aws

사용자 가이드

Amazon ElastiCache

API 버전 2015-02-02

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon ElastiCache: 사용자 가이드

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

Table of Contents

Amazon이란 무엇입니까 ElastiCache?

Amazon ElastiCache 사용 설명서 를 시작합니다. Amazon ElastiCache 은 클라우드에서 분산 메모리 내 데이터 스토어 또는 캐시 환경을 쉽게 설정, 관리 및 확장할 수 있는 웹 서비스입니다. 확장 가능하고 비용 효율적인 고성능 캐싱 솔루션을 제공합니다. 또한 분산된 캐시 환경의 배포 및 관리와 관련된 복잡성을 해소할 수 있습니다.

Amazon ElastiCache 은 두 가지 형식으로 작동할 수 있습니다. 서버리스 캐시로 시작하거나 자체 캐시 클러스터를 설계하도록 선택할 수 있습니다.

> **Note**
>
> Amazon은 Valkey, Redis OSS및 Memcached 엔진과 함께 ElastiCache 작동합니다. 사용하고 싶은 엔진을 결정하기 어렵다면 이 가이드의 Valkey, Redis OSS및 Memcached 자체 설계 캐시 비교 섹션을 참조하세요.

서버리스 캐시

ElastiCache 는 애플리케이션의 캐시 추가 및 운영을 간소화하는 서버리스 캐싱을 제공합니다. ElastiCache Serverless를 사용하면 1분 이내에 고가용성 캐시를 생성할 수 있으므로 인스턴스를 프로비저닝하거나 노드 또는 클러스터를 구성할 필요가 없습니다. 개발자는 ElastiCache 콘솔 SDK 또는 를 사용하여 캐시 이름을 지정하여 서버리스 캐시를 생성할 수 있습니다CLI.

ElastiCache 또한 Serverless는 캐싱 용량을 계획하고 관리할 필요가 없습니다. 캐시의 메모리를 ElastiCache 지속적으로 모니터링합니다. 컴퓨팅, 및 애플리케이션에서 사용하는 네트워크 대역폭, 및 는 애플리케이션의 요구 사항을 충족하도록 확장됩니다. 는 개발자에게 간단한 엔드포인트 환경을 ElastiCache 제공합니다. 기본 캐시 인프라 및 클러스터 설계를 추상화하여 하드웨어 프로비저닝을 ElastiCache 관리합니다. 모니터링, 노드 교체, 소프트웨어 패치를 자동으로 투명하게 적용하면 애플리케이션 개발에 집중할 수 있도록 캐시를 작동하는 대신

ElastiCache Serverless는 Valkey 7.2, Redis OSS 7.1 이상 및 Memcached 1.6.21 이상과 호환됩니다.

자체 ElastiCache 클러스터 설계

ElastiCache 클러스터를 세밀하게 제어해야 하는 경우 를 사용하여 자체 Valkey, Redis OSS 또는 Memcached 클러스터를 설계하도록 선택할 수 있습니다 ElastiCache. 를 ElastiCache 사용하면 클러

스터의 AWS 가용 영역 전체에서 노드 유형, 노드 수 및 노드 배치를 선택하여 클러스터를 설계할 수 있습니다. ElastiCache 는 완전 관리형 서비스이므로 클러스터의 하드웨어 프로비저닝, 모니터링, 노드 교체 및 소프트웨어 패치를 자동으로 관리합니다.

자체 ElastiCache 클러스터를 설계하면 클러스터에 대한 유연성과 제어력이 향상됩니다. 예를 들어 필요에 따라 단일 AZ 가용성 또는 다중 AZ 가용성으로 클러스터를 운영하도록 선택할 수 있습니다. 클러스터 모드에서 Valkey, Redis OSS 또는 Memcached를 실행하여 수평 조정을 활성화하거나 클러스터 모드 없이 수직 조정만 수행하도록 선택할 수도 있습니다. 클러스터를 직접 설계할 때는 애플리케이션에 필요한 만큼 캐시 용량이 충분하도록 노드 유형과 수를 올바르게 선택해야 합니다. Valkey 또는 Redis OSS 클러스터에 새 소프트웨어 패치를 적용할 시기를 선택할 수도 있습니다.

자체 ElastiCache 클러스터를 설계할 때 Valkey 7.2, Redis OSS 4.0~7.1 또는 Memcached 1.4 이상을 실행하도록 선택할 수 있습니다.

관련 서비스

MemoryDB

ElastiCache 또는 MemoryDB를 사용할지 여부를 결정할 때 다음 비교를 고려하세요.

- ElastiCache 는 Valkey, Redis OSS 또는 Memcached를 사용하여 다른 데이터베이스 및 데이터 스토어의 데이터를 캐시하는 데 일반적으로 사용되는 서비스입니다. 기존 기본 데이터베이스 또는 데이터 스토어(마이크로초 읽기 및 쓰기 성능)를 사용하여 데이터 액세스를 가속화하려는 워크로드 캐싱에 ElastiCache 고려해야 합니다. 또한 Valkey 또는 Redis OSS 데이터 구조를 사용하고 기본 데이터베이스 또는 데이터 스토어에 저장된 데이터에 APIs 액세스하려는 ElastiCache 사용 사례를 고려해야 합니다.
- ElastiCache 또한 는 자주 액세스하는 데이터를 캐시에 저장하여 데이터베이스 비용을 절감하는 데 도움이 됩니다. 애플리케이션의 읽기 처리량 요구 사항이 높으면 기본 데이터베이스를 확장하는 ElastiCache대신 를 사용하여 대규모, 빠른 성능 및 데이터 스토리지 비용 절감을 달성할 수 있습니다.
- MemoryDB는 매우 빠른 기본 데이터베이스가 필요한 워크로드를 위한 내구성이 뛰어난 인메모리 데이터베이스입니다. Valkey 및 Redis 와 호환됩니다OSS. 워크로드에 초고속 성능(마이크로초 단위 읽기 및 10밀리초의 쓰기 지연 시간)을 제공하는 내구성이 뛰어난 데이터베이스가 필요한 경우, MemoryDB 사용을 고려해야 합니다. 또한 MemoryDB는 Valkey 또는 Redis OSS 데이터 구조를 사용하고 기본 내구성 데이터베이스를 APIs 사용하여 애플리케이션을 빌드하려는 경우 사용 사례에 적합할 수 있습니다. 마지막으로, 내구성과 성능을 위해 데이터베이스 사용을 캐시로 대체하여 애플리케이션 아키텍처를 단순화하고 비용을 절감하려면 MemoryDB 사용을 고려해야 합니다.

Amazon Relational Database Service

ElastiCache 는 자주 액세스하는 데이터를 캐시에 저장하여 데이터베이스 비용을 절감하는 데 도움이 됩니다. 애플리케이션의 읽기 처리량 요구 사항이 높으면 기본 데이터베이스를 확장하는 ElastiCache 대신 를 사용하여 대규모, 빠른 성능 및 데이터 스토리지 비용 절감을 달성할 수 있습니다.

관련 Amazon Relational Database Service 서비스에 대한 자세한 내용은 AmazonRDS을 참조하세요.

ElastiCache 는 자주 액세스하는 데이터를 캐시에 저장하여 데이터베이스 비용을 절감하는 데 도움이 됩니다. 애플리케이션의 읽기 처리량 요구 사항이 높으면 기본 데이터베이스를 확장하는 ElastiCache 대신 를 사용하여 대규모, 빠른 성능 및 데이터 스토리지 비용 절감을 달성할 수 있습니다.

ElastiCache 작동 방식

여기서 ElastiCache 배포의 주요 구성 요소에 대한 개요를 확인할 수 있습니다.

캐시 및 캐싱 엔진

캐시는 캐시된 데이터를 저장하는 데 사용할 수 있는 메모리 내 데이터 스토어입니다. 일반적으로 애플리케이션은 응답 시간을 최적화하기 위해 자주 액세스하는 데이터를 캐시에 캐시합니다. ElastiCache 는 서버리스 클러스터와 자체 설계된 클러스터라는 두 가지 배포 옵션을 제공합니다. 배포 옵션 간 선택을 참조하세요.

> **Note**
>
> Amazon은 Valkey, Redis OSS 및 Memcached 엔진과 함께 ElastiCache 작동합니다. 사용하고 싶은 엔진을 결정하기 어렵다면 이 가이드의 Valkey, Redis OSS및 Memcached 자체 설계 캐시 비교 섹션을 참조하세요.

주제

ElastiCache 작동 방식

ElastiCache 서버리스

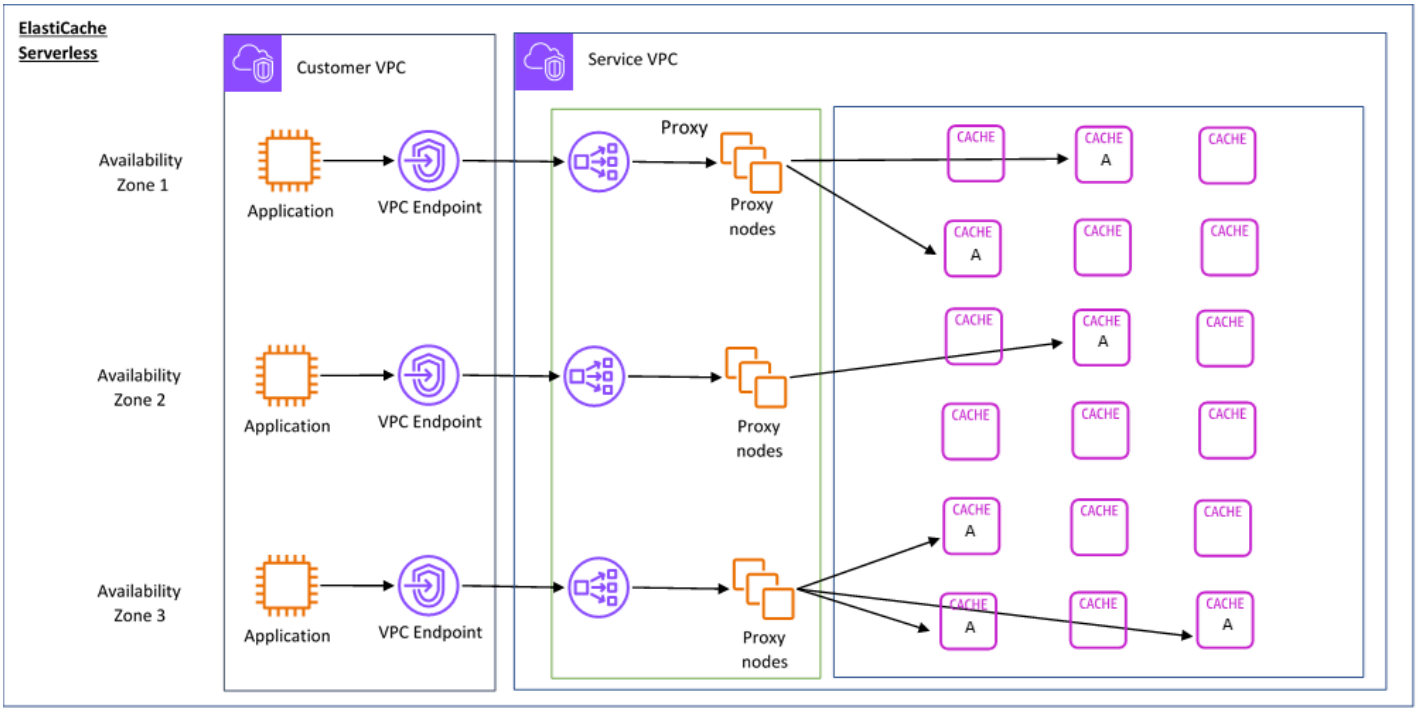
ElastiCache Serverless를 사용하면 용량 계획, 하드웨어 관리 또는 클러스터 설계에 대한 걱정 없이 캐시를 생성할 수 있습니다. 캐시의 이름을 제공하면 Valkey, Redis OSS 또는 Memcached 클라이언트에서 캐시 액세스를 시작하도록 구성할 수 있는 단일 엔드포인트가 수신됩니다.

> **Note**
>
> - ElastiCache Serverless는 클러스터 모드에서 Valkey, Redis OSS 또는 Memcached를 실행하며 를 지원하는 클라이언트와만 호환됩니다TLS.

주요 이점

- 용량 계획 없음: ElastiCache Serverless를 사용하면 용량을 계획할 필요가 없습니다. ElastiCache Serverless는 캐시의 메모리, 컴퓨팅 및 네트워크 대역폭 사용률을 지속적으로 모니터링하고 수직 및 수평으로 확장합니다. 이렇게 하면 캐시 노드의 크기를 늘리는 동시에 스케일 아웃 작업을 시작하여 항상 애플리케이션 요구 사항에 맞게 캐시 규모를 조정할 수 있습니다.
- P ay-per-use: ElastiCache Serverless를 사용하면 캐시의 워크로드에서 사용하는 저장 데이터 및 컴퓨팅 비용을 지불합니다. 차원 사용을 참조하세요.
- 고가용성: ElastiCache Serverless는 고가용성을 위해 여러 가용 영역(AZ)에서 데이터를 자동으로 복제합니다. 기본 캐시 노드를 자동으로 모니터링하고 장애 발생 시 이를 대체합니다. 모든 캐시에 대해 99.99%의 가용성SLA을 제공합니다.
- 자동 소프트웨어 업그레이드: ElastiCache Serverless는 애플리케이션에 대한 가용성 영향 없이 캐시를 최신 마이너 및 패치 소프트웨어 버전으로 자동으로 업그레이드합니다. 새 메이저 버전을 사용할 수 있게 되면 ElastiCache 에서 알림을 보냅니다.
- 보안: 서버리스는 전송 중 데이터와 저장된 데이터를 항상 암호화합니다. 서비스 관리형 키를 사용하거나 자체 고객 관리형 키를 사용하여 저장 데이터를 암호화할 수 있습니다.

다음 다이어그램은 ElastiCache Serverless의 작동 방식을 보여줍니다.



새 서버리스 캐시를 생성할 때 는 에서 선택한 서브넷에 Virtual Private Cloud(VPC) 엔드포인트를 ElastiCache 생성합니다VPC. 애플리케이션은 이러한 VPC 엔드포인트를 통해 캐시에 연결할 수 있습니다.

ElastiCache Serverless를 사용하면 애플리케이션이 연결하는 단일 DNS 엔드포인트를 받게 됩니다. 엔드포인트에 대한 새 연결을 요청하면 ElastiCache Serverless는 프록시 계층을 통해 모든 캐시 연결을 처리합니다. 프록시 계층을 사용하면 기본 클러스터가 변경될 경우 클라이언트가 클러스터 토폴로지를 재검색할 필요가 없으므로 복잡한 클라이언트 구성을 줄일 수 있습니다. 프록시 계층은 Network Load Balancer를 사용하여 연결을 처리하는 프록시 노드 집합입니다.

애플리케이션이 새 캐시 연결을 생성하면 Network Load Balancer가 요청을 프록시 노드로 보냅니다. 애플리케이션이 캐시 명령을 실행하면 애플리케이션에 연결된 프록시 노드가 캐시의 캐시 노드에서 요청을 실행합니다. 프록시 계층은 클라이언트의 캐시 클러스터 토폴로지와 노드를 추상화합니다. 이렇게 하면 가용성 ElastiCache 이 애플리케이션에 영향을 미치거나 연결을 재설정하지 않고도 가 지능적으로 로드 밸런싱, 스케일 아웃 및 새 캐시 노드 추가, 실패 시 캐시 노드 교체, 캐시 노드의 소프트웨어 업데이트가 가능합니다.

자체 설계된 ElastiCache 클러스터

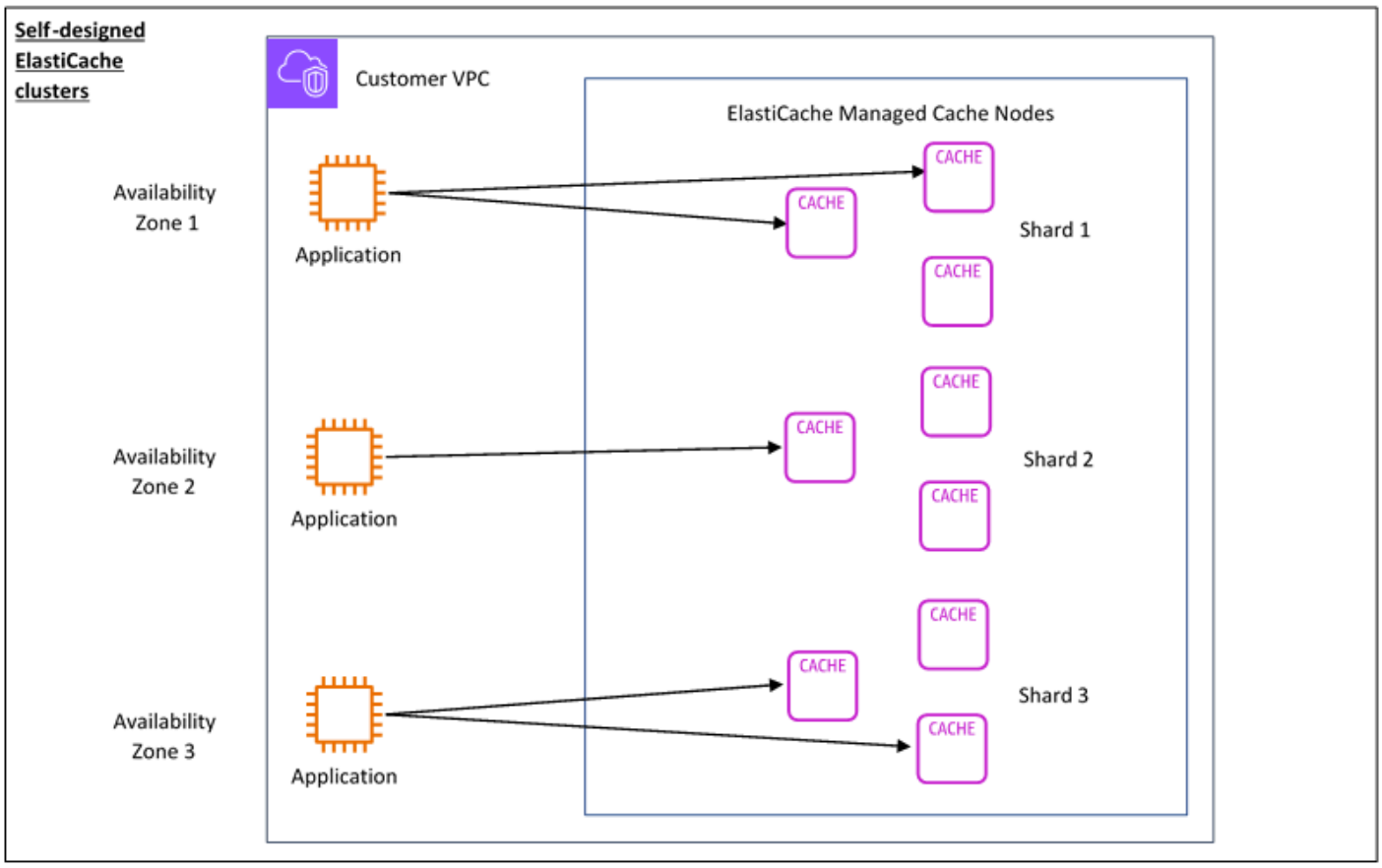
ElastiCache 클러스터의 캐시 노드 패밀리, 크기 및 노드 수를 선택하여 자체 클러스터를 설계할 수 있습니다. 클러스터를 직접 설계하면 보다 세밀하게 제어할 수 있으며 캐시의 샤드 수와 각 샤드의 노드 수(기본 및 복제본)를 선택할 수 있습니다. 여러 개의 샤드가 있는 OSS 클러스터를 생성하여 클러스터

모드에서 Valkey 또는 Redis를 작동하거나 단일 샤드가 있는 비클러스터 모드에서 작동하도록 선택할 수 있습니다.

주요 이점

- 자체 클러스터 설계: 를 사용하면 자체 클러스터를 설계하고 캐시 노드를 배치할 위치를 선택할 ElastiCache수 있습니다. 예를 들어 고가용성을 낮은 지연 시간과 절충하려는 애플리케이션을 보유하고 있는 경우 단일 AZ에 캐시 노드를 배포하도록 선택할 수 있습니다. 또는 여러 노드가 있는 클러스터를 설계AZs하여 고가용성을 달성할 수 있습니다.
- 세밀한 제어: 자체 클러스터를 설계할 때 캐시의 설정을 더 세밀하게 조정할 수 있습니다. 예를 들어 Valkey 및 Redis OSS 파라미터 또는 Memcached 특정 파라미터 를 사용하여 캐시 엔진을 구성할 수 있습니다.
- 수직 및 수평으로 규모 조정: 필요할 때 캐시 노드 크기를 늘리거나 줄여 수동으로 클러스터 규모를 조정하도록 선택할 수 있습니다. 새 샤드를 추가하거나 샤드에 복제본을 추가하여 수평적으로 규모를 조정할 수도 있습니다. Auto-Scaling 기능을 사용하여 캐시의 CPU 및 메모리 사용량과 같은 지표를 기반으로 일정 또는 크기 조정을 기반으로 크기 조정을 구성할 수도 있습니다.

다음 다이어그램은 ElastiCache 자체 설계된 클러스터의 작동 방식을 보여줍니다.



차원 사용

두 ElastiCache 가지 배포 옵션으로 배포할 수 있습니다. ElastiCache Serverless를 배포할 때 GB-시간 단위로 저장된 데이터의 사용량과 ElastiCache 처리 단위()의 컴퓨팅 비용을 지불합니다ECPU. 자체 ElastiCache 클러스터를 설계하기로 선택하면 캐시 노드 사용량의 시간당 요금을 지불합니다. 요금 관련 세부 사항은 여기를 참조하세요.

데이터 스토리지

기가바이트 시간(GB 시간) 단위로 청구되는 ElastiCache Serverless에 저장된 데이터에 대해 비용을 지불합니다. ElastiCache Serverless는 캐시에 저장된 데이터를 지속적으로 모니터링하여 분당 여러 번 샘플링하고 시간당 평균을 계산하여 캐시의 데이터 스토리지 사용량을 GB-hr 단위로 결정합니다. 각 ElastiCache Serverless 캐시는 저장된 최소 1GB의 데이터에 대해 측정됩니다.

ElastiCache 처리 단위(ECPUs)

vCPU 시간과 전송된 데이터가 모두 포함된 단위인 ElastiCache 처리 단위(ECPUs)의 ElastiCache Serverless에서 애플리케이션이 실행하는 요청에 대해 비용을 지불합니다.

- 간단한 읽기 및 쓰기에는 전송된 데이터의 킬로바이트(KB)ECPU당 1개가 필요합니다. 예를 들어 최대 1KB의 데이터를 전송하는 GET 명령은 1개를 사용합니다ECPU. 3.2KB의 데이터를 전송하는 SET 요청은 3.2를 사용합니다ECPUs.
- Valkey 및 Redis 를 사용하면 더 많은 vCPU 시간을 소비하고 더 많은 데이터를 전송하는 OSS명령이 두 차원 중 더 높은 차원을 ECPUs 기반으로 소비합니다. 예를 들어 애플리케이션이 HMGET 명령을 사용하고, 가 vCPU 시간의 3배를 단순 SET/GET 명령으로 소비하고, 3.2KB의 데이터를 전송하는 경우 3.2를 소비합니다ECPU. 또는 2KB의 데이터만 전송하는 경우 3개를 사용합니다ECPUs.
- Valkey 및 Redis 를 사용하면 추가 vCPU 시간이 필요한 OSS명령은 비례적으로 더 많은 를 소비합니다ECPUs. 예를 들어 애플리케이션이 Valkey 또는 Redis OSS HMGET 명령 을 사용하고 vCPU 시간을 단순 SET/GET 명령으로 3배 사용하는 경우 3개를 사용합니다ECPUs.
- Memcached를 사용하면 여러 항목에서 작동하는 명령이 비례적으로 더 많은 를 소비합니다ECPUs. 예를 들어 애플리케이션이 3개 항목에 대해 멀티게트를 수행하는 경우 3개를 사용합니다ECPUs.
- Memcached를 사용하면 더 많은 항목에서 작동하고 더 많은 데이터를 전송하는 명령이 두 차원 중 더 높은 차원을 ECPUs 기반으로 소비됩니다. 예를 들어 애플리케이션이 GET 명령을 사용하고, 3개의 항목을 검색하고, 3.2KB의 데이터를 전송하는 경우 3.2를 사용합니다ECPU. 또는 2KB의 데이터만 전송하는 경우 3개를 사용합니다ECPUs.

ElastiCache Serverless는 워크로드에서 ECPUs 소비되는 를 이해하는 데 도움이 `ElastiCacheProcessingUnits`되는 라는 새 지표를 내보냅니다.

노드 시간

가용 영역에 대한 EC2 노드 패밀리, 크기, 노드 수 및 배치를 선택하여 자체 캐시 클러스터를 설계할 수 있습니다. 클러스터를 자체 설계할 때는 각 캐시 노드에 대해 시간당 요금을 지불합니다.

ElastiCache 백업

백업은 point-in-time 서버리스 캐시 또는 Valkey 또는 Redis OSS 자체 설계 클러스터의 사본입니다. ElastiCache 를 사용하면 언제든지 데이터를 백업하거나 자동 백업을 설정할 수 있습니다. 백업을 사용하여 기존 캐시를 복원하거나 새 캐시를 시드할 수 있습니다. 백업은 캐시의 모든 데이터와 일부 메타데이터로 구성됩니다. 자세한 정보는 스냅샷 및 복원 섹션을 참조하세요.

배포 옵션 간 선택

Amazon ElastiCache 에는 두 가지 배포 옵션이 있습니다.

- 서버리스 캐시
- 자체 설계된 클러스터

둘 다에 대해 지원되는 명령 목록은 섹션을 참조하세요지원 및 제한된 Valkey, Redis OSS 및 Memcached 명령.

서버리스 캐시

Amazon ElastiCache Serverless는 캐시 생성을 간소화하고 고객의 가장 까다로운 애플리케이션을 지원하도록 즉시 확장합니다. ElastiCache Serverless를 사용하면 1분 이내에 가용성과 확장성이 뛰어난 캐시를 생성할 수 있으므로 캐시 클러스터 용량을 프로비저닝, 계획 및 관리할 필요가 없습니다. ElastiCache Serverless는 3개의 가용 영역에 걸쳐 데이터를 자동으로 중복 저장하고 99.99% 가용성 서비스 수준 계약(SLA)을 제공합니다. 자체 설계된 Valkey 또는 Redis OSS 클러스터의 백업을 서버리스 구성으로 복원할 수 있습니다.

자체 설계된 클러스터

Valkey, Redis OSS또는 Memcached 클러스터를 세밀하게 제어해야 하는 경우 를 사용하여 자체 클러스터를 설계하도록 선택할 수 ElastiCache있습니다. 를 ElastiCache 사용하면 클러스터의 AWS 가용 영역 전체에서 노드 유형, 노드 수 및 노드 배치를 선택하여 노드 기반 클러스터를 운영할 수 있습니다. ElastiCache 는 완전 관리형 서비스이므로 클러스터의 하드웨어 프로비저닝, 모니터링, 노드 교체 및 소프트웨어 패치를 관리하는 데 도움이 됩니다. 자체 설계된 클러스터는 최대 99.99%의 가용성을 제공하도록 설계할 수 있습니다SLA. 서버리스 Valkey 또는 Redis OSS 캐시의 백업은 자체 설계된 클러스터로 복원할 수 있습니다.

배포 옵션 간 선택

다음과 같은 경우 서버리스 캐싱을 선택합니다.

- 새롭거나 예측하기 어려운 워크로드에 대한 캐시를 생성하고 있습니다.
- 애플리케이션 트래픽이 예측 불가능한 경우
- 캐시를 가장 쉽게 시작하는 방법을 찾고 있는 경우

다음과 같은 경우 자체 ElastiCache 클러스터를 설계하도록 선택합니다.

- 이미 ElastiCache Serverless를 실행 중이며 Valkey, Redis 또는 Memcached를 실행하는 노드 유형 OSS, 노드 수 및 해당 노드의 배치에 대해 더 세분화된 제어를 원합니다.

- 애플리케이션 트래픽은 비교적 예측 가능하며 성능, 가용성 및 비용에 대한 세분화된 제어를 원합니다.
- 비용 제어를 위해 용량 요구 사항을 예측할 수 있는 경우

서버리스 캐싱과 자체 설계된 클러스터 비교

기능	서버리스 캐시	자체 설계된 클러스터
캐시 설정	1분 이내에 이름만 있는 캐시 생성	캐시 클러스터 설계에 대한 세분화된 제어를 제공합니다. 사용자는 노드 유형, 노드 수 및 AWS 가용 영역 간 배치를 선택할 수 있습니다.
지원되는 ElastiCache 버전	Valkey 7.2 이상, Redis OSS 버전 7.1 이상, Memcached 1.6.21 이상	Valkey 7.2 이상, Redis OSS 버전 4.0 이상, Memcached 1.4 이상
클러스터 모드(발키 및 RedisOSS)	에서 엔진`cluster mode enabled`만 작동합니다. 클라이언트는 ElastiCache Serverless에 연결`cluster mode enabled`하려면 를 지원해야 합니다.	클러스터 모드가 활성화되거나 클러스터 모드가 비활성화된 상태에서 작동하도록 구성할 수 있습니다.
스케일링	용량 관리 없이 수직 및 수평으로 엔진을 자동으로 확장합니다.	현재 용량이 수요를 적절하게 충족하는지 확인하기 위해 모니터링이 필요한 동시에 조정을 제어합니다. Valkey 및 Redis 의 경우 필요한 경우 캐시 노드 크기를 늘리거나 줄여 수직으로 확장하도록 OSS선택할 수 있습니다. 새 샤드를 추가하거나 샤드에 복제본을 더 추가하여 수평으로 확장할 수도 있습니다.

기능	서버리스 캐시	자체 설계된 클러스터
		Memcached에서는 이 기능을 사용할 수 없습니다. Auto-Scaling 기능을 사용하면 일정에 따라 조정을 구성하거나 캐시의 CPU 및 메모리 사용량과 같은 지표를 기반으로 조정을 구성할 수도 있습니다.
클라이언트 연결	클라이언트는 단일 엔드포인트에 연결됩니다. 이렇게 하면 기본 캐시 노드 토폴로지(스케일링, 교체 및 업그레이드)가 클라이언트 연결을 해제하지 않고 변경될 수 있습니다.	클라이언트는 각 개별 캐시 노드에 연결됩니다. 노드를 교체하면 클라이언트가 클러스터 토폴로지를 다시 검색하고 연결을 다시 설정합니다.
구성 가능성	세분화된 구성을 사용할 수 없습니다. 고객은 캐시에 액세스할 수 있는 서브넷, 자동 백업을 켜거나 끌지 여부, 최대 캐시 사용 제한을 포함한 기본 설정을 구성할 수 있습니다.	자체 설계된 클러스터는 세분화된 구성 옵션을 제공합니다. 고객은 파라미터 그룹을 사용하여 세분화된 제어를 수행할 수 있습니다. 노드 유형별 파라미터 값의 표는 엔진별 파라미터 섹션을 참조하세요.
다중 AZ	데이터는 가용성을 높이고 읽기 지연 시간을 개선하기 위해 여러 가용 영역에 비동기적으로 복제됩니다.	단일 가용 영역 또는 여러 가용 영역()에서 클러스터를 설계하는 옵션을 제공합니다AZs. Valkey 또는 Redis 를 사용하는 경우 OSS는 다중 가용 영역에 비동기적으로 복제된 데이터가 포함된 다중 AZ 클러스터를 제공하여 가용성을 높이고 읽기 지연 시간을 개선합니다.

기능	서버리스 캐시	자체 설계된 클러스터
저장 중 암호화	항상 활성화됨. 고객은 에서 AWS 관리형 키 또는 고객 관리형 키를 사용할 수 있습니다 AWS KMS.	저장 시 암호화를 활성화 또는 비활성화하는 옵션입니다. 활성화되면 고객은 에서 AWS 관리형 키 또는 고객 관리형 키를 사용할 수 있습니다 AWS KMS.
전송 중 암호화(TLS)	항상 활성화됨. 클라이언트는 TLS 연결을 지원해야 합니다.	활성화 또는 비활성화 옵션입니다.
백업	성능에 영향을 주지 않고 캐시의 자동 및 수동 백업을 지원합니다. Valkey 및 Redis OSS 백업은 교차 호환되며 ElastiCache Serverless 캐시 또는 자체 설계된 클러스터로 복원할 수 있습니다.	Valkey 및 Redis 에 대한 자동 및 수동 백업을 지원합니다 OSS. 클러스터는 사용 가능한 예약 메모리에 따라 성능에 약간의 영향을 미칠 수 있습니다. 자세한 내용은 Valkey 및 Redis 용 예약 메모리 관리 OSS 단원을 참조하십시오. Valkey 및 Redis OSS 백업은 교차 호환되며 ElastiCache Serverless 캐시 또는 자체 설계된 클러스터로 복원할 수 있습니다.

기능	서버리스 캐시	자체 설계된 클러스터
모니터링	캐시 적중률, 캐시 누락률, 데이터 크기 및 ECPUs 소비를 포함한 캐시 수준 지표를 지원합니다. ElastiCache Serverless는 캐시에서 중요한 이벤트가 발생할 EventBridge 때 를 사용하여 이벤트를 전송합니다. Amazon 를 사용하여 ElastiCache 이벤트를 모니터링, 수집, 변환 및 작업하도록 선택할 수 있습니다 EventBridge. 자세한 내용은 서버리스 캐시 이벤트 단원을 참조하십시오.	ElastiCache 자체 설계된 클러스터는 호스트 수준 지표와 캐시 지표를 포함하여 각 노드 수준에서 지표를 내보냅니다. 자체 설계된 클러스터는 중요한 이벤트에 대한 SNS 알림을 내보냅니다. Memcached 지표 및 Valkey 및 Redis에 대한 지표 OSS 단원을 참조하세요.
가용성	99.99% 가용성 서비스 수준 계약(SLA)	자체 설계된 클러스터는 구성에 따라 최대 99.99%의 가용성 서비스 수준 계약(SLA)을 달성하도록 설계할 수 있습니다.
소프트웨어 업그레이드 및 패치 적용	애플리케이션에 영향을 주지 않고 캐시 소프트웨어를 최신 마이너 및 패치 버전으로 자동 업그레이드합니다. 고객은 메이저 버전 업그레이드에 대한 알림을 받으며 고객은 원하는 경우 최신 메이저 버전으로 업그레이드할 수 있습니다.	자체 설계된 클러스터는 마이너 및 패치 버전 업그레이드와 메이저 버전 업그레이드를 위한 고객 지원 셀프 서비스를 제공합니다. 관리형 업데이트는 고객이 정의한 유지 관리 기간 동안 자동으로 적용됩니다. 고객은 마이너 또는 패치 버전 업그레이드를 온디맨드로 적용하도록 선택할 수도 있습니다.
글로벌 데이터 스토어	지원되지 않음	단일 리전 쓰기 및 다중 리전 읽기로 리전 간 복제를 지원하는 Global Data Store 지원

기능	서버리스 캐시	자체 설계된 클러스터
데이터 계층화	지원되지 않음	r6gd 패밀리의 노드를 사용하여 설계된 클러스터에는 메모리와 로컬SSD(솔리드 스테이트 드라이브) 스토리지 간에 계층화된 데이터가 있습니다. 데이터 계층화는 메모리에 데이터를 저장하는 것 외에도 각 클러스터 노드의 저렴한 솔리드 스테이트 드라이브(SSDs)를 활용하여 Valkey 및 Redis OSS 워크로드에 대한 가격 대비 성능 옵션을 제공합니다.
요금 모델	P ay-per-use, GB-시간 단위로 저장된 데이터와 ElastiCache 처리 단위()의 요청을 기반으로 합니다ECPU. 요금 관련 세부 사항은 여기를 참조하세요.	P ay-per-hour, 캐시 노드 사용량 기준. 요금 관련 세부 사항은 여기를 참조하세요.

관련 항목:

- 자체 ElastiCache 클러스터 설계 및 관리

처음 사용자를 위한 Amazon ElastiCache 리소스

처음 사용하는 사용자는 다음 섹션을 읽고 필요에 따라 참조하는 것이 좋습니다.

- 서비스 하이라이트 및 요금 - 제품 세부 정보 페이지에서는 ElastiCache, 서비스 하이라이트 및 요금에 대한 일반적인 제품 개요를 제공합니다.
- ElastiCache 비디오 - ElastiCache 비디오 섹션에는 Amazon 를 소개하는 비디오가 있습니다 ElastiCache. 이 동영상에서는 ElastiCache 및 데모의 일반적인 사용 사례를 다루며 를 사용하여 ElastiCache 지연 시간을 줄이고 애플리케이션의 처리량을 개선하는 방법을 설명합니다.

- 시작하기 - Amazon 시작하기 ElastiCache 섹션에는 캐시 클러스터 생성에 대한 정보가 나와 있습니다. 여기에는 캐시 클러스터에 액세스할 수 있는 권한을 부여하는 방법과 캐시 노드에 연결하고 캐시 클러스터를 삭제하는 방법도 포함되어 있습니다.
- 대규모 성능 - Amazon 백서를 사용한 대규모 성능 ElastiCache은 애플리케이션이 대규모로 잘 수행하는 데 도움이 되는 캐싱 전략을 다룹니다.

이전 섹션을 완료한 후에는 다음 섹션을 읽어 보십시오.

- 노드 크기 선택

  캐시하려는 모든 데이터를 수용할 만큼의 충분한 노드를 원하며, 동시에 필요한 것보다 더 많은 캐시의 비용을 지불하고 싶지 않은 경우 이 항목을 사용하여 최적의 노드 크기를 선택할 수 있습니다.
- ElastiCache 모범 사례 및 캐싱 전략

  클러스터의 효율성에 영향을 줄 수 있는 문제를 확인하고 해결하세요.

AWS Command Line Interface (AWS CLI)를 사용하려면 다음 문서를 사용하여 시작할 수 있습니다.

- AWS Command Line Interface 설명서

  이 섹션에서는 를 다운로드하고 AWS CLI, 시스템에서 AWS CLI 작업을 수행하고, AWS 자격 증명을 제공하는 방법에 대한 정보를 제공합니다.
- AWS CLI 설명서 ElastiCache

  이 별도의 문서는 구문 및 예제를 포함하여 ElastiCache 명령에 AWS CLI 대한 모든 를 다룹니다.

애플리케이션 프로그램을 작성하여 널리 사용되는 다양한 프로그래밍 언어와 함께 ElastiCache API를 사용할 수 있습니다. 다음과 같은 몇 가지 리소스가 있습니다.

- Amazon Web Services용 도구

  Amazon Web Services는 에 대한 지원과 함께 여러 소프트웨어 개발 키트(SDKs)를 제공합니다 ElastiCache. Java, ., NETPHP, Ruby 및 기타 언어를 ElastiCache 사용하도록 코딩할 수 있습니다. 이렇게 하면 요청을 로 포맷하고 ElastiCache, 응답을 구문 분석하고, 재시도 로직 및 오류 처리를 제공하여 애플리케이션 개발을 크게 간소화할 SDKs 수 있습니다.
- 사용 ElastiCache API

를 사용하지 않으려면 쿼리 를 사용하여 와 ElastiCache 직접 상호 작용 AWS SDKs할 수 있습니다 API. 이 섹션에서 요청 생성과 인증 및 응답 처리에 대한 정보와 문제 해결 팁을 찾을 수 있습니다.

- Amazon ElastiCache API 참조

  이 별도의 문서는 구문 및 예제를 ElastiCache API 포함한 모든 작업을 다룹니다.

AWS 리전 및 가용 영역

Amazon 클라우드 컴퓨팅 리소스는 전 세계 여러 리전의 가용성이 높은 데이터 센터 시설에 하우징됩니다(예: 북미, 유럽 또는 아시아). 각 데이터 센터 위치를 AWS 리전이라고 합니다.

각 AWS 리전에는 가용 영역 또는 라는 여러 개의 고유한 위치가 포함되어 있습니다AZs. 각 가용 영역은 다른 가용 영역에서 발생한 장애에서 격리되도록 설계되었습니다. 각 는 동일한 AWS 리전의 다른 가용 영역에 저렴하고 지연 시간이 짧은 네트워크 연결을 제공하도록 설계되었습니다. 별도의 가용 영역에서 인스턴스를 시작함으로써 단일 위치에서 장애가 발생할 경우 애플리케이션을 보호할 수 있습니다. 자세한 내용은 리전 및 가용 영역 선택 섹션을 참조하세요.

여러 가용 영역에서 클러스터를 생성할 수 있습니다. 다중 AZ 배포라는 옵션입니다. 이 옵션을 선택하면 Amazon은 다른 가용 영역에서 보조 예비 노드 인스턴스를 자동으로 프로비저닝하고 유지합니다. 기본 노드 인스턴스는 가용 영역 전체에서 보조 인스턴스로 비동기식으로 복제됩니다. 이 접근 방식을 통해 데이터 중복 및 장애 조치 지원을 제공하고, I/O 중지를 없애고, 시스템 백업 중에 지연 시간 스파이크를 최소화할 수 있습니다. 자세한 내용은 다중 AZ 를 사용한 ElastiCache (Redis OSS)의 가동 중지 시간 최소화를 참조하세요.

일반적인 ElastiCache 사용 사례 및 가 도울 ElastiCache 수 있는 방법

최신 뉴스, 10대 리더보드, 제품 카탈로그를 게재할 때나 이벤트 티켓을 판매할 때 가장 중요한 것은 속도입니다. 웹 사이트와 비즈니스의 성공 여부는 콘텐츠를 제공하는 속도에 상당한 영향을 받습니다.

뉴욕 타임즈의 "For Impatient Web Users, an Eye Blink Is Just Too Long to Wait(참을성 없는 웹 사용자에게는 눈 깜박하는 시간조차 너무 길게 느껴져)"라는 기사에 따르면 사용자는 경쟁 사이트 간의 250밀리초(4/4초) 차이를 인지합니다. 사용자는 결국 속도가 느린 사이트를 떠나 빠른 사이트로 이동합니다. How Webpage Load Time Is Related to Visitor Loss에 따르면 Amazon에서 실시한 테스트에서 로드 시간이 100밀리초(10/10초) 증가할 때마다 매출이 1% 감소하는 결과가 나왔습니다.

다른 사용자가 데이터를 원할 경우 캐시된 데이터를 훨씬 더 빠르게 제공할 수 있습니다. 웹 페이지든 비즈니스 결정을 주도하는 보고서용이든 이는 동일하게 적용되는 사실입니다. 귀사에서는 웹 페이지를 캐시하지 않고 지연 시간을 최소화하여 제공할 여유가 있습니까?

가장 많이 요청되는 항목을 캐시하려 할 것이라는 사실은 어쩌면 자명한 것일 수 있습니다. 요청 빈도가 낮은 항목을 캐시하려고 하지 않는 이유는 무엇입니까? 가장 최적화된 데이터베이스 쿼리 또는 원격 API 호출도 인 메모리 캐시에서 플랫 키를 검색하는 것보다 훨씬 느립니다. 현저히 느려지면 고객이 다른 곳으로 떠나는 경향이 있습니다.

다음 예제에서는 를 사용하여 애플리케이션의 전반적인 성능을 개선할 ElastiCache 수 있는 몇 가지 방법을 보여줍니다.

주제

- 인 메모리 데이터 스토어
- 게임 리더보드
- 메시징(Pub/Sub)
- 권장 데이터(해시)
- ElastiCache 고객 추천

인 메모리 데이터 스토어

인 메모리 키-값 스토어의 주된 목적은 지연 시간이 1밀리초 미만인 매우 빠른 속도와 저렴한 비용으로 데이터 복사본에 액세스할 수 있게 하는 것입니다. 대부분의 데이터 스토어에는 자주 액세스하지만 거의 업데이트하지 않는 데이터 영역이 있습니다. 또한 데이터베이스를 쿼리하면 키-값 페어 캐시에서 키를 찾는 것에 비해 확실히 속도가 느리고 비용이 많이 듭니다. 일부 데이터베이스 쿼리는 특히 수행하

는 데 있어 많은 비용이 듭니다. 예로는 여러 표에 걸친 조인이나 복잡한 계산이 포함된 쿼리를 들 수 있습니다. 이러한 쿼리 결과를 캐시하여 쿼리 가격을 한 번만 지불합니다. 그런 다음 쿼리를 다시 실행할 필요 없이 여러 번 데이터를 빠르게 검색할 수 있습니다.

어떻게 캐시해야 합니까?

캐시할 데이터를 결정할 때 다음과 같은 요인을 고려합니다.

속도 및 비용 - 캐시가 아닌 데이터베이스에서 데이터를 얻는 것이 항상 더 느리고 비용도 많이 듭니다. 다른 데이터베이스 쿼리에 비해 원래 느리고 비용이 많이 드는 데이터베이스 쿼리도 있습니다. 예를 들어, 여러 테이블에서 조인을 수행하는 쿼리는 간단한 싱글 테이블 쿼리에 비해 훨씬 더 느리고 비용이 많이 소요됩니다. 속도가 느리고 비용이 많이 드는 쿼리를 통해서만 얻을 수 있는 데이터라면 캐시가 필요합니다. 비교적 빠르고 간단한 쿼리로 얻을 수 있는 데이터라도 다른 요인에 따라 캐싱이 필요할 수 있습니다.

데이터 및 액세스 패턴 - 캐시할 항목을 결정할 때는 데이터 자체와 액세스 패턴을 이해해야 합니다. 예를 들어, 빠르게 변하거나 거의 액세스하지 않는 데이터는 캐시할 필요가 없습니다. 캐시를 통해 실질적인 이점을 누리려면 비교적 정적이고 자주 액세스하는 데이터여야 합니다. 소셜 미디어 사이트의 개인 프로필을 예로 들 수 있습니다. 반대로, 캐시해도 속도나 비용이 나아지지 않는다면 데이터를 캐시할 필요가 없습니다. 예를 들어 검색 결과를 반환하는 웹 페이지의 쿼리와 결과는 대부분 고유하기 때문에 그러한 웹 페이지를 캐시하는 것은 의미가 없습니다.

기한 경과 - 정의하자면 캐시된 데이터는 기한이 경과한 데이터입니다. 경우에 따라 데이터 기한이 지나지 않았더라도 언제나 기한이 지난 데이터로 간주하고 취급해야 합니다. 데이터를 캐시할지 여부를 결정할 때는 애플리케이션의 데이터 기한 경과 허용 범위를 확인해야 합니다.

애플리케이션에서 기한이 지난 데이터를 허용할 수 있는 상황도 있고 그렇지 않은 상황도 있습니다. 예를 들어 사이트에서 공개적으로 거래된 주가를 제공한다고 가정합니다. 고객이 가격이 n분 지연될 수 있다는 고지 사항과 함께 어느 정도의 기한이 경과할 수 있음을 받아들일 수 있습니다. 하지만 주식을 매각하거나 매입하는 중개인에게 주식의 가격을 제공할 때는 실시간 데이터가 필요합니다.

다음에 해당하는 경우 데이터 캐시를 고려하세요.

- 캐시 검색에 비해 느린 속도와 높은 비용으로 데이터를 얻습니다.
- 사용자가 자주 데이터에 액세스합니다.
- 데이터가 비교적 동일하게 유지되거나 빠르게 변경되어도 기한 경과가 큰 문제가 되지 않습니다.

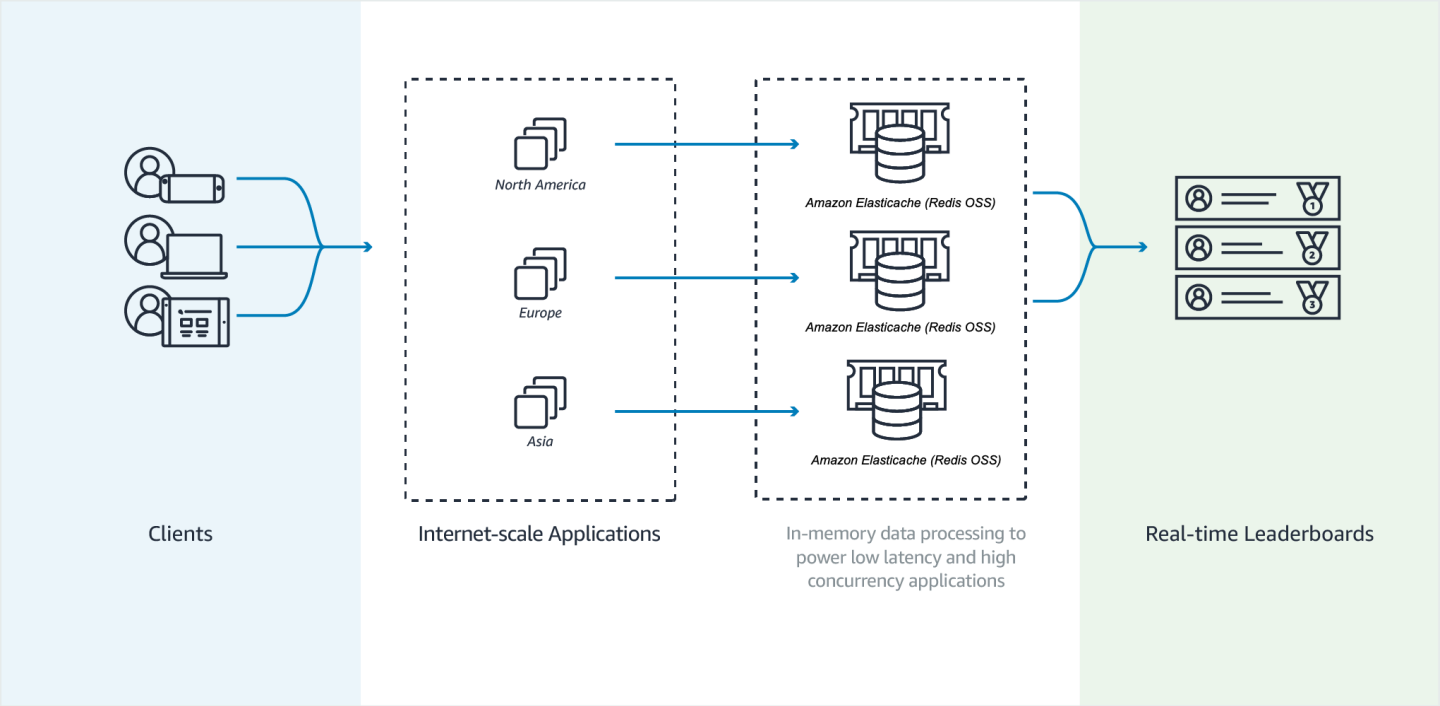
자세한 내용은 Memcached에 대한 캐싱 전략 단원을 참조하세요.

게임 리더보드

Valkey 또는 Redis OSS 정렬 세트를 사용하면 리더보드의 계산 복잡성을 애플리케이션에서 클러스터로 이동할 수 있습니다.

게임의 상위 10개 점수와 같은 리더보드는 계산적으로 복잡합니다. 이는 특히 동시 플레이어가 많고 점수가 지속적으로 바뀌는 경우에 그렇습니다. Valkey 및 Redis OSS 정렬 세트는 고유성과 요소 순서를 모두 보장합니다. 정렬된 세트를 사용하면 정렬된 세트에 새 요소가 추가될 때마다 실시간으로 순위가 다시 매겨집니다. 그다음에 올바른 숫자 순서의 세트에 해당 요소가 추가됩니다.

다음 다이어그램에서 ElastiCache 게임 리더보드의 작동 방식을 확인할 수 있습니다.



Example Valkey 또는 Redis OSS 리더보드

이 예제에서는 ZADD를 사용하여 게이머 4명과 이들의 점수를 정렬 목록에 입력합니다. ZREVRANGEBYSCORE 명령이 높은 점수에서 낮은 점수 순서로 플레이어를 나열합니다. 그런 다음 ZADD를 사용해 기존의 항목을 덮어써 June의 점수를 업데이트합니다. 마지막으로, ZREVRANGEBYSCORE에서 높은 점수에서 낮은 점수 순서로 플레이어를 나열합니다. 이 목록에서는 June이 순위가 상승했음을 알 수 있습니다.

```
ZADD leaderboard 132 Robert
ZADD leaderboard 231 Sandra
ZADD leaderboard 32 June
```

```
ZADD leaderboard 381 Adam

ZREVRANGEBYSCORE leaderboard +inf -inf
1) Adam
2) Sandra
3) Robert
4) June

ZADD leaderboard 232 June

ZREVRANGEBYSCORE leaderboard +inf -inf
1) Adam
2) June
3) Sandra
4) Robert
```

다음 명령을 통해 June은 모든 플레이어 중 자신의 순위를 알 수 있습니다. 순위는 0기반이므로 는 6월에 두 번째 위치에 있는 1을 ZREVRANK 반환합니다.

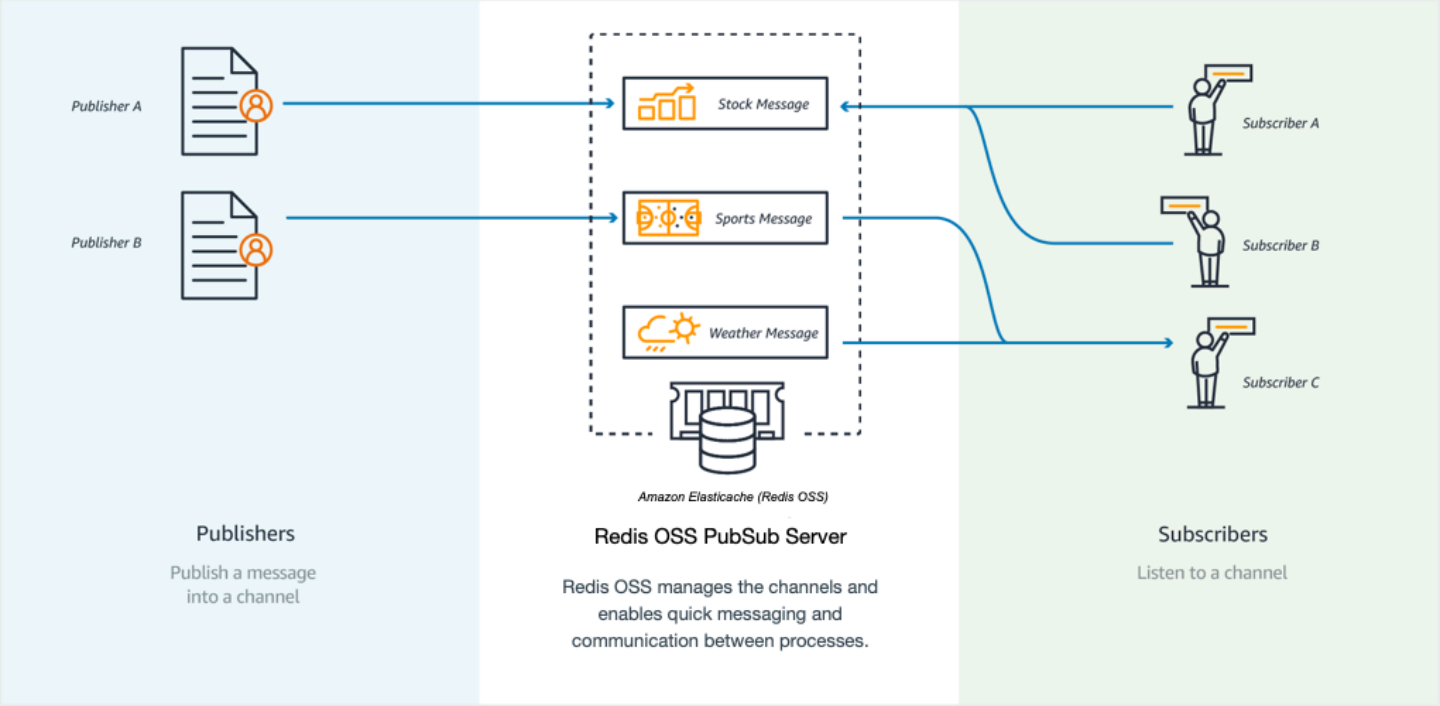
```
ZREVRANK leaderboard June
1
```

자세한 내용은 정렬된 세트에 대한 Valkey 설명서를 참조하세요.

메시징(Pub/Sub)

이메일 메시지를 보낼 때 지정된 수신자 한 명 이상에게 메시지가 전송됩니다. Valkey 및 Redis OSS pub/sub 패러다임에서 누가 수신할지 모르는 특정 채널에 메시지를 보냅니다. 메시지를 받는 사람은 채널을 구독하는 사람입니다. 예를 들어 news.sports.golf 채널을 구독한다고 가정해 보겠습니다. news.sports.golf 채널을 구독하는 모든 사람이 news.sports.golf에 게시되는 메시지를 받게 됩니다.

Pub/sub 기능은 키 스페이스와 관련이 없습니다. 어떤 수준에서도 간섭이 발생하지 않습니다. 다음 다이어그램에서 Valkey 및 Redis 를 사용한 메시징 그림 ElastiCache을 찾을 수 있습니다OSS.



구독

채널의 메시지를 받으려면 해당 채널을 구독합니다. 단일 채널, 지정된 여러 채널 또는 패턴에 맞는 모든 채널을 구독할 수 있습니다. 구독을 취소하려면 구독할 때 지정한 채널에서 구독을 취소해야 합니다. 또는 패턴 일치를 사용하여 구독한 경우 이전에 사용했던 패턴과 동일한 패턴으로 구독을 취소합니다.

Example - 단일 채널 구독

단일 채널을 구독하려면 구독하려는 채널을 지정하는 SUBSCRIBE 명령을 사용합니다. 다음 예제에서는 클라이언트가 news.sports.golf 채널을 구독합니다.

```
SUBSCRIBE news.sports.golf
```

잠시 후 클라이언트는 구독을 취소할 채널을 지정하는 UNSUBSCRIBE 명령을 사용하여 채널에 대한 구독을 취소합니다.

```
UNSUBSCRIBE news.sports.golf
```

Example - 지정된 여러 채널 구독

여러 특정 채널을 구독하려면 SUBSCRIBE 명령을 사용하여 채널을 나열합니다. 다음 예제에서는 클라이언트가 news.sports.golf, news.sports.soccer 및 news.sports.skiing 채널을 구독합니다.

```
SUBSCRIBE news.sports.golf news.sports.soccer news.sports.skiing
```

특정 채널에 대한 구독을 취소하려면 UNSUBSCRIBE 명령을 사용하고 구독을 취소할 채널을 지정합니다.

```
UNSUBSCRIBE news.sports.golf
```

여러 채널에 대한 구독을 취소하려면 UNSUBSCRIBE 명령을 사용하고 구독을 취소할 채널을 지정합니다.

```
UNSUBSCRIBE news.sports.golf news.sports.soccer
```

모든 구독을 취소하려면 UNSUBSCRIBE를 사용하고 각 채널을 지정합니다. 또는 UNSUBSCRIBE를 사용하고 채널을 지정하지 않습니다.

```
UNSUBSCRIBE news.sports.golf news.sports.soccer news.sports.skiing
```

또는

```
UNSUBSCRIBE
```

Example - 패턴 일치를 사용하는 구독

클라이언트는 PSUBSCRIBE 명령을 사용하여 패턴과 일치하는 모든 채널을 구독할 수 있습니다.

다음 예제에서는 클라이언트가 모든 스포츠 채널을 구독합니다. SUBSCRIBE를 사용할 때처럼 모든 스포츠 채널을 개별적으로 나열하는 것은 아닙니다. 대신 PSUBSCRIBE 명령을 사용하여 패턴 일치를 사용합니다.

```
PSUBSCRIBE news.sports.*
```

Example 구독 취소

이 채널의 구독을 취소하려면 PUNSUBSCRIBE 명령을 사용하세요.

```
PUNSUBSCRIBE news.sports.*
```

> **Important**
>
> [P]SUBSCRIBE 명령과 [P]UNSUBSCRIBE 명령으로 전송된 채널 문자열이 일치해야 합니다. news.*에 PSUBSCRIBE한 후 news.sports.*에서 PUNSUBSCRIBE하거나 news.sports.golf에서 UNSUBSCRIBE할 수 없습니다.

게시

채널의 모든 구독자에게 메시지를 보내려면 채널과 메시지를 지정하는 PUBLISH 명령을 사용하세요. 다음 예제에서는 "It's Saturday and sunny. I'm headed to the links." 메시지를 news.sports.golf 채널에 게시합니다.

```
PUBLISH news.sports.golf "It's Saturday and sunny. I'm headed to the links."
```

클라이언트는 구독 중인 채널에 게시할 수 없습니다.

자세한 내용은 Valkey 설명서의 Pub/Sub를 참조하세요.

권장 데이터(해시)

Valkey 또는 RedisDECR에서 INCR 또는 를 사용하면 컴파일 권장 사항이 간단OSS해집니다. 사용자가 제품을 "좋아할" 때마다 item:productID:like 카운터가 증가하고 "싫어할" 때마다 item:productID:dislike 카운터가 증가합니다. 해시를 사용하면 제품을 좋아하거나 싫어하는 모든 사람의 목록을 유지할 수도 있습니다.

Example - 호불호

```
INCR item:38923:likes
HSET item:38923:ratings Susan 1
INCR item:38923:dislikes
HSET item:38923:ratings Tommy -1
```

ElastiCache 고객 추천

Airbnb, PBS, Esri 등과 같은 기업이 Amazon을 사용하여 고객 경험을 개선 ElastiCache 하여 비즈니스를 성장시키는 방법에 대해 알아보려면 다른 사람이 Amazon 를 사용하는 방법을 ElastiCache 참조하세요.

자습서 비디오에서 추가 ElastiCache 고객 사용 사례를 볼 수도 있습니다.

Amazon 시작하기 ElastiCache

이 섹션의 실습 자습서를 사용하여 사용을 시작하고 사용에 대해 자세히 알아봅니다 ElastiCache.

주제

설정 ElastiCache

ElastiCache 웹 서비스를 사용하려면 다음 단계를 따르세요.

주제

에 가입 AWS 계정

가 없는 경우 다음 단계를 AWS 계정완료하여 를 생성합니다.

에 가입하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/가입 을 엽니다.

2. 온라인 지시 사항을 따릅니다.

   등록 절차 중 전화를 받고 전화 키패드로 확인 코드를 입력하는 과정이 있습니다.

   에 가입하면 AWS 계정AWS 계정 루트 사용자가 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.

AWS 는 가입 프로세스가 완료된 후 확인 이메일을 보냅니다. 언제든지 https://aws.amazon.com/로 이동하여 내 계정 을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

에 가입한 후 일상적인 작업에 루트 사용자를 사용하지 않도록 를 AWS 계정보호하고, 를 AWS 계정 루트 사용자활성화하고 AWS IAM Identity Center, 관리 사용자를 생성합니다.

보안 AWS 계정 루트 사용자

1. 루트 사용자를 선택하고 AWS 계정 이메일 주소를 입력하여 계정 소유자AWS Management Console로 에 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

   루트 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 User Guide의 루트 사용자로 로그인을 참조하십시오.
2. 루트 사용자에 대해 다중 인증(MFA)을 켭니다.

   지침은 IAM 사용 설명서의 AWS 계정 루트 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 사용 설명서의 AWS IAM Identity Center설정을 참조하세요.
2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

   를 자격 증명 소스 IAM Identity Center 디렉터리 로 사용하는 방법에 대한 자습서는 AWS IAM Identity Center 사용 설명서의 기본값으로 사용자 액세스 구성을 IAM Identity Center 디렉터리 참조하세요.

관리 액세스 권한이 있는 사용자로 로그인

- IAM Identity Center 사용자로 로그인하려면 IAM Identity Center 사용자를 생성할 때 이메일 주소로 전송URL된 로그인을 사용합니다.

  IAM Identity Center 사용자를 사용하여 로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의 AWS 액세스 포털에 로그인을 참조하세요.

추가 사용자에게 액세스 권한 할당

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

   지침은AWS IAM Identity Center 사용 설명서의 Create a permission set를 참조하세요.
2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

   지침은AWS IAM Identity Center 사용 설명서의 Add groups를 참조하세요.

프로그래밍 방식 액세스 권한 부여

사용자는 AWS 외부에서 와 상호 작용하려는 경우 프로그래밍 방식으로 액세스해야 합니다 AWS Management Console. 프로그래밍 방식 액세스를 부여하는 방법은 에 액세스하는 사용자 유형에 따라 다릅니다 AWS.

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

프로그래밍 방식 액세스가 필요한 사용자는 누구인가요?	To	액세스 권한을 부여하는 사용자
작업 인력 ID (IAMID 센터에서 관리하는 사용자)	임시 자격 증명을 사용하여 AWS CLI AWS SDKs, 또는 에 대한 프로그래밍 요청에 서명합니다 AWS APIs.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • 의 경우 AWS Command Line Interface 사용 설명서의 AWS CLI 를 사용하도록 구성을 AWS IAM Identity Center AWS CLI참조하세요. • AWS SDKs, 도구 및 의 경우 AWS SDKs 및 도구 참조 가

프로그래밍 방식 액세스가 필요한 사용자는 누구인가요?	To	액세스 권한을 부여하는 사용자
		이드의 IAM Identity Center 인증을 AWS APIs참조하세요.
IAM	임시 자격 증명을 사용하여 AWS CLI AWS SDKs, 또는 에 대한 프로그래밍 요청에 서명합니다 AWS APIs.	IAM 사용 설명서의 AWS 리소스와 함께 임시 자격 증명 사용의 지침을 따릅니다.
IAM	(권장되지 않음) 장기 보안 인증 정보를 사용하여 AWS CLI AWS SDKs, 또는 에 대한 프로그래밍 요청에 서명합니다 AWS APIs.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • 의 경우 AWS Command Line Interface 사용 설명서의 IAM 사용자 자격 증명을 사용하여 인증을 AWS CLI참조하세요. • 및 도구에 대한 AWS SDKs 자세한 내용은 AWS SDKs 및 도구 참조 가이드의 장기 보안 인증 정보를 사용하여 인증을 참조하세요. • 의 경우 IAM 사용 설명서의 IAM 사용자에 대한 액세스 키 관리를 AWS APIs참조하세요.

관련 항목:

- IAM 사용 설명서의 내용IAM.
- AWS 일반 참조AWS 의 보안 자격 증명.

권한 설정(신규 ElastiCache 사용자만 해당)

액세스 권한을 제공하려면 사용자, 그룹 또는 역할에 권한을 추가하세요:

- 의 사용자 및 그룹 AWS IAM Identity Center:

  권한 세트를 생성합니다. AWS IAM Identity Center 사용 설명서의 권한 세트 생성의 지침을 따릅니다.

- 자격 증명 공급자를 IAM 통해 에서 관리되는 사용자:

  ID 페더레이션을 위한 역할을 생성합니다. IAM 사용 설명서의 타사 자격 증명 공급자(페더레이션)에 대한 역할 생성의 지침을 따릅니다.

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

Amazon은 서비스 연결 역할을 ElastiCache 생성하고 사용하여 리소스를 프로비저닝하고 사용자를 대신하여 다른 AWS 리소스 및 서비스에 액세스합니다. 에서 서비스 연결 역할을 생성 ElastiCache 하려면 라는 AWS관리형 정책을 사용합니다`AmazonElastiCacheFullAccess`. 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 호출할 권한이 `iam:createServiceLinkedRole` 있거나 서비스 연결 역할을 생성 ElastiCache했는지 확인합니다.

자세한 내용은 다음 자료를 참조하세요.

- 새 정책 생성(IAM)
- AWS Amazon용 관리형 정책 ElastiCache
- Amazon용 서비스 연결 역할 사용 ElastiCache

설정 EC2

캐시에 연결할 EC2 인스턴스를 설정해야 합니다.

- 아직 EC2 인스턴스가 없는 경우 여기에서 EC2 인스턴스를 설정하는 방법을 알아봅니다. 시작하기 EC2.
- EC2 인스턴스는 캐시VPC와 동일한 보안 그룹 설정이어야 합니다. 기본적으로 Amazon은 기본값에 캐시를 ElastiCache 생성하고 기본 보안 그룹을 VPC 사용합니다. 이 자습서를 따르려면 EC2 인스턴스가 기본값이고 기본 보안 그룹이 VPC 있는지 확인하세요.

Amazon VPC 보안 그룹에서 캐시로 네트워크 액세스 권한 부여

ElastiCache 자체 설계된 클러스터는 Valkey 및 Redis OSS 명령에 포트 6379를 사용하고 서버 ElastiCache 리스는 포트 6379와 포트 6380을 모두 사용합니다. EC2 인스턴스에서 Valkey 또는 Redis OSS 명령을 성공적으로 연결하고 실행하려면 보안 그룹이 필요에 따라 이러한 포트에 대한 액세스를 허용해야 합니다.

ElastiCache (Memcached)는 11211 및 11212 포트를 사용하여 Memcached 명령을 수락합니다. EC2 인스턴스에서 Memcached 명령을 성공적으로 연결하고 실행하려면 보안 그룹이 이러한 포트에 대한 액세스를 허용해야 합니다.

1. 에 로그인 AWS Command Line Interface 하고 Amazon EC2 콘솔 을 엽니다.
2. 탐색 창의 [Network & Security] 아래에서 [Security Groups]를 선택합니다.
3. 보안 그룹 목록에서 Amazon 의 보안 그룹을 선택합니다VPC. ElastiCache 사용할 보안 그룹을 생성하지 않는 한 이 보안 그룹의 이름은 기본 입니다.
4. 인바운드 탭을 선택한 후 다음과 같이 하세요.

    a. 편집을 선택합니다.
    b. 규칙 추가를 선택합니다.
    c. 유형 열에서 사용자 지정 TCP 규칙 을 선택합니다.
    d. Valkey 또는 Redis 를 사용하는 경우 포트 범위 상자에 OSS를 입력합니다6379.

        Memcached를 사용하는 경우 포트 범위 상자에 를 입력합니다11211.
    e. 소스 상자에서 포트 범위(0.0.0.0/0)가 있는 Anywhere를 선택하여 Amazon 내에서 시작하는 모든 Amazon EC2 인스턴스가 캐시에 연결할 VPC 수 있도록 합니다.
    f. ElastiCache 서버리스를 사용하는 경우 규칙 추가를 선택하여 다른 규칙을 추가합니다.
    g. 유형 열에서 사용자 지정 TCP 규칙을 선택합니다.
    h. ElastiCache (Redis OSS)를 사용하는 경우 포트 범위 상자에 를 입력합니다6380.

        ElastiCache (Memcached)를 사용하는 경우 포트 범위 상자에 를 입력합니다11212.

i. 소스 상자에서 포트 범위(0.0.0.0/0)가 있는 Anywhere를 선택하여 Amazon 내에서 시작하는 모든 Amazon EC2 인스턴스가 캐시에 연결할 VPC 수 있도록 합니다.

j. 저장을 선택합니다.

명령줄 액세스 다운로드 및 설정

valkey-cli 유틸리티를 다운로드하여 설치합니다.

를 Valkey ElastiCache 와 함께 사용하는 경우 valkey-cli 유틸리티가 유용할 수 있습니다. ElastiCache (Redis OSS)를 redis-cli와 함께 사용하는 경우 RedisOSS에서도 작동하므로 valkey-cli로 전환하는 것이 좋습니다.

1. 원하는 연결 유틸리티를 사용하여 Amazon EC2 인스턴스에 연결합니다. Amazon EC2 인스턴스에 연결하는 방법에 대한 지침은 Amazon EC2 시작 안내서를 참조하세요.
2. 설정에 적합한 명령을 실행하여 valkey-cli 유틸리티를 다운로드하고 설치합니다.

   Amazon Linux 2023

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

   Amazon Linux 2

   ```
   sudo amazon-linux-extras install epel -y
   sudo yum install gcc jemalloc-devel openssl-devel tcl tcl-devel -y
   wget https://github.com/valkey-io/valkey/archive/refs/tags/7.2.6.tar.gz
   tar xvzf valkey-7.2.6.tar.gz
   cd valkey-7.2.6
   make BUILD_TLS=yes
   ```

> **Note**
>
> - redis6 패키지를 설치하면 기본 암호화 지원과 함께 redis6-cli가 설치됩니다.
> - valkey-cli 또는 redis-cli를 설치할 TLS 때 에 대한 빌드 지원을 받는 것이 중요합니다. ElastiCache Serverless는 이 활성화된 경우에만 액세스할 수 TLS 있습니다.
> - 암호화되지 않은 클러스터에 연결하는 경우 `Build_TLS=yes` 옵션이 필요하지 않습니다.

Valkey 서버리스 캐시 생성

이 단계에서는 Amazon 에서 새 캐시를 생성합니다 ElastiCache.

AWS Management Console

ElastiCache 콘솔을 사용하여 새 캐시를 생성하려면:

1. 에 로그인 AWS Management Console 하고 를 엽니다https://console.aws.amazon.com/connect/.
2. 콘솔 왼쪽의 탐색 창에서 Valkey 캐시 를 선택합니다.
3. 콘솔 오른쪽에서 Valkey 캐시 생성을 선택합니다.
4. 캐시 설정에서 이름을 입력합니다. 필요에 따라 캐시에 대한 설명을 입력할 수 있습니다.
5. 기본 설정이 선택된 상태로 유지합니다.
6. 생성을 클릭하여 캐시를 생성합니다.
7. 캐시가 “ACTIVE” 상태가 되면 캐시에 데이터 쓰기 및 읽기를 시작할 수 있습니다.

AWS CLI

다음 AWS CLI 예제에서는 를 사용하여 새 캐시를 생성합니다 create-serverless-cache.

Linux

```
aws elasticache create-serverless-cache \
    --serverless-cache-name CacheName \
    --engine valkey
```

Windows

```
aws elasticache create-serverless-cache ^
    --serverless-cache-name CacheName ^
    --engine valkey
```

상태 필드 값은 CREATING으로 설정됩니다.

ElastiCache 가 캐시 생성을 완료했는지 확인하려면 describe-serverless-caches 명령을 사용합니다.

Linux

```
aws elasticache describe-serverless-caches --serverless-cache-name CacheName
```

Windows

```
aws elasticache describe-serverless-caches --serverless-cache-name CacheName
```

새 캐시를 생성한 후에는 캐시에 데이터 읽기 및 쓰기 섹션으로 이동합니다.

캐시에 데이터 읽기 및 쓰기

이 섹션에서는 Amazon EC2 인스턴스를 생성하여 연결할 수 있다고 가정합니다. 이 작업을 수행하는 방법에 대한 지침은 Amazon EC2 시작 안내서를 참조하세요.

또한 이 섹션에서는 캐시에 연결하는 EC2 인스턴스에 대한 VPC 액세스 및 보안 그룹 설정을 설정하고 EC2 인스턴스에 valkey-cli를 설정했다고 가정합니다. 이 단계에 대한 자세한 내용은 설정 ElastiCache 섹션을 참조하세요.

캐시 엔드포인트 확인

AWS Management Console

ElastiCache 콘솔을 사용하여 캐시의 엔드포인트를 찾으려면:

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 콘솔 왼쪽의 탐색 창에서 Valkey 캐시 를 선택합니다.
3. 콘솔의 오른쪽에서 앞서 생성한 캐시의 이름을 클릭합니다.
4. 캐시 세부 정보에서 캐시 엔드포인트를 찾아 복사합니다.

AWS CLI

다음 AWS CLI 예제는 명령을 사용하여 새 캐시의 엔드포인트를 describe-serverless-caches 찾는 방법을 보여줍니다. 명령을 실행한 후 '엔드포인트' 필드를 찾습니다.

Linux

```
aws elasticache describe-serverless-caches \
  --serverless-cache-name CacheName
```

Windows

```
aws elasticache describe-serverless-caches ^
  --serverless-cache-name CacheName
```

Valkey Cache(Linux)에 연결

이제 필요한 엔드포인트가 있으므로 EC2 인스턴스에 로그인하고 캐시에 연결할 수 있습니다. 다음 예제에서는 valkey-cli 유틸리티를 사용하여 클러스터에 연결합니다. 다음 명령은 캐시에 연결됩니다(참고: cache-endpoint를 이전 단계에서 검색한 엔드포인트로 교체).

```
src/valkey-cli -h cache-endpoint --tls -p 6379
set a "hello"          // Set key "a" with a string value and no expiration
OK
get a                  // Get value for key "a"
"hello"
```

Valkey Cache에 연결(Windows)

이제 필요한 엔드포인트가 있으므로 EC2 인스턴스에 로그인하고 캐시에 연결할 수 있습니다. 다음 예제에서는 valkey-cli 유틸리티를 사용하여 클러스터에 연결합니다. 다음 명령을 사용하여 캐시에 연결합니다. 명령 프롬프트를 열고 Valkey 또는 Redis OSS 디렉터리로 변경한 다음 명령을 실행합니다(참고: Cache_Endpoint를 이전 단계에서 검색한 엔드포인트로 교체).

```
c:\Valkey>valkey-cli -h Valkey_Cluster_Endpoint --tls -p 6379
set a "hello"          // Set key "a" with a string value and no expiration
OK
get a                  // Get value for key "a"
"hello"
```

이제 (선택 사항) 정리 섹션으로 이동합니다.

(선택 사항) 정리

생성한 Amazon ElastiCache 캐시가 더 이상 필요하지 않은 경우 삭제할 수 있습니다. 이렇게 하면 사용하지 않는 리소스에 요금이 청구되지 않습니다. ElastiCache 콘솔, AWS CLI또는 를 사용하여 캐시를 ElastiCache API 삭제할 수 있습니다.

AWS Management Console

콘솔을 사용하여 캐시를 삭제하려면 다음과 같이 하세요.

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 콘솔 왼쪽의 탐색 창에서 Valkey 캐시 를 선택합니다.
3. 삭제하려는 캐시 옆에 있는 라디오 버튼을 선택합니다.
4. 오른쪽 상단에서 작업을 선택하고 삭제를 선택합니다.
5. 캐시를 삭제하기 전에 필요에 따라 최종 스냅샷을 찍을 수도 있습니다.
6. 삭제 확인 화면에서 캐시 이름을 다시 입력하고 삭제를 선택하여 클러스터를 삭제하거나 취소를 선택하여 클러스터를 유지합니다.

캐시가 DELETING 상태로 전환되는 즉시 해당 캐시에 대한 요금 발생을 중지합니다.

AWS CLI

다음 AWS CLI 예제에서는 명령을 사용하여 캐시를 delete-serverless-cache 삭제합니다.

Linux

```
aws elasticache delete-serverless-cache \
  --serverless-cache-name CacheName
```

Windows

```
aws elasticache delete-serverless-cache ^
  --serverless-cache-name CacheName
```

상태 필드의 값은 로 설정되어 있습니다DELETING.

이제 다음 단계 섹션으로 이동합니다.

다음 단계

에 대한 자세한 내용은 다음 페이지를 ElastiCache 참조하세요.

- 작업 ElastiCache
- 크기 조정 ElastiCache
- Amazon에서 로깅 및 모니터링 ElastiCache
- ElastiCache 모범 사례 및 캐싱 전략
- 스냅샷 및 복원

- Amazon ElastiCache 이벤트 SNS 모니터링

Valkey 또는 Redis OSS 서버리스 캐시 생성

이 단계에서는 Amazon 에서 새 캐시를 생성합니다 ElastiCache.

AWS Management Console

ElastiCache 콘솔을 사용하여 새 캐시를 생성하려면:

1. 에 로그인 AWS Management Console 하고 를 엽니다https://console.aws.amazon.com/connect/.
2. 콘솔 왼쪽의 탐색 창에서 Valkey 캐시 또는 Redis OSS 캐시를 선택합니다.
3. 콘솔 오른쪽에서 Valkey 캐시 생성 또는 Redis OSS 캐시 생성을 선택합니다.
4. 캐시 설정에서 이름을 입력합니다. 필요에 따라 캐시에 대한 설명을 입력할 수 있습니다.
5. 기본 설정이 선택된 상태로 유지합니다.
6. 생성을 클릭하여 캐시를 생성합니다.
7. 캐시가 “ACTIVE” 상태가 되면 캐시에 데이터 쓰기 및 읽기를 시작할 수 있습니다.

AWS CLI

다음 AWS CLI 예제에서는 를 사용하여 새 캐시를 생성합니다 create-serverless-cache.

Linux

```
aws elasticache create-serverless-cache \
    --serverless-cache-name CacheName \
    --engine redis
```

Windows

```
aws elasticache create-serverless-cache ^
    --serverless-cache-name CacheName ^
    --engine redis
```

상태 필드 값은 CREATING으로 설정됩니다.

ElastiCache 가 캐시 생성을 완료했는지 확인하려면 describe-serverless-caches 명령을 사용합니다.

Linux

```
aws elasticache describe-serverless-caches --serverless-cache-name CacheName
```

Windows

```
aws elasticache describe-serverless-caches --serverless-cache-name CacheName
```

새 캐시를 생성한 후에는 캐시에 데이터 읽기 및 쓰기 섹션으로 이동합니다.

캐시에 데이터 읽기 및 쓰기

이 섹션에서는 Amazon EC2 인스턴스를 생성하여 연결할 수 있다고 가정합니다. 이 작업을 수행하는 방법에 대한 지침은 Amazon EC2 시작 안내서를 참조하세요.

또한 이 섹션에서는 캐시에 연결하는 EC2 인스턴스에 대한 VPC 액세스 및 보안 그룹 설정을 설정하고 EC2 인스턴스에 valkey-cli를 설정했다고 가정합니다. 이 단계에 대한 자세한 내용은 설정 ElastiCache 섹션을 참조하세요.

캐시 엔드포인트 확인

AWS Management Console

ElastiCache 콘솔을 사용하여 캐시의 엔드포인트를 찾으려면:

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 콘솔 왼쪽의 탐색 창에서 Valkey 캐시 Redis OSS 캐시 를 선택합니다.
3. 콘솔의 오른쪽에서 앞서 생성한 캐시의 이름을 클릭합니다.
4. 캐시 세부 정보에서 캐시 엔드포인트를 찾아 복사합니다.

AWS CLI

다음 AWS CLI 예제는 명령을 사용하여 새 캐시의 엔드포인트를 describe-serverless-caches 찾는 방법을 보여줍니다. 명령을 실행한 후 '엔드포인트' 필드를 찾습니다.

Linux

```
aws elasticache describe-serverless-caches \
```

```
  --serverless-cache-name CacheName
```

Windows

```
aws elasticache describe-serverless-caches ^
  --serverless-cache-name CacheName
```

Valkey 또는 Redis OSS Cache(Linux)에 연결

이제 필요한 엔드포인트가 있으므로 EC2 인스턴스에 로그인하고 캐시에 연결할 수 있습니다. 다음 예제에서는 valkey-cli 유틸리티를 사용하여 클러스터에 연결합니다. 다음 명령은 캐시에 연결됩니다(참고: cache-endpoint를 이전 단계에서 검색한 엔드포인트로 교체).

```
src/valkey-cli -h cache-endpoint --tls -p 6379
set a "hello"          // Set key "a" with a string value and no expiration
OK
get a                  // Get value for key "a"
"hello"
```

Valkey 또는 Redis OSS Cache에 연결(Windows)

이제 필요한 엔드포인트가 있으므로 EC2 인스턴스에 로그인하고 캐시에 연결할 수 있습니다. 다음 예제에서는 valkey-cli 유틸리티를 사용하여 클러스터에 연결합니다. 다음 명령을 사용하여 캐시에 연결합니다. 명령 프롬프트를 열고 Valkey 디렉터리로 변경하고 명령을 실행합니다(참고: Cache_Endpoint를 이전 단계에서 검색한 엔드포인트로 교체).

```
c:\Redis>valkey-cli -h Redis_Cluster_Endpoint --tls -p 6379
set a "hello"          // Set key "a" with a string value and no expiration
OK
get a                  // Get value for key "a"
"hello"
```

이제 (선택 사항) 정리 섹션으로 이동합니다.

(선택 사항) 정리

생성한 Amazon ElastiCache 캐시가 더 이상 필요하지 않은 경우 삭제할 수 있습니다. 이렇게 하면 사용하지 않는 리소스에 요금이 청구되지 않습니다. ElastiCache 콘솔, AWS CLI또는 를 사용하여 캐시를 ElastiCache API 삭제할 수 있습니다.

AWS Management Console

콘솔을 사용하여 캐시를 삭제하려면 다음과 같이 하세요.

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 콘솔 왼쪽의 탐색 창에서 Valkey 또는 Redis OSS Caches를 선택합니다.
3. 삭제하려는 캐시 옆에 있는 라디오 버튼을 선택합니다.
4. 오른쪽 상단에서 작업을 선택하고 삭제를 선택합니다.
5. 캐시를 삭제하기 전에 필요에 따라 최종 스냅샷을 찍을 수도 있습니다.
6. 삭제 확인 화면에서 캐시 이름을 다시 입력하고 삭제를 선택하여 클러스터를 삭제하거나 취소를 선택하여 클러스터를 유지합니다.

캐시가 DELETING 상태로 전환되는 즉시 해당 캐시에 대한 요금 발생을 중지합니다.

AWS CLI

다음 AWS CLI 예제에서는 명령을 사용하여 캐시를 delete-serverless-cache 삭제합니다.

Linux

```
aws elasticache delete-serverless-cache \
  --serverless-cache-name CacheName
```

Windows

```
aws elasticache delete-serverless-cache ^
  --serverless-cache-name CacheName
```

상태 필드의 값은 로 설정되어 있습니다DELETING.

이제 다음 단계 섹션으로 이동합니다.

다음 단계

에 대한 자세한 내용은 다음 페이지를 ElastiCache 참조하세요.

- 작업 ElastiCache

- 크기 조정 ElastiCache
- Amazon에서 로깅 및 모니터링 ElastiCache
- ElastiCache 모범 사례 및 캐싱 전략
- 스냅샷 및 복원
- Amazon ElastiCache 이벤트 SNS 모니터링

Memcached 서버리스 캐시 생성

AWS Management Console

ElastiCache 콘솔을 사용하여 새 Memcached 서버리스 캐시를 생성하려면:

1. 에 로그인 AWS Management Console 하고 에서 ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 콘솔 왼쪽의 탐색 창에서 Memcached 캐시를 선택합니다.
3. 콘솔의 오른쪽에서 Memcached 캐시 생성을 선택합니다.
4. 캐시 설정에서 이름을 입력합니다. 필요에 따라 캐시에 대한 설명을 입력할 수 있습니다.
5. 기본 설정이 선택된 상태로 유지합니다.
6. 생성을 클릭하여 캐시를 생성합니다.
7. 캐시가 “ACTIVE” 상태가 되면 캐시에 데이터 쓰기 및 읽기를 시작할 수 있습니다.

를 사용하여 새 캐시를 생성하려면 AWS CLI

다음 AWS CLI 예제에서는 를 사용하여 새 캐시를 생성합니다 create-serverless-cache.

Linux

```
aws elasticache create-serverless-cache \
  --serverless-cache-name CacheName \
  --engine memcached
```

Windows

```
aws elasticache create-serverless-cache ^
  --serverless-cache-name CacheName ^
  --engine memcached
```

상태 필드 값은 CREATING으로 설정됩니다.

ElastiCache 가 캐시 생성을 완료했는지 확인하려면 describe-serverless-caches 명령을 사용합니다.

Linux

```
aws elasticache describe-serverless-caches --serverless-cache-name CacheName
```

Windows

```
aws elasticache describe-serverless-caches --serverless-cache-name CacheName
```

새 캐시를 생성한 후에는 캐시에 데이터 읽기 및 쓰기 섹션으로 이동합니다.

캐시에 데이터 읽기 및 쓰기

이 섹션에서는 Amazon EC2 인스턴스를 생성하여 연결할 수 있다고 가정합니다. 이 작업을 수행하는 방법에 대한 지침은 Amazon EC2 시작 안내서를 참조하세요.

기본적으로 는 기본 에서 캐시를 ElastiCache 생성합니다VPC. 캐시에 연결할 수 VPC있도록 EC2 인스턴스도 기본 에 생성되어 있는지 확인합니다.

캐시 엔드포인트 확인

AWS Management Console

ElastiCache 콘솔을 사용하여 캐시의 엔드포인트를 찾으려면:

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 콘솔 왼쪽의 탐색 창에서 Memcached 캐시를 선택합니다.
3. 콘솔의 오른쪽에서 앞서 생성한 캐시의 이름을 클릭합니다.
4. 캐시 세부 정보에서 캐시 엔드포인트를 찾아 복사합니다.

AWS CLI

다음 AWS CLI 예제는 명령을 사용하여 새 캐시의 엔드포인트를 describe-serverless-caches 찾는 방법을 보여줍니다. 명령을 실행한 후 '엔드포인트' 필드를 찾습니다.

Linux

```
aws elasticache describe-serverless-caches \
  --serverless-cache-name CacheName
```

Windows

```
aws elasticache describe-serverless-caches ^
  --serverless-cache-name CacheName
```

열기를 사용하여 연결SSL

열기를 사용하여 연결하는 방법에 대한 자세한 내용은 섹션을 SSL참조하세요. ElastiCache 전송 중 암호화(TLS)

Memcached Java 클라이언트를 사용하여 연결

Memcached Java 클라이언트를 사용하여 연결하는 방법에 대한 자세한 내용은 ElastiCache 전송 중 암호화(TLS) 섹션을 참조하세요.

Memcached PHP 클라이언트를 사용하여 연결

```
<?php
$cluster_endpoint = "mycluster.serverless.use1.cache.amazonaws.com";
$server_port = 11211;

/* Initialize a persistent Memcached client in TLS mode */
$tls_client = new Memcached('persistent-id');
$tls_client->addServer($cluster_endpoint, $server_port);
if(!$tls_client->setOption(Memcached::OPT_USE_TLS, 1)) {
    echo $tls_client->getLastErrorMessage(), "\n";
    exit(1);
}
$tls_config = new MemcachedTLSContextConfig();
$tls_config->hostname = '*.serverless.use1.cache.amazonaws.com';
$tls_config->skip_cert_verify = false;
$tls_config->skip_hostname_verify = false;
$tls_client->createAndSetTLSContext((array)$tls_config);

 /* store the data for 60 seconds in the cluster */
$tls_client->set('key', 'value', 60);
?>
```

Memcached Python 클라이언트(Pymemcache)를 사용하여 연결

https://pymemcache.readthedocs.io/en/latest/getting_started.html 참조하세요.

```
import ssl
from pymemcache.client.base import Client

context = ssl.create_default_context()
cluster_endpoint = <To be taken from the AWS CLI / console>
target_port = 11211
memcached_client = Client(("{cluster_endpoint}", target_port), tls_context=context)
memcached_client.set("key", "value", expire=500, noreply=False)
assert self.memcached_client.get("key").decode() == "value"
```

Memcached NodeJS/TS 클라이언트(전극 I/O Memcache)를 사용하여 연결

https://github.com/electrode-io/memcache 및 https://www.npmjs.com/package/memcache-client 보기

`npm i memcache-client`를 통해 설치합니다.

애플리케이션에서 다음과 같이 memcached TLS 클라이언트를 생성합니다.

```
var memcache = require("memcache-client");
const client = new memcache.MemcacheClient({server: "{cluster_endpoint}:11211", tls:
 {}});
client.set("key", "value");
```

Memcached Rust 클라이언트를 사용하여 연결(rust-memcache)

https://crates.io/crates/memcache 및 https://github.com/aisk/rust-memcache를 참조하세요.

```
// create connection with to memcached server node:
let client = memcache::connect("memcache+tls://<cluster_endpoint>:11211?
verify_mode=none").unwrap();

// set a string value
client.set("foo", "bar", 0).unwrap();
```

Memcached Go 클라이언트를 사용하여 연결(Gomemcache)

https://github.com/bradfitz/gomemcache 보기

```
c := New(net.JoinHostPort("{cluster_endpoint}", strconv.Itoa(port)))
c.DialContext = func(ctx context.Context, network, addr string) (net.Conn, error) {
var td tls.Dialer
td.Config = &tls.Config{}
return td.DialContext(ctx, network, addr)
}
foo := &Item{Key: "foo", Value: []byte("fooval"), Flags: 123}
err := c.Set(foo)
```

Memcached Ruby 클라이언트를 사용하여 연결(Dalli)

https://github.com/petergoldstein/달리 참조

```
require 'dalli'
ssl_context = OpenSSL::SSL::SSLContext.new
ssl_context.ssl_version = :SSLv23
ssl_context.verify_hostname = true
ssl_context.verify_mode = OpenSSL::SSL::VERIFY_PEER
client = Dalli::Client.new("<cluster_endpoint>:11211", :ssl_context => ssl_context);
client.get("abc")
```

Memcached .NET 클라이언트(EnyimMemcachedCore)를 사용하여 연결

참조 https://github.com/cnblogs/EnyimMemcachedCore

```
"MemcachedClient": {
"Servers": [
{
"Address": "{cluster_endpoint}",
"Port": 11211
}
],
"UseSslStream":  true
}
```

이제 (선택 사항) 정리 섹션으로 이동합니다.

(선택 사항) 정리

사용 AWS Management Console

다음은 배포에서 캐시 하나를 삭제하는 절차입니다. 캐시를 여러 개 삭제하려면 삭제할 캐시마다 절차를 반복합니다. 캐시 하나를 다 삭제한 후 다른 캐시 삭제 절차가 시작될 때까지 기다릴 필요는 없습니다.

캐시를 삭제하려면 다음과 같이 하세요.

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. ElastiCache 콘솔 대시보드에서 삭제하려는 캐시가 실행 중인 엔진을 선택합니다. 해당 엔진을 실행하고 있는 모든 캐시의 목록이 표시됩니다.
3. 삭제할 캐시를 선택하려면 캐시 목록에서 캐시 이름을 선택합니다.

   > **Important**
   >
   > ElastiCache 콘솔에서 한 번에 하나의 캐시만 삭제할 수 있습니다. 여러 캐시를 선택하면 삭제 작업이 비활성화됩니다.

4. 작업에 대해 삭제를 선택합니다.
5. 캐시 삭제 확인 화면에서 삭제를 선택하여 캐시를 삭제하거나 취소를 선택하여 클러스터를 유지합니다.
6. 삭제를 선택한 경우 캐시 상태가 삭제 중으로 바뀝니다.

캐시가 DELETING 상태로 전환되는 즉시 해당 캐시에 대한 요금 발생을 중지합니다.

사용 AWS CLI

다음 코드는 my-cache 캐시를 삭제합니다.

```
aws elasticache delete-serverless-cache --serverless-cache-name my-cache
```

CLI 작업은 delete-serverless-cache 서버리스 캐시 하나만 삭제합니다. 여러 캐시를 삭제하려면 삭제하려는 각 서버리스 캐시에 대해 를 호출 delete-serverless-cache합니다. 서버리스 캐시 하나를 다 삭제한 후 다른 캐시를 삭제할 때까지 기다릴 필요는 없습니다.

Linux, macOS, Unix의 경우:

```
aws elasticache delete-serverless-cache \
  --serverless-cache-name my-cache
```

Windows의 경우:

```
aws elasticache delete-serverless-cache ^
  --serverless-cache-name my-cache
```

자세한 내용은 ElastiCache 주제 의 섹션을 참조 AWS CLI 하세요 delete-serverless-cache.

이제 다음 단계 섹션으로 이동합니다.

다음 단계

에 대한 자세한 내용은 다음을 ElastiCache 참조하세요.

- 작업 ElastiCache
- 크기 조정 ElastiCache
- 에 대한 할당량 ElastiCache
- ElastiCache 모범 사례 및 캐싱 전략
- ElastiCache 이벤트 보기

자습서: Python 및 시작하기 ElastiCache

이 섹션에는 Valkey 및 Redis 에 대해 배우는 데 도움이 되는 실습 자습서 ElastiCache 가 포함되어 있습니다OSS. 언어별 자습서에 따라 학습하는 것을 권장합니다.

> **Note**
>
> AWS SDKs 는 다양한 언어에서 사용할 수 있습니다. 완전한 목록은 Tools for Amazon Web Services를 참조하세요.

주제

- Python 및 ElastiCache

Python 및 ElastiCache

이 자습서에서는 for Python(Boto3)을 AWS SDK 사용하여 다음 ElastiCache (Redis OSS) 작업을 수행하는 간단한 프로그램을 작성합니다.

- 클러스터 생성 ElastiCache (Redis OSS)(클러스터 모드 활성화 및 클러스터 모드 비활성화)
- 사용자 또는 사용자 그룹이 존재하는지 확인하고 그렇지 않으면 생성합니다. (이 기능은 Valkey 7.2 이상 및 Redis OSS 6.0 이상에서 사용할 수 있습니다.)
- 에 연결 ElastiCache
- 문자열 설정 및 가져오기, 스트림에서 읽기 및 쓰기, 게시/구독 채널의 게시 및 구독과 같은 작업을 수행합니다.

이 자습서를 진행하면서 Python용(Boto) 설명서를 참조 AWS SDK할 수 있습니다. 다음 섹션은 에 따라 다릅니다 ElastiCache. ElastiCache 하위 수준 클라이언트

자습서 사전 요구 사항

- 를 사용하도록 AWS 액세스 키를 설정합니다 AWS SDKs. 자세한 내용은 설정 ElastiCache 단원을 참조하십시오.
- Python 3.0 이상을 설치합니다. 자세한 내용은 https://www.python.org/downloads 참조하세요. 지침은 Boto 3 설명서의 Quickstart 섹션을 참조하세요.

주제

- 자습서: ElastiCache 클러스터 및 사용자 생성
- 자습서: 에 연결 ElastiCache
- 사용 예제:

자습서: ElastiCache 클러스터 및 사용자 생성

다음 예제에서는 ElastiCache (Redis OSS) 관리 작업(클러스터 또는 사용자 생성)에 boto3를 사용하고 데이터 처리를 SDK 위해 redis-py/redis-py-cluster 를 사용합니다.

주제

- 클러스터 모드가 비활성화된 클러스터 생성
- TLS 및 를 사용하여 클러스터 모드 비활성화 클러스터 생성 RBAC

- 클러스터 모드가 활성화된 클러스터 생성
- TLS 및 를 사용하여 클러스터 모드 활성화 클러스터 생성 RBAC
- 사용자/사용자 그룹이 있는지 확인하고 없으면 생성

클러스터 모드가 비활성화된 클러스터 생성

다음 프로그램을 복사하여 CreateClusterModeDisabledCluster.py라는 파일에 붙여 넣습니다.

```
import boto3
import logging

logging.basicConfig(level=logging.INFO)
client = boto3.client('elasticache')

def
 create_cluster_mode_disabled(CacheNodeType='cache.t3.small',EngineVersion='6.0',NumCacheCluste
 cache cluster',ReplicationGroupId=None):
    """Creates an ElastiCache Cluster with cluster mode disabled

    Returns a dictionary with the API response

    :param CacheNodeType: Node type used on the cluster. If not specified,
 cache.t3.small will be used
    Refer to https://docs.aws.amazon.com/AmazonElastiCache/latest/dg/
CacheNodes.SupportedTypes.html for supported node types
    :param EngineVersion: Engine version to be used. If not specified, latest will be
 used.
    :param NumCacheClusters: Number of nodes in the cluster. Minimum 1 (just a primary
 node) and maximun 6 (1 primary and 5 replicas).
    If not specified, cluster will be created with 1 primary and 1 replica.
    :param ReplicationGroupDescription: Description for the cluster.
    :param ReplicationGroupId: Name for the cluster
    :return: dictionary with the API results

    """
    if not ReplicationGroupId:
        return 'ReplicationGroupId parameter is required'

    response = client.create_replication_group(
        AutomaticFailoverEnabled=True,
        CacheNodeType=CacheNodeType,
        Engine='valkey',
```

```
        EngineVersion=EngineVersion,
        NumCacheClusters=NumCacheClusters,
        ReplicationGroupDescription=ReplicationGroupDescription,
        ReplicationGroupId=ReplicationGroupId,
        SnapshotRetentionLimit=30,
    )
    return response


if __name__ == '__main__':

    # Creates an ElastiCache Cluster mode disabled cluster, based on cache.m6g.large
 nodes, Valkey 7.2, one primary and two replicas
    elasticacheResponse = create_cluster_mode_disabled(
        #CacheNodeType='cache.m6g.large',
        EngineVersion='7.2',
        NumCacheClusters=3,
        ReplicationGroupDescription='Valkey cluster mode disabled with replicas',
        ReplicationGroupId='valkey202104053'
        )

    logging.info(elasticacheResponse)
```

프로그램을 실행하려면 다음 명령을 입력합니다.

```
python CreateClusterModeDisabledCluster.py
```

자세한 내용은 에서 클러스터 관리 ElastiCache 단원을 참조하십시오.

TLS 및 를 사용하여 클러스터 모드 비활성화 클러스터 생성 RBAC

보안을 보장하기 위해 클러스터 모드 비활성화 클러스터를 생성할 때 전송 계층 보안(TLS) 및 역할 기반 액세스 제어(RBAC)를 사용할 수 있습니다. 토큰이 인증된 경우 모든 인증된 클라이언트OSSAUTH가 전체 복제 그룹 액세스 권한을 갖는 Valkey 또는 Redis와 달리 RBAC를 사용하면 사용자 그룹을 통해 클러스터 액세스를 제어할 수 있습니다. 이러한 사용자 그룹은 복제 그룹에 대한 액세스를 구성하는 방법으로 설계되었습니다. 자세한 내용은 역할 기반 액세스 제어(RBAC) 단원을 참조하십시오.

다음 프로그램을 복사하여 ClusterModeDisabledWithRBAC.py라는 파일에 붙여 넣습니다.

```
import boto3
import logging

logging.basicConfig(level=logging.INFO)
```

```
client = boto3.client('elasticache')

def
 create_cluster_mode_disabled_rbac(CacheNodeType='cache.t3.small',EngineVersion='6.0',NumCacheC
 cache cluster',ReplicationGroupId=None, UserGroupIds=None,
 SecurityGroupIds=None,CacheSubnetGroupName=None):
    """Creates an ElastiCache Cluster with cluster mode disabled and RBAC

    Returns a dictionary with the API response

    :param CacheNodeType: Node type used on the cluster. If not specified,
 cache.t3.small will be used
    Refer to https://docs.aws.amazon.com/AmazonElastiCache/latest/dg/
CacheNodes.SupportedTypes.html for supported node types
    :param EngineVersion: Engine version to be used. If not specified, latest will be
 used.
    :param NumCacheClusters: Number of nodes in the cluster. Minimum 1 (just a primary
 node) and maximun 6 (1 primary and 5 replicas).
    If not specified, cluster will be created with 1 primary and 1 replica.
    :param ReplicationGroupDescription: Description for the cluster.
    :param ReplicationGroupId: Mandatory name for the cluster.
    :param UserGroupIds: The ID of the user group to be assigned to the cluster.
    :param SecurityGroupIds: List of security groups to be assigned. If not defined,
 default will be used
    :param CacheSubnetGroupName: subnet group where the cluster will be placed. If not
 defined, default will be used.
    :return: dictionary with the API results

    """
    if not ReplicationGroupId:
        return {'Error': 'ReplicationGroupId parameter is required'}
    elif not isinstance(UserGroupIds,(list)):
        return {'Error': 'UserGroupIds parameter is required and must be a list'}

    params={'AutomaticFailoverEnabled': True,
            'CacheNodeType': CacheNodeType,
            'Engine': 'valkey',
            'EngineVersion': EngineVersion,
            'NumCacheClusters': NumCacheClusters,
            'ReplicationGroupDescription': ReplicationGroupDescription,
            'ReplicationGroupId': ReplicationGroupId,
            'SnapshotRetentionLimit': 30,
            'TransitEncryptionEnabled': True,
            'UserGroupIds':UserGroupIds
```

```
        }

    # defaults will be used if CacheSubnetGroupName or SecurityGroups are not explicit.
    if isinstance(SecurityGroupIds,(list)):
        params.update({'SecurityGroupIds':SecurityGroupIds})
    if CacheSubnetGroupName:
        params.update({'CacheSubnetGroupName':CacheSubnetGroupName})

    response = client.create_replication_group(**params)
    return response

if __name__ == '__main__':

    # Creates an ElastiCache Cluster mode disabled cluster, based on cache.m6g.large
 nodes, Valkey 7.2, one primary and two replicas.
    # Assigns the existent user group "mygroup" for RBAC authentication

    response=create_cluster_mode_disabled_rbac(
        CacheNodeType='cache.m6g.large',
        EngineVersion='7.2',
        NumCacheClusters=3,
        ReplicationGroupDescription='Valkey cluster mode disabled with replicas',
        ReplicationGroupId='valkey202104',
        UserGroupIds=[
            'mygroup'
        ],
        SecurityGroupIds=[
            'sg-7cc73803'
        ],
        CacheSubnetGroupName='default'
    )

    logging.info(response)
```

프로그램을 실행하려면 다음 명령을 입력합니다.

```
python ClusterModeDisabledWithRBAC.py
```

자세한 내용은 에서 클러스터 관리 ElastiCache 단원을 참조하십시오.

클러스터 모드가 활성화된 클러스터 생성

다음 프로그램을 복사하여 ClusterModeEnabled.py라는 파일에 붙여 넣습니다.

```
import boto3
import logging

logging.basicConfig(level=logging.INFO)
client = boto3.client('elasticache')

def
 create_cluster_mode_enabled(CacheNodeType='cache.t3.small',EngineVersion='6.0',NumNodeGroups=1
 ReplicationGroupDescription='Sample cache with cluster mode
 enabled',ReplicationGroupId=None):
    """Creates an ElastiCache Cluster with cluster mode enabled

    Returns a dictionary with the API response

    :param CacheNodeType: Node type used on the cluster. If not specified,
 cache.t3.small will be used
    Refer to https://docs.aws.amazon.com/AmazonElastiCache/latest/dg/
CacheNodes.SupportedTypes.html for supported node types
    :param EngineVersion: Engine version to be used. If not specified, latest will be
 used.
    :param NumNodeGroups: Number of shards in the cluster. Minimum 1 and maximun 90.
    If not specified, cluster will be created with 1 shard.
    :param ReplicasPerNodeGroup: Number of replicas per shard. If not specified 1
 replica per shard will be created.
    :param ReplicationGroupDescription: Description for the cluster.
    :param ReplicationGroupId: Name for the cluster
    :return: dictionary with the API results

    """
    if not ReplicationGroupId:
        return 'ReplicationGroupId parameter is required'

    response = client.create_replication_group(
        AutomaticFailoverEnabled=True,
        CacheNodeType=CacheNodeType,
        Engine='valkey',
        EngineVersion=EngineVersion,
        ReplicationGroupDescription=ReplicationGroupDescription,
        ReplicationGroupId=ReplicationGroupId,
    #   Creates a cluster mode enabled cluster with 1 shard(NumNodeGroups), 1 primary
 node (implicit) and 2 replicas (replicasPerNodeGroup)
        NumNodeGroups=NumNodeGroups,
        ReplicasPerNodeGroup=ReplicasPerNodeGroup,
```

```
        CacheParameterGroupName='default.valkey7.2.cluster.on'
    )

    return response


# Creates a cluster mode enabled
response = create_cluster_mode_enabled(
    CacheNodeType='cache.m6g.large',
    EngineVersion='6.0',
    ReplicationGroupDescription='Valkey cluster mode enabled with replicas',
    ReplicationGroupId='valkey20210',
#   Creates a cluster mode enabled cluster with 1 shard(NumNodeGroups), 1 primary
 (implicit) and 2 replicas (replicasPerNodeGroup)
    NumNodeGroups=2,
    ReplicasPerNodeGroup=1,
)

logging.info(response)
```

프로그램을 실행하려면 다음 명령을 입력합니다.

```
python ClusterModeEnabled.py
```

자세한 내용은 에서 클러스터 관리 ElastiCache 단원을 참조하십시오.

TLS 및 를 사용하여 클러스터 모드 활성화 클러스터 생성 RBAC

보안을 보장하기 위해 클러스터 모드 활성화 클러스터를 생성할 때 전송 계층 보안(TLS) 및 역할 기반 액세스 제어(RBAC)를 사용할 수 있습니다. 토큰이 인증된 경우 모든 인증된 클라이언트OSSAUTH가 전체 복제 그룹 액세스 권한을 갖는 Valkey 또는 Redis와 달리 RBAC를 사용하면 사용자 그룹을 통해 클러스터 액세스를 제어할 수 있습니다. 이러한 사용자 그룹은 복제 그룹에 대한 액세스를 구성하는 방법으로 설계되었습니다. 자세한 내용은 역할 기반 액세스 제어(RBAC) 단원을 참조하십시오.

다음 프로그램을 복사하여 ClusterModeEnabledWithRBAC.py라는 파일에 붙여 넣습니다.

```
import boto3
import logging

logging.basicConfig(level=logging.INFO)
client = boto3.client('elasticache')
```

```
def
 create_cluster_mode_enabled(CacheNodeType='cache.t3.small',EngineVersion='6.0',NumNodeGroups=1
 ReplicationGroupDescription='Sample cache with cluster
 mode enabled',ReplicationGroupId=None,UserGroupIds=None,
 SecurityGroupIds=None,CacheSubnetGroupName=None,CacheParameterGroupName='default.valkey7.2.clu
    """Creates an ElastiCache Cluster with cluster mode enabled and RBAC

    Returns a dictionary with the API response

    :param CacheNodeType: Node type used on the cluster. If not specified,
 cache.t3.small will be used
    Refer to https://docs.aws.amazon.com/AmazonElastiCache/latest/dg/
CacheNodes.SupportedTypes.html for supported node types
    :param EngineVersion: Engine version to be used. If not specified, latest will be
 used.
    :param NumNodeGroups: Number of shards in the cluster. Minimum 1 and maximun 90.
    If not specified, cluster will be created with 1 shard.
    :param ReplicasPerNodeGroup: Number of replicas per shard. If not specified 1
 replica per shard will be created.
    :param ReplicationGroupDescription: Description for the cluster.
    :param ReplicationGroupId: Name for the cluster.
    :param CacheParameterGroupName: Parameter group to be used. Must be compatible with
 the engine version and cluster mode enabled.
    :return: dictionary with the API results

    """
    if not ReplicationGroupId:
        return 'ReplicationGroupId parameter is required'
    elif not isinstance(UserGroupIds,(list)):
        return {'Error': 'UserGroupIds parameter is required and must be a list'}

    params={'AutomaticFailoverEnabled': True,
            'CacheNodeType': CacheNodeType,
            'Engine': 'valkey',
            'EngineVersion': EngineVersion,
            'ReplicationGroupDescription': ReplicationGroupDescription,
            'ReplicationGroupId': ReplicationGroupId,
            'SnapshotRetentionLimit': 30,
            'TransitEncryptionEnabled': True,
            'UserGroupIds':UserGroupIds,
            'NumNodeGroups': NumNodeGroups,
            'ReplicasPerNodeGroup': ReplicasPerNodeGroup,
            'CacheParameterGroupName': CacheParameterGroupName
        }
```

```
    # defaults will be used if CacheSubnetGroupName or SecurityGroups are not explicit.
    if isinstance(SecurityGroupIds,(list)):
        params.update({'SecurityGroupIds':SecurityGroupIds})
    if CacheSubnetGroupName:
        params.update({'CacheSubnetGroupName':CacheSubnetGroupName})

    response = client.create_replication_group(**params)
    return response

if __name__ == '__main__':
    # Creates a cluster mode enabled cluster
    response = create_cluster_mode_enabled(
        CacheNodeType='cache.m6g.large',
        EngineVersion='7.2',
        ReplicationGroupDescription='Valkey cluster mode enabled with replicas',
        ReplicationGroupId='valkey2021',
    #   Creates a cluster mode enabled cluster with 1 shard(NumNodeGroups), 1 primary
 (implicit) and 2 replicas (replicasPerNodeGroup)
        NumNodeGroups=2,
        ReplicasPerNodeGroup=1,
        UserGroupIds=[
            'mygroup'
        ],
        SecurityGroupIds=[
            'sg-7cc73803'
        ],
        CacheSubnetGroupName='default'

    )

    logging.info(response)
```

프로그램을 실행하려면 다음 명령을 입력합니다.

```
python ClusterModeEnabledWithRBAC.py
```

자세한 내용은 에서 클러스터 관리 ElastiCache 단원을 참조하십시오.

사용자/사용자 그룹이 있는지 확인하고 없으면 생성

를 사용하면 사용자를 RBAC생성하고 액세스 문자열을 사용하여 특정 권한을 할당할 수 있습니다. 사용자를 특정 역할(관리자, 인적 자원)과 정렬된 사용자 그룹에 할당한 다음 하나 이상의 ElastiCache (Redis OSS) 복제 그룹에 배포합니다. 이렇게 하면 동일한 Valkey 또는 Redis OSS 복제 그룹 또는 그

룹을 사용하여 클라이언트 간에 보안 경계를 설정하고 클라이언트가 서로의 데이터에 액세스하지 못하게 할 수 있습니다. 자세한 내용은 역할 기반 액세스 제어(RBAC) 단원을 참조하십시오.

다음 프로그램을 복사하여 UserAndUserGroups.py라는 파일에 붙여 넣습니다. 보안 인증 정보 제공 메커니즘을 업데이트합니다. 이 예시의 보안 인증 정보는 교체 가능한 것으로 표시되며 선언되지 않은 항목이 할당됩니다. 보안 인증 정보를 하드 코딩하지 마시기 바랍니다.

이 예제에서는 사용자에 대한 권한이 있는 액세스 문자열을 사용합니다. 액세스 문자열에 대한 자세한 내용은 섹션을 참조하세요액세스 문자열을 사용하여 권한 지정.

```
import boto3
import logging

logging.basicConfig(level=logging.INFO)
client = boto3.client('elasticache')

def check_user_exists(UserId):
    """Checks if UserId exists

    Returns True if UserId exists, otherwise False
    :param UserId: ElastiCache User ID
    :return: True|False
    """
    try:
        response = client.describe_users(
            UserId=UserId,
        )
        if response['Users'][0]['UserId'].lower() == UserId.lower():
            return True
    except Exception as e:
        if e.response['Error']['Code'] == 'UserNotFound':
            logging.info(e.response['Error'])
            return False
        else:
            raise

def check_group_exists(UserGroupId):
    """Checks if UserGroupID exists

    Returns True if Group ID exists, otherwise False
    :param UserGroupId: ElastiCache User ID
    :return: True|False
    """
```

```
    try:
        response = client.describe_user_groups(
            UserGroupId=UserGroupId
        )
        if response['UserGroups'][0]['UserGroupId'].lower() == UserGroupId.lower():
            return True
    except Exception as e:
        if e.response['Error']['Code'] == 'UserGroupNotFound':
            logging.info(e.response['Error'])
            return False
        else:
            raise

def create_user(UserId=None,UserName=None,Password=None,AccessString=None):
    """Creates a new user

    Returns the ARN for the newly created user or the error message
    :param UserId: ElastiCache user ID. User IDs must be unique
    :param UserName: ElastiCache user name. ElastiCache allows multiple users with the
 same name as long as the associated user ID is unique.
    :param Password: Password for user. Must have at least 16 chars.
    :param AccessString: Access string with the permissions for the user.
    :return: user ARN
    """
    try:
        response = client.create_user(
            UserId=UserId,
            UserName=UserName,
            Engine='Redis',
            Passwords=[Password],
            AccessString=AccessString,
            NoPasswordRequired=False
        )
        return response['ARN']
    except Exception as e:
        logging.info(e.response['Error'])
        return e.response['Error']

def create_group(UserGroupId=None, UserIds=None):
    """Creates a new group.
    A default user is required (mandatory) and should be specified in the UserIds list

    Return: Group ARN
```

```
    :param UserIds: List with user IDs to be associated with the new group. A default
 user is required
    :param UserGroupId: The ID (name) for the group
    :return: Group ARN
    """
    try:
        response = client.create_user_group(
            UserGroupId=UserGroupId,
            Engine='Redis',
            UserIds=UserIds
        )
        return response['ARN']
    except Exception as e:
        logging.info(e.response['Error'])


if __name__ == '__main__':

    groupName='mygroup2'
    userName = 'myuser2'
    userId=groupName+'-'+userName

    # Creates a new user if the user ID does not exist.
    for tmpUserId,tmpUserName in [ (userId,userName), (groupName+'-
default','default')]:
        if not check_user_exists(tmpUserId):
            response=create_user(UserId=tmpUserId,
 UserName=EXAMPLE,Password=EXAMPLE,AccessString='on ~* +@all')
            logging.info(response)
        # assigns the new user ID to the user group
    if not check_group_exists(groupName):
        UserIds = [ userId , groupName+'-default']
        response=create_group(UserGroupId=groupName,UserIds=UserIds)
        logging.info(response)
```

프로그램을 실행하려면 다음 명령을 입력합니다.

```
python UserAndUserGroups.py
```

자습서: 에 연결 ElastiCache

다음 예제에서는 Valkey 또는 Redis OSS 클라이언트를 사용하여 에 연결합니다 ElastiCache.

주제

클러스터 모드가 비활성화된 클러스터에 연결

다음 프로그램을 복사하여 ConnectClusterModeDisabled.py라는 파일에 붙여 넣습니다. 보안 인증 정보 제공 메커니즘을 업데이트합니다. 이 예시의 보안 인증 정보는 교체 가능한 것으로 표시되며 선언되지 않은 항목이 할당됩니다. 보안 인증 정보를 하드 코딩하지 마시기 바랍니다.

```
from redis import Redis
import logging

logging.basicConfig(level=logging.INFO)
redis = Redis(host='primary.xxx.yyyyyy.zzz1.cache.amazonaws.com', port=6379,
 decode_responses=True, ssl=True, username=example, password=EXAMPLE)

if redis.ping():
    logging.info("Connected to Redis")
```

프로그램을 실행하려면 다음 명령을 입력합니다.

```
python ConnectClusterModeDisabled.py
```

클러스터 모드가 활성화된 클러스터에 연결

다음 프로그램을 복사하여 ConnectClusterModeEnabled.py라는 파일에 붙여 넣습니다.

```
from rediscluster import RedisCluster
import logging

logging.basicConfig(level=logging.INFO)
redis = RedisCluster(startup_nodes=[{"host":
 "xxx.yyy.clustercfg.zzz1.cache.amazonaws.com","port": "6379"}],
 decode_responses=True,skip_full_coverage_check=True)

if redis.ping():
    logging.info("Connected to Redis")
```

프로그램을 실행하려면 다음 명령을 입력합니다.

```
python ConnectClusterModeEnabled.py
```

사용 예제:

다음 예제에서는 의 boto3SDK ElastiCache 를 사용하여 ElastiCache (Redis OSS)를 사용합니다.

주제

- 문자열 설정 및 가져오기
- 여러 항목이 있는 해시 설정 및 가져오기
- 게시/구독 채널의 게시(쓰기) 및 구독(읽기)
- 스트림의 쓰기 및 읽기

문자열 설정 및 가져오기

다음 프로그램을 복사하여 SetAndGetStrings.py라는 파일에 붙여 넣습니다.

```
import time
import logging
logging.basicConfig(level=logging.INFO,format='%(asctime)s: %(message)s')

keyName='mykey'
currTime=time.ctime(time.time())

# Set the key 'mykey' with the current date and time as value.
# The Key will expire and removed from cache in 60 seconds.
redis.set(keyName, currTime, ex=60)

# Sleep just for better illustration of TTL (expiration) value
time.sleep(5)

# Retrieve the key value and current TTL
keyValue=redis.get(keyName)
keyTTL=redis.ttl(keyName)

logging.info("Key {} was set at {} and has {} seconds until expired".format(keyName,
 keyValue, keyTTL))
```

프로그램을 실행하려면 다음 명령을 입력합니다.

```
python SetAndGetStrings.py
```

여러 항목이 있는 해시 설정 및 가져오기

다음 프로그램을 복사하여 SetAndGetHash.py라는 파일에 붙여 넣습니다.

```
import logging
import time

logging.basicConfig(level=logging.INFO,format='%(asctime)s: %(message)s')

keyName='mykey'
keyValues={'datetime': time.ctime(time.time()), 'epochtime': time.time()}

# Set the hash 'mykey' with the current date and time in human readable format
 (datetime field) and epoch number (epochtime field).
redis.hset(keyName, mapping=keyValues)

# Set the key to expire and removed from cache in 60 seconds.
redis.expire(keyName, 60)

# Sleep just for better illustration of TTL (expiration) value
time.sleep(5)

# Retrieves all the fields and current TTL
keyValues=redis.hgetall(keyName)
keyTTL=redis.ttl(keyName)

logging.info("Key {} was set at {} and has {} seconds until expired".format(keyName,
 keyValues, keyTTL))
```

프로그램을 실행하려면 다음 명령을 입력합니다.

```
python SetAndGetHash.py
```

게시/구독 채널의 게시(쓰기) 및 구독(읽기)

다음 프로그램을 복사하여 PubAndSub.py라는 파일에 붙여 넣습니다.

```
import logging
import time

def handlerFunction(message):
    """Prints message got from PubSub channel to the log output

    Return None
```

```
    :param message: message to log
    """
    logging.info(message)

logging.basicConfig(level=logging.INFO)
redis = Redis(host="redis202104053.tihewd.ng.0001.use1.cache.amazonaws.com", port=6379,
 decode_responses=True)


# Creates the subscriber connection on "mychannel"
subscriber = redis.pubsub()
subscriber.subscribe(**{'mychannel': handlerFunction})

# Creates a new thread to watch for messages while the main process continues with its
 routines
thread = subscriber.run_in_thread(sleep_time=0.01)

# Creates publisher connection on "mychannel"
redis.publish('mychannel', 'My message')

# Publishes several messages. Subscriber thread will read and print on log.
while True:
    redis.publish('mychannel',time.ctime(time.time()))
    time.sleep(1)
```

프로그램을 실행하려면 다음 명령을 입력합니다.

```
python PubAndSub.py
```

스트림의 쓰기 및 읽기

다음 프로그램을 복사하여 ReadWriteStream.py라는 파일에 붙여 넣습니다.

```
from redis import Redis
import redis.exceptions as exceptions
import logging
import time
import threading

logging.basicConfig(level=logging.INFO)

def writeMessage(streamName):
    """Starts a loop writting the current time and thread name to 'streamName'
```

```
    :param streamName: Stream (key) name to write messages.
    """
    fieldsDict={'writerId':threading.currentThread().getName(),'myvalue':None}
    while True:
        fieldsDict['myvalue'] = time.ctime(time.time())
        redis.xadd(streamName,fieldsDict)
        time.sleep(1)

def readMessage(groupName=None,streamName=None):
    """Starts a loop reading from 'streamName'
    Multiple threads will read from the same stream consumer group. Consumer group is
 used to coordinate data distribution.
    Once a thread acknowleges the message, it won't be provided again. If message
 wasn't acknowledged, it can be served to another thread.

    :param groupName: stream group were multiple threads will read.
    :param streamName: Stream (key) name where messages will be read.
    """

    readerID=threading.currentThread().getName()
    while True:
        try:
            # Check if the stream has any message
            if redis.xlen(streamName)>0:
                # Check if if the messages are new (not acknowledged) or not (already
 processed)
                streamData=redis.xreadgroup(groupName,readerID,
{streamName:'>'},count=1)
                if len(streamData) > 0:
                    msgId,message = streamData[0][1][0]
                    logging.info("{}: Got {} from ID
 {}".format(readerID,message,msgId))
                    #Do some processing here. If the message has been processed
 sucessfuly, acknowledge it and (optional) delete the message.
                    redis.xack(streamName,groupName,msgId)
                    logging.info("Stream message ID {} read and processed successfuly
 by {}".format(msgId,readerID))
                    redis.xdel(streamName,msgId)
            else:
                pass
        except:
            raise

        time.sleep(0.5)
```

```
# Creates the stream 'mystream' and consumer group 'myworkergroup' where multiple
 threads will write/read.
try:
    redis.xgroup_create('mystream','myworkergroup',mkstream=True)
except exceptions.ResponseError as e:
    logging.info("Consumer group already exists. Will continue despite the error:
 {}".format(e))
except:
    raise

# Starts 5 writer threads.
for writer_no in range(5):
    writerThread = threading.Thread(target=writeMessage, name='writer-'+str(writer_no),
 args=('mystream',),daemon=True)
    writerThread.start()

# Starts 10 reader threads
for reader_no in range(10):
    readerThread = threading.Thread(target=readMessage, name='reader-'+str(reader_no),
 args=('myworkergroup','mystream',),daemon=True)
    readerThread.daemon = True
    readerThread.start()

# Keep the code running for 30 seconds
time.sleep(30)
```

프로그램을 실행하려면 다음 명령을 입력합니다.

```
python ReadWriteStream.py
```

자습서: ElastiCache 에서 에 액세스하도록 Lambda 구성 VPC

이 자습서에서는 ElastiCache 서버리스 캐시를 생성하고, Lambda 함수를 생성한 다음, Lambda 함수를 테스트하고, 필요에 따라 정리하는 방법을 배울 수 있습니다.

주제

- 1단계: ElastiCache 서버리스 캐시 생성.
- 2단계: 에 대한 Lambda 함수 생성 ElastiCache
- 3단계: 를 사용하여 Lambda 함수 테스트 ElastiCache
- 4단계: 정리(선택 사항)

1단계: ElastiCache 서버리스 캐시 생성.

서버리스 캐시를 생성하려면 다음 단계를 따르세요.

1.1단계: 서버리스 캐시 생성

이 단계에서는 AWS Command Line Interface ()를 사용하여 계정의 us-east-1 리전VPC에 있는 기본 Amazon에서 서버리스 캐시를 생성합니다CLI. ElastiCache 콘솔 또는 를 사용하여 서버리스 캐시를 생성하는 방법에 대한 자세한 내용은 섹션을 API참조하세요Valkey 서버리스 캐시 생성.

```
aws elasticache create-serverless-cache \
  --serverless-cache-name cache-01  \
--description "ElastiCache IAM auth application" \
--engine valkey
```

상태 필드 값은 CREATING으로 설정됩니다. 캐시 생성을 완료하는 ElastiCache 데 1분 정도 걸릴 수 있습니다.

1.2단계: 서버리스 캐시 엔드포인트 복사

ElastiCache (Redis OSS)가 `describe-serverless-caches` 명령을 사용하여 캐시 생성을 완료했는지 확인합니다.

```
aws elasticache describe-serverless-caches \
--serverless-cache-name cache-01
```

출력에 표시된 엔드포인트 주소를 복사합니다. Lambda 함수의 배포 패키지를 생성할 때 이 주소가 필요합니다.

1.3단계: IAM 역할 생성

1. 계정이 새 역할을 수임할 수 있도록 아래에 표시된 대로 역할에 대한 IAM 신뢰 정책 문서를 생성합니다. 정책을 trust-policy.json 파일에 저장합니다.

   ```
   {
   "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. 아래와 같이 IAM 정책 문서를 생성합니다. 정책을 policy.json 파일에 저장합니다.

```
{
"Version": "2012-10-17",
  "Statement": [
    {
"Effect" : "Allow",
      "Action" : [
        "elasticache:Connect"
      ],
      "Resource" : [
        "arn:aws:elasticache:us-east-1:123456789012:serverlesscache:cache-01",
        "arn:aws:elasticache:us-east-1:123456789012:user:iam-user-01"
      ]
    }
  ]
}
```

3. IAM 역할을 생성합니다.

```
aws iam create-role \
--role-name "elasticache-iam-auth-app" \
--assume-role-policy-document file://trust-policy.json
```

4. IAM 정책을 생성합니다.

```
aws iam create-policy \
  --policy-name "elasticache-allow-all" \
  --policy-document file://policy.json
```

5. IAM 정책을 역할에 연결합니다.

```
aws iam attach-role-policy \
 --role-name "elasticache-iam-auth-app" \
```

```
 --policy-arn "arn:aws:iam::123456789012:policy/elasticache-allow-all"
```

1.4단계: 서버리스 캐시 생성

1. 새 기본 사용자를 생성합니다.

```
aws elasticache create-user \
  --user-name default \
--user-id default-user-disabled \
--engine redis \
--authentication-mode Type=no-password-required \
--access-string "off +get ~keys*"
```

2. 새 IAM사용 가능 사용자를 생성합니다.

```
aws elasticache create-user \
  --user-name iam-user-01 \
--user-id iam-user-01 \
--authentication-mode Type=iam \
--engine redis \
--access-string "on ~* +@all"
```

3. 사용자 그룹을 생성하고 사용자를 연결합니다.

```
aws elasticache create-user-group \
  --user-group-id iam-user-group-01 \
--engine redis \
--user-ids default-user-disabled iam-user-01

aws elasticache modify-serverless-cache \
  --serverless-cache-name cache-01  \
--user-group-id iam-user-group-01
```

2단계: 에 대한 Lambda 함수 생성 ElastiCache

ElastiCache 캐시에 액세스할 Lambda 함수를 생성하려면 다음 단계를 수행합니다.

2.1단계: Lambda 함수 생성

이 자습서에서는 Lambda 함수에 대한 예제 코드를 Python으로 제공합니다.

Python

다음 예제 Python 코드는 ElastiCache 캐시에 항목을 읽고 씁니다. 코드를 복사한 후 app.py 파일에 저장합니다. 코드의 elasticache_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 ElastiCacheIAMProvider(redis.CredentialProvider):
    def __init__(self, user, cache_name, is_serverless=False, region="us-east-1"):
        self.user = user
        self.cache_name = cache_name
        self.is_serverless = is_serverless
        self.region = region

        session = botocore.session.get_session()
        self.request_signer = RequestSigner(
            ServiceId("elasticache"),
            self.region,
            "elasticache",
            "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}
        if self.is_serverless:
            query_params["ResourceType"] = "ServerlessCache"
        url = urlunparse(
            ParseResult(
                scheme="https",
                netloc=self.cache_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
        # Elasticache 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
    cache_name = "cache-01" # replace with your cache name
    elasticache_endpoint = "cache-01-xxxxx.serverless.use1.cache.amazonaws.com" #
 replace with your cache endpoint
    creds_provider = ElastiCacheIAMProvider(user=username, cache_name=cache_name,
 is_serverless=True)
    redis_client = redis.Redis(host=elasticache_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 cache
    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 cache and print
    # the results
    if decoded_result == uuid_in:
        print(f"Success: Inserted {uuid_in}. Fetched {decoded_result} from Valkey.")
    else:
        raise Exception(f"Bad value retrieved. Expected {uuid_in}, got
 {decoded_result}")

    return "Fetched value from Valkey"
```

이 코드는 Python redis-py 라이브러리를 사용하여 항목을 캐시에 넣고 검색합니다. 이 코드는 캐시 도구를 사용하여 생성된 IAM 인증 토큰을 15분 동안 캐시합니다. redis-py 및 캐시 도구가 포함된 배포 패키지를 생성하려면 다음 단계를 수행합니다.

app.py 소스 코드 파일이 포함된 프로젝트 디렉터리에서 redis-py 및 cachetools 라이브러리를 설치할 폴더 패키지를 생성합니다.

```
mkdir package
```

pip를 사용하여 redis-py, cachetools를 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. Windows에서는 선호하는 zip 유틸리티를 사용하여 루트에 redis-py 및 cachetools 라이브러리가 있는 .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.py 추가합니다.

```
cd ..
zip my_deployment_package.zip app.py
```

2.2단계: IAM 역할 생성(실행 역할)

라는 AWS 관리형 정책을 `AWSLambdaVPCAccessExecutionRole` 역할에 연결합니다.

```
aws iam attach-role-policy \
 --role-name "elasticache-iam-auth-app" \
 --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaVPCAccessExecutionRole"
```

2.3단계: 배포 패키지 업로드(Lambda 함수 생성)

이 단계에서는 create-function AWS CLI 명령을 사용하여 Lambda 함수(AccessValkey)를 생성합니다.

배포 패키지 .zip 파일이 포함된 프로젝트 디렉터리에서 다음 Lambda CLI `create-function` 명령을 실행합니다.

역할 옵션의 경우 이전 단계에서 생성한 실행 역할ARN의 를 사용합니다. vpc-config에 기본 서브넷의 쉼표로 구분VPC된 목록과 기본 보안 그룹 IDVPC를 입력합니다. Amazon VPC 콘솔에서 이러한 값을 찾을 수 있습니다. 기본 VPC의 서브넷을 찾으려면 VPCs를 선택한 다음 AWS 계정의 기본 를 선택합니다VPC. 이 에 대한 보안 그룹을 찾으려면 보안으로 VPC이동하여 보안 그룹 을 선택합니다. us-east-1 리전이 선택되어 있어야 합니다.

```
aws lambda create-function \
--function-name AccessValkey  \
--region us-east-1 \
--zip-file fileb://my_deployment_package.zip \
--role arn:aws:iam::123456789012:role/elasticache-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: 를 사용하여 Lambda 함수 테스트 ElastiCache

이 단계에서는 호출 명령을 사용하여 Lambda 함수를 수동으로 호출합니다. Lambda 함수가 실행되면 가 생성UUID되어 Lambda 코드에 지정한 ElastiCache 캐시에 기록됩니다. 그런 다음 Lambda 함수는 캐시에서 해당 항목을 검색합니다.

1. 호출 명령을 사용하여 Lambda 함수(AccessValkey) AWS Lambda 를 호출합니다.

   ```
   aws lambda invoke \
   --function-name AccessValkey  \
   --region us-east-1 \
   output.txt
   ```

2. 다음과 같이 Lambda 함수가 성공적으로 실행되었는지 확인합니다.

   - output.txt 파일을 검토합니다.
   - CloudWatch 콘솔을 열고 함수(/aws/lambda/)의 로그 그룹을 선택하여 CloudWatch 로그에서 결과를 확인합니다AccessValkey. 로그 스트림의 출력은 다음과 유사해야 합니다.

```
Success: Inserted 826e70c5f4d2478c8c18027125a3e01e. Fetched
 826e70c5f4d2478c8c18027125a3e01e from Valkey.
```

- AWS Lambda 콘솔에서 결과를 검토합니다.

4단계: 정리(선택 사항)

정리하려면 다음 단계를 수행합니다.

4.1단계: Lambda 함수 삭제

```
aws lambda delete-function \
 --function-name AccessValkey
```

4.2단계: 서버리스 캐시 삭제

캐시를 삭제합니다.

```
aws elasticache delete-serverless-cache \
 --serverless-cache-name cache-01
```

사용자 및 사용자 그룹을 제거합니다.

```
aws elasticache delete-user \
 --user-id default-user-disabled

aws elasticache delete-user \
 --user-id iam-user-01

aws elasticache delete-user-group \
 --user-group-id iam-user-group-01
```

4.3단계: IAM 역할 및 정책 제거

```
aws iam detach-role-policy \
 --role-name "elasticache-iam-auth-app" \
 --policy-arn "arn:aws:iam::123456789012:policy/elasticache-allow-all"

aws iam detach-role-policy \
```

```
--role-name "elasticache-iam-auth-app" \
--policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaVPCAccessExecutionRole"

aws iam delete-role \
 --role-name "elasticache-iam-auth-app"

 aws iam delete-policy \
  --policy-arn "arn:aws:iam::123456789012:policy/elasticache-allow-all"
```

자체 ElastiCache 클러스터 설계 및 관리

ElastiCache 클러스터에 대한 세분화된 제어가 필요한 경우 자체 클러스터를 설계하도록 선택할 수 있습니다. ElastiCache 를 사용하면 클러스터의 AWS 가용 영역 전체에서 노드 유형, 노드 수 및 노드 배치를 선택하여 노드 기반 클러스터를 운영할 수 있습니다. ElastiCache 는 완전 관리형 서비스이므로 클러스터의 하드웨어 프로비저닝, 모니터링, 노드 교체 및 소프트웨어 패치를 자동으로 관리합니다.

설정 방법에 대한 자세한 설명은 설정 ElastiCache 섹션을 참조하세요. 노드나 클러스터의 관리, 업데이트 또는 삭제에 대한 세부 사항은 에서 노드 관리 ElastiCache 섹션을 참조하세요. 자체 ElastiCache 클러스터를 설계할 때 Amazon ElastiCache 배포의 주요 구성 요소에 대한 개요는 다음 주요 개념을 참조하세요.

주제

- ElastiCache 구성 요소 및 기능
- ElastiCache 용어
- 자습서: 자체 클러스터를 설계하는 방법
- 클러스터 삭제
- 기타 ElastiCache 자습서 및 동영상
- 에서 노드 관리 ElastiCache
- 에서 클러스터 관리 ElastiCache
- Valkey, Redis OSS및 Memcached 자체 설계 캐시 비교
- Valkey 또는 Redis의 온라인 마이그레이션 OSS
- 에 대한 리전 및 가용 영역 선택 ElastiCache

ElastiCache 구성 요소 및 기능

다음은 Amazon ElastiCache 배포의 주요 구성 요소에 대한 개요입니다.

주제

- ElastiCache 노드
- ElastiCache 샤드
- ElastiCache 클러스터
- ElastiCache 복제

- ElastiCache 엔드포인트
- ElastiCache 파라미터 그룹
- ElastiCache 보안
- ElastiCache 서브넷 그룹
- ElastiCache 백업
- ElastiCache 이벤트

ElastiCache 노드

노드는 ElastiCache 배포의 가장 작은 구성 요소입니다. 노드는 다른 노드와 독립적으로 존재하거나 다른 노드와 일부 관련되어 존재할 수 있습니다.

노드는 네트워크에 연결된 안전한 의 고정 크기 청크입니다RAM. 각 노드는 클러스터를 생성할 때 선택한 엔진 및 버전의 인스턴스를 실행합니다. 필요한 경우 클러스터의 노드를 다른 인스턴스 유형으로 스케일 업하거나 스케일 다운할 수 있습니다. 자세한 내용은 크기 조정 ElastiCache 단원을 참조하십시오.

클러스터 내 모든 노드는 인스턴스 유형이 동일하며, 동일한 캐시 엔진을 실행합니다. 각 캐시 노드에는 고유한 도메인 이름 서비스(DNS) 이름과 포트가 있습니다. 여러 유형의 캐시 노드가 지원되며 연결된 메모리 양이 각각 다릅니다. 지원되는 노드 인스턴스 유형의 목록은 지원되는 노드 유형 섹션을 참조하세요.

노드를 pay-as-you-go 기준으로 구매할 수 있으며, 노드 사용에 대한 비용만 지불하면 됩니다. 또는 대폭 인하된 시간당 요금으로 예약 노드를 구입할 수 있습니다. 사용률이 높은 경우, 예약 노드를 구입하면 비용을 절약할 수 있습니다. 클러스터가 항상 사용 중에 있고 사용량 폭증을 처리하기 위해 가끔 노드를 추가하는 경우를 생각해 봅시다. 이 경우, 대부분의 시간을 실행하기 위해 여러 개의 예약된 노드를 구입할 수 있습니다. 그런 다음 때때로 노드를 추가해야 하는 시간 동안 노드를 구매할 pay-as-you-go 수 있습니다. 예약 노드에 대한 자세한 내용은 예약 노드 섹션을 참조하세요.

노드에 대한 자세한 내용은 에서 노드 관리 ElastiCache 섹션을 참조하세요.

ElastiCache 샤드

Valkey 또는 Redis OSS 샤드(API 및 에서 노드 그룹이라고 함CLI)는 1~6개의 관련 노드 그룹입니다. OSS 클러스터 모드가 활성화된 Valkey 또는 Redis 클러스터에는 항상 샤드가 하나 이상 있습니다.

샤딩은 대규모 데이터베이스를 데이터 샤드라고 하는 더 작고 빠르며 쉽게 관리되는 부분으로 분리하는 데이터베이스 파티셔닝 방법입니다. 이렇게 하면 여러 별도의 섹션에 작업을 분산하여 데이터베이

스 효율성을 높일 수 있습니다. 샤드를 사용하면 향상된 성능, 확장성 및 비용 효율성을 비롯한 많은 이점을 얻을 수 있습니다.

클러스터 모드가 활성화된 Valkey 및 Redis OSS 클러스터에는 최대 500개의 샤드가 있을 수 있으며 데이터는 샤드에 분할됩니다. Valkey 또는 Redis OSS 엔진 버전이 5.0.6 이상인 경우 노드 또는 샤드 제한을 클러스터당 최대 500개로 늘릴 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. 일반적인 위험에는 서브넷 그룹의 서브넷 CIDR 범위가 너무 작거나 서브넷이 공유되어 다른 클러스터에서 많이 사용됩니다. 자세한 내용은 서브넷 그룹 생성 단원을 참조하십시오. 5.0.6 이하의 버전에서 한도는 클러스터당 250개입니다.

한도 증가를 요청하려면 AWS 서비스 한도를 참조하고 한도 유형을 인스턴스 유형별 클러스터당 노드로 선택하세요.

다중 노드 샤드는 읽기/쓰기 기본 노드 하나와 1~5개의 복제본 노드를 통해 복제를 구현합니다. 자세한 내용은 고가용성을 위한 복제 그룹 사용 단원을 참조하십시오.

샤드에 대한 자세한 내용은 에서 샤드 작업 ElastiCache 섹션을 참조하세요.

ElastiCache 클러스터

클러스터는 하나 이상의 노드 의 논리적 그룹입니다. 데이터는 Memcached 클러스터의 노드와 클러스터 모드가 활성화된 Valkey 또는 Redis OSS 클러스터의 샤드 간에 분할됩니다.

많은 ElastiCache 작업이 클러스터를 대상으로 합니다.

- 클러스터 생성
- 클러스터 수정
- 클러스터의 스냅샷 생성(모든 Redis 버전)
- 클러스터 삭제
- 클러스터의 요소 보기
- 비용 할당 태그를 클러스터에 추가 및 클러스터에서 삭제

자세한 내용은 다음 관련 항목을 참조하세요.

- 에서 클러스터 관리 ElastiCache 및 에서 노드 관리 ElastiCache

  클러스터, 노드 및 관련 작업에 대한 정보입니다.

- AWS 서비스 제한: Amazon ElastiCache

  최대 노드 또는 클러스터 수와 같은 ElastiCache 제한에 대한 정보입니다. 이러한 특정 제한을 초과하려면 Amazon ElastiCache 캐시 노드 요청 양식 을 사용하여 요청할 수 있습니다.
- 장애 완화

  클러스터 및 Valkey 또는 Redis OSS 복제 그룹의 내결함성을 개선하는 방법에 대한 정보입니다.

일반적인 클러스터 구성

다음은 일반적인 클러스터 구성입니다.

Valkey 또는 Redis OSS 클러스터

클러스터 모드가 비활성화된 Valkey 또는 Redis OSS 클러스터에는 항상 샤드가 하나만 포함됩니다(API 및 에서는 노드 그룹 CLI하나). Valkey 또는 Redis OSS 샤드에는 1~6개의 노드가 포함되어 있습니다. 한 샤드에 노드가 두 개 이상 있는 경우 샤드에서는 복제를 지원합니다. 이 경우 한 노드는 읽기/쓰기 기본 노드이고 다른 노드는 읽기 전용 복제 노드입니다.

내결함성을 개선하려면 Valkey 또는 Redis OSS 클러스터에 노드가 두 개 이상 있고 다중 AZ를 활성화하는 것이 좋습니다. 자세한 내용은 장애 완화 단원을 참조하십시오.

Valkey 또는 Redis OSS 클러스터에 대한 수요가 변경되면 확장하거나 축소할 수 있습니다. 이렇게 하려면 클러스터를 다른 노드 인스턴스 유형으로 이동합니다. 애플리케이션이 읽기 집약적인 경우 클러스터에 읽기 전용 복제본을 추가하는 것이 좋습니다. 이렇게 하면 보다 적절한 수의 노드로 읽기를 분산할 수 있습니다.

데이터 계층화를 사용할 수도 있습니다. 자주 액세스하는 데이터는 메모리에 저장되고 자주 액세스하지 않는 데이터는 디스크에 저장됩니다. 데이터 계층화를 사용하면 메모리 요구 사항이 줄어들 수 있다는 장점이 있습니다. 자세한 내용은 의 데이터 계층화 ElastiCache 단원을 참조하십시오.

ElastiCache 는 Valkey 또는 Redis OSS 클러스터의 노드 유형을 더 큰 노드 유형으로 동적으로 변경하는 것을 지원합니다. 스케일 업 또는 스케일 다운에 대한 정보는 Valkey 또는 Redis에 대한 단일 노드 클러스터 크기 조정OSS(클러스터 모드 비활성화됨) 또는 Valkey 또는 Redis용 복제본 노드 크기 조정 OSS(클러스터 모드 비활성화됨) 섹션을 참조하세요.

Memcached의 일반적인 클러스터 구성

Memcached는 각 클러스터에 1~60개의 노드가 있는 각 AWS 리전에 대해 고객당 최대 300개의 노드를 지원합니다. 사용자는 데이터를 Memcached 클러스터의 노드에 두루 분할합니다.

Memcached 엔진을 실행할 때 클러스터는 1~60개의 노드로 구성될 수 있습니다. 사용자는 데이터베이스를 노드에 두루 분할합니다. 애플리케이션은 각 노드의 엔드포인트를 읽고 씁니다. 자세한 내용은 Auto Discovery를 참조하세요.

내결함성을 개선하려면 클러스터 AWS 리전 내의 다양한 가용 영역(AZs)에서 Memcached 노드를 찾습니다. 이렇게 하면 한 AZ에서 발생한 오류가 전체 클러스터 및 애플리케이션에 미치는 영향을 최대한 줄일 수 있습니다. 자세한 내용은 장애 완화 단원을 참조하십시오.

Memcached 클러스터 변경 시 요구 사항에 따라 노드를 추가 또는 제거하여 확장 또는 축소할 수 있습니다. 이로써 데이터는 새 노드 수에 두루 재분할됩니다. 데이터를 분할할 때 일관적 해싱을 사용하는 것이 좋습니다. 일관적 해싱에 대한 자세한 내용은 효율적인 로드 밸런싱을 위해 ElastiCache 클라이언트 구성(Memcached) 섹션을 참조하세요.

ElastiCache 복제

Valkey 및 Redis 의 경우 OSS복제는 샤드(API 및 에서는 노드 그룹CLI이라고 함)에 있는 2~6개의 노드를 그룹화하여 구현됩니다. 이러한 노드 중 하나는 읽기/쓰기 기본 노드입니다. 다른 모든 노드는 읽기 전용 복제본 노드입니다. 복제는 ElastiCache Valkey 및 Redis 에서만 사용할 수 OSS있으며 ElastiCache (Memcached)에서는 사용할 수 없습니다.

각 복제본 노드는 기본 노드에서 데이터 사본을 유지합니다. 복제본 노드는 비동기식 복제 메커니즘을 사용하여 기본 노드와의 동기화를 유지합니다. 애플리케이션은 클러스터에 있는 모든 노드에서 읽을 수 있지만 기본 노드에만 쓸 수 있습니다. 읽기 전용 복제본은 여러 엔드포인트에 읽기를 분산하여 확장성을 향상합니다. 또한 읽기 전용 복제본은 여러 데이터 사본을 유지 관리하여 내결함성을 개선합니다. 다중 가용 영역에서 읽기 전용 복제본을 찾으면 내결함성이 더욱 개선됩니다. 내결함성에 대한 자세한 내용은 장애 완화 섹션을 참조하세요.

Valkey 또는 Redis OSS 클러스터는 샤드 하나를 지원합니다(API 및 에서는 노드 그룹CLI이라고 함).

API 및 CLI 관점에서의 복제는 이전 버전과의 호환성을 유지하기 위해 다른 용어를 사용하지만 결과는 동일합니다. 다음 표에는 복제 구현을 위한 API 및 CLI 용어가 나와 있습니다.

복제 비교: Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 및 Valkey 또는 RedisOSS(클러스터 모드 활성화됨)--> 클러스터 모드가 활성화된 Valkey 또는 Redis OSS 클러스터와 클러스터 모드가 비활성화된 Valkey 또는 Redis OSS 클러스터 비교

다음 표에서는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨)와 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹의 기능을 비교할 수 있습니다.

	OSS 클러스터 모드가 비활성화된 Valkey 또는 Redis 클러스터	OSS 클러스터 모드가 활성화된 Valkey 또는 Redis 클러스터
샤드(노드 그룹)	1	1~500
각 샤드(노드 그룹)의 복제본 수	0~5	0~5
데이터 파티셔닝	아니요	예
복제본 추가/삭제	예	예
노드 그룹 추가/삭제	아니요	예
확장 지원	예	예
엔진 업그레이드 지원	예	예
복제본을 기본 노드로 승격	예	자동
다중 AZ	선택 사항	필수
백업/복구	예	예
참고:		
복제본이 없는 기본 노드에서 장애가 발생할 경우 기본 노드에 저장된 데이터가 모두 손실됩니다.		
백업 및 복원을 사용하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨)로 마이그레이션할 수 있습니다.		
백업 및 복원을 사용하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 크기를 조정할 수 있습니다.		

모든 샤드(API 및 CLI, 노드 그룹)와 노드는 동일한 AWS 리전에 있어야 합니다. 그러나 해당 AWS 리전 내의 여러 가용 영역에 개별 노드를 프로비저닝할 수 있습니다.

읽기 전용 복제본은 데이터가 둘 이상의 노드(기본 및 하나 이상의 읽기 전용 복제본)에 복제되므로 잠재적인 데이터 손실이 방지됩니다. 안정성과 복구 속도를 높이려면 여러 가용 영역에 읽기 전용 복제본을 하나 이상 생성하는 것이 좋습니다.

글로벌 데이터 스토어를 활용할 수도 있습니다. Global Datastore for Redis OSS 기능을 사용하면 AWS 리전 간에 완전 관리형, 빠르고 안정적이며 안전한 복제를 수행할 수 있습니다. 이 기능을 사용하면 에 대한 리전 간 읽기 전용 복제본 클러스터를 생성 ElastiCache 하여 AWS 리전 간 지연 시간이 짧은 읽기 및 재해 복구를 활성화할 수 있습니다. 자세한 내용은 글로벌 데이터 스토어를 사용하여 AWS 리전 간 복제를 참조하세요.

복제: 제한 및 제외

- 노드 유형 T1에서는 다중 AZ가 지원되지 않습니다.

ElastiCache 엔드포인트

엔드포인트는 애플리케이션이 ElastiCache 노드 또는 클러스터에 연결하는 데 사용하는 고유한 주소입니다.

클러스터 모드OSS가 비활성화된 Valkey 또는 Redis의 단일 노드 엔드포인트

단일 노드 Valkey 또는 Redis OSS 클러스터의 엔드포인트는 읽기 및 쓰기 모두에 대해 클러스터에 연결하는 데 사용됩니다.

클러스터 모드OSS가 비활성화된 Valkey 또는 Redis의 다중 노드 엔드포인트

OSS 클러스터 모드가 비활성화된 다중 노드 Valkey 또는 Redis 클러스터에는 두 가지 유형의 엔드포인트가 있습니다. 기본 엔드포인트는 기본 역할의 특정 노드가 변경된 경우에도 항상 클러스터의 기본 노드에 연결됩니다. 클러스터에 대한 모든 쓰기를 위해 기본 엔드포인트를 사용합니다.

리더 엔드포인트를 사용하여 모든 읽기 전용 복제본 사이에 수신 연결을 고르게 분할합니다. 읽기 작업에 개별 노드 엔드포인트를 사용합니다(API/CLI에서는 이러한 엔드포인트를 읽기 엔드포인트라고 함).

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 엔드포인트

OSS 클러스터 모드가 활성화된 Valkey 또는 Redis 클러스터에는 단일 구성 엔드포인트가 있습니다. 구성 엔드포인트에 연결되면 애플리케이션이 클러스터의 각 샤드에 대한 기본 및 읽기 엔드포인트를 찾을 수 있습니다.

자세한 내용은 에서 연결 엔드포인트 찾기 ElastiCache 단원을 참조하십시오.

ElastiCache (Memcached) 엔드포인트

Memcached 클러스터의 각 노드에는 고유한 엔드포인트가 있습니다. 클러스터에도 구성 엔드포인트라는 엔드포인트가 있습니다. Auto Discovery를 사용하기로 설정하고 구성 엔드포인트를 연결한 경우에는 클러스터에서 노드를 추가하거나 제거한 후에도 애플리케이션이 자동으로 각 노드 엔드포인트를 파악합니다. 자세한 내용은 Auto Discovery를 참조하세요.

자세한 내용은 엔드포인트 단원을 참조하십시오.

ElastiCache 파라미터 그룹

캐시 파라미터 그룹은 지원되는 엔진 소프트웨어에 대한 런타임 설정을 관리하는 간단한 방법입니다. 메모리 사용량, 제거 정책, 항목 크기 등을 제어하는 데 여러 가지 파라미터가 사용됩니다. ElastiCache 파라미터 그룹은 클러스터에 적용할 수 있는 엔진별 파라미터의 명명된 모음입니다. 이를 통해 해당 클러스터에 있는 모든 노드가 정확히 동일한 방법으로 구성되게 할 수 있습니다.

지원되는 파라미터 목록, 기본값 및 수정할 수 있는 파라미터 목록은 섹션을 참조하세요.DescribeEngineDefaultParameters (CLI: describe-engine-default-parameters).

ElastiCache 파라미터 그룹에 대한 자세한 내용은 섹션을 참조하세요파라미터 그룹을 사용하여 엔진 ElastiCache 파라미터 구성.

ElastiCache 보안

향상된 보안을 위해 ElastiCache 노드 액세스는 사용자가 허용하는 Amazon EC2 인스턴스에서 실행되는 애플리케이션으로 제한됩니다. 보안 그룹을 사용하여 클러스터에 액세스할 수 있는 Amazon EC2 인스턴스를 제어할 수 있습니다.

기본적으로 모든 새 ElastiCache 클러스터는 Amazon Virtual Private Cloud(AmazonVPC) 환경에서 시작됩니다. 서브넷 그룹을 사용하여 특정 서브넷에서 실행되는 Amazon EC2 인스턴스에서 클러스터 액세스 권한을 부여할 수 있습니다.

노드 액세스를 제한하는 것 외에도 는 지정된 버전의 를 실행하는 노드에 대한 암호화를 ElastiCache 지원TLS하며 제자리에 있습니다 ElastiCache. 자세한 내용은 다음 자료를 참조하세요.

- Amazon의 데이터 보안 ElastiCache
- Valkey 및 Redis OSS AUTH 명령을 사용한 인증

ElastiCache 서브넷 그룹

서브넷 그룹은 Amazon VPC 환경에서 실행되는 클러스터에 대해 지정할 수 있는 서브넷(일반적으로 프라이빗) 모음입니다.

Amazon 에서 클러스터를 생성하는 VPC경우 캐시 서브넷 그룹을 지정해야 합니다. ElastiCache 는 해당 캐시 서브넷 그룹을 사용하여 해당 서브넷 내에서 캐시 노드와 연결할 서브넷 및 IP 주소를 선택합니다.

Amazon VPC 환경에서의 캐시 서브넷 그룹 사용에 대한 자세한 내용은 다음을 참조하세요.

- Amazon VPCs 및 ElastiCache 보안
- 단계 3. 클러스터에 대한 액세스 권한 부여
- 서브넷 및 서브넷 그룹

ElastiCache 백업

백업은 point-in-time Valkey 또는 Redis OSS 클러스터 또는 서버리스 캐시 또는 Memcached 서버리스 캐시의 사본입니다. 백업을 사용하여 기존 클러스터를 복원하거나 새 클러스터를 시드할 수 있습니다. 백업은 클러스터의 모든 데이터와 일부 메타데이터로 구성됩니다.

클러스터에서 OSS 실행되는 Valkey 또는 Redis의 버전에 따라 백업 프로세스에 성공하려면 예약 메모리의 양이 달라집니다. 자세한 내용은 다음 자료를 참조하세요.

- 스냅샷 및 복원
- 동기화 및 백업 구현 방법
- 자체 설계된 클러스터 백업이 성능에 미치는 영향
- Valkey 또는 Redis OSS 스냅샷을 생성하기에 충분한 메모리 확보

ElastiCache 이벤트

캐시 클러스터에서 중요한 이벤트가 발생하면 는 특정 Amazon SNS 주제에 알림을 ElastiCache 보냅니다. 이러한 이벤트로는 노드 추가 실패 또는 성공, 보안 그룹 수정 등을 들 수 있습니다. 주요 이벤트를 모니터링하면 클러스터의 현재 상태를 파악할 수 있으며, 많은 경우에 교정 작업을 수행할 수도 있습니다.

ElastiCache 이벤트에 대한 자세한 내용은 섹션을 참조하세요Amazon ElastiCache 이벤트 SNS 모니터링.

ElastiCache 용어

2016년 10월 Amazon은 Redis OSS 3.2에 대한 지원을 ElastiCache 시작했습니다. 이때 최대 500개의 샤드(및 에서 노드 그룹이라고 함)에 걸쳐 데이터를 분할하는 ElastiCache API 지원이 추가되었습니다 AWS CLI. 이전 버전과의 호환성을 유지하기 위해 새로운 Redis OSS 기능을 포함하도록 API 버전 2015-02-02 작업을 확장했습니다.

동시에 이 새로운 기능에 사용되고 업계 전반에서 일반적인 ElastiCache 콘솔에서 용어를 사용하기 시작했습니다. 이러한 변경 사항은 일부 시점에서 및 에 사용되는 용어가 콘솔에 사용되는 용어API와 다를 CLI 수 있음을 의미합니다. 다음 목록은 API 및 콘솔 간에 다를 수 있는 용어CLI를 식별합니다.

캐시 클러스터 또는 노드와 노드 비교

복제본 노드가 one-to-one 없는 경우 노드와 캐시 클러스터 간에 관계가 있습니다. 따라서 ElastiCache 콘솔은 종종 용어를 상호 교환적으로 사용했습니다. 콘솔에서는 이제 노드 처리량이라는 용어를 사용합니다. 한 가지 예외는 [Create Cluster] 버튼입니다. 이 버튼은 복제본 노드를 포함 또는 포함하지 않고 클러스터를 생성하는 프로세스를 시작합니다.

및 는 ElastiCache API 이전과 동일한 용어를 AWS CLI 계속 사용합니다.

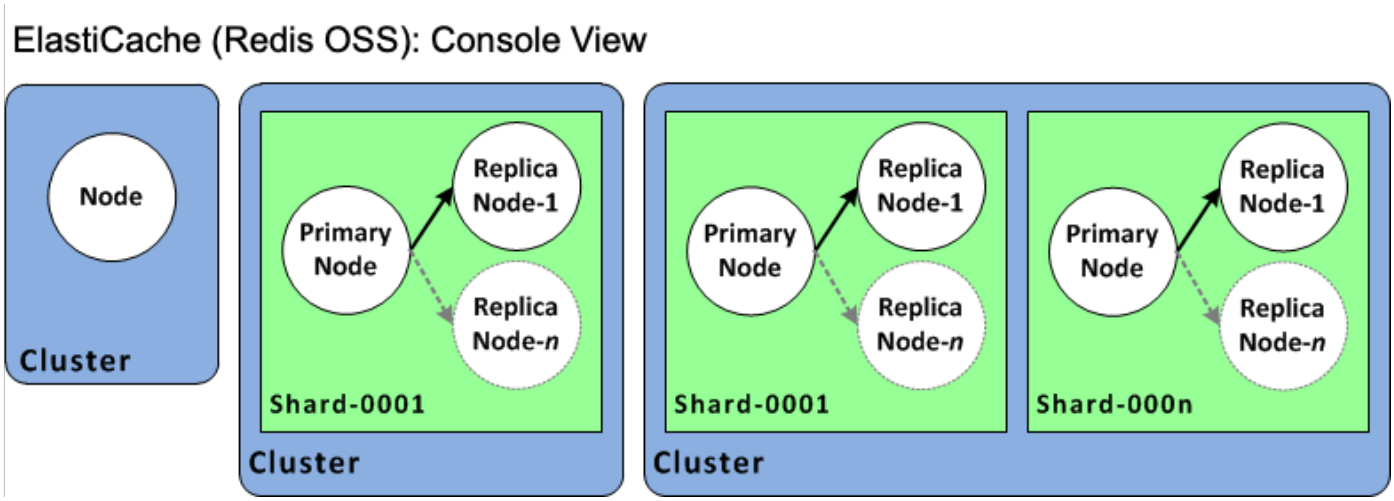
클러스터 대 Valkey 또는 Redis OSS 복제 그룹

이제 콘솔은 모든 ElastiCache (Redis OSS) 클러스터에 대해 용어 클러스터를 사용합니다. 콘솔에서는 다음과 같은 모든 환경에서 클러스터 용어를 사용합니다.

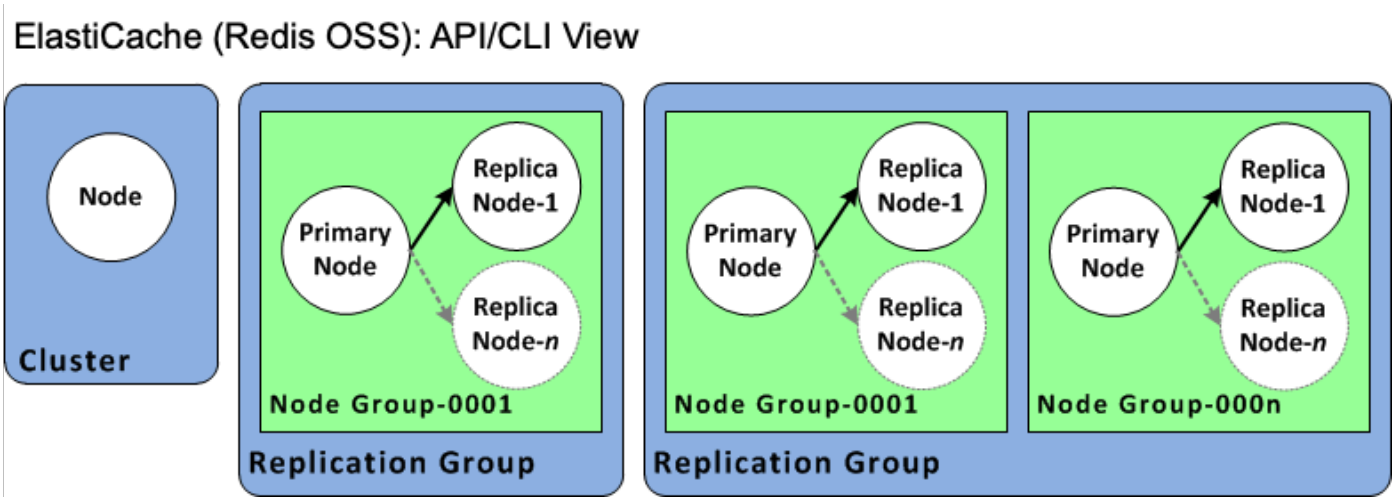
- 클러스터가 단일 노드 Valkey 또는 Redis OSS 클러스터인 경우.
- 클러스터가 단일 샤드 내에서 복제를 지원하는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터인 경우(API 및 에서는 노드 그룹 CLI이라고 함).
- 클러스터가 1~90개의 샤드 또는 한도 증가 요청으로 최대 500개의 샤드 내에서 복제를 지원하는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터인 경우. 한도 증가를 요청하려면 AWS 서비스 한도를 참조하고 한도 유형을 인스턴스 유형별 클러스터당 노드로 선택하세요.

Valkey 또는 Redis OSS 복제 그룹에 대한 자세한 내용은 섹션을 참조하세요고가용성을 위한 복제 그룹 사용.

다음 다이어그램은 콘솔의 관점에서 ElastiCache (Redis OSS) 클러스터의 다양한 토폴로지를 보여줍니다.



ElastiCache API 및 AWS CLI 작업은 여전히 단일 노드 ElastiCache (Redis OSS) 클러스터를 다중 노드 Valkey 또는 Redis OSS 복제 그룹과 구분합니다. 다음 다이어그램은 및 AWS CLI 관점에서 다양한 ElastiCache (Redis OSS) 토폴로지를 ElastiCache API 보여줍니다.



Valkey 또는 Redis OSS Replication 그룹과 글로벌 데이터 스토어 비교

글로벌 데이터 스토어는 리전 간에 서로 복제하는 하나 이상의 클러스터 모음인 반면, Valkey 또는 Redis OSS 복제 그룹은 여러 샤드가 있는 클러스터 모드 지원 클러스터 전체에서 데이터를 복제합니다. 글로벌 데이터 스토어는 다음과 같이 구성됩니다.

- 기본(활성) 클러스터 - 기본 클러스터는 글로벌 데이터 스토어 내의 모든 클러스터에 복제되는 쓰기를 허용합니다. 기본 클러스터는 읽기 요청도 허용합니다.
- 보조(수동) 클러스터 - 보조 클러스터는 읽기 요청만 허용하고 기본 클러스터에서 데이터 업데이트를 복제합니다. 보조 클러스터는 기본 클러스터와 다른 AWS 리전에 있어야 합니다.

글로벌 데이터 스토어에 대한 자세한 내용은 글로벌 데이터 스토어를 사용하여 AWS 리전 간 복제 섹션을 참조하세요.

자습서: 자체 클러스터를 설계하는 방법

다음은 Valkey 및 Redis용 자체 클러스터를 설계하는 방법입니다OSS.

주제

- 자체 ElastiCache (Valkey) 클러스터 설계
- 자체 ElastiCache (Redis OSS) 클러스터 설계

자체 ElastiCache (Valkey) 클러스터 설계

다음은 ElastiCache (Valkey) 클러스터 설계를 시작하기 위해 수행해야 하는 일회성 작업입니다.

1단계: 서브넷 그룹 생성

ElastiCache (Valkey) 클러스터를 생성하기 전에 먼저 서브넷 그룹을 생성합니다. 캐시 서브넷 그룹은 에서 캐시 클러스터에 대해 지정할 수 있는 서브넷 모음입니다VPC. 에서 캐시 클러스터를 시작할 때 캐시 서브넷 그룹을 선택해야 VPC합니다. 그런 다음 해당 캐시 서브넷 그룹을 ElastiCache 사용하여 해당 서브넷 내의 IP 주소를 클러스터의 각 캐시 노드에 할당합니다.

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.

설정에 대한 자세한 내용은 섹션을 ElastiCache 참조하세요설정 ElastiCache.

다음 절차는 `mysubnetgroup`(콘솔)이라는 서브넷 그룹과 AWS CLI를 생성하는 방법을 보여 줍니다.

서브넷 그룹 생성(콘솔)

다음 절차는 서브넷 그룹을 생성하는 방법을 보여줍니다(콘솔).

서브넷 그룹을 생성하려면(콘솔)

1. AWS 관리 콘솔에 로그인하고 에서 ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 탐색 목록에서 [Subnet Groups]를 선택합니다.
3. [Create Subnet Group]을 선택합니다.

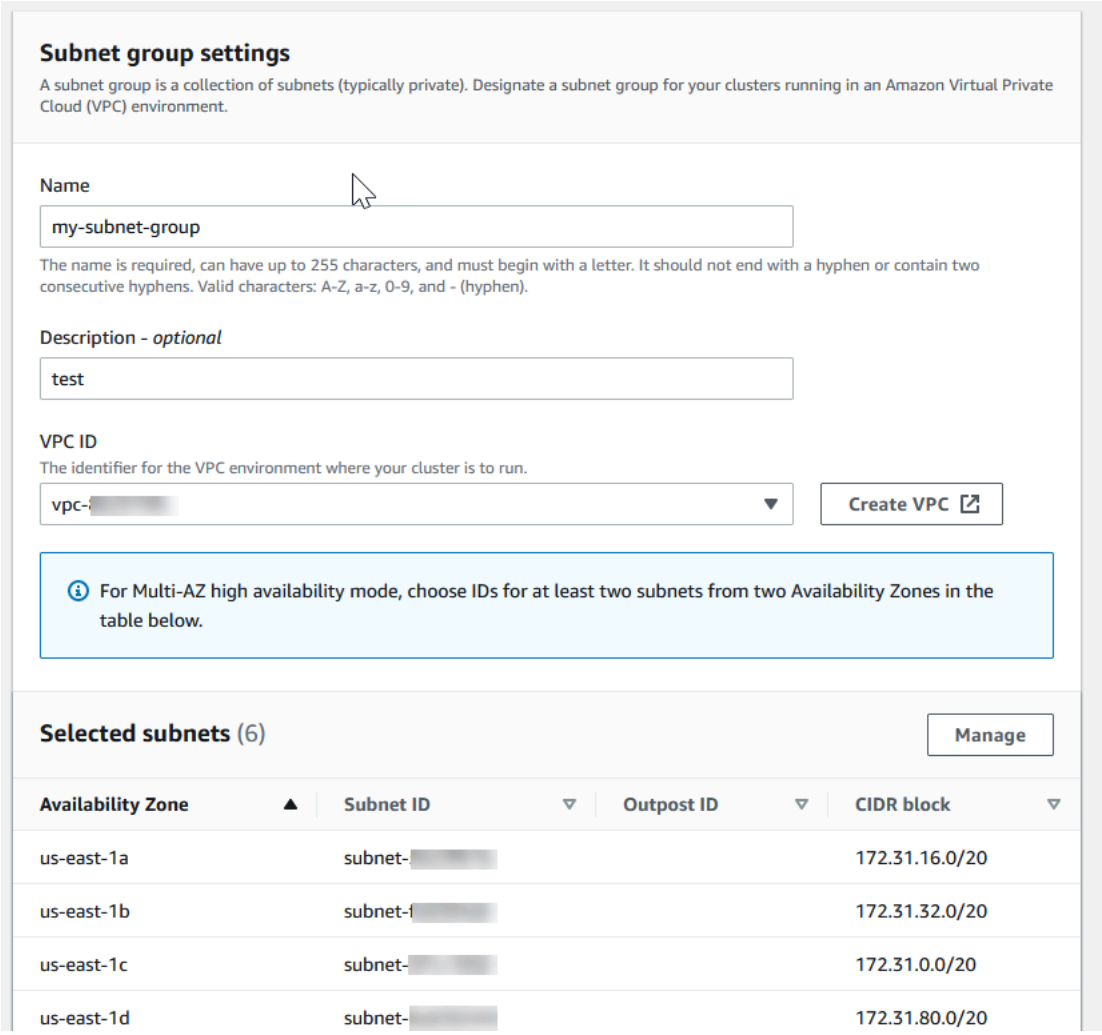
4. [Create Subnet Group] 마법사에서 다음을 수행합니다. 모든 설정이 원하는 대로 설정되었으면 [Yes, Create]를 선택합니다.

   a. [Name] 상자에 서브넷 그룹의 이름을 입력합니다.

   b. [Description] 상자에 서브넷 그룹에 대한 설명을 입력합니다.

   c. VPC ID 상자에서 VPC 생성한 Amazon을 선택합니다.

   d. 가용 영역 및 서브넷 ID 목록에서 프라이빗 서브넷의 가용 영역 또는 에서 로컬 영역 사용 ElastiCache 및 ID를 선택한 다음 추가를 선택합니다.



5. 나타나는 확인 메시지에서 [Close]를 선택합니다.

새 서브넷 그룹이 ElastiCache 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-cache-subnet-group` 명령을 사용하여 서브넷 그룹을 생성합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache create-cache-subnet-group \
    --cache-subnet-group-name mysubnetgroup \
    --cache-subnet-group-description "Testing" \
    --subnet-ids subnet-53df9c3a
```

Windows의 경우:

```
aws elasticache create-cache-subnet-group ^
    --cache-subnet-group-name mysubnetgroup ^
    --cache-subnet-group-description "Testing" ^
    --subnet-ids subnet-53df9c3a
```

이 명령은 다음과 유사한 출력을 생성합니다.

```
{
    "CacheSubnetGroup": {
        "VpcId": "vpc-37c3cd17",
        "CacheSubnetGroupDescription": "Testing",
        "Subnets": [
            {
                "SubnetIdentifier": "subnet-53df9c3a",
                "SubnetAvailabilityZone": {
                    "Name": "us-west-2a"
                }
            }
        ],
        "CacheSubnetGroupName": "mysubnetgroup"
    }
}
```

자세한 내용은 AWS CLI 주제를 참조하세요.create-cache-subnet-group.

2단계: 클러스터 생성

프로덕션 용도로 사용할 클러스터를 생성하기 전에 비즈니스 요구 사항에 맞게 클러스터를 구성할 방법을 고려해야 합니다. 이러한 문제에 대해서는 에서 클러스터 준비 ElastiCache 섹션에서 다룹니다.

이 시작하기 연습의 목적에 따라, 클러스터 모드가 비활성화된 클러스터를 생성하고 적용되는 모든 위치에서 기본 구성 값을 그대로 사용합니다.

생성하려는 클러스터는 활성화되고 샌드박스에서 실행되지 않습니다. 인스턴스를 삭제할 때까지 인스턴스에 대한 표준 ElastiCache 사용 요금이 발생합니다. 여기에 설명된 연습을 한 번에 끝내고 연습을 마칠 때 클러스터를 삭제하면 총 청구 비용이 가장 적게 듭니다(일반적으로 1달러 미만). ElastiCache 사용률에 대한 자세한 내용은 Amazon 섹션을 ElastiCache 참조하세요.

클러스터는 Amazon VPC 서비스를 기반으로 가상 프라이빗 클라우드(VPC)에서 시작됩니다.

Valkey(클러스터 모드 비활성화됨) 클러스터 생성(콘솔)

ElastiCache 콘솔을 사용하여 Valkey(클러스터 모드 비활성화됨) 클러스터를 생성하려면

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 오른쪽 상단 모서리의 목록에서 이 클러스터를 시작할 AWS 리전을 선택합니다.
3. 탐색 창에서 시작하기(Get started)를 선택하세요.
4. 생성을 VPC 선택하고 Virtual Private Cloud 생성(VPC)에 설명된 단계를 따릅니다.
5. ElastiCache 대시보드 페이지에서 Valkey 캐시 또는 Redis OSS 캐시를 선택한 다음 Valkey 캐시 생성 또는 Redis OSS 캐시를 선택합니다.
6. 클러스터 설정(Cluster settings)에서 다음을 수행합니다.
   a. Configure and create a new cluster(새 클러스터 구성 및 생성)를 선택합니다.
   b. Cluster mode(클러스터 모드)에서 Enabled(사용 설정됨)를 선택합니다.
   c. Cluster info(클러스터 정보)에 Name(이름) 값을 입력합니다.
   d. (선택 사항) 설명(Description) 값을 입력합니다.
7. Location(위치)에서 다음을 수행합니다.

   AWS Cloud

   1. AWS Cloud의 경우 Multi-AZ(다중 AZ) 및 Auto-failover(자동 장애 조치)의 기본 설정을 수락하는 것이 좋습니다. 자세한 내용은 다중 AZ 를 사용한 ElastiCache (Redis OSS)의 가동 중지 시간 최소화를 참조하세요.
   2. 클러스터 설정(Cluster settings)에서 다음을 수행합니다.
      a. 엔진 버전(Engine version)의 경우 사용 가능한 버전을 선택합니다.

b. 포트(Port)의 경우 기본 포트인 6379를 사용합니다. 다른 포트를 사용해야 하는 경우 포트 번호를 입력합니다.

c. 파라미터 그룹에서 파라미터 그룹을 선택하거나 새 파라미터 그룹을 만듭니다. 파라미터 그룹은 클러스터의 런타임 파라미터를 제어합니다. 파라미터 그룹에 대한 자세한 정보는 Valkey 및 Redis OSS 파라미터 및 ElastiCache 파라미터 그룹 생성 섹션을 참조하세요.

> **Note**
>
> 파라미터 그룹을 선택하여 엔진 구성 값을 설정하면 해당 파라미터 그룹이 글로벌 데이터 스토어의 모든 클러스터에 적용됩니다. 파라미터 그룹 페이지에서 yes/no 글로벌 속성은 파라미터 그룹이 글로벌 데이터 스토어의 일부인지 여부를 나타냅니다.

d. 노드 유형에서 아래쪽 화살표
(▼)
를 선택합니다. 노드 유형 변경 대화 상자에서 원하는 노드 유형의 인스턴스 패밀리 값을 선택합니다. 그런 다음 이 클러스터에 사용할 노드 유형을 선택한 다음 저장을 선택합니다.

자세한 정보는 노드 크기 선택 섹션을 참조하세요.

r6gd 노드 유형을 선택하는 경우 데이터 계층화가 자동으로 사용 설정됩니다. 자세한 내용은 의 데이터 계층화 ElastiCache 단원을 참조하십시오.

e. Number of replicas(복제본 개수)의 경우 원하는 읽기 전용 복제본 개수를 선택합니다. 다중 AZ를 사용 설정한 경우 숫자는 1~5 사이가 되어야 합니다.

3. 연결 아래

a. 네트워크 유형에서 이 클러스터가 지원할 IP 버전을 선택합니다.

b. 서브넷 그룹 의 경우 이 클러스터에 적용할 서브넷을 선택합니다. 는 해당 서브넷 그룹을 ElastiCache 사용하여 해당 서브넷 내의 서브넷 및 IP 주소를 선택하여 노드와 연결합니다. ElastiCache 클러스터에는 듀얼 스택 모드에서 작동하려면 IPv4 및 IPv6 주소가 할당된 듀얼 스택 서브넷이 필요하며, IPv6전용 서브넷은 로IPv6만 작동합니다.

새 서브넷 그룹을 생성할 때 해당 서브넷 그룹이 속한 VPC ID를 입력합니다.

자세한 내용은 다음을 참조하세요.

- 에서 네트워크 유형 선택 ElastiCache.
- 에서 서브넷을 생성합니다VPC.

에서 로컬 영역 사용 ElastiCache를 사용하는 경우 로컬 영역에 있는 서브넷을 선택하거나 생성해야 합니다.

자세한 내용은 서브넷 및 서브넷 그룹 단원을 참조하십시오.

4. 가용 영역 배치(Availability zone placements)의 경우 다음 두 가지 옵션이 있습니다.

   - 기본 설정 없음 - 가용 영역을 ElastiCache 선택합니다.
   - 가용 영역 지정 – 각 클러스터의 가용 영역을 지정합니다.

     가용 영역을 지정하도록 선택한 경우 샤드에 있는 각 클러스터에 대해 목록에서 가용 영역을 선택합니다.

   자세한 내용은 에 대한 리전 및 가용 영역 선택 ElastiCache 단원을 참조하십시오.

5. 다음(Next)을 선택합니다.
6. 고급 Valkey 또는 Redis OSS 설정에서

   - 보안(Security)의 경우

     i. 데이터를 암호화하려면 다음과 같은 옵션이 있습니다.

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

       > **Note**
       >
       > Customer Managed AWS KMS 키를 선택하고 키를 선택하여 다른 암호화 키를 제공할 수 있습니다. 자세한 내용은 에서 AWS 고객 관리형 키 사용을 KMS참조하세요.

     - 전송 중 데이터 암호화 – 전송 데이터 암호화를 활성화합니다. 자세한 정보는 전송 중 데이터 암호화를 참조하세요. Valkey 및 Redis OSS 엔진 버전 6.0 이상의 경우 전송 중 암호화를 활성화하면 다음 액세스 제어 옵션 중 하나를 지정하라는 메시지가 표시됩니다.

- 액세스 제어 안 함 – 기본 설정입니다. 이 옵션은 클러스터에 대한 사용자 액세스를 제한하지 않는다는 의미입니다.
- 사용자 그룹 액세스 제어 목록 - 클러스터에 액세스할 수 있는 사용자 집합이 정의된 사용자 그룹을 선택합니다. 자세한 내용은 콘솔 및 를 사용하여 사용자 그룹 관리 CLI 단원을 참조하십시오.
- AUTH 기본 사용자 - Redis OSS 서버의 인증 메커니즘입니다. 자세한 내용은 AUTH를 참조하세요.

- AUTH – Redis OSS 서버의 인증 메커니즘입니다. 자세한 내용은 AUTH를 참조하세요.

> **Note**
>
> OSS 버전 3.2.10을 제외한 Valkey 및 3.2.6 이상 Redis 버전의 경우 Redis가 유일한 옵션OSSAUTH입니다.

ii. 보안 그룹에서 이 클러스터에 사용할 보안 그룹을 선택합니다. 보안 그룹은 클러스터에 대한 네트워크 액세스를 제어하는 방화벽 역할을 합니다. 에 기본 보안 그룹을 사용하거나 새 보안 그룹을 VPC 생성할 수 있습니다.

보안 그룹에 대한 자세한 내용은 Amazon 사용 설명서의 에 대한 보안 그룹을 VPC 참조하세요. VPC

7. 정기적인 자동 백업을 예약할 경우 Enable automatic backups(자동 백업 활성화)를 선택한 후 자동으로 삭제되기 전에 각 자동 백업을 보존할 기간(일)을 입력합니다. 정기적인 자동 백업을 예약하지 않으려면 [Enable automatic backups] 확인란의 선택을 취소합니다. 어떤 경우든 수동 백업을 항상 생성할 수 있습니다.

   Redis OSS 백업 및 복원에 대한 자세한 내용은 섹션을 참조하세요스냅샷 및 복원.

8. (선택 사항) 유지 관리 기간을 지정합니다. 유지 관리 기간은 클러스터에 대한 시스템 유지 관리 일정을 예약할 때마다 ElastiCache 일반적으로 1시간의 시간입니다. ElastiCache 에서 유지 관리 기간의 날짜 및 시간을 선택하거나(기본 설정 없음) 직접 날짜, 시간 및 기간을 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 ElastiCache 클러스터 유지 관리 단원을 참조하십시오.

9. (선택 사항) 로그의 경우:

- 로그 형식에서 텍스트 또는 를 선택합니다JSON.
- 대상 유형에서 CloudWatch 로그 또는 Kinesis Firehose를 선택합니다.
- 로그 대상 에서 새로 만들기를 선택하고 CloudWatch 로그 로그 그룹 이름 또는 Firehose 스트림 이름을 입력하거나 기존 선택을 선택한 다음 CloudWatch 로그 로그 로그 그룹 이름 또는 Firehose 스트림 이름을 선택합니다.

10. 태그 의 경우 클러스터 및 기타 ElastiCache 리소스를 관리하는 데 도움이 되도록 태그 형식으로 각 리소스에 자체 메타데이터를 할당할 수 있습니다. 자세한 정보는 ElastiCache 리소스 태그 지정 섹션을 참조하세요.
11. Next(다음)를 선택합니다.
12. 입력 및 선택한 내용을 모두 검토한 다음 필요한 내용을 수정합니다. 준비가 되었으면 생성(Create)e을 선택합니다.

On premises

1. On premises(온프레미스)의 경우 Auto-failover(자동 장애 조치)를 사용 설정하는 것이 좋습니다. 자세한 내용은 다중 AZ를 사용한 ElastiCache (Redis OSS)의 가동 중지 시간 최소화를 참조하세요.
2. 클러스터 생성을 완료하려면 Outposts 사용의 단계를 수행합니다.

클러스터의 상태를 사용할 수 있게 되면 즉시 Amazon에 EC2 액세스 권한을 부여하고 연결한 다음 사용을 시작할 수 있습니다. 자세한 내용은 단계 3. 클러스터에 대한 액세스 권한 부여 및 4단계. 클러스터의 노드에 연결 단원을 참조하세요.

> **Important**
>
> 클러스터를 사용할 수 있으면 클러스터를 활동적으로사용하지 않더라도 클러스터가 사용 설정되어 있는 매 시간 또는 60분 미만 단위로 비용이 청구됩니다. 이 클러스터의 요금 발생을 중지하려면 클러스터를 삭제해야 합니다. 에서 클러스터 삭제 ElastiCache을 참조하세요.

Valkey(클러스터 모드 비활성화됨) 클러스터 생성(AWS CLI)

Example

다음 CLI 코드는 복제본이 없는 Valkey(클러스터 모드 비활성화됨) 캐시 클러스터를 생성합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache create-cache-cluster \
--cache-cluster-id my-cluster \
--cache-node-type cache.r4.large \
--engine valkey \
--num-cache-nodes 1 \
--snapshot-arns arn:aws:s3:::my_bucket/snapshot.rdb
```

Windows의 경우:

```
aws elasticache create-cache-cluster ^
--cache-cluster-id my-cluster ^
--cache-node-type cache.r4.large ^
--engine valkey ^
--num-cache-nodes 1 ^
--snapshot-arns arn:aws:s3:::my_bucket/snapshot.rdb
```

클러스터 모드가 활성화된 상태로 작업하려면 다음 주제를 참조하세요.

- 콘솔을 사용하려면 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 생성(콘솔)을 참조하세요.
- 를 사용하려면 섹션을 AWS CLI참조하세요Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 생성(AWS CLI).

단계 3. 클러스터에 대한 액세스 권한 부여

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

모든 ElastiCache 클러스터는 Amazon EC2 인스턴스에서 액세스하도록 설계되었습니다. 가장 일반적인 시나리오는 동일한 Amazon Virtual Private Cloud(Amazon VPC)의 Amazon EC2 인스턴스에서 ElastiCache 클러스터에 액세스하는 것입니다. 이 연습의 경우입니다.

기본적으로 클러스터에 대한 네트워크 액세스는 생성할 때 사용한 계정에만 제한됩니다. EC2 인스턴스에서 클러스터에 연결하려면 먼저 EC2 인스턴스가 클러스터에 액세스할 수 있는 권한을 부여해야 합니다.

가장 일반적인 사용 사례는 EC2 인스턴스에 배포된 애플리케이션이 동일한 의 클러스터에 연결해야 하는 경우입니다VPC. 동일한 에서 EC2 인스턴스와 클러스터 간의 액세스를 관리하는 가장 간단한 방법은 다음을 수행하는 VPC 것입니다.

1. 클러스터의 VPC 보안 그룹을 생성합니다. 이 보안 그룹을 사용해 클러스터 인스턴스에 대한 액세스를 제한할 수 있습니다. 예를 들어 클러스터를 생성할 때 클러스터에 할당한 포트와 클러스터 TCP에 액세스하는 데 사용할 IP 주소를 사용하여 액세스를 허용하는 이 보안 그룹에 대한 사용자 지정 규칙을 생성할 수 있습니다.

   Valkey 또는 Redis OSS 클러스터 및 복제 그룹의 기본 포트는 입니다6379.

   > **Important**
   >
   > Amazon ElastiCache 보안 그룹은 Amazon Virtual Private Cloud 환경()에서 실행되지 않는 클러스터에만 적용됩니다VPC. Amazon Virtual Private Cloud를 실행하는 경우 콘솔 탐색 창에서 보안 그룹을 사용할 수 없습니다.
   > Amazon 에서 ElastiCache 노드를 실행하는 경우 VPC보안 그룹과 다른 Amazon VPC 보안 ElastiCache 그룹을 사용하여 클러스터에 대한 액세스를 제어합니다. Amazon ElastiCache 에서 를 사용하는 방법에 대한 자세한 내용은 섹션을 VPC참조하세요. Amazon VPCs 및 ElastiCache 보안

2. EC2 인스턴스(웹 및 애플리케이션 서버)에 대한 VPC 보안 그룹을 생성합니다. 이 보안 그룹은 필요한 경우 의 라우팅 테이블을 통해 인터넷에서 EC2 인스턴스에 대한 액세스를 허용할 수 VPC있습니다. 예를 들어 포트 22를 통해 EC2 인스턴스에 TCP 액세스할 수 있도록 이 보안 그룹에 규칙을 설정할 수 있습니다.

3. 클러스터의 보안 그룹에서 EC2 인스턴스에 대해 생성한 보안 그룹의 연결을 허용하는 사용자 지정 규칙을 생성합니다. 그러면 보안 그룹의 모든 구성원이 클러스터에 액세스하도록 허용됩니다.

> **Note**
>
> Local Zones를 사용할 계획이라면 활성화했는지 확인합니다. 해당 로컬 영역에서 서브넷 그룹을 생성하면 VPC가 해당 로컬 영역으로 확장되고 VPC 는 서브넷을 다른 가용 영역의 서브넷으로 취급합니다. 모든 관련 게이트웨이 및 라우팅 테이블이 자동으로 조정됩니다.

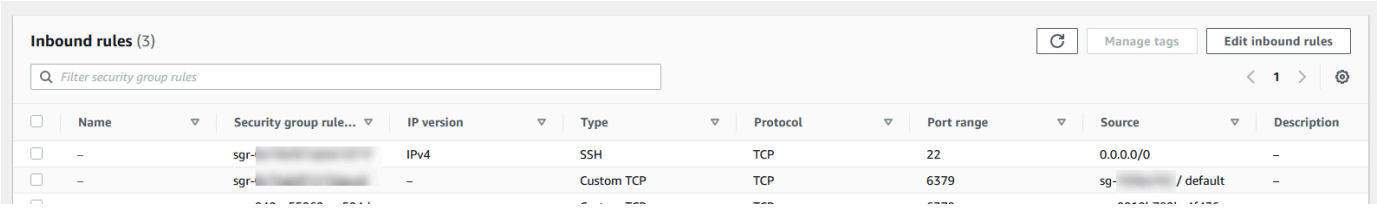
다른 VPC 보안 그룹의 연결을 허용하는 규칙을 보안 그룹에 생성하려면

1. AWS 관리 콘솔에 로그인하고 https://console.aws.amazon.com/vpc 에서 Amazon VPC 콘솔을 엽니다.
2. 탐색 창에서 보안 그룹을 선택합니다.
3. 클러스터 인스턴스에 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. 유형에서 사용자 지정 TCP 규칙 을 선택합니다.

   a. [Port Range]에 대해 클러스터를 만들 때 사용한 포트를 지정합니다.

      Valkey 또는 Redis OSS 클러스터 및 복제 그룹의 기본 포트는 입니다6379.

   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 노드에 연결할 수 있습니다.

다른 Amazon VPC, 다른 AWS 리전 또는 회사 네트워크에서 ElastiCache 클러스터에 액세스하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- Amazon에서 ElastiCache 캐시에 액세스하기 위한 액세스 패턴 VPC
- 외부에서 ElastiCache 리소스에 액세스 AWS

4단계. 클러스터의 노드에 연결

계속하기 전에 단계 3. 클러스터에 대한 액세스 권한 부여 단계를 완료하세요.

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

Amazon EC2 인스턴스는 클러스터 노드에 연결하도록 권한을 부여한 경우에만 연결할 수 있습니다.

노드 엔드포인트 찾기

클러스터가 사용 가능한 상태이고 이에 대한 액세스 권한을 부여한 경우 Amazon EC2 인스턴스에 로그인하여 클러스터에 연결할 수 있습니다. 이를 수행하려면 먼저 엔드포인트를 결정해야 합니다.

Valkey(클러스터 모드 비활성화됨) 클러스터의 엔드포인트 찾기(콘솔)

Valkey(클러스터 모드 비활성화됨) 클러스터에 노드가 하나만 있는 경우 노드의 엔드포인트는 읽기 및 쓰기 모두에 사용됩니다. 클러스터에 여러 노드가 있는 경우 세 가지 유형의 엔드포인트(기본 엔드포인트, 리더 엔드포인트 및 노드 엔드포인트)가 있습니다.

기본 엔드포인트는 항상 클러스터의 기본 노드로 확인되는 DNS 이름입니다. 기본 엔드포인트는 읽기 전용 복제본을 기본 역할로 승격하는 것과 같은 클러스터 변경의 영향을 받지 않습니다. 쓰기 활동의 경우 애플리케이션을 기본 엔드포인트에 연결하는 것이 좋습니다.

리더 엔드포인트는 들어오는 연결을 ElastiCache 클러스터의 모든 읽기 전용 복제본 간에 균등하게 분할합니다. 애플리케이션이 연결을 생성하는 시기 또는 애플리케이션에서 연결을 다시 사용하는 방법과 같은 추가 요소가 트래픽 분산을 결정합니다. 리더 엔드포인트는 복제본이 추가 또는 제거되는 클러스터의 변경 사항을 실시간으로 반영합니다. ElastiCache 클러스터의 여러 읽기 전용 복제본을 서로 다른 AWS 가용 영역(AZ)에 배치하여 리더 엔드포인트의 고가용성을 보장할 수 있습니다.

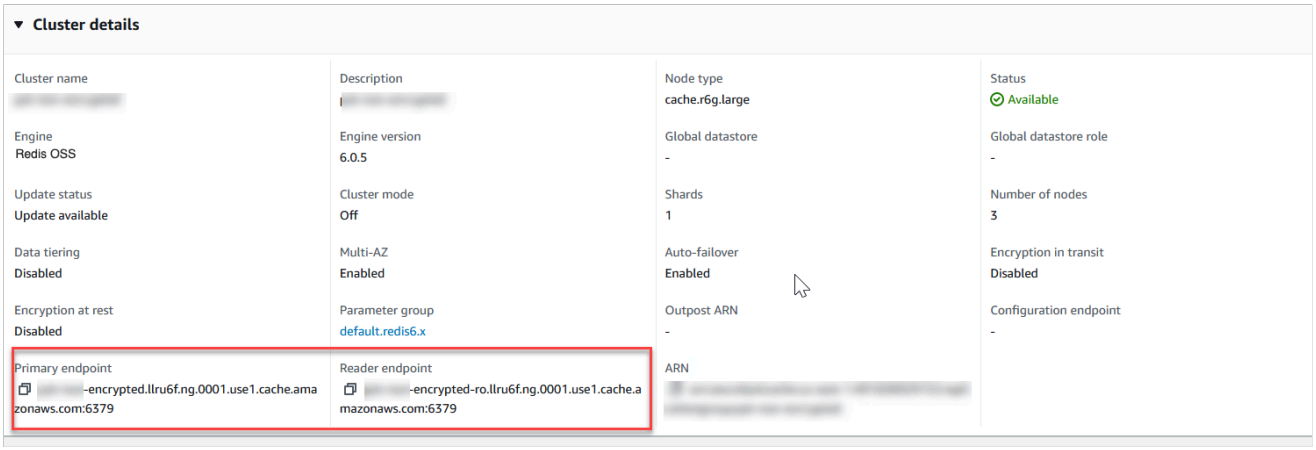
> **Note**
>
> 리더 엔드포인트는 로드 밸런서가 아닙니다. 이는 라운드 로빈 방식으로 복제본 노드 중 하나의 IP 주소로 확인되는 DNS 레코드입니다.

읽기 활동의 경우 애플리케이션은 클러스터의 어떤 노드에도 연결할 수 있습니다. 기본 엔드포인트와 달리, 노드 엔드포인트는 특정 엔드포인트로 확인됩니다. 복제본을 추가하거나 삭제하는 것과 같이 클러스터를 변경하면 애플리케이션에서 노드 엔드포인트를 업데이트해야 합니다.

Valkey(클러스터 모드 비활성화됨) 클러스터의 엔드포인트를 찾으려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 Valkey 캐시 또는 Redis OSS 캐시를 선택합니다.

   클러스터 화면에 기존 Valkey 또는 Redis OSS 서버리스 캐시, Valkey(클러스터 모드 비활성화됨) 및 Valkey(클러스터 모드 활성화됨) 클러스터가 포함된 목록이 표시됩니다. Valkey(클러스터 모드 비활성화됨) 클러스터 생성(콘솔) 섹션에서 생성한 클러스터를 선택합니다.
3. 클러스터의 기본 엔드포인트 및/또는 리더 엔드포인트를 찾으려면 클러스터 이름(라디오 버튼 아님)을 선택합니다.

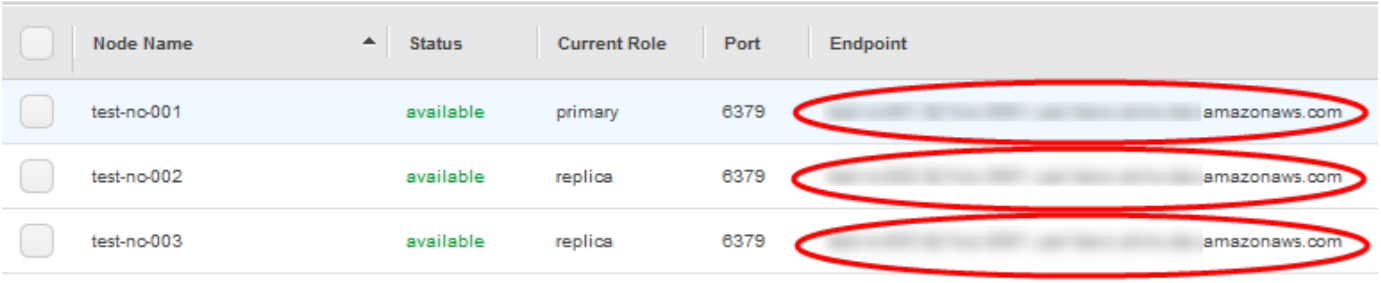


Valkey(클러스터 모드 비활성화됨) 클러스터의 기본 및 리더 엔드포인트

클러스터에 노드가 하나 뿐이면 기본 엔드포인트가 없으며 다음 단계에서 계속할 수 있습니다.

4. Valkey(클러스터 모드 비활성화됨) 클러스터에 복제본 노드가 있는 경우 클러스터의 이름을 선택한 다음 노드 탭을 선택하여 클러스터의 복제본 노드 엔드포인트를 찾을 수 있습니다.

   노드 화면에 클러스터의 각 노드, 기본 및 복제본이 엔드포인트와 함께 나열됩니다.



Valkey(클러스터 모드 비활성화됨) 클러스터의 노드 엔드포인트

5. 엔드포인트를 클립보드에 복사하려면

   a. 한 번에 엔드포인트 하나씩, 복사할 엔드포인트를 찾습니다.

   b. 엔드포인트 바로 앞에 있는 복사 아이콘을 선택합니다.

   엔드포인트가 클립보드에 복사됩니다. 엔드포인트를 사용하여 노드에 연결하는 방법에 대한 자세한 내용은 노드에 연결 섹션을 참조하세요.

Valkey(클러스터 모드 비활성화됨) 기본 엔드포인트는 다음과 같습니다. 전송 중 데이터 암호화가 활성화되어 있는지 여부에 따라 차이가 있습니다.

전송 중 데이터 암호화가 비활성화된 경우

```
clusterName.xxxxxx.nodeId.regionAndAz.cache.amazonaws.com:port

redis-01.7abc2d.0001.usw2.cache.amazonaws.com:6379
```

전송 중 데이터 암호화가 활성화된 경우

```
master.clusterName.xxxxxx.regionAndAz.cache.amazonaws.com:port

master.ncit.ameaqx.use1.cache.amazonaws.com:6379
```

사용자의 엔드포인트를 찾는 방법을 더 자세히 알아보려면 엔진과 실행 중인 클러스터 유형에 관한 주제를 참조하세요.

- 에서 연결 엔드포인트 찾기 ElastiCache
- Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터(콘솔)의 엔드포인트 찾기 - 클러스터의 구성 엔드포인트가 필요합니다.

- 엔드포인트 찾기(AWS CLI)
- 엔드포인트 찾기(ElastiCache API)

Valkey 또는 Redis OSS 클러스터 또는 복제 그룹에 연결(Linux)

이제 필요한 엔드포인트가 있으므로 EC2 인스턴스에 로그인하고 클러스터 또는 복제 그룹에 연결할 수 있습니다. 다음 예제에서는 valkey-cli 유틸리티를 사용하여 클러스터에 연결합니다. 최신 버전의 valkey-cli는 암호화/인증 지원 클러스터 연결을 위해 SSL/TLS도 지원합니다.

다음 예제에서는 Amazon Linux 및 Amazon Linux 2를 실행하는 Amazon EC2 인스턴스를 사용합니다. valkey-cli를 설치하고 다른 Linux 배포와 컴파일하는 방법에 대한 자세한 내용은 특정 운영 체제 설명서를 참조하세요.

> **Note**
>
> 이 프로세스는 계획되지 않은 용도로만 valkey-cli 유틸리티를 사용하여 연결을 테스트하는 것을 다룹니다. 지원되는 Valkey 및 Redis OSS 클라이언트 목록은 Valkey 설명서 를 참조하세요. 와 함께 를 AWS SDKs 사용하는 예제는 섹션을 ElastiCache참조하세요자습서: Python 및 시작하기 ElastiCache.

클러스터 모드가 비활성화된 암호화되지 않은 클러스터에 연결

1. 다음 명령을 실행하여 클러스터에 연결하고 바꿉니다.*primary-endpoint* 그리고 *port number* 클러스터의 엔드포인트와 포트 번호를 사용합니다. (Valkey 또는 Redis의 기본 포트OSS는 6379입니다.)

   ```
   src/valkey-cli -h primary-endpoint -p port number
   ```

   Valkey 또는 Redis OSS 명령 프롬프트의 결과는 다음과 비슷합니다.

   ```
   primary-endpoint:port number
   ```

2. 이제 Valkey 또는 Redis OSS 명령을 실행할 수 있습니다.

   ```
   set x Hello
   OK

   get x
   ```

```
"Hello"
```

클러스터 모드가 활성화된 암호화되지 않은 클러스터에 연결

1. 다음 명령을 실행하여 클러스터에 연결하고 바꿉니다.*configuration-endpoint* 그리고 *port number* 클러스터의 엔드포인트와 포트 번호를 사용합니다. (Valkey 또는 Redis의 기본 포트OSS는 6379입니다.)

   ```
   src/valkey-cli -h configuration-endpoint -c -p port number
   ```

   > **Note**
   >
   > 앞의 명령에서 옵션 -c는 - 및MOVED -ASK 리디렉션 다음에 클러스터 모드를 활성화합니다.

   Valkey 또는 Redis OSS 명령 프롬프트의 결과는 다음과 비슷합니다.

   ```
   configuration-endpoint:port number
   ```

2. 이제 Valkey 또는 Redis OSS 명령을 실행할 수 있습니다. 리디렉션은 -c 옵션을 사용하여 활성화했기 때문에 발생합니다. 리디렉션이 활성화되지 않은 경우 명령은 MOVED 오류를 반환합니다. MOVED 오류에 대한 자세한 내용은 Redis OSS 클러스터 사양 섹션을 참조하세요.

   ```
   set x Hi
   -> Redirected to slot [16287] located at 172.31.28.122:6379
   OK
   set y Hello
   OK
   get y
   "Hello"
   set z Bye
   -> Redirected to slot [8157] located at 172.31.9.201:6379
   OK
   get z
   "Bye"
   get x
   -> Redirected to slot [16287] located at 172.31.28.122:6379
   "Hi"
   ```

암호화/인증이 활성화된 클러스터에 연결

기본적으로 valkey-cli는 Valkey 또는 Redis 에 연결할 때 암호화되지 않은 TCP 연결을 사용합니다 OSS. 이 옵션은 이전 명령줄 액세스 다운로드 및 설정 섹션에 표시된 대로 valkey-cli 컴파일 시 SSL/TLS를 BUILD_TLS=yes 활성화합니다. 활성화AUTH는 선택 사항입니다. 그러나 를 활성화하려면 전송 중 암호화를 활성화해야 합니다AUTH. 암호화 및 인증에 대한 ElastiCache 자세한 내용은 섹션을 참조하세요ElastiCache 전송 중 암호화(TLS).

> **Note**
>
> valkey-cli와 --tls 함께 옵션을 사용하여 클러스터 모드 활성화 및 비활성화된 암호화된 클러스터 모두에 연결할 수 있습니다. 클러스터에 AUTH 토큰 세트가 있는 경우 옵션을 사용하여 AUTH 암호를 -a 제공할 수 있습니다.

다음 예에서는 를 교체해야 합니다.*cluster-endpoint* 그리고 *port number* 클러스터의 엔드포인트와 포트 번호를 사용합니다. (Valkey 또는 Redis의 기본 포트OSS는 6379입니다.)

클러스터 모드가 비활성화된 암호화된 클러스터에 연결

다음은 암호화 및 인증이 활성화된 클러스터에 연결하는 예제입니다.

```
src/valkey-cli -h cluster-endpoint --tls -a your-password -p port number
```

다음은 암호화만 활성화된 클러스터에 연결하는 예제입니다.

```
src/valkey-cli -h cluster-endpoint --tls -p port number
```

클러스터 모드가 활성화된 암호화된 클러스터에 연결

다음은 암호화 및 인증이 활성화된 클러스터에 연결하는 예제입니다.

```
src/valkey-cli -c -h cluster-endpoint --tls -a your-password -p port number
```

다음은 암호화만 활성화된 클러스터에 연결하는 예제입니다.

```
src/valkey-cli -c -h cluster-endpoint --tls -p port number
```

클러스터에 연결한 후 암호화되지 않은 클러스터에 대해 앞의 예제와 같이 Valkey 또는 Redis OSS 명령을 실행할 수 있습니다.

valkey-cli 대체

클러스터 모드가 활성화되어 있지 않고 valkey-cli 컴파일을 거치지 않고 짧은 테스트를 위해 클러스터에 연결해야 하는 경우 텔넷 또는 openssl을 사용할 수 있습니다. 다음 예제 명령에서 *cluster-endpoint* 그리고 *port number* 클러스터의 엔드포인트와 포트 번호를 사용합니다. (Valkey 또는 Redis의 기본 포트OSS는 6379입니다.)

다음은 암호화 및/또는 인증이 활성화되어 있으며 클러스터 모드가 비활성화된 클러스터에 연결하는 예제입니다.

```
openssl s_client -connect cluster-endpoint:port number
```

클러스터에 암호가 설정되어 있으면 먼저 클러스터에 연결합니다. 연결 후 다음 명령을 사용하여 클러스터를 인증한 다음 Enter 키를 누릅니다. 다음 예에서는 를 바꿉니다.*your-password* 클러스터의 암호와 함께 사용합니다.

```
Auth your-password
```

다음은 암호화 또는 인증이 활성화되어 있지 않으며 클러스터 모드가 비활성화된 클러스터에 연결하는 예제입니다.

```
telnet cluster-endpoint port number
```

Valkey 또는 Redis OSS 클러스터 또는 복제 그룹에 연결(Windows)

Valkey 또는 Redis 를 사용하여 EC2 Windows 인스턴스에서 Valkey CLI 또는 Redis OSS 클러스터에 연결하려면 valkey-cli 패키지를 다운로드하고 valkey-cli.exe를 사용하여 EC2 Windows 인스턴스에서 Valkey 또는 Redis OSS 클러스터에 연결OSSCLI해야 합니다.

다음 예제에서는 valkey-cli 유틸리티를 사용하여 암호화가 활성화되지 않고 Valkey 또는 Redis 를 실행하는 클러스터에 연결합니다OSS. Valkey 또는 Redis OSS 및 사용 가능한 명령에 대한 자세한 내용은 Valkey 웹 사이트의 Valkey 및 Redis OSS 명령을 참조하세요.

valkey-cli를 사용하여 암호화가 활성화되지 않은 Valkey 또는 Redis OSS 클러스터에 연결하려면

1. 원하는 연결 유틸리티를 사용하여 Amazon EC2 인스턴스에 연결합니다. Amazon EC2 인스턴스에 연결하는 방법에 대한 지침은 Amazon EC2 시작 안내서를 참조하세요.
2. 인터넷 브라우저https://github.com/microsoftarchive/redis/releases/download/win-3.0.504/Redis-x64-3.0.504.zip에 링크를 복사하여 붙여넣어 의 사용 가능한 릴리스에서 Valkey 클라이언트

의 zip 파일을 다운로드합니다. GitHub https://github.com/microsoftarchive/redis/releases/tag/win-3.0.504

원하는 폴더/경로에 zip 파일의 압축을 풉니다.

명령 프롬프트를 열고 Valkey 디렉터리로 변경한 다음 명령을 실행합니다`c:\Valkey>valkey-cli -h Redis_Cluster_Endpoint -p 6379`.

예:

```
c:\Valkey>valkey-cli -h cmd.xxxxxxx.ng.0001.usw2.cache.amazonaws.com -p 6379
```

3. Valkey 또는 Redis OSS 명령을 실행합니다.

   이제 클러스터에 연결되어 다음과 같이 Valkey 또는 Redis OSS 명령을 실행할 수 있습니다.

```
set a "hello"           // Set key "a" with a string value and no expiration
OK
get a                   // Get value for key "a"
"hello"
get b                   // Get value for key "b" results in miss
(nil)
set b "Good-bye" EX 5   // Set key "b" with a string value and a 5 second expiration
"Good-bye"
get b                   // Get value for key "b"
"Good-bye"
                        // wait >= 5 seconds
get b
(nil)                   // key has expired, nothing returned
quit                    // Exit from valkey-cli
```

추가 정보

이제 시작하기 연습을 시도했으므로 다음 섹션을 탐색하여 ElastiCache 및 사용 가능한 도구에 대해 자세히 알아볼 수 있습니다.

- 시작하기 AWS
- Amazon Web Services용 도구
- AWS Command Line Interface
- Amazon ElastiCache API 참조

시작하기 연습을 완료한 후 다음 섹션을 읽고 ElastiCache 관리에 대해 자세히 알아볼 수 있습니다.

- 노드 크기 선택

  캐시하려는 모든 데이터를 수용할 만큼의 충분한 캐시를 원하며, 동시에 필요한 것보다 더 많은 캐시의 비용을 지불하고 싶지 않은 경우 이 섹션을 읽어보면 최적의 노드 크기를 선택하는 데 도움이 됩니다.

- ElastiCache 모범 사례 및 캐싱 전략

  클러스터의 효율성에 부정적인 영향을 미칠 수 있는 문제를 확인하고 해결하세요.

자체 ElastiCache (Redis OSS) 클러스터 설계

다음은 자체 ElastiCache (Redis OSS) 클러스터를 설계하기 위해 수행해야 하는 일회성 작업입니다.

설정에 대한 자세한 내용은 섹션을 ElastiCache 참조하세요설정 ElastiCache.

주제

- 1단계: 서브넷 그룹 생성
- 2단계: 클러스터 생성
- 3단계: 클러스터에 대한 액세스 허가
- 4단계: 클러스터 노드에 연결

1단계: 서브넷 그룹 생성

클러스터를 생성하기 전에 먼저 서브넷 그룹을 생성합니다. 캐시 서브넷 그룹은 에서 캐시 클러스터에 대해 지정할 수 있는 서브넷 모음입니다VPC. 에서 캐시 클러스터를 시작할 때 캐시 서브넷 그룹을 선택해야 VPC합니다. 그런 다음 해당 캐시 서브넷 그룹을 ElastiCache 사용하여 해당 서브넷 내의 IP 주소를 클러스터의 각 캐시 노드에 할당합니다.

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 그룹과 AWS CLI를 생성하는 방법을 보여 줍니다.

서브넷 그룹 생성(콘솔)

다음 절차는 서브넷 그룹을 생성하는 방법을 보여줍니다(콘솔).

서브넷 그룹을 생성하려면(콘솔)

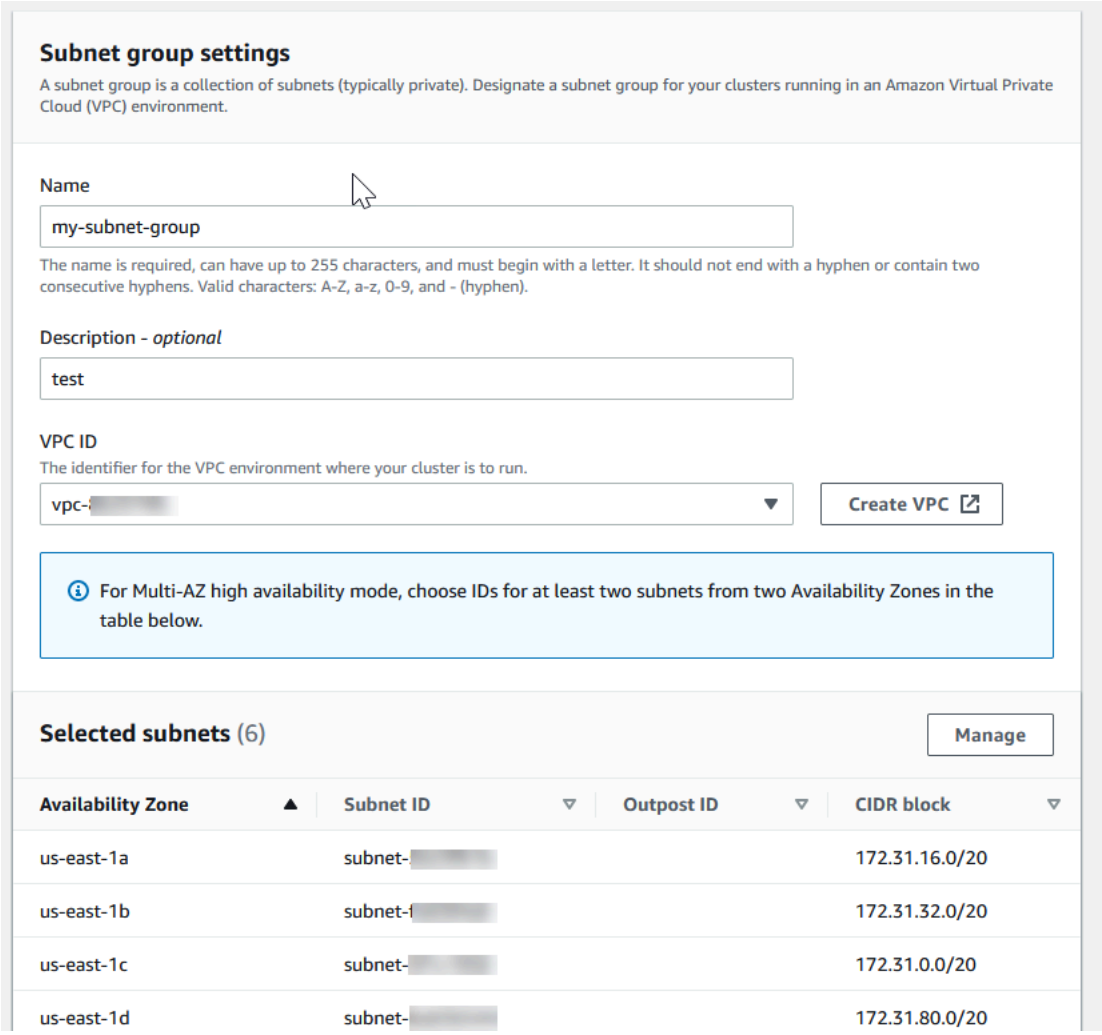
1. AWS 관리 콘솔에 로그인하고 에서 ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 탐색 목록에서 [Subnet Groups]를 선택합니다.
3. [Create Subnet Group]을 선택합니다.
4. [Create Subnet Group] 마법사에서 다음을 수행합니다. 모든 설정이 원하는 대로 설정되었으면 [Yes, Create]를 선택합니다.

    a. [Name] 상자에 서브넷 그룹의 이름을 입력합니다.

    b. [Description] 상자에 서브넷 그룹에 대한 설명을 입력합니다.

    c. VPC ID 상자에서 VPC 생성한 Amazon을 선택합니다.

    d. 가용 영역 및 서브넷 ID 목록에서 가용 영역 또는 로컬 영역과 프라이빗 서브넷의 ID를 선택한 다음 추가를 선택합니다.



5. 나타나는 확인 메시지에서 [Close]를 선택합니다.

새 서브넷 그룹이 ElastiCache 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-cache-subnet-group` 명령을 사용하여 서브넷 그룹을 생성합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache create-cache-subnet-group \
    --cache-subnet-group-name mysubnetgroup \
    --cache-subnet-group-description "Testing" \
    --subnet-ids subnet-53df9c3a
```

Windows의 경우:

```
aws elasticache create-cache-subnet-group ^
    --cache-subnet-group-name mysubnetgroup ^
    --cache-subnet-group-description "Testing" ^
    --subnet-ids subnet-53df9c3a
```

이 명령은 다음과 유사한 출력을 생성합니다.

```
{
    "CacheSubnetGroup": {
        "VpcId": "vpc-37c3cd17",
        "CacheSubnetGroupDescription": "Testing",
        "Subnets": [
            {
                "SubnetIdentifier": "subnet-53df9c3a",
                "SubnetAvailabilityZone": {
                    "Name": "us-west-2a"
                }
            }
        ],
        "CacheSubnetGroupName": "mysubnetgroup"
    }
}
```

자세한 내용은 AWS CLI 주제를 참조하세요.create-cache-subnet-group.

2단계: 클러스터 생성

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

생성하려는 클러스터는 활성화되고 샌드박스에서 실행되지 않습니다. 인스턴스를 삭제할 때까지 인스턴스에 대한 표준 ElastiCache 사용 요금이 발생합니다. 여기에 설명된 연습을 한 번에 끝내고 연습을 마칠 때 클러스터를 삭제하면 총 청구 비용이 가장 적게 듭니다(일반적으로 1달러 미만). ElastiCache 사용률에 대한 자세한 내용은 Amazon 섹션을 ElastiCache 참조하세요.

클러스터는 Amazon VPC 서비스를 기반으로 가상 프라이빗 클라우드(VPC)에서 시작됩니다.

RedisOSS(클러스터 모드 비활성화됨) 클러스터 생성(콘솔)

ElastiCache 콘솔을 사용하여 RedisOSS(클러스터 모드 비활성화됨) 클러스터를 생성하려면

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 오른쪽 상단 모서리의 목록에서 이 클러스터를 시작할 AWS 리전을 선택합니다.
3. 탐색 창에서 시작하기(Get started)를 선택하세요.
4. 생성을 VPC 선택하고 Virtual Private Cloud 생성(VPC)에 설명된 단계를 따릅니다.
5. ElastiCache 대시보드 페이지에서 Valkey 캐시 또는 Redis OSS 캐시를 선택합니다. 이 연습에서는 Redis OSS 캐시 를 선택한 다음 Redis OSS 캐시 생성 을 선택합니다.
6. 클러스터 설정(Cluster settings)에서 다음을 수행합니다.
   a. Configure and create a new cluster(새 클러스터 구성 및 생성)를 선택합니다.
   b. Cluster mode(클러스터 모드)에서 Enabled(사용 설정됨)를 선택합니다.
   c. Cluster info(클러스터 정보)에 Name(이름) 값을 입력합니다.
   d. (선택 사항) 설명(Description) 값을 입력합니다.
7. Location(위치)에서 다음을 수행합니다.

   AWS Cloud

   1. AWS Cloud의 경우 Multi-AZ(다중 AZ) 및 Auto-failover(자동 장애 조치)의 기본 설정을 수락하는 것이 좋습니다. 자세한 내용은 다중 AZ 를 사용한 ElastiCache (Redis OSS)의 가동 중지 시간 최소화를 참조하세요.
   2. 클러스터 설정(Cluster settings)에서 다음을 수행합니다.
      a. 엔진 버전(Engine version)의 경우 사용 가능한 버전을 선택합니다.
      b. 포트(Port)의 경우 기본 포트인 6379를 사용합니다. 다른 포트를 사용해야 하는 경우 포트 번호를 입력합니다.
      c. 파라미터 그룹에서 파라미터 그룹을 선택하거나 새 파라미터 그룹을 만듭니다. 파라미터 그룹은 클러스터의 런타임 파라미터를 제어합니다. 파라미터 그룹에 대한 자세한 정보는 Valkey 및 Redis OSS 파라미터 및 ElastiCache 파라미터 그룹 생성 섹션을 참조하세요.

> **Note**
>
> 파라미터 그룹을 선택하여 엔진 구성 값을 설정하면 해당 파라미터 그룹이 글로벌 데이터 스토어의 모든 클러스터에 적용됩니다. 파라미터 그룹 페이지에서 yes/no 글로벌 속성은 파라미터 그룹이 글로벌 데이터 스토어의 일부인지 여부를 나타냅니다.

d. 노드 유형에서 아래쪽 화살표 ( ) 를 선택합니다. 노드 유형 변경 대화 상자에서 원하는 노드 유형의 인스턴스 패밀리 값을 선택합니다. 그런 다음 이 클러스터에 사용할 노드 유형을 선택한 다음 저장을 선택합니다.

   자세한 정보는 노드 크기 선택 섹션을 참조하세요.

   r6gd 노드 유형을 선택하는 경우 데이터 계층화가 자동으로 사용 설정됩니다. 자세한 내용은 의 데이터 계층화 ElastiCache 단원을 참조하십시오.

e. Number of replicas(복제본 개수)의 경우 원하는 읽기 전용 복제본 개수를 선택합니다. 다중 AZ를 사용 설정한 경우 숫자는 1~5 사이가 되어야 합니다.

3. 연결 아래

   a. 네트워크 유형에서 이 클러스터가 지원할 IP 버전을 선택합니다.

   b. 서브넷 그룹 의 경우 이 클러스터에 적용할 서브넷을 선택합니다. 는 해당 서브넷 그룹을 ElastiCache 사용하여 해당 서브넷 내의 서브넷 및 IP 주소를 선택하여 노드와 연결합니다. ElastiCache 클러스터에는 듀얼 스택 모드에서 작동하려면 IPv4 및 IPv6 주소가 할당된 듀얼 스택 서브넷이 필요하며, IPv6는 IPv6전용으로 작동하려면 전용 서브넷이 필요합니다.

      새 서브넷 그룹을 생성할 때 해당 서브넷 그룹이 속한 VPC ID를 입력합니다.

      자세한 내용은 다음을 참조하세요.

      - 에서 네트워크 유형 선택 ElastiCache.
      - 에서 서브넷을 생성합니다VPC.

      에서 로컬 영역 사용 ElastiCache를 사용하는 경우 로컬 영역에 있는 서브넷을 선택하거나 생성해야 합니다.

자세한 내용은 서브넷 및 서브넷 그룹 단원을 참조하십시오.

4. 가용 영역 배치(Availability zone placements)의 경우 다음 두 가지 옵션이 있습니다.

   - 기본 설정 없음 - 가용 영역을 ElastiCache 선택합니다.
   - 가용 영역 지정 – 각 클러스터의 가용 영역을 지정합니다.

     가용 영역을 지정하도록 선택한 경우 샤드에 있는 각 클러스터에 대해 목록에서 가용 영역을 선택합니다.

   자세한 내용은 에 대한 리전 및 가용 영역 선택 ElastiCache 단원을 참조하십시오.

5. 다음(Next)을 선택합니다.
6. 고급 Redis OSS 설정에서

   - 보안(Security)의 경우

     i. 데이터를 암호화하려면 다음과 같은 옵션이 있습니다.

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

       > **Note**
       >
       > Customer Managed AWS KMS 키를 선택하고 키를 선택하여 다른 암호화 키를 제공할 수 있습니다. 자세한 내용은 에서 AWS 고객 관리형 키 사용을 KMS참조하세요.

     - 전송 중 데이터 암호화 – 전송 데이터 암호화를 활성화합니다. 자세한 정보는 전송 중 데이터 암호화를 참조하세요. Redis OSS 엔진 버전 6.0 이상의 경우 전송 중 암호화를 활성화하면 다음 액세스 제어 옵션 중 하나를 지정하라는 메시지가 표시됩니다.
       - 액세스 제어 안 함 – 기본 설정입니다. 이 옵션은 클러스터에 대한 사용자 액세스를 제한하지 않는다는 의미입니다.
       - 사용자 그룹 액세스 제어 목록 - 클러스터에 액세스할 수 있는 사용자 집합이 정의된 사용자 그룹을 선택합니다. 자세한 내용은 콘솔 및 를 사용하여 사용자 그룹 관리 CLI 단원을 참조하십시오.

- AUTH 기본 사용자 - Valkey 및 Redis OSS 서버의 인증 메커니즘입니다. 자세한 내용은 AUTH를 참조하세요.
- AUTH – Redis OSS 서버의 인증 메커니즘입니다. 자세한 내용은 AUTH를 참조하세요.

> **Note**
>
> OSS 버전 3.2.10을 제외한 3.2.6 이상 Redis 버전의 경우 Redis가 유일한 옵션OSSAUTH입니다.

ii. 보안 그룹에서 이 클러스터에 사용할 보안 그룹을 선택합니다. 보안 그룹은 클러스터에 대한 네트워크 액세스를 제어하는 방화벽 역할을 합니다. 에 기본 보안 그룹을 사용하거나 새 보안 그룹을 VPC 생성할 수 있습니다.

보안 그룹에 대한 자세한 내용은 Amazon 사용 설명서의 에 대한 보안 그룹을 VPC 참조하세요. VPC

7. 정기적인 자동 백업을 예약할 경우 Enable automatic backups(자동 백업 활성화)를 선택한 후 자동으로 삭제되기 전에 각 자동 백업을 보존할 기간(일)을 입력합니다. 정기적인 자동 백업을 예약하지 않으려면 [Enable automatic backups] 확인란의 선택을 취소합니다. 어떤 경우든 수동 백업을 항상 생성할 수 있습니다.

   백업 및 복원에 대한 자세한 내용은 섹션을 참조하세요스냅샷 및 복원.

8. (선택 사항) 유지 관리 기간을 지정합니다. 유지 관리 기간은 클러스터에 대한 시스템 유지 관리를 예약하는 ElastiCache 매주 일반적으로 1시간의 시간입니다. ElastiCache 에서 유지 관리 기간의 날짜 및 시간을 선택하거나(기본 설정 없음) 직접 날짜, 시간 및 기간을 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 ElastiCache 클러스터 유지 관리 단원을 참조하십시오.

9. (선택 사항) 로그의 경우:

   - 로그 형식에서 텍스트 또는 를 선택합니다JSON.
   - 대상 유형에서 CloudWatch 로그 또는 Kinesis Firehose를 선택합니다.

- 로그 대상 에서 새로 만들기를 선택하고 CloudWatch 로그 로그 그룹 이름 또는 Firehose 스트림 이름을 입력하거나 기존 선택을 선택한 다음 CloudWatch 로그 로그 로그 그룹 이름 또는 Firehose 스트림 이름을 선택합니다.

10. 태그 의 경우 클러스터 및 기타 ElastiCache 리소스를 관리하는 데 도움이 되도록 태그 형식으로 각 리소스에 자체 메타데이터를 할당할 수 있습니다. 자세한 정보는 ElastiCache 리소스 태그 지정 섹션을 참조하세요.
11. Next(다음)를 선택합니다.
12. 입력 및 선택한 내용을 모두 검토한 다음 필요한 내용을 수정합니다. 준비가 되었으면 생성(Create)e을 선택합니다.

On premises

1. On premises(온프레미스)의 경우 Auto-failover(자동 장애 조치)를 사용 설정하는 것이 좋습니다. 자세한 내용은 다중 AZ를 사용한 ElastiCache (Redis OSS)의 가동 중지 시간 최소화를 참조하세요.
2. 클러스터 생성을 완료하려면 Outposts 사용의 단계를 수행합니다.

클러스터의 상태를 사용할 수 있게 되면 즉시 Amazon에 EC2 액세스 권한을 부여하고 연결한 다음 사용을 시작할 수 있습니다. 자세한 내용은 단계 3. 클러스터에 대한 액세스 권한 부여 및 4단계. 클러스터의 노드에 연결 단원을 참조하세요.

> **Important**
>
> 클러스터를 사용할 수 있으면 클러스터를 활동적으로사용하지 않더라도 클러스터가 사용 설정되어 있는 매 시간 또는 60분 미만 단위로 비용이 청구됩니다. 이 클러스터의 요금 발생을 중지하려면 클러스터를 삭제해야 합니다. 에서 클러스터 삭제 ElastiCache을 참조하세요.

RedisOSS(클러스터 모드 비활성화됨) 클러스터 생성(AWS CLI)

Example

다음 CLI 코드는 복제본이 없는 RedisOSS(클러스터 모드 비활성화됨) 캐시 클러스터를 생성합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache create-cache-cluster \
```

```
--cache-cluster-id my-cluster \
--cache-node-type cache.r4.large \
--engine redis \
--num-cache-nodes 1 \
--snapshot-arns arn:aws:s3:::my_bucket/snapshot.rdb
```

Windows의 경우:

```
aws elasticache create-cache-cluster ^
--cache-cluster-id my-cluster ^
--cache-node-type cache.r4.large ^
--engine redis ^
--num-cache-nodes 1 ^
--snapshot-arns arn:aws:s3:::my_bucket/snapshot.rdb
```

클러스터 모드가 활성화된 상태로 작업하려면 다음 주제를 참조하세요.

- 콘솔을 사용하려면 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 생성(콘솔)을 참조하세요.
- 를 사용하려면 섹션을 AWS CLI참조하세요Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 생성(AWS CLI).

3단계: 클러스터에 대한 액세스 허가

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

모든 ElastiCache 클러스터는 Amazon EC2 인스턴스에서 액세스하도록 설계되었습니다. 가장 일반적인 시나리오는 동일한 Amazon Virtual Private Cloud(Amazon VPC)의 Amazon EC2 인스턴스에서 ElastiCache 클러스터에 액세스하는 것입니다. 이 연습의 경우입니다.

기본적으로 클러스터에 대한 네트워크 액세스는 생성할 때 사용한 계정에만 제한됩니다. EC2 인스턴스에서 클러스터에 연결하려면 먼저 EC2 인스턴스가 클러스터에 액세스할 수 있는 권한을 부여해야 합니다. 필요한 단계는 클러스터를 EC2-VPC 또는 EC2-Classic으로 시작했는지에 따라 달라집니다.

가장 일반적인 사용 사례는 EC2 인스턴스에 배포된 애플리케이션이 동일한 의 클러스터에 연결해야 하는 경우입니다VPC. 동일한 에서 EC2 인스턴스와 클러스터 간의 액세스를 관리하는 가장 간단한 방법은 다음을 수행하는 VPC 것입니다.

1. 클러스터의 VPC 보안 그룹을 생성합니다. 이 보안 그룹을 사용해 클러스터 인스턴스에 대한 액세스를 제한할 수 있습니다. 예를 들어 클러스터를 생성할 때 클러스터에 할당한 포트와 클러스터 TCP에 액세스하는 데 사용할 IP 주소를 사용하여 액세스를 허용하는 이 보안 그룹에 대한 사용자 지정 규칙을 생성할 수 있습니다.

   Redis OSS 클러스터 및 복제 그룹의 기본 포트는 입니다6379.

   > **Important**
   >
   > Amazon ElastiCache 보안 그룹은 Amazon Virtual Private Cloud 환경()에서 실행되지 않는 클러스터에만 적용됩니다VPC. Amazon Virtual Private Cloud를 실행하는 경우 콘솔 탐색 창에서 보안 그룹을 사용할 수 없습니다.
   > Amazon 에서 ElastiCache 노드를 실행하는 경우 VPC보안 그룹과 다른 Amazon VPC 보안 ElastiCache 그룹을 사용하여 클러스터에 대한 액세스를 제어합니다. Amazon ElastiCache 에서 를 사용하는 방법에 대한 자세한 내용은 섹션을 VPC참조하세요. Amazon VPCs 및 ElastiCache 보안

2. EC2 인스턴스(웹 및 애플리케이션 서버)에 대한 VPC 보안 그룹을 생성합니다. 이 보안 그룹은 필요한 경우 의 라우팅 테이블을 통해 인터넷에서 EC2 인스턴스에 대한 액세스를 허용할 수 VPC있습니다. 예를 들어 포트 22를 통해 EC2 인스턴스에 TCP 액세스할 수 있도록 이 보안 그룹에 규칙을 설정할 수 있습니다.

3. 클러스터의 보안 그룹에서 EC2 인스턴스에 대해 생성한 보안 그룹의 연결을 허용하는 사용자 지정 규칙을 생성합니다. 그러면 보안 그룹의 모든 구성원이 클러스터에 액세스하도록 허용됩니다.

> **Note**
>
> 를 사용할 계획이라면 활성화했는지 에서 로컬 영역 사용 ElastiCache확인하세요. 해당 로컬 영역에서 서브넷 그룹을 생성하면 VPC가 해당 로컬 영역으로 확장되고 VPC 는 서브넷을 다른 가용 영역의 서브넷으로 취급합니다. 모든 관련 게이트웨이 및 라우팅 테이블이 자동으로 조정됩니다.

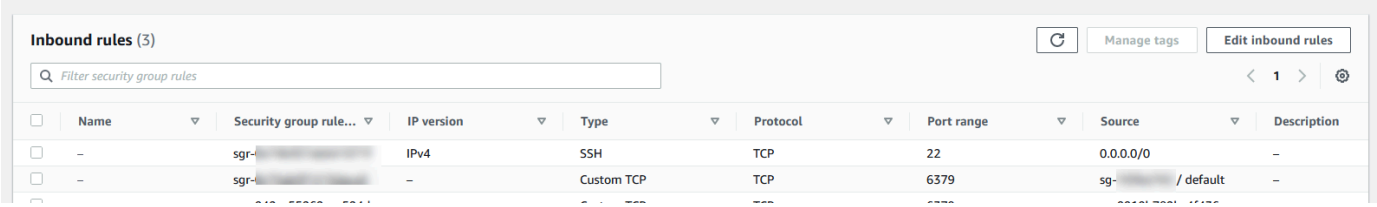
다른 VPC 보안 그룹의 연결을 허용하는 규칙을 보안 그룹에 생성하려면

1. AWS 관리 콘솔에 로그인하고 https://console.aws.amazon.com/vpc 에서 Amazon VPC 콘솔을 엽니다.
2. 탐색 창에서 보안 그룹을 선택합니다.
3. 클러스터 인스턴스에 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. 유형에서 사용자 지정 TCP 규칙 을 선택합니다.

   a. [Port Range]에 대해 클러스터를 만들 때 사용한 포트를 지정합니다.

      Redis OSS 클러스터 및 복제 그룹의 기본 포트는 입니다6379.

   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 노드에 연결할 수 있습니다.

다른 Amazon VPC, 다른 AWS 리전 또는 회사 네트워크에서 ElastiCache 클러스터에 액세스하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- Amazon에서 ElastiCache 캐시에 액세스하기 위한 액세스 패턴 VPC
- 외부에서 ElastiCache 리소스에 액세스 AWS

4단계: 클러스터 노드에 연결

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

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

Amazon EC2 인스턴스는 클러스터 노드에 연결하도록 권한을 부여한 경우에만 연결할 수 있습니다.

노드 엔드포인트 찾기

클러스터가 사용 가능한 상태이고 이에 대한 액세스 권한을 부여한 경우 Amazon EC2 인스턴스에 로그인하여 클러스터에 연결할 수 있습니다. 이를 수행하려면 먼저 엔드포인트를 결정해야 합니다.

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터의 엔드포인트 찾기(콘솔)

RedisOSS(클러스터 모드 비활성화됨) 클러스터에 노드가 하나만 있는 경우 노드의 엔드포인트는 읽기 및 쓰기 모두에 사용됩니다. 클러스터에 여러 노드가 있는 경우 세 가지 유형의 엔드포인트(기본 엔드포인트, 리더 엔드포인트 및 노드 엔드포인트)가 있습니다.

기본 엔드포인트는 항상 클러스터의 기본 노드로 확인되는 DNS 이름입니다. 기본 엔드포인트는 읽기 전용 복제본을 기본 역할로 승격하는 것과 같은 클러스터 변경의 영향을 받지 않습니다. 쓰기 활동의 경우 애플리케이션을 기본 엔드포인트에 연결하는 것이 좋습니다.

리더 엔드포인트는 ElastiCache (Redis OSS) 클러스터의 모든 읽기 전용 복제본 간에 엔드포인트에 대한 수신 연결을 균등하게 분할합니다. 애플리케이션이 연결을 생성하는 시기 또는 애플리케이션에서 연결을 다시 사용하는 방법과 같은 추가 요소가 트래픽 분산을 결정합니다. 리더 엔드포인트는 복제본이 추가 또는 제거되는 클러스터의 변경 사항을 실시간으로 반영합니다. ElastiCache (Redis OSS) 클러스터의 여러 읽기 전용 복제본을 서로 다른 AWS 가용 영역(AZ)에 배치하여 리더 엔드포인트의 고가용성을 보장할 수 있습니다.

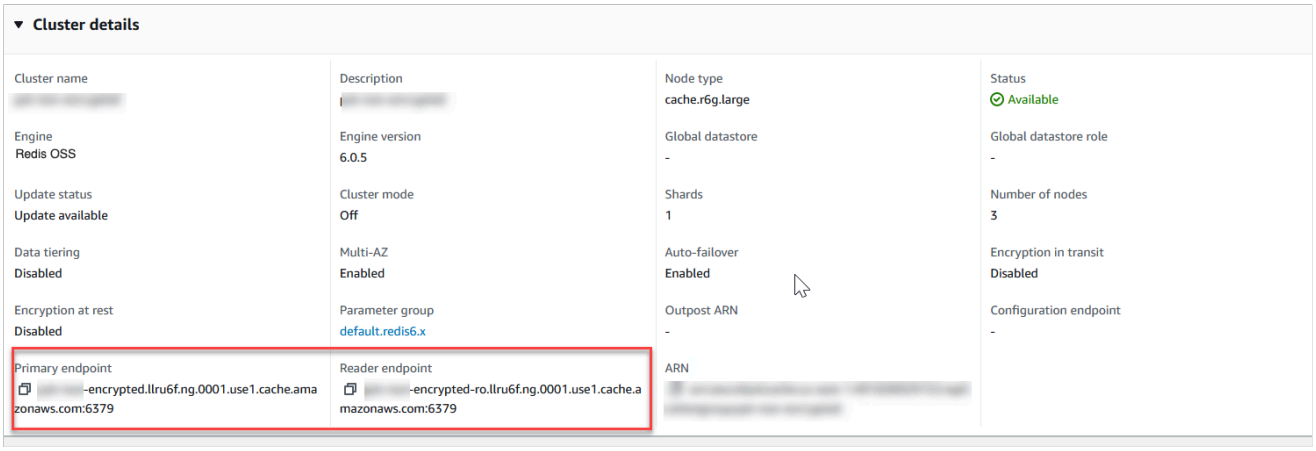
> **Note**
>
> 리더 엔드포인트는 로드 밸런서가 아닙니다. 이는 라운드 로빈 방식으로 복제본 노드 중 하나의 IP 주소로 확인되는 DNS 레코드입니다.

읽기 활동의 경우 애플리케이션은 클러스터의 어떤 노드에도 연결할 수 있습니다. 기본 엔드포인트와 달리, 노드 엔드포인트는 특정 엔드포인트로 확인됩니다. 복제본을 추가하거나 삭제하는 것과 같이 클러스터를 변경하면 애플리케이션에서 노드 엔드포인트를 업데이트해야 합니다.

RedisOSS(클러스터 모드 비활성화됨) 클러스터의 엔드포인트를 찾으려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 OSS 캐시 재분배 를 선택합니다.

   클러스터 화면에는 기존 Valkey 또는 Redis OSS 서버리스 캐시, RedisOSS(클러스터 모드 비활성화됨) 클러스터 및 RedisOSS(클러스터 모드 활성화됨) 클러스터가 포함된 목록이 표시됩니다. RedisOSS(클러스터 모드 비활성화됨) 클러스터 생성(콘솔) 섹션에서 생성한 클러스터를 선택합니다.
3. 클러스터의 기본 엔드포인트 및/또는 리더 엔드포인트를 찾으려면 클러스터 이름(라디오 버튼 아님)을 선택합니다.

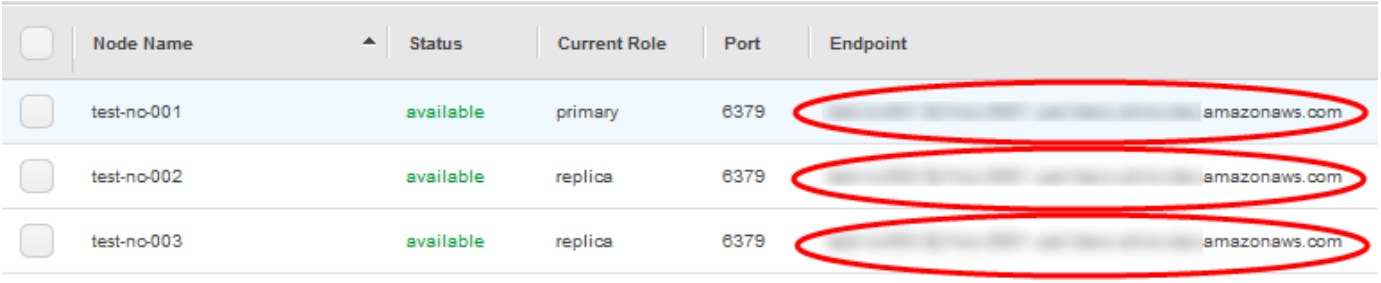


RedisOSS(클러스터 모드 비활성화됨) 클러스터의 프라이머리 및 리더 엔드포인트

클러스터에 노드가 하나 뿐이면 기본 엔드포인트가 없으며 다음 단계에서 계속할 수 있습니다.

4. RedisOSS(클러스터 모드 비활성화됨) 클러스터에 복제본 노드가 있는 경우 클러스터의 이름을 선택한 다음 노드 탭을 선택하여 클러스터의 복제본 노드 엔드포인트를 찾을 수 있습니다.

   노드 화면에 클러스터의 각 노드, 기본 및 복제본이 엔드포인트와 함께 나열됩니다.



RedisOSS(클러스터 모드 비활성화됨) 클러스터의 노드 엔드포인트

5. 엔드포인트를 클립보드에 복사하려면

   a. 한 번에 엔드포인트 하나씩, 복사할 엔드포인트를 찾습니다.

   b. 엔드포인트 바로 앞에 있는 복사 아이콘을 선택합니다.

   엔드포인트가 클립보드에 복사됩니다. 엔드포인트를 사용하여 노드에 연결하는 방법에 대한 자세한 내용은 노드에 연결 섹션을 참조하세요.

RedisOSS(클러스터 모드 비활성화됨) 기본 엔드포인트는 다음과 같습니다. 전송 중 데이터 암호화가 활성화되어 있는지 여부에 따라 차이가 있습니다.

전송 중 데이터 암호화가 비활성화된 경우

```
clusterName.xxxxxx.nodeId.regionAndAz.cache.amazonaws.com:port

redis-01.7abc2d.0001.usw2.cache.amazonaws.com:6379
```

전송 중 데이터 암호화가 활성화된 경우

```
master.clusterName.xxxxxx.regionAndAz.cache.amazonaws.com:port

master.ncit.ameaqx.use1.cache.amazonaws.com:6379
```

사용자의 엔드포인트를 찾는 방법을 더 자세히 알아보려면 엔진과 실행 중인 클러스터 유형에 관한 주제를 참조하세요.

- 에서 연결 엔드포인트 찾기 ElastiCache
- Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터(콘솔)의 엔드포인트 찾기 - 클러스터의 구성 엔드포인트가 필요합니다.

- 엔드포인트 찾기(AWS CLI)
- 엔드포인트 찾기(ElastiCache API)

Valkey 또는 Redis OSS 클러스터 또는 복제 그룹에 연결(Linux)

이제 필요한 엔드포인트가 있으므로 EC2 인스턴스에 로그인하고 클러스터 또는 복제 그룹에 연결할 수 있습니다. 다음 예제에서는 valkey-cli 유틸리티를 사용하여 클러스터에 연결합니다. 최신 버전의 valkey-cli는 암호화/인증 지원 클러스터를 연결하기 위해 SSL/TLS도 지원합니다.

다음 예제에서는 Amazon Linux 및 Amazon Linux 2를 실행하는 Amazon EC2 인스턴스를 사용합니다. 다른 Linux 배포와 함께 valkey-cli를 설치하고 컴파일하는 방법에 대한 자세한 내용은 특정 운영 체제에 대한 설명서를 참조하세요.

> **Note**
>
> 이 프로세스는 계획되지 않은 용도로만 valkey-cli 유틸리티를 사용하여 연결을 테스트하는 것을 다룹니다. 지원되는 클라이언트 목록은 Valkey 설명서 를 참조하세요. 와 함께 를 AWS SDKs 사용하는 예제는 섹션을 ElastiCache참조하세요자습서: Python 및 시작하기 ElastiCache.

클러스터 모드가 비활성화된 암호화되지 않은 클러스터에 연결

1. 다음 명령을 실행하여 클러스터에 연결하고 바꿉니다.*primary-endpoint* 그리고 *port number* 클러스터의 엔드포인트와 포트 번호를 사용합니다. (Valkey 및 Redis의 기본 포트OSS는 6379입니다.)

   ```
   src/valkey-cli -h primary-endpoint -p port number
   ```

   명령 프롬프트의 결과는 다음과 비슷합니다.

   ```
   primary-endpoint:port number
   ```

2. 이제 Valkey 및 Redis OSS 명령을 실행할 수 있습니다.

   ```
   set x Hello
   OK

   get x
   ```

```
"Hello"
```

클러스터 모드가 활성화된 암호화되지 않은 클러스터에 연결

1. 다음 명령을 실행하여 클러스터에 연결하고 바꿉니다.*configuration-endpoint* 그리고 *port number* 클러스터의 엔드포인트와 포트 번호를 사용합니다. (Valkey 및 Redis의 기본 포트OSS는 6379입니다.)

   ```
   src/valkey-cli -h configuration-endpoint -c -p port number
   ```

   > **Note**
   >
   > 앞의 명령에서 옵션 -c는 - 및MOVED -ASK 리디렉션 다음에 클러스터 모드를 활성화합니다.

   명령 프롬프트의 결과는 다음과 비슷합니다.

   ```
   configuration-endpoint:port number
   ```

2. 이제 Valkey 및 Redis OSS 명령을 실행할 수 있습니다. 리디렉션은 -c 옵션을 사용하여 활성화했기 때문에 발생합니다. 리디렉션이 활성화되지 않은 경우 명령은 MOVED 오류를 반환합니다. MOVED 오류에 대한 자세한 내용은 클러스터 사양 섹션을 참조하세요.

   ```
   set x Hi
   -> Redirected to slot [16287] located at 172.31.28.122:6379
   OK
   set y Hello
   OK
   get y
   "Hello"
   set z Bye
   -> Redirected to slot [8157] located at 172.31.9.201:6379
   OK
   get z
   "Bye"
   get x
   -> Redirected to slot [16287] located at 172.31.28.122:6379
   "Hi"
   ```

암호화/인증이 활성화된 클러스터에 연결

기본적으로 valkey-cli는 Valkey 및 Redis 에 연결할 때 암호화되지 않은 TCP 연결을 사용합니다OSS. 이 옵션은 이전 명령줄 액세스 다운로드 및 설정 섹션에 표시된 대로 valkey-cli 컴파일 시 SSL/TLS를 `BUILD_TLS=yes` 활성화합니다. 활성화AUTH는 선택 사항입니다. 그러나 를 활성화하려면 전송 중 암호화를 활성화해야 합니다AUTH. 암호화 및 인증에 대한 ElastiCache 자세한 내용은 섹션을 참조하세요ElastiCache 전송 중 암호화(TLS).

> **Note**
>
> valkey-cli와 `--tls` 함께 옵션을 사용하여 클러스터 모드 활성화 및 비활성화된 암호화된 클러스터 모두에 연결할 수 있습니다. 클러스터에 AUTH 토큰 세트가 있는 경우 옵션을 사용하여 AUTH 암호를 `-a` 제공할 수 있습니다.

다음 예제에서는 를 교체해야 합니다.*cluster-endpoint* 그리고 *port number* 클러스터의 엔드포인트와 포트 번호를 사용합니다. (Redis의 기본 포트OSS는 6379입니다.)

클러스터 모드가 비활성화된 암호화된 클러스터에 연결

다음은 암호화 및 인증이 활성화된 클러스터에 연결하는 예제입니다.

```
src/valkey-cli -h cluster-endpoint --tls -a your-password -p port number
```

다음은 암호화만 활성화된 클러스터에 연결하는 예제입니다.

```
src/valkey-cli -h cluster-endpoint --tls -p port number
```

클러스터 모드가 활성화된 암호화된 클러스터에 연결

다음은 암호화 및 인증이 활성화된 클러스터에 연결하는 예제입니다.

```
src/valkey-cli -c -h cluster-endpoint --tls -a your-password -p port number
```

다음은 암호화만 활성화된 클러스터에 연결하는 예제입니다.

```
src/valkey-cli -c -h cluster-endpoint --tls -p port number
```

클러스터에 연결한 후 암호화되지 않은 클러스터에 대해 앞의 예제와 같이 Valkey 또는 Redis OSS 명령을 실행할 수 있습니다.

valkey-cli 또는 Redis-cli의 대안

클러스터 모드가 활성화되어 있지 않고 짧은 테스트를 위해 클러스터에 연결해야 하지만 valkey-cli 또는 redis-cli 컴파일을 거치지 않는 경우 텔넷 또는 openssl을 사용할 수 있습니다. 다음 예제 명령에서 *cluster-endpoint* 그리고 *port number* 클러스터의 엔드포인트와 포트 번호를 사용합니다. (Redis의 기본 포트OSS는 6379입니다.)

다음은 암호화 및/또는 인증이 활성화되어 있으며 클러스터 모드가 비활성화된 클러스터에 연결하는 예제입니다.

```
openssl s_client -connect cluster-endpoint:port number
```

클러스터에 암호가 설정되어 있으면 먼저 클러스터에 연결합니다. 연결 후 다음 명령을 사용하여 클러스터를 인증한 다음 Enter 키를 누릅니다. 다음 예에서는 를 바꿉니다.*your-password* 클러스터의 암호와 함께 사용합니다.

```
Auth your-password
```

다음은 암호화 또는 인증이 활성화되어 있지 않으며 클러스터 모드가 비활성화된 클러스터에 연결하는 예제입니다.

```
telnet cluster-endpoint port number
```

Valkey 또는 Redis OSS 클러스터 또는 복제 그룹에 연결(Windows)

Valkey 또는 Redis OSS 를 사용하여 EC2 Windows 인스턴스에서 클러스터에 연결하려면 valkey-cli 패키지를 다운로드하고 valkey-cli.exe를 사용하여 EC2 Windows 인스턴스에서 Valkey 또는 Redis OSS 클러스터에 연결CLI해야 합니다.

다음 예제에서는 valkey-cli 유틸리티를 사용하여 암호화가 활성화되지 않고 Valkey 또는 Redis 를 실행하는 클러스터에 연결합니다OSS. Valkey 및 사용 가능한 명령에 대한 자세한 내용은 Valkey 웹 사이트의 Valkey 명령을 참조하세요.

valkey-cli를 사용하여 암호화가 활성화되지 않은 Valkey 또는 Redis OSS 클러스터에 연결하려면

1. 원하는 연결 유틸리티를 사용하여 Amazon EC2 인스턴스에 연결합니다. Amazon EC2 인스턴스에 연결하는 방법에 대한 지침은 Amazon EC2 시작 안내서를 참조하세요.

2. 인터넷 브라우저https://github.com/microsoftarchive/redis/releases/download/win-3.0.504/Redis-x64-3.0.504.zip에 링크를 복사하여 붙여넣어 의 사용 가능한 릴리스에서 Redis OSS 클라이언트의 zip 파일을 다운로드합니다. GitHub https://github.com/microsoftarchive/redis/releases/tag/win-3.0.504

   원하는 폴더/경로에 zip 파일의 압축을 풉니다.

   명령 프롬프트를 열고 Valkey 디렉터리로 변경하고 명령을 실행합니다`c:\Valkey>valkey-cli -h Valkey_Cluster_Endpoint -p 6379`.

   예:

```
c:\Valkey>valkey-cli -h cmd.xxxxxxx.ng.0001.usw2.cache.amazonaws.com -p 6379
```

3. Valkey 또는 Redis OSS 명령을 실행합니다.

   이제 클러스터에 연결되어 다음과 같이 Valkey 또는 Redis OSS 명령을 실행할 수 있습니다.

```
set a "hello"          // Set key "a" with a string value and no expiration
OK
get a                  // Get value for key "a"
"hello"
get b                  // Get value for key "b" results in miss
(nil)
set b "Good-bye" EX 5  // Set key "b" with a string value and a 5 second expiration
"Good-bye"
get b                  // Get value for key "b"
"Good-bye"
                       // wait >= 5 seconds
get b
(nil)                  // key has expired, nothing returned
quit                   // Exit from valkey-cli
```

클러스터 삭제

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

> **Warning**
>
> - ElastiCache 클러스터를 삭제하면 수동 스냅샷이 유지됩니다. 클러스터가 삭제되기 전에 최종 스냅샷을 생성할 수도 있습니다. 자동 캐시 스냅샷은 보존되지 않습니다. 자세한 내용은 스냅샷 및 복원 단원을 참조하십시오.
> - `CreateSnapshot` 최종 스냅샷을 생성하려면 권한이 필요합니다. 이 권한이 없으면 `Access Denied` 예외를 제외하고 API 호출이 실패합니다.

사용 AWS Management Console

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.

클러스터를 삭제하려면

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. ElastiCache 엔진 대시보드에서 Valkey 또는 Redis 를 선택합니다OSS.

   해당 엔진에서 실행되는 모든 캐시 목록이 나타납니다.
3. 삭제할 클러스터를 선택하려면 클러스터 목록에서 해당 클러스터의 이름을 선택합니다. 이 경우 2단계: 클러스터 생성에서 생성된 클러스터의 이름입니다.

   > **Important**
   >
   > ElastiCache 콘솔에서 한 번에 하나의 클러스터만 삭제할 수 있습니다. 여러 클러스터를 선택하면 삭제 작업이 비활성화됩니다.
4. 작업에 대해 삭제를 선택합니다.
5. 클러스터 삭제 확인 화면에서 클러스터 이름을 입력하고 최종 백업을 선택합니다. 그런 다음 삭제를 선택하여 클러스터를 삭제하거나 취소를 선택하여 클러스터를 유지합니다.

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

클러스터가 클러스터 목록에서 제거되는 즉시 요금 부과가 중단됩니다.

사용 AWS CLI

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

```
aws elasticache delete-cache-cluster --cache-cluster-id my-cluster
```

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

Linux, macOS, Unix의 경우:

```
aws elasticache delete-cache-cluster \
    --cache-cluster-id my-cluster \
    --region us-east-2
```

Windows의 경우:

```
aws elasticache delete-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --region us-east-2
```

자세한 내용은 ElastiCache 주제 의 섹션을 참조 AWS CLI 하세요`delete-cache-cluster`.

기타 ElastiCache 자습서 및 동영상

다음 자습서에서는 Amazon ElastiCache 사용자가 관심 있는 작업을 다룹니다.

- ElastiCache 비디오
- 자습서: Amazon ElastiCache 에서 Amazon에 액세스하도록 Lambda 함수 구성 VPC

ElastiCache 비디오

다음은 기본 및 고급 Amazon ElastiCache 개념을 배우는 데 도움이 되는 비디오입니다. AWS 훈련에 대한 자세한 내용은 AWS 훈련 및 인증을 참조하세요.

주제

- 입문자용 동영상
- 고급 동영상

입문자용 동영상

다음 동영상에서는 Amazon 를 소개합니다 ElastiCache.

주제

- AWS re:Invent 2020: Amazon의 새로운 기능 ElastiCache
- AWS re:Invent 2019: Amazon의 새로운 기능 ElastiCache
- AWS re:Invent 2017: Amazon의 새로운 기능 ElastiCache
- DAT204 - 서비스 AWS 없는SQL 확장 가능한 애플리케이션 구축(re:Invent 2015)
- DAT207 - Amazon을 통한 애플리케이션 성능 가속화 ElastiCache (AWS re:Invent 2013)

AWS re:Invent 2020: Amazon의 새로운 기능 ElastiCache

AWS re:Invent 2020: Amazon의 새로운 기능 ElastiCache

AWS re:Invent 2019: Amazon의 새로운 기능 ElastiCache

AWS re:Invent 2019: Amazon의 새로운 기능 ElastiCache

AWS re:Invent 2017: Amazon의 새로운 기능 ElastiCache

AWS re:Invent 2017: Amazon의 새로운 기능 ElastiCache

DAT204 - 서비스 AWS 없는SQL 확장 가능한 애플리케이션 구축(re:Invent 2015)

이 세션에서는 데이터베이스 없음SQL의 이점에 대해 논의하고 AWS—Amazon DynamoDB 및 Amazon 에서 제공하는 주요 서비스 없음SQL에 대해 살펴봅니다 ElastiCache. 그런 다음 Expedia와 Mapbox라는 두 주요 고객으로부터 사용 사례와 아키텍처 문제, 설계 패턴 및 모범 사례를 포함하여 서비스 AWS 없음을SQL 사용하여 문제를 해결한 방법에 대해 들어봅니다. NoSQL와 강력한 기능을 더 잘 이해하여 데이터베이스 문제를 자신 있게 해결할 준비가 된 상태로 이 세션에서 나가야 합니다.

DAT204 - 서비스 AWS 없는SQL 확장 가능한 애플리케이션 구축(re:Invent 2015)

DAT207 - Amazon을 통한 애플리케이션 성능 가속화 ElastiCache (AWS re:Invent 2013)

이 동영상에서는 Amazon ElastiCache 을 사용하여 인 메모리 캐싱 시스템을 쉽게 배포하여 애플리케이션 성능을 높이는 방법을 알아봅니다. Amazon을 사용하여 애플리케이션 지연 시간을 ElastiCache 개선하고 데이터베이스 서버의 부하를 줄이는 방법을 보여줍니다. 또한 애플리케이션이 성장함에 따라 쉽게 관리 및 조정할 수 있는 캐싱 레이어를 구축하는 방법도 알아봅니다. 이 세션에서는 캐싱을 활성화하여 이점을 얻을 수 있는 다양한 시나리오와 사용 사례를 살펴보고 Amazon 에서 제공하는 기능에 대해 설명합니다 ElastiCache.

DAT207 - Amazon을 통한 애플리케이션 성능 가속화 ElastiCache (re:Invent 2013)

고급 동영상

다음 동영상에서는 고급 Amazon ElastiCache 주제를 다룹니다.

주제

- Amazon ElastiCache 모범 사례의 성공을 위한 설계(re:Invent 2020)
- Amazon ElastiCache (re:Invent 2019)을 사용하여 실시간 앱 강화
- 모범 사례: Amazon에서 ElastiCache (re:Invent 2019)EC2로 Redis OSS 클러스터 마이그레이션
- Amazon ElastiCache 및 Amazon Aurora를 사용하여 Fantasy Sports 플랫폼 확장STP11(re:Invent 2018)
- Amazon을 통한 클라우드OSS의 안정적이고 확장 가능한 Redis ElastiCache (re:Invent 2018)
- ElastiCache Deep Dive: 인 메모리 데이터 스토어의 설계 패턴(re:Invent 2018)
- DAT305—Amazon ElastiCache Deep Dive(re:Invent 2017)
- DAT306 - Amazon ElastiCache Deep Dive(re:Invent 2016)
- DAT317- ElastiCache (Redis OSS)를 IFTTT 사용하여 이벤트를 예측하는 방법(re:Invent 2016)
- DAT407—Amazon ElastiCache Deep Dive(re:Invent 2015)
- SDD402 - Amazon ElastiCache Deep Dive(re:Invent 2014)
- DAT307 - Amazon ElastiCache 아키텍처 및 설계 패턴 심층 분석(re:Invent 2013)

Amazon ElastiCache 모범 사례의 성공을 위한 설계(re:Invent 2020)

Redis 기반 비즈니스 크리티컬 실시간 애플리케이션이 폭발적으로 증가함에 따라 가용성OSS, 확장성 및 보안이 가장 중요한 고려 사항이 되었습니다. 온라인 확장, 다중 AZ 배포 전반의 고가용성 및 보안 구성을 통해 성공을 ElastiCache 위해 Amazon을 설정하는 모범 사례를 알아봅니다.

Amazon ElastiCache 모범 사례의 성공을 위한 설계(re:Invent 2020)

Amazon ElastiCache (re:Invent 2019)을 사용하여 실시간 앱 강화

클라우드 도입과 이에 기반한 새로운 시나리오가 급속히 증가함에 따라 애플리케이션에는 초당 수백만 건의 요청을 지원하기 위해 마이크로초 단위의 지연 시간과 높은 처리량이 필요합니다. 개발자들은 전통적으로 데이터 감소 기술이 결합된 디스크 기반 데이터베이스와 같은 특수 하드웨어 및 해결 방법을 사용하여 실시간 애플리케이션을 위한 데이터를 관리해 왔습니다. 그러나 이러한 접근 방식은 비용이 많이 들고 확장성이 떨어질 수 있습니다. 최고의 성능, 높은 확장성, 가용성 및 보안을 ElastiCache 위해 완전 관리형 인메모리 Amazon을 사용하여 실시간 애플리케이션의 성능을 향상시킬 수 있는 방법을 알아봅니다.

Amazon ElastiCache (re:Invent 2019:)으로 실시간 앱을 강화하세요.

모범 사례: Amazon에서 ElastiCache (re:Invent 2019)EC2로 Redis OSS 클러스터 마이그레이션

Redis OSS 클러스터를 직접 관리하는 것은 어려울 수 있습니다. 하드웨어를 프로비저닝하고, 소프트웨어를 패치하고, 데이터를 백업하고, 워크로드를 지속적으로 모니터링해야 합니다. Amazon 에 대해 새로 출시된 온라인 마이그레이션 기능을 사용하면 클러스터 모드가 비활성화된 상태에서 AmazonOSS의 자체 호스팅 Redis에서 EC2 완전 관리형 Amazon ElastiCache로 데이터를 쉽게 이동할 ElastiCache수 있습니다. 이 세션에서는 새로운 온라인 마이그레이션 도구에 대해 알아보고, 데모를 보고, 더 중요한 것은 Amazon 로 원활하게 마이그레이션하기 위한 실습 모범 사례를 배우는 것입니다 ElastiCache.

모범 사례: Amazon에서 ElastiCache (re:Invent 2019)EC2로 Redis OSS 클러스터 마이그레이션

Amazon ElastiCache 및 Amazon Aurora를 사용하여 Fantasy Sports 플랫폼 확장STP11(re:Invent 2018)

Dream11은 인도의 선도적인 스포츠 기술 스타트업입니다. 판타지 크리켓, 축구, 농구 등 여러 스포츠를 즐기는 4천만 명 이상의 사용자 기반이 계속 증가하고 있으며 현재 백만 명의 동시 사용자에게 서비스를 제공하며, 이러한 사용자들이 50밀리초 미만의 응답 시간으로 분당 3백만 건의 요청을 생성하고 있습니다. 이 대화에서 Dream11 CTO Amit Sharma는 회사가 Amazon Aurora와 Amazon ElastiCache 을 사용하여 플래시 트래픽을 처리하는 방법을 설명합니다. 이 트래픽은 30초의 응답 기간 내에 3배로 늘어날 수 있습니다. 또한 Sharma는 잠금이 필요하지 않은 트랜잭션 확장/축소 방법에 대해 이야기하며 매일 5백만 명의 활성 사용자에게 서비스를 제공하고 있는 플래시 트래픽 처리 단계를 공유합니다. 전체 제목: AWS re:Invent 2018: Amazon ElastiCache 및 Amazon Aurora를 사용하여 Fantasy Sports 플랫폼 확장(STP11)

Amazon ElastiCache 및 Amazon Aurora를 사용하여 Fantasy Sports 플랫폼 확장STP11(re:Invent 2018)

Amazon을 통한 클라우드OSS의 안정적이고 확장 가능한 Redis ElastiCache (re:Invent 2018)

이 세션에서는 Redis OSS호환 서비스인 Amazon ElastiCache (Redis)의 기능과 개선 사항을 다룹니다 OSS. Redis OSS 5, 확장성 및 성능 개선, 보안 및 규정 준수 등과 같은 주요 기능을 다룹니다. 또한 예정된 기능 및 고객 사례 연구에 대해서도 논의합니다.

Amazon을 통한 클라우드OSS의 안정적이고 확장 가능한 Redis ElastiCache (re:Invent 2018)

ElastiCache Deep Dive: 인 메모리 데이터 스토어의 설계 패턴(re:Invent 2018)

이 세션에서는 Amazon 의 설계 및 아키텍처에 대해 알아볼 수 있는 이면의 을 제공합니다 ElastiCache. Redis OSS 및 Memcached 오퍼링의 일반적인 설계 패턴과 고객이 인 메모리 데이터 처리에 이를 사용하여 지연 시간을 줄이고 애플리케이션 처리량을 개선하는 방법을 확인하세요. ElastiCache 모범 사례, 설계 패턴 및 패턴 방지를 검토합니다.

ElastiCache Deep Dive: 인 메모리 데이터 스토어의 설계 패턴(re:Invent 2018)

DAT305—Amazon ElastiCache Deep Dive(re:Invent 2017)

Amazon 의 ElastiCache설계 및 아키텍처에 대해 알아보려면 뒤를 쳐다 보세요. Memcached 및 Redis OSS 오퍼링을 통해 일반적인 설계 패턴과 고객이 인메모리 작업에 이를 사용하여 지연 시간을 줄이고 애플리케이션 처리량을 개선하는 방법을 확인하세요. 이 동영상에서는 ElastiCache 모범 사례, 설계 패턴 및 패턴 방지를 검토합니다.

비디오는 다음의 내용을 소개합니다.

- ElastiCache (Redis OSS) 온라인 리샤딩
- ElastiCache 보안 및 암호화
- ElastiCache (Redis OSS) 버전 3.2.10

DAT305 - Amazon ElastiCache Deep Dive(re:Invent 2017)

DAT306 - Amazon ElastiCache Deep Dive(re:Invent 2016)

Amazon 의 ElastiCache설계 및 아키텍처에 대해 알아보려면 뒤를 쳐다 보세요. Memcached 및 Redis OSS 오퍼링을 통해 일반적인 설계 패턴과 고객이 인메모리 작업에 이를 사용하여 지연 시간을 줄이고 애플리케이션 처리량을 개선하는 방법을 확인하세요. 이 세션에서는 ElastiCache 모범 사례, 설계 패턴 및 패턴 방지를 검토합니다.

DAT306—Amazon ElastiCache Deep Dive(re:Invent 2016)

DAT317- ElastiCache (Redis OSS)를 IFTTT 사용하여 이벤트를 예측하는 방법(re:Invent 2016)

IFTTT 는 간단한 작업을 자동화하는 것부터 누군가 집과 상호 작용하고 제어하는 방식을 바꾸는 것까지 사람들이 좋아하는 서비스를 더 많이 수행할 수 있는 무료 서비스입니다. IFTTT 는 ElastiCache (Redis OSS)를 사용하여 트랜잭션 실행 기록을 저장하고 예측을 예약하며 Amazon S3의 로그 문서에 대한 인덱스도 저장합니다. 이 세션을 보고 Lua의 스크립팅 성능과 Redis의 데이터 유형을 OSS 통해 사람들이 다른 곳에서는 할 수 없었던 일을 어떻게 달성할 수 있었는지 알아보세요.

DAT317- ElastiCache (Redis OSS)를 IFTTT 사용하여 이벤트를 예측하는 방법(re:Invent 2016)

DAT407—Amazon ElastiCache Deep Dive(re:Invent 2015)

Amazon 의 설계 및 아키텍처에 대해 알아보려면 보이지 않는 장면 ElastiCache을 살펴보세요. Memcached 및 Redis OSS 제품의 일반적인 설계 패턴과 고객이 인메모리 작업에 Memcached 및 Redis를 사용하고 애플리케이션의 지연 시간과 처리량을 개선한 방법을 확인하세요. 이 세션에서는 Amazon 와 관련된 모범 사례, 설계 패턴 및 안티 패턴을 검토합니다 ElastiCache.

DAT407 - Amazon ElastiCache Deep Dive(re:Invent 2015)

SDD402 - Amazon ElastiCache Deep Dive(re:Invent 2014)

이 동영상에서는 일반적인 캐싱 사용 사례, Memcached 및 Redis OSS 엔진, 요구 사항에 더 적합한 엔진을 결정하는 데 도움이 되는 패턴, 일관된 해싱 등을 빠르고 확장 가능한 애플리케이션을 구축하는 수단으로 살펴봅니다. Adobe의 수석 과학자인 Frank Wiebe는 Adobe가 Amazon ElastiCache 을 사용하여 고객 경험을 개선하고 비즈니스를 확장하는 방법을 자세히 설명합니다.

DAT402 - Amazon ElastiCache Deep Dive(re:Invent 2014)

DAT307 - Amazon ElastiCache 아키텍처 및 설계 패턴 심층 분석(re:Invent 2013)

이 동영상에서는 캐싱, 캐싱 전략, 확장, 모니터링을 살펴봅니다. 또한 Memcached 엔진과 Redis OSS 엔진을 비교합니다. 이 세션에서는 Amazon 와 관련된 모범 사례 및 설계 패턴도 검토합니다 ElastiCache.

DAT307 - Amazon ElastiCache 아키텍처 및 설계 패턴에 대한 심층 분석(AWS re:Invent 2013)

에서 노드 관리 ElastiCache

노드는 Amazon ElastiCache 배포의 가장 작은 구성 요소입니다. 안전한 네트워크 연결 의 고정 크기 청크입니다RAM. 각 노드는 클러스터가 생성되거나 마지막으로 수정되었을 때 선택한 엔진을 실행합니다. 각 노드에는 고유한 도메인 이름 서비스(DNS) 이름과 포트가 있습니다. 여러 유형의 ElastiCache 노드가 지원되며, 각 노드는 서로 다른 양의 관련 메모리와 컴퓨팅 성능을 갖습니다.

사용할 노드 크기에 대한 자세한 내용은 노드 크기 선택 섹션을 참조하세요.

일반적으로 샤딩 지원으로 인해 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 배포에는 더 작은 노드가 여러 개 있습니다. 반면 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 배포에는 클러스터에 더 적고 더 큰 노드가 있습니다. 사용할 노드 크기에 대한 자세한 내용은 노드 크기 선택 섹션을 참조하세요.

주제

- ElastiCache 노드 상태 보기
- Valkey 또는 Redis OSS 노드 및 샤드
- 노드에 연결
- 지원되는 노드 유형
- 노드 재부팅
- 노드 교체(Valkey 및 RedisOSS)
- 노드 교체(Memcached)
- 예약 노드
- 이전 세대 노드 마이그레이션

노드와 관련된 몇 가지 중요한 작업은 다음과 같습니다.

- ElastiCache 클러스터에 노드 추가
- ElastiCache 클러스터에서 노드 제거
- 크기 조정 ElastiCache
- 에서 연결 엔드포인트 찾기 ElastiCache
- 클러스터의 노드 자동 식별(Memcached)

ElastiCache 노드 상태 보기

ElastiCache 콘솔 을 사용하면 ElastiCache 노드의 상태에 빠르게 액세스할 수 있습니다. ElastiCache 노드의 상태는 노드의 상태를 나타냅니다. 다음 절차를 사용하여 Amazon ElastiCache 콘솔, AWS CLI 명령 또는 API 작업에서 ElastiCache 노드 상태를 볼 수 있습니다.

ElastiCache 노드의 가능한 상태 값은 다음 표에 나와 있습니다. 이 표는 ElastiCache 노드에 대한 요금이 청구되는지 여부도 보여줍니다.

유형	청구	설명
available	청구	ElastiCache 노드가 정상이고 사용 가능합니다.
creating	미청구	ElastiCache 노드가 생성 중입니다. 생성 중인 노드에는 액세스할 수 없습니다.
deleting	미청구	ElastiCache 노드가 삭제되고 있습니다.
modifying	청구	ElastiCache 노드를 수정하라는 고객 요청으로 인해 노드가 수정되고 있습니다.
updating	청구	업데이트 상태는 다음 중 하나 이상이 Amazon ElastiCache 노드에 해당함을 나타냅니다. • ElastiCache 노드가 서비스 udpate의 일부로 패치되고 있습니다. 서비스 업데이트에 대한 자세한 내용은 Amazon ElastiCache Managed Maintenance and Service Updates 도움말 페이지 섹션을 참조하세요. • ElastiCache 클러스터에 대한 VPC 보안 그룹이 업데이트 중입니다. • ElastiCache 클러스터가 스케일 업 또는 스케일 다운되고 있습니다. • ElastiCache 클러스터에 대한 로그 전송 구성이 수정되고 있습니다.

유형	청구	설명
		• ElastiCache 노드에 대한 삭제 작업이 보류 중입니다. • Valkey 또는 Redis OSS 암호가 ElastiCache 포함된 가를 사용하여 업데이트/회전되고 있습니다AWS Secrets Manager.
`rebooting cache cluster nodes`	청구	고객 요청 또는 ElastiCache 노드 재부팅이 필요한 Amazon ElastiCache 프로세스로 인해 노드가 재부팅되고 있습니다.
`incompatible_parameters`	미청구	노드의 파라미터 그룹에 지정된 파라미터가 노드와 호환되지 않으므로 Amazon은 노드를 시작할 ElastiCache 수 없습니다. 노드에 대한 액세스 권한을 다시 얻으려면 파라미터 변경 사항을 되돌리거나 노드와 호환되는 파라미터를 정의해야 합니다. 호환되지 않는 파라미터에 대한 자세한 내용은 ElastiCache 노드의 이벤트 목록을 확인하세요.

유형	청구	설명
`incompatible_network`	미청구	호환되지 않는 네트워크 상태는 다음 중 하나 이상이 Amazon ElastiCache 노드에 해당함을 나타냅니다. • 노드가 ElastiCache 시작된 서브넷에 사용 가능한 IP 주소가 없습니다. • 서브넷 그룹에 언급된 ElastiCache 서브넷은 Amazon Virtual Private Cloud(Amazon)에 더 이상 존재하지 않습니다VPC.

유형	청구	설명
`restore_failed`	미청구	복원 실패 상태는 다음 중 하나가 Amazon ElastiCache 노드에 해당함을 나타냅니다. • 인스턴스 용량 부족이 반복되어 노드 교체가 실패했습니다. 이는 일반적으로 인 이전 세대 노드를 실행할 때 발생합니다 end-of-life. 그러나 가 지정된 가용 영역에서 요청을 이행할 만큼 충분한 온디맨드 용량이 AWS 없는 경우 현재 세대 노드를 교체할 때도 발생할 수 있습니다. 이러한 노드를 수정하거나 제거하는 방법에 대한 자세한 내용은 섹션을 참조하세요이전 세대 노드 마이그레이션. • 지정된 RDB 스냅샷을 복원하지 못했습니다. • ElastiCache 클러스터의 AWS 계정이 일시 중지되었습니다. • 노드가 실패하여 복구할 수 없습니다.
`snapshotting`	청구	ElastiCache 는 Valkey 또는 Redis OSS 노드의 스냅샷을 생성합니다.

콘솔을 사용하여 ElastiCache 노드 상태 보기

콘솔을 사용하여 ElastiCache 노드의 상태를 보려면:

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 탐색 창에서 Redis OSS 클러스터 또는 Memcached 클러스터를 선택합니다. 캐시 페이지가 ElastiCache 노드 목록과 함께 나타납니다. 각 노드에 대한 상태 값이 표시됩니다.
3. 그런 다음 캐시의 서비스 업데이트 탭으로 이동하여 캐시에 적용되는 서비스 업데이트 목록을 표시할 수 있습니다.

를 사용하여 ElastiCache 노드 상태 보기 AWS CLI

를 사용하여 ElastiCache 노드 및 상태 정보를 보려면 `describe-cache-cluster` 명령을 AWS CLI 사용합니다. 예를 들어 다음 AWS CLI 명령은 각 ElastiCache 노드를 표시합니다.

```
aws elasticache describe-cache-clusters
```

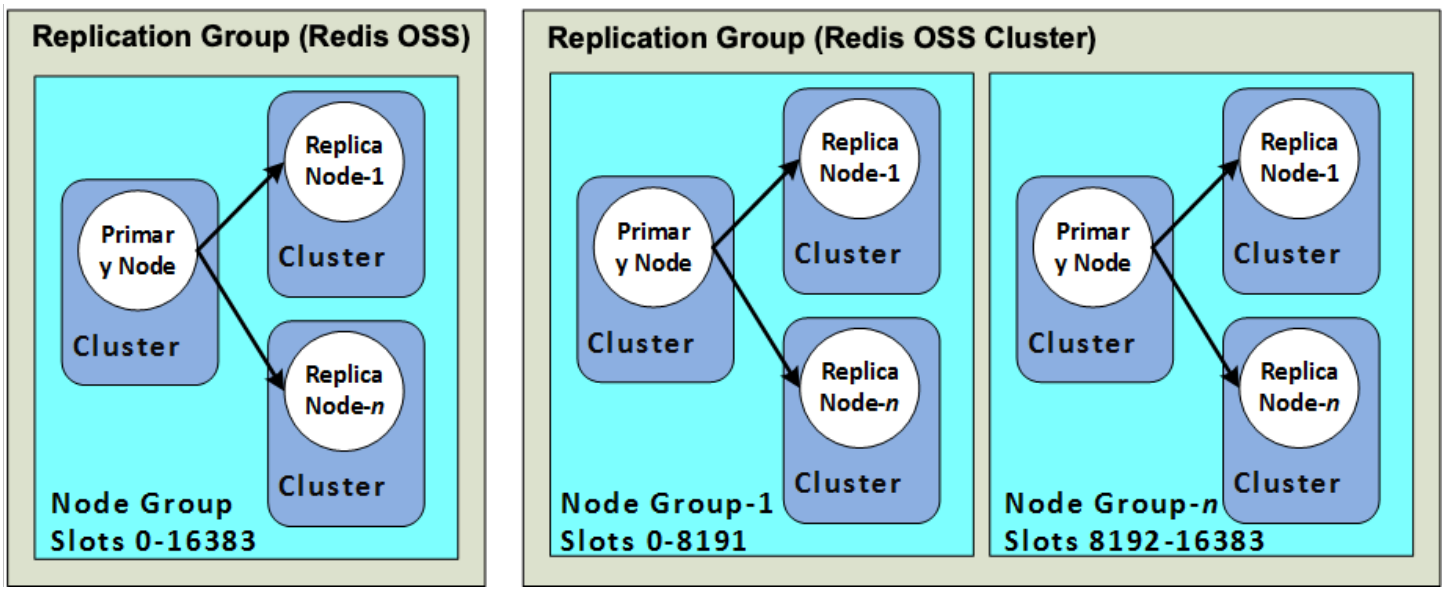
를 통해 ElastiCache 노드 상태 보기 API

Amazon 를 사용하여 ElastiCache 노드의 상태를 보려면 `ShowCacheNodeInfo` 플래그가 `DescribeCacheClusteroperation` 있는 를 ElastiCache API호출하여 개별 캐시 노드에 대한 정보를 검색합니다.

Valkey 또는 Redis OSS 노드 및 샤드

샤드(API 및 에서 CLI노드 그룹)는 각각 클러스터로 래핑되는 노드의 계층적 배열입니다. 샤드는 복제를 지원합니다. 샤드에서 노드 하나가 읽기/쓰기 기본 노드로 사용됩니다. 샤드에 있는 다른 모든 노드는 기본 노드의 읽기 전용 복제본 역할을 합니다. Valkey 또는 Redis OSS 버전 3.2 이상은 클러스터 내에서 여러 샤드를 지원합니다(API 및 에서는 CLI복제 그룹). 이 지원을 사용하면 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터에서 데이터를 파티셔닝할 수 있습니다.

다음 다이어그램은 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터와 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 간의 차이점을 보여줍니다.



Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터는 샤드를 통한 복제를 지원합니다. API 작업DescribeReplicationGroups(CLI: describe-replication-groups)에는 멤버 노드가 있는 노드 그룹, 노드 그룹 내 노드의 역할 및 기타 정보가 나열됩니다.

Valkey 또는 Redis OSS 클러스터를 생성할 때 클러스터링이 활성화된 클러스터를 생성할지 여부를 지정합니다. Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터에는 샤드가 두 개 이상 있어서는 안 됩니다. 이 샤드는 읽기 전용 복제본 노드를 추가(최대 총 5개)하거나 삭제하여 수평으로 확장할 수 있습니다. 자세한 내용은 고가용성을 위한 복제 그룹 사용, Valkey 또는 Redis에 대한 읽기 전용 복제본 추가OSS(클러스터 모드 비활성화됨) 또는 Valkey 또는 Redis에 대한 읽기 전용 복제본 삭제OSS(클러스터 모드 비활성화됨) 섹션을 참조하세요. Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터는 노드 유형을 변경하여 수직으로 확장할 수도 있습니다. 자세한 내용은 Valkey 또는 Redis용 복제본 노드 크기 조정OSS(클러스터 모드 비활성화됨) 단원을 참조하십시오.

엔진이 Valkey 또는 Redis OSS 버전 5.0.6 이상인 경우 노드 또는 샤드 제한을 클러스터당 최대 500개로 늘릴 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. 일반적인 위험에는 서브넷 그룹의 서브넷 CIDR 범위가 너무 작거나 서브넷이 공유되어 다른 클러스터에서 많이 사용되는 경우가 포함됩니다. 자세한 내용은 서브넷 그룹 생성 단원을 참조하십시오.

5.0.6 이하의 버전에서 한도는 클러스터당 250개입니다.

한도 증가를 요청하려면 AWS 서비스 한도를 참조하고 한도 유형을 인스턴스 유형별 클러스터당 노드로 선택하세요.

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터가 생성된 후에는 클러스터를 변경(스케일 인 또는 아웃)할 수 있습니다. 자세한 내용은 크기 조정 ElastiCache 및 노드 교체(Valkey 및 RedisOSS) 단원을 참조하세요.

새 클러스터를 생성할 때 빈 상태로 시작되지 않도록 이전 클러스터의 데이터를 시드할 수 있습니다. 이는 클러스터 그룹에 이전 클러스터와 동일한 수의 샤드가 있는 경우에만 사용할 수 있는 방식입니다. 이렇게 하면 노드 유형이나 엔진 버전을 변경해야 하는 경우 도움이 됩니다. 자세한 정보는 수동 백업 지원 및 백업에서 새 캐시로 복원 섹션을 참조하세요.

노드에 연결

Valkey 또는 Redis OSS 노드에 연결

클러스터의 Valkey 또는 Redis OSS 노드에 연결을 시도하기 전에 노드에 대한 엔드포인트가 있어야 합니다. 엔드포인트를 찾으려면 다음을 참조하세요.

- Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터의 엔드포인트 찾기(콘솔)
- Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터(콘솔)의 엔드포인트 찾기
- 엔드포인트 찾기(AWS CLI)
- 엔드포인트 찾기(ElastiCache API)

다음 예제에서는 valkey-cli 유틸리티를 사용하여 Valkey 또는 Redis 를 실행하는 클러스터에 연결합니다OSS.

> **Note**
>
> 사용 가능한 명령에 대한 자세한 내용은 명령 웹 페이지를 참조하세요.

valkey-cli를 사용하여 Valkey 또는 Redis OSS 클러스터에 연결하려면

1. 원하는 연결 유틸리티를 사용하여 Amazon EC2 인스턴스에 연결합니다.

   > **Note**
   >
   > Amazon EC2 인스턴스에 연결하는 방법에 대한 지침은 Amazon EC2 시작 안내서를 참조하세요.

2. 를 빌드하려면 GNU 컴파일러 컬렉션()을 `valkey-cli`다운로드하여 설치합니다`gcc`. EC2 인스턴스의 명령 프롬프트에서 다음 명령을 입력하고 확인 프롬프트`y`에 를 입력합니다.

   ```
   sudo yum install gcc
   ```

   다음과 같이 유사한 출력이 나타납니다.

   ```
   Loaded plugins: priorities, security, update-motd, upgrade-helper
   ```

```
Setting up Install Process
Resolving Dependencies
--> Running transaction check

...(output omitted)...

Total download size: 27 M
Installed size: 53 M
Is this ok [y/N]: y
Downloading Packages:
(1/11): binutils-2.22.52.0.1-10.36.amzn1.x86_64.rpm      | 5.2 MB     00:00
(2/11): cpp46-4.6.3-2.67.amzn1.x86_64.rpm                | 4.8 MB     00:00
(3/11): gcc-4.6.3-3.10.amzn1.noarch.rpm                  | 2.8 kB     00:00

...(output omitted)...

Complete!
```

3. valkey-cli 유틸리티를 다운로드하고 컴파일합니다. 이 유틸리티는 Valkey 소프트웨어 배포에 포함됩니다. EC2 인스턴스의 명령 프롬프트에 다음 명령을 입력합니다.

> **Note**
>
> Ubuntu 시스템의 경우 `make`를 실행하기 전에 `make distclean`을 실행합니다.

```
wget https://github.com/valkey-io/valkey/archive/refs/tags/7.2.6.tar.gz
tar xvzf valkey-7.2.6.tar.gz
cd valkey-7.2.6
make distclean      # ubuntu systems only
make
```

4. EC2 인스턴스의 명령 프롬프트에 다음 명령을 입력합니다.

```
src/valkey-cli -c -h mycachecluster.eaogs8.0001.usw2.cache.amazonaws.com -p 6379
```

다음과 유사한 Valkey 또는 Redis OSS 명령 프롬프트가 나타납니다.

```
redis mycachecluster.eaogs8.0001.usw2.cache.amazonaws.com 6379>
```

5. Valkey 또는 Redis OSS 명령을 실행하여 연결을 테스트합니다.

   이제 클러스터에 연결되어 Valkey 또는 Redis OSS 명령을 실행할 수 있습니다. 다음은 Valkey 또는 Redis OSS 응답이 있는 명령의 몇 가지 예입니다.

```
set a "hello"          // Set key "a" with a string value and no expiration
OK
get a                  // Get value for key "a"
"hello"
get b                  // Get value for key "b" results in miss
(nil)
set b "Good-bye" EX 5  // Set key "b" with a string value and a 5 second expiration
get b
"Good-bye"
                    // wait 5 seconds
get b
(nil)                  // key has expired, nothing returned
quit                   // Exit from valkey-cli
```

Secure Sockets Layer(SSL) 암호화(운송 중 활성화)가 있는 노드 또는 클러스터에 연결하는 방법은 섹션을 참조하세요ElastiCache 전송 중 암호화(TLS).

Memcached 노드에 연결

Memcached 클러스터에 연결하기 전에 노드의 엔드포인트가 있어야 합니다. 엔드포인트를 찾으려면 다음을 참조하세요.

- 클러스터의 엔드포인트 찾기(콘솔)(Memcached)
- 엔드포인트 찾기(AWS CLI)
- 엔드포인트 찾기(ElastiCache API)

다음 예제에서 telnet 유틸리티를 사용하여 Memcached를 실행하는 노드에 연결합니다.

> **Note**
>
> Memcached 및 사용 가능한 Memcached 명령에 대한 자세한 내용은 Memcached 웹 사이트를 참조하세요.

telnet을 사용하여 노드에 연결하려면

1. 원하는 연결 유틸리티를 사용하여 Amazon EC2 인스턴스에 연결합니다.

   > **Note**
   >
   > Amazon EC2 인스턴스에 연결하는 방법에 대한 지침은 Amazon EC2 시작 안내서를 참조하세요.

2. Amazon EC2 인스턴스에 텔넷 유틸리티를 다운로드하고 설치합니다. Amazon EC2 인스턴스의 명령 프롬프트에서 다음 명령을 입력하고 명령 프롬프트에 y를 입력합니다.

   ```
   sudo yum install telnet
   ```

   다음과 같이 유사한 출력이 나타납니다.

   ```
   Loaded plugins: priorities, security, update-motd, upgrade-helper
   Setting up Install Process
   Resolving Dependencies
   --> Running transaction check

   ...(output omitted)...

   Total download size: 63 k
   Installed size: 109 k
   Is this ok [y/N]: y
   Downloading Packages:
   telnet-0.17-47.7.amzn1.x86_64.rpm                          |  63 kB     00:00

   ...(output omitted)...

   Complete!
   ```

3. Amazon EC2 인스턴스의 명령 프롬프트에 다음 명령을 입력하여 이 예제에 표시된 노드의 엔드포인트를 대체합니다.

   ```
   telnet mycachecluster.eaogs8.0001.usw2.cache.amazonaws.com 11211
   ```

   다음과 같이 유사한 출력이 나타납니다.

```
Trying 128.0.0.1...
Connected to mycachecluster.eaogs8.0001.usw2.cache.amazonaws.com.
Escape character is '^]'.
>
```

4. Memcached 명령을 실행하여 연결을 테스트합니다.

   이제 노드에 연결되어 Memcached 명령을 실행할 수 있습니다. 다음은 예입니다.

```
set a 0 0 5      // Set key "a" with no expiration and 5 byte value
hello            // Set value as "hello"
STORED
get a            // Get value for key "a"
VALUE a 0 5
hello
END
get b            // Get value for key "b" results in miss
END
>
```

지원되는 노드 유형

ElastiCache 는 다음 노드 유형을 지원합니다. 일반적으로, 현재 세대 유형은 이전 세대의 동급 제품에 비해 더 많은 메모리와 컴퓨팅 파워를 더 저렴하게 제공합니다.

각 노드 유형의 성능 세부 정보에 대한 자세한 내용은 Amazon EC2 인스턴스 유형 섹션을 참조하세요.

사용할 노드 크기에 대한 자세한 내용은 노드 크기 선택 섹션을 참조하세요.

주제

현재 세대(Memcached)

다음 표는 네트워크 I/O 크레딧 메커니즘을 사용하여 기준 대역폭을 초과하여 버스트하는 인스턴스 유형에 대한 기준 및 버스트 대역폭을 보여줍니다.

> **Note**
>
> 버스트 가능 네트워크 성능을 발휘하는 인스턴스 유형은 네트워크 I/O 크레딧 메커니즘을 사용하여 최선의 노력을 기준으로 기준 대역폭을 초과하여 버스트할 수 있습니다.

일반

인스턴스 유형	지원되는 최소 Memcached 버전	기준 대역폭(Gbps)	버스트 대역폭(Gbps)
cache.m7g.large		0.937	12.5
cache.m7g.xlarge		1.876	12.5
cache.m7g.2xlarge		3.75	15

인스턴스 유형	지원되는 최소 Memcached 버전	기준 대역폭(Gbps)	버스트 대역폭(Gbps)
cache.m7g.4xlarge		7.5	15
cache.m7g.8xlarge		15	N/A
cache.m7g.12xlarge		22.5	N/A
cache.m7g.16xlarge		30	N/A
cache.m6g.large	1.5.16	0.75	10.0
cache.m6g.xlarge	1.5.16	1.25	10.0
cache.m6g.2xlarge	1.5.16	2.5	10.0
cache.m6g.4xlarge	1.5.16	5.0	10.0
cache.m6g.8xlarge	1.5.16	12	N/A
cache.m6g.12xlarge	1.5.16	20	N/A
cache.m6g.16xlarge	1.5.16	25	N/A
cache.m5.large	1.5.16	0.75	10.0
cache.m5.xlarge	1.5.16	1.25	10.0
cache.m5.2xlarge	1.5.16	2.5	10.0
cache.m5.4xlarge	1.5.16	5.0	10.0
cache.m5.12xlarge	1.5.16	N/A	N/A
cache.m5.24xlarge	1.5.16	N/A	N/A
cache.m4.large	1.5.16	0.45	1.2
cache.m4.xlarge	1.5.16	0.75	2.8
cache.m4.2xlarge	1.5.16	1.0	10.0

인스턴스 유형	지원되는 최소 Memcached 버전	기준 대역폭(Gbps)	버스트 대역폭(Gbps)
cache.m4.4xlarge	1.5.16	2.0	10.0
cache.m4.10xlarge	1.5.16	5.0	10.0
cache.t4g.micro	1.5.16	0.064	5.0
cache.t4g.small	1.5.16	0.128	5.0
cache.t4g.medium	1.5.16	0.256	5.0
cache.t3.micro	1.5.16	0.064	5.0
cache.t3.small	1.5.16	0.128	5.0
cache.t3.medium	1.5.16	0.256	5.0
cache.t2.micro	1.5.16	0.064	1.024
cache.t2.small	1.5.16	0.128	1.024
cache.t2.medium	1.5.16	0.256	1.024

Memcached에 최적화된 메모리

인스턴스 유형	지원되는 최소 버전	기준 대역폭(Gbps)	버스트 대역폭(Gbps)
cache.r7g.large		0.937	12.5
cache.r7g.xlarge		1.876	12.5
cache.r7g.2xlarge		3.75	15
cache.r7g.4xlarge		7.5	15
cache.r7g.8xlarge		15	N/A
cache.r7g.12xlarge		22.5	N/A

인스턴스 유형	지원되는 최소 버전	기준 대역폭(Gbps)	버스트 대역폭(Gbps)
cache.r7g.16xlarge		30	N/A
cache.r6g.large	1.5.16	0.75	10.0
cache.r6g.xlarge	1.5.16	1.25	10.0
cache.r6g.2xlarge	1.5.16	2.5	10.0
cache.r6g.4xlarge	1.5.16	5.0	10.0
cache.r6g.8xlarge	1.5.16	12	N/A
cache.r6g.12xlarge	1.5.16	20	N/A
cache.r6g.16xlarge	1.5.16	25	N/A
cache.r5.large	1.5.16	0.75	10.0
cache.r5.xlarge	1.5.16	1.25	10.0
cache.r5.2xlarge	1.5.16	2.5	10.0
cache.r5.4xlarge	1.5.16	5.0	10.0
cache.r5.12xlarge	1.5.16	20	N/A
cache.r5.24xlarge	1.5.16	25	N/A
cache.r4.large	1.5.16	0.75	10.0
cache.r4.xlarge	1.5.16	1.25	10.0
cache.r4.2xlarge	1.5.16	2.5	10.0
cache.r4.4xlarge	1.5.16	5.0	10.0
cache.r4.8xlarge	1.5.16	12	N/A
cache.r4.16xlarge	1.5.16	25	N/A

Memcached에 최적화된 네트워크

인스턴스 유형	지원되는 최소 버전	기준 대역폭(Gbps)	버스트 대역폭(Gbps)
cache.c7gn.large	1.6.6	6.25	30
cache.c7gn.xlarge	1.6.6	12.5	40
cache.c7gn.2xlarge	1.6.6	25	50
cache.c7gn.4xlarge	1.6.6	50	N/A
cache.c7gn.8xlarge	1.6.6	100	N/A
cache.c7gn.12xlarge	1.6.6	150	N/A
cache.c7gn.16xlarge	1.6.6	200	N/A

현재 세대(Valkey 또는 RedisOSS)

이전 세대에 대한 자세한 내용은 이전 세대 노드를 참조하세요.

> **Note**
>
> 버스트 가능 네트워크 성능을 발휘하는 인스턴스 유형은 네트워크 I/O 크레딧 메커니즘을 사용하여 최선의 노력을 기준으로 기준 대역폭을 초과하여 버스트할 수 있습니다.

일반

인스턴스 유형	지원되는 최소 Redis OSS 버전	향상된 I/O(Redis OSS 5.0.6 이상)	TLS 오프로드(Redis OSS 6.2.5 이상)	향상된 I/O 멀티플렉싱 (Redis OSS 7.0.4 이상)	기준 대역폭 (Gbps)	버스트 대역폭 (Gbps)
cache.m7g.large	6.2	N	N	N	0.937	12.5
cache.m7g.xlarge	6.2	Y	Y	Y	1.876	12.5
cache.m7g.2xlarge	6.2	Y	Y	Y	3.75	15
cache.m7g.4xlarge	6.2	Y	Y	Y	7.5	15
cache.m7g.8xlarge	6.2	Y	Y	Y	15	N/A
cache.m7g.12xlarge	6.2	Y	Y	Y	22.5	N/A
cache.m7g.16xlarge	6.2	Y	Y	Y	30	N/A
cache.m6g.large	5.0.6	N	N	N	0.75	10.0

인스턴스 유형	지원되는 최소 Redis OSS 버전	향상된 I/O(Redis OSS 5.0.6 이상)	TLS 오프로드(Redis OSS 6.2.5 이상)	향상된 I/O 멀티플렉싱(Redis OSS 7.0.4 이상)	기준 대역폭(Gbps)	버스트 대역폭(Gbps)
cache.m6g.xlarge	5.0.6	Y	Y	Y	1.25	10.0
cache.m6g.2xlarge	5.0.6	Y	Y	Y	2.5	10.0
cache.m6g.4xlarge	5.0.6	Y	Y	Y	5.0	10.0
cache.m6g.8xlarge	5.0.6	Y	Y	Y	12	N/A
cache.m6g.12xlarge	5.0.6	Y	Y	Y	20	N/A
cache.m6g.16xlarge	5.0.6	Y	Y	Y	25	N/A
cache.m5.large	3.2.4	N	N	N	0.75	10.0
cache.m5.xlarge	3.2.4	Y	N	N	1.25	10.0

인스턴스 유형	지원되는 최소 Redis OSS 버전	향상된 I/O(Redis OSS 5.0.6 이상)	TLS 오프로드(Redis OSS 6.2.5 이상)	향상된 I/O 멀티플렉싱(Redis OSS 7.0.4 이상)	기준 대역폭(Gbps)	버스트 대역폭(Gbps)
cache.m5.2xlarge	3.2.4	Y	Y	Y	2.5	10.0
cache.m5.4xlarge	3.2.4	Y	Y	Y	5.0	10.0
cache.m5.12xlarge	3.2.4	Y	Y	Y	12	N/A
cache.m5.24xlarge	3.2.4	Y	Y	Y	25	N/A
cache.m4.large	3.2.4	N	N	N	0.45	1.2
cache.m4.xlarge	3.2.4	Y	N	N	0.75	2.8
cache.m4.2xlarge	3.2.4	Y	Y	Y	1.0	10.0
cache.m4.4xlarge	3.2.4	Y	Y	Y	2.0	10.0

인스턴스 유형	지원되는 최소 Redis OSS 버전	향상된 I/O(Redis OSS 5.0.6 이상)	TLS 오프로드(Redis OSS 6.2.5 이상)	향상된 I/O 멀티플렉싱(Redis OSS 7.0.4 이상)	기준 대역폭(Gbps)	버스트 대역폭(Gbps)
cache.m4.10xlarge	3.2.4	Y	Y	Y	5.0	10.0
cache.t4g.micro	3.2.4	N	N	N	0.064	5.0
cache.t4g.small	5.0.6	N	N	N	0.128	5.0
cache.t4g.medium	5.0.6	N	N	N	0.256	5.0
cache.t3.micro	3.2.4	N	N	N	0.064	5.0
cache.t3.small	3.2.4	N	N	N	0.128	5.0
cache.t3.medium	3.2.4	N	N	N	0.256	5.0
cache.t2.micro	3.2.4	N	N	N	0.064	1.024
cache.t2.small	3.2.4	N	N	N	0.128	1.024
cache.t2.medium	3.2.4	N	N	N	0.256	1.024

메모리 최적화

인스턴스 유형	지원되는 최소 Redis OSS 버전	향상된 I/O(Redis OSS 5.0.6 이상)	TLS 오프로드(Redis OSS 6.2.5 이상)	향상된 I/O 멀티플렉싱(Redis OSS 7.0.4 이상)	기준 대역폭(Gbps)	버스트 대역폭(Gbps)
cache.r7g.large	6.2	N	N	N	0.937	12.5
cache.r7g.xlarge	6.2	Y	Y	Y	1.876	12.5
cache.r7g.2xlarge	6.2	Y	Y	Y	3.75	15
cache.r7g.4xlarge	6.2	Y	Y	Y	7.5	15
cache.r7g.8xlarge	6.2	Y	Y	Y	15	N/A
cache.r7g.12xlarge	6.2	Y	Y	Y	22.5	N/A
cache.r7g.16xlarge	6.2	Y	Y	Y	30	N/A
cache.r6g.large	5.0.6	N	N	N	0.75	10.0

인스턴스 유형	지원되는 최소 Redis OSS 버전	향상된 I/O(Redis OSS 5.0.6 이상)	TLS 오프로드(Redis OSS 6.2.5 이상)	향상된 I/O 멀티플렉싱(Redis OSS 7.0.4 이상)	기준 대역폭(Gbps)	버스트 대역폭(Gbps)
cache.r6g.xlarge	5.0.6	Y	Y	Y	1.25	10.0
cache.r6g.2xlarge	5.0.6	Y	Y	Y	2.5	10.0
cache.r6g.4xlarge	5.0.6	Y	Y	Y	5.0	10.0
cache.r6g.8xlarge	5.0.6	Y	Y	Y	12	N/A
cache.r6g.12xlarge	5.0.6	Y	Y	Y	20	N/A
cache.r6g.16xlarge	5.0.6	Y	Y	Y	25	N/A
cache.r5.large	3.2.4	N	N	N	0.75	10.0
cache.r5.xlarge	3.2.4	Y	N	N	1.25	10.0
cache.r5.2xlarge	3.2.4	Y	Y	Y	2.5	10.0

인스턴스 유형	지원되는 최소 Redis OSS 버전	향상된 I/O(Redis OSS 5.0.6 이상)	TLS 오프로드(Redis OSS 6.2.5 이상)	향상된 I/O 멀티플렉싱(Redis OSS 7.0.4 이상)	기준 대역폭(Gbps)	버스트 대역폭(Gbps)
cache.r5.4xlarge	3.2.4	Y	Y	Y	5.0	10.0
cache.r5.12xlarge	3.2.4	Y	Y	Y	12	N/A
cache.r5.24xlarge	3.2.4	Y	Y	Y	25	N/A
cache.r4.large	3.2.4	N	N	N	0.75	10.0
cache.r4.xlarge	3.2.4	Y	N	N	1.25	10.0
cache.r4.2xlarge	3.2.4	Y	Y	Y	2.5	10.0
cache.r4.4xlarge	3.2.4	Y	Y	Y	5.0	10.0
cache.r4.8xlarge	3.2.4	Y	Y	Y	12	N/A
cache.r4.16xlarge	3.2.4	Y	Y	Y	25	N/A

데이터 계층화에 최적화된 메모리

인스턴스 유형	지원되는 최소 Redis OSS 버전	향상된 I/O(Redis OSS 5.0.6 이상)	TLS 오프로드(Redis OSS 6.2.5 이상)	향상된 I/O 멀티플렉싱(Redis OSS 7.0.4 이상)	기준 대역폭(Gbps)	버스트 대역폭(Gbps)
cache.r6gd.xlarge	6.2.0	Y	N	N	1.25	10
cache.r6gd.2xlarge	6.2.0	Y	Y	Y	2.5	10
cache.r6gd.4xlarge	6.2.0	Y	Y	Y	5.0	10
cache.r6gd.8xlarge	6.2.0	Y	Y	Y	12	N/A
cache.r6gd.12xlarge	6.2.0	Y	Y	Y	20	N/A
cache.r6gd.16xlarge	6.2.0	Y	Y	Y	25	N/A

네트워크 최적화

인스턴스 유형	지원되는 최소 Redis OSS 버전	향상된 I/O(Redis OSS 5.0.6 이상)	TLS 오프로드(Redis OSS 6.2.5 이상)	향상된 I/O 멀티플렉싱(Redis OSS 7.0.4 이상)	기준 대역폭(Gbps)	버스트 대역폭(Gbps)
cache.c7gn.large	6.2	N	N	N	6.25	30
cache.c7gn.xlarge	6.2	Y	Y	Y	12.5	40
cache.c7gn.2xlarge	6.2	Y	Y	Y	25	50
cache.c7gn.4xlarge	6.2	Y	Y	Y	50	N/A
cache.c7gn.8xlarge	6.2	Y	Y	Y	100	N/A
cache.c7gn.12xlarge	6.2	Y	Y	Y	150	N/A
cache.c7gn.16xlarge	6.2	Y	Y	Y	200	N/A

AWS 리전별로 지원되는 노드 유형

지원되는 노드 유형은 AWS 리전마다 다를 수 있습니다. 자세한 내용은 Amazon ElastiCache 요금 섹션을 참조하세요.

성능 순간 확장 가능 인스턴스

Amazon 에서 범용 버스트 가능 T4g, T3-Standard 및 T2-Standard 캐시 노드를 시작할 수 있습니다 ElastiCache. 이러한 노드는 누적 크레딧이 소진될 때까지 언제든지 CPU 사용량을 버스트할 수 있는 기능과 함께 기준 수준의 CPU 성능을 제공합니다. CPU 크레딧은 1분 동안 전체 CPU 코어의 성능을 제공합니다.

Amazon ElastiCache의 T4g, T3 및 T2 노드는 표준으로 구성되며 평균 CPU 사용률이 인스턴스의 기준 성능보다 일관되게 낮은 워크로드에 적합합니다. 기준 이상으로 버스트하기 위해 노드는 크레딧 잔액에 누적된 CPU 크레딧을 사용합니다. 누적된 크레딧에서 노드가 부족한 경우, 성능이 점진적으로 기준 성능 수준으로 저하됩니다. 이렇게 점진적으로 낮추면 누적 CPU 크레딧 잔액이 고갈될 때 노드가 급격한 성능 저하를 경험하지 않습니다. 자세한 내용은 Amazon 사용 설명서의 CPU Credits and Baseline Performance for Burstable Performance Instances를 참조하세요. EC2

다음 표에는 버스트 가능한 성능 노드 유형, 시간당 CPU 크레딧이 적립되는 속도가 나열되어 있습니다. 또한 노드가 누적할 수 있는 최대 획득 CPU 크레딧 수와 노드 vCPUs 당 수를 보여줍니다. 또한 전체 코어 성능의 백분율로 기준 성능 수준을 제공합니다(단일 v 사용CPU).

CPU 시간당 적립된 크레딧	누적 가능한 최대 지급된 크레딧*	vCPUs	v당 기준 성능CPU	메모리(GiB)	네트워크 성능
12	288	2	10%	0.5	최대 5 기가비트
24	576	2	20%	1.37	최대 5 기가비트
24	576	2	20%	3.09	최대 5 기가비트
12	288	2	10%	0.5	최대 5 기가비트

CPU 시간당 적립된 크레딧	누적 가능한 최대 지급된 크레딧*	vCPUs	v당 기준 성능CPU	메모리(GiB)	네트워크 성능
24	576	2	20%	1.37	최대 5 기가비트
24	576	2	20%	3.09	최대 5 기가비트
6	144	1	10%	0.5	낮음에서 중간
12	288	1	20%	1.55	낮음에서 중간
24	576	2	20%	3.22	낮음에서 중간

* 누적될 수 있는 크레딧은 수는 24시간 동안 획득할 수 있는 크레딧의 수와 동일합니다.

** 테이블의 기준 성능은 v당 입니다CPU. v가 두 개 이상인 일부 노드 크기입니다CPU. 이를 위해 vCPU 백분율에 수를 곱하여 노드의 기준 CPU 사용률을 계산합니다vCPUs.

T3 및 T4g 버스트 가능한 성능 인스턴스에 대해 다음 CPU 크레딧 지표를 사용할 수 있습니다.

> **Note**
>
> T2 성능 순간 확장 가능 인스턴스에는 이러한 지표를 사용할 수 없습니다.

- `CPUCreditUsage`
- `CPUCreditBalance`

이러한 지표에 대한 자세한 내용은 CPU 크레딧 지표를 참조하세요.

또한 다음 사항을 숙지해야 합니다.

- 모든 현재 세대 노드 유형은 VPC 기본적으로 Amazon을 기반으로 가상 프라이빗 클라우드(VPC)에서 생성됩니다.
- Redis OSS 추가 전용 파일(AOF)은 T2 인스턴스에 지원되지 않습니다. Redis OSS 구성 변수 `appendonly` 및 `appendfsync`는 Redis OSS 버전 2.8.22 이상에서는 지원되지 않습니다.

관련 정보

- Amazon ElastiCache 제품 기능 및 세부 정보
- Memcached에 대한 Memcached 노드 유형별 파라미터
- Valkey 및 Redis OSS 파라미터
- 전송 중 암호화(TLS)

노드 재부팅

일부 변경 사항을 적용하려면 Redis OSS 또는 Memcached 클러스터를 재부팅해야 합니다. 예를 들어, 일부 파라미터는 파라미터 그룹의 파라미터 값을 변경할 경우 재부팅해야 변경 사항이 적용됩니다.

주제

- Redis OSS 노드 재부팅(클러스터 모드만 비활성화됨)
- Memcached용 클러스터 재부팅

Redis OSS 노드 재부팅(클러스터 모드만 비활성화됨)

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터의 경우 재부팅 후에만 적용되는 파라미터 그룹의 파라미터는 다음과 같습니다.

- activerehashing
- 데이터베이스

Redis 노드는 ElastiCache 콘솔을 통해서만 업데이트할 수 있습니다. 한 번에 하나의 노드만 재부팅할 수 있습니다. 여러 노드를 재부팅하려면 각 노드에 대해 프로세스를 반복해야 합니다.

> **Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 파라미터 변경 사항**
>
> Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터에서 다음 파라미터를 변경하는 경우 다음 단계를 따릅니다.

- activerehashing
- 데이터베이스

1. 클러스터의 수동 백업을 만듭니다. 수동 백업 지원을 참조하세요.
2. Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터를 삭제합니다. 에서 클러스터 삭제 ElastiCache을 참조하세요.
3. 변경한 파라미터 그룹 및 백업을 사용해 클러스터를 복원하여 새로운 클러스터를 시드합니다. 백업에서 새 캐시로 복원을 참조하세요.

다른 파라미터를 변경한 경우에는 필요하지 않습니다.

사용 AWS Management Console

ElastiCache 콘솔을 사용하여 노드를 재부팅할 수 있습니다.

노드를 재부팅하려면(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 오른쪽 상단 모서리의 목록에서 해당하는 AWS 리전을 선택합니다.
3. 왼쪽 탐색 창에서 Redis OSS를 선택합니다.

   Redis를 실행하는 클러스터 목록이 OSS 나타납니다.
4. 클러스터 이름에서 클러스터를 선택합니다.
5. 노드 이름에서 재부팅할 노드 옆에 있는 라디오 버튼을 선택합니다.
6. 작업을 선택한 후 노드 재부팅을 선택합니다.

여러 노드를 재부팅하려면 재부팅할 노드마다 2단계에서 5단계까지 반복합니다. 한 노드의 재부팅이 완료되기를 기다렸다가 그다음 노드를 재부팅할 필요는 없습니다.

Memcached용 클러스터 재부팅

Memcached 클러스터를 재부팅하면 클러스터가 모든 데이터를 플러시하고 엔진을 다시 시작합니다. 이 프로세스 중에는 클러스터에 액세스할 수 없습니다. 클러스터가 해당 데이터를 모두 플러시하기 때문에 클러스터가 다시 사용 가능한 상태가 되면 빈 클러스터로 시작됩니다.

ElastiCache 콘솔, AWS CLI또는 를 사용하여 클러스터를 재부팅할 수 있습니다 ElastiCache API. ElastiCache 콘솔, AWS CLI 또는 를 사용하든 ElastiCache API간에 단일 클러스터 재부팅만 시작할 수 있습니다. 여러 클러스터를 재부팅하려면 프로세스나 작업에서 반복해야 합니다.

사용 AWS Management Console

ElastiCache 콘솔을 사용하여 클러스터를 재부팅할 수 있습니다.

클러스터를 재부팅하려면(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 오른쪽 상단 모서리의 목록에서 관심 있는 AWS 리전을 선택합니다.
3. 탐색 창에서 재부팅하려는 클러스터에서 실행 중인 엔진을 선택합니다.

   선택한 엔진을 실행하는 클러스터 목록이 표시됩니다.
4. 클러스터 이름 왼쪽의 상자를 선택하여 재부팅할 클러스터를 선택합니다.

   Reboot(재부팅) 버튼이 활성화됩니다.

   클러스터를 2개 이상 선택하면 Reboot(재부팅) 버튼이 비활성화됩니다.
5. 재부팅을 선택합니다.

   클러스터 재부팅 확인 화면이 나타납니다.
6. 클러스터를 재부팅하려면 [Reboot]를 선택합니다. 클러스터 상태가 rebooting cluster nodes(클러스터 노드 재부팅)로 변경됩니다.

   클러스터를 재부팅하지 않으려면 [Cancel]을 선택합니다.

여러 클러스터를 재부팅하려면 재부팅할 클러스터마다 2단계에서 5단계까지 반복합니다. 한 클러스터의 재부팅이 완료되기를 기다렸다가 그다음 클러스터를 재부팅할 필요는 없습니다.

특정 노드를 재부팅하려면 노드를 선택한 다음재부팅을 선택합니다.

사용 AWS CLI

클러스터(AWS CLI)를 재부팅하려면 `reboot-cache-cluster` CLI 작업을 사용합니다.

클러스터의 특정 노드를 재부팅하려면 `--cache-node-ids-to-reboot`를 사용하여 재부팅할 특정 클러스터를 나열하세요. 다음 명령을 통해 my-cluster의 노드 0001, 0002 및 0004가 재부팅됩니다.

Linux, macOS, Unix의 경우:

```
aws elasticache reboot-cache-cluster \
    --cache-cluster-id my-cluster \
    --cache-node-ids-to-reboot 0001 0002 0004
```

Windows의 경우:

```
aws elasticache reboot-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --cache-node-ids-to-reboot 0001 0002 0004
```

클러스터의 모든 노드를 재부팅하려면 `--cache-node-ids-to-reboot` 파라미터를 사용하고 클러스터의 모든 노드 ID를 나열하세요. 자세한 내용은 reboot-cache-cluster를 참조하세요.

사용 ElastiCache API

를 사용하여 클러스터를 재부팅하려면 `RebootCacheCluster` 작업을 ElastiCache API사용합니다.

클러스터의 특정 노드를 재부팅하려면 `CacheNodeIdsToReboot`를 사용하여 재부팅할 특정 클러스터를 나열하세요. 다음 명령을 통해 my-cluster의 노드 0001, 0002 및 0004가 재부팅됩니다.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=RebootCacheCluster
   &CacheClusterId=my-cluster
   &CacheNodeIdsToReboot.member.1=0001
   &CacheNodeIdsToReboot.member.2=0002
   &CacheNodeIdsToReboot.member.3=0004
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

클러스터의 모든 노드를 재부팅하려면 `CacheNodeIdsToReboot` 파라미터를 사용하고 클러스터의 모든 노드 ID를 나열하세요. 자세한 내용은 RebootCacheCluster를 참조하세요.

노드 교체(Valkey 및 RedisOSS)

Amazon은 패치와 업그레이드가 인스턴스에 원활하게 적용되어 플릿을 ElastiCache 자주 업그레이드 합니다. 그러나 기본 호스트에 필수 OS 업데이트를 적용하려면 때때로 ElastiCache 노드를 다시 시작해야 합니다. 보안, 안정성 및 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**
>
> Amazon에서 자동으로 생성된 교체 캐시 노드의 IP 주소는 다를 ElastiCache 수 있습니다. 애플리케이션 구성을 검토하여 캐시 노드가 적절한 IP 주소와 연결되어 있는지 확인해야 합니다.

다음 목록은 가 Valkey 또는 Redis OSS 노드 중 하나를 교체하도록 ElastiCache 예약할 때 수행할 수 있는 작업을 식별합니다. 상황에 맞는 정보를 신속하게 찾으려면 다음 메뉴에서 선택하세요.

- Do nothing - Amazon이 노드를 일정에 따라 ElastiCache 교체하도록 합니다.
- Change your maintenance window - 더 적합한 시간으로 유지 관리 기간을 변경합니다.
- Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 구성
  - Replace the only node in any Valkey or Redis OSS cluster - 백업 및 복원을 사용하여 Valkey 또는 Redis OSS 클러스터의 노드를 교체하는 절차입니다.
  - Replace a replica node in any Valkey or Redis OSS cluster - OSS 클러스터 가동 중지 없이 복제본 수를 늘리고 줄여 Valkey 또는 Redis 클러스터의 읽기 전용 복제본을 교체하는 절차입니다.
  - Replace any node in a Valkey or Redis OSS (cluster mode enabled) shard - 스케일 아웃 및 스케일 인을 통해 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 노드를 교체하기 위한 클러스터 가동 중지 시간이 없는 동적 절차입니다.
- Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 구성
  - Replace the only node in any Valkey or Redis OSS cluster - 백업 및 복원을 사용하여 Valkey 또는 Redis OSS 클러스터의 모든 노드를 교체하는 절차입니다.
  - Replace a replica node in any Valkey or Redis OSS cluster - OSS 클러스터 가동 중지 없이 복제본 수를 늘리고 줄여 Valkey 또는 Redis 클러스터의 읽기 전용 복제본을 교체하는 절차입니다.

- Replace a node in a Valkey or Redis OSS (cluster mode disabled) cluster - 복제를 사용하여 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터의 노드를 교체하는 절차입니다.
- Replace a Valkey or Redis OSS (cluster mode disabled) read-replica – Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹에서 읽기 전용 복제본을 수동으로 교체하는 절차입니다.
- Replace a Valkey or Redis OSS (cluster mode disabled) primary node - Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹의 기본 노드를 수동으로 교체하는 절차입니다.

Valkey 및 Redis OSS 노드 교체 옵션

- 아무것도 하지 않음 - 아무것도 하지 않으면 는 노드를 일정에 따라 ElastiCache 교체합니다.

  Autofailover가 활성화된 클러스터가 아닌 구성의 경우 클러스터가 계속 온라인 상태를 유지하고 수신 쓰기 요청을 처리하는 동안 Valkey 7.2 이상 및 Redis OSS 5.0.6 이상의 클러스터가 교체를 완료합니다. Redis OSS 4.0.10 이하에서 자동 장애 조치가 활성화된 클러스터의 경우 DNS 업데이트와 관련하여 최대 몇 초 동안 짧은 쓰기 중단이 발생할 수 있습니다.

  노드가 자동 장애 조치 활성화 클러스터의 구성원인 경우 Valkey 또는 RedisOSS를 ElastiCache 사용하면 패치, 업데이트 및 기타 유지 관리 관련 노드 교체 중에 가용성이 향상됩니다.

  Valkey 또는 Redis OSS 클러스터 OSS 클라이언트 ElastiCache 와 ElastiCache 함께 사용하도록 설정된 의 경우 클러스터가 수신 쓰기 요청을 처리하는 동안 교체가 완료됩니다.

  Autofailover가 활성화된 클러스터가 아닌 구성의 경우 클러스터가 계속 온라인 상태를 유지하고 수신 쓰기 요청을 처리하는 동안 Valkey 7.2 이상 및 Redis OSS 5.0.6 이상의 클러스터가 교체를 완료합니다. Redis OSS 4.0.10 이하에서 자동 장애 조치가 활성화된 클러스터의 경우 DNS 업데이트와 관련하여 최대 몇 초 동안 짧은 쓰기 중단이 발생할 수 있습니다.

  노드가 독립 실행형인 경우 Amazon은 ElastiCache 먼저 대체 노드를 시작한 다음 기존 노드와 동기화합니다. 그 동안은 서비스 요청에 기존 노드를 사용할 수 없습니다. 동기화가 완료되면 기존 노드가 종료되고 새 노드가 대신됩니다. ElastiCache 는 이 작업 중에 데이터를 유지하기 위해 최선의 노력을 기울입니다.

- 유지 관리 기간 변경 - 예약된 유지 관리 이벤트의 경우 에서 이메일 또는 알림 이벤트를 수신합니다 ElastiCache. 이러한 경우 예약된 대체 시간 전에 유지 관리 기간을 변경하면 이제 노드가 새 시간에 대체됩니다. 자세한 내용은 다음 자료를 참조하세요.
  - ElastiCache 클러스터 수정
  - 복제 그룹 수정

  > **Note**
  >
  > 유지 관리 기간을 이동하여 교체 기간을 변경하는 기능은 ElastiCache 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 15:00, 다음 유지 관리 기간은 11월 10일 금요일 17:00라고 가정해 보겠습니다. 다음을 이러한 가정의 결과를 보여주는 3가지 시나리오입니다.
  - 유지 관리 기간을 현재 날짜/시간 이후 및 예약된 다음 유지 관리 기간 이전인 금요일 16:00으로 변경합니다. 11월 10일 금요일 16:00에 노드가 대체됩니다.
  - 유지 관리 기간을 현재 날짜/시간 이후 및 예약된 다음 유지 관리 기간 이전인 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 ElastiCache 클러스터 유지 관리 단원을 참조하십시오.

- Valkey 또는 Redis OSS 클러스터의 유일한 노드 교체 - 클러스터에 읽기 전용 복제본이 없는 경우 다음 절차를 사용하여 노드를 교체할 수 있습니다.

  백업 및 복원을 사용하여 노드만을 대체하려면 다음을 수행합니다.

  1. 노드 클러스터의 스냅샷을 생성합니다. 지침은 수동 백업 지원 단원을 참조하십시오.
  2. 스냅샷에서 시드하여 새 클러스터를 생성합니다. 지침은 백업에서 새 캐시로 복원 단원을 참조하십시오.
  3. 대체 예약한 노드가 포함된 클러스터를 삭제합니다. 지침은 에서 클러스터 삭제 ElastiCache 단원을 참조하십시오.

4. 애플리케이션에서 이전 노드의 엔드포인트를 새 노드의 엔드포인트로 대체합니다.

- Valkey 또는 Redis OSS 클러스터의 복제본 노드 교체 - 복제본 클러스터를 교체하려면 복제본 수를 늘리세요. 이렇게 하려면 복제본을 추가한 다음 대체할 복제본을 제거하여 복제본 수를 줄입니다. 이 프로세스는 동적이며 클러스터 중단 시간이 없습니다.

  > **Note**
  >
  > 샤드 또는 복제 그룹에 이미 5개 복제본이 있는 경우 1단계와 2단계를 반대로 합니다.

  Valkey 또는 Redis OSS 클러스터에서 복제본을 교체하려면

  1. 샤드 또는 복제 그룹에 복제본을 추가하여 복제본 수를 늘립니다. 자세한 내용은 샤드의 복제본 수 늘리기 섹션을 참조하세요.
  2. 대체하려는 복제본을 삭제합니다. 자세한 내용은 샤드의 복제본 수 줄이기 섹션을 참조하세요.
  3. 애플리케이션에서 엔드포인트를 업데이트합니다.

- Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 샤드의 노드 교체 - 클러스터의 노드를 가동 중지 없이 교체하려면 온라인 리샤딩을 사용합니다. 먼저 확장하여 샤드를 추가한 다음 축소하여 대체할 노드로 샤드를 삭제합니다.

  Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 노드를 교체하려면

  1. 확장: 대체할 노드가 포함된 기존 샤드와 동일한 구성의 추가 샤드를 추가합니다. 자세한 내용은 온라인 리샤딩을 사용하여 샤드 추가 섹션을 참조하세요.
  2. 축소: 대체할 노드가 포함된 샤드를 삭제합니다. 자세한 내용은 온라인 리샤딩을 사용하여 샤드 제거 섹션을 참조하세요.
  3. 애플리케이션에서 엔드포인트를 업데이트합니다.

- Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터의 노드 교체 - 클러스터가 읽기 전용 복제본이 없는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터인 경우 다음 절차에 따라 노드를 교체합니다.

  복제를 사용하여 노드 대체하려면(클러스터 모드 비활성화 전용)

  1. 기본으로 대체하도록 예약한 노드가 있는 클러스터에 복제를 추가합니다. 이 클러스터에서 다중 AZ를 활성화하지 마십시오. 지침은 샤드 없이 Valkey 또는 Redis OSS 클러스터에 복제를 추가하려면 단원을 참조하십시오.
  2. 클러스터에 읽기 전용 복제본을 추가합니다. 지침은 ElastiCache 클러스터에 노드를 추가하려면(콘솔) 단원을 참조하십시오.
  3. 읽기 전용 복제본을 기본으로 승격합니다. 지침은 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹의 읽기 전용 복제본을 기본 복제본으로 승격 단원을 참조하십시오.
  4. 대체 예약한 노드를 삭제합니다. 지침은 ElastiCache 클러스터에서 노드 제거 단원을 참조하십시오.
  5. 애플리케이션에서 이전 노드의 엔드포인트를 새 노드의 엔드포인트로 대체합니다.

- Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 읽기 전용 복제본 교체 - 노드가 읽기 전용 복제본인 경우 노드를 교체합니다.

  클러스터에 복제본 노드가 한 개뿐이고 다중 AZ가 활성화되어 있으면 다중 AZ를 비활성화해야 복제본을 삭제할 수 있습니다. 지침은 복제 그룹 수정 단원을 참조하십시오.

  Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 읽기 전용 복제본을 교체하려면

  1. 대체 예약된 복제본을 삭제합니다. 지침은 다음을 참조하세요.
     - 샤드의 복제본 수 줄이기
     - ElastiCache 클러스터에서 노드 제거
  2. 대체 예약된 복제본을 대체할 새 복제본을 추가합니다. 삭제한 복제본과 같은 이름을 사용하는 경우 3단계를 건너뛸 수 있습니다. 지침은 다음을 참조하세요.
     - 샤드의 복제본 수 늘리기
     - Valkey 또는 Redis에 대한 읽기 전용 복제본 추가OSS(클러스터 모드 비활성화됨)
  3. 애플리케이션에서 이전 복제본의 엔드포인트를 새 복제본의 엔드포인트로 대체합니다.

4. 시작할 때 다중 AZ를 비활성화한 경우 다시 활성화합니다. 지침은 다중 AZ 활성화 단원을 참조하십시오.

- Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 기본 노드 교체 - 노드가 기본 노드인 경우 먼저 읽기 전용 복제본을 기본 노드로 승격합니다. 그런 다음 기본 노드였던 복제본을 삭제합니다.

  클러스터에 복제본이 한 개뿐이고 다중 AZ가 활성화되어 있으면 2단계에서 다중 AZ를 비활성화해야 복제본을 삭제할 수 있습니다. 지침은 복제 그룹 수정 단원을 참조하십시오.

  Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 기본 노드를 교체하려면

  1. 읽기 전용 복제본을 기본으로 승격합니다. 지침은 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹의 읽기 전용 복제본을 기본 복제본으로 승격 단원을 참조하십시오.
  2. 대체 예약된 노드(이전의 기본)를 삭제합니다. 지침은 ElastiCache 클러스터에서 노드 제거 단원을 참조하십시오.
  3. 대체 예약된 복제본을 대체할 새 복제본을 추가합니다. 삭제한 노드와 같은 이름을 사용하는 경우 애플리케이션에서 엔드포인트 변경을 건너뛸 수 있습니다.

     지침은 Valkey 또는 Redis에 대한 읽기 전용 복제본 추가OSS(클러스터 모드 비활성화됨) 단원을 참조하십시오.
  4. 애플리케이션에서 이전 노드의 엔드포인트를 새 노드의 엔드포인트로 대체합니다.
  5. 시작할 때 다중 AZ를 비활성화한 경우 다시 활성화합니다. 지침은 다중 AZ 활성화 단원을 참조하세요.

노드 교체(Memcached)

Amazon ElastiCache (Memcached)은 패치와 업그레이드를 인스턴스에 원활하게 적용하여 플릿을 자주 업그레이드합니다. 그러나 기본 호스트에 필수 OS 업데이트를 적용하려면 때때로 ElastiCache (Memcached) 노드를 다시 시작해야 합니다. 보안, 안정성 및 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**
>
> Amazon에서 자동으로 생성된 교체 캐시 노드의 IP 주소는 다를 ElastiCache 수 있습니다. 애플리케이션 구성을 검토하여 캐시 노드가 적절한 IP 주소와 연결되어 있는지 확인해야 합니다.

다음 목록은 가 Memcached 노드 중 하나를 교체하도록 ElastiCache 예약할 때 수행할 수 있는 작업을 식별합니다.

- 아무것도 하지 않음 - 아무것도 ElastiCache 하지 않으면 가 노드를 일정에 따라 바꿉니다. 가 자동으로 노드를 새 노드로 ElastiCache 교체하면 새 노드는 처음에 비어 있습니다.
- 유지 관리 기간 변경 - 예약된 유지 관리 이벤트의 경우 에서 이메일 또는 알림 이벤트를 수신합니다 ElastiCache. 이 경우 예약된 대체 시간 전에 유지 관리 기간을 변경하면 이제 노드가 새 시간에 대체됩니다. 자세한 내용은 ElastiCache 클러스터 수정 단원을 참조하십시오.

  > **Note**
  >
  > 유지 관리 기간을 이동하여 교체 기간을 변경하는 기능은 ElastiCache 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 15:00, 다음 유지 관리 기간은 11월 10일 금요일 17:00라고 가정해 보겠습니다. 다음을 이러한 가정의 결과를 보여주는 3가지 시나리오입니다.

  - 유지 관리 기간을 현재 날짜/시간 이후 및 예약된 다음 유지 관리 기간 이전인 금요일 16:00으로 변경합니다. 11월 10일 금요일 16:00에 노드가 대체됩니다.

- 유지 관리 기간을 현재 날짜/시간 이후 및 예약된 다음 유지 관리 기간 이전인 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 ElastiCache 클러스터 유지 관리 단원을 참조하십시오.

- 수동으로 노드 대체 - 다음 유지 관리 기간 전에 노드를 대체하려면 노드를 수동으로 대체하세요.

  노드를 수동으로 대체하면 키는 재분산됩니다. 이 재분산으로 인해 캐시가 누락될 수 있습니다.

  Memcached 노드를 수동으로 대체하려면

  1. 대체 예약한 노드를 삭제합니다. 지침은 ElastiCache 클러스터에서 노드 제거 단원을 참조하십시오.
  2. 클러스터에 새 노드를 추가합니다. 지침은 ElastiCache 클러스터에 노드 추가 단원을 참조하십시오.
  3. 이 클러스터에서 자동 검색 기능을 사용하지 않으려면 애플리케이션을 확인하고 기존 노드의 모든 엔드포인트 인스턴스를 새로운 노드 엔드포인트로 대체합니다.

예약 노드

하나 이상의 ElastiCache 노드를 예약하면 비용을 절감할 수 있습니다. 노드 유형과 예약 기간(1년 또는 3년)에 따라 예약 노드에 선결제 요금이 부과됩니다.

예약된 노드가 사용 사례에 대해 비용이 절감되는지 확인하려면 먼저 필요한 노드 수와 노드 크기를 결정합니다. 그런 다음 노드의 사용량을 예측하고 온디맨드 노드와 예약된 노드의 총 비용을 비교합니다. 클러스터에서 예약 노드와 온디맨드 노드를 함께 사용할 수 있습니다. 요금 정보는 Amazon ElastiCache 요금 섹션을 참조하세요.

주제

- 예약 노드를 통해 비용 관리
- 표준 예약 노드 제품
- 유연한 크기의 예약 노드
- 예약 노드 삭제
- 이전 예약 노드 제품
- 예약 노드 제품에 대한 정보 가져오기

- 예약 노드 구입
- 예약 노드에 대한 정보 가져오기

예약 노드를 통해 비용 관리

하나 이상의 노드를 예약하여 비용을 줄일 수 있습니다. 노드 유형과 예약 기간(1년 또는 3년)에 따라 예약 노드에 선결제 요금이 부과됩니다. 이 요금은 온디맨드 노드에서 발생하는 시간당 사용 요금보다 훨씬 낮습니다.

예약된 노드가 사용 사례에 대해 비용이 절감되는지 확인하려면 먼저 필요한 노드 수와 노드 크기를 결정합니다. 그런 다음 노드의 사용량을 예측하고 온디맨드 노드와 예약된 노드의 총 비용을 비교합니다. 클러스터에서 예약 노드와 온디맨드 노드를 함께 사용할 수 있습니다. 요금 정보는 Amazon ElastiCache 요금 섹션을 참조하세요.

AWS 구매 시 리전, 노드 유형 및 기간 길이를 선택해야 하며 나중에 변경할 수 없습니다.

AWS Management Console, AWS CLI또는 ElastiCache API 를 사용하여 사용 가능한 예약 노드 제품을 나열하고 구매할 수 있습니다.

예약 노드에 대한 자세한 내용은 Amazon ElastiCache 예약 노드 를 참조하세요.

표준 예약 노드 제품

Amazon 에서 예약 노드 인스턴스(RI)를 구매할 때 예약 노드 인스턴스의 기간 동안 특정 노드 인스턴스 유형 및 AWS 리전에 대해 할인된 요금을 얻기 위한 약정을 구매할 ElastiCache수 있습니다. Amazon ElastiCache 예약 노드 인스턴스를 사용하려면 온디맨드 인스턴스와 마찬가지로 새 ElastiCache 노드 인스턴스를 생성합니다.

새 예약 노드 인스턴스의 사양이 계정의 기존 예약 노드 인스턴스와 일치하는 경우 예약 노드 인스턴스에 제공되는 할인 요금이 청구됩니다. 그렇지 않으면 노드 인스턴스에 대해 온디맨드 요금이 청구됩니다. 이러한 표준RIs은 R5 및 M5 인스턴스 패밀리부터 사용할 수 있습니다.

> **Note**
>
> 다음에 설명된 모든 제안 유형은 1년 및 3년 기간으로 제공됩니다.

제공 유형

선결제 RI는 선결제 없이 예약 ElastiCache 인스턴스에 대한 액세스를 제공하지 않습니다. 예약되지 않은 ElastiCache 인스턴스는 사용량에 관계없이 기간 내 매시간 할인된 시간당 요금을 청구합니다.

부분 선불 RI를 사용하려면 예약된 ElasticCache 인스턴스의 일부를 선불로 지불해야 합니다. 결제하지 않은 시간에 대해서는 사용 기간 동안 사용량에 상관없이 할인된 시간당 요금이 청구됩니다. 이 옵션은 다음 섹션에서 설명할 이전 Heavy 사용률 옵션을 대신합니다.

전체 선결제 RI의 경우 RI 사용 기간이 시작될 때 전액 지불해야 합니다. 사용 시간과 관계없이 남은 기간 동안 다른 비용은 발생하지 않습니다.

유연한 크기의 예약 노드

예약된 모든 노드의 크기는 유연합니다. 예약 노드를 구매할 때 지정하는 한 가지는 cache.r6g.xlarge와 같은 노드 유형입니다. 노드 유형에 대한 자세한 내용은 Amazon ElastiCache 요금 섹션을 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. 크기 조정이 가능한 예약 노드는 AWS 리전이 동일한 노드에서 사용할 수 있습니다. 유연한 크기의 예약 노드는 해당 노드 제품군에서만 확장할 수 있습니다. 예를 들어 cache.r6g.xlarge에 대해 예약된 노드는 cache.r6g.2xlarge에 적용될 수 있지만 cache.r6gd.large에는 적용할 수 없습니다. cache.r6g와 cache.r6gd는 노드 패밀리가 다르기 때문입니다.

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

예약된 노드를 Redis에서 ValkeyOSS로 업그레이드

에서 Valkey를 시작 ElastiCache하면 이제 Valkey 캐시 엔진에 Redis OSS 예약 노드 할인을 적용할 수 있습니다. 기존 계약 및 예약의 이점을 유지하면서 Redis에서 ValkeyOSS로 업그레이드할 수 있습니다. 캐시 노드 패밀리 및 엔진 내에서 이점을 적용할 수 있을 뿐만 아니라 더 많은 증분 값을 얻을 수도 있습니다. Valkey는 Redis 에 비해 20% 할인된 가격으로 제공되며, 예약된 노드 유연성OSS으로 Redis OSS 예약 노드를 사용하여 실행 중인 Valkey 노드를 20% 더 많이 커버할 수 있습니다.

할인율을 계산하기 위해 각 ElastiCache 노드와 엔진 조합에는 단위로 측정되는 정규화 계수가 있습니다. 예약 노드 유닛은 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 포함할 수 있습니다. Valkey는 Redis OSS 및 Memcached에 비해 할인된 가격으로 가격이 책정되므로 지정된 인스턴스 유형의 단위가 낮아 Redis OSS 예약 노드가 더 많은 Valkey 노드를 커버할 수 있습니다.

예를 들어, 캐시.r7g.4xlarge for the Redis OSS 엔진(32개 유닛)에 대해 예약 노드를 구매했고 캐시.r7g.4xlarge Redis OSS 노드(32개 유닛) 하나를 실행 중이라고 가정해 보겠습니다. 노드를 Valkey로 업그레이드하면 실행 중인 노드의 정규화 인수가 25.6단위로 떨어지며, 기존 예약 노드는 리전의 cache.r7g 패밀리 내에서 실행 중인 다른 Valkey 또는 Redis OSS 노드에 사용할 추가 6.4단위를 제공합니다. 이를 사용하여 계정의 다른 cache.r7g.4xlarge Valkey 노드의 25%(25.6개 단위) 또는 cache.r7g.xlarge Valkey 노드의 100%(6.4개 단위)를 포함할 수 있습니다.

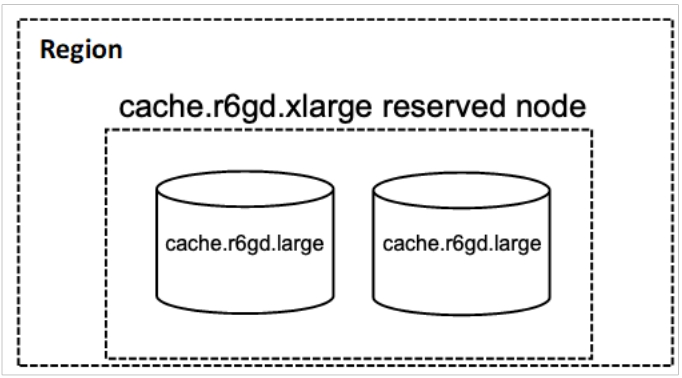
사용량과 정규화된 단위 비교

예약 노드의 크기에 따른 사용량은 정규화 유닛을 사용하여 비교할 수 있습니다. 예를 들어, cache.r6g.4xlarge 노드 2개에서 1시간 사용은 cache.r6g.large 1개에서 16시간 사용에 해당합니다. 다음 표는 노드 크기에 따른 정규화 유닛의 수를 나타낸 것입니다.

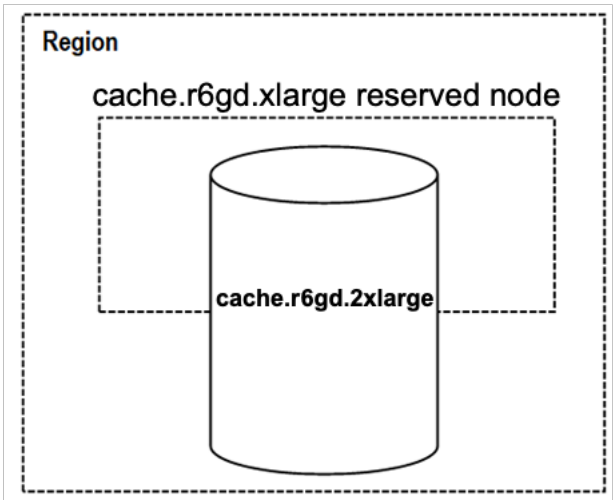
노드 크기	정규화된 단위(Redis OSS, Memcached)	정규화된 단위(Valkey)
micro	0.5	0.4
small	1	8.
medium	2	1.6
large	4	3.2
xlarge	8	6.4
2xlarge	16	12.8
4xlarge	32	25.6
6xlarge	48	38.4
8xlarge	64	51.2
10xlarge	80	64
12xlarge	96	76.8
16xlarge	128	102.4

노드 크기	정규화된 단위(Redis OSS, Memcached)	정규화된 단위(Valkey)
24xlarge	192	153.6

예를 들어 cache.r6gd.xlarge 예약 노드를 구매하고 동일한 AWS 리전의 계정에 실행 중인 cache.r6gd.large 예약 노드가 두 개 있습니다. 이 경우, 결제 혜택은 두 노드에 100% 적용됩니다.



또는 동일한 AWS 리전의 계정에서 실행 중인 cache.r6gd.2xlarge 인스턴스가 하나 있는 경우 예약된 노드 사용의 50%에 청구 혜택이 적용됩니다.



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.

이전 예약 노드 제품

세 가지의 이전 노드 예약 수준에는 Heavy 사용률, Medium 사용률 및 Light 사용률이 있습니다. 어느 사용률 수준으로나 1년 또는 3년간 노드를 예약할 수 있습니다. 노드 유형, 사용률 수준 및 예약 기간에 따라 총 비용이 정해집니다. 예약 노드를 구입하기 전에 다양한 모델을 비교하여 예약 노드가 회사에 줄 수 있는 절약 효과를 확인하세요.

특정 사용률 수준이나 기간으로 구입한 노드를 다른 사용률 수준이나 기간으로 바꿀 수 없습니다.

사용률 수준

Heavy 사용률 예약 노드는 용량 기준이 일관된 워크로드를 가능하게 하거나 되거나 상태가 꾸준한 워크로드를 실행합니다. Heavy 사용률 예약 노드는 높은 선납금 약정이 필요하지만 예약 노드 기간의 79% 이상을 실행하려는 경우 가장 크게 비용을 절감할 수 있습니다(온디맨드 요금의 70%까지). Heavy 사용률 예약 노드의 경우, 일회성 요금을 지불합니다. 이는 노드가 실행 여부에 관계없이 사용 기간 동안 시간당 요금이 더 낮습니다.

Medium 사용률 예약 노드는 많은 기간 예약 노드를 이용할 계획이고 더 저렴한 일회성 요금을 원하거나 노드 사용을 종료할 때 노드 요금 지불을 중지할 수 있기를 원하는 경우 가장 적합한 옵션입니다. Medium 사용률 예약 노드는 예약 노드 기간의 40% 이상을 실행하려는 경우 더욱 비용 효율적인 옵션입니다. 이 옵션은 온 디맨드 가격의 최대 64%까지 절감할 수 있습니다. Medium 사용률 예약 노드를 사용하면 Light 사용률 예약 노드를 사용하는 것보다 약간 높은 일회성 요금을 지불하지만 노드를 실행할 때 시간당 사용 요금을 낮출 수 있습니다.

Light 사용률 예약 노드는 매일 두세 시간 또는 일주일에 며칠 정도 실행하는 정기 작업에 적합합니다. Light 사용률 예약 노드를 사용하면 일회성 요금을 지불한 후 노드를 실행할 때 할인된 시간당 요금을 지불하게 됩니다. 노드가 예약된 노드 사용 기간의 17% 이상을 실행 중인 경우, 비용을 절감할 수 있습니다. 예약된 노드의 전체 사용 기간 동안 온디맨드 요금의 최대 56%까지 절감할 수 있습니다.

이전 예약 노드 제품

제공 유형	선결제 비용	사용료	이점
Heavy 사용률	가장 높음	시간당 가장 낮은 요금. 예약 노드를 사용하고 있는지 상관없이	예상되는 예약 노드 사용률이 3년 약정 기준으로 79% 이상인 경우

제공 유형	선결제 비용	사용료	이점
		전체 기간에 적용됩니다.	전체적인 비용이 가장 낮습니다.
Medium 사용률	중간	노드를 실행하는 매 시간마다 시간당 사용 요금이 청구됩니다. 노드가 실행되지 않을 때는 시간당 요금이 청구되지 않습니다.	탄력적인 워크로드 또는 3년 약정 기간에 40% 이상의 보통 사용량이 필요한 경우에 적합합니다.
Light 사용률	가장 낮음	노드를 실행하는 매 시간마다 시간당 사용 요금이 청구됩니다. 노드가 실행되지 않을 때는 시간당 요금이 청구되지 않습니다. 모든 제공 유형 중에서 시간당 요금이 가장 높지만 예약 노드가 실행되고 있을 때만 요금이 적용됩니다.	항상 실행할 계획이라면 전체 비용이 가장 높습니다. 그러나, 이는 3년 약정 기간의 약 15% 이상 예약 노드를 자주 사용하지 않는 경우 전체 비용이 가장 낮습니다.
온디맨드 사용(예약 노드 없음)	None	시간당 요금이 가장 높습니다. 노드가 실행 중일 때마다 적용됩니다.	시간당 비용이 가장 높습니다.

자세한 내용은 Amazon ElastiCache 요금 섹션을 참조하세요.

예약 노드 제품에 대한 정보 가져오기

예약 노드를 구입하기 전에 사용 가능한 예약 노드 제품에 대한 정보를 확인할 수 있습니다.

다음 예제에서는 AWS Management Console AWS CLI, 및 를 사용하여 사용 가능한 예약 노드 제품에 대한 요금 및 정보를 가져오는 방법을 보여줍니다 ElastiCache API.

주제

예약 노드 제품에 대한 정보 가져오기(콘솔)

를 사용하여 사용 가능한 예약 클러스터 제품에 대한 요금 및 기타 정보를 가져오려면 다음 절차를 AWS Management Console사용합니다.

사용 가능한 예약 노드 제품에 대한 정보를 가져오려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 예약 노드를 선택합니다.
3. 예약 노드 구매(Purchase Reserved Node)를 선택합니다.
4. 엔진 에서 Valkey, Redis OSS또는 Memcached를 선택합니다.
5. 사용 가능한 제품을 확인하려면 다음 옵션 중에서 선택합니다.

   - 노드 유형(Node Type)
   - 용어
   - Offering Type(제공 유형)

   선택한 후 선택 항목의 노드당 비용과 총 비용이 예약 세부 정보(Reservation details) 아래에 표시됩니다.
6. [Cancel]을 선택하면 이 노드를 구입하지 않으며 요금이 발생하지 않습니다.

예약 노드 제품에 대한 정보 가져오기(AWS CLI)

Valkey 또는 Redis 에 대해 사용 가능한 예약 노드 제품에 대한 요금 및 기타 정보를 가져오려면 명령 프롬프트에 다음 명령을 OSS입력합니다.

```
aws elasticache describe-reserved-cache-nodes-offerings
```

이 작업은 다음과 유사한 출력을 생성합니다(JSON 형식).

```
{
          "ReservedCacheNodesOfferingId": "0xxxxxxxx-xxeb-44ex-xx3c-xxxxxxxx072",
          "CacheNodeType": "cache.xxx.large",
          "Duration": 94608000,
          "FixedPrice": XXXX.X,
          "UsagePrice": X.X,
          "ProductDescription": "redis",
          "OfferingType": "All Upfront",
          "RecurringCharges": [
              {
                  "RecurringChargeAmount": X.X,
                  "RecurringChargeFrequency": "Hourly"
              }
          ]
 },
 {
          "ReservedCacheNodesOfferingId": "0xxxxxxxx-xxeb-44ex-xx3c-xxxxxxxx072",
          "CacheNodeType": "cache.xxx.xlarge",
          "Duration": 94608000,
          "FixedPrice": XXXX.X,
          "UsagePrice": X.X,
          "ProductDescription": "redis",
          "OfferingType": "Partial Upfront",
          "RecurringCharges": [
              {
                  "RecurringChargeAmount": X.XXX,
                  "RecurringChargeFrequency": "Hourly"
              }
          ]
 },
 {
          "ReservedCacheNodesOfferingId": "0xxxxxxxx-xxeb-44ex-xx3c-xxxxxxxx072",
          "CacheNodeType": "cache.xxx.large",
          "Duration": 31536000,
```

```
            "FixedPrice": X.X,
            "UsagePrice": X.X,
            "ProductDescription": "redis",
            "OfferingType": "No Upfront",
            "RecurringCharges": [
                {
                    "RecurringChargeAmount": X.XXX,
                    "RecurringChargeFrequency": "Hourly"
                }
            ]
}
```

Memcached에 대해 사용 가능한 예약 노드 제품에 대한 요금 및 기타 정보를 가져오려면 명령 프롬프트에 다음 명령을 입력합니다.

```
 {
            "ReservedCacheNodesOfferingId": "0xxxxxxxx-xxeb-44ex-xx3c-xxxxxxxx072",
            "CacheNodeType": "cache.xxx.large",
            "Duration": 94608000,
            "FixedPrice": XXXX.X,
            "UsagePrice": X.X,
            "ProductDescription": "memcached",
            "OfferingType": "All Upfront",
            "RecurringCharges": [
                {
                    "RecurringChargeAmount": X.X,
                    "RecurringChargeFrequency": "Hourly"
                }
            ]
    },
 {
            "ReservedCacheNodesOfferingId": "0xxxxxxxx-xxeb-44ex-xx3c-xxxxxxxx072",
            "CacheNodeType": "cache.xxx.xlarge",
            "Duration": 94608000,
            "FixedPrice": XXXX.X,
            "UsagePrice": X.X,
            "ProductDescription": "memcached",
            "OfferingType": "Partial Upfront",
            "RecurringCharges": [
                {
                    "RecurringChargeAmount": X.XXXX,
                    "RecurringChargeFrequency": "Hourly"
                }
```

```
            ]
        },
        {
            "ReservedCacheNodesOfferingId": "0xxxxxxxx-xxeb-44ex-xx3c-xxxxxxxx072",
            "CacheNodeType": "cache.xx.12xlarge",
            "Duration": 31536000,
            "FixedPrice": X.X,
            "UsagePrice": X.X,
            "ProductDescription": "memcached",
            "OfferingType": "No Upfront",
            "RecurringCharges": [
                {
                    "RecurringChargeAmount": X.XXXX,
                    "RecurringChargeFrequency": "Hourly"
                }
            ]
}
```

자세한 내용은 참조의 describe-reserved-cache-nodes-offerings를 AWS CLI 참조하세요.

예약된 노드 제공에 대한 정보 가져오기(ElastiCache API)

사용 가능한 예약 노드 제품의 요금과 정보를 가져오려면 `DescribeReservedCacheNodesOfferings` 작업을 호출하세요.

Example

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeReservedCacheNodesOfferings
    &Version=2014-12-01
    &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>
```

자세한 내용은 참조DescribeReservedCacheNodesOfferings의 섹션을 참조하세요 ElastiCache API.

예약 노드 구입

다음 예제에서는 AWS Management Console, AWS CLI및 를 사용하여 예약 노드 제품을 구매하는 방법을 보여줍니다 ElastiCache API.

> **Important**
>
> 이 섹션의 예제를 따르면 되돌릴 수 없는 AWS 요금이 계정에 발생합니다.

주제

예약 노드 구매(콘솔)

이 예제에서는 예약 노드 ID가 myreservationID인 특정 예약 노드 제품 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f를 구입하는 방법을 보여줍니다.

다음 절차에서는 AWS Management Console 를 사용하여 id를 제공하여 예약 노드 제품을 구매합니다.

예약 노드를 구입하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 목록에서 예약 노드(Reserved Nodes) 링크를 선택합니다.
3. 예약 노드 구매(Purchase reserved nodes) 버튼을 선택합니다.
4. 엔진 에서 Valkey, Redis OSS또는 Memcached를 선택합니다.
5. 사용 가능한 제품을 확인하려면 다음 옵션 중에서 선택합니다.
   - 노드 유형(Node Type)
   - 용어
   - Offering Type(제공 유형)
   - 선택적 예약 노드 ID(Reserved node ID)

선택한 후 선택 항목의 노드당 비용과 총 비용이 예약 세부 정보(Reservation details) 아래에 표시됩니다.

6. 구입을 선택하세요.

예약 노드 구입(AWS CLI)

다음 예제에서는 예약 노드 ID가 myreservationID인 특정 예약 클러스터 상품 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f를 구입하는 방법을 보여줍니다.

명령 프롬프트에서 다음 명령을 입력합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache purchase-reserved-cache-nodes-offering \
    --reserved-cache-nodes-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f \
    --reserved-cache-node-id myreservationID
```

Windows의 경우:

```
aws elasticache purchase-reserved-cache-nodes-offering ^
    --reserved-cache-nodes-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f ^
    --reserved-cache-node-id myreservationID
```

이 명령은 다음과 비슷한 출력을 반환합니다.

```
RESERVATION  ReservationId      Class           Start Time                Duration
 Fixed Price  Usage Price  Count  State            Description      Offering Type
RESERVATION  myreservationid    cache.xx.small  2013-12-19T00:30:23.247Z  1y
 XXX.XX USD   X.XXX USD    1      payment-pending  memcached        Medium Utilization
```

자세한 내용은 참조의 purchase-reserved-cache-nodes-offering을 AWS CLI 참조하세요.

예약 노드 구입(ElastiCache API)

다음 예제에서는 예약 클러스터 ID가 myreservationID인 특정 예약 노드 제품 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f를 구입하는 방법을 보여줍니다.

다음 파라미터와 함께 `PurchaseReservedCacheNodesOffering` 작업을 호출합니다.

- ReservedCacheNodesOfferingId = 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f
- ReservedCacheNodeID = myreservationID
- CacheNodeCount = 1

Example

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=PurchaseReservedCacheNodesOffering
    &ReservedCacheNodesOfferingId=649fd0c8-cf6d-47a0-bfa6-060f8e75e95f
    &ReservedCacheNodeID=myreservationID
    &CacheNodeCount=1
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

자세한 내용은 참조PurchaseReservedCacheNodesOffering의 섹션을 참조하세요 ElastiCache API.

예약 노드에 대한 정보 가져오기

AWS Management Console, 및 를 사용하여 구매한 예약 노드에 대한 정보를 얻을 수 AWS CLI있습니다 ElastiCache API.

주제

예약 노드에 대한 정보 가져오기(콘솔)

다음 절차에서는 를 사용하여 구매한 예약 노드에 대한 정보를 AWS Management Console 가져오는 방법을 설명합니다.

구입한 예약 노드에 대한 정보를 가져오려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 목록에서 예약 노드(Reserved nodes) 링크를 선택합니다.

   계정의 예약 노드가 예약 노드 목록에 나타납니다. 목록에서 예약 노드를 선택하면 콘솔 아래쪽의 세부 정보 창에 예약 노드에 대한 자세한 정보가 표시됩니다.

예약 노드에 대한 정보 가져오기(AWS CLI)

AWS 계정에 예약된 노드에 대한 정보를 가져오려면 명령 프롬프트에 다음 명령을 입력합니다.

```
aws elasticache describe-reserved-cache-nodes
```

이 작업은 다음과 유사한 출력을 생성합니다(JSON 형식).

```
{
    "ReservedCacheNodeId": "myreservationid",
    "ReservedCacheNodesOfferingId": "649fd0c8-cf6d-47a0-bfa6-060f8e75e95f",
    "CacheNodeType": "cache.xx.small",
    "DataTiering": "disabled",
    "Duration": "31536000",
    "ProductDescription": "memcached",
```

```
    "OfferingType": "Medium Utilization",
    "MaxRecords": 0
}
```

자세한 내용은 AWS CLI 참조의 describe--reserved-cache-nodes를 참조하세요.

예약 노드에 대한 정보 가져오기(ElastiCache API)

AWS 계정에 예약된 노드에 대한 정보를 가져오려면 `DescribeReservedCacheNodes` 작업을 호출합니다.

Example

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeReservedCacheNodes
    &Version=2014-12-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

자세한 내용은 참조DescribeReservedCacheNodes의 섹션을 참조하세요 ElastiCache API.

이전 세대 노드 마이그레이션

이전 세대 노드는 단계적으로 제거되고 있는 노드 유형입니다. 이전 세대 노드 유형을 사용하는 기존 클러스터가 없는 경우 ElastiCache 는 해당 노드 유형을 사용하는 새 클러스터 생성을 지원하지 않습니다.

이전 세대 노드 유형의 수가 제한되어 있기 때문에 클러스터에서 노드가 비정상 상태가 될 때 성공적인 교체를 보장할 수 없습니다. 이러한 시나리오에서는 클러스터 가용성에 부정적인 영향을 줄 수 있습니다.

가용성 및 성능 향상을 위해 클러스터를 새 노드 유형으로 마이그레이션하는 것이 좋습니다. 마이그레이션할 권장 노드 유형에 대해서는 업그레이드 경로를 참조하세요. 에서 지원되는 노드 유형 및 이전 세대 노드 유형의 전체 목록은 섹션을 ElastiCache참조하세요지원되는 노드 유형.

Valkey 또는 Redis OSS 클러스터의 노드 마이그레이션

다음 절차에서는 ElastiCache 콘솔을 사용하여 Valkey 또는 Redis OSS 클러스터 노드 유형을 마이그레이션하는 방법을 설명합니다. 이 프로세스 중에 Valkey 또는 Redis OSS 클러스터는 가동 중지 시간을 최소화하면서 요청을 계속 처리합니다. 클러스터 구성에 따라 다음과 같은 가동 중지 시간이 나타날 수 있습니다. 다음은 예상치이며 특정 구성에 따라 달라질 수 있습니다.

- 클러스터 모드 비활성화됨(단일 노드)은 주로 DNS 전파로 인해 약 60초가 걸릴 수 있습니다.
- 클러스터 모드 비활성화됨(복제 노드 사용)은 Valkey 7.2 이상 또는 Redis 5.0.6 이상을 실행하는 클러스터의 경우 약 OSS 1초 정도 걸릴 수 있습니다. 모든 하위 버전에는 약 10초가 걸릴 수 있습니다.
- 클러스터 모드 활성화됨이 약 1초 동안 표시될 수 있습니다.

콘솔을 사용하여 Valkey 또는 Redis OSS 클러스터 노드 유형을 수정하려면:

1. 콘솔에 로그인하고 에서 ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 탐색 창에서 Valkey 클러스터 또는 Redis OSS 클러스터를 선택합니다.
3. 클러스터 목록에서 마이그레이션할 클러스터를 선택합니다.
4. 작업을 선택한 다음 수정을 선택합니다.
5. 노드 유형 목록에서 새 노드 유형을 선택합니다.
6. 마이그레이션 프로세스를 즉시 수행하려면 즉시 적용을 선택합니다. 즉시 적용을 선택하지 않으면 클러스터의 다음 유지 관리 기간 중에 마이그레이션션 프로세스가 수행됩니다.
7. 수정을 선택합니다. 이전 단계에서 [Apply immediately]를 선택한 경우 클러스터의 상태가 수정 중으로 변경됩니다. 상태가 사용 가능으로 변경되면 수정이 완료되고 새 클러스터의 사용을 시작할 수 있습니다.

를 사용하여 Valkey 또는 Redis OSS 클러스터 노드 유형을 수정하려면 AWS CLI:

다음과 modify-replication-group API 같이 를 사용합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-replication-group /
 --replication-group-id my-replication-group /
 --cache-node-type new-node-type /
 --apply-immediately
```

Windows의 경우:

```
aws elasticache modify-replication-group ^
 --replication-group-id my-replication-group ^
 --cache-node-type new-node-type ^
 --apply-immediately
```

이 시나리오에서 의 값은 *new-node-type* 는 마이그레이션하려는 노드 유형입니다. `--apply-immediately` 파라미터를 전달하면 복제 그룹이 수정 중에서 사용 가능 상태로 전환되는 즉시 업데이트가 적용됩니다. 즉시 적용을 선택하지 않으면 클러스터의 다음 유지 관리 기간 중에 마이그레이션션 프로세스가 수행됩니다.

> **Note**
>
> `InvalidCacheClusterState` 오류가 있는 클러스터를 수정할 수 없는 경우 먼저 복원이 실패한 노드를 제거해야 합니다.

수정 또는 제거 restore-failed-node()

다음 절차에서는 Valkey 또는 Redis OSS 클러스터에서 복원 실패 노드(들)를 수정하거나 제거하는 방법을 설명합니다. ElastiCache 노드가 복원 실패 상태로 전환되는 방법에 대한 자세한 내용은 섹션을 참조하세요 ElastiCache 노드 상태 보기. 먼저 복원 실패 상태의 노드를 제거한 다음 ElastiCache 클러스터의 나머지 이전 세대 노드를 최신 세대 노드 유형으로 마이그레이션하고 마지막으로 필요한 수의 노드를 다시 추가하는 것이 좋습니다.

복원이 실패한 노드를 제거하려면(콘솔)

1. 콘솔에 로그인하고 에서 ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 탐색 창에서 Valkey 클러스터 또는 Redis OSS 클러스터를 선택합니다.
3. 클러스터 목록에서 노드를 제거할 클러스터를 선택합니다.
4. 샤드 목록에서 노드를 제거할 샤드를 선택합니다. 클러스터에서 클러스터 모드가 비활성화된 경우 이 단계를 건너뜁니다.
5. 노드 목록에서 상태가 `restore-failed`인 노드를 선택합니다.

6. 작업을 선택하고 노드 삭제를 선택합니다.

ElastiCache 클러스터에서 복원 실패 노드(들)를 제거한 후에는 이제 최신 세대 유형으로 마이그레이션할 수 있습니다. 자세한 내용은 Valkey 또는 Redis OSS 클러스터의 노드 마이그레이션 섹션을 참조하세요.

ElastiCache 클러스터에 노드를 다시 추가하려면 섹션을 참조하세요 ElastiCache 클러스터에 노드 추가.

Memcached 클러스터에서 노드 마이그레이션

다른 노드 유형으로 마이그레이션 ElastiCache (Memcached)하려면 새 클러스터를 생성해야 합니다. 이 클러스터는 애플리케이션이 채울 수 있도록 항상 비어 있게 시작됩니다.

콘솔을 사용하여 ElastiCache ElastiCache (Memcached) 클러스터 노드 유형을 마이그레이션하려면:

- 새 노드 유형으로 새 클러스터를 생성합니다. 자세한 내용은 Memcached 클러스터 생성(콘솔) 섹션을 참조하세요.
- 애플리케이션에서 엔드포인트를 새 클러스터의 엔드포인트로 업데이트합니다. 자세한 내용은 클러스터의 엔드포인트 찾기(콘솔)(Memcached) 섹션을 참조하세요.
- 이전 클러스터를 삭제합니다. 자세한 내용은 에서 클러스터 삭제 ElastiCache 섹션을 참조하세요.

에서 클러스터 관리 ElastiCache

클러스터는 하나 이상의 캐시 노드 모음으로, 모두 Valkey, Redis OSS 또는 Memcached 엔진 소프트웨어의 인스턴스를 실행합니다. 클러스터를 만들 때 모든 노드에서 사용할 엔진과 버전을 지정합니다.

Valkey 및 Redis OSS 클러스터

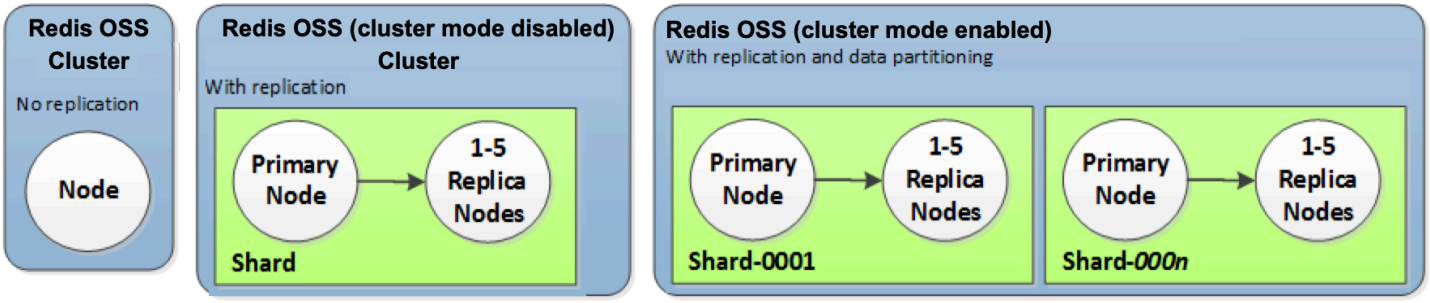
다음 다이어그램은 일반적인 Valkey 또는 Redis OSS 클러스터를 보여줍니다. 이러한 클러스터에는 단일 노드 또는 샤드(API/CLI: 노드 그룹) 내에 최대 6개의 노드가 포함될 수 있으며, 단일 노드 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터에는 샤드가 없고, 다중 노드 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터에는 단일 샤드가 있습니다. Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터에는 최대 500개의 샤드가 있을 수 있으며 데이터는 샤드를 통해 분할됩니다. 엔진 버전이 Valkey 7.2 이상 또는 Redis 5.0.6 이상인 경우 노드 또는 샤드 제한을 클러스터당 최대 OSS 500개로 늘릴 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. 일반적인 위험에는 서브넷 그룹

의 서브넷 CIDR 범위가 너무 작거나 서브넷이 공유되어 다른 클러스터에서 많이 사용됩니다. 자세한 내용은 서브넷 그룹 생성 단원을 참조하십시오. 5.0.6 이하의 버전에서 한도는 클러스터당 250개입니다.

한도 증가를 요청하려면 AWS 서비스 한도를 참조하고 한도 유형을 인스턴스 유형별 클러스터당 노드로 선택하세요.

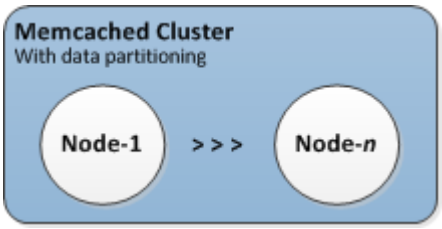
Valkey 또는 Redis OSS 샤드에 여러 노드가 있는 경우 노드 중 하나는 읽기/쓰기 기본 노드입니다. 샤드의 나머지 노드는 모두 읽기 전용 복제본입니다.

일반적인 Valkey 또는 Redis OSS 클러스터는 다음과 같습니다.



Memcached 클러스터

일반적인 Memcached 클러스터는 다음과 같습니다. Memcached 클러스터에는 1~60개의 노드가 포함되어 있으며, 이 노드를 통해 데이터를 수평으로 분할할 수 있습니다.



Valkey, Redis OSS및 Memcached에 대한 Elasticache 작업

대부분의 ElastiCache 작업은 클러스터 수준에서 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. 이 식별자는 및 AWS CLI 명령과 상호 작용할 때 특정 클러스터를 ElastiCache API 지정합니다. 클러스터 식별자는 AWS 리전의 해당 고객에 대해 고유해야 합니다.

ElastiCache 는 여러 엔진 버전을 지원합니다. 특별한 이유가 없으면 최신 버전을 사용하는 것이 좋습니다.

ElastiCache 클러스터는 Amazon EC2 인스턴스를 사용하여 액세스하도록 설계되었습니다. Amazon VPC 서비스를 기반으로 가상 프라이빗 클라우드(VPC)에서 클러스터를 시작하는 경우 외부에서 클러스터에 액세스할 수 있습니다 AWS. 자세한 내용은 외부에서 ElastiCache 리소스에 액세스 AWS 단원을 참조하십시오.

지원되는 버전 목록은 , 지원되는 Redis OSS 버전및 지원되는 엔진 및 버전섹션을 참조하세요지원되는 ElastiCache (Memcached) 버전.

에서 네트워크 유형 선택 ElastiCache

ElastiCache 는 Internet Protocol 버전 4 및 6(IPv4 및 IPv6)을 지원하므로 클러스터에서 다음을 수락하도록 구성할 수 있습니다.

- IPv4 연결만
- IPv6 연결만
- IPv4 및 IPv6 연결 모두(듀얼 스택)

IPv6 는 Nitro 시스템 에 구축된 모든 인스턴스에서 Valkey 7.2 이상 또는 Redis OSS 엔진 버전 6.2 이상을 사용하는 워크로드에 지원됩니다. 를 ElastiCache 통한 액세스에는 추가 요금이 부과되지 않습니다IPv6.

> **Note**
>
> IPV6 / 듀얼 스택의 가용성 이전에 생성된 클러스터의 마이그레이션은 지원되지 않습니다. 새로 생성된 클러스터에서의 네트워크 유형 간 전환도 지원되지 않습니다.

IPv6 는 Nitro 시스템 에 구축된 모든 인스턴스에서 Memcached 엔진 버전 1.6.6 이상을 사용하는 워크로드에 대해 지원됩니다. 를 ElastiCache 통한 액세스에는 추가 요금이 부과되지 않습니다IPv6.

네트워크 유형에 맞는 서브넷 구성

Amazon 에서 클러스터를 생성하는 경우 서브넷 그룹을 지정VPC해야 합니다. 해당 서브넷 그룹을 ElastiCache 사용하여 해당 서브넷 내에서 서브넷 및 IP 주소를 선택하여 노드와 연결해야 합니다. ElastiCache 클러스터에는 듀얼 스택 모드에서 작동하려면 IPv4 및 IPv6 주소가 할당된 듀얼 스택 서브넷이 필요하며IPv6, 전용 서브넷은 로IPv6만 작동해야 합니다.

듀얼 스택 사용

클러스터 모드가 활성화된 상태에서 ElastiCache (Redis OSS)를 사용하는 경우 애플리케이션의 관점에서 구성 엔드포인트를 통해 모든 클러스터 노드에 연결하는 것은 개별 캐시 노드에 직접 연결하는 것과 다르지 않습니다. 이를 위해 클러스터 인식 클라이언트가 클러스터 검색 프로세스에 참여하고 모든 노드에 대한 구성 정보를 요청해야 합니다. Redis의 검색 프로토콜은 노드당 하나의 IP만 지원합니다.

ElastiCache (Memcached)를 사용하여 캐시 클러스터를 생성하고 듀얼 스택을 네트워크 유형으로 선택하면 IP 검색 유형을 또는 IPv4 로 지정해야 합니다IPv6. ElastiCache 는 네트워크 유형과 IP 검색을

로 기본 설정IPv6하지만 변경할 수 있습니다. 자동 검색을 사용하는 경우, 선택된 IP 유형의 IP 주소만 Memcached 클라이언트로 반환됩니다. 자세한 내용은 클러스터의 노드 자동 식별(Memcached) 단원을 참조하십시오.

기존 모든 클라이언트와의 이전 버전과의 호환성을 유지하기 위해 IP 검색이 도입되어 검색 프로토콜에서 광고할 IP 유형(예: IPv4 또는 IPv6)을 선택할 수 있습니다. 이렇게 하면 자동 검색이 하나의 IP 유형으로 제한되지만 듀얼 스택은 다운타임 없이 에서 IPv6 Discovery IP 유형으로 마이그레이션(또는 롤백)IPv4할 수 있으므로 클러스터 모드 지원 워크로드에 여전히 유용합니다.

TLS 활성화된 듀얼 스택 ElastiCache 클러스터

TLS 가 ElastiCache 클러스터에 대해 활성화되면 Valkey 또는 Rediscluster nodes가 있는 `cluster shards`, 및 `cluster slots`OSS와 Memcached가 `config get cluster` 있는 와 같은 클러스터 검색 함수는 대신 호스트 이름을 반환합니다IPs. 그러면 호스트 이름이 ElastiCache 클러스터IPs에 연결하고 TLS 핸드셰이크를 수행하는 대신 사용됩니다. 따라서, 클라이언트가 IP Discovery 파라미터의 영향을 받지 않습니다. TLS 활성화된 클러스터의 경우 IP 검색 파라미터는 기본 IP 프로토콜에 영향을 주지 않습니다. 대신 사용되는 IP 프로토콜은 클라이언트가 DNS 호스트 이름을 확인할 때 선호하는 IP 프로토콜에 따라 결정됩니다.

DNS 호스트 이름을 확인할 때 IP 프로토콜 기본 설정을 구성하는 방법에 대한 예는 섹션을 참조하세요TLS 활성화된 듀얼 스택 ElastiCache 클러스터.

AWS Management Console (Valkey 및 RedisOSS) 사용

를 사용하여 클러스터를 생성할 때 연결 AWS Management Console에서 네트워크 유형 IPv4, IPv6 또는 듀얼 스택 을 선택합니다. Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터를 생성하고 듀얼 스택을 선택하는 경우 IPv6 또는 에서 검색 IP 유형을 선택해야 합니다IPv4.

자세한 내용은 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 생성(콘솔) 또는 Valkey 또는 Redis 생성OSS(클러스터 모드 비활성화됨)(콘솔)을 참조하세요.

를 사용하여 복제 그룹을 생성할 때 네트워크 유형IPv4인 IPv6 또는 듀얼 스택 을 AWS Management Console선택합니다. 듀얼 스택을 선택하는 경우 IPv6 또는 중 하나에서 검색 IP 유형을 선택해야 합니다IPv4.

자세한 내용은 처음부터 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹 생성 또는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨)에서 처음부터 복제 그룹 생성을 참조하세요.

AWS Management Console (Memcached) 사용

를 사용하여 캐시 클러스터를 생성할 때 연결 AWS Management Console에서 또는 IPv4 IPv6 듀얼 스택 중 하나를 선택합니다. 듀얼 스택을 선택하는 경우 IPv6 또는 중 하나에서 검색 IP 유형을 선택해야 합니다IPv4.

자세한 내용은 Memcached 클러스터 생성(콘솔) 단원을 참조하십시오.

Valkey, Redis OSS 또는 Memcached와 CLI 함께 사용

Redis OSS

를 OSS 사용하여 Valkey 또는 Redis로 캐시 클러스터를 생성할 때 create-cache-cluster 명령을 CLI사용하고 `NetworkType` 및 `IPDiscovery` 파라미터를 지정합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache create-cache-cluster \
    --cache-cluster-id "cluster-test" \
    --engine redis \
    --cache-node-type cache.m5.large \
    --num-cache-nodes 1 \
    --network-type dual_stack \
    --ip-discovery ipv4
```

Windows의 경우:

```
aws elasticache create-cache-cluster ^
    --cache-cluster-id "cluster-test" ^
    --engine redis ^
    --cache-node-type cache.m5.large ^
    --num-cache-nodes 1 ^
    --network-type dual_stack ^
    --ip-discovery ipv4
```

를 사용하여 클러스터 모드가 비활성화된 복제 그룹을 생성할 때 create-replication-group 명령을 CLI 사용하고 `NetworkType` 및 `IPDiscovery` 파라미터를 지정합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache create-replication-group \
   --replication-group-id sample-repl-group \
   --replication-group-description "demo cluster with replicas" \
   --num-cache-clusters 3 \
   --primary-cluster-id redis01 \
   --network-type dual_stack \
   --ip-discovery ipv4
```

Windows의 경우:

```
aws elasticache create-replication-group ^
   --replication-group-id sample-repl-group ^
   --replication-group-description "demo cluster with replicas" ^
   --num-cache-clusters 3 ^
   --primary-cluster-id redis01 ^
   --network-type dual_stack ^
   --ip-discovery ipv4
```

클러스터 모드가 활성화된 복제 그룹을 생성하고 를 사용하여 IP 검색IPv4에 사용할 때 create-replication-group 명령을 CLI사용하고 `NetworkType` 및 `IPDiscovery` 파라미터를 지정합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache create-replication-group \
  --replication-group-id demo-cluster \
  --replication-group-description "demo cluster" \
  --cache-node-type cache.m5.large \
  --num-node-groups 2 \
  --engine redis \
  --cache-subnet-group-name xyz \
  --network-type dual_stack \
  --ip-discovery ipv4 \
  --region us-east-1
```

Windows의 경우:

```
aws elasticache create-replication-group ^
  --replication-group-id demo-cluster ^
  --replication-group-description "demo cluster" ^
  --cache-node-type cache.m5.large ^
```

```
  --num-node-groups 2 ^
  --engine redis ^
  --cache-subnet-group-name xyz ^
  --network-type dual_stack ^
  --ip-discovery ipv4 ^
  --region us-east-1
```

클러스터 모드가 활성화된 복제 그룹을 생성하고 를 사용하여 IP 검색IPv6에 사용할 때 create-replication-group 명령을 CLI사용하고 `NetworkType` 및 `IPDiscovery` 파라미터를 지정합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache create-replication-group \
  --replication-group-id demo-cluster \
  --replication-group-description "demo cluster" \
  --cache-node-type cache.m5.large \
  --num-node-groups 2 \
  --engine redis \
  --cache-subnet-group-name xyz \
  --network-type dual_stack \
  --ip-discovery ipv6 \
  --region us-east-1
```

Windows의 경우:

```
aws elasticache create-replication-group ^
  --replication-group-id demo-cluster ^
  --replication-group-description "demo cluster" ^
  --cache-node-type cache.m5.large ^
  --num-node-groups 2 ^
  --engine redis ^
  --cache-subnet-group-name xyz ^
  --network-type dual_stack ^
  --ip-discovery ipv6 ^
  --region us-east-1
```

Memcached

를 사용하여 Memcached로 캐시 클러스터를 생성할 때 create-cache-cluster 명령을 CLI사용하고 `NetworkType` 및 `IPDiscovery` 파라미터를 지정합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache create-cache-cluster \
    --cache-cluster-id "cluster-test" \
    --engine memcached \
    --cache-node-type cache.m5.large \
    --num-cache-nodes 1 \
    --network-type dual_stack \
    --ip-discovery ipv4
```

Windows의 경우:

```
aws elasticache create-cache-cluster ^
    --cache-cluster-id "cluster-test" ^
    --engine memcached ^
    --cache-node-type cache.m5.large ^
    --num-cache-nodes 1 ^
    --network-type dual_stack ^
    --ip-discovery ipv4
```

클러스터의 노드 자동 식별(Memcached)

Memcached 엔진을 실행하는 클러스터의 경우 는 클라이언트 프로그램이 캐시 클러스터의 모든 노드를 자동으로 식별하고 이러한 모든 노드에 대한 연결을 시작하고 유지하는 기능인 Auto Discovery를 ElastiCache 지원합니다.

> **Note**
>
> Amazon ElastiCache Memcached에서 실행되는 캐시 클러스터에 Auto Discovery가 추가됩니다. Valkey 또는 Redis OSS 엔진에는 Auto Discovery를 사용할 수 없습니다.

Auto Discovery를 사용하면 애플리케이션이 개별 캐시 노드에 수동으로 연결할 필요가 없으며, 대신 애플리케이션이 Memcached 노드 하나에 연결하고 노드 목록을 검색합니다. 그 목록에서 애플리케이션은 클러스터의 나머지 노드를 인식하고 이러한 모든 노드에 연결할 수 있습니다. 애플리케이션에서 개별 캐시 노드 엔드포인트를 하드 코딩할 필요가 없습니다.

클러스터에서 듀얼 스택 네트워크 유형을 사용하는 경우 Auto Discovery는 선택한 주소에 따라 IPv4 또는 IPv6 주소만 반환합니다. 자세한 정보는 에서 네트워크 유형 선택 ElastiCache을 참조하십시오.

클러스터에 있는 모든 캐시 노드는 모든 다른 노드에 대한 메타데이터 목록을 유지 관리합니다. 이 메타데이터는 클러스터에서 노드가 추가되거나 제거될 때마다 업데이트됩니다.

주제

Memcached를 사용한 Auto Discovery의 이점

Memcached를 사용할 때 Auto Discovery는 다음과 같은 이점을 제공합니다.

- 캐시 클러스터에서 노드 개수를 늘리면 새로운 노드가 구성 Endpoint 및 모든 다른 노드를 사용하여 자신을 등록합니다. 캐시 클러스터에서 노드를 제거하면 제거되는 노드가 자신을 등록 해제합니다. 두 가지 경우 모두에서 클러스터의 다른 노드 모두는 최신 캐시 노드 메타데이터를 사용하여 업데이트됩니다.
- 캐시 노드 실패가 자동으로 감지되고 실패한 노드는 자동으로 대체됩니다.

  > **Note**
  >
  > 노드 대체가 완료될 때까지 노드가 계속 실패합니다.

- 클라이언트 프로그램은 구성 endpoint에 연결해야만 합니다. 그 후에 Auto Discovery 라이브러리가 클러스터의 다른 모든 노드에 연결됩니다.
- 클라이언트 프로그램은 분당(이 간격은 필요한 경우 조정 가능) 한 번씩 클러스터를 폴링합니다. 노드가 추가 또는 삭제되는 등의 클러스터 구성에 변경 사항이 있는 경우 클라이언트는 업데이트된 메타데이터 목록을 받습니다. 그러면 클라이언트가 필요에 따라 이 노드에 연결하거나 연결을 해제합니다.

Auto Discovery는 모든 ElastiCache Memcached 캐시 클러스터에서 활성화됩니다. 이 기능을 사용하기 위해 캐시 노드 중 어느 것도 재부팅할 필요가 없습니다.

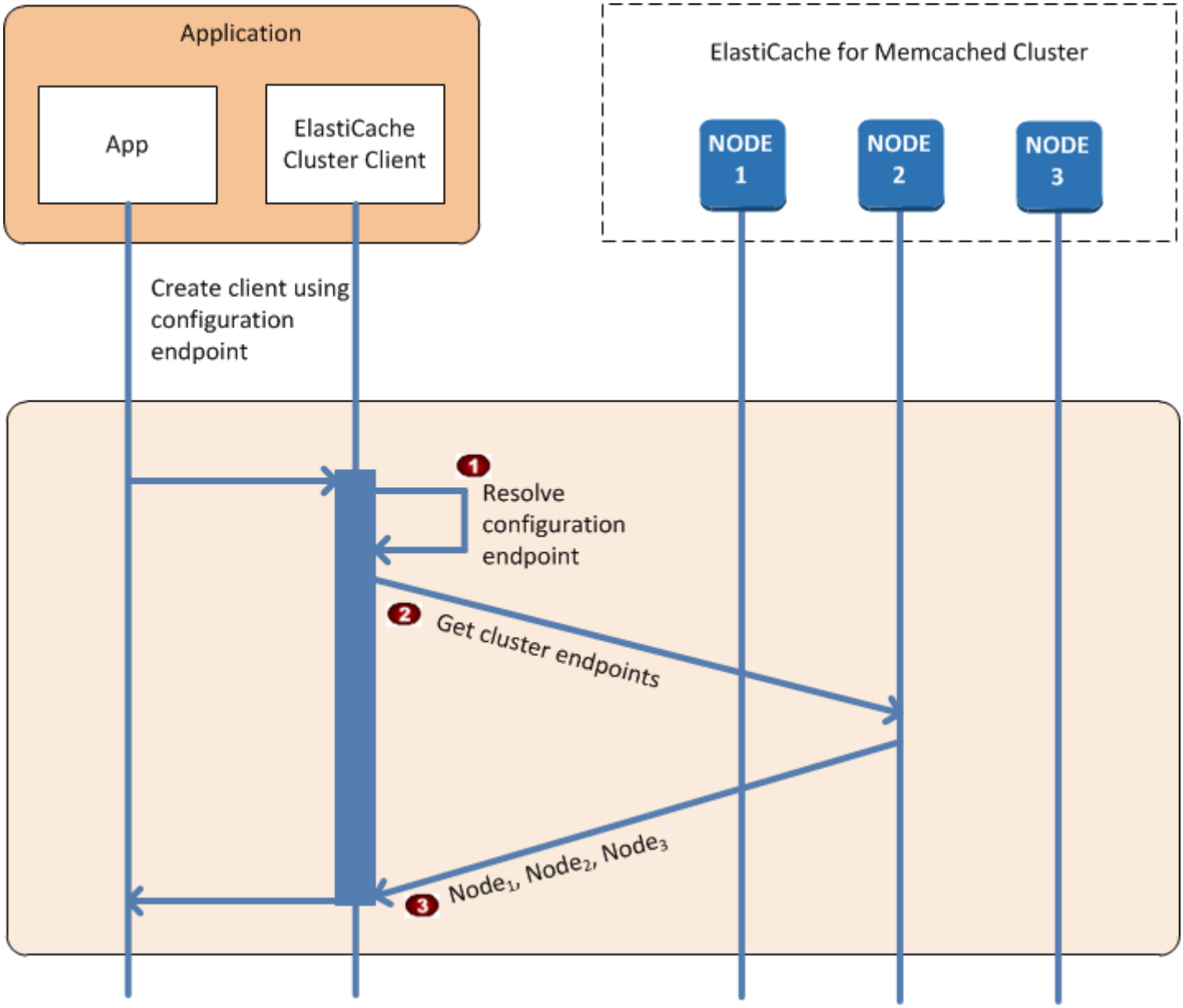
Auto Discovery 작동 방법

주제

이 섹션에서는 클라이언트 애플리케이션이 ElastiCache 클러스터 클라이언트를 사용하여 캐시 노드 연결을 관리하고 캐시의 데이터 항목과 상호 작용하는 방법을 설명합니다.

캐시 노드에 연결

애플리케이션 관점에서 클러스터 구성 엔드포인트에 대한 연결은 개별 캐시 노드에 대한 직접 연결과 다르지 않습니다. 다음 시퀀스 다이어그램은 캐시 노드에 연결하는 과정을 보여 줍니다.



캐시 노드에 연결하는 과정

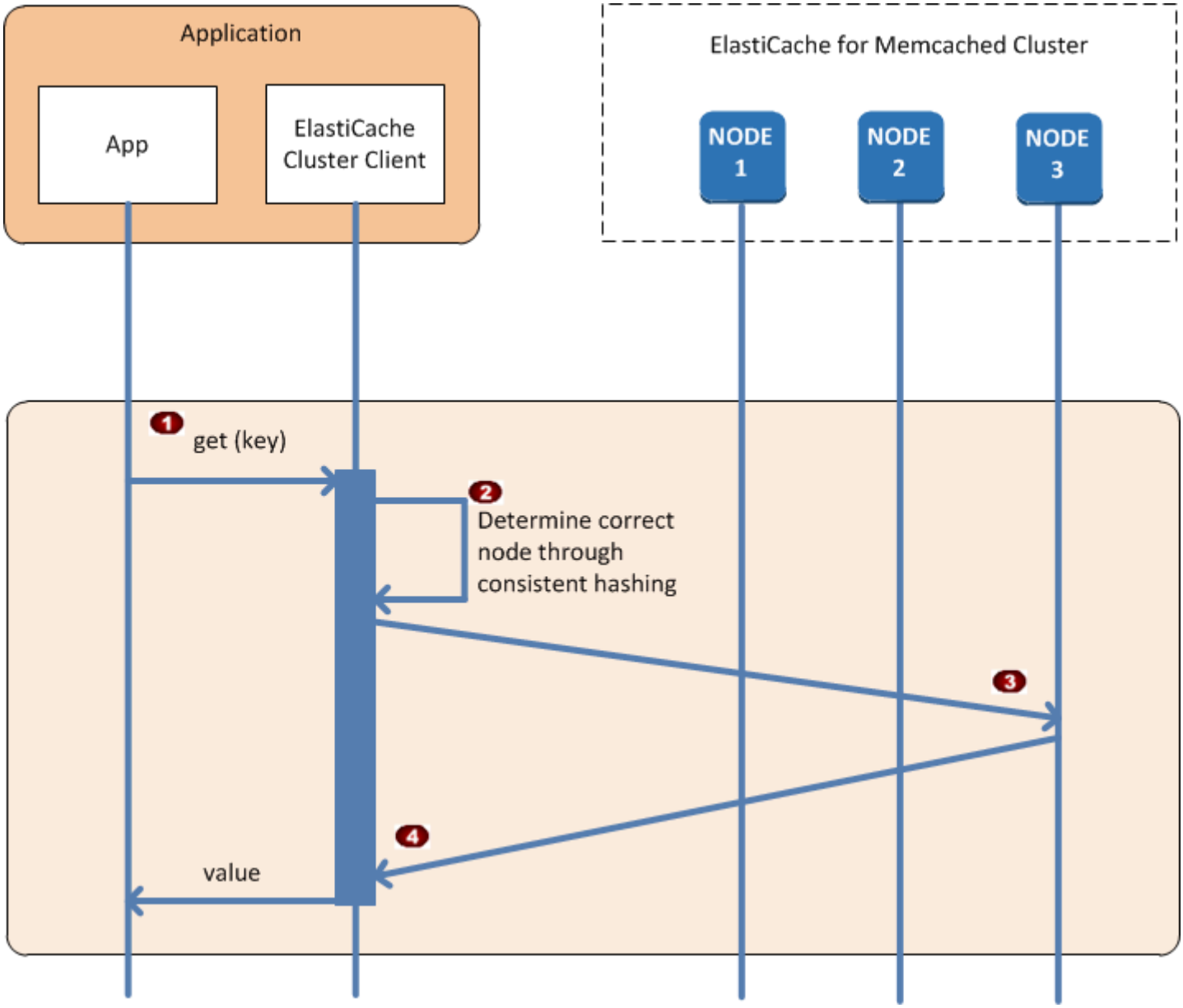
- 애플리케이션에서 구성 엔드포인트의 DNS 이름을 확인합니다. 구성 엔드포인트는 모든 캐시 노드에 대한 CNAME 항목을 유지하므로 DNS 이름은 노드 중 하나로 확인되며 클라이언트는 해당 노드에 연결할 수 있습니다.
- 클라이언트는 모든 다른 노드에 대한 구성 정보를 요청합니다. 각 노드가 클러스터에 있는 모든 노드에 대한 구성 정보를 유지 관리하므로 요청 시 모든 노드가 구성 정보를 클라이언트에 전달할 수 있습니다.

- 클라이언트는 캐시 노드 호스트 이름 및 IP 주소의 현재 목록을 받습니다. 그런 다음 클러스터에 있는 모든 다른 노드에 연결할 수 있습니다.

> **Note**
>
> 클라이언트 프로그램은 캐시 노드 호스트 이름 및 IP 주소 목록을 분당 한 번씩 새로 고칩니다. 이 폴링 간격은 필요한 경우 조정될 수 있습니다.

일반적인 클러스터 작업

애플리케이션이 모든 캐시 노드에 연결되면 ElastiCache 클러스터 클라이언트는 개별 데이터 항목을 저장해야 하는 노드와 나중에 해당 데이터 항목에 대해 쿼리해야 하는 노드를 결정합니다. 다음 시퀀스 다이어그램은 일반 클러스터 작업 과정을 보여 줍니다.



일반적인 클러스터 작업 과정

- 애플리케이션이 키로 식별되는 특정 데이터 항목에 대해 Get 요청을 실행합니다.
- 클라이언트가 키에 대해 해시 알고리즘을 사용하여 해당 데이터 항목을 포함하는 캐시 노드를 결정합니다.
- 데이터 항목이 적절한 노드에서 요청됩니다.
- 데이터 항목이 애플리케이션으로 반환됩니다.

기타 작업

일부 경우에는 클러스터의 노드를 변경할 수 있습니다. 예를 들어, 추가 수요를 수용하기 위해 노드를 추가하거나, 수요가 감소하는 기간 동안 비용을 절약하기 위해 노드를 삭제할 수 있습니다. 또는 이런저런 노드 장애로 인해 노드를 대체할 수도 있습니다.

클러스터가 변경되어 클러스터 엔드포인트로 메타데이터를 업데이트해야 하는 경우 동시에 모든 노드가 변경되므로 제공된 모든 노드의 메타데이터는 클러스터에 있는 다른 모든 노드의 메타데이터와 일치합니다.

이런 경우 클러스터의 모든 노드에서 동시에 메타데이터가 업데이트되므로 메타데이터는 항상 모든 노드에서 일치합니다. 항상 구성 엔드포인트를 사용하여 클러스터에 있는 여러 노드의 엔드포인트를 가져와야 합니다. 구성 엔드포인트를 사용하여 "사라지는" 노드에서 엔드포인트 데이터를 가져오지 않을 수 있습니다.

노드 추가

노드가 실행되는 동안 노드의 엔드포인트가 메타데이터에 포함되지 않습니다. 노드가 사용 가능 상태가 되면 각 클러스터 노드의 메타데이터에 노드가 추가됩니다. 이 시나리오에서는 모든 노드에서 메타데이터가 일치하고 노드가 사용 가능해진 후에야 새로운 노드와 상호 작용할 수 있습니다. 노드가 사용 가능 상태가 되기 전에는 노드에 대해 알 수 없으며 새로운 노드가 존재하지 않는 것처럼 클러스터에 있는 노드와 상호 작용합니다.

노드 삭제

노드가 제거되면 노드의 엔드포인트가 먼저 메타데이터에서 제거된 후 클러스터에서 노드가 제거됩니다. 이 시나리오에서는 모든 노드의 메타데이터가 일치하고 노드가 사용 가능 상태가 아닐 때 제거할 노드의 엔드포인트를 포함할 시간이 없습니다. 노드 제거 시간 중에는 메타데이터에 노드가 보고되지 않으므로 애플리케이션이 노드가 존재하지 않는 것처럼 나머지 노드 n-1개에 한해 상호 작용합니다.

노드 대체

노드가 실패하면 ElastiCache 는 해당 노드를 취소하고 대체 노드를 스핀업합니다. 대체 프로세스는 몇 분 정도 걸립니다. 이 시간 동안 모든 노트의 메타데이터가 실패한 노드의 엔드포인트를 계속 표시하지만 이 노드와 상호 작용하려는 모든 시도가 실패합니다. 따라서 로직에 항상 재시도 로직을 포함해야 합니다.

Auto Discovery 사용

Auto Discovery with ElastiCache (Memcached) 사용을 시작하려면 다음 단계를 따르세요.

- 구성 엔드포인트 가져오기
- ElastiCache 클러스터 클라이언트 다운로드
- 애플리케이션 프로그램 수정

구성 엔드포인트 가져오기

클라이언트 프로그램은 클러스터에 연결하기 위해 클러스터 구성 Endpoint를 알아야 합니다. 클러스터의 엔드포인트 찾기(콘솔)(Memcached) 항목을 참조하세요.

또한 다음과 같이 `aws elasticache describe-cache-clusters` 파라미터를 포함하여 `--show-cache-node-info` 명령을 사용할 수 있습니다.

클러스터의 엔드포인트를 찾기 위해 어떤 방법을 사용하든지 구성 엔드포인트는 항상 주소에 .cfg를 포함합니다.

Example AWS CLI 용 를 사용하여 엔드포인트 찾기 ElastiCache

Linux, macOS, Unix의 경우:

```
aws elasticache describe-cache-clusters \
    --cache-cluster-id mycluster \
    --show-cache-node-info
```

Windows의 경우:

```
aws elasticache describe-cache-clusters ^
    --cache-cluster-id mycluster ^
    --show-cache-node-info
```

이 작업은 다음과 유사한 출력을 생성합니다(JSON 형식).

```
{
    "CacheClusters": [
        {
            "Engine": "memcached",
            "CacheNodes": [
                {
```

```
                    "CacheNodeId": "0001",
                    "Endpoint": {
                        "Port": 11211,
                        "Address": "mycluster.fnjyzo.cfg.0001.use1.cache.amazonaws.com"
                    },
                    "CacheNodeStatus": "available",
                    "ParameterGroupStatus": "in-sync",
                    "CacheNodeCreateTime": "2016-10-12T21:39:28.001Z",
                    "CustomerAvailabilityZone": "us-east-1e"
                },
                {
                    "CacheNodeId": "0002",
                    "Endpoint": {
                        "Port": 11211,
                        "Address": "mycluster.fnjyzo.cfg.0002.use1.cache.amazonaws.com"
                    },
                    "CacheNodeStatus": "available",
                    "ParameterGroupStatus": "in-sync",
                    "CacheNodeCreateTime": "2016-10-12T21:39:28.001Z",
                    "CustomerAvailabilityZone": "us-east-1a"
                }
            ],
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "CacheParameterGroupName": "default.memcached1.4",
                "ParameterApplyStatus": "in-sync"
            },
            "CacheClusterId": "mycluster",
            "PreferredAvailabilityZone": "Multiple",
            "ConfigurationEndpoint": {
                "Port": 11211,
                "Address": "mycluster.fnjyzo.cfg.use1.cache.amazonaws.com"
            },
            "CacheSecurityGroups": [],
            "CacheClusterCreateTime": "2016-10-12T21:39:28.001Z",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterStatus": "available",
            "NumCacheNodes": 2,
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "CacheSubnetGroupName": "default",
            "EngineVersion": "1.4.24",
            "PendingModifiedValues": {},
            "PreferredMaintenanceWindow": "sat:06:00-sat:07:00",
```

```
            "CacheNodeType": "cache.r3.large"
        }
    ]
}
```

ElastiCache 클러스터 클라이언트 다운로드

Auto Discovery를 활용하려면 클라이언트 프로그램이 ElastiCache 클러스터 클라이언트 를 사용해야 합니다. ElastiCache 클러스터 클라이언트는 Java, PHP및 에서 사용할 수 있습니다.NET 에는 모든 캐시 노드를 검색하고 연결하는 데 필요한 모든 로직이 포함되어 있습니다.

ElastiCache 클러스터 클라이언트를 다운로드하려면

1. AWS 관리 콘솔에 로그인하고 에서 ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. ElastiCache 콘솔에서 ElastiCache 클러스터 클라이언트를 선택한 다음 다운로드를 선택합니다.

Java용 ElastiCache 클러스터 클라이언트의 소스 코드는 https://github.com/amazonwebservices/aws-elasticache-cluster-client-memcached-for-java에서 확인할 수 있습니다. 이 라이브러리는 일반적으로 많이 사용되는 Spymemcached 클라이언트를 기반으로 합니다. ElastiCache 클러스터 클라이언트는 Amazon 소프트웨어 라이선스https://aws.amazon.com/asl 에 따라 릴리스됩니다. 원하는 대로 자유롭게 소스 코드를 수정할 수 있습니다. 소스를 다른 오픈 소스 Memcached 라이브러리 또는 자체 클라이언트 코드에 통합할 수도 있습니다.

> **Note**
>
> ElastiCache 클러스터 클라이언트를 에 사용하려면 PHP먼저 Amazon EC2 인스턴스에 클러스터 클라이언트를 설치해야 합니다. 자세한 내용은 PHP용 ElastiCache 클러스터 클라이언트 설치 단원을 참조하십시오.
> TLS 지원되는 클라이언트의 경우 PHP 버전 7.4 이상의 바이너리를 다운로드합니다.
> ElastiCache 클러스터 클라이언트를 에 사용하려면 NET먼저 Amazon EC2 인스턴스에 클러스터 클라이언트를 설치해야 합니다. 자세한 내용은 용 ElastiCache 클러스터 클라이언트 설치.NET 단원을 참조하십시오.

애플리케이션 프로그램 수정

Auto Discovery를 사용하도록 애플리케이션 프로그램을 수정합니다. 다음 섹션에서는 Java, PHP및 용 ElastiCache 클러스터 클라이언트를 사용하는 방법을 보여줍니다NET.

Important

클러스터의 구성 엔드포인트를 지정할 때 다음과 같이 엔드포인트의 주소에 ".cfg"를 포함해야 합니다. CNAME 또는 엔드포인트에 “.cfg”가 없는 엔드포인트를 사용하지 마세요.

```
"mycluster.fnjyzo.cfg.use1.cache.amazonaws.com";
```

클러스터의 구성 엔드포인트를 명시적으로 지정하지 못하면 특정 노드로 구성됩니다.

Java용 ElastiCache 클러스터 클라이언트 사용

아래 프로그램은 ElastiCache 클러스터 클라이언트를 사용하여 클러스터 구성 엔드포인트에 연결하고 캐시에 데이터 항목을 추가하는 방법을 보여줍니다. 프로그램은 Auto Discovery를 사용하여 추가 개입 없이 클러스터에 있는 모든 노드에 연결합니다.

```
package com.amazon.elasticache;

import java.io.IOException;
import java.net.InetSocketAddress;

// Import the &AWS;-provided library with Auto Discovery support
import net.spy.memcached.MemcachedClient;

public class AutoDiscoveryDemo {

    public static void main(String[] args) throws IOException {

        String configEndpoint = "mycluster.fnjyzo.cfg.use1.cache.amazonaws.com";
        Integer clusterPort = 11211;

        MemcachedClient client = new MemcachedClient(
                                 new InetSocketAddress(configEndpoint,
                                                       clusterPort));
        // The client will connect to the other cache nodes automatically.

        // Store a data item for an hour.
        // The client will decide which cache host will store this item.
        client.set("theKey", 3600, "This is the data value");
    }
}
```

에 대한 ElastiCache 클러스터 클라이언트 사용 PHP

아래 프로그램은 ElastiCache 클러스터 클라이언트를 사용하여 클러스터 구성 엔드포인트에 연결하고 캐시에 데이터 항목을 추가하는 방법을 보여줍니다. 프로그램은 Auto Discovery를 사용하여 추가 개입 없이 클러스터에 있는 모든 노드에 연결합니다.

ElastiCache 클러스터 클라이언트를 에 사용하려면 PHP먼저 Amazon EC2 인스턴스에 클러스터 클라이언트를 설치해야 합니다. 자세한 내용은 PHP용 ElastiCache 클러스터 클라이언트 설치 단원을 참조하세요.

```
<?php

 /**
  * Sample PHP code to show how to integrate with the Amazon ElastiCache
  * Auto Discovery feature.
  */

  /* Configuration endpoint to use to initialize memcached client.
   * This is only an example.  */
  $server_endpoint = "mycluster.fnjyzo.cfg.use1.cache.amazonaws.com";

  /* Port for connecting to the ElastiCache cluster.
   * This is only an example  */
  $server_port = 11211;

 /**
  * The following will initialize a Memcached client to utilize the Auto Discovery
 feature.
  *
  * By configuring the client with the Dynamic client mode with single endpoint, the
  * client will periodically use the configuration endpoint to retrieve the current
 cache
  * cluster configuration. This allows scaling the cache cluster up or down in number
 of nodes
  * without requiring any changes to the PHP application.
  *
  * By default the Memcached instances are destroyed at the end of the request.
  * To create an instance that persists between requests,
  *    use persistent_id to specify a unique ID for the instance.
  * All instances created with the same persistent_id will share the same connection.
  * See http://php.net/manual/en/memcached.construct.php for more information.
  */
  $dynamic_client = new Memcached('persistent-id');
```

```
 $dynamic_client->setOption(Memcached::OPT_CLIENT_MODE,
Memcached::DYNAMIC_CLIENT_MODE);
 $dynamic_client->addServer($server_endpoint, $server_port);

 /**
 * Store the data for 60 seconds in the cluster.
 * The client will decide which cache host will store this item.
 */
 $dynamic_client->set('key', 'value', 60);


/**
 * Configuring the client with Static client mode disables the usage of Auto Discovery
 * and the client operates as it did before the introduction of Auto Discovery.
 * The user can then add a list of server endpoints.
 */
 $static_client = new Memcached('persistent-id');
 $static_client->setOption(Memcached::OPT_CLIENT_MODE, Memcached::STATIC_CLIENT_MODE);
 $static_client->addServer($server_endpoint, $server_port);

/**
 * Store the data without expiration.
 * The client will decide which cache host will store this item.
 */
 $static_client->set('key', 'value');
 ?>
```

TLS 활성화된 에서 ElastiCache 클러스터 클라이언트를 사용하는 방법에 대한 예는 PHP 및 Memcached 에서 전송 중 암호화 사용을 참조하세요.

용 ElastiCache 클러스터 클라이언트 사용.NET

> **Note**
>
> ElastiCache .NET 클러스터 클라이언트는 2022년 5월 부로 더 이상 사용되지 않습니다.

.NET 의 클라이언트 ElastiCache 는 의 오픈 소스입니다https://github.com/awslabs/elasticache-cluster-config-net.

.NET 애플리케이션은 일반적으로 구성 파일에서 구성을 가져옵니다. 다음은 샘플 애플리케이션 구성 파일입니다.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <configSections>
        <section
            name="clusterclient"
            type="Amazon.ElastiCacheCluster.ClusterConfigSettings,
 Amazon.ElastiCacheCluster" />
    </configSections>

    <clusterclient>
        <!-- the hostname and port values are from step 1 above -->
        <endpoint hostname="mycluster.fnjyzo.cfg.use1.cache.amazonaws.com"
 port="11211" />
    </clusterclient>
</configuration>
```

아래 C# 프로그램은 ElastiCache 클러스터 클라이언트를 사용하여 클러스터 구성 엔드포인트에 연결하고 캐시에 데이터 항목을 추가하는 방법을 보여줍니다. 프로그램은 Auto Discovery를 사용하여 추가 개입 없이 클러스터에 있는 모든 노드에 연결합니다.

```
// *****************
// Sample C# code to show how to integrate with the Amazon ElastiCcache Auto Discovery
 feature.

using System;

using Amazon.ElastiCacheCluster;

using Enyim.Caching;
using Enyim.Caching.Memcached;

public class DotNetAutoDiscoveryDemo  {

    public static void Main(String[] args)  {

        // instantiate a new client.
        ElastiCacheClusterConfig config = new ElastiCacheClusterConfig();
        MemcachedClient memClient = new MemcachedClient(config);

        // Store the data for 3600 seconds (1hour) in the cluster.
        // The client will decide which cache host will store this item.
        memClient.Store(StoreMode.Set, 3600, "This is the data value.");
```

```
    }  // end Main

}  // end class DotNetAutoDiscoverDemo
```

수동으로 Memached Cache 노드에 연결

클라이언트 프로그램이 Auto Discovery를 사용하지 않는 경우 각 Memcached 캐시 노드에 수동으로 연결할 수 있습니다. 이 방법이 Memcached 클라이언트의 기본 방법입니다.

AWS 관리 콘솔에서 캐시 노드 호스트 이름 및 포트 번호 목록을 확보할 수 있습니다. `--show-cache-node-info` 파라미터와 AWS CLI `aws elasticache describe-cache-clusters` 함께 명령을 사용할 수도 있습니다.

Example

다음 Java 코드 조각은 노드가 4개인 캐시 클러스터에 있는 모든 노드에 연결하는 방법을 보여 줍니다.

```
...

ArrayList<String> cacheNodes = new ArrayList<String>(
 Arrays.asList(
     "mycachecluster.fnjyzo.0001.use1.cache.amazonaws.com:11211",
     "mycachecluster.fnjyzo.0002.use1.cache.amazonaws.com:11211",
     "mycachecluster.fnjyzo.0003.use1.cache.amazonaws.com:11211",
     "mycachecluster.fnjyzo.0004.use1.cache.amazonaws.com:11211"));

MemcachedClient cache = new MemcachedClient(AddrUtil.getAddresses(cacheNodes));

...
```

Important

노드를 추가 또는 삭제하여 캐시 클러스터를 확장 또는 축소하는 경우 클라이언트 노드에서 노드 목록을 업데이트해야 합니다.

Memcached 클라이언트 라이브러리에 Auto Discovery 추가

Auto Discovery의 구성 정보는 각 Memcached 캐시 클러스터 노드에 중복 저장됩니다. 클라이언트 애플리케이션은 모든 캐시 노드를 쿼리하고 클러스터에서 모든 노드의 구성 정보를 확보할 수 있습니다.

애플리케이션이 이 작업을 수행하는 방법은 캐시 엔진 버전에 따라 달라집니다.

- 캐시 엔진 버전이 1.4.14 이상인 경우 `config` 명령을 사용합니다.
- 캐시 엔진 버전이 1.4.14 미만인 경우 `get AmazonElastiCache:cluster` 명령을 사용합니다.

이러한 두 명령의 출력은 동일하며 아래 출력 형식 섹션에서 설명됩니다.

캐시 엔진 버전 1.4.14 이상

캐시 엔진 버전 1.4.14 이상인 경우 `config` 명령을 사용합니다. 이 명령은 에 의해 Memcached ASCII 및 바이너리 프로토콜에 추가되었으며 ElastiCache 클러스터 클라이언트에서 구현 ElastiCache됩니다. 다른 클라이언트 라이브러리에 Auto Discovery를 사용하려면 해당 라이브러리가 `config` 명령을 지원하도록 확장될 필요가 있습니다.

> **Note**
>
> 다음 설명서는 ASCII 프로토콜과 관련이 있지만 `config` 명령은 ASCII 및 바이너리를 모두 지원합니다. 바이너리 프로토콜을 사용하여 Auto Discovery 지원을 추가하려면 ElastiCache 클러스터 클라이언트 의 소스 코드를 참조하세요.

구문

```
config [sub-command] [key]
```

옵션

명칭	설명	필수
`sub-command`	캐시 노드와 상호 작용하는 데 사용되는 하위 명령입니다. Auto Discovery의 경우 이 하위 명령은 `get`입니다.	예
`key`	클러스터 구성이 저장되어 있는 키입니다. Auto Discovery의 경우 이 키 이름은 `cluster`입니다.	예

명칭	설명	필수

클러스터 구성 정보를 보려면 다음 명령을 사용합니다.

```
config get cluster
```

캐시 엔진 버전 1.4.14 미만

클러스터 구성 정보를 보려면 다음 명령을 사용합니다.

```
get AmazonElastiCache:cluster
```

> **Note**
>
> 클러스터 구성 정보가 상주하므로 'AmazonElastiCache:cluster' 키를 조작하지 마세요. 이 키를 덮어쓰면 구성 정보를 ElastiCache 자동으로 올바르게 업데이트하기 전에 클라이언트가 짧은 기간(15초 이하) 동안 잘못 구성될 수 있습니다.

출력 형식

`config get cluster`를 사용하든, `get AmazonElastiCache:cluster`를 사용하든 응답은 두 줄로 구성됩니다.

- 구성 정보의 버전 번호. 매번 노드가 캐시 클러스터에서 추가 또는 제거될 때마다 버전 번호가 하나씩 증가합니다.
- 캐시 노드 목록. 목록의 각 노드는 hostname|ip-address|port 그룹으로 표현되며 각 노드는 공백으로 구분됩니다.

캐리지 리턴 및 줄 바꿈 문자(CR + LF)는 각 행의 끝에 표시됩니다. 데이터 행 끝에 줄 바꿈 문자(LF)가 포함되며 여기에 CR + LF가 추가됩니다. 구성 버전 행은 CR 없이 LF로 종결됩니다.

노드가 세 개인 캐시 클러스터는 다음과 같이 표현됩니다.

```
configversion\n
hostname|ip-address|port hostname|ip-address|port hostname|ip-address|port\n\r\n
```

각 노드는 CNAME 및 프라이빗 IP 주소와 함께 표시됩니다. CNAME 는 항상 존재합니다. 프라이빗 IP 주소를 사용할 수 없는 경우 표시되지 않지만 파이프 문자 "|"는 계속 인쇄됩니다.

Example

다음은 구성 정보를 쿼리할 때 반환되는 페이로드의 예입니다.

```
CONFIG cluster 0 136\r\n
12\n
myCluster.pc4ldq.0001.use1.cache.amazonaws.com|10.82.235.120|11211
 myCluster.pc4ldq.0002.use1.cache.amazonaws.com|10.80.249.27|11211\n\r\n
END\r\n
```

> **Note**
>
> - 두 번째 행은 구성 정보가 이제까지 12회 수정되었음을 나타냅니다.
> - 세 번째 행에서는 노드 목록이 호스트 이름의 사전순으로 지정됩니다. 이 정렬 순서는 클라이언트 애플리케이션에서 현재 사용 중인 것과 다른 순서로 되어 있을 수도 있습니다.

ElastiCache 자동 검색 기능이 있는 클라이언트

클러스터 클라이언트 프로그램은 Memcached 엔진을 실행하는 모든 캐시 클러스터 노드를 자동으로 식별하고 연결할 수 있습니다.

이 섹션에서는 자동 검색과 함께 사용할 및 .NET 클라이언트의 ElastiCache PHP 설치 및 구성에 대해 설명합니다.

주제

- 클러스터 클라이언트 설치 및 컴파일
- ElastiCache 클라이언트 구성

클러스터 클라이언트 설치 및 컴파일

이 섹션에서는 PHP 및 의 설치, 구성 및 컴파일을 다룹니다.NET Amazon ElastiCache 자동 검색 클러스터 클라이언트.

주제

- 용 ElastiCache 클러스터 클라이언트 설치.NET

- PHP용 ElastiCache 클러스터 클라이언트 설치
- 에 대한 ElastiCache 클러스터 클라이언트의 소스 코드 컴파일 PHP

용 ElastiCache 클러스터 클라이언트 설치.NET

를 찾을 수 있습니다 ElastiCache .NET 클러스터 클라이언트 코드를 의 오픈 소스로 사용합니다https://github.com/awslabs/elasticache-cluster-config-net.

이 섹션에서는 Amazon EC2 인스턴스의 클러스터 클라이언트에 대한 ElastiCache .NET 구성 요소를 설치, 업데이트 및 제거하는 방법을 설명합니다. Auto Discovery에 대한 자세한 내용은 클러스터의 노드 자동 식별(Memcached) 섹션을 참조하세요. 클라이언트를 사용하는 샘플 .NET 코드는 섹션을 참조하세요용 ElastiCache 클러스터 클라이언트 사용.NET.

주제

- 설치.NET
- 에 대한 ElastiCache .NET 클러스터 클라이언트 다운로드 ElastiCache
- 를 사용하여 AWS 어셈블리 설치 NuGet

설치.NET

가 있어야 합니다.NET 용 를 사용하도록 NET3.5 이상이 설치 AWS SDK되었습니다 ElastiCache. 가 없는 경우.NET 3.5 이상에서는 http://www.microsoft.com/net 최신 버전을 다운로드하여 설치할 수 있습니다.

에 대한 ElastiCache .NET 클러스터 클라이언트 다운로드 ElastiCache

ElastiCache .NET 클러스터 클라이언트를 다운로드하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 ElastiCache 클러스터 클라이언트 를 클릭합니다.
3. ElastiCache Memcached 클러스터 클라이언트 다운로드 목록에서 를 선택한 NET다음 다운로드 를 클릭합니다.

를 사용하여 AWS 어셈블리 설치 NuGet

NuGet 는 .NET 플랫폼의 패키지 관리 시스템입니다. NuGet 는 어셈블리 종속성을 인식하고 필요한 모든 파일을 자동으로 설치합니다. NuGet 설치된 어셈블리는 와 같은 중앙 위치가 아닌 솔루션과 함께

저장Program Files되므로 호환성 문제를 일으키지 않고 애플리케이션별 버전을 설치할 수 있습니다.

설치 NuGet

NuGet 의 설치 갤러리에서 설치할 수 있습니다. https://visualstudiogallery.msdn.microsoft.com/27077b70-9dad-4c64-adcf-c7cf6bc9970c 단원을 MSDN참조하세요. Visual Studio 2010 이상을 사용하는 경우 가 자동으로 설치 NuGet 됩니다.

Solution Explorer 또는 Package Manager 콘솔 NuGet 에서 를 사용할 수 있습니다.

Solution Explorer NuGet 에서 사용

Visual Studio 2010의 Solution Explorer NuGet 에서 사용하려면

1. [Tools] 메뉴에서 [Library Package Manager]를 선택합니다.
2. [Package Manager Console]을 클릭합니다.

Visual Studio 2012 또는 Visual Studio 2013의 Solution Explorer NuGet 에서 사용하려면

1. 도구 메뉴에서 NuGet 패키지 관리자 를 선택합니다.
2. [Package Manager Console]을 클릭합니다.

명령줄에서 다음에 표시된 대로 Install-Package를 사용하여 어셈블리를 설치할 수 있습니다.

```
Install-Package Amazon.ElastiCacheCluster
```

및 AWS.Extensions 어셈블리 NuGet와 같이 를 통해 사용할 수 있는 모든 패키지의 AWS SDK 페이지를 보려면 NuGet 웹 사이트 http://www.nuget.org 참조하세요. 각 패키지의 페이지에는 콘솔을 사용하여 패키지를 설치하기 위한 샘플 명령줄과 를 통해 사용할 수 있는 패키지의 이전 버전 목록이 포함되어 있습니다NuGet.

Package Manager Console 명령에 대한 자세한 내용은 http://nuget.codeplex.com/wikipage?title=Package%20Manager%20Console%20Command%20Reference%20%28v1.3%29를 참조하세요.

PHP용 ElastiCache 클러스터 클라이언트 설치

이 섹션에서는 Amazon EC2 인스턴스에서 ElastiCache 클러스터 클라이언트에 대한 PHP 구성 요소를 설치, 업데이트 및 제거하는 방법에 대해 설명합니다. Auto Discovery에 대한 자세한 내용은 클러스터의 노드 자동 식별(Memcached) 섹션을 참조하세요. 클라이언트를 사용할 샘플 PHP 코드는 에 대한 ElastiCache 클러스터 클라이언트 사용 PHP 섹션을 참조하세요.

주제

- 설치 패키지 다운로드
- 이미 php-memcached 확장 프로그램을 설치한 사용자의 경우
- 신규 사용자를 위한 설치 단계
- PHP 클러스터 클라이언트 제거

설치 패키지 다운로드

용 ElastiCache 클러스터 클라이언트의 올바른 버전을 사용하려면 Amazon EC2 인스턴스에 PHP 설치된 의 버전을 PHP알아야 합니다. 또한 Amazon EC2 인스턴스가 64비트 또는 32비트 Linux 버전을 실행 중인지도 알아야 합니다.

Amazon EC2 인스턴스에 설치된 PHP 버전을 확인하려면

- 명령 프롬프트에서 다음 명령을 실행합니다.

  ```
  php -v
  ```

  이 예제와 같이 출력에 PHP 버전이 표시됩니다.

  ```
  PHP 5.4.10 (cli) (built: Jan 11 2013 14:48:57)
  Copyright (c) 1997-2012 The PHP Group
  Zend Engine v2.4.0, Copyright (c) 1998-2012 Zend Technologies
  ```

  > **Note**
  >
  > PHP 및 Memcached 버전이 호환되지 않는 경우 다음과 같은 오류 메시지가 표시됩니다.
  >
  > ```
  > PHP Warning: PHP Startup: memcached: Unable to initialize module
  > Module compiled with module API=20100525
  > PHP compiled with module API=20131226
  > ```

```
These options need to match
in Unknown on line 0
```

이러한 경우 소스 코드에서 모듈을 컴파일해야 합니다. 자세한 내용은 에 대한 ElastiCache 클러스터 클라이언트의 소스 코드 컴파일 PHP 단원을 참조하십시오.

Amazon EC2 AMI 아키텍처를 확인하려면(64비트 또는 32비트)

1. 에 로그인 AWS Management Console 하고 에서 Amazon EC2 콘솔을 엽니다https://console.aws.amazon.com/ec2/.
2. 인스턴스 목록에서 Amazon EC2 인스턴스를 클릭합니다.
3. 설명 탭에서 AMI: 필드를 찾습니다. 64비트 인스턴스는 설명에 x86_64를 포함해야 하며 32비트 인스턴스는 이 필드에 i386 또는 i686를 포함합니다.

이제 ElastiCache 클러스터 클라이언트를 다운로드할 준비가 되었습니다.

에 대한 ElastiCache 클러스터 클라이언트를 다운로드하려면 PHP

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. ElastiCache 콘솔에서 ElastiCache 클러스터 클라이언트 를 선택합니다.
3. ElastiCache Memcached 클러스터 클라이언트 다운로드 목록에서 PHP 버전 및 AMI 아키텍처와 일치하는 ElastiCache 클러스터 클라이언트를 선택한 다음 다운로드 버튼을 선택합니다.

이미 php-memcached 확장 프로그램을 설치한 사용자의 경우

php-memcached 설치를 업데이트하려면

1. PHP 클러스터 클라이언트 제거 항목에 명시된 대로 이전에 설치된 PHP용 Memcached 확장명을 제거합니다.
2. 이전에 신규 사용자를 위한 설치 단계에 명시된 대로 새 ElastiCache php-memcached 확장 프로그램을 설치합니다.

신규 사용자를 위한 설치 단계

주제

- 새 사용자를 위한 PHP 7.x 설치
- 새 사용자를 위한 PHP 5.x 설치

새 사용자를 위한 PHP 7.x 설치

주제

- Ubuntu 서버 PHP 14.04LTSAMI(64비트 및 32비트)에 7을 설치하려면
- Amazon Linux PHP 201609에 7을 설치하려면 AMI
- SUSE Linux에 PHP 7을 설치하려면 AMI

Ubuntu 서버 PHP 14.04LTSAMI(64비트 및 32비트)에 7을 설치하려면

1. 에서 새 인스턴스를 시작합니다AMI.
2. 다음 명령을 실행합니다.

```
sudo apt-get update
sudo apt-get install gcc g++
```

3. 7PHP을 설치합니다.

```
sudo yum install php70
```

4. Amazon ElastiCache 클러스터 클라이언트를 다운로드합니다.

```
wget https://elasticache-downloads.s3.amazonaws.com/ClusterClient/PHP-7.0/
latest-64bit
```

5. `latest-64bit`를 추출합니다.

```
tar -zxvf latest-64bit
```

6. 루트 권한으로 추출된 아티팩트 파일 `amazon-elasticache-cluster-client.so`를 `/usr/lib/php/20151012`로 복사합니다.

```
sudo mv artifact/amazon-elasticache-cluster-client.so /usr/lib/php/20151012
```

7. `extension=amazon-elasticache-cluster-client.so` 라인을 `/etc/php/7.0/cli/php.ini` 파일에 삽입합니다.

```
echo "extension=amazon-elasticache-cluster-client.so" | sudo tee --append /etc/
php/7.0/cli/php.ini
```

8. Apache 서버를 시작하거나 다시 시작합니다.

```
sudo /etc/init.d/httpd start
```

Amazon Linux PHP 201609에 7을 설치하려면 AMI

1. 에서 새 인스턴스를 시작합니다AMI.
2. 다음 명령 실행:

```
sudo yum install gcc-c++
```

3. 7PHP을 설치합니다.

```
sudo yum install php70
```

4. Amazon ElastiCache 클러스터 클라이언트를 다운로드합니다.

```
wget https://elasticache-downloads.s3.amazonaws.com/ClusterClient/PHP-7.0/
latest-64bit
```

5. `latest-64bit`를 추출합니다.

```
tar -zxvf latest-64bit
```

6. 루트 권한으로 추출된 아티팩트 파일 `amazon-elasticache-cluster-client.so`를 `/usr/lib64/php/7.0/modules/`로 복사합니다.

```
sudo mv artifact/amazon-elasticache-cluster-client.so /usr/lib64/php/7.0/modules/
```

7. `50-memcached.ini` 파일을 생성합니다.

```
echo "extension=amazon-elasticache-cluster-client.so" | sudo tee --append /etc/
php-7.0.d/50-memcached.ini
```

8. Apache 서버를 시작하거나 다시 시작합니다.

```
sudo /etc/init.d/httpd start
```

SUSE Linux에 PHP 7을 설치하려면 AMI

1. 에서 새 인스턴스를 시작합니다AMI.
2. 다음 명령 실행:

```
sudo zypper install gcc
```

3. 7PHP을 설치합니다.

```
sudo yum install php70
```

4. Amazon ElastiCache 클러스터 클라이언트를 다운로드합니다.

```
wget https://elasticache-downloads.s3.amazonaws.com/ClusterClient/PHP-7.0/
latest-64bit
```

5. `latest-64bit`를 추출합니다.

```
tar -zxvf latest-64bit
```

6. 루트 권한으로 추출된 아티팩트 파일 `amazon-elasticache-cluster-client.so`를 `/usr/lib64/php7/extensions/`로 복사합니다.

```
sudo mv artifact/amazon-elasticache-cluster-client.so /usr/lib64/php7/extensions/
```

7. `extension=amazon-elasticache-cluster-client.so` 라인을 `/etc/php7/cli/php.ini` 파일에 삽입합니다.

```
echo "extension=amazon-elasticache-cluster-client.so" | sudo tee --append /etc/
php7/cli/php.ini
```

8. Apache 서버를 시작하거나 다시 시작합니다.

```
sudo /etc/init.d/httpd start
```

새 사용자를 위한 PHP 5.x 설치

주제

- Amazon Linux PHP 2014.03(64비트 및 32비트)에 AMI 5를 설치하려면
- Red Hat Enterprise Linux PHP 7.0AMI(64비트 및 32비트)에 5를 설치하려면
- Ubuntu 서버 PHP 14.04LTSAMI(64비트 및 32비트)에 5를 설치하려면
- SUSE Linux 엔터프라이즈 서버 PHP 11AMI(64비트 또는 32비트)용 5를 설치하려면
- 기타 Linux 배포

Amazon Linux PHP 2014.03(64비트 및 32비트)에 AMI 5를 설치하려면

1. Amazon Linux 인스턴스(64비트 또는 32비트)를 시작하여 인스턴스에 로그인합니다.
2. PHP 종속성 설치:

   ```
   sudo yum install gcc-c++ php php-pear
   ```

3. Amazon EC2 인스턴스 및 PHP 버전에 맞는 php-memcached 패키지를 다운로드합니다. 자세한 내용은 설치 패키지 다운로드 단원을 참조하십시오.
4. php-memcached을 설치합니다. 는 설치 패키지의 다운로드 경로URI여야 합니다.

   ```
   sudo pecl install <package download path>
   ```

   다음은 5.4, PHP 64비트 Linux에 대한 샘플 설치 명령입니다. 이 샘플에서는 를 바꿉니다.*X.Y.Z* 실제 버전 번호:

   ```
   sudo pecl install /home/AmazonElastiCacheClusterClient-X.Y.Z-PHP54-64bit.tgz
   ```

   > **Note**
   >
   > 최신 버전의 설치 아티팩트를 사용하세요.

5. 루트/수도 권한을 사용하여 /etc/php.d 디렉터리memcached.ini에 라는 새 파일을 추가하고 파일에 'extension=amazon-elasticache-cluster-client.so'를 삽입합니다.

```
echo "extension=amazon-elasticache-cluster-client.so" | sudo tee --append /etc/
php.d/memcached.ini
```

6. Apache 서버를 시작하거나 다시 시작합니다.

```
sudo /etc/init.d/httpd start
```

Red Hat Enterprise Linux PHP 7.0AMI(64비트 및 32비트)에 5를 설치하려면

1. Red Hat Enterprise Linux 인스턴스(64비트 또는 32비트)를 시작하여 인스턴스에 로그인합니다.
2. PHP 종속성 설치:

```
sudo yum install gcc-c++ php php-pear
```

3. Amazon EC2 인스턴스 및 PHP 버전에 맞는 php-memcached 패키지를 다운로드합니다. 자세한 내용은 설치 패키지 다운로드 단원을 참조하십시오.
4. php-memcached을 설치합니다. 는 설치 패키지의 다운로드 경로URI여야 합니다.

```
sudo pecl install <package download path>
```

5. 루트/sudo 권한으로 /etc/php.d 디렉터리에 이름이 memcached.ini인 새 파일을 추가하고 파일에 extension=amazon-elasticache-cluster-client.so를 삽입합니다.

```
echo "extension=amazon-elasticache-cluster-client.so" | sudo tee --append /etc/
php.d/memcached.ini
```

6. Apache 서버를 시작하거나 다시 시작합니다.

```
sudo /etc/init.d/httpd start
```

Ubuntu 서버 PHP 14.04LTSAMI(64비트 및 32비트)에 5를 설치하려면

1. Ubuntu Linux 인스턴스(64비트 또는 32비트)를 시작하여 인스턴스에 로그인합니다.
2. PHP 종속성 설치:

```
sudo apt-get update
sudo apt-get install gcc g++ php5 php-pear
```

3. Amazon EC2 인스턴스 및 PHP 버전에 맞는 php-memcached 패키지를 다운로드합니다. 자세한 내용은 설치 패키지 다운로드 단원을 참조하십시오.
4. php-memcached을 설치합니다. 는 설치 패키지의 다운로드 경로여야 URI 합니다.

```
sudo pecl install <package download path>
```

> **Note**
>
> 이 설치 단계에서는 빌드 아티팩트 amazon-elasticache-cluster-client.so를 /usr/lib/php5/20121212* 디렉터리에 설치합니다. 다음 단계에서 필요한 빌드 아티팩트의 절대 경로를 확인하세요.

이전 명령이 작동하지 않는 경우 다운로드한 *.tgz 파일amazon-elasticache-cluster-client.so에서 PHP 클라이언트 아티팩트를 수동으로 추출하여 /usr/lib/php5/20121212* 디렉터리에 복사해야 합니다.

```
tar -xvf <package download path>
cp amazon-elasticache-cluster-client.so /usr/lib/php5/20121212/
```

5. 루트/수도 권한을 사용하여 /etc/php5/cli/conf.d 디렉터리memcached.ini에 라는 새 파일을 추가하고 파일에 'extension=<abso amazon-elasticache-cluster-client>' 경로를 삽입합니다.

```
echo "extension=<absolute path to amazon-elasticache-cluster-client.so>" | sudo tee
 --append /etc/php5/cli/conf.d/memcached.ini
```

6. Apache 서버를 시작하거나 다시 시작합니다.

```
sudo /etc/init.d/httpd start
```

SUSE Linux 엔터프라이즈 서버 PHP 11AMI(64비트 또는 32비트)용 5를 설치하려면

1. SUSE Linux 인스턴스(64비트 또는 32비트)를 시작하고 로그인합니다.
2. PHP 종속성 설치:

```
sudo zypper install gcc php53-devel
```

3. Amazon EC2 인스턴스 및 PHP 버전에 맞는 php-memcached 패키지를 다운로드합니다. 자세한 내용은 설치 패키지 다운로드 단원을 참조하십시오.
4. php-memcached을 설치합니다. 는 설치 패키지의 다운로드 경로여야 URI 합니다.

```
sudo pecl install <package download path>
```

5. 루트/sudo 권한으로 /etc/php5/conf.d 디렉터리에 이름이 memcached.ini인 새 파일을 추가하고 파일에 **extension=amazon-elasticache-cluster-client.so**를 삽입합니다.

```
echo "extension=amazon-elasticache-cluster-client.so" | sudo tee --append /etc/
php5/conf.d/memcached.ini
```

6. Apache 서버를 시작하거나 다시 시작합니다.

```
sudo /etc/init.d/httpd start
```

> **Note**
>
> 이전의 모든 플랫폼에 대해 5단계가 작동하지 않으면 amazon-elasticache-cluster-client.so의 설치 경로를 확인하고 확장명에 바이너리의 전체 경로를 지정하세요. 또한 사용 PHP 중인 이 지원되는 버전인지 확인합니다. 5.3~5.5의 버전이 지원됩니다.

기타 Linux 배포

일부 시스템에서는 특히 CentOS7 및 Red Hat Enterprise Linux(RHEL) 7.1libsasl2.so.3이를 대체했습니다libsasl2.so.2. 이러한 시스템에서 ElastiCache 는 클러스터 클라이언트를 로드할 때 를 찾아 로드하려고 시도하고 실패합니다libsasl2.so.2. 이 문제를 해결하려면

libsasl2.so.3에 대한 심볼 링크를 생성합니다. 그러면 클라이언트가 libsasl2.so.2를 로드하려고 시도할 때 libsasl2.so.3으로 리디렉션됩니다. 다음 코드는 이 심볼 링크를 생성합니다.

```
cd /usr/lib64
sudo ln libsasl2.so.3 libsasl2.so.2
```

PHP 클러스터 클라이언트 제거

주제

- 이전 버전의 PHP 7 제거
- 이전 버전의 PHP 5 제거

이전 버전의 PHP 7 제거

이전 버전의 PHP 7을 제거하려면

1. 설치 지침에 앞서 설명한 대로 적절한 PHP lib 디렉터리에서 amazon-elasticache-cluster-client.so 파일을 제거합니다. 이미 php-memcached 확장 프로그램을 설치한 사용자의 경우의 설치에 대한 섹션을 참조하세요.
2. php.ini 파일에서 extension=amazon-elasticache-cluster-client.so 라인을 제거합니다.
3. Apache 서버를 시작하거나 다시 시작합니다.

   ```
   sudo /etc/init.d/httpd start
   ```

이전 버전의 PHP 5 제거

이전 버전의 PHP 5를 제거하려면

1. php-memcached 확장 프로그램을 삭제합니다.

   ```
   sudo pecl uninstall __uri/AmazonElastiCacheClusterClient
   ```

2. 이전 설치 단계에 표시된 대로 해당하는 디렉터리에 추가된 memcached.ini 파일을 제거합니다.

에 대한 ElastiCache 클러스터 클라이언트의 소스 코드 컴파일 PHP

이 섹션에서는 용 ElastiCache 클러스터 클라이언트의 소스 코드를 가져오고 컴파일하는 방법을 다룹니다PHP.

-aws-elasticache-cluster-clientlibmemcached GitHub 및 -라는 두 가지 패키지를 가져와 컴파일해야 합니다aws-elasticache-cluster-clientmemcached-for-php.

주제

- libmemcached 라이브러리 컴파일
- 에 대한 ElastiCache Memcached 자동 검색 클라이언트 컴파일 PHP

libmemcached 라이브러리 컴파일

aws-elasticache-cluster-client-libmemcached 라이브러리를 컴파일하려면

1. Amazon EC2 인스턴스를 시작합니다.
2. 라이브러리 종속 항목을 설치합니다.
   - Amazon Linux 201509 AMI

     ```
     sudo yum install gcc gcc-c++ autoconf libevent-devel
     ```

   - Ubuntu 14.04에서 AMI

     ```
     sudo apt-get update
     sudo apt-get install libevent-dev gcc g++ make autoconf libsasl2-dev
     ```

3. 리포지토리를 가져오고 코드를 컴파일합니다.

   ```
   Download and install  https://github.com/awslabs/aws-elasticache-cluster-client-
   libmemcached/archive/v1.0.18.tar.gz
   ```

에 대한 ElastiCache Memcached 자동 검색 클라이언트 컴파일 PHP

다음 섹션에서는 ElastiCache Memcached Auto Discovery Client를 컴파일하는 방법을 설명합니다.

주제

- PHP 7에 대한 ElastiCache Memcached 클라이언트 컴파일

- PHP 5에 대한 ElastiCache Memcached 클라이언트 컴파일

PHP 7에 대한 ElastiCache Memcached 클라이언트 컴파일

코드 디렉터리에서 다음 명령 세트를 실행합니다.

```
git clone https://github.com/awslabs/aws-elasticache-cluster-client-memcached-for-
php.git
cd aws-elasticache-cluster-client-memcached-for-php
git checkout php7
sudo yum install php70-devel
phpize
./configure --with-libmemcached-dir=<libmemcached-install-directory> --disable-
memcached-sasl
make
make install
```

> **Note**
>
> libmemcached 라이브러리를 PHP 바이너리에 고정적으로 연결하여 다양한 Linux 플랫폼에 포팅할 수 있습니다. 이렇게 하려면 make 전에 다음 명령을 실행하세요.
>
> ```
> sed -i "s#-lmemcached#<libmemcached-install-directory>/lib/libmemcached.a -
> lcrypt -lpthread -lm -lstdc++ -lsasl2#" Makefile
> ```

PHP 5에 대한 ElastiCache Memcached 클라이언트 컴파일

aws-elasticache-cluster-client-memcached-for-php/ 폴더에서 다음 명령을 실행하여 aws-elasticache-cluster-client-memcached-for-php를 컴파일합니다.

```
git clone https://github.com/awslabs/aws-elasticache-cluster-client-memcached-for-
php.git
cd aws-elasticache-cluster-client-memcached-for-php
sudo yum install zlib-devel
phpize
./configure --with-libmemcached-dir=<libmemcached-install-directory>
make
make install
```

ElastiCache 클라이언트 구성

ElastiCache 클러스터는 프로토콜에 따라 Valkey, Redis OSS및 Memcached를 준수합니다. 현재 기존 환경에서 사용하는 코드, 애플리케이션 및 가장 인기 있는 도구는 서비스와 원활하게 작동합니다.

이 섹션에서는 의 캐시 노드에 연결하기 위한 특정 고려 사항에 대해 설명합니다 ElastiCache.

주제

제한된 명령

관리형 서비스 경험을 제공하기 위해 는 고급 권한이 필요한 특정 캐시 엔진별 명령에 대한 액세스를 ElastiCache 제한합니다. Redis를 실행하는 캐시 클러스터에서는 다음 명령을 사용할 수 없습니다.

- `bgrewriteaof`
- `bgsave`
- `config`
- `debug`
- `migrate`
- `replicaof`
- `save`
- `slaveof`
- `shutdown`
- `sync`

노드 엔드포인트 및 포트 번호 찾기

캐시 노드에 연결하려는 애플리케이션은 해당 노드의 Endpoint 및 포트 번호를 알아야 합니다.

노드 엔드포인트 및 포트 번호 찾기(콘솔)

노드 엔드포인트 및 포트 번호를 확인하려면

1. Amazon ElastiCache 관리 콘솔에 로그인하고 클러스터에서 실행되는 엔진을 선택합니다.

   선택한 엔진을 실행하고 있는 모든 클러스터의 목록이 표시됩니다.
2. 실행 중인 엔진 및 구성에 대해 다음을 계속합니다.
3. 관심 있는 클러스터의 이름을 선택합니다.
4. 관심이 있는 노드에 대한 [Port] 및 [Endpoint] 열을 찾습니다.

캐시 노드 엔드포인트 및 포트 번호 찾기(AWS CLI)

캐시 노드 엔드포인트 및 포트 번호를 확인하려면 `describe-cache-clusters` 명령을 `--show-cache-node-info` 파라미터와 함께 사용하세요.

```
aws elasticache describe-cache-clusters --show-cache-node-info
```

정규화된 DNS 이름과 포트 번호는 출력의 엔드포인트 섹션에 있습니다.

캐시 노드 엔드포인트 및 포트 번호 찾기(ElastiCache API)

캐시 노드 엔드포인트 및 포트 번호를 확인하려면 `DescribeCacheClusters` 작업을 `ShowCacheNodeInfo=true` 파라미터와 함께 사용하세요.

Example

```
https://elasticache.us-west-2.amazonaws.com /
    ?Action=DescribeCacheClusters
    &ShowCacheNodeInfo=true
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20140421T220302Z
    &Version=2014-09-30
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Credential=<credential>
    &X-Amz-Date=20140421T220302Z
    &X-Amz-Expires=20140421T220302Z
    &X-Amz-Signature=<signature>
    &X-Amz-SignedHeaders=Host
```

자동 검색 사용을 위한 연결

애플리케이션이 Auto Discovery를 사용하는 경우에는 각 캐시 노드에 대한 개별 Endpoint가 아니라 클러스터에 대한 구성 Endpoint만 알면 됩니다. 자세한 내용은 클러스터의 노드 자동 식별(Memcached) 단원을 참조하십시오.

> **Note**
>
> 이때, Auto Discovery는 Memcached를 실행하는 캐시 클러스터에 대해서만 사용할 수 있습니다.

Valkey 또는 Redis OSS 클러스터의 노드에 연결

> **Note**
>
> 현재 복제 및 읽기 전용 복제본을 지원하는 클러스터(API/CLI: 복제 그룹)는 Valkey 또는 Redis를 실행하는 클러스터에서만 지원됩니다OSS.

클러스터의 경우 콘솔, CLI및 API 인터페이스를 ElastiCache 제공하여 개별 노드에 대한 연결 정보를 가져옵니다.

읽기 전용 활동의 경우 애플리케이션은 클러스터의 모든 노드에 연결할 수 있습니다. 하지만 쓰기 활동의 경우 애플리케이션이 노드에 직접 연결하는 대신 클러스터의 기본 엔드포인트(Valkey 또는 RedisOSS(클러스터 모드 비활성화됨)) 또는 구성 엔드포인트(Valkey 또는 RedisOSS(클러스터 모드 활성화됨))에 연결하는 것이 좋습니다. 이렇게 하면 읽기 전용 복제본을 기본 역할로 승격하여 클러스터를 재구성할 경우에도 애플리케이션이 항상 올바른 노드를 찾을 수 있습니다.

복제 그룹의 클러스터에 연결(콘솔)

Endpoint 및 포트 번호를 확인하려면

- Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터의 엔드포인트 찾기(콘솔) 항목을 참조하세요.

복제 그룹의 클러스터에 연결(AWS CLI)

캐시 노드 Endpoint 및 포트 번호를 확인하려면

describe-replication-groups 명령을 복제 그룹 이름과 함께 사용합니다.

```
aws elasticache describe-replication-groups redis2x2
```

이 명령은 다음과 유사한 출력을 생성합니다.

```
{
    "ReplicationGroups": [
        {
            "Status": "available",
            "Description": "2 shards, 2 nodes (1 + 1 replica)",
            "NodeGroups": [
                {
                    "Status": "available",
                    "Slots": "0-8191",
                    "NodeGroupId": "0001",
                    "NodeGroupMembers": [
                        {
                            "PreferredAvailabilityZone": "us-west-2c",
                            "CacheNodeId": "0001",
                            "CacheClusterId": "redis2x2-0001-001"
                        },
                        {
                            "PreferredAvailabilityZone": "us-west-2a",
                            "CacheNodeId": "0001",
                            "CacheClusterId": "redis2x2-0001-002"
                        }
                    ]
                },
                {
                    "Status": "available",
                    "Slots": "8192-16383",
                    "NodeGroupId": "0002",
                    "NodeGroupMembers": [
                        {
                            "PreferredAvailabilityZone": "us-west-2b",
                            "CacheNodeId": "0001",
                            "CacheClusterId": "redis2x2-0002-001"
                        },
                        {
                            "PreferredAvailabilityZone": "us-west-2a",
                            "CacheNodeId": "0001",
                            "CacheClusterId": "redis2x2-0002-002"
```

```
                        }
                    ]
                }
            ],
            "ConfigurationEndpoint": {
                "Port": 6379,
                "Address": "redis2x2.9dcv5r.clustercfg.usw2.cache.amazonaws.com"
            },
            "ClusterEnabled": true,
            "ReplicationGroupId": "redis2x2",
            "SnapshotRetentionLimit": 1,
            "AutomaticFailover": "enabled",
            "SnapshotWindow": "13:00-14:00",
            "MemberClusters": [
                "redis2x2-0001-001",
                "redis2x2-0001-002",
                "redis2x2-0002-001",
                "redis2x2-0002-002"
            ],
            "CacheNodeType": "cache.m3.medium",
            "PendingModifiedValues": {}
        }
    ]
}
```

복제 그룹의 클러스터에 연결(ElastiCache API)

캐시 노드 Endpoint 및 포트 번호를 확인하려면

`DescribeReplicationGroups`를 다음 파라미터를 사용하여 호출합니다.

`ReplicationGroupId` = 복제 그룹 이름

Example

```
https://elasticache.us-west-2.amazonaws.com /
    ?Action=DescribeCacheClusters
    &ReplicationGroupId=repgroup01
    &Version=2014-09-30
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20140421T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20140421T220302Z
```

```
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20140421T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

DNS 이름 및 기본 IP

클라이언트는 캐시 데이터를 보관 중인 서버의 주소 및 포트 번호를 포함하는 서버 목록을 유지 관리합니다. ElastiCache를 DescribeCacheClusters API 사용할 때 (또는 describe-cache-clusters 명령줄 유틸리티)는 서버 목록에 사용할 수 있는 정규화된 DNS 항목 및 포트 번호를 반환합니다.

> **Important**
>
> 캐시 노드 엔드포인트에 연결을 시도할 때 캐시 노드의 DNS 이름을 자주 확인하도록 클라이언트 애플리케이션을 구성하는 것이 중요합니다.

VPC 설치

ElastiCache 는 실패 시 캐시 노드가 복구될 때 캐시 노드의 DNS 이름과 IP 주소가 모두 동일하게 유지되도록 합니다.

비VPC설치

ElastiCache 는 실패 시 캐시 노드가 복구될 때 캐시 노드의 DNS 이름이 변경되지 않도록 하지만 캐시 노드의 기본 IP 주소가 변경될 수 있습니다.

클라이언트 라이브러리는 대부분 기본적으로 영구 캐시 노드 연결을 지원합니다. 를 사용할 때는 영구 캐시 노드 연결을 사용하는 것이 좋습니다 ElastiCache. 클라이언트 측 DNS 캐싱은 클라이언트 라이브러리, 언어 런타임 또는 클라이언트 운영 체제를 비롯한 여러 위치에서 발생할 수 있습니다. 캐시 노드의 IP 주소를 자주 확인하도록 하려면 각 계층의 애플리케이션 구성을 검토해야 합니다.

의 데이터 계층화 ElastiCache

ElastiCache 복제 그룹을 구성하고 r6gd 패밀리의 노드 유형을 사용하는 Valkey 또는 Redis OSS 클러스터를 사용하면 메모리와 로컬SSD(솔리드 스테이트 드라이브) 스토리지 간에 데이터가 계층화됩니다. 데이터 계층화는 메모리에 데이터를 저장하는 것 외에도 각 클러스터 노드의 저렴한 솔리드 스테이트 드라이브(SSDs)를 활용하여 Valkey 또는 Redis OSS 워크로드에 대한 새로운 가격 대비 성능 옵션을 제공합니다. 전체 데이터 세트의 최대 20%에 정기적으로 액세스하는 워크로드와 에서 데이터에 액세스할 때 추가 지연 시간을 견딜 수 있는 애플리케이션에 적합합니다SSD.

데이터 계층화가 있는 ElastiCache 클러스터에서 는 저장하는 모든 항목의 마지막 액세스 시간을 ElastiCache 모니터링합니다. 사용 가능한 메모리(DRAM)가 완전히 소모되면 ElastiCache는 최근에 가장 적게 사용되는 (LRU) 알고리즘을 사용하여 자주 액세스하지 않는 항목을 메모리에서 로 자동으로 이동합니다SSD. 이후에 SSD 의 데이터에 액세스하면 요청을 처리하기 전에 ElastiCache 자동 및 비동기적으로 데이터를 메모리로 다시 이동합니다. 데이터의 하위 집합에만 정기적으로 액세스하는 워크로드가 있는 경우 데이터 계층화는 용량을 비용 효율적으로 확장할 수 있는 최적의 방법입니다.

데이터 계층화를 사용하는 경우 키 자체는 항상 메모리에 남아 있는 반면 는 메모리 대 디스크에 값 배치를 LRU 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 메모리에 저장된 데이터에 대한 요청과 SSD 비교하여 에 저장된 데이터에 대한 요청의 지연 시간이 평균 300마이크로초 더 걸릴 수 있습니다.

가장 큰 데이터 계층화 노드 크기(cache.r6gd.16xlarge)를 사용하면 단일 500노드 클러스터에 최대 1페타바이트까지 저장할 수 있습니다(읽기 전용 복제본 1개를 사용하는 경우는 500TB). 데이터 계층화는 에서 지원되는 모든 Valkey 또는 Redis OSS 명령 및 데이터 구조와 호환됩니다 ElastiCache. 이 기능을 사용하려면 클라이언트 측 변경 사항이 필요하지 않습니다.

주제

모범 사례

다음 모범 사례를 따르는 것이 좋습니다.

- 데이터 계층화는 전체 데이터 세트의 최대 20%에 정기적으로 액세스하는 워크로드와 에서 데이터에 액세스할 때 추가 지연 시간을 견딜 수 있는 애플리케이션에 적합합니다SSD.
- 데이터 계층 노드에서 사용 가능한 SSD 용량을 사용할 때는 값 크기가 키 크기보다 큰 것이 좋습니다. 항목이 DRAM 및 사이에서 이동되면 SSD키는 항상 메모리에 남아 있으며 값만 SSD 계층으로 이동합니다.

제한 사항

데이터 계층화에는 다음과 같은 제한 사항이 있습니다.

- 복제 그룹에 속하는 클러스터에서만 데이터 계층화를 사용할 수 있습니다.
- 사용하는 노드 유형은 `us-east-2`, `us-east-1`, `us-west-2`, `us-west-1`, `eu-west-1`, `eu-central-1`, `eu-north-1`, `eu-west-3`, `ap-northeast-1`, `ap-southeast-1`, `ap-southeast-2`, `ap-south-1`, `ca-central-1` 및 `sa-east-1`과 같은 리전에서 사용할 수 있는 r6gd 패밀리의 노드 유형이어야 합니다.
- Valkey 7.2 이상 또는 Redis OSS 6.2 이상의 엔진을 사용해야 합니다.
- r6gd 클러스터를 사용하지 않는 한 r6gd 클러스터의 백업을 다른 클러스터로 복원할 수 없습니다.
- 데이터 계층화 클러스터를 위해 백업을 Amazon S3로 내보낼 수 없습니다.
- r6gd 노드 유형에서 실행되는 클러스터에는 온라인 마이그레이션이 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. 자세한 내용은 크기 조정 ElastiCache 단원을 참조하십시오.
- Auto Scaling은 Valkey 버전 7.2 이상 및 Redis OSS 버전 7.0.7 이상의 데이터 계층화를 사용하는 클러스터에서 지원됩니다. 자세한 내용은 Auto Scaling Valkey 및 Redis OSS 클러스터 단원을 참조하세요.
- 데이터 계층화는 `volatile-lru`, `allkeys-lru`, `volatile-lfu`, `allkeys-lfu` 및 `noeviction` 메모리 사용량 제한 정책만 지원합니다.
- Forkless 저장은 Valkey 버전 7.2 이상 및 Redis OSS 버전 7.0.7 이상에서 지원됩니다. 자세한 내용은 동기화 및 백업 구현 방법 단원을 참조하십시오.
- 128MiB보다 큰 항목은 로 이동되지 않습니다SSD.

요금

R6gd 노드는 총 용량(메모리 + SSD)의 4.8배가 더 많으며 R6g 노드(메모리만 해당)에 비해 최대 사용률로 실행 시 60% 이상 절감할 수 있습니다. 자세한 내용은 ElastiCache 요금 섹션을 참조하세요.

모니터링

ElastiCache 는 데이터 계층화를 사용하는 성능 클러스터를 모니터링하도록 특별히 설계된 지표를 제공합니다. 와 DRAM 비교하여 의 항목 비율을 모니터링하려면 Valkey 및 Redis `CurrItems` 에 대한 지표에서 지표를 사용할 SSD수 있습니다. OSS 백분율은 (차원: 계층 = 메모리 * CurrItems 100) / (차원 필터 CurrItems 없음)로 계산할 수 있습니다.

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

메모리의 항목 비율이 5% 미만으로 감소할 경우 클러스터 모드 활성화 클러스터에 대해 스케일 아웃을 고려하거나 클러스터 모드 비활성화 클러스터에 대해 스케일 업을 고려하는 것이 좋습니다. 크기 조정에 대한 자세한 내용은 섹션을 참조하세요Valkey 또는 Redis에서 클러스터 크기 조정OSS(클러스터 모드 활성화됨). 데이터 계층화를 사용하는 Valkey 또는 Redis OSS 클러스터의 지표에 대한 자세한 내용은 섹션을 참조하세요Valkey 및 Redis에 대한 지표 OSS.

데이터 계층화 사용

를 사용한 데이터 계층화 사용 AWS Management Console

복제 그룹에 속하는 클러스터를 생성할 때 cache.r6gd.xlarge와 같은 r6gd 패밀리의 노드 유형을 선택하여 데이터 계층화를 사용합니다. 해당 노드 유형을 선택하면 데이터 계층화가 자동으로 사용됩니다.

클러스터 생성에 대한 자세한 내용은 Valkey 또는 Redis용 클러스터 생성 OSS 섹션을 참조하세요.

를 사용하여 데이터 계층화 활성화 AWS CLI

를 사용하여 복제 그룹을 생성할 때 cache.r6gd.xlarge와 같은 r6gd 패밀리에서 노드 유형을 선택하고 `--data-tiering-enabled` 파라미터를 설정하여 데이터 계층화를 AWS CLI사용합니다.

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

Linux, macOS, Unix의 경우:

```
aws elasticache create-replication-group \
   --replication-group-id redis-dt-cluster \
   --replication-group-description "Redis OSS cluster with data tiering" \
   --num-node-groups 1 \
   --replicas-per-node-group 1 \
   --cache-node-type cache.r6gd.xlarge \
   --engine redis \
   --cache-subnet-group-name default \
   --automatic-failover-enabled \
   --data-tiering-enabled
```

Windows의 경우:

```
aws elasticache create-replication-group ^
   --replication-group-id redis-dt-cluster ^
```

```
    --replication-group-description "Redis OSS cluster with data tiering" ^
    --num-node-groups 1 ^
    --replicas-per-node-group 1 ^
    --cache-node-type cache.r6gd.xlarge ^
    --engine redis ^
    --cache-subnet-group-name default ^
    --automatic-failover-enabled ^
    --data-tiering-enabled
```

이 작업을 실행하면 다음과 유사한 응답이 표시됩니다.

```
{
    "ReplicationGroup": {
        "ReplicationGroupId": "redis-dt-cluster",
        "Description": "Redis OSS cluster with data tiering",
        "Status": "creating",
        "PendingModifiedValues": {},
        "MemberClusters": [
            "redis-dt-cluster"
        ],
        "AutomaticFailover": "enabled",
        "DataTiering": "enabled",
        "SnapshotRetentionLimit": 0,
        "SnapshotWindow": "06:00-07:00",
        "ClusterEnabled": false,
        "CacheNodeType": "cache.r6gd.xlarge",
        "TransitEncryptionEnabled": false,
        "AtRestEncryptionEnabled": false
    }
}
```

데이터 계층화가 활성화된 상태에서 백업에서 클러스터로 데이터 복원

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

데이터 계층화가 활성화된 상태로 백업에서 클러스터로 데이터 복원(콘솔)

데이터 계층화가 활성화된 새 클러스터로 백업을 복원하려면(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.

2. 탐색 창에서 [Backups]를 선택합니다.
3. 백업 목록에서 복원할 백업의 이름 왼쪽에 있는 상자를 선택합니다.
4. 복원을 선택합니다.
5. [Restore Cluster] 대화 상자를 완료합니다. 모든 필수 필드와 기본값을 변경하려는 다른 필드를 완료해야 합니다.

    1. 클러스터 ID - 필수입니다. 새 클러스터의 이름입니다.
    2. 클러스터 모드 활성화(스케일 아웃) - Valkey 또는 RedisOSS(클러스터 모드 활성화) 클러스터에 대해 이 옵션을 선택합니다.
    3. 노드 유형 - cache.r6gd.xlarge 또는 r6gd 패밀리의 다른 노드 유형을 지정합니다.
    4. 샤드 수 - 새 클러스터(API/CLI: 노드 그룹)에서 원하는 샤드 수를 선택합니다.
    5. 샤드 당 복제본 - 각 샤드에 포함할 읽기 전용 복제본 노드 수를 선택합니다.
    6. 슬롯 및 키스페이스 - 샤드에 키를 배포할 방법을 선택합니다. 키 배포를 지정하도록 선택할 경우 각 샤드의 키 범위를 지정하는 표를 완료합니다.
    7. 가용 영역 - 클러스터의 가용 영역 선택 방법을 지정합니다.
    8. 포트 - 이 클러스터에 다른 포트를 사용하려는 경우에만 변경합니다.
    9. 선택 VPC- 이 클러스터를 생성할 VPC 를 선택합니다.
    10. 파라미터 그룹 - 선택한 노드 유형에 대한 Valkey 또는 Redis OSS 오버헤드에 충분한 메모리를 예약하는 파라미터 그룹을 선택합니다.
6. 원하는 대로 설정되었으면 [Create]를 선택합니다.

클러스터 생성에 대한 자세한 내용은 Valkey 또는 Redis용 클러스터 생성 OSS 섹션을 참조하세요.

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

를 사용하여 복제 그룹을 생성할 때 AWS CLI기본적으로 데이터 계층화는 cache.r6gd.xlarge와 같은 r6gd 패밀리에서 노드 유형을 선택하고 `--data-tiering-enabled` 파라미터를 설정하여 사용됩니다.

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

Linux, macOS, Unix의 경우:

```
aws elasticache create-replication-group \
   --replication-group-id redis-dt-cluster \
```

```
   --replication-group-description "Redis OSS cluster with data tiering" \
   --num-node-groups 1 \
   --replicas-per-node-group 1 \
   --cache-node-type cache.r6gd.xlarge \
   --engine redis \
   --cache-subnet-group-name default \
   --automatic-failover-enabled \
   --data-tiering-enabled \
   --snapshot-name my-snapshot
```

Linux, macOS, Unix의 경우:

```
aws elasticache create-replication-group ^
   --replication-group-id redis-dt-cluster ^
   --replication-group-description "Redis OSS cluster with data tiering" ^
   --num-node-groups 1 ^
   --replicas-per-node-group 1 ^
   --cache-node-type cache.r6gd.xlarge ^
   --engine redis ^
   --cache-subnet-group-name default ^
   --automatic-failover-enabled ^
   --data-tiering-enabled ^
   --snapshot-name my-snapshot
```

이 작업을 실행하면 다음과 유사한 응답이 표시됩니다.

```
{
    "ReplicationGroup": {
        "ReplicationGroupId": "redis-dt-cluster",
        "Description": "Redis OSS cluster with data tiering",
        "Status": "creating",
        "PendingModifiedValues": {},
        "MemberClusters": [
            "redis-dt-cluster"
        ],
        "AutomaticFailover": "enabled",
        "DataTiering": "enabled",
        "SnapshotRetentionLimit": 0,
        "SnapshotWindow": "06:00-07:00",
        "ClusterEnabled": false,
        "CacheNodeType": "cache.r6gd.xlarge",
        "TransitEncryptionEnabled": false,
        "AtRestEncryptionEnabled": false
```

```
    }
}
```

에서 클러스터 준비 ElastiCache

다음은 ElastiCache 콘솔, AWS CLI또는 를 사용하여 클러스터를 생성하는 방법에 대한 지침을 확인할 수 있습니다 ElastiCache API.

를 사용하여 ElastiCache 클러스터를 생성할 수도 있습니다AWS CloudFormation. 자세한 내용은 해당 접근 방식을 구현하는 방법에 대한 지침이 포함된 AWS Cloud Formation 사용 설명서의 AWS:ElastiCache:::CacheCluster를 참조하세요.

클러스터 또는 복제 그룹을 생성할 때마다 준비 작업을 미리 하면 즉시 업그레이드하거나 변경할 필요가 없어 좋습니다.

주제

- ElastiCache 클러스터 요구 사항 확인
- 노드 크기 선택

ElastiCache 클러스터 요구 사항 확인

준비

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

- 필요한 노드 인스턴스 유형은 무엇입니까?

  인스턴스 노드 유형 선택에 도움이 필요한 경우 노드 크기 선택를 참조하세요.
- Amazon 기반 가상 프라이빗 클라우드(VPC)에서 클러스터를 시작하시겠습니까VPC?

> **Important**
>
> 에서 클러스터를 시작하려면 클러스터 생성을 시작하기 VPC 전에 동일한 에서 서브넷 그룹을 생성해야 VPC합니다. 자세한 내용은 서브넷 및 서브넷 그룹 단원을 참조하십시오. ElastiCache 는 Amazon 를 AWS 사용하여 내부에서 액세스하도록 설계되었습니다EC2. 그러나 Amazon VPC 기반 에서 를 시작하고 VPC 클러스터가 에 있는 경우 외부 에서 액세스를 제공할 VPC수 있습니다 AWS. 자세한 내용은 외부에서 ElastiCache 리소스에 액세스 AWS 단원을 참조하십시오.

- 파라미터 값을 사용자 지정해야 합니까?

  그렇다면 사용자 지정 파라미터 그룹을 만듭니다. 자세한 내용은 ElastiCache 파라미터 그룹 생성 단원을 참조하십시오.

  Valkey 또는 Redis 를 실행하는 경우 `reserved-memory` 또는 를 설정하는 것이 OSS좋습니다`reserved-memory-percent`. 자세한 내용은 Valkey 및 Redis용 예약 메모리 관리 OSS 단원을 참조하십시오.
- 자체 VPC 보안 그룹을 생성해야 합니까?

  자세한 내용은 의 보안을 참조하세요VPC.
- 어떤 방법으로 내결함성을 구현하시겠습니까?

  자세한 내용은 장애 완화 단원을 참조하십시오.

주제

- ElastiCache 메모리 및 프로세서 요구 사항
- Memcached 클러스터 구성
- Valkey 및 Redis OSS 클러스터 구성
- ElastiCache 크기 조정 요구 사항
- ElastiCache 액세스 요구 사항
- 에 대한 리전, 가용 영역 및 로컬 영역 요구 사항 ElastiCache

ElastiCache 메모리 및 프로세서 요구 사항

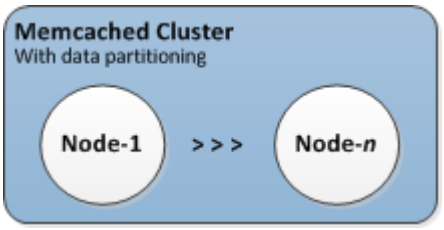
Amazon의 기본 구성 요소는 노드 ElastiCache 입니다. 노드는 개별적으로 또는 그룹으로 구성되어 클러스터를 형성합니다. 클러스터에 사용할 노드 유형을 결정할 때 클러스터의 노드 구성과 저장해야하는 데이터의 양을 고려합니다.

Memcached 엔진은 다중 스레드이므로 노드의 코어 수가 클러스터에 사용할 수 있는 컴퓨팅 파워에 영향을 줍니다.

Memcached 클러스터 구성

ElastiCache (Memcached) 클러스터는 1~60개의 노드로 구성됩니다. Memcached 클러스터의 데이터는 클러스터의 노드로 분할됩니다. 애플리케이션은 엔드포인트라는 네트워크 주소를 사용하여 Memcached 클러스터에 연결됩니다. Memcached 클러스터의 각 노드에는 애플리케이션이 특정 노드

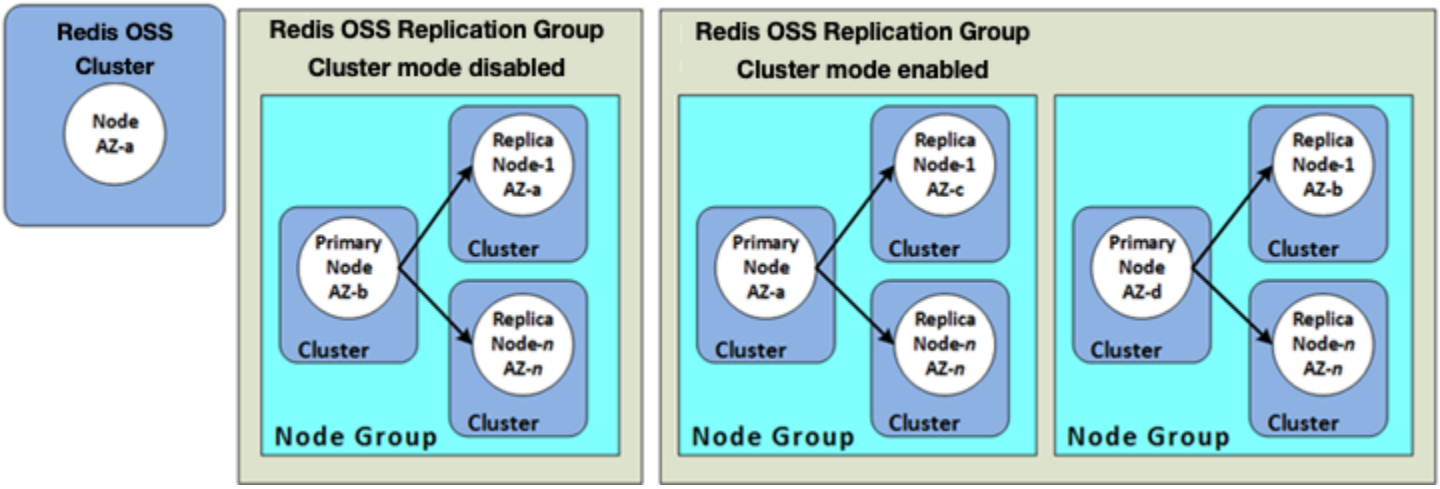
에(서) 읽고 쓰는 데 사용하는 자체 엔드포인트가 있습니다. 노드 엔드포인트 외에도 Memcached 클러스터는 구성 엔드포인트라는 엔드포인트를 가지고 있습니다. 애플리케이션에서는 이 엔드포인트를 사용하여 클러스터에서 읽거나 쓸 수 있으며, 읽을 노드 또는 쓸 노드에 대한 결정은 자동 검색에 맡깁니다.



자세한 내용은 에서 클러스터 관리 ElastiCache 단원을 참조하십시오.

Valkey 및 Redis OSS 클러스터 구성

ElastiCache 와 Valkey 및 Redis OSS 클러스터는 0~500개의 샤드(노드 그룹이라고도 함)로 구성됩니다. Valkey 또는 Redis OSS 클러스터의 데이터는 클러스터의 샤드를 통해 분할됩니다. 애플리케이션은 엔드포인트라는 네트워크 주소를 사용하여 Valkey 또는 Redis OSS 클러스터에 연결됩니다. Valkey 또는 Redis OSS 샤드의 노드는 읽기/쓰기 기본 및 다른 모든 노드 읽기 전용 보조(읽기 전용 복제본이라고도 함)의 두 역할 중 하나를 수행합니다. 노드 엔드포인트 외에도 Valkey 또는 Redis OSS 클러스터 자체에는 구성 엔드포인트 라는 엔드포인트가 있습니다. 애플리케이션은 이 엔드포인트를 사용하여 클러스터에서 읽거나 클러스터에 쓸 수 있으므로 (Redis)에서 읽거나 최대 ElastiCache (Redis OSS)에 쓸 노드를 결정할 수 있습니다.



자세한 내용은 에서 클러스터 관리 ElastiCache 단원을 참조하십시오.

ElastiCache 크기 조정 요구 사항

더 크고 새로운 노드 유형으로 새 클러스터를 생성하여 모든 클러스터를 조정할 수 있습니다. Memcached 클러스터를 확장하면 새 클러스터가 비어 있게 시작됩니다. Valkey 또는 Redis OSS 클러

스터를 스케일 업할 때 백업에서 클러스터를 시드하고 새 클러스터가 비어 있지 않도록 할 수 있습니다.

Amazon ElastiCache for Memcached 클러스터는 스케일 아웃하거나 스케일 인할 수 있습니다. Memcached 클러스터를 스케일 아웃하거나 스케일 인하려면 클러스터에서 노드를 추가하거나 삭제하세요. Auto Discovery를 사용하고 애플리케이션이 클러스터의 구성 엔드포인트와 연결된 경우 노드를 추가하거나 제거할 때 애플리케이션을 변경할 필요가 없습니다.

자세한 내용은 이 가이드의 크기 조정 ElastiCache을 참조하세요.

ElastiCache 액세스 요구 사항

설계상 Amazon ElastiCache 클러스터는 Amazon EC2 인스턴스에서 액세스할 수 있습니다. ElastiCache 클러스터에 대한 네트워크 액세스는 클러스터를 생성한 계정으로 제한됩니다. 따라서 Amazon EC2 인스턴스에서 클러스터에 액세스하려면 먼저 Amazon EC2 인스턴스가 클러스터에 액세스하도록 권한을 부여해야 합니다. 이를 수행하는 단계는 EC2-VPC 또는 EC2-Classic에서 를 시작했는지 여부에 따라 달라집니다.

에서VPC 클러스터를 시작한 경우 클러스터에 네트워크 수신을 부여EC2해야 합니다. 클러스터를 EC2-Classic으로 시작한 경우 인스턴스와 연결된 Amazon Elastic Compute Cloud 보안 그룹에 ElastiCache 보안 그룹에 대한 액세스 권한을 부여해야 합니다. 자세한 지침은 이 가이드의 단계 3. 클러스터에 대한 액세스 권한 부여를 참조하세요.

에 대한 리전, 가용 영역 및 로컬 영역 요구 사항 ElastiCache

Amazon은 모든 AWS 리전을 ElastiCache 지원합니다. 애플리케이션과 가까운 AWS 리전에서 ElastiCache 클러스터를 찾으면 지연 시간을 줄일 수 있습니다. 클러스터에 다중 노드가 있는 경우 다른 가용 영역이나 Local Zones에 노드를 배치하면 클러스터에 장애가 미치는 영향을 줄일 수 있습니다.

자세한 내용은 다음을 참조하세요.

- 에 대한 리전 및 가용 영역 선택 ElastiCache
- 에서 로컬 영역 사용 ElastiCache
- 장애 완화

노드 크기 선택

ElastiCache 클러스터에 대해 선택한 노드 크기는 비용, 성능 및 내결함성에 영향을 미칩니다.

노드 크기(Valkey 및 RedisOSS)

Graviton 프로세서의 이점에 대한 자세한 내용은 AWS Graviton 프로세서를 참조하세요.

다음 질문에 답하면 Valkey 또는 Redis OSS 구현에 필요한 최소 노드 유형을 결정하는 데 도움이 될 수 있습니다.

- 여러 클라이언트 연결을 사용하는 처리량 제한 워크로드를 기대하십니까?

  이 경우 Redis OSS 버전 5.0.6 이상을 실행하는 경우 Redis OSS 엔진을 대신하여 클라이언트 연결을 오프로드하는 CPUs 데 사용할 수 있는 경우 향상된 I/O 기능을 사용하여 처리량과 지연 시간을 개선할 수 있습니다. Redis OSS 버전 7.0.4 이상을 실행하는 경우 향상된 I/O 외에도 향상된 I/O 멀티플렉싱을 통해 추가 가속화를 얻을 수 있습니다. 각 전용 네트워크 IO 스레드는 여러 클라이언트의 명령을 Redis OSS 엔진으로 전달하여 Redis 의 명령을 배치로 효율적으로 처리할 수 있는 OSS기능을 활용합니다. ElastiCache (Redis OSS) v7.1 이상에서는 향상된 I/O 스레드 기능을 확장하여 프레젠테이션 계층 로직도 처리합니다. 프레젠테이션 계층이란 향상된 I/O 스레드가 이제 클라이언트 입력을 읽을 뿐만 아니라 입력을 Redis OSS 바이너리 명령 형식으로 구문 분석하여 실행을 위해 메인 스레드로 전달되어 성능 향상을 제공한다는 의미입니다. 자세한 내용은 블로그 게시물과 지원되는 버전 페이지를 참조하세요.
- 소량의 데이터에 정기적으로 액세스하는 워크로드가 있나요?

  이 경우 Redis OSS 엔진 버전 6.2 이상에서 실행 중인 경우 r6gd 노드 유형을 선택하여 데이터 계층화를 활용할 수 있습니다. 데이터 계층화를 사용하면 최근에 가장 적게 사용된 데이터가 에 저장됩니다SSD. 검색 시 대기 시간 비용이 적게 들고 비용 절감으로 균형을 이룹니다. 자세한 내용은 의 데이터 계층화 ElastiCache 단원을 참조하십시오.

  자세한 내용은 지원되는 노드 유형 단원을 참조하십시오.
- 데이터에 필요한 총 메모리는 얼마나 됩니까?

  일반적인 에상치를 알아보려면 캐시할 항목의 크기를 선택합니다. 이 크기를 동시에 캐시에 보관할 항목 수로 곱합니다. 적당한 항목 크기 예상치를 구하고 캐시 항목을 직렬화한 후 문자 수를 계산합니다. 그런 다음 클러스터에 있는 샤드의 수에 대해 이를 나눕니다.

  자세한 내용은 지원되는 노드 유형 단원을 참조하십시오.
- 어떤 버전의 RedisOSS를 실행하고 있습니까?

  2.8.22 이전의 Redis OSS 버전에서는 장애 조치, 스냅샷, 동기화 및 복제본을 기본 작업으로 승격하기 위해 더 많은 메모리를 예약해야 합니다. 프로세스 중에 발생하는 모든 쓰기에 충분한 메모리를 사용할 수 있어야 하기 때문입니다.

Redis OSS 버전 2.8.22 이상에서는 이전 프로세스보다 사용 가능한 메모리가 더 적은 포크리스 저장 프로세스를 사용합니다.

자세한 내용은 다음 자료를 참조하세요.

- 동기화 및 백업 구현 방법
- Valkey 또는 Redis OSS 스냅샷을 생성하기에 충분한 메모리 확보

- 애플리케이션의 쓰기 작업이 얼마나 많습니까?

  쓰기 작업이 많은 애플리케이션에서는 스냅샷을 만들거나 장애 조치를 할 때 데이터에 사용되지 않으며 사용할 수 있는 메모리가 훨씬 더 많이 필요할 수 있습니다. BGSAVE 프로세스가 수행될 때마다 BGSAVE 프로세스 중에 발생하는 모든 쓰기를 수용할 수 있도록 데이터가 사용하지 않는 메모리가 충분해야 합니다. 예를 들어 스냅샷을 찍을 때, 기본 클러스터를 클러스터의 복제본과 동기화할 때, 추가 전용 파일(AOF) 기능을 활성화할 때가 있습니다. 또 다른 예로는 복제본을 기본으로 승격하는 경우입니다(활성화된 다중 AZ가 있는 경우). 최악의 경우는 프로세스 중에 모든 데이터가 다시 작성되는 경우입니다. 이 경우 데이터에만 필요한 메모리의 두 배가 되는 노드 인스턴스 크기가 필요합니다.

  더 자세한 내용은 Valkey 또는 Redis OSS 스냅샷을 생성하기에 충분한 메모리 확보 섹션을 참조하세요.

- 구현이 독립 실행형 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터이거나 여러 개의 샤드가 있는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터입니까?

  Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터

  Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터를 구현하는 경우 이전 글머리 기호에 설명된 대로 노드 유형이 모든 데이터와 필요한 오버헤드를 수용할 수 있어야 합니다.

  예를 들어, 모든 항목의 총 크기를 12GB로 추정합니다. 이 경우, 메모리가 13.3GB인 `cache.m3.xlarge` 노드 또는 메모리가 13.5GB인 `cache.r3.large` 노드를 사용할 수 있습니다. 하지만 BGSAVE 작업에는 메모리가 더 필요할 수 있습니다. 애플리케이션이 쓰기 작업이 많은 경우 메모리 요구 사항을 최소 24GB로 두 배로 늘립니다. 따라서 27.9GB의 메모리가 있는 `cache.m3.2xlarge` 또는 30.5GB의 메모리가 있는 `cache.r3.xlarge`를 사용합니다.

  샤드가 여러 개인 Valkey 또는 RedisOSS(클러스터 모드 활성화됨)

  샤드가 여러 개인 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터를 구현하는 경우 노드 유형이 `bytes-for-data-and-overhead / number-of-shards`바이트의 데이터를 수용할 수 있어야 합니다.

예를 들어, 모든 항목의 총 크기를 12GB로 추정하고 샤드가 2개입니다. 이 경우, 메모리가 6.05GB인 cache.m3.large 노드를 사용할 수 있습니다(12GB/2). 하지만 BGSAVE 작업에는 메모리가 더 필요할 수 있습니다. 애플리케이션이 쓰기 작업이 많은 경우 메모리 요구 사항을 최소 샤드당 12GB로 두 배로 늘립니다. 따라서 13.3GB의 메모리가 있는 cache.m3.xlarge 또는 13.5GB의 메모리가 있는 cache.r3.large를 사용합니다.

- Local Zones를 사용하고 있습니까?

  로컬 영역을 사용하면 ElastiCache 클러스터와 같은 리소스를 사용자와 가까운 여러 위치에 배치할 수 있습니다. 그러나 노드 크기를 선택할 때 사용 가능한 노드 크기는 용량 요구 사항에 관계없이 현재 다음과 같이 제한된다는 것에 주의하세요.

  - 현재 세대:

    M5 노드 유형: cache.m5.large, cache.m5.xlarge, cache.m5.2xlarge, cache.m5.4xlarge, cache.m5.12xlarge, cache.m5.24xlarge

    R5 노드 유형: cache.r5.large, cache.r5.xlarge, cache.r5.2xlarge, cache.r5.4xlarge, cache.r5.12xlarge, cache.r5.24xlarge

    T3 노드 유형: cache.t3.micro, cache.t3.small, cache.t3.medium

클러스터가 실행되는 동안 에 게시된 메모리 사용량, 프로세서 사용률, 캐시 히트 및 캐시 누락 지표를 모니터링할 수 있습니다 CloudWatch. 클러스터의 적중률이 기대에 미치지 못하거나 키가 너무 자주 제거되고 있음을 알 수 있습니다. 이러한 경우 CPU 및 메모리 사양이 더 큰 다른 노드 크기를 선택할 수 있습니다.

CPU 사용량을 모니터링할 때 Valkey와 RedisOSS는 단일 스레드임을 기억하세요. 따라서 보고된 CPU 사용량에 CPU 코어 수를 곱하여 실제 사용량을 구합니다. 예를 들어 20% 사용률을 CPU 보고하는 4개 코어는 실제로 1개 코어 Redis가 80% 사용률로 실행OSS되고 있습니다.

노드 크기(Memcached)

Memcached 클러스터에는 여러 노드로 분할된 클러스터 데이터와 한 개 이상의 노드가 포함되어 있습니다. 클러스터의 메모리 요구와 노드의 메모리가 서로 관련은 있지만 동일하지는 않습니다. 큰 노드 두세 개나 작은 노드 여러 개를 두어 필요한 클러스터 메모리 용량을 확보할 수 있습니다. 또한 요구량이 달라지면 클러스터에서 노드를 추가하거나 제거하여 필요한 만큼만 비용을 지불할 수 있습니다.

클러스터의 총 메모리 용량은 시스템 오버헤드를 공제한 후 클러스터의 노드 수에 각 노드의 RAM 용량을 곱하여 계산됩니다. 각 노드의 용량은 노드 유형에 따라 다릅니다.

```
cluster_capacity = number_of_nodes * (node_capacity - system_overhead)
```

클러스터의 노드 수는 Memcached를 실행하는 클러스터의 가용성을 결정하는 핵심 요인입니다. 단일 노드의 오류는 애플리케이션 가용성과 백엔드 데이터베이스에 대한 로드에 영향을 미칠 수 있습니다. 이 경우 ElastiCache는 실패한 노드에 대한 대체를 프로비저닝하고 다시 채워집니다. 이러한 가용성 영향을 줄이려면 적은 수의 대용량 노드를 사용하는 대신 더 적은 용량의 더 여러 개의 노드로 메모리 및 컴퓨팅 용량을 분산해야 합니다.

캐시 메모리가 35GB인 시나리오에서 다음과 같은 구성으로 설정할 수 있습니다.

- 메모리 3.22GB인 `cache.t2.medium` 노드 11개에 스레드가 각각 2개이면 35.42GB, 스레드 22개와 같습니다.
- 메모리 6.42GB인 `cache.m4.large` 노드 6개에 스레드가 각각 2개이면 38.52GB, 스레드 12개와 같습니다.
- 메모리 12.3GB인 `cache.r4.large` 노드 3개에 스레드가 각각 2개이면 36.90GB, 스레드 6개와 같습니다.
- 메모리 14.28GB인 `cache.m4.xlarge` 노드 3개에 스레드가 각각 4개이면 42.84GB, 스레드 12개와 같습니다.

노드 옵션 비교

노드 유형	메모리 (GB)	코어	시간당 비용*	필요한 노드	총 메모리 (GiB)	총 코어	월별 비용
cache.t2.medium	3.22	2	0.068 USD	11	35.42	22	538.56 USD
cache.m4.large	6.42	2	0.156 USD	6	38.52	12	673.92 USD
cache.m4.xlarge	14.28	4	0.311 USD	3	42.84	12	671.76 USD
cache.m5.xlarge	12.93	4	0.311 USD	3	38.81	12	671.76 USD
cache.m6g.large	6.85	2	$ 0.147	6	41.1	12	$635

노드 유형	메모리 (GB)	코어	시간당 비용*	필요한 노드	총 메모리 (GiB)	총 코어	월별 비용
cache.r4.large	12.3	2	0.228 USD	3	36.9	6	492.48 USD
cache.r5.large	13.07	2	0.216 USD	3	39.22	6	466.56 USD
cache.r6g.large	13.07	2	$ 0.205	3	42.12	6	$442
* 2020년 10월 8일 기준 각 노드의 시간당 비용							
30일간(720시간) 사용량 100%의 월별 비용							

여기에 나와 있는 옵션은 각각 비슷한 메모리 용량을 제공하지만 컴퓨팅 용량과 비용은 다릅니다. 특정 옵션의 비용을 비교하려면 Amazon ElastiCache 요금 섹션을 참조하세요.

Memcached를 실행하는 클러스터의 경우 각 노드의 사용 가능한 메모리 일부가 연결 오버헤드에 사용됩니다. 자세한 내용은 Memcached 연결 오버헤드 섹션을 참조하세요.

여러 노드를 사용하면 노드에 키를 분산시켜야 합니다. 노드마다 고유한 엔드포인트가 있습니다. 엔드포인트를 쉽게 관리하기 위해 ElastiCache Auto Discovery 기능을 사용할 수 있습니다. 이 기능을 사용하면 클라이언트 프로그램이 클러스터의 모든 노드를 자동으로 식별할 수 있습니다. 자세한 내용은 클러스터의 노드 자동 식별(Memcached) 단원을 참조하십시오.

경우에 따라 얼마나 많은 용량이 필요한지 확신할 수 없을 수도 있습니다. 그렇다면, 테스트를 위해 1개의 `cache.m5.large` 노드를 사용하는 것이 좋습니다. 그런 다음 Amazon 에 게시된 ElastiCache 지표를 사용하여 메모리 사용량, CPU 사용률 및 캐시 적중률을 모니터링합니다 CloudWatch. 의 CloudWatch 지표에 대한 자세한 내용은 섹션을 ElastiCache참조하세요 CloudWatch 지표 사용 모니터링. 프로덕션 및 대규모 워크로드의 경우 R5 노드는 최상의 성능과 RAM 비용 가치를 제공합니다.

클러스터의 적중률이 기대에 미치지 못하면 간편하게 노드를 더 추가하여 클러스터의 총 가용 메모리를 늘릴 수 있습니다.

클러스터에 바인딩되어 CPU 있지만 적중률이 충분한 경우 더 많은 컴퓨팅 성능을 제공하는 노드 유형을 사용하여 새 클러스터를 설정합니다.

Valkey 또는 Redis용 클러스터 생성 OSS

다음 예제에서는 AWS Management Console, AWS CLI 및 를 사용하여 Valkey 또는 Redis OSS 클러스터를 생성하는 방법을 보여줍니다 ElastiCache API.

Valkey 또는 Redis 생성OSS(클러스터 모드 비활성화됨)(콘솔)

ElastiCache 는 Valkey 또는 Redis OSS 엔진을 사용할 때 복제를 지원합니다. 데이터가 Valkey 또는 Redis OSS 읽기/쓰기 기본 클러스터에 기록되는 시점과 읽기 전용 보조 클러스터에 전파되는 시점 사이의 지연 시간을 모니터링하려면 클러스터에 특수 키를 ElastiCache 추가합니다`ElastiCacheMasterReplicationTimestamp`. 이 키는 현재 Universal Time(UTC) 시간입니다. Valkey 또는 Redis OSS 클러스터는 나중에 복제 그룹에 추가될 수 있으므로 이 키는 처음에 복제 그룹의 멤버가 아니더라도 모든 Valkey 또는 Redis OSS 클러스터에 포함됩니다. 복제 그룹에 대한 자세한 정보는 고가용성을 위한 복제 그룹 사용 섹션을 참조하세요.

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터를 생성하려면 의 단계를 따릅니다Valkey(클러스터 모드 비활성화됨) 클러스터 생성(콘솔).

클러스터의 상태를 사용할 수 있게 되면 즉시 Amazon에 EC2 액세스 권한을 부여하고 연결한 다음 사용을 시작할 수 있습니다. 자세한 내용은 단계 3. 클러스터에 대한 액세스 권한 부여 및 4단계. 클러스터의 노드에 연결 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. 에서 클러스터 삭제 ElastiCache을 참조하세요.

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 생성(콘솔)

Redis OSS 3.2.4 이상을 실행하는 경우 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터를 생성할 수 있습니다. Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터는 1~500개의 샤드(API/CLI: 노드 그룹)에서 데이터를 파티셔닝할 수 있지만 일부 제한이 있습니다. Valkey 또는 RedisOSS(클러스터 모드 비활성화됨)와 Valkey 또는 RedisOSS(클러스터 모드 활성화됨)의 비교는 섹션을 참조하세요지원되는 엔진 및 버전.

ElastiCache 콘솔을 사용하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터를 생성하려면

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 오른쪽 상단 모서리의 목록에서 이 클러스터를 시작할 AWS 리전을 선택합니다.
3. 탐색 창에서 시작하기(Get started)를 선택하세요.
4. 생성을 VPC 선택하고 Virtual Private Cloud 생성(VPC)에 설명된 단계를 따릅니다.
5. ElastiCache 대시보드 페이지에서 클러스터 생성을 선택한 다음 Valkey 클러스터 생성 또는 Redis OSS 클러스터 생성을 선택합니다.
6. 클러스터 설정(Cluster settings)에서 다음을 수행합니다.
   a. Configure and create a new cluster(새 클러스터 구성 및 생성)를 선택합니다.
   b. Cluster mode(클러스터 모드)에서 Enabled(사용 설정됨)를 선택합니다.
   c. Cluster info(클러스터 정보)에 Name(이름) 값을 입력합니다.
   d. (선택 사항) 설명(Description) 값을 입력합니다.
7. Location(위치)에서 다음을 수행합니다.

   AWS Cloud

   1. AWS Cloud의 경우 Multi-AZ(다중 AZ) 및 Auto-failover(자동 장애 조치)의 기본 설정을 수락하는 것이 좋습니다. 자세한 내용은 다중 AZ 를 사용하여 ElastiCache (Redis OSS)의 가동 중지 시간 최소화를 참조하세요.
   2. 클러스터 설정(Cluster settings)에서 다음을 수행합니다.
      a. 엔진 버전(Engine version)의 경우 사용 가능한 버전을 선택합니다.
      b. 포트(Port)의 경우 기본 포트인 6379를 사용합니다. 다른 포트를 사용해야 하는 경우 포트 번호를 입력합니다.
      c. 파라미터 그룹에서 파라미터 그룹을 선택하거나 새 파라미터 그룹을 만듭니다. 파라미터 그룹은 클러스터의 런타임 파라미터를 제어합니다. 파라미터 그룹에 대한 자세한 정보는 Valkey 및 Redis OSS 파라미터 및 ElastiCache 파라미터 그룹 생성 섹션을 참조하세요.

> **Note**
>
> 파라미터 그룹을 선택하여 엔진 구성 값을 설정하면 해당 파라미터 그룹이 글로벌 데이터 스토어의 모든 클러스터에 적용됩니다. 파라미터 그룹 페이지에서 yes/no 글로벌 속성은 파라미터 그룹이 글로벌 데이터 스토어의 일부인지 여부를 나타냅니다.

d. 노드 유형에서 아래쪽 화살표 ( ) 를 선택합니다. 노드 유형 변경 대화 상자에서 원하는 노드 유형의 인스턴스 패밀리 값을 선택합니다. 그런 다음 이 클러스터에 사용할 노드 유형을 선택한 다음 저장을 선택합니다.

   자세한 정보는 노드 크기 선택 섹션을 참조하세요.

   r6gd 노드 유형을 선택하는 경우 데이터 계층화가 자동으로 사용 설정됩니다. 자세한 내용은 의 데이터 계층화 ElastiCache 단원을 참조하십시오.

e. 샤드 수 에서 이 Valkey 또는 Redis(클러스터 모드 활성화됨) 클러스터에 사용할 샤드 수OSS(파티션/노드 그룹)를 선택합니다.

   Valkey 또는 Redis의 일부 버전OSS(클러스터 모드 활성화됨)의 경우 클러스터의 샤드 수를 동적으로 변경할 수 있습니다.

   - Redis OSS 3.2.10 이상 - 클러스터가 Redis OSS 3.2.10 이상 버전을 실행하는 경우 클러스터의 샤드 수를 동적으로 변경할 수 있습니다. 자세한 내용은 Valkey 또는 Redis에서 클러스터 크기 조정OSS(클러스터 모드 활성화됨) 단원을 참조하십시오.
   - 기타 Redis OSS 버전 - 클러스터가 버전 3.2.10 OSS 이전의 Redis 버전을 실행하는 경우 다른 접근 방식이 있습니다. 이 경우 클러스터의 샤드 수를 변경하려면 새 샤드 수로 새 클러스터를 만듭니다. 자세한 내용은 백업에서 새 캐시로 복원 단원을 참조하십시오.

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

   Valkey 또는 RedisOSS(클러스터 모드 활성화됨)에는 다음과 같은 제한 사항이 있습니다.

   - 다중 AZ를 활성화한 경우 샤드당 복제본이 하나 이상 있어야 합니다.

- 콘솔을 사용하여 클러스터를 생성할 때 샤드마다 복제본 수가 동일합니다.
- 샤드당 읽기 전용 복제본 수가 고정되어 변경할 수 없습니다. 샤드당 복제본(API/CLI: 노드 그룹)이 더 많거나 더 적게 필요한 경우 새 복제본 수로 새 클러스터를 생성해야 합니다. 자세한 내용은 자습서: 외부에서 생성된 백업을 사용하여 자체 설계된 새 클러스터 검색 단원을 참조하십시오.

3. 연결 아래

   a. 네트워크 유형에서 이 클러스터가 지원할 IP 버전을 선택합니다.

   b. 서브넷 그룹 에서 이 클러스터에 적용할 서브넷을 선택합니다. 는 해당 서브넷 그룹을 ElastiCache 사용하여 해당 서브넷 내의 서브넷 및 IP 주소를 선택하여 노드와 연결합니다. ElastiCache 클러스터에는 듀얼 스택 모드에서 작동하려면 IPv4 및 IPv6 주소가 할당된 듀얼 스택 서브넷이 필요하며, IPv6IPv6는 전용으로 작동하려면 전용 서브넷이 필요합니다.

      새 서브넷 그룹을 생성할 때 해당 서브넷 그룹이 속한 VPC ID를 입력합니다.

      검색 IP 유형을 선택합니다. 선택한 프로토콜의 IP 주소만 반환됩니다.

      자세한 내용은 다음을 참조하세요.

      - 에서 네트워크 유형 선택 ElastiCache.
      - 에서 서브넷을 생성합니다VPC.

      에서 로컬 영역 사용 ElastiCache를 사용하는 경우 로컬 영역에 있는 서브넷을 선택하거나 생성해야 합니다.

      자세한 내용은 서브넷 및 서브넷 그룹 단원을 참조하십시오.

4. 가용 영역 배치(Availability zone placements)의 경우 다음 두 가지 옵션이 있습니다.

   - 기본 설정 없음 - 가용 영역을 ElastiCache 선택합니다.
   - 가용 영역 지정 – 각 클러스터의 가용 영역을 지정합니다.

     가용 영역을 지정하도록 선택한 경우 샤드에 있는 각 클러스터에 대해 목록에서 가용 영역을 선택합니다.

   자세한 내용은 에 대한 리전 및 가용 영역 선택 ElastiCache 단원을 참조하십시오.

5. 다음(Next)을 선택합니다.

6. 고급 Valkey 설정 또는 고급 Redis OSS 설정 또는

   - 보안(Security)의 경우

     i. 데이터를 암호화하려면 다음과 같은 옵션이 있습니다.

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

       > **Note**
       >
       > Customer Managed AWS KMS 키를 선택하고 키를 선택하여 다른 암호화 키를 제공할 수 있습니다. 자세한 내용은 에서 AWS 고객 관리형 키 사용을 KMS참조하세요.

     - 전송 중 데이터 암호화 – 전송 데이터 암호화를 활성화합니다. 자세한 정보는 전송 중 데이터 암호화를 참조하세요. Valkey 7.2 이상 또는 Redis OSS 엔진 버전 6.0 이상의 경우 전송 중 암호화를 활성화하면 다음 액세스 제어 옵션 중 하나를 지정하라는 메시지가 표시됩니다.
       - 액세스 제어 안 함 – 기본 설정입니다. 이 옵션은 클러스터에 대한 사용자 액세스를 제한하지 않는다는 의미입니다.
       - 사용자 그룹 액세스 제어 목록 - 클러스터에 액세스할 수 있는 사용자 집합이 정의된 사용자 그룹을 선택합니다. 자세한 내용은 콘솔 및 를 사용하여 사용자 그룹 관리 CLI 단원을 참조하십시오.
       - AUTH 기본 사용자 - Valkey 또는 Redis OSS 서버의 인증 메커니즘입니다. 자세한 내용은 AUTH를 참조하세요.
     - AUTH – Valkey 또는 Redis OSS 서버의 인증 메커니즘입니다. 자세한 내용은 AUTH를 참조하세요.

       > **Note**
       >
       > OSS 버전 3.2.10을 제외한 3.2.6 이상 Redis 버전의 경우 유일한 옵션 AUTH입니다.

     ii. 보안 그룹에서 이 클러스터에 사용할 보안 그룹을 선택합니다. 보안 그룹은 클러스터에 대한 네트워크 액세스를 제어하는 방화벽 역할을 합니다. 에 기본 보안 그룹을 사용하거나 새 보안 그룹을 VPC 생성할 수 있습니다.

보안 그룹에 대한 자세한 내용은 Amazon 사용 설명서의 에 대한 보안 그룹을 VPC 참조하세요. VPC

7. 정기적인 자동 백업을 예약할 경우 Enable automatic backups(자동 백업 활성화)를 선택한 후 자동으로 삭제되기 전에 각 자동 백업을 보존할 기간(일)을 입력합니다. 정기적인 자동 백업을 예약하지 않으려면 [Enable automatic backups] 확인란의 선택을 취소합니다. 어떤 경우든 수동 백업을 항상 생성할 수 있습니다.

   백업 및 복원에 대한 자세한 내용은 섹션을 참조하세요스냅샷 및 복원.

8. (선택 사항) 유지 관리 기간을 지정합니다. 유지 관리 기간은 클러스터에 대한 시스템 유지 관리를 예약하는 ElastiCache 매주 일반적으로 1시간의 시간입니다. ElastiCache 에서 유지 관리 기간의 날짜 및 시간을 선택하거나(기본 설정 없음) 직접 날짜, 시간 및 기간을 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 ElastiCache 클러스터 유지 관리 단원을 참조하십시오.

9. (선택 사항) 로그의 경우:

   - 로그 형식에서 텍스트 또는 를 선택합니다JSON.
   - 대상 유형에서 CloudWatch 로그 또는 Kinesis Firehose를 선택합니다.
   - 로그 대상 에서 새로 만들기를 선택하고 CloudWatch 로그 로그 그룹 이름 또는 Firehose 스트림 이름을 입력하거나 기존 선택을 선택한 다음 CloudWatch 로그 로그 로그 그룹 이름 또는 Firehose 스트림 이름을 선택합니다.

10. 태그 의 경우 클러스터 및 기타 ElastiCache 리소스를 관리하는 데 도움이 되도록 태그 형식으로 각 리소스에 자체 메타데이터를 할당할 수 있습니다. 자세한 정보는 ElastiCache 리소스 태그 지정 섹션을 참조하세요.

11. Next(다음)를 선택합니다.

12. 입력 및 선택한 내용을 모두 검토한 다음 필요한 내용을 수정합니다. 준비가 되었으면 생성(Create)e을 선택합니다.

On premises

1. On premises(온프레미스)의 경우 Auto-failover(자동 장애 조치)를 사용 설정하는 것이 좋습니다. 자세한 내용은 다중 AZ를 사용한 ElastiCache (Redis OSS)의 가동 중지 시간 최소화를 참조하세요.

2. Outposts 사용의 단계를 수행하세요.

ElastiCache 콘솔 AWS CLI 대신 또는 를 ElastiCache API 사용하여 동등한 를 생성하려면 다음을 참조하세요.

- API: CreateReplicationGroup
- CLI: create-replication-group

클러스터의 상태를 사용할 수 있게 되면 즉시 클러스터에 대한 EC2 액세스 권한을 부여하고 연결한 다음 사용을 시작할 수 있습니다. 자세한 내용은 단계 3. 클러스터에 대한 액세스 권한 부여 및 4단계. 클러스터의 노드에 연결 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. 에서 클러스터 삭제 ElastiCache을 참조하세요.

클러스터 생성(AWS CLI)

를 사용하여 클러스터를 생성하려면 `create-cache-cluster` 명령을 AWS CLI사용합니다.

> **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. 에서 클러스터 삭제 ElastiCache을 참조하세요.

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터 생성(CLI)

Example – 읽기 전용 복제본이 없는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터

다음 CLI 코드는 복제본이 없는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 캐시 클러스터를 생성합니다.

> **Note**
>
> r6gd 패밀리의 노드 유형을 사용하여 클러스터를 생성하는 경우 `data-tiering-enabled` 파라미터를 전달해야 합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache create-cache-cluster \
--cache-cluster-id my-cluster \
--cache-node-type cache.r4.large \
--engine redis \
--num-cache-nodes 1 \
--cache-parameter-group default.redis6.x \
--snapshot-arns arn:aws:s3:::my_bucket/snapshot.rdb
```

Windows의 경우:

```
aws elasticache create-cache-cluster ^
--cache-cluster-id my-cluster ^
--cache-node-type cache.r4.large ^
--engine redis ^
--num-cache-nodes 1 ^
```

```
--cache-parameter-group default.redis6.x ^
--snapshot-arns arn:aws:s3:::my_bucket/snapshot.rdb
```

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 생성(AWS CLI)

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터(API/CLI: 복제 그룹)는 `create-cache-cluster` 작업을 사용하여 생성할 수 없습니다. Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터(API/CLI: 복제 그룹)를 생성하려면 섹션을 참조하세요처음부터 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹 생성(AWS CLI).

자세한 내용은 ElastiCache 참조 주제 를 참조 AWS CLI 하세요`create-replication-group`.

Valkey 또는 Redis용 클러스터 생성OSS(ElastiCache API)

를 사용하여 클러스터를 생성하려면 `CreateCacheCluster` 작업을 ElastiCache API사용합니다.

> **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. 에서 클러스터 삭제 ElastiCache을 참조하세요.

주제

- Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 캐시 클러스터 생성(ElastiCache API)
- Valkey 또는 Redis에서 캐시 클러스터 생성OSS(클러스터 모드 활성화됨)(ElastiCache API)

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 캐시 클러스터 생성(ElastiCache API)

다음 코드는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 캐시 클러스터()를 생성합니다 ElastiCache API.

줄바꿈은 가독성을 높이기 위해 추가되었습니다.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=CreateCacheCluster
    &CacheClusterId=my-cluster
    &CacheNodeType=cache.r4.large
    &CacheParameterGroup=default.redis3.2
    &Engine=redis
```

```
    &EngineVersion=3.2.4
    &NumCacheNodes=1
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &SnapshotArns.member.1=arn%3Aaws%3As3%3A%3A%3AmyS3Bucket%2Fdump.rdb
    &Timestamp=20150508T220302Z
    &Version=2015-02-02
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Credential=<credential>
    &X-Amz-Date=20150508T220302Z
    &X-Amz-Expires=20150508T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Signature=<signature>
```

Valkey 또는 Redis에서 캐시 클러스터 생성OSS(클러스터 모드 활성화됨)(ElastiCache API)

`CreateCacheCluster` 작업을 사용하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터(API/CLI: 복제 그룹)를 생성할 수 없습니다. Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터(API/CLI: 복제 그룹)를 생성하려면 섹션을 참조하세요Valkey 또는 RedisOSS(클러스터 모드 활성화됨)에서 처음부터 복제 그룹 생성(ElastiCache API).

자세한 내용은 참조 주제 를 ElastiCache API 참조하세요`CreateReplicationGroup`.

Memcached용 클러스터 생성

다음 예제에서는 및 를 사용하여 클러스터를 생성하는 방법을 보여줍니다 AWS Management Console AWS CLI ElastiCache API.

Memcached 클러스터 생성(콘솔)

Memcached 엔진을 사용하는 경우 Amazon은 여러 노드를 통해 데이터를 수평으로 분할할 수 있도록 ElastiCache 지원합니다. Memcached는 Auto Discovery를 지원하므로 각 노드의 엔드포인트를 추적할 필요가 없습니다. Memcached가 각 노드의 엔드포인트를 추적하여 노드가 추가되거나 제거되면 엔드포인트 목록을 업데이트합니다. 클러스터와 상호 작용이 필요한 모든 애플리케이션은 구성 엔드포인트입니다.

Memcached 클러스터를 생성하려면 클러스터 생성의 단계를 수행하세요.

클러스터의 상태를 사용할 수 있게 되면 즉시 Amazon에 EC2 액세스 권한을 부여하고 연결한 다음 사용을 시작할 수 있습니다. 자세한 내용은 단계 3. 클러스터에 대한 액세스 권한 부여 및 4단계. 클러스터의 노드에 연결 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. 에서 클러스터 삭제 ElastiCache을 참조하세요.

클러스터 생성(AWS CLI)

를 사용하여 클러스터를 생성하려면 `create-cache-cluster` 명령을 AWS CLI사용합니다.

> **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. 에서 클러스터 삭제 ElastiCache을 참조하세요.

Memcached 캐시 클러스터 생성(AWS CLI)

다음 CLI 코드는 3개의 노드로 Memcached 캐시 클러스터를 생성합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache create-cache-cluster \
--cache-cluster-id my-cluster \
--cache-node-type cache.r4.large \
--engine memcached \
--engine-version 1.4.24 \
--cache-parameter-group default.memcached1.4 \
--num-cache-nodes 3
```

Windows의 경우:

```
aws elasticache create-cache-cluster ^
--cache-cluster-id my-cluster ^
--cache-node-type cache.r4.large ^
--engine memcached ^
--engine-version 1.4.24 ^
--cache-parameter-group default.memcached1.4 ^
```

```
--num-cache-nodes 3
```

Memcached용 클러스터 생성(ElastiCache API)

를 사용하여 클러스터를 생성하려면 `CreateCacheCluster` 작업을 ElastiCache API사용합니다.

> **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. 에서 클러스터 삭제 ElastiCache을 참조하세요.

주제

- Memcached 캐시 클러스터 생성(ElastiCache API)

Memcached 캐시 클러스터 생성(ElastiCache API)

다음 코드는 3개의 노드()가 있는 Memcached 클러스터를 생성합니다ElastiCache API.

줄바꿈은 가독성을 높이기 위해 추가되었습니다.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=CreateCacheCluster
    &CacheClusterId=my-cluster
    &CacheNodeType=cache.r4.large
    &Engine=memcached
    &NumCacheNodes=3
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150508T220302Z
    &Version=2015-02-02
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Credential=<credential>
    &X-Amz-Date=20150508T220302Z
    &X-Amz-Expires=20150508T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Signature=<signature>
```

ElastiCache 클러스터의 세부 정보 보기

ElastiCache 콘솔 AWS CLI또는 를 사용하여 하나 이상의 클러스터에 대한 세부 정보를 볼 수 있습니다 ElastiCache API.

Memcached 클러스터의 세부 정보 보기(콘솔)

ElastiCache 콘솔, AWS CLI 용 ElastiCache또는 를 사용하여 Memcached 클러스터의 세부 정보를 볼 수 있습니다 ElastiCache API.

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

Memcached 클러스터의 세부 정보를 보려면

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 오른쪽 상단 모서리의 목록에서 관심 있는 AWS 리전을 선택합니다.
3. ElastiCache 엔진 대시보드에서 Memcached 를 선택합니다. 이렇게 하면 Memcached 엔진에서 실행 중인 모든 클러스터 목록이 표시됩니다.
4. 클러스터의 세부 정보를 보려면 클러스터 이름의 왼쪽에 있는 상자를 선택합니다.
5. 노드 정보를 보려면 노드 상태 및 엔드포인트에 대한 정보를 표시하는 노드 탭을 선택합니다.
6. 지표를 보려면 클러스터 내 모든 노드 관련 지표를 표시하는 지표 탭을 선택합니다. 자세한 내용은 CloudWatch 지표 사용 모니터링 단원을 참조하세요.
7. 네트워크 및 보안 탭을 선택하여 클러스터의 네트워크 연결 및 서브넷 그룹 구성과 VPC 보안 그룹에 대한 세부 정보를 봅니다. 자세한 내용은 서브넷 및 서브넷 그룹 단원을 참조하십시오.
8. 유지 관리 탭을 선택하면 클러스터의 유지 관리 설정에 대한 세부 정보를 볼 수 있습니다. 자세한 내용은 ElastiCache 클러스터 유지 관리 단원을 참조하십시오.
9. 태그 탭을 선택하면 클러스터 리소스에 적용된 모든 태그에 대한 세부 정보를 볼 수 있습니다. 자세한 내용은 ElastiCache 리소스 태그 지정 단원을 참조하십시오.

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 세부 정보 보기(콘솔)

ElastiCache 콘솔, AWS CLI 용 또는 를 사용하여 Valkey ElastiCache또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터의 세부 정보를 볼 수 있습니다 ElastiCache API.

다음 절차에서는 ElastiCache 콘솔을 사용하여 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터의 세부 정보를 보는 방법을 자세히 설명합니다.

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터의 세부 정보를 보려면

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. ElastiCache 엔진 대시보드에서 Valkey 또는 RedisOSS를 선택하여 해당 엔진에서 실행 중인 모든 클러스터 목록을 표시합니다.
3. 클러스터의 세부 정보를 보려면 클러스터 이름의 왼쪽에 있는 확인란을 선택합니다. 클러스터링된 Valkey 또는 클러스터링된 Redis가 아닌 Valkey 또는 Redis OSS 엔진을 실행하는 클러스터를 선택해야 합니다OSS. 그러면 클러스터의 기본 엔드포인트를 포함하여 클러스터에 대한 세부 정보가 표시됩니다.
4. 노드 정보를 보려면
   a. 클러스터의 이름을 선택합니다.
   b. 샤드 및 노드(Shards and nodes) 탭을 선택합니다. 그러면 클러스터에서 읽으려면 사용해야 하는 노드의 엔드포인트를 포함하여 각 노드의 세부 정보가 표시됩니다.
5. 지표를 보려면 클러스터 내 모든 노드 관련 지표를 표시하는 지표 탭을 선택합니다. 자세한 내용은 CloudWatch 지표 사용 모니터링 단원을 참조하세요.
6. 로그를 보려면 클러스터가 느린 로그를 선택하는지, 엔진 로그를 선택하는지를 보여주고 관련 세부 정보를 제공해주는 로그 탭을 선택합니다. 자세한 내용은 로그 전달 단원을 참조하십시오.
7. 네트워크 및 보안 탭을 선택하면 클러스터의 네트워크 연결 및 서브넷 그룹 구성에 대한 세부 정보를 볼 수 있습니다. 자세한 내용은 서브넷 및 서브넷 그룹 단원을 참조하십시오.
8. 유지 관리 탭을 선택하면 클러스터의 유지 관리 설정에 대한 세부 정보를 볼 수 있습니다. 자세한 내용은 ElastiCache 클러스터 유지 관리 단원을 참조하십시오.
9. 서비스 업데이트 탭을 선택하면 사용 가능한 모든 서비스 업데이트에 대한 세부 정보와 권장 적용 기한을 볼 수 있습니다. 자세한 내용은 의 서비스 업데이트 ElastiCache 단원을 참조하십시오.
10. 태그 탭을 선택하면 클러스터 리소스에 적용된 모든 태그에 대한 세부 정보를 볼 수 있습니다. 자세한 내용은 ElastiCache 리소스 태그 지정 단원을 참조하십시오.

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터(콘솔)에 대한 세부 정보 보기

ElastiCache 콘솔, AWS CLI 용 또는 를 사용하여 Valkey ElastiCache또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 세부 정보를 볼 수 있습니다 ElastiCache API.

다음 절차에서는 ElastiCache 콘솔을 사용하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 세부 정보를 보는 방법을 자세히 설명합니다.

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 세부 정보를 보려면

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 오른쪽 상단 모서리의 목록에서 관심 있는 AWS 리전을 선택합니다.
3. ElastiCache 엔진 대시보드에서 Valkey 또는 RedisOSS를 선택하여 해당 엔진에서 실행 중인 모든 클러스터 목록을 표시합니다.
4. Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 세부 정보를 보려면 클러스터 이름 왼쪽의 상자를 선택합니다. Valkey 또는 Clustered Redis OSS 엔진을 실행하는 클러스터를 선택해야 합니다.

   화면에서 아래의 클러스터를 확대하여 클러스터 구성 엔드포인트를 포함해 클러스터에 대한 세부 정보를 표시합니다.
5. 클러스터의 샤드 목록과 각 샤드 내 노드 개수를 보려면 샤드 및 노드 탭을 선택합니다.
6. 노드의 특정 정보를 보려면

   - 샤드 ID를 선택합니다.

     그러면 클러스터에서 데이터를 읽는 데 사용해야 하는 각 노드의 엔드포인트를 포함하여 각 노드에 대한 정보가 표시됩니다.
7. 지표를 보려면 클러스터 내 모든 노드 관련 지표를 표시하는 지표 탭을 선택합니다. 자세한 내용은 CloudWatch 지표 사용 모니터링 단원을 참조하세요.
8. 로그를 보려면 클러스터가 느린 로그를 선택하는지, 엔진 로그를 선택하는지를 보여주고 관련 세부 정보를 제공해주는 로그 탭을 선택합니다. 자세한 내용은 로그 전달 단원을 참조하십시오.
9. 네트워크 및 보안 탭을 선택하여 클러스터의 네트워크 연결 및 서브넷 그룹 구성, VPC 보안 그룹 및 클러스터에서 활성화된 암호화 방법에 대한 세부 정보를 확인합니다. 자세한 내용은 서브넷 및 서브넷 그룹 및 Amazon의 데이터 보안 ElastiCache 단원을 참조하세요.
10. 유지 관리 탭을 선택하면 클러스터의 유지 관리 설정에 대한 세부 정보를 볼 수 있습니다. 자세한 내용은 ElastiCache 클러스터 유지 관리 단원을 참조하십시오.
11. 서비스 업데이트 탭을 선택하면 사용 가능한 모든 서비스 업데이트에 대한 세부 정보와 권장 적용 기한을 볼 수 있습니다. 자세한 내용은 의 서비스 업데이트 ElastiCache 단원을 참조하십시오.
12. 태그 탭을 선택하면 클러스터 리소스에 적용된 모든 태그에 대한 세부 정보를 볼 수 있습니다. 자세한 내용은 ElastiCache 리소스 태그 지정 단원을 참조하십시오.

ElastiCache 클러스터의 세부 정보 보기(AWS CLI)

다음 코드는 에 대한 세부 정보를 나열합니다.*my-cluster*:

```
aws elasticache describe-cache-clusters --cache-cluster-id my-cluster
```

Replace *my-cluster* `create-cache-cluster` 명령을 사용하여 1개의 캐시 노드와 0개의 샤드로 클러스터가 생성되는 경우의 클러스터 이름.

```
{
    "CacheClusters": [
        {
            "CacheClusterStatus": "available",
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "wed:12:00-wed:13:00",
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "08:30-09:30",
            "TransitEncryptionEnabled": false,
            "AtRestEncryptionEnabled": false,
            "CacheClusterId": "my-cluster1",
            "CacheClusterCreateTime": "2018-02-26T21:06:43.420Z",
            "PreferredAvailabilityZone": "us-west-2c",
            "AuthTokenEnabled": false,
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
           "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis3.2"
            },
            "SnapshotRetentionLimit": 0,
            "AutoMinorVersionUpgrade": true,
            "EngineVersion": "3.2.10",
            "CacheSecurityGroups": [],
```

```
            "NumCacheNodes": 1
        }
```

```
{
    "CacheClusters": [
        {
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "AuthTokenEnabled": false,
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "12:30-13:30",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterCreateTime": "2018-02-26T21:13:24.250Z",
            "CacheClusterStatus": "available",
            "AtRestEncryptionEnabled": false,
            "PreferredAvailabilityZone": "us-west-2a",
            "TransitEncryptionEnabled": false,
            "ReplicationGroupId": "my-cluster2",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "sun:08:30-sun:09:30",
            "CacheClusterId": "my-cluster2-001",
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
            "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis6.x"
            },
            "SnapshotRetentionLimit": 0,
            "EngineVersion": "6.0",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        },
        {
            "SecurityGroups": [
                {
```

```
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "AuthTokenEnabled": false,
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "12:30-13:30",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterCreateTime": "2018-02-26T21:13:24.250Z",
            "CacheClusterStatus": "available",
            "AtRestEncryptionEnabled": false,
            "PreferredAvailabilityZone": "us-west-2b",
            "TransitEncryptionEnabled": false,
            "ReplicationGroupId": "my-cluster2",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "sun:08:30-sun:09:30",
            "CacheClusterId": "my-cluster2-002",
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
            "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis6.x"
            },
            "SnapshotRetentionLimit": 0,
            "EngineVersion": "6.0",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        },
        {
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "AuthTokenEnabled": false,
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "12:30-13:30",
```

```
            "AutoMinorVersionUpgrade": true,
            "CacheClusterCreateTime": "2018-02-26T21:13:24.250Z",
            "CacheClusterStatus": "available",
            "AtRestEncryptionEnabled": false,
            "PreferredAvailabilityZone": "us-west-2c",
            "TransitEncryptionEnabled": false,
            "ReplicationGroupId": "my-cluster2",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "sun:08:30-sun:09:30",
            "CacheClusterId": "my-cluster2-003",
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
            "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis3.2"
            },
            "SnapshotRetentionLimit": 0,
            "EngineVersion": "3.2.10",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        }
```

```
{
    "CacheClusters": [
        {
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "AuthTokenEnabled": true,
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "12:30-13:30",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterCreateTime": "2018-02-26T21:17:01.439Z",
            "CacheClusterStatus": "available",
            "AtRestEncryptionEnabled": true,
            "PreferredAvailabilityZone": "us-west-2a",
```

```
            "TransitEncryptionEnabled": true,
            "ReplicationGroupId": "my-cluster3",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "thu:11:00-thu:12:00",
            "CacheClusterId": "my-cluster3-0001-001",
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
            "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis6.x.cluster.on"
            },
            "SnapshotRetentionLimit": 0,
            "EngineVersion": "6.0",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        },
        {
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "AuthTokenEnabled": true,
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "12:30-13:30",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterCreateTime": "2018-02-26T21:17:01.439Z",
            "CacheClusterStatus": "available",
            "AtRestEncryptionEnabled": true,
            "PreferredAvailabilityZone": "us-west-2b",
            "TransitEncryptionEnabled": true,
            "ReplicationGroupId": "my-cluster3",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "thu:11:00-thu:12:00",
            "CacheClusterId": "my-cluster3-0001-002",
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
             "DataTiering": "disabled",
            "CacheParameterGroup": {
```

```
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis3.2.cluster.on"
            },
            "SnapshotRetentionLimit": 0,
            "EngineVersion": "3.2.6",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        },
        {
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "AuthTokenEnabled": true,
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "12:30-13:30",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterCreateTime": "2018-02-26T21:17:01.439Z",
            "CacheClusterStatus": "available",
            "AtRestEncryptionEnabled": true,
            "PreferredAvailabilityZone": "us-west-2c",
            "TransitEncryptionEnabled": true,
            "ReplicationGroupId": "my-cluster3",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "thu:11:00-thu:12:00",
            "CacheClusterId": "my-cluster3-0001-003",
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
             "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis6.x.cluster.on"
            },
            "SnapshotRetentionLimit": 0,
            "EngineVersion": "6.0",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        },
```

```
        {
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "AuthTokenEnabled": true,
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "12:30-13:30",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterCreateTime": "2018-02-26T21:17:01.439Z",
            "CacheClusterStatus": "available",
            "AtRestEncryptionEnabled": true,
            "PreferredAvailabilityZone": "us-west-2b",
            "TransitEncryptionEnabled": true,
            "ReplicationGroupId": "my-cluster3",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "thu:11:00-thu:12:00",
            "CacheClusterId": "my-cluster3-0002-001",
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
             "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis6.x.cluster.on"
            },
            "SnapshotRetentionLimit": 0,
            "EngineVersion": "6.0",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        },
        {
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
```

```
            "AuthTokenEnabled": true,
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "12:30-13:30",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterCreateTime": "2018-02-26T21:17:01.439Z",
            "CacheClusterStatus": "available",
            "AtRestEncryptionEnabled": true,
            "PreferredAvailabilityZone": "us-west-2c",
            "TransitEncryptionEnabled": true,
            "ReplicationGroupId": "my-cluster3",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "thu:11:00-thu:12:00",
            "CacheClusterId": "my-cluster3-0002-002",
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
             "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis3.2.cluster.on"
            },
            "SnapshotRetentionLimit": 0,
            "EngineVersion": "3.2.6",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        },
        {
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "AuthTokenEnabled": true,
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "12:30-13:30",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterCreateTime": "2018-02-26T21:17:01.439Z",
            "CacheClusterStatus": "available",
            "AtRestEncryptionEnabled": true,
            "PreferredAvailabilityZone": "us-west-2a",
            "TransitEncryptionEnabled": true,
```

```
            "ReplicationGroupId": "my-cluster3",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "thu:11:00-thu:12:00",
            "CacheClusterId": "my-cluster3-0002-003",
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
             "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis6.x.cluster.on"
            },
            "SnapshotRetentionLimit": 0,
            "EngineVersion": "6.0",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        }
    ]
}
```

클러스터가 를 사용하여 생성되는 경우 AWS Management Console (클러스터 노드가 하나 이상의 샤드로 활성화 또는 비활성화됨) 다음 명령을 사용하여 클러스터의 세부 정보를 설명합니다(교체 *my-cluster* 복제 그룹 이름(클러스터 이름):

```
aws elasticache describe-replication-groups --replication-group-id my-cluster
```

자세한 내용은 ElastiCache 주제 의 섹션을 참조 AWS CLI 하세요describe-cache-clusters.

ElastiCache 클러스터의 세부 정보 보기(ElastiCache API)

DescribeCacheClusters 작업을 사용하여 클러스터의 세부 정보를 볼 수 있습니다 ElastiCache API. CacheClusterId 파라미터가 포함되면 지정한 클러스터의 세부 정보가 반환됩니다. CacheClusterId 파라미터가 생략되면 클러스터 최대 MaxRecords개(기본값 100)의 세부 정보가 반환됩니다. MaxRecords의 값은 20 이상 또는 100 이하여야 합니다.

다음 코드는 my-cluster의 세부 정보를 나열합니다.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheClusters
   &CacheClusterId=my-cluster
   &Version=2015-02-02
   &SignatureVersion=4
```

```
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

다음 코드는 클러스터 최대 25개의 세부 정보를 나열합니다.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheClusters
   &MaxRecords=25
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

자세한 내용은 참조 주제 를 ElastiCache API 참조하세요DescribeCacheClusters.

ElastiCache 클러스터 수정

ElastiCache 클러스터에서 노드를 추가하거나 제거하는 것 외에도 보안 그룹 추가, 유지 관리 기간 또는 파라미터 그룹 변경과 같은 다른 변경을 수행해야 하는 경우가 있을 수 있습니다.

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

클러스터의 파라미터를 변경하면 변경 사항은 또는 클러스터가 재시작된 즉시 또는 그 이후에 클러스터에 적용됩니다. 이는 클러스터의 파라미터 그룹 자체에서 변경하든 파라미터 값을 클러스터의 파라미터 그룹 내에서 변경하든 마찬가지입니다. 특정 파라미터 변경 사항이 적용되는 시기를 확인하려면 Memcached 특정 파라미터 및 의 테이블에서 세부 정보 열의 변경 사항 적용 섹션을 참조하세요Valkey 및 Redis OSS 파라미터. 클러스터의 노드 재부팅에 관한 자세한 정보는 노드 재부팅 섹션을 참조하세요.

사용 ElastiCache AWS Management Console

클러스터를 수정하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 오른쪽 상단 모서리의 목록에서 수정하려는 클러스터가 있는 AWS 리전을 선택합니다.
3. 탐색 창에서 수정하려는 클러스터에서 실행 중인 엔진을 선택합니다.

   선택한 엔진의 클러스터 목록이 나타납니다.
4. 클러스터 목록에서 수정할 클러스터의 해당 이름을 선택합니다.
5. 작업을 선택한 다음 수정을 선택합니다.

   [Modify Cluster] 창이 나타납니다.
6. Modify Cluster(클러스터 수정) 창에서 원하는 내용을 수정합니다. 옵션에는 다음이 포함됩니다.

   - 설명
   - 클러스터 모드 - 클러스터 모드를 비활성화에서 활성화로 수정하려면 먼저 클러스터 모드를 호환으로 설정해야 합니다.

     호환 모드를 사용하면 클러스터 모드가 활성화되고 클러스터 모드가 비활성화된 상태에서 Valkey 또는 Redis OSS 클라이언트를 연결할 수 있습니다. 클러스터 모드 활성화를 사용하도록 모든 Valkey 또는 Redis OSS 클라이언트를 마이그레이션한 후 클러스터 모드 구성을 완료하고 클러스터 모드를 활성화됨으로 설정할 수 있습니다.
   - 엔진 버전 호환성

> **Important**
>
> 새로운 엔진 버전으로 업그레이드할 수 있습니다. 메이저 엔진 버전을 업그레이드하는 경우(예: 5.0.6에서 6.0으로 업그레이드) 새 엔진 버전과 호환되는 파라미터 그룹 패밀리를 선택해야 합니다. 이에 대한 자세한 내용은 용 버전 관리 ElastiCache 섹션을 참조하세요. 하지만 기존의 클러스터를 삭제하고 새로 만들지 않는 한 이전 엔진 버전으로 다운그레이드할 수 없습니다.

- VPC 보안 그룹(들)
- Parameter Group
- 노드 유형(Node Type)

> **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 계층화를 참조하세요.

- 다중 AZ
- 자동 장애 조치(클러스터 모드 비활성화됨 전용)
- 자동 백업 활성화
- Backup 노드 ID
- 백업 보존 기간
- 백업 기간
- SNS 알림 주제

- Memcached 엔진 버전 호환성
- 네트워크 유형

> **Note**
>
> 에서 IPv4로 전환하는 경우 와 호환되는 서브넷 그룹을 선택하거나 생성IPv6해야 합니다IPv6. 자세한 내용은 에서 네트워크 유형 선택 ElastiCache 단원을 참조하십시오.

- VPC 보안 그룹(들)

- Parameter Group
- 유지 관리 기간
- SNS 알림 주제

[Apply Immediately] 상자는 엔진 버전 수정에만 적용됩니다. 변경 사항을 즉시 적용하려면 Apply Immediately(즉시 적용) 확인란을 선택합니다. 이 상자를 선택하지 않으면 다음 번 유지 관리 기간에 엔진 버전 수정이 적용됩니다. 유지 관리 기간 변경과 같은 다른 수정은 즉시 적용됩니다.

Redis에 대한 로그 전송을 활성화/비활성화하려면

1. 클러스터 목록에서 수정할 클러스터를 선택합니다. 클러스터 이름을 선택합니다(그 옆의 확인란 아님).
2. 클러스터 세부 정보 페이지에서 로그 탭을 선택합니다.
3. 느린 로그를 활성화 또는 비활성화하려면 활성화 또는 비활성화를 선택합니다.

   사용 설정을 선택한 경우

   a. 로그 형식에서 JSON 또는 텍스트 중 하나를 선택합니다.
   b. 로그 대상 유형에서 CloudWatch 로그 또는 Kinesis Firehose를 선택합니다.
   c. 로그 대상 에서 새로 생성을 선택하고 CloudWatchLogs 로그 그룹 이름 또는 Kinesis Data Firehose 스트림 이름을 입력할 수 있습니다. 기존 선택 을 선택한 다음 CloudWatchLogs 로그 그룹 이름 또는 Kinesis Data Firehose 스트림 이름을 선택할 수도 있습니다.
   d. 활성화를 선택합니다.

Redis에 대한 구성을 변경하려면:

1. 수정을 선택합니다.
2. 로그 형식에서 JSON 또는 텍스트 중 하나를 선택합니다.
3. 대상 유형에서 CloudWatch 로그 또는 Kinesis Firehose를 선택합니다.
4. 로그 대상 에서 새로 생성을 선택하고 CloudWatchLogs 로그 그룹 이름 또는 Kinesis Data Firehose 스트림 이름을 입력합니다. 또는 기존 선택을 선택한 다음 CloudWatchLogs 로그 그룹 이름 또는 Kinesis Data Firehose 스트림 이름을 선택합니다.

와 AWS CLI 함께 사용 ElastiCache

작업을 사용하여 기존 클러스터를 AWS CLI `modify-cache-cluster` 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.

> **Important**
>
> 최신 Memcached 엔진 버전으로 업그레이드할 수 있습니다. 이에 대한 자세한 내용은 용 버전 관리 ElastiCache 섹션을 참조하세요. 하지만 기존의 클러스터를 삭제하고 새로 만들지 않는 한 이전 엔진 버전으로 다운그레이드할 수 없습니다.

> **Important**
>
> 최신 Valkey 또는 Redis OSS 엔진 버전으로 업그레이드할 수 있습니다. Redis OSS 5.0.6에서 Redis OSS 6.0으로 주요 엔진 버전을 업그레이드하는 경우 새 엔진 버전과 호환되는 파라미터 그룹 패밀리를 선택해야 합니다. 이에 대한 자세한 내용은 용 버전 관리 ElastiCache 섹션을 참조하세요. 하지만 기존의 클러스터를 삭제하고 새로 만들지 않는 한 이전 엔진 버전으로 다운그레이드할 수 없습니다.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-cache-cluster \
    --cache-cluster-id my-cluster \
    --preferred-maintenance-window sun:23:00-mon:02:00
```

Windows의 경우:

```
aws elasticache modify-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --preferred-maintenance-window sun:23:00-mon:02:00
```

`--apply-immediately` 파라미터는 노드 유형, 엔진 버전의 수정 및 클러스터의 노드 수 변경에만 적용됩니다. 이 변경 사항을 즉시 적용하려면 `--apply-immediately` 파라미터를 사용하세요. 다음 번 유지 관리 기간으로 이 변경을 연기하려면 `--no-apply-immediately` 파라미터를 사용하세요. 유지 관리 기간 변경과 같은 다른 수정은 즉시 적용됩니다.

자세한 내용은 ElastiCache 주제 의 섹션을 참조 AWS CLI 하세요`modify-cache-cluster`.

사용 ElastiCache API

`ModifyCacheCluster` 작업을 사용하여 기존 클러스터를 ElastiCache API 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.

> **Important**
>
> 최신 Memcached 엔진 버전으로 업그레이드할 수 있습니다. 이에 대한 자세한 내용은 용 버전 관리 ElastiCache 섹션을 참조하세요. 하지만 기존의 클러스터를 삭제하고 새로 만들지 않는 한 이전 엔진 버전으로 다운그레이드할 수 없습니다.

> **Important**
>
> 최신 Valkey 또는 Redis OSS 엔진 버전으로 업그레이드할 수 있습니다. Redis OSS 5.0.6에서 Redis OSS 6.0으로 주요 엔진 버전을 업그레이드하는 경우 새 엔진 버전과 호환되는 파라미터 그룹 패밀리를 선택해야 합니다. 이에 대한 자세한 내용은 용 버전 관리 ElastiCache 섹션을 참조하세요. 하지만 기존의 클러스터를 삭제하고 새로 만들지 않는 한 이전 엔진 버전으로 다운그레이드할 수 없습니다.

줄바꿈은 가독성을 높이기 위해 추가되었습니다.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyCacheCluster
    &CacheClusterId=my-cluster
    &PreferredMaintenanceWindow=sun:23:00-mon:02:00
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150901T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20150202T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20150901T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

`ApplyImmediately` 파라미터는 노드 유형, 엔진 버전의 수정 및 클러스터의 노드 수 변경에만 적용됩니다. 이 변경 사항을 즉시 적용하려면 `ApplyImmediately` 파라미터를 `true`로 설정하세요. 다음 번 유지 관리 기간으로 이 변경을 연기하려면 `ApplyImmediately` 파라미터를 `false`로 설정하세요. 유지 관리 기간 변경과 같은 다른 수정은 즉시 적용됩니다.

자세한 내용은 참조 주제 를 ElastiCache API 참조하세요`ModifyCacheCluster`.

ElastiCache 클러스터에 노드 추가

Memcached 클러스터에 노드를 추가하면 클러스터의 파티션 수가 증가합니다. 클러스터의 파티션 수를 변경할 때는 키스페이스의 일부를 다시 매핑해야 정확한 노드에 매핑됩니다. 키스페이스를 일시적으로 다시 매핑하면 클러스터의 캐시 누락 수가 증가합니다. 자세한 내용은 효율적인 로드 밸런싱을 위해 ElastiCache 클라이언트 구성(Memcached) 단원을 참조하십시오.

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터를 재구성하려면 섹션을 참조하세요. Valkey 또는 Redis에서 클러스터 크기 조정OSS(클러스터 모드 활성화됨)

ElastiCache 관리 콘솔, AWS CLI 또는 ElastiCache API 를 사용하여 클러스터에 노드를 추가할 수 있습니다.

사용 ElastiCache AWS Management Console

단일 노드 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터에 노드를 추가하려면(복제 활성화되지 않은 클러스터) 먼저 복제를 추가한 다음 복제 노드를 추가하는 2단계 프로세스입니다.

주제

- 샤드 없이 Valkey 또는 Redis OSS 클러스터에 복제를 추가하려면
- ElastiCache 클러스터에 노드를 추가하려면(콘솔)

다음 절차에서는 복제가 활성화되지 OSS 않은 단일 노드 Valkey 또는 Redis에 복제를 추가합니다. 복제를 추가할 때 기존의 노드는 복제가 활성화된 클러스터의 기본 노드가 됩니다. 복제를 추가한 후에는 최대 5개의 복제본 노드를 클러스터에 추가할 수 있습니다.

샤드 없이 Valkey 또는 Redis OSS 클러스터에 복제를 추가하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 Valkey 클러스터 또는 Redis OSS 클러스터를 선택합니다.

   해당 엔진을 실행하는 클러스터 목록이 표시됩니다.
3. 노드를 추가하고자 하는 클러스터의 이름 왼쪽에 있는 상자가 아닌 클러스터의 이름을 선택합니다.

   복제가 활성화되지 않은 Redis OSS 클러스터의 경우 다음 사항이 적용됩니다.

   - 클러스터링된 Redis 가 OSS아닌 Redis 를 실행 중입니다OSS.

- 제로 샤드를 포함합니다.

  클러스터에 샤드가 있으면 복제가 이미 활성화되어 있으며 ElastiCache 클러스터에 노드를 추가하려면(콘솔)에서 계속할 수 있습니다.

4. [Add replication]을 선택합니다.
5. 복제가 활성화된 이 클러스터에 대한 설명을 Add Replication(복제 추가)에 입력합니다.
6. 추가를 선택합니다.

   클러스터 상태가 [available]로 돌아오면 다음 절차에서 계속 진행하고 복제본을 클러스터에 추가할 수 있습니다.

ElastiCache 클러스터에 노드를 추가하려면(콘솔)

다음 절차에 따라 클러스터에 노드를 추가할 수 있습니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 노드를 추가하려는 클러스터에서 실행 중인 엔진을 탐색 창에서 선택합니다.

   선택한 엔진을 실행하는 클러스터 목록이 표시됩니다.
3. 클러스터 목록에서 노드를 추가할 클러스터의 해당 이름을 선택합니다.

   클러스터가 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터인 경우 섹션을 참조하세요Valkey 또는 Redis에서 클러스터 크기 조정OSS(클러스터 모드 활성화됨).

   클러스터가 샤드가 없는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터인 경우 먼저 에서 단계를 완료합니다샤드 없이 Valkey 또는 Redis OSS 클러스터에 복제를 추가하려면.
4. [Add node]를 선택합니다.
5. Add Node(노드 추가) 대화 상자에 요청을 받은 정보를 입력합니다.
6. 이 노드를 즉시 추가하려면 [Apply Immediately - Yes] 버튼을 선택하고, 클러스터의 다음 유지 관리 기간 중에 이 노드를 추가하려면 [No]를 선택합니다.

신규 추가 및 제거 요청이 대기 중 요청에 미치는 영향

시나리오	대기 중 작업	신규 요청	결과
시나리오 1	삭제	삭제	신규 삭제 요청(대기 중 또는 즉시)은 대기 중 삭제 요청을 대체합니다. 예를 들어 노드 0001, 0003 및 0007에서 삭제 요청이 대기 중일 때 노드 0002 및 0004를 삭제하는 신규 요청이 생성될 경우 노드 0002 및 0004만 삭제됩니다. 노드 0001, 0003 및 0007은 삭제되지 않습니다.
시나리오 2	삭제	생성	신규 생성 요청(대기 중 또는 즉시)은 대기 중 삭제 요청을 대체합니다. 예를 들어 노드 0001, 0003 및 0007에서 삭제 요청이 대기 중일 때 노드를 생성하는 신규 요청이 생성될 경우 새 노드가 생성되고 노드 0001, 0003 및 0007은 삭제되지 않습니다.
시나리오 3	생성	삭제	신규 삭제 요청(대기 중 또는 즉시)은 대기 중 생성 요청을 대체합니다. 예를 들어 노드 2개를 생성하는 요청이 대기 중일 때 노드 0003을 삭제하는 요청이 생성될 경우 새 노드는 생성되지 않고 노드 0003이 삭제됩니다.

시나리오	대기 중 작업	신규 요청	결과
시나리오 4	생성	생성	신규 생성 요청은 대기 중 생성 요청에 추가됩니다. 예를 들어 노드 2개를 생성하는 요청이 대기 중일 때 노드 3개를 생성하는 신규 요청이 생성될 경우 신규 요청이 대기 중 요청에 추가되어 노드 5개가 생성됩니다. **Important** 신규 생성 요청이 Apply Immediately - Yes(즉시 적용 - 예)로 설정된 경우 모든 생성 요청이 즉시 실행됩니다. 신규 생성 요청이 Apply Immediately - No(즉시 적용 - 아니요)로 설정된 경우 모든 생성 요청은 대기 중 작업입니다.

어떤 작업이 대기 중인지 알아보려면 설명 탭을 선택하여 대기 중 생성 또는 삭제 작업이 몇 개인지 확인합니다. 대기 중 생성 작업과 대기 중 삭제 작업이 동시에 있을 수는 없습니다.

7. [Add] 버튼을 선택합니다.

   잠시 후 [creating] 상태로 새로운 노드가 노드 목록에 나타납니다. 노드가 표시되지 않으면 브라우저 페이지를 새로 고치십시오. 노드가 사용 가능 상태가 되면 새로운 노드를 사용할 수 있습니다.

와 AWS CLI 함께 사용 ElastiCache

를 사용하여 클러스터에 노드를 추가하려면 다음 파라미터`modify-cache-cluster`로 AWS CLI 작업을 AWS CLI사용합니다.

- `--cache-cluster-id` 노드를 추가할 캐시 클러스터의 ID입니다.
- `--num-cache-nodes` `--num-cache-nodes` 파라미터는 수정이 적용된 후 이 클러스터에 포함할 노드 수를 지정합니다. 이 클러스터에 노드를 추가하려면 `--num-cache-nodes`가 이 클러스터의 현재 노드 수보다 커야 합니다. 이 값이 현재 노드 수보다 작으면 는 파라미터`cache-node-ids-`

to-remove와 클러스터에서 제거할 노드 목록을 ElastiCache 예상합니다. 자세한 내용은 와 AWS CLI 함께 사용 ElastiCache 단원을 참조하십시오.

- --apply-immediately 또는 --no-apply-immediately 이 노드를 즉시 추가할지 아니면 다음 번 유지 관리 기간에 추가할지 지정합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-cache-cluster \
    --cache-cluster-id my-cluster \
    --num-cache-nodes 5 \
    --apply-immediately
```

Windows의 경우:

```
aws elasticache modify-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --num-cache-nodes 5 ^
    --apply-immediately
```

이 작업은 다음과 유사한 출력을 생성합니다(JSON 형식).

```
{
    "CacheCluster": {
        "Engine": "memcached",
        "CacheParameterGroup": {
            "CacheNodeIdsToReboot": [],
            "CacheParameterGroupName": "default.memcached1.4",
            "ParameterApplyStatus": "in-sync"
        },
        "CacheClusterId": "my-cluster",
        "PreferredAvailabilityZone": "us-west-2b",
        "ConfigurationEndpoint": {
            "Port": 11211,
            "Address": "rlh-mem000.7alc7bf-example.cfg.usw2.cache.amazonaws.com"
        },
        "CacheSecurityGroups": [],
        "CacheClusterCreateTime": "2016-09-21T16:28:28.973Z",
        "AutoMinorVersionUpgrade": true,
        "CacheClusterStatus": "modifying",
        "NumCacheNodes": 2,
```

```
        "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
        "SecurityGroups": [
            {
                "Status": "active",
                "SecurityGroupId": "sg-dbe93fa2"
            }
        ],
        "CacheSubnetGroupName": "default",
        "EngineVersion": "1.4.24",
        "PendingModifiedValues": {
            "NumCacheNodes": 5
        },
        "PreferredMaintenanceWindow": "sat:09:00-sat:10:00",
        "CacheNodeType": "cache.m3.medium",
         "DataTiering": "disabled",
    }
}
```

자세한 내용은 AWS CLI 주제를 참조하세요modify-cache-cluster.

와 AWS CLI 함께 사용 ElastiCache

복제가 활성화되지 않은 기존 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터에 노드를 추가하려면 먼저 기존 클러스터를 기본 클러스터로 지정하는 복제 그룹을 생성해야 합니다. 자세한 내용은 사용 가능한 Valkey 또는 Redis OSS 캐시 클러스터를 사용하여 복제 그룹 생성(AWS CLI) 단원을 참조하십시오. 복제 그룹이 [available] 상태가 되면 다음 프로세스로 계속 진행할 수 있습니다.

를 사용하여 클러스터에 노드를 추가하려면 다음 파라미터와 `increase-replica-count` 함께 AWS CLI 작업을 AWS CLI사용합니다.

- `--replication-group-id` 노드를 추가할 복제 그룹의 ID입니다.
- `--new-replica-count`는 수정이 적용된 후 이 복제 그룹에 포함할 노드 수를 지정합니다. 이 클러스터에 노드를 추가하려면 `--new-replica-count`가 이 클러스터의 현재 노드 수보다 커야 합니다.
- `--apply-immediately` 또는 `--no-apply-immediately` 이 노드를 즉시 추가할지 아니면 다음 번 유지 관리 기간에 추가할지 지정합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache increase-replica-count \
```

```
    --replication-group-id my-replication-group \
    --new-replica-count 4 \
    --apply-immediately
```

Windows의 경우:

```
aws elasticache increase-replica-count ^
    --replication-group-id my-replication-group ^
    --new-replica-count 4 ^
    --apply-immediately
```

이 작업은 다음과 유사한 출력을 생성합니다(JSON 형식).

```
{
    "ReplicationGroup": {
        "ReplicationGroupId": "node-test",
        "Description": "node-test",
        "Status": "modifying",
        "PendingModifiedValues": {},
        "MemberClusters": [
            "node-test-001",
            "node-test-002",
            "node-test-003",
            "node-test-004",
            "node-test-005"
        ],
        "NodeGroups": [
            {
                "NodeGroupId": "0001",
                "Status": "modifying",
                "PrimaryEndpoint": {
                    "Address": "node-test.zzzzzz.ng.0001.usw2.cache.amazonaws.com",
                    "Port": 6379
                },
                "ReaderEndpoint": {
                    "Address": "node-test.zzzzzz.ng.0001.usw2.cache.amazonaws.com",
                    "Port": 6379
                },
                "NodeGroupMembers": [
                    {
                        "CacheClusterId": "node-test-001",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
```

```
                            "Address": "node-
test-001.zzzzzz.0001.usw2.cache.amazonaws.com",
                            "Port": 6379
                        },
                        "PreferredAvailabilityZone": "us-west-2a",
                        "CurrentRole": "primary"
                    },
                    {
                        "CacheClusterId": "node-test-002",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Address": "node-
test-002.zzzzzz.0001.usw2.cache.amazonaws.com",
                            "Port": 6379
                        },
                        "PreferredAvailabilityZone": "us-west-2c",
                        "CurrentRole": "replica"
                    },
                    {
                        "CacheClusterId": "node-test-003",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Address": "node-
test-003.zzzzzz.0001.usw2.cache.amazonaws.com",
                            "Port": 6379
                        },
                        "PreferredAvailabilityZone": "us-west-2b",
                        "CurrentRole": "replica"
                    }
                ]
            }
        ],
        "SnapshottingClusterId": "node-test-002",
        "AutomaticFailover": "enabled",
        "MultiAZ": "enabled",
        "SnapshotRetentionLimit": 1,
        "SnapshotWindow": "07:30-08:30",
        "ClusterEnabled": false,
        "CacheNodeType": "cache.r5.large",
         "DataTiering": "disabled",
        "TransitEncryptionEnabled": false,
        "AtRestEncryptionEnabled": false,
        "ARN": "arn:aws:elasticache:us-west-2:123456789012:replicationgroup:node-test"
    }
```

```
}
```

자세한 내용은 AWS CLI 주제를 참조하세요increase-replica-count.

사용 ElastiCache API

복제가 활성화되지 않은 기존 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터에 노드를 추가하려면 먼저 기존 클러스터를 기본 클러스터로 지정하는 복제 그룹을 생성해야 합니다. 자세한 내용은 독립 실행형 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터에 복제본 추가 (ElastiCache API) 단원을 참조하십시오. 복제 그룹이 [available] 상태가 되면 다음 프로세스로 계속 진행할 수 있습니다.

클러스터에 노드를 추가하려면(ElastiCache API)

- 다음 파라미터를 사용하여 `IncreaseReplicaCount` API 작업을 호출합니다.
  - `ReplicationGroupId` 노드를 추가할 클러스터의 ID입니다.
  - `NewReplicaCount` `NewReplicaCount` 파라미터는 수정이 적용된 후 이 클러스터에 포함할 노드 수를 지정합니다. 이 클러스터에 노드를 추가하려면 `NewReplicaCount`가 이 클러스터의 현재 노드 수보다 커야 합니다. 이 값이 현재 노드 수보다 작은 경우 클러스터에서 제거할 노드 수`DecreaseReplicaCount`API와 함께 를 사용합니다.
  - `ApplyImmediately` 이 노드를 즉시 추가할지 아니면 다음 번 유지 관리 기간에 추가할지 지정합니다.
  - `Region` 노드를 추가할 클러스터의 AWS 리전을 지정합니다.

  다음 예제에서는 클러스터에 노드를 추가하기 위한 호출을 보여줍니다.

  Example

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=IncreaseReplicaCount
    &ApplyImmediately=true
    &NumCacheNodes=4
    &ReplicationGroupId=my-replication-group
    &Region=us-east-2
    &Version=2014-12-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
```

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

자세한 내용은 주제를 참조 ElastiCache API하세요IncreaseReplicaCount.

사용 ElastiCache API

클러스터에 노드를 추가하려면(ElastiCache API)

- 다음 파라미터를 사용하여 ModifyCacheCluster API 작업을 호출합니다.
  - CacheClusterId 노드를 추가할 클러스터의 ID입니다.
  - NumCacheNodes NumCachNodes 파라미터는 수정이 적용된 후 이 클러스터에 포함할 노드 수를 지정합니다. 이 클러스터에 노드를 추가하려면 NumCacheNodes가 이 클러스터의 현재 노드 수보다 커야 합니다. 이 값이 현재 노드 수보다 작으면 클러스터에서 제거할 노드 목록이 CacheNodeIdsToRemove 있는 파라미터가 ElastiCache 예상됩니다(참조Memcached와 ElastiCache API 함께 사용).
  - ApplyImmediately 이 노드를 즉시 추가할지 아니면 다음 번 유지 관리 기간에 추가할지 지정합니다.
  - Region 노드를 추가할 클러스터의 AWS 리전을 지정합니다.

  다음 예제에서는 클러스터에 노드를 추가하기 위한 호출을 보여줍니다.

  Example

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyCacheCluster
    &ApplyImmediately=true
    &NumCacheNodes=5
 &CacheClusterId=my-cluster
 &Region=us-east-2
    &Version=2014-12-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
```

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

자세한 내용은 주제를 참조 ElastiCache API하세요ModifyCacheCluster.

ElastiCache 클러스터에서 노드 제거

, 또는 를 사용하여 ValkeyOSS, Redis AWS CLI또는 Memcached 클러스터에서 노드 AWS Management Console를 삭제할 수 있습니다 ElastiCache API.

> **Note**
>
> Memcached 클러스터의 노드 수를 변경할 때마다 키스페이스의 일부라도 다시 매핑해야 정확한 노드에 매핑됩니다. Memcached 클러스터의 로드 밸런싱에 대한 자세한 내용은 효율적인 로드 밸런싱을 위해 ElastiCache 클라이언트 구성(Memcached) 섹션을 참조하세요.

사용 ElastiCache AWS Management Console

클러스터 에서 노드를 제거하려면(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 오른쪽 상단 모서리의 목록에서 노드를 제거할 클러스터의 AWS 리전을 선택합니다.
3. 노드를 제거하려는 클러스터에서 실행 중인 엔진을 탐색 창에서 선택합니다.

   선택한 엔진을 실행하는 클러스터 목록이 표시됩니다.
4. 클러스터 목록에서 노드를 제거할 클러스터 이름을 선택합니다.

   클러스터의 노드 목록이 나타납니다.
5. 제거할 노드의 노드 ID 왼쪽에 있는 상자를 선택합니다. ElastiCache 콘솔을 사용하면 한 번에 하나의 노드만 삭제할 수 있으므로 여러 노드를 선택하면 노드 삭제 버튼을 사용할 수 없습니다.

   노드 삭제 페이지가 나타납니다.
6. 노드를 삭제하려면 노드 삭제 페이지를 완료하고 노드 삭제를 선택합니다. 노드를 유지하려면 취소를 선택합니다.

   > **Important**
   >
   > Valkey 또는 Redis 를 사용하여 클러스터에서 더 이상 다중 AZ를 준수하지 않는 노드 결과를 삭제하는 OSS경우 먼저 다중 AZ 확인란을 선택 취소한 다음 노드를 삭제해야 합니다. 다중 AZ 확인란을 선택 취소하면 Auto failover(자동 장애 조치)를 활성화하도록 선택할 수 있습니다.

신규 추가 및 제거 요청이 대기 중 요청에 미치는 영향

시나리오	대기 중 작업	신규 요청	결과
시나리오 1	삭제	삭제	신규 삭제 요청(대기 중 또는 즉시)은 대기 중 삭제 요청을 대체합니다. 예를 들어 노드 0001, 0003 및 0007에서 삭제 요청이 대기 중일 때 노드 0002 및 0004를 삭제하는 신규 요청이 생성될 경우 노드 0002 및 0004만 삭제됩니다. 노드 0001, 0003 및 0007은 삭제되지 않습니다.
시나리오 2	삭제	생성	신규 생성 요청(대기 중 또는 즉시)은 대기 중 삭제 요청을 대체합니다. 예를 들어 노드 0001, 0003 및 0007에서 삭제 요청이 대기 중일 때 노드를 생성하는 신규 요청이 생성될 경우 새 노드가 생성되고 노드 0001, 0003 및 0007은 삭제되지 않습니다.
시나리오 3	생성	삭제	신규 삭제 요청(대기 중 또는 즉시)은 대기 중 생성 요청을 대체합니다. 예를 들어 노드 2개를 생성하는 요청이 대기 중일 때 노드 0003을 삭제하는 요청이 생성될 경우 새 노드는 생성되지 않고 노드 0003이 삭제됩니다.
시나리오 4	생성	생성	신규 생성 요청은 대기 중 생성 요청에 추가됩니다. 예를 들어 노드 2개를 생성하는 요청이 대기 중일 때 노드 3개를 생성하는 신규 요청이 생성될 경우 신규 요청이 대기 중 요청에 추가되어 노드 5개가 생성됩니다. **Important** 신규 생성 요청이 Apply Immediately - Yes(즉시 적용 - 예)로 설정된 경우 모든 생성 요청이 즉시 실행됩니다. 신규 생성 요청이 Apply

시나리오	대기 중 작업	신규 요청	결과
			Immediately - No(즉시 적용 - 아니요)로 설정된 경우 모든 생성 요청은 대기 중 작업입니다.

어떤 작업이 대기 중인지 알아보려면 설명 탭을 선택하여 대기 중 생성 또는 삭제 작업이 몇 개인지 확인합니다. 대기 중 생성 작업과 대기 중 삭제 작업이 동시에 있을 수는 없습니다.

와 AWS CLI 함께 사용 ElastiCache

1. 제거할 노드IDs의 를 식별합니다. 자세한 내용은 ElastiCache 클러스터의 세부 정보 보기 단원을 참조하십시오.
2. 다음 예제와 같이 제거할 노드 목록과 함께 `decrease-replica-count` CLI 작업을 사용합니다.

   명령줄 인터페이스를 사용하여 클러스터에서 노드를 제거하려면 다음 파라미터와 함께 `decrease-replica-count` 명령을 사용하세요.

   - `--replication-group-id` 노드를 제거할 복제 그룹의 ID입니다.
   - `--new-replica-count` `--new-replica-count` 파라미터는 수정이 적용된 후 이 클러스터에 포함할 노드 수를 지정합니다.
   - `--replicas-to-remove` 이 클러스터에서 제거IDs하려는 노드 목록입니다.
   - `--apply-immediately` 또는 `--no-apply-immediately` 이 노드를 즉시 제거할지 아니면 다음 번 유지 관리 기간에 제거할지 지정합니다.
   - `--region` 노드를 제거할 클러스터의 AWS 리전을 지정합니다.

   > **Note**
   >
   > 이 작업을 호출 할 때 `--replicas-to-remove` 또는 `--new-replica-count` 파라미터 중 하나만 전달할 수 있습니다.

   Linux, macOS, Unix의 경우:

```
aws elasticache decrease-replica-count \
    --replication-group-id my-replication-group \
```

```
    --new-replica-count 2 \
    --region us-east-2 \
    --apply-immediately
```

Windows의 경우:

```
aws elasticache decrease-replica-count ^
    --replication-group-id my-replication-group ^
    --new-replica-count 3 ^
    --region us-east-2 ^
    --apply-immediately
```

이 작업은 다음과 유사한 출력을 생성합니다(JSON 형식).

```
{
    "ReplicationGroup": {
        "ReplicationGroupId": "node-test",
        "Description": "node-test"
       },
        "Status": "modifying",
        "PendingModifiedValues": {},
        "MemberClusters": [
            "node-test-001",
            "node-test-002",
            "node-test-003",
            "node-test-004",
            "node-test-005",
            "node-test-006"
        ],
        "NodeGroups": [
            {
                "NodeGroupId": "0001",
                "Status": "modifying",
                "PrimaryEndpoint": {
                    "Address": "node-test.zzzzzz.ng.0001.usw2.cache.amazonaws.com",
                    "Port": 6379
                },
                "ReaderEndpoint": {
                    "Address": "node-test-
ro.zzzzzz.ng.0001.usw2.cache.amazonaws.com",
                    "Port": 6379
                },
```

```
                "NodeGroupMembers": [
                    {
                        "CacheClusterId": "node-test-001",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Address": "node-
test-001.zzzzzz.0001.usw2.cache.amazonaws.com",
                            "Port": 6379
                        },
                        "PreferredAvailabilityZone": "us-west-2a",
                        "CurrentRole": "primary"
                    },
                    {
                        "CacheClusterId": "node-test-002",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Address": "node-
test-002.zzzzzz.0001.usw2.cache.amazonaws.com",
                            "Port": 6379
                        },
                        "PreferredAvailabilityZone": "us-west-2c",
                        "CurrentRole": "replica"
                    },
                    {
                        "CacheClusterId": "node-test-003",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Address": "node-
test-003.zzzzzz.0001.usw2.cache.amazonaws.com",
                            "Port": 6379
                        },
                        "PreferredAvailabilityZone": "us-west-2b",
                        "CurrentRole": "replica"
                    },
                    {
                        "CacheClusterId": "node-test-004",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Address": "node-
test-004.zzzzzz.0001.usw2.cache.amazonaws.com",
                            "Port": 6379
                        },
                        "PreferredAvailabilityZone": "us-west-2c",
                        "CurrentRole": "replica"
```

```
                    },
                    {
                        "CacheClusterId": "node-test-005",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Address": "node-
test-005.zzzzzz.0001.usw2.cache.amazonaws.com",
                            "Port": 6379
                        },
                        "PreferredAvailabilityZone": "us-west-2b",
                        "CurrentRole": "replica"
                    },
                    {
                        "CacheClusterId": "node-test-006",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Address": "node-
test-006.zzzzzz.0001.usw2.cache.amazonaws.com",
                            "Port": 6379
                        },
                        "PreferredAvailabilityZone": "us-west-2b",
                        "CurrentRole": "replica"
                    }
                ]
            }
        ],
        "SnapshottingClusterId": "node-test-002",
        "AutomaticFailover": "enabled",
        "MultiAZ": "enabled",
        "SnapshotRetentionLimit": 1,
        "SnapshotWindow": "07:30-08:30",
        "ClusterEnabled": false,
        "CacheNodeType": "cache.r5.large",
         "DataTiering": "disabled",
        "TransitEncryptionEnabled": false,
        "AtRestEncryptionEnabled": false,
        "ARN": "arn:aws:elasticache:us-west-2:123456789012:replicationgroup:node-
test"
    }
}
```

또는 decrease-replica-count를 호출하고 --new-replica-count 파라미터를 전달하는 대신 다음과 같이 --replicas-to-remove 파라미터를 전달할 수 있습니다.

Linux, macOS, Unix의 경우:

```
aws elasticache decrease-replica-count \
    --replication-group-id my-replication-group \
    --replicas-to-remove node-test-003 \
    --region us-east-2 \
    --apply-immediately
```

Windows의 경우:

```
aws elasticache decrease-replica-count ^
    --replication-group-id my-replication-group ^
    --replicas-to-remove node-test-003 ^
    --region us-east-2 ^
    --apply-immediately
```

자세한 내용은 AWS CLI 주제를 참조하세요decrease-replica-count.

Valkey 또는 Redis와 ElastiCache API 함께 사용 OSS

를 사용하여 노드를 제거하려면 다음과 같이 복제 그룹 ID와 제거할 노드 목록을 사용하여 DecreaseReplicaCount API 작업을 ElastiCache API호출합니다.

- ReplicationGroupId 노드를 제거할 복제 그룹의 ID입니다.
- ReplicasToRemove ReplicasToRemove 파라미터는 수정이 적용된 후 이 클러스터에 포함할 노드 수를 지정합니다.
- ApplyImmediately 이 노드를 즉시 제거할지 아니면 다음 번 유지 관리 기간에 제거할지 지정합니다.
- Region 노드를 제거할 클러스터의 AWS 리전을 지정합니다.

다음 예제에서는 my-cluster 클러스터에서 노드 0004 및 0005를 즉시 제거합니다.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DecreaseReplicaCount
    &ReplicationGroupId=my-replication-group
    &ApplyImmediately=true
```

```
    &ReplicasToRemove=node-test-003
    &Region us-east-2
    &Version=2014-12-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

자세한 내용은 주제를 참조 ElastiCache API하세요DecreaseReplicaCount.

Memcached와 ElastiCache API 함께 사용

를 사용하여 노드를 제거하려면 다음과 같이 캐시 클러스터 ID와 제거할 노드 목록을 사용하여 ModifyCacheCluster API 작업을 ElastiCache API호출합니다.

- CacheClusterId 노드를 제거할 캐시 클러스터의 ID입니다.
- NumCacheNodes NumCacheNodes 파라미터는 수정이 적용된 후 이 클러스터에 포함할 노드 수를 지정합니다.
- CacheNodeIdsToRemove.member.n 클러스터에서 IDs 제거할 노드 목록입니다.
  - CacheNodeIdsToRemove.member.1=0004
  - CacheNodeIdsToRemove.member.1=0005
- ApplyImmediately 이 노드를 즉시 제거할지 아니면 다음 번 유지 관리 기간에 제거할지 지정합니다.
- Region 노드를 제거할 클러스터의 AWS 리전을 지정합니다.

다음 예제에서는 my-cluster 클러스터에서 노드 0004 및 0005를 즉시 제거합니다.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyCacheCluster
    &CacheClusterId=my-cluster
    &ApplyImmediately=true
    &CacheNodeIdsToRemove.member.1=0004
    &CacheNodeIdsToRemove.member.2=0005
    &NumCacheNodes=3
```

```
    &Region us-east-2
    &Version=2014-12-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

자세한 내용은 주제를 참조 ElastiCache API하세요ModifyCacheCluster.

에서 보류 중인 노드 추가 또는 삭제 작업 취소 ElastiCache

ElastiCache 클러스터 변경 사항을 즉시 적용하지 않기로 선택한 경우 다음 유지 관리 기간에 수행될 때까지 작업이 보류 중입니다. 대기 중인 작업을 취소할 수 있습니다.

대기 중인 작업을 취소하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 오른쪽 상단 모서리의 목록에서 보류 중인 노드 추가 또는 삭제 작업을 취소할 AWS 리전을 선택합니다.
3. 취소하려는 대기 중 작업이 있는 클러스터에서 실행 중인 엔진을 탐색 창에서 선택합니다. 선택한 엔진을 실행하는 클러스터 목록이 표시됩니다.
4. 클러스터 목록에서 보류 중인 작업을 취소하고자 하는 클러스터의 이름 왼쪽에 있는 상자가 아닌 클러스터의 이름을 선택합니다.
5. 어떤 작업이 대기 중인지 알아보려면 설명 탭을 선택하여 대기 중 생성 또는 삭제 작업이 몇 개인지 확인합니다. 대기 중 생성 작업과 대기 중 삭제 작업이 동시에 있을 수는 없습니다.
6. [Nodes] 탭을 선택합니다.
7. 대기 중인 작업을 모두 취소하려면 [Cancel Pending]을 클릭합니다. [Cancel Pending] 대화 상자가 나타납니다.
8. [Cancel Pending] 버튼을 선택하여 대기 중인 작업을 모두 취소하도록 확인하거나 [Cancel]을 선택하여 작업을 유지합니다.

에서 클러스터 삭제 ElastiCache

ElastiCache 클러스터가 사용 가능한 상태인 한 클러스터를 적극적으로 사용하고 있는지 여부에 관계없이 요금이 부과됩니다. 요금 발생을 중지하려면 클러스터를 삭제하세요.

> **Warning**
>
> ElastiCache 클러스터를 삭제하면 수동 스냅샷이 유지됩니다. 클러스터가 삭제되기 전에 최종 스냅샷을 생성할 수도 있습니다. 자동 캐시 스냅샷은 보존되지 않습니다.

사용 AWS Management Console

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.

클러스터를 삭제하려면

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. ElastiCache 엔진 대시보드에서 삭제하려는 클러스터가 실행 중인 엔진을 선택합니다.

   해당 엔진을 실행하고 있는 모든 클러스터의 목록이 표시됩니다.
3. 삭제할 클러스터를 선택하려면 클러스터 목록에서 해당 클러스터의 이름을 선택합니다.

   > **Important**
   >
   > ElastiCache 콘솔에서 한 번에 하나의 클러스터만 삭제할 수 있습니다. 여러 클러스터를 선택하면 삭제 작업이 비활성화됩니다.
4. 작업에 대해 삭제를 선택합니다.
5. 클러스터 삭제 확인 화면에서 삭제를 선택하여 클러스터를 삭제하거나 취소를 선택하여 클러스터를 유지합니다.

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

클러스터가 클러스터 목록에서 제거되는 즉시 요금 부과가 중단됩니다.

AWS CLI 를 사용하여 ElastiCache 클러스터 삭제

다음 코드는 ElastiCache 캐시 클러스터 를 삭제합니다`my-cluster`.

```
aws elasticache delete-cache-cluster --cache-cluster-id my-cluster
```

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

Linux, macOS, Unix의 경우:

```
aws elasticache delete-cache-cluster \
    --cache-cluster-id my-cluster \
    --region us-east-2
```

Windows의 경우:

```
aws elasticache delete-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --region us-east-2
```

자세한 내용은 ElastiCache 주제 의 섹션을 참조 AWS CLI 하세요`delete-cache-cluster`.

사용 ElastiCache API

다음 코드는 클러스터 `my-cluster`를 삭제합니다.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DeleteCacheCluster
    &CacheClusterId=my-cluster
    &Region us-east-2
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20150202T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20150202T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

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

자세한 내용은 참조 주제 를 ElastiCache API 참조하세요`DeleteCacheCluster`.

ElastiCache 클러스터 또는 복제 그룹에 액세스

Amazon ElastiCache 인스턴스는 Amazon EC2 인스턴스를 통해 액세스하도록 설계되었습니다.

Amazon Virtual Private Cloud(Amazon VPC)에서 ElastiCache 인스턴스를 시작한 경우 동일한 Amazon 의 Amazon 인스턴스에서 EC2 인스턴스에 액세스할 수 있습니다 ElastiCacheVPC. 또는 VPC 피어링을 사용하여 다른 Amazon 의 Amazon에서 ElastiCache 인스턴스EC2에 액세스할 수 있습니다VPC.

EC2 Classic에서 ElastiCache 인스턴스를 시작한 경우 EC2 인스턴스와 연결된 Amazon EC2 보안 그룹에 캐시 보안 그룹에 대한 액세스 권한을 부여하여 인스턴스가 클러스터에 액세스하도록 허용합니다. 기본적으로 클러스터에 대한 액세스는 클러스터를 시작한 계정으로 제한됩니다.

주제

클러스터 또는 복제 그룹에 액세스 권한 부여

클러스터를 EC2- 로 시작했습니다.VPC

클러스터를 Amazon Virtual Private Cloud(Amazon VPC)로 시작한 경우 동일한 Amazon 에서 실행 중인 Amazon EC2 인스턴스에서만 클러스터에 연결할 수 있습니다 ElastiCacheVPC. 이 경우 클러스터에 네트워크 진입을 허용해야 합니다.

> **Note**
>
> Local Zones를 사용 중인 경우 활성화했는지 확인합니다. 자세한 내용은 Local Zones 활성화를 참조하세요. 이렇게 하면 VPC가 해당 로컬 영역으로 확장되고 VPC 는 서브넷을 다른 가용 영역 및 관련 게이트웨이, 라우팅 테이블 및 기타 보안 그룹 고려 사항의 서브넷으로 취급합니다. 는 자동으로 조정됩니다.

Amazon VPC 보안 그룹에서 클러스터로 네트워크 수신을 부여하려면

1. 에 로그인 AWS Management Console 하고 에서 Amazon EC2 콘솔을 엽니다https://console.aws.amazon.com/ec2/.
2. 탐색 창의 [Network & Security] 아래에서 [Security Groups]를 선택합니다.
3. 보안 그룹 목록에서 Amazon 의 보안 그룹을 선택합니다VPC. ElastiCache 사용할 보안 그룹을 생성하지 않는 한 이 보안 그룹의 이름은 기본 입니다.

4. [Inbound] 탭을 선택하고 다음을 수행합니다.

   a. 편집을 선택합니다.

   b. 규칙 추가를 선택합니다.

   c. 유형 열에서 사용자 지정 TCP 규칙 을 선택합니다.

   d. [Port range] 상자에 클러스터 노드의 포트 번호를 입력합니다. 이 번호는 클러스터를 시작할 때 지정한 번호와 동일해야 합니다. Memcached의 기본 포트는 **11211** Valkey의 기본 포트이고 RedisOSS는 **6379**입니다.

   e. 소스 상자에서 포트 범위(0.0.0.0/0)가 있는 Anywhere를 선택하여 Amazon 내에서 시작하는 모든 Amazon EC2 인스턴스가 ElastiCache 노드에 연결할 VPC 수 있도록 합니다.

      > **Important**
      >
      > ElastiCache 클러스터를 0.0.0.0/0으로 열면 퍼블릭 IP 주소가 없으므로 외부에서 액세스할 수 없으므로 클러스터가 인터넷에 노출되지 않습니다VPC. 그러나 기본 보안 그룹은 고객 계정의 다른 Amazon EC2 인스턴스에 적용될 수 있으며 해당 인스턴스에는 퍼블릭 IP 주소가 있을 수 있습니다. 기본 포트에서 무언가를 실행하면 비의도적으로 해당 서비스가 노출될 수 있습니다. 따라서 에서만 사용할 VPC 보안 그룹을 생성하는 것이 좋습니다 ElastiCache. 자세한 정보는 사용자 지정 보안 그룹을 참조하세요.

   f. 저장(Save)을 선택합니다.

Amazon EC2 인스턴스를 Amazon 로 시작하면 VPC해당 인스턴스가 ElastiCache 클러스터에 연결할 수 있습니다.

외부에서 ElastiCache 리소스에 액세스 AWS

Amazon ElastiCache 은 클라우드 기반 인 메모리 키-값 스토어를 제공하는 AWS 서비스입니다. 서비스는 내에서만 액세스하도록 설계되었습니다 AWS. 그러나 ElastiCache 클러스터가 내에서 호스팅되는 경우 Network Address Translation(NAT) 인스턴스를 사용하여 외부 액세스를 제공할 VPC수 있습니다.

요구 사항

외부 에서 ElastiCache 리소스에 액세스하려면 다음 요구 사항을 충족해야 합니다. AWS

- 클러스터는 내에 있어야 VPC 하며 Network Address Translation(NAT) 인스턴스를 통해 액세스해야 합니다. 이 요구 사항에는 예외가 없습니다.
- NAT 인스턴스는 클러스터VPC와 동일한 에서 시작해야 합니다.
- NAT 인스턴스는 클러스터와 별도의 퍼블릭 서브넷에서 시작해야 합니다.
- 탄력적 IP 주소(EIP)를 NAT 인스턴스와 연결해야 합니다. iptables의 포트 전달 기능은 NAT 인스턴스의 포트를 내의 캐시 노드 포트로 전달하는 데 사용됩니다VPC.

고려 사항

외부에서 ElastiCache 리소스에 액세스할 때는 다음 사항을 염두에 두어야 합니다 ElastiCache.

- 클라이언트는 NAT 인스턴스의 EIP 및 캐시 포트에 연결합니다. NAT 인스턴스의 포트 전달은 트래픽을 적절한 캐시 클러스터 노드로 전달합니다.
- 클러스터 노드가 추가되거나 대체되면 이 변경 사항을 반영하도록 iptables 규칙을 업데이트해야 합니다.

제한 사항

테스트 및 개발 목적으로만 이 접근 방식을 사용해야 합니다. 다음과 같은 제한으로 인해 프로덕션용으로는 사용하지 않는 것이 좋습니다.

- NAT 인스턴스는 클라이언트와 여러 클러스터 간의 프록시 역할을 합니다. 프록시를 추가하면 캐시 클러스터 성능에 영향을 줍니다. 인스턴스를 통해 액세스하는 캐시 클러스터 수에 따라 영향이 증가합니다NAT.
- 클라이언트에서 NAT 인스턴스로의 트래픽은 암호화되지 않습니다. 따라서 NAT 인스턴스를 통해 민감한 데이터를 전송하지 않아야 합니다.

- NAT 인스턴스는 다른 인스턴스를 유지 관리하는 오버헤드를 추가합니다.
- NAT 인스턴스는 단일 장애 지점 역할을 합니다. NAT 에서 고가용성을 설정하는 방법에 대한 자세한 내용은 Amazon 인스턴스의 고가용성: 예제 를 VPC참조하세요. VPC NAT

외부에서 ElastiCache 리소스에 액세스하는 방법 AWS

다음 절차에서는 NAT 인스턴스를 사용하여 ElastiCache 리소스에 연결하는 방법을 보여줍니다.

이 단계에서는 다음과 같이 가정합니다.

- `iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 6380 -j DNAT --to 10.0.1.231:6379`
- `iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 6381 -j DNAT --to 10.0.1.232:6379`

다음으로 NAT 반대 방향으로 다음과 같이 해야 합니다.

```
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source 10.0.0.55
```

기본적으로 비활성화된 IP 전달도 활성화해야 합니다.

```
sudo sed -i 's/net.ipv4.ip_forward=0/net.ipv4.ip_forward=1/g' /etc/
sysctl.conf sudo sysctl --system
```

- 다음을 사용하여 Memcached 클러스터에 액세스합니다.
  - IP 주소 – 10.0.1.230
  - 기본 Memcached 포트 – 11211
  - 보안 그룹 – *10\.0\.0\.55*
- 다음을 사용하여 Valkey 또는 Redis OSS 클러스터에 액세스합니다.
  - IP 주소 – 10.0.1.230
  - 기본 포트 - 6379
  - 보안 그룹 – sg-bd56b7da
  - AWS 인스턴스 IP 주소 – sg-bd56b7da
- 신뢰할 수 있는 클라이언트의 IP 주소가 198.51.100.27입니다.
- NAT 인스턴스에는 탄력적 IP 주소 203.0.113.73이 있습니다.

- NAT 인스턴스에 보안 그룹 sg-ce56b7a9가 있습니다.

NAT 인스턴스를 사용하여 ElastiCache 리소스에 연결하려면

1. 캐시 클러스터VPC와 동일한 에서 퍼블릭 서브넷에 NAT 인스턴스를 생성합니다.

   기본적으로 VPC 마법사는 cache.m1.small 노드 유형을 시작합니다. 필요에 따라 노드 크기를 선택해야 합니다. 외부 ElastiCache 에서 에 액세스할 수 있으EC2NATAMI려면 를 사용해야 합니다 AWS.

   NAT 인스턴스 생성에 대한 자세한 내용은 사용 설명서의 NAT 인스턴스를 AWS VPC 참조하세요.
2. 캐시 클러스터 및 NAT 인스턴스에 대한 보안 그룹 규칙을 생성합니다.

   NAT 인스턴스 보안 그룹과 클러스터 인스턴스에는 다음 규칙이 있어야 합니다.

   - 인바운드 규칙 2개
     - Memcached의 경우 첫 번째 규칙은 신뢰할 수 있는 클라이언트에서 NAT 인스턴스(11211 - 11213)에서 전달된 각 캐시 포트로의 TCP 연결을 허용하는 것입니다.
     - Valkey 및 Redis OSS의 경우 첫 번째 규칙은 신뢰할 수 있는 클라이언트에서 NAT 인스턴스(6379 - 6381)에서 전달된 각 캐시 포트로의 TCP 연결을 허용하는 것입니다.
     - 신뢰할 수 있는 클라이언트에 대한 SSH 액세스를 허용하는 두 번째 규칙입니다.

   NAT 인스턴스 보안 그룹 - Memcached를 사용한 인바운드 규칙

유형	프로토콜	포트 범위	소스
사용자 지정 TCP 규칙	TCP	11211-11213	198.51.100.27-32
SSH	TCP	22	198.51.100.27-32

   NAT 인스턴스 보안 그룹 - Valkey 또는 Redis를 사용하는 인바운드 규칙 OSS

유형	프로토콜	포트 범위	소스
사용자 지정 TCP 규칙	TCP	6379-6380	198.51.100.27-32

유형	프로토콜	포트 범위	소스
SSH	TCP	22	203.0.113.73/32

- Memcached를 사용하면 캐시 포트(11211)에 대한 TCP 연결을 허용하는 아웃바운드 규칙입니다.

  NAT 인스턴스 보안 그룹 - 아웃바운드 규칙

유형	프로토콜	포트 범위	대상
사용자 지정 TCP 규칙	TCP	11211	sg-ce56b7a9(클러스터 인스턴스 보안 그룹)

- Valkey 또는 Redis OSS를 사용하면 캐시 포트(6379)에 대한 TCP 연결을 허용하는 아웃바운드 규칙입니다.

  NAT 인스턴스 보안 그룹 - 아웃바운드 규칙

유형	프로토콜	포트 범위	대상
사용자 지정 TCP 규칙	TCP	6379	sg-ce56b7a9(클러스터 인스턴스 보안 그룹)

- Memcached를 사용하면 NAT 인스턴스에서 캐시 포트(11211)로의 TCP 연결을 허용하는 클러스터의 보안 그룹에 대한 인바운드 규칙입니다.

  클러스터 인스턴스 보안 그룹 - 인바운드 규칙

유형	프로토콜	포트 범위	소스
사용자 지정 TCP 규칙	TCP	11211	sg-bd56b7da(NAT보안 그룹)

- Valkey 또는 Redis OSS를 사용하면 NAT 인스턴스에서 캐시 포트(6379)로의 TCP 연결을 허용하는 클러스터의 보안 그룹에 대한 인바운드 규칙입니다.

클러스터 인스턴스 보안 그룹 - 인바운드 규칙

유형	프로토콜	포트 범위	소스
사용자 지정 TCP 규칙	TCP	6379	sg-bd56b7da(클러스터 보안 그룹)

3. 규칙을 검증합니다.

   - 신뢰할 수 있는 클라이언트가 NAT 인스턴스에 SSH 액세스할 수 있는지 확인합니다.
   - 신뢰할 수 있는 클라이언트가 NAT 인스턴스에서 클러스터에 연결할 수 있는지 확인합니다.

4. Memcached

   NAT 인스턴스에 iptables 규칙을 추가합니다.

   NAT 인스턴스에서 클러스터 노드로 캐시 포트를 전달하려면 클러스터의 각 노드에 대한 NAT 테이블에 iptables 규칙을 추가해야 합니다. 예제는 다음과 같습니다.

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 11211 -j DNAT --to
 10.0.1.230:11211
```

   포트 번호는 클러스터의 노드마다 고유해야 합니다. 예를 들어, 포트 11211 - 11213을 사용하여 노드가 3개인 Memcached 클러스터로 작업하는 경우 규칙은 다음과 같습니다.

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 11211 -j DNAT --to
 10.0.1.230:11211
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 11212 -j DNAT --to
 10.0.1.231:11211
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 11213 -j DNAT --to
 10.0.1.232:11211
```

   신뢰할 수 있는 클라이언트가 클러스터에 연결할 수 있음을 확인합니다.

   신뢰할 수 있는 클라이언트는 NAT 인스턴스와 EIP 연결된 와 해당 클러스터 노드에 해당하는 클러스터 포트에 연결해야 합니다. 예를 들어 의 연결 문자열은 다음과 같을 PHP 수 있습니다.

```
$memcached->connect( '203.0.113.73', 11211 );
$memcached->connect( '203.0.113.73', 11212 );
$memcached->connect( '203.0.113.73', 11213 );
```

telnet 클라이언트를 사용하여 연결을 확인할 수도 있습니다. 예:

```
telnet 203.0.113.73 11211
telnet 203.0.113.73 11212
telnet 203.0.113.73 11213
```

Valkey 또는 Redis OSS

NAT 인스턴스에 iptables 규칙을 추가합니다.

NAT 인스턴스에서 클러스터 노드로 캐시 포트를 전달하려면 클러스터의 각 노드에 대한 NAT 테이블에 iptables 규칙을 추가해야 합니다. 예제는 다음과 같습니다.

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 6379 -j DNAT --to
 10.0.1.230:6379
```

포트 번호는 클러스터의 노드마다 고유해야 합니다. 예를 들어 포트 6379 - 6381을 사용하여 3노드 Redis OSS 클러스터로 작업하는 경우 규칙은 다음과 같습니다.

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 6379 -j DNAT --to
 10.0.1.230:6379
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 6380 -j DNAT --to
 10.0.1.231:6379
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 6381 -j DNAT --to
 10.0.1.232:6379
```

신뢰할 수 있는 클라이언트가 클러스터에 연결할 수 있음을 확인합니다.

신뢰할 수 있는 클라이언트는 NAT 인스턴스와 EIP 연결된 와 해당 클러스터 노드에 해당하는 클러스터 포트에 연결해야 합니다. 예를 들어 의 연결 문자열은 다음과 같을 PHP 수 있습니다.

```
redis->connect( '203.0.113.73', 6379 );
redis->connect( '203.0.113.73', 6380 );
redis->connect( '203.0.113.73', 6381 );
```

telnet 클라이언트를 사용하여 연결을 확인할 수도 있습니다. 예:

```
telnet 203.0.113.73 6379
telnet 203.0.113.73 6380
```

```
telnet 203.0.113.73 6381
```

5. iptables 구성을 저장합니다.

   규칙을 테스트하고 확인한 후 저장합니다. Redhat 기반 Linux 배포(예: Amazon Linux)를 사용하는 경우 다음 명령을 실행합니다.

   ```
   service iptables save
   ```

관련 주제

더 자세히 알고 싶으시면 다음 단원을 참조하십시오.

- Amazon에서 ElastiCache 캐시에 액세스하기 위한 액세스 패턴 VPC
- 고객의 데이터 센터에서 실행되는 애플리케이션에서 ElastiCache 캐시에 액세스
- NAT 인스턴스
- ElastiCache 클라이언트 구성
- Amazon VPC NAT 인스턴스의 고가용성: 예제

에서 연결 엔드포인트 찾기 ElastiCache

애플리케이션은 엔드포인트를 사용하여 ElastiCache 클러스터에 연결됩니다. 엔드포인트는 노드나 클러스터의 고유한 주소입니다.

Valkey 또는 Redis와 함께 사용할 엔드포인트 OSS

- 독립형 노드 , 읽기 및 쓰기 작업 모두에 노드의 엔드포인트를 사용합니다.

- Valkery 또는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터 에서 모든 쓰기 작업에 기본 엔드포인트를 사용합니다. 리더 엔드포인트를 사용하여 모든 읽기 전용 복제본 사이에 수신 연결을 고르게 분할합니다. 읽기 작업에 개별 노드 엔드포인트를 사용합니다(API/CLI에서는 이러한 엔드포인트를 읽기 엔드포인트라고 함).

- Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 , 클러스터 모드 활성화 명령을 지원하는 모든 작업에 대해 클러스터의 구성 엔드포인트를 사용합니다. Valkey 클러스터 또는 Redis OSS 클

러스터(Redis OSS 3.2)를 지원하는 클라이언트를 사용해야 합니다. 개별 노드 엔드포인트에서 계속 읽을 수 있습니다(API/CLI에서는 엔드포인트 읽기라고 함).

다음 섹션에서는 실행 중인 엔진에 필요한 엔드포인트를 찾는 방법을 안내합니다.

Memcached와 함께 사용할 엔드포인트

Memcached 를 사용한 ElastiCache 서버리스 캐시의 경우 콘솔에서 클러스터 엔드포인트DNS와 포트를 얻기만 하면 됩니다.

에서 `describe-serverless-caches` 명령을 AWS CLI사용하여 엔드포인트 정보를 가져옵니다.

Linux

```
aws elasticache describe-serverless-caches --serverless-cache-name CacheName
```

Windows

```
aws elasticache describe-serverless-caches --serverless-cache-name CacheName
```

위 작업의 출력은 다음과 같아야 합니다(JSON 형식).

```
{
    "ServerlessCaches": [
        {
            "ServerlessCacheName": "serverless-memcached",
            "Description": "test",
            "CreateTime": 1697659642.136,
            "Status": "available",
            "Engine": "memcached",
            "MajorEngineVersion": "1.6",
            "FullEngineVersion": "21",
            "SecurityGroupIds": [
                "sg-083eda453e1e51310"
            ],
            "Endpoint": {
                "Address": "serverless-memcached-01.amazonaws.com",
                "Port":11211
            },
```

```
            "ARN": "<the ARN>",
            "SubnetIds": [
                "subnet-0cf759df15bd4dc65",
                "subnet-09e1307e8f1560d17"
            ],
            "SnapshotRetentionLimit": 0,
            "DailySnapshotTime": "03:00"
        }
    ]
}
```

인스턴스 기반 Memcached 클러스터에서 Auto Discovery를 사용하는 경우 클러스터의 구성 엔드포인트를 사용하여 Memcached 클라이언트를 구성할 수 있습니다. Auto Discovery를 지원하는 클라이언트를 사용해야 합니다.

Auto Discovery를 사용하지 않으면 읽기 및 쓰기를 위해 개별 노드 엔드포인트를 사용하도록 클라이언트를 구성해야 합니다. 또한 노드를 추가 및 제거할 때 엔드포인트를 추적해야 합니다.

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터의 엔드포인트 찾기(콘솔)

Valkery 또는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터에 노드가 하나만 있는 경우 노드의 엔드포인트는 읽기 및 쓰기 모두에 사용됩니다. Valkery 또는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터에 여러 노드가 있는 경우 세 가지 유형의 엔드포인트가 있습니다. 즉, 기본 엔드포인트 , 리더 엔드포인트 및 노드 엔드포인트 입니다.

기본 엔드포인트는 항상 클러스터의 기본 노드로 확인되는 DNS 이름입니다. 기본 엔드포인트는 읽기 전용 복제본을 기본 역할로 승격하는 것과 같은 클러스터 변경의 영향을 받지 않습니다. 쓰기 활동의 경우 애플리케이션을 기본 엔드포인트에 연결하는 것이 좋습니다.

리더 엔드포인트는 ElastiCache (Redis OSS) 클러스터의 모든 읽기 전용 복제본 간에 엔드포인트에 대한 수신 연결을 균등하게 분할합니다. 애플리케이션이 연결을 생성하는 시기 또는 애플리케이션에서 연결을 다시 사용하는 방법과 같은 추가 요소가 트래픽 분산을 결정합니다. 리더 엔드포인트는 복제본이 추가 또는 제거되는 클러스터의 변경 사항을 실시간으로 반영합니다. ElastiCache (Redis OSS) 클러스터의 여러 읽기 전용 복제본을 서로 다른 AWS 가용 영역(AZ)에 배치하여 리더 엔드포인트의 고가용성을 보장할 수 있습니다.

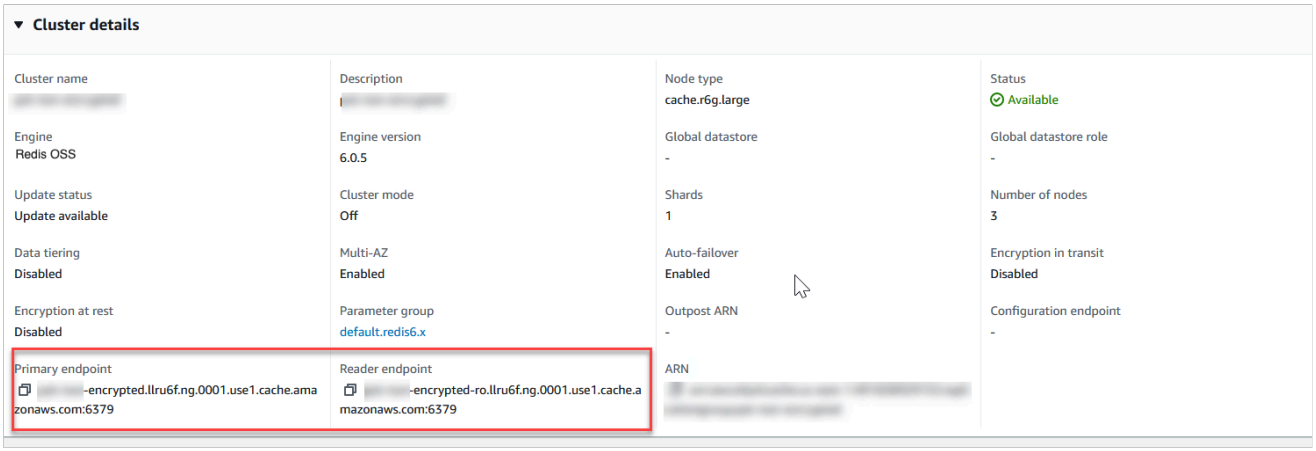
> **Note**
>
> 리더 엔드포인트는 로드 밸런서가 아닙니다. 라운드 로빈 방식으로 복제본 노드 중 하나의 IP 주소로 확인되는 DNS 레코드입니다.

읽기 활동의 경우 애플리케이션은 클러스터의 어떤 노드에도 연결할 수 있습니다. 기본 엔드포인트와 달리, 노드 엔드포인트는 특정 엔드포인트로 확인됩니다. 복제본을 추가하거나 삭제하는 것과 같이 클러스터를 변경하면 애플리케이션에서 노드 엔드포인트를 업데이트해야 합니다.

Valkery 또는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터의 엔드포인트를 찾으려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 Valkey 클러스터 또는 Redis OSS 클러스터를 선택합니다.

   클러스터 화면에 Valkery 또는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 및 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 목록이 표시됩니다.
3. 클러스터의 기본 엔드포인트 및/또는 리더 엔드포인트를 찾으려면 클러스터 이름(왼쪽에 있는 버튼 아님)을 선택합니다.

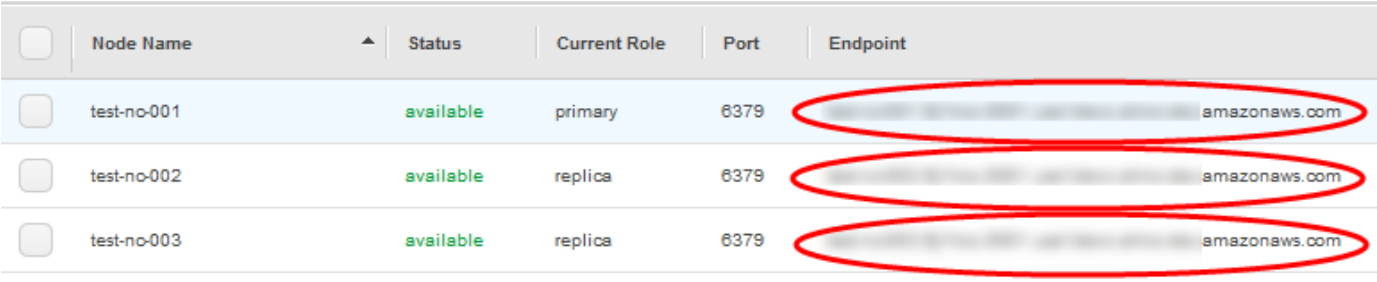


Valkery 또는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터의 프라이머리 및 리더 엔드포인트

클러스터에 노드가 하나 뿐이면 기본 엔드포인트가 없으며 다음 단계에서 계속할 수 있습니다.

4. Valkery 또는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터에 복제본 노드가 있는 경우 클러스터의 이름을 선택한 다음 노드 탭을 선택하여 클러스터의 복제본 노드 엔드포인트를 찾을 수 있습니다.

   노드 화면에 클러스터의 각 노드, 기본 및 복제본이 엔드포인트와 함께 나열됩니다.



   Valkery 또는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터의 노드 엔드포인트

5. 엔드포인트를 클립보드에 복사하려면

   a. 한 번에 엔드포인트 하나씩, 복사할 엔드포인트를 찾습니다.

   b. 엔드포인트 바로 앞에 있는 복사 아이콘을 선택합니다.

엔드포인트가 클립보드에 복사됩니다. 엔드포인트를 사용하여 노드에 연결하는 방법에 대한 자세한 내용은 Memcached 노드에 연결 섹션을 참조하세요.

Valkery 또는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 기본 엔드포인트는 다음과 같습니다. 전송 중 데이터 암호화가 활성화되어 있는지 여부에 따라 차이가 있습니다.

전송 중 데이터 암호화가 비활성화된 경우

```
clusterName.xxxxxx.nodeId.regionAndAz.cache.amazonaws.com:port

redis-01.7abc2d.0001.usw2.cache.amazonaws.com:6379
```

전송 중 데이터 암호화가 활성화된 경우

```
master.clusterName.xxxxxx.regionAndAz.cache.amazonaws.com:port

master.ncit.ameaqx.use1.cache.amazonaws.com:6379
```

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터(콘솔)의 엔드포인트 찾기

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터에는 단일 구성 엔드포인트가 있습니다. 구성 엔드포인트에 연결되면 애플리케이션이 클러스터의 각 샤드에 대한 기본 및 읽기 엔드포인트를 찾을 수 있습니다.

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 엔드포인트를 찾으려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 Valkey 클러스터 또는 Redis OSS 클러스터를 선택합니다.

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

클러스터의 엔드포인트 찾기(콘솔)(Memcached)

모든 Memcached 엔드포인트는 읽기/쓰기 엔드포인트입니다. Memcached 클러스터에 있는 노드에 연결하기 위해 애플리케이션은 각 노드의 엔드포인트를 사용하거나 Auto Discovery와 함께 클러스터의 구성 엔드포인트를 사용할 수 있습니다. Auto Discovery를 사용하려면 Auto Discovery를 지원하는 클라이언트를 사용해야 합니다.

Auto Discovery를 사용하면 클라이언트 애플리케이션이 구성 엔드포인트를 사용하여 Memcached 클러스터에 연결합니다. 노드를 추가하거나 제거하여 클러스터를 조정할 때는 애플리케이션에서 클러스터의 모든 노드를 자동으로 "인식"하고 그 중에서 어디에나 연결할 수 있습니다. Auto Discovery를 사용하지 않으면 애플리케이션에서 이 작업을 수행하거나, 사용자가 노드를 추가하거나 제거할 때마다 애플리케이션의 엔드포인트를 수동으로 업데이트해야 합니다.

엔드포인트를 복사하려면 엔드포인트 주소 바로 앞에 있는 복사 아이콘을 선택합니다. 엔드포인트를 사용하여 노드에 연결하는 방법에 대한 자세한 내용은 Memcached 노드에 연결 섹션을 참조하세요.

구성 엔드포인트와 노드 엔드포인트는 매우 비슷합니다. 두 엔드포인트의 차이는 다음과 같이 굵게 표시됩니다.

```
myclustername.xxxxxx.cfg.usw2.cache.amazonaws.com:port   # configuration endpoint
 contains "cfg"
myclustername.xxxxxx.0001.usw2.cache.amazonaws.com:port  # node endpoint for node 0001
```

> **Important**
>
> Memcached 구성 엔드포인트에 CNAME 대한 를 생성하도록 선택한 경우 자동 검색 클라이언트가 를 구성 엔드포인트CNAME로 인식하려면 `.cfg.`에 를 포함해야 합니다CNAME.

엔드포인트 찾기(AWS CLI)

Memcached의 경우 AWS CLI for Amazon ElastiCache 을 사용하여 노드 및 클러스터의 엔드포인트를 검색할 수 있습니다.

Redis의 경우 AWS CLI for Amazon을 사용하여 노드, 클러스터 및 복제 그룹의 엔드포인트를 ElastiCache 검색할 수 있습니다.

주제

- 노드 및 클러스터의 엔드포인트 찾기(AWS CLI)
- Valkey 또는 Redis OSS 복제 그룹의 엔드포인트 찾기(AWS CLI)

노드 및 클러스터의 엔드포인트 찾기(AWS CLI)

AWS CLI 를 사용하여 `describe-cache-clusters` 명령을 사용하여 클러스터 및 해당 노드의 엔드포인트를 검색할 수 있습니다. Valkey 또는 Redis OSS 클러스터의 경우 명령은 클러스터 엔드포인트를 반환합니다. Memcached 클러스터의 경우 명령이 구성 엔드포인트를 반환합니다. 또한 선택적 파라미터 `--show-cache-node-info`를 포함할 경우 명령이 클러스터에 있는 개별 노드의 엔드포인트를 반환합니다.

Example

다음 명령은 Memcached 클러스터 mycluster의 구성 엔드포인트(`ConfigurationEndpoint`) 및 개별 노드 엔드포인트(`Endpoint`)를 검색합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache describe-cache-clusters \
    --cache-cluster-id mycluster \
    --show-cache-node-info
```

Windows의 경우:

```
aws elasticache describe-cache-clusters ^
    --cache-cluster-id mycluster ^
    --show-cache-node-info
```

위 작업의 출력은 다음과 같아야 합니다(JSON 형식).

```
{
```

```
    "CacheClusters": [
    {
        "Engine": "memcached",
        "CacheNodes": [
           {
              "CacheNodeId": "0001",
              "Endpoint": {
                 "Port": 11211,
                 "Address": "mycluster.amazonaws.com"
              },
                 "CacheNodeStatus": "available",
                 "ParameterGroupStatus": "in-sync",
                 "CacheNodeCreateTime": "2016-09-22T21:30:29.967Z",
                 "CustomerAvailabilityZone": "us-west-2b"
           },
           {
              "CacheNodeId": "0002",
              "Endpoint": {
                 "Port": 11211,
                 "Address": "mycluster.amazonaws.com"
              },
                 "CacheNodeStatus": "available",
                 "ParameterGroupStatus": "in-sync",
                 "CacheNodeCreateTime": "2016-09-22T21:30:29.967Z",
                 "CustomerAvailabilityZone": "us-west-2b"
           },
           {
                 "CacheNodeId": "0003",
                 "Endpoint": {
                    "Port": 11211,
                    "Address": "mycluster.amazonaws.com"
                 },
                    "CacheNodeStatus": "available",
                    "ParameterGroupStatus": "in-sync",
                    "CacheNodeCreateTime": "2016-09-22T21:30:29.967Z",
                    "CustomerAvailabilityZone": "us-west-2b"
           }
        ],
        "CacheParameterGroup": {
        "CacheNodeIdsToReboot": [],
        "CacheParameterGroupName": "default.memcached1.4",
        "ParameterApplyStatus": "in-sync"
             },
             "CacheClusterId": "mycluster",
```

```
            "PreferredAvailabilityZone": "us-west-2b",
            "ConfigurationEndpoint": {
                "Port": 11211,
                "Address": "mycluster.amazonaws.com"
            },
            "CacheSecurityGroups": [],
            "CacheClusterCreateTime": "2016-09-22T21:30:29.967Z",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterStatus": "available",
            "NumCacheNodes": 3,
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "CacheSubnetGroupName": "default",
            "EngineVersion": "1.4.24",
            "PendingModifiedValues": {},
            "PreferredMaintenanceWindow": "mon:09:00-mon:10:00",
            "CacheNodeType": "cache.m4.large",
             "DataTiering": "disabled"
        }
    ]
}
```

Important

Memcached 구성 엔드포인트에 CNAME 대한 를 생성하도록 선택한 경우 자동 검색 클라이언트가 를 구성 엔드포인트CNAME로 인식하려면 `.cfg.`에 를 포함해야 합니다CNAME. 예를 들어, `mycluster.cfg.local`가 `session.save_path` 파라미터의 php.ini 파일에 있습니다.

Example

Valkey 및 Redis OSS의 경우 다음 명령은 단일 노드 클러스터 mycluster 에 대한 클러스터 정보를 검색합니다.

Important

파라미터는 단일 노드 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터 ID 또는 복제 그룹의 특정 노드 ID와 함께 사용할 `--cache-cluster-id` 수 있습니다. `--cache-cluster-id` 복제 그룹의 는 와 같은 4자리 값입니다`0001`. `--cache-cluster-id` 가 복제 그룹에 있는 클러스터(노드)의 ID인 경우 `replication-group-id`가 출력에 포함됩니다.

Linux, macOS, Unix의 경우:

```
aws elasticache describe-cache-clusters \
    --cache-cluster-id redis-cluster \
    --show-cache-node-info
```

Windows의 경우:

```
aws elasticache describe-cache-clusters ^
    --cache-cluster-id redis-cluster ^
    --show-cache-node-info
```

위 작업의 출력은 다음과 같아야 합니다(JSON 형식).

```
{
    "CacheClusters": [
        {
            "CacheClusterStatus": "available",
            "SecurityGroups": [
                {
                    "SecurityGroupId": "sg-77186e0d",
                    "Status": "active"
                }
            ],
            "CacheNodes": [
                {
                    "CustomerAvailabilityZone": "us-east-1b",
                    "CacheNodeCreateTime": "2018-04-25T18:19:28.241Z",
                    "CacheNodeStatus": "available",
                    "CacheNodeId": "0001",
                    "Endpoint": {
                        "Address": "redis-cluster.amazonaws.com",
                        "Port": 6379
                    },
                    "ParameterGroupStatus": "in-sync"
                }
            ],
            "AtRestEncryptionEnabled": false,
            "CacheClusterId": "redis-cluster",
            "TransitEncryptionEnabled": false,
            "CacheParameterGroup": {
                "ParameterApplyStatus": "in-sync",
```

```
                "CacheNodeIdsToReboot": [],
                "CacheParameterGroupName": "default.redis3.2"
            },
            "NumCacheNodes": 1,
            "PreferredAvailabilityZone": "us-east-1b",
            "AutoMinorVersionUpgrade": true,
            "Engine": "redis",
            "AuthTokenEnabled": false,
            "PendingModifiedValues": {},
            "PreferredMaintenanceWindow": "tue:08:30-tue:09:30",
            "CacheSecurityGroups": [],
            "CacheSubnetGroupName": "default",
            "CacheNodeType": "cache.t2.small",
             "DataTiering": "disabled"
            "EngineVersion": "3.2.10",
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "CacheClusterCreateTime": "2018-04-25T18:19:28.241Z"
        }
    ]
}
```

자세한 내용은 주제를 참조하세요describe-cache-clusters.

Valkey 또는 Redis OSS 복제 그룹의 엔드포인트 찾기(AWS CLI)

AWS CLI 를 사용하여 `describe-replication-groups` 명령을 사용하여 복제 그룹 및 해당 클러스터의 엔드포인트를 검색할 수 있습니다. 이 명령은 리더 엔드포인트와 함께, 복제 그룹의 기본 엔드포인트와 복제 그룹에 있는 모든 클러스터(노드)의 목록 및 해당 엔드포인트를 반환합니다.

다음 작업은 복제 그룹 `myreplgroup`의 기본 엔드포인트 및 리더 엔드포인트를 검색합니다. 모든 쓰기 작업에 기본 엔드포인트를 사용합니다.

```
aws elasticache describe-replication-groups \
    --replication-group-id myreplgroup
```

Windows의 경우:

```
aws elasticache describe-replication-groups ^
    --replication-group-id myreplgroup
```

이 작업의 출력은 다음과 같아야 합니다(JSON 형식).

```
{
   "ReplicationGroups": [
     {
       "Status": "available",
       "Description": "test",
       "NodeGroups": [
         {
            "Status": "available",
               "NodeGroupMembers": [
                  {
                     "CurrentRole": "primary",
                     "PreferredAvailabilityZone": "us-west-2a",
                     "CacheNodeId": "0001",
                     "ReadEndpoint": {
                        "Port": 6379,
                        "Address": "myreplgroup-001.amazonaws.com"
                     },
                     "CacheClusterId": "myreplgroup-001"
                  },
                  {
                     "CurrentRole": "replica",
                     "PreferredAvailabilityZone": "us-west-2b",
                     "CacheNodeId": "0001",
                     "ReadEndpoint": {
                        "Port": 6379,
                        "Address": "myreplgroup-002.amazonaws.com"
                     },
                     "CacheClusterId": "myreplgroup-002"
                  },
                  {
                     "CurrentRole": "replica",
                     "PreferredAvailabilityZone": "us-west-2c",
                     "CacheNodeId": "0001",
                     "ReadEndpoint": {
                        "Port": 6379,
                        "Address": "myreplgroup-003.amazonaws.com"
                     },
                     "CacheClusterId": "myreplgroup-003"
                  }
               ],
               "NodeGroupId": "0001",
               "PrimaryEndpoint": {
                  "Port": 6379,
```

```
                "Address": "myreplgroup.amazonaws.com"
             },
             "ReaderEndpoint": {
                "Port": 6379,
                "Address": "myreplgroup-ro.amazonaws.com"
             }
          }
       ],
       "ReplicationGroupId": "myreplgroup",
       "AutomaticFailover": "enabled",
       "SnapshottingClusterId": "myreplgroup-002",
       "MemberClusters": [
          "myreplgroup-001",
          "myreplgroup-002",
          "myreplgroup-003"
       ],
       "PendingModifiedValues": {}
    }
  ]
}
```

자세한 내용은 명령 참조describe-replication-groups의 섹션을 참조하세요. AWS CLI

엔드포인트 찾기(ElastiCache API)

Memcached의 경우 Amazon ElastiCache API을 사용하여 노드 및 클러스터의 엔드포인트를 검색할 수 있습니다.

Redis의 경우 Amazon ElastiCache API을 사용하여 노드, 클러스터 및 복제 그룹의 엔드포인트를 검색할 수 있습니다.

주제

- 노드 및 클러스터의 엔드포인트 찾기(ElastiCache API)
- Valkey 또는 Redis OSS 복제 그룹의 엔드포인트 찾기(ElastiCache API)

노드 및 클러스터의 엔드포인트 찾기(ElastiCache API)

를 ElastiCache API 사용하여 `DescribeCacheClusters` 작업과 함께 클러스터 및 해당 노드의 엔드포인트를 검색할 수 있습니다. Valkey 또는 Redis OSS 클러스터의 경우 명령은 클러스터 엔드포인트를 반환합니다. Memcached 클러스터의 경우 명령이 구성 엔드포인트를 반환합니다. 또한 선택적 파라미터 `ShowCacheNodeInfo`를 포함할 경우 작업이 클러스터에 있는 개별 노드의 엔드포인트를 반환합니다.

Example

Memcached의 경우 다음 명령은 Memcached 클러스터 mycluster 에 대한 구성 엔드포인트(`ConfigurationEndpoint`) 및 개별 노드 엔드포인트(`Endpoint`)를 검색합니다.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeCacheClusters
    &CacheClusterId=mycluster
    &ShowCacheNodeInfo=true
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &Version=2015-02-02
    &X-Amz-Credential=<credential>
```

> **Important**
>
> Memcached 구성 엔드포인트에 CNAME 대한 를 생성하도록 선택한 경우 자동 검색 클라이언트가 를 구성 엔드포인트CNAME로 인식하려면 .cfg.에 를 포함해야 합니다CNAME. 예를 들어, mycluster.*cfg.*local가 session.save_path 파라미터의 php.ini 파일에 있습니다.

Valkey 또는 Redis OSS 복제 그룹의 엔드포인트 찾기(ElastiCache API)

를 ElastiCache API 사용하여 DescribeReplicationGroups 작업과 함께 복제 그룹 및 해당 클러스터의 엔드포인트를 검색할 수 있습니다. 이 작업은 리더 엔드포인트와 함께, 복제 그룹의 기본 엔드포인트와 복제 그룹에 있는 모든 클러스터의 목록 및 해당 엔드포인트를 반환합니다.

다음 작업은 복제 그룹 에 대한 기본 엔드포인트(PrimaryEndpoint), 리더 엔드포인트(ReaderEndpoint) 및 개별 노드 엔드포인트(ReadEndpoint)를 검색합니다myreplgroup. 모든 쓰기 작업에 기본 엔드포인트를 사용합니다.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeReplicationGroups
    &ReplicationGroupId=myreplgroup
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &Version=2015-02-02
    &X-Amz-Credential=<credential>
```

자세한 내용은 DescribeReplicationGroups를 참조하세요.

에서 샤드 작업 ElastiCache

샤드(API/CLI: 노드 그룹)는 Valkey 또는 Redis OSS 노드가 ElastiCache 있는 1~6개의 모음입니다. Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터에는 샤드가 두 개 이상 없습니다. 샤드를 사용하면 대규모 데이터베이스를 데이터 샤드라고 하는 더 작고 빠르며 쉽게 관리되는 부분으로 분리할 수 있습니다. 이렇게 하면 여러 별도의 섹션에 작업을 분산하여 데이터베이스 효율성을 높일 수 있습니다. 샤드를 사용하면 향상된 성능, 확장성 및 비용 효율성을 비롯한 많은 이점을 얻을 수 있습니다.

하나의 클러스터당 최대 90개의 노드로 구성된 더 많은 수의 샤드와 더 적은 수의 복제본을 가진 클러스터를 생성할 수 있습니다. 이 클러스터 구성은 90개의 샤드 및 0개의 복제본부터 15개의 샤드 및 5개의 복제본까지 해당될 수 있으며, 이는 허용되는 최대 복제본 수입니다. 클러스터의 데이터는 클러스터

의 샤드로 분할됩니다. 샤드에 둘 이상의 노드가 있는 경우 샤드는 한 노드가 읽기/쓰기 기본 노드가 되고 다른 노드가 읽기 전용 복제본 노드인 복제를 구현합니다.

엔진 버전 Valkey 7.2 또는 Redis 5.0.6 이상인 경우 노드 또는 샤드 제한을 클러스터당 최대 OSS 500개로 늘릴 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. 일반적인 위험에는 서브넷 그룹의 서브넷 CIDR 범위가 너무 작거나 서브넷이 공유되어 다른 클러스터에서 많이 사용되는 경우가 포함됩니다. 자세한 내용은 서브넷 그룹 생성 단원을 참조하십시오.

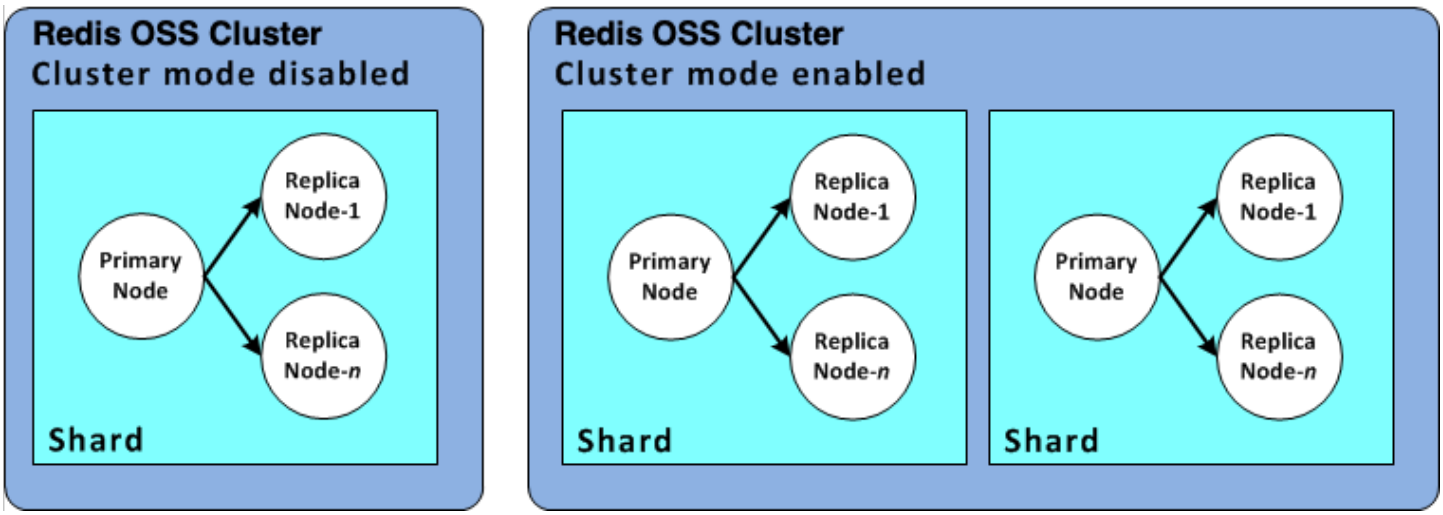
5.0.6 이하의 버전에서 한도는 클러스터당 250개입니다.

한도 증가를 요청하려면 AWS 서비스 한도를 참조하고 한도 유형을 인스턴스 유형별 클러스터당 노드로 선택하세요.

ElastiCache 콘솔을 사용하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터를 생성할 때 클러스터의 샤드 수와 샤드의 노드 수를 지정합니다. 자세한 내용은 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 생성(콘솔) 단원을 참조하십시오. 또는 를 ElastiCache API 사용하여 클러스터(API/에서 복제 그룹이라고 함CLI)를 AWS CLI 생성하는 경우 샤드(API/CLI: 노드 그룹)의 노드 수를 독립적으로 구성할 수 있습니다. 자세한 내용은 다음 자료를 참조하세요.

- API: CreateReplicationGroup
- CLI: create-replication-group

샤드의 각 노드는 컴퓨팅, 스토리지 및 메모리 사양이 동일합니다. 를 ElastiCache API 사용하면 노드 수, 보안 설정 및 시스템 유지 관리 기간과 같은 샤드 전체 속성을 제어할 수 있습니다.



Valkey 또는 Redis OSS 샤드 구성

자세한 내용은 Valkey 또는 Redis에 대한 오프라인 리샤딩OSS(클러스터 모드 활성화됨) 및 Valkey 또는 Redis에 대한 온라인 리샤딩OSS(클러스터 모드 활성화됨) 단원을 참조하세요.

샤드 ID 찾기

AWS Management Console, AWS CLI 또는 를 사용하여 샤드의 ID를 찾을 수 있습니다 ElastiCache API.

사용 AWS Management Console

주제

- Valkey 또는 Redis의 경우OSS(클러스터 모드 비활성화됨)
- Valkey 또는 Redis의 경우OSS(클러스터 모드 활성화됨)

Valkey 또는 Redis의 경우OSS(클러스터 모드 비활성화됨)

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹 샤드IDs는 항상 입니다`0001`.

Valkey 또는 Redis의 경우OSS(클러스터 모드 활성화됨)

다음 절차에서는 AWS Management Console 를 사용하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨)의 복제 그룹의 샤드 ID를 찾습니다.

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹에서 샤드 ID를 찾으려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 Valkey 또는 Redis OSS를 선택한 다음 샤드를 찾을 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹의 이름을 선택합니다IDs.
3. 샤드 이름 열에서 샤드 ID는 샤드 이름의 마지막 네 자리 숫자입니다.

사용 AWS CLI

Valkey 또는 Redis(클러스터 모드 비활성화됨) 또는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹에 대한 샤드OSS(노드 그룹) ID를 찾으려면 다음 선택적 파라미터를 `describe-replication-groups` 사용하여 AWS CLI 작업을 사용합니다.

- **`--replication-group-id`** - 사용되면 지정된 복제 그룹의 세부 정보 출력을 제한하는 선택적 파라미터입니다. 이 파라미터가 생략되면 최대 100개의 복제 그룹의 세부 정보가 반환됩니다.

Example

이 명령은 `sample-repl-group`의 세부 정보를 반환합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache describe-replication-groups \
    --replication-group-id sample-repl-group
```

Windows의 경우:

```
aws elasticache describe-replication-groups ^
    --replication-group-id sample-repl-group
```

이 명령의 출력은 다음과 같습니다. 샤드(노드 그룹) ID는 다음과 같습니다.*highlighted* 더 쉽게 찾을 수 있습니다.

```
{
    "ReplicationGroups": [
        {
            "Status": "available",
            "Description": "2 shards, 2 nodes (1 + 1 replica)",
            "NodeGroups": [
                {
                    "Status": "available",
                    "Slots": "0-8191",
                    "NodeGroupId": "0001",
                    "NodeGroupMembers": [
                        {
                            "PreferredAvailabilityZone": "us-west-2c",
                            "CacheNodeId": "0001",
                            "CacheClusterId": "sample-repl-group-0001-001"
                        },
                        {
                            "PreferredAvailabilityZone": "us-west-2a",
                            "CacheNodeId": "0001",
                            "CacheClusterId": "sample-repl-group-0001-002"
                        }
                    ]
                },
                {
                    "Status": "available",
                    "Slots": "8192-16383",
```

```
                    "NodeGroupId": "0002",
                    "NodeGroupMembers": [
                        {
                            "PreferredAvailabilityZone": "us-west-2b",
                            "CacheNodeId": "0001",
                            "CacheClusterId": "sample-repl-group-0002-001"
                        },
                        {
                            "PreferredAvailabilityZone": "us-west-2a",
                            "CacheNodeId": "0001",
                            "CacheClusterId": "sample-repl-group-0002-002"
                        }
                    ]
                }
            ],
            "ConfigurationEndpoint": {
                "Port": 6379,
                "Address": "sample-repl-
group.9dcv5r.clustercfg.usw2.cache.amazonaws.com"
            },
            "ClusterEnabled": true,
            "ReplicationGroupId": "sample-repl-group",
            "SnapshotRetentionLimit": 1,
            "AutomaticFailover": "enabled",
            "SnapshotWindow": "13:00-14:00",
            "MemberClusters": [
                "sample-repl-group-0001-001",
                "sample-repl-group-0001-002",
                "sample-repl-group-0002-001",
                "sample-repl-group-0002-002"
            ],
            "CacheNodeType": "cache.m3.medium",
            "DataTiering": "disabled",
            "PendingModifiedValues": {}
        }
    ]
}
```

사용 ElastiCache API

Valkey 또는 Redis(클러스터 모드 비활성화됨) 또는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹에 대한 샤드OSS(노드 그룹) ID를 찾으려면 다음 선택적 파라미터를 `describe-replication-groups` 사용하여 AWS CLI 작업을 사용합니다.

- **ReplicationGroupId** - 사용되면 지정된 복제 그룹의 세부 정보 출력을 제한하는 선택적 파라미터입니다. 이 파라미터가 생략된 경우 최대 *xxx* 복제 그룹이 반환됩니다.

Example

이 명령은 sample-repl-group의 세부 정보를 반환합니다.

Linux, macOS, Unix의 경우:

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeReplicationGroup
   &ReplicationGroupId=sample-repl-group
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

Valkey, Redis OSS및 Memcached 자체 설계 캐시 비교

Amazon은 Valkey, Redis OSS및 Memcached 캐시 엔진을 ElastiCache 지원합니다. 각 엔진에는 몇 가지 장점이 있습니다. 이 항목의 정보를 활용하면 요구 사항에 가장 잘 맞는 엔진과 버전을 선택하는 데 도움이 됩니다.

> **Important**
>
> 캐시, 자체 설계된 클러스터 또는 복제 그룹을 생성한 후에는 최신 엔진 버전으로 업그레이드할 수 있지만 이전 엔진 버전으로 다운그레이드할 수는 없습니다. 이전 엔진 버전을 사용하려면 기존 캐시, 자체 설계된 클러스터 또는 복제 그룹을 삭제하고 이전 엔진 버전으로 다시 생성해야 합니다.

표면적으로는 엔진이 유사하게 보입니다. 각 엔진은 인 메모리 키-값 저장소입니다. 그러나 실제로 상당한 차이점이 있습니다.

다음과 같은 경우 Memcached를 선택합니다.

- 가능한 가장 단순한 모델이 필요한 경우
- 여러 코어 또는 스레드가 있는 큰 노드를 실행해야 하는 경우

- 시스템의 요구 사항이 증가하고 감소함에 따라 노드를 추가 및 제거하는 확장 및 축소 기능이 필요한 경우
- 객체를 캐시에 저장해야 하는 경우

다음 사항이 적용되는 OSS ElastiCache 경우 Valkey 또는 Redis with를 선택합니다.

- ElastiCache Valkey 7.2 또는 Redis OSS 버전 7.0(향상됨) 사용

  함수 , 샤딩된 Pub/Sub 또는 ACL 개선 사항을 사용하려고 합니다. 자세한 내용은 Redis OSS 버전 7.0(향상됨)을 참조하세요.
- ElastiCache (Redis OSS) 버전 6.2(향상됨)

  r6gd 노드 유형을 SSD 사용하여 메모리와 간에 데이터를 계층화하는 기능을 원합니다. 자세한 내용은 데이터 계층화를 참조하세요.
- ElastiCache (Redis OSS) 버전 6.0(향상됨)

  역할 기반 액세스 제어로 사용자를 인증하려는 경우

  자세한 내용은 Redis OSS 버전 6.0(향상됨)을 참조하세요.
- ElastiCache (Redis OSS) 버전 5.0.0(향상됨)

  생산자가 실시간으로 새 항목을 추가할 수 있고 소비자가 차단 또는 비차단 방식으로 메시지를 사용할 수 있는 로그 데이터 구조인 Redis OSS 스트림을 사용하려고 합니다.

  자세한 내용은 Redis OSS 버전 5.0.0(향상됨)을 참조하세요.
- ElastiCache (Redis OSS) 버전 4.0.10(향상됨)

  Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터에서 암호화와 동적으로 샤드 추가 또는 제거를 모두 지원합니다.

  자세한 내용은 Redis OSS 버전 4.0.10(향상됨)을 참조하세요.

다음 버전은 더 이상 사용되지 않거나 수명이 다했거나 곧 종료될 예정입니다.

- ElastiCache (Redis OSS) 버전 3.2.10(향상됨)

  Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터에서 샤드를 동적으로 추가하거나 제거하는 기능을 지원합니다.

> **Important**
>
> 현재 ElastiCache (Redis OSS) 3.2.10는 암호화를 지원하지 않습니다.

자세한 내용은 다음 자료를 참조하세요.

- Redis OSS 버전 3.2.10(향상됨)
- Redis 에 대한 온라인 리샤딩 모범 사례OSS, 자세한 내용은 다음을 참조하세요.
  - 모범 사례: 온라인 리샤딩
  - Valkey 또는 Redis에 대한 온라인 공유 및 샤드 리밸런싱OSS(클러스터 모드 활성화됨)
- Redis OSS 클러스터 크기 조정에 대한 자세한 내용은 크기 조정을 참조하세요.

- ElastiCache (Redis OSS) 버전 3.2.6(향상됨)

  이전 Redis OSS 버전의 기능과 다음 기능이 필요한 경우 ElastiCache (Redis OSS) 3.2.6을 선택합니다.

  - 전송 중 데이터 암호화. 자세한 내용은 Amazon ElastiCache (Redis OSS) 전송 중 암호화를 참조하세요.
  - 미사용 데이터 암호화. 자세한 내용은 Amazon ElastiCache (Redis OSS) At-Rest Encryption을 참조하세요.
- ElastiCache (Redis OSS) (클러스터 모드 활성화됨) 버전 3.2.4

  Redis OSS 2.8.x의 기능과 다음 기능이 필요한 경우 Redis OSS 3.2.4(클러스터 모드)를 선택합니다.

  - 2~500개의 노드 그룹으로 데이터를 분할해야 하는 경우(클러스터 모드에만 해당)
  - 지역 검색 인덱싱이 필요한 경우(클러스터 모드 또는 비클러스터 모드)
  - 여러 데이터베이스를 지원할 필요가 없는 경우
- ElastiCache (Redis OSS) (클러스터링되지 않은 모드) 2.8.x 및 3.2.4(향상됨)

  다음 사항이 적용되는 경우 Redis OSS 2.8.x 또는 Redis OSS 3.2.4(클러스터링되지 않은 모드)를 선택합니다.

  - 문자열, 해시, 목록, 세트, 정렬된 세트 및 비트맵과 같은 복잡한 데이터 유형이 필요한 경우
  - 인 메모리 데이터 세트를 정렬하거나 순위를 지정해야 하는 경우
  - 키 저장소의 지속성을 원할 경우

- 읽기 집약적 애플리케이션을 위해 기본 항목에서 하나 이상의 읽기 전용 복제본으로 데이터를 복제해야 하는 경우
- 기본 노드가 실패할 때 자동 장애 조치가 필요한 경우
- 서버에 대한 이벤트를 클라이언트에 알리기 위해 게시 및 구독(게시/구독) 기능이 필요합니다.
- 자체 설계된 클러스터와 서버리스 캐시를 위한 백업 및 복원 기능이 필요합니다.
- 여러 데이터베이스를 지원해야 하는 경우

Memcached, Valkey 또는 RedisOSS(클러스터 모드 비활성화됨)와 Valkey 또는 RedisOSS(클러스터 모드 활성화됨)의 비교 요약

	Memcached	Valkey 또는 RedisOSS(클러스터 모드 비활성화됨)	Valkey 또는 RedisOSS(클러스터 모드 활성화됨)
엔진 버전+	1.4.5 이상	4.0.10 이상	4.0.10 이상
데이터 타입	간단함	2.8.x - 복합 *	3.2.x 이상 - 복합
		복합	
데이터 파티셔닝	예	아니요	예
클러스터 수정 가능	예	예	3.2.10이상 - 제한
온라인 리샤딩	아니요	아니요	3.2.10 이상
암호화(Encryption)	전송 중 1.6.12 이상	4.0.10 이상	4.0.10 이상
데이터 계층화	아니요	6.2 이상	6.2 이상
규정 준수 인증			
규정 준수 인증			
급식RAMP	예 - 1.6.12 이상	4.0.10 이상	4.0.10 이상
HIPAA	예 - 1.6.12 이상	4.0.10 이상	4.0.10 이상
PCI DSS	예	4.0.10 이상	4.0.10 이상
다중 스레드	예	아니요	아니요
노드 유형 업그레이드	아니요	예	예
엔진 업그레이드	예	예	예
고가용성(복제)	아니요	예	예
자동 장애조치(fail over)	아니요	선택 사항	필수

	Memcached	Valkey 또는 RedisOSS(클러스터 모드 비활성화됨)	Valkey 또는 RedisOSS(클러스터 모드 활성화됨)
게시/구독 기능	아니요	예	예
정렬된 세트	아니요	예	예
백업 및 복원	Serverless Memcached 전용, 자체 설계된 Memcached 클러스터 제외	예	예
지역 검색 인덱싱	아니요	4.0.10 이상	예

참고:

문자열, 객체(예: 데이터베이스)

* 문자열, 세트, 정렬된 세트, 목록, 해시, 비트맵, HyperLogLog

문자열, 세트, 정렬된 세트, 목록, 해시, 비트맵, hyperloglog, 지역 검색 인덱스

+ 더 이상 사용되지 않거나 수명이 다했거나 곧 종료될 버전은 제외합니다.

클러스터에 대한 엔진을 선택한 후 해당 엔진의 최신 버전을 사용하는 것이 좋습니다. 자세한 내용은 지원되는 노드 유형 단원을 참조하십시오.

Valkey 또는 Redis의 온라인 마이그레이션 OSS

온라인 마이그레이션을 사용하면 AmazonOSS의 자체 호스팅 오픈 소스 Valkey 또는 Redis에서 Amazon 로 데이터를 마이그레이션EC2할 수 있습니다 ElastiCache.

> **Note**
>
> r6gd 노드 유형에서 실행되는 ElastiCache 서버리스 캐시 또는 클러스터에서는 온라인 마이그레이션이 지원되지 않습니다.

개요

Amazon에서 OSS 실행되는 오픈 소스 Valkey 또는 Redis에서 AmazonEC2으로 데이터를 마이그레이션하려면 기존 또는 새로 생성된 Amazon ElastiCache 배포가 ElastiCache 필요합니다. 이 배포는 즉시 마이그레이션이 가능한 구성을 가지고 있어야 합니다. 또한 인스턴스 유형, 샤드 수 및 복제본 개수 같은 속성을 포함하여 원하는 구성과 비슷한 구성을 가져야 합니다.

온라인 마이그레이션은 ElastiCache 클러스터 간에 데이터를 이동하기 위한 것이 ElastiCache아니라 AmazonOSS의 자체 호스팅 오픈 소스 Valkey 또는 Redis에서 로 데이터를 마이그레이션EC2하기 위해 설계되었습니다.

> **Important**
>
> 온라인 마이그레이션 프로세스를 시작하기 전에 다음 섹션을 전부 검토하는 것이 좋습니다

마이그레이션은 `StartMigration` API 작업 또는 AWS CLI 명령을 호출할 때 시작됩니다. Valkey 또는 Redis OSS 클러스터 모드 비활성화 클러스터를 마이그레이션할 때 마이그레이션 프로세스는 ElastiCache Valkey 또는 Redis OSS 클러스터의 기본 노드를 소스 Valkey 또는 Redis OSS 프라이머리의 복제본으로 만듭니다. Valkey 또는 Redis OSS 클러스터 모드 지원 클러스터를 마이그레이션할 때 마이그레이션 프로세스는 각 ElastiCache 샤드의 기본 노드를 동일한 슬롯을 소유한 소스 클러스터의 해당 샤드 복제본으로 만듭니다.

클라이언트 측 변경 사항이 준비되면 `CompleteMigration` API 작업을 호출합니다. 이 API 작업은 기본 및 복제본 노드(해당하는 경우) ElastiCache 를 사용하여 기본 Valkey 또는 Redis OSS 배포로 배포를 승격합니다. 이제 클라이언트 애플리케이션을 리디렉션하여 에 데이터 쓰기를 시작할 수 있습니다 ElastiCache. 마이그레이션 중에 Valkey 노드와 기본 노드에서 valkey-cli INFO 명령을 실행하여 복제 상태를 확인할 수 있습니다. ElastiCache

마이그레이션 단계

다음 주제에서는 데이터 마이그레이션을 위한 프로세스에 대해 개략적으로 알아봅니다.

- 마이그레이션을 위한 소스 및 대상 준비
- 데이터 마이그레이션 테스트
- 마이그레이션 시작
- 데이터 마이그레이션 진행 상황 확인
- 데이터 마이그레이션 완료

마이그레이션을 위한 소스 및 대상 준비

이 단계를 통해 자체 호스팅된 Valkey 또는 Redis 소스에서 로 ElastiCache또는 Redis OSS 클러스터에서 ElastiCache Valkey 클러스터로 데이터를 마이그레이션EC2할 준비를 할 수 있습니다.

ElastiCache 콘솔 API 또는 에서 마이그레이션을 시작하기 전에 다음 네 가지 사전 조건을 모두 충족해야 합니다 AWS CLI.

마이그레이션을 위해 소스 및 대상 Valkey 또는 Redis OSS 노드를 준비하려면

1. 대상 ElastiCache 배포를 식별하고 데이터를 마이그레이션할 수 있는지 확인합니다.

   기존 배포 또는 새로 생성된 ElastiCache 배포는 마이그레이션에 대한 다음 요구 사항을 충족해야 합니다.

   - Valkey 또는 Redis OSS 엔진 버전 5.0.6 이상을 사용하고 있습니다.
   - 전송 중 암호화가 활성화되지 않았습니다.
   - 다중 AZ가 활성화되어 있습니다.
   - Valkey 또는 Redis OSS 클러스터의 데이터에 맞게 사용할 수 있는 메모리가 충분합니다. 예약 메모리 설정을 올바르게 구성하는 방법은 Valkey 및 Redis용 예약 메모리 관리 OSS 섹션을 참조하세요.
   - 클러스터 모드가 비활성화된 경우 또는 CLI 콘솔을 사용하여 CLI 또는 Valkey 또는 Redis OSS 버전 OSS5.0.6 이상을 사용하는 경우 Valkey 또는 Redis 버전 2.8.21 이상에서 Valkey 또는 Redis OSS 버전 5.0.6 이상으로 직접 마이그레이션할 수 있습니다. 클러스터 모드가 활성화된 경우 가 CLI 또는 콘솔을 사용하여 CLI 또는 Redis OSS 버전 5.0.6 이상을 사용하는 경우 클러스터 모드가 활성화된 Valkey 또는 Redis OSS 버전에서 Redis OSS 버전 5.0.6 이상으로 직접 마이그레이션할 수 있습니다.
   - 소스와 대상의 샤드 수가 일치해야 합니다.
   - 글로벌 데이터 스토어에 포함되면 안 됩니다.
   - 데이터 계층화가 비활성화되어 있어야 합니다.
2. 오픈 소스 Valkey 또는 Redis OSS 및 ElastiCache 배포의 구성이 호환되는지 확인합니다.

   최소한 대상 ElastiCache 배포의 다음 모든 항목은 복제를 위해 Valkey 또는 Redis OSS 구성과 호환되어야 합니다.

   - 클러스터가 AUTH 활성화되지 않았어야 합니다.
   - 구성은 로 설정되어야 `protected-mode` 합니다no.

- Valkey 또는 Redis OSS 구성에 bind 구성이 있는 경우 ElastiCache 노드 요청을 허용하도록 업데이트해야 합니다.
- 논리적 데이터베이스 수는 ElastiCache 노드와 Valkey 또는 Redis OSS 클러스터에서 동일해야 합니다. 이 값은 Valkey 또는 Redis 구성databases에서 OSS 를 사용하여 설정됩니다.
- 데이터 수정을 수행하는 Valkey 또는 Redis OSS 명령은 , sync, , psync, 등의 데이터 복제가 성공할 수 있도록 이름을 바꾸면 안 됩니다infoconfigcommandcluster.
- Valkey 또는 Redis OSS 클러스터의 데이터를 로 복제하려면 이 추가 로드를 처리할 수 있는 충분한 CPU 메모리와 메모리가 있어야 ElastiCache합니다. 이 로드는 Valkey 또는 Redis OSS 클러스터에서 생성하고 네트워크를 통해 ElastiCache 노드로 전송된 RDB 파일에서 가져옵니다.
- 소스 클러스터의 모든 Valkey 또는 Redis OSS 인스턴스는 동일한 포트에서 실행되어야 합니다.

3. 다음을 ElastiCache 수행하여 인스턴스가 에 연결할 수 있는지 확인합니다.
   - 각 인스턴스의 IP 주소가 프라이빗 주소인지 확인합니다.
   - 인스턴스의 Valkey 또는 Redis와 동일한 가상 프라이빗 클라우드(VPC)OSS에 ElastiCache 배포를 할당하거나 생성합니다(권장).
   - VPCs 가 다른 경우 노드 간 액세스를 허용하도록 VPC 피어링을 설정합니다. VPC 피어링에 대한 자세한 내용은 섹션을 참조하세요Amazon에서 ElastiCache 캐시에 액세스하기 위한 액세스 패턴 VPC.
   - Valkey 또는 Redis OSS 인스턴스에 연결된 보안 그룹은 ElastiCache 노드의 인바운드 트래픽을 허용해야 합니다.
4. 데이터 마이그레이션이 완료된 후 애플리케이션이 ElastiCache 노드로 트래픽을 전달할 수 있는지 확인합니다. 자세한 내용은 Amazon에서 ElastiCache 캐시에 액세스하기 위한 액세스 패턴 VPC 단원을 참조하십시오.

데이터 마이그레이션 테스트

모든 사전 요구 사항이 완료되면 , AWS Management Console ElastiCache API또는 를 사용하여 마이그레이션 설정을 검증할 수 있습니다 AWS CLI. 다음 예제는 를 사용하는 방법을 보여줍니다CLI.

test-migration 명령을 다음 파라미터와 함께 호출하여 마이그레이션을 테스트합니다.

- --replication-group-id - 데이터를 마이그레이션할 복제 그룹의 ID입니다.
- --customer-node-endpoint-list - 데이터를 마이그레이션해야 하는 엔드포인트의 목록입니다. 목록에는 한 개의 요소만 있어야 합니다.

다음은 를 사용하는 예제입니다CLI.

```
aws elasticache test-migration --replication-group-id test-cluster --customer-node-endpoint-list "Address='10.0.0.241',Port=6379"
```

ElastiCache 는 실제 데이터 마이그레이션 없이 마이그레이션 설정을 검증합니다.

마이그레이션 시작

모든 사전 요구 사항이 완료되면 , AWS Management Console ElastiCache API또는 를 사용하여 데이터 마이그레이션을 시작할 수 있습니다 AWS CLI. 클러스터 모드가 활성화되어 있는 경우 슬롯 마이그레이션이 다르면 실시간 마이그레이션 전에 리샤딩이 수행됩니다. 다음 예제는 를 사용하는 방법을 보여줍니다CLI.

> **Note**
>
> `TestMigration` API 를 사용하여 마이그레이션 설정을 검증하는 것이 좋습니다. 그러나 이는 전적으로 선택 사항입니다.

`start-migration` 명령을 다음 파라미터로 호출하여 마이그레이션을 시작합니다.

- `--replication-group-id` - 대상 ElastiCache 복제 그룹의 식별자
- `--customer-node-endpoint-list` - DNS 또는 IP 주소가 있는 엔드포인트 목록과 소스 Valkey 또는 Redis OSS 클러스터가 실행 중인 포트입니다. 클러스터 모드가 비활성화된 경우와 클러스터 모드가 활성화된 경우에 모두 하나의 요소만 이 목록에 포함할 수 있습니다. 체인 복제를 활성화한 경우 엔드포인트는 Valkey 또는 Redis OSS 클러스터의 기본 노드 대신 복제본을 가리킬 수 있습니다.

다음은 를 사용하는 예제입니다CLI.

```
aws elasticache start-migration --replication-group-id test-cluster --customer-node-endpoint-list "Address='10.0.0.241',Port=6379"
```

이 명령을 실행하면 ElastiCache 기본 노드(각 샤드에 있음)가 Valkey 또는 Redis OSS 인스턴스(클러스터 지원 redis에서 동일한 슬롯을 소유한 해당 샤드에 있음)의 복제본이 되도록 자체 구성합니다. ElastiCache 클러스터 상태가 마이그레이션으로 변경되고 데이터가 Valkey 또는 Redis OSS 인스턴스에서 ElastiCache 기본 노드로 마이그레이션되기 시작합니다. Valkey 또는 Redis OSS 인스턴스의

데이터 크기와 로드에 따라 마이그레이션을 완료하는 데 시간이 걸릴 수 있습니다. Valkey 인스턴스 및 기본 노드에서 valkey-cli INFO 명령을 실행하여 마이그레이션 진행 상황을 확인할 수 있습니다. ElastiCache

복제에 성공하면 Valkey 또는 Redis OSS 인스턴스에 대한 모든 쓰기가 ElastiCache 클러스터로 전파됩니다. 읽기에 ElastiCache 노드를 사용할 수 있습니다. 하지만 클러스터에 ElastiCache 쓸 수는 없습니다. ElastiCache 기본 노드에 연결된 다른 복제본 노드가 있는 경우 이러한 복제본 노드는 ElastiCache 기본 노드에서 계속 복제됩니다. 이렇게 하면 Valkey 또는 Redis OSS 클러스터의 모든 데이터가 ElastiCache 클러스터의 모든 노드에 복제됩니다.

ElastiCache 기본 노드가 Valkey 또는 Redis OSS 인스턴스의 복제본이 될 수 없는 경우 여러 번 재시도한 후 최종적으로 기본 노드로 다시 승격합니다. 그러면 클러스터 상태가 ElastiCache 사용 가능한 로 변경되고 마이그레이션 시작 실패에 대한 복제 그룹 이벤트가 전송됩니다. 이러한 문제를 해결하려면 다음과 같이 하세요.

- 복제 그룹 이벤트를 확인합니다. 이벤트에서 나온 모든 구체적인 정보를 이용해 마이그레이션 실패 문제를 해결합니다.
- 이벤트가 구체적인 정보를 제공하지 않는 경우에는 마이그레이션을 위한 소스 및 대상 준비의 지침을 준수했는지 확인합니다.
- VPC 및 서브넷의 라우팅 구성이 ElastiCache 노드와 Valkey 또는 Redis OSS 인스턴스 간의 트래픽을 허용하는지 확인합니다.
- Valkey 또는 Redis OSS 인스턴스에 연결된 보안 그룹이 ElastiCache 노드의 입력 바인딩 트래픽을 허용하는지 확인합니다.
- 복제 관련 장애에 대한 자세한 내용은 인스턴스의 Valkey 또는 Redis OSS 로그를 확인하세요.

데이터 마이그레이션 진행 상황 확인

데이터 마이그레이션이 시작된 이후에 다음을 통해 진행 상황을 확인할 수 있습니다.

- Valkey 또는 RedisOSS`master_link_status`가 ElastiCache 기본 노드(들)up의 `INFO` 명령에 있는지 확인합니다. ElastiCache 콘솔에서도 이 정보를 찾을 수 있습니다. 클러스터를 선택하고 CloudWatch 지표 아래에서 기본 링크 상태 를 관찰합니다. 값이 1에 도달한 후 데이터가 동기화됩니다.
- Valkey 또는 Redis OSS 인스턴스에서 `INFO` 명령을 실행하여 ElastiCache 복제본이 온라인 상태인지 확인할 수 있습니다. 이렇게 하면 복제 지연 시간에 대한 정보도 제공됩니다.
- Valkey 또는 Redis OSS 인스턴스에서 CLIENT LIST 명령을 사용하여 클라이언트 출력 버퍼가 낮은지 확인합니다.

데이터 마이그레이션이 완료되면 데이터는 Valkey 또는 Redis OSS 클러스터의 프라이머리 노드(들)에 들어오는 새 쓰기와 동기화됩니다.

데이터 마이그레이션 완료

ElastiCache 클러스터를 잘라낼 준비가 되면 다음 파라미터와 함께 `complete-migration` CLI 명령을 사용합니다.

- `--replication-group-id` - 복제 그룹의 식별자입니다.
- `--force` - 데이터가 동기화 중인지 확인하지 않고 마이그레이션을 강제 중단하는 값입니다.

다음은 예입니다.

```
aws elasticache complete-migration --replication-group-id test-cluster
```

이 명령을 실행하면 ElastiCache 기본 노드(각 샤드에 있음)가 Valkey 또는 Redis OSS 인스턴스에서 복제를 중지하고 기본 노드로 승격합니다. 이러한 승격은 보통 몇 분 내에 완료됩니다. 기본 노드로의 승격을 확정하려면 이벤트 `Complete Migration successful for test-cluster`를 확인합니다. 이 시점에서 애플리케이션을 ElastiCache 쓰기 및 읽기로 지시할 수 있습니다. ElastiCache 클러스터 상태는 마이그레이션에서 사용 가능한 로 변경되어야 합니다.

프라이머리로 승격이 실패하면 ElastiCache 프라이머리 노드가 Valkey 또는 Redis OSS 인스턴스에서 계속 복제됩니다. ElastiCache 클러스터는 계속 마이그레이션 상태이며 실패에 대한 복제 그룹 이벤트 메시지가 전송됩니다. 이 문제를 해결하려면 다음과 같이 하세요.

- 복제 그룹 이벤트를 확인합니다. 이벤트에서 나온 구체적인 정보를 이용해 마이그레이션 실패 문제를 해결합니다.
- 데이터가 동기화 중이 아니라는 이벤트 메시지가 나타날 수 있습니다. 그렇다면 ElastiCache 기본 가 Valkey 또는 Redis OSS 인스턴스에서 복제할 수 있고 둘 다 동기화되어 있는지 확인합니다. 여전히 동기화를 중단하고 싶은 마음이 있으면 `–force` 옵션을 통해 이전의 명령을 실행할 수 있습니다.
- ElastiCache 노드 중 하나가 교체 중인 경우 이벤트 메시지가 표시될 수 있습니다. 대체 작업이 완료되고 난 후에 전체 마이그레이션 단계를 다시 시도할 수 있습니다.

콘솔을 사용해 온라인 데이터 마이그레이션 수행

AWS Management Console 를 사용하여 클러스터에서 Valkey 또는 Redis OSS 클러스터로 데이터를 마이그레이션할 수 있습니다.

콘솔을 사용해 온라인 데이터 마이그레이션을 수행하려면

1. 콘솔에 로그인하고 에서 ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 새 Valkey 또는 Redis OSS 클러스터를 생성하거나 기존 클러스터를 선택합니다. 클러스터가 다음 요구 사항을 충족하는지 확인합니다.

   - 엔진 버전은 Valkey 7.2 이상 또는 Redis OSS 5.0.6 이상이어야 합니다.
   - 클러스터가 AUTH 활성화되지 않았어야 합니다.
   - 구성은 로 설정되어야 `protected-mode` 합니다no.
   - Valkey 또는 Redis OSS 구성에 `bind` 구성이 있는 경우 ElastiCache 노드 요청을 허용하도록 업데이트해야 합니다.
   - 데이터베이스 수는 ElastiCache 노드와 Valkey 또는 Redis OSS 클러스터 간에 동일해야 합니다. 이 값은 엔진 구성`databases`에서 를 사용하여 설정됩니다.
   - 데이터 수정을 수행하는 Valkey 또는 Redis OSS 명령은 데이터 복제가 성공할 수 있도록 이름을 변경해서는 안 됩니다.
   - Valkey 또는 Redis OSS 클러스터의 데이터를 로 복제하려면 이 추가 로드를 처리할 수 있는 충분한 CPU 메모리와 메모리가 있어야 ElastiCache합니다. 이 로드는 Valkey 또는 Redis OSS 클러스터에서 생성하고 네트워크를 통해 ElastiCache 노드로 전송된 RDB 파일에서 가져옵니다.
   - 클러스터는 사용 가능 상태입니다.
3. 클러스터가 선택된 상태에서 작업에서 엔드포인트에서 데이터 마이그레이션을 선택합니다.
4. 엔드포인트에서 데이터 마이그레이션 대화 상자에서 IP 주소와 Valkey 또는 Redis OSS 클러스터를 사용할 수 있는 포트를 입력합니다.

   > **Important**
   >
   > IP 주소는 정확해야 합니다. 주소를 잘못 입력하면 마이그레이션이 실패합니다.

5. Start Migration(마이그레이션 시작)을 선택합니다.

   클러스터에서 마이그레이션이 시작되면 상태가 설정 변경으로 바뀐 다음, Migrating(마이그레이션 중)으로 바뀝니다.
6. 탐색 창에서 이벤트를 선택해 마이그레이션 진행 상황을 모니터링합니다.

마이그레이션이 진행되는 동안 언제라도 마이그레이션을 중단할 수 있습니다. 이렇게 하려면 클러스터를 선택하고 작업에서 데이터 마이그레이션 중지를 선택합니다. 그러면 클러스터의 상태가 사용 가능로 바뀝니다.

마이그레이션이 성공하면 클러스터의 상태가 사용 가능으로 바뀌고 이벤트 로그에 다음과 같은 내용이 표시됩니다.

`Migration operation succeeded for replication group` *`ElastiCacheClusterName`*`.`

마이그레이션이 실패하면 클러스터의 상태가 사용 가능으로 바뀌고 이벤트 로그에 다음과 같은 내용이 표시됩니다.

`Migration operation failed for replication group` *`ElastiCacheClusterName`*`.`

에 대한 리전 및 가용 영역 선택 ElastiCache

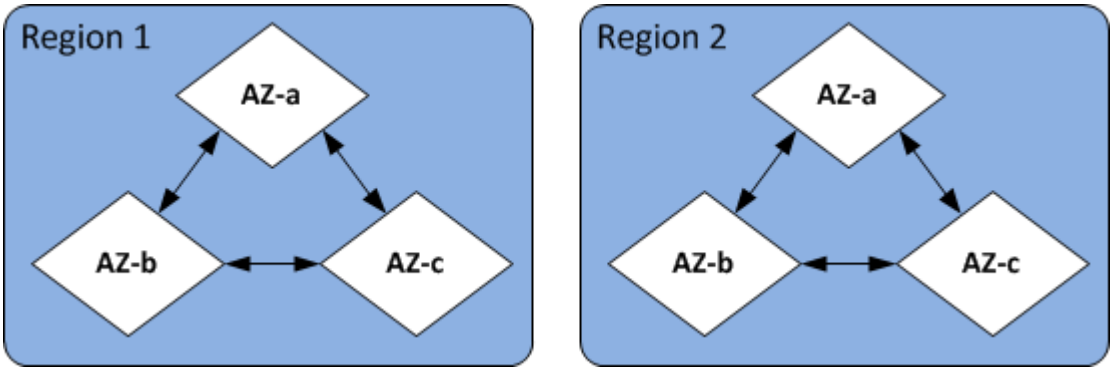
해당 엔드포인트를 사용하여 리전 및 가용 영역을 지정하여 ElastiCache 클러스터에 추가 확장성과 안정성을 제공할 수 있습니다.

AWS 클라우드 컴퓨팅 리소스는 가용성이 높은 데이터 센터 시설에 수용됩니다. 추가 확장성 및 안정성을 제공하기 위해 이러한 데이터 센터 시설은 여러 물리적 위치에 배치됩니다. 이러한 위치는 리전 및 가용 영역으로 분류됩니다.

AWS 리전은 크고 별도의 지리적 위치에 널리 분산되어 있습니다. 가용 영역은 다른 가용 영역의 장애로부터 격리되도록 설계된 AWS 리전 내의 고유한 위치입니다. 동일한 AWS 리전의 다른 가용 영역에 저렴하고 지연 시간이 짧은 네트워크 연결을 제공합니다.

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

특정 리전에서 클러스터를 생성하거나 사용하려면 해당하는 리전 서비스 엔드포인트를 사용하세요. 서비스 엔드포인트는 지원되는 리전 및 엔드포인트 섹션을 참조하세요.



리전 및 가용 영역

주제

- Memcached의 가용 영역 고려 사항
- 노드 찾기
- 지원되는 리전 및 엔드포인트
- 에서 로컬 영역 사용 ElastiCache
- 에서 Outposts 사용 ElastiCache

Memcached의 가용 영역 고려 사항

한 리전 내의 여러 가용 영역에 Memcached 노드를 배포하면 한 가용 영역 내의 정전과 같은 치명적인 장애의 영향으로부터 사용자를 보호할 수 있습니다.

서버리스 캐시

ElastiCache 서버리스 캐싱은 여러 가용 영역에 걸쳐 고가용성 캐시를 생성합니다. 서버리스 클러스터를 생성할 VPC 때와 동일한 다양한 가용 영역의 서브넷을 지정하거나 기본 에서 서브넷을 자동으로 ElastiCache 선택할 수 있습니다VPC.

자체 ElastiCache (Memcached) 클러스터 설계

Memcached 클러스터에는 최대 300개의 노드가 있을 수 있습니다. Memcached 클러스터에 노드를 생성하거나 추가할 때 모든 노드에 대해 단일 가용 영역을 지정하거나, ElastiCache 가 모든 노드에 대해 단일 가용 영역을 선택하거나, 각 노드에 대해 가용 영역을 지정하거나, 가 각 노드에 대해 가용 영역을 선택 ElastiCache 하도록 허용할 수 있습니다. 기존 Memcached 클러스터에 새 노드를 추가할 때 다른 가용 영역에 노드를 생성할 수 있습니다. 캐시 노드가 생성되면 가용 영역을 수정할 수 없습니다.

단일 가용 영역 클러스터의 클러스터에 노드가 여러 가용 영역에 분산되도록 하려면 에서 다양한 가용 영역에 새 노드를 생성할 ElastiCache 수 있습니다. 그런 다음 원래 캐시 노드의 일부 또는 전체를 삭제할 수 있습니다. 이 방법이 권장 방법입니다.

Memcached 노드를 단일 가용 영역에서 여러 가용 영역으로 마이그레이션하려면

1. 원하는 가용 영역에서 새 캐시 노드를 생성하여 클러스터를 수정합니다. 요청에서 다음을 수행합니다.

   - AZMode (CLI: - -az-mode)를 로 설정합니다cross-az.
   - NumCacheNodes (CLI: - -num-cache-nodes)를 현재 활성 캐시 노드 수와 생성하려는 새 캐시 노드 수로 설정합니다.
   - NewAvailabilityZones (CLI: - -new-availability-zones)를 에서 새 캐시 노드를 생성할 영역 목록으로 설정합니다. 각 새 노드의 가용 영역을 ElastiCache 확인하려면 목록을 지정하지 마세요.
   - ApplyImmediately (CLI: - -apply-immediately)를 true로 설정합니다.

   > **Note**
   >
   > Auto Discovery를 사용하지 않는 경우 클라이언트 애플리케이션을 새 캐시 노드 엔드포인트로 업데이트해야 합니다.

   다음 단계로 넘어가기 전에 Memcached 노드가 완전히 생성되어 사용 가능한지 확인합니다.

2. 원래 가용 영역에서 더 이상 필요하지 않은 노드를 제거하여 클러스터를 수정합니다. 요청에서 다음을 수행합니다.

   - 이 수정이 적용된 후 원하는 활성 캐시 노드 수로 NumCacheNodes (CLI: - -num-cache-nodes)를 설정합니다.
   - CacheNodeIdsToRemove (CLI: - -nodes-to-remove)를 클러스터에서 제거할 캐시 노드 목록으로 설정합니다.

     IDs 나열된 캐시 노드 수는 현재 활성 노드 수에서 의 값을 뺀 수와 같아야 합니다NumCacheNodes.
   - (선택 사항) ApplyImmediately (CLI: - -apply-immediately)를 true로 설정합니다.

`ApplyImmediately` (CLI: `- -apply-immediately`)를 true로 설정하지 않으면 다음 유지 관리 기간에 노드 삭제가 수행됩니다.

노드 찾기

Amazon은 단일 또는 여러 가용 영역()에서 클러스터의 모든 노드를 찾는 것을 ElastiCache 지원합니다 AZs. 또한 여러AZs(권장)에서 노드를 찾으려는 경우 각 노드에 대해 AZ ElastiCache 를 선택하거나 에서 노드ElastiCache 를 선택할 수 있습니다.

서로 다른 에서 노드를 찾AZs음으로써 한 AZ에서 정전과 같은 장애로 인해 전체 시스템이 실패할 가능성을 제거합니다. 테스트 결과 한 AZ에서 모든 노드를 찾거나 여러 에 분산하는 간에 상당한 지연 시간 차이가 없는 것으로 나타났습니다AZs.

기존 클러스터를 수정할 때 노드를 추가거나 클러스터를 생성할 때 각 노드에 대한 AZ를 지정할 수 있습니다. 자세한 내용은 다음 자료를 참조하세요.

- Memcached용 클러스터 생성
- Valkey 또는 Redis용 클러스터 생성 OSS
- ElastiCache 클러스터 수정
- ElastiCache 클러스터에 노드 추가

지원되는 리전 및 엔드포인트

Amazon ElastiCache 은 여러 AWS 리전에서 사용할 수 있습니다. 즉, 요구 사항을 충족하는 위치에서 ElastiCache 클러스터를 시작할 수 있습니다. 예를 들어 고객에게 가장 가까운 AWS 리전에서 를 시작하거나 특정 법적 요구 사항을 충족하기 위해 특정 AWS 리전에서 를 시작할 수 있습니다.

각 리전은 다른 리전에서 완전히 격리되도록 설계되었습니다. 각 리전 안에는 가용 영역(AZ)이 여러 개 있습니다. ElastiCache 서버리스 캐시는 고가용성을 위해 여러 가용 영역(두 가용 영역에서 데이터가 복제`us-west-1`되는 는 제외)에서 데이터를 자동으로 복제합니다. 자체 ElastiCache 클러스터를 설계할 때 내결함성을 달성하기 위해 다른 에서 노드AZs를 시작하도록 선택할 수 있습니다. 리전 및 가용 영역에 대한 자세한 내용은 이 주제의 맨 위에 있는 에 대한 리전 및 가용 영역 선택 ElastiCache 섹션을 참조하십시오.

ElastiCache 가 지원되는 리전

리전 이름/리전	Endpoint	프로토콜	
US East (Ohio) Region `us-east-2`	`elasticache.us-east-2.amazonaws.com`	HTTPS	

리전 이름/리전	Endpoint	프로토콜	
미국 동부(버지니아 북부) 리전 us-east-1	elasticache.us-east-1.amazonaws.com	HTTPS	
미국 서부(캘리포니아 북부) 리전 us-west-1	elasticache.us-west-1.amazonaws.com	HTTPS	
미국 서부(오레곤) 리전 us-west-2	elasticache.us-west-2.amazonaws.com	HTTPS	
캐나다(중부) 리전 ca-central-1	elasticache.ca-central-1.amazonaws.com	HTTPS	
캐나다(서부) 리전 ca-west-1	elasticache.ca-west-1.amazonaws.com	HTTPS	
아시아 태평양(자카르타) ap-southeast-3	elasticache.ap-southeast-3.amazonaws.com	HTTPS	
Asia Pacific (Mumbai) Region ap-south-1	elasticache.ap-south-1.amazonaws.com	HTTPS	
아시아 태평양(하이데라바드) 리전 ap-south-2	elasticache.ap-south-2.amazonaws.com	HTTPS	

리전 이름/리전	Endpoint	프로토콜	
아시아 태평양(도쿄) 리전 ap-northeast-1	elasticache.ap-northeast-1.amazonaws.com	HTTPS	
Asia Pacific (Seoul) Region ap-northeast-2	elasticache.ap-northeast-2.amazonaws.com	HTTPS	
Asia Pacific (Osaka) Region ap-northeast-3	elasticache.ap-northeast-3.amazonaws.com	HTTPS	
아시아 태평양(싱가포르) 리전 ap-southeast-1	elasticache.ap-southeast-1.amazonaws.com	HTTPS	
아시아 태평양(시드니) 리전 ap-southeast-2	elasticache.ap-southeast-2.amazonaws.com	HTTPS	
Europe (Frankfurt) Region eu-central-1	elasticache.eu-central-1.amazonaws.com	HTTPS	
유럽(취리히) 리전 eu-central-2	elasticache.eu-central-2.amazonaws.com	HTTPS	
Europe (Stockholm) Region eu-north-1	elasticache.eu-north-1.amazonaws.com	HTTPS	

리전 이름/리전	Endpoint	프로토콜	
Middle East (Bahrain) Region me-south-1	elasticache.me-south-1.amazonaws.com	HTTPS	
중동(UAE) 리전 me-central-1	elasticache.me-central-1.amazonaws.com	HTTPS	
Europe (Ireland) Region eu-west-1	elasticache.eu-west-1.amazonaws.com	HTTPS	
Europe (London) Region eu-west-2	elasticache.eu-west-2.amazonaws.com	HTTPS	
EU(파리) 리전 eu-west-3	elasticache.eu-west-3.amazonaws.com	HTTPS	
Europe (Milan) Region eu-south-1	elasticache.eu-south-1.amazonaws.com	HTTPS	
유럽(스페인) 리전 eu-south-2	elasticache.eu-south-2.amazonaws.com	HTTPS	
South America (São Paulo) Region sa-east-1	elasticache.sa-east-1.amazonaws.com	HTTPS	

리전 이름/리전	Endpoint	프로토콜	
중국(베이징) 리전 cn-north-1	elasticache.cn-north-1.amazonaws.com.cn	HTTPS	
중국(닝샤) 리전 cn-northwest-1	elasticache.cn-northwest-1.amazonaws.com.cn	HTTPS	
Asia Pacific (Hong Kong) Region ap-east-1	elasticache.ap-east-1.amazonaws.com	HTTPS	
아프리카(케이프타운) 리전 af-south-1	elasticache.af-south-1.amazonaws.com	HTTPS	
Israel (Tel Aviv) Region il-central-1	elasticache.il-central-1.amazonaws.com	HTTPS	
AWS GovCloud (미국 서부) us-gov-west-1	elasticache.us-gov-west-1.amazonaws.com	HTTPS	
AWS GovCloud (미국 동부) us-gov-east-1	elasticache.us-gov-east-1.amazonaws.com	HTTPS	

에서 AWS GovCloud (미국)을 사용하는 방법에 대한 자세한 내용은 AWS GovCloud (미국) 리전의 서비스를 ElastiCache ElastiCache참조하세요.

일부 리전은 노드 유형의 하위 집합을 지원합니다. AWS 리전별 지원되는 노드 유형 표는 섹션을 참조하세요AWS 리전별로 지원되는 노드 유형.

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

에서 로컬 영역 사용 ElastiCache

로컬 영역은 사용자와 지리적으로 가까운 AWS 리전의 확장입니다. 새 서브넷을 생성하고 로컬 영역에 할당하여 상위 AWS 리전에서 로컬 영역으로 가상 프라이빗 클라우드(VPC)를 확장할 수 있습니다. 로컬 영역에서 서브넷을 생성하면 가 해당 로컬 영역으로 확장VPC됩니다. 로컬 영역의 서브넷은 의 다른 서브넷과 동일하게 작동합니다VPC.

로컬 영역을 사용하면 ElastiCache 클러스터와 같은 리소스를 사용자와 가까운 여러 위치에 배치할 수 있습니다.

ElastiCache 클러스터를 생성할 때 로컬 영역에서 서브넷을 선택할 수 있습니다. Local Zones는 인터넷에 대한 자체 연결을 가지고 있으며 AWS Direct Connect를 지원합니다. 따라서 로컬 영역에서 생성된 리소스는 지연 시간이 매우 짧은 통신으로 로컬 사용자에게 서비스를 제공할 수 있습니다. 자세한 내용은 AWS Local Zones를 참조하세요.

로컬 영역은 AWS 리전 코드와 같은 위치를 나타내는 식별자로 표시됩니다`us-west-2-lax-1a`.

현재 사용 가능한 Local Zones는 `us-west-2-lax-1a` 및 `us-west-2-lax-1b`입니다.

ElastiCache 로컬 영역에는 다음 제한이 적용됩니다.

- 글로벌 데이터 스토어는 지원되지 않습니다.
- 온라인 마이그레이션은 지원되지 않습니다.
- 현재 Local Zones에서 지원되는 노드 유형은 다음과 같습니다.
  - 현재 세대:

    M5 노드 유형: `cache.m5.large`, `cache.m5.xlarge`, `cache.m5.2xlarge`, `cache.m5.4xlarge`, `cache.m5.12xlarge`, `cache.m5.24xlarge`

    R5 노드 유형: `cache.r5.large`, `cache.r5.xlarge`, `cache.r5.2xlarge`, `cache.r5.4xlarge`, `cache.r5.12xlarge`, `cache.r5.24xlarge`

    T3 노드 유형: `cache.t3.micro`, `cache.t3.small`, `cache.t3.medium`

로컬 영역 활성화

1. Amazon EC2 콘솔에서 로컬 영역을 활성화합니다.

   자세한 내용은 Amazon EC2 사용 설명서의 로컬 영역 활성화를 참조하세요.
2. 로컬 영역에서 서브넷을 만듭니다.

   자세한 내용은 Amazon VPC 사용 설명서의 에서 서브넷 생성을 VPC 참조하세요.
3. 로컬 영역에서 ElastiCache 서브넷 그룹을 생성합니다.

   ElastiCache 서브넷 그룹을 생성할 때 로컬 영역의 가용 영역 그룹을 선택합니다.

   자세한 내용은 서브넷 그룹 생성 단원을 참조하십시오.
4. 로컬 영역에서 ElastiCache 서브넷을 사용하는 ElastiCache (Memcached) 클러스터를 생성합니다.

   자세한 내용은 Memcached 클러스터 생성(콘솔) 단원을 참조하십시오.
5. 로컬 영역에서 ElastiCache 서브넷을 사용하는 ElastiCache (Redis OSS) 클러스터를 생성합니다. 자세한 내용은 다음 중 하나의 주제를 참조하세요:

   - Valkey(클러스터 모드 비활성화됨) 클러스터 생성(콘솔)
   - Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 생성(콘솔)

에서 Outposts 사용 ElastiCache

AWS Outposts를 와 함께 사용할 수 있습니다 ElastiCache. Outposts는 AWS 인프라, 서비스, APIs및 도구를 고객 구내로 확장하는 완전 관리형 서비스입니다. AWS 관리형 인프라에 로컬 액세스를 제공함으로써 AWS Outposts를 사용하면 고객은 AWS 리전과 동일한 프로그래밍 인터페이스를 사용하여 온프레미스에서 애플리케이션을 구축하고 실행할 수 있으며 지연 시간 및 로컬 데이터 처리 요구 사항을 줄이기 위해 로컬 컴퓨팅 및 스토리지 리소스를 사용할 수 있습니다. Outpost는 고객 사이트에 배포된 AWS 컴퓨팅 및 스토리지 용량 풀입니다. 는 이 용량을 AWS 리전의 일부로 AWS 운영, 모니터링 및 관리합니다. Outpost에서 서브넷을 생성하고 ElastiCache 클러스터와 같은 AWS 리소스를 생성할 때 지정할 수 있습니다.

> **Note**
>
> 이 버전에는 다음과 같은 제한 사항이 적용됩니다.

- ElastiCache for Outposts는 M5 및 R5 노드 패밀리만 지원합니다.
- 다중 AZ(Outpost 간 복제)는 지원되지 않습니다.
- 실시간 마이그레이션은 지원되지 않습니다.
- 로컬 스냅샷은 지원되지 않습니다.
- 엔진 로그와 느린 로그는 활성화할 수 없습니다.
- ElastiCache Outposts의 는 CoIP 를 지원하지 않습니다.
- ElastiCache cn-north-1, cn-northwest-1 및 ap-northeast-3 리전에서는 for Outposts가 지원되지 않습니다.

ElastiCache 콘솔에서 Outposts 사용

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 Valkey 캐시 , Redis OSS 캐시 또는 Memcached 캐시 를 선택합니다.
3. Valkey 캐시 를 선택한 경우 Valkey 캐시 생성 을 선택합니다. Redis OSS 캐시 를 선택한 경우 Redis OSS 캐시 생성 을 선택합니다. Memcached 캐시 를 선택한 경우 Memcached 캐시 생성을 선택합니다.
4. 클러스터 설정 에서 자체 캐시 및 클러스터 캐시 설계를 선택합니다. 클러스터 모드를 비활성화로 설정한 상태로 둡니다. 그런 다음 캐시에 대한 이름과 선택적 설명을 생성합니다.
5. 위치 에서 온프레미스 를 선택합니다.
6. 온프레미스 섹션에 Outpost ID 필드가 표시됩니다. 클러스터가 실행될 ID를 입력합니다.

   클러스터 설정의 모든 추가 설정은 기본값으로 유지될 수 있습니다.
7. 연결 에서 새 서브넷 그룹 생성을 선택하고 VPC ID 를 입력합니다. 나머지는 기본값으로 두고 다음 를 선택합니다.

온프레미스 옵션 구성

사용 가능한 Outpost를 선택하여 캐시 클러스터를 추가하거나 사용 가능한 Outpost가 없는 경우 다음 단계를 사용하여 새 Outpost를 생성할 수 있습니다.

온프레미스 옵션에서 다음을 수행합니다.

1. 원하는 엔진에 따라 Valkey 설정 , Redis OSS 설정 또는 Memcached 설정 에서 다음을 수행합니다.

   a. 이름: 클러스터의 이름을 입력합니다.

   b. 설명 : 클러스터에 대한 설명을 입력합니다.

   c. 엔진 버전 호환성: 엔진 버전은 AWS Outpost 리전을 기반으로 합니다.

   d. 포트 : Valkey 또는 Redis 의 경우 기본 포트 6379를 OSS수락합니다. Memcached의 경우 기본 포트 11211을 수락합니다. 다른 포트를 사용하려면 포트 번호를 입력합니다.

   e. 파라미터 그룹: 드롭다운을 사용하여 기본 또는 사용자 지정 파라미터 그룹을 선택합니다.

   f. 노드 유형: 사용 가능한 인스턴스는 Outposts 가용성에 기반합니다. Valkey 또는 Redis OSS를 사용하는 경우 용 Porting Assistant.NET for Outposts는 M5 및 R5 노드 패밀리만 지원합니다. 드롭다운 목록에서 Outposts를 선택한 다음 이 클러스터에 사용할 사용 가능한 노드 유형을 선택합니다. 그런 다음 저장을 선택합니다.

   g. 복제본 수: 이 복제 그룹에 대해 생성할 읽기 전용 복제본 수를 입력합니다. 1개 이상 5개 이하의 읽기 전용 복제본이 있어야 합니다. 기본값은 2입니다.

      읽기 전용 복제본의 자동 생성된 이름은 기본 클러스터 이름과 동일한 패턴을 따릅니다. 즉, 끝에 대시와 `-002`부터 시작하는 순차적인 3자리 숫자가 추가됩니다. 예를 들어, 복제 그룹의 이름이 `MyGroup`인 경우 2차 복제본의 이름은 `MyGroup-002`, `MyGroup-003`, `MyGroup-004`, `MyGroup-005`, `MyGroup-006`입니다.

2. 연결에서:

   a. 서브넷 그룹: 목록에서 새로 생성을 선택합니다.

      - 이름: 서브넷 그룹의 이름을 입력합니다.
      - 설명: 서브넷 그룹에 대한 설명을 입력합니다.
      - VPC ID : VPC ID는 Outpost 와 일치해야 합니다VPC. OutpostsIDs에 서브넷VPC이 없는 를 선택하면 목록이 비어 있습니다.
      - 가용 영역 또는 Outpost: 사용 중인 Outpost를 선택합니다.
      - 서브넷 ID: Outpost에 사용할 수 있는 서브넷 ID를 선택합니다. IDs 사용 가능한 서브넷이 없는 경우 서브넷을 생성해야 합니다. 자세한 내용은 서브넷 생성을 참조하세요.

   b. 생성을 선택합니다.

Outpost 클러스터 세부 정보 보기

목록 페이지에서 AWS Outpost에 속하는 클러스터를 선택하고 클러스터 세부 정보 를 볼 때 다음 사항에 유의합니다.

- 가용 영역 : (ARNAmazon 리소스 이름) 및 AWS 리소스 번호를 사용하여 Outpost를 나타냅니다.
- Outpost 이름: AWS Outpost의 이름입니다.

에서 Outposts 사용 AWS CLI

AWS Command Line Interface (AWS CLI)를 사용하여 명령줄에서 여러 AWS 서비스를 제어하고 스크립트를 통해 자동화할 수 있습니다. 임시(일회성) 작업에 를 AWS CLI 사용할 수 있습니다.

다운로드 및 구성 AWS CLI

는 Windows, macOS 또는 Linux에서 AWS CLI 실행됩니다. 다음 절차에 따라 다운로드 및 구성합니다.

를 다운로드, 설치 및 구성하려면 CLI

1. AWS 명령줄 인터페이스 웹 페이지에서 AWS CLI를 다운로드합니다.
2. AWS Command Line Interface 사용 설명서의 설치 AWS CLI 및 구성 AWS CLI 지침을 따릅니다.

Outposts와 AWS CLI 함께 사용

다음 CLI 작업을 사용하여 Outposts를 사용하는 캐시 클러스터를 생성합니다.

- create-cache-cluster - 이 작업을 사용하면 `outpost-mode` 파라미터는 캐시 클러스터의 노드가 단일 Outpost에서 생성되는지 아니면 여러 Outpost에서 생성되는지를 지정하는 값을 수락합니다.

> **Note**
>
> 현재,`single-outpost` 모드만 지원됩니다.

```
aws elasticache create-cache-cluster \
   --cache-cluster-id cache cluster id \
   --outpost-mode single-outpost \
```

작업 ElastiCache

이 섹션에서는 ElastiCache 구현의 다양한 구성 요소를 관리하는 방법에 대한 세부 정보를 찾을 수 있습니다.

주제

스냅샷 및 복원

Valkey, Redis OSS또는 Serverless Memcached를 실행하는 Amazon ElastiCache 캐시는 스냅샷을 생성하여 데이터를 백업할 수 있습니다. 백업을 사용하여 캐시를 복원하거나 데이터를 새 캐시에 시드할 수 있습니다. 백업은 캐시의 모든 데이터와 캐시의 메타데이터로 구성됩니다. 모든 백업은 Amazon Simple Storage Service(S3)에 쓰여지므로 내구성 있는 스토리지가 확보됩니다. 언제든지 새 Valkey, Redis OSS또는 Serverless Memcached 캐시를 생성하고 백업의 데이터로 채워 데이터를 복원할 수 있습니다. 를 사용하면 ElastiCache, AWS Management Console AWS Command Line Interface (AWS CLI) 및 를 사용하여 백업을 관리할 수 있습니다 ElastiCache API.

캐시를 삭제할 계획을 세우고 데이터를 보존하는 것이 중요한 경우 추가적인 예방 조치를 취할 수 있습니다. 이렇게 하려면 먼저 수동 백업을 생성하고 사용 가능한 상태인지 확인한 다음 캐시를 삭제합니다. 이렇게 하면 백업에 실패하더라도 캐시 데이터를 계속 사용할 수 있습니다. 앞서 설명한 모범 사례에 따라 백업을 다시 시도할 수 있습니다.

주제

- 자동 백업 예약
- 수동 백업 지원
- 최종 백업 생성
- 백업 설명
- 백업 복사
- 백업 내보내기
- 백업에서 새 캐시로 복원
- 백업 삭제
- 백업 태그 지정
- 자습서: 외부에서 생성된 백업을 사용하여 자체 설계된 새 클러스터 검색

백업 제약 조건

백업을 계획하거나 만들려는 경우 다음 제약 조건을 고려해야 합니다.

- 백업 및 복원은 Valkey, Redis OSS 또는 Serverless Memcached에서 실행되는 캐시에만 지원됩니다.
- Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터의 경우 `cache.t1.micro` 노드에서 백업 및 복원이 지원되지 않습니다. 다른 모든 캐시 노드 유형은 지원됩니다.
- Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 경우 모든 노드 유형에 대해 백업 및 복원이 지원됩니다.
- 연속 24시간 동안 서버리스 캐시당 24개 이하의 수동 백업을 생성할 수 있습니다. Valkey 및 Redis OSS 자체 설계 클러스터의 경우 클러스터의 노드당 20개 이하의 수동 백업을 생성할 수 있습니다.
- Valkey 또는 RedisOSS(클러스터 모드 활성화됨)는 클러스터 수준(API 또는 의 경우 복제 그룹 수준)에서만 백업 가져오기CLI를 지원합니다. Valkey 또는 RedisOSS(클러스터 모드 활성화됨)는 샤드 수준(API 또는 의 경우 노드 그룹 수준)에서 백업 가져오기CLI를 지원하지 않습니다.
- 백업 프로세스 중에는 서버리스 캐시에서 다른 API 또는 CLI 작업을 실행할 수 없습니다. 백업 중에 자체 설계된 클러스터에서 API 또는 CLI 작업을 실행할 수 있습니다.
- 데이터 계층화와 함께 Valkey 또는 Redis OSS 캐시를 사용하는 경우 백업을 Amazon S3로 내보낼 수 없습니다.
- r6gd 노드 유형을 사용하는 클러스터의 백업은 r6gd 노드 유형을 사용하는 클러스터에만 복원할 수 있습니다.

자체 설계된 클러스터 백업이 성능에 미치는 영향

서버리스 캐시의 백업은 성능뿐 아니라 애플리케이션에도 영향을 미치지 않습니다. 하지만 자체 설계된 클러스터의 백업을 생성할 때는 가용할 수 있는 예약 메모리에 따라 성능에 어느 정도 영향을 미칠 수 있습니다. 자체 설계된 클러스터에 대한 백업은 ElastiCache (Memcached)에서는 사용할 수 없지만 ElastiCache (Redis)에서는 사용할 수 있습니다OSS.

다음은 자체 설계된 클러스터에서 백업 성능을 개선하기 위한 지침입니다.

- `reserved-memory-percent` 파라미터 설정 - 과도한 호출을 완화하려면 reserved-memory-percent 파라미터를 설정하는 것이 좋습니다. 이 파라미터는 Valkey와 Redis가 노드의 사용 가능한 모든 메모리를 OSS 사용하지 못하도록 하고 호출량을 줄이는 데 도움이 될 수 있습니다. 더 큰 노드를 사용하기만 해도 성능 개선을 확인할 수 있습니다. 예약 메모리 및 reserved-memory-percent 파라미터에 대한 자세한 내용은 섹션을 참조하세요Valkey 및 Redis용 예약 메모리 관리 OSS.

- 읽기 전용 복제본에서 백업 생성 - 노드가 둘 이상인 노드 OSS 그룹에서 Valkey 또는 Redis를 실행하는 경우 기본 노드 또는 읽기 전용 복제본 중 하나에서 백업을 가져올 수 있습니다. 에 필요한 시스템 리소스로 인해 읽기 전용 복제본 중 하나에서 백업을 생성하는 것이 BGSAVE좋습니다. 복제본에서 백업을 생성하는 동안 기본 노드는 BGSAVE 리소스 요구 사항의 영향을 받지 않습니다. 기본 노드는 속도를 늦추지 않고 계속해서 요청을 처리할 수 있습니다.

  이렇게 하려면 수동 백업 생성(콘솔) 섹션을 참조하고, 백업 생성 창의 클러스터 이름 필드에서 기본 값인 기본 노드 대신 복제본을 선택합니다.

복제 그룹을 삭제하고 최종 백업을 요청하는 경우 ElastiCache 항상 기본 노드에서 백업을 가져옵니다. 이렇게 하면 복제 그룹이 삭제되기 전에 최신 Valkey 또는 Redis OSS 데이터를 캡처할 수 있습니다.

자동 백업 예약

모든 Valkey 또는 Redis OSS 서버리스 캐시 또는 자체 설계된 클러스터에 자동 백업을 활성화할 수 있습니다. 자동 백업이 활성화되면 는 매일 캐시의 백업을 ElastiCache 생성합니다. 캐시에는 영향을 주지 않으며 변경 사항이 즉시 적용됩니다. 자동 백업은 데이터 손실을 막는 데 도움이 됩니다. 실패할 경우 새로운 캐시를 생성해 최신 백업에서 모든 데이터를 복원할 수 있습니다. 그 결과 웜 부팅 캐시로 전환되고 데이터가 사전 로드되어 사용할 수 있게 됩니다. 자세한 내용은 백업에서 새 캐시로 복원 단원을 참조하십시오.

모든 Memcached Serverless 캐시에 자동 백업을 활성화할 수 있습니다. 자동 백업이 활성화되면 는 매일 캐시의 백업을 ElastiCache 생성합니다. 캐시에는 영향을 주지 않으며 변경 사항이 즉시 적용됩니다. 자동 백업은 데이터 손실을 막는 데 도움이 됩니다. 실패할 경우 새로운 캐시를 생성해 최신 백업에서 모든 데이터를 복원할 수 있습니다. 그 결과 웜 부팅 캐시로 전환되고 데이터가 사전 로드되어 사용할 수 있게 됩니다. 자세한 내용은 백업에서 새 캐시로 복원 단원을 참조하십시오.

자동 백업을 예약할 때는 다음 설정을 계획해야 합니다.

- 백업 시작 시간 - 가 백업 생성을 ElastiCache 시작하는 시간입니다. 가장 편리한 시간에 백업 기간을 설정할 수 있습니다. 백업 기간을 지정하지 않으면 가 백업 기간을 자동으로 ElastiCache 할당합니다.

- 백업 보존 제한 - Amazon S3에서 백업을 보존되는 일수입니다. 예를 들어, 보존 제한을 5로 설정하면 오늘 만든 백업이 5일간 보존됩니다. 보존 제한이 만료되면 백업이 자동으로 삭제됩니다.

  최대 백업 보존 제한은 35일입니다. 백업 보존 제한을 0으로 설정하면 캐시의 자동 백업이 비활성화됩니다.

자동 백업을 예약하면 ElastiCache 가 백업 생성을 시작합니다. 가장 편리한 시간에 백업 기간을 설정할 수 있습니다. 백업 기간을 지정하지 않으면 가 백업 기간을 자동으로 ElastiCache 할당합니다.

ElastiCache 콘솔, 또는 를 사용하여 새 캐시를 생성하거나 기존 캐시를 업데이트할 때 자동 백업을 활성화 AWS CLI하거나 비활성화할 수 있습니다 ElastiCache API. Valkey 및 Redis 의 경우 고급 Valkey 설정 또는 고급 Redis OSS 설정 섹션에서 자동 백업 활성화 상자를 선택하면 OSS됩니다. Memcached의 경우 고급 Memcached 설정 섹션에서 자동 백업 활성화 상자를 선택하면 됩니다.

수동 백업 지원

자동 백업 외에도 언제든지 수동 백업을 만들 수 있습니다. 지정한 보존 기간 후에 자동으로 삭제되는 자동 백업과 달리 수동 백업에는 나중에 자동으로 삭제되는 보존 기간이 없습니다. 캐시를 삭제하더라도 해당 캐시의 모든 수동 백업은 보존됩니다. 수동 백업을 더 이상 보존하지 않으려면 이 백업을 직접 명시적으로 삭제해야 합니다.

수동 백업을 직접 생성할 뿐 아니라 다음 방법 중 하나로 수동 백업을 생성할 수 있습니다.

- 백업 복사. 소스 백업을 자동으로 생성했는지 수동으로 생성했는지는 중요하지 않습니다.
- 최종 백업 생성. 클러스터나 노드를 삭제하기 직전에 백업을 생성합니다.

AWS Management Console, AWS CLI또는 를 사용하여 캐시의 수동 백업을 생성할 수 있습니다 ElastiCache API.

수동 백업 생성(콘솔)

캐시의 백업을 생성하려면 다음과 같이 하세요(콘솔).

1. 에 로그인 AWS Management Console 하고 에서 Amazon EC2 콘솔을 엽니다https://console.aws.amazon.com/ec2/.
2. 탐색 창에서 기본 설정에 따라 Valkey 캐시 , Redis OSS 캐시 또는 Memcached 캐시 를 선택합니다.
3. 백업하려는 캐시 이름 왼쪽의 상자를 선택합니다.
4. [Backup]을 선택합니다.
5. [Create Backup] 대화 상자의 [Backup Name] 상자에 백업 이름을 입력합니다. 이름은 백업된 클러스터와 백업 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. [Create Backup]을 선택합니다.

   클러스터 상태가 snapshotting으로 바뀝니다.

수동 백업 생성(AWS CLI)

를 사용하여 서버리스 캐시 수동 백업 AWS CLI

를 사용하여 캐시의 수동 백업을 생성하려면 다음 파라미터로 `create-serverless-snapshot` AWS CLI 작업을 AWS CLI사용합니다.

- `--serverless-cache-name` - 백업하는 서버리스 캐시 이름입니다.
- `--serverless-cache-snapshot-name` - 생성할 스냅샷의 이름입니다.

Linux, macOS, Unix의 경우:

```
aws elasticache create-serverless-snapshot \
                        --serverless-cache-name CacheName \
                        --serverless-cache-snapshot-name bkup-20231127
```

Windows의 경우:

```
aws elasticache create-serverless-snapshot ^
    --serverless-cache-name CacheName ^
    --serverless-cache-snapshot-name bkup-20231127
```

를 사용하여 자체 설계된 클러스터의 수동 백업 AWS CLI

를 사용하여 자체 설계된 클러스터의 수동 백업을 생성하려면 다음 파라미터로 `create-snapshot` AWS CLI 작업을 AWS CLI사용합니다.

- `--cache-cluster-id`
  - 백업할 클러스터에 복제본 노드가 없는 경우 `--cache-cluster-id`는 백업할 클러스터의 이름입니다. 예를 들어 *`mycluster`*.
  - 백업 중인 클러스터에 복제본 노드가 하나 이상 있으면 `--cache-cluster-id`는 백업에 사용하려는 클러스터의 노드 이름입니다. 예를 들어 이름은 *`mycluster-002`*.

  Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터를 백업할 때만 이 파라미터를 사용합니다.

- `--replication-group-id` – 백업의 소스로 사용할 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터(CLI/API: 복제 그룹)의 이름입니다. Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터를 백업할 때 이 파라미터를 사용합니다.

- `--snapshot-name` - 생성할 스냅샷의 이름입니다.

  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.

예제 1: 복제본 노드가 없는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터 백업

다음 AWS CLI 작업은 읽기 전용 복제본`myNonClusteredRedis`이 없는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터`bkup-20150515`에서 백업을 생성합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache create-snapshot \
    --cache-cluster-id myNonClusteredRedis \
    --snapshot-name bkup-20150515
```

Windows의 경우:

```
aws elasticache create-snapshot ^
    --cache-cluster-id myNonClusteredRedis ^
    --snapshot-name bkup-20150515
```

예제 2: 복제본 노드를 사용하여 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터 백업

다음 AWS CLI 작업은 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터 `bkup-20150515`에서 백업을 생성합니다`myNonClusteredRedis`. 이 백업에는 하나 이상의 읽기 전용 복제본이 있습니다.

Linux, macOS, Unix의 경우:

```
aws elasticache create-snapshot \
```

```
    --cache-cluster-id myNonClusteredRedis-001 \
    --snapshot-name bkup-20150515
```

Windows의 경우:

```
aws elasticache create-snapshot ^
    --cache-cluster-id myNonClusteredRedis-001 ^
    --snapshot-name bkup-20150515
```

예제 출력: 복제본 노드를 사용하여 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터 백업

이 작업의 출력은 다음과 같습니다.

```
{
    "Snapshot": {
        "Engine": "redis",
        "CacheParameterGroupName": "default.redis6.x",
        "VpcId": "vpc-91280df6",
        "CacheClusterId": "myNonClusteredRedis-001",
        "SnapshotRetentionLimit": 0,
        "NumCacheNodes": 1,
        "SnapshotName": "bkup-20150515",
        "CacheClusterCreateTime": "2017-01-12T18:59:48.048Z",
        "AutoMinorVersionUpgrade": true,
        "PreferredAvailabilityZone": "us-east-1c",
        "SnapshotStatus": "creating",
        "SnapshotSource": "manual",
        "SnapshotWindow": "08:30-09:30",
        "EngineVersion": "6.0",
        "NodeSnapshots": [
            {
                "CacheSize": "",
                "CacheNodeId": "0001",
                "CacheNodeCreateTime": "2017-01-12T18:59:48.048Z"
            }
        ],
        "CacheSubnetGroupName": "default",
        "Port": 6379,
        "PreferredMaintenanceWindow": "wed:07:30-wed:08:30",
        "CacheNodeType": "cache.m3.2xlarge",
        "DataTiering": "disabled"
    }
}
```

예제 3: Valkey 또는 Redis용 클러스터 백업OSS(클러스터 모드 활성화됨)

다음 AWS CLI 작업은 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 bkup-20150515에서 백업을 생성합니다myClusteredRedis. --replication-group-id 대신 --cache-cluster-id를 사용하여 원본을 식별하세요.

Linux, macOS, Unix의 경우:

```
aws elasticache create-snapshot \
    --replication-group-id myClusteredRedis \
    --snapshot-name bkup-20150515
```

Windows의 경우:

```
aws elasticache create-snapshot ^
    --replication-group-id myClusteredRedis ^
    --snapshot-name bkup-20150515
```

예제 출력: Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 백업

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
    "Snapshot": {
        "Engine": "redis",
        "CacheParameterGroupName": "default.redis6.x.cluster.on",
        "VpcId": "vpc-91280df6",
        "NodeSnapshots": [
            {
                "CacheSize": "",
                "NodeGroupId": "0001"
            },
            {
                "CacheSize": "",
                "NodeGroupId": "0002"
            }
        ],
        "NumNodeGroups": 2,
        "SnapshotName": "bkup-20150515",
        "ReplicationGroupId": "myClusteredRedis",
        "AutoMinorVersionUpgrade": true,
        "SnapshotRetentionLimit": 1,
```

```
        "AutomaticFailover": "enabled",
        "SnapshotStatus": "creating",
        "SnapshotSource": "manual",
        "SnapshotWindow": "10:00-11:00",
        "EngineVersion": "6.0",
        "CacheSubnetGroupName": "default",
        "ReplicationGroupDescription": "2 shards 2 nodes each",
        "Port": 6379,
        "PreferredMaintenanceWindow": "sat:03:30-sat:04:30",
        "CacheNodeType": "cache.r3.large",
        "DataTiering": "disabled"
    }
}
```

관련 주제

자세한 내용은 AWS CLI 명령 참조의 create-snapshot을 참조하세요.

최종 백업 생성

ElastiCache 콘솔, AWS CLI또는 를 사용하여 최종 백업을 생성할 수 있습니다 ElastiCache API.

최종 백업 생성(콘솔)

ElastiCache 콘솔을 사용하여 Valkey 또는 Redis OSS 서버리스 캐시, Valkey 또는 Redis OSS 자체 설계 클러스터 또는 Memcached 서버리스 캐시를 삭제할 때 최종 백업을 생성할 수 있습니다.

캐시를 삭제할 때 최종 백업을 생성하려면 삭제 대화 상자에서 백업 생성에서 예를 선택하고 백업에 이름을 지정합니다.

관련 주제

- 사용 AWS Management Console
- 복제 그룹 삭제(콘솔)

최종 백업 생성(AWS CLI)

를 사용하여 캐시를 삭제할 때 최종 백업을 생성할 수 있습니다 AWS CLI.

주제

- Valkey 캐시, Redis OSS 캐시 또는 Memcached 서버리스 캐시를 삭제할 때
- 읽기 전용 복제본이 없는 Valkey 또는 Redis OSS 클러스터를 삭제하는 경우
- 읽기 전용 복제본을 사용하여 Valkey 또는 Redis OSS 클러스터를 삭제하는 경우

Valkey 캐시, Redis OSS 캐시 또는 Memcached 서버리스 캐시를 삭제할 때

최종 백업을 생성하려면 다음 파라미터로 `delete-serverless-cache` AWS CLI 작업을 사용합니다.

- `--serverless-cache-name` - 삭제 중인 캐시 이름입니다.
- `--final-snapshot-name` - 백업 이름입니다.

다음 코드는 캐시 `myserverlesscache`를 삭제할 때 최종 백업 `bkup-20231127-final`을 생성합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache delete-serverless-cache \
        --serverless-cache-name myserverlesscache \
        --final-snapshot-name bkup-20231127-final
```

Windows의 경우:

```
aws elasticache delete-serverless-cache ^
        --serverless-cache-name myserverlesscache ^
        --final-snapshot-name bkup-20231127-final
```

자세한 내용은 명령 참조delete-serverless-cache의 섹션을 참조하세요. AWS CLI

읽기 전용 복제본이 없는 Valkey 또는 Redis OSS 클러스터를 삭제하는 경우

읽기 전용 복제본이 없는 자체 설계 클러스터에 대한 최종 백업을 생성하려면 다음 파라미터와 함께 `delete-cache-cluster` AWS CLI 작업을 사용합니다.

- `--cache-cluster-id` - 삭제 중인 클러스터의 이름입니다.
- `--final-snapshot-identifier` - 백업 이름입니다.

다음 코드는 클러스터 `myRedisCluster`를 삭제할 때 최종 백업 `bkup-20150515-final`을 생성합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache delete-cache-cluster \
        --cache-cluster-id myRedisCluster \
        --final-snapshot-identifier bkup-20150515-final
```

Windows의 경우:

```
aws elasticache delete-cache-cluster ^
        --cache-cluster-id myRedisCluster ^
        --final-snapshot-identifier bkup-20150515-final
```

자세한 내용은 명령 참조delete-cache-cluster의 섹션을 참조하세요. AWS CLI

읽기 전용 복제본을 사용하여 Valkey 또는 Redis OSS 클러스터를 삭제하는 경우

복제 그룹을 삭제할 때 최종 백업을 생성하려면 다음 파라미터와 함께 `delete-replication-group` AWS CLI 작업을 사용합니다.

- `--replication-group-id` - 삭제 중인 복제 그룹의 이름입니다.
- `--final-snapshot-identifier` - 최종 백업 이름입니다.

다음 코드는 복제 그룹 `myReplGroup`을 삭제할 때 최종 백업 `bkup-20150515-final`을 만듭니다.

Linux, macOS, Unix의 경우:

```
aws elasticache delete-replication-group \
        --replication-group-id myReplGroup \
        --final-snapshot-identifier bkup-20150515-final
```

Windows의 경우:

```
aws elasticache delete-replication-group ^
        --replication-group-id myReplGroup ^
        --final-snapshot-identifier bkup-20150515-final
```

자세한 내용은 명령 참조delete-replication-group의 섹션을 참조하세요. AWS CLI

백업 설명

다음 절차는 백업 목록을 표시하는 방법을 보여줍니다. 원한다면 특정 백업의 세부 정보를 볼 수도 있습니다.

백업 설명(콘솔)

를 사용하여 백업을 표시하려면 AWS Management Console

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 [Backups]를 선택합니다.
3. 특정 백업의 세부 정보를 보려면 백업 이름 왼쪽의 상자를 선택합니다.

서버리스 백업 설명(AWS CLI)

서버리스 백업 목록과 선택적으로 특정 백업에 대한 세부 정보를 표시하려면 `describe-serverless-cache-snapshots` CLI 작업을 사용합니다.

예제

다음 작업에서는 `--max-records` 파라미터를 사용하여 계정에 연결된 백업을 20개까지 나열합니다. `--max-records` 파라미터를 생략하면 백업이 50개까지 나열됩니다.

```
aws elasticache describe-serverless-cache-snapshots --max-records 20
```

다음 작업에서는 `--serverless-cache-name` 파라미터를 사용하여 캐시 `my-cache`에 연결된 백업만 나열합니다.

```
aws elasticache describe-serverless-cache-snapshots --serverless-cache-name my-cache
```

다음 작업에서는 `--serverless-cache-snapshot-name` 파라미터를 사용하여 백업 `my-backup`의 세부 정보를 표시합니다.

```
aws elasticache describe-serverless-cache-snapshots --serverless-cache-snapshot-
name my-backup
```

자세한 내용은 AWS CLI 명령 참조describe-serverless-cache-snapshots의 섹션을 참조하세요.

자체 설계된 클러스터 백업 설명(AWS CLI)

자체 설계된 클러스터 백업 목록과 선택적으로 특정 백업에 대한 세부 정보를 표시하려면 `describe-snapshots` CLI 작업을 사용합니다.

예제

다음 작업에서는 `--max-records` 파라미터를 사용하여 계정에 연결된 백업을 20개까지 나열합니다. `--max-records` 파라미터를 생략하면 백업이 50개까지 나열됩니다.

```
aws elasticache describe-snapshots --max-records 20
```

다음 작업에서는 `--cache-cluster-id` 파라미터를 사용하여 클러스터 `my-cluster`에 연결된 백업만 나열합니다.

```
aws elasticache describe-snapshots --cache-cluster-id my-cluster
```

다음 작업에서는 `--snapshot-name` 파라미터를 사용하여 백업 `my-backup`의 세부 정보를 표시합니다.

```
aws elasticache describe-snapshots --snapshot-name my-backup
```

자세한 내용은 AWS CLI 명령 참조의 describe-snapshots를 참조하세요.

백업 복사

자동 또는 수동으로 백업을 생성했을 때 모든 백업 복사본을 생성할 수 있습니다. 외부에서 액세스할 수 있도록 백업을 내보낼 수도 있습니다 ElastiCache. 백업을 내보내기 위한 지침은 백업 내보내기 섹션을 참조하세요.

다음 단계에서는 백업을 복사하는 방법을 보여 줍니다.

백업 복사(콘솔)

백업을 복사하려면(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 백업 목록을 표시하려면 왼쪽 탐색 창에서 [Backups]를 선택합니다.
3. 백업 목록에서 복사할 백업의 이름 왼쪽에 있는 상자를 선택합니다.
4. 작업, 복사 순으로 선택합니다.
5. [New backup name] 상자에 새 백업의 이름을 입력합니다.
6. 복사를 선택합니다.

서버리스 백업 복사(AWS CLI)

서버리스 캐시의 백업을 복사하려면 `copy-serverless-cache-snapshot` 작업을 사용합니다.

파라미터

- `--source-serverless-cache-snapshot-name` - 복사할 백업의 이름입니다.
- `--target-serverless-cache-snapshot-name` - 백업 복사본의 이름입니다.

다음 예제에서는 자동 백업 복사본을 만듭니다.

Linux, macOS, Unix의 경우:

```
aws elasticache copy-serverless-cache-snapshot \
    --source-serverless-cache-snapshot-name automatic.my-cache-2023-11-27-03-15 \
    --target-serverless-cache-snapshot-name my-backup-copy
```

Windows의 경우:

```
aws elasticache copy-serverless-cache-snapshot ^
    --source-serverless-cache-snapshot-name automatic.my-cache-2023-11-27-03-15 ^
    --target-serverless-cache-snapshot-name my-backup-copy
```

자세한 내용은 *copy-serverless-cache-snapshot*의 AWS CLI 섹션을 참조하세요.

자체 설계된 클러스터 백업 복사(AWS CLI)

자체 설계된 클러스터의 백업을 복사하려면 copy-snapshot 작업을 사용합니다.

파라미터

- `--source-snapshot-name` - 복사할 백업의 이름입니다.
- `--target-snapshot-name` - 백업 복사본의 이름입니다.
- `--target-bucket` - 백업을 내보내기 위해 예약됩니다. 백업 복사본을 만들 때 이 파라미터를 사용하지 마십시오. 자세한 내용은 백업 내보내기 단원을 참조하십시오.

다음 예제에서는 자동 백업 복사본을 만듭니다.

Linux, macOS, Unix의 경우:

```
aws elasticache copy-snapshot  \
    --source-snapshot-name automatic.my-redis-primary-2014-03-27-03-15 \
    --target-snapshot-name my-backup-copy
```

Windows의 경우:

```
aws elasticache copy-snapshot ^
    --source-snapshot-name automatic.my-redis-primary-2014-03-27-03-15 ^
    --target-snapshot-name my-backup-copy
```

자세한 내용은 *copy-snapshot*의 AWS CLI 섹션을 참조하세요.

백업 내보내기

Amazon ElastiCache 은 ElastiCache (Redis OSS) 백업을 Amazon Simple Storage Service(Amazon S3) 버킷으로 내보내는 것을 지원하므로 외부에서 백업에 액세스할 수 있습니다 ElastiCache. ElastiCache 콘솔, AWS CLI또는 를 사용하여 백업을 내보낼 수 있습니다 ElastiCache API.

다른 AWS 리전에서 클러스터를 시작해야 하는 경우 백업을 내보내는 것이 도움이 될 수 있습니다. 데이터를 한 AWS 리전으로 내보내고 .rdb 파일을 새 AWS 리전으로 복사한 다음 해당 .rdb 파일을 사용하여 새 클러스터가 사용을 통해 채워질 때까지 기다리는 대신 새 캐시를 시드할 수 있습니다. 새 클러스터 시드에 대한 자세한 내용은 자습서: 외부에서 생성된 백업을 사용하여 자체 설계된 새 클러스터 검색 섹션을 참조하세요. 캐시의 데이터를 내보내려는 또 다른 이유는 오프라인 처리에 .rdb 파일을 사용하기 때문입니다.

> **Important**
>
> - 복사하려는 ElastiCache 백업과 Amazon S3 버킷은 동일한 AWS 리전에 있어야 합니다.
>
> Amazon S3 버킷으로 복사된 백업이 암호화되긴 하지만 백업을 저장하려는 Amazon S3 버킷에 대한 액세스 권한을 다른 사람에게 부여하지 않는 것이 좋습니다.
> - 데이터 계층화를 사용하는 클러스터는 Amazon S3로 백업 내보내기가 지원되지 않습니다. 자세한 내용은 의 데이터 계층화 ElastiCache 단원을 참조하십시오.
> - 백업 내보내기는 Valkey 및 Redis OSS 자체 설계 클러스터, Serverless Valkey 및 RedisOSS, Serverless Memcached에 사용할 수 있습니다. 자체 설계된 Memcached 클러스터에는 백업 내보내기를 사용할 수 없습니다.

백업을 Amazon S3 버킷으로 내보내려면 먼저 백업과 동일한 AWS 리전에 Amazon S3 버킷이 있어야 합니다. 버킷에 대한 ElastiCache 액세스 권한을 부여합니다. 처음 두 단계에서는 이 작업을 수행할 방법을 보여줍니다.

Amazon S3 버킷 생성

다음 단계에서는 Amazon S3 콘솔을 사용하여 ElastiCache 백업을 내보내고 저장할 Amazon S3 버킷을 생성합니다.

Amazon S3 버킷을 생성하려면

1. 에 로그인 AWS Management Console 하고 에서 Amazon S3 콘솔을 엽니다https://console.aws.amazon.com/s3/.

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.

      Amazon S3 버킷의 이름은 DNS규정을 준수해야 합니다. 그렇지 않으면 ElastiCache 는 백업 파일에 액세스할 수 없습니다. DNS 규정 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.

   b. 리전 목록에서 Amazon S3 버킷의 AWS 리전을 선택합니다. 이 AWS 리전은 내보내려는 ElastiCache 백업과 동일한 AWS 리전이어야 합니다.

   c. 생성(Create)을 선택합니다.

Amazon S3 버킷 생성에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 버킷 생성을 참조하세요.

Amazon S3 버킷에 대한 ElastiCache 액세스 권한 부여

스냅샷을 Amazon S3 버킷에 복사 ElastiCache 하려면 버킷 정책을 업데이트하여 버킷에 대한 ElastiCache 액세스 권한을 부여해야 합니다.

> **Warning**
>
> Amazon S3 버킷으로 복사된 백업이 암호화되더라도 Amazon S3 버킷에 대한 액세스 권한이 있는 사람이 데이터에 액세스할 수 있습니다. 따라서 이 Amazon S3 버킷에 대한 무단 액세스를 방지하기 위한 IAM 정책을 설정하는 것이 좋습니다. 자세한 정보는 Amazon S3 사용 설명서의 액세스 관리를 참조하세요.

Amazon S3 버킷에 대한 적절한 권한을 생성하려면 다음 단계를 수행합니다.

S3 버킷에 대한 ElastiCache 액세스 권한을 부여하려면

1. 에 로그인 AWS Management Console 하고 에서 Amazon S3 콘솔을 엽니다https://console.aws.amazon.com/s3/.
2. 백업을 복사할 Amazon S3 버킷 이름을 선택합니다. Amazon S3 버킷 생성에서 생성한 S3 버킷이어야 합니다.
3. 권한 탭을 선택하고 권한 아래에서 액세스 제어 목록(ACL)을 선택한 다음 편집을 선택합니다.
4. 다음 옵션을 사용하여 피부여자 정식 ID `540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353`을 추가합니다.

   - 객체: 나열, 쓰기(Objects: List, Write)
   - 버킷ACL: 읽기, 쓰기

   > **Note**
   > - PDT GovCloud 리전의 경우 표준 ID는 입니다 `40fa568277ad703bd160f66ae4f83fc9dfdfd06c2f1b5060ca22442ac3ef8be6`.
   > - OSU GovCloud 리전의 경우 표준 ID는 입니다 `c54286759d2a83da9c480405349819c993557275cf37d820d514b42da6893f5c`.

5. 저장(Save)을 선택합니다.

ElastiCache 백업 내보내기

이제 S3 버킷을 생성하고 액세스할 수 있는 ElastiCache 권한을 부여했습니다. 다음으로 ElastiCache 콘솔, AWS CLI또는 ElastiCache API 를 사용하여 스냅샷을 내보낼 수 있습니다. 다음 예제에서는 호출자의 IAM ID에 다음과 같은 추가 S3 특정 IAM 권한이 있다고 가정합니다.

```
{
 "Version": "2012-10-17",
 "Statement": [{
  "Effect": "Allow",
  "Action": [
   "s3:GetBucketLocation",
   "s3:ListAllMyBuckets",
   "s3:PutObject",
   "s3:GetObject",
```

```
   "s3:DeleteObject",
   "s3:ListBucket"
  ],
  "Resource": "arn:aws:s3:::*"
 }]
}
```

옵트인 리전의 경우, 업데이트된 S3 버킷 정책은 다음 예와 유사합니다. (이 예에서는 아시아 태평양 (홍콩) 리전을 사용합니다.)

```
{
    "Version": "2012-10-17",
    "Id": "Policy15397346",
    "Statement": [
        {
            "Sid": "Stmt15399483",
            "Effect": "Allow",
            "Principal": {
                "Service": "elasticache.amazonaws.com"
            },
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::hkg-elasticache-backup",
                "arn:aws:s3:::hkg-elasticache-backup/*"
            ]
        },
        {
            "Sid": "Stmt15399484",
            "Effect": "Allow",

            "Principal": {
                "Service": "ap-east-1.elasticache-snapshot.amazonaws.com"
            },
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::hkg-elasticache-backup",
                "arn:aws:s3:::hkg-elasticache-backup/*"
            ]
        }
    ]
}
```

ElastiCache 백업 내보내기(콘솔)

다음 단계에서는 외부에서 액세스할 수 있도록 ElastiCache 콘솔을 사용하여 Amazon S3 버킷으로 백업을 내보냅니다 ElastiCache. Amazon S3 버킷은 ElastiCache 백업과 동일한 AWS 리전에 있어야 합니다.

Amazon S3 버킷으로 ElastiCache 백업을 내보내려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 백업 목록을 표시하려면 왼쪽 탐색 창에서 [Backups]를 선택합니다.
3. 백업 목록에서 내보낼 백업의 이름 왼쪽에 있는 상자를 선택합니다.
4. 복사를 선택합니다.
5. 백업 복사본을 생성에서 다음 작업을 수행하세요.

   a. [New backup name] 상자에 새 백업의 이름을 입력합니다.

      이름은 1~1,000자여야 하며 UTF-8 인코딩이 가능합니다.

      ElastiCache 는 여기에 입력하는 값에 인스턴스 식별자 및 `.rdb`를 추가합니다. 예를 들어 `my-exported-backup`을 입력하면 ElastiCache 에서 `my-exported-backup-0001.rdb`를 생성합니다.

   b. 대상 S3 위치 목록에서 백업을 복사할 Amazon S3 버킷(Amazon S3 버킷 생성에서 생성한 버킷)을 선택합니다.

      내보내기 프로세스가 성공하려면 대상 S3 위치가 백업 AWS 리전의 Amazon S3 버킷이어야 합니다.

      - 객체 액세스 - 읽기 및 쓰기.
      - 권한 액세스 - 읽기.

      자세한 내용은 Amazon S3 버킷에 대한 ElastiCache 액세스 권한 부여 단원을 참조하십시오.

   c. 복사를 선택합니다.

> **Note**
>
> S3 버킷에 백업을 내보내는 ElastiCache 데 필요한 권한이 없는 경우 다음 오류 메시지 중 하나를 받게 됩니다. Amazon S3 버킷에 대한 ElastiCache 액세스 권한 부여로 돌아가 지정한 권한을 추가하고 백업 내보내기를 다시 시도하세요.
>
> - ElastiCache 에 S3 버킷에 대한 %s READ 권한이 부여되지 않았습니다.
>
> 해결 방법: 버킷에 대한 읽기 권한을 추가합니다.
> - ElastiCache 에 S3 버킷에 대한 %s WRITE 권한이 부여되지 않았습니다.
>
> 해결 방법: 버킷에 대한 쓰기 권한을 추가합니다.
> - ElastiCache S3 버킷에 READ_ACP 권한 %s가 부여되지 않았습니다.
>
> 해결 방법: 버킷에 대한 권한 액세스로 [Read]를 추가합니다.

백업을 다른 AWS 리전에 복사하려면 Amazon S3를 사용하여 복사합니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 객체 복사를 참조하세요.

ElastiCache 서버리스 백업 내보내기(AWS CLI)

서버리스 캐시 백업 내보내기

다음 파라미터로 `export-serverless-cache-snapshot` CLI 작업을 사용하여 Amazon S3 버킷으로 백업을 내보냅니다.

파라미터

- `--serverless-cache-snapshot-name` - 복사할 백업의 이름입니다.
- `--s3-bucket-name` - 백업을 내보낼 Amazon S3 버킷의 이름입니다. 지정한 버킷에 백업 복사본이 생성됩니다.

  내보내기 프로세스가 성공하려면 가 다음 권한이 AWS 있는 백업 리전의 Amazon S3 버킷이어야 `--s3-bucket-name` 합니다.
  - 객체 액세스 - 읽기 및 쓰기.
  - 권한 액세스 - 읽기.

다음 작업은 my-s3-bucket에 백업을 복사합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache export-serverless-cache-snapshot \
    --serverless-cache-snapshot-name automatic.my-redis-2023-11-27 \
    --s3-bucket-name my-s3-bucket
```

Windows의 경우:

```
aws elasticache export-serverless-cache-snapshot ^
    --serverless-cache-snapshot-name automatic.my-redis-2023-11-27 ^
    --s3-bucket-name my-s3-bucket
```

자체 설계된 ElastiCache 클러스터 백업 내보내기(AWS CLI)

자체 설계된 클러스터 백업 내보내기

다음 파라미터로 copy-snapshot CLI 작업을 사용하여 Amazon S3 버킷으로 백업을 내보냅니다.

파라미터

- --source-snapshot-name - 복사할 백업의 이름입니다.
- --target-snapshot-name - 백업 복사본의 이름입니다.

  이름은 1~1,000자여야 하며 UTF-8 인코딩이 가능합니다.

  ElastiCache 는 여기에 입력한 값에 인스턴스 식별자 및 .rdb를 추가합니다. 예를 들어 my-exported-backup을 입력하면 ElastiCache 에서 my-exported-backup-0001.rdb를 생성합니다.
- --target-bucket - 백업을 내보낼 Amazon S3 버킷의 이름입니다. 지정한 버킷에 백업 복사본이 생성됩니다.

  내보내기 프로세스가 성공하려면 가 다음 권한이 있는 백업 AWS 리전의 Amazon S3 버킷이어야 --target-bucket 합니다.
  - 객체 액세스 - 읽기 및 쓰기.
  - 권한 액세스 - 읽기.

  자세한 내용은 Amazon S3 버킷에 대한 ElastiCache 액세스 권한 부여 단원을 참조하십시오.

다음 작업은 my-s3-bucket에 백업을 복사합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache copy-snapshot \
    --source-snapshot-name automatic.my-redis-primary-2016-06-27-03-15 \
    --target-snapshot-name my-exported-backup \
    --target-bucket my-s3-bucket
```

Windows의 경우:

```
aws elasticache copy-snapshot ^
    --source-snapshot-name automatic.my-redis-primary-2016-06-27-03-15 ^
    --target-snapshot-name my-exported-backup ^
    --target-bucket my-s3-bucket
```

백업에서 새 캐시로 복원

Valkey의 기존 백업을 새 Valkey 캐시 또는 자체 설계된 클러스터로 복원하고 기존 Redis OSS 백업을 새 Redis OSS 캐시 또는 자체 설계된 클러스터로 복원할 수 있습니다. 기존 Memcached 서버리스 캐시 베이크업을 새 Memcached 서버리스 캐시로 복원할 수도 있습니다.

서버리스 캐시로 백업 복원(콘솔)

> **Note**
>
> ElastiCache Serverless는 Valkey 7.2 이상 및 5.0과 사용 가능한 최신 OSS 버전 사이의 Redis 버전과 호환되는 RDB 파일을 지원합니다.

서버리스 캐시로 백업을 복원하려면 다음과 같이 하세요(콘솔).

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 [Backups]를 선택합니다.
3. 백업 목록에서 복원할 백업의 이름 왼쪽에 있는 상자를 선택합니다.
4. 작업을 선택한 다음 복원을 선택합니다.
5. 새 서버리스 캐시의 이름과 설명(선택 사항)을 입력합니다.
6. 생성을 클릭하여 새 캐시를 생성하고 백업에서 데이터를 가져옵니다.

백업을 자체 설계된 클러스터로 복원(콘솔)

백업을 자체 설계된 클러스터로 복원하려면 다음과 같이 하세요(콘솔).

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 [Backups]를 선택합니다.
3. 백업 목록에서 복원할 백업의 이름 왼쪽에 있는 상자를 선택합니다.
4. 작업을 선택한 다음 복원을 선택합니다.
5. 자체 캐시 설계를 선택하고 노드 유형, 크기, 샤드 수, 복제본, AZ 배치, 보안 설정 등 클러스터 설정을 사용자 지정합니다.

6. 생성을 선택해 자체 설계된 캐시를 새로 생성하고 백업에서 데이터를 가져옵니다.

서버리스 캐시로 백업 복원(AWS CLI)

> **Note**
>
> ElastiCache Serverless는 Valkey 7.2 이상 및 5.0과 사용 가능한 최신 OSS 버전 사이의 Redis 버전과 호환되는 RDB 파일을 지원합니다.

서버리스 캐시로 백업을 복원하려면 다음과 같이 하세요(AWS CLI).

다음 AWS CLI 예제에서는 를 사용하여 새 캐시를 생성하고 백업에서 데이터를 `create-serverless-cache` 가져옵니다.

Linux, macOS, Unix의 경우:

```
aws elasticache create-serverless-cache \

    --serverless-cache-name CacheName \
    --engine redis
    --snapshot-arns-to-restore Snapshot-ARN
```

Windows의 경우:

```
aws elasticache create-serverless-cache ^

    --serverless-cache-name CacheName ^
    --engine redis ^
    --snapshot-arns-to-restore Snapshot-ARN
```

백업을 자체 설계된 클러스터로 복원(AWS CLI)

백업을 자체 설계된 클러스터로 복원하려면 다음과 같이 하세요(AWS CLI).

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 백업을 두 가지 방법으로 복원할 수 있습니다.

-
```
aws elasticache create-serverless-cache \
    --serverless-cache-name CacheName \
```

```
    --engine redis
    --snapshot-arns-to-restore Snapshot-ARN
```

- Windows의 경우:

```
aws elasticache create-serverless-cache ^
    --serverless-cache-name CacheName ^
    --engine redis ^
    --snapshot-arns-to-restore Snapshot-ARN
```

백업을 자체 설계된 클러스터로 복원(AWS CLI)

백업을 자체 설계된 클러스터로 복원하려면 다음과 같이 하세요(AWS CLI).

Valkey 또는 Redis OSS 서버리스 캐시 백업을 복원할 수 있으며 Valkey 또는 Redis OSS 자체 설계 클러스터를 복원할 수도 있습니다.

Valkey 또는 Redis OSS 서버리스 캐시 백업을 두 가지 방법으로 복원할 수 있습니다.

- AWS CLI 작업을 사용하여 단일 노드 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터로 복원할 수 있습니다 create-cache-cluster.
- 읽기 전용 복제본(복제 그룹)을 사용하여 Valkey 또는 Redis OSS 클러스터로 복원할 수 있습니다. 이렇게 하려면 AWS CLI 작업 에서 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 또는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨)를 사용할 수 있습니다 create-replication-group. 이 경우 Valkey 또는 Redis OSS .rdb 파일로 복원을 시드합니다. 자체 설계된 새 클러스터를 시드하는 방법에 대한 자세한 내용은 자습서: 외부에서 생성된 백업을 사용하여 자체 설계된 새 클러스터 검색 섹션을 참조하세요.

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 백업을 두 가지 방법으로 복원할 수 있습니다.

- AWS CLI 작업을 사용하여 단일 노드 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터로 복원할 수 있습니다 create-cache-cluster.
- 읽기 전용 복제본(복제 그룹)을 사용하여 Valkey 또는 Redis OSS 클러스터로 복원할 수 있습니다. 이렇게 하려면 AWS CLI 작업 에서 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 또는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨)를 사용할 수 있습니다 create-replication-group. 이 경우 Valkey 또는 Redis OSS .rdb 파일로 복원을 시드합니다. 자체 설계된 새 클러스터를 시드하는 방법에 대한 자세한 내용은 자습서: 외부에서 생성된 백업을 사용하여 자체 설계된 새 클러스터 검색 섹션을 참조하세요.

`create-cache-cluster` 또는 `create-replication-group` 작업 중 하나를 사용할 때는 `--snapshot-name` 또는 `--snapshot-arn` 파라미터를 포함하여 백업의 데이터로 새 클러스터나 복제 그룹을 시드해야 합니다.

백업 삭제

보존 기간 제한이 만료되면 자동 백업이 자동으로 삭제됩니다. 클러스터를 삭제하면 모든 자동 백업도 삭제됩니다. 복제 그룹을 삭제하면 해당 그룹에 속한 클러스터의 모든 자동 백업도 삭제됩니다.

ElastiCache 는 백업이 자동으로 생성되었는지 아니면 수동으로 생성되었는지에 관계없이 언제든지 백업을 삭제할 수 있는 삭제 API 작업을 제공합니다. 수동 백업에는 보존 제한이 없으므로 수동 삭제를 통해서만 수동 백업을 제거할 수 있습니다.

ElastiCache 콘솔, AWS CLI또는 를 사용하여 백업을 삭제할 수 있습니다 ElastiCache API.

백업 삭제(콘솔)

다음 절차에서는 ElastiCache 콘솔을 사용하여 백업을 삭제합니다.

백업 삭제

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 백업을 선택합니다.

   백업 화면에 백업 목록이 나타납니다.
3. 삭제할 백업의 이름 왼쪽에 있는 상자를 선택합니다.
4. Delete(삭제)를 선택합니다.
5. 이 백업을 삭제하려면 [Delete Backup] 확인 화면에서 [Delete]를 선택합니다. 상태가 [deleting]으로 변경됩니다.

서버리스 백업 삭제(AWS CLI)

다음 파라미터와 함께 Delete-snapshot AWS CLI 작업을 사용하여 서버리스 백업을 삭제합니다.

- `--serverless-cache-snapshot-name` - 삭제할 백업의 이름입니다.

다음 코드는 `myBackup` 백업을 삭제합니다.

```
aws elasticache delete-serverless-cache-snapshot --serverless-cache-snapshot-
name myBackup
```

자세한 내용은 명령 참조delete-serverless-cache-snapshot의 섹션을 참조하세요. AWS CLI

자체 설계된 클러스터 백업 삭제(AWS CLI)

다음 파라미터와 함께 Delete-snapshot AWS CLI 작업을 사용하여 자체 설계된 클러스터 백업을 삭제합니다.

- `--snapshot-name` - 삭제할 백업의 이름입니다.

다음 코드는 myBackup 백업을 삭제합니다.

```
aws elasticache delete-snapshot --snapshot-name myBackup
```

자세한 내용은 AWS CLI 명령 참조의 delete-snapshot을 참조하세요.

백업 태그 지정

사용자 고유의 메타데이터를 태그 형태로 각 백업에 할당할 수 있습니다. 태그를 사용하면 용도, 소유자 또는 환경을 기준으로 하는 등 백업을 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 송장에 대한 비용을 태그 값으로 그룹화하여 여러 AWS 서비스에서 비용을 추적하는 수단입니다. 비용 할당 태그에 대해 자세히 알아보려면 비용 할당 태그 사용을 참조하세요.

ElastiCache 콘솔, AWS CLI또는 ElastiCache API 를 사용하여 백업에 비용 할당 태그를 추가, 나열, 수정, 제거 또는 복사할 수 있습니다. 자세한 내용은 비용 할당 태그를 사용하여 비용을 모니터링합니다. 단원을 참조하십시오.

자습서: 외부에서 생성된 백업을 사용하여 자체 설계된 새 클러스터 검색

새 Valkey 또는 Redis OSS 자체 설계 클러스터를 생성할 때 Valkey 또는 Redis OSS .rdb 백업 파일의 데이터와 함께 클러스터를 시드할 수 있습니다. 클러스터 시드는 현재 외부에서 Valkey 또는 Redis OSS 인스턴스를 관리하고 기존 Valkey 또는 Redis OSS 데이터로 새 ElastiCache (Redis OSS) 자체 설계 클러스터를 채우 ElastiCache 려는 경우에 유용합니다.

Amazon 내에서 생성된 Valkey 또는 Redis OSS 백업에서 새 Valkey 또는 Redis OSS 자체 설계 클러스터를 시드하려면 섹션을 ElastiCache참조하세요백업에서 새 캐시로 복원.

Valkey 또는 Redis OSS .rdb 파일을 사용하여 자체 설계된 새 클러스터를 시드하는 경우 다음을 수행할 수 있습니다.

- 파티셔닝되지 않은 클러스터에서 Redis OSS 버전 3.2.4를 실행하는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 자체 설계된 클러스터로 업그레이드합니다.
- 새 자체 설계된 클러스터에서 여러 샤드(API 및 에서 노드 그룹이라고 함CLI)를 지정합니다. 이 숫자는 백업 파일을 생성하는 데 사용된 자체 설계된 클러스터의 샤드 수와 다를 수 있습니다.
- 백업을 만든 클러스터에 사용된 것보다 크거나 작은 새로운 자체 설계된 클러스터의 다른 노드 유형을 지정합니다. 더 작은 노드 유형으로 확장하는 경우 새 노드 유형에 데이터 및 Valkey 또는 Redis OSS 오버헤드를 위한 충분한 메모리가 있는지 확인합니다. 자세한 내용은 Valkey 또는 Redis OSS 스냅샷을 생성하기에 충분한 메모리 확보 단원을 참조하십시오.
- 새 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 슬롯에 백업 파일을 생성하는 데 사용된 클러스터와 다르게 키를 배포합니다.

> **Note**
>
> Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터에서 생성된 .rdb 파일에서 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터를 시드할 수 없습니다.

> **Important**
>
> - Valkey 또는 Redis OSS 백업 데이터가 노드의 리소스를 초과하지 않도록 해야 합니다. 예를 들어 Valkey 또는 Redis OSS 데이터가 5GB인 .rdb 파일을 메모리가 2.9GB인 cache.m3.medium 노드에 업로드할 수 없습니다.

백업이 너무 크면 결과 클러스터의 상태는 `restore-failed`가 됩니다. 이 경우 클러스터를 삭제하고 다시 시작해야 합니다.

노드 유형 및 사양의 전체 목록은 OSS 노드 유형별 파라미터 재정의 및 Amazon ElastiCache 제품 기능 및 세부 정보 섹션을 참조하세요.

- Amazon S3 서버 측 암호화(SSE-S3)로만 Valkey 또는 Redis OSS .rdb 파일을 암호화할 수 있습니다. 자세한 내용은 서버 측 암호화를 사용하여 데이터 보호를 참조하세요.

다음은 외부에서 Valkey 또는 Redis ElastiCache 용 클러스터를 ElastiCache (Redis OSS)로 마이그레이션OSS하는 방법을 안내하는 주제를 찾을 수 있습니다.

로 마이그레이션 ElastiCache (Redis OSS)

- 1단계: Valkey 또는 Redis OSS 백업 생성
- 2단계: Amazon S3 버킷 및 폴더 생성
- 3단계: Amazon S3에 백업 업로드
- 4단계: .rdb 파일에 대한 ElastiCache 읽기 액세스 권한 부여

외부 서비스에서 ElastiCache (Redis OSS)로 마이그레이션.

- 1단계: Valkey 또는 Redis OSS 백업 생성
- 2단계: Amazon S3 버킷 및 폴더 생성
- 3단계: Amazon S3에 백업 업로드
- 4단계: .rdb 파일에 대한 ElastiCache 읽기 액세스 권한 부여

1단계: Valkey 또는 Redis OSS 백업 생성

Valkey 또는 Redis OSS 백업을 생성하여 ElastiCache (Redis OSS) 인스턴스를 시드하려면

1. 기존 Valkey 또는 Redis OSS 인스턴스에 연결합니다.
2. `BGSAVE` 또는 `SAVE` 작업을 실행하여 백업을 생성합니다. .rdb 파일의 위치를 메모합니다.

   `BGSAVE`는 비동기식이며 처리하는 동안 다른 클라이언트를 차단하지 않습니다. 자세한 내용은 Valkey 웹 BGSAVE 사이트의 섹션을 참조하세요.

SAVE는 동기식이며 마칠 때까지 다른 프로세스를 차단합니다. 자세한 내용은 Valkey 웹 SAVE 사이트의 섹션을 참조하세요.

백업 생성에 대한 자세한 내용은 Valkey 웹 사이트의 지속성을 참조하세요.

2단계: Amazon S3 버킷 및 폴더 생성

백업 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 백업 업로드 섹션으로 건너뛸 수 있습니다.

Amazon S3 버킷을 생성하려면

1. 에 로그인 AWS Management Console 하고 에서 Amazon S3 콘솔을 엽니다https://console.aws.amazon.com/s3/.
2. Amazon Simple Storage Service 사용 설명서에서 버킷 생성의 Amazon S3 버킷 생성 지침을 따릅니다.

   Amazon S3 버킷의 이름은 DNS규정을 준수해야 합니다. 그렇지 않으면 ElastiCache 는 백업 파일에 액세스할 수 없습니다. DNS 규정 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 ElastiCache (Redis OSS) 클러스터와 동일한 AWS 리전에서 Amazon S3 버킷을 생성해야 합니다. 이 접근 방식은 가 Amazon S3에서 .rdb 파일을 ElastiCache 읽을 때 가장 높은 데이터 전송 속도를 보장합니다.

> **Note**
>
> 데이터를 최대한 안전하게 유지하려면 Amazon S3 버킷에 대한 권한을 최대한 제한적으로 설정합니다. 동시에 권한은 버킷과 해당 콘텐츠를 새 Valkey 또는 Redis OSS 클러스터에 시드하는 데 사용할 수 있도록 허용해야 합니다.

Amazon S3 버킷에 폴더를 추가하려면

1. 에 로그인 AWS Management Console 하고 에서 Amazon S3 콘솔을 엽니다https://console.aws.amazon.com/s3/.
2. .rdb 파일을 업로드할 버킷 이름을 선택합니다.
3. 폴더 생성을 선택합니다.
4. 새 폴더의 이름을 입력합니다.
5. 저장(Save)을 선택합니다.

   버킷 이름과 폴더 이름을 모두 메모합니다.

3단계: Amazon S3에 백업 업로드

이제 1단계: Valkey 또는 Redis OSS 백업 생성에서 생성한 .rdb 파일을 업로드합니다. 2단계: Amazon S3 버킷 및 폴더 생성에서 생성한 Amazon S3 버킷과 폴더로 업로드합니다. 이 작업에 대한 자세한 내용은 버킷에 객체 추가를 참조하세요. 2단계와 3단계 사이에 생성된 폴더 이름을 선택합니다.

.rdb 파일을 Amazon S3 폴더에 업로드하려면

1. 에 로그인 AWS Management Console 하고 에서 Amazon S3 콘솔을 엽니다https://console.aws.amazon.com/s3/.
2. 2단계에서 만든 Amazon S3 버킷 이름을 선택합니다.
3. 2단계에서 만든 폴더 이름을 선택합니다.
4. 업로드를 선택합니다.
5. [Add Files]를 선택합니다.
6. 업로드할 파일을 찾아 선택합니다. 파일을 여러 개 선택하려면 Ctrl 키를 누른 상태로 각 파일 이름을 선택합니다.
7. Open을 선택합니다.

8. [Upload] 대화 상자에서 정확한 파일 이름이 표시되는지 확인하고 [Upload]를 선택합니다.

.rdb 파일에 대한 경로를 기록합니다. 예를 들어 버킷 이름이 `myBucket`이고 경로가 `myFolder/redis.rdb`이면 `myBucket/myFolder/redis.rdb`를 입력합니다. 이 백업의 데이터로 새 클러스터를 시드하려면 이 경로가 필요합니다.

자세한 내용은 Amazon Simple Storage Service 사용 설명서의 버킷 규제 및 제한을 참조하세요.

4단계: .rdb 파일에 대한 ElastiCache 읽기 액세스 권한 부여

이제 .rdb 백업 파일에 대한 ElastiCache 읽기 액세스 권한을 부여합니다. 버킷이 기본 AWS 리전 또는 옵트인 AWS 리전에 있는지에 따라 다른 방식으로 백업 파일에 대한 ElastiCache 액세스 권한을 부여합니다.

AWS 2019년 3월 20일 이전에 도입된 리전은 기본적으로 활성화됩니다. 이러한 AWS 리전에서 즉시 작업을 시작할 수 있습니다. 아시아 태평양(홍콩) 및 중동(바레인)과 같이 2019년 3월 20일 이후에 도입된 리전은 기본적으로 비활성 상태입니다. 사용하려면 먼저 AWS 일반 참조의 AWS 리전 관리에 설명된 대로 리전을 활성화하거나 옵트인해야 합니다.

AWS 리전에 따라 접근 방식을 선택합니다.

- 기본 리전의 경우 기본 리전의 .rdb 파일에 대한 ElastiCache 읽기 액세스 권한 부여의 절차를 사용합니다.
- 옵트인 리전의 경우 옵트인 리전의 .rdb 파일에 대한 ElastiCache 읽기 액세스 권한 부여의 절차를 사용합니다.

기본 리전의 .rdb 파일에 대한 ElastiCache 읽기 액세스 권한 부여

AWS 2019년 3월 20일 이전에 도입된 리전은 기본적으로 활성화됩니다. 이러한 AWS 리전에서 즉시 작업을 시작할 수 있습니다. 아시아 태평양(홍콩) 및 중동(바레인)과 같이 2019년 3월 20일 이후에 도입된 리전은 기본적으로 비활성 상태입니다. 사용하려면 먼저 AWS 일반 참조의 AWS 리전 관리에 설명된 대로 리전을 활성화하거나 옵트인해야 합니다.

기본적으로 활성화된 AWS 리전의 백업 파일에 대한 ElastiCache 읽기 액세스 권한을 부여하려면

1. 에 로그인 AWS Management Console 하고 에서 Amazon S3 콘솔을 엽니다https://console.aws.amazon.com/s3/.
2. .rdb 파일이 포함된 S3 버킷 이름을 선택합니다.

3. .rdb 파일이 포함된 폴더 이름을 선택합니다.
4. .rdb 백업 파일 이름을 선택합니다. 선택한 파일 이름은 페이지 맨 위의 탭 위에 나타납니다.
5. 권한을 선택합니다.
6. aws-scs-s3-readonly 또는 IDs 다음 목록의 정식 중 하나가 사용자로 나열되지 않은 경우 다음을 수행합니다.

   a. 다른 AWS 계정에 대한 액세스에서 권한 부여자 추가를 선택합니다.

   b. 상자에 다음과 같이 AWS 리전의 정식 ID를 추가합니다.

      - AWS GovCloud (미국 서부) 리전:

      ```
      40fa568277ad703bd160f66ae4f83fc9dfdfd06c2f1b5060ca22442ac3ef8be6
      ```

      > **Important**
      >
      > 의 Valkey 또는 Redis OSS 클러스터에 다운로드 AWS GovCloud (US) 하려면 의 S3 버킷에 백업이 있어야 합니다 AWS GovCloud (US).

      - AWS 기본적으로 활성화된 리전:

      ```
      540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353
      ```

   c. 다음에 대해 예를 선택하여 버킷에 대한 권한을 설정합니다.

      - 나열/쓰기 객체(List/write object)
      - 객체 읽기/쓰기 ACL 권한

   d. 저장(Save)을 선택합니다.

7. 개요를 선택한 다음 다운로드를 선택합니다.

옵트인 리전의 .rdb 파일에 대한 ElastiCache 읽기 액세스 권한 부여

AWS 2019년 3월 20일 이전에 도입된 리전은 기본적으로 활성화됩니다. 이러한 AWS 리전에서 즉시 작업을 시작할 수 있습니다. 아시아 태평양(홍콩) 및 중동(바레인)과 같이 2019년 3월 20일 이후에 도입된 리전은 기본적으로 비활성 상태입니다. 사용하려면 먼저 AWS 일반 참조의 AWS 리전 관리에 설명된 대로 리전을 활성화하거나 옵트인해야 합니다.

이제 .rdb 백업 파일에 대한 ElastiCache 읽기 액세스 권한을 부여합니다.

백업 파일에 대한 ElastiCache 읽기 액세스 권한을 부여하려면

1. 에 로그인 AWS Management Console 하고 에서 Amazon S3 콘솔을 엽니다https://console.aws.amazon.com/s3/.
2. .rdb 파일이 포함된 S3 버킷 이름을 선택합니다.
3. .rdb 파일이 포함된 폴더 이름을 선택합니다.
4. .rdb 백업 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 작업을 수행하는 데 ElastiCache 필요한 권한을 부여합니다.

   - `[ "Service" : "region-full-name.elasticache-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.

```
{
    "Version": "2012-10-17",
    "Id": "Policy15397346",
    "Statement": [
        {
            "Sid": "Stmt15399483",
            "Effect": "Allow",
            "Principal": {
                "Service": "ap-east-1.elasticache-snapshot.amazonaws.com"
            },
            "Action": [
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketAcl"
            ],
            "Resource": [
                "arn:aws:s3:::example-bucket",
                "arn:aws:s3:::example-bucket/backup1.rdb",
```

```
                "arn:aws:s3:::example-bucket/backup2.rdb"
            ]
        }
    ]
}
```

8. Save changes(변경 사항 저장)를 선택합니다.

.rdb 파일 데이터를 사용하여 ElastiCache 클러스터를 시드했습니다.

이제 ElastiCache 클러스터를 생성하고 .rdb 파일의 데이터와 함께 클러스터를 시드할 준비가 되었습니다. 클러스터를 생성하려면 Valkey 또는 Redis용 클러스터 생성 OSS 또는 처음부터 Valkey 또는 Redis OSS 복제 그룹 생성의 지시를 따릅니다. 클러스터 엔진OSS으로 Valkey 또는 Redis를 선택해야 합니다.

Amazon S3에 업로드한 백업을 어디에서 찾을 ElastiCache 수 있는지 확인하는 데 사용하는 방법은 클러스터를 생성하는 데 사용하는 방법에 따라 달라집니다.

ElastiCache (Redis OSS) 클러스터 또는 복제 그룹을 .rdb 파일 데이터로 시드했습니다.

- ElastiCache 콘솔 사용

  클러스터 설정을 선택할 때 클러스터 생성 방법으로 백업에서 복원을 선택한 다음, 백업 소스 섹션에서 소스로 기타 백업을 선택합니다. 시드 RDB 파일 S3 위치 상자에 파일(들)의 Amazon S3 경로를 입력합니다. .rdb 파일이 여러 개 있으면 쉼표로 구분된 목록에 각 파일의 경로를 입력합니다. Amazon S3 경로는 *myBucket*/*myFolder*/*myBackupFilename*.rdb처럼 표시될 수 있습니다.

- 사용 AWS CLI

  `create-cache-cluster` 또는 `create-replication-group` 작업을 사용하는 경우 파라미터를 사용하여 각 .rdb 파일에 ARN 대해 정규화된 `--snapshot-arns`를 지정합니다. 예: `arn:aws:s3:::`*myBucket*`/`*myFolder*`/`*myBackupFilename*`.rdb`. 는 Amazon S3에 저장한 백업 파일로 확인ARN되어야 합니다.

- 사용 ElastiCache API

  `CreateCacheCluster` 또는 `CreateReplicationGroup` ElastiCache API 작업을 사용하는 경우 파라미터를 사용하여 각 .rdb 파일에 ARN 대해 정규화된 `SnapshotArns`을 지정합니다. 예: `arn:aws:s3:::`*myBucket*`/`*myFolder*`/`*myBackupFilename*`.rdb`. 는 Amazon S3에 저장한 백업 파일로 확인ARN되어야 합니다.

> **Important**
>
> Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터를 시드할 때는 새 클러스터 또는 복제 그룹에서 각 노드 그룹(샤드)을 구성해야 합니다. 파라미터`--node-group-configuration`(API: `NodeGroupConfiguration`)를 사용하여 이 작업을 수행합니다. 자세한 내용은 다음 자료를 참조하세요.
>
> - CLI AWS CLI 참조create-replication-group의 :
> - API참조CreateReplicationGroup의 ElastiCache API :

클러스터를 생성하는 과정에서 Valkey 또는 Redis OSS 백업의 데이터가 클러스터에 기록됩니다. ElastiCache 이벤트 메시지를 확인하여 진행 상황을 모니터링할 수 있습니다. 이렇게 하려면 ElastiCache 콘솔을 참조하고 캐시 이벤트 를 선택합니다. AWS ElastiCache 명령줄 인터페이스 또는 를 사용하여 이벤트 메시지를 ElastiCache API 가져올 수도 있습니다. 자세한 내용은 ElastiCache 이벤트 보기 단원을 참조하십시오.

의 엔진 버전 및 업그레이드 ElastiCache

이 섹션에서는 지원되는 Valkey, Redis OSS 및 Memcached 엔진과 업그레이드 방법을 다룹니다. Redis OSS 7.2에서 사용할 수 있는 모든 기능은 기본적으로 Valkey 7.2 이상에서 사용할 수 있습니다. Redis OSS 엔진이 ElastiCache 있는 일부 기존 에서 Valkey 엔진으로 업그레이드할 수도 있습니다.

주제

용 버전 관리 ElastiCache

Valkey, Redis 및 Memcached 엔진에 대해 업데이트된 ElastiCache 캐시 OSS 및 자체 설계된 클러스터를 업데이트하는 방법을 관리합니다.

ElastiCache 서버리스 캐시 버전 관리

ElastiCache 서버리스 캐시가 업그레이드되었는지 여부와 언제 업그레이드되는지 관리하고 사용자 고유의 조건과 일정에 따라 버전 업그레이드를 수행합니다.

ElastiCache Serverless는 애플리케이션에 미치는 영향이나 가동 중지 없이 최신 MINOR 및 PATCH 소프트웨어 버전을 캐시에 자동으로 적용합니다. 여러분은 아무 작업도 수행할 필요가 없습니다.

새 MAJOR 버전을 사용할 수 있게 되면 ElastiCache Serverless는 콘솔에서 알림을 보내고 에서 이벤트를 보냅니다 EventBridge. 콘솔, CLI또는 를 사용하여 캐시를 수정API하고 최신 엔진 버전을 선택하여 캐시를 최신 메이저 버전으로 업그레이드할 수 있습니다.

자체 설계된 ElastiCache 클러스터의 버전 관리

자체 설계된 ElastiCache 클러스터로 작업할 때 캐시 클러스터에 전원을 공급하는 소프트웨어가 에서 지원하는 새 버전으로 업그레이드되는 시기를 제어할 수 있습니다 ElastiCache . 캐시를 사용 가능한 최신 MAJOR, MINOR및 PATCH 버전으로 업그레이드할 시기를 제어할 수 있습니다. 클러스터 또는 복제 그룹을 수정하고 새 엔진 버전을 지정하여 엔진 버전 업그레이드를 시작합니다.

캐시 클러스터에 전원을 공급하는 프로토콜 준수 소프트웨어가 에서 지원하는 새 버전으로 업그레이드되는지 여부와 시기를 제어할 수 있습니다 ElastiCache. 이 제어 수준을 사용하면 특정 버전과의 호환성을 유지하고, 프로덕션에 배포하기 전에 애플리케이션으로 새 버전을 테스트하고, 원하는 조건과 일정에 맞춰 버전 업그레이드를 수행할 수 있습니다.

버전 업그레이드에는 약간의 호환성 위험이 있을 수 있으므로 업그레이드가 자동으로 이루어지지 않기 때문에 업그레이드는 사용자가 시작해야 합니다.

Valkey 및 Redis OSS 클러스터

> **Note**
>
> - Valkey 또는 Redis OSS 클러스터가 하나 이상의 리전에 복제되면 엔진 버전이 보조 리전에 대해 업그레이드된 다음 기본 리전에 대해 업그레이드됩니다.
> - ElastiCache (Redis OSS) 버전은 MAJOR 및 MINOR 구성 요소로 구성된 의미 버전으로 식별됩니다. 예를 들어 Redis OSS 6.2에서 메이저 버전은 6이고 마이너 버전은 2입니다. 자체 설계된 클러스터를 작동할 때 ElastiCache (Redis OSS)는 Redis OSS 6.2.1과 같은 PATCH 구성 요소도 노출하고 패치 버전은 1입니다.
>
> MAJOR 버전은 호환API되지 않는 변경 사항용이고 MINOR 버전은 이전 버전과 호환되는 방식으로 추가된 새 기능용입니다. PATCH 버전은 백워드 호환 버그 수정 및 비기능적 변경에 사용됩니다.

Valkey 및 Redis 를 사용하면 클러스터 또는 복제 그룹을 수정하고 새 엔진 버전을 지정하여 클러스터 또는 복제 그룹에 대한 엔진 버전 업그레이드를 OSS시작할 수 있습니다. 자세한 내용은 복제 그룹 수정 단원을 참조하십시오.

Memcached

Memcached를 사용하여 최신 버전으로 업그레이드하려면 캐시 클러스터를 수정하고 사용하려는 새 엔진 버전을 지정해야 합니다. 최신 Memcached 버전으로의 업그레이드는 안전하지 않은 프로세스로, 데이터가 손상되고 콜드 캐시로 시작합니다. 자세한 내용은 ElastiCache 클러스터 수정 단원을 참조하십시오.

이전 Memcached 버전을 Memcached 버전 1.4.33 이후로 업그레이드할 때 다음과 같은 요구 사항을 주의해야 합니다. 다음 조건에서는 `CreateCacheCluster` 및 `ModifyCacheCluster`에 실패합니다.

- `slab_chunk_max > max_item_size`의 경우.
- `max_item_size modulo slab_chunk_max != 0`의 경우.
- `max_item_size > ((max_cache_memory - memcached_connections_overhead) / 4)`의 경우.

  `(max_cache_memory - memcached_connections_overhead)` 값은 데이터에 사용할 수 있는 노드의 메모리입니다. 자세한 내용은 Memcached 연결 오버헤드 단원을 참조하십시오.

자체 설계된 클러스터를 사용할 때의 업그레이드 고려 사항

> **Note**
>
> 다음 고려 사항은 자체 설계된 클러스터를 업그레이드할 때만 적용됩니다. ElastiCache Serverless에는 적용되지 않습니다.

Valkey 및 Redis OSS 고려 사항

자체 설계된 Valkey 또는 Redis OSS 클러스터를 업그레이드할 때는 다음을 고려하세요.

- 엔진 버전 관리는 패치 발생 방법을 최대한 제어할 수 있도록 설계되었습니다. 그러나 는 시스템 또는 캐시 소프트웨어에서 심각한 보안 취약성이 발생할 가능성이 낮을 경우 클러스터를 패치할 수 있는 권한을 ElastiCache 보유합니다.
- Valkey 7.2 및 Redis OSS 6.0부터 ElastiCache 는 여러 패치 버전을 제공하는 대신 각 마이너 릴리스에 대해 단일 버전을 제공합니다.
- Redis OSS 엔진 버전 5.0.6부터는 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.
- 5.0.6 이전 버전으로 ElastiCache 클러스터를 업그레이드할 수도 있습니다. 관련된 프로세스는 동일하지만 DNS 전파 중에 장애 조치 시간이 길어질 수 있습니다(30초~1분).
- Redis OSS 7부터는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨)와 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 간의 전환을 ElastiCache 지원합니다.
- Amazon ElastiCache (Redis OSS) 엔진 업그레이드 프로세스는 기존 데이터를 유지하기 위해 최선의 노력을 기울일 수 있도록 설계되었으며 Redis OSS 복제를 성공적으로 수행해야 합니다.

- 엔진을 업그레이드할 때 는 기존 클라이언트 연결을 종료 ElastiCache 합니다. 엔진 업그레이드 중 가동 중지 시간을 최소화하려면 오류 재시도 및 지수 백오프가 있는 Redis OSS 클라이언트에 대한 모범 사례와 유지 관리 중 가동 중지 시간을 최소화하기 위한 모범 사례를 구현하는 것이 좋습니다.
- 엔진을 업그레이드할 때는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨)에서 Valkey 또는 RedisOSS(클러스터 모드 활성화됨)로 직접 업그레이드할 수 없습니다. 다음 절차에서는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨)에서 Valkey 또는 RedisOSS(클러스터 모드 활성화됨)로 업그레이드하는 방법을 보여줍니다.

  Valkey 또는 RedisOSS(클러스터 모드 비활성화됨)에서 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 엔진 버전으로 업그레이드하려면

  1. Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터 또는 복제 그룹을 백업합니다. 자세한 내용은 수동 백업 지원 단원을 참조하십시오.
  2. 백업을 사용하여 하나의 샤드OSS(노드 그룹)로 Valkey 또는 Redis(클러스터 모드 활성화됨) 클러스터를 생성하고 시드합니다. 클러스터 또는 복제 그룹을 생성할 때 새 엔진 버전을 지정하고 클러스터 모드를 활성화합니다. 자세한 내용은 자습서: 외부에서 생성된 백업을 사용하여 자체 설계된 새 클러스터 검색 단원을 참조하십시오.
  3. 이전 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터 또는 복제 그룹을 삭제합니다. 자세한 내용은 에서 클러스터 삭제 ElastiCache 또는 복제 그룹 삭제을 참조하세요.
  4. 새 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 또는 복제 그룹을 필요한 샤드(노드 그룹) 수로 조정합니다. 자세한 내용은 Valkey 또는 Redis에서 클러스터 크기 조정OSS(클러스터 모드 활성화됨) 단원을 참조하세요.
- 메이저 엔진 버전을 업그레이드하는 경우(예: 5.0.6에서 6.0으로 업그레이드) 새 엔진 버전과 호환되는 새 파라미터 그룹도 선택해야 합니다.
- 다중 AZ가 비활성화된 단일 Redis OSS 클러스터 및 클러스터의 경우 에 설명된 OSS 대로 Redis에 충분한 메모리를 제공하는 것이 좋습니다Valkey 또는 Redis OSS 스냅샷을 생성하기에 충분한 메모리 확보. 이러한 경우 업그레이드 프로세스 중에는 서비스 요청에 기본 항목을 사용할 수 없습니다.
- 다중 AZ가 활성화된 Redis OSS 클러스터의 경우 들어오는 쓰기 트래픽이 적은 기간 동안 엔진 업그레이드를 예약하는 것이 좋습니다. Redis OSS 5.0.6 이상으로 업그레이드하는 경우 업그레이드 프로세스 중에 기본 클러스터를 서비스 요청에 계속 사용할 수 있습니다.

  샤드가 여러 개인 클러스터 및 복제 그룹은 다음과 같이 처리되고 패치가 적용됩니다.

  - 모든 샤드는 병렬로 처리됩니다. 언제든지 하나의 샤드에서 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 기본 노드가 연속하여 처리됩니다. 한번에 오직 하나의 기본 노드가 업그레이드됩니다.
- 현재 클러스터 또는 복제 그룹에서 암호화가 활성화되어 있는 경우에는 암호화를 지원하지 않는 엔진 버전으로 업그레이드할 수 없습니다(예를 들면 3.2.6에서 3.2.10로 업그레이드 불가능).

Memcached 고려 사항

자체 설계된 Memcached 클러스터를 업그레이드할 때는 다음을 고려하세요.

- 엔진 버전 관리는 패치 발생 방법을 최대한 제어할 수 있도록 설계되었습니다. 그러나 는 시스템 또는 캐시 소프트웨어에서 심각한 보안 취약성이 발생할 가능성이 낮을 경우 클러스터를 패치할 수 있는 권한을 ElastiCache 보유합니다.
- Memcached 엔진은 지속성을 지원하지 않으므로 Memcached 엔진 버전 업그레이드는 항상 클러스터에서 모든 캐시 데이터를 지우는 방해가 되는 프로세스입니다.

엔진 버전 업그레이드 방법

Valkey 및 Redis OSS

Valkey 및 Redis 를 사용하면 ElastiCache 콘솔OSS, 또는 를 사용하여 수정하고 최신 엔진 버전을 지정하여 클러스터 AWS CLI또는 ElastiCache API 복제 그룹에 대한 버전 업그레이드를 시작할 수 있습니다. 자세한 내용은 다음 항목을 참조하십시오.

클러스터 및 복제 그룹을 수정하는 방법	
클러스터	복제 그룹
사용 ElastiCache AWS Management Console	사용 AWS Management Console
와 AWS CLI 함께 사용 ElastiCache	사용 AWS CLI
사용 ElastiCache API	사용 ElastiCache API

Memcached

Memcached를 사용하면 클러스터에 대한 버전 업그레이드를 시작하려면 클러스터를 수정하고 최신 엔진 버전을 지정합니다. ElastiCache 콘솔, AWS CLI또는 를 사용하여 이 작업을 수행할 수 있습니다 ElastiCache API.

- 를 사용하려면 – 단원을 AWS Management Console참조하세요사용 ElastiCache AWS Management Console.
- 를 사용하려면 섹션을 AWS CLI참조하세요와 AWS CLI 함께 사용 ElastiCache.
- 를 사용하려면 섹션을 ElastiCache API참조하세요사용 ElastiCache API.

Redis에서 Valkey로 엔진 간 업그레이드를 트리거OSS하는 방법

콘솔 또는 API 를 사용하여 기존 Redis OSS 복제 그룹(v4 이상)을 Valkey 엔진으로 업그레이드할 수 있습니다CLI.

> **Note**
>
> 기존 RedisOSS(클러스터 모드 비활성화됨) 단일 노드 클러스터를 Valkey 엔진으로 업그레이드하려면 먼저 다음 사전 필수 단계에 따라 복제 그룹에 추가해야 합니다기존 클러스터를 사용하여 복제 그룹 생성.

기본 캐시 파라미터 그룹을 사용하는 기존 Redis OSS 복제 그룹이 있는 경우 를 사용하여 새 엔진 및 엔진 버전을 지정하여 Valkey로 modify-replication-group 업그레이드할 수 있습니다API.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-replication-group \
   --replication-group-id myReplGroup \
   --engine valkey \
   --engine-version 7.2
```

Windows의 경우:

```
aws elasticache modify-replication-group ^
   --replication-group-id myReplGroup ^
   --engine valkey ^
   --engine-version 7.2
```

업그레이드하려는 기존 redis 복제 그룹에 사용자 지정 캐시 파라미터 그룹이 적용된 경우 요청에서 사용자 지정 Valkey 캐시 파라미터 그룹도 전달해야 합니다. 입력 Valkey 사용자 지정 파라미터 그룹은 기존 Redis 사용자 지정 파라미터 그룹과 동일한 Redis 정적 파라미터 값을 가져야 합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-replication-group \
   --replication-group-id myReplGroup \
   --engine valkey \
   --engine-version 7.2 \
   --cache-parameter-group-name myParamGroup
```

Windows의 경우:

```
aws elasticache modify-replication-group ^
   --replication-group-id myReplGroup ^
   --engine valkey ^
   --engine-version 7.2 ^
   --cache-parameter-group-name myParamGroup
```

ElastiCache ServerlessOSS용 Valkey 교차 엔진 업그레이드로 Redis

를 사용하여 새 엔진 및 메이저 엔진 버전을 지정CLI하여 콘솔 API 또는 를 사용하여 기존 Redis OSS 서버리스 캐시를 Valkey 엔진으로 업그레이드할 modify-serverless-cache 수 있습니다API.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-serverless-cache \
   --serverless-cache-name myCluster \
   --engine valkey \
   --major-engine-version 7
```

Windows의 경우:

```
aws elasticache modify-serverless-cache ^
   --serverless-cache-name myCluster ^
   --engine valkey ^
   --major-engine-version 7
```

지원되는 엔진 및 버전

ElastiCache 서버리스 캐시는 Valkey 7.2 이상, Redis OSS 버전 7.0 및 Memcached 1.6 이상을 지원합니다.

ElastiCache 자체 설계된 캐시는 Valkey 7.2 이상, 모든 Redis OSS 버전 4.0.10 이상 및 Memcached 버전 1.4.5 이상을 지원합니다.

자체 설계된 ElastiCache 클러스터는 다음 Valkey 버전을 지원합니다.

- 지원되는 Valkey 버전
- 지원되는 Redis OSS 버전
- Redis OSS 버전 수명 종료 일정
- 지원되는 ElastiCache (Memcached) 버전

지원되는 Valkey 버전

아래에서 지원되는 Valkey 버전입니다. Valkey는 기본적으로 Redis OSS 7.2에서 사용할 수 있는 대부분의 기능을 지원합니다.

ElastiCache (발키) 버전 7.2.6

2024년 10월 10일 Valkey 7.2.6 ElastiCache 이 출시되었습니다. 다음은 Valkey 7.2에 도입된 몇 가지 새로운 기능입니다(Redis OSS 7.1과 비교).

- ZRANK 및 ZREVRANK 명령에 대한 새 WITHSCORE 옵션
- CLIENT 아니요 -TOUCH 클라이언트가 키의 LRU/LFU에 영향을 주지 않고 명령을 실행할 수 있습니다.
- 복제를 기반으로 클러스터 모드에서 노드를 논리적으로 그룹화하도록 노드의 샤드 ID를 반환 CLUSTERMYSHARDID하는 새 명령입니다.
- 다양한 데이터 유형에 대한 성능 및 메모리 최적화.

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

- 동일한 채널을 구독하는 RESP3 클라이언트PUBLISH를 호출하면 순서가 변경되고 게시된 메시지보다 먼저 응답이 전송됩니다.
- 스크립트에 대한 클라이언트 측 추적은 이제 EVAL/ 호출자가 선언한 키 대신 스크립트가 읽는 키를 추적합니다FCALL.

- 동결 시간 샘플링은 명령 실행 중 및 스크립트에서 발생합니다.
- 차단된 명령이 차단 해제되면 ACL, OOM및 기타와 같은 검사가 재평가됩니다.
- ACL 오류 오류 메시지 텍스트와 오류 코드가 통합됩니다.
- 키가 더 이상 존재하지 않을 때 해제되는 차단된 스트림 명령에는 다른 오류 코드(-NOGROUP 대신 또는 -WRONGTYPE)가 있습니다UNBLOCKED.
- 명령 통계는 명령이 실제로 실행되는 경우에만 차단된 명령에 대해 업데이트됩니다.
- ACL 사용자의 내부 스토리지는 더 이상 중복 명령 및 범주 규칙을 제거하지 않습니다. 이렇게 하면 이러한 규칙이 ACL SAVE, ACL GETUSER 및 ACL 의 일부로 표시되는 방식이 변경될 수 있습니다 LIST.
- 가능한 SNI 경우 TLS기반 복제를 위해 생성된 모든 클라이언트 연결입니다.
- XINFO STREAM: 이제 확인된 시간 응답 필드는 마지막으로 성공한 상호 작용 대신 마지막으로 시도한 상호 작용을 나타냅니다. 이제 새 활성 시간 응답 필드는 마지막으로 성공한 상호 작용을 나타냅니다.
- XREADGROUP 및 X[AUTO]CLAIM는 일부 읽기/청구를 수행할 수 있었는지 여부에 관계없이 소비자를 생성합니다. [TBD - '여기서'란 무엇인가요?]
- ACL ACLLIST/에서 새로 생성된 기본 사용자 세트 sanitize-payload 플래그입니다GETUSER.
- 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 의 Redis OSS 7.2.4 릴리스 정보(Valkey 7.2에는 Redis에서 버전 7.2.4OSS까지의 모든 변경 사항이 포함됨) 및 Valkey의 Valkey 7.2 릴리스 정보를 참조하세요 GitHub.

지원되는 Redis OSS 버전

ElastiCache 서버리스 캐시와 자체 설계된 캐시는 모든 Redis OSS 버전 7.1 이하를 지원합니다.

- ElastiCache (발키) 버전 7.2.6
- ElastiCache (Redis OSS) 버전 7.1(향상됨)

자체 설계된 ElastiCache 클러스터는 다음 Valkey 및 Redis OSS 버전을 지원합니다.

- ElastiCache (Redis OSS) 버전 7.1(향상됨)

- ElastiCache (Redis OSS) 버전 7.0(향상됨)
- ElastiCache (Redis OSS) 버전 6.2(향상됨)
- ElastiCache (Redis OSS) 버전 6.0(향상됨)
- ElastiCache (Redis OSS) 버전 5.0.6(향상됨)
- ElastiCache (Redis OSS) 버전 5.0.5(사용되지 않음, 버전 5.0.6 사용)
- ElastiCache (Redis OSS) 버전 5.0.4(사용되지 않음, 버전 5.0.6 사용)
- ElastiCache (Redis OSS) 버전 5.0.3(사용되지 않음, 버전 5.0.6 사용)
- ElastiCache (Redis OSS) 버전 5.0.0(사용되지 않음, 버전 5.0.6 사용)
- ElastiCache (Redis OSS) 버전 4.0.10(향상됨)
- 과거 수명 종료(EOL) 버전(3.x)
- 과거 수명 종료(EOL) 버전(2.x)

ElastiCache (Redis OSS) 버전 7.1(향상됨)

이 릴리스에는 워크로드가 처리량을 높이고 운영 지연 시간을 줄일 수 있도록 하는 성능 개선 사항이 포함되어 있습니다. ElastiCache 7.1에는 두 가지 주요 개선 사항이 도입되었습니다.

프레젠테이션 계층 로직도 처리하도록 향상된 I/O 스레드 기능이 확장되었습니다. 프레젠테이션 계층이란 이제 클라이언트 입력을 읽을 뿐만 아니라 입력을 Redis OSS 바이너리 명령 형식으로 구문 분석하는 향상된 I/O 스레드를 의미합니다. 그런 다음 기본 스레드로 전달되어 실행되므로 성능이 향상됩니다. Redis OSS 메모리 액세스 패턴이 개선되었습니다. 여러 데이터 구조 작업의 실행 단계가 삽입되므로 병렬 메모리 액세스가 보장되고 메모리 액세스 지연 시간이 단축됩니다. Graviton3-based `R7g.4xlarge` 이상 ElastiCache 에서 를 실행할 때 고객은 노드당 초당 1백만 개 이상의 요청을 달성할 수 있습니다. ElastiCache (Redis OSS) v7.1의 성능 개선으로 고객은 ElastiCache (Redis OSS) v7.0에 비해 최대 100% 더 많은 처리량과 50% 더 낮은 P99 지연 시간을 달성할 수 있습니다. 이러한 개선 사항은 CPU 유형에 관계없이 물리적 코어가 8개 이상인 노드 크기(`2xlarge`Graaviton 및 x86)`4xlarge`에서 활성화되며 클라이언트를 변경할 필요가 없습니다.

> **Note**
>
> ElastiCache v7.1은 Redis OSS v7.0과 호환됩니다.

ElastiCache (Redis OSS) 버전 7.0(향상됨)

ElastiCache (Redis OSS) 7.0에는 새로운 기능에 대한 여러 개선 사항과 지원이 추가되었습니다.

- 함수 : ElastiCache (Redis OSS) 7은 Redis OSS 함수에 대한 지원을 추가하고, 개발자가 클라이언트가 모든 연결로 스크립트를 서버에 재전송할 필요 없이 클러스터에 ElastiCache 저장된 애플리케이션 로직LUA으로 스크립트를 실행할 수 있는 관리형 환경을 제공합니다.
- ACL 개선 사항 : Valkey 및 Redis OSS 7은 다음 버전의 액세스 제어 목록()에 대한 지원을 추가합니다ACLs. 이제 클라이언트는 Valkey 및 Redis 의 특정 키 또는 키스페이스에 대해 여러 권한 세트를 지정할 수 있습니다OSS.
- Sharded Pub/Sub : ElastiCache with Valkey and Redis OSS 7은 클러스터 모드 활성화() ElastiCache 에서 를 실행할 때 샤딩된 방식으로 Pub/Sub 기능을 실행할 수 있는 지원을 추가합니다CME. Pub/Sub 기능을 사용하면 게시자가 채널의 구독자 수에 관계없이 메시지를 발행할 수 있습니다. 채널은 ElastiCache 클러스터의 샤드에 바인딩되므로 샤드 간에 채널 정보를 전파할 필요가 없으므로 확장성이 향상됩니다.
- 향상된 I/O 멀티플렉싱:Valkey 및 Redis OSS 7 ElastiCache 을 사용하면 향상된 I/O 멀티플렉싱이 도입되어 ElastiCache 클러스터에 대한 동시 클라이언트 연결이 많은 고처리량 워크로드에 처리량을 늘리고 지연 시간을 줄일 수 있습니다. 예를 들어 r6g.xlarge 노드 클러스터를 사용하고 5,200개의 동시 클라이언트를 실행하는 경우 Redis OSS 버전 6과 비교하여 처리량(초당 읽기 및 쓰기 작업)을 최대 72% 높이고 P99 지연 시간을 최대 71% 줄일 수 ElastiCache 있습니다.

Valkey에 대한 자세한 내용은 Valkey 단원을 참조하세요. Redis OSS 7.0 릴리스에 대한 자세한 내용은 의 Redis에서 Redis OSS 7.0 릴리스 정보를 참조OSS하세요 GitHub.

ElastiCache (Redis OSS) 버전 6.2(향상됨)

ElastiCache (Redis OSS) 6.2에는 8 vCPUs 개 이상의 x86 노드 유형 또는 4개 vCPUs 이상의 Graviton2 노드 유형을 사용하는 TLS활성화된 클러스터에 대한 성능 개선이 포함되어 있습니다. 이러한 향상된 기능은 다른 로 암호화를 오프로드하여 처리량을 개선하고 클라이언트 연결 설정 시간을 줄입니다vCPUs. Redis OSS 6.2를 사용하면 액세스 제어 목록(ACL) 규칙을 사용하여 Pub/Sub 채널에 대한 액세스를 관리할 수도 있습니다.

이 버전에서는 로컬로 연결된 NVMe 가 포함된 클러스터 노드의 데이터 계층화에 대한 지원도 소개합니다SSD. 자세한 내용은 의 데이터 계층화 ElastiCache 단원을 참조하십시오.

또한 Redis OSS 엔진 버전 6.2.6은 Redis OSS 클러스터 내에서 복잡한 데이터 세트를 인코딩하는 간단하고 스키마 없는 방법인 네이티브 JavaScript 객체 표기법(JSON) 형식에 대한 지원을 도입했습니다. JSON 지원을 통해 에서 작동하는 애플리케이션의 성능 및 RedisOSSAPIs를 활용할 수 있습니다JSON. 자세한 내용은 시작하기를 참조하세요JSON. 또한 이 데이터 유형의 사용을 모니터링 CloudWatch 하기 위해 에 `JsonBasedCmdsLatency`통합된 JSON관련 지표 `JsonBasedCmds` 및 도 포함되어 있습니다. 자세한 내용은 Valkey 및 Redis에 대한 지표 OSS 단원을 참조하십시오.

6.2. ElastiCache (를 사용하여 엔진 버전을 지정합니다.Redis OSS)는 사용 가능한 Redis OSS 6.2의 기본 패치 버전을 자동으로 호출합니다. 예를 들어 캐시 클러스터를 생성/수정할 때 `--engine-version` 파라미터를 6.2로 설정합니다. 클러스터는 생성/수정 시 현재 사용 가능한 Redis OSS 6.2 선호 패치 버전으로 시작됩니다. 에서 엔진 버전 6.x를 지정API하면 최신 마이너 버전의 Redis OSS 6이 생성됩니다.

기존 6.0 클러스터의 경우 , `CreateCacheCluster` `ModifyCacheCluster` `CreateReplicationGroup` 또는 `yes` 에서 `AutoMinorVersionUpgrade` 파라미터를 로 설정하여 다음 자동 마이너 버전 업그레이드를 선택할 수 있습니다`ModifyReplicationGroup`APIs. ElastiCache (Redis OSS)는 셀프 서비스 업데이트를 사용하여 기존 6.0 클러스터의 마이너 버전을 6.2로 업그레이드합니다. 자세한 내용은 Amazon 의 셀프 서비스 업데이트를 ElastiCache 참조하세요.

를 호출하면 DescribeCacheEngineVersions API `EngineVersion` 파라미터 값이 6.2로 설정되고 패치 버전이 있는 실제 엔진 버전이 `CacheEngineVersionDescription` 필드에 반환됩니다.

Redis OSS 6.2 릴리스에 대한 자세한 내용은 의 Redis에서 Redis OSS 6.2 릴리스 정보를 참조OSS하세요 GitHub.

ElastiCache (Redis OSS) 버전 6.0(향상됨)

Amazon ElastiCache (Redis OSS)은 역할 기반 액세스 제어를 통한 사용자 인증, 클라이언트 측 캐싱 및 상당한 운영 개선 사항을 포함하는 다음 버전의 Redis OSS 엔진을 도입합니다.

Redis OSS 6.0부터 ElastiCache (Redis OSS)는 여러 패치 버전을 제공하는 대신 각 Redis OSS 마이너 릴리스에 대해 단일 버전을 제공합니다. ElastiCache (Redis OSS)는 실행 중인 캐시 클러스터의 패치 버전을 자동으로 관리하여 성능을 개선하고 보안을 강화합니다.

`AutoMinorVersionUpgrade` 파라미터를 로 설정하여 다음 자동 마이너 버전 업그레이드에 옵트인할 수도 `yes` 있으며, ElastiCache (Redis OSS)는 셀프 서비스 업데이트를 통해 마이너 버전 업그레이드를 관리합니다. 자세한 내용은 의 서비스 업데이트 ElastiCache 단원을 참조하십시오.

를 사용하여 엔진 버전을 지정합니다`6.0`. ElastiCache (Redis OSS)는 사용 가능한 Redis OSS 6.0의 기본 패치 버전을 자동으로 호출합니다. 예를 들어 캐시 클러스터를 생성/수정하는 경우 `--engine-version` 파라미터를 6.0으로 설정합니다. 클러스터는 생성/수정 시 현재 사용 가능한 기본 Redis OSS 6.0 패치 버전으로 시작됩니다. 특정 패치 버전 값을 사용한 모든 요청이 거부되고 예외가 발생한 후 프로세스가 실패합니다.

를 호출하면 DescribeCacheEngineVersions API `EngineVersion` 파라미터 값이 6.0으로 설정되고 패치 버전이 있는 실제 엔진 버전이 `CacheEngineVersionDescription` 필드에 반환됩니다.

Redis OSS 6.0 릴리스에 대한 자세한 내용은 의 Redis에서 Redis OSS 6.0 릴리스 정보를 참조OSS하세요 GitHub.

ElastiCache (Redis OSS) 버전 5.0.6(향상됨)

Amazon ElastiCache (Redis OSS)은 다음 버전의 Redis OSS 엔진을 도입합니다. 여기에는 버그 수정 및 다음과 같은 누적 업데이트가 포함됩니다.

- 특별한 조건에서 엔진 안정성 보장.
- 향상된 Hyperloglog 오류 처리.
- 안정적인 복제를 위한 향상된 핸드셰이크 명령
- XCLAIM 명령을 통한 일관된 메시지 배달 추적.
- 객체에서의 향상된 LFU 필드 관리.
- ZPOP 사용 시 향상된 트랜잭션 관리.
- 명령 이름 바꾸기 기능: FLUSHALL 또는 와 같이 실수로 데이터가 손실될 수 rename-commands 있는 잠재적으로 위험하거나 비용이 많이 드는 Redis OSS 명령의 이름을 바꿀 수 있는 라는 파라미터입니다FLUSHDB. 이는 오픈 소스 Redis 의 rename-command 구성과 유사합니다OSS. 그러나 ElastiCache는 완전 관리형 워크플로를 제공하여 경험을 개선했습니다. 명령 이름 변경은 즉시 적용되며, 명령 목록을 포함하는 클러스터의 모든 노드에 자동으로 전파됩니다. 사용자의 개입(노드 재부팅 등)은 필요 없습니다.

  다음 예제에서는 기존 파라미터 그룹을 수정하는 방법을 보여줍니다. 이러한 예제에는 이름을 변경하려는 명령 목록(공백으로 구분)인 rename-commands 파라미터가 포함됩니다.

```
aws elasticache modify-cache-parameter-group --cache-parameter-group-
name custom_param_group
--parameter-name-values "ParameterName=rename-commands,  ParameterValue='flushall
 restrictedflushall'" --region region
```

  이 예제에서는 rename-commands 파라미터를 사용하여 flushall 명령을 restrictedflushall로 이름 변경합니다.

  여러 명령의 이름을 변경하려면 다음을 사용하세요.

```
aws elasticache modify-cache-parameter-group --cache-parameter-group-
name custom_param_group
--parameter-name-values "ParameterName=rename-commands,  ParameterValue='flushall
 restrictedflushall flushdb restrictedflushdb'" --region region
```

변경을 되돌리려면 다음과 같이 명령을 다시 실행하고, 유지하려는 ParameterValue 목록에서 이름 변경된 값을 제외시킵니다.

```
aws elasticache modify-cache-parameter-group --cache-parameter-group-
name custom_param_group
--parameter-name-values "ParameterName=rename-commands,  ParameterValue='flushall
 restrictedflushall'" --region region
```

이 경우, flushall 명령은 restrictedflushall로 이름이 변경되고 이름 변경된 다른 명령은 원래 명령 이름으로 되돌려집니다.

> **Note**
>
> 명령 이름 변경 시 다음과 같은 제한이 따릅니다.
>
> - 이름 변경된 모든 명령은 영숫자여야 합니다.
> - 새 명령 이름의 최대 길이는 20자(영숫자)입니다.
> - 명령 이름을 변경할 경우 해당 클러스터와 연결된 파라미터 그룹을 업데이트해야 합니다.
> - 명령 사용을 전체적으로 차단하려면 다음과 같이 blocked 키워드를 사용합니다.
>
> ```
> aws elasticache modify-cache-parameter-group --cache-parameter-group-
> name custom_param_group
> --parameter-name-values "ParameterName=rename-commands,
>  ParameterValue='flushall blocked'" --region region
> ```

파라미터 변경에 대한 정보와 이름을 변경할 수 있는 명령 목록을 보려면 Redis OSS 5.0.3 파라미터 변경 사항 섹션을 참조하세요.

- Redis OSS Streams: 이 모델은 생산자가 새 항목을 실시간으로 추가할 수 있는 로그 데이터 구조를 모델링합니다. 또한 소비자가 차단 또는 차단하지 않는 방식으로 메시지를 소비할 수 있습니다. 또한 스트림을 사용하여 클라이언트 그룹을 대표하는 소비자 그룹이 Apache Kafka와 비슷한 메시지 스트림의 서로 다른 부분을 공동으로 사용할 수 있습니다. 자세한 내용은 스트림 을 참조하세요.
- XADD, XRANGE 및 XREAD와 같은 스트림 명령군 지원. 자세한 내용은 Streams Commands를 참조하세요.
- 새 파라미터 및 이름이 변경된 파라미터의 수. 자세한 내용은 Redis OSS 5.0.0 파라미터 변경 사항 단원을 참조하십시오.

- 새 Redis 지표, OSS `StreamBasedCmds`.
- Redis OSS 노드의 스냅샷 시간이 약간 빨라졌습니다.

> **Important**
>
> ElastiCache (Redis OSS)는 Redis OSS 오픈 소스 버전 5.0.1의 두 가지 중요한 버그 수정을 백포트했습니다. 해당되는 사항은 다음과 같습니다.
>
> - RESTORE 특정 키가 이미 만료된 경우 불일치 응답.
> - `XCLAIM` 명령은 잠재적으로 잘못된 항목을 반환하거나 프로토콜을 동기화 해제할 수 있습니다.
>
> 이러한 버그 수정은 모두 Redis OSS 엔진 버전 5.0.0에 대한 ElastiCache (Redis OSS) 지원에 포함되며 향후 버전 업데이트에서 사용됩니다.

자세한 내용은 의 Redis에서 Redis OSS 5.0.6 릴리스 정보를 참조OSS하세요 GitHub.

ElastiCache (Redis OSS) 버전 5.0.5(사용되지 않음, 버전 5.0.6 사용)

Amazon ElastiCache (Redis OSS)은 다음 버전의 Redis OSS 엔진을 도입합니다. 여기에는 계획된 모든 작업 중 자동 장애 조치 클러스터의 ElastiCache (Redis OSS)에 대한 온라인 구성 변경 사항이 포함됩니다. 이제 클러스터를 확장하고, Redis OSS 엔진 버전을 업그레이드하고, 클러스터가 온라인 상태를 유지하고 수신 요청을 계속 제공하는 동안 패치 및 유지 관리 업데이트를 적용할 수 있습니다. 여기에는 버그 수정도 포함되어 있습니다.

자세한 내용은 의 Redis에서 Redis OSS 5.0.5 릴리스 정보를 참조OSS하세요 GitHub.

ElastiCache (Redis OSS) 버전 5.0.4(사용되지 않음, 버전 5.0.6 사용)

Amazon ElastiCache (Redis OSS)은 Amazon 에서 지원하는 다음 버전의 Redis OSS 엔진을 도입합니다 ElastiCache. 다음과 같은 향상된 기능을 포함합니다.

- 특별한 조건에서 엔진 안정성 보장.
- 향상된 Hyperloglog 오류 처리.
- 안정적인 복제를 위한 향상된 핸드셰이크 명령
- `XCLAIM` 명령을 통한 일관된 메시지 배달 추적.

- 객체에서의 향상된 LFU 필드 관리.
- ZPOP 사용 시 향상된 트랜잭션 관리.

자세한 내용은 의 Redis에서 Redis OSS 5.0.4 릴리스 정보를 참조OSS하세요 GitHub.

ElastiCache (Redis OSS) 버전 5.0.3(사용되지 않음, 버전 5.0.6 사용)

Amazon ElastiCache (Redis OSS)은 Amazon ElastiCache에서 지원하는 다음 버전의 Redis OSS 엔진을 도입합니다. 여기에는 버그 수정이 포함됩니다.

ElastiCache (Redis OSS) 버전 5.0.0(사용되지 않음, 버전 5.0.6 사용)

Amazon ElastiCache (Redis OSS)은 Amazon 에서 지원하는 Redis OSS 엔진의 다음 메이저 버전을 도입합니다 ElastiCache. ElastiCache (Redis OSS) 5.0.0은 다음과 같은 개선 사항을 지원합니다.

- Redis OSS Streams: 이 모델은 생산자가 새 항목을 실시간으로 추가할 수 있는 로그 데이터 구조를 모델링합니다. 또한 소비자가 차단 또는 차단하지 않는 방식으로 메시지를 소비할 수 있습니다. 또한 스트림을 사용하여 클라이언트 그룹을 대표하는 소비자 그룹이 Apache Kafka와 비슷한 메시지 스트림의 서로 다른 부분을 공동으로 사용할 수 있습니다. 자세한 내용은 스트림 을 참조하세요.
- XADD, XRANGE 및 XREAD와 같은 스트림 명령군 지원. 자세한 내용은 Streams Commands를 참조하세요.
- 새 파라미터 및 이름이 변경된 파라미터의 수. 자세한 내용은 Redis OSS 5.0.0 파라미터 변경 사항 단원을 참조하십시오.
- 새 Redis 지표, OSS StreamBasedCmds.
- Redis OSS 노드의 스냅샷 시간이 약간 빨라졌습니다.

ElastiCache (Redis OSS) 버전 4.0.10(향상됨)

Amazon ElastiCache (Redis OSS)은 Amazon 에서 지원하는 Redis OSS 엔진의 다음 메이저 버전을 도입합니다 ElastiCache. ElastiCache (Redis OSS) 4.0.10은 다음과 같은 개선 사항을 지원합니다.

- 온라인 클러스터 크기 조정과 암호화를 모두 단일 ElastiCache (Redis OSS) 버전으로 제공합니다. 자세한 내용은 다음 자료를 참조하세요.
  - Valkey 또는 Redis에서 클러스터 크기 조정OSS(클러스터 모드 활성화됨)
  - Valkey 또는 Redis에 대한 온라인 리샤딩OSS(클러스터 모드 활성화됨)
  - Amazon의 데이터 보안 ElastiCache

- 새 파라미터의 수입니다. 자세한 내용은 Redis OSS 4.0.10 파라미터 변경 사항 단원을 참조하십시오.
- MEMORY와 같은 메모리 명령군 지원. 자세한 내용은 명령(에서 검색)을 참조하세요MEMO.
- 온라인 상태에서 메모리 조각 모음을 지원하여 더욱 효율적인 메모리 사용률과 데이터에 대해 더 많이 사용 가능한 메모리가 허용됩니다.
- 비동기 플러시 및 삭제를 지원합니다. ElastiCache (Redis OSS)는 UNLINK FLUSHDB 및 와 같은 명령을 지원FLUSHALL하여 메인 스레드와 다른 스레드에서 실행합니다. 이렇게 하면 비동기식으로 메모리를 확보하여 애플리케이션의 성능 및 응답 시간을 향상시킬 수 있습니다.
- 새 Redis 지표, OSS ActiveDefragHits. 자세한 내용은 Redis 지표를 참조하세요OSS.

Redis OSS 버전 3.2.10을 실행하는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 사용자는 콘솔을 사용하여 온라인 업그레이드를 통해 클러스터를 업그레이드할 수 있습니다.

ElastiCache (Redis OSS) 클러스터 크기 조정 및 암호화 지원 비교

기능	3.2.6	3.2.10	4.0.10 이상
온라인 클러스터 크기 조정 *	아니요	예	예
전송 중 데이터 암호화 **	예	아니요	예
미사용 데이터 암호화 **	예	아니요	예
* 샤드 추가, 제거 및 재분배			
** Fed RAMP, HIPAA및 PCI DSS 규정 준수 애플리케이션에 필요합니다. 자세한 내용은 Amazon에 대한 규정 준수 검증 ElastiCache 단원을 참조하십시오.			

과거 수명 종료(EOL) 버전(3.x)

ElastiCache (Redis OSS) 버전 3.2.10(향상됨)

Amazon ElastiCache (Redis OSS)은 Amazon 에서 지원하는 Redis OSS 엔진의 다음 메이저 버전을 도입합니다 ElastiCache. ElastiCache (Redis OSS) 3.2.10에서는 수신 I/O 요청을 계속 처리하는 동안 클러스터에서 샤드를 추가하거나 제거하기 위한 온라인 클러스터 크기 조정을 도입합니다. ElastiCache (Redis OSS) 3.2.10 사용자는 데이터를 암호화하는 기능을 제외하고 이전 Redis OSS 버전의 모든 기능을 사용할 수 있습니다. 이 기능은 현재 버전 3.2.6에서만 사용할 수 있습니다.

ElastiCache (Redis OSS) 버전 3.2.6과 3.2.10 비교

기능	3.2.6	3.2.10
온라인 클러스터 크기 조정 *	아니요	예
전송 중 데이터 암호화 **	예	아니요
미사용 데이터 암호화 **	예	아니요
* 샤드 추가, 제거 및 재분배		
** Fed RAMP, HIPAA및 PCI DSS 규정 준수 애플리케이션에 필요합니다. 자세한 내용은 Amazon에 대한 규정 준수 검증 ElastiCache 단원을 참조하십시오.		

자세한 내용은 다음 자료를 참조하세요.

- Valkey 또는 Redis에 대한 온라인 리샤딩OSS(클러스터 모드 활성화됨)
- 온라인 클러스터 크기 조정

ElastiCache (Redis OSS) 버전 3.2.6(향상됨)

Amazon ElastiCache (Redis OSS)은 Amazon 에서 지원하는 Redis OSS 엔진의 다음 메이저 버전을 도입합니다 ElastiCache. ElastiCache (Redis OSS) 3.2.6 사용자에게는 이전 Redis OSS 버전의 모든 기능과 데이터를 암호화하는 옵션이 있습니다. 자세한 내용은 다음 자료를 참조하세요.

- ElastiCache 전송 중 암호화(TLS)
- 의 저장 시 암호화 ElastiCache
- Amazon에 대한 규정 준수 검증 ElastiCache

ElastiCache (Redis OSS) 버전 3.2.4(향상됨)

Amazon ElastiCache (Redis OSS) 버전 3.2.4에는 Amazon 에서 지원하는 Redis OSS 엔진의 다음 메이저 버전이 도입되었습니다 ElastiCache. ElastiCache (Redis OSS) 3.2.4 사용자는 이전 Redis OSS 버전의 모든 기능과 클러스터 모드 또는 비클러스터 모드 에서 실행할 수 있는 옵션을 사용할 수 있습니다. 다음 표에는 이에 대해 요약되어 있습니다.

Redis OSS 3.2.4 비클러스터 모드 및 클러스터 모드 비교

기능	비클러스터 모드	클러스터 모드
데이터 파티셔닝	아니요	예
지역 검색 인덱싱	예	예
노드 유형 변경	예	예 *
복제본 조정	예	예 *
스케일 아웃	아니요	예 *
데이터베이스 지원	다양함	단일
Parameter Group	`default.redis3.2` **	`default.redis3.2.cluster.on` **
* 백업에서 새 캐시로 복원 섹션 참조		
** 또는 해당 그룹에서 파생된 파라미터		

참고:

- 분할 - 각 노드 그룹에 대한 복제 지원을 통해 데이터를 2~500개의 노드 그룹(샤드)으로 분할할 수 있는 기능입니다.
- 지리 공간 인덱싱 – Redis OSS 3.2.4는 6개의 GEO 명령을 통한 지리 공간 인덱싱에 대한 지원을 도입합니다. 자세한 내용은 Valkey 명령 페이지(에 대해 필터링됨)의 Redis OSS GEO* 명령 설명서 명령: GEO 를 참조하세요GEO.

추가 Redis OSS 3 기능에 대한 자세한 내용은 Redis OSS 3.2 릴리스 정보 및 Redis OSS 3.0 릴리스 정보 섹션을 참조하세요.

현재 ElastiCache 관리형 Valkey 또는 RedisOSS(클러스터 모드 활성화됨)는 다음 Redis OSS 3.2 기능을 지원하지 않습니다.

- 복제본 마이그레이션
- 클러스터 재분배

- Lua 디버거

ElastiCache 는 다음 Redis OSS 3.2 관리 명령을 비활성화합니다.

- `cluster meet`
- `cluster replicate`
- `cluster flushslots`
- `cluster addslots`
- `cluster delslots`
- `cluster setslot`
- `cluster saveconfig`
- `cluster forget`
- `cluster failover`
- `cluster bumpepoch`
- `cluster set-config-epoch`
- `cluster reset`

Redis OSS 3.2.4 파라미터에 대한 자세한 내용은 섹션을 참조하세요Redis OSS 3.2.4 파라미터 변경 사항.

과거 수명 종료(EOL) 버전(2.x)

ElastiCache (Redis OSS) 버전 2.8.24(향상됨)

버전 2.8.23 이후 추가된 Redis OSS 개선 사항에는 버그 수정 및 잘못된 메모리 액세스 주소 로깅이 포함됩니다. 자세한 내용은 Redis OSS 2.8 릴리스 정보 섹션을 참조하세요.

ElastiCache (Redis OSS) 버전 2.8.23(향상됨)

버전 2.8.22 이후 추가된 Redis OSS 개선 사항에 버그 수정이 포함되어 있습니다. 자세한 내용은 Redis OSS 2.8 릴리스 정보 섹션을 참조하세요. 이 릴리스에는 새 파라미터 `close-on-slave-write`에 대한 지원도 포함됩니다. 이 파라미터가 활성화되면 읽기 전용 복제본에 쓰려고 시도하는 클라이언트를 연결 해제합니다.

Redis OSS 2.8.23 파라미터에 대한 자세한 내용은 ElastiCache 사용 설명서Redis OSS 2.8.23(향상됨) 추가 파라미터의 섹션을 참조하세요.

ElastiCache (Redis OSS) 버전 2.8.22(향상됨)

버전 2.8.21 이후 추가된 Redis OSS 개선 사항은 다음과 같습니다.

- 백업 오버헤드에 대해 메모리를 적게 할당하고 애플리케이션에 많이 할당할 수 있는 forkless 백업 및 동기화에 대해 지원합니다. 자세한 내용은 동기화 및 백업 구현 방법 단원을 참조하십시오. forkless 프로세스는 지연 시간과 처리량 모두에 영향을 줄 수 있습니다. 높은 쓰기 처리량의 경우 복제본이 다시 동기화되면, 동기화되는 전체 시간에 대해 접속 불가능하게 될 수 있습니다.
- 장애 조치가 발생한 경우, 가능하면 언제든지 복제본이 기본 항목과 전체 동기화가 아닌 부분적인 동기화를 수행하므로 이제 복제 그룹이 더 빠르게 복구됩니다. 또한, 기본 항목 및 복제본 모두 동기화 중 더 이상 디스크를 사용하지 않으므로 속도가 향상됩니다.
- 두 가지 새로운 CloudWatch 지표를 지원합니다.
  - `ReplicationBytes` - 읽기 전용 복제본으로 전송되는 복제 그룹 기본 클러스터의 바이트 수.
  - `SaveInProgress` - 백그라운드 저장 프로세스가 실행 중인지 여부를 나타내는 이진 값.

  자세한 내용은 CloudWatch 지표 사용 모니터링 단원을 참조하십시오.
- 복제 PSYNC 동작에서 여러 가지 중요한 버그 수정. 자세한 내용은 Redis OSS 2.8 릴리스 정보 섹션을 참조하세요.
- 다중 AZ 복제 그룹에서 향상된 복제 성능을 유지하고 클러스터 안정성을 높이기 위해 비복 ElastiCache 제본은 더 이상 지원되지 않습니다.
- 복제 그룹에서 기본 클러스터와 복제본 간의 데이터 일관성을 향상하기 위해 복제본에서는 기본 클러스터와 별도로 더 이상 키를 제거하지 않습니다.
- Redis OSS 구성 변수 `appendonly` 및 `appendfsync`는 Redis OSS 버전 2.8.22 이상에서는 지원되지 않습니다.
- 메모리가 부족한 상황에서 큰 출력 버퍼가 있는 클라이언트는 복제본 클러스터에서 연결이 해제될 수 있습니다. 연결이 해제되면 클라이언트가 다시 연결해야 합니다. 이러한 상황은 PUBSUB 클라이언트에게 발생할 가능성이 가장 높습니다.

ElastiCache (Redis OSS) 버전 2.8.21

버전 2.8.19 이후 추가된 Redis OSS 개선 사항에는 여러 버그 수정 사항이 포함되어 있습니다. 자세한 내용은 Redis OSS 2.8 릴리스 정보 섹션을 참조하세요.

ElastiCache (Redis OSS) 버전 2.8.19

버전 2.8.6 이후 추가된 Redis OSS 개선 사항은 다음과 같습니다.

- 에 대한 지원 HyperLogLog. 자세한 내용은 Redis OSS 새 데이터 구조: HyperLogLog를 참조하세요.
- 정렬된 세트 데이터 유형은 이제 ZRANGEBYLEX, ZLEXCOUNT 및 ZREMRANGEBYLEX의 새 명령을 통해 사전 순 범위 쿼리를 지원합니다.
- 기본 노드가 복제본 노드로 오래된 데이터를 전송하지 못하도록 하려면 백그라운드 저장(bgsave) 하위 프로세스가 중단되면 마스터가 SYNC 실패합니다.
- HyperLogLogBasedCommands CloudWatch 지표에 대한 지원. 자세한 내용은 Valkey 및 Redis에 대한 지표 OSS 단원을 참조하십시오.

ElastiCache (Redis OSS) 버전 2.8.6

버전 2.6.13 이후 추가된 Redis OSS 개선 사항은 다음과 같습니다.

- 읽기 전용 복제본에 대한 복원성 및 내결함성이 개선되었습니다.
- 부분적 재동기화를 지원합니다.
- 항상 사용할 수 있어야 하는 읽기 전용 복제본의 사용자 정의 최소 숫자를 지원합니다.
- 게시/구독에 대한 전체 지원 - 서버에서의 이벤트를 클라이언트에 알리는 기능입니다.
- 기본 노드 장애의 자동 감지 및 기본 노드에서 보조 노드로 장애 조치

ElastiCache (Redis OSS) 버전 2.6.13

Redis OSS 버전 2.6.13은 Amazon에서 OSS 지원하는 Redis의 초기 버전이었습니다 ElastiCache (Redis OSS). Redis 2.6.13에서는 다중 OSS AZ가 지원되지 않습니다.

Redis OSS 버전 수명 종료 일정

이 섹션에서는 이전 메이저 버전이 발표되는 대로 수명 종료(EOL) 날짜를 정의합니다. 이를 통해 향후 버전 및 업그레이드 결정을 내릴 수 있습니다.

> **Note**
>
> ElastiCache 5.0.0에서 5.0.5까지의 (Redis OSS) 패치 버전은 더 이상 사용되지 않습니다. 버전 5.0.6 이상을 사용하세요.

다음 표에는 각 버전과 발표 EOL 날짜, 권장 업그레이드 대상 버전이 요약되어 있습니다.

과거 EOL

원본 엔진	원본 마이너 버전	권장 업그레이드 대상	EOL 날짜
버전 3	3.2.4, 3.2.6 및 3.2.10	버전 6.2 이상 **Note** US-ISO-EAST-1, US-ISO-WEST-1 및 US-ISOB-EAST-1 리전의 경우 5.0.6 이상을 사용하는 것이 좋습니다.	2023년 7월 31일
버전 2	2.8.24, 2.8.23, 2.8.22, 2.8.21, 2.8.19, , 2.8.12, 2.8.6, 2.6.13	버전 6.2 이상 **Note** US-ISO-EAST-1, US-ISO-WEST-1 및 US-ISOB-EAST-1 리전의 경우 5.0.6 이상을 사용하는 것이 좋습니다.	2023년 1월 13일

지원되는 ElastiCache (Memcached) 버전

ElastiCache 는 다음 Memcached 버전과 최신 버전으로 업그레이드를 지원합니다. 새 버전으로 업그레이드할 때 충족되지 않을 경우 업그레이드가 실패하는 조건에 주의를 기울이십시오.

ElastiCache Memcached 버전용

- Memcached 버전 1.6.22

- Memcached 버전 1.6.17
- Memcached 버전 1.6.12
- Memcached 버전 1.6.6
- Memcached 버전 1.5.16
- Memcached 버전 1.5.10
- Memcached 버전 1.4.34
- Memcached 버전 1.4.33
- Memcached 버전 1.4.24
- Memcached 버전 1.4.14
- Memcached 버전 1.4.5

Memcached 버전 1.6.22

ElastiCache (Memcached)는 Memcached 버전 1.6.22에 대한 지원을 추가합니다. 새로운 기능은 없지만 Memcached 1.6.18의 버그 수정 및 누적 업데이트가 포함되어 있습니다.

자세한 내용은 의 Memcached에서 ReleaseNotes1622를 참조하세요 GitHub.

Memcached 버전 1.6.17

ElastiCache (Memcached)는 Memcached 버전 1.6.17에 대한 지원을 추가합니다. 새로운 기능은 없지만 Memcached 1.6.17의 버그 수정 및 누적 업데이트가 포함되어 있습니다.

자세한 내용은 의 Memcached에서 ReleaseNotes1617을 참조하세요 GitHub.

Memcached 버전 1.6.12

ElastiCache (Memcached)는 Memcached 버전 1.6.12 및 전송 중 암호화에 대한 지원을 추가합니다. Memcached 1.6.6부터의 버그 해결 및 누적 업데이트도 포함되었습니다.

자세한 내용은 의 Memcached에서 ReleaseNotes1612를 참조하세요 GitHub.

Memcached 버전 1.6.6

ElastiCache (Memcached)는 Memcached 버전 1.6.6에 대한 지원을 추가합니다. 여기에는 새 기능이 포함되지 않지만 Memcached 1.5.16의 버그 수정 및 누적 업데이트가 포함됩니다. ElastiCache (Memcached)에는 Extstore 에 대한 지원이 포함되지 않습니다.

자세한 내용은 의 Memcached에서 ReleaseNotes166을 참조하세요 GitHub.

Memcached 버전 1.5.16

ElastiCache for Memcached는 Memcached 버전 1.5.16에 대한 지원을 추가합니다. 새로운 기능은 없지만 Memcached 1.5.14 및 Memcached 1.5.15의 버그 수정 및 누적 업데이트가 포함되어 있습니다.

자세한 내용은 의 Memcached에서 Memcached 1.5.16 릴리스 정보를 참조하세요 GitHub.

Memcached 버전 1.5.10

ElastiCache for Memcached 버전 1.5.10은 다음과 같은 Memcached 기능을 지원합니다.

- 자동화된 슬래브 재분배 기능.
- `murmur3` 알고리즘으로 더 빠른 해시 테이블 조회.
- 세분화된 LRU 알고리즘.
- LRU 크롤러를 백그라운드 복구 메모리에 연결합니다.
- `--enable-seccomp`: 컴파일 시간 옵션.

또한 `no_modern` 및 `inline_ascii_resp` 파라미터를 도입합니다. 자세한 내용은 Memcached 1.5.10 파라미터 변경 단원을 참조하십시오.

부터 Memcached 버전 1.4.34 ElastiCache 에 추가된 Memcached 개선 사항에는 다음이 포함됩니다.

- ASCII 멀티게트, CVE-2017-9951 및 에 대한 제한 크롤과 같은 누적 수정`metadumper`.
- 연결 한도에서 연결을 닫는 방식으로 연결 관리 향상.
- 1MB 이상의 항목 크기에 대한 항목 크기 관리 개선.
- 항목당 메모리 요구 사항을 몇 바이트 줄임으로써 성능 및 메모리 오버헤드 개선.

자세한 내용은 의 Memcached에서 Memcached 1.5.10 릴리스 정보를 참조하세요 GitHub.

Memcached 버전 1.4.34

ElastiCache Memcached 버전 1.4.34의 경우 버전 1.4.33에 새 기능이 추가되지 않습니다. 버전 1.4.34는 일반적인 릴리스보다 큰 버그 수정 릴리스입니다.

자세한 내용은 의 Memcached에서 Memcached 1.4.34 릴리스 정보를 참조하세요 GitHub.

Memcached 버전 1.4.33

버전 1.4.24부터 추가된 Memcached 개선 사항에는 다음이 포함됩니다.

- 특정 슬래브 클래스, 슬래브 클래스 목록 또는 모든 슬래브 클래스에 대한 모든 메타데이터를 덤프할 수 있습니다. 자세한 내용은 Memcached 1.4.31 릴리스 정보를 참조하세요.
- 1메가바이트 기본값보다 큰 항목에 대한 지원이 개선되었습니다. 자세한 내용은 Memcached 1.4.29 릴리스 정보를 참조하세요.
- 종료하라는 메시지가 표시되기 전에 클라이언트가 유휴 상태로 있을 수 있는 기간을 지정할 수 있습니다.

  클러스터를 다시 시작하지 않고 Memcached에 사용할 수 있는 메모리의 양을 동적으로 늘릴 수 있습니다. 자세한 내용은 Memcached 1.4.27 릴리스 정보를 참조하세요.
- 이제 `fetchers`, `mutations` 및 `evictions`의 로깅이 지원됩니다. 자세한 내용은 Memcached 1.4.26 릴리스 정보를 참조하세요.
- 빈 메모리를 전역 풀로 다시 회수하여 새 슬래브 클래스로 재할당할 수 있습니다. 자세한 내용은 Memcached 1.4.25 릴리스 정보를 참조하세요.
- 여러 가지 버그 수정.
- 일부 새 명령 및 파라미터. 목록을 보려면 Memcached 1.4.33 추가 파라미터 섹션을 참조하세요.

Memcached 버전 1.4.24

버전 1.4.14부터 추가된 Memcached 개선 사항에는 다음이 포함됩니다.

- 백그라운드 프로세스를 사용하여 최근에 사용한 (LRU) 관리가 가장 적습니다.
- 해시 알고리즘으로 jenkins 또는 murmur3의 옵션이 추가되었습니다.
- 일부 새 명령 및 파라미터. 목록을 보려면 Memcached 1.4.24 추가 파라미터 섹션을 참조하세요.
- 여러 가지 버그 수정.

Memcached 버전 1.4.14

버전 1.4.5부터 추가된 Memcached 개선 사항에는 다음이 포함됩니다.

- 슬래브 재분배 기능이 개선되었습니다.
- 성능 및 확장성 개선.
- 기존 항목을 가져오지 않고 해당 항목의 만료 시간을 업데이트하기 위해 터치 명령이 도입되었습니다.
- 자동 검색 - 클라이언트 프로그램이 클러스터의 모든 캐시 노드를 자동으로 확인하고 이러한 모든 노드에 대한 연결을 시작하고 유지 관리할 수 있는 기능입니다.

Memcached 버전 1.4.5

Memcached 버전 1.4.5는 Amazon ElastiCache (Memcached)에서 지원하는 초기 엔진 및 버전이었습니다.

Valkey와의 주요 버전 동작 및 호환성 차이

Valkey 7.2.6은 이전 버전의 Redis OSS 7.2.5와 유사한 호환성 차이를 보입니다. 지원되는 최신 버전의 Valkey는 섹션을 참조하세요지원되는 엔진 및 버전.

Redis와의 주요 버전 동작 및 호환성 차이 OSS

> **Important**
>
> 다음 페이지에서는 버전 간의 호환되지 않는 차이점을 모두 보여주며 최신 버전으로 업그레이드할 때 고려해야 할 사항을 알려줍니다. 이 목록에는 업그레이드 시 발생할 수 있는 모든 버전 비호환 문제가 포함되어 있습니다.
>
> 순차 업그레이드 없이 현재 Redis OSS 버전에서 사용 가능한 최신 Redis OSS 버전으로 직접 업그레이드할 수 있습니다. 예를 들어 Redis OSS 버전 3.0에서 버전 7.0으로 직접 업그레이드할 수 있습니다.

Redis OSS 버전은 MAJOR, MINOR및 PATCH 구성 요소로 구성된 의미 버전으로 식별됩니다. 예를 들어 Redis OSS 4.0.10에서 메이저 버전은 4이고 마이너 버전은 0이며 패치 버전은 10입니다. 이러한 값은 일반적으로 다음 규칙에 따라 증분됩니다.

- MAJOR 버전은 호환API되지 않는 변경에 사용됩니다.
- MINOR 버전은 이전 버전과 호환되는 방식으로 추가된 새 기능을 위한 것입니다.
- PATCH 버전은 백워드 호환 버그 수정 및 비기능적 변경에 사용됩니다.

최신 성능 및 안정성 개선을 위해 항상 지정된 MAJOR.MINOR 버전 내에서 최신 패치 버전을 유지하는 것이 좋습니다. Redis OSS 6.0부터 ElastiCache (Redis OSS)는 여러 패치 버전을 제공하는 대신 각 Redis OSS 마이너 릴리스에 대해 단일 버전을 제공합니다. ElastiCache (Redis OSS)는 실행 중인 캐시 클러스터의 패치 버전을 자동으로 관리하여 성능을 개선하고 보안을 강화합니다.

또한 대부분의 주요 개선 사항은 이전 버전으로 다시 포팅되지 않으므로 주기적으로 최신 메이저 버전으로 업그레이드하는 것이 좋습니다. 가 새 AWS 리전으로 가용성을 ElastiCache 확장함에 따라 ElastiCache (Redis OSS)는 해당 시점에 새 리전에 대한 두 개의 최신 MAJOR.MINOR 버전을 지원합니다. 예를 들어 새 AWS 리전이 시작되고 최신 가 시작되는 경우MAJORMINOR ElastiCache (Redis OSS) 버전은 7.0 및 6.2이며, ElastiCache (Redis OSS)는 새 AWS 리전에서 버전 7.0 및 6.2를 지원합니다. 최신 MAJOR.MINOR ElastiCache (Redis OSS) 버전이 릴리스되면 ElastiCache 는 새로 릴리스된 ElastiCache (Redis OSS) 버전에 대한 지원을 계속 추가합니다. 의 리전 선택에 대한 자세한 내용은

리전 및 가용 영역 선택을 ElastiCache참조하세요. https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/RegionsAndAZs.html#SupportedRegions

메이저 버전 또는 마이너 버전에 걸친 업그레이드를 수행할 때는 시간 OSS 경과에 따라 Redis와 릴리스된 동작 및 이전 버전과 호환되지 않는 변경 사항이 포함된 다음 목록을 고려하세요.

Redis OSS 7.0 동작 및 이전 버전과 호환되지 않는 변경 사항

전체 변경 목록은 Redis OSS 7.0 릴리스 정보 섹션을 참조하세요.

- `SCRIPT LOAD` 및 `SCRIPT FLUSH`는 더 이상 복제본으로 전파되지 않습니다. 스크립트에 약간의 내구성이 필요한 경우 Redis OSS 함수 사용을 고려하는 것이 좋습니다.
- 이제 Pubsub 채널이 새 ACL 사용자에 대해 기본적으로 차단됩니다.
- `STRALGO` 명령이 `LCS` 명령으로 대체되었습니다.
- `ACL GETUSER`의 형식이 변경되어 모든 필드에 표준 액세스 문자열 패턴이 표시됩니다. `ACL GETUSER`를 사용하여 자동화한 경우, 두 형식 중 하나가 처리되는지 검증해야 합니다.
- `SELECT`, , `WAIT`, `ROLE`, `LASTSAVE`, `READWRITE`, 및 `READONLY`의 ACL 범주`ASKING`가 변경되었습니다.
- 이제 `INFO` 명령은 최상위 컨테이너 명령 대신 하위 명령별 명령 통계를 표시합니다.
- 특정 엣지 상황에서 `LPOP`, `RPOP`, `ZPOPMIN`, `ZPOPMAX` 명령의 반환 값이 변경되었습니다. 이들 명령을 사용한다면, 릴리스 정보를 확인하고 영향이 있는지 평가해야 합니다.
- 이제 `SORT` 및 `SORT_RO` 명령이 `GET` 및 `BY` 인수를 사용하려면 키스페이스 전체에 액세스할 수 있어야 합니다.

Redis OSS 6.2 동작 및 이전 버전과 호환되지 않는 변경 사항

전체 변경 사항은 Redis OSS 6.2 릴리스 정보 섹션을 참조하세요.

- `TIME`, `ECHO`, `ROLE`및 `LASTSAVE` 명령의 ACL 플래그가 변경되었습니다. 이로 인해 이전에 허용된 명령이 거부될 수 있으며 그 반대의 경우도 마찬가지입니다.

> **Note**
>
> 이러한 명령 중 어느 것도 데이터를 수정하거나 액세스 권한을 부여하지 않습니다.

- Redis OSS 6.0에서 업그레이드할 때 맵 응답에서 루아 스크립트로 반환되는 키/값 페어의 순서가 변경됩니다. 스크립트가 맵(Redis OSS 6.0의 새 맵)을 사용하거나 `redis.setresp()` 반환하는 경우 업그레이드 시 스크립트가 중단될 수 있는 영향을 고려합니다.

Redis OSS 6.0 동작 및 이전 버전과 호환되지 않는 변경 사항

전체 변경 사항은 Redis OSS 6.0 릴리스 정보 섹션을 참조하세요.

- 허용되는 최대 데이터베이스 수가 120만 개에서 1만 개로 감소했습니다. 기본값은 16입니다. 성능 및 메모리 문제가 발견되었으므로 이보다 훨씬 큰 값은 사용하지 않는 것이 좋습니다.
- `AutoMinorVersionUpgrade` 파라미터를 예로 설정하면 ElastiCache (Redis OSS)가 셀프 서비스 업데이트를 통해 마이너 버전 업그레이드를 관리합니다. 이러한 관리는 셀프 서비스 업데이트 캠페인을 사용하는 표준 고객 알림 채널을 통해 처리됩니다. 자세한 내용은 의 셀프 서비스 업데이트를 ElastiCache 참조하세요.

Redis OSS 5.0 동작 및 이전 버전과 호환되지 않는 변경 사항

전체 변경 목록은 Redis OSS 5.0 릴리스 정보 섹션을 참조하세요.

- 스크립트는 복제본에서 스크립트를 다시 실행하지 않고 효과에 의해 복제됩니다. 이렇게 하면 일반적으로 성능이 향상되지만 기본 및 복제본 간에 복제된 데이터의 양이 증가할 수 있습니다. ElastiCache (Redis OSS) 5.0에서만 사용할 수 있는 이전 동작으로 되돌릴 수 있는 옵션이 있습니다.
- Redis OSS 4.0에서 업그레이드하는 경우 LUA 스크립트의 일부 명령은 이전 버전과 다른 순서로 인수를 반환합니다. Redis OSS 4.0에서 RedisOSS는 응답을 결정적으로 만들기 위해 일부 응답을 어휘로 정렬합니다. 스크립트가 효과로 복제될 때는 이 순서가 적용되지 않습니다.
- Inn Redis OSS 5.0.3 이상 ElastiCache (Redis OSS)은 4개 이상의 인스턴스 유형의 백그라운드 코어로 일부 IO 작업을 오프로드합니다VCPUs. 이렇게 하면 성능 특성 Redis가 변경OSS되고 일부 지표의 값이 변경될 수 있습니다. 자세한 정보는 어떤 지표를 모니터링해야 합니까? 섹션을 참조하여 지켜보는 지표를 변경해야 할 경우를 이해하세요.

Redis OSS 4.0 동작 및 이전 버전과 호환되지 않는 변경 사항

전체 변경 사항은 Redis OSS 4.0 릴리스 정보 섹션을 참조하세요.

- 슬로우 로그는 이제 클라이언트 이름과 주소라는 두 개의 인수를 추가로 기록합니다. 이 변경 사항은 3개의 값을 포함하는 각 슬로우 로그 항목에 명시적으로 의존하지 않는 한 이전 버전과 호환되어야 합니다.

- CLUSTER NODES 명령은 이제 약간 다른 형식을 반환하는데 이는 이전 버전과 호환되지 않습니다. 클라이언트는 클러스터에 있는 노드에 대해 알기 위해 이 명령을 사용하지 않는 것이 좋습니다. 대신 CLUSTER SLOTS를 사용해야 합니다.

과거 EOL

Redis OSS 3.2 동작 및 이전 버전과 호환되지 않는 변경 사항

전체 변경 사항은 Redis OSS 3.2 릴리스 정보 섹션을 참조하세요.

- 이 버전에 대해 호출할 호환성 변경 사항은 없습니다.

자세한 내용은 Redis OSS 버전 수명 종료 일정 단원을 참조하십시오.

Redis OSS 2.8 동작 및 이전 버전과 호환되지 않는 변경 사항

전체 변경 사항은 Redis OSS 2.8 릴리스 정보 섹션을 참조하세요.

- Redis OSS 2.8.22부터 RedisOSSAOF는 더 이상 ElastiCache (Redis OSS)에서 지원되지 않습니다. 데이터를 안정적으로 유지해야 하는 경우 MemoryDB를 사용하는 것이 좋습니다.
- Redis OSS 2.8.22부터 ElastiCache (Redis OSS)는 더 이상 내에서 호스팅되는 프라이머리에 복제본 연결을 지원하지 않습니다 ElastiCache. 업그레이드하는 동안 외부 복제본의 연결이 끊어지고 다시 연결할 수 없게 됩니다. 외부 복제본 대신 Redis OSS 6.0에서 사용할 수 있는 클라이언트 측 캐싱을 사용하는 것이 좋습니다.
- TTL 및 PTTL 명령은 이제 키가 있지 않은 경우 -2를 반환하고, 키는 있지만 관련 만료 기간이 없으면 -1을 반환합니다. Redis OSS 2.6 및 이전 버전은 두 조건 모두에 대해 -1을 반환하는 데 사용됩니다.
- ALPHA를 사용하는 SORT는 이제 STORE 옵션이 사용되지 않는 경우 로컬 데이터 정렬 로캘에 따라 정렬합니다.

자세한 내용은 Redis OSS 버전 수명 종료 일정 단원을 참조하십시오.

차단된 Valkey 또는 Redis OSS 엔진 업그레이드 해결

다음 표와 같이 스케일 업 작업이 보류 중인 경우 Valkey 또는 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

차단된 Redis OSS 엔진 업그레이드를 해결하려면

- 다음 중 하나를 수행합니다.
  - 즉시 적용 확인란을 선택 취소하여 다음 유지 관리 기간에 대한 Redis OSS 엔진 업그레이드 작업을 예약합니다.

    에서 를 CLI사용합니다`--no-apply-immediately`. 에서 를 API사용합니다`ApplyImmediately=false`.
  - 다음 유지 관리 기간(또는 이후)까지 기다렸다가 Redis OSS 엔진 업그레이드 작업을 수행합니다.
  - 즉시 적용 확인란을 선택한 상태에서 이 클러스터 수정에 Redis OSS 스케일 업 작업을 추가합니다.

    에서 를 CLI사용합니다`--apply-immediately`. 에서 를 API사용합니다`ApplyImmediately=true`.

    이러한 접근 방식에서는 이를 즉시 수행하여 다음 유지 관리 기간 동안 엔진 업그레이드를 효과적으로 취소합니다.

ElastiCache 모범 사례 및 캐싱 전략

아래에서 Amazon 에 대한 권장 모범 사례를 찾을 수 있습니다 ElastiCache. 다음 모범 사례를 준수하면 클러스터의 성능과 신뢰성을 향상시킬 수 있습니다.

주제

- 지원 및 제한된 Valkey, Redis OSS 및 Memcached 명령
- Valkey 및 Redis OSS 구성 및 제한
- IPv6 Valkey, Redis OSS 및 Memcached에 대한 클라이언트 예제
- 클라이언트 모범 사례(Valkey 및 RedisOSS)
- 클라이언트 모범 사례(Memcached)
- TLS 활성화된 듀얼 스택 ElastiCache 클러스터
- Valkey 및 Redis용 예약 메모리 관리 OSS
- Valkey 및 Redis OSS 자체 설계 클러스터 작업 모범 사례
- Memcached에 대한 캐싱 전략

전체 모범 사례

아래에서 에서 Valkey, Redis OSS 및 Memcached 인터페이스를 사용하기 위한 모범 사례에 대한 정보를 찾을 수 있습니다 ElastiCache.

- 클러스터 모드 지원 구성 사용 - 클러스터 모드 활성화를 사용하면 캐시를 수평으로 확장하여 클러스터 모드 비활성화 구성보다 더 높은 스토리지 및 처리량을 달성할 수 있습니다. ElastiCache 서버리스는 클러스터 모드 활성화 구성에서만 사용할 수 있습니다.
- 장기 연결 사용 - 새 연결을 생성하는 데 비용이 많이 들고 캐시에서 시간과 CPU 리소스가 걸립니다. 가능하면 연결을 재사용하여(예: 연결 풀링 사용) 여러 명령에서 발생하는 이 비용을 분할 상환합니다.
- 복제본에서 읽기 - ElastiCache 서버리스를 사용 중이거나 읽기 전용 복제본(자체 설계 클러스터)을 프로비저닝한 경우 읽기를 복제본으로 직접 전달하여 확장성을 높이고 지연 시간을 줄입니다. 복제본에서의 읽기는 최종적으로 프라이머리와 일치합니다.

  자체 설계된 클러스터의 경우 노드에 장애가 발생하면 일시적으로 읽기가 불가능할 수 있으므로 읽기 요청을 단일 읽기 전용 복제본으로 보내지 않도록 합니다. 적어도 2개 이상의 읽기 전용 복제복으로 읽기 요청을 보내거나, 단일 복제복과 프라이머리로 읽기 전용 복제복으로 보내도록 클라이언트를 구성할 수 있습니다.

  ElastiCache 서버리스에서는 복제본 포트(6380)에서 읽기를 수행하면 가능하면 읽기가 클라이언트의 로컬 가용 영역으로 전달되어 검색 지연 시간이 줄어듭니다. 장애가 발생하면 자동으로 다른 노드로 대체됩니다.

- 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.
- 샤딩된 pub/sub 사용 - 처리량이 높은 pub/sub 워크로드를 지원하기 OSS 위해 Valkey 또는 Redis를 사용하는 경우 샤딩된 pub/sub(Valkey 및 Redis OSS 7 이상에서 사용 가능)를 사용하는 것이 좋습니다. 클러스터 모드가 활성화된 클러스터의 기존 pub/sub는 클러스터의 모든 노드에 메시지를 브로드캐스트하므로 높은 EngineCPUUtilization 결과가 발생할 수 있습니다. ElastiCache 서버리스에서 기존 pub/sub 명령은 내부적으로 샤딩된 pub/sub 명령을 사용합니다.

지원 및 제한된 Valkey, Redis OSS 및 Memcached 명령

지원되는 Valkey 및 Redis OSS 명령

지원되는 Valkey 및 Redis OSS 명령

다음 Valkey 및 Redis OSS 명령은 서버리스 캐시에서 지원됩니다. 이러한 명령 외에도 이러한 지원되는 Valkey 및 Redis OSS 명령 명령도 지원됩니다.

비트맵 명령

- BITCOUNT

  문자열에 설정된 비트 수(인구 수 계산)를 계산합니다.

  자세히 알아보기
- BITFIELD

  문자열에 대해 임의의 비트필드 정수 연산을 수행합니다.

  자세히 알아보기
- BITFIELD_RO

  문자열에 대해 임의의 읽기 전용 비트필드 정수 연산을 수행합니다.

  자세히 알아보기

- BITOP

  여러 문자열에 대해 비트 논리곱 연산을 수행하고 결과를 저장합니다.

  자세히 알아보기

- BITPOS

  문자열에서 첫 번째 세트(1) 또는 클리어(0) 비트를 찾습니다.

  자세히 알아보기

- GETBIT

  오프셋을 기준으로 비트 값을 반환합니다.

  자세히 알아보기

- SETBIT

  문자열 값의 오프셋에서 비트를 설정하거나 지웁니다. 존재하지 않으면 키를 생성합니다.

  자세히 알아보기

클러스터 관리 명령

- CLUSTER COUNTKEYSINSLOT

  해시 슬롯의 키 수를 반환합니다.

  자세히 알아보기

- CLUSTER GETKEYSINSLOT

  해시 슬롯의 키 이름을 반환합니다.

  자세히 알아보기

- CLUSTER INFO

  노드 상태에 대한 정보를 반환합니다. 서버리스 캐시에서는 클라이언트에 노출된 단일 가상 '샤드'에 대한 상태를 반환합니다.

  자세히 알아보기

- CLUSTER KEYSLOT

키의 해시 슬롯을 반환합니다.

자세히 알아보기

- CLUSTER MYID

  노드의 ID를 반환합니다. 서버리스 캐시에서는 클라이언트에 노출된 단일 가상 '샤드'에 대한 상태를 반환합니다.

  자세히 알아보기

- CLUSTER NODES

  노드의 클러스터 구성을 반환합니다. 서버리스 캐시에서는 클라이언트에 노출된 단일 가상 '샤드'에 대한 상태를 반환합니다.

  자세히 알아보기

- CLUSTER REPLICAS

  프라이머리 노드의 복제 노드를 나열합니다. 서버리스 캐시에서는 클라이언트에 노출된 단일 가상 '샤드'에 대한 상태를 반환합니다.

  자세히 알아보기

- CLUSTER SHARDS

  클러스터 슬롯의 매핑을 샤드에 반환합니다. 서버리스 캐시에서는 클라이언트에 노출된 단일 가상 '샤드'에 대한 상태를 반환합니다.

  자세히 알아보기

- CLUSTER SLOTS

  클러스터 슬롯의 매핑을 노드에 반환합니다. 서버리스 캐시에서는 클라이언트에 노출된 단일 가상 '샤드'에 대한 상태를 반환합니다.

  자세히 알아보기

- READONLY

  Valkey 또는 Redis OSS 클러스터 복제본 노드에 대한 연결에 대해 읽기 전용 쿼리를 활성화합니다.

  자세히 알아보기

- READWRITE

Valkey 또는 Redis OSS 클러스터 복제본 노드에 대한 연결에 대해 읽기-쓰기 쿼리를 활성화합니다.

자세히 알아보기

연결 관리 명령

- AUTH

  연결을 인증합니다.

  자세히 알아보기
- CLIENT GETNAME

  연결 이름을 반환합니다.

  자세히 알아보기
- CLIENT REPLY

  명령에 응답할지 여부를 서버에 지시합니다.

  자세히 알아보기
- CLIENT SETNAME

  연결 이름을 설정합니다.

  자세히 알아보기
- ECHO

  주어진 문자열을 반환합니다.

  자세히 알아보기
- HELLO

  Valkey 또는 Redis OSS 서버를 사용한 핸드셰이크.

  자세히 알아보기
- PING

  서버의 활성 응답을 반환합니다.

자세히 알아보기

- QUIT

  연결을 종료합니다.

  자세히 알아보기

- RESET

  연결을 초기화합니다.

  자세히 알아보기

- SELECT

  선택한 데이터베이스를 변경합니다.

  자세히 알아보기

일반 명령

- COPY

  키 값을 새 키로 복사합니다.

  자세히 알아보기

- DEL

  하나 이상의 키를 삭제합니다.

  자세히 알아보기

- DUMP

  키에 저장된 값의 직렬화된 표현을 반환합니다.

  자세히 알아보기

- EXISTS

  키가 하나 이상 존재하는지 확인합니다.

  자세히 알아보기

- EXPIRE

  키의 만료 시간(초 기준)을 설정합니다.

  자세히 알아보기

- EXPIREAT

  키의 만료 시간을 Unix 타임스탬프로 설정합니다.

  자세히 알아보기

- EXPIRETIME

  키의 만료 시간을 Unix 타임스탬프로 반환합니다.

  자세히 알아보기

- PERSIST

  키의 만료 시간을 제거합니다.

  자세히 알아보기

- PEXPIRE

  키의 만료 시간(밀리초 기준)을 설정합니다.

  자세히 알아보기

- PEXPIREAT

  키의 만료 시간을 Unix 밀리초 타임스탬프로 설정합니다.

  자세히 알아보기

- PEXPIRETIME

  키의 만료 시간을 Unix 밀리초 타임스탬프로 반환합니다.

  자세히 알아보기

- PTTL

  키의 만료 시간(밀리초 기준)을 반환합니다.

  자세히 알아보기

- RANDOMKEY

  데이터베이스에서 임의의 키 이름을 반환합니다.

  자세히 알아보기

- RENAME

  키 이름을 바꾸고 대상을 덮어씁니다.

  자세히 알아보기

- RENAMENX

  대상 키 이름이 존재하지 않는 경우에만 키 이름을 변경합니다.

  자세히 알아보기

- RESTORE

  직렬화된 값 표현에서 키를 만듭니다.

  자세히 알아보기

- SCAN

  데이터베이스의 키 이름을 반복합니다.

  자세히 알아보기

- SORT

  목록, 집합 또는 정렬된 집합의 요소를 정렬하고 필요에 따라 결과를 저장합니다.

  자세히 알아보기

- SORT_RO

  목록, 집합 또는 정렬된 집합의 정렬된 요소를 반환합니다.

  자세히 알아보기

- TOUCH

  마지막으로 액세스한 시각을 업데이트한 후 지정된 키 중에서 기존 키의 수를 반환합니다.

  자세히 알아보기

- TTL

  키의 만료 시간(초 기준)을 반환합니다.

  자세히 알아보기

- TYPE

  키에 저장된 값의 유형을 결정합니다.

  자세히 알아보기

- UNLINK

  하나 이상의 키를 비동기적으로 삭제합니다.

  자세히 알아보기

지리공간 명령

- GEOADD

  지리공간 인덱스에 하나 이상의 멤버를 추가합니다. 키가 존재하지 않으면 생성됩니다.

  자세히 알아보기

- GEODIST

  지리공간 인덱스의 두 멤버 간 거리를 반환합니다.

  자세히 알아보기

- GEOHASH

  지리공간 인덱스의 멤버를 geohash 문자열로 반환합니다.

  자세히 알아보기

- GEOPOS

  지리공간 인덱스에서 멤버의 경도와 위도를 반환합니다.

  자세히 알아보기

- GEORADIUS

좌표로부터 거리 이내에 있는 멤버의 지리공간 인덱스를 쿼리하고 결과를 필요에 따라 저장합니다.

자세히 알아보기

- GEORADIUS_RO

  좌표로부터 거리 이내에 있는 지리공간 인덱스에서 멤버를 반환합니다.

  자세히 알아보기

- GEORADIUSBYMEMBER

  멤버의 거리 이내에 있는 멤버에 대한 지리공간 인덱스를 쿼리하고 결과를 필요에 따라 저장합니다.

  자세히 알아보기

- GEORADIUSBYMEMBER_RO

  멤버의 거리 이내에 있는 지리공간 인덱스에서 멤버를 반환합니다.

  자세히 알아보기

- GEOSEARCH

  상자 또는 원 영역 안에 있는 멤버에 대한 지리공간 인덱스를 쿼리합니다.

  자세히 알아보기

- GEOSEARCHSTORE

  상자 또는 원 영역 안에 있는 멤버에 대한 지리공간 인덱스를 쿼리하고, 필요에 따라 결과를 저장합니다.

  자세히 알아보기

해시 명령

- HDEL

  해시에서 하나 이상의 필드와 해당 값을 삭제합니다. 남아 있는 필드가 없으면 해시를 삭제합니다.

  자세히 알아보기

- HEXISTS

필드가 해시에 존재하는지 여부를 결정합니다.

자세히 알아보기

- HGET

  해시의 필드 값을 반환합니다.

  자세히 알아보기

- HGETALL

  해시의 모든 필드와 값을 반환합니다.

  자세히 알아보기

- HINCRBY

  해시에 있는 필드의 정수 값을 숫자만큼 증가시킵니다. 필드가 존재하지 않는 경우 0을 초기값으로 사용합니다.

  자세히 알아보기

- HINCRBYFLOAT

  필드의 부동 소수점 값을 숫자만큼 증가시킵니다. 필드가 존재하지 않는 경우 0을 초기값으로 사용합니다.

  자세히 알아보기

- HKEYS

  해시의 모든 필드를 반환합니다.

  자세히 알아보기

- HLEN

  해시의 필드 수를 반환합니다.

  자세히 알아보기

- HMGET

  해시의 모든 필드 값을 반환합니다.

자세히 알아보기

- HMSET

  여러 필드의 값을 설정합니다.

  자세히 알아보기

- HRANDFIELD

  해시에서 하나 이상의 임의 필드를 반환합니다.

  자세히 알아보기

- HSCAN

  해시의 필드와 값을 반복합니다.

  자세히 알아보기

- HSET

  해시에서 필드 값을 만들거나 수정합니다.

  자세히 알아보기

- HSETNX

  필드가 존재하지 않는 경우에만 해시의 필드 값을 설정합니다.

  자세히 알아보기

- HSTRLEN

  필드 값의 길이를 반환합니다.

  자세히 알아보기

- HVALS

  해시의 모든 값을 반환합니다.

  자세히 알아보기

HyperLogLog 명령

- PFADD

  HyperLogLog 키에 요소를 추가합니다. 존재하지 않으면 키를 생성합니다.

  자세히 알아보기

- PFCOUNT

  HyperLogLog 키(들)에서 관찰한 세트(들)의 근사 카디널리티를 반환합니다.

  자세히 알아보기

- PFMERGE

  하나 이상의 HyperLogLog 값을 단일 키로 병합합니다.

  자세히 알아보기

목록 명령

- BLMOVE

  목록에서 요소를 가져와 다른 목록으로 푸시한 다음 반환합니다. 다른 방법으로 요소를 사용할 수 있을 때까지 차단합니다. 마지막 요소가 이동된 경우 목록을 삭제합니다.

  자세히 알아보기

- BLMPOP

  여러 목록 중 하나에서 첫 번째 요소를 팝업합니다. 다른 방법으로 요소를 사용할 수 있을 때까지 차단합니다. 마지막 요소가 팝업된 경우 목록을 삭제합니다.

  자세히 알아보기

- BLPOP

  목록에서 첫 번째 요소를 제거하고 반환합니다. 다른 방법으로 요소를 사용할 수 있을 때까지 차단합니다. 마지막 요소가 팝업된 경우 목록을 삭제합니다.

  자세히 알아보기

- BRPOP

  목록에서 마지막 요소를 제거하고 반환합니다. 다른 방법으로 요소를 사용할 수 있을 때까지 차단합니다. 마지막 요소가 팝업된 경우 목록을 삭제합니다.

자세히 알아보기

- BRPOPLPUSH

  목록에서 요소를 가져와 다른 목록으로 푸시한 다음 반환합니다. 다른 방법으로 요소를 사용할 수 있을 때까지 차단합니다. 마지막 요소가 팝업된 경우 목록을 삭제합니다.

  자세히 알아보기

- LINDEX

  인덱스를 기준으로 목록에서 요소를 반환합니다.

  자세히 알아보기

- LINSERT

  목록에서 다른 요소 앞 또는 뒤에 요소를 삽입합니다.

  자세히 알아보기

- LLEN

  목록의 길이를 반환합니다.

  자세히 알아보기

- LMOVE

  한 목록에서 요소를 가져와 다른 목록으로 푸시한 후 요소를 반환합니다. 마지막 요소가 이동된 경우 목록을 삭제합니다.

  자세히 알아보기

- LMPOP

  요소를 제거한 후 목록에서 여러 요소를 반환합니다. 마지막 요소가 팝업된 경우 목록을 삭제합니다.

  자세히 알아보기

- LPOP

  목록의 첫 번째 요소를 제거한 후 해당 요소를 반환합니다. 마지막 요소가 팝업된 경우 목록을 삭제합니다.

  자세히 알아보기

- LPOS

  목록에서 일치하는 요소의 인덱스를 반환합니다.

  자세히 알아보기

- LPUSH

  하나 이상의 요소를 목록 앞에 추가합니다. 존재하지 않으면 키를 생성합니다.

  자세히 알아보기

- LPUSHX

  목록이 있는 경우에만 목록 앞에 요소를 하나 이상 추가합니다.

  자세히 알아보기

- LRANGE

  목록에서 요소 범위를 반환합니다.

  자세히 알아보기

- LREM

  목록에서 요소를 제거합니다. 마지막 요소가 제거된 경우 목록을 삭제합니다.

  자세히 알아보기

- LSET

  인덱스를 기준으로 목록의 요소 값을 설정합니다.

  자세히 알아보기

- LTRIM

  목록의 양쪽 끝에서 요소를 제거합니다. 모든 요소가 잘린 경우 목록을 삭제합니다.

  자세히 알아보기

- RPOP

  목록에서 마지막 요소를 반환하고 제거합니다. 마지막 요소가 팝업된 경우 목록을 삭제합니다.

  자세히 알아보기

- RPOPLPUSH

  목록의 마지막 요소를 제거하고 다른 목록으로 푸시한 후 해당 요소를 반환합니다. 마지막 요소가 팝업된 경우 목록을 삭제합니다.

  자세히 알아보기

- RPUSH

  하나 이상의 요소를 목록 앞에 추가합니다. 존재하지 않으면 키를 생성합니다.

  자세히 알아보기

- RPUSHX

  목록이 있는 경우에만 목록에 요소를 추가합니다.

  자세히 알아보기

Pub/Sub 명령

> **Note**
>
> PUBSUB 명령은 내부적으로 샤딩된 를 사용PUBSUB하므로 채널 이름이 혼합됩니다.

- PUBLISH

  채널에 메시지를 게시합니다.

  자세히 알아보기

- PUBSUB CHANNELS

  활성 채널을 반환합니다.

  자세히 알아보기

- PUBSUB NUMSUB

  채널 구독자 수를 반환합니다.

  자세히 알아보기

- PUBSUB SHARDCHANNELS

활성 샤드 채널을 반환합니다.

PUBSUB-SHARDCHANNELS

- PUBSUB SHARDNUMSUB

  샤드 채널 구독자 수를 반환합니다.

  PUBSUB-SHARDNUMSUB

- SPUBLISH

  샤드 채널에 메시지를 게시합니다.

  자세히 알아보기

- SSUBSCRIBE

  샤드 채널에 게시된 메시지를 수신합니다.

  자세히 알아보기

- SUBSCRIBE

  채널에 게시된 메시지를 수신합니다.

  자세히 알아보기

- SUNSUBSCRIBE

  샤드 채널에 게시된 메시지 수신을 중단합니다.

  자세히 알아보기

- UNSUBSCRIBE

  채널에 게시된 메시지 수신을 중단합니다.

  자세히 알아보기

스크립팅 명령

- EVAL

  서버 측 Lua 스크립트를 실행합니다.

자세히 알아보기

- EVAL_RO

  읽기 전용 서버 측 Lua 스크립트를 실행합니다.

  자세히 알아보기

- EVALSHA

  다이제스트를 통해 서버 측 Lua 스크립트SHA1를 실행합니다.

  자세히 알아보기

- EVALSHA_RO

  다이제스트를 통해 읽기 전용 서버 측 Lua 스크립트SHA1를 실행합니다.

  자세히 알아보기

- SCRIPT EXISTS

  서버 측 Lua 스크립트가 스크립트 캐시에 존재하는지 여부를 결정합니다.

  자세히 알아보기

- SCRIPT FLUSH

  현재 운영되지 않는 스크립트 캐시는 서비스에서 관리합니다.

  자세히 알아보기

- SCRIPT LOAD

  서버 측 Lua 스크립트를 스크립트 캐시에 로드합니다.

  자세히 알아보기

서버 관리 명령

- ACL CAT

  ACL 범주 또는 범주 내 명령을 나열합니다.

  자세히 알아보기

- ACL GENPASS

  ACL 사용자를 식별하는 데 사용할 수 있는 가상의 보안 암호를 생성합니다.

  자세히 알아보기

- ACL GETUSER

  사용자의 ACL 규칙을 나열합니다.

  자세히 알아보기

- ACL LIST

  유효한 규칙을 ACL 파일 형식으로 덤프합니다.

  자세히 알아보기

- ACL USERS

  모든 ACL 사용자를 나열합니다.

  자세히 알아보기

- ACL WHOAMI

  현재 연결의 인증된 사용자 이름을 반환합니다.

  자세히 알아보기

- DBSIZE

  현재 선택한 데이터베이스의 키 수를 반환합니다. 이 작업이 모든 슬롯에서 세부적으로 수행된다고 보장할 수는 없습니다.

  자세히 알아보기

- COMMAND

  모든 명령에 대한 자세한 정보를 반환합니다.

  자세히 알아보기

- COMMAND COUNT

  명령 개수를 반환합니다.

자세히 알아보기

- COMMAND DOCS

  하나, 여러 개 또는 모든 명령에 대한 문서 정보를 반환합니다.

  자세히 알아보기

- COMMAND GETKEYS

  임의의 명령에서 키 이름을 추출합니다.

  자세히 알아보기

- COMMAND GETKEYSANDFLAGS

  임의 명령의 키 이름과 액세스 플래그를 추출합니다.

  자세히 알아보기

- COMMAND INFO

  하나, 여러 개 또는 모든 명령에 대한 정보를 반환합니다.

  자세히 알아보기

- COMMAND LIST

  명령 이름 목록을 반환합니다.

  자세히 알아보기

- FLUSHALL

  모든 데이터베이스에서 모든 키를 제거합니다. 이 작업이 모든 슬롯에서 세부적으로 수행된다고 보장할 수는 없습니다.

  자세히 알아보기

- FLUSHDB

  현재 데이터베이스에서 모든 키를 제거합니다. 이 작업이 모든 슬롯에서 세부적으로 수행된다고 보장할 수는 없습니다.

  자세히 알아보기

- INFO

서버에 대한 정보와 통계를 반환합니다.

자세히 알아보기

- LOLWUT

  컴퓨터 아트와 Valkey 또는 Redis OSS 버전을 표시합니다.

  자세히 알아보기

- ROLE

  복제 역할을 반환합니다.

  자세히 알아보기

- TIME

  서버 시각을 반환합니다.

  자세히 알아보기

설정 명령

- SADD

  세트에 하나 이상의 멤버를 추가합니다. 존재하지 않으면 키를 생성합니다.

  자세히 알아보기

- SCARDT

  세트에 멤버 수를 반환합니다.

  자세히 알아보기

- SDIFF

  여러 세트의 차이를 반환합니다.

  자세히 알아보기

- SDIFFSTORE

  여러 세트의 차이를 키에 저장합니다.

자세히 알아보기

- SINTER

  여러 세트의 교차점을 반환합니다.

  자세히 알아보기

- SINTERCARD

  여러 세트의 교차점에 있는 멤버 수를 반환합니다.

  자세히 알아보기

- SINTERSTORE

  여러 세트의 교차점을 키에 저장합니다.

  자세히 알아보기

- SISMEMBER

  멤버가 세트에 속하는지 여부를 결정합니다.

  자세히 알아보기

- SMEMBERS

  세트의 모든 멤버를 반환합니다.

  자세히 알아보기

- SMISMEMBER

  멤버가 세트에 속하는지 여부를 결정합니다.

  자세히 알아보기

- SMOVE

  한 세트에서 다른 세트로 멤버를 이동합니다.

  자세히 알아보기

- SPOP

하나 이상의 무작위 멤버를 제거한 후 세트에서 해당 멤버를 반환합니다. 마지막 멤버가 팝업된 경우 세트를 삭제합니다.

자세히 알아보기

- SRANDMEMBER

  세트에서 하나 또는 여러 개의 무작위 멤버를 가져옵니다.

  자세히 알아보기

- SREM

  세트에서 하나 이상의 멤버를 제거합니다. 마지막 멤버가 제거된 경우 세트를 삭제합니다.

  자세히 알아보기

- SSCAN

  세트의 멤버를 반복합니다.

  자세히 알아보기

- SUNION

  여러 세트의 결합을 반환합니다.

  자세히 알아보기

- SUNIONSTORE

  여러 세트의 결합을 키에 저장합니다.

  자세히 알아보기

정렬된 세트 명령

- BZMPOP

  하나 이상의 정렬된 세트에서 점수별로 멤버를 제거하고 반환합니다. 다른 방법으로 멤버를 사용할 수 있을 때까지 차단합니다. 마지막 요소가 팝업된 경우 정렬된 세트를 삭제합니다.

  자세히 알아보기

- BZPOPMAX

하나 이상의 정렬된 세트에서 높은 점수별로 멤버를 제거하고 반환합니다. 다른 방법으로 멤버를 사용할 수 있을 때까지 차단합니다. 마지막 요소가 팝업된 경우 정렬된 세트를 삭제합니다.

자세히 알아보기

- BZPOPMIN

  하나 이상의 정렬된 세트에서 낮은 점수별로 멤버를 제거하고 반환합니다. 다른 방법으로 멤버를 사용할 수 있을 때까지 차단합니다. 마지막 요소가 팝업된 경우 정렬된 세트를 삭제합니다.

  자세히 알아보기

- ZADD

  정렬된 세트에 하나 이상의 멤버를 추가하거나 멤버의 점수를 업데이트합니다. 존재하지 않으면 키를 생성합니다.

  자세히 알아보기

- ZCARD

  정렬된 세트에서 멤버 수를 반환합니다.

  자세히 알아보기

- ZCOUNT

  일정 범위 내에 점수가 있는 정렬된 세트의 멤버 수를 반환합니다.

  자세히 알아보기

- ZDIFF

  여러 정렬된 세트의 차이를 반환합니다.

  자세히 알아보기

- ZDIFFSTORE

  여러 정렬된 세트의 차이를 키에 저장합니다.

  자세히 알아보기

- ZINCRBY

  정렬된 세트에 있는 멤버의 점수를 증가시킵니다.

자세히 알아보기

- ZINTER

  여러 정렬된 세트의 교차점을 반환합니다.

  자세히 알아보기

- ZINTERCARD

  여러 정렬된 세트의 교차점에 있는 멤버 수를 반환합니다.

  자세히 알아보기

- ZINTERSTORE

  여러 정렬된 세트의 교차점을 키에 저장합니다.

  자세히 알아보기

- ZLEXCOUNT

  사전 범위 내에 있는 정렬된 세트의 멤버 수를 반환합니다.

  자세히 알아보기

- ZMPOP

  하나 이상의 정렬된 세트에서 가장 높은 점수 또는 가장 낮은 점수를 받은 멤버를 제거한 후 해당 멤버를 반환합니다. 마지막 멤버가 팝업된 경우 정렬된 세트를 삭제합니다.

  자세히 알아보기

- ZMSCORE

  정렬된 세트에 있는 하나 이상의 멤버 점수를 반환합니다.

  자세히 알아보기

- ZPOPMAX

  가장 높은 점수를 받은 멤버를 제거한 후 정렬된 세트에서 해당 멤버를 반환합니다. 마지막 멤버가 팝업된 경우 정렬된 세트를 삭제합니다.

  자세히 알아보기

- ZPOPMIN

가장 낮은 점수를 받은 멤버를 제거한 후 정렬된 세트에서 해당 멤버를 반환합니다. 마지막 멤버가 팝업된 경우 정렬된 세트를 삭제합니다.

자세히 알아보기

- ZRANDMEMBER

  정렬된 세트에서 하나 이상의 임의 멤버를 반환합니다.

  자세히 알아보기

- ZRANGE

  인덱스 범위 내에 있는 정렬된 세트의 멤버를 반환합니다.

  자세히 알아보기

- ZRANGEBYLEX

  사전 범위 내에 있는 정렬된 세트의 멤버를 반환합니다.

  자세히 알아보기

- ZRANGEBYSCORE

  점수 범위 내에 있는 정렬된 세트의 멤버를 반환합니다.

  자세히 알아보기

- ZRANGESTORE

  정렬된 세트의 멤버 범위를 키에 저장합니다.

  자세히 알아보기

- ZRANK

  오름차순 점수를 기준으로 정렬된 세트의 멤버 인덱스를 반환합니다.

  자세히 알아보기

- ZREM

  정렬된 세트에서 하나 이상의 멤버를 제거합니다. 모든 멤버가 제거된 경우 정렬된 세트를 삭제합니다.

자세히 알아보기

- ZREMRANGEBYLEX

  사전 범위 내에 있는 정렬된 세트의 멤버를 제거합니다. 모든 멤버가 제거된 경우 정렬된 세트를 삭제합니다.

  자세히 알아보기

- ZREMRANGEBYRANK

  인덱스 범위 내에 있는 정렬된 세트의 멤버를 제거합니다. 모든 멤버가 제거된 경우 정렬된 세트를 삭제합니다.

  자세히 알아보기

- ZREMRANGEBYSCORE

  점수 범위 내에 있는 정렬된 세트의 멤버를 제거합니다. 모든 멤버가 제거된 경우 정렬된 세트를 삭제합니다.

  자세히 알아보기

- ZREVRANGE

  인덱스 범위 내에 있는 정렬된 세트의 멤버를 역순으로 반환합니다.

  자세히 알아보기

- ZREVRANGEBYLEX

  사전 범위 내에 있는 정렬된 세트의 멤버를 역순으로 반환합니다.

  자세히 알아보기

- ZREVRANGEBYSCORE

  점수 범위 내에 있는 정렬된 세트의 멤버를 역순으로 반환합니다.

  자세히 알아보기

- ZREVRANK

  내림차순 점수를 기준으로 정렬된 세트의 멤버 인덱스를 반환합니다.

  자세히 알아보기

- ZSCAN

  정렬된 세트의 멤버와 점수를 반복합니다.

  자세히 알아보기

- ZSCORE

  정렬된 세트에 있는 멤버의 점수를 반환합니다.

  자세히 알아보기

- ZUNION

  여러 정렬된 세트의 결합을 반환합니다.

  자세히 알아보기

- ZUNIONSTORE

  여러 정렬된 세트의 결합을 키에 저장합니다.

  자세히 알아보기

스트림 명령

- XACK

  스트림의 소비자 그룹 멤버가 성공적으로 확인한 메시지 수를 반환합니다.

  자세히 알아보기

- XADD

  스트림에 새 메시지를 추가합니다. 존재하지 않으면 키를 생성합니다.

  자세히 알아보기

- XAUTOCLAIM

  메시지가 소비자 그룹 멤버로 전달된 것처럼 소비자 그룹의 메시지 소유권을 변경하거나 획득합니다.

  자세히 알아보기

- XCLAIM

메시지가 소비자 그룹 멤버로 전달된 것처럼 소비자 그룹의 메시지 소유권을 변경하거나 획득합니다.

자세히 알아보기

- XDEL

  스트림에서 메시지를 제거한 후 메시지 수를 반환합니다.

  자세히 알아보기

- XGROUP CREATE

  소비자 그룹을 생성합니다.

  자세히 알아보기

- XGROUP CREATECONSUMER

  소비자 그룹에 소비자를 생성합니다.

  자세히 알아보기

- XGROUP DELCONSUMER

  소비자 그룹에서 소비자를 삭제합니다.

  자세히 알아보기

- XGROUP DESTROY

  소비자 그룹을 제거합니다.

  자세히 알아보기

- XGROUP SETID

  소비자 그룹에 마지막으로 전달된 ID를 설정합니다.

  자세히 알아보기

- XINFO CONSUMERS

  소비자 그룹의 소비자 목록을 반환합니다.

  자세히 알아보기

- XINFO GROUPS

  스트림의 소비자 그룹 목록을 반환합니다.

  자세히 알아보기

- XINFO STREAM

  스트림에 대한 정보를 반환합니다.

  자세히 알아보기

- XLEN

  스트림의 메시지 수를 반환합니다.

  자세히 알아보기

- XPENDING

  스트림 소비자 그룹의 보류 중인 항목 목록에서 정보와 항목을 반환합니다.

  자세히 알아보기

- XRANGE

  범위 내의 스트림에서 메시지를 반환합니다IDs.

  자세히 알아보기

- XREAD

  요청된 것보다 IDs 큰 여러 스트림의 메시지를 반환합니다. 다른 방법으로 메시지를 사용할 수 있을 때까지 차단합니다.

  자세히 알아보기

- XREADGROUP

  스트림에서 그룹 내 소비자에게 새 메시지 또는 과거 메시지를 반환합니다. 다른 방법으로 메시지를 사용할 수 있을 때까지 차단합니다.

  자세히 알아보기

- XREVRANGE

  의 범위 내에서 스트림의 메시지를 IDs 역순으로 반환합니다.

자세히 알아보기

- XTRIM

  스트림의 시작 부분부터 메시지를 삭제합니다.

  자세히 알아보기

문자열 명령

- APPEND

  키 값에 문자열을 추가합니다. 존재하지 않으면 키를 생성합니다.

  자세히 알아보기

- DECR

  키의 정수 값을 1씩 줄입니다. 키가 존재하지 않는 경우 0을 초기값으로 사용합니다.

  자세히 알아보기

- DECRBY

  키의 정수 값에서 숫자를 줄입니다. 키가 존재하지 않는 경우 0을 초기값으로 사용합니다.

  자세히 알아보기

- GET

  키의 문자열 값을 반환합니다.

  자세히 알아보기

- GETDEL

  키를 삭제한 후 키의 문자열 값을 반환합니다.

  자세히 알아보기

- GETEX

  만료 시각을 설정한 후 키의 문자열 값을 반환합니다.

  자세히 알아보기

- GETRANGE

  키에 저장된 문자열의 하위 문자열을 반환합니다.

  자세히 알아보기

- GETSET

  키를 새 값으로 설정한 후 키의 이전 문자열 값을 반환합니다.

  자세히 알아보기

- INCR

  키의 정수 값을 1씩 증가시킵니다. 키가 존재하지 않는 경우 0을 초기값으로 사용합니다.

  자세히 알아보기

- INCRBY

  키의 정수 값을 숫자만큼 증가시킵니다. 키가 존재하지 않는 경우 0을 초기값으로 사용합니다.

  자세히 알아보기

- INCRBYFLOAT

  필드의 부동 소수점 값을 숫자만큼 증가시킵니다. 키가 존재하지 않는 경우 0을 초기값으로 사용합니다.

  자세히 알아보기

- LCS

  가장 긴 공통 하위 문자열을 찾습니다.

  자세히 알아보기

- MGET

  하나 이상인 키의 문자열 값을 세부적으로 반환합니다.

  자세히 알아보기

- MSET

  하나 이상인 키의 문자열 값을 세부적으로 생성 또는 수정합니다.

자세히 알아보기

- MSETNX

  모든 키가 존재하지 않는 경우에만 하나 이상인 키의 문자열 값을 세부적으로 수정합니다.

  자세히 알아보기

- PSETEX

  키의 문자열 값과 만료 시각(밀리초 기준)을 모두 설정합니다. 키가 존재하지 않으면 생성됩니다.

  자세히 알아보기

- SET

  유형을 무시하고 키의 문자열 값을 설정합니다. 키가 존재하지 않으면 생성됩니다.

  자세히 알아보기

- SETEX

  키의 문자열 값과 만료 시각을 설정합니다. 존재하지 않으면 키를 생성합니다.

  자세히 알아보기

- SETNX

  키가 존재하지 않는 경우에만 키의 문자열 값을 설정합니다.

  자세히 알아보기

- SETRANGE

  문자열 값의 일부를 오프셋에서 다른 값으로 덮어씁니다. 존재하지 않으면 키를 생성합니다.

  자세히 알아보기

- STRLEN

  문자열 값의 길이를 반환합니다.

  자세히 알아보기

- SUBSTR

  문자열 값에서 하위 문자열을 반환합니다.

자세히 알아보기

트랜잭션 명령

- DISCARD

  트랜잭션을 폐기합니다.

  자세히 알아보기
- EXEC

  트랜잭션의 모든 명령을 실행합니다.

  자세히 알아보기
- MULTI

  트랜잭션을 시작합니다.

  자세히 알아보기

제한된 Valkey 및 Redis OSS 명령

관리형 서비스 경험을 제공하기 위해 는 고급 권한이 필요한 특정 캐시 엔진별 명령에 대한 액세스를 ElastiCache 제한합니다. Redis를 실행하는 캐시의 경우 다음 명령을 사용할 수 없습니다.

- acl setuser
- acl load
- acl save
- acl deluser
- bgrewriteaof
- bgsave
- cluster addslot
- cluster addslotsrange
- cluster bumpepoch
- cluster delslot
- cluster delslotsrange

- cluster failover
- cluster flushslots
- cluster forget
- cluster links
- cluster meet
- cluster setslot
- config
- debug
- migrate
- psync
- replicaof
- save
- slaveof
- shutdown
- sync

또한 서버리스 캐시에는 다음 명령을 사용할 수 없습니다.

- acl log
- client caching
- client getredir
- client id
- client info
- client kill
- client list
- client no-evict
- client pause
- client tracking
- client trackinginfo
- client unblock

- client unpause
- cluster count-failure-reports
- fcall
- fcall_ro
- function
- function delete
- function dump
- function flush
- function help
- function kill
- function list
- function load
- function restore
- function stats
- keys
- lastsave
- latency
- latency doctor
- latency graph
- latency help
- latency histogram
- latency history
- latency latest
- latency reset
- memory
- memory doctor
- memory help
- memory malloc-stats
- memory purge
- memory stats

- memory usage
- monitor
- move
- object
- object encoding
- object freq
- object help
- object idletime
- object refcount
- pfdebug
- pfselftest
- psubscribe
- pubsub numpat
- punsubscribe
- script kill
- slowlog
- slowlog get
- slowlog help
- slowlog len
- slowlog reset
- swapdb
- unwatch
- wait
- watch

지원되는 Memcached 명령

ElastiCache Serverless for Memcached는 다음을 제외하고 오픈 소스 memcached 1.6의 모든 memcached 명령을 지원합니다.

- 클라이언트 연결에는 가 필요합니다. TLS따라서 UDP 프로토콜이 지원되지 않습니다.

- 바이너리 프로토콜은 memcached 1.6에서 공식적으로 지원 중단되었으므로 지원되지 않습니다.
- 많은 수의 키를 가져와서 서버에 대한 잠재적인 DoS 공격을 피하기 위해 GET/GETS 명령은 16KB로 제한됩니다.
- CLIENT_ERROR에서는 지연된 flush_all 명령이 거부됩니다.
- 다음과 같이 엔진을 구성하거나 엔진 상태 또는 로그에 대한 내부 정보를 표시하는 명령은 지원되지 않습니다.
  - STATS 명령의 경우 stats, stats reset만 지원됩니다. 다른 변형은 ERROR를 반환합니다.
  - lru / lru_crawler - LRU 및 LRU크롤러 설정 수정
  - watch - memcached 서버 로그 감시
  - verbosity - 서버 로그 수준 구성
  - me - 메타 디버그(me) 명령은 지원되지 않습니다.

Valkey 및 Redis OSS 구성 및 제한

Valkey 및 Redis OSS 엔진은 각각 여러 구성 파라미터를 제공하며, 그 중 일부는 ElastiCache (Redis OSS)에서 수정할 수 있고 일부는 안정적인 성능과 신뢰성을 제공하도록 수정할 수 없습니다.

서버리스 캐시

서버리스 캐시의 경우 파라미터 그룹이 사용되지 않으며 모든 Valkey 또는 Redis OSS 구성을 수정할 수 없습니다. 다음 Valkey 또는 Redis OSS 파라미터가 있습니다.

이름	Details	설명
acl-pubsub-default	allchannels	캐시의 ACL 사용자에 대한 기본 pubsub 채널 권한입니다.
client-output-buffer-limit	normal 0 0 0 pubsub 32mb 8mb 60	일반 클라이언트에는 버퍼 제한이 없습니다. PUB/SUB 클라이언트가 32MiB 백로그를 위반하거나 60초 동안 8MiB 백로그를 위반하면 연결이 해제됩니다.
client-query-buffer-limit	1GiB	단일 클라이언트 쿼리 버퍼의 최대 크기입니다. 또한 클라이언트는 4,000개가 넘는 인수가 포함된 요청을 발행할 수 없습니다.

이름	Details	설명
cluster-allow-pubsubshard-when-down	yes	이렇게 하면 캐시가 부분적으로 다운된 상태에서도 캐시가 pubsub 트래픽을 처리할 수 있습니다.
cluster-allow-reads-when-down	yes	이렇게 하면 캐시가 부분적으로 다운된 상태에서도 캐시가 읽기 트래픽을 처리할 수 있습니다.
cluster-enabled	yes	모든 서버리스 캐시는 클러스터 모드를 지원하므로 데이터를 여러 백엔드 샤드에 투명하게 분할할 수 있습니다. 모든 슬롯은 단일 가상 노드에 포함된 것으로 클라이언트에 표시됩니다.
cluster-require-full-coverage	no	키스페이스가 부분적으로 다운된 경우(즉, 적어도 하나 이상의 해시 슬롯에 액세스할 수 없는 경우) 캐시는 여전히 포함되는 키스페이스 부분의 쿼리를 계속 수락합니다. 전체 키스페이스는 항상 `cluster slots`의 단일 가상 노드에서 '포함' 상태로 존재합니다.
lua-time-limit	5000	가 스크립트를 중지하는 작업을 ElastiCache 수행하기 전에 Lua 스크립트의 최대 실행 시간을 밀리초 단위로 표시합니다. `lua-time-limit` 이 초과되면 모든 Valkey 또는 Redis OSS 명령이 양식 ____-BUSY의 오류를 반환할 수 있습니다. 이 상태는 많은 필수 Valkey 또는 Redis OSS 작업에 간섭을 일으킬 수 있으므로 ElastiCache 는 먼저 SCRIPT KILL 명령을 실행합니다. 이 작업이 실패하면 Valkey 또는 Redis ElastiCache 를 강제로 다시 시작합니다OSS.

이름	Details	설명
maxclients	65000	한 번에 캐시에 연결할 수 있는 최대 클라이언트 수입니다. 설정된 추가 연결이 성공하거나 성공하지 못할 수 있습니다.
maxmemory-policy	volatile-lru	TTL 세트가 있는 항목은 캐시의 메모리 한도에 도달하면 (LRU) 추정 후 least-recently-used 제거됩니다.
notify-keyspace-events	(빈 문자열)	현재 키스페이스 이벤트는 서버리스 캐시에서는 지원되지 않습니다.
port	기본 포트: 6379 읽기 포트: 6380	서버리스 캐시는 동일한 호스트 이름이 있는 포트 2개로 제시됩니다. 기본 포트에서는 쓰기 및 읽기가 가능한 반면, 읽기 포트는 READONLY 명령을 사용하여 짧은 지연 시간으로 최종 읽기 일관성을 지원합니다.
proto-max-bulk-len	512MiB	단일 요소 요청의 최대 크기입니다.
timeout	0	클라이언트는 특정 유휴 시간에 강제로 연결이 해제되지는 않지만 로드 밸런싱을 위해 정상 상태일 때 연결이 해제되는 경우도 있을 수 있습니다.

또한 다음과 같은 제한 사항이 있습니다.

이름	Details	설명
키 이름 길이	4KiB	단일 Valkey 또는 Redis OSS 키 또는 채널 이름의 최대 크기입니다. 이 기준보다 큰 키를 참조하는 클라이언트에는 오류가 발생합니다.

이름	Details	설명
Lua 스크립트 크기	4MiB	단일 Valkey 또는 Redis OSS Lua 스크립트의 최대 크기입니다. 이 기준보다 큰 Lua 스크립트를 로드하려고 하면 오류가 발생합니다.
슬롯 크기	32GiB	단일 Valkey 또는 Redis OSS해시 슬롯의 최대 크기입니다. 단일 Valkey 또는 Redis OSS 슬롯에서 이보다 더 많은 데이터를 설정하려는 클라이언트는 슬롯에서 제거 정책을 트리거하고 키를 제거할 수 없는 경우 메모리 부족(OOM) 오류를 수신합니다.

자체 설계된 클러스터

자체 설계된 클러스터의 경우 구성 가능한 구성 파라미터의 기본값에 대해 알아보려면 Valkey 및 Redis OSS 파라미터 섹션을 참조하세요. 기본값을 재정의해야 하는 특정 사용 사례가 없는 한 일반적으로 기본값을 사용하는 것이 좋습니다.

IPv6 Valkey, Redis OSS 및 Memcached에 대한 클라이언트 예제

ElastiCache 는 Valkey, Redis OSS 및 Memcached와 호환됩니다. 즉, IPv6 연결을 지원하는 클라이언트는 IPv6 활성화된 ElastiCache (Memcached) 클러스터에 연결할 수 있어야 합니다. IPv6 활성화된 리소스와 상호 작용할 때 주의해야 할 몇 가지 주의 사항이 있습니다.

ElastiCache 리소스에 대한 Valkey 및 Redis 클라이언트 구성에 대한 권장 사항은 데이터베이스 블로그에서 Valkey 및 Redis 클라이언트 블로그 게시물의 모범 사례를 볼 수 있습니다. AWS OSS

다음은 일반적으로 사용되는 오픈 소스 클라이언트 라이브러리를 사용하여 IPv6 활성화된 ElastiCache 리소스와 상호 작용하는 모범 사례입니다.

Valkey 및 Redis로 검증된 클라이언트 OSS

ElastiCache 는 Valkey 및 오픈 소스 Redis 와 호환됩니다OSS. 즉, IPv6 연결을 지원하는 Valkey 및 오픈 소스 Redis OSS 클라이언트는 IPv6 활성화된 ElastiCache (Redis OSS) 클러스터에 연결할 수 있어야 합니다. 또한 가장 인기 있는 Python 및 Java 클라이언트 중 몇 개는 지원되는 모든 네트워크 유형 구성(IPv4만, IPv6 만 및 듀얼 스택)에서 작동하도록 특별히 테스트되고 검증되었습니다.

다음 클라이언트는 Valkey 및 Redis 에 대해 지원되는 모든 네트워크 유형 구성에서 작동하도록 특별히 검증되었습니다OSS.

검증된 클라이언트:

- Redis Py () – 4.1.2
- 양상추 - 버전: 6.1.6.RELEASE
- Jedis – 버전: 3.6.0

클라이언트 모범 사례(Valkey 및 RedisOSS)

일반적인 시나리오의 모범 사례를 알아보고 가장 인기 있는 오픈 소스 Valkey 및 Redis OSS 클라이언트 라이브러리(redis-py, PHPRedis및 Lettuce)의 코드 예제와 함께 일반적으로 사용되는 오픈 소스 Memcached 클라이언트 라이브러리를 사용하여 ElastiCache 리소스와 상호 작용하기 위한 모범 사례를 따르세요.

주제

- 많은 수의 연결(Valkey 및 RedisOSS)
- 클러스터 클라이언트 검색 및 지수 백오프(Valkey 및 RedisOSS)
- 클라이언트 측 제한 시간 구성(Valkey 및 RedisOSS)
- 서버 측 유휴 제한 시간 구성(Valkey 및 RedisOSS)
- Lua 스크립트
- 대형 복합 항목 저장(Valkey 및 RedisOSS)
- Lettuce 클라이언트 구성(Valkey 및 RedisOSS)
- 듀얼 스택 클러스터(Valkey 및 RedisOSS)에 대한 기본 프로토콜 구성

많은 수의 연결(Valkey 및 RedisOSS)

서버리스 캐시 및 개별 ElastiCache (Redis OSS) 노드는 최대 65,000개의 동시 클라이언트 연결을 지원합니다. 하지만 성능을 최적화하려면 클라이언트 애플리케이션이 해당 연결 수준에서 계속 작동하지 않는 것이 좋습니다. Valkey와 Redis에는 OSS 각각 수신 클라이언트 요청이 순차적으로 처리되는 이벤트 루프를 기반으로 하는 단일 스레드 프로세스가 있습니다. 즉, 연결된 클라이언트 수가 늘어날수록 해당 클라이언트의 응답 시간이 길어집니다.

Valkey 또는 Redis OSS 서버에서 연결 병목 현상이 발생하지 않도록 다음 일련의 작업을 수행할 수 있습니다.

- 읽기 전용 복제본에서 읽기 작업을 수행합니다. 이는 클러스터 모드가 비활성화된 ElastiCache 리더 엔드포인트를 사용하거나 서버리스 캐시를 포함하여 클러스터 모드가 활성화된 읽기에 복제본을 사용하여 수행할 수 있습니다.
- 쓰기 트래픽을 여러 프라이머리 노드에 분산합니다. 2가지 방법으로 수행할 수 있습니다. OSS 클러스터 모드 지원 클라이언트와 함께 다중 샤딩된 Valkey 또는 Redis 클러스터를 사용할 수 있습니다. 클라이언트 측 샤딩이 비활성화된 클러스터 모드에서 여러 프라이머리 노드에 쓸 수도 있습니다. 이 작업은 서버리스 캐시에서 자동으로 수행됩니다.
- 클라이언트 라이브러리에서 사용 가능한 경우 연결 풀을 사용하세요.

일반적으로 TCP 연결 생성은 일반적인 Valkey 또는 Redis OSS 명령에 비해 계산 비용이 많이 드는 작업입니다. 예를 들어, SET/GET 요청을 처리하는 것은 기존 연결을 재사용할 때 훨씬 더 빨라집니다. 크기가 한정된 클라이언트 연결 풀을 사용하면 연결 관리 시의 오버헤드가 줄어듭니다. 또한 클라이언트 애플리케이션에서 동시에 들어오는 연결 수를 제한합니다.

의 다음 코드 예제는 각 새 사용자 요청에 대해 새 연결이 생성되었음을 PHPRedis 보여줍니다.

```
$redis = new Redis();
if ($redis->connect($HOST, $PORT) != TRUE) {
 //ERROR: connection failed
 return;
}
$redis->set($key, $value);
unset($redis);
$redis = NULL;
```

Graviton2(m6g.2xlargeEC2)(ElastiCache Redis) 노드에 연결된 Amazon Elastic Compute Cloud(Amazon OSS) 인스턴스의 루프에서 이 코드를 벤치마킹했습니다. 클라이언트와 서버를 동일 가용 영역에 배치했습니다. 전체 작업의 평균 지연 시간은 2.82밀리초였습니다.

코드를 업데이트하고 영구 연결과 연결 풀을 사용했을 때 전체 작업의 평균 지연 시간은 0.21밀리초였습니다.

```
$redis = new Redis();
if ($redis->pconnect($HOST, $PORT) != TRUE) {
 // ERROR: connection failed
 return;
}
$redis->set($key, $value);
unset($redis);
```

```
$redis = NULL;
```

필수 redis.ini 구성:

- `redis.pconnect.pooling_enabled=1`
- `redis.pconnect.connection_limit=10`

다음 코드는 Redis-py 연결 풀의 예제입니다.

```
conn = Redis(connection_pool=redis.BlockingConnectionPool(host=HOST,
 max_connections=10))
conn.set(key, value)
```

다음 코드는 Lettuce 연결 풀의 예제입니다.

```
RedisClient client = RedisClient.create(RedisURI.create(HOST, PORT));
GenericObjectPool<StatefulRedisConnection> pool =
 ConnectionPoolSupport.createGenericObjectPool(() -> client.connect(), new
 GenericObjectPoolConfig());
pool.setMaxTotal(10); // Configure max connections to 10
try (StatefulRedisConnection connection = pool.borrowObject()) {
 RedisCommands syncCommands = connection.sync();
 syncCommands.set(key, value);
}
```

클러스터 클라이언트 검색 및 지수 백오프(Valkey 및 RedisOSS)

클러스터 모드가 활성화된 상태에서 ElastiCache Valkey 또는 Redis OSS 클러스터에 연결할 때는 해당 클라이언트 라이브러리가 클러스터를 인식해야 합니다. 요청을 올바른 노드로 보내고 클러스터 리디렉션을 처리하는 데 따른 성능 오버헤드를 피하려면 클라이언트가 클러스터의 해당 노드에 대한 해시 슬롯 맵을 가져와야 합니다. 따라서 클라이언트는 다음과 같은 2가지 상황에서 전체 슬롯 목록과 매핑된 노드를 검색해야 합니다.

- 클라이언트가 초기화되었으므로 초기 슬롯 구성을 채워야 합니다.
- 이전 프라이머리 노드에서 제공하는 모든 슬롯을 복제본에서 인계받을 때 장애 조치 상황이나 슬롯을 소스 프라이머리에서 대상 프라이머리 노드로 이동할 때 재분배하는 경우와 같이 서버로부터 리 MOVED디렉션이 수신됩니다.

클라이언트 검색은 일반적으로 Valkey 또는 Redis OSS 서버에 CLUSTER SLOT 또는 CLUSTER NODE 명령을 발급하여 수행됩니다. 슬롯 범위 세트와 연결된 기본 및 복제본 노드를 클라이언트로 반환하기 때문에 CLUSTER SLOT 메서드를 사용하는 것이 좋습니다. 이렇게 하면 클라이언트의 추가 구문 분석이 필요하지 않아 더 효율적입니다.

클러스터 토폴로지에 따라 CLUSTER SLOT 명령에 대한 응답 크기는 클러스터 크기에 따라 다를 수 있습니다. 큰 클러스터의 노드 수가 많을수록 응답 크기도 커집니다. 따라서 클러스터 토폴로지 검색을 수행하는 클라이언트 수가 무제한으로 증가하지 않도록 하는 것이 중요합니다. 예를 들어 클라이언트 애플리케이션이 시작되거나 서버와의 연결이 끊기고 클러스터 검색을 수행해야 하는 경우, 일반적인 실수 중 하나는 클라이언트 애플리케이션이 재시도 시 지수 백오프를 추가하지 않고 여러 번 재연결 및 검색 요청을 실행하는 것입니다. 이렇게 하면 Valkey 또는 Redis OSS 서버가 100%의 CPU 사용률로 장기간 응답하지 않을 수 있습니다. 각 CLUSTER SLOT 명령이 클러스터 버스에서 많은 수의 노드를 처리해야 하는 경우 중단이 연장됩니다. Python(redis-py-cluster) 및 Java(Lettuce 및 Redisson)를 비롯한 여러 언어에서 이러한 동작으로 인해 과거에 여러 클라이언트가 중단된 것이 관찰되었습니다.

서버리스 캐시의 경우, 알려진 클러스터 토폴로지가 정적이고 쓰기 엔드포인트와 읽기 엔드포인트의 두 항목으로 구성되어 있으므로 많은 문제가 자동으로 완화됩니다. 또한 캐시 엔드포인트를 사용하면 클러스터 검색이 여러 노드에 자동으로 분산됩니다. 하지만 여전히 다음 권장 사항을 따르는 것이 좋습니다.

연결 및 검색 요청의 갑작스러운 유입으로 인한 영향을 완화하려면 다음 내용을 따르세요.

- 제한된 크기의 클라이언트 연결 풀을 구현하여 클라이언트 애플리케이션에서 동시에 들어오는 연결 수를 제한합니다.
- 제한 시간 초과로 인해 서버에서 클라이언트 연결이 끊어지면 지터가 있는 지수 백오프를 사용하여 다시 시도합니다. 이렇게 하면 여러 클라이언트가 동시에 서버에 과부하를 가하지 않도록 할 수 있습니다.
- 에서 연결 엔드포인트 찾기 ElastiCache 섹션의 가이드를 사용하여 클러스터 검색을 수행할 클러스터 엔드포인트를 확인합니다. 이렇게 하면 클러스터에서 몇 개의 하드코딩된 시드 노드에 도달하지 않고 클러스터의 모든 노드(최대 90개)에 검색 부하를 분산할 수 있습니다.

다음은 redis-py, PHPRedis및 Lettuce의 지수 백오프 재시도 로직에 대한 몇 가지 코드 예제입니다.

백오프 로직 샘플 1: redis-py

redis-py에는 실패하자마자 한 번 재시도하는 재시도 메커니즘이 내장되어 있습니다. 이 메커니즘은 Redis OSS 객체를 생성할 때 제공된 `retry_on_timeout` 인수를 통해 활성화할 수 있습니다. 여기서는 지수 백오프와 지터를 사용하는 사용자 지정 재시도 메커니즘을 보여 줍니다. redis-py(#1494)에서

기본적으로 지수 백오프를 구현하기 위한 풀 요청이 제출된 상태입니다. 향후에는 수동으로 구현할 필요가 없을 수도 있습니다.

```
def run_with_backoff(function, retries=5):
base_backoff = 0.1 # base 100ms backoff
max_backoff = 10 # sleep for maximum 10 seconds
tries = 0
while True:
try:
  return function()
except (ConnectionError, TimeoutError):
  if tries >= retries:
 raise
  backoff = min(max_backoff, base_backoff * (pow(2, tries) + random.random()))
  print(f"sleeping for {backoff:.2f}s")
  sleep(backoff)
  tries += 1
```

그런 다음, 다음 코드를 사용하여 값을 설정할 수 있습니다.

```
client = redis.Redis(connection_pool=redis.BlockingConnectionPool(host=HOST,
 max_connections=10))
res = run_with_backoff(lambda: client.set("key", "value"))
print(res)
```

워크로드에 따라 지연 시간에 민감한 워크로드의 기본 백오프 값을 1초에서 수십 또는 수백 밀리초로 변경하는 것을 고려할 수 있습니다.

백오프 로직 샘플 2: PHPRedis

PHPRedis 에는 최대 10회 (구성 불가)를 재시도하는 재시도 메커니즘이 내장되어 있습니다. 시도 사이에는 지연을 구성할 수 있습니다(두 번째 재시도부터 지터 사용). 자세한 내용은 다음 샘플 예제를 참조하세요. 이후 병합되고 문서화된 PHPredis (#1986)에 지수 백오프를 기본적으로 구현하기 위한 풀 요청을 제출했습니다. 의 최신 릴리스에 있는 사용자의 경우 수동으로 구현PHPRedis할 필요가 없지만 이전 버전에 있는 사용자의 참조를 여기에 포함했습니다. 다음은 현재 재시도 메커니즘의 지연을 구성하는 코드 예제입니다.

```
$timeout = 0.1; // 100 millisecond connection timeout
$retry_interval = 100; // 100 millisecond retry interval
$client = new Redis();
if($client->pconnect($HOST, $PORT, $timeout, NULL, $retry_interval) != TRUE) {
```

```
 return; // ERROR: connection failed
}
$client->set($key, $value);
```

백오프 로직 샘플 3: Lettuce

Lettuce에는 지수 백오프 및 Jitter 관련 게시물에 명시된 지수 백오프 전략을 기반으로 하는 재시도 메커니즘이 내장되어 있습니다. 다음은 코드 발췌문은 전체적인 지터의 접근 방식을 보여 줍니다.

```
public static void main(String[] args)
{
 ClientResources resources = null;
 RedisClient client = null;

 try {
  resources = DefaultClientResources.builder()
    .reconnectDelay(Delay.fullJitter(
   Duration.ofMillis(100),     // minimum 100 millisecond delay
   Duration.ofSeconds(5),      // maximum 5 second delay
   100, TimeUnit.MILLISECONDS) // 100 millisecond base
  ).build();

  client = RedisClient.create(resources, RedisURI.create(HOST, PORT));
  client.setOptions(ClientOptions.builder()
 .socketOptions(SocketOptions.builder().connectTimeout(Duration.ofMillis(100)).build()) //
 100 millisecond connection timeout
 .timeoutOptions(TimeoutOptions.builder().fixedTimeout(Duration.ofSeconds(5)).build()) //
 5 second command timeout
 .build());

     // use the connection pool from above example
 } finally {
  if (connection != null) {
   connection.close();
  }

  if (client != null){
   client.shutdown();
  }

  if (resources != null){
   resources.shutdown();
  }
```

```
 }
}
```

클라이언트 측 제한 시간 구성(Valkey 및 RedisOSS)

클라이언트 측 제한 시간 구성

서버가 요청을 처리하고 응답을 생성하는 데 충분한 시간을 할애할 수 있도록 클라이언트 측 제한 시간을 적절하게 구성합니다. 또한 서버와의 연결을 설정할 수 없는 경우 빠른 실패로 이어질 수 있습니다. 특정 Valkey 또는 Redis OSS 명령은 다른 명령보다 계산 비용이 더 높을 수 있습니다. 예를 들어 원자적으로 실행해야 하는 여러 명령이 포함된 Lua 스크립트 또는 MULTI/EXEC 트랜잭션이 있습니다. 일반적으로 서버로부터 응답을 받기 전에 클라이언트 제한 시간이 초과되지 않도록 하려면 다음을 포함하여 클라이언트 측 제한 시간을 높게 설정하는 것이 좋습니다.

- 여러 키에서 명령 실행
- 여러 개의 개별 Valkey 또는 Redis OSS 명령으로 구성된 실행 중인 MULTI/EXEC 트랜잭션 또는 Lua 스크립트
- 큰 값 읽기
- 다음과 같은 차단 작업 수행 BLPOP

와 같은 차단 작업의 경우 BLPOP명령 제한 시간을 소켓 제한 시간보다 낮은 숫자로 설정하는 것이 가장 좋습니다.

다음은 redis-py, PHPRedis및 Lettuce에서 클라이언트 측 제한 시간을 구현하기 위한 코드 예제입니다.

제한 시간 구성 샘플 1: redis-py

다음은 redis-py를 사용한 코드 예제입니다.

```
# connect to Redis server with a 100 millisecond timeout
# give every Redis command a 2 second timeout
client = redis.Redis(connection_pool=redis.BlockingConnectionPool(host=HOST,
 max_connections=10,socket_connect_timeout=0.1,socket_timeout=2))

res = client.set("key", "value") # will timeout after 2 seconds
print(res)                       # if there is a connection error
```

```
res = client.blpop("list", timeout=1) # will timeout after 1 second
                                      # less than the 2 second socket timeout
print(res)
```

제한 시간 구성 샘플 2: PHPRedis

다음은 를 사용하는 코드 예제입니다PHPRedis.

```
// connect to Redis server with a 100ms timeout
// give every Redis command a 2s timeout
$client = new Redis();
$timeout = 0.1; // 100 millisecond connection timeout
$retry_interval = 100; // 100 millisecond retry interval
$client = new Redis();
if($client->pconnect($HOST, $PORT, 0.1, NULL, 100, $read_timeout=2) != TRUE){
 return; // ERROR: connection failed
}
$client->set($key, $value);

$res = $client->set("key", "value"); // will timeout after 2 seconds
print "$res\n";                      // if there is a connection error

$res = $client->blpop("list", 1); // will timeout after 1 second
print "$res\n";                   // less than the 2 second socket timeout
```

제한 시간 구성 샘플 3: Lettuce

다음은 Lettuce를 사용한 코드 예제입니다.

```
// connect to Redis server and give every command a 2 second timeout
public static void main(String[] args)
{
 RedisClient client = null;
 StatefulRedisConnection<String, String> connection = null;
 try {
  client = RedisClient.create(RedisURI.create(HOST, PORT));
  client.setOptions(ClientOptions.builder()
 .socketOptions(SocketOptions.builder().connectTimeout(Duration.ofMillis(100)).build()) //
 100 millisecond connection timeout
 .timeoutOptions(TimeoutOptions.builder().fixedTimeout(Duration.ofSeconds(2)).build()) //
 2 second command timeout
 .build());
```

```
  // use the connection pool from above example

  commands.set("key", "value"); // will timeout after 2 seconds
  commands.blpop(1, "list"); // BLPOP with 1 second timeout
 } finally {
  if (connection != null) {
   connection.close();
  }

  if (client != null){
   client.shutdown();
  }
 }
}
```

서버 측 유휴 제한 시간 구성(Valkey 및 RedisOSS)

고객 애플리케이션에 연결된 유휴 클라이언트 수가 많지만 적극적으로 명령이 전송되지 않는 경우가 있습니다. 이러한 시나리오의 경우 유휴 클라이언트 수가 많으면 65,000개의 연결이 모두 소진될 수 있습니다. 이러한 시나리오가 발생하지 않도록 하려면 Valkey 및 Redis OSS 파라미터를 통해 서버의 제한 시간 설정을 적절하게 구성하세요. 이렇게 하면 서버가 유휴 클라이언트의 연결을 적극적으로 해제하여 연결 수가 증가하지 않도록 할 수 있습니다. 서버리스 캐시에서는 이 설정을 사용할 수 없습니다.

Lua 스크립트

Valkey 및 Redis는 Lua 스크립트를 실행하는 명령을 포함하여 200개 이상의 명령을 OSS 지원합니다. 그러나 Lua 스크립트의 경우 Valkey 또는 Redis 의 메모리 및 가용성에 영향을 미칠 수 있는 몇 가지 위험이 있습니다OSS.

파라미터화되지 않은 Lua 스크립트

각 Lua 스크립트는 실행되기 전에 Valkey 또는 Redis OSS 서버에 캐시됩니다. 파라미터화되지 않은 Lua 스크립트는 고유하므로 Valkey 또는 Redis OSS 서버가 많은 수의 Lua 스크립트를 저장하고 더 많은 메모리를 소비할 수 있습니다. 이를 완화하려면 모든 Lua 스크립트에 파라미터가 지정되고 필요한 경우 정기적으로 캐시된 Lua 스크립트를 정리SCRIPTFLUSH해야 합니다.

다음 예제는 파라미터화된 스크립트를 사용하는 방법을 보여 줍니다. 먼저 3개의 캐시된 Lua 스크립트가 생성되는 파라미터화되지 않은 접근 방식의 예제가 있으며 이 방식은 권장하지 않습니다.

```
eval "return redis.call('set','key1','1')" 0
```

```
eval "return redis.call('set','key2','2')" 0
eval "return redis.call('set','key3','3')" 0
```

대신 다음 패턴을 사용하여 전달된 파라미터를 사용할 수 있는 단일 스크립트를 생성합니다.

```
eval "return redis.call('set',KEYS[1],ARGV[1])" 1 key1 1
eval "return redis.call('set',KEYS[1],ARGV[1])" 1 key2 2
eval "return redis.call('set',KEYS[1],ARGV[1])" 1 key3 3
```

장기 실행 Lua 스크립트

Lua 스크립트는 원자적으로 여러 명령을 실행할 수 있으므로 일반 Valkey 또는 Redis OSS 명령보다 완료하는 데 시간이 더 걸릴 수 있습니다. Lua 스크립트가 읽기 전용 작업만 실행하는 경우 중간에 중지할 수 있습니다. 하지만 Lua 스크립트는 쓰기 작업을 수행하는 즉시 종료할 수 없으므로 실행을 완료해야 합니다. 장기 실행 중인 Lua 스크립트를 변경하면 Valkey 또는 Redis OSS 서버가 장기간 응답하지 않을 수 있습니다. 이 문제를 완화하려면 장기 실행 Lua 스크립트를 사용하지 말고 사전 프로덕션 환경에서 스크립트를 테스트해 보세요.

Stealth 쓰기 기능이 있는 Lua 스크립트

Valkey 또는 Redis가 를 초과OSS하더라도 Lua 스크립트OSS가 Valkey 또는 Redis에 새 데이터를 계속 쓸 수 있는 몇 가지 방법이 있습니다`maxmemory`.

- 스크립트는 Valkey 또는 Redis OSS 서버가 미만일 때 시작`maxmemory`되며 내부에 여러 쓰기 작업이 포함되어 있습니다.
- 스크립트의 첫 번째 쓰기 명령은 메모리를 소비하지 않으며(예: DEL) 메모리를 소비하는 쓰기 작업이 더 많음
- 이외의 Valkey 또는 Redis OSS 서버에서 적절한 제거 정책을 구성하여 이 문제를 완화할 수 있습니다`noeviction`. 이렇게 하면 RedisOSS가 항목을 제거하고 Lua 스크립트 간에 메모리를 확보할 수 있습니다.

대형 복합 항목 저장(Valkey 및 RedisOSS)

일부 시나리오에서 애플리케이션은 큰 복합 항목을 Valkey 또는 RedisOSS(예: 멀티 GB 해시 데이터 세트)에 저장할 수 있습니다. 이는 Valkey 또는 Redis 에서 성능 문제를 초래하는 경우가 많기 때문에 권장 관행이 아닙니다OSS. 예를 들어 클라이언트는 HGETALL 명령을 수행하여 전체 멀티 GB 해시 컬렉션을 검색할 수 있습니다. 이렇게 하면 클라이언트 출력 버퍼에서 큰 항목을 버퍼링하는 Valkey 또는 Redis OSS 서버에 상당한 메모리 압력이 발생할 수 있습니다. 또한 클러스터 모드의 슬롯 마이그레

이션의 경우 는 직렬화된 크기가 256MB보다 큰 항목이 포함된 슬롯을 마이그레이션 ElastiCache 하지 않습니다.

대규모 항목 문제를 해결하기 위한 권장 사항은 다음과 같습니다.

- 대규모 복합 항목을 여러 개의 작은 항목으로 나눕니다. 예를 들어 항목 컬렉션을 식별하기 위해 키 이름에 공통 접두사를 사용하는 등 컬렉션을 적절하게 반영하는 키 이름 스키마를 사용하여 대규모 해시 컬렉션을 개별 키-값 필드로 나눕니다. 동일한 컬렉션에서 여러 필드에 원자적으로 액세스해야 하는 경우 명령을 사용하여 동일한 MGET 명령에서 여러 키 값을 검색할 수 있습니다.
- 모든 옵션을 평가했는데도 여전히 대규모 컬렉션 데이터 세트를 분리할 수 없으면 전체 컬렉션 대신 컬렉션의 데이터 하위 집합에서 작동하는 명령어를 사용해 보세요. 동일한 명령어로 전체 다중 GB 컬렉션을 세부적으로 검색해야 하는 사용 사례는 피하도록 합니다. 한 가지 예는 HGETALL해시 컬렉션 대신 HGET 또는 HMGET 명령을 사용하는 것입니다.

Lettuce 클라이언트 구성(Valkey 및 RedisOSS)

이 섹션에서는 권장 Java 및 Lettuce 구성 옵션과 ElastiCache 클러스터에 적용하는 방법을 설명합니다.

이 섹션의 권장 사항은 Lettuce 버전 6.2.2에서 테스트되었습니다.

주제

- 예: 클러스터 모드에 대한 양상추 구성, TLS 활성화됨
- 예: 클러스터 모드에 대한 Lettuce 구성이 비활성화됨, TLS 활성화됨

Java DNS 캐시 TTL

Java 가상 머신(JVM)은 DNS 이름 조회를 캐시합니다. 가 호스트 이름을 IP 주소로 JVM 확인하면 time-to-live ()라고 하는 지정된 기간 동안 IP 주소를 캐싱합니다TTL.

TTL 가치 선택은 지연 시간과 변화에 대한 대응성 간의 균형입니다. 가 짧을수록 TTLs DNS 해석자는 클러스터의 업데이트가 DNS 더 빨라집니다. 이렇게 하면 클러스터가 수행하는 교체 또는 기타 워크플로에 애플리케이션이 더 빠르게 응답할 수 있습니다. 그러나 TTL 이 너무 낮으면 쿼리 볼륨이 증가하여 애플리케이션의 지연 시간이 늘어날 수 있습니다. 올바른 TTL 값은 없지만 TTL 값을 설정할 때 변경 사항이 적용될 때까지 기다릴 수 있는 시간을 고려하는 것이 좋습니다.

ElastiCache 노드는 변경될 수 있는 DNS 이름 항목을 사용하기 때문에 를 TTL 5~10초 JVM 미만으로 구성하는 것이 좋습니다. 이렇게 하면 노드의 IP 주소가 변경되면 애플리케이션에서 DNS 항목을 다시 쿼리하여 리소스의 새 IP 주소를 수신하고 사용할 수 있습니다.

일부 Java 구성에서는 TTL가 다시 시작될 때까지 DNS 항목을 새로 고치지 않도록 JVM 기본값JVM이 설정됩니다.

를 설정하는 방법에 대한 자세한 내용은 를 설정하는 방법을 JVM TTL참조하세요. JVM TTL

Lettuce 버전

Lettuce 버전 6.2.2 이상을 사용하는 것이 좋습니다.

엔드포인트

클러스터 모드가 활성화된 클러스터를 사용하는 경우 `redisUri`를 클러스터 구성 엔드포인트로 설정합니다. 이를 DNS 조회하면 클러스터에서 사용 가능한 모든 노드 목록이 URI 반환되며 클러스터 초기화 중에 해당 노드 중 하나로 무작위로 확인됩니다. 토폴로지 새로 고침의 작동 방식에 대한 자세한 내용은 이 주제의 dynamicRefreshResources뒷부분을 참조하세요.

SocketOption

를 활성화합니다KeepAlive. 이 옵션을 활성화하면 명령 런타임 중에 실패한 연결을 처리할 필요가 줄어듭니다.

애플리케이션 요구 사항 및 워크로드에 따라 연결 제한 시간을 설정해야 합니다. 자세한 내용은 이 주제의 후반부에서 연결 제한 섹션을 참조하세요.

ClusterClientOption: 클러스터 모드 활성화된 클라이언트 옵션

연결이 끊AutoReconnect어지면 활성화합니다.

를 설정합니다CommandTimeout. 자세한 내용은 이 주제의 후반부에서 연결 제한 섹션을 참조하세요.

토폴로지에서 실패한 노드를 필터링nodeFilter하도록 를 설정합니다. Lettuce는 클라이언트의 '파티션'(샤드라고도 함)의 '클러스터 노드' 출력(PFAIL/FAIL 상태의 노드 포함)에 있는 모든 노드를 저장합니다. 클러스터 토폴로지를 생성하는 동안 모든 파티션 노드에 연결을 시도합니다. 장애가 발생한 노드를 추가하는 이러한 Lettuce 동작은 어떤 이유로든 노드를 교체할 때 연결 오류(또는 경고)를 유발할 수 있습니다.

예를 들어 장애 조치가 완료되고 클러스터가 복구 프로세스를 시작한 후 clusterTopology 가 새로 고쳐지는 동안 클러스터 버스 노드 맵은 다운 노드가 노드로 나열되는 시간이 짧FAIL아 토폴로지에서 완전히 제거됩니다. 이 기간 동안 Lettuce 클라이언트는 이를 정상 노드로 간주하고 지속적으로 연결합니다. 이로 인해 재시도가 모두 끝난 후 오류가 발생합니다.

예:

```
final ClusterClientOptions clusterClientOptions =
    ClusterClientOptions.builder()
    ... // other options
    .nodeFilter(it ->
        ! (it.is(RedisClusterNode.NodeFlag.FAIL)
        || it.is(RedisClusterNode.NodeFlag.EVENTUAL_FAIL)
        || it.is(RedisClusterNode.NodeFlag.HANDSHAKE)
        || it.is(RedisClusterNode.NodeFlag.NOADDR)))
    .validateClusterNodeMembership(false)
    .build();
redisClusterClient.setOptions(clusterClientOptions);
```

> **Note**
>
> 노드 필터링은 true로 DynamicRefreshSources 설정된 와 함께 사용하는 것이 가장 좋습니다. 그러지 않으면 문제가 있는 단일 시드 노드에서 토폴로지 보기를 가져와서 일부 샤드의 프라이머리 노드에 장애가 있는 것으로 간주하면 이 프라이머리 노드가 필터링되어 슬롯이 포함되지 않습니다. 시드 노드가 여러 개 있으면(true DynamicRefreshSources 인 경우) 새로 승격된 프라이머리를 사용한 장애 조치 후 시드 노드의 일부 이상이 업데이트된 토폴로지 뷰를 가져야 하므로 이 문제가 발생할 가능성이 줄어듭니다.

ClusterTopologyRefreshOptions: 클러스터 모드 활성화 클라이언트의 클러스터 토폴로지 새로 고침을 제어하는 옵션

> **Note**
>
> 클러스터 모드가 비활성화된 클러스터는 클러스터 검색 명령을 지원하지 않으며 모든 클라이언트 동적 토폴로지 검색 기능과 호환되지 않습니다.
> 에서 비활성화된 클러스터 모드 ElastiCache 는 Lettuce의 와 호환되지 않습니다`MasterSlaveTopologyRefresh`. 대신, 비활성화된 클러스터 모드에는 `StaticMasterReplicaTopologyProvider`를 구성하여 클러스터 읽기 및 쓰기 엔드포인트를 제공할 수 있습니다.
> 클러스터 모드 비활성화 클러스터 연결에 대한 자세한 내용은 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터의 엔드포인트 찾기(콘솔)를 참조하십시오.
> Lettuce의 동적 토폴로지 검색 기능을 사용하려면 클러스터 모드 활성화 클러스터를 만들고 기존 클러스터와 샤드 구성을 동일하게 해야 합니다. 하지만 클러스터 모드 활성화 클러스터에는

최소 3개의 샤드와 1개 이상의 복제본을 구성하는 것을 권장하며, 이렇게 해야 빠른 장애 조치를 지원할 수 있습니다.

를 활성화합니다enablePeriodicRefresh. 이렇게 하면 주기적인 클러스터 토폴로지 업데이트가 활성화되어 클라이언트가 클러스터 토폴로지를 의 간격으로 업데이트합니다refreshPeriod (기본값: 60초). 비활성화된 경우 클라이언트는 클러스터에 대해 명령을 실행하려고 할 때 오류가 발생한 경우에만 클러스터 토폴로지를 업데이트합니다.

이 옵션을 활성화하면 이 작업을 백그라운드 작업에 추가하여 클러스터 토폴로지 새로 고침과 관련된 지연 시간을 줄일 수 있습니다. 토폴로지 새로 고침은 백그라운드 작업에서 수행되지만 노드가 많은 클러스터의 경우 다소 느릴 수 있습니다. 이는 가장 업데이트된 클러스터 보기를 얻기 위해 모든 노드에 보기를 쿼리하기 때문입니다. 대규모 클러스터를 실행하는 경우 기간을 늘리는 것이 좋습니다.

를 활성화합니다enableAllAdaptiveRefreshTriggers. 이렇게 하면 _REDIRECT, MOVED_, _, _, ASK_ 등의 모든 트리거를 사용하는 적응형 토폴로지 새로 고침이 활성화됩니다 REDIRECTPERSISTENTRECONNECTSUNCOVEREDSLOTUNKNOWNNODE. 적응형 새로 고침 트리거는 Valkey 또는 Redis OSS 클러스터 작업 중에 발생하는 이벤트를 기반으로 토폴로지 보기 업데이트를 시작합니다. 이 옵션을 활성화하면 위 트리거 중 하나가 발생할 때 즉시 토폴로지가 새로 고쳐집니다. 이벤트가 대규모로 발생할 수 있으므로 적응형 트리거 새로 고침은 제한 시간을 사용하여 속도를 제한합니다(업데이트 간 기본 제한 시간: 30).

를 활성화합니다closeStaleConnections. 이렇게 하면 클러스터 토폴로지를 새로 고칠 때 오래된 연결을 닫을 수 있습니다. ClusterTopologyRefreshOptions.isPeriodicRefreshEnabled()가 true인 경우에만 적용됩니다. 활성화되면 클라이언트는 오래된 연결을 닫고 백그라운드에서 새 연결을 만들 수 있습니다. 그러면 명령 런타임 중에 실패한 연결을 처리할 필요가 줄어듭니다.

를 활성화합니다dynamicRefreshResources. 소형 클러스터의 경우 를 활성화 dynamicRefreshResources하고 대형 클러스터의 경우 비활성화하는 것이 좋습니다. dynamicRefreshResources 는 제공된 시드 노드(예: 클러스터 구성 엔드포인트)에서 클러스터 노드를 검색할 수 있습니다. 검색된 모든 노드를 소스로 사용하여 클러스터 토폴로지를 새로 고칩니다.

동적 새로 고침을 사용하면 클러스터 토폴로지에 대해 검색된 모든 노드를 쿼리하고 가장 정확한 클러스터 보기를 선택하려고 시도합니다. 거짓으로 설정하면 초기 시드 노드만 토폴로지 검색의 소스로 사용되고 초기 시드 노드에 대한 클라이언트 수만 가져옵니다. 비활성화된 경우 클러스터 구성 엔드포인트가 장애가 발생한 노드로 확인되면 클러스터 보기를 새로 고치려는 시도가 실패하고 예외가 발생합니다. 이 시나리오는 장애가 발생한 노드의 항목이 클러스터 구성 엔드포인트에서 제거될 때까지 어느 정도 시간이 걸리기 때문에 발생할 수 있습니다. 따라서 구성 엔드포인트는 여전히 짧은 시간 동안 장애가 발생한 노드로 무작위로 확인될 수 있습니다.

하지만 활성화되면 클러스터 보기에서 수신한 모든 클러스터 노드를 사용하여 현재 보기를 쿼리합니다. 해당 보기에서 장애가 발생한 노드를 필터링하므로 토폴로지 새로 고침이 성공적으로 이루어집니다. 하지만 dynamicRefreshSources 가 true이면 Lettuce는 모든 노드를 쿼리하여 클러스터 보기를 가져온 다음 결과를 비교합니다. 따라서 노드가 많은 클러스터의 경우 비용이 많이 들 수 있습니다. 노드가 많은 클러스터의 경우 이 기능을 끄는 것이 좋습니다.

```
final ClusterTopologyRefreshOptions topologyOptions =
    ClusterTopologyRefreshOptions.builder()
    .enableAllAdaptiveRefreshTriggers()
    .enablePeriodicRefresh()
    .dynamicRefreshSources(true)
    .build();
```

ClientResources

DnsResolver 를 사용하여 를 구성합니다DirContextDnsResolver. DNS 해석기는 Java의 com.sun.jndi.dns를 기반으로 합니다DnsContextFactory.

지수 백오프 및 전체 지터reconnectDelay를 사용하여 를 구성합니다. Lettuce에는 지수 백오프 전략을 기반으로 하는 재시도 메커니즘이 내장되어 있습니다. 자세한 내용은 AWS 아키텍처 블로그의 지수 백오프 및 지터를 참조하세요. 재시도 백오프 전략의 중요성에 대한 자세한 내용은 AWS 데이터베이스 블로그의 모범 사례 블로그 게시물의 백오프 로직 섹션을 참조하세요.

```
ClientResources clientResources = DefaultClientResources.builder()
   .dnsResolver(new DirContextDnsResolver())
    .reconnectDelay(
        Delay.fullJitter(
            Duration.ofMillis(100),     // minimum 100 millisecond delay
            Duration.ofSeconds(10),      // maximum 10 second delay
            100, TimeUnit.MILLISECONDS)) // 100 millisecond base
    .build();
```

제한 시간

명령 제한 시간보다 낮은 연결 제한 시간 값을 사용합니다. Lettuce는 지연 연결 설정을 사용합니다. 따라서 연결 제한 시간이 명령 제한 시간보다 긴 경우 Lettuce가 비정상 노드에 연결하려고 시도하고 명령 제한 시간이 항상 초과하면 토폴로지 새로 고침 후에도 오류가 계속 발생하는 기간이 있을 수 있습니다.

서로 다른 명령에 동적 명령 제한 시간을 사용합니다. 명령에 기대되는 기간에 따라 명령 제한 시간을 설정하는 것이 좋습니다. 예를 들어 , , FLUSHDB, FLUSHALL KEYS SMEMBERS또는 Lua 스크립트

와 같은 여러 키를 반복하는 명령에는 더 긴 제한 시간을 사용합니다. , 및 와 같은 단일 키 명령에는 더 짧은 제한 시간을 사용합니다SETGETHSET.

> **Note**
>
> 다음 예제에서 구성된 제한 시간은 최대 20바이트 길이의 키 및 값으로 실행SET/GET명령하는 테스트에 대한 것입니다. 명령이 복잡하거나 키와 값이 크면 처리 시간이 더 오래 걸릴 수 있습니다. 애플리케이션의 사용 사례에 따라 제한 시간을 설정해야 합니다.

```
private static final Duration META_COMMAND_TIMEOUT = Duration.ofMillis(1000);
private static final Duration DEFAULT_COMMAND_TIMEOUT = Duration.ofMillis(250);
// Socket connect timeout should be lower than command timeout for Lettuce
private static final Duration CONNECT_TIMEOUT = Duration.ofMillis(100);

SocketOptions socketOptions = SocketOptions.builder()
    .connectTimeout(CONNECT_TIMEOUT)
    .build();


class DynamicClusterTimeout extends TimeoutSource {
     private static final Set<ProtocolKeyword> META_COMMAND_TYPES =
 ImmutableSet.<ProtocolKeyword>builder()
          .add(CommandType.FLUSHDB)
          .add(CommandType.FLUSHALL)
          .add(CommandType.CLUSTER)
          .add(CommandType.INFO)
          .add(CommandType.KEYS)
          .build();

    private final Duration defaultCommandTimeout;
    private final Duration metaCommandTimeout;

    DynamicClusterTimeout(Duration defaultTimeout, Duration metaTimeout)
    {
        defaultCommandTimeout = defaultTimeout;
        metaCommandTimeout = metaTimeout;
    }

    @Override
    public long getTimeout(RedisCommand<?, ?, ?> command) {
        if (META_COMMAND_TYPES.contains(command.getType())) {
```

```
            return metaCommandTimeout.toMillis();
        }
        return defaultCommandTimeout.toMillis();
    }
}

// Use a dynamic timeout for commands, to avoid timeouts during
// cluster management and slow operations.
TimeoutOptions timeoutOptions = TimeoutOptions.builder()
.timeoutSource(
    new DynamicClusterTimeout(DEFAULT_COMMAND_TIMEOUT, META_COMMAND_TIMEOUT))
.build();
```

예: 클러스터 모드에 대한 양상추 구성, TLS 활성화됨

> **Note**
>
> 다음 예제의 제한 시간은 최대 20바이트 길이의 키 및 값으로 실행SET/GET명령하는 테스트에 대한 것입니다. 명령이 복잡하거나 키와 값이 크면 처리 시간이 더 오래 걸릴 수 있습니다. 애플리케이션의 사용 사례에 따라 제한 시간을 설정해야 합니다.

```
// Set DNS cache TTL
public void setJVMProperties() {
    java.security.Security.setProperty("networkaddress.cache.ttl", "10");
}

private static final Duration META_COMMAND_TIMEOUT = Duration.ofMillis(1000);
private static final Duration DEFAULT_COMMAND_TIMEOUT = Duration.ofMillis(250);
// Socket connect timeout should be lower than command timeout for Lettuce
private static final Duration CONNECT_TIMEOUT = Duration.ofMillis(100);

// Create RedisURI from the cluster configuration endpoint
clusterConfigurationEndpoint = <cluster-configuration-endpoint> // TODO: add your
 cluster configuration endpoint
final RedisURI redisUriCluster =
    RedisURI.Builder.redis(clusterConfigurationEndpoint)
        .withPort(6379)
        .withSsl(true)
        .build();

// Configure the client's resources
```

```
ClientResources clientResources = DefaultClientResources.builder()
    .reconnectDelay(
        Delay.fullJitter(
            Duration.ofMillis(100),     // minimum 100 millisecond delay
            Duration.ofSeconds(10),      // maximum 10 second delay
            100, TimeUnit.MILLISECONDS)) // 100 millisecond base
    .dnsResolver(new DirContextDnsResolver())
    .build();

// Create a cluster client instance with the URI and resources
RedisClusterClient redisClusterClient =
    RedisClusterClient.create(clientResources, redisUriCluster);

// Use a dynamic timeout for commands, to avoid timeouts during
// cluster management and slow operations.
class DynamicClusterTimeout extends TimeoutSource {
     private static final Set<ProtocolKeyword> META_COMMAND_TYPES =
 ImmutableSet.<ProtocolKeyword>builder()
          .add(CommandType.FLUSHDB)
          .add(CommandType.FLUSHALL)
          .add(CommandType.CLUSTER)
          .add(CommandType.INFO)
          .add(CommandType.KEYS)
          .build();

    private final Duration metaCommandTimeout;
    private final Duration defaultCommandTimeout;

    DynamicClusterTimeout(Duration defaultTimeout, Duration metaTimeout)
    {
        defaultCommandTimeout = defaultTimeout;
        metaCommandTimeout = metaTimeout;
    }

    @Override
    public long getTimeout(RedisCommand<?, ?, ?> command) {
        if (META_COMMAND_TYPES.contains(command.getType())) {
            return metaCommandTimeout.toMillis();
        }
        return defaultCommandTimeout.toMillis();
    }
}

TimeoutOptions timeoutOptions = TimeoutOptions.builder()
```

```
    .timeoutSource(new DynamicClusterTimeout(DEFAULT_COMMAND_TIMEOUT,
 META_COMMAND_TIMEOUT))
     .build();

// Configure the topology refreshment optionts
final ClusterTopologyRefreshOptions topologyOptions =
    ClusterTopologyRefreshOptions.builder()
    .enableAllAdaptiveRefreshTriggers()
    .enablePeriodicRefresh()
    .dynamicRefreshSources(true)
    .build();

// Configure the socket options
final SocketOptions socketOptions =
    SocketOptions.builder()
    .connectTimeout(CONNECT_TIMEOUT)
    .keepAlive(true)
    .build();

// Configure the client's options
final ClusterClientOptions clusterClientOptions =
    ClusterClientOptions.builder()
    .topologyRefreshOptions(topologyOptions)
    .socketOptions(socketOptions)
    .autoReconnect(true)
    .timeoutOptions(timeoutOptions)
    .nodeFilter(it ->
        ! (it.is(RedisClusterNode.NodeFlag.FAIL)
        || it.is(RedisClusterNode.NodeFlag.EVENTUAL_FAIL)
        || it.is(RedisClusterNode.NodeFlag.NOADDR)))
    .validateClusterNodeMembership(false)
    .build();

redisClusterClient.setOptions(clusterClientOptions);

// Get a connection
final StatefulRedisClusterConnection<String, String> connection =
    redisClusterClient.connect();

// Get cluster sync/async commands
RedisAdvancedClusterCommands<String, String> sync = connection.sync();
RedisAdvancedClusterAsyncCommands<String, String> async = connection.async();
```

예: 클러스터 모드에 대한 Lettuce 구성이 비활성화됨, TLS 활성화됨

> **Note**
>
> 다음 예제의 제한 시간은 최대 20바이트 길이의 키 및 값으로 실행SET/GET명령하는 테스트에 대한 것입니다. 명령이 복잡하거나 키와 값이 크면 처리 시간이 더 오래 걸릴 수 있습니다. 애플리케이션의 사용 사례에 따라 제한 시간을 설정해야 합니다.

```
// Set DNS cache TTL
public void setJVMProperties() {
    java.security.Security.setProperty("networkaddress.cache.ttl", "10");
}

private static final Duration META_COMMAND_TIMEOUT = Duration.ofMillis(1000);
private static final Duration DEFAULT_COMMAND_TIMEOUT = Duration.ofMillis(250);
// Socket connect timeout should be lower than command timeout for Lettuce
private static final Duration CONNECT_TIMEOUT = Duration.ofMillis(100);

// Create RedisURI from the primary/reader endpoint
clusterEndpoint = <primary/reader-endpoint> // TODO: add your node endpoint
RedisURI redisUriStandalone =

 RedisURI.Builder.redis(clusterEndpoint).withPort(6379).withSsl(true).withDatabase(0).build();

ClientResources clientResources =
    DefaultClientResources.builder()
        .dnsResolver(new DirContextDnsResolver())
        .reconnectDelay(
            Delay.fullJitter(
                Duration.ofMillis(100), // minimum 100 millisecond delay
                Duration.ofSeconds(10), // maximum 10 second delay
                100,
                TimeUnit.MILLISECONDS)) // 100 millisecond base
        .build();

// Use a dynamic timeout for commands, to avoid timeouts during
// slow operations.
class DynamicTimeout extends TimeoutSource {
     private static final Set<ProtocolKeyword> META_COMMAND_TYPES =
 ImmutableSet.<ProtocolKeyword>builder()
          .add(CommandType.FLUSHDB)
```

```
            .add(CommandType.FLUSHALL)
            .add(CommandType.INFO)
            .add(CommandType.KEYS)
            .build();

    private final Duration metaCommandTimeout;
    private final Duration defaultCommandTimeout;

    DynamicTimeout(Duration defaultTimeout, Duration metaTimeout)
    {
        defaultCommandTimeout = defaultTimeout;
        metaCommandTimeout = metaTimeout;
    }

    @Override
    public long getTimeout(RedisCommand<?, ?, ?> command) {
        if (META_COMMAND_TYPES.contains(command.getType())) {
            return metaCommandTimeout.toMillis();
        }
        return defaultCommandTimeout.toMillis();
    }
}

TimeoutOptions timeoutOptions = TimeoutOptions.builder()
    .timeoutSource(new DynamicTimeout(DEFAULT_COMMAND_TIMEOUT, META_COMMAND_TIMEOUT))
     .build();

final SocketOptions socketOptions =
    SocketOptions.builder().connectTimeout(CONNECT_TIMEOUT).keepAlive(true).build();

ClientOptions clientOptions =

 ClientOptions.builder().timeoutOptions(timeoutOptions).socketOptions(socketOptions).build();

RedisClient redisClient = RedisClient.create(clientResources, redisUriStandalone);
redisClient.setOptions(clientOptions);
```

듀얼 스택 클러스터(Valkey 및 RedisOSS)에 대한 기본 프로토콜 구성

클러스터 모드가 활성화된 Valkey 또는 Redis OSS 클러스터의 경우 IP Discovery 파라미터를 사용하여 클러스터의 노드에 연결하는 데 사용할 프로토콜 클라이언트를 제어할 수 있습니다. IP 검색 파라미터는 IPv4 또는 로 설정할 수 있습니다IPv6.

Valkey 또는 Redis OSS 클러스터의 경우 IP 검색 파라미터는 클러스터 슬롯(), 클러스터 샤드() 및 클러스터 노드() 출력에 사용되는 IP 프로토콜을 설정합니다. 이러한 명령은 클라이언트가 클러스터 토폴로지를 검색하는 데 사용됩니다. 클라이언트는 이러한 명령IPs의 를 사용하여 클러스터의 다른 노드에 연결합니다.

IP Discovery를 변경해도 연결된 클라이언트는 가동 중지되지 않습니다. 하지만 변경 사항이 전파되려면 다소 시간이 소요됩니다. 변경 사항이 Valkey 또는 Redis OSS 클러스터에 대해 완전히 전파된 시기를 확인하려면 의 출력을 모니터링합니다`cluster slots`. 새 프로토콜과 IPs 함께 클러스터 슬롯 명령 보고서에서 반환된 모든 노드가 전송을 완료하면 변경 사항이 적용됩니다.

Redis-Py 사용 예제:

```
cluster = RedisCluster(host="xxxx", port=6379)
target_type = IPv6Address # Or IPv4Address if changing to IPv4

nodes = set()
while len(nodes) == 0 or not all((type(ip_address(host)) is target_type) for host in
 nodes):
    nodes = set()

   # This refreshes the cluster topology and will discovery any node updates.
   # Under the hood it calls cluster slots
    cluster.nodes_manager.initialize()
    for node in cluster.get_nodes():
        nodes.add(node.host)
    self.logger.info(nodes)

    time.sleep(1)
```

Lettuce 사용 예제:

```
RedisClusterClient clusterClient = RedisClusterClient.create(RedisURI.create("xxxx",
 6379));

Class targetProtocolType = Inet6Address.class; // Or Inet4Address.class if you're
 switching to IPv4

Set<String> nodes;

do {
   // Check for any changes in the cluster topology.
   // Under the hood this calls cluster slots
```

```
    clusterClient.refreshPartitions();
    Set<String> nodes = new HashSet<>();

    for (RedisClusterNode node : clusterClient.getPartitions().getPartitions()) {
        nodes.add(node.getUri().getHost());
    }

    Thread.sleep(1000);
} while (!nodes.stream().allMatch(node -> {
            try {
                return finalTargetProtocolType.isInstance(InetAddress.getByName(node));
            } catch (UnknownHostException ignored) {}
            return false;
}));
```

클라이언트 모범 사례(Memcached)

효율적인 로드 밸런싱을 위해 ElastiCache 클라이언트 구성(Memcached)

> **Note**
>
> 이 섹션의 내용은 자체 설계된 다중 노드 Memcached 클러스터에 적용됩니다.

여러 ElastiCache Memcached 노드를 효과적으로 사용하려면 캐시 키를 노드에 분산할 수 있어야 합니다. n 노드 사용 클러스터를 로드 밸런스하는 간단한 방법은 객체 키의 해시를 계산하고 n - `hash(key) mod n`으로 결과를 모드하는 것입니다. 결과 값(0~n–1)은 객체가 배치되는 노드의 수입니다.

노드 수(n)가 일정하면 이 방법이 간편하고 효과적입니다. 하지만 클러스터에서 노드를 추가하거나 제거할 때마다 이동할 키의 수는 (n - 1) / n입니다. 여기서 n은 새로운 노드 수입니다. 따라서 이 방법을 사용하면 많은 키가 이동하며 특히 노드 수가 커질수록 다량의 초기 캐시가 누락됩니다. 1개에서 2개로 노드를 조정하면 이동할 키는 (2-1)/2(50%)가 되는 것이 모범 사례입니다. 9개에서 10개로 노드를 조정하면 이동할 키는 (10-1)/10(90%)가 됩니다. 트래픽의 스파이크로 인해 확장하는 경우 캐시가 대량으로 누락되는 상황을 방지할 필요가 없습니다. 캐시가 대량으로 누락되면 데이터베이스에 대한 히트가 발생하지만 이미 트래픽의 스파이크로 인해 오버로드되어 있습니다.

이 문제의 해결 방법은 일관적 해싱입니다. 일관적 해싱에는 클러스터에서 노드가 추가되거나 제거될 때마다 이동할 키의 수가 대략 1/n(여기서 n은 새로운 노드 수)인 알고리즘이 사용됩니다. 1개에서 2개

로 노드를 조정하면 이동할 키는 1/2(50%)이 되는 것이 최악의 경우입니다. 9개에서 10개로 노드를 조정하면 이동할 키는 1/10(10%)가 됩니다.

사용자는 다중 노드 클러스터에 사용되는 해시 알고리즘을 제어합니다. 일관적 해싱을 사용하도록 클라이언트를 구성하는 것이 좋습니다. 일관적 해싱을 구현하는 데 자주 사용되는 언어로 여러 Memcached 클라이언트 라이브러리가 제공됩니다. 사용할 라이브러리의 설명서에서 일관적 해싱 지원 여부와 그 구현 방법을 참조하세요.

Java, PHP또는 에서 작업하는 경우 Amazon ElastiCache 클라이언트 라이브러리 중 하나를 사용하는 NET것이 좋습니다.

Java를 사용한 일관적 해싱

ElastiCache Memcached Java 클라이언트는 일관된 해싱 기능이 내장된 오픈 소스 spymemcached Java 클라이언트를 기반으로 합니다. 라이브러리에는 일관된 해싱을 구현하는 KetamaConnectionFactory 클래스가 포함되어 있습니다. 기본적으로 spymemcached에서는 일관적 해싱이 해제됩니다.

자세한 내용은 의 KetamaConnectionFactory 설명서를 참조하세요KetamaConnectionFactory.

를 MemcachedPHP와 함께 사용하여 일관된 해싱

ElastiCache Memcached PHP 클라이언트는 내장 Memcached PHP 라이브러리 주변의 래퍼입니다. 기본적으로 일관된 해싱은 Memcached PHP 라이브러리에서 해제됩니다.

다음 코드를 사용하여 일관적 해싱을 설정하세요.

```
$m = new Memcached();
$m->setOption(Memcached::OPT_DISTRIBUTION, Memcached::DISTRIBUTION_CONSISTENT);
```

또한 php.ini 파일에서 `memcached.sess_consistent_hash`를 설정할 수도 있습니다.

자세한 내용은 http://php.net/manual/en/memcached.configuration.php Memcached의 런타임 구성 설명서를 참조PHP하세요. 특히 `memcached.sess_consistent_hash` 파라미터에 유의하시기 바랍니다.

를 사용한 일관된 해싱.NET 및 Memcached

ElastiCache Memcached .NET 클라이언트는 Enyim Memcached 주변의 래퍼입니다. 기본적으로 Enyim Memcached 클라이언트에서는 일관적 해싱이 설정됩니다.

자세한 내용은 https://github.com/enyim/EnyimMemcached/wiki/MemcachedClient-Configuration#user-content-memcachedlocator의 `memcached/locator` 설명서를 참조하세요.

Memcached를 사용하여 검증된 클라이언트

다음 클라이언트는 Memcached에 대해 지원되는 모든 네트워크 유형 구성에서 작동하도록 특별히 검증되었습니다.

검증된 클라이언트:

- AWS ElastiCache Php용 클러스터 클라이언트 Memcached - 버전 *3.6.2
- AWS ElastiCache Java용 클러스터 클라이언트 Memcached - Github의 최신 마스터

듀얼 스택 클러스터에 대한 기본 프로토콜 구성(Memcached)

Memcached 클러스터의 경우, 클라이언트가 클러스터 내 노드에 연결하는 데 사용할 프로토콜을 IP Discovery 파라미터로 제어할 수 있습니다. IP Discovery 파라미터는 IPv4 또는 로 설정할 수 있습니다 IPv6.

IP Discovery 파라미터는 config get 클러스터 출력에 사용되는 IP 프로토콜을 제어합니다. 그러면 ElastiCache (Memcached) 클러스터에 대한 자동 검색을 지원하는 클라이언트에서 사용하는 IP 프로토콜이 결정됩니다.

IP Discovery를 변경해도 연결된 클라이언트는 가동 중지되지 않습니다. 하지만 변경 사항이 전파되려면 다소 시간이 소요됩니다.

Java의 경우 `getAvailableNodeEndPoints` 출력을, Php의 경우 `getServerList` 출력을 모니터링합니다. 이러한 함수의 출력이 업데이트된 프로토콜을 사용하는 클러스터의 모든 노드에 IPs 대해 보고되면 변경 사항의 전파가 완료되었습니다.

Java 예제:

```
MemcachedClient client = new MemcachedClient(new InetSocketAddress("xxxx", 11211));

Class targetProtocolType = Inet6Address.class; // Or Inet4Address.class if you're
 switching to IPv4

Set<String> nodes;

do {
```

```
    nodes =
 client.getAvailableNodeEndPoints().stream().map(NodeEndPoint::getIpAddress).collect(Collectors

    Thread.sleep(1000);
} while (!nodes.stream().allMatch(node -> {
            try {
                return finalTargetProtocolType.isInstance(InetAddress.getByName(node));
            } catch (UnknownHostException ignored) {}
            return false;
        }));
```

Php 예제:

```
$client = new Memcached;
$client->setOption(Memcached::OPT_CLIENT_MODE, Memcached::DYNAMIC_CLIENT_MODE);
$client->addServer("xxxx", 11211);

$nodes = [];
$target_ips_count = 0;
do {
    # The PHP memcached client only updates the server list if the polling interval has
 expired and a
    # command is sent
    $client->get('test');

    $nodes = $client->getServerList();

    sleep(1);
    $target_ips_count = 0;

    // For IPv4 use FILTER_FLAG_IPV4
    $target_ips_count = count(array_filter($nodes, function($node) { return
 filter_var($node["ipaddress"], FILTER_VALIDATE_IP, FILTER_FLAG_IPV6); }));

} while (count($nodes) !== $target_ips_count);
```

IP Discovery가 업데이트되기 전에 생성된 기존 클라이언트 연결은 업데이트 전의 프로토콜을 사용하여 연결됩니다. 클러스터 검색 명령의 출력에서 변경 사항이 감지되면, 검증된 모든 클라이언트가 새 IP 프로토콜을 사용하여 자동으로 클러스터에 재연결됩니다. 하지만, 클라이언트 구현에 따라 달라질 수 있습니다.

TLS 활성화된 듀얼 스택 ElastiCache 클러스터

ElastiCache 클러스터에 대해 TLS 가 활성화되면 클러스터 검색 함수(`cluster slots`Redis의 `cluster nodes` 경우 , `cluster shards`,)가 활성화되고 `config get cluster` Memcached의 경우 대신 호스트 이름을 반환합니다IPs. 그런 다음 ElastiCache 클러스터IPs에 연결하고 TLS 핸드셰이크를 수행하는 대신 호스트 이름이 사용됩니다. 따라서 클라이언트가 IP Discovery 파라미터의 영향을 받지 않습니다. TLS 활성화된 클러스터의 경우 IP 검색 파라미터는 기본 IP 프로토콜에 영향을 주지 않습니다. 대신 사용되는 IP 프로토콜은 클라이언트가 DNS 호스트 이름을 확인할 때 선호하는 IP 프로토콜에 따라 결정됩니다.

Java 클라이언트

IPv4 및 를 모두 지원하는 Java 환경에서 연결할 때 IPv6Java는 기본적으로 이전 버전과의 호환성 IPv4IPv6을 선호합니다. 그러나 JVM 인수를 통해 IP 프로토콜 기본 설정을 구성할 수 있습니다. IPv4를 선호하려면 가 `-Djava.net.preferIPv4Stack=true` 및 를 JVM 수락하고 IPv6 를 선호합니다`-Djava.net.preferIPv6Stack=true`. 설정`-Djava.net.preferIPv4Stack=true`하면 JVM 가 더 이상 IPv6 연결하지 않습니다. Valkey 또는 Redis 의 경우 OSS여기에는 다른 비-Valkey 및 비-Redis OSS 애플리케이션에 대한 애플리케이션이 포함됩니다.

호스트 수준 기본 설정

일반적으로 클라이언트 또는 클라이언트 런타임이 IP 프로토콜 기본 설정을 위한 구성 옵션을 제공하지 않는 경우 DNS 해결을 수행할 때 IP 프로토콜은 호스트의 구성에 따라 달라집니다. 기본적으로 대부분의 호스트는 IPv6 우선하지만 IPv4 이 기본 설정은 호스트 수준에서 구성할 수 있습니다. 이는 ElastiCache 클러스터에 대한 DNS 요청뿐만 아니라 해당 호스트의 모든 요청에 영향을 미칩니다.

Linux 호스트

Linux의 경우, `gai.conf` 파일을 수정하여 IP 프로토콜 기본 설정을 구성할 수 있습니다. `gai.conf` 파일은 `/etc/gai.conf`에서 찾을 수 있습니다. 지정된 `gai.conf`가 없으면 예제를 `/usr/share/doc/glibc-common-x.xx/gai.conf`에서 찾을 수 있는데, 이를 `/etc/gai.conf`에 복사하면 기본 구성의 주석을 제거할 수 있습니다. ElastiCache 클러스터에 연결할 IPv4 때 선호하도록 구성을 업데이트하려면 클러스터를 포함하는 CIDR 범위의 우선 순위를 기본 IPv6 연결의 우선 순위보다 높 IPs게 업데이트합니다. 기본적으로 IPv6 연결의 우선 순위는 40입니다. 예를 들어 클러스터가 CIDR 172.31.0.0:0/16이 있는 서브넷에 있다고 가정하면 아래 구성으로 인해 클라이언트는 해당 클러스터에 대한 IPv4 연결을 선호합니다.

```
label ::1/128       0
label ::/0          1
label 2002::/16     2
```

```
label ::/96         3
label ::ffff:0:0/96 4
label fec0::/10     5
label fc00::/7      6
label 2001:0::/32   7
label ::ffff:172.31.0.0/112 8
#
#    This default differs from the tables given in RFC 3484 by handling
#    (now obsolete) site-local IPv6 addresses and Unique Local Addresses.
#    The reason for this difference is that these addresses are never
#    NATed while IPv4 site-local addresses most probably are.  Given
#    the precedence of IPv6 over IPv4 (see below) on machines having only
#    site-local IPv4 and IPv6 addresses a lookup for a global address would
#    see the IPv6 be preferred.  The result is a long delay because the
#    site-local IPv6 addresses cannot be used while the IPv4 address is
#    (at least for the foreseeable future) NATed.  We also treat Teredo
#    tunnels special.
#
# precedence  <mask>   <value>
#    Add another rule to the RFC 3484 precedence table.  See section 2.1
#    and 10.3 in RFC 3484.  The default is:
#
precedence  ::1/128       50
precedence  ::/0          40
precedence  2002::/16     30
precedence ::/96          20
precedence ::ffff:0:0/96  10
precedence ::ffff:172.31.0.0/112 100
```

`gai.conf`에 대한 자세한 내용은 Linux 기본 페이지에서 확인할 수 있습니다.

Windows 호스트

Windows 호스트의 프로세스도 비슷합니다. Windows 호스트의 경우, `netsh interface ipv6 set prefix CIDR_CONTAINING_CLUSTER_IPS PRECEDENCE LABEL`을 실행할 수 있습니다. 이는 Linux 호스트에서 `gai.conf` 파일을 수정할 때와 효과가 동일합니다.

이렇게 하면 지정된 CIDR 범위의 IPv4 연결보다 IPv6 연결을 선호하도록 기본 설정 정책이 업데이트됩니다. 예를 들어 클러스터가 172.31.0.0:0/16이 CIDR 실행 중인 서브넷에 있다고 가정`netsh interface ipv6 set prefix ::ffff:172.31.0.0:0/112 100 15`하면 다음과 같은 우선 순위 테이블이 생성되어 클러스터에 연결할 IPv4 때 클라이언트가 선호합니다.

```
C:\Users\Administrator>netsh interface ipv6 show prefixpolicies
```

```
Querying active state...

Precedence Label Prefix
---------- ----- --------------------------------
100 15 ::ffff:172.31.0.0:0/112
20 4 ::ffff:0:0/96
50 0 ::1/128
40 1 ::/0
30 2 2002::/16
5 5 2001::/32
3 13 fc00::/7
1 11 fec0::/10
1 12 3ffe::/16
1 3 ::/96
```

Valkey 및 Redis용 예약 메모리 관리 OSS

예약된 메모리는 비데이터 사용을 위해 구분된 메모리입니다. 백업 또는 장애 조치를 수행할 때 Valkey와 Redis는 클러스터의 데이터가 .rdb 파일에 기록되는 동안 사용 가능한 메모리를 OSS 사용하여 클러스터에 대한 쓰기 작업을 기록합니다. 모든 쓰기에 사용 가능한 메모리를 충분히 확보하지 못하면 프로세스가 실패합니다. 다음은 ElastiCache (Redis OSS)용 예약 메모리를 관리하기 위한 옵션과 이러한 옵션을 적용하는 방법에 대한 정보를 찾을 수 있습니다.

주제

필요한 예약된 메모리의 양

OSS 이전에 Redis 버전을 실행하는 경우 Redis OSS2.8.22 이상을 실행하는 경우보다 백업 및 장애 조치를 위해 더 많은 메모리를 예약합니다. 이 요구 사항은 ElastiCache (Redis OSS)가 백업 프로세스를 구현하는 다양한 방법으로 인한 것입니다. 엄지손가락 규칙은 노드 유형 `maxmemory` 값의 절반을 2.8.22 이전 버전의 경우 Redis OSS 오버헤드에, 1/4은 Redis OSS 버전 2.8.22 이상에 예약하는 것입니다.

백업 및 복제 프로세스를 ElastiCache 구현하는 다양한 방법으로 인해 썸의 규칙은 `reserved-memory-percent` 파라미터를 사용하여 노드 유형 `maxmemory` 값의 25%를 예약하는 것입니다. 이는 기본값이며 대부분의 경우에 권장됩니다.

버스트 가능한 마이크로 및 스몰 인스턴스 유형이 `maxmemory` 제한 근처에서 작동하는 경우 스왑 사용량이 발생할 수 있습니다. 백업, 복제 및 트래픽이 많은 동안 이러한 인스턴스 유형의 운영 안정성을 개선하려면 작은 인스턴스 유형의 경우 `reserved-memory-percent` 파라미터 값을 최대 30%, 마이크로 인스턴스 유형의 경우 최대 50% 늘리는 것이 좋습니다.

데이터 계층화가 있는 ElastiCache 클러스터의 쓰기 작업이 많은 워크로드의 경우 노드에서 사용할 수 있는 메모리의 최대 50%`reserved-memory-percent`까지 를 늘리는 것이 좋습니다.

자세한 내용은 다음 자료를 참조하세요.

- Valkey 또는 Redis OSS 스냅샷을 생성하기에 충분한 메모리 확보
- 동기화 및 백업 구현 방법
- 의 데이터 계층화 ElastiCache

예약된 메모리를 관리하기 위한 파라미터

2017년 3월 16일부터 Amazon ElastiCache 은 Valkey 또는 Redis OSS 메모리를 관리하기 위한 두 개의 상호 배타적인 파라미터 `reserved-memory` 및 를 제공합니다`reserved-memory-percent`. 이러한 파라미터 중 어느 것도 Valkey 또는 Redis OSS 배포의 일부가 아닙니다.

ElastiCache 고객이 된 시기에 따라 이러한 파라미터 중 하나 또는 다른 파라미터가 기본 메모리 관리 파라미터입니다. 이 파라미터는 새 Valkey 또는 Redis OSS 클러스터 또는 복제 그룹을 생성하고 기본 파라미터 그룹을 사용할 때 적용됩니다.

- 2017년 3월 16일 이전에 시작한 고객의 경우 - 기본 파라미터 그룹을 사용하여 Redis OSS 클러스터 또는 복제 그룹을 생성할 때 메모리 관리 파라미터는 입니다`reserved-memory`. 이 경우 0바이트의 메모리가 예약됩니다.
- 2017년 3월 16일 이후에 시작한 고객의 경우 - 기본 파라미터 그룹을 사용하여 Valkey 또는 Redis OSS 클러스터 또는 복제 그룹을 생성할 때 메모리 관리 파라미터는 입니다`reserved-memory-percent`. 이 경우 노드 `maxmemory` 값의 25%가 비데이터 용도로 예약됩니다.

두 가지 Valkey 또는 Redis OSS 메모리 관리 파라미터에 대해 읽은 후 기본값이 아니거나 기본값이 아닌 파라미터를 사용하는 것이 좋습니다. 이 경우, 다른 예약된 메모리 관리 파라미터로 변경할 수 있습니다.

이 파라미터의 값을 변경하려면 사용자 정의 파라미터 그룹을 생성하고 기본 설정 메모리 관리 파라미터 및 값을 사용하도록 수정할 수 있습니다. 그런 다음 새 Valkey 또는 Redis OSS 클러스터 또는 복제 그룹을 생성할 때마다 사용자 지정 파라미터 그룹을 사용할 수 있습니다. 기존 클러스터나 복제 그룹의 경우 사용자 지정 파라미터 그룹을 사용하도록 수정할 수 있습니다.

자세한 내용은 다음 자료를 참조하세요.

- 예약된 메모리 관리 파라미터 지정
- ElastiCache 파라미터 그룹 생성
- ElastiCache 파라미터 그룹 수정
- ElastiCache 클러스터 수정
- 복제 그룹 수정

reserved-memory 파라미터

2017년 3월 16일 이전에는 파라미터 를 사용하여 모든 ElastiCache (Redis OSS) 예약 메모리 관리를 수행했습니다`reserved-memory`. `reserved-memory`의 기본값은 0입니다. 이 기본값은 Valkey 또

는 Redis OSS 오버헤드에 대한 메모리를 예약하지 않으며 Valkey 또는 Redis가 데이터와 함께 노드의 모든 메모리OSS를 사용할 수 있도록 허용합니다.

백업 및 장애 조치를 위해 충분한 메모리를 사용할 수 있도록 reserved-memory를 변경할 경우 사용자 지정 파라미터 그룹을 생성해야 합니다. 이 사용자 지정 파라미터 그룹에서는 클러스터 및 클러스터의 노드 유형에서 실행되는 Valkey 또는 Redis OSS 버전에 적합한 reserved-memory 값으로 설정합니다. 자세한 내용은 필요한 예약된 메모리의 양 단원을 참조하세요.

파라미터는 에 reserved-memory ElastiCache 특정되며 일반 Redis OSS 배포의 일부가 아닙니다.

다음 절차에서는 를 사용하여 Valkey 또는 Redis OSS 클러스터의 메모리reserved-memory를 관리하는 방법을 보여줍니다.

reserved-memory를 사용하여 메모리를 예약하려면

1. 실행 중인 엔진 버전과 일치하는 파라미터 그룹 패밀리를 지정하는 사용자 지정 파라미터 그룹을 생성합니다. 예를 들어 redis2.8 파라미터 그룹 패밀리를 지정합니다. 자세한 내용은 ElastiCache 파라미터 그룹 생성 단원을 참조하십시오.

   ```
   aws elasticache create-cache-parameter-group \
      --cache-parameter-group-name redis6x-m3xl \
      --description "Redis OSS 2.8.x for m3.xlarge node type" \
      --cache-parameter-group-family redis6.x
   ```

2. Valkey 또는 Redis OSS 오버헤드에 대해 예약할 메모리 바이트 수를 계산합니다. OSS 노드 유형별 파라미터 재정의에서 노드 유형에 대한 maxmemory 값을 찾을 수 있습니다.
3. reserved-memory 파라미터가 이전 단계에서 계산된 바이트 수가 되도록 사용자 지정 파라미터 그룹을 수정합니다. 다음 AWS CLI 예제에서는 2.8.22 OSS 이전에 Redis 버전을 실행 중이며 노드 의 절반을 예약해야 한다고 가정합니다maxmemory. 자세한 내용은 ElastiCache 파라미터 그룹 수정 단원을 참조하십시오.

   ```
   aws elasticache modify-cache-parameter-group \
      --cache-parameter-group-name redis28-m3xl \
      --parameter-name-values "ParameterName=reserved-memory,
    ParameterValue=7130316800"
   ```

   각 노드 유형에는 다른 maxmemory 값이 있으므로 사용할 각 노드 유형에 대해 별도의 사용자 지정 파라미터 그룹이 필요합니다. 따라서 각 노드 유형에는 reserved-memory에 대해 서로 다른 값이 필요합니다.
4. 사용자 지정 파라미터 그룹을 사용하도록 Redis OSS 클러스터 또는 복제 그룹을 수정합니다.

다음 CLI 예제에서는 사용자 지정 파라미터 그룹을 redis28-m3xl 즉시 my-redis-cluster 사용하도록 클러스터를 수정합니다. 자세한 내용은 ElastiCache 클러스터 수정 단원을 참조하십시오.

```
aws elasticache modify-cache-cluster \
   --cache-cluster-id my-redis-cluster \
   --cache-parameter-group-name redis28-m3xl \
   --apply-immediately
```

다음 CLI 예제에서는 즉시 redis28-m3xl 시작되는 사용자 지정 파라미터 그룹을 my-redis-repl-grp 사용하도록 복제 그룹을 수정합니다. 자세한 설명은 복제 그룹 수정 섹션을 참조하십시오.

```
aws elasticache modify-replication-group \
   --replication-group-id my-redis-repl-grp \
   --cache-parameter-group-name redis28-m3xl \
   --apply-immediately
```

reserved-memory-percent 파라미터

2017년 3월 16일 Amazon은 파라미터를 ElastiCache 도입reserved-memory-percent하여 ElastiCache (Redis)의 모든 버전에서 사용할 수 있도록 했습니다OSS. reserved-memory-percent의 목적은 모든 클러스터에 대해 예약된 메모리 관리를 간소화하는 것입니다. 노드 유형과 상관없이 클러스터의 예약된 메모리를 관리하기 위해 각 파라미터 그룹 패밀리에 대해 단일 파라미터 그룹(예: redis2.8)을 보유할 수 있도록 하여 이를 수행합니다. reserved-memory-percent에 대한 기본값은 25(25%)입니다.

파라미터는 에 reserved-memory-percent 특정 ElastiCache 되며 일반 Redis OSS 배포의 일부가 아닙니다.

클러스터가 r6gd 패밀리의 노드 유형을 사용하고 있고 메모리 사용량이 75%에 도달하는 경우 데이터 계층화가 자동으로 트리거됩니다. 자세한 내용은 의 데이터 계층화 ElastiCache 단원을 참조하십시오.

를 사용하여 메모리를 예약하려면 reserved-memory-percent

reserved-memory-percent 를 사용하여 ElastiCache (Redis OSS) 클러스터의 메모리를 관리하려면 다음 중 하나를 수행합니다.

- Redis OSS 2.8.22 이상을 실행하는 경우 클러스터에 기본 파라미터 그룹을 할당합니다. 기본 25%가 적절합니다. 그렇지 않은 경우 다음 설명된 단계에 따라 값을 변경합니다.
- 2.8.22 OSS 이전에 Redis 버전을 실행하는 경우 `reserved-memory-percent`의 기본 25%보다 많은 메모리를 예약해야 할 수 있습니다. 이렇게 하려면 다음 절차를 사용하세요.

의 백분율 값을 변경하려면 reserved-memory-percent

1. 실행 중인 엔진 버전과 일치하는 파라미터 그룹 패밀리를 지정하는 사용자 지정 파라미터 그룹을 생성합니다. 예를 들어 `redis2.8` 파라미터 그룹 패밀리를 지정합니다. 기본 파라미터 그룹을 수정할 수 없으므로 사용자 지정 파라미터 그룹이 필요합니다. 자세한 내용은 ElastiCache 파라미터 그룹 생성 단원을 참조하십시오.

   ```
   aws elasticache create-cache-parameter-group \
      --cache-parameter-group-name redis28-50 \
      --description "Redis OSS 2.8.x 50% reserved" \
      --cache-parameter-group-family redis2.8
   ```

   `reserved-memory-percent`는 노드의 `maxmemory`%로 메모리를 예약하므로 각 노드 유형에 대해 사용자 지정 파라미터 그룹이 필요하지 않습니다.

2. `reserved-memory-percent`가 50(50%)이 되도록 사용자 지정 파라미터 그룹을 수정합니다. 자세한 내용은 ElastiCache 파라미터 그룹 수정 단원을 참조하십시오.

   ```
   aws elasticache modify-cache-parameter-group \
      --cache-parameter-group-name redis28-50 \
      --parameter-name-values "ParameterName=reserved-memory-percent,
    ParameterValue=50"
   ```

3. 2.8.22 OSS 이전의 Redis 버전을 실행하는 모든 Redis OSS 클러스터 또는 복제 그룹에 이 사용자 지정 파라미터 그룹을 사용합니다.

   다음 CLI 예제에서는 사용자 지정 파라미터 그룹을 `redis28-50` 즉시 `my-redis-cluster` 사용하도록 Redis OSS 클러스터를 수정합니다. 자세한 내용은 ElastiCache 클러스터 수정 단원을 참조하십시오.

   ```
   aws elasticache modify-cache-cluster \
      --cache-cluster-id my-redis-cluster \
      --cache-parameter-group-name redis28-50 \
      --apply-immediately
   ```

다음 CLI 예제에서는 사용자 지정 파라미터 그룹을 `redis28-50` 즉시 `my-redis-repl-grp` 사용하도록 Redis OSS 복제 그룹을 수정합니다. 자세한 내용은 복제 그룹 수정 단원을 참조하십시오.

```
aws elasticache modify-replication-group \
   --replication-group-id my-redis-repl-grp \
   --cache-parameter-group-name redis28-50 \
   --apply-immediately
```

예약된 메모리 관리 파라미터 지정

2017년 3월 16일에 현재 ElastiCache 고객인 경우 기본 예약 메모리 관리 파라미터는 예약 메모리의 영(0) 바이트`reserved-memory`입니다. 2017년 3월 16일 이후에 ElastiCache 고객이 된 경우 기본 예약 메모리 관리 파라미터는 노드 메모리 예약의 25%`reserved-memory-percent`입니다. (ElastiCache Redis OSS) 클러스터 또는 복제 그룹을 생성한 시기와 관계없이 마찬가지입니다. 그러나 AWS CLI 또는 를 사용하여 예약 메모리 관리 파라미터를 변경할 수 있습니다 ElastiCache API.

파라미터 `reserved-memory` 및 `reserved-memory-percent`는 함께 사용할 수 없습니다. 파라미터 그룹에는 항상 한 개의 파라미터가 있으며 둘 다 있을 수 없습니다. 파라미터 그룹을 수정하여 파라미터 그룹에서 예약된 메모리 관리에 사용할 파라미터를 변경할 수 있습니다. 기본 파라미터 그룹을 수정할 수 없으므로 파라미터 그룹은 사용자 지정 파라미터 그룹이어야 합니다. 자세한 내용은 ElastiCache 파라미터 그룹 생성 단원을 참조하십시오.

지정하려면 reserved-memory-percent

`reserved-memory-percent`를 예약된 메모리 관리 파라미터로 사용하려면 `modify-cache-parameter-group` 명령어를 사용하여 사용자 지정 파라미터 그룹을 수정해야 합니다. `reserved-memory-percent` 파라미터를 사용하여 `parameter-name-values` 및 해당 값을 지정합니다.

다음 CLI 예제에서는 가 예약된 메모리를 관리하는 `reserved-memory-percent` 데 사용하도록 사용자 지정 파라미터 그룹을 수정`redis32-cluster-on`합니다. 예약된 메모리 관리를 위해 파라미터 그룹에서 `ParameterName` 파라미터를 사용할 수 있도록 `ParameterValue`에 값을 지정해야 합니다. 자세한 내용은 ElastiCache 파라미터 그룹 수정 단원을 참조하십시오.

```
aws elasticache modify-cache-parameter-group \
   --cache-parameter-group-name redis32-cluster-on \
   --parameter-name-values "ParameterName=reserved-memory-percent, ParameterValue=25"
```

reserved-memory를 지정하려면

reserved-memory를 예약된 메모리 관리 파라미터로 사용하려면 modify-cache-parameter-group 명령어를 사용하여 사용자 지정 파라미터 그룹을 수정해야 합니다. reserved-memory 파라미터를 사용하여 parameter-name-values 및 해당 값을 지정합니다.

다음 CLI 예제에서는 가 예약 메모리를 관리하는 reserved-memory 데 사용하도록 사용자 지정 파라미터 그룹을 수정redis32-m3xl합니다. 예약된 메모리 관리를 위해 파라미터 그룹에서 ParameterName 파라미터를 사용할 수 있도록 ParameterValue에 값을 지정해야 합니다. 엔진 버전이 2.8.22보다 더 새로운 버전이므로 값을 cache.m3.xlarge의 maxmemory 중 25퍼센트에 해당하는 3565158400으로 설정했습니다. 자세한 내용은 ElastiCache 파라미터 그룹 수정 단원을 참조하십시오.

```
aws elasticache modify-cache-parameter-group \
   --cache-parameter-group-name redis32-m3xl \
   --parameter-name-values "ParameterName=reserved-memory, ParameterValue=3565158400"
```

Valkey 및 Redis OSS 자체 설계 클러스터 작업 모범 사례

다중 AZ 사용, 충분한 메모리, 클러스터 크기 조정 및 가동 중지 최소화는 모두 Valkey 또는 Redis 에서 자체 설계된 클러스터를 사용할 때 기억해야 할 유용한 개념입니다OSS. 여러 모범 사례를 검토하고 따르도록 하세요.

주제

- 다중 AZ로 가동 중지 시간 최소화
- Valkey 또는 Redis OSS 스냅샷을 생성하기에 충분한 메모리 확보
- 온라인 클러스터 크기 조정
- 유지 관리 중 가동 중지 최소화

다중 AZ로 가동 중지 시간 최소화

ElastiCache Valkey 또는 Redis가 기본 노드를 교체해야 할 OSS 수 있는 여러 인스턴스가 있습니다. 여기에는 특정 유형의 계획된 유지 관리와 가능성이 낮은 기본 노드 또는 가용 영역 장애 이벤트가 포함됩니다.

이러한 교체로 인해 클러스터에 약간의 가동 중지가 발생하지만 다중 AZ를 활성화한 경우 가동 중지 시간이 최소화됩니다. 기본 노드의 역할은 자동으로 읽기 전용 복제본 중 하나로 장애 조치됩니다. 는

이를 투명하게 ElastiCache 처리하기 때문에 새 기본 노드를 생성하고 프로비저닝할 필요가 없습니다. 이 장애 조치 및 복제본 승격을 통해 승격이 완료되는 즉시 새 기본 노드에 작성을 재개할 수 있습니다.

다중 AZ 및 가동 중지 시간 최소화에 대한 자세한 Valkey 및 Redis와 함께 다중 AZ ElastiCache 를 사용하여 의 가동 중지 시간 최소화 OSS내용은 섹션을 참조하세요.

Valkey 또는 Redis OSS 스냅샷을 생성하기에 충분한 메모리 확보

Valkey 7.2 이상 및 Redis OSS 버전 2.8.22 이상의 스냅샷 및 동기화

Valkey는 스냅샷 및 동기화를 기본적으로 지원합니다. Redis OSS 2.8.22에서는 동기화 및 저장 중에 스왑 사용량을 늘리지 않고 애플리케이션 사용에 더 많은 메모리를 할당할 수 있는 포크리스 저장 프로세스를 도입합니다. 자세한 내용은 동기화 및 백업 구현 방법 단원을 참조하십시오.

버전 2.8.22 이전의 Redis OSS 스냅샷 및 동기화

ElastiCache (Redis OSS)로 작업할 때 Redis는 다음과 같은 여러 경우에 백그라운드 쓰기 명령을 OSS 호출합니다.

- 백업을 위해 스냅샷을 생성하는 경우
- 복제 그룹에서 기본을 사용하여 복제본을 동기화하는 경우
- Redis 에 대한 추가 전용 파일 기능(AOF)을 활성화하는 경우OSS.
- 복제본을 기본으로 승격하는 경우(기본/복제본 동기화 발생)

Redis가 백그라운드 쓰기 프로세스를 OSS 실행할 때마다 프로세스 오버헤드를 수용할 수 있는 충분한 메모리가 있어야 합니다. 사용 가능한 메모리를 충분히 확보하지 못하면 프로세스가 실패합니다. 따라서 Redis OSS 클러스터를 생성할 때 메모리가 충분한 노드 인스턴스 유형을 선택하는 것이 중요합니다.

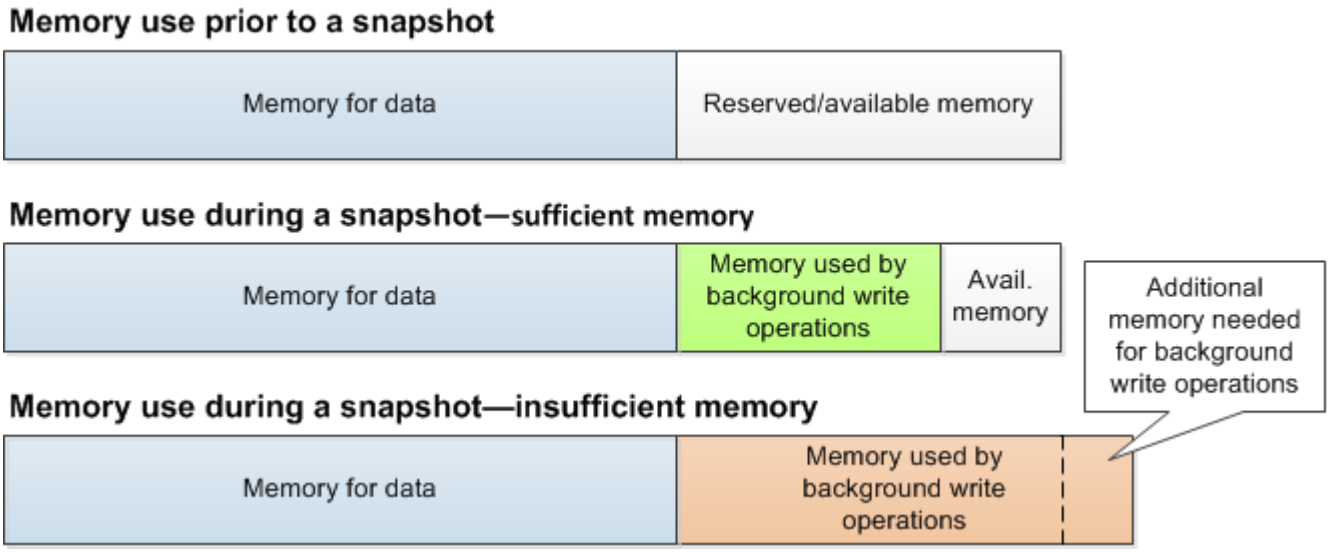
Valkey 및 Redis를 사용한 백그라운드 쓰기 프로세스 및 메모리 사용 OSS

백그라운드 쓰기 프로세스가 호출될 때마다 Valkey와 Redis는 프로세스를 OSS 포크합니다(단일 스레드 엔진임). 하나의 포크는 Redis OSS .rdb 스냅샷 파일의 디스크에 데이터를 유지합니다. 나머지 포크가 모든 읽기 및 쓰기 작업을 처리합니다. 스냅샷이 point-in-time 스냅샷인지 확인하기 위해 모든 데이터 업데이트 및 추가는 데이터 영역과 별도로 사용 가능한 메모리 영역에 기록됩니다.

사용 가능한 메모리가 충분하여 데이터가 계속 디스크에 쓰여지는 동안 모든 쓰기 작업을 기록할 수 있으면 메모리 부족 문제가 발생하지 않습니다. 다음과 같은 경우에 해당되며 메모리 부족 문제가 생기기 쉽습니다.

- 애플리케이션이 여러 쓰기 작업을 수행하여 새로운 데이터나 업데이트된 데이터를 저장하기 위해 사용 가능한 메모리가 대량으로 필요합니다.
- 새로운 데이터나 업데이트된 데이터를 쓸 사용 가능한 메모리가 거의 없습니다.
- 디스크에 계속 쓰기 위해 시간이 오래 걸리는 큰 데이터 세트가 있어 많은 쓰기 작업이 필요합니다.

다음 다이어그램에서는 백그라운드 쓰기 프로세스를 실행할 때의 메모리 사용량을 보여줍니다.



백업이 성능에 미치는 영향에 대한 정보는 자체 설계된 클러스터 백업이 성능에 미치는 영향 섹션을 참조하세요.

Valkey 및 Redis가 스냅샷을 OSS 수행하는 방법에 대한 자세한 내용은 http://valkey.io 참조하세요.

리전 및 가용 영역에 대한 자세한 내용은 에 대한 리전 및 가용 영역 선택 ElastiCache 섹션을 참조하세요.

백그라운드 쓰기를 실행할 때 메모리 부족 방지

BGSAVE 또는 와 같은 백그라운드 쓰기 프로세스가 BGREWRITEAOF 호출될 때마다 프로세스가 실패하지 않도록 하려면 프로세스 중에 쓰기 작업에서 사용할 메모리보다 더 많은 메모리를 사용할 수 있어야 합니다. 최악의 시나리오는 백그라운드 쓰기 작업 중에 모든 레코드가 업데이트되고 일부 새 레코드가 캐시에 추가된다는 것입니다. 따라서 2.8.22 이전의 Redis OSS 버전은 50(50%)reserved-memory-percent으로, Valkey 및 모든 Redis OSS 버전 2.8.22 이상은 25(25%)로 설정하는 것이 좋습니다.

maxmemory 값은 데이터 및 작업 오버헤드에 사용할 수 있는 메모리를 나타냅니다. 기본 파라미터 그룹에서 reserved-memory 파라미터를 수정할 수 없으므로 클러스터의 사용자 지정 파라미터 그룹을 생성해야 합니다. 의 기본값은 0reserved-memory으로, Redis가 모든 최대 메모리OSS를 데이터와

함께 사용할 수 있도록 하므로 백그라운드 쓰기 프로세스와 같은 다른 용도로 메모리가 너무 적을 수 있습니다. 노드 인스턴스 유형에 따른 maxmemory 값은 OSS 노드 유형별 파라미터 재정의 섹션을 참조하세요.

reserved-memory 파라미터를 사용하여 상자에 사용되는 메모리 양을 줄일 수도 있습니다.

의 Valkey 및 Redis별 파라미터에 대한 자세한 내용은 섹션을 ElastiCache참조하세요Valkey 및 Redis OSS 파라미터.

파라미터 그룹 생성 및 수정에 대한 정보는 ElastiCache 파라미터 그룹 생성 및 ElastiCache 파라미터 그룹 수정 섹션을 참조하세요.

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. 따라서 리샤딩 중에 멀티코어 인스턴스의 CPU 사용률을 80% 미만으로 유지하고 단일 코어 인스턴스의 사용률을 50% 미만으로 유지하는 것이 좋습니다. ElastiCache (Redis OSS) 지표를 모니터링하고 애플리케이션이 조정 문제를 관찰하기 전에 재분배를 시작합니다. CPUUtilization, NetworkBytesIn, NetworkBytesOut, CurrConnections, NewConnections, FreeableMemory, SwapUsage 및 BytesUsedForCacheItems 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 다

시 연결합니다. 이렇게 하면 동시에 서버에 연결하는 새로운 연결이 급격하게 증가하는 것을 방지할 수 있습니다.

- 모든 샤드에 함수 로드 - 클러스터를 확장할 때 ElastiCache 는 기존 노드(임의로 선택됨) 중 하나에 로드된 함수를 새 노드(들)에 자동으로 복제합니다. 클러스터에 Valkey 7.2 이상 또는 Redis OSS 7.0 이상이 있고 애플리케이션이 함수 를 사용하는 경우 클러스터가 다른 샤드에서 다른 함수로 끝나지 않도록 스케일 아웃하기 전에 모든 함수를 모든 샤드에 로드하는 것이 좋습니다.

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. 대상 샤드의 데이터는 삭제되지 않습니다.
- 항목이 많은 슬롯은 마이그레이션되지 않습니다. 특히, 직렬화 후 크기가 256MB 이상인 항목이 있는 슬롯은 마이그레이션되지 않습니다.
- 리샤딩 작업 중에는 Lua 스크립트 내에서 FLUSHALL과 FLUSHDB 명령이 지원되지 않습니다. Redis OSS 6 이전에BRPOPLPUSH는 마이그레이션 중인 슬롯에서 작동하는 명령은 지원되지 않습니다.

유지 관리 중 가동 중지 최소화

클러스터 모드 구성은 관리 또는 비관리 작업 중에도 최고의 가용성을 제공합니다. 클러스터 탐색 엔드포인트에 연결되는 클러스터 모드 지원 클라이언트를 사용하는 것이 좋습니다. 클러스터 모드가 비활성화된 경우 모든 쓰기 작업에 기본 엔드포인트를 사용하는 것이 좋습니다.

읽기 활동의 경우 애플리케이션은 클러스터의 어떤 노드에도 연결할 수 있습니다. 기본 엔드포인트와 달리, 노드 엔드포인트는 특정 엔드포인트로 확인됩니다. 복제본을 추가하거나 삭제하는 것과 같이 클러스터를 변경하면 애플리케이션에서 노드 엔드포인트를 업데이트해야 합니다. 따라서 클러스터 모드를 비활성화한 경우 읽기 작업에 리더 엔드포인트를 사용하는 것이 좋습니다.

클러스터에서 AutoFailover 이 활성화된 경우 기본 노드가 변경될 수 있습니다. 따라서, 애플리케이션은 노드의 역할을 확인하고 모든 읽기 엔드포인트를 업데이트해야 합니다. 이렇게 하면 기본에 큰 로드가 발생하지 않도록 하는 데 도움이 됩니다. 비활성화된 노드 AutoFailover의 역할은 변경되지 않습니다. 그러나 관리형 또는 비관리형 작업의 가동 중지 시간은 활성화된 클러스터 AutoFailover에 비해 더 높습니다.

노드를 사용할 수 없으면 읽기 중단이 발생할 수 있으므로 읽기 요청을 단일 읽기 전용 복제본 노드로 보내지 않습니다. 프라이머리에서 읽기 전용으로 대체하거나, 적어도 2개 이상의 읽기 전용 복제본을 마련하여 유지 관리 중 읽기 중단이 일어나지 않도록 합니다.

Memcached에 대한 캐싱 전략

다음 주제에서는 Memcached 캐시를 채우고 유지하기 위한 전략을 찾을 수 있습니다.

캐시를 채우고 유지 관리하기 위해 구현하려는 전략은 캐싱되는 데이터의 유형과 해당 데이터에 대한 액세스 패턴에 따라 달라집니다. 예를 들어, 게임 사이트와 새 이야기 추세의 상위 10개 리더보드에 대해 동일한 전략을 사용하려고 하지 않을 수 있습니다. 이 섹션의 나머지 부분에서는 일반적인 캐시 유지 관리 전략, 이에 대한 장점 및 단점에 대해 살펴봅니다.

주제

지연 로딩

이름에서 알 수 있듯이 지연 로딩은 필요할 때에만 데이터를 캐시에 로드하는 캐싱 전략입니다. 다음 설명과 같이 작동합니다.

Amazon ElastiCache 은 애플리케이션과 애플리케이션이 액세스하는 데이터 스토어(데이터베이스) 사이에 있는 인 메모리 키 값 스토어입니다. 애플리케이션이 데이터를 요청할 때마다 먼저 ElastiCache 캐시에 요청합니다. 캐시에 데이터가 존재하고 최신 상태인 경우 는 데이터를 애플리케이션에 ElastiCache 반환합니다. 데이터가 캐시에 없거나 만료된 경우 애플리케이션에서 데이터 스토어에 데이터를 요청합니다. 데이터 스토어에서 데이터를 애플리케이션에 반환합니다. 다음으로 애플리케이션은 스토어에서 수신한 데이터를 캐시에 씁니다. 이렇게 하면 다음에 요청이 있을 때 데이터를 더 빨리 검색할 수 있습니다.

캐시 적중률은 데이터가 캐시에 있고 만료되지 않은 경우에 발생합니다.

1. 애플리케이션은 캐시에서 데이터를 요청합니다.
2. 캐시는 애플리케이션으로 데이터를 반환합니다.

캐시 누락은 데이터가 캐시에 없거나 만료된 경우에 발생합니다.

1. 애플리케이션은 캐시에서 데이터를 요청합니다.
2. 캐시에 데이터가 요청되지 않으므로 `null`을 반환합니다.

3. 애플리케이션은 데이터베이스에 데이터를 요청하고 수신합니다.
4. 애플리케이션은 새 데이터로 캐시를 업데이트합니다.

지연 로딩의 장점 및 단점

지연 로딩의 장점은 다음과 같습니다.

- 요청된 데이터만 캐싱됩니다.

  대부분의 데이터가 요청되지 않으므로 지연 로딩은 요청되지 않은 데이터가 있는 캐시를 채우지 않습니다.
- 노드 장애가 애플리케이션에 치명적인 영향을 주지 않습니다.

  노드 장애가 발생하여 새로운 빈 노드로 대체될 경우 애플리케이션에서는 지연 시간 증가를 통해 계속 작동합니다. 새 노드에 대한 요청이 발생하면, 각 캐시가 누락될 때마다 데이터베이스 쿼리가 생성됩니다. 동시에 데이터 복사본이 캐시에 추가되어 이후의 요청이 캐시에서 검색됩니다.

지연 로딩의 단점은 다음과 같습니다.

- 캐시 누락 패널티가 있습니다. 각 캐시 누락은 세 개의 이동으로 나타납니다.
  1. 캐시에서 데이터에 대한 초기 요청
  2. 데이터에 대한 데이터베이스의 쿼리
  3. 캐시에 데이터 작성

  이러한 누락으로 인해 애플리케이션으로 데이터를 가져오는 것이 눈에 띄게 지연될 수 있습니다.
- 기한 경과된 데이터

  캐시 누락 시에만 데이터를 캐시에 쓰면, 캐시의 데이터가 기한이 경과할 수 있습니다. 이 결과는 데이터베이스에서 데이터가 변경될 때 캐시에 대한 업데이트가 없기 때문에 발생합니다. 이 문제를 해결하려면 라이트-스루 및 추가 TTL 전략을 사용할 수 있습니다.

지연 로딩 유사 코드 예제

다음은 지연 로딩 로직의 의사(pseudo) 코드 예제입니다.

```
// ****************************************
// function that returns a customer's record.
```

```
// Attempts to retrieve the record from the cache.
// If it is retrieved, the record is returned to the application.
// If the record is not retrieved from the cache, it is
//    retrieved from the database,
//    added to the cache, and
//    returned to the application
// ************************************
get_customer(customer_id)

    customer_record = cache.get(customer_id)
    if (customer_record == null)

        customer_record = db.query("SELECT * FROM Customers WHERE id = {0}",
 customer_id)
        cache.set(customer_id, customer_record)

    return customer_record
```

이 예제의 경우, 데이터를 갖는 애플리케이션 코드는 다음과 같습니다.

```
customer_record = get_customer(12345)
```

라이트-스루

라이트-스루 전략은 데이터베이스에 데이터를 작성할 때마다 데이터를 추가하거나 캐시의 데이터를 업데이트합니다.

라이트-스루의 장점 및 단점

라이트-스루의 장점은 다음과 같습니다.

- 캐시의 데이터가 기한 경과되지 않습니다.

  캐시의 데이터는 데이터베이스에 쓰일 때마다 업데이트되므로 항상 최신 상태입니다.
- 쓰기 패널티 vs. 읽기 패널티

  모든 쓰기에는 다음 2개의 이동이 수반됩니다.

  1. 캐시에 쓰기
  2. 데이터베이스에 쓰기

이로 인해 프로세스에 지연 시간이 추가됩니다. 즉, 최종 사용자는 일반적으로 데이터를 검색할 때보다 데이터를 업데이트할 때 지연 시간에 더 관대합니다. 업데이트는 더 많이 작동하므로 오래 걸릴 수 있다는 고유한 생각이 있습니다.

라이트-스루의 단점은 다음과 같습니다.

- 누락된 데이터

  노드 장애 또는 확장으로 인해 새 노드를 스핀업하면 데이터가 누락됩니다. 이 데이터는 데이터베이스에 추가되거나 업데이트될 때까지 계속 누락됩니다. 라이트-스루로 지연 로딩을 구현하여 이 문제를 최소화할 수 있습니다.

- 캐시 이탈

  대부분의 데이터는 절대 읽히지 않으며 이는 리소스 낭비입니다. 라이브(TTL) 값 에 시간을 추가하여 공간 낭비를 최소화할 수 있습니다.

라이트-스루 의사 코드 예제

다음은 라이트-스루 로직의 의사(pseudo) 코드 예제입니다.

```
// ****************************************
// function that saves a customer's record.
// ****************************************
save_customer(customer_id, values)

    customer_record = db.query("UPDATE Customers WHERE id = {0}", customer_id, values)
    cache.set(customer_id, customer_record)
    return success
```

이 예제의 경우, 데이터를 갖는 애플리케이션 코드는 다음과 같습니다.

```
save_customer(12345,{"address":"123 Main"})
```

추가 TTL

지연 로딩은 기한 경과 데이터에 대해 허용되지만 빈 노드로 인해 실패하지 않습니다. 라이트-스루는 데이터를 항상 최신 상태로 유지하지만, 빈 노드로 인해 실패할 수 있으며 불필요한 데이터로 캐시를

채울 수 있습니다. 각 쓰기에 실시간(TTL) 값을 추가하여 각 전략의 이점을 얻을 수 있습니다. 동시에 추가 데이터로 캐시를 복잡하게 만들지 않을 수 있습니다.

Time to Live(TTL)는 키가 만료될 때까지 초 수를 지정하는 정수 값입니다. Valkey 또는 Redis는 이 값에 초 또는 밀리초를 지정할 OSS 수 있습니다. Memcached는 초 단위로 이 값을 지정합니다. 애플리케이션에서 만료된 키를 읽으려고 하면 키가 없는 것으로 처리됩니다. 데이터베이스가 키에 대해 쿼리되고 캐시가 업데이트됩니다. 이는 값이 기한 경과가 아님을 보장하지 않습니다. 그러나 데이터가 너무 기간 경과되지 않도록 방지하며, 경우에 따라 캐시의 값이 데이터베이스에서 새로 고침되어야 합니다.

자세한 내용은 Valkey 및 Redis OSS 명령 또는 Memcached `set` 명령을 참조하세요.

TTL 의사 코드 예제

다음은 를 사용하는 쓰기 로직의 의사 코드 예제입니다TTL.

```
// ****************************************
// function that saves a customer's record.
// The TTL value of 300 means that the record expires
//    300 seconds (5 minutes) after the set command
//    and future reads will have to query the database.
// ****************************************
save_customer(customer_id, values)

    customer_record = db.query("UPDATE Customers WHERE id = {0}", customer_id, values)
    cache.set(customer_id, customer_record, 300)

    return success
```

다음은 를 사용한 레이지 로드 로직의 의사 코드 예제입니다TTL.

```
// ****************************************
// function that returns a customer's record.
// Attempts to retrieve the record from the cache.
// If it is retrieved, the record is returned to the application.
// If the record is not retrieved from the cache, it is
//    retrieved from the database,
//    added to the cache, and
//    returned to the application.
// The TTL value of 300 means that the record expires
//    300 seconds (5 minutes) after the set command
//    and subsequent reads will have to query the database.
// ****************************************
get_customer(customer_id)
```

```
    customer_record = cache.get(customer_id)

    if (customer_record != null)
        if (customer_record.TTL < 300)
            return customer_record        // return the record and exit function

    // do this only if the record did not exist in the cache OR
    //    the TTL was >= 300, i.e., the record in the cache had expired.
    customer_record = db.query("SELECT * FROM Customers WHERE id = {0}", customer_id)
    cache.set(customer_id, customer_record, 300)  // update the cache
    return customer_record                // return the newly retrieved record and exit
function
```

이 예제의 경우, 데이터를 갖는 애플리케이션 코드는 다음과 같습니다.

```
save_customer(12345,{"address":"123 Main"})
```

```
customer_record = get_customer(12345)
```

관련 주제

- 인 메모리 데이터 스토어
- 엔진 및 버전 선택
- 크기 조정 ElastiCache

에서 자체 설계된 클러스터 관리 ElastiCache

ElastiCache 는 서버리스 캐싱과 자체 설계된 클러스터라는 두 가지 배포 옵션을 제공합니다. 각 에는 고유한 기능과 요구 사항이 있습니다.

이 섹션에는 자체 설계된 클러스터를 관리하는 데 도움이 되는 주제가 포함되어 있습니다.

> **Note**
>
> 이러한 주제는 ElastiCache Serverless에는 적용되지 않습니다.

주제

Auto Scaling Valkey 및 Redis OSS 클러스터

사전 조건

ElastiCache Auto Scaling은 다음으로 제한됩니다.

- Valkey 7.2 이상을 실행하거나 Redis OSS 엔진 버전 6.0 이상을 실행하는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터
- Valkey 7.2 이상을 실행하거나 Redis OSS 엔진 버전 7.0.7 이상을 실행하는 데이터 계층화(클러스터 모드 활성화됨) 클러스터
- 인스턴스 크기 - 대형, XLarge, 2XLarge
- 인스턴스 유형 패밀리 - R7g, R6g, R6gd, R5, M7g, M6g, M5, C7gn
- 의 Auto Scaling ElastiCache 은 글로벌 데이터 스토어, Outpost 또는 로컬 영역에서 실행되는 클러스터에는 지원되지 않습니다.

Valkey 또는 Redis를 사용한 ElastiCache Auto Scaling으로 용량 자동 관리 OSS

ElastiCache Valkey 또는 Redis를 사용한 Auto ScalingOSS은 ElastiCache 서비스에서 원하는 샤드 또는 복제본을 자동으로 늘리거나 줄이는 기능입니다. 는 Application Auto Scaling 서비스를 ElastiCache 활용하여 이 기능을 제공합니다. 자세한 내용은 Application Auto Scaling 섹션을 참조하세요. 자동 조정을 사용하려면 할당한 CloudWatch 지표와 대상 값을 사용하는 조정 정책을 정의하고 적용합니다. ElastiCache 자동 조정은 정책을 사용하여 실제 워크로드에 대한 응답으로 인스턴스 수를 늘리거나 줄입니다.

AWS Management Console 를 사용하여 사전 정의된 지표를 기반으로 조정 정책을 적용할 수 있습니다. `predefined metric`이 열거 형식으로 정의되어 이를 코드의 이름별로 지정하거나 AWS Management Console에서 사용할 수 있습니다. AWS Management Console을 사용하여 선택하는 경

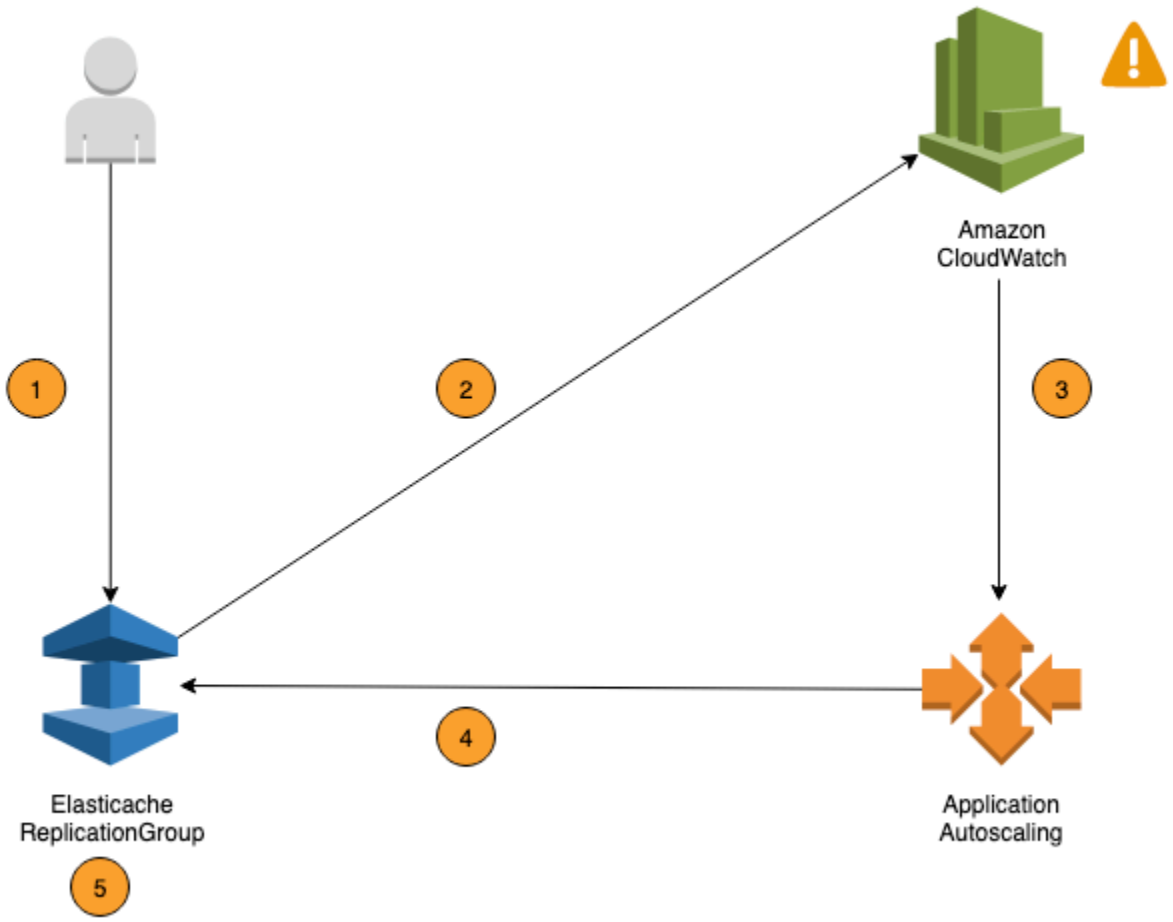
우에는 사용자 지정 지표를 사용할 수 없습니다. 또는 AWS CLI 또는 Application Auto ScalingAPI을 사용하여 사전 정의된 지표 또는 사용자 지정 지표를 기반으로 조정 정책을 적용할 수 있습니다.

ElastiCache 와 함께 Valkey 또는 Redis는 다음 차원에 대한 크기 조정을 OSS 지원합니다.

- 샤드 - 수동 온라인 리샤딩과 유사하게 클러스터에서 샤드를 자동으로 추가/제거합니다. 이 경우 ElastiCache 자동 조정은 사용자를 대신하여 조정을 트리거합니다.
- 복제본 - 수동 복제본 증가/감소 작업과 유사하게 클러스터에서 복제본을 자동으로 추가/제거합니다. ElastiCache Valkey 또는 Redis OSS Auto Scaling을 사용하면 클러스터의 모든 샤드에서 복제본을 균일하게 추가/제거할 수 있습니다.

ElastiCache 와 함께 Valkey 또는 Redis는 다음과 같은 유형의 자동 조정 정책을 OSS 지원합니다.

- 대상 추적 조정 정책 - 특정 지표에 대한 대상 값을 기준으로 서비스가 실행하는 샤드/복제본의 수를 늘리거나 줄입니다. 이 과정은 온도 조절기를 사용하여 집안 온도를 유지하는 방법과 비슷합니다. 사용자가 온도를 선택하면 나머지는 모두 온도 조절기에서 자동으로 수행됩니다.
- Valkey 또는 Redis OSS Auto Scaling ElastiCache 을 사용한 애플리케이션의 예약된 조정 - 날짜 및 시간을 기준으로 서비스가 실행되는 샤드/복제본 수를 늘리거나 줄입니다.



다음 단계에서는 이전 다이어그램과 같이 Valkey 또는 Redis OSS Auto Scaling 프로세스를 ElastiCache 사용하여 를 요약합니다.

1. 복제 그룹에 대한 ElastiCache 자동 조정 정책을 생성합니다.
2. ElastiCache Valkey 또는 Redis를 사용한 자동 조정은 사용자를 대신하여 한 쌍의 CloudWatch 경보를 OSS 생성합니다. 각 쌍은 지표의 상한값과 하한값을 나타냅니다. 이러한 CloudWatch 경보는 클러스터의 실제 사용률이 일정 기간 동안 목표 사용률에서 벗어나면 트리거됩니다. 콘솔에서 경보를 볼 수 있습니다.
3. 구성된 지표 값이 특정 기간 동안 목표 사용률을 초과(또는 목표 미만으로)하는 경우 는 자동 조정을 호출하여 조정 정책을 평가하는 경보를 CloudWatch 트리거합니다.
4. ElastiCache Valkey 또는 Redis OSS Auto Scaling을 사용하면 클러스터 용량을 조정하기 위한 수정 요청이 발생합니다.

5. ElastiCache Valkey 또는 Redis를 사용하면 수정 요청을 OSS 처리하여 클러스터 샤드/복제본 용량을 동적으로 증가(또는 감소)하여 대상 사용률에 도달합니다.

Valkey 또는 Redis OSS Auto Scaling ElastiCache 의 작동 방식을 이해하려면 라는 클러스터가 있다고 가정해 보겠습니다`UsersCluster`. 에 대한 CloudWatch 지표를 모니터링하여 트래픽이 피크에 있을 때 클러스터에 필요한 최대 샤드와 트래픽이 가장 낮은 지점에 있을 때 필요한 최소 샤드를 `UsersCluster`결정합니다. `UsersCluster` 클러스터의 CPU 사용률에 대한 목표 값도 결정합니다. ElastiCache Auto Scaling`UsersCluster`은 대상 추적 알고리즘을 사용하여 의 프로비저닝된 샤드가 필요에 따라 조정되어 사용률이 대상 값과 같거나 가까운 상태로 유지되도록 합니다.

> **Note**
>
> 스케일링에는 상당한 시간이 걸릴 수 있으며 샤드가 리밸런싱되려면 추가 클러스터 리소스가 필요합니다. Valkey 또는 Redis OSS Auto Scaling을 ElastiCache 사용하면 실제 워크로드가 몇 분 동안 상승(또는 우울)된 상태로 유지되는 경우에만 리소스 설정을 수정합니다. Auto Scaling 대상 추적 알고리즘은 장기적으로 대상 사용률을 선택한 값 또는 그 근처에서 유지하려고 합니다.

Auto Scaling 정책

스케일링 정책에는 다음과 같은 구성 요소가 있습니다.

- 대상 지표 - Valkey 또는 Redis OSS Auto Scaling ElastiCache 을 사용하여 언제 얼마나 확장해야 하는지 결정하는 CloudWatch 지표입니다.
- 최소 및 최대 용량 - 크기 조정에 사용할 최소 및 최대 샤드 또는 복제본 수입니다.

> **Important**
>
> Auto Scaling 정책 을 생성하는 동안 현재 용량이 구성된 최대 용량보다 높으면 정책 생성 MaxCapacity 중에 scaleIn 로 이동합니다. 마찬가지로 현재 용량이 구성된 최소 용량보다 낮으면 scaleOut 로 이동합니다 MinCapacity.

- 휴지 기간 - 축소 또는 확장 활동이 완료되고 다른 확장 활동이 시작되기 전의 시간(초 단위)입니다.
- 서비스 연결 역할 - 특정 AWS 서비스에 연결된 AWS 자격 증명 및 액세스 관리(IAM) 역할입니다. 서비스 연결 역할에는 서비스가 사용자를 대신하여 다른 AWS 서비스를 호출하는 데 필요한 모든 권한

이 포함됩니다. Valkey 또는 Redis OSS Auto Scaling을 ElastiCache 사용하면 자동으로 이 역할인 `AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG`가 생성됩니다.

- 스케일 인 활동 활성화 또는 비활성화 - 정책의 스케일 인 활동을 활성화하거나 비활성화할 수 있는 기능입니다.

주제

Auto Scaling을 위한 대상 지표

이 유형의 정책에서는 사전 정의되거나 사용자 지정 지표와 지표의 대상 값이 대상 추적 조정 정책 구성에 지정됩니다. Valkey 또는 Redis OSS Auto Scaling을 ElastiCache 사용하면 조정 정책을 트리거하는 CloudWatch 경보를 생성 및 관리하고 지표 및 대상 값을 기반으로 조정 조정을 계산합니다. 조정 정책은 필요에 따라 샤드/복제본을 추가하거나 제거하여 지표를 지정한 대상 값으로 또는 대상 값에 가깝게 유지합니다. 대상 추적 조정 정책은 지표를 대상 값에 가깝게 유지하는 것 외에도 워크로드 변화로 인한 지표의 변동에 따라 조정되기도 합니다. 이 정책은 클러스터의 사용 가능한 샤드/복제본 수의 급격한 변동을 최소화하기도 합니다.

미리 정의된 평균 `ElastiCachePrimaryEngineCPUUtilization` 지표가 사용되는 조정 정책을 예로 든다면, 이러한 정책은 CPU 사용률을 70%와 같은 지정된 사용률 백분율로 유지하거나 그에 근접할 수 있습니다.

> **Note**
>
> 클러스터마다 대상 지표에 대해 Auto Scaling 정책을 하나씩만 생성할 수 있습니다.

최소 및 최대 용량

샤드

Valkey 또는 Redis OSS Auto Scaling을 사용하여 로 조정할 수 있는 최대 샤드 수 ElastiCache 를 지정할 수 있습니다. 이 값은 250보다 작거나 같아야 하며 최소값은 1입니다. 자동 크기 조정을 통해 관리할

최소 샤드 수를 지정할 수도 있습니다. 이 값은 최소 1이어야 하고, 최대 샤드 수(250)에 지정된 값과 동일하거나 그보다 작아야 합니다.

복제본

Valkey 또는 Redis OSS Auto Scaling ElastiCache 으로 관리할 최대 복제본 수를 지정할 수 있습니다. 이 값은 5보다 작거나 같아야 합니다. 또한 자동 크기 조정을 통해 관리할 최소 복제본 수를 지정할 수도 있습니다. 이 값은 최소 1이어야 하고, 최대 복제본 수(5)에 지정된 값과 동일하거나 그보다 작아야 합니다.

일반 트래픽에서 필요한 샤드/복제본의 최소 및 최대 수를 결정하려면 모델에 대한 예상 트래픽 레이트를 이용해 Auto Scaling 구성을 테스트합니다.

> **Note**
>
> ElastiCache Valkey 또는 Redis OSS Auto Scaling 정책을 사용하면 정의된 최대 크기에 도달하거나 서비스 제한이 적용될 때까지 클러스터 용량이 증가합니다. 한도 증가를 요청하려면 AWS 서비스 한도를 참조하고 한도 유형을 인스턴스 유형별 클러스터당 노드로 선택하세요.

> **Important**
>
> 트래픽이 없는 경우 축소가 발생합니다. 변형의 트래픽이 0이 되면 ElastiCache Valkey 또는 Redis가 지정된 최소 인스턴스 수로 OSS 자동으로 확장됩니다.

휴지 기간

클러스터의 조정에 영향을 미치는 휴지 기간을 추가하여 대상 추적 조정 정책의 응답성을 조정할 수 있습니다. 휴지 기간은 기간이 만료될 때까지 후속 스케일 인 또는 스케일 아웃 요청을 차단합니다. 이렇게 하면 스케일 인 요청의 경우 Valkey 또는 Redis OSS 클러스터 ElastiCache 를 사용하여 에서 샤드/복제본을 삭제하고 스케일 아웃 요청의 경우 샤드/복제본을 생성하는 속도가 느려집니다. 다음과 같은 휴지 기간을 지정할 수 있습니다.

- 스케일 인 활동은 클러스터의 샤드/복제본 수를 줄입니다. 스케일 인 휴지 기간은 스케일 인 활동이 완료되고 다른 스케일 인 활동이 시작되기 전의 시간을 초 단위로 지정합니다.
- 스케일 아웃 활동은 클러스터의 샤드/복제본 수를 증가시킵니다. 스케일 아웃 휴지 기간은 스케일 아웃 활동이 완료되고 다른 스케일 아웃 활동이 시작되기 전의 시간을 초 단위로 지정합니다.

스케일 인 또는 스케일 아웃 휴지 기간을 지정하지 않은 경우 스케일 인의 기본값은 600초이고 스케일 아웃의 기본값은 900초입니다.

스케일 인 활동 활성화 또는 비활성화

정책의 스케일 인 활동을 활성화하거나 비활성화할 수 있습니다. 스케일 인 활동을 활성화하면 조정 정책을 통해 샤드/복제본을 삭제할 수 있습니다. 스케일 인 활동이 활성화되면 조정 정책의 스케일 인 휴지 기간이 스케일 인 활동에 적용됩니다. 스케일 인 활동을 비활성화하면 조정 정책을 통해 샤드/복제본을 삭제할 수 없습니다.

> **Note**
>
> 스케일 아웃 활동은 스케일 아웃 정책이 필요에 따라 Valkey 또는 Redis OSS 샤드/복제본 ElastiCache 을 사용하여 생성할 수 있도록 항상 활성화됩니다.

IAM Auto Scaling에 필요한 권한

ElastiCache Valkey 또는 Redis OSS Auto Scaling을 사용하면 ElastiCache CloudWatch, 및 Application Auto Scaling을 조합하여 사용할 수 있습니다APIs. 클러스터는 ElastiCache (Redis OSS)를 사용하여 생성 및 업데이트되고, 경보는 를 사용하여 생성되며 CloudWatch, 조정 정책은 Application Auto Scaling 을 사용하여 생성됩니다. 클러스터 생성 및 업데이트에 대한 표준 IAM 권한 외에도 ElastiCache Auto Scaling 설정에 액세스하는 IAM 사용자는 동적 조정을 지원하는 서비스에 대한 적절한 권한을 가지고 있어야 합니다. IAM 사용자는 다음 예제 정책에 표시된 작업을 사용할 수 있는 권한이 있어야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:*",
                "elasticache:DescribeReplicationGroups",
                "elasticache:ModifyReplicationGroupShardConfiguration",
                "elasticache:IncreaseReplicaCount",
                "elasticache:DecreaseReplicaCount",
                "elasticache:DescribeCacheClusters",
                "elasticache:DescribeCacheParameters",
                "cloudwatch:DeleteAlarms",
                "cloudwatch:DescribeAlarmHistory",
```

```
                "cloudwatch:DescribeAlarms",
                "cloudwatch:DescribeAlarmsForMetric",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics",
                "cloudwatch:PutMetricAlarm",
                "cloudwatch:DisableAlarmActions",
                "cloudwatch:EnableAlarmActions",
                "iam:CreateServiceLinkedRole",
                "sns:CreateTopic",
                "sns:Subscribe",
                "sns:Get*",
                "sns:List*"
            ],
            "Resource": "arn:aws:iam::123456789012:role/autoscaling-roles-for-cluster"
        }
    ]
}
```

서비스 연결 역할

또한 Valkey 또는 Redis OSS Auto Scaling 서비스가 ElastiCache 포함된 에는 클러스터 및 CloudWatch 경보를 설명하는 권한과 사용자를 대신하여 ElastiCache 대상 용량을 수정할 수 있는 권한이 필요합니다. 클러스터에 대해 Auto Scaling을 활성화하면 라는 서비스 연결 역할이 생성됩니다`AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG`. 이 서비스 연결 역할은 정책의 경보를 설명하고, 플릿의 현재 용량을 모니터링하고, 플릿의 용량을 수정할 수 있는 ElastiCache 자동 조정 권한을 부여합니다. 서비스 연결 역할은 ElastiCache 자동 조정의 기본 역할입니다. 자세한 내용은 Application Auto Scaling 사용 설명서의 [ElastiCache (Redis OSS) Auto Scaling에 대한 서비스 연결 역할을] 참조하세요. Auto Scaling

Auto Scaling 모범 사례

Auto Scaling에 등록하기 전에 다음 작업을 수행하는 것이 좋습니다.

1. 하나의 추적 지표만 사용 - 클러스터에 CPU 또는 데이터 집약적 워크로드가 있는지 식별하고 해당하는 사전 정의된 지표를 사용하여 조정 정책을 정의합니다.
   - 엔진 CPU: `ElastiCachePrimaryEngineCPUUtilization` (샤드 차원) 또는 `ElastiCacheReplicaEngineCPUUtilization` (복제 차원)
   - 데이터베이스 사용: `ElastiCacheDatabaseCapacityUsageCountedForEvictPercentage` 이 크기 조정 정책은 클러스터에서 maxmemory-policy를 noeviction으로 설정한 경우에 가장 잘 작동합니다.

클러스터의 차원당 여러 정책을 피하는 것이 좋습니다. Valkey 또는 Redis OSS Auto 스케일링 ElastiCache 을 사용하면 대상 추적 정책이 스케일 아웃할 준비가 되면 확장 가능한 대상을 스케일 아웃하지만 모든 대상 추적 정책(스케일 인 부분이 활성화된 상태)이 스케일 인할 준비가 된 경우에만 스케일 인됩니다. 여러 정책이 조정 가능한 대상에 스케일 아웃 또는 인을 동시에 지시하는 경우 대상은 스케일 인과 스케일 아웃 모두에 대해 가장 큰 용량을 제공하는 정책에 따라 조정합니다.

2. 대상 추적을 위한 사용자 지정 지표 - Auto Scaling은 정책에 대해 선택한 지표의 변화에 비례하는 스케일 아웃/인에 가장 적합하므로 대상 추적에 사용자 지정 지표를 사용할 경우 주의해야 합니다. 이와 같이 크기 조정 작업에 비례하여 변경되지 않는 지표가 정책 생성에 사용되는 경우 가용성이나 비용에 영향을 줄 수 있는 지속적인 스케일 아웃 또는 스케일 인 작업이 발생할 수 있습니다.

   데이터 계층화 클러스터(r6gd 패밀리 인스턴스 유형)의 경우 크기 조정에 메모리 기반 지표를 사용하지 마세요.

3. 예약된 조정(Scheduled Scaling) - 워크로드가 결정적인지(특정 시간에 상한/하한에 도달) 확인되면 예약 조정을 사용하고 필요에 따라 대상 용량을 구성하는 것이 좋습니다. 대상 추적은 비결정적 워크로드와 필요한 대상 지표에 따라 더 많은 리소스가 필요할 때 스케일 아웃되고 더 적은 리소스가 필요할 때 스케일 인되는 방식으로 작동하는 클러스터에 가장 적합합니다.

4. 스케일 인 사용 중지(Disable Scale-In) - 지표의 급속한 증가/감소는 연속적인 스케일 아웃/인 변동을 트리거할 수 있으므로 대상 추적의 자동 크기 조정은 워크로드가 점진적으로 증가/감소하는 클러스터에 가장 적합합니다. 이러한 변동을 방지하기 위해 스케일 인을 사용 중지한 상태로 시작하고 나중에 언제든지 필요에 따라 수동으로 스케일 인할 수 있습니다.

5. 애플리케이션 테스트(Test your application) - 예상 최소/최대 워크로드를 사용하여 애플리케이션을 테스트하여 가용성 문제를 방지하기 위한 조정 정책을 생성하는 동안 클러스터에 필요한 최소, 최대 샤드/복제본의 절대 수를 확인하는 것이 좋습니다. 자동 크기 조정은 대상에 구성된 최대 임계값까지 스케일 아웃하고 최소 임계값까지 스케일 인할 수 있습니다.

6. 목표 값 정의 - 4주 기간 동안 클러스터 사용률에 대한 해당 CloudWatch 지표를 분석하여 목표 값 임계값을 결정할 수 있습니다. 어떤 값을 선택할지 잘 모르는 경우 지원되는 최소 사전 정의 지표 값으로 시작하는 것이 좋습니다.

7. AutoScaling 대상 추적은 샤드/복제본 차원에 워크로드가 균일하게 분산된 클러스터에 가장 적합합니다. 균일하게 분산되지 않으면 다음과 같은 결과가 발생할 수 있습니다.

   - 사용량이 많은 몇 개의 샤드/복제본에서 워크로드의 급격한 증가/감소로 인해 필요하지 않을 때 조정이 발생할 수 있습니다.
   - 사용량이 많은 샤드/복제본이 있어도 전체 평균이 대상에 가깝기 때문에 필요할 때 조정이 발생하지 않을 수 있습니다.

> **Note**
>
> 클러스터를 확장할 때 ElastiCache 는 기존 노드(임의로 선택됨) 중 하나에 로드된 함수를 새 노드(들)에 자동으로 복제합니다. 클러스터에 Valkey 또는 Redis OSS 7.0 이상이 있고 애플리케이션이 함수 를 사용하는 경우 클러스터가 다른 샤드에서 다른 함수로 끝나지 않도록 스케일 아웃하기 전에 모든 함수를 모든 샤드에 로드하는 것이 좋습니다.

에 등록한 후 다음 사항에 AutoScaling유의하세요.

- 자동 크기 조정의 지원되는 구성에 제한이 있으므로 자동 크기 조정에 등록된 복제 그룹의 구성을 변경하지 않는 것이 좋습니다. 예를 들어, 다음과 같습니다.
  - 인스턴스 유형을 지원되지 않는 유형으로 수동으로 수정합니다.
  - 복제 그룹을 글로벌 데이터 스토어에 연결합니다.
  - `ReservedMemoryPercent` 파라미터를 변경합니다.
  - 정책 생성 중에 구성된 최소/최대 용량을 초과하여 샤드/복제본을 수동으로 늘리거나 줄입니다.

샤드에 Auto Scaling 사용

ElastiCache를 사용하면 Valkey 또는 Redis OSS 엔진에서 추적 및 예약 정책을 사용할 AutoScaling 수 있습니다.

다음은 대상 추적 및 예약된 정책에 대한 세부 정보와 AWS Management Console AWS CLI 및 를 사용하여 정책을 적용하는 방법을 제공합니다APIs.

주제

- 대상 추적 조정 정책
- 조정 정책 추가
- 확장 가능 목표 등록
- 조정 정책 정의
- 스케일 인 활동 비활성화
- 조정 정책 적용
- 조정 정책 편집
- 조정 정책 삭제

- AWS CloudFormation Auto Scaling 정책에 사용
- 예약된 조정

대상 추적 조정 정책

대상 추적 조정 정책을 사용하는 경우 지표를 선택하고 목표 값을 설정합니다. ElastiCache 와 함께 Valkey 또는 Redis OSS Auto Scaling은 조정 정책을 트리거하는 CloudWatch 경보를 생성 및 관리하고 지표와 대상 값을 기반으로 조정 조정을 계산합니다. 조정 정책은 필요에 따라 샤드를 추가하거나 제거하여 지표를 지정한 대상 값으로 또는 대상 값에 가깝게 유지합니다. 대상 추적 조정 정책은 지표를 목표 값에 가깝게 유지하는 것 외에도 로드 패턴의 변화로 인한 지표 변동에 따라 반응하여 플릿의 용량이 갑작스럽게 바뀌는 것을 최소화합니다.

목표 값이 구성되어 있으며 미리 정의된 평균 `ElastiCachePrimaryEngineCPUUtilization` 지표가 사용되는 조정 정책을 예로 든다면, 이러한 정책은 지정된 목표 값에 가깝게 CPU 사용률을 유지할 수 있습니다.

사전 정의된 지표

사전 정의된 지표는 지정된 CloudWatch 지표의 특정 이름, 차원 및 통계(`average`)를 참조하는 구조입니다. Auto Scaling 정책은 클러스터에 대해 다음 사전 정의된 지표 중 하나를 정의합니다.

사전 정의된 지표 유형	CloudWatch 지표 이름	CloudWatch 지표 차원	부적격 인스턴스 유형
`ElastiCachePrimaryEngineCPUUtilization`	`EngineCPUUtilization`	ReplicationGroupId, 역할 = 기본	None
`ElastiCacheDatabaseCapacityUsageCountedForEvictPercentage`	`DatabaseCapacityUsageCountedForEvictPercentage`	Valkey 또는 Redis OSS Replication 그룹 지표	None

사전 정의된 지표 유형	CloudWatch 지표 이름	CloudWatch 지표 차원	부적격 인스턴스 유형
ElastiCacheDatabaseMemoryUsageCountedForEvictPercentage	DatabaseMemoryUsageCountedForEvictPercentage	Valkey 또는 Redis OSS Replication 그룹 지표	R6gd

데이터 계층 인스턴스 유형은 메모리와 에 데이터를 모두 저장`ElastiCacheDatabaseMemoryUsageCountedForEvictPercentage`하므로 를 사용할 수 없습니다SSD. 데이터 계층 인스턴스의 예상 사용 사례는 메모리 사용량이 100%이고 필요에 SSD 따라 채우는 것입니다.

샤드의 Auto Scaling 기준

서비스에서 사전 정의된 지표가 목표 설정보다 크거나 같음을 감지하면 자동으로 샤드 용량을 증가시킵니다. ElastiCache Valkey 또는 Redis를 사용하면 클러스터 샤드의 OSS 스케일 아웃이 두 숫자 중 큰 숫자, 즉 대상과의 백분율 편차와 현재 샤드의 20%와 동일한 수로 계산됩니다. 스케일 인의 경우 전체 지표 값이 정의된 대상의 75% 미만이 아닌 한 자동 스케일 인되지 ElastiCache 않습니다.

스케일 아웃 예제로, 샤드가 50개 있다고 가정합니다.

- 대상 위반이 30%인 경우 ElastiCache Valkey 또는 Redis는 30% OSS 확장되어 클러스터당 샤드가 65개입니다.
- 대상 위반이 10%인 경우 ElastiCache Valkey 또는 Redis는 기본적으로 최소 20%로 OSS 스케일 아웃되어 클러스터당 샤드가 60개입니다.

스케일 인 예제의 경우 Valkey 또는 Redis ElastiCache 를 사용하여 목표 값을 60%로 선택한 경우 지표가 45%(목표 60%보다 25%) 이하가 OSS 될 때까지 자동 스케일 인되지 않습니다.

Auto Scaling 고려 사항

다음 사항에 유의하세요.

- 대상 추적 조정 정책은 지정한 지표가 목표 값을 초과할 때 한해서 확장을 수행해야 합니다. 지정된 지표가 대상 값보다 낮으면 대상 추적 조정 정책을 사용하여 스케일 아웃할 수 없습니다. Valkey 또는 Redis ElastiCache 를 사용하면 클러스터의 기존 샤드 대상에서 최소 20% 편차만큼 샤드를 OSS 스케일 아웃합니다.
- 대상 추적 조정 정책에서는 지정한 지표에 데이터가 부족할 때 조정을 수행하지 않습니다. 데이터가 부족하다고 해서 사용량이 낮은 것으로 해석하지 않기 때문에 축소를 수행하지 않습니다.
- 목표 값과 실제 지표 데이터 포인트 사이에는 차이가 발생할 수 있습니다. 이는 Valkey 또는 Redis OSS Auto Scaling ElastiCache 을 사용하면 추가 또는 제거할 용량이 결정될 때 항상 반올림 또는 반내림하여 보수적으로 작동하기 때문입니다. 이는 용량을 부족하게 추가하거나 너무 많이 제거하는 일을 방지하기 위해서입니다.
- 애플리케이션 가용성을 보장하기 위해 서비스는 지표에 비례하여 가능한 한 빠르게 스케일 아웃하지만, 스케일 인은 훨씬 보수적으로 수행합니다.
- 각각 다른 지표를 사용하는 경우 Valkey 또는 Redis OSS 클러스터 ElastiCache 가 있는 에 대해 여러 대상 추적 조정 정책을 사용할 수 있습니다. ElastiCache (Redis OSS) Auto Scaling의 의도는 항상 가용성의 우선 순위를 지정하기 위한 것이므로 대상 추적 정책이 스케일 아웃 또는 스케일 인 준비가 되었는지 여부에 따라 동작이 달라집니다. 대상 추적 정책 중 하나라도 확장을 허용할 경우 서비스를 확장하지만, 모든 대상 추적 정책(축소 부분이 활성화됨)이 축소를 허용하는 경우에만 서비스를 축소합니다.
- Valkey 또는 Redis OSS Auto Scaling을 ElastiCache 사용하여 대상 추적 조정 정책에 대해 관리하는 CloudWatch 경보는 편집하거나 삭제하지 마십시오. ElastiCache Auto Scaling은 조정 정책을 삭제하면 경보를 자동으로 삭제합니다.
- ElastiCache Auto Scaling은 클러스터 샤드를 수동으로 수정하는 것을 금지하지 않습니다. 이러한 수동 조정은 조정 정책에 연결된 기존 CloudWatch 경보에는 영향을 미치지 않지만 이러한 CloudWatch 경보를 트리거할 수 있는 지표에는 영향을 미칠 수 있습니다.
- Auto Scaling에서 관리하는 이러한 CloudWatch 경보는 클러스터의 모든 샤드에 대한 AVG 지표를 통해 정의됩니다. 따라서 사용량이 많은 샤드가 있으면 다음 시나리오 중 하나가 발생할 수 있습니다.
  - CloudWatch 경보를 트리거하는 몇 개의 핫 샤드에 대한 로드로 인해 필요하지 않은 경우 크기 조정
  - 경보가 위반되지 않도록 영향을 미치는 모든 샤드AVG에 걸쳐 집계되어 필요한 경우 확장되지 않습니다.
- ElastiCache 클러스터당 노드에 대한 Valkey 또는 Redis OSS 기본 제한이 여전히 적용됩니다. 따라서 Auto Scaling을 선택할 때 최대 노드 수가 기본 제한보다 클 것으로 예상되는 경우 AWS 서비스 한도에서 한도 향상을 요청하고 한도 유형을 인스턴스 유형별 클러스터당 노드로 선택합니다.

- 스케일 아웃 중에 필요한 ENIs (Elastic Network Interfaces)VPC를 에서 충분히 사용할 수 있는지 확인합니다. 자세한 내용은 탄력적 네트워크 인터페이스를 참조하세요.
- 에서 사용할 수 있는 용량이 충분하지 않으면 EC2 ElastiCache Auto Scaling이 확장되지 않고 용량을 사용할 수 있을 때까지 지연됩니다.
- ElastiCache (Redis OSS) 확장 중 Auto Scaling은 직렬화 후 항목 크기가 256MB보다 큰 슬롯이 있는 샤드를 제거하지 않습니다.
- 스케일 인 중에 결과 샤드 구성에서 사용할 수 있는 메모리가 충분하지 않으면 샤드를 제거하지 않습니다.

조정 정책 추가

를 사용하여 조정 정책을 추가할 수 있습니다 AWS Management Console.

Valkey 또는 Redis OSS 클러스터를 ElastiCache 사용하여 Auto Scaling 정책을 에 추가하려면

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 탐색 창에서 Valkey 또는 Redis를 OSS선택합니다.
3. 정책을 추가할 클러스터를 선택합니다(클러스터 이름 왼쪽에 있는 버튼이 아닌 클러스터 이름 선택).
4. Auto Scaling 정책 탭을 선택합니다.
5. 동적 크기 조정 추가(add dynamic scaling)를 선택합니다.
6. 정책 이름에 정책 이름을 입력합니다.
7. 조정 가능 차원에서 샤드를 선택합니다.
8. 대상 지표로 다음 중 하나를 선택합니다.
   - 기본 CPU 사용률은 평균 CPU 사용률을 기반으로 정책을 생성합니다.
   - 메모리 - 평균 데이터베이스 메모리를 기반으로 정책을 생성합니다.
   - 용량 - 평균 데이터베이스 용량 사용률을 기반으로 정책을 생성합니다. 용량 지표에는 데이터 계층 인스턴스의 메모리 및 SSD 사용률과 다른 모든 인스턴스 유형의 메모리 사용률이 포함됩니다.
9. 목표값으로 35 이상 70 이하의 값을 선택합니다. Auto Scaling은 ElastiCache 샤드 전체에서 선택한 대상 지표에 대해 이 값을 유지합니다.
   - 기본 CPU 사용률: 기본 노드의 `EngineCPUUtilization` 지표에 대한 목표 값을 유지합니다.

- 메모리: `DatabaseMemoryUsageCountedForEvictPercentage` 지표의 목표 값을 유지합니다.
- 용량: `DatabaseCapacityUsageCountedForEvictPercentage` 지표의 목표 값을 유지합니다.

클러스터 샤드가 추가되거나 제거되어 지정한 값에 가깝게 지표가 유지됩니다.

10. (선택 사항) 콘솔에서는 스케일 인 또는 스케일 아웃 휴지 기간이 지원되지 않습니다. AWS CLI 를 사용하여 냉각 값을 수정합니다.
11. 최소 용량 의 경우 ElastiCache Auto Scaling 정책이 유지해야 하는 최소 샤드 수를 입력합니다.
12. 최대 용량 에 ElastiCache Auto Scaling 정책이 유지 관리하는 데 필요한 최대 샤드 수를 입력합니다. 이 값은 250보다 작거나 같아야 합니다.
13. 생성(Create)을 선택합니다.

확장 가능 목표 등록

Valkey 또는 Redis OSS 클러스터에서 Auto Scaling ElastiCache 을 사용하려면 먼저 클러스터를 ElastiCache Auto Scaling으로 등록해야 합니다. 이렇게 하면 해당 클러스터에 적용할 크기 조정 차원과 제한을 정의할 수 있습니다. ElastiCache 자동 크기 조정은 클러스터 샤드 수를 나타내는 `elasticache:replication-group:NodeGroups` 확장 가능한 차원을 따라 클러스터를 동적으로 조정합니다.

사용 AWS CLI

를 Valkey 또는 Redis OSS 클러스터 ElastiCache 에 등록하려면 다음 파라미터와 함께 register-scalable-target 명령을 사용합니다.

- `--service-namespace` 이 값을 로 설정하세요.`elasticache`
- `--resource-id` - 클러스터의 리소스 식별자입니다. 이 파라미터의 경우 리소스 유형은 `ReplicationGroup`이고 고유 식별자는 클러스터의 이름입니다. 예를 들어 입니다`replication-group/myscalablecluster`.
- `--scalable-dimension` - 이 값을 로 설정하세요.`elasticache:replication-group:NodeGroups`
- `--max-capacity` - ElastiCache 자동 크기 조정으로 관리할 최대 샤드 수입니다. `--min-capacity`, `--max-capacity` 및 클러스터의 샤드 수 간의 관계에 대한 자세한 내용은 최소 및 최대 용량 섹션을 참조하세요.

- `--min-capacity` - ElastiCache 자동 크기 조정으로 관리할 최소 샤드 수입니다. `--min-capacity`, `--max-capacity` 및 클러스터의 샤드 수 간의 관계에 대한 자세한 내용은 최소 및 최대 용량 섹션을 참조하세요.

Example

다음 예제에서는 라는 이름의 Valkey 또는 Redis OSS 클러스터 ElastiCache 에 를 등록합니다`myscalablecluster`. 등록은 1개에서 10개까지 샤드를 포함하도록 클러스터 크기를 동적으로 조정해야 함을 나타냅니다.

Linux, macOS, Unix의 경우:

```
aws application-autoscaling register-scalable-target \
    --service-namespace elasticache \
    --resource-id replication-group/myscalablecluster \
    --scalable-dimension elasticache:replication-group:NodeGroups \
    --min-capacity 1 \
    --max-capacity 10 \
```

Windows의 경우:

```
aws application-autoscaling register-scalable-target ^
    --service-namespace elasticache ^
    --resource-id replication-group/myscalablecluster ^
    --scalable-dimension elasticache:replication-group:NodeGroups ^
    --min-capacity 1 ^
    --max-capacity 10 ^
```

사용 API

ElastiCache 클러스터를 등록하려면 다음 파라미터와 함께 register-scalable-target 명령을 사용합니다.

- ServiceNamespace - 이 값을 elasticache로 설정합니다.
- ResourceID - ElastiCache 클러스터의 리소스 식별자입니다. 이 파라미터의 경우 리소스 유형은 ReplicationGroup 이고 고유 식별자는 클러스터의 이름입니다. 예를 들어 입니다`replication-group/myscalablecluster`.
- ScalableDimension – 이 값을 로 설정합니다`elasticache:replication-group:NodeGroups`.

- MinCapacity - ElastiCache 자동 크기 조정으로 관리할 최소 샤드 수입니다. --min-capacity, --max-capacity 및 클러스터의 복제본 수 간의 관계에 대한 자세한 내용은 최소 및 최대 용량 섹션을 참조하세요.
- MaxCapacity - ElastiCache 자동 크기 조정으로 관리할 최대 샤드 수입니다. --min-capacity, --max-capacity 및 클러스터의 복제본 수 간의 관계에 대한 자세한 내용은 최소 및 최대 용량 섹션을 참조하세요.

Example

다음 예제에서는 Application Auto Scaling ElastiCache 로 이름이 지정된 Valkey 또는 Redis OSS 클러스터myscalablecluster에 를 등록합니다API. 이 등록은 1개에서 5개까지 복제본을 포함하도록 클러스터 크기를 동적으로 조정해야 함을 나타냅니다.

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.RegisterScalableTarget
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS
{
    "ServiceNamespace": "elasticache",
    "ResourceId": "replication-group/myscalablecluster",
    "ScalableDimension": "elasticache:replication-group:NodeGroups",
    "MinCapacity": 1,
    "MaxCapacity": 5
}
```

조정 정책 정의

대상 추적 조정 정책 구성은 지표와 대상 값이 정의된 JSON 블록으로 표시됩니다. 크기 조정 정책 구성을 텍스트 파일의 JSON 블록으로 저장할 수 있습니다. AWS CLI 또는 Application Auto Scaling을 호출할 때 해당 텍스트 파일을 사용합니다API. 정책 구성 구문에 대한 자세한 내용은 Application Auto Scaling API 참조TargetTrackingScalingPolicyConfiguration의 섹션을 참조하세요.

대상 추적 조정 정책 구성을 정의하기 위해 다음과 같은 옵션을 사용할 수 있습니다.

주제

- 미리 정의된 지표 사용
- 사용자 지정 지표 사용
- 휴지 기간 사용

미리 정의된 지표 사용

미리 정의된 지표를 사용하여 ElastiCache (Redis OSS) Auto Scaling 에서 대상 추적과 함께 작동하는 Valkey 또는 Redis OSS 클러스터 ElastiCache 가 있는 에 대한 대상 추적 조정 정책을 빠르게 정의할 수 있습니다. Auto Scaling

현재 는 NodeGroup Auto Scaling에서 다음과 같은 사전 정의된 지표를 ElastiCache 지원합니다.

- ElastiCachePrimaryEngineCPUUtilization - 클러스터의 모든 기본 노드에서 `EngineCPUUtilization` 의 CloudWatch 지표 평균 값입니다.
- ElastiCacheDatabaseMemoryUsageCountedForEvictPercentage - 클러스터의 모든 기본 노드 CloudWatch 에서 `DatabaseMemoryUsageCountedForEvictPercentage` 의 지표 평균 값입니다.
- ElastiCacheDatabaseCapacityUsageCountedForEvictPercentage - 클러스터의 모든 기본 노드 CloudWatch 에서 `ElastiCacheDatabaseCapacityUsageCountedForEvictPercentage` 의 지표 평균 값입니다.

`EngineCPUUtilization`, `DatabaseMemoryUsageCountedForEvictPercentage` 및 `DatabaseCapac` 표에 대한 자세한 정보는 CloudWatch 지표 사용 모니터링 섹션을 참조하세요. 조정 정책에서 미리 정의된 지표를 사용하려면 조정 정책을 위한 대상 추적 구성을 생성합니다. 이 구성에는 사전 정의된 지표에 `PredefinedMetricSpecification` 대한 와 해당 지표의 대상 값에 TargetValue 대한 가 포함되어야 합니다.

Example

다음 예제에서는 Valkey 또는 Redis OSS 클러스터가 ElastiCache 있는 의 대상 추적 조정을 위한 일반적인 정책 구성을 설명합니다. 이 구성에서는 사전 `ElastiCachePrimaryEngineCPUUtilization` 정의된 지표를 사용하여 클러스터의 모든 프라이머리 노드에서 평균 CPU 사용률 40%를 기준으로 클러스터를 조정합니다.

```
{
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
```

```
        "PredefinedMetricType": "ElastiCachePrimaryEngineCPUUtilization"
    }
}
```

사용자 지정 지표 사용

사용자 지정 지표를 사용하여 사용자 지정 요구 사항에 맞는 대상 추적 조정 정책을 정의할 수 있습니다. 조정에 비례하여 변경되는 모든 지표를 기반으로 사용자 지정 ElastiCache 지표를 정의할 수 있습니다. 모든 ElastiCache 지표가 대상 추적에 사용되는 것은 아닙니다. 지표는 유효한 사용량 수치로서 인스턴스의 사용량을 설명해야 합니다. 클러스터에 있는 샤드 수에 따라 지표 값이 증가하거나 줄어들어야 합니다. 지표 데이터를 사용하여 샤드 수를 비례적으로 확장 또는 축소하려면 이 비례적인 증가나 감소가 필요합니다.

Example

다음 예제에서는 조정 정책의 대상 추적 구성을 설명합니다. 이 구성에서 사용자 지정 지표는 라는 클러스터의 모든 샤드에서 평균 CPU 사용률 50%를 기준으로 ElastiCache (Redis OSS) 클러스터를 조정합니다`my-db-cluster`.

```
{
    "TargetValue": 50,
    "CustomizedMetricSpecification":
    {
        "MetricName": "EngineCPUUtilization",
        "Namespace": "AWS/ElastiCache",
        "Dimensions": [
            {
                "Name": "RelicationGroup","Value": "my-db-cluster"
            },
            {
                "Name": "Role","Value": "PRIMARY"
            }
        ],
        "Statistic": "Average",
        "Unit": "Percent"
    }
}
```

휴지 기간 사용

`ScaleOutCooldown`에 초 단위로 값을 지정하여 클러스터를 확장하기 위한 휴지 기간을 추가할 수 있습니다. 마찬가지로 `ScaleInCooldown`에 초 단위로 값을 추가하여 클러스터를 축

소하기 위한 휴지 기간을 추가할 수 있습니다. 자세한 내용은 Application Auto Scaling API 참조TargetTrackingScalingPolicyConfiguration의 섹션을 참조하세요.

다음 예제에서는 조정 정책의 대상 추적 구성을 설명합니다. 이 구성에서는 ElastiCachePrimaryEngineCPUUtilization 사전 정의된 지표를 사용하여 해당 클러스터의 모든 기본 노드에서 평균 CPU 사용률 40%를 기준으로 ElastiCache (Redis OSS) 클러스터를 조정합니다. 구성에서는 스케일 인 휴지 기간 10분과 스케일 아웃 휴지 기간 5분을 제공합니다.

```
{
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "ElastiCachePrimaryEngineCPUUtilization"
    },
    "ScaleInCooldown": 600,
    "ScaleOutCooldown": 300
}
```

스케일 인 활동 비활성화

스케일 인 활동을 비활성화하여 대상 추적 스케일링 정책 구성이 클러스터에서 스케일링되지 않도록 할 수 있습니다. 스케일 인 활동을 비활성화하면 조정 정책에서 필요에 따라 샤드를 생성할 수 있지만 삭제할 수는 없습니다.

DisableScaleIn에 부울 값을 지정하여 클러스터에 대한 스케일 인 활동을 활성화하거나 비활성화할 수 있습니다. 자세한 내용은 Application Auto Scaling API 참조TargetTrackingScalingPolicyConfiguration의 섹션을 참조하세요.

다음 예제에서는 조정 정책의 대상 추적 구성을 설명합니다. 이 구성에서 ElastiCachePrimaryEngineCPUUtilization 사전 정의된 지표는 해당 OSS 클러스터의 모든 기본 노드에서 평균 CPU 사용률 40%를 기준으로 Valkey 또는 Redis 클러스터 ElastiCache 를 사용하여 를 조정합니다. 구성에서는 조정 정책의 스케일 인 활동을 비활성화합니다.

```
{
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "ElastiCachePrimaryEngineCPUUtilization"
    },
    "DisableScaleIn": true
}
```

조정 정책 적용

Valkey 또는 Redis OSS Auto Scaling ElastiCache 을 사용하여 에 클러스터를 등록하고 스케일링 정책을 정의한 후 등록된 클러스터에 스케일링 정책을 적용합니다. ElastiCache (Redis OSS) 클러스터에 조정 정책을 적용하려면 AWS CLI 또는 Application Auto Scaling을 사용할 수 있습니다API.

를 사용하여 조정 정책 적용 AWS CLI

Valkey 또는 Redis OSS 클러스터를 ElastiCache 사용하여 에 조정 정책을 적용하려면 다음 파라미터와 함께 put-scaling-policy 명령을 사용합니다.

- --policy-name – 조정 정책의 이름입니다.
- --policy-type – 이 값을 `TargetTrackingScaling`으로 설정합니다.
- --resource-id – 리소스 식별자입니다. 이 파라미터의 경우 리소스 유형은 `ReplicationGroup`이고 고유 식별자는 클러스터의 이름입니다. 예를 들어 입니다`replication-group/myscalablecluster`.
- --service-namespace – 이 값을 `elasticache`로 설정합니다.
- --scalable-dimension – 이 값을 `elasticache:replication-group:NodeGroups`로 설정합니다.
- -target-tracking-scaling-policy--configuration – 클러스터에 사용할 대상 추적 조정 정책 구성입니다.

다음 예제에서는 ElastiCache 자동 조정으로 이름이 지정된 Valkey 또는 Redis OSS 클러스터`myscalablecluster`를 ElastiCache 사용하여 `myscalablepolicy` 에 이름이 지정된 대상 추적 조정 정책을 적용합니다. 이를 위해 `config.json`이라는 파일에 저장된 정책 구성을 사용합니다.

Linux, macOS 또는 Unix의 경우는 다음과 같습니다.

```
aws application-autoscaling put-scaling-policy \
    --policy-name myscalablepolicy \
    --policy-type TargetTrackingScaling \
    --resource-id replication-group/myscalablecluster \
    --service-namespace elasticache \
    --scalable-dimension elasticache:replication-group:NodeGroups \
    --target-tracking-scaling-policy-configuration file://config.json
```

Windows의 경우:

```
aws application-autoscaling put-scaling-policy ^
    --policy-name myscalablepolicy ^
    --policy-type TargetTrackingScaling ^
    --resource-id replication-group/myscalablecluster ^
    --service-namespace elasticache ^
    --scalable-dimension elasticache:replication-group:NodeGroups ^
    --target-tracking-scaling-policy-configuration file://config.json
```

를 사용하여 조정 정책 적용 API

Valkey 또는 Redis OSS 클러스터를 ElastiCache 사용하여 에 조정 정책을 적용하려면 다음 파라미터와 함께 PutScalingPolicy AWS CLI 명령을 사용합니다.

- --policy-name – 조정 정책의 이름입니다.
- --resource-id – 리소스 식별자입니다. 이 파라미터의 경우 리소스 유형은 `ReplicationGroup`이고 고유 식별자는 클러스터의 이름입니다. 예를 들어 입니다`replication-group/myscalablecluster`.
- --service-namespace – 이 값을 `elasticache`로 설정합니다.
- --scalable-dimension – 이 값을 `elasticache:replication-group:NodeGroups`로 설정합니다.
- --target-tracking-scaling-policy-configuration – 클러스터에 사용할 대상 추적 조정 정책 구성입니다.

다음 예제에서는 ElastiCache 자동 조정으로 이름이 지정된 Valkey 또는 Redis OSS 클러스터`myscalablecluster`를 ElastiCache 사용하여 `myscalablepolicy` 에 이름이 지정된 대상 추적 조정 정책을 적용합니다. `ElastiCachePrimaryEngineCPUUtilization` 사전 정의 지표를 기반으로 하는 정책 구성을 사용합니다.

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.PutScalingPolicy
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS
{
    "PolicyName": "myscalablepolicy",
    "ServiceNamespace": "elasticache",
```

```
    "ResourceId": "replication-group/myscalablecluster",
    "ScalableDimension": "elasticache:replication-group:NodeGroups",
    "PolicyType": "TargetTrackingScaling",
    "TargetTrackingScalingPolicyConfiguration": {
        "TargetValue": 40.0,
        "PredefinedMetricSpecification":
        {
            "PredefinedMetricType": "ElastiCachePrimaryEngineCPUUtilization"
        }
    }
}
```

조정 정책 편집

AWS Management Console, AWS CLI또는 Application Auto Scaling을 사용하여 조정 정책을 편집할 수 있습니다API.

를 사용하여 조정 정책 편집 AWS Management Console

Valkey 또는 Redis OSS 클러스터가 ElastiCache 있는 에 대한 Auto Scaling 정책을 편집하려면

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 탐색 창에서 적절한 엔진을 선택합니다.
3. 정책을 추가할 클러스터를 선택합니다(클러스터 이름 왼쪽에 있는 버튼이 아닌 클러스터 이름 선택).
4. Auto Scaling 정책 탭을 선택합니다.
5. 조정 정책(Scaling policies)에서 변경할 자동 크기 조정 정책 왼쪽에 있는 버튼을 선택한 다음 수정(Modify)을 선택합니다.
6. 정책을 필요에 따라 변경합니다.
7. 수정을 선택합니다.

AWS CLI 및 를 사용하여 조정 정책 편집 API

AWS CLI 또는 Application Auto ScalingAPI을 사용하여 조정 정책을 적용하는 것과 동일한 방식으로 조정 정책을 편집할 수 있습니다.

- 를 사용할 때 `--policy-name` 파라미터에 편집하려는 정책의 이름을 AWS CLI지정합니다. 변경할 파라미터의 새로운 값을 지정합니다.

- Application Auto Scaling을 사용할 때 `PolicyName` 파라미터에 편집하려는 정책의 이름을 API지정합니다. 변경할 파라미터의 새로운 값을 지정합니다.

자세한 내용은 조정 정책 적용 단원을 참조하십시오.

조정 정책 삭제

AWS Management Console, AWS CLI또는 Application Auto Scaling을 사용하여 조정 정책을 삭제할 수 있습니다API.

를 사용하여 조정 정책 삭제 AWS Management Console

ElastiCache (RedisOSS) 클러스터에 대한 Auto Scaling 정책을 삭제하려면

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 탐색 창에서 Valkey 또는 Redis를 OSS선택합니다.
3. 자동 크기 조정 정책을 편집할 클러스터를 선택합니다(클러스터 이름 왼쪽에 있는 버튼이 아닌 클러스터 이름 선택).
4. Auto Scaling 정책 탭을 선택합니다.
5. 조정 정책(Scaling policies)에서 자동 크기 조정 정책을 선택한 후 삭제(Delete)를 선택합니다.

를 사용하여 조정 정책 삭제 AWS CLI

Valkey 또는 Redis OSS 클러스터를 ElastiCache 사용하여 에 대한 조정 정책을 삭제하려면 다음 파라미터와 함께 delete-scaling-policy AWS CLI 명령을 사용합니다.

- --policy-name – 조정 정책의 이름입니다.
- --resource-id – 리소스 식별자입니다. 이 파라미터의 경우 리소스 유형은 `ReplicationGroup`이고 고유 식별자는 클러스터의 이름입니다. 예를 들어 입니다`replication-group/myscalablecluster`.
- --service-namespace – 이 값을 `elasticache`로 설정합니다.
- --scalable-dimension – 이 값을 `elasticache:replication-group:NodeGroups`로 설정합니다.

다음 예제에서는 라는 클러스터`myscalablepolicy`에서 라는 대상 추적 조정 정책을 삭제합니다`myscalablecluster`.

Linux, macOS, Unix의 경우:

```
aws application-autoscaling delete-scaling-policy \
    --policy-name myscalablepolicy \
    --resource-id replication-group/myscalablecluster \
    --service-namespace elasticache \
    --scalable-dimension elasticache:replication-group:NodeGroups
```

Windows의 경우:

```
aws application-autoscaling delete-scaling-policy ^
    --policy-name myscalablepolicy ^
    --resource-id replication-group/myscalablecluster ^
    --service-namespace elasticache ^
    --scalable-dimension elasticache:replication-group:NodeGroups
```

를 사용하여 조정 정책 삭제 API

Valkey 또는 Redis OSS 클러스터를 ElastiCache 사용하여 에 대한 조정 정책을 삭제하려면 다음 파라미터와 함께 DeleteScalingPolicy AWS CLI 명령을 사용합니다.

- --policy-name – 조정 정책의 이름입니다.
- --resource-id – 리소스 식별자입니다. 이 파라미터의 경우 리소스 유형은 `ReplicationGroup`이고 고유 식별자는 클러스터의 이름입니다. 예를 들어 입니다`replication-group/myscalablecluster`.
- --service-namespace – 이 값을 `elasticache`로 설정합니다.
- --scalable-dimension – 이 값을 `elasticache:replication-group:NodeGroups`로 설정합니다.

다음 예제에서는 라는 클러스터`myscalablepolicy`에서 라는 대상 추적 조정 정책을 삭제합니다`myscalablecluster`.

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.DeleteScalingPolicy
X-Amz-Date: 20160506T182145Z
```

```
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS
{
    "PolicyName": "myscalablepolicy",
    "ServiceNamespace": "elasticache",
    "ResourceId": "replication-group/myscalablecluster",
    "ScalableDimension": "elasticache:replication-group:NodeGroups"
}
```

AWS CloudFormation Auto Scaling 정책에 사용

이 코드 조각은 AWS:::: 리소스를 사용하여 대상 추적 정책을 생성하고 ::ElastiCache:ReplicationGroupAWSApplicationAutoScaling:ScalableTarget 리소스에 적용하는 방법을 보여줍니다. Fn::Join 및 Ref 내장 함수를 사용하여 동일한 템플릿에 지정된 `AWS::ElastiCache::ReplicationGroup` 리소스의 논리적 이름으로 `ResourceId` 속성을 구성합니다.

```
ScalingTarget:
   Type: 'AWS::ApplicationAutoScaling::ScalableTarget'
   Properties:
     MaxCapacity: 3
     MinCapacity: 1
     ResourceId: !Sub replication-group/${logicalName}
     ScalableDimension: 'elasticache:replication-group:NodeGroups'
     ServiceNamespace: elasticache
     RoleARN: !Sub "arn:aws:iam::${AWS::AccountId}:role/aws-
service-role/elasticache.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG"

  ScalingPolicy:
    Type: "AWS::ApplicationAutoScaling::ScalingPolicy"
    Properties:
      ScalingTargetId: !Ref ScalingTarget
      ServiceNamespace: elasticache
      PolicyName: testpolicy
      PolicyType: TargetTrackingScaling
      ScalableDimension: 'elasticache:replication-group:NodeGroups'
      TargetTrackingScalingPolicyConfiguration:
        PredefinedMetricSpecification:
          PredefinedMetricType: ElastiCachePrimaryEngineCPUUtilization
        TargetValue: 40
```

예약된 조정

일정을 기반으로 조정을 수행하면 수요에 따른 로드 변경에 맞게 애플리케이션을 조정할 수 있습니다. 예약된 조정을 사용하려면 Valkey 또는 Redis ElastiCache 를 사용하여 특정 시간에 조정 활동을 OSS 수행하도록 지시하는 예약된 작업을 생성합니다. 예약된 작업을 생성할 때 조정 활동이 발생하는 시점, 최소 용량 및 최대 용량인 기존 ElastiCache (Redis OSS) 클러스터를 지정합니다. 규모를 한 번만 조정하거나 반복되는 일정으로 조정하도록 예약된 작업을 생성할 수 있습니다.

이미 존재하는 ElastiCache (Redis OSS) 클러스터에 대해서만 예약된 작업을 생성할 수 있습니다. 클러스터를 생성하는 동시에 예약된 작업을 생성할 수는 없습니다.

예약된 작업 생성, 관리 및 삭제와 관련된 용어에 대한 자세한 내용은 예약된 작업 생성, 관리 및 삭제에 일반적으로 사용되는 명령을 참조하세요.

반복되는 일정으로 생성하려면

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 탐색 창에서 Valkey 또는 Redis를 OSS선택합니다.
3. 정책을 추가할 클러스터를 선택합니다.
4. 작업 드롭다운 목록에서 Auto Scaling 정책 관리를 선택합니다.
5. Auto Scaling 정책 탭을 선택합니다.
6. Auto Scaling 정책 섹션에서 조정 정책 추가 대화 상자가 나타납니다. 예약된 조정을 선택합니다.
7. 정책 이름에 정책 이름을 입력합니다.
8. 조정 가능 차원에서 샤드를 선택합니다.
9. 대상 샤드에서 값을 선택합니다.
10. 반복에서 반복을 선택합니다.
11. 빈도에서 해당하는 값을 선택합니다.
12. 시작일 및 시작 시간에서 정책이 시행될 시간을 선택합니다.
13. 정책 추가를 선택합니다.

1회성 예약된 작업을 생성하려면

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 탐색 창에서 Valkey 또는 Redis를 OSS선택합니다.

3. 정책을 추가할 클러스터를 선택합니다.
4. 작업 드롭다운 목록에서 Auto Scaling 정책 관리를 선택합니다.
5. Auto Scaling 정책 탭을 선택합니다.
6. Auto Scaling 정책 섹션에서 조정 정책 추가 대화 상자가 나타납니다. 예약된 조정을 선택합니다.
7. 정책 이름에 정책 이름을 입력합니다.
8. 조정 가능 차원에서 샤드를 선택합니다.
9. 대상 샤드에서 값을 선택합니다.
10. 반복에서 한 번을 선택합니다.
11. 시작일 및 시작 시간에서 정책이 시행될 시간을 선택합니다.
12. 종료일에서 정책이 시행되는 기한을 선택합니다.
13. 정책 추가를 선택합니다.

예약된 작업 삭제

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 탐색 창에서 Valkey 또는 Redis를 OSS선택합니다.
3. 정책을 추가할 클러스터를 선택합니다.
4. 작업 드롭다운 목록에서 Auto Scaling 정책 관리를 선택합니다.
5. Auto Scaling 정책 탭을 선택합니다.
6. Auto Scaling 정책 섹션에서 Auto Scaling 정책을 선택한 다음 작업 메뉴에서 삭제를 선택합니다.

AWS CLI 를 사용하여 예약된 조정을 관리하려면

다음 애플리케이션 자동 크기 조정을 사용합니다APIs.

- put-scheduled-action
- describe-scheduled-actions
- delete-scheduled-action

AWS CloudFormation 을 사용하여 예약된 작업 생성

이 코드 조각은 AWS:::: 리소스를 사용하여 대상 추적 정책을 생성하고 ::ElastiCache:ReplicationGroupAWSApplicationAutoScaling:ScalableTarget 리소스에 적

용하는 방법을 보여줍니다. Fn::Join 및 Ref 내장 함수를 사용하여 동일한 템플릿에 지정된 AWS::ElastiCache::ReplicationGroup 리소스의 논리적 이름으로 ResourceId 속성을 구성합니다.

```
ScalingTarget:
   Type: 'AWS::ApplicationAutoScaling::ScalableTarget'
   Properties:
     MaxCapacity: 3
     MinCapacity: 1
     ResourceId: !Sub replication-group/${logicalName}
     ScalableDimension: 'elasticache:replication-group:NodeGroups'
     ServiceNamespace: elasticache
     RoleARN: !Sub "arn:aws:iam::${AWS::AccountId}:role/aws-
service-role/elasticache.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG"
     ScheduledActions:
       - EndTime: '2020-12-31T12:00:00.000Z'
         ScalableTargetAction:
           MaxCapacity: '5'
           MinCapacity: '2'
         ScheduledActionName: First
         Schedule: 'cron(0 18 * * ? *)'
```

복제본에 Auto Scaling 사용

ElastiCache 복제 그룹은 단일 논리적 노드로 작동하도록 하나 이상의 캐시를 설정할 수 있습니다.

다음은 대상 추적 및 예약된 정책에 대한 세부 정보와 AWS Management Console AWS CLI 및 를 사용하여 정책을 적용하는 방법을 제공합니다APIs.

대상 추적 조정 정책

대상 추적 조정 정책을 사용하는 경우 지표를 선택하고 목표 값을 설정합니다. ElastiCache 와 함께 Valkey 또는 Redis는 조정 정책을 트리거하는 CloudWatch 경보를 OSS AutoScaling 생성 및 관리하고 지표와 대상 값을 기반으로 조정 조정을 계산합니다. 조정 정책은 필요에 따라 모든 샤드에서 균등하게 복제본을 추가하거나 제거하여 지표를 지정한 대상 값으로 또는 대상 값에 가깝게 유지합니다. 대상 추적 조정 정책은 지표를 목표 값에 가깝게 유지하는 것 외에도 로드 패턴의 변화로 인한 지표 변동에 따라 반응하여 플릿의 용량이 갑작스럽게 바뀌는 것을 최소화합니다.

복제본에 대한 Auto Scaling 기준

Auto Scaling 정책은 클러스터에 대해 다음과 같은 사전 정의된 지표를 정의합니다.

`ElastiCacheReplicaEngineCPUUtilization`: 가 자동 크기 조정 작업을 트리거하는 데 ElastiCache 사용하는 모든 복제본에 걸쳐 집계된 AVG 엔진CPU 사용률 임계값입니다. 사용률 목표를 35%에서 70% 사이로 설정할 수 있습니다.

서비스에서 `ElastiCacheReplicaEngineCPUUtilization` 지표가 대상 설정과 같거나 크다는 것을 감지하면 샤드 전체에서 복제본이 자동으로 증가합니다. Valkey 또는 Redis ElastiCache 를 사용하면 클러스터 복제본이 대상과의 백분율 편차와 복제본 1개 중 큰 숫자와 동일한 수만큼 OSS 스케일 아웃됩니다. 스케일 인의 경우 전체 지표 값이 정의된 대상의 75% 미만인 경우를 제외하고 Valkey 또는 Redis를 ElastiCache 사용하면 자동 스케일 인되지 OSS 않습니다.

스케일 아웃 예제로, 샤드가 5개 있고 각각 복제본 1개가 있다고 가정합니다.

대상 위반이 30%인 경우 ElastiCache Valkey 또는 Redis는 모든 샤드에서 복제본 1개(최대(0.3, 기본값 1))만큼 OSS 스케일 아웃되며, 이로 인해 각 복제본 2개가 있는 샤드 5개가 생성됩니다.

스케일 인 예제에서 목표 값을 60%로 선택한 경우 Valkey 또는 Redis ElastiCache 를 사용하면 지표가 45%(목표 60%보다 25%) 이하가 OSS 될 때까지 자동 스케일 인되지 않습니다.

Auto Scaling 고려 사항

다음 사항에 유의하세요.

- 대상 추적 조정 정책은 지정한 지표가 목표 값을 초과할 때 한해서 확장을 수행해야 합니다. 지정된 지표가 대상 값보다 낮으면 대상 추적 조정 정책을 사용하여 스케일 아웃할 수 없습니다. Valkey 또는 Redis ElastiCache 는 클러스터의 모든 샤드에 걸쳐 기존 복제본의 최대 (대상에서 반올림된 편차 %, 기본값 1)까지 복제본을 OSS 스케일 아웃합니다.
- 대상 추적 조정 정책에서는 지정한 지표에 데이터가 부족할 때 조정을 수행하지 않습니다. 데이터가 부족하다고 해서 사용량이 낮은 것으로 해석하지 않기 때문에 축소를 수행하지 않습니다.
- 목표 값과 실제 지표 데이터 포인트 사이에는 차이가 발생할 수 있습니다. 이는 Valkey 또는 Redis OSS Auto Scaling ElastiCache 을 사용하면 추가 또는 제거할 용량이 결정될 때 항상 반올림 또는 반내림하여 보수적으로 작동하기 때문입니다. 이는 용량을 부족하게 추가하거나 너무 많이 제거하는 일을 방지하기 위해서입니다.
- 애플리케이션 가용성을 보장하기 위해 서비스는 지표에 비례하여 가능한 한 빠르게 스케일 아웃하지만, 스케일 인은 클러스터의 전체 샤드에서 복제본 1개의 최대 스케일 인을 사용하여 비교적 점진적으로 진행됩니다.

- 각각 다른 지표를 사용하는 경우, Valkey 또는 Redis OSS 클러스터 ElastiCache 가 있는 에 대해 여러 대상 추적 조정 정책을 사용할 수 있습니다. Auto Scaling의 목적은 항상 가용성의 우선 순위를 지정하기 위한 것이므로 대상 추적 정책이 스케일 아웃 또는 스케일 인 준비가 되었는지 여부에 따라 동작이 달라집니다. 대상 추적 정책 중 하나라도 확장을 허용할 경우 서비스를 확장하지만, 모든 대상 추적 정책(축소 부분이 활성화됨)이 축소를 허용하는 경우에만 서비스를 축소합니다.
- Valkey 또는 Redis OSS Auto Scaling을 ElastiCache 사용하여 대상 추적 조정 정책에 대해 관리하는 CloudWatch 경보는 편집하거나 삭제하지 마십시오. Auto Scaling은 조정 정책을 삭제하거나 클러스터를 삭제할 때 경보를 자동으로 삭제합니다.
- ElastiCache Valkey 또는 Redis OSS Auto Scaling을 사용하면 샤드 간에 복제본을 수동으로 수정할 수 없습니다. 이러한 수동 조정은 조정 정책에 연결된 기존 CloudWatch 경보에는 영향을 미치지 않지만 이러한 CloudWatch 경보를 트리거할 수 있는 지표에는 영향을 미칠 수 있습니다.
- Auto Scaling에서 관리하는 이러한 CloudWatch 경보는 클러스터의 모든 샤드에 대한 AVG 지표를 통해 정의됩니다. 따라서 사용량이 많은 샤드가 있으면 다음 시나리오 중 하나가 발생할 수 있습니다.
  - CloudWatch 경보를 트리거하는 몇 개의 핫 샤드에 대한 로드로 인해 필요하지 않은 경우 크기 조정
  - 경보가 위반되지 않도록 영향을 미치는 모든 샤드AVG에 걸쳐 집계되어 필요한 경우 확장되지 않습니다.
- ElastiCache 클러스터당 노드에 대한 Valkey 또는 Redis OSS 기본 제한이 여전히 적용됩니다. 따라서 Auto Scaling을 선택할 때 최대 노드 수가 기본 제한보다 클 것으로 예상되는 경우 AWS 서비스 한도에서 한도 향상을 요청하고 한도 유형을 인스턴스 유형별 클러스터당 노드로 선택합니다.
- 스케일 아웃 중에 필요한 ENIs (Elastic Network Interfaces)VPC를 에서 충분히 사용할 수 있는지 확인합니다. 자세한 내용은 탄력적 네트워크 인터페이스를 참조하세요.
- 에서 사용할 수 있는 용량이 충분하지 않은 경우 Valkey 또는 Redis OSS Auto Scaling을 EC2 ElastiCache 사용하면 용량을 사용할 수 있을 때까지 또는 용량이 충분한 인스턴스 유형으로 클러스터를 수동으로 수정할 때까지 확장되지 않습니다.
- ElastiCache Valkey 또는 Redis OSS Auto Scaling을 사용하면 25% `ReservedMemoryPercent` 미만의 클러스터로 복제본 크기 조정을 지원하지 않습니다. 자세한 내용은 Valkey 및 Redis용 예약 메모리 관리 OSS 단원을 참조하십시오.

조정 정책 추가

를 사용하여 조정 정책을 추가할 수 있습니다 AWS Management Console.

를 사용하여 조정 정책 추가 AWS Management Console

Valkey 또는 Redis를 ElastiCache 사용하여 에 Auto Scaling 정책을 추가하려면 OSS

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 탐색 창에서 Valkey 또는 Redis를 OSS선택합니다.
3. 정책을 추가할 클러스터를 선택합니다(클러스터 이름 왼쪽에 있는 버튼이 아닌 클러스터 이름 선택).
4. Auto Scaling 정책 탭을 선택합니다.
5. 동적 크기 조정 추가(add dynamic scaling)를 선택합니다.
6. 조정 정책(Scaling policies)에서 동적 크기 조정 추가(Add dynamic scaling)를 선택합니다.
7. 정책 이름에 정책 이름을 입력합니다.
8. 조정 가능 차원의 대화 상자에서 복제본을 선택합니다.
9. 대상 값에 ElastiCache 복제본에서 유지하려는 평균 CPU 사용률을 입력합니다. 이 값은 >=35 및 <=70이어야 합니다. 클러스터 복제본이 추가되거나 제거되어 지정한 값에 가깝게 지표가 유지됩니다.
10. (선택 사항) 콘솔에서는 스케일 인 또는 스케일 아웃 휴지 기간이 지원되지 않습니다. AWS CLI 를 사용하여 냉각 값을 수정합니다.
11. 최소 용량 에 Valkey 또는 Redis OSS Auto Scaling 정책 ElastiCache 이 있는 가 유지 관리하는 데 필요한 최소 복제본 수를 입력합니다.
12. 최대 용량 에 Valkey 또는 Redis OSS Auto Scaling 정책을 ElastiCache 사용하여 가 유지 관리하는 데 필요한 최대 복제본 수를 입력합니다. 이 값은 >=5이어야 합니다.
13. 생성(Create)을 선택합니다.

확장 가능 목표 등록

미리 정의된 지표나 사용자 지정 지표를 기반으로 조정 정책을 적용할 수 있습니다. 이렇게 하려면 AWS CLI 또는 Application Auto Scaling을 사용할 수 있습니다API. 첫 번째 단계는 Auto Scaling ElastiCache 을 사용하여 를 Valkey 또는 Redis OSS 복제 그룹에 등록하는 것입니다.

클러스터에서 ElastiCache Auto Scaling을 사용하려면 먼저 Valkey 또는 Redis OSS Auto Scaling을 사용하여 ElastiCache 에 클러스터를 등록해야 합니다. 이렇게 하면 해당 클러스터에 적용할 크기 조정 차원과 제한을 정의할 수 있습니다. Valkey 또는 Redis OSS Auto Scaling을 ElastiCache 사용하면 샤

드당 클러스터 복제본 수를 나타내는 `elasticache:replication-group:Replicas` 확장 가능한 차원을 따라 클러스터를 동적으로 크기 조정합니다.

사용 CLI

ElastiCache 클러스터를 등록하려면 다음 파라미터와 함께 register-scalable-target 명령을 사용합니다.

- --service-namespace – 이 값을 elasticache로 설정합니다.
- --resource-id – ElastiCache 클러스터의 리소스 식별자입니다. 이 파라미터의 경우 리소스 유형은 ReplicationGroup 이고 고유 식별자는 클러스터의 이름입니다. 예를 들어 입니다`replication-group/myscalablecluster`.
- --scalable-dimension – 이 값을 `elasticache:replication-group:Replicas`로 설정합니다.
- --min-capacity – Valkey 또는 Redis OSS Auto Scaling ElastiCache 으로 관리할 최소 복제본 수입니다. --min-capacity, --max-capacity 및 클러스터의 복제본 수 간의 관계에 대한 자세한 내용은 최소 및 최대 용량 섹션을 참조하세요.
- --max-capacity – Valkey 또는 Redis OSS Auto Scaling ElastiCache 으로 관리할 최대 복제본 수입니다. --min-capacity, --max-capacity 및 클러스터의 복제본 수 간의 관계에 대한 자세한 내용은 최소 및 최대 용량 섹션을 참조하세요.

Example

다음 예제에서는 라는 이름의 Valkey 또는 Redis OSS 클러스터 ElastiCache 에 를 등록합니다`myscalablecluster`. 등록은 1개에서 5개까지 복제본을 포함하도록 클러스터 크기를 동적으로 조정해야 함을 나타냅니다.

Linux, macOS, Unix의 경우:

```
aws application-autoscaling register-scalable-target \
    --service-namespace elasticache \
    --resource-id replication-group/myscalablecluster \
    --scalable-dimension elasticache:replication-group:Replicas \
    --min-capacity 1 \
    --max-capacity 5 \
```

Windows의 경우:

```
aws application-autoscaling register-scalable-target ^
```

```
    --service-namespace elasticache ^
    --resource-id replication-group/myscalablecluster ^
    --scalable-dimension elasticache:replication-group:Replicas ^
    --min-capacity 1 ^
    --max-capacity 5 ^
```

사용 API

ElastiCache 클러스터를 등록하려면 다음 파라미터와 함께 register-scalable-target 명령을 사용합니다.

- ServiceNamespace - 이 값을 elasticache로 설정합니다.
- ResourceID - ElastiCache 클러스터의 리소스 식별자입니다. 이 파라미터의 경우 리소스 유형은 ReplicationGroup 이고 고유 식별자는 클러스터의 이름입니다. 예를 들어 입니다`replication-group/myscalablecluster`.
- ScalableDimension – 이 값을 로 설정합니다`elasticache:replication-group:Replicas`.
- MinCapacity - Valkey 또는 Redis OSS Auto Scaling ElastiCache 으로 관리할 최소 복제본 수입니다. --min-capacity, --max-capacity 및 클러스터의 복제본 수 간의 관계에 대한 자세한 내용은 최소 및 최대 용량 섹션을 참조하세요.
- MaxCapacity - Valkey 또는 Redis OSS Auto Scaling ElastiCache 으로 관리할 최대 복제본 수입니다. --min-capacity, --max-capacity 및 클러스터의 복제본 수 간의 관계에 대한 자세한 내용은 최소 및 최대 용량 섹션을 참조하세요.

Example

다음 예제에서는 Application Auto Scaling `myscalablecluster`에 라는 클러스터를 등록합니다API. 이 등록은 1개에서 5개까지 복제본을 포함하도록 클러스터 크기를 동적으로 조정해야 함을 나타냅니다.

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.RegisterScalableTarget
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS
```

```
{
    "ServiceNamespace": "elasticache",
    "ResourceId": "replication-group/myscalablecluster",
    "ScalableDimension": "elasticache:replication-group:Replicas",
    "MinCapacity": 1,
    "MaxCapacity": 5
}
```

조정 정책 정의

대상 추적 조정 정책 구성은 지표와 대상 값이 정의된 JSON 블록으로 표시됩니다. 크기 조정 정책 구성을 텍스트 파일의 JSON 블록으로 저장할 수 있습니다. AWS CLI 또는 Application Auto Scaling을 호출할 때 해당 텍스트 파일을 사용합니다API. 정책 구성 구문에 대한 자세한 내용은 Application Auto Scaling 참조TargetTrackingScalingPolicyConfiguration의 섹션을 참조하세요. Auto Scaling API

대상 추적 조정 정책 구성을 정의하기 위해 다음과 같은 옵션을 사용할 수 있습니다.

주제

- 미리 정의된 지표 사용
- 조정 정책 편집
- 조정 정책 삭제
- AWS CloudFormation Auto Scaling 정책에 사용
- 예약된 조정

미리 정의된 지표 사용

대상 추적 조정 정책 구성은 지표와 대상 값이 정의된 JSON 블록으로 표시됩니다. 크기 조정 정책 구성을 텍스트 파일의 JSON 블록으로 저장할 수 있습니다. AWS CLI 또는 Application Auto Scaling을 호출할 때 해당 텍스트 파일을 사용합니다API. 정책 구성 구문에 대한 자세한 내용은 Application Auto Scaling 참조TargetTrackingScalingPolicyConfiguration의 섹션을 참조하세요. Auto Scaling API

대상 추적 조정 정책 구성을 정의하기 위해 다음과 같은 옵션을 사용할 수 있습니다.

주제

- 미리 정의된 지표 사용
- 사용자 지정 지표 사용
- 휴지 기간 사용

- 스케일 인 활동 비활성화
- Valkey 또는 Redis OSS 클러스터를 ElastiCache 사용하여 에 조정 정책 적용

미리 정의된 지표 사용

미리 정의된 지표를 사용하여 Valkey 또는 Redis OSS Auto Scaling ElastiCache 에서 대상 추적을 사용하는 와 Valkey 또는 Redis OSS 클러스터 ElastiCache 에 대한 대상 추적 조정 정책을 빠르게 정의할 수 있습니다. Auto Scaling 현재 는 ElastiCache 복제본 Auto Scaling에서 다음과 같은 사전 정의된 지표를 ElastiCache 지원합니다.

`ElastiCacheReplicaEngineCPUUtilization` - 클러스터의 모든 복제본 CloudWatch 에서 의 EngineCPUUtilization 지표의 평균 값입니다. 필수 ReplicationGroupId 및 역할 복제본에 대한 집계 지표 값은 CloudWatch ElastiCache `ReplicationGroupId, Role`의 에서 확인할 수 있습니다.

조정 정책에서 미리 정의된 지표를 사용하려면 조정 정책을 위한 대상 추적 구성을 생성합니다. 미리 정의된 지표의 `PredefinedMetricSpecification` 및 해당 지표의 대상 값에 대한 `TargetValue`를 이 구성에 포함해야 합니다.

사용자 지정 지표 사용

사용자 지정 지표를 사용하여 사용자 지정 요구 사항에 맞는 대상 추적 조정 정책을 정의할 수 있습니다. 크기 조정에 비례하여 변경되는 Valkey 또는 Redis 지표 ElastiCache 를 사용하여 OSS 를 기반으로 사용자 지정 지표를 정의할 수 있습니다. 모든 ElastiCache 지표가 대상 추적에 사용되는 것은 아닙니다. 지표는 유효한 사용량 수치로서 인스턴스의 사용량을 설명해야 합니다. 클러스터에 있는 복제본 수에 비례하여 지표 값이 증가하거나 줄어들어야 합니다. 지표 데이터를 사용하여 복제본 수를 비례적으로 늘리거나 줄이려면 이 비례적인 증가나 감소가 필요합니다.

Example

다음 예제에서는 조정 정책의 대상 추적 구성을 설명합니다. 이 구성에서 사용자 지정 지표는 라는 클러스터의 모든 복제본에서 평균 CPU 사용률 50%를 기준으로 클러스터를 조정합니다`my-db-cluster`.

```
{"TargetValue": 50,
    "CustomizedMetricSpecification":
    {"MetricName": "EngineCPUUtilization",
        "Namespace": "AWS/ElastiCache",
        "Dimensions": [
            {"Name": "RelicationGroup","Value": "my-db-cluster"},
            {"Name": "Role","Value": "REPLICA"}
```

```
        ],
        "Statistic": "Average",
        "Unit": "Percent"
    }
}
```

휴지 기간 사용

ScaleOutCooldown에 초 단위로 값을 지정하여 클러스터를 확장하기 위한 휴지 기간을 추가할 수 있습니다. 마찬가지로 ScaleInCooldown에 초 단위로 값을 추가하여 클러스터를 축소하기 위한 휴지 기간을 추가할 수 있습니다. ScaleInCooldown 및 에 대한 자세한 내용은 Application Auto Scaling 참조TargetTrackingScalingPolicyConfiguration의 섹션을 ScaleOutCooldown참조하세요. Auto Scaling API 다음 예제에서는 조정 정책의 대상 추적 구성을 설명합니다. 이 구성에서는 ElastiCacheReplicaEngineCPUUtilization사전 정의된 지표를 사용하여 해당 클러스터의 모든 복제본에서 평균 CPU 사용률을 40%로 기준으로 클러스터를 조정합니다. 구성에서는 스케일 인 휴지 기간 10분과 스케일 아웃 휴지 기간 5분을 제공합니다.

```
{"TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {"PredefinedMetricType": "ElastiCacheReplicaEngineCPUUtilization"
    },
    "ScaleInCooldown": 600,
    "ScaleOutCooldown": 300
}
```

스케일 인 활동 비활성화

스케일 인 활동을 비활성화하여 Valkey 또는 Redis OSS 클러스터 ElastiCache 를 사용하여 에서 대상 추적 조정 정책 구성이 조정되지 않도록 할 수 있습니다. 스케일 인 활동을 비활성화하면 조정 정책에서 필요에 따라 복제본을 추가할 수 있지만 삭제할 수는 없습니다.

DisableScaleIn에 부울 값을 지정하여 클러스터에 대한 스케일 인 활동을 활성화하거나 비활성화할 수 있습니다. 에 대한 자세한 내용은 Application Auto Scaling 참조TargetTrackingScalingPolicyConfiguration의 섹션을 DisableScaleIn참조하세요. Auto Scaling API

Example

다음 예제에서는 조정 정책의 대상 추적 구성을 설명합니다. 이 구성에서 ElastiCacheReplicaEngineCPUUtilization 사전 정의된 지표는 해당 클러스터의 모든 복제본

에서 평균 CPU 사용률 40%를 기준으로 클러스터를 조정합니다. 구성에서는 조정 정책의 스케일 인 활동을 비활성화합니다.

```
{"TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {"PredefinedMetricType": "ElastiCacheReplicaEngineCPUUtilization"
    },
    "DisableScaleIn": true
}
```

Valkey 또는 Redis OSS 클러스터를 ElastiCache 사용하여 에 조정 정책 적용

Valkey 또는 Redis OSS Auto Scaling ElastiCache 을 사용하여 에 클러스터를 등록하고 스케일링 정책을 정의한 후 등록된 클러스터에 스케일링 정책을 적용합니다. Valkey 또는 Redis OSS 클러스터가 ElastiCache 있는 에 조정 정책을 적용하려면 AWS CLI 또는 Application Auto Scaling 을 사용할 수 있습니다API.

사용 AWS CLI

Valkey 또는 Redis OSS 클러스터를 ElastiCache 사용하여 에 조정 정책을 적용하려면 다음 파라미터와 함께 put-scaling-policy 명령을 사용합니다.

- --policy-name – 조정 정책의 이름입니다.
- --policy-type – 이 값을 `TargetTrackingScaling`으로 설정합니다.
- --resource-id – 클러스터의 리소스 식별자입니다. 이 파라미터의 경우 리소스 유형은 ReplicationGroup 이고 고유 식별자는 클러스터의 이름입니다. 예를 들어 입니다`replication-group/myscalablecluster`.
- --service-namespace – 이 값을 elasticache로 설정합니다.
- --scalable-dimension – 이 값을 `elasticache:replication-group:Replicas`로 설정합니다.
- --target-tracking-scaling-policy-configuration – 클러스터에 사용할 대상 추적 조정 정책 구성입니다.

Example

다음 예제에서는 Valkey 또는 Redis OSS Auto Scaling을 `myscalablecluster` ElastiCache 사용하여 라는 클러스터`myscalablepolicy`에 라는 대상 추적 조정 정책을 적용합니다. 이를 위해 `config.json`이라는 파일에 저장된 정책 구성을 사용합니다.

Linux, macOS 또는 Unix의 경우는 다음과 같습니다.

```
aws application-autoscaling put-scaling-policy \
    --policy-name myscalablepolicy \
    --policy-type TargetTrackingScaling \
    --resource-id replication-group/myscalablecluster \
    --service-namespace elasticache \
    --scalable-dimension elasticache:replication-group:Replicas \
    --target-tracking-scaling-policy-configuration file://config.json
```

```
{"TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {"PredefinedMetricType": "ElastiCacheReplicaEngineCPUUtilization"
    },
    "DisableScaleIn": true
}
```

Windows의 경우:

```
aws application-autoscaling put-scaling-policy ^
    --policy-name myscalablepolicy ^
    --policy-type TargetTrackingScaling ^
    --resource-id replication-group/myscalablecluster ^
    --service-namespace elasticache ^
    --scalable-dimension elasticache:replication-group:Replicas ^
    --target-tracking-scaling-policy-configuration file://config.json
```

사용 API

Application Auto Scaling 을 ElastiCache 사용하여 Valkey 또는 Redis OSS 클러스터를 사용하여 에 조정 정책을 적용하려면 다음 파라미터와 함께 PutScalingPolicy Application Auto Scaling API 작업을 API사용합니다. Auto Scaling

- PolicyName - 조정 정책의 이름입니다.
- PolicyType – 이 값을 로 설정합니다`TargetTrackingScaling`.
- ResourceID - 클러스터의 리소스 식별자입니다. 이 파라미터의 경우 리소스 유형은 ReplicationGroup 이고 고유 식별자는 와 같은 ElastiCache (Redis OSS) 클러스터의 이름입니다`replication-group/myscalablecluster`.

- ServiceNamespace - 이 값을 elasticache로 설정합니다.
- ScalableDimension – 이 값을 로 설정합니다`elasticache:replication-group:Replicas`.
- TargetTrackingScalingPolicyConfiguration – 클러스터에 사용할 대상 추적 조정 정책 구성입니다.

Example

다음 예제에서는 Valkey 또는 Redis OSS Auto Scaling을 `myscalablecluster` ElastiCache 사용하여 라는 클러스터`scalablepolicy`에 라는 대상 추적 조정 정책을 적용합니다. `ElastiCacheReplicaEngineCPUUtilization` 사전 정의 지표를 기반으로 하는 정책 구성을 사용합니다.

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.PutScalingPolicy
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS
{
    "PolicyName": "myscalablepolicy",
    "ServiceNamespace": "elasticache",
    "ResourceId": "replication-group/myscalablecluster",
    "ScalableDimension": "elasticache:replication-group:Replicas",
    "PolicyType": "TargetTrackingScaling",
    "TargetTrackingScalingPolicyConfiguration": {
        "TargetValue": 40.0,
        "PredefinedMetricSpecification":
        {
            "PredefinedMetricType": "ElastiCacheReplicaEngineCPUUtilization"
        }
    }
}
```

조정 정책 편집

AWS Management Console, AWS CLI또는 Application Auto Scaling을 사용하여 조정 정책을 편집할 수 있습니다API.

를 사용하여 조정 정책 편집 AWS Management Console

AWS Management Console을 사용하여 미리 정의된 지표 유형이 있는 정책을 편집할 수 있습니다.

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 탐색 창에서 Valkey 또는 RedisOSS를 선택합니다.
3. 정책을 추가할 클러스터를 선택합니다(클러스터 이름 왼쪽에 있는 버튼이 아닌 클러스터 이름 선택).
4. Auto Scaling 정책 탭을 선택합니다.
5. 조정 정책(Scaling policies)에서 변경할 자동 크기 조정 정책 왼쪽에 있는 버튼을 선택한 다음 수정(Modify)을 선택합니다.
6. 정책을 필요에 따라 변경합니다.
7. 수정을 선택합니다.
8. 정책을 변경합니다.
9. 수정을 선택합니다.

AWS CLI 또는 Application Auto Scaling을 사용하여 조정 정책 편집 API

AWS CLI 또는 Application Auto ScalingAPI을 사용하여 조정 정책을 적용하는 것과 동일한 방식으로 조정 정책을 편집할 수 있습니다.

- Application Auto Scaling을 사용할 때 `PolicyName` 파라미터에 편집하려는 정책의 이름을 API지정합니다. 변경할 파라미터의 새로운 값을 지정합니다.

자세한 내용은 Valkey 또는 Redis OSS 클러스터를 ElastiCache 사용하여 에 조정 정책 적용 단원을 참조하십시오.

조정 정책 삭제

AWS Management Console, AWS CLI 또는 Application Auto Scaling을 사용하여 조정 정책을 삭제할 수 있습니다. API

를 사용하여 조정 정책 삭제 AWS Management Console

AWS Management Console을 사용하여 미리 정의된 지표 유형이 있는 정책을 편집할 수 있습니다.

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 탐색 창에서 Valkey 또는 RedisOSS를 선택합니다.
3. 자동 크기 조정 정책을 삭제할 클러스터를 선택합니다.
4. Auto Scaling 정책 탭을 선택합니다.
5. 조정 정책(Scaling policies)에서 자동 크기 조정 정책을 선택한 후 삭제(Delete)를 선택합니다.

AWS CLI 또는 Application Auto Scaling을 사용하여 조정 정책 삭제 API

AWS CLI 또는 Application Auto Scaling을 사용하여 ElastiCache 클러스터에서 조정 정책을 API 삭제할 수 있습니다.

CLI

Valkey 또는 Redis OSS 클러스터를 ElastiCache 사용하여 에서 조정 정책을 삭제하려면 다음 파라미터와 함께 delete-scaling-policy 명령을 사용합니다.

- --policy-name – 조정 정책의 이름입니다.
- --resource-id – 클러스터의 리소스 식별자입니다. 이 파라미터의 경우 리소스 유형은 ReplicationGroup 이고 고유 식별자는 클러스터의 이름입니다. 예를 들어 입니다`replication-group/myscalablecluster`.
- --service-namespace – 이 값을 elasticache로 설정합니다.
- --scalable-dimension – 이 값을 `elasticache:replication-group:Replicas`로 설정합니다.

Example

다음 예제에서는 라는 ELC; 클러스터`myscalablepolicy`에서 라는 대상 추적 조정 정책을 삭제합니다`myscalablecluster`.

Linux, macOS, Unix의 경우:

```
aws application-autoscaling delete-scaling-policy \
    --policy-name myscalablepolicy \
    --resource-id replication-group/myscalablecluster \
    --service-namespace elasticache \
    --scalable-dimension elasticache:replication-group:Replicas \
```

Windows의 경우:

```
aws application-autoscaling delete-scaling-policy ^
    --policy-name myscalablepolicy ^
    --resource-id replication-group/myscalablecluster ^
    --service-namespace elasticache ^
    --scalable-dimension elasticache:replication-group:Replicas ^
```

API

Valkey 또는 Redis OSS 클러스터를 ElastiCache 사용하여 에서 조정 정책을 삭제하려면 다음 파라미터와 함께 DeleteScalingPolicy Application Auto Scaling API 작업을 사용합니다.

- PolicyName - 조정 정책의 이름입니다.
- ResourceID - 클러스터의 리소스 식별자입니다. 이 파라미터의 경우 리소스 유형은 ReplicationGroup 이고 고유 식별자는 클러스터의 이름입니다. 예를 들어 입니다`replication-group/myscalablecluster`.
- ServiceNamespace - 이 값을 elasticache로 설정합니다.
- ScalableDimension – 이 값을 로 설정합니다`elasticache:replication-group:Replicas`.

다음 예제에서는 Application Auto Scaling `myscalablecluster`로 명명된 클러스터`myscalablepolicy`에서 명명된 대상 추적 조정 정책을 삭제합니다API. Auto Scaling

```
POST / HTTP/1.1
>>>>>>> mainline
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.DeleteScalingPolicy
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS
{
    "PolicyName": "myscalablepolicy",
    "ServiceNamespace": "elasticache",
    "ResourceId": "replication-group/myscalablecluster",
    "ScalableDimension": "elasticache:replication-group:Replicas"
```

```
}
```

AWS CloudFormation Auto Scaling 정책에 사용

이 조각은 AWS:::: 리소스를 사용하여 예약된 작업을 생성하고 ::ElastiCache: 리소스 ReplicationGroup에 적용하는 방법을 보여줍니다AWSApplicationAutoScalingScalableTarget. Fn::Join 및 Ref 내장 함수를 사용하여 동일한 템플릿에 지정된 `AWS::ElastiCache::ReplicationGroup` 리소스의 논리적 이름으로 `ResourceId` 속성을 구성합니다.

```
ScalingTarget:
   Type: 'AWS::ApplicationAutoScaling::ScalableTarget'
   Properties:
     MaxCapacity: 0
     MinCapacity: 0
     ResourceId: !Sub replication-group/${logicalName}
     ScalableDimension: 'elasticache:replication-group:Replicas'
     ServiceNamespace: elasticache
     RoleARN: !Sub "arn:aws:iam::${AWS::AccountId}:role/aws-
service-role/elasticache.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG"

  ScalingPolicy:
    Type: "AWS::ApplicationAutoScaling::ScalingPolicy"
    Properties:
      ScalingTargetId: !Ref ScalingTarget
      ServiceNamespace: elasticache
      PolicyName: testpolicy
      PolicyType: TargetTrackingScaling
      ScalableDimension: 'elasticache:replication-group:Replicas'
      TargetTrackingScalingPolicyConfiguration:
        PredefinedMetricSpecification:
          PredefinedMetricType: ElastiCacheReplicaEngineCPUUtilization
        TargetValue: 40
```

예약된 조정

일정을 기반으로 조정을 수행하면 수요에 따른 로드 변경에 맞게 애플리케이션을 조정할 수 있습니다. 예약된 크기 조정을 사용하려면 Valkey 또는 Redis ElastiCache 를 사용하여 특정 시간에 크기 조정 활동을 OSS 수행하도록 지시하는 예약된 작업을 생성합니다. 예약된 작업을 생성할 때 스케일링 활동이 발생하는 시점, 최소 용량 및 최대 용량인 Valkey 또는 Redis OSS 클러스터가 ElastiCache 있는 기존를 지정합니다. 규모를 한 번만 조정하거나 반복되는 일정으로 조정하도록 예약된 작업을 생성할 수 있습니다.

이미 존재하는 ElastiCache Valkey 또는 Redis OSS 클러스터에서만 에 대한 예약된 작업을 생성할 수 있습니다. 클러스터를 생성하는 동시에 예약된 작업을 생성할 수는 없습니다.

예약된 작업 생성, 관리 및 삭제와 관련된 용어에 대한 자세한 내용은 예약된 작업 생성, 관리 및 삭제에 일반적으로 사용되는 명령을 참조하세요.

1회성 예약된 작업을 생성하려면

샤드 차원과 유사합니다. 예약된 조정 을 참조하세요.

예약된 작업 삭제

샤드 차원과 유사합니다. 예약된 조정 을 참조하세요.

AWS CLI 를 사용하여 예약된 조정을 관리하려면

다음 애플리케이션 자동 크기 조정을 사용합니다APIs.

- put-scheduled-action
- describe-scheduled-actions
- delete-scheduled-action

Auto Scaling 정책을 생성하는 AWS CloudFormation 데 사용

이 조각은 AWS:::: 리소스를 사용하여 예약된 작업을 생성하고 ::ElastiCache: 리소스 ReplicationGroup에 적용하는 방법을 보여줍니다AWSApplicationAutoScalingScalableTarget. Fn::Join 및 Ref 내장 함수를 사용하여 동일한 템플릿에 지정된 `AWS::ElastiCache::ReplicationGroup` 리소스의 논리적 이름으로 `ResourceId` 속성을 구성합니다.

```
ScalingTarget:
   Type: 'AWS::ApplicationAutoScaling::ScalableTarget'
   Properties:
     MaxCapacity: 0
     MinCapacity: 0
     ResourceId: !Sub replication-group/${logicalName}
     ScalableDimension: 'elasticache:replication-group:Replicas'
     ServiceNamespace: elasticache
     RoleARN: !Sub "arn:aws:iam::${AWS::AccountId}:role/aws-
service-role/elasticache.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG"
```

```
    ScheduledActions:
      - EndTime: '2020-12-31T12:00:00.000Z'
        ScalableTargetAction:
          MaxCapacity: '5'
          MinCapacity: '2'
        ScheduledActionName: First
        Schedule: 'cron(0 18 * * ? *)'
```

클러스터 모드 수정

Valkey 및 RedisOSS는 샤딩 및 복제를 지원하는 분산 인 메모리 데이터베이스입니다. ElastiCache Valkey 및 Redis OSS 클러스터는 여러 노드에서 데이터를 분할할 수 있는 분산 구현입니다. ElastiCache (Redis OSS) 클러스터에는 클러스터 모드 활성화()와 클러스터 모드 비활성화(CME)라는 두 가지 작동 모드가 있습니다CMD. 에서 Valkey 및 Redis OSS 엔진CME은 여러 샤드 및 노드가 있는 분산 데이터베이스로 작동하는 반면 CMD에서 Valkey 및 Redis는 단일 노드로 OSS 작동합니다.

에서 CMD로 마이그레이션하기 전에 CME다음 조건을 충족해야 합니다.

> **Important**
>
> 클러스터 모드 구성은 클러스터 모드 비활성화에서 클러스터 모드 활성화로만 변경할 수 있습니다. 이 구성은 되돌릴 수 없습니다.

- 클러스터에는 데이터베이스 0에만 키가 있을 수 있습니다.
- 애플리케이션은 클러스터 프로토콜을 사용하고 구성 엔드포인트를 사용할 수 있는 Valkey 또는 Redis OSS 클라이언트를 사용해야 합니다.
- 최소 1개의 복제본이 있는 클러스터에서 자동 장애 조치를 활성화해야 합니다.
- 마이그레이션에 필요한 최소 엔진 버전은 Valkey 7.2 이상 또는 Redis OSS 7.0 이상입니다.

에서 CMD로 마이그레이션하려면 CME클러스터 모드 구성을 클러스터 모드 비활성화에서 클러스터 모드 활성화로 변경해야 합니다. 이는 마이그레이션 프로세스 중에 클러스터 가용성을 보장하는 2단계 절차입니다.

> **Note**
>
> 파라미터 그룹에 클러스터 지원 구성을 제공해야 합니다. 즉, 클러스터 지원 파라미터는 yes로 설정되어 있어야 합니다. 기본 파라미터 그룹을 사용하는 경우 ElastiCache (Redis OSS)는 클

러스터가 활성화된 구성으로 해당 기본 파라미터 그룹을 자동으로 선택합니다. 클러스터 활성화 파라미터 값은 CMD 클러스터에 no 대해 로 설정됩니다. 클러스터가 호환 모드로 이동하면 수정 작업의 일부로 클러스터 지원 파라미터 값이 yes로 업데이트됩니다.
자세한 내용은 파라미터 그룹을 사용하여 엔진 ElastiCache 파라미터 구성 단원을 참조하세요.

1. 준비 - 테스트 CME 클러스터를 생성하고 스택이 사용할 준비가 되었는지 확인합니다. ElastiCache (Redis OSS)는 준비 상태를 확인할 수 없습니다. 자세한 내용은 Valkey 또는 Redis용 클러스터 생성 OSS 단원을 참조하십시오.
2. 기존 CMD 클러스터 구성을 클러스터 모드 호환으로 수정 - 이 모드에서는 단일 샤드가 배포되고 ElastiCache (Redis OSS)가 단일 노드로 작동하지만 단일 샤드 클러스터로도 작동합니다. 호환 모드란 클라이언트 애플리케이션이 두 프로토콜 중 하나를 사용하여 클러스터와 통신할 수 있음을 의미합니다. 이 모드에서는 애플리케이션을 재구성하여 Valkey 또는 Redis OSS 클러스터 프로토콜 및 구성 엔드포인트 사용을 시작해야 합니다. Valkey 또는 Redis OSS 클러스터 모드를 클러스터 모드 호환으로 변경하려면 다음 단계를 따르세요.

   > **Note**
   >
   > 호환 모드에서는 크기 조정 및 엔진 버전과 같은 다른 수정 작업이 클러스터에 허용되지 않습니다. 또한 ModifyReplicationGroup 요청 내에서 클러스터 모드 파라미터를 정의할 때 파라미터(제외`cacheParameterGroupName`)를 수정할 수 없습니다.

   a. 를 사용하여 를 AWS Management Console참조복제 그룹 수정하고 클러스터 모드를 호환됨으로 설정합니다.
   b. 를 사용하여 를 API참조ModifyReplicationGroup하고 `ClusterMode` 파라미터를 로 업데이트합니다`compatible`.
   c. 를 사용하여 를 AWS CLI참조modify-replication-group하고 `cluster-mode` 파라미터를 로 업데이트합니다`compatible`.

   Valkey 또는 Redis OSS 클러스터 모드를 클러스터 모드 호환으로 변경하면 DescribeReplicationGroupsAPI가 ElastiCache (Redis OSS) 클러스터 구성 엔드포인트를 반환합니다. 클러스터 구성 엔드포인트는 애플리케이션이 클러스터에 연결하는 데 사용할 수 있는 단일 엔드포인트입니다. 자세한 내용은 에서 연결 엔드포인트 찾기 ElastiCache 단원을 참조하십시오.
3. 클러스터 구성을 클러스터 모드활성화로 수정 – 클러스터 모드가 클러스터 모드 호환으로 설정되면 두 번째 단계는 클러스터 모드 활성화로 클러스터 구성을 수정하는 것입니다. 이 모드에서는 단

일 샤드가 실행되고 고객은 이제 클러스터 크기를 조정하거나 다른 클러스터 구성을 수정할 수 있습니다.

클러스터 모드를 활성화로 변경하려면 아래 단계를 따릅니다.

시작하기 전에 Valkey 또는 Redis OSS 클라이언트가 클러스터 프로토콜을 사용하여 로 마이그레이션되었고 클러스터의 구성 엔드포인트가 사용되지 않는지 확인합니다.

a. 를 사용하여 를 AWS Management Console참조복제 그룹 수정하고 클러스터 모드를 활성화됨으로 설정합니다.
b. 를 사용하여 를 API참조ModifyReplicationGroup하고 `ClusterMode` 파라미터를 로 업데이트합니다`enabled`.
c. 를 사용하여 를 AWS CLI참조modify-replication-group하고 `cluster-mode` 파라미터를 로 업데이트합니다`enabled`.

클러스터 모드를 활성화로 변경하면 엔드포인트가 Valkey 또는 Redis OSS 클러스터 사양에 따라 구성됩니다. DescribeReplicationGroups API 는 클러스터 모드 파라미터를 로 반환`enabled`하고 이제 애플리케이션에서 클러스터에 연결하는 데 사용할 수 있는 클러스터 엔드포인트를 반환합니다.

클러스터 모드가 활성화로 변경되면 클러스터 엔드포인트가 변경된다는 점에 유의하세요. 새 엔드포인트로 애플리케이션을 업데이트해야 합니다.

클러스터 모드 호환에서 비활성화된 클러스터 모드(CMD)로 되돌리고 원래 구성을 보존하도록 선택할 수도 있습니다.

클러스터 모드 호환에서 클러스터 모드 비활성화로 클러스터 구성을 수정합니다.

1. 를 사용하여 를 AWS Management Console참조복제 그룹 수정하고 클러스터 모드를 비활성화됨으로 설정합니다.
2. 를 사용하여 를 API참조ModifyReplicationGroup하고 `ClusterMode` 파라미터를 로 업데이트합니다`disabled`.
3. 를 사용하여 를 AWS CLI참조modify-replication-group하고 `cluster-mode` 파라미터를 로 업데이트합니다`disabled`.

클러스터 모드를 비활성화로 변경하면 DescribeReplicationGroupsAPI는 클러스터 모드 파라미터를 로 반환합니다`disabled`.

글로벌 데이터 스토어를 사용하여 AWS 리전 간 복제

> **Note**
>
> 글로벌 데이터 스토어는 현재 자체 설계된 클러스터에만 사용할 수 있습니다.

Global Datastore 기능을 사용하면 AWS 리전 간 완전 관리형, 빠르고 안정적이며 안전한 Valkey 또는 Redis OSS 클러스터 복제를 수행할 수 있습니다. 이 기능을 사용하면 리전 간 읽기 전용 복제본 클러스터를 생성하여 AWS 리전 간 지연 시간이 짧은 읽기 및 재해 복구를 활성화할 수 있습니다.

다음 섹션에서는 글로벌 데이터 스토어로 작업하는 방법에 대한 설명을 찾을 수 있습니다.

주제

개요

각 글로벌 데이터 스토어는 서로 복제하는 하나 이상의 클러스터 모음입니다.

글로벌 데이터 스토어는 다음과 같이 구성됩니다.

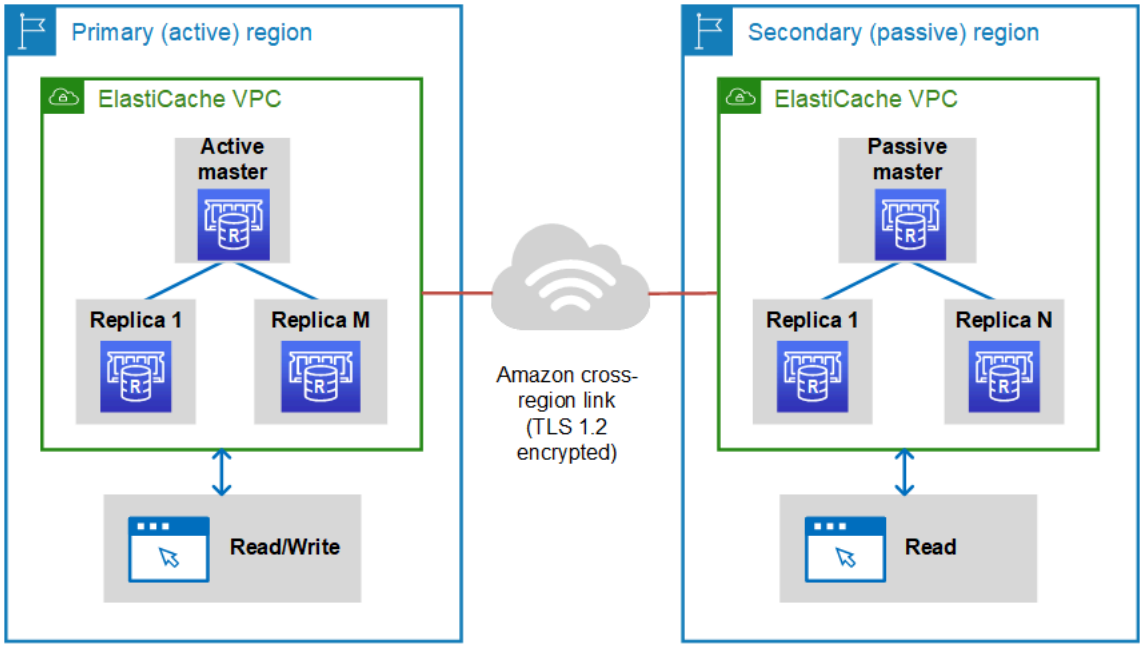
- 기본(활성) 클러스터 - 기본 클러스터는 글로벌 데이터 스토어 내의 모든 클러스터에 복제되는 쓰기를 허용합니다. 기본 클러스터는 읽기 요청도 허용합니다.
- 보조(수동) 클러스터 - 보조 클러스터는 읽기 요청만 허용하고 기본 클러스터에서 데이터 업데이트를 복제합니다. 보조 클러스터는 기본 클러스터와 다른 AWS 리전에 있어야 합니다.

Valkey 또는 Redis ElastiCache 를 사용하여 에서 글로벌 데이터 스토어를 생성하면 기본 클러스터에서 보조 클러스터로 데이터가 OSS자동으로 복제됩니다. Valkey 또는 Redis OSS 데이터를 복제해야 하는 AWS 리전을 선택한 다음 해당 AWS 리전에 보조 클러스터를 생성합니다. ElastiCache 그런 다음는 두 클러스터 간에 데이터의 자동 비동기 복제를 설정하고 관리합니다.

Valkey 또는 Redis용 글로벌 데이터 스토어를 사용하면 다음과 같은 이점이 OSS 있습니다.

- 지리로컬 성능 - 추가 AWS 리전에 원격 복제본 클러스터를 설정하고 이들 간에 데이터를 동기화하면 해당 AWS 리전의 데이터 액세스 지연 시간을 줄일 수 있습니다. 글로벌 데이터 스토어는 AWS 리전 간에 지연 시간이 짧은 지리로컬 읽기를 제공하여 애플리케이션의 응답성을 높이는 데 도움이 될 수 있습니다.
- 재해 복구 - 글로벌 데이터 스토어의 기본 클러스터에서 성능 저하가 발생하는 경우 보조 클러스터를 새 기본 클러스터로 승격할 수 있습니다. 보조 클러스터가 포함된 모든 AWS 리전에 연결하여 연결할 수 있습니다.

다음 다이어그램은 글로벌 데이터 스토어가 작동하는 방식을 보여줍니다.



사전 조건 및 제한 사항

글로벌 데이터 스토어를 시작하기 전에 다음 사항에 유의하세요.

- 글로벌 데이터 스토어는 AWS 아시아 태평양(서울, 도쿄, 싱가포르, 시드니, 뭄바이, 오사카), 유럽(프랑크푸르트, 파리, 런던, 아일랜드, 스톡홀름), 미국 동부(버지니아 북부 및 오하이오), 미국 서부(캘리포니아 북부 및 오리건), 남미(상파울루), AWS GovCloud (미국 서부 및 미국 동부), 캐나다(중부) 리전, 중국(베이징 및 닝샤) 리전에서 지원됩니다.
- 글로벌 데이터 스토어의 모든 클러스터(기본 및 보조)에는 동일한 수의 기본 노드, 노드 유형, 엔진 버전 및 샤드 수(클러스터 모드가 활성화된 경우)가 있어야 합니다. 글로벌 데이터 스토어의 각 클러스터에는 해당 클러스터에 대한 로컬 읽기 트래픽을 수용하기 위해 다른 수의 읽기 복제본이 있을 수 있습니다.

기존 단일 노드 클러스터를 사용하려는 경우 복제를 활성화해야 합니다.

- 글로벌 데이터 스토어는 크기가 큰 인스턴스 이상에서 지원됩니다.
- 한 AWS 리전에서 최대 2개의 다른 AWS 리전의 보조 클러스터로 프라이머리 클러스터에 대한 복제를 설정할 수 있습니다.

> **Note**
>
> 중국(베이징) 리전과 중국(닝샤) 리전은 예외로, 두 리전 간에서만 복제가 실행될 수 있습니다.

- VPC 클러스터에서만 글로벌 데이터 스토어로 작업할 수 있습니다. 자세한 내용은 Amazon에서 ElastiCache 캐시에 액세스하기 위한 액세스 패턴 VPC 단원을 참조하십시오. EC2-Classic을 사용하는 경우 글로벌 데이터 스토어는 지원되지 않습니다. 자세한 내용은 Amazon 사용 설명서의 EC2-Classic을 참조하세요. EC2

> **Note**
>
> 현재 에서 로컬 영역 사용 ElastiCache에서 글로벌 데이터 스토어를 사용할 수 없습니다.

- ElastiCache 는 한 리전에서 다른 AWS 리전으로의 자동 장애 조치를 지원하지 않습니다. 필요한 경우 보조 클러스터를 수동으로 승격할 수 있습니다. 예시는 보조 클러스터를 기본 클러스터로 승격에서 확인하십시오.
- 기존 데이터에서 부트스트랩하려면 기존 클러스터를 기본 클러스터로 사용하여 글로벌 데이터 스토어를 생성합니다. 기존 클러스터를 보조 클러스터로 추가하는 것은 지원하지 않습니다. 클러스터를 보조 클러스터로 추가하는 프로세스로 데이터가 지워져 데이터가 손실될 수 있습니다.
- 파라미터 업데이트는 글로벌 데이터 스토어에 속한 클러스터의 로컬 파라미터 그룹을 수정할 때 모든 클러스터에 적용됩니다.
- 리전 클러스터를 수직(확장 및 축소) 및 수평(확장 및 축소)으로 확장할 수 있습니다. 글로벌 데이터 스토어를 수정하여 클러스터를 조정할 수 있습니다. 그러면 글로벌 데이터 스토어의 모든 리전 클러스터가 중단 없이 확장됩니다. 자세한 내용은 크기 조정 ElastiCache 단원을 참조하십시오.
- 글로벌 데이터 스토어는 저장 시 암호화, 전송 중 암호화 및 를 지원합니다AUTH.
- 글로벌 데이터 스토어는 인터넷 프로토콜 버전 6()을 지원하지 않습니다IPv6.
- 글로벌 데이터 스토어는 AWS KMS 키를 지원합니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 AWS key management service concepts를 참조하세요.

> **Note**
>
> 글로벌 데이터 스토어는 다음 규정에 따라 pub/sub 메시지를 지원합니다.
>
> - 클러스터 모드가 비활성화된 경우 pub/sub가 완전히 지원됩니다. 기본 AWS 리전의 기본 클러스터에 게시된 이벤트는 보조 AWS 리전으로 전파됩니다.
> - 클러스터 모드가 활성화된 경우 다음 사항이 적용됩니다.
>   - 키스페이스에 없는 게시된 이벤트의 경우 동일한 AWS 리전의 구독자만 이벤트를 수신합니다.
>   - 게시된 키스페이스 이벤트의 경우 모든 AWS 리전의 구독자는 이벤트를 수신합니다.

글로벌 데이터 스토어 사용(콘솔)

콘솔을 사용하여 글로벌 데이터 스토어를 생성하려면 다음 2단계 프로세스를 수행합니다.

1. 기존 클러스터를 사용하거나 새 클러스터를 생성하여 기본 클러스터를 생성합니다. 엔진은 Valkey 7.2 이상 또는 Redis OSS 5.0.6 이상이어야 합니다.
2. Valkey 7.2 이상 또는 Redis 5.0.6 엔진 이상을 사용하여 다른 AWS 리전에 최대 OSS 2개의 보조 클러스터를 추가합니다.

다음 절차에서는 Valkey 또는 Redis용 글로벌 데이터 스토어를 생성하고 ElastiCache 콘솔을 사용하여 다른 작업을 OSS 수행하는 방법을 안내합니다.

주제

- 기존 클러스터를 사용하여 글로벌 데이터 스토어 생성
- 새 기본 클러스터를 사용하여 새 글로벌 데이터 스토어 생성
- 글로벌 데이터 스토어 세부 정보 보기
- 글로벌 데이터 스토어에 리전 추가
- 글로벌 데이터 스토어 수정
- 보조 클러스터를 기본 클러스터로 승격
- 글로벌 데이터 스토어에서 리전 제거
- 글로벌 데이터 스토어 삭제

기존 클러스터를 사용하여 글로벌 데이터 스토어 생성

이 시나리오에서는 기존 클러스터를 사용하여 새 글로벌 데이터 스토어의 기본 클러스터 역할을 합니다. 그런 다음 별도의 AWS 리전에 보조 읽기 전용 클러스터를 생성합니다. 이 보조 클러스터는 기본 클러스터에서 자동 및 비동기 업데이트를 받습니다.

> **Important**
>
> 기존 클러스터는 Valkey 7.2 이상 또는 Redis OSS 5.0.6 이상의 엔진을 사용해야 합니다.

기존 클러스터를 사용하여 글로벌 데이터 스토어를 생성하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 글로벌 데이터 스토어를 선택한 다음 글로벌 데이터 스토어 생성 을 선택합니다.
3. 기본 클러스터 설정 페이지에서 다음을 수행합니다.
   - 글로벌 데이터 스토어 정보 필드에 새 글로벌 데이터 스토어의 이름을 입력합니다.
   - (선택 사항) 설명 값을 입력합니다.
4. 리전 클러스터 에서 기존 리전 클러스터 사용을 선택합니다.
5. 기존 클러스터 에서 사용하려는 기존 클러스터를 선택합니다.
6. 다음 옵션을 그대로 유지하세요. 기본 클러스터 구성과 일치하도록 미리 채워져 있으므로 변경할 수 없습니다.
   - 엔진 버전
   - 노드 유형
   - Parameter Group

     > **Note**
     >
     > ElastiCache 는 제공된 파라미터 그룹의 값에서 새 파라미터 그룹을 자동 생성하고 클러스터에 새 파라미터 그룹을 적용합니다. 글로벌 데이터 스토어의 파라미터를 수정하려면 이 새 파라미터 그룹을 사용합니다. 자동 생성된 각 파라미터 그룹은 하나의 클러스터에만 연결되므로 하나의 글로벌 데이터 스토어에만 연결됩니다.

   - 샤드 수

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

  > **Note**
  >
  > Customer Managed AWS KMS 키를 선택하고 키를 선택하여 다른 암호화 키를 제공할 수 있습니다. 자세한 내용은 Customer Managed AWS KMS 키 사용을 참조하세요.

- 전송 중 데이터 암호화 – 전송 데이터 암호화를 활성화합니다. 자세한 내용은 전송 중 데이터 암호화를 참조하세요. Valkey 7.2 이상 및 Redis OSS 엔진 버전 6.0 이상의 경우 전송 중 암호화를 활성화하면 다음 액세스 제어 옵션 중 하나를 지정하라는 메시지가 표시됩니다.
  - 액세스 제어 안 함 – 기본 설정입니다. 이 옵션은 제한하지 않는다는 의미입니다.
  - 사용자 그룹 액세스 제어 목록 - 사용 가능한 작업에 대한 사용자 및 권한 집합이 정의된 사용자 그룹을 선택합니다. 자세한 내용은 콘솔 및 를 사용하여 사용자 그룹 관리 CLI 단원을 참조하십시오.
  - AUTH 기본 사용자 - Valkey 또는 Redis OSS 서버의 인증 메커니즘입니다. 자세한 내용은 AUTH를 참조하세요.

7. (선택 사항) 필요에 따라 나머지 보조 클러스터 설정을 업데이트합니다. 기본 클러스터와 동일한 값으로 미리 채워지지만 해당 클러스터에 대한 특정 요구 사항을 충족하도록 업데이트할 수 있습니다.

   - Port
   - 복제본 개수
   - 서브넷 그룹
   - 기본 가용 영역
   - 보안 그룹
   - 고객 관리형(AWS KMS 키)
   - AUTH 토큰
   - 자동 백업 활성화
   - 백업 보관 기간
   - 백업 기간
   - 유지보수 윈도우
   - SNS 알림 주제

8. 생성(Create)을 선택합니다. 이렇게 하면 글로벌 데이터 스토어의 상태가 생성 중으로 설정됩니다. 기본 클러스터가 글로벌 데이터 스토어에 연결되고 보조 클러스터가 연결 중(Associating) 상태가 된 후 상태가 수정 중(Modifying)으로 전환됩니다.

   기본 클러스터 및 보조 클러스터가 글로벌 데이터 스토어와 연결되면 상태가 사용 가능으로 변경됩니다. 이 시점에서 읽기 및 쓰기를 허용하는 기본 클러스터와 기본 클러스터에서 복제된 읽기를 허용하는 보조 클러스터가 있습니다.

   이 페이지는 클러스터가 글로벌 데이터 스토어의 일부인지 여부를 나타내도록 업데이트되며, 여기에는 다음이 포함됩니다.

   - 글로벌 데이터 스토어 - 클러스터가 속한 글로벌 데이터 스토어의 이름입니다.
   - 글로벌 데이터 스토어 역할 - 클러스터의 역할(기본 또는 보조)입니다.

다른 AWS 리전에 최대 1개의 보조 클러스터를 추가할 수 있습니다. 자세한 내용은 글로벌 데이터 스토어에 리전 추가 단원을 참조하십시오.

새 기본 클러스터를 사용하여 새 글로벌 데이터 스토어 생성

새 클러스터를 사용하여 글로벌 데이터 스토어를 생성하도록 선택한 경우 다음 절차를 따르십시오.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 글로벌 데이터 스토어를 선택한 다음 글로벌 데이터 스토어 생성 을 선택합니다.
3. 기본 클러스터 설정(Primary cluster settings)에서 다음을 수행합니다.

   a. 클러스터 모드(Cluster mode)에서 사용 설정됨(Enabled) 또는 사용 중지됨(Disabled)을 선택합니다.

   b. 글로벌 데이터 스토어 정보의 경우 이름 값을 입력합니다. 는 접미사를 ElastiCache 사용하여 글로벌 데이터 스토어의 고유한 이름을 생성합니다. 여기에서 지정하는 접미사를 사용하여 글로벌 데이터 스토어를 검색할 수 있습니다.

   c. (선택 사항) 글로벌 데이터 스토어 설명에 값을 입력합니다.
4. 리전 클러스터(Regional cluster)에서 다음을 수행합니다.

   a. 리전 에서 사용 가능한 AWS 리전을 선택합니다.

   b. 새 리전 클러스터 생성(Create new regional cluster) 또는 기존 리전 클러스터 사용(Use existing regional cluster)을 선택합니다.

c. 새 리전 클러스터 생성(Create new regional cluster)을 선택한 경우 클러스터 정보(Cluster info)에서 클러스터의 이름과 설명(선택 사항)을 입력합니다.

d. 위치(Location)에서 다중 AZ(Multi-AZ) 및 자동 장애 조치(Auto-failover)의 기본 설정을 수락하는 것이 좋습니다.

5. 클러스터 설정(Cluster settings)에서 다음을 수행합니다.

a. 엔진 버전(Engine version)의 경우 사용 가능한 버전(5.0.6 이상)을 선택합니다.

b. 포트(Port)의 경우 기본 포트인 6379를 사용합니다. 다른 포트를 사용해야 하는 경우 포트 번호를 입력합니다.

c. 파라미터 그룹에서 파라미터 그룹을 선택하거나 새 파라미터 그룹을 만듭니다. 파라미터 그룹은 클러스터의 런타임 파라미터를 제어합니다. 파라미터 그룹에 대한 자세한 정보는 Valkey 및 Redis OSS 파라미터 및 ElastiCache 파라미터 그룹 생성 섹션을 참조하세요.

> **Note**
>
> 파라미터 그룹을 선택하여 엔진 구성 값을 설정하면 해당 파라미터 그룹이 글로벌 데이터 스토어의 모든 클러스터에 적용됩니다. 파라미터 그룹 페이지에서 yes/no 글로벌 속성은 파라미터 그룹이 글로벌 데이터 스토어의 일부인지 여부를 나타냅니다.

d. 노드 유형에서 아래쪽 화살표
(▼)
를 선택합니다. 노드 유형 변경 대화 상자에서 원하는 노드 유형의 인스턴스 패밀리 값을 선택합니다. 그런 다음 이 클러스터에 사용할 노드 유형을 선택한 다음 저장을 선택합니다.

자세한 정보는 노드 크기 선택 섹션을 참조하세요.

r6gd 노드 유형을 선택하는 경우 데이터 계층화가 자동으로 사용 설정됩니다. 자세한 내용은 의 데이터 계층화 ElastiCache 단원을 참조하십시오.

e. Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터를 생성하는 경우:

복제본 개수(Number of replicas)의 경우 이 클러스터에 대해 원하는 복제본 개수를 선택합니다.

f. Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터를 생성하는 경우:

i. 샤드 수 에서 이 Valkey 또는 Redis(클러스터 모드 활성화됨) 클러스터에 사용할 샤드 수 OSS(파티션/노드 그룹)를 선택합니다.

일부 버전의 Valkey 또는 RedisOSS(클러스터 모드 활성화됨)의 경우 클러스터의 샤드 수를 동적으로 변경할 수 있습니다.

- Redis OSS 3.2.10 이상 - 클러스터가 Redis OSS 3.2.10 이상 버전을 실행하는 경우 클러스터의 샤드 수를 동적으로 변경할 수 있습니다. 자세한 내용은 Valkey 또는 Redis에서 클러스터 크기 조정OSS(클러스터 모드 활성화됨) 단원을 참조하십시오.
- 기타 Redis OSS 버전 - 클러스터가 버전 3.2.10 OSS 이전에 Redis 버전을 실행하는 경우 다른 접근 방식이 있습니다. 이 경우 클러스터의 샤드 수를 변경하려면 새 샤드 수로 새 클러스터를 만듭니다. 자세한 내용은 백업에서 새 캐시로 복원 단원을 참조하십시오.

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

Valkey 또는 RedisOSS(클러스터 모드 활성화됨)에는 다음과 같은 제한 사항이 있습니다.

- 다중 AZ를 활성화한 경우 샤드당 복제본이 하나 이상 있어야 합니다.
- 콘솔을 사용하여 클러스터를 생성할 때 샤드마다 복제본 수가 동일합니다.
- 샤드당 읽기 전용 복제본 수가 고정되어 변경할 수 없습니다. 샤드당 복제본(API/CLI: 노드 그룹)이 더 많거나 적어야 하는 경우 새 복제본 수로 새 클러스터를 생성해야 합니다. 자세한 내용은 자습서: 외부에서 생성된 백업을 사용하여 자체 설계된 새 클러스터 검색 단원을 참조하십시오.

6. 서브넷 그룹 설정 에서 이 클러스터에 적용할 서브넷을 선택합니다. 는 기본 서브넷 그룹을 ElastiCache 제공하거나 새 IPv4 서브넷을 생성하도록 선택할 수 있습니다. 의 경우 IPv6 CIDR 블록을 사용하여 서브넷 그룹을 생성IPv6해야 합니다. 듀얼 스택 을 선택하는 경우 IPv6 또는 중에서 검색 IP 유형을 선택해야 합니다IPv4.

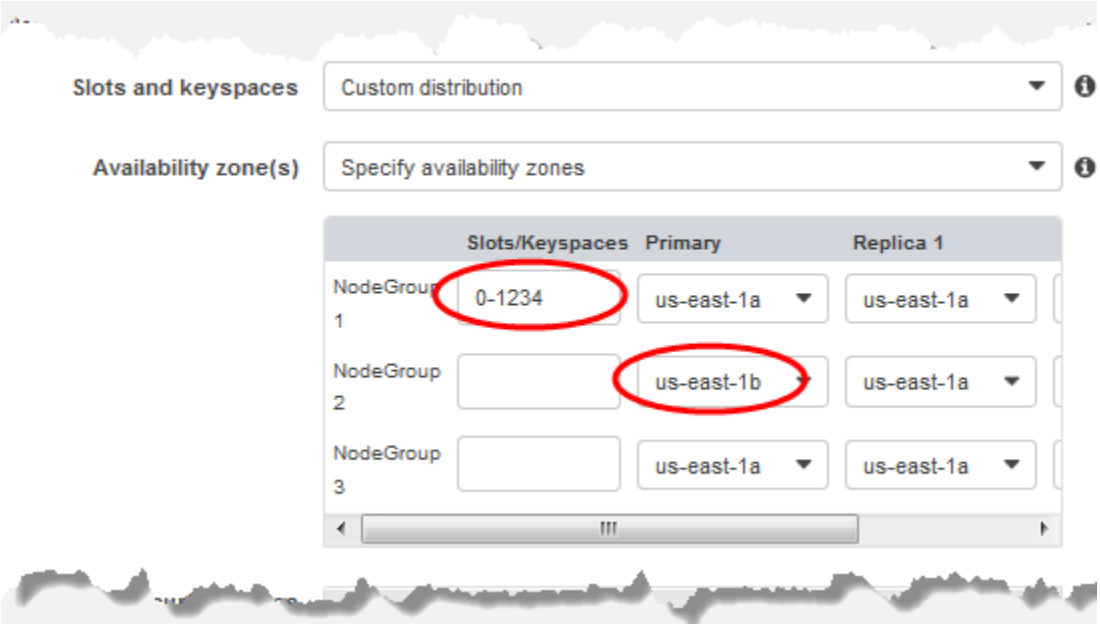
   자세한 내용은 에서 서브넷 생성을 참조하세요VPC.

7. 가용 영역 배치(Availability zone placements)의 경우 다음 두 가지 옵션이 있습니다.

   - 기본 설정 없음 - 가용 영역을 ElastiCache 선택합니다.
   - 가용 영역 지정 – 각 클러스터의 가용 영역을 지정합니다.

     가용 영역을 지정하도록 선택한 경우 샤드에 있는 각 클러스터에 대해 목록에서 가용 영역을 선택합니다.

   자세한 내용은 에 대한 리전 및 가용 영역 선택 ElastiCache 단원을 참조하십시오.



Keyspaces 및 가용 영역 지정

8. 다음(Next)을 선택합니다.
9. 고급 Valkey 및 Redis OSS 설정에서

   - 보안(Security)의 경우

     i. 데이터를 암호화하려면 다음과 같은 옵션이 있습니다.

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

       > **Note**
       >
       > Customer Managed AWS KMS 키를 선택하고 키를 선택하여 다른 암호화 키를 제공할 수 있습니다. 자세한 내용은 에서 AWS 고객 관리형 키 사용을 KMS참조하세요.

     - 전송 중 데이터 암호화 – 전송 데이터 암호화를 활성화합니다. 자세한 정보는 전송 중 데이터 암호화를 참조하세요. Valkey 7.2 이상 및 Redis OSS 엔진 버전 6.0 이상의 경우 전송 중 암호화를 활성화하면 다음 액세스 제어 옵션 중 하나를 지정하라는 메시지가 표시됩니다.
       - 액세스 제어 안 함 – 기본 설정입니다. 이 옵션은 클러스터에 대한 사용자 액세스를 제한하지 않는다는 의미입니다.

- 사용자 그룹 액세스 제어 목록 - 클러스터에 액세스할 수 있는 사용자 집합이 정의된 사용자 그룹을 선택합니다. 자세한 내용은 콘솔 및 를 사용하여 사용자 그룹 관리 CLI 단원을 참조하십시오.
- AUTH 기본 사용자 - Valkey 또는 Redis OSS 서버의 인증 메커니즘입니다. 자세한 내용은 AUTH를 참조하세요.

- AUTH – Valkey 또는 Redis OSS 서버의 인증 메커니즘입니다. 자세한 내용은 AUTH를 참조하세요.

> **Note**
>
> OSS 버전 3.2.10을 제외한 3.2.6 이상 Redis 버전의 경우 유일한 옵션AUTH입니다.

ii. 보안 그룹에서 이 클러스터에 사용할 보안 그룹을 선택합니다. 보안 그룹은 클러스터에 대한 네트워크 액세스를 제어하는 방화벽 역할을 합니다. 에 기본 보안 그룹을 사용하거나 새 보안 그룹을 VPC 생성할 수 있습니다.

보안 그룹에 대한 자세한 내용은 Amazon 사용 설명서의 에 대한 보안 그룹을 VPC 참조하세요. VPC

10. 정기적인 자동 백업을 예약할 경우 Enable automatic backups(자동 백업 활성화)를 선택한 후 자동으로 삭제되기 전에 각 자동 백업을 보존할 기간(일)을 입력합니다. 정기적인 자동 백업을 예약하지 않으려면 [Enable automatic backups] 확인란의 선택을 취소합니다. 어떤 경우든 수동 백업을 항상 생성할 수 있습니다.

    백업 및 복원에 대한 자세한 내용은 섹션을 참조하세요스냅샷 및 복원.

11. (선택 사항) 유지 관리 기간을 지정합니다. 유지 관리 기간은 클러스터에 대한 시스템 유지 관리를 ElastiCache 예약하는 매주 일반적으로 1시간의 시간입니다. ElastiCache 에서 유지 관리 기간의 날짜 및 시간을 선택하거나(기본 설정 없음) 직접 날짜, 시간 및 기간을 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 ElastiCache 클러스터 유지 관리 단원을 참조하십시오.

12. (선택 사항) 로그의 경우:

    - 로그 형식에서 텍스트 또는 를 선택합니다JSON.
    - 대상 유형에서 CloudWatch 로그 또는 Kinesis Firehose를 선택합니다.

- 로그 대상 에서 새로 만들기를 선택하고 CloudWatch 로그 로그 그룹 이름 또는 Firehose 스트림 이름을 입력하거나 기존 선택을 선택한 다음 CloudWatch 로그 로그 로그 그룹 이름 또는 Firehose 스트림 이름을 선택합니다.

13. 태그 의 경우 클러스터 및 기타 ElastiCache 리소스를 관리하는 데 도움이 되도록 태그 형식으로 각 리소스에 자체 메타데이터를 할당할 수 있습니다. 자세한 정보는 ElastiCache 리소스 태그 지정 섹션을 참조하세요.
14. 입력 및 선택한 내용을 모두 검토한 다음 필요한 내용을 수정합니다. 준비가 되면 다음을 선택합니다.
15. 이전 단계에서 클러스터를 구성한 후 이제 보조 클러스터 세부 정보를 구성합니다.
16. 리전 클러스터 에서 클러스터가 위치한 AWS 리전을 선택합니다.
17. 클러스터 정보(Cluster info)에 클러스터의 이름과 설명(선택 사항)을 입력합니다.
18. 다음 옵션은 기본 클러스터 구성과 일치하도록 미리 채워지며 변경할 수 없습니다.

    - 위치
    - 엔진 버전
    - 인스턴스 유형
    - 노드 유형
    - 샤드 수
    - Parameter Group

    > **Note**
    >
    > ElastiCache 는 제공된 파라미터 그룹의 값에서 새 파라미터 그룹을 자동 생성하고 클러스터에 새 파라미터 그룹을 적용합니다. 글로벌 데이터 스토어의 파라미터를 수정하려면 이 새 파라미터 그룹을 사용합니다. 자동 생성된 각 파라미터 그룹은 하나의 클러스터에만 연결되므로 하나의 글로벌 데이터 스토어에만 연결됩니다.

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

    > **Note**
    >
    > Customer Managed AWS KMS 키를 선택하고 키를 선택하여 다른 암호화 키를 제공할 수 있습니다. 자세한 내용은 Customer Managed AWS KMS 키 사용을 참조하세요.

- 전송 중 데이터 암호화 – 전송 데이터 암호화를 활성화합니다. 자세한 내용은 전송 중 데이터 암호화를 참조하세요. Valkey 7.2 이상 및 Redis OSS 엔진 버전 6.4 이상의 경우 전송 중 암호화를 활성화하면 다음 액세스 제어 옵션 중 하나를 지정하라는 메시지가 표시됩니다.
  - 액세스 제어 안 함 – 기본 설정입니다. 이 옵션은 클러스터에 대한 사용자 액세스를 제한하지 않는다는 의미입니다.
  - 사용자 그룹 액세스 제어 목록 - 클러스터에 액세스할 수 있는 사용자 집합이 정의된 사용자 그룹을 선택합니다. 자세한 내용은 콘솔 및 를 사용하여 사용자 그룹 관리 CLI 단원을 참조하십시오.
  - AUTH 기본 사용자 - Valkey 또는 Redis OSS 서버의 인증 메커니즘입니다. 자세한 내용은 AUTH를 참조하세요.

> **Note**
>
> 4.0.2 사이의 Redis OSS 버전에서는 전송 중 암호화가 처음 지원되고 6.0.4AUTH가 유일한 옵션입니다.

나머지 보조 클러스터 설정은 기본 클러스터와 동일한 값으로 미리 채워지지만, 다음은 해당 클러스터에 대한 특정 요구 사항을 충족하도록 업데이트할 수 있습니다.

- Port
- 복제본 개수
- 서브넷 그룹
- 기본 가용 영역
- 보안 그룹
- 고객 관리형(AWS KMS 키)
- AUTH 토큰
- 자동 백업 활성화
- 백업 보관 기간
- 백업 기간
- 유지보수 윈도우
- SNS 알림 주제

19. 생성(Create)을 선택합니다. 이렇게 하면 글로벌 데이터 스토어의 상태가 생성 중으로 설정됩니다. 기본 클러스터 및 보조 클러스터가 글로벌 데이터 스토어와 연결되면 상태가 사용 가능으로 변경됩니다. 읽기 및 쓰기를 허용하는 기본 클러스터와 기본 클러스터에서 복제된 읽기를 허용하는 보조 클러스터가 있습니다.

    또한 클러스터가 다음을 포함하여 글로벌 데이터 스토어의 일부인지 여부를 나타내도록 페이지가 업데이트됩니다.

    - 글로벌 데이터 스토어 - 클러스터가 속한 글로벌 데이터 스토어의 이름입니다.
    - 글로벌 데이터 스토어 역할 - 클러스터의 역할(기본 또는 보조)입니다.

다른 AWS 리전에 최대 1개의 보조 클러스터를 추가할 수 있습니다. 자세한 내용은 글로벌 데이터 스토어에 리전 추가 단원을 참조하십시오.

글로벌 데이터 스토어 세부 정보 보기

기존 글로벌 데이터 스토어의 세부 정보를 보고 글로벌 데이터 스토어 페이지에서 수정할 수도 있습니다.

글로벌 데이터 스토어 세부 정보를 보려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 글로벌 데이터 스토어를 선택한 다음 사용 가능한 글로벌 데이터 스토어를 선택합니다.

그러면 다음 글로벌 데이터 스토어 속성을 검사할 수 있습니다.

- 글로벌 데이터 스토어 이름: 글로벌 데이터 스토어의 이름입니다.
- 설명: 글로벌 데이터 스토어에 대한 설명입니다.
- 상태: 옵션은 다음과 같습니다.
    - [생성 중]
    - 수정 중
    - 사용 가능
    - [삭제 중]
    - 기본 전용 - 이 상태는 글로벌 데이터 스토어에 기본 클러스터만 포함되어 있음을 나타냅니다. 모든 보조 클러스터가 삭제되거나 성공적으로 생성되지 않습니다.

- 클러스터 모드: 활성화되거나 비활성화됩니다.
- 엔진 버전: 글로벌 데이터 스토어를 실행하는 Valkey 또는 Redis OSS 엔진 버전
- 인스턴스 노드 유형: 글로벌 데이터 스토어에 사용되는 노드 유형입니다.
- 미사용 데이터 암호화: 활성화되거나 비활성화됩니다.
- 전송 중 데이터 암호화: 활성화되거나 비활성화됩니다.
- AUTH: 활성화 또는 비활성화

글로벌 데이터 스토어를 다음과 같이 변경할 수 있습니다.

- 글로벌 데이터 스토어에 리전 추가
- 글로벌 데이터 스토어에서 리전 제거
- 보조 클러스터를 기본 클러스터로 승격
- 글로벌 데이터 스토어 수정

글로벌 데이터 스토어 페이지에는 글로벌 데이터 스토어를 구성하는 개별 클러스터와 각각에 해당하는 다음 속성도 나열됩니다.

- 리전 - 클러스터가 저장된 AWS 리전
- 역할 - 기본 또는 보조입니다.
- 클러스터 이름 - 클러스터의 이름입니다.
- 상태 - 옵션은 다음과 같습니다.
  - 연결 중 - 클러스터가 글로벌 데이터 스토어에 연결되는 중입니다.
  - 연관됨 - 클러스터가 글로벌 데이터 스토어에 연결되어 있습니다.
  - 연결 해제 중 - 글로벌 데이터 스토어 이름을 사용하여 글로벌 데이터 스토어에서 보조 클러스터를 제거하는 중입니다. 이후 보조 클러스터는 더 이상 기본 클러스터로부터 업데이트를 수신하지 않지만 해당 AWS 리전에서 독립 실행형 클러스터로 유지됩니다.
  - 연결 해제됨 - 보조 클러스터가 글로벌 데이터 스토어에서 제거되었으며 이제 AWS 리전에서 독립 실행형 클러스터가 되었습니다.
- 글로벌 데이터 스토어 복제본 지연 - 글로벌 데이터 스토어의 보조 AWS 리전당 하나의 값을 표시합니다. 보조 리전의 기본 노드와 기본 리전의 기본 노드 간의 지연입니다. 클러스터 모드가 활성화된 Valkey 또는 Redis OSS의 경우 지연은 샤드 간의 최대 지연 시간을 초 단위로 나타냅니다.

글로벌 데이터 스토어에 리전 추가

기존 글로벌 데이터 스토어에 최대 1개의 AWS 리전을 추가할 수 있습니다. 이 시나리오에서는 기본 클러스터로부터 자동 및 비동기 업데이트를 수신하는 별도의 AWS 리전에 읽기 전용 클러스터를 생성합니다.

글로벌 데이터 스토어에 AWS 리전을 추가하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 글로벌 데이터 스토어 를 선택한 다음 기존 글로벌 데이터 스토어를 선택합니다.
3. 리전 클러스터 추가를 선택하고 보조 클러스터가 상주할 AWS 리전을 선택합니다.
4. 클러스터 정보 에서 이름에 값을 입력하고 클러스터에 대한 설명에 선택적으로 값을 입력합니다.
5. 다음 옵션을 그대로 유지하세요. 기본 클러스터 구성과 일치하도록 미리 채워져 있으므로 변경할 수 없습니다.

   - 엔진 버전
   - 인스턴스 유형
   - 노드 유형
   - 샤드 수
   - Parameter Group

     > **Note**
     >
     > ElastiCache 는 제공된 파라미터 그룹의 값에서 새 파라미터 그룹을 자동 생성하고 클러스터에 새 파라미터 그룹을 적용합니다. 글로벌 데이터 스토어의 파라미터를 수정하려면 이 새 파라미터 그룹을 사용합니다. 자동 생성된 각 파라미터 그룹은 하나의 클러스터에만 연결되므로 하나의 글로벌 데이터 스토어에만 연결됩니다.

   - 저장 중 암호화

     > **Note**
     >
     > Customer Managed AWS KMS 키를 선택하고 키를 선택하여 다른 암호화 키를 제공할 수 있습니다.

   - 전송 중 암호화

- AUTH

6. (선택 사항) 나머지 보조 클러스터 설정을 업데이트합니다. 기본 클러스터와 동일한 값으로 미리 채워지지만 해당 클러스터에 대한 특정 요구 사항을 충족하도록 업데이트할 수 있습니다.

   - Port
   - 복제본 개수
   - 서브넷 그룹
   - 기본 가용 영역
   - 보안 그룹
   - 고객 관리 AWS KMS형 키)
   - AUTH 토큰
   - 자동 백업 활성화
   - 백업 보관 기간
   - 백업 기간
   - 유지보수 윈도우
   - SNS 알림 주제

7. 추가를 선택합니다.

글로벌 데이터 스토어 수정

리전 클러스터의 속성을 수정할 수 있습니다. 보조 클러스터를 기본 클러스터로 승격하는 경우를 제외하고 글로벌 데이터 스토어에서는 하나의 수정 작업만 진행 중일 수 있습니다. 자세한 내용은 보조 클러스터를 기본 클러스터로 승격 단원을 참조하십시오.

글로벌 데이터 스토어를 수정하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 글로벌 데이터 스토어 를 선택한 다음 글로벌 데이터 스토어 이름 에서 글로벌 데이터 스토어를 선택합니다.
3. 수정을 선택하고 다음 옵션 중에서 선택합니다.

   - 설명 수정 - 글로벌 데이터 스토어에 대한 설명을 업데이트합니다.
   - 엔진 버전 수정 - Valkey 7.2 이상 또는 Redis OSS 엔진 버전 5.0.6 이상만 사용할 수 있습니다.

- 노드 유형 수정 - 리전 클러스터를 수직(확장 및 축소) 및 수평(확장 및 축소)으로 확장합니다. 옵션에는 R5 및 M5 노드 패밀리가 포함됩니다. 노드 유형에 대한 자세한 내용은 지원되는 노드 유형 섹션을 참조하세요.
- 자동 장애 조치 수정 - 자동 장애 조치를 활성화하거나 비활성화합니다. 리전 클러스터에서 장애 조치 및 기본 노드를 활성화하면 가 예기치 않게 종료되어 리전 복제본 중 하나로 ElastiCache 장애 조치가 수행됩니다. 자세한 내용은 자동 장애 조치를 참조하세요.

OSS 클러스터 모드가 활성화된 Valkey 또는 Redis 클러스터의 경우:

- 샤드 추가 - 추가할 샤드 수를 입력하고 선택적으로 하나 이상의 가용 영역을 지정합니다.
- 샤드 삭제 - 각 AWS 리전에서 삭제할 샤드를 선택합니다.
- 샤드 재분배 - 슬롯 분포를 재분배하여 클러스터의 기존 샤드 간에 균일한 분포를 보장합니다.

글로벌 데이터 스토어의 파라미터를 수정하려면 글로벌 데이터 스토어에 대한 멤버 클러스터의 파라미터 그룹을 수정합니다. 는 이 변경 사항을 해당 글로벌 데이터 스토어 내의 모든 클러스터에 자동으로 ElastiCache 적용합니다. 해당 클러스터의 파라미터 그룹을 수정하려면 Valkey 또는 Redis OSS 콘솔 또는 ModifyCacheCluster API 작업을 사용합니다. 자세한 내용은 ElastiCache 파라미터 그룹 수정 단원을 참조하십시오. 글로벌 데이터 스토어 내에 포함된 클러스터의 파라미터 그룹을 수정하면 해당 글로벌 데이터 스토어 내의 모든 클러스터에 적용됩니다.

전체 파라미터 그룹 또는 특정 파라미터를 재설정하려면 ResetCacheParameterGroup API 작업을 사용합니다.

보조 클러스터를 기본 클러스터로 승격

기본 클러스터 또는 AWS 리전을 사용할 수 없게 되거나 성능 문제가 발생하는 경우 보조 클러스터를 기본 클러스터로 승격할 수 있습니다. 다른 수정이 진행 중이더라도 언제든지 승격이 허용됩니다. 또한 여러 승력을 병렬로 실행할 수 있으며 글로벌 데이터 스토어가 최종적으로 하나의 기본 클러스터가 됩니다. 여러 보조 클러스터를 동시에 승격하는 경우 ElastiCache Valkey 또는 RedisOSS는 어떤 클러스터가 최종적으로 기본 클러스터로 확인되는지 보장하지 않습니다.

보조 클러스터를 기본 클러스터로 승격하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 글로벌 데이터 스토어 를 선택합니다.
3. 세부 정보를 보려면 글로벌 데이터 스토어 이름을 선택합니다.

4. 보조 클러스터를 선택합니다.
5. 기본 클러스터로 승격을 선택합니다.

   그러면 `Promoting a region to primary will make the cluster in this region as read/writable. Are you sure you want to promote the secondary cluster to primary?` 경고와 함께 결정을 확인하라는 메시지가 표시됩니다.

   `The current primary cluster in primary region will become secondary and will stop accepting writes after this operation completes. Please ensure you update your application stack to direct traffic to the new primary region.`
6. 승력을 계속하려면 확인을 선택하고 그렇지 않으면 취소를 선택합니다.

확인하려면 글로벌 데이터 스토어가 수정 중 상태로 전환되어 승격이 완료될 때까지 사용할 수 없습니다.

글로벌 데이터 스토어에서 리전 제거

다음 절차를 사용하여 글로벌 데이터 스토어에서 AWS 리전을 제거할 수 있습니다.

글로벌 데이터 스토어에서 AWS 리전을 제거하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 글로벌 데이터 스토어 를 선택합니다.
3. 글로벌 데이터 스토어를 선택합니다.
4. 제거할 리전을 선택합니다.
5. 리전 제거를 선택합니다.

> **Note**
>
> 이 옵션은 보조 클러스터에만 사용할 수 있습니다.

그러면 `Removing the region will remove your only available cross region replica for the primary cluster. Your primary cluster will no longer be set up for disaster recovery and improved read latency in remote`

region. Are you sure you want to remove the selected region from the global datastore? 경고와 함께 결정을 확인하라는 메시지가 표시됩니다.

6. 승력을 계속하려면 확인을 선택하고 그렇지 않으면 취소를 선택합니다.

확인을 선택하면 AWS 리전이 제거되고 보조 클러스터가 더 이상 복제 업데이트를 수신하지 않습니다.

글로벌 데이터 스토어 삭제

글로벌 데이터 스토어를 삭제하려면 먼저 모든 보조 클러스터를 제거합니다. 자세한 내용은 글로벌 데이터 스토어에서 리전 제거 단원을 참조하십시오. 이렇게 하면 글로벌 데이터 스토어가 기본 전용 상태로 유지됩니다.

글로벌 데이터 스토어를 삭제하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 글로벌 데이터 스토어 를 선택합니다.
3. 글로벌 데이터 스토어 이름에서 삭제할 글로벌 데이터 스토어를 선택한 다음 삭제를 선택합니다.

   그러면 Are you sure you want to delete this Global Datastore? 경고와 함께 결정을 확인하라는 메시지가 표시됩니다.
4. Delete(삭제)를 선택합니다.

글로벌 데이터 스토어가 삭제 중 상태로 전환됩니다.

글로벌 데이터 스토어 사용(CLI)

AWS Command Line Interface (AWS CLI)를 사용하여 명령줄에서 여러 AWS 서비스를 제어하고 스크립트를 통해 자동화할 수 있습니다. 임시(일회성) 작업에 를 AWS CLI 사용할 수 있습니다.

다운로드 및 구성 AWS CLI

는 Windows, macOS 또는 Linux에서 AWS CLI 실행됩니다. 다음 절차에 따라 다운로드 및 구성합니다.

를 다운로드, 설치 및 구성하려면 CLI

1. AWS 명령줄 인터페이스 웹 페이지에서 AWS CLI를 다운로드합니다.

2. CLI AWS Command Line Interface 사용 설명서의 설치 AWS CLI 및 구성 AWS 지침을 따릅니다.

글로벌 데이터 스토어에서 AWS CLI 사용

글로벌 데이터 스토어에서 작업하려면 다음 CLI 작업을 사용합니다.

- create-global-replication-group

```
aws elasticache create-global-replication-group \
   --global-replication-group-id-suffix my global datastore  \
   --primary-replication-group-id sample-repl-group  \
   --global-replication-group-description an optional description of the global
 datastore
```

Amazon은 글로벌 데이터 스토어 ID가 생성될 때 자동으로 접두사를 ElastiCache 적용합니다. 각 AWS 리전에는 고유한 접두사가 있습니다. 예를 들어 미국 서부(캘리포니아 북부) 리전에서 생성된 글로벌 데이터 스토어 ID는 사용자가 제공한 접미사 이름과 함께 'virxk'로 시작합니다. 접미사는 자동으로 생성된 접두사와 결합되어 여러 리전에서 글로벌 데이터 스토어 이름의 고유성을 보장합니다.

다음 표에는 각 AWS 리전과 해당 글로벌 데이터 스토어 ID 접두사가 나열되어 있습니다.

리전 이름/리전	접두사
US East (Ohio) Region `us-east-2`	`fpkhr`
미국 동부(버지니아 북부) 리전 `us-east-1`	`ldgnf`
미국 서부(캘리포니아 북부) 리전 `us-west-1`	`virxk`
미국 서부(오레곤) 리전 `us-west-2`	`sgaui`

리전 이름/리전	접두사
캐나다(중부) 리전 `ca-central-1`	`bxodz`
Asia Pacific (Mumbai) Region `ap-south-1`	`erpgt`
아시아 태평양(도쿄) 리전 `ap-northeast-1`	`quwsw`
Asia Pacific (Seoul) Region `ap-northeast-2`	`lfqnh`
Asia Pacific (Osaka) Region `ap-northeast-3`	`nlapn`
아시아 태평양(싱가포르) 리전 `ap-southeast-1`	`vlqxn`
아시아 태평양(시드니) 리전 `ap-southeast-2`	`vbgxd`
Europe (Frankfurt) Region `eu-central-1`	`iudkw`
Europe (Ireland) Region `eu-west-1`	`gxeiz`
Europe (London) Region `eu-west-2`	`okuqm`

리전 이름/리전	접두사
EU(파리) 리전 `eu-west-3`	`fgjhi`
South America (São Paulo) Region `sa-east-1`	`juxlw`
중국(베이징) 리전 `cn-north-1`	`emvgo`
중국(닝샤) 리전 `cn-northwest-1`	`ckbem`
Asia Pacific (Hong Kong) Region `ap-east-1`	`knjmp`
AWS GovCloud (미국 서부) `us-gov-west-1`	`sgwui`

- create-replication-group - 이 작업을 사용하여 글로벌 데이터 스토어의 이름을 `--global-replication-group-id` 파라미터에 제공하여 글로벌 데이터 스토어에 대한 보조 클러스터를 생성합니다.

```
aws elasticache create-replication-group \
  --replication-group-id secondary replication group name \
  --replication-group-description “Replication group description" \
  --global-replication-group-id global datastore name
```

이 작업을 호출하고 `--global-replication-group-id` 값을 전달할 때 ElastiCache 는 다음 파라미터에 대해 전역 복제 그룹의 기본 복제 그룹에서 값을 추론합니다. 다음 파라미터에 대한 값을 전달하지 마십시오.

`"PrimaryClusterId",`

```
  "AutomaticFailoverEnabled",

  "NumNodeGroups",

  "CacheParameterGroupName",

  "CacheNodeType",

  "Engine",

  "EngineVersion",

  "CacheSecurityGroupNames",

  "EnableTransitEncryption",

  "AtRestEncryptionEnabled",

  "SnapshotArns",

  "SnapshotName"
```

- describe-global-replication-groups

```
aws elasticache describe-global-replication-groups \
   --global-replication-group-id my global datastore  \
   --show-member-info an optional parameter that returns a list of the primary and
 secondary clusters that make up the global datastore
```

- modify-global-replication-group

```
aws elasticache modify-global-replication-group \
   --global-replication-group-id my global datastore  \
   --automatic-failover-enabled \
   --cache-node-type node type \
   --cache-parameter-group-name parameter group name \
   --engine-version engine version \
   --–apply-immediately \
   --global-replication-group-description description
```

에 대한 OSS Valkey 교차 엔진 업그레이드로 재분배 ElastiCache GlobalDataStore

콘솔 API 또는 를 사용하여 기존 Redis OSS 글로벌 복제 그룹을 Valkey 엔진으로 업그레이드할 수 있습니다CLI.

기존 Redis OSS 글로벌 복제 그룹이 있는 경우 를 사용하여 새 엔진 및 엔진 버전을 지정하여 Valkey로 modify-global-replication-group 업그레이드할 수 있습니다API.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-global-replication-group \
   --global-replication-group-id myGlobalReplGroup \
   --engine valkey \
   --apply-immediately \
   --engine-version 7.2
```

Windows의 경우:

```
aws elasticache modify-global-replication-group ^
   --global-replication-group-id myGlobalReplGroup ^
   --engine valkey ^
   --apply-immediately ^
   --engine-version 7.2
```

업그레이드하려는 기존 Redis OSS 글로벌 복제 그룹에 사용자 지정 캐시 파라미터 그룹이 적용된 경우 요청에서도 사용자 지정 Valkey 캐시 파라미터 그룹을 전달해야 합니다. 입력 Valkey 사용자 지정 파라미터 그룹은 기존 Redis 사용자 OSS 지정 파라미터 그룹과 동일한 Redis OSS 정적 파라미터 값을 가져야 합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-global-replication-group \
   --global-replication-group-id myGlobalReplGroup \
   --engine valkey \
   --engine-version 7.2 \
   --apply-immediately \
   --cache-parameter-group-name myParamGroup
```

Windows의 경우:

```
aws elasticache modify-global-replication-group ^
```

```
   --global-replication-group-id myGlobalReplGroup ^
   --engine valkey ^
   --engine-version 7.2 ^
   --apply-immediately ^
   --cache-parameter-group-name myParamGroup
```

- delete-global-replication-group

```
aws elasticache delete-global-replication-group \
   --global-replication-group-id my global datastore  \
   --retain-primary-replication-group defaults to true
```

- disassociate-global-replication-group

```
aws elasticache disassociate-global-replication-group \
   --global-replication-group-id my global datastore  \
   --replication-group-id my secondary cluster  \
   --replication-group-region the AWS Region in which the secondary cluster resides
```

- failover-global-replication-group

```
aws elasticache failover-replication-group \
   --global-replication-group-id my global datastore \
   --primary-region The AWS Region of the primary cluster \
   --primary-replication-group-id  The name of the global datastore, including the
 suffix.
```

- increase-node-groups-in-global-replication-group

```
aws elasticache increase-node-groups-in-global-replication-group \
   --apply-immediately yes \
   --global-replication-group-id global-replication-group-name \
   --node-group-count 3
```

- decrease-node-groups-in-global-replication-group

```
aws elasticache decrease-node-groups-in-global-replication-group \
   --apply-immediately yes \
   --global-replication-group-id global-replication-group-name \
   --node-group-count 3
```

- rebalance-shards-in-global-복제-그룹

```
aws elasticache rebalance-shards-in-global-replication-group \
   --apply-immediately yes \
   --global-replication-group-id global-replication-group-name
```

도움말을 사용하여 Valkey 또는 Redis 에서 에 사용할 수 ElastiCache 있는 모든 명령을 나열합니다 OSS.

```
aws elasticache help
```

도움말을 사용하면 특정 명령을 설명하고 그 사용법에 대해 자세히 알아볼 수도 있습니다.

```
aws elasticache create-global-replication-group help
```

고가용성을 위한 복제 그룹 사용

단일 노드 Amazon ElastiCache Valkey 및 Redis OSS 클러스터는 데이터 보호 서비스가 제한된 메모리 내 엔터티입니다(AOF). 어떤 이유로든 클러스터에 장애가 발생하면 클러스터의 모든 데이터가 손실됩니다. 하지만 Valkey 또는 Rediss OSS 엔진을 실행하는 경우 2~6개의 노드를 복제본이 있는 클러스터로 그룹화할 수 있습니다. 여기서 1~5개의 읽기 전용 노드에는 그룹의 단일 읽기/쓰기 기본 노드의 복제 데이터가 포함됩니다. 이 시나리오에서는 어떤 이유로든 한 노드에 장애가 발생해도 데이터가 모두 손실되지는 않습니다. 왜냐하면 한 노드가 하나 이상의 다른 노드에 복제되어 있기 때문입니다. 복제 지연 시간으로 인해 기본 읽기/쓰기 노드가 실패할 경우 일부 데이터가 손실될 수 있습니다.

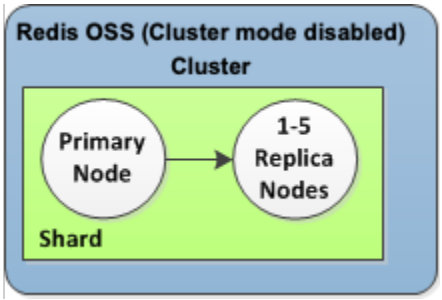
다음 그래픽에서 볼 수 있듯이 복제 구조는 Valkey 또는 Redis OSS 클러스터 내에 포함된 샤드(API/의 노드 그룹이라고 함CLI) 내에 포함됩니다. Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터에는 항상 샤드가 하나 있습니다. Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터에는 클러스터의 데이터가 샤드에 분할되어 있는 최대 500개의 샤드가 있을 수 있습니다. 하나의 클러스터당 최대 90개의 노드로 구성된 더 많은 수의 샤드와 더 적은 수의 복제본을 가진 클러스터를 생성할 수 있습니다. 이 클러스터 구성은 90개의 샤드 및 0개의 복제본부터 15개의 샤드 및 5개의 복제본까지 해당될 수 있으며, 이는 허용되는 최대 복제본 수입니다.

노드 또는 샤드 제한은 Valkey 및 Redis OSS 엔진 버전 5.0.6 이상에서 클러스터당 최대 500개까지 늘릴 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. 일반적인 위험에는 서브넷 그룹의 서브넷 CIDR 범위가 너무 작거나

서브넷이 공유되어 다른 클러스터에서 많이 사용되는 경우가 포함됩니다. 자세한 내용은 서브넷 그룹 생성 단원을 참조하십시오.

5.0.6 이하의 버전에서 한도는 클러스터당 250개입니다.

한도 증가를 요청하려면 AWS 서비스 한도를 참조하고 한도 유형을 인스턴스 유형별 클러스터당 노드로 선택하세요.



Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터에 샤드가 하나 있고 복제본 노드가 0~5개 있음

복제본이 있는 클러스터에 다중 AZ가 활성화되어 있고 기본 노드에 장애가 발생하면 기본 노드가 읽기 전용 복제본으로 장애 조치됩니다. 복제본 노드의 데이터가 비동기적으로 업데이트되기 때문에 복제본 노드를 업데이트할 때 지연 시간으로 인해 일부 데이터가 손실될 수 있습니다. 자세한 내용은 Valkey 또는 Redis 실행 시 장애 완화 OSS 단원을 참조하십시오.

주제

- Valkey 및 Redis OSS 복제 이해
- 복제: Valkey 및 Redis OSS 클러스터 모드 비활성화됨 vs. 활성화됨
- Valkey 및 Redis와 함께 다중 AZ ElastiCache 를 사용하여 의 가동 중지 시간 최소화 OSS
- 동기화 및 백업 구현 방법
- Valkey 또는 Redis OSS 복제 그룹 생성
- 복제 그룹의 세부 정보 보기
- 복제 그룹 엔드포인트 찾기
- 복제 그룹 수정
- 복제 그룹 삭제
- 복제본 수 변경
- Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹의 읽기 전용 복제본을 기본 복제본으로 승격

Valkey 및 Redis OSS 복제 이해

Redis는 두 가지 방법으로 복제를 OSS 구현합니다.

- 각 노드에 있는 클러스터의 모든 데이터가 포함된 단일 샤드가 있음 - Valkey 또는 RedisOSS(클러스터 모드 비활성화됨)
- 최대 500개의 샤드에 걸쳐 데이터가 분할된 경우 - Valkey 또는 RedisOSS(클러스터 모드 활성화됨)

복제 그룹의 각 샤드에는 읽기/쓰기 기본 노드 하나와 최대 5개의 읽기 전용 복제본 노드가 있습니다. 하나의 클러스터당 최대 90개의 노드로 구성된 더 많은 수의 샤드와 더 적은 수의 복제본을 가진 클러스터를 생성할 수 있습니다. 이 클러스터 구성은 90개의 샤드 및 0개의 복제본부터 15개의 샤드 및 5개의 복제본까지 해당될 수 있으며, 이는 허용되는 최대 복제본 수입니다.

Redis OSS 엔진 버전이 5.0.6 이상인 경우 노드 또는 샤드 제한을 클러스터당 최대 500개로 늘릴 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. 일반적인 위험에는 서브넷 그룹의 서브넷 CIDR 범위가 너무 작거나 서브넷이 공유되어 다른 클러스터에서 많이 사용되는 경우가 포함됩니다. 자세한 내용은 서브넷 그룹 생성 단원을 참조하십시오.

5.0.6 이하의 버전에서 한도는 클러스터당 250개입니다.

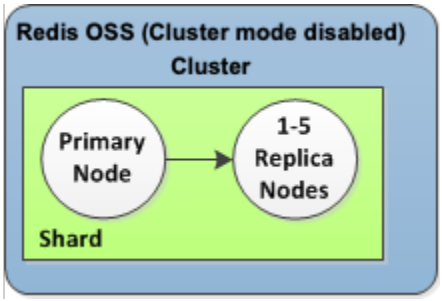
한도 증가를 요청하려면 AWS 서비스 한도를 참조하고 한도 유형을 인스턴스 유형별 클러스터당 노드로 선택하세요.

주제

- Valkey 또는 RedisOSS(클러스터 모드 비활성화됨)
- Valkey 또는 RedisOSS(클러스터 모드 활성화됨)

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨)

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터에는 노드 모음인 단일 샤드가 있으며, 그 안에는 기본 읽기/쓰기 노드 1개와 최대 5개의 보조 읽기 전용 복제본 노드가 있습니다. 각 읽기 전용 복제본은 클러스터의 기본 노드에서 가져온 데이터 사본을 유지합니다. 비동기식 복제 메커니즘은 읽기 전용 복제본이 기본 노드와 동기화되어 있는 상태를 유지하는 데 사용됩니다. 애플리케이션은 클러스터의 모든 노드로부터 읽을 수 있습니다. 애플리케이션은 기본 노드에만 쓸 수 있습니다. 읽기 전용 복제본은 읽기 처리량을 높이고 노드에 장애가 발생할 경우 데이터 손실을 방지합니다.



단일 샤드 및 복제본 노드가 있는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터

복제본 노드가 있는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터를 사용하여 읽기 집약적인 애플리케이션을 처리하거나 동일한 클러스터에서 동시에 읽는 많은 수의 클라이언트를 지원하기 위해 솔루션을 ElastiCache 로 확장할 수 있습니다.

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터의 모든 노드는 동일한 리전에 있어야 합니다.

읽기 전용 복제본을 클러스터에 추가하면 기본 노드에 있는 모든 데이터가 새 노드로 복사됩니다. 이 시점부터 기본 노드에 데이터를 쓸 때마다 변경 사항이 모든 읽기 전용 복제본에 비동기식으로 전파됩니다.

내결함성을 개선하고 쓰기 다운타임을 줄이려면 복제본이 있는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터에 대해 자동 장애 조치로 다중 AZ를 활성화합니다. 자세한 내용은 Valkey 및 Redis와 함께 다중 AZ ElastiCache 를 사용하여 의 가동 중지 시간 최소화 OSS 단원을 참조하십시오.

기본 및 복제본 교환 역할 중 하나를 사용하여 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터 내의 노드 역할을 변경할 수 있습니다. 성능 튜닝을 위해 이런 방식을 선택할 수 있습니다. 예를 들어, 쓰기 작업이 많은 웹 애플리케이션의 경우 네트워크 지연 시간이 가장 짧은 노드를 선택할 수 있습니다. 자세한 내용은 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹의 읽기 전용 복제본을 기본 복제본으로 승격 단원을 참조하십시오.

Valkey 또는 RedisOSS(클러스터 모드 활성화됨)

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터는 1~500개의 샤드(API/CLI: 노드 그룹)로 구성됩니다. 각 샤드에는 읽기/쓰기 기본 노드 및 최대 5개의 읽기 전용 복제본 노드가 있습니다. 이 구성은 90개의 샤드 및 0개의 복제본부터 15개의 샤드 및 5개의 복제본까지 해당될 수 있으며, 이는 허용되는 최대 복제본 수입니다.

엔진 버전이 Valkey 7.2 이상 또는 Redis 5.0.6 이상인 경우 노드 또는 샤드 제한을 클러스터당 최대 OSS 500개로 늘릴 수 있습니다. 예를 들어 83개 샤드(샤드당 기본 1개와 복제본 5개)에서 500개 샤드

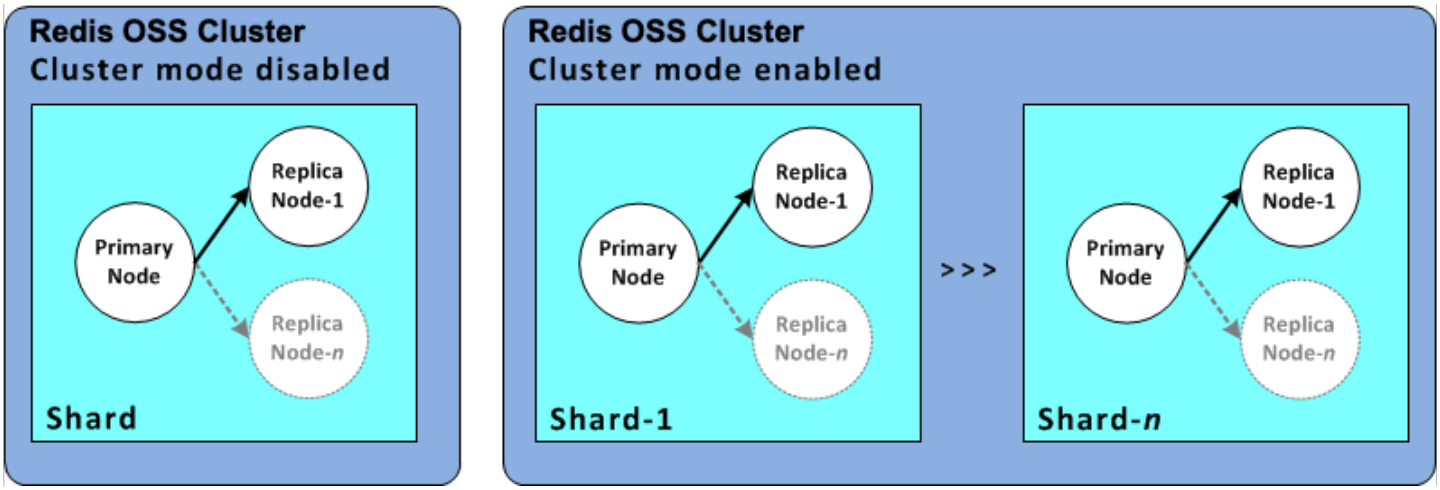
(기본 1개와 복제본 없음) 범위의 500개 노드 클러스터를 구성하도록 선택할 수 있습니다. 증가를 수용할 수 있는 IP 주소가 충분한지 확인해야 합니다. 일반적인 위험에는 서브넷 그룹의 서브넷 CIDR 범위가 너무 작거나 서브넷이 공유되어 다른 클러스터에서 많이 사용되는 경우가 포함됩니다. 자세한 내용은 서브넷 그룹 생성 단원을 참조하십시오.

5.0.6 이하의 버전에서 한도는 클러스터당 250개입니다.

한도 증가를 요청하려면 AWS 서비스 한도를 참조하고 한도 유형을 인스턴스 유형별 클러스터당 노드로 선택하세요.

샤드의 각 읽기 전용 복제본은 샤드의 기본 노드에서 가져온 데이터 사본을 유지합니다. 비동기식 복제 메커니즘은 읽기 전용 복제본이 기본 노드와 동기화되어 있는 상태를 유지하는 데 사용됩니다. 애플리케이션은 클러스터의 모든 노드로부터 읽을 수 있습니다. 애플리케이션은 기본 노드에만 쓸 수 있습니다. 읽기 전용 복제본은 읽기 확장성을 개선하고 데이터 손실을 방지합니다. 데이터는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 샤드를 통해 분할됩니다.

애플리케이션은 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 구성 엔드포인트를 사용하여 클러스터의 노드에 연결합니다. 자세한 내용은 에서 연결 엔드포인트 찾기 ElastiCache 단원을 참조하십시오.



여러 샤드 및 복제본 노드가 있는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터

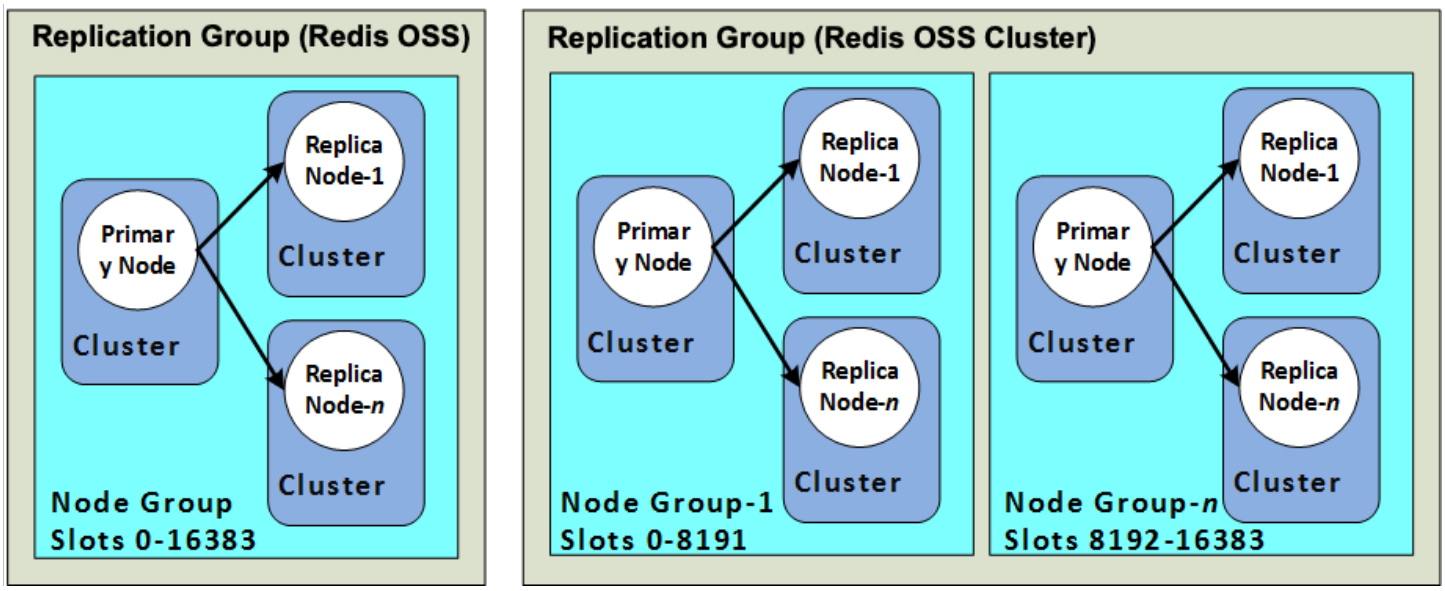
Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 모든 노드는 동일한 리전에 있어야 합니다. 내결함성을 개선하기 위해 해당 리전 내의 여러 가용 영역에서 기본 노드와 읽기 전용 복제본을 모두 프로비저닝할 수 있습니다.

현재 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 기능에는 몇 가지 제한이 있습니다.

- 복제본 노드는 수동으로 기본 노드로 승격할 수 없습니다.

복제: Valkey 및 Redis OSS 클러스터 모드 비활성화됨 vs. 활성화됨

Valkey 7.2 및 Redis OSS 버전 3.2부터는 두 가지 유형의 클러스터(API/CLI: 복제 그룹) 중 하나를 생성할 수 있습니다. Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터에는 항상 최대 5개의 읽기 전용 복제본 노드가 있는 단일 샤드(API/CLI: 노드 그룹)가 있습니다. Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터에는 각각 1~5개의 읽기 전용 복제본 노드가 있는 최대 500개의 샤드가 있습니다.



Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 및 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터

다음 표에는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터와 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 간의 중요한 차이점이 요약되어 있습니다.

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨)와 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 비교

기능	Valkey 또는 RedisOSS(클러스터 모드 비활성화됨)	Valkey 또는 RedisOSS(클러스터 모드 활성화됨)
수정 가능	예. 복제본 노드 추가/삭제 및 노드 유형 확장을 지원합니다.	제한. 자세한 내용은 용 버전 관리 ElastiCache 및 Valkey 또는 Redis에서 클러스터 크기 조정 OSS(클러스터 모드 활성화됨) 섹션을 참조하세요.

기능	Valkey 또는 RedisOSS(클러스터 모드 비활성화됨)	Valkey 또는 RedisOSS(클러스터 모드 활성화됨)
데이터 파티셔닝	아니요	예
샤드	1	1~500
읽기 전용 복제본	0~5 **Important** 복제본이 없으며 노드에 장애가 발생하면 전체 데이터가 손실됩니다.	샤드당 0~5개. **Important** 복제본이 없으며 노드에 장애가 발생하면 전체 데이터가 손실됩니다.
다중 AZ	예, 최소 1개의 복제본이 있어야 합니다. 선택 사항입니다. 기본적으로 활성화되어 있습니다.	예 선택 사항입니다. 기본적으로 활성화되어 있습니다.
스냅샷(백업)	예, 단일 .rdb 파일을 생성합니다.	예, 각 샤드에 고유한 .rdb 파일을 생성합니다.
복원	예, Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터의 단일 .rdb 파일을 사용합니다.	예, Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 또는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 .rdb 파일을 사용합니다.
지원되는 버전	모든 Valkey 및 Redis OSS 버전	모든 Valkey 버전 및 Redis OSS 3.2 이상
엔진 업그레이드 가능 여부	예, 하지만 몇 가지 제약이 있습니다. 자세한 내용은 용 버전 관리 ElastiCache 단원을 참조하십시오.	예, 하지만 몇 가지 제약이 있습니다. 자세한 내용은 용 버전 관리 ElastiCache 단원을 참조하십시오.

기능	Valkey 또는 RedisOSS(클러스터 모드 비활성화됨)	Valkey 또는 RedisOSS(클러스터 모드 활성화됨)
암호화(Encryption)	버전 3.2.6(에 대해 예약됨EOL, Redis OSS 버전 수명 종료 일정 참조) 및 4.0.10 이상.	버전 3.2.6(에 대해 예약됨EOL, Redis OSS 버전 수명 종료 일정 참조) 및 4.0.10 이상.
HIPAA 적격	버전 3.2.6(에 대해 예약됨EOL, Redis OSS 버전 수명 종료 일정 참조) 및 4.0.10 이상.	버전 3.2.6(에 대해 예약됨EOL, Redis OSS 버전 수명 종료 일정 참조) 및 4.0.10 이상.
PCI DSS 규정 준수	버전 3.2.6(에 대해 예약됨EOL, Redis OSS 버전 수명 종료 일정 참조) 및 4.0.10 이상.	버전 3.2.6(에 대해 예약됨EOL, Redis OSS 버전 수명 종료 일정 참조) 및 4.0.10 이상.
온라인 리샤딩	N/A	버전 3.2.10(에 대해 예약됨 EOL, Redis OSS 버전 수명 종료 일정 참조) 이상.

어떤 클러스터를 선택해야 합니까?

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 또는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 중에서 선택할 때는 다음 요인을 고려하세요.

- 조정과 파티셔닝 - 비즈니스 요구는 변화합니다. 최고 수요가 발생할 때를 대비하거나 수요가 변화함에 따라 조정해야 합니다. Valkey 또는 RedisOSS(클러스터 모드 비활성화됨)는 크기 조정을 지원합니다. 복제본 노드를 추가하거나 삭제하여 읽기 용량을 조정하거나 더 큰 노드 유형까지 확장하여 용량을 조정할 수 있습니다. 이 두 작업은 모두 시간이 소요됩니다. 자세한 내용은 섹션을 참조Valkey 또는 Redis용 복제본 노드 크기 조정OSS(클러스터 모드 비활성화됨)하세요.

  Valkey 또는 RedisOSS(클러스터 모드 활성화됨)는 최대 500개의 노드 그룹에서 데이터 파티셔닝을 지원합니다. 비즈니스에 변경이 필요할 때마다 샤드 수를 동적으로 변경할 수 있습니다. 파티셔닝의 이점 중 하나는 로드를 더 많은 엔드포인트로 분산시켜 최고 수요가 발생할 때 액세스 병목 현상을 줄이는 것입니다. 또한 데이터가 여러 서버에 분산될 수 있으므로 더 큰 데이터 세트를 수용할 수 있습니다. 파티션 크기 조정에 대한 자세한 내용은 섹션을 참조하세요Valkey 또는 Redis에서 클러스터 크기 조정OSS(클러스터 모드 활성화됨).

- 노드 크기 대 노드 수 - Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터에는 샤드가 하나뿐이므로 노드 유형은 모든 클러스터의 데이터와 필요한 오버헤드를 수용할 수 있을 만큼 커야 합니다. 반면 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터를 사용할 때 여러 샤드에 데이터를 분할할 수 있으므로 노드 유형은 더 작지만 더 많이 필요할 수 있습니다. 자세한 내용은 노드 크기 선택 단원을 참조하십시오.

- 읽기 대 쓰기 - 클러스터의 기본 로드가 데이터를 읽는 애플리케이션인 경우 읽기 전용 복제본을 추가하고 삭제하여 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터를 확장할 수 있습니다. 그러나 5개 읽기 전용 복제본의 최대치가 있습니다. 클러스터의 로드가 쓰기 사용량이 많은 경우 샤드가 여러 개 있는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 추가 쓰기 엔드포인트를 활용할 수 있습니다.

어떤 유형의 클러스터를 구현하도록 선택하든 현재와 미래의 요구에 적합한 노드 유형을 선택해야 합니다.

Valkey 및 Redis와 함께 다중 AZ ElastiCache 를 사용하여 의 가동 중지 시간 최소화 OSS

Valkey 및 Redis ElastiCache 를 사용하면 기본 노드를 교체해야 할 OSS 수 있는 인스턴스가 여러 개 있습니다. 여기에는 특정 유형의 계획된 유지 관리와 기본 노드 또는 가용 영역 장애의 가능성이 낮은 이벤트가 포함됩니다.

이러한 교체로 인해 클러스터에 약간의 가동 중지가 발생하지만 다중 AZ를 활성화한 경우 가동 중지 시간이 최소화됩니다. 기본 노드의 역할은 자동으로 읽기 전용 복제본 중 하나로 장애 조치됩니다. 는 이를 투명하게 ElastiCache 처리하기 때문에 새 기본 노드를 생성하고 프로비저닝할 필요가 없습니다. 이 장애 조치 및 복제본 승격을 통해 승격이 완료되는 즉시 새 기본 노드에 작성을 재개할 수 있습니다.

ElastiCache 또한 승격된 복제본의 도메인 이름 서비스(DNS) 이름도 전파합니다. 이렇게 하면 애플리케이션이 기본 엔드포인트에 쓰는 경우 애플리케이션에서 엔드포인트를 변경할 필요가 없기 때문입니다. 개별 엔드포인트를 읽을 경우 기본으로 승격된 복제본의 읽기 엔드포인트를 새 복제본의 엔드포인트로 변경해야 합니다.

계획된 노드 교체의 경우, 유지 관리 업데이트 또는 셀프 서비스 업데이트로 인해 시작되었으며 다음 사항에 유의하세요.

- ElastiCache Valkey 및 Redis OSS 클러스터의 경우 클러스터가 수신 쓰기 요청을 처리하는 동안 계획된 노드 교체가 완료됩니다.
- 5.0.6 이상 엔진에서 실행되는 다중 AZ가 활성화된 Valkey 및 Redis OSS 클러스터 모드 비활성화 클러스터의 경우 클러스터가 수신 쓰기 요청을 처리하는 동안 계획된 노드 교체가 완료됩니다.
- 4.0.10 이하 엔진에서 실행되는 다중 AZ가 활성화된 Valkey 및 Redis OSS 클러스터 모드 비활성화 클러스터의 경우 DNS 업데이트와 관련된 짧은 쓰기 중단이 발생할 수 있습니다. 이 중단은 최대 몇 초가 걸릴 수 있습니다. 이 프로세스는 다중 AZ를 활성화하지 않은 경우 발생하는, 새 기본 노드를 다시 생성하고 프로비저닝하는 것보다 훨씬 빠릅니다.

ElastiCache 관리 콘솔, AWS CLI또는 를 사용하여 다중 AZ를 활성화할 수 있습니다 ElastiCache API.

Valkey 또는 Redis OSS 클러스터(API 및 CLI, 복제 그룹)에서 ElastiCache 다중 AZ를 활성화하면 내결함성이 향상됩니다. 특히 클러스터의 읽기/쓰기 기본 클러스터 노드에 접속할 수 없거나 어떤 이유로든 실패하는 경우에 특히 그렇습니다. 다중 AZ는 각 샤드에 둘 이상의 노드가 있는 Valkey 및 Redis OSS 클러스터에서만 지원됩니다.

주제

- 다중 AZ 활성화
- 다중 AZ 응답이 있는 장애 시나리오
- 자동 장애 조치 테스트
- 다중 AZ에 대한 제한 사항

다중 AZ 활성화

ElastiCache 콘솔 또는 를 사용하여 클러스터(API 또는 CLI, 복제 그룹)를 생성 AWS CLI하거나 수정할 때 다중 AZ를 활성화할 ElastiCache 수 있습니다API.

사용 가능한 읽기 전용 복제본이 하나 이상 있는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터에서만 다중 AZ를 활성화할 수 있습니다. 읽기 전용 복제본이 없는 클러스터는 고가용성 또는 내결함성을 제공하지 않습니다. 복제하여 클러스터를 생성에 대한 정보는 Valkey 또는 Redis OSS 복제 그룹 생성을 참조하세요. 복제하여 있는 클러스터에 읽기 전용 복제본 추가에 대한 정보는 Valkey 또는 Redis에 대한 읽기 전용 복제본 추가OSS(클러스터 모드 비활성화됨)를 참조하세요.

주제

- 다중 AZ 활성화(콘솔)
- 다중 AZ 활성화(AWS CLI)
- 다중 AZ 활성화(ElastiCache API)

다중 AZ 활성화(콘솔)

새 Valkey 또는 Redis OSS 클러스터를 생성할 때 ElastiCache 콘솔을 사용하거나 복제를 사용하여 기존 클러스터를 수정하여 다중 AZ를 활성화할 수 있습니다.

다중 AZ는 기본적으로 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터에서 활성화됩니다.

> **Important**
>
> ElastiCache 클러스터에 모든 샤드의 기본 샤드와 다른 가용 영역에 하나 이상의 복제본이 포함된 경우에만 가 다중 AZ를 자동으로 활성화합니다.

ElastiCache 콘솔을 사용하여 클러스터를 생성할 때 다중 AZ 활성화

이 프로세스에 대한 자세한 내용은 Valkey(클러스터 모드 비활성화됨) 클러스터 생성(콘솔)을 참조하세요. 복제본이 하나 이상 있어야 하고 다중 AZ를 활성화해야 합니다.

기존 클러스터에서 다중 AZ 활성화(콘솔)

이 프로세스에 대한 자세한 내용은 클러스터 수정 사용 ElastiCache AWS Management Console섹션을 참조하세요.

다중 AZ 활성화(AWS CLI)

다음 코드 예제에서는 AWS CLI 를 사용하여 복제 그룹 에 대해 다중 AZ를 활성화합니다`redis12`.

> **Important**
>
> 복제 그룹 `redis12`가 이미 존재해야 하며 사용할 수 있는 읽기 전용 복제본이 하나 이상 있어야 합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-replication-group \
    --replication-group-id redis12 \
    --automatic-failover-enabled \
    --multi-az-enabled \
    --apply-immediately
```

Windows의 경우:

```
aws elasticache modify-replication-group ^
    --replication-group-id redis12 ^
    --automatic-failover-enabled ^
    --multi-az-enabled ^
    --apply-immediately
```

이 명령의 JSON 출력은 다음과 같아야 합니다.

```
{
    "ReplicationGroup": {
        "Status": "modifying",
        "Description": "One shard, two nodes",
        "NodeGroups": [
            {
                "Status": "modifying",
                "NodeGroupMembers": [
                    {
```

```
                        "CurrentRole": "primary",
                        "PreferredAvailabilityZone": "us-west-2b",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Port": 6379,
                            "Address":
 "redis12-001.v5r9dc.0001.usw2.cache.amazonaws.com"
                        },
                        "CacheClusterId": "redis12-001"
                    },
                    {
                        "CurrentRole": "replica",
                        "PreferredAvailabilityZone": "us-west-2a",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Port": 6379,
                            "Address":
 "redis12-002.v5r9dc.0001.usw2.cache.amazonaws.com"
                        },
                        "CacheClusterId": "redis12-002"
                    }
                ],
                "NodeGroupId": "0001",
                "PrimaryEndpoint": {
                    "Port": 6379,
                    "Address": "redis12.v5r9dc.ng.0001.usw2.cache.amazonaws.com"
                }
            }
        ],
        "ReplicationGroupId": "redis12",
        "SnapshotRetentionLimit": 1,
        "AutomaticFailover": "enabling",
        "MultiAZ": "enabled",
        "SnapshotWindow": "07:00-08:00",
        "SnapshottingClusterId": "redis12-002",
        "MemberClusters": [
            "redis12-001",
            "redis12-002"
        ],
        "PendingModifiedValues": {}
    }
}
```

자세한 내용은 AWS CLI 명령 참조의 다음 항목을 참조하세요.

- create-cache-cluster
- create-replication-group
- modify-replication-group AWS CLI 명령 참조의 .

다중 AZ 활성화(ElastiCache API)

다음 코드 예제에서는 ElastiCache API를 사용하여 복제 그룹 에 대해 다중 AZ를 활성화합니다.`redis12`

> **Note**
>
> 이 예제를 사용하려면 복제 그룹 `redis12`가 이미 존재해야 하며 사용할 수 있는 읽기 전용 복제본이 하나 이상 있어야 합니다.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyReplicationGroup
    &ApplyImmediately=true
    &AutoFailover=true
    &MultiAZEnabled=true
    &ReplicationGroupId=redis12
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20140401T192317Z
    &X-Amz-Credential=<credential>
```

자세한 내용은 ElastiCache API 참조 의 다음 주제를 참조하세요.

- CreateCacheCluster
- CreateReplicationGroup
- ModifyReplicationGroup

다중 AZ 응답이 있는 장애 시나리오

다중 AZ를 도입하기 전에 는 실패한 노드를 다시 생성하고 다시 프로비저닝하여 클러스터의 실패한 노드를 ElastiCache 감지하고 교체했습니다. 다중 AZ를 활성화하면 장애가 발생한 기본 노드가 복제 지연이 가장 짧은 복제본으로 장애 조치됩니다. 선택한 복제본이 자동으로 승격되기 때문에 새 기본 노드를 생성하고 프로비저닝하는 것보다 훨씬 빠릅니다. 이 프로세스는 보통 클러스터에 다시 작성하려면 몇 초 정도 소요됩니다.

다중 AZ가 활성화되면 는 기본 노드의 상태를 ElastiCache 지속적으로 모니터링합니다. 기본 노드에 장애가 발생하면 장애 유형에 따라 다음 작업 중 하나가 수행됩니다.

주제

- 프라이머리 노드에만 장애가 발생한 경우의 장애 시나리오
- 기본 노드 및 일부 읽기 전용 복제본에 장애가 발생한 경우의 장애 시나리오
- 전체 클러스터에 장애가 발생한 경우의 장애 시나리오

프라이머리 노드에만 장애가 발생한 경우의 장애 시나리오

기본 노드에 장애가 발생하면 복제 지연 시간이 가장 짧은 읽기 전용 복제본을 기본 노드로 승격시킵니다. 그러면 대체 읽기 전용 복제본이 생성되어 장애가 발생한 기본 노드와 동일한 가용 영역에 프로비저닝됩니다.

기본 노드만 실패하면 ElastiCache Multi-AZ는 다음을 수행합니다.

1. 장애가 발생한 기본 노드는 오프라인 상태로 전환됩니다.
2. 복제 지연 시간이 가장 짧은 읽기 전용 복제본을 기본으로 승격시킵니다.

   승격 프로세스가 완료되는 즉시 쓰기를 재개할 수 있으며 일반적으로 몇 초 정도 소요됩니다. 애플리케이션이 기본 엔드포인트에 쓰는 경우 쓰기 또는 읽기에 대한 엔드포인트를 변경할 필요가 없습니다. ElastiCache 는 승격된 복제본의 DNS 이름을 전파합니다.
3. 대체 읽기 전용 복제본을 시작하고 프로비저닝합니다.

   장애가 발생한 기본 노드가 있는 가용 영역에서 대체 읽기 전용 복제본을 시작하여 노드 배포를 유지합니다.
4. 복제본이 새 기본 노드와 동기화됩니다.

새 복제본을 사용할 수 있게 되면 다음 효과에 유의하세요.

- 기본 엔드포인트 - 새 기본 노드의 DNS 이름이 기본 엔드포인트에 전파되므로 애플리케이션을 변경할 필요가 없습니다.
- 읽기 엔드포인트 - 리더 엔드포인트는 새 복제본 노드를 가리키도록 자동으로 업데이트됩니다.

클러스터의 엔드포인트를 찾는 방법에 대한 정보는 다음 항목을 참조하세요.

- Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터의 엔드포인트 찾기(콘솔)
- Valkey 또는 Redis OSS 복제 그룹의 엔드포인트 찾기(AWS CLI)
- Valkey 또는 Redis OSS 복제 그룹의 엔드포인트 찾기(ElastiCache API)

기본 노드 및 일부 읽기 전용 복제본에 장애가 발생한 경우의 장애 시나리오

기본 복제본 및 하나 이상의 복제본에 장애가 발생하면 지연 시간이 가장 짧은 사용 가능한 복제본이 기본 클러스터로 승격됩니다. 또한 기본으로 승격된 복제본 및 장애가 발생한 노드로 새로운 읽기 전용 복제본이 동일 가용 영역에 생성되고 프로비저닝됩니다.

기본 노드와 일부 읽기 전용 복제본이 실패하면 ElastiCache Multi-AZ는 다음을 수행합니다.

1. 장애가 발생한 기본 노드 및 읽기 전용 복제본이 오프라인 상태로 전환됩니다.
2. 복제 지연 시간이 가장 짧은 사용 가능한 복제본을 기본 노드로 승격시킵니다.

   승격 프로세스가 완료되는 즉시 쓰기를 재개할 수 있으며 일반적으로 몇 초 정도 소요됩니다. 애플리케이션이 기본 엔드포인트에 쓰는 경우 writes. ElastiCache propagates에 대한 엔드포인트DNS를 변경할 필요가 없습니다.
3. 교체용 복제본을 생성하고 프로비저닝합니다.

   장애가 발생한 노드의 가용 영역에서 교체용 복제본을 생성하여 노드 배포를 유지합니다.
4. 모든 클러스터가 새 기본 노드와 동기화됩니다.

새 노드를 사용할 수 있게 되면 애플리케이션을 다음과 같이 변경합니다.

- 기본 엔드포인트 - 애플리케이션을 변경하지 마십시오. 새 기본 노드의 DNS 이름이 기본 엔드포인트에 전파됩니다.
- 읽기 엔드포인트 - 읽기 엔드포인트는 새 복제본 노드를 가리키도록 자동으로 업데이트됩니다.

복제 그룹의 엔드포인트를 찾는 방법에 대한 정보는 다음 항목을 참조하세요.

- Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터의 엔드포인트 찾기(콘솔)
- Valkey 또는 Redis OSS 복제 그룹의 엔드포인트 찾기(AWS CLI)
- Valkey 또는 Redis OSS 복제 그룹의 엔드포인트 찾기(ElastiCache API)

전체 클러스터에 장애가 발생한 경우의 장애 시나리오

모든 것에 장애가 발생하면 모든 노드를 동일한 가용 영역에 원본 노드로 재생성하고 프로비저닝합니다.

이 시나리오에서는 클러스터의 모든 노드에 장애가 발생하여 클러스터의 모든 데이터가 손실됩니다. 이는 거의 발생하지 않습니다.

전체 클러스터가 실패하면 ElastiCache Multi-AZ는 다음을 수행합니다.

1. 장애가 발생한 기본 노드 및 읽기 전용 복제본이 오프라인 상태로 전환됩니다.
2. 대체 기본 노드를 생성하고 프로비저닝합니다.
3. 교체용 복제본을 생성하고 프로비저닝합니다.

   장애가 발생한 노드의 가용 영역에서 대체를 생성하여 노드 배포를 유지합니다.

   전체 클러스터에 장애가 발생했으므로 데이터가 손실되고 모든 새 노드가 콜드를 시작합니다.

각각의 교체 노드에는 교체하는 노드와 동일한 엔드포인트가 있기 때문에 애플리케이션에서 엔드포인트를 변경할 필요가 없습니다.

복제 그룹의 엔드포인트를 찾는 방법에 대한 정보는 다음 항목을 참조하세요.

- Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터의 엔드포인트 찾기(콘솔)
- Valkey 또는 Redis OSS 복제 그룹의 엔드포인트 찾기(AWS CLI)
- Valkey 또는 Redis OSS 복제 그룹의 엔드포인트 찾기(ElastiCache API)

내결함성 수준을 높이려면 다른 가용 영역에 기본 노드 및 읽기 전용 복제본을 생성하는 것이 좋습니다.

자동 장애 조치 테스트

자동 장애 조치를 활성화한 후 ElastiCache 콘솔, 및 를 사용하여 테스트 AWS CLI할 수 있습니다 ElastiCache API.

테스트 시 다음 사항에 유의하세요.

- 이 작업을 사용하여 24시간 동안 최대 15개의 샤드(및 에서 ElastiCache API 노드 그룹이라고 함 AWS CLI)에 대해 자동 장애 조치를 테스트할 수 있습니다.
- 다른 클러스터의 샤드(API 및 에서 복제 그룹이라고 함CLI)에서 이 작업을 호출하는 경우 동시에 호출할 수 있습니다.
- 경우에 따라 동일한 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹의 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 Amazon ElastiCache 콘솔, AWS CLI또는 를 사용하여 이벤트를 확인합니다 ElastiCache API. 발생 순서대로 나열되어 있는 아래 목록에서 다음과 같은 자동 장애 조치 관련 이벤트를 찾습니다.

  1. 복제 그룹 메시지: `Test Failover API called for node group <node-group-id>`
  2. 캐시 클러스터 메시지: `Failover from primary node <primary-node-id> to replica node <node-id> completed`
  3. 복제 그룹 메시지: `Failover from primary node <primary-node-id> to replica node <node-id> completed`
  4. 캐시 클러스터 메시지: `Recovering cache nodes <node-id>`
  5. 캐시 클러스터 메시지: `Finished recovery for cache nodes <node-id>`

  자세한 내용은 다음 자료를 참조하세요.

  - ElastiCache 이벤트 보기(출처: ElastiCache 사용 설명서)
  - DescribeEvents ElastiCache API 참조의
  - AWS CLI 명령 참조의 describe-events
- 이는 ElastiCache 장애 조치 시 애플리케이션의 동작을 테스트하기 위해 API 설계되었습니다. 클러스터 문제를 해결하기 위해 장애 조치를 시작하는 운영 도구로 설계되지 않았습니다. 또한 대규모 운영 이벤트와 같은 특정 조건에서 이 를 차단할 AWS 수 있습니다API.

주제

- 를 사용하여 자동 장애 조치 테스트 AWS Management Console

- 를 사용하여 자동 장애 조치 테스트 AWS CLI
- 를 사용하여 자동 장애 조치 테스트 ElastiCache API

를 사용하여 자동 장애 조치 테스트 AWS Management Console

다음 절차에 따라 콘솔로 자동 장애 조치를 테스트합니다.

자동 장애 조치를 테스트하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 Valkey 또는 Redis를 OSS선택합니다.
3. 클러스터 목록에서 테스트하려는 클러스터 왼쪽의 상자를 선택합니다. 이 클러스터에는 읽기 전용 복제본 노드가 하나 이상 있어야 합니다.
4. [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 사용 ElastiCache AWS Management Console 단원을 참조하십시오.

5. Valkey 또는 RedisOSS(클러스터 모드 비활성화됨)에서 클러스터 이름을 선택합니다.

   Valkey 또는 RedisOSS(클러스터 모드 활성화됨)의 경우 다음을 수행합니다.

   a. 클러스터의 이름을 선택합니다.
   b. 샤드 페이지에서 장애 조치를 테스트하려는 샤드(API 및 에서 노드 그룹이라고 함CLI)에 대해 샤드의 이름을 선택합니다.
6. 노드 페이지에서 [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(`Test Failover API called`) 및 완료(`Recovery completed`)를 나타내는 이벤트를 주시합니다.

를 사용하여 자동 장애 조치 테스트 AWS CLI

AWS CLI 작업을 사용하여 다중 AZ 지원 클러스터에서 자동 장애 조치를 테스트할 수 있습니다`test-failover`.

파라미터

- `--replication-group-id` - 필수입니다. 테스트할 복제 그룹(콘솔, 클러스터)입니다.
- `--node-group-id` - 필수입니다. 자동 장애 조치를 테스트할 노드 그룹의 이름입니다. 24시간 동안 최대 15개의 노드 그룹을 테스트할 수 있습니다.

다음 예제에서는 AWS CLI 를 사용하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터`redis00-0003`의 노드 그룹에서 자동 장애 조치를 테스트합니다`redis00`.

Example 자동 장애 조치 테스트

Linux, macOS, Unix의 경우:

```
aws elasticache test-failover \
   --replication-group-id redis00 \
   --node-group-id redis00-0003
```

Windows의 경우:

```
aws elasticache test-failover ^
   --replication-group-id redis00 ^
   --node-group-id redis00-0003
```

이전 명령의 출력은 다음과 같습니다.

```
{
    "ReplicationGroup": {
        "Status": "available",
        "Description": "1 shard, 3 nodes (1 + 2 replicas)",
        "NodeGroups": [
            {
                "Status": "available",
                "NodeGroupMembers": [
                    {
                        "CurrentRole": "primary",
                        "PreferredAvailabilityZone": "us-west-2c",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Port": 6379,
                            "Address":
 "redis1x3-001.7ekv3t.0001.usw2.cache.amazonaws.com"
                        },
                        "CacheClusterId": "redis1x3-001"
                    },
                    {
                        "CurrentRole": "replica",
                        "PreferredAvailabilityZone": "us-west-2a",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Port": 6379,
                            "Address":
 "redis1x3-002.7ekv3t.0001.usw2.cache.amazonaws.com"
                        },
                        "CacheClusterId": "redis1x3-002"
                    },
                    {
                        "CurrentRole": "replica",
                        "PreferredAvailabilityZone": "us-west-2b",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Port": 6379,
                            "Address":
 "redis1x3-003.7ekv3t.0001.usw2.cache.amazonaws.com"
                        },
                        "CacheClusterId": "redis1x3-003"
                    }
                ],
                "NodeGroupId": "0001",
```

```
                "PrimaryEndpoint": {
                    "Port": 6379,
                    "Address": "redis1x3.7ekv3t.ng.0001.usw2.cache.amazonaws.com"
                }
            }
        ],
        "ClusterEnabled": false,
        "ReplicationGroupId": "redis1x3",
        "SnapshotRetentionLimit": 1,
        "AutomaticFailover": "enabled",
        "MultiAZ": "enabled",
        "SnapshotWindow": "11:30-12:30",
        "SnapshottingClusterId": "redis1x3-002",
        "MemberClusters": [
            "redis1x3-001",
            "redis1x3-002",
            "redis1x3-003"
        ],
        "CacheNodeType": "cache.m3.medium",
        "DataTiering": "disabled",
        "PendingModifiedValues": {}
    }
}
```

장애 조치 진행 상황을 추적하려면 작업을 사용합니다 AWS CLI `describe-events`.

자세한 내용은 다음 자료를 참조하세요.

- AWS CLI 명령 참조의 test-failover
- AWS CLI 명령 참조의 describe-events

를 사용하여 자동 장애 조치 테스트 ElastiCache API

작업을 사용하여 다중 AZ로 활성화된 모든 클러스터에서 자동 장애 조치를 테스트할 ElastiCache API 수 있습니다`TestFailover`.

파라미터

- `ReplicationGroupId` - 필수입니다. 테스트할 복제 그룹(콘솔, 클러스터)입니다.

- `NodeGroupId` - 필수입니다. 자동 장애 조치를 테스트할 노드 그룹의 이름입니다. 24시간 동안 최대 15개의 노드 그룹을 테스트할 수 있습니다.

다음 예제에서는 복제 그룹(콘솔, 클러스터에서) `redis00`의 노드 그룹 `redis00-0003`에 대한 자동 장애 조치를 테스트합니다.

Example 자동 장애 조치 테스트

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=TestFailover
    &NodeGroupId=redis00-0003
    &ReplicationGroupId=redis00
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20140401T192317Z
    &X-Amz-Credential=<credential>
```

장애 조치 진행 상황을 추적하려면 API 작업을 사용합니다 ElastiCache `DescribeEvents`.

자세한 내용은 다음 자료를 참조하세요.

- TestFailover ElastiCache API 참조의
- DescribeEvents ElastiCache API 참조의

다중 AZ에 대한 제한 사항

다중 AZ에 대한 다음 제한 사항에 유의하세요.

- 다중 AZ는 Valkey 및 Redis OSS 버전 2.8.6 이상에서 지원됩니다.
- 다중 AZ는 T1 노드 유형에서 지원되지 않습니다.
- Valkey 및 Redis OSS 복제는 비동기식입니다. 따라서 기본 노드를 복제본으로 장애 조치하면 복제 지연으로 인해 소량의 데이터가 손실될 수 있습니다.

  기본으로 승격할 복제본을 선택할 때 복제 지연이 가장 적은 복제본을 ElastiCache 선택합니다. 즉, 가장 최신 복제본을 선택합니다. 이로써 손실 데이터 양을 최소화할 수 있습니다. 복제 지연 시간이 가장 짧은 복제본은 실패한 기본 노드와 같은 가용 영역에 있을 수도 있고 다른 가용 영역에 있을 수도 있습니다.

- 클러스터 모드가 비활성화된 Valkey 또는 Redis OSS 클러스터에서 읽기 전용 복제본을 기본으로 수동으로 승격하는 경우 다중 AZ 및 자동 장애 조치가 비활성화된 경우에만 승격할 수 있습니다. 읽기 전용 복제본을 기본으로 승격하려면 다음 단계를 따릅니다.

  1. 클러스터에서 다중 AZ를 비활성화합니다.
  2. 클러스터에서 자동 장애 조치를 비활성화합니다. 복제 그룹에 대한 자동 장애 조치 확인란을 선택 취소하여 콘솔을 통해 이 작업을 수행할 수 있습니다. `ModifyReplicationGroup` 작업을 호출할 `false` 때 `AutomaticFailoverEnabled` 속성을 로 설정 AWS CLI 하여 를 사용하여 이 작업을 수행할 수도 있습니다.
  3. 읽기 전용 복제본을 기본으로 승격합니다.
  4. 다중 AZ를 다시 활성화합니다.

- ElastiCache (Redis OSS) 다중 AZ 및 추가 전용 파일(AOF)은 상호 배타적입니다. 하나를 활성화하면 다른 하나를 활성화할 수 없습니다.
- 노드 장애는 드물지만 전체 가용 영역에 장애가 발생하는 경우로 인해 발생할 수 있습니다. 이 경우 장애가 발생한 기본 서버를 대체하는 복제본은 가용 영역이 백업된 경우에만 생성됩니다. 예를 들어, AZ-a에 기본 노드가 있고 AZ-b 및 AZ-c에 복제본이 있는 복제 그룹을 가정해 보겠습니다. 기본 노드에 문제가 발생하면 복제 지연 시간이 가장 짧은 사용 가능한 복제본을 기본 노드로 승격시킵니다. 그런 다음 AZ-a를 백업하고 사용할 수 있는 경우에만 AZ-a(실패한 프라이머리가 있는 위치)에서 새 복제본을 ElastiCache 생성합니다.
- 고객이 실행한 기본 재부팅은 자동 장애 조치를 트리거하지 않습니다. 다른 재부팅 및 장애는 자동 장애 조치를 트리거합니다.
- 기본을 재부팅하는 경우 온라인 상태가 되면 데이터가 지워집니다. 읽기 전용 복제본은 기본 클러스터가 지워진 것을 확인하면 데이터 복제본을 지우기 때문에 데이터가 손실됩니다.
- 읽기 전용 복제본이 승격된 후 다른 복제본은 새 기본 복제본과 동기화됩니다. 초기 동기화 후 복제본의 콘텐츠가 삭제되고 새 기본 복제본의 데이터가 동기화됩니다. 이 동기화 프로세스로 인해 복제본에 액세스할 수 없는 잠깐 중단이 발생합니다. 또한 이 동기화 프로세스로 인해 복제본과 동기화되는 동안 기본에 임시 로드가 증가합니다. 이 동작은 Valkey 및 Redis의 기본 동작OSS이며 다중 AZ의 ElastiCache 고유 동작은 아닙니다. 이 동작에 대한 자세한 내용은 Valkey 웹 사이트의 복제를 참조하세요.

> **Important**
>
> Valkey 7.2.6 이상 또는 Redis OSS 버전 2.8.22 이상의 경우 외부 복제본을 생성할 수 없습니다.

2.8.22 이전의 Redis OSS 버전의 경우 다중 AZ가 활성화된 ElastiCache 클러스터에 외부 복제본을 연결하지 않는 것이 좋습니다. 지원되지 않는 이 구성은 ElastiCache 가 장애 조치 및 복구를 제대로 수행하지 못하게 하는 문제를 일으킬 수 있습니다. 외부 복제본을 클러스터에 ElastiCache 연결하려면 연결하기 전에 다중 AZ가 활성화되어 있지 않은지 확인하세요.

동기화 및 백업 구현 방법

지원되는 모든 버전의 Valkey 및 Redis는 기본 노드와 복제본 노드 간의 백업 및 동기화를 OSS 지원합니다. 그러나 백업 및 동기화가 구현되는 방식은 버전에 따라 다릅니다.

Redis OSS 버전 2.8.22 이상

버전 2.8.22 이상에서 Redis OSS 복제는 두 가지 방법 중에서 선택합니다. 자세한 내용은 2.8.22 이전 Redis OSS 버전 및 스냅샷 및 복원 단원을 참조하세요.

포크 없는 프로세스 중 쓰기 로드가 많으면 변경 사항이 너무 많이 누적되어 성공적인 스냅샷을 방해하는 일이 발생하지 않도록 클러스터에 대한 쓰기가 지연됩니다.

2.8.22 이전 Redis OSS 버전

2.8.22 이전의 버전에서 Redis OSS 백업 및 동기화는 3단계 프로세스입니다.

1. 포크하고 백그라운드 프로세스에서 클러스터 데이터를 디스크에 직렬화합니다. 그러면 스냅샷이 point-in-time 생성됩니다.
2. 포그라운드에서 클라이언트 출력 버퍼에 변경 로그를 누적합니다.

   > **Important**
   >
   > 변경 로그가 클라이언트 출력 버퍼 크기를 초과하면 백업 또는 동기화가 실패합니다. 자세한 내용은 Valkey 또는 Redis OSS 스냅샷을 생성하기에 충분한 메모리 확보 단원을 참조하십시오.

3. 마지막으로 캐시 데이터와 변경 로그를 순서대로 복제본 클러스터에 전송합니다.

Valkey 또는 Redis OSS 복제 그룹 생성

복제본 노드가 있는 클러스터를 생성하기 위한 다음과 같은 옵션이 있습니다. 하나는 복제본을 기본 노드로 사용할 클러스터와 연결되지 않은 가용 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터가 이미 있는 경우에 적용됩니다. 다른 옵션은 클러스터와 읽기 전용 복제본으로 기본 노드를 생성해야할 때 적용됩니다. 현재 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터를 처음부터 생성해야 합니다.

옵션 1: 기존 클러스터를 사용하여 복제 그룹 생성

이 옵션을 사용하여 기존 단일 노드 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터를 활용합니다. 이 기존 노드를 새 클러스터의 기본 노드로 지정하고 클러스터에 1개~5개의 읽기 전용 복제본을 개별적으로 추가합니다. 기존 클러스터가 활성 상태인 경우 읽기 복제본은 생성되는 대로 해당 클러스터와 동기화됩니다. 기존 클러스터를 사용하여 복제 그룹 생성을 참조하세요.

> **Important**
>
> 기존 클러스터를 사용하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터를 생성할 수 없습니다. ElastiCache 콘솔을 사용하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터(API/CLI: 복제 그룹)를 생성하려면 섹션을 참조하세요Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 생성(콘솔).

옵션 2: 처음부터 Valkey 또는 Redis OSS 복제 그룹 생성

클러스터의 기본 노드로 사용할 수 있는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터가 아직 없거나 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터를 생성하려는 경우 이 옵션을 사용합니다. 처음부터 Valkey 또는 Redis OSS 복제 그룹 생성을(를) 참조하세요.

기존 클러스터를 사용하여 복제 그룹 생성

사용 가능한 클러스터는 기존 단일 노드 Valkey 또는 Redis OSS 클러스터입니다. 현재 Valkey 또는 RedisOSS(클러스터 모드 활성화됨)는 사용 가능한 단일 노드 클러스터를 사용하여 복제본이 있는 클러스터 생성을 지원하지 않습니다. Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터를 생성하려면 섹션을 참조하세요Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 생성(콘솔).

다음 절차는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 단일 노드 클러스터가 있는 경우에만 사용할 수 있습니다. 이 클러스터의 노드는 새 클러스터의 기본 노드가 됩니다. 새 클러스터의 기본으로 사용할 수 있는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터가 없는 경우 섹션을 참조하세요처음부터 Valkey 또는 Redis OSS 복제 그룹 생성.

기존 클러스터를 사용하여 복제 그룹 생성(콘솔)

사용 ElastiCache AWS Management Console 항목을 참조하세요.

사용 가능한 Valkey 또는 Redis OSS 캐시 클러스터를 사용하여 복제 그룹 생성(AWS CLI)

를 사용할 때 기본 에 사용 가능한 Valkey 또는 Redis OSS Cache 클러스터를 사용할 때 읽기 전용 복제본을 사용하여 복제 그룹을 생성하는 데는 두 단계가 있습니다 AWS CLI.

를 사용하는 AWS CLI 경우 사용 가능한 독립 실행형 노드를 클러스터의 기본 노드로 지정`--primary-cluster-id`하고 CLI 명령을 사용하여 클러스터에서 원하는 노드 수를 지정하는 복제 그룹을 생성합니다`create-replication-group`. 다음 파라미터를 포함합니다.

--replication-group-id

생성하는 복제 그룹의 이름입니다. 이 파라미터의 값은 추가되는 노드의 이름을 지정하는 기준으로 사용되는데, `--replication-group-id` 끝에 3자리 일련 번호가 추가됩니다. 예: `sample-repl-group-001`.

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹 이름 지정 제약 조건은 다음과 같습니다.

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

--replication-group-description

복제 그룹에 대한 설명입니다.

--num-node-groups

이 클러스터에 있는 노드의 수. 이 값에는 기본 노드가 포함됩니다. 이 파라미터의 최대값은 6입니다.

--primary-cluster-id

이 복제 그룹의 기본 노드로 사용할 수 있는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터 노드의 이름입니다.

다음 명령은 사용 가능한 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터를 복제 그룹의 기본 노드`redis01`로 `sample-repl-group` 사용하여 복제 그룹을 생성합니다. 이렇게 하면 읽기 전용 복제본인 새 노드 2개가 생성됩니다. `redis01`의 설정(즉, 파라미터 그룹, 보안 그룹, 노드 유형, 엔진 버전 등)은 복제 그룹의 모든 노드에 적용됩니다.

Linux, macOS, Unix의 경우:

```
aws elasticache create-replication-group \
   --replication-group-id sample-repl-group \
   --replication-group-description "demo cluster with replicas" \
   --num-cache-clusters 3 \
   --primary-cluster-id redis01
```

Windows의 경우:

```
aws elasticache create-replication-group ^
   --replication-group-id sample-repl-group ^
   --replication-group-description "demo cluster with replicas" ^
   --num-cache-clusters 3 ^
   --primary-cluster-id redis01
```

사용하려는 추가 정보 및 파라미터는 AWS CLI 주제를 참조하세요.create-replication-group.

다음으로 복제 그룹에 읽기 전용 복제본을 추가합니다.

복제 그룹이 생성된 후 `create-cache-cluster` 명령을 사용하여 해당 복제 그룹에 1~5개의 읽기 전용 복제본을 추가하여 다음 파라미터를 포함해야 합니다.

--cache-cluster-id

복제 그룹에 추가하는 클러스터의 이름입니다.

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.

--replication-group-id

이 캐시 클러스터를 추가하는 복제 그룹의 이름입니다.

`--cache-cluster-id` 파라미터 값만 변경하여 복제 그룹에 추가할 각 읽기 전용 복제본마다 이 명령을 반복합니다.

> **Note**
>
> 복제 그룹에는 읽기 전용 복제본이 최대 5개로 제한됩니다. 읽기 전용 복제본 5개가 이미 있는 복제 그룹에 읽기 전용 복제본을 추가하려고 하면 작업이 실패합니다.

다음 코드는 읽기 전용 복제본 `my-replica01`을 복제 그룹 `sample-repl-group`에 추가합니다. 기본 클러스터의 설정(즉, 파라미터 그룹, 보안 그룹, 노드 유형 등)은 복제 그룹에 추가될 때 노드에도 적용됩니다.

Linux, macOS, Unix의 경우:

```
aws elasticache create-cache-cluster \
   --cache-cluster-id my-replica01 \
   --replication-group-id sample-repl-group
```

Windows의 경우:

```
aws elasticache create-cache-cluster ^
   --cache-cluster-id my-replica01 ^
   --replication-group-id sample-repl-group
```

이 명령의 출력은 다음과 같습니다.

```
{
    "ReplicationGroup": {
        "Status": "creating",
        "Description": "demo cluster with replicas",
        "ClusterEnabled": false,
        "ReplicationGroupId": "sample-repl-group",
        "SnapshotRetentionLimit": 1,
        "AutomaticFailover": "disabled",
        "SnapshotWindow": "00:00-01:00",
        "SnapshottingClusterId": "redis01",
        "MemberClusters": [
            "sample-repl-group-001",
            "sample-repl-group-002",
            "redis01"
        ],
        "CacheNodeType": "cache.m4.large",
        "DataTiering": "disabled",
        "PendingModifiedValues": {}
    }
}
```

자세한 내용은 다음 AWS CLI 주제를 참조하세요.

- create-replication-group
- modify-replication-group

독립 실행형 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터에 복제본 추가(ElastiCache API)

를 사용할 때 클러스터의 기본 노드로 사용 가능한 독립 실행형 노드를 지정`PrimaryClusterId`하고 CLI 명령을 사용하여 클러스터에서 원하는 노드 수를 지정하는 복제 그룹을 ElastiCache API생성합니다`CreateReplicationGroup`. 다음 파라미터를 포함합니다.

ReplicationGroupId

생성하는 복제 그룹의 이름입니다. 이 파라미터의 값은 추가되는 노드의 이름을 지정하는 기준으로 사용되는데, `ReplicationGroupId` 끝에 3자리 일련 번호가 추가됩니다. 예: `sample-repl-group-001`.

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹 이름 지정 제약 조건은 다음과 같습니다.

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

ReplicationGroupDescription

복제본이 있는 클러스터에 대한 설명입니다.

NumCacheClusters

이 클러스터에 있는 노드의 수. 이 값에는 기본 노드가 포함됩니다. 이 파라미터의 최대값은 6입니다.

PrimaryClusterId

이 클러스터의 기본 노드로 사용할 수 있는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터의 이름입니다.

다음 명령은 사용 가능한 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터를 복제 그룹의 기본 노드`redis01`로 `sample-repl-group` 사용하여 복제본이 있는 클러스터를 생성합니다. 이렇게 하면 읽기 전용 복제본인 새 노드 2개가 생성됩니다. `redis01`의 설정(즉, 파라미터 그룹, 보안 그룹, 노드 유형, 엔진 버전 등)은 복제 그룹의 모든 노드에 적용됩니다.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=CreateReplicationGroup
   &Engine=redis
   &EngineVersion=6.0
   &ReplicationGroupDescription=Demo%20cluster%20with%20replicas
   &ReplicationGroupId=sample-repl-group
   &PrimaryClusterId=redis01
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

자세한 내용은 ElastiCache APL 주제를 참조하세요.

- CreateReplicationGroup
- ModifyReplicationGroup

다음으로 복제 그룹에 읽기 전용 복제본을 추가합니다.

복제 그룹이 생성된 후 `CreateCacheCluster` 작업을 사용하여 해당 복제 그룹에 1~5개의 읽기 전용 복제본을 추가하여 다음 파라미터를 포함해야 합니다.

CacheClusterId

복제 그룹에 추가하는 클러스터의 이름입니다.

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.

ReplicationGroupId

이 캐시 클러스터를 추가하는 복제 그룹의 이름입니다.

`CacheClusterId` 파라미터 값만 변경하여 복제 그룹에 추가할 각 읽기 전용 복제본마다 이 작업을 반복합니다.

다음 코드는 읽기 전용 복제본 `myReplica01`을 복제 그룹 `myReplGroup`에 추가합니다. 기본 클러스터의 설정(즉, 파라미터 그룹, 보안 그룹, 노드 유형 등)은 복제 그룹에 추가될 때 노드에도 적용됩니다.

```
https://elasticache.us-west-2.amazonaws.com/
 ?Action=CreateCacheCluster
 &CacheClusterId=myReplica01
 &ReplicationGroupId=myReplGroup
 &SignatureMethod=HmacSHA256
 &SignatureVersion=4
 &Version=2015-02-02
 &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
 &X-Amz-Credential=[your-access-key-id]/20150202/us-west-2/elasticache/aws4_request
 &X-Amz-Date=20150202T170651Z
 &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
 &X-Amz-Signature=[signature-value]
```

사용하려는 추가 정보 및 파라미터는 ElastiCache API 주제를 참조하세요.CreateCacheCluster.

처음부터 Valkey 또는 Redis OSS 복제 그룹 생성

다음은 기존 Valkey 또는 Redis OSS 클러스터를 기본으로 사용하지 않고 Valkey 또는 Redis OSS 복제 그룹을 생성하는 방법을 확인할 수 있습니다. ElastiCache 콘솔, 또는 를 사용하여 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 또는 Valkey AWS CLI또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹을 처음부터 생성할 수 있습니다 ElastiCacheAPI.

계속하기 전에 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹을 생성할지 아니면 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹을 생성할지 결정합니다. 결정에 대한 지침은 복제: Valkey 및 Redis OSS 클러스터 모드 비활성화됨 vs. 활성화됨를 참조하세요.

주제

- 처음부터 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹 생성
- Valkey 또는 RedisOSS(클러스터 모드 활성화됨)에서 처음부터 복제 그룹 생성

처음부터 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹 생성

콘솔, 또는 를 사용하여 ElastiCache Valkey AWS CLI또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹을 처음부터 생성할 수 있습니다 ElastiCache API. Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹에는 항상 하나의 노드 그룹, 기본 클러스터 및 최대 5개의 읽기 전용 복제본이 있습니다. Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹은 데이터 파티셔닝을 지원하지 않습니다.

> **Note**
>
> 노드/샤드 한도는 클러스터당 최대 500개로 늘릴 수 있습니다. 제한을 높이도록 요청하려면 AWS 서비스 제한을 참조하고 요청에 인스턴스 유형을 포함하세요.

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹을 처음부터 생성하려면 다음 방법 중 하나를 사용합니다.

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹을 처음부터 생성(AWS CLI)

다음 절차에서는 를 사용하여 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹을 생성합니다 AWS CLI.

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹을 처음부터 생성하면 `create-replication-group` 명령에 대한 단일 호출로 복제 그룹과 해당 노드를 AWS CLI 모두 생성합니다. 다음 파라미터를 포함합니다.

--replication-group-id

생성하는 복제 그룹의 이름입니다.

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹 이름 지정 제약 조건은 다음과 같습니다.

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

--replication-group-description

복제 그룹에 대한 설명입니다.

--num-cache-clusters

이 복제 그룹, 기본 및 읽기 전용 복제본과 함께 생성하려는 노드의 수입니다.

다중 AZ(`--automatic-failover-enabled`)를 활성화하는 경우 `--num-cache-clusters`의 값은 2 이상이어야 합니다.

--cache-node-type

복제 그룹에 있는 각 노드의 노드 유형입니다.

ElastiCache 는 다음 노드 유형을 지원합니다. 일반적으로, 현재 세대 유형은 이전 세대의 동급 제품에 비해 더 많은 메모리와 컴퓨팅 파워를 더 저렴하게 제공합니다.

각 노드 유형의 성능 세부 정보에 대한 자세한 내용은 Amazon EC2 인스턴스 유형 섹션을 참조하세요.

--data-tiering-enabled

r6gd 노드 유형을 사용하는 경우 이 파라미터를 설정합니다. 데이터 계층화를 원하지 않는 경우 `--no-data-tiering-enabled`를 설정합니다. 자세한 내용은 의 데이터 계층화 ElastiCache 단원을 참조하십시오.

--cache-parameter-group

엔진 버전에 해당하는 파라미터 그룹을 지정합니다. Redis OSS 3.2.4 이상을 실행하는 경우 `default.redis3.2` 파라미터 그룹 또는 에서 파생된 파라미터 그룹을 지정`default.redis3.2`하여 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹을 생성합니다. 자세한 내용은 Valkey 및 Redis OSS 파라미터 단원을 참조하십시오.

--network-type

`ipv4`, `ipv6`, `dual-stack` 중 하나입니다. 듀얼 스택을 선택한 경우, `--IpDiscovery` 파라미터를 `ipv4` 또는 `ipv6`로 설정해야 합니다.

--엔진

redis

--engine-version

다양한 기능 세트를 사용하려면 최신 엔진 버전을 선택합니다.

`-00#`을 복제 그룹 이름 뒤에 붙이면 복제 그룹 이름에서 노드 이름이 파생됩니다. 예를 들어, 복제 그룹 이름 `myReplGroup`을 사용하는 경우 기본 이름은 `myReplGroup-001`이 되고, 읽기 전용 복제본 이름은 `myReplGroup-002`에서 `myReplGroup-006` 사이가 됩니다.

이 복제 그룹에서 전송 중 데이터 암호화 또는 미사용 데이터 암호화를 활성화하려면 `--transit-encryption-enabled` 또는 `--at-rest-encryption-enabled` 파라미터 중 하나 또는 둘 다를 추가하고 다음 조건을 충족해야 합니다.

- 복제 그룹이 Redis OSS 버전 3.2.6 또는 4.0.10을 실행 중이어야 합니다.
- 복제 그룹은 Amazon 에서 생성해야 합니다VPC.
- 또한 `--cache-subnet-group` 파라미터도 포함해야 합니다.
- 또한 이 복제 그룹에서 작업을 수행하는 데 필요한 AUTH 토큰(암호)에 대해 고객 지정 문자열 값과 `--auth-token` 함께 파라미터를 포함해야 합니다.

다음 작업은 기본 복제본 1개와 복제본 2개로 구성된 3개의 노드`sample-repl-group`로 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹을 생성합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache create-replication-group \
   --replication-group-id sample-repl-group \
   --replication-group-description "Demo cluster with replicas" \
   --num-cache-clusters 3 \
   --cache-node-type cache.m4.large \
   --engine redis
```

Windows의 경우:

```
aws elasticache create-replication-group ^
   --replication-group-id sample-repl-group ^
   --replication-group-description "Demo cluster with replicas" ^
   --num-cache-clusters 3 ^
   --cache-node-type cache.m4.large ^
   --engine redis
```

이 명령의 출력은 다음과 같습니다.

```
{
    "ReplicationGroup": {
        "Status": "creating",
        "Description": "Demo cluster with replicas",
        "ClusterEnabled": false,
        "ReplicationGroupId": "sample-repl-group",
```

```
        "SnapshotRetentionLimit": 0,
        "AutomaticFailover": "disabled",
        "SnapshotWindow": "01:30-02:30",
        "MemberClusters": [
            "sample-repl-group-001",
            "sample-repl-group-002",
            "sample-repl-group-003"
        ],
        "CacheNodeType": "cache.m4.large",
        "DataTiering": "disabled",
        "PendingModifiedValues": {}
    }
}
```

사용하려는 추가 정보 및 파라미터는 AWS CLI 주제를 참조하세요create-replication-group.

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹을 처음부터 생성(ElastiCache API)

다음 절차에서는 를 사용하여 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹을 생성합니다 ElastiCache API.

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹을 처음부터 생성하면 `CreateReplicationGroup` 작업에 대한 단일 호출로 복제 그룹과 해당 노드를 ElastiCache API 모두 생성합니다. 다음 파라미터를 포함합니다.

ReplicationGroupId

생성하는 복제 그룹의 이름입니다.

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹 이름 지정 제약 조건은 다음과 같습니다.

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

ReplicationGroupDescription

복제 그룹에 대한 설명입니다.

NumCacheClusters

이 복제 그룹, 기본 및 읽기 전용 복제본과 함께 생성하려는 총 노드 수입니다.

다중 AZ(AutomaticFailoverEnabled=true)를 활성화하는 경우 NumCacheClusters의 값은 2 이상이어야 합니다.

CacheNodeType

복제 그룹에 있는 각 노드의 노드 유형입니다.

ElastiCache 는 다음 노드 유형을 지원합니다. 일반적으로, 현재 세대 유형은 이전 세대의 동급 제품에 비해 더 많은 메모리와 컴퓨팅 파워를 더 저렴하게 제공합니다.

각 노드 유형의 성능 세부 정보에 대한 자세한 내용은 Amazon EC2 인스턴스 유형 섹션을 참조하세요.

--data-tiering-enabled

r6gd 노드 유형을 사용하는 경우 이 파라미터를 설정합니다. 데이터 계층화를 원하지 않는 경우 --no-data-tiering-enabled를 설정합니다. 자세한 내용은 의 데이터 계층화 ElastiCache 단원을 참조하십시오.

CacheParameterGroup

엔진 버전에 해당하는 파라미터 그룹을 지정합니다. Redis OSS 3.2.4 이상을 실행하는 경우 default.redis3.2 파라미터 그룹 또는 에서 파생된 파라미터 그룹을 지정default.redis3.2하여 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹을 생성합니다. 자세한 내용은 Valkey 및 Redis OSS 파라미터 단원을 참조하십시오.

--network-type

ipv4, ipv, dual-stack 중 하나입니다. 듀얼 스택을 선택한 경우, --IpDiscovery 파라미터를 ipv4 또는 ipv6로 설정해야 합니다.

엔진

redis

EngineVersion

6.0

-00#을 복제 그룹 이름 뒤에 붙이면 복제 그룹 이름에서 노드 이름이 파생됩니다. 예를 들어, 복제 그룹 이름 myReplGroup을 사용하는 경우 기본 이름은 myReplGroup-001이 되고, 읽기 전용 복제본 이름은 myReplGroup-002에서 myReplGroup-006 사이가 됩니다.

이 복제 그룹에서 전송 중 데이터 암호화 또는 미사용 데이터 암호화를 활성화하려면 `TransitEncryptionEnabled=true` 또는 `AtRestEncryptionEnabled=true` 파라미터 중 하나 또는 둘 다를 추가하고 다음 조건을 충족해야 합니다.

- 복제 그룹이 Redis OSS 버전 3.2.6 또는 4.0.10을 실행 중이어야 합니다.
- 복제 그룹은 Amazon 에서 생성해야 합니다VPC.
- 또한 `CacheSubnetGroup` 파라미터도 포함해야 합니다.
- 또한 이 복제 그룹에서 작업을 수행하는 데 필요한 AUTH 토큰(암호)에 대해 고객이 지정한 문자열 값과 `AuthToken` 함께 파라미터를 포함해야 합니다.

다음 작업은 기본 복제본 1개와 복제본 2개로 구성된 3개의 노드`myReplGroup`로 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹을 생성합니다.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=CreateReplicationGroup
   &CacheNodeType=cache.m4.large
   &CacheParameterGroup=default.redis6.x
   &Engine=redis
   &EngineVersion=6.0
   &NumCacheClusters=3
   &ReplicationGroupDescription=test%20group
   &ReplicationGroupId=myReplGroup
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

사용하려는 추가 정보 및 파라미터는 ElastiCache API 주제를 참조하세요.CreateReplicationGroup.

Valkey 또는 RedisOSS(클러스터 모드 활성화됨)에서 처음부터 복제 그룹 생성

ElastiCache 콘솔, 또는 를 사용하여 Valkey AWS CLI또는 RedisOSS(클러스터 모드 활성화됨) 클러스터(API/CLI: 복제 그룹)를 생성할 수 있습니다 ElastiCache API. Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹에는 1~500개의 샤드(API/CLI: 노드 그룹), 각 샤드에 기본 노드, 각 샤드에 최대 5개의 읽기 전용 복제본이 있습니다. 하나의 클러스터당 최대 90개의 노드로 구성된 더 많은 수의 샤드와 더 적은 수의 복제본을 가진 클러스터를 생성할 수 있습니다. 이 클러스터 구성은 90개의 샤드 및 0개의 복제본부터 15개의 샤드 및 5개의 복제본까지 해당될 수 있으며, 이는 허용되는 최대 복제본 수입니다.

Valkey 또는 Redis OSS 엔진 버전이 5.0.6 이상인 경우 노드 또는 샤드 제한을 클러스터당 최대 500개로 늘릴 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. 일반적인 위험에는 서브넷 그룹의 서브넷 CIDR 범위가 너무 작거나 서브넷이 공유되어 다른 클러스터에서 많이 사용되는 경우가 포함됩니다. 자세한 내용은 서브넷 그룹 생성 단원을 참조하십시오.

5.0.6 이하의 버전에서 한도는 클러스터당 250개입니다.

한도 증가를 요청하려면 AWS 서비스 한도를 참조하고 한도 유형을 인스턴스 유형별 클러스터당 노드로 선택하세요.

Valkey 또는 Redis에서 클러스터 생성OSS(클러스터 모드 활성화됨)

- Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 생성(콘솔)
- 처음부터 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹 생성(AWS CLI)
- Valkey 또는 RedisOSS(클러스터 모드 활성화됨)에서 처음부터 복제 그룹 생성(ElastiCache API)

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 생성(콘솔)

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터를 생성하려면 섹션을 참조하세요Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 생성(콘솔). 클러스터 모드 활성화(스케일 아웃)에서 클러스터 모드를 활성화하고 두 개 이상의 샤드와 한 개의 복제본 노드를 지정합니다.

처음부터 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹 생성(AWS CLI)

다음 절차에서는 를 사용하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹을 생성합니다 AWS CLI.

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹을 처음부터 생성하면 `create-replication-group` 명령에 대한 단일 호출로 복제 그룹과 모든 노드를 AWS CLI 생성합니다. 다음 파라미터를 포함합니다.

--replication-group-id

생성하는 복제 그룹의 이름입니다.

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹 이름 지정 제약 조건은 다음과 같습니다.

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

--replication-group-description

복제 그룹에 대한 설명입니다.

--cache-node-type

복제 그룹에 있는 각 노드의 노드 유형입니다.

ElastiCache 는 다음 노드 유형을 지원합니다. 일반적으로, 현재 세대 유형은 이전 세대의 동급 제품에 비해 더 많은 메모리와 컴퓨팅 파워를 더 저렴하게 제공합니다.

각 노드 유형의 성능 세부 정보에 대한 자세한 내용은 Amazon EC2 인스턴스 유형 섹션을 참조하세요.

--data-tiering-enabled

r6gd 노드 유형을 사용하는 경우 이 파라미터를 설정합니다. 데이터 계층화를 원하지 않는 경우 `--no-data-tiering-enabled`를 설정합니다. 자세한 내용은 의 데이터 계층화 ElastiCache 단원을 참조하십시오.

--cache-parameter-group

`default.redis6.x.cluster.on` 파라미터 그룹 또는 에서 파생된 파라미터 그룹을 지정`default.redis6.x.cluster.on`하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹을 생성합니다. 자세한 내용은 Redis OSS 6.x 파라미터 변경 사항 단원을 참조하십시오.

--엔진

redis

--engine-version

3.2.4

--num-node-groups

이 복제 그룹의 노드 그룹 수입니다. 유효한 값은 1~500입니다.

> **Note**
>
> 노드/샤드 한도는 클러스터당 최대 500개로 늘릴 수 있습니다. 한도 증가를 요청하는 방법에 대한 지침은 AWS 서비스 제한을 참조하고 한도 유형을 '인스턴스 유형별 클러스터당 노드'로 선택하세요.

--replicas-per-node-group

각 노드 그룹의 복제본 노드 수입니다. 유효한 값은 0~5입니다.

--network-type

`ipv4`, `ipv`, `dual-stack` 중 하나입니다. 듀얼 스택을 선택한 경우, `--IpDiscovery` 파라미터를 `ipv4` 또는 `ipv6`로 설정해야 합니다.

이 복제 그룹에서 전송 중 데이터 암호화 또는 미사용 데이터 암호화를 활성화하려면 `--transit-encryption-enabled` 또는 `--at-rest-encryption-enabled` 파라미터 중 하나 또는 둘 다를 추가하고 다음 조건을 충족해야 합니다.

- 복제 그룹이 Redis OSS 버전 3.2.6 또는 4.0.10을 실행 중이어야 합니다.
- 복제 그룹은 Amazon 에서 생성해야 합니다VPC.
- 또한 `--cache-subnet-group` 파라미터도 포함해야 합니다.
- 또한 이 복제 그룹에서 작업을 수행하는 데 필요한 AUTH 토큰(암호)의 고객 지정 문자열 값과 `--auth-token` 함께 파라미터를 포함해야 합니다.

다음 작업은 노드 그룹/샤드OSS(--)가 각각 기본 복제본 `sample-repl-group` 1개와 읽기 전용 복제본 2개로 구성된 노드 그룹 3개(--)num-node-groups로 Valkey 또는 Redis(클러스터 모드 활성화됨) 복제 그룹을 생성합니다replicas-per-node-group.

Linux, macOS, Unix의 경우:

```
aws elasticache create-replication-group \
   --replication-group-id sample-repl-group \
   --replication-group-description "Demo cluster with replicas" \
   --num-node-groups 3 \
   --replicas-per-node-group 2 \
   --cache-node-type cache.m4.large \
   --engine redis \
   --security-group-ids SECURITY_GROUP_ID \
   --cache-subnet-group-name SUBNET_GROUP_NAME>
```

Windows의 경우:

```
aws elasticache create-replication-group ^
   --replication-group-id sample-repl-group ^
   --replication-group-description "Demo cluster with replicas" ^
   --num-node-groups 3 ^
   --replicas-per-node-group 2 ^
   --cache-node-type cache.m4.large ^
   --engine redis ^
   --security-group-ids SECURITY_GROUP_ID ^
   --cache-subnet-group-name SUBNET_GROUP_NAME>
```

앞에 나온 명령은 다음 출력을 생성합니다.

```
{
    "ReplicationGroup": {
        "Status": "creating",
        "Description": "Demo cluster with replicas",
        "ReplicationGroupId": "sample-repl-group",
        "SnapshotRetentionLimit": 0,
        "AutomaticFailover": "enabled",
        "SnapshotWindow": "05:30-06:30",
        "MemberClusters": [
            "sample-repl-group-0001-001",
            "sample-repl-group-0001-002",
            "sample-repl-group-0001-003",
            "sample-repl-group-0002-001",
            "sample-repl-group-0002-002",
            "sample-repl-group-0002-003",
            "sample-repl-group-0003-001",
```

```
            "sample-repl-group-0003-002",
            "sample-repl-group-0003-003"
        ],
        "PendingModifiedValues": {}
    }
}
```

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹을 처음부터 생성할 때 다음 예제와 같이 두 노드 그룹(콘솔: 샤드)을 구성하는 `--node-group-configuration` 파라미터를 사용하여 클러스터의 각 샤드를 구성할 수 있습니다. 첫 번째 샤드에는 2개의 노드(기본 1개, 읽기 전용 복제본 1개)가 있습니다. 두 번째 샤드에는 세 개의 노드(기본 한 개와 읽기 전용 복제본 두 개)가 있습니다.

--node-group-configuration

각 노드 그룹의 구성입니다. `--node-group-configuration` 파라미터는 다음 필드로 구성됩니다.

- `PrimaryAvailabilityZone` - 이 노드 그룹의 기본 노드가 있는 가용 영역입니다. 이 파라미터가 생략된 경우 기본 노드의 가용 영역을 ElastiCache 선택합니다.

  예: us-west-2a.

- `ReplicaAvailabilityZones` - 읽기 전용 복제본이 있는 가용 영역의 쉼표로 구분된 목록입니다. 이 목록의 가용 영역 수는 `ReplicaCount` 값과 일치해야 합니다. 이 파라미터가 생략된 경우 는 복제본 노드의 가용 영역을 ElastiCache 선택합니다.

  예: "us-west-2a,us-west-2b,us-west-2c"

- `ReplicaCount` - 이 노드 그룹의 복제본 노드 수입니다.
- `Slots` - 노드 그룹의 키스페이스를 지정하는 문자열입니다. 문자열 형식은 `startKey-endKey`입니다. 이 파라미터가 생략된 경우 는 노드 그룹 간에 키를 균등하게 ElastiCache 할당합니다.

  예: "0-4999"

다음 작업은 두 개의 노드 그룹/샤드OSS()를 `new-group` 사용하여 Valkey 또는 Redis(클러스터 모드 활성화됨) 복제 그룹을 생성합니다`--num-node-groups`. 위 예제와 달리 각 노드 그룹은 다른 노드 그룹(`--node-group-configuration`)과 다르게 구성됩니다.

Linux, macOS, Unix의 경우:

```
aws elasticache create-replication-group \
  --replication-group-id new-group \
  --replication-group-description "Sharded replication group" \
  --engine redis \
  --snapshot-retention-limit 8 \
  --cache-node-type cache.m4.medium \
  --num-node-groups 2 \
  --node-group-configuration \
      "ReplicaCount=1,Slots=0-8999,PrimaryAvailabilityZone='us-
east-1c',ReplicaAvailabilityZones='us-east-1b'" \
      "ReplicaCount=2,Slots=9000-16383,PrimaryAvailabilityZone='us-
east-1a',ReplicaAvailabilityZones='us-east-1a','us-east-1c'"
```

Windows의 경우:

```
aws elasticache create-replication-group ^
  --replication-group-id new-group ^
  --replication-group-description "Sharded replication group" ^
  --engine redis ^
  --snapshot-retention-limit 8 ^
  --cache-node-type cache.m4.medium ^
  --num-node-groups 2 ^
  --node-group-configuration \
      "ReplicaCount=1,Slots=0-8999,PrimaryAvailabilityZone='us-
east-1c',ReplicaAvailabilityZones='us-east-1b'" \
      "ReplicaCount=2,Slots=9000-16383,PrimaryAvailabilityZone='us-
east-1a',ReplicaAvailabilityZones='us-east-1a','us-east-1c'"
```

앞에 나온 작업은 다음 출력을 생성합니다.

```
{
    "ReplicationGroup": {
        "Status": "creating",
        "Description": "Sharded replication group",
        "ReplicationGroupId": "rc-rg",
        "SnapshotRetentionLimit": 8,
        "AutomaticFailover": "enabled",
        "SnapshotWindow": "10:00-11:00",
        "MemberClusters": [
            "rc-rg-0001-001",
            "rc-rg-0001-002",
            "rc-rg-0002-001",
```

```
            "rc-rg-0002-002",
            "rc-rg-0002-003"
        ],
        "PendingModifiedValues": {}
    }
}
```

사용하려는 추가 정보 및 파라미터는 AWS CLI 주제를 참조하세요.create-replication-group.

Valkey 또는 RedisOSS(클러스터 모드 활성화됨)에서 처음부터 복제 그룹 생성(ElastiCache API)

다음 절차에서는 를 사용하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹을 생성합니다 ElastiCache API.

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹을 처음부터 생성하면 `CreateReplicationGroup` 작업에 대한 단일 호출로 복제 그룹과 모든 노드를 ElastiCache API 생성합니다. 다음 파라미터를 포함합니다.

ReplicationGroupId

생성하는 복제 그룹의 이름입니다.

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹 이름 지정 제약 조건은 다음과 같습니다.

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

ReplicationGroupDescription

복제 그룹에 대한 설명입니다.

NumNodeGroups

이 복제 그룹과 함께 생성할 노드 그룹 수입니다. 유효한 값은 1~500입니다.

ReplicasPerNodeGroup

각 노드 그룹의 복제본 노드 수입니다. 유효한 값은 1~5입니다.

NodeGroupConfiguration

각 노드 그룹의 구성입니다. `NodeGroupConfiguration` 파라미터는 다음 필드로 구성됩니다.

- `PrimaryAvailabilityZone` - 이 노드 그룹의 기본 노드가 있는 가용 영역입니다. 이 파라미터가 생략된 경우 기본 노드의 가용 영역을 ElastiCache 선택합니다.

  예: us-west-2a.

- `ReplicaAvailabilityZones` - 읽기 전용 복제본이 있는 가용 영역 목록입니다. 이 목록의 가용 영역 수는 `ReplicaCount` 값과 일치해야 합니다. 이 파라미터가 생략된 경우 는 복제본 노드의 가용 영역을 ElastiCache 선택합니다.
- `ReplicaCount` - 이 노드 그룹의 복제본 노드 수입니다.
- `Slots` - 노드 그룹의 키스페이스를 지정하는 문자열입니다. 문자열 형식은 `startKey-endKey`입니다. 이 파라미터가 생략된 경우 는 노드 그룹 간에 키를 균등하게 ElastiCache 할당합니다.

  예: "0-4999"

CacheNodeType

복제 그룹에 있는 각 노드의 노드 유형입니다.

ElastiCache 는 다음 노드 유형을 지원합니다. 일반적으로, 현재 세대 유형은 이전 세대의 동급 제품에 비해 더 많은 메모리와 컴퓨팅 파워를 더 저렴하게 제공합니다.

각 노드 유형의 성능 세부 정보에 대한 자세한 내용은 Amazon EC2 인스턴스 유형 섹션을 참조하세요.

--data-tiering-enabled

r6gd 노드 유형을 사용하는 경우 이 파라미터를 설정합니다. 데이터 계층화를 원하지 않는 경우 `--no-data-tiering-enabled`를 설정합니다. 자세한 내용은 의 데이터 계층화 ElastiCache 단원을 참조하십시오.

CacheParameterGroup

`default.redis6.x.cluster.on` 파라미터 그룹 또는 에서 파생된 파라미터 그룹을 지정`default.redis6.x.cluster.on`하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹을 생성합니다. 자세한 내용은 Redis OSS 6.x 파라미터 변경 사항 단원을 참조하십시오.

--network-type

`ipv4`, `ipv`, `dual-stack` 중 하나입니다. 듀얼 스택을 선택한 경우, `--IpDiscovery` 파라미터를 `ipv4` 또는 `ipv6`로 설정해야 합니다.

엔진

redis

EngineVersion

6.0

이 복제 그룹에서 전송 중 데이터 암호화 또는 미사용 데이터 암호화를 활성화하려면 `TransitEncryptionEnabled=true` 또는 `AtRestEncryptionEnabled=true` 파라미터 중 하나 또는 둘 다를 추가하고 다음 조건을 충족해야 합니다.

- 복제 그룹이 Redis OSS 버전 3.2.6 또는 4.0.10을 실행 중이어야 합니다.
- 복제 그룹은 Amazon 에서 생성해야 합니다VPC.
- 또한 `CacheSubnetGroup` 파라미터도 포함해야 합니다.
- 또한 이 복제 그룹에서 작업을 수행하는 데 필요한 AUTH 토큰(암호)에 대해 고객이 지정한 문자열 값과 `AuthToken` 함께 파라미터를 포함해야 합니다.

줄바꿈은 가독성을 높이기 위해 추가되었습니다.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=CreateReplicationGroup
   &CacheNodeType=cache.m4.large
   &CacheParemeterGroup=default.redis6.xcluster.on
   &Engine=redis
   &EngineVersion=6.0
   &NumNodeGroups=3
   &ReplicasPerNodeGroup=2
   &ReplicationGroupDescription=test%20group
   &ReplicationGroupId=myReplGroup
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

사용하려는 추가 정보 및 파라미터는 ElastiCache API 주제를 참조하세요.CreateReplicationGroup.

복제 그룹의 세부 정보 보기

복제 그룹의 세부 정보를 보려는 경우가 있습니다. ElastiCache 콘솔, AWS CLI 용 ElastiCache또는 를 사용할 수 있습니다 ElastiCache API. Valkey 또는 RedisOSS(클러스터 모드 비활성화됨)와 Valkey 또는 RedisOSS(클러스터 모드 활성화됨)의 콘솔 프로세스는 다릅니다.

복제 그룹의 세부 정보 보기

- 복제본을 사용하여 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 보기
  - Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹 보기(콘솔)
  - Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹 보기(AWS CLI)
  - Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹 보기(ElastiCache API)
- 복제 그룹 보기: Valkey 또는 RedisOSS(클러스터 모드 활성화됨)
  - Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 보기(콘솔)
  - Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 보기(AWS CLI)
  - Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 보기(ElastiCache API)
- 복제 그룹의 세부 정보 보기(AWS CLI)
- 복제 그룹의 세부 정보 보기(ElastiCache API)

복제본을 사용하여 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 보기

ElastiCache 콘솔, 용 또는 를 사용하여 복제본OSS(API/CLI: 복제 그룹)이 있는 Valkey ElastiCache 또는 Redis(클러스터 모드 비활성화됨) 클러스터의 세부 정보를 볼 수 AWS CLI 있습니다 ElastiCache API.

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터의 세부 정보 보기

- Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹 보기(콘솔)
- Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹 보기(AWS CLI)
- Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹 보기(ElastiCache API)

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹 보기(콘솔)

ElastiCache 콘솔을 사용하여 복제본이 있는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터의 세부 정보를 보려면 주제를 참조하세요Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 세부 정보 보기(콘솔).

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹 보기(AWS CLI)

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹의 세부 정보를 표시하는 AWS CLI 예제는 섹션을 참조하세요복제 그룹의 세부 정보 보기(AWS CLI).

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹 보기(ElastiCache API)

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹의 세부 정보를 표시하는 ElastiCache API 예제는 섹션을 참조하세요복제 그룹의 세부 정보 보기(ElastiCache API).

복제 그룹 보기: Valkey 또는 RedisOSS(클러스터 모드 활성화됨)

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 보기(콘솔)

ElastiCache 콘솔을 사용하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 세부 정보를 보려면 섹션을 참조하세요Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터(콘솔)에 대한 세부 정보 보기.

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 보기(AWS CLI)

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹의 세부 정보를 표시하는 ElastiCache CLI 예제는 섹션을 참조하세요복제 그룹의 세부 정보 보기(AWS CLI).

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 보기(ElastiCache API)

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹의 세부 정보를 표시하는 ElastiCache API 예제는 섹션을 참조하세요복제 그룹의 세부 정보 보기(ElastiCache API).

복제 그룹의 세부 정보 보기(AWS CLI)

명령을 사용하여 복제 그룹의 세부 정보를 볼 수 있습니다 AWS CLI `describe-replication-groups`. 목록을 구체화하려면 다음과 같은 선택적 파라미터를 사용합니다. 파라미터가 생략되면 최대 100개의 복제 그룹에 대한 세부 정보가 반환됩니다.

선택 사항 파라미터

- `--replication-group-id` - 이 파라미터를 사용하여 지정한 복제 그룹의 세부 정보를 나열합니다. 지정한 복제 그룹의 노드 그룹이 둘 이상인 경우 결과가 노드 그룹별로 그룹화되어 반환됩니다.
- `--max-items` - 이 파라미터를 사용하여 나열된 복제 그룹 수를 제한합니다. `--max-items`의 값은 20 이상 또는 100 이하여야 합니다.

Example

다음 코드는 최대 100개의 복제 그룹에 대한 세부 정보를 나열합니다.

```
aws elasticache describe-replication-groups
```

다음 코드는 sample-repl-group의 세부 정보를 나열합니다.

```
aws elasticache describe-replication-groups --replication-group-id sample-repl-group
```

다음 코드는 sample-repl-group의 세부 정보를 나열합니다.

```
aws elasticache describe-replication-groups --replication-group-id sample-repl-group
```

다음 코드는 최대 25개의 복제 그룹에 대한 세부 정보를 나열합니다.

```
aws elasticache describe-replication-groups --max-items 25
```

이 작업의 출력은 다음과 같아야 합니다(JSON 형식).

```
{
   "ReplicationGroups": [
     {
       "Status": "available",
       "Description": "test",
       "NodeGroups": [
         {
            "Status": "available",
               "NodeGroupMembers": [
                  {
                     "CurrentRole": "primary",
                     "PreferredAvailabilityZone": "us-west-2a",
                     "CacheNodeId": "0001",
                     "ReadEndpoint": {
                        "Port": 6379,
                        "Address": "rg-name-001.1abc4d.0001.usw2.cache.amazonaws.com"
                     },
                     "CacheClusterId": "rg-name-001"
                  },
                  {
                     "CurrentRole": "replica",
```

```
                    "PreferredAvailabilityZone": "us-west-2b",
                    "CacheNodeId": "0001",
                    "ReadEndpoint": {
                       "Port": 6379,
                       "Address": "rg-name-002.1abc4d.0001.usw2.cache.amazonaws.com"
                    },
                    "CacheClusterId": "rg-name-002"
                 },
                 {
                    "CurrentRole": "replica",
                    "PreferredAvailabilityZone": "us-west-2c",
                    "CacheNodeId": "0001",
                    "ReadEndpoint": {
                       "Port": 6379,
                       "Address": "rg-name-003.1abc4d.0001.usw2.cache.amazonaws.com"
                    },
                    "CacheClusterId": "rg-name-003"
                 }
              ],
              "NodeGroupId": "0001",
              "PrimaryEndpoint": {
                 "Port": 6379,
                 "Address": "rg-name.1abc4d.ng.0001.usw2.cache.amazonaws.com"
              }
           }
        ],
        "ReplicationGroupId": "rg-name",
        "AutomaticFailover": "enabled",
        "SnapshottingClusterId": "rg-name-002",
        "MemberClusters": [
           "rg-name-001",
           "rg-name-002",
           "rg-name-003"
        ],
        "PendingModifiedValues": {}
     },
     {
     ... some output omitted for brevity
     }
  ]
}
```

자세한 내용은 주제 섹션을 참조 AWS CLI 하세요 ElastiCache .describe-replication-groups.

복제 그룹의 세부 정보 보기(ElastiCache API)

작업을 사용하여 복제에 대한 세부 정보를 볼 수 있습니다 AWS CLI `DescribeReplicationGroups`. 목록을 구체화하려면 다음과 같은 선택적 파라미터를 사용합니다. 파라미터가 생략되면 최대 100개의 복제 그룹에 대한 세부 정보가 반환됩니다.

선택 사항 파라미터

- `ReplicationGroupId` - 이 파라미터를 사용하여 지정한 복제 그룹의 세부 정보를 나열합니다. 지정한 복제 그룹의 노드 그룹이 둘 이상인 경우 결과가 노드 그룹별로 그룹화되어 반환됩니다.
- `MaxRecords` - 이 파라미터를 사용하여 나열된 복제 그룹 수를 제한합니다. `MaxRecords`의 값은 20 이상 또는 100 이하여야 합니다. 기본값은 100입니다.

Example

다음 코드는 최대 100개의 복제 그룹에 대한 세부 정보를 나열합니다.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeReplicationGroups
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

다음 코드는 `myReplGroup`의 세부 정보를 나열합니다.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeReplicationGroups
   &ReplicationGroupId=myReplGroup
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

다음 코드는 클러스터 최대 25개의 세부 정보를 나열합니다.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeReplicationGroups
```

```
    &MaxRecords=25
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

자세한 내용은 참조 주제를 참조하세요. ElastiCache API DescribeReplicationGroups.

복제 그룹 엔드포인트 찾기

애플리케이션은 복제 그룹의 모든 노드에 연결할 수 있습니다. 단, 해당 노드에 대한 DNS 엔드포인트 및 포트 번호가 있어야 합니다. Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 또는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹을 실행하는지 여부에 따라 다양한 엔드포인트에 관심이 있습니다.

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨)

복제본이 있는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터에는 기본 엔드포인트 , 리더 엔드포인트 및 노드 엔드포인트 라는 세 가지 유형의 엔드포인트가 있습니다. 기본 엔드포인트는 항상 클러스터의 기본 노드로 확인되는 DNS 이름입니다. 기본 엔드포인트는 읽기 전용 복제본을 기본 역할로 승격하는 것과 같은 클러스터 변경의 영향을 받지 않습니다. 쓰기 활동의 경우 애플리케이션을 기본 엔드포인트에 연결하는 것이 좋습니다.

리더 엔드포인트는 들어오는 연결을 ElastiCache 클러스터의 모든 읽기 전용 복제본 간에 균등하게 분할합니다. 애플리케이션이 연결을 생성하는 시기 또는 애플리케이션에서 연결을 다시 사용하는 방법과 같은 추가 요소가 트래픽 분산을 결정합니다. 리더 엔드포인트는 복제본이 추가 또는 제거되는 클러스터의 변경 사항을 실시간으로 반영합니다. ElastiCache (Redis OSS) 클러스터의 여러 읽기 전용 복제본을 서로 다른 AWS 가용 영역(AZ)에 배치하여 리더 엔드포인트의 고가용성을 보장할 수 있습니다.

> **Note**
>
> 리더 엔드포인트는 로드 밸런서가 아닙니다. 이는 라운드 로빈 방식으로 복제본 노드 중 하나의 IP 주소로 확인되는 DNS 레코드입니다.

읽기 활동의 경우 애플리케이션은 클러스터의 어떤 노드에도 연결할 수 있습니다. 기본 엔드포인트와 달리, 노드 엔드포인트는 특정 엔드포인트로 확인됩니다. 복제본을 추가하거나 삭제하는 것과 같이 클러스터를 변경하면 애플리케이션에서 노드 엔드포인트를 업데이트해야 합니다.

Valkey 또는 RedisOSS(클러스터 모드 활성화됨)

복제본이 있는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터는 샤드(API/CLI: 노드 그룹)가 여러 개 있으므로 기본 노드도 여러 개 있고 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터와 엔드포인트 구조가 다릅니다. Valkey 또는 RedisOSS(클러스터 모드 활성화됨)에는 클러스터의 모든 기본 및 노드 엔드포인트를 “알고” 있는 구성 엔드포인트가 있습니다. 애플리케이션은 이 구성 엔드포인트에 연결됩니다. 애플리케이션이 클러스터의 구성 엔드포인트인 Valkey 및 Redis 에 쓰거나 읽을 때마다 장면 OSS뒤에서 키가 속한 샤드와 사용할 샤드의 엔드포인트를 결정합니다. 이 모든 것이 애플리케이션에 매우 투명하게 진행됩니다.

ElastiCache 콘솔, AWS CLI또는 를 사용하여 클러스터의 엔드포인트를 찾을 수 있습니다 ElastiCache API.

복제 그룹 엔드포인트 찾기

복제 그룹의 엔드포인트를 찾으려면 다음 항목 중 하나를 참조하세요.

- Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터의 엔드포인트 찾기(콘솔)
- Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터(콘솔)의 엔드포인트 찾기
- Valkey 또는 Redis OSS 복제 그룹의 엔드포인트 찾기(AWS CLI)
- Valkey 또는 Redis OSS 복제 그룹의 엔드포인트 찾기(ElastiCache API)

복제 그룹 수정

⚠ 중요한 제약

- 현재 는 API 작업OSS(CLI:)을 사용하여 엔진 버전 변경과 같은 Valkey 또는 RedisModifyReplicationGroup(클러스터 모드 활성화됨) 복제 그룹의 제한된 수정을 ElastiCache 지원합니다modify-replication-group. API 작업(ModifyReplicationGroupShardConfigurationCLI:)을 사용하여 Valkey 또는 Redis(클러스터 모드 활성화됨) 클러스터의 샤드 OSS 수(노드 그룹)를 수정할 수 있습니다modify-replication-group-shard-configuration. 자세한 내용은 Valkey 또는 Redis에서 클러스터 크기 조정OSS(클러스터 모드 활성화됨) 단원을 참조하십시오.

  Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터에 대한 다른 수정 사항에서는 변경 사항을 통합하는 새 클러스터가 포함된 클러스터를 생성해야 합니다.
- Valkey 또는 RedisOSS(클러스터 모드 비활성화됨)와 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 및 복제 그룹을 최신 엔진 버전으로 업그레이드할 수 있습니다. 하지만 기존의 클러스터 또는 복제 그룹을 삭제하고 새로 만들지 않는 한 이전 엔진 버전으로 다운그레이드할 수 없습니다. 자세한 내용은 용 버전 관리 ElastiCache 단원을 참조하십시오.
- 아래 예제 ElastiCache 와 같이 OSS 클러스터 모드를 비활성화하여 콘솔 또는 modify-replication-group CLI 명령을 사용하여 클러스터 모드를 활성화한 기존 를 Valkey ModifyReplicationGroup API 또는 Redis 클러스터로 업그레이드할 수 있습니다. 또는 클러스터 모드 수정의 단계를 따를 수 있습니다.

ElastiCache 콘솔, 또는 를 사용하여 Valkey AWS CLI또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터의 설정을 수정할 수 있습니다 ElastiCache API. 현재 는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹에 대한 제한된 수의 수정을 ElastiCache 지원합니다. 다른 수정 사항에서는 현재 복제 그룹의 백업을 생성한 다음 해당 백업을 사용하여 새 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹을 시드해야 합니다.

주제

- 사용 AWS Management Console
- 사용 AWS CLI
- 사용 ElastiCache API

사용 AWS Management Console

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터를 수정하려면 섹션을 참조하세요 ElastiCache 클러스터 수정.

사용 AWS CLI

다음은 `modify-replication-group` 명령의 AWS CLI 예입니다. 동일한 명령을 사용하여 복제 그룹을 수정할 수 있습니다.

기존 Valkey 또는 Redis OSS 복제 그룹에서 다중 AZ 활성화:

Linux, macOS, Unix의 경우:

```
aws elasticache modify-replication-group \
   --replication-group-id myReplGroup \
   --multi-az-enabled = true
```

Windows의 경우:

```
aws elasticache modify-replication-group ^
   --replication-group-id myReplGroup ^
   --multi-az-enabled
```

클러스터 모드를 비활성화에서 활성화로 수정:

클러스터 모드를 비활성화에서 활성화로 수정하려면 먼저 클러스터 모드를 호환으로 설정해야 합니다. 호환 모드를 사용하면 클러스터 모드가 활성화되고 클러스터 모드가 비활성화된 상태에서 Valkey 또는 Redis OSS 클라이언트를 연결할 수 있습니다. 클러스터 모드 활성화를 사용하도록 모든 Valkey 또는 Redis OSS 클라이언트를 마이그레이션한 후 클러스터 모드 구성을 완료하고 클러스터 모드를 활성화 로 설정할 수 있습니다.

Linux, macOS, Unix의 경우:

클러스터 모드를 호환으로 설정합니다.

```
aws elasticache modify-replication-group \
   --replication-group-id myReplGroup \
   --cache-parameter-group-name myParameterGroupName \
   --cluster-mode compatible
```

클러스터 모드를 활성화로 설정합니다.

```
aws elasticache modify-replication-group \
   --replication-group-id myReplGroup \
   --cluster-mode enabled
```

Windows의 경우:

클러스터 모드를 호환으로 설정합니다.

```
aws elasticache modify-replication-group ^
   --replication-group-id myReplGroup ^
   --cache-parameter-group-name myParameterGroupName ^
   --cluster-mode compatible
```

클러스터 모드를 활성화로 설정합니다.

```
aws elasticache modify-replication-group ^
   --replication-group-id myReplGroup ^
   --cluster-mode enabled
```

명령에 대한 자세한 내용은 섹션을 참조하세요. AWS CLI `modify-replication-group` modify-replication-group 또는 ElastiCache (Redis OSS) 사용 설명서에서 클러스터 모드 수정.

사용 ElastiCache API

다음 ElastiCache API 작업은 기존 Valkey 또는 Redis OSS 복제 그룹에서 다중 AZ를 활성화합니다. 동일한 작업을 사용하여 복제 그룹을 수정할 수 있습니다.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=ModifyReplicationGroup
   &AutomaticFailoverEnabled=true
   &Mutli-AZEnabled=true
   &ReplicationGroupId=myReplGroup
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20141201T220302Z
   &Version=2014-12-01
   &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
   &X-Amz-Date=20141201T220302Z
   &X-Amz-SignedHeaders=Host
```

```
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

`ModifyReplicationGroup` 작업에 대한 자세한 내용은 섹션을 참조하세요. ElastiCache API [ModifyReplicationGroup](#).

복제 그룹 삭제

복제본(API/에서 복제 그룹이라고 함CLI)이 있는 클러스터 중 하나가 더 이상 필요하지 않은 경우 삭제할 수 있습니다. 복제 그룹을 삭제하면 는 해당 그룹의 모든 노드를 ElastiCache 삭제합니다.

작업을 시작한 후에는 중단하거나 취소할 수 없습니다.

> ⚠ Warning
>
> - ElastiCache (Redis OSS) 클러스터를 삭제하면 수동 스냅샷이 유지됩니다. 클러스터를 삭제하기 전에 최종 스냅샷을 생성할 수 있는 옵션도 있습니다. 자동 캐시 스냅샷은 보존되지 않습니다.
> - `CreateSnapshot` 최종 스냅샷을 생성하려면 권한이 필요합니다. 이 권한이 없으면 `Access Denied` 예외를 제외하고 API 호출이 실패합니다.

복제 그룹 삭제(콘솔)

복제본이 있는 클러스터를 삭제하려면 에서 클러스터 삭제 ElastiCache를 참조하세요.

복제 그룹 삭제(AWS CLI)

명령 사용 delete-replication-group 복제 그룹을 삭제합니다.

```
aws elasticache delete-replication-group --replication-group-id my-repgroup
```

결정을 확인하라는 메시지가 나타납니다. 즉시 작업을 시작하려면 [y](예)를 입력합니다. 프로세스가 시작되면 되돌릴 수 없습니다.

```
  After you begin deleting this replication group, all of its nodes will be deleted as
well.
  Are you sure you want to delete this replication group? [Ny]y

REPLICATIONGROUP  my-repgroup  My replication group  deleting
```

복제 그룹 삭제(ElastiCache API)

전화 DeleteReplicationGroup `ReplicationGroup` 파라미터를 사용합니다.

Example

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DeleteReplicationGroup
   &ReplicationGroupId=my-repgroup
   &Version=2014-12-01
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20141201T220302Z
   &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
   &X-Amz-Date=20141201T220302Z
   &X-Amz-SignedHeaders=Host
   &X-Amz-Expires=20141201T220302Z
   &X-Amz-Credential=<credential>
   &X-Amz-Signature=<signature>
```

> **Note**
>
> `RetainPrimaryCluster` 파라미터를 `true`로 설정하면 모든 읽기 전용 복제본이 삭제되지만 기본 클러스터는 보존됩니다.

복제본 수 변경

AWS Management Console, 또는 를 사용하여 Valkey 또는 Redis OSS 복제 그룹의 읽기 전용 복제본 수를 동적으로 늘리 AWS CLI거나 줄일 수 있습니다 ElastiCache API. 복제 그룹이 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹인 경우 복제본 수를 늘리거나 줄일 샤드(노드 그룹)를 선택할 수 있습니다.

복제 그룹의 복제본 수를 동적으로 변경하려면 다음 테이블에서 상황에 맞는 작업을 선택합니다.

방법	Valkey 또는 Redis의 경우 OSS(클러스터 모드 활성화됨)	Valkey 또는 Redis의 경우 OSS(클러스터 모드 비활성화됨)
복제본 추가	샤드의 복제본 수 늘리기	샤드의 복제본 수 늘리기 Valkey 또는 Redis에 대한 읽기 전용 복제본 추가OSS(클러스터 모드 비활성화됨)
복제본 삭제	샤드의 복제본 수 줄이기	샤드의 복제본 수 줄이기 Valkey 또는 Redis에 대한 읽기 전용 복제본 삭제OSS(클러스터 모드 비활성화됨)

샤드의 복제본 수 늘리기

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 샤드 또는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹의 복제본 수를 최대 5개까지 늘릴 수 있습니다. AWS Management Console, AWS CLI또는 를 사용하여 할 수 있습니다 ElastiCache API.

주제

- 사용 AWS Management Console
- 사용 AWS CLI
- 사용 ElastiCache API

사용 AWS Management Console

다음 절차에서는 콘솔을 사용하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹의 복제본 수를 늘립니다.

샤드의 복제본 수를 늘리려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 Valkey 또는 Redis OSS를 선택한 다음 복제본을 추가할 복제 그룹의 이름을 선택합니다.
3. 복제본을 추가할 각 샤드의 상자를 선택합니다.
4. Add replicas(복제본 추가)를 선택합니다.
5. Add Replicas to Shards(샤드에 복제본 추가) 페이지를 완료합니다.

   - New number of replicas/shard(새 복제본/샤드 수)에 선택한 모든 샤드에 있도록 하려는 복제본 수를 입력합니다. 이 값은 Current Number of Replicas per shard(샤드당 현재 복제본 수)보다 크거나 같아야 하며 5보다 작거나 같아야 합니다. 최소한 두 개의 복제본을 사용하는 것이 좋습니다.
   - 가용 영역 에서 각 새 복제본에 대해 가용 영역을 ElastiCache 선택하도록 기본 설정 없음을 선택하거나, 각 새 복제본에 대해 가용 영역을 선택하도록 가용 영역 지정을 선택합니다.

     가용 영역 지정를 선택할 경우 목록을 사용해 각각의 새 복제본에 대해 가용 영역을 지정하세요.
6. 추가를 선택하여 복제본을 추가하거나 취소를 선택하여 작업을 취소합니다.

사용 AWS CLI

Valkey 또는 Redis OSS 샤드의 복제본 수를 늘리려면 다음 파라미터와 함께 `increase-replica-count` 명령을 사용합니다.

- `--replication-group-id` - 필수입니다. 복제본 수를 늘리려는 복제 그룹을 식별합니다.
- `--apply-immediately` 또는 `--no-apply-immediately` – 필수입니다. 복제본 수를 즉시 늘릴 것인지(`--apply-immediately`) 아니면 다음 번 유지 관리 기간에 늘릴 것인지(`--no-apply-immediately`) 지정합니다. 현재 `--no-apply-immediately`는 지원되지 않습니다.
- `--new-replica-count` – 선택 사항입니다. 완료된 경우 원하는 복제본 노드의 수를 최대 5개까지 지정합니다. 노드 그룹 또는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 그룹이 하나만 있거나 모든 노드 그룹에 동일한 수의 복제본이 있어야 하는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹에 이 파라미터를 사용합니다. 이 값이 노드 그룹의 현재 복제본 수보다 크지 않은 경우 호출이 실패하고 예외가 발생합니다.
- `--replica-configuration` – 선택 사항입니다. 각 노드 그룹에 대해 독립적으로 복제본 수와 가용 영역을 설정할 수 있도록 합니다. 각 노드 그룹을 독립적으로 구성하려는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 그룹에 이 파라미터를 사용합니다.

  `--replica-configuration`에는 다음의 선택 멤버 3개가 있습니다.

  - `NodeGroupId` - 구성하는 노드 그룹의 4자리 ID입니다. Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹의 경우 샤드 ID는 항상 입니다`0001`. Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 노드 그룹의 (샤드) ID를 찾으려면 섹션을 참조하세요샤드 ID 찾기.
  - `NewReplicaCount` - 이 작업이 끝날 때 이 노드 그룹에 둘 복제본의 수입니다. 값은 현재 복제본 수보다 커야 하며, 최대 5개까지입니다. 이 값이 노드 그룹의 현재 복제본 수보다 크지 않은 경우 호출이 실패하고 예외가 발생합니다.
  - `PreferredAvailabilityZones` - 복제 그룹의 노드가 있을 가용 영역을 지정하는 `PreferredAvailabilityZone` 문자열의 목록입니다. `PreferredAvailabilityZone` 값의 수는 기본 노드를 고려하여 `NewReplicaCount`에 1을 더한 값과 같아야 합니다. 의 이 멤버`--replica-configuration`가 생략된 경우 ElastiCache (Redis OSS)는 각 새 복제본의 가용 영역을 선택합니다.

> **Important**
>
> 호출에 `--new-replica-count` 또는 `--replica-configuration` 파라미터를 포함해야 하지만, 둘 다 포함해서는 안 됩니다.

Example

다음은 복제 그룹 sample-repl-group의 복제본 수를 3으로 늘리는 예입니다. 예제가 완료되면 각 노드 그룹에 복제본 3개가 있습니다. 이 숫자는 단일 노드 그룹이 있는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 그룹인지 아니면 여러 노드 그룹이 있는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 그룹인지에 관계없이 적용됩니다.

Linux, macOS, Unix의 경우:

```
aws elasticache increase-replica-count \
    --replication-group-id sample-repl-group \
    --new-replica-count 3 \
    --apply-immediately
```

Windows의 경우:

```
aws elasticache increase-replica-count ^
    --replication-group-id sample-repl-group ^
    --new-replica-count 3 ^
    --apply-immediately
```

다음은 복제 그룹 sample-repl-group의 복제본 수를 지정된 2개의 노드 그룹에 대해 지정된 값으로 늘리는 예입니다. 노드 그룹이 여러 개 있는 경우 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹입니다. 선택적 PreferredAvailabilityZones를 지정할 때 나열된 가용 영역 수는 NewReplicaCount에 1 이상을 더한 값과 같아야 합니다. 이러한 접근 방식은 NodeGroupId에서 식별한 그룹에 대한 기본 노드를 설명합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache increase-replica-count \
    --replication-group-id sample-repl-group \
    --replica-configuration \
        NodeGroupId=0001,NewReplicaCount=2,PreferredAvailabilityZones=us-east-1a,us-east-1c,us-east-1b \
        NodeGroupId=0003,NewReplicaCount=3,PreferredAvailabilityZones=us-east-1a,us-east-1b,us-east-1c,us-east-1c \
    --apply-immediately
```

Windows의 경우:

```
aws elasticache increase-replica-count ^
```

```
    --replication-group-id sample-repl-group ^
    --replica-configuration ^
        NodeGroupId=0001,NewReplicaCount=2,PreferredAvailabilityZones=us-east-1a,us-
east-1c,us-east-1b ^
        NodeGroupId=0003,NewReplicaCount=3,PreferredAvailabilityZones=us-east-1a,us-
east-1b,us-east-1c,us-east-1c \
    --apply-immediately
```

를 사용하여 복제본 수를 늘리는 방법에 대한 자세한 내용은 Amazon 명령줄 참조increase-replica-count의 CLI섹션을 참조하세요. ElastiCache

사용 ElastiCache API

Valkey 또는 Redis OSS 샤드의 복제본 수를 늘리려면 다음 파라미터와 함께 `IncreaseReplicaCount` 작업을 사용합니다.

- `ReplicationGroupId` - 필수입니다. 복제본 수를 늘리려는 복제 그룹을 식별합니다.
- `ApplyImmediately` - 필수입니다. 복제본 수를 즉시 늘릴 것인지(`ApplyImmediately=True`) 아니면 다음 번 유지 관리 기간에 늘릴 것인지(`ApplyImmediately=False`) 지정합니다. 현재 `ApplyImmediately=False`는 지원되지 않습니다.
- `NewReplicaCount` – 선택 사항입니다. 완료된 경우 원하는 복제본 노드의 수를 최대 5개까지 지정합니다. 노드 그룹이 하나만 있는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹 또는 모든 노드 그룹에 동일한 수의 복제본을 포함하도록 하려는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 그룹에 이 파라미터를 사용합니다. 이 값이 노드 그룹의 현재 복제본 수보다 크지 않은 경우 호출이 실패하고 예외가 발생합니다.
- `ReplicaConfiguration` – 선택 사항입니다. 각 노드 그룹에 대해 독립적으로 복제본 수와 가용 영역을 설정할 수 있도록 합니다. 각 노드 그룹을 독립적으로 구성하려는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 그룹에 이 파라미터를 사용합니다.

  `ReplicaConfiguraion`에는 다음의 선택 멤버 3개가 있습니다.

  - `NodeGroupId` - 구성하는 노드 그룹의 4자리 ID입니다. Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹의 경우 노드 그룹(샤드) ID는 항상 입니다`0001`. Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 노드 그룹의 (샤드) ID를 찾으려면 섹션을 참조하세요샤드 ID 찾기.
  - `NewReplicaCount` - 이 작업이 끝날 때 이 노드 그룹에 둘 복제본의 수입니다. 값은 현재 복제본 수보다 커야 하며 최대 5개까지입니다. 이 값이 노드 그룹의 현재 복제본 수보다 크지 않은 경우 호출이 실패하고 예외가 발생합니다.
  - `PreferredAvailabilityZones` - 복제 그룹의 노드가 있을 가용 영역을 지정하는 `PreferredAvailabilityZone` 문자열의 목록입니다. `PreferredAvailabilityZone` 값

의 수는 기본 노드를 고려하여 NewReplicaCount에 1을 더한 값과 같아야 합니다. 의 이 멤버ReplicaConfiguration가 생략된 경우 ElastiCache (Redis OSS)는 각 새 복제본의 가용 영역을 선택합니다.

> **Important**
>
> 호출에 NewReplicaCount 또는 ReplicaConfiguration 파라미터를 포함해야 하지만, 둘 다 포함해서는 안 됩니다.

Example

다음은 복제 그룹 sample-repl-group의 복제본 수를 3으로 늘리는 예입니다. 예제가 완료되면 각 노드 그룹에 복제본 3개가 있습니다. 이 숫자는 단일 노드 그룹이 있는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 그룹이든, 여러 노드 그룹이 있는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 그룹이든 관계없이 적용됩니다.

```
https://elasticache.us-west-2.amazonaws.com/
      ?Action=IncreaseReplicaCount
      &ApplyImmediately=True
      &NewReplicaCount=3
      &ReplicationGroupId=sample-repl-group
      &Version=2015-02-02
      &SignatureVersion=4
      &SignatureMethod=HmacSHA256
      &Timestamp=20150202T192317Z
      &X-Amz-Credential=<credential>
```

다음은 복제 그룹 sample-repl-group의 복제본 수를 지정된 2개의 노드 그룹에 대해 지정된 값으로 늘리는 예입니다. 노드 그룹이 여러 개 있는 경우 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹입니다. 선택적 PreferredAvailabilityZones를 지정할 때 나열된 가용 영역 수는 NewReplicaCount에 1 이상을 더한 값과 같아야 합니다. 이러한 접근 방식은 NodeGroupId에서 식별한 그룹에 대한 기본 노드를 설명합니다.

```
https://elasticache.us-west-2.amazonaws.com/
      ?Action=IncreaseReplicaCount
      &ApplyImmediately=True
      &ReplicaConfiguration.ConfigureShard.1.NodeGroupId=0001
      &ReplicaConfiguration.ConfigureShard.1.NewReplicaCount=2
```

```
 &ReplicaConfiguration.ConfigureShard.1.PreferredAvailabilityZones.PreferredAvailabilityZone.1=
east-1a

 &ReplicaConfiguration.ConfigureShard.1.PreferredAvailabilityZones.PreferredAvailabilityZone.2=
east-1c

 &ReplicaConfiguration.ConfigureShard.1.PreferredAvailabilityZones.PreferredAvailabilityZone.3=
east-1b
      &ReplicaConfiguration.ConfigureShard.2.NodeGroupId=0003
      &ReplicaConfiguration.ConfigureShard.2.NewReplicaCount=3

 &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.1=
east-1a

 &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.2=
east-1b

 &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.3=
east-1c

 &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.4=
east-1c
      &ReplicationGroupId=sample-repl-group
      &Version=2015-02-02
      &SignatureVersion=4
      &SignatureMethod=HmacSHA256
      &Timestamp=20150202T192317Z
      &X-Amz-Credential=<credential>
```

를 사용하여 복제본 수를 늘리는 방법에 대한 자세한 내용은 Amazon 참조IncreaseReplicaCount의 API섹션을 참조하세요. ElastiCache API

샤드의 복제본 수 줄이기

Valkey 또는 Redis용 샤드OSS(클러스터 모드 활성화됨) 또는 Valkey 또는 Redis용 복제 그룹OSS(클러스터 모드 비활성화됨)의 복제본 수를 줄일 수 있습니다.

- Valkey 또는 RedisOSS(클러스터 모드 비활성화됨)의 경우 다중 AZ가 활성화된 경우 복제본 수를 1로 줄이고 활성화되지 않은 경우 0으로 줄일 수 있습니다.
- Valkey 또는 RedisOSS(클러스터 모드 활성화됨)의 경우 복제본 수를 0으로 줄일 수 있습니다. 그러나 기본 노드가 실패할 경우 복제본으로 장애 조치를 수행할 수 없습니다.

AWS Management Console, AWS CLI 또는 ElastiCache API 를 사용하여 노드 그룹(샤드) 또는 복제 그룹의 복제본 수를 줄일 수 있습니다.

주제

- 사용 AWS Management Console
- 사용 AWS CLI
- 사용 ElastiCache API

사용 AWS Management Console

다음 절차에서는 콘솔을 사용하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹의 복제본 수를 줄입니다.

Valkey 또는 Redis OSS 샤드의 복제본 수를 줄이려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 Valkey 또는 Redis OSS를 선택한 다음 복제본을 삭제할 복제 그룹의 이름을 선택합니다.
3. 복제본 노드를 제거할 각 샤드의 상자를 선택합니다.
4. Delete replicas(복제본 삭제)를 선택합니다.
5. Delete Replicas from Shards(샤드에서 복제본 삭제) 페이지를 완료합니다.

    a. New number of replicas/shard(새 복제본/샤드 수)에 선택한 샤드에 있도록 하려는 복제본 수를 입력합니다. 이 숫자는 1보다 크거나 같아야 합니다. 샤드마다 최소한 두 개의 복제본을 사용하는 것이 좋습니다.

    b. 삭제를 선택하여 복제본을 삭제하거나 취소를 선택하여 작업을 취소합니다.

> **Important**
>
> - 삭제할 복제본 노드를 지정하지 않으면 ElastiCache (Redis OSS)에서 삭제할 복제본 노드를 자동으로 선택합니다. 이때 ElastiCache (Redis OSS)는 복제 그룹에 대한 다중 AZ 아키텍처를 유지한 다음 복제본을 기본 복제 지연이 최소인 상태로 유지하려고 시도합니다.
> - 복제 그룹의 기본 노드는 삭제할 수 없습니다. 기본 노드를 삭제하도록 지정하면 작업이 실패하고, 기본 노드가 삭제되도록 선택되었음을 나타내는 오류 이벤트가 발생합니다.

사용 AWS CLI

Valkey 또는 Redis OSS 샤드의 복제본 수를 줄이려면 다음 파라미터와 함께 `decrease-replica-count` 명령을 사용합니다.

- `--replication-group-id` - 필수입니다. 복제본 수를 줄이려는 복제 그룹을 식별합니다.
- `--apply-immediately` 또는 `--no-apply-immediately` – 필수입니다. 복제본 수를 즉시 줄일 것인지(`--apply-immediately`) 아니면 다음 번 유지 관리 기간에 줄일 것인지(`--no-apply-immediately`) 지정합니다. 현재 `--no-apply-immediately`는 지원되지 않습니다.
- `--new-replica-count` – 선택 사항입니다. 원하는 복제본 노드의 수를 지정합니다. `--new-replica-count`의 값은 유효해야 하며, 노드 그룹의 현재 복제본 수보다 작아야 합니다. 허용된 최소값은 샤드의 복제본 수 줄이기 섹션을 참조하세요. `--new-replica-count`의 값이 이 요구 사항을 충족하지 않는 경우 호출이 실패합니다.
- `--replicas-to-remove` – 선택 사항입니다. 제거할 복제본 노드를 IDs 지정하는 노드 목록을 포함합니다.
- `--replica-configuration` – 선택 사항입니다. 각 노드 그룹에 대해 독립적으로 복제본 수와 가용 영역을 설정할 수 있도록 합니다. 각 노드 그룹을 독립적으로 구성하려는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 그룹에 이 파라미터를 사용합니다.

  `--replica-configuration`에는 다음의 선택 멤버 3개가 있습니다.

  - `NodeGroupId` - 구성하는 노드 그룹의 4자리 ID입니다. Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹의 경우 샤드 ID는 항상 입니다`0001`. Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 노드 그룹의 (샤드) ID를 찾으려면 섹션을 참조하세요샤드 ID 찾기.
  - `NewReplicaCount` - 선택적 파라미터로, 원하는 복제본 노드의 수를 지정합니다. `NewReplicaCount`의 값은 유효해야 하며, 노드 그룹의 현재 복제본 수보다 작아야 합니다. 허용된 최소값은 샤드의 복제본 수 줄이기 섹션을 참조하세요. `NewReplicaCount`의 값이 이 요구 사항을 충족하지 않는 경우 호출이 실패합니다.

- `PreferredAvailabilityZones` - 복제 그룹의 노드가 있는 가용 영역을 지정하는 `PreferredAvailabilityZone` 문자열의 목록입니다. `PreferredAvailabilityZone` 값의 수는 기본 노드를 고려하여 `NewReplicaCount`에 1을 더한 값과 같아야 합니다. 의 이 멤버--`replica-configuration`가 생략된 경우 ElastiCache (Redis OSS)는 각 새 복제본의 가용 영역을 선택합니다.

> **Important**
>
> `--new-replica-count`, `--replicas-to-remove` 또는 `--replica-configuration` 파라미터 중 하나만 포함해야 합니다.

Example

다음은 `--new-replica-count`를 사용해 복제 그룹 `sample-repl-group`의 복제본 수를 1로 줄이는 예입니다. 예제가 완료되면 각 노드 그룹에 복제본 1개가 있습니다. 이 숫자는 단일 노드 그룹이 있는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 그룹인지 아니면 여러 노드 그룹이 있는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 그룹인지에 관계없이 적용됩니다.

Linux, macOS, Unix의 경우:

```
aws elasticache decrease-replica-count
    --replication-group-id sample-repl-group \
    --new-replica-count 1 \
    --apply-immediately
```

Windows의 경우:

```
aws elasticache decrease-replica-count ^
    --replication-group-id sample-repl-group ^
    --new-replica-count 1 ^
    --apply-immediately
```

다음은 노드 그룹에서 지정된 복제본 2개(`0001` 및 `0003`)를 제거하여 복제 그룹 `sample-repl-group`의 복제본 수를 줄이는 예입니다.

Linux, macOS, Unix의 경우:

```
aws elasticache decrease-replica-count \
```

```
    --replication-group-id sample-repl-group \
    --replicas-to-remove 0001,0003 \
    --apply-immediately
```

Windows의 경우:

```
aws elasticache decrease-replica-count ^
    --replication-group-id sample-repl-group ^
    --replicas-to-remove 0001,0003 \
    --apply-immediately
```

다음은 `--replica-configuration`을 사용해 복제 그룹 `sample-repl-group`의 복제본 수를 지정된 2개의 노드 그룹에 대해 지정된 값으로 줄이는 예입니다. 노드 그룹이 여러 개 있는 경우 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹입니다. 선택적 `PreferredAvailabilityZones`를 지정할 때 나열된 가용 영역 수는 `NewReplicaCount`에 1 이상을 더한 값과 같아야 합니다. 이러한 접근 방식은 `NodeGroupId`에서 식별한 그룹에 대한 기본 노드를 설명합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache decrease-replica-count \
    --replication-group-id sample-repl-group \
    --replica-configuration \
        NodeGroupId=0001,NewReplicaCount=1,PreferredAvailabilityZones=us-east-1a,us-
east-1c \
        NodeGroupId=0003,NewReplicaCount=2,PreferredAvailabilityZones=us-east-1a,us-
east-1b,us-east-1c \
    --apply-immediately
```

Windows의 경우:

```
aws elasticache decrease-replica-count ^
    --replication-group-id sample-repl-group ^
    --replica-configuration ^
        NodeGroupId=0001,NewReplicaCount=2,PreferredAvailabilityZones=us-east-1a,us-
east-1c ^
        NodeGroupId=0003,NewReplicaCount=3,PreferredAvailabilityZones=us-east-1a,us-
east-1b,us-east-1c \
    --apply-immediately
```

를 사용하여 복제본 수를 줄이는 방법에 대한 자세한 내용은 Amazon 명령줄 참조decrease-replica-count의 CLI섹션을 참조하세요. ElastiCache

사용 ElastiCache API

Valkey 또는 Redis OSS 샤드의 복제본 수를 줄이려면 다음 파라미터와 함께 `DecreaseReplicaCount` 작업을 사용합니다.

- `ReplicationGroupId` - 필수입니다. 복제본 수를 줄이려는 복제 그룹을 식별합니다.
- `ApplyImmediately` - 필수입니다. 복제본 수를 즉시 줄일 것인지(`ApplyImmediately=True`) 아니면 다음 번 유지 관리 기간에 줄일 것인지(`ApplyImmediately=False`) 지정합니다. 현재 `ApplyImmediately=False`는 지원되지 않습니다.
- `NewReplicaCount` – 선택 사항입니다. 원하는 복제본 노드의 수를 지정합니다. `NewReplicaCount`의 값은 유효해야 하며, 노드 그룹의 현재 복제본 수보다 작아야 합니다. 허용된 최소값은 샤드의 복제본 수 줄이기 섹션을 참조하세요. `--new-replica-count`의 값이 이 요구 사항을 충족하지 않는 경우 호출이 실패합니다.
- `ReplicasToRemove` – 선택 사항입니다. 제거할 복제본 노드를 IDs 지정하는 노드 목록을 포함합니다.
- `ReplicaConfiguration` – 선택 사항입니다. 각 노드 그룹에 대해 독립적으로 복제본 수와 가용 영역을 설정할 수 있도록 허용하는 노드 그룹의 목록을 포함합니다. 각 노드 그룹을 독립적으로 구성하려는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 그룹에 이 파라미터를 사용합니다.

  `ReplicaConfiguraion`에는 다음의 선택 멤버 3개가 있습니다.

  - `NodeGroupId` - 구성하는 노드 그룹의 4자리 ID입니다. Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹의 경우 노드 그룹 ID는 항상 입니다`0001`. Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 노드 그룹의 (샤드) ID를 찾으려면 섹션을 참조하세요샤드 ID 찾기.
  - `NewReplicaCount` - 이 작업이 끝날 때 이 노드 그룹에 둘 복제본의 수입니다. 값은 현재 복제본 수보다 작아야 하며, 다중 AZ가 활성화된 경우 최소 1 또는 자동 장애 조치가 있는 다중 AZ가 활성화되지 않은 경우 0까지 줄입니다. 이 값이 노드 그룹의 현재 복제본 수보다 작지 않은 경우 호출이 실패하고 예외가 발생합니다.
  - `PreferredAvailabilityZones` - 복제 그룹의 노드가 있는 가용 영역을 지정하는 `PreferredAvailabilityZone` 문자열의 목록입니다. `PreferredAvailabilityZone` 값의 수는 기본 노드를 고려하여 `NewReplicaCount`에 1을 더한 값과 같아야 합니다. 의 이 멤버`ReplicaConfiguration`가 생략된 경우 ElastiCache (Redis OSS)는 각 새 복제본의 가용 영역을 선택합니다.

> **Important**
>
> NewReplicaCount, ReplicasToRemove 또는 ReplicaConfiguration 파라미터 중 하나만 포함해야 합니다.

Example

다음은 NewReplicaCount를 사용해 복제 그룹 sample-repl-group의 복제본 수를 1로 줄이는 예입니다. 예제가 완료되면 각 노드 그룹에 복제본 1개가 있습니다. 이 숫자는 단일 노드 그룹이 있는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 그룹인지 아니면 여러 노드 그룹이 있는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 그룹인지에 관계없이 적용됩니다.

```
https://elasticache.us-west-2.amazonaws.com/
      ?Action=DecreaseReplicaCount
      &ApplyImmediately=True
      &NewReplicaCount=1
      &ReplicationGroupId=sample-repl-group
      &Version=2015-02-02
      &SignatureVersion=4
      &SignatureMethod=HmacSHA256
      &Timestamp=20150202T192317Z
      &X-Amz-Credential=<credential>
```

다음은 노드 그룹에서 지정된 복제본 2개(0001 및 0003)를 제거하여 복제 그룹 sample-repl-group의 복제본 수를 줄이는 예입니다.

```
https://elasticache.us-west-2.amazonaws.com/
      ?Action=DecreaseReplicaCount
      &ApplyImmediately=True
      &ReplicasToRemove.ReplicaToRemove.1=0001
      &ReplicasToRemove.ReplicaToRemove.2=0003
      &ReplicationGroupId=sample-repl-group
      &Version=2015-02-02
      &SignatureVersion=4
      &SignatureMethod=HmacSHA256
      &Timestamp=20150202T192317Z
      &X-Amz-Credential=<credential>
```

다음은 ReplicaConfiguration을 사용해 복제 그룹 sample-repl-group의 복제본 수를 지정된 2개의 노드 그룹에 대해 지정된 값으로 줄이는 예입니다. 노드 그룹이 여러 개 있는 경우 Valkey 또는

RedisOSS(클러스터 모드 활성화됨) 복제 그룹입니다. 선택적 PreferredAvailabilityZones를 지정할 때 나열된 가용 영역 수는 NewReplicaCount에 1 이상을 더한 값과 같아야 합니다. 이러한 접근 방식은 NodeGroupId에서 식별한 그룹에 대한 기본 노드를 설명합니다.

```
https://elasticache.us-west-2.amazonaws.com/
      ?Action=DecreaseReplicaCount
      &ApplyImmediately=True
      &ReplicaConfiguration.ConfigureShard.1.NodeGroupId=0001
      &ReplicaConfiguration.ConfigureShard.1.NewReplicaCount=1

 &ReplicaConfiguration.ConfigureShard.1.PreferredAvailabilityZones.PreferredAvailabilityZone.1=
east-1a

 &ReplicaConfiguration.ConfigureShard.1.PreferredAvailabilityZones.PreferredAvailabilityZone.2=
east-1c
      &ReplicaConfiguration.ConfigureShard.2.NodeGroupId=0003
      &ReplicaConfiguration.ConfigureShard.2.NewReplicaCount=2

 &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.1=
east-1a

 &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.2=
east-1b

 &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.4=
east-1c
      &ReplicationGroupId=sample-repl-group
      &Version=2015-02-02
      &SignatureVersion=4
      &SignatureMethod=HmacSHA256
      &Timestamp=20150202T192317Z
      &X-Amz-Credential=<credential>
```

를 사용하여 복제본 수를 줄이는 방법에 대한 자세한 내용은 Amazon 참조DecreaseReplicaCount의 API섹션을 참조하세요. ElastiCache API

Valkey 또는 Redis에 대한 읽기 전용 복제본 추가OSS(클러스터 모드 비활성화됨)

다음 주제의 정보는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹에만 적용됩니다.

읽기 트래픽이 증가함에 따라 이러한 읽기를 더 많은 노드로 분산시켜 어느 한 노드에 대한 읽기 압력을 줄이려고 할 수 있습니다. 이 주제에서는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터에 읽기 전용 복제본을 추가하는 방법을 찾을 수 있습니다.

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹에는 최대 5개의 읽기 전용 복제본이 있을 수 있습니다. 읽기 전용 복제본 5개가 이미 있는 복제 그룹에 읽기 전용 복제본을 추가하려고 하면 작업이 실패합니다.

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹에 복제본을 추가하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- Valkey 또는 Redis에서 클러스터 크기 조정OSS(클러스터 모드 활성화됨)
- 샤드의 복제본 수 늘리기

ElastiCache 콘솔 AWS CLI, 또는 를 사용하여 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터에 읽기 전용 복제본을 추가할 수 있습니다 ElastiCache API.

관련 주제

- ElastiCache 클러스터에 노드 추가
- 복제 그룹에 읽기 전용 복제본 추가(AWS CLI)
- 를 사용하여 복제 그룹에 읽기 전용 복제본 추가 API

복제 그룹에 읽기 전용 복제본 추가(AWS CLI)

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹에 읽기 전용 복제본을 추가하려면 파라미터와 함께 명령을 사용하여 AWS CLI `create-cache-cluster` 클러스터(노드)를 추가할 복제 그룹을 `--replication-group-id` 지정합니다.

다음 예제에서는 `my-read replica`클러스터를 생성하고 해당 클러스터를 `my-replication-group` 복제 그룹에 추가합니다. 읽기 전용 복제본의 노드 유형, 파라미터 그룹, 보안 그룹, 유지 관리 기간 및 기타 설정이 `my-replication-group`의 다른 노드와 동일해집니다.

Linux, macOS, Unix의 경우:

```
aws elasticache create-cache-cluster \
      --cache-cluster-id my-read-replica \
      --replication-group-id my-replication-group
```

Windows의 경우:

```
aws elasticache create-cache-cluster ^
      --cache-cluster-id my-read-replica ^
```

```
    --replication-group-id my-replication-group
```

를 사용하여 읽기 전용 복제본을 추가하는 방법에 대한 자세한 내용은 섹션을 CLI참조하세요.create-cache-cluster Amazon ElastiCache 명령줄 참조의 .

를 사용하여 복제 그룹에 읽기 전용 복제본 추가 API

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹에 읽기 전용 복제본을 추가하려면 파라미터와 함께 작업을 사용하여 ElastiCache `CreateCacheCluster` 클러스터(노드)를 추가할 복제 그룹을 `ReplicationGroupId` 지정합니다.

다음 예제에서는 `myReadReplica`클러스터를 생성하고 해당 클러스터를 `myReplicationGroup` 복제 그룹에 추가합니다. 읽기 전용 복제본의 노드 유형, 파라미터 그룹, 보안 그룹, 유지 관리 기간 및 기타 설정이 `myReplicationGroup`의 다른 노드와 동일해집니다.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=CreateCacheCluster
    &CacheClusterId=myReadReplica
    &ReplicationGroupId=myReplicationGroup
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

를 사용하여 읽기 전용 복제본을 추가하는 방법에 대한 자세한 내용은 섹션을 API참조하세요.CreateCacheCluster Amazon ElastiCache API 참조의 .

Valkey 또는 Redis에 대한 읽기 전용 복제본 삭제OSS(클러스터 모드 비활성화됨)

다음 주제의 정보는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹에만 적용됩니다.

Valkey 또는 Redis OSS 복제 그룹의 읽기 트래픽이 변경되면 읽기 전용 복제본을 추가하거나 제거할 수 있습니다. 복제 그룹에서 노드를 제거하는 것은 클러스터를 삭제하는 것과 동일하지만 다음과 같은 제한이 있습니다.

- 복제 그룹에서 기본을 제거할 수 없습니다. 기본을 삭제하려면 다음을 수행하세요.

  1. 읽기 전용 복제본을 기본으로 승격합니다. 기본으로 읽기 전용 복제본 승격에 대한 자세한 내용은 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹의 읽기 전용 복제본을 기본 복제본으로 승격을 참조하세요.

2. 이전 기본을 삭제합니다. 이 메서드에 대한 제한 사항은 다음 요점을 참조하세요.

- 복제 그룹에서 다중 AZ가 활성화된 경우 이 복제 그룹에서 마지막 읽기 전용 복제본을 제거할 수 없습니다. 이 경우 다음과 같이 합니다.

1. 복제 그룹을 수정하여 다중 AZ를 비활성화합니다. 자세한 내용은 복제 그룹 수정 단원을 참조하십시오.
2. 읽기 전용 복제본을 삭제합니다.

ElastiCache 콘솔, AWS CLI 용 또는 를 사용하여 Valkey ElastiCache또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹에서 읽기 전용 복제본을 제거할 수 있습니다 ElastiCache API.

Valkey 또는 Redis OSS 복제 그룹에서 클러스터를 삭제하는 방법에 대한 지침은 다음을 참조하세요.

- 사용 AWS Management Console
- AWS CLI 를 사용하여 ElastiCache 클러스터 삭제
- 사용 ElastiCache API
- Valkey 또는 Redis에서 클러스터 크기 조정OSS(클러스터 모드 활성화됨)
- 샤드의 복제본 수 줄이기

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹의 읽기 전용 복제본을 기본 복제본으로 승격

다음 주제의 정보는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹에만 적용됩니다.

AWS Management Console, 또는 를 사용하여 Valkey AWS CLI또는 RedisOSS(클러스터 모드 비활성화됨) 읽기 전용 복제본을 기본 복제본으로 승격할 수 있습니다 ElastiCache API. 자동 장애 조치가 포함된 다중 AZ가 복제 그룹에서 활성화되어 있는 동안에는 읽기 전용 복제본을 기본으로 승격할 수 없습니다. 다중 AZ 활성화 복제 그룹의 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제본을 기본 복제본으로 승격하려면 다음을 수행합니다.

1. 다중 AZ를 비활성하도록 복제 그룹을 수정합니다(수정할 때 모든 클러스터가 동일 가용 영역에 있을 필요는 없음). 자세한 내용은 복제 그룹 수정 단원을 참조하십시오.
2. 읽기 전용 복제본을 기본으로 승격합니다.
3. 다중 AZ를 다시 활성화하도록 복제 그룹을 수정합니다.

Redis 2.6.13 이하를 실행하는 복제 그룹에서는 다중 OSS AZ를 사용할 수 없습니다.

사용 AWS Management Console

다음 절차에서는 콘솔을 사용해 복제본 노드를 기본 노드로 승격합니다.

읽기 전용 복제본을 기본으로 승격하려면(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 승격하려는 복제본이 다중 AZ가 활성화된 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹의 멤버인 경우 계속하기 전에 복제 그룹을 수정하여 다중 AZ를 비활성화합니다. 자세한 내용은 복제 그룹 수정 단원을 참조하십시오.
3. Valkey 또는 Redis OSS를 선택한 다음 클러스터 목록에서 수정하려는 복제 그룹을 선택합니다. 이 복제 그룹은 "Clustered Redis" 엔진이 아닌 "Redis" 엔진에서 실행되어야 하며, 2개 이상의 노드가 있어야 합니다.
4. 노드 목록에서 기본으로 승격할 복제본 노드를 선택한 후 작업에서 Promote(승격)를 선택합니다.
5. Promote Read Replica(읽기 전용 복제본 승격) 대화 상자에서 다음을 수행합니다.

    a. Apply Immediately(즉시 적용)에서 예를 선택하여 읽기 전용 복제본을 즉시 승격하거나 아니요를 선택하여 클러스터의 다음 번 유지 관리 기간에 승격합니다.

b. [Promote]를 선택하여 읽기 전용 복제본을 승격하거나 [Cancel]을 선택하여 작업을 취소합니다.

6. 승격 프로세스를 시작하기 전에 클러스터에 다중 AZ가 활성화된 경우 복제 그룹의 상태가 사용 가능으로 될 때까지 기다린 후 클러스터를 수정하여 다중 AZ를 재활성화합니다. 자세한 내용은 복제 그룹 수정 단원을 참조하십시오.

사용 AWS CLI

복제 그룹에 다중 AZ가 활성화되어 있으면 읽기 전용 복제본을 기본으로 승격할 수 없습니다. 일부 경우에 승격하려는 복제본은 다중 AZ가 활성화되어 있는 복제 그룹의 일원일 수 있습니다. 이러한 경우 계속하기 전에 다중 AZ를 비활성화하도록 복제 그룹을 수정해야 합니다. 수정할 때 모든 클러스터가 동일 가용 영역에 있을 필요는 없습니다. 복제 그룹 수정에 대한 자세한 내용은 복제 그룹 수정을 참조하세요.

다음 AWS CLI 명령은 복제 그룹 를 수정`sample-repl-group`하여 읽기 전용 복제본`my-replica-1`을 복제 그룹의 기본으로 만듭니다.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-replication-group \
   --replication-group-id sample-repl-group \
   --primary-cluster-id my-replica-1
```

Windows의 경우:

```
aws elasticache modify-replication-group ^
   --replication-group-id sample-repl-group ^
   --primary-cluster-id my-replica-1
```

복제 그룹 수정에 대한 자세한 내용은 섹션을 참조하세요.modify-replication-group Amazon ElastiCache 명령줄 참조의 .

사용 ElastiCache API

복제 그룹에 다중 AZ가 활성화되어 있으면 읽기 전용 복제본을 기본으로 승격할 수 없습니다. 일부 경우에 승격하려는 복제본은 다중 AZ가 활성화되어 있는 복제 그룹의 일원일 수 있습니다. 이러한 경우 계속하기 전에 다중 AZ를 비활성화하도록 복제 그룹을 수정해야 합니다. 수정할 때 모든 클러스터가 동일 가용 영역에 있을 필요는 없습니다. 복제 그룹 수정에 대한 자세한 내용은 복제 그룹 수정을 참조하세요.

다음 ElastiCache API 작업은 복제 그룹 를 수정myReplGroup하여 읽기 전용 복제본myReplica-1을 복제 그룹의 기본으로 만듭니다.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=ModifyReplicationGroup
   &ReplicationGroupId=myReplGroup
   &PrimaryClusterId=myReplica-1
   &Version=2014-12-01
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20141201T220302Z
   &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
   &X-Amz-Date=20141201T220302Z
   &X-Amz-SignedHeaders=Host
   &X-Amz-Expires=20141201T220302Z
   &X-Amz-Credential=<credential>
   &X-Amz-Signature=<signature>
```

복제 그룹 수정에 대한 자세한 내용은 섹션을 참조하세요.ModifyReplicationGroup Amazon ElastiCache API 참조의 .

ElastiCache 클러스터 유지 관리

모든 클러스터에는 시스템 변경 내용이 적용되는 주 단위 유지 관리 기간이 있습니다. Valkey 및 Redis OSS를 사용하면 복제 그룹에 동일한 주간 유지 관리 기간이 있습니다. 클러스터 또는 복제 그룹을 생성하거나 수정할 때 기본 유지 관리 기간을 지정하지 않으면 는 무작위로 선택한 요일에 리전의 유지 관리 기간 내에 60분의 유지 관리 기간을 ElastiCache 할당합니다.

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	아시아 태평양(도쿄) 리전	13:00~21:00 UTC
ap-northeast-2	Asia Pacific (Seoul) Region	12:00~20:00 UTC
ap-northeast-3	Asia Pacific (Osaka) Region	12:00~20:00 UTC
ap-southeast-3	Asia Pacific (Jakarta) Region	14:00~22:00 UTC

리전 코드	리전 이름	리전 유지 관리 기간
ap-south-1	Asia Pacific (Mumbai) Region	17:30~1:30 UTC
ap-southeast-1	아시아 태평양(싱가포르) 리전	14:00~22:00 UTC
cn-north-1	중국(베이징) 리전	14:00~22:00 UTC
cn-northwest-1	중국(닝샤) 리전	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
eu-west-3	EU(파리) 리전	23:59~07:29 UTC
af-south-1	아프리카(케이프타운) 리전	13:00~21:00 UTC
eu-central-1	Europe (Frankfurt) Region	23:00~07:00 UTC
eu-west-1	Europe (Ireland) Region	22:00~06:00 UTC
eu-west-2	Europe (London) Region	23:00~07:00 UTC
me-south-1	Middle East (Bahrain) Region	13:00~21:00 UTC
me-central-1	중동(UAE) 리전	13:00~21:00 UTC
eu-south-1	Europe (Milan) Region	21:00~05:00 UTC
sa-east-1	South America (São Paulo) Region	01:00~09:00 UTC
us-east-1	미국 동부(버지니아 북부) 리전	03:00~11:00 UTC
us-east-2	US East (Ohio) Region	04:00~12:00 UTC
us-gov-west-1	AWS GovCloud (US) 리전	06:00~14:00 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

클러스터 또는 복제 그룹의 유지 관리 기간 변경

유지 관리 기간은 사용률이 가장 낮은 시간에 할당되어야 하므로 수시로 수정되어야 할 수 있습니다. 클러스터나 복제 그룹을 수정하여 요청한 유지 관리 활동이 이루어지는 기간을 최대 24시간까지 지정할 수 있습니다. 이 시간 동안 사용자가 요청한 지연된 또는 대기 중인 클러스터 수정이 발생합니다.

> **Note**
>
> 지금 적용 상자를 선택하여 노드 유형 수정 및/또는 엔진 업그레이드를 즉시 적용하려면 를 AWS Management Console 선택합니다. 그러지 않으면 이러한 수정 사항은 다음 예약된 유지 관리 기간에 적용됩니다. 를 사용하려면 modify-replication-group 또는 섹션을 API참조하세요modify-cache-cluster.

추가 정보

유지 관리 기간 및 노드 대체에 대한 자세한 내용은 다음을 참조하세요.

- ElastiCache 유지 관리 -FAQ 유지 관리 및 노드 교체 시
- 노드 교체(Memcached)—Memcached에 대한 노드 교체 관리
- ElastiCache 클러스터 수정 - 클러스터의 유지 관리 기간 변경
- 노드 교체(Valkey 및 RedisOSS) - 노드 교체 관리
- 복제 그룹 수정 - 복제 그룹의 유지 관리 기간 변경

파라미터 그룹을 사용하여 엔진 ElastiCache 파라미터 구성

Amazon ElastiCache 은 파라미터를 사용하여 노드 및 클러스터의 런타임 속성을 제어합니다. 일반적으로 최신 엔진 버전에는 새로운 기능을 기원하는 추가 파라미터가 포함됩니다. Memcached 파라미터 표는 섹션을 참조하세요Memcached 특정 파라미터. Valkey 및 Redis OSS 파라미터 표는 섹션을 참조하세요Valkey 및 Redis OSS 파라미터.

`maxmemory`와 같은 일부 파라미터 값은 엔진 및 노드 유형에 의해 결정됩니다. 노드 유형별 이러한 Memcached 파라미터 값의 표는 섹션을 참조하세요Memcached 노드 유형별 파라미터. 노드 유형별 이러한 Valkey 및 Redis OSS 파라미터 값의 표는 섹션을 참조하세요OSS 노드 유형별 파라미터 재정의.

> **Note**
>
> Memcached 특정 파라미터 목록은 Memcached 특정 파라미터를 참조하세요.

주제

- 의 파라미터 관리 ElastiCache
- 의 캐시 파라미터 그룹 계층 ElastiCache
- ElastiCache 파라미터 그룹 생성
- 이름으로 ElastiCache 파라미터 그룹 나열
- ElastiCache 파라미터 그룹의 값 나열
- ElastiCache 파라미터 그룹 수정
- ElastiCache 파라미터 그룹 삭제
- 엔진별 파라미터

의 파라미터 관리 ElastiCache

ElastiCache 파라미터는 명명된 파라미터 그룹으로 그룹화되어 파라미터 관리가 더 쉬워집니다. 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.
- Memcached의 경우 파라미터 그룹 패밀리와 할당하려는 클러스터가 호환되어야 합니다. 예를 들어 클러스터에서 Memcached 버전 1.4.8을 실행 중이라면 Memcached 1.4 패밀리의 기본 또는 사용자 지정 파라미터 그룹만 사용할 수 있습니다.

  Redis의 경우 파라미터 그룹 패밀리와 할당하려는 클러스터가 호환되어야 합니다. 예를 들어 클러스터가 Redis OSS 버전 3.2.10을 실행하는 경우 Redis OSS 3.2 패밀리의 기본 또는 사용자 지정 파라미터 그룹만 사용할 수 있습니다.
- 클러스터의 파라미터 그룹을 변경하면 조건부로 수정 가능한 파라미터의 값이 현재 및 새 파라미터 그룹에서 동일해야 합니다.
- Memcached의 경우 클러스터의 파라미터를 변경하면 변경 사항이 클러스터에 즉시 적용됩니다. 이는 클러스터의 파라미터 그룹 자체에서 변경하든 파라미터 값을 클러스터의 파라미터 그룹 내에서 변경하든 마찬가지입니다. 특정 파라미터 변경 사항이 적용되는 시점을 확인하려면 Memcached 특정 파라미터에 대한 테이블의 변경 적용 열을 참조하세요. 클러스터의 노드 재부팅에 관한 자세한 정보는 클러스터 재부팅을 참조하세요.
- Redis의 경우 클러스터의 파라미터를 변경하면 클러스터 노드가 재부팅된 후 변경 사항이 클러스터에 즉시 적용되거나 다음 예외가 적용됩니다. 이는 클러스터의 파라미터 그룹 자체에서 변경하든 파라미터 값을 클러스터의 파라미터 그룹 내에서 변경하든 마찬가지입니다. 특정 파라미터 변경 사항이 적용되는 시점을 확인하려면 Valkey 및 Redis OSS 파라미터에 대한 테이블의 변경 적용 열을 참조하세요.

  Valkey 또는 Redis OSS 노드 재부팅에 대한 자세한 내용은 섹션을 참조하세요노드 재부팅.

> **Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 파라미터 변경 사항**
>
> Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터에서 다음 파라미터를 변경하는 경우 다음 단계를 따릅니다.

- activerehashing
- 데이터베이스

1. 클러스터의 수동 백업을 만듭니다. 수동 백업 지원을 참조하세요.
2. 클러스터를 삭제합니다. 클러스터 삭제를 참조하세요.
3. 변경한 파라미터 그룹 및 백업을 사용해 클러스터를 저장하여 새로운 클러스터를 시드합니다. 백업에서 새 캐시로 복원을 참조하세요.

다른 파라미터를 변경한 경우에는 필요하지 않습니다.

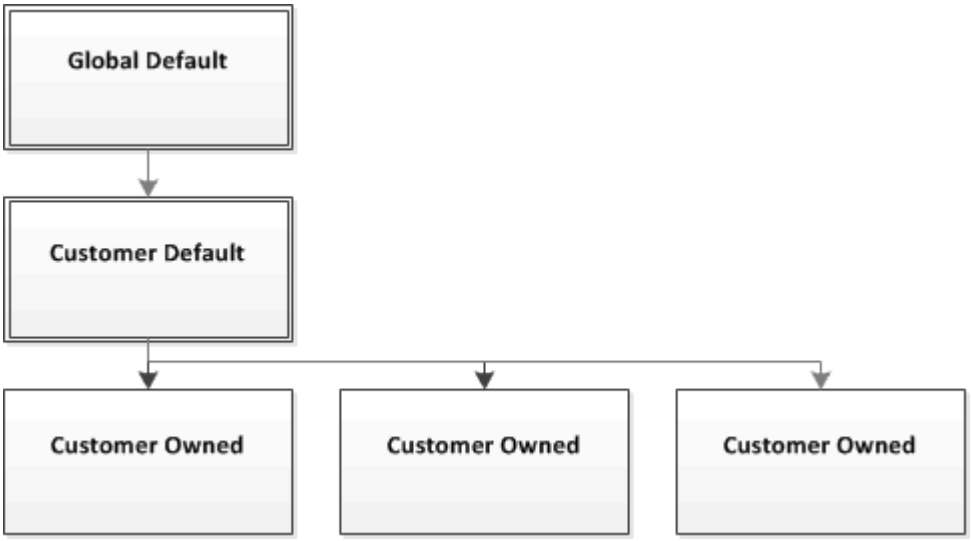
- 파라미터 그룹을 Valkey 및 Redis OSS 글로벌 데이터 스토어와 연결할 수 있습니다. 글로벌 데이터 스토어는 AWS 리전에 걸쳐 있는 하나 이상의 클러스터 모음입니다. 이 경우 파라미터 그룹은 글로벌 데이터 스토어를 구성하는 모든 클러스터에서 공유됩니다. 기본 클러스터의 파라미터 그룹에 대한 모든 수정 사항은 글로벌 데이터 스토어의 나머지 모든 클러스터에 복제됩니다. 자세한 내용은 글로벌 데이터 스토어를 사용하여 AWS 리전 간 복제 단원을 참조하십시오.

  다음 위치를 보면 파라미터 그룹이 글로벌 데이터 스토어의 일부인지 확인할 수 있습니다.

  - 파라미터 그룹 페이지의 ElastiCache 콘솔에서 예/아니요 전역 속성
  - CacheParameterGroup API 작업의 예/아니요 `IsGlobal` 속성

의 캐시 파라미터 그룹 계층 ElastiCache

Amazon ElastiCache 에는 다음과 같이 세 가지 계층의 캐시 파라미터 그룹이 있습니다.



Amazon ElastiCache 파라미터 그룹 계층

전역 기본값

리전의 모든 Amazon ElastiCache 고객을 위한 최상위 루트 파라미터 그룹입니다.

전역 기본 캐시 파라미터 그룹:

- 예약되어 ElastiCache 있으며 고객이 사용할 수 없습니다.

고객 기본값

고객의 사용을 위해 생성된 전역 기본 캐시 파라미터 그룹의 사본입니다.

고객 기본 캐시 파라미터 그룹:

- 가 생성하고 소유합니다 ElastiCache.
- 고객이 이 캐시 파라미터 그룹에서 지원하는 엔진 버전을 실행 중인 모든 클러스터의 캐시 파라미터 그룹으로 사용할 수 있습니다.
- 고객이 편집할 수 없습니다.

고객 소유

고객 기본 캐시 파라미터 그룹의 사본입니다. 고객 소유 캐시 파라미터 그룹은 고객이 캐시 파라미터 그룹을 생성할 때마다 만들어집니다.

고객 소유 캐시 파라미터 그룹:

- 고객이 생성하고 소유합니다.
- 고객의 호환 가능한 모든 클러스터에 할당할 수 있습니다.
- 고객이 사용자 지정 캐시 파라미터 그룹을 생성하기 위해 수정할 수 있습니다.

  모든 파라미터 값을 수정할 수 있는 것은 아닙니다. Memcached 값에 대한 자세한 내용은 섹션을 참조하세요Memcached 특정 파라미터. Valkey 및 Redis OSS 값에 대한 자세한 내용은 섹션을 참조하세요Valkey 및 Redis OSS 파라미터.

ElastiCache 파라미터 그룹 생성

기본값에서 변경하려는 파라미터 값이 하나 이상이면 새 파라미터 그룹을 생성해야 합니다. ElastiCache 콘솔, AWS CLI또는 를 사용하여 파라미터 그룹을 생성할 수 있습니다 ElastiCache API.

ElastiCache 파라미터 그룹 생성(콘솔)

다음 절차에서는 콘솔을 사용하여 ElastiCache 파라미터 그룹을 생성하는 방법을 보여줍니다.

ElastiCache 콘솔을 사용하여 파라미터 그룹을 생성하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 사용 가능한 모든 파라미터 목록을 표시하려면 왼쪽 탐색 창에서 [Parameter Groups]를 선택합니다.
3. 파라미터 그룹을 생성하려면 [Create Parameter Group]을 선택합니다.

   파라미터 그룹 생성 화면이 나타납니다.
4. [Family] 목록에서 파라미터 그룹의 템플릿이 될 파라미터 그룹 패밀리를 선택합니다.

   memcached1.4 또는 redis3.2와 같은 파라미터 그룹 패밀리는 파라미터 그룹의 실제 파라미터와 초기 값을 정의합니다. 파라미터 그룹 패밀리는 클러스터의 엔진 및 버전과 일치해야 합니다.
5. [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.

6. [Description] 상자에 파라미터 그룹에 대한 설명을 입력합니다.
7. 파라미터 그룹을 생성하려면 [Create]를 선택합니다.

   파라미터 그룹을 생성하지 않고 프로세스를 종료하려면 [Cancel]을 선택합니다.
8. 파라미터 그룹을 생성하면 패밀리의 기본값이 부여됩니다. 기본값을 변경하려면 파라미터 그룹을 수정해야 합니다. 자세한 내용은 ElastiCache 파라미터 그룹 수정 단원을 참조하십시오.

ElastiCache 파라미터 그룹 생성(AWS CLI)

를 사용하여 파라미터 그룹을 생성하려면 다음 파라미터와 `create-cache-parameter-group` 함께 명령을 AWS CLI사용합니다.

- `--cache-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.
- `--cache-parameter-group-family` - 파라미터 그룹의 엔진 및 버전 패밀리입니다.
- `--description` - 사용자가 정의한 파라미터 그룹에 대한 설명입니다.

Example

다음 예제에서는 memcachedmyMem1.4 패밀리를 템플릿으로 사용하여 14라는 파라미터 그룹을 생성합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache create-cache-parameter-group \
    --cache-parameter-group-name myMem14  \
    --cache-parameter-group-family memcached1.4 \
    --description "My first parameter group"
```

Windows의 경우:

```
aws elasticache create-cache-parameter-group ^
    --cache-parameter-group-name myMem14  ^
    --cache-parameter-group-family memcached1.4 ^
    --description "My first parameter group"
```

이 명령의 출력은 다음과 유사해야 합니다.

```
{
    "CacheParameterGroup": {
        "CacheParameterGroupName": "myMem14",
        "CacheParameterGroupFamily": "memcached1.4",
        "Description": "My first  parameter group"
    }
}
```

Example

다음 예제에서는 redismyRed2.8 패밀리를 템플릿으로 사용하여 28이라는 파라미터 그룹을 생성합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache create-cache-parameter-group \
    --cache-parameter-group-name myRed28  \
    --cache-parameter-group-family redis2.8 \
    --description "My first parameter group"
```

Windows의 경우:

```
aws elasticache create-cache-parameter-group ^
    --cache-parameter-group-name myRed28  ^
    --cache-parameter-group-family redis2.8 ^
    --description "My first parameter group"
```

이 명령의 출력은 다음과 유사해야 합니다.

```
{
    "CacheParameterGroup": {
        "CacheParameterGroupName": "myRed28",
        "CacheParameterGroupFamily": "redis2.8",
        "Description": "My first parameter group"
    }
}
```

파라미터 그룹을 생성하면 패밀리의 기본값이 부여됩니다. 기본값을 변경하려면 파라미터 그룹을 수정해야 합니다. 자세한 내용은 ElastiCache 파라미터 그룹 수정 단원을 참조하십시오.

자세한 내용은 `create-cache-parameter-group` 단원을 참조하십시오.

ElastiCache 파라미터 그룹 생성(ElastiCache API)

를 사용하여 파라미터 그룹을 생성하려면 다음 파라미터와 함께 `CreateCacheParameterGroup` 작업을 ElastiCache API사용합니다.

- `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.
- `CacheParameterGroupFamily` - 파라미터 그룹의 엔진 및 버전 패밀리입니다. 예: `memcached1.4`.
- `CacheParameterGroupFamily` - 파라미터 그룹의 엔진 및 버전 패밀리입니다. 예: `redis2.8`.
- `Description` - 사용자가 정의한 파라미터 그룹에 대한 설명입니다.

Example

다음 예제에서는 memcachedmyMem1.4 패밀리를 템플릿으로 사용하여 14라는 파라미터 그룹을 생성합니다.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=CreateCacheParameterGroup
   &CacheParameterGroupFamily=memcached1.4
   &CacheParameterGroupName=myMem14
   &Description=My%20first%20parameter%20group
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

이 작업의 응답은 다음과 유사해야 합니다.

```
<CreateCacheParameterGroupResponse xmlns="http://elasticache.amazonaws.com/
doc/2013-06-15/">
  <CreateCacheParameterGroupResult>
    <CacheParameterGroup>
      <CacheParameterGroupName>myMem14</CacheParameterGroupName>
      <CacheParameterGroupFamily>memcached1.4</CacheParameterGroupFamily>
      <Description>My first  parameter group</Description>
    </CacheParameterGroup>
  </CreateCacheParameterGroupResult>
  <ResponseMetadata>
    <RequestId>d8465952-af48-11e0-8d36-859edca6f4b8</RequestId>
  </ResponseMetadata>
</CreateCacheParameterGroupResponse>
```

Example

다음 예제에서는 redismyRed2.8 패밀리를 템플릿으로 사용하여 28이라는 파라미터 그룹을 생성합니다.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=CreateCacheParameterGroup
   &CacheParameterGroupFamily=redis2.8
   &CacheParameterGroupName=myRed28
   &Description=My%20first%20parameter%20group
```

```
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

이 작업의 응답은 다음과 유사해야 합니다.

```
<CreateCacheParameterGroupResponse xmlns="http://elasticache.amazonaws.com/
doc/2013-06-15/">
  <CreateCacheParameterGroupResult>
    <CacheParameterGroup>
      <CacheParameterGroupName>myRed28</CacheParameterGroupName>
      <CacheParameterGroupFamily>redis2.8</CacheParameterGroupFamily>
      <Description>My first parameter group</Description>
    </CacheParameterGroup>
  </CreateCacheParameterGroupResult>
  <ResponseMetadata>
    <RequestId>d8465952-af48-11e0-8d36-859edca6f4b8</RequestId>
  </ResponseMetadata>
</CreateCacheParameterGroupResponse>
```

파라미터 그룹을 생성하면 패밀리의 기본값이 부여됩니다. 기본값을 변경하려면 파라미터 그룹을 수정해야 합니다. 자세한 내용은 ElastiCache 파라미터 그룹 수정 단원을 참조하십시오.

자세한 내용은 `CreateCacheParameterGroup` 단원을 참조하십시오.

이름으로 ElastiCache 파라미터 그룹 나열

ElastiCache 콘솔, AWS CLI또는 를 사용하여 파라미터 그룹을 나열할 ElastiCache 수 있습니다API.

이름별로 파라미터 그룹 목록 조회(콘솔)

다음 절차에서는 ElastiCache 콘솔을 사용하여 파라미터 그룹 목록을 보는 방법을 보여줍니다.

ElastiCache 콘솔을 사용하여 파라미터 그룹을 나열하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 사용 가능한 모든 파라미터 목록을 표시하려면 왼쪽 탐색 창에서 [Parameter Groups]를 선택합니다.

이름별 ElastiCache 파라미터 그룹 나열(AWS CLI)

를 사용하여 파라미터 그룹 목록을 생성하려면 명령 을 AWS CLI사용합니다`describe-cache-parameter-groups`. 파라미터 그룹의 이름을 입력하면 그 파라미터 그룹만 나열됩니다. 파라미터 그룹의 이름을 입력하지 않으면 최대 `--max-records`개의 파라미터 그룹이 나열됩니다. 두 경우 모두 파라미터 그룹의 이름, 패밀리 및 설명이 나열됩니다.

Example

다음 샘플 코드는 파라미터 그룹 myMem14를 나열합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache describe-cache-parameter-groups \
    --cache-parameter-group-name myMem14
```

Windows의 경우:

```
aws elasticache describe-cache-parameter-groups ^
    --cache-parameter-group-name myMem14
```

이 명령의 출력은 파라미터 그룹의 이름, 패밀리 및 설명을 나열하는 것과 같습니다.

```
{
```

```
    "CacheParameterGroups": [
     {
         "CacheParameterGroupName": "myMem14",
         "CacheParameterGroupFamily": "memcached1.4",
         "Description": "My first parameter group"
     }
    ]
}
```

Example

다음 샘플 코드는 파라미터 그룹 myRed28을 나열합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache describe-cache-parameter-groups \
    --cache-parameter-group-name myRed28
```

Windows의 경우:

```
aws elasticache describe-cache-parameter-groups ^
    --cache-parameter-group-name myRed28
```

이 명령의 출력은 파라미터 그룹의 이름, 패밀리 및 설명을 나열하는 것과 같습니다.

```
{
    "CacheParameterGroups": [
     {
         "CacheParameterGroupName": "myRed28",
         "CacheParameterGroupFamily": "redis2.8",
         "Description": "My first parameter group"
     }
    ]
}
```

Example

다음 샘플 코드는 Redis OSS 엔진 버전 myRed5.0.6 이상에서 실행되는 파라미터 그룹에 대한 파라미터 그룹 56을 나열합니다. 파라미터 그룹이 글로벌 데이터 스토어를 사용하여 AWS 리전 간 복제의 일부인 경우 출력에서 반환되는 `IsGlobal` 속성 값은 `Yes`가 됩니다.

Linux, macOS, Unix의 경우:

```
aws elasticache describe-cache-parameter-groups \
    --cache-parameter-group-name myRed56
```

Windows의 경우:

```
aws elasticache describe-cache-parameter-groups ^
    --cache-parameter-group-name myRed56
```

이 명령의 출력은 파라미터 그룹의 이름, 패밀리 isGlobal 및 설명을 나열하는 다음과 같습니다.

```
{
    "CacheParameterGroups": [
     {
         "CacheParameterGroupName": "myRed56",
         "CacheParameterGroupFamily": "redis5.0",
         "Description": "My first parameter group",
         "IsGlobal": "yes"
     }
    ]
}
```

Example

다음 샘플 코드는 최대 10개의 파라미터 그룹을 나열합니다.

```
aws elasticache describe-cache-parameter-groups --max-records 10
```

이 명령의 JSON 출력은 각 파라미터 그룹에 대해 이름, 패밀리, 설명, 그리고 redis5.6의 경우 파라미터 그룹이 글로벌 데이터 스토어(isGlobal)의 일부인지 여부를 나열하는 다음과 같습니다.

```
{
    "CacheParameterGroups": [
        {
            "CacheParameterGroupName": "custom-redis32",
            "CacheParameterGroupFamily": "redis3.2",
            "Description": "custom parameter group with reserved-memory > 0"
        },
        {
```

```
            "CacheParameterGroupName": "default.memcached1.4",
            "CacheParameterGroupFamily": "memcached1.4",
            "Description": "Default parameter group for memcached1.4"
        },
        {
            "CacheParameterGroupName": "default.redis2.6",
            "CacheParameterGroupFamily": "redis2.6",
            "Description": "Default parameter group for redis2.6"
        },
        {
            "CacheParameterGroupName": "default.redis2.8",
            "CacheParameterGroupFamily": "redis2.8",
            "Description": "Default parameter group for redis2.8"
        },
        {
            "CacheParameterGroupName": "default.redis3.2",
            "CacheParameterGroupFamily": "redis3.2",
            "Description": "Default parameter group for redis3.2"
        },
        {
            "CacheParameterGroupName": "default.redis3.2.cluster.on",
            "CacheParameterGroupFamily": "redis3.2",
            "Description": "Customized default parameter group for redis3.2 with
 cluster mode on"
        },
        {
            "CacheParameterGroupName": "default.redis5.6.cluster.on",
            "CacheParameterGroupFamily": "redis5.0",
            "Description": "Customized default parameter group for redis5.6 with
 cluster mode on",
            "isGlobal": "yes"
        },
    ]
}
```

자세한 내용은 [describe-cache-parameter-groups](#) 단원을 참조하십시오.

이름별 ElastiCache 파라미터 그룹 나열(ElastiCache API)

를 사용하여 파라미터 그룹 목록을 생성하려면 `DescribeCacheParameterGroups` 작업을 ElastiCache API사용합니다. 파라미터 그룹의 이름을 입력하면 그 파라미터 그룹만 나열됩니다. 파라미터 그룹의 이름을 입력하지 않으면 최대 `MaxRecords`개의 파라미터 그룹이 나열됩니다. 두 경우 모두 파라미터 그룹의 이름, 패밀리 및 설명이 나열됩니다.

Example

다음 샘플 코드는 파라미터 그룹 myMem14를 나열합니다.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheParameterGroups
   &CacheParameterGroupName=myMem14
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

이 작업의 응답은 각 파라미터 그룹의 이름, 패밀리 및 설명을 나열하는 것과 같습니다.

```
<DescribeCacheParameterGroupsResponse xmlns="http://elasticache.amazonaws.com/
doc/2013-06-15/">
  <DescribeCacheParameterGroupsResult>
    <CacheParameterGroups>
      <CacheParameterGroup>
        <CacheParameterGroupName>myMem14</CacheParameterGroupName>
        <CacheParameterGroupFamily>memcached1.4</CacheParameterGroupFamily>
        <Description>My custom Memcached 1.4 parameter group</Description>
      </CacheParameterGroup>
    </CacheParameterGroups>
  </DescribeCacheParameterGroupsResult>
  <ResponseMetadata>
    <RequestId>3540cc3d-af48-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeCacheParameterGroupsResponse>
```

Example

다음 샘플 코드는 최대 10개의 파라미터 그룹을 나열합니다.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheParameterGroups
   &MaxRecords=10
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
```

```
   &X-Amz-Credential=<credential>
```

이 작업의 응답은 각 파라미터 그룹에 대해 이름, 패밀리, 설명, 그리고 파라미터 그룹이 글로벌 데이터 스토어(isGlobal)에 속하는 경우 redis5.6을 나열하는 것과 같습니다.

```
<DescribeCacheParameterGroupsResponse xmlns="http://elasticache.amazonaws.com/
doc/2013-06-15/">
  <DescribeCacheParameterGroupsResult>
    <CacheParameterGroups>
      <CacheParameterGroup>
        <CacheParameterGroupName>myRedis28</CacheParameterGroupName>
        <CacheParameterGroupFamily>redis2.8</CacheParameterGroupFamily>
        <Description>My custom Redis 2.8 parameter group</Description>
      </CacheParameterGroup>
      <CacheParameterGroup>
        <CacheParameterGroupName>myMem14</CacheParameterGroupName>
        <CacheParameterGroupFamily>memcached1.4</CacheParameterGroupFamily>
        <Description>My custom Memcached 1.4 parameter group</Description>
      </CacheParameterGroup>
       <CacheParameterGroup>
        <CacheParameterGroupName>myRedis56</CacheParameterGroupName>
        <CacheParameterGroupFamily>redis5.0</CacheParameterGroupFamily>
        <Description>My custom redis 5.6 parameter group</Description>
        <isGlobal>yes</isGlobal>
      </CacheParameterGroup>
    </CacheParameterGroups>
  </DescribeCacheParameterGroupsResult>
  <ResponseMetadata>
    <RequestId>3540cc3d-af48-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeCacheParameterGroupsResponse>
```

Example

다음 샘플 코드는 파라미터 그룹 myRed28을 나열합니다.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheParameterGroups
   &CacheParameterGroupName=myRed28
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
```

```
   &X-Amz-Credential=<credential>
```

이 작업의 응답은 각 파라미터 그룹의 이름, 패밀리 및 설명을 나열하는 것과 같습니다.

```
<DescribeCacheParameterGroupsResponse xmlns="http://elasticache.amazonaws.com/
doc/2013-06-15/">
  <DescribeCacheParameterGroupsResult>
    <CacheParameterGroups>
      <CacheParameterGroup>
        <CacheParameterGroupName>myRed28</CacheParameterGroupName>
        <CacheParameterGroupFamily>redis2.8</CacheParameterGroupFamily>
        <Description>My custom Redis 2.8 parameter group</Description>
      </CacheParameterGroup>
    </CacheParameterGroups>
  </DescribeCacheParameterGroupsResult>
  <ResponseMetadata>
    <RequestId>3540cc3d-af48-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeCacheParameterGroupsResponse>
```

Example

다음 샘플 코드는 파라미터 그룹 myRed56을 나열합니다.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheParameterGroups
   &CacheParameterGroupName=myRed56
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

이 작업의 응답은 이름, 패밀리, 설명 및 파라미터 그룹이 글로벌 데이터 스토어()의 일부인지 여부를 나열하는 다음과 같습니다isGlobal.

```
<DescribeCacheParameterGroupsResponse xmlns="http://elasticache.amazonaws.com/
doc/2013-06-15/">
  <DescribeCacheParameterGroupsResult>
    <CacheParameterGroups>
      <CacheParameterGroup>
        <CacheParameterGroupName>myRed56</CacheParameterGroupName>
```

```
        <CacheParameterGroupFamily>redis5.0</CacheParameterGroupFamily>
        <Description>My custom Redis 5.6 parameter group</Description>
        <isGlobal>yes</isGlobal>
      </CacheParameterGroup>
    </CacheParameterGroups>
  </DescribeCacheParameterGroupsResult>
  <ResponseMetadata>
    <RequestId>3540cc3d-af48-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeCacheParameterGroupsResponse>
```

자세한 내용은 DescribeCacheParameterGroups 단원을 참조하십시오.

ElastiCache 파라미터 그룹의 값 나열

콘솔, AWS CLI또는 를 사용하여 ElastiCache 파라미터 그룹의 파라미터와 해당 값을 나열할 수 있습니다 ElastiCache API.

ElastiCache 파라미터 그룹의 값 나열(콘솔)

다음 절차에서는 ElastiCache 콘솔을 사용하여 파라미터 그룹의 파라미터와 해당 값을 나열하는 방법을 보여줍니다.

콘솔을 사용하여 ElastiCache 파라미터 그룹의 파라미터와 해당 값을 나열하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 사용 가능한 모든 파라미터 목록을 표시하려면 왼쪽 탐색 창에서 [Parameter Groups]를 선택합니다.
3. 파라미터 그룹의 이름 왼쪽에 있는 확인란을 선택하여 파라미터 및 값을 나열할 파라미터 그룹을 선택합니다.

   파라미터 및 그 값이 화면 하단에 나열됩니다. 파라미터의 수가 많으면 위아래로 스크롤하여 원하는 파라미터를 찾습니다.

파라미터 그룹 값 나열(AWS CLI)

를 사용하여 파라미터 그룹의 파라미터와 해당 값을 나열하려면 명령 을 AWS CLI사용합니다`describe-cache-parameters`.

Example

다음 샘플 코드는 파라미터 그룹 myMem14에 대한 모든 Memcached 파라미터와 해당 값을 나열합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache describe-cache-parameters \
    --cache-parameter-group-name myMem14
```

Windows의 경우:

```
aws elasticache describe-cache-parameters ^
```

```
    --cache-parameter-group-name myMem14
```

Example

다음 샘플 코드는 파라미터 그룹 myRedis28에 대한 모든 파라미터와 해당 값을 나열합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache describe-cache-parameters \
    --cache-parameter-group-name myRedis28
```

Windows의 경우:

```
aws elasticache describe-cache-parameters ^
    --cache-parameter-group-name myRed28
```

자세한 내용은 describe-cache-parameters 단원을 참조하십시오.

파라미터 그룹 값 나열(ElastiCache API)

를 사용하여 파라미터 그룹의 파라미터와 해당 값을 나열하려면 DescribeCacheParameters 작업을 ElastiCache API사용합니다.

Example

다음 샘플 코드는 파라미터 그룹 myMem14에 대한 모든 Memcached 파라미터를 나열합니다.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheParameters
   &CacheParameterGroupName=myMem14
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

이 작업의 응답은 다음과 유사합니다. 응답이 잘렸습니다.

```
<DescribeCacheParametersResponse xmlns="http://elasticache.amazonaws.com/
doc/2013-06-15/">
  <DescribeCacheParametersResult>
```

```
    <CacheClusterClassSpecificParameters>
      <CacheNodeTypeSpecificParameter>
        <DataType>integer</DataType>
        <Source>system</Source>
        <IsModifiable>false</IsModifiable>
        <Description>The maximum configurable amount of memory to use to store items,
 in megabytes.</Description>
        <CacheNodeTypeSpecificValues>
          <CacheNodeTypeSpecificValue>
            <Value>1000</Value>
            <CacheClusterClass>cache.c1.medium</CacheClusterClass>
          </CacheNodeTypeSpecificValue>
          <CacheNodeTypeSpecificValue>
            <Value>6000</Value>
            <CacheClusterClass>cache.c1.xlarge</CacheClusterClass>
          </CacheNodeTypeSpecificValue>
          <CacheNodeTypeSpecificValue>
            <Value>7100</Value>
            <CacheClusterClass>cache.m1.large</CacheClusterClass>
          </CacheNodeTypeSpecificValue>
          <CacheNodeTypeSpecificValue>
            <Value>1300</Value>
            <CacheClusterClass>cache.m1.small</CacheClusterClass>
          </CacheNodeTypeSpecificValue>

...output omitted...

    </CacheClusterClassSpecificParameters>
  </DescribeCacheParametersResult>
  <ResponseMetadata>
    <RequestId>6d355589-af49-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeCacheParametersResponse>
```

Example

다음 샘플 코드는 파라미터 그룹 myRed28의 모든 파라미터를 나열합니다.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheParameters
   &CacheParameterGroupName=myRed28
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
```

```
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

이 작업의 응답은 다음과 유사합니다. 응답이 잘렸습니다.

```
<DescribeCacheParametersResponse xmlns="http://elasticache.amazonaws.com/
doc/2013-06-15/">
  <DescribeCacheParametersResult>
    <CacheClusterClassSpecificParameters>
      <CacheNodeTypeSpecificParameter>
        <DataType>integer</DataType>
        <Source>system</Source>
        <IsModifiable>false</IsModifiable>
        <Description>The maximum configurable amount of memory to use to store items,
 in megabytes.</Description>
        <CacheNodeTypeSpecificValues>
          <CacheNodeTypeSpecificValue>
            <Value>1000</Value>
            <CacheClusterClass>cache.c1.medium</CacheClusterClass>
          </CacheNodeTypeSpecificValue>
          <CacheNodeTypeSpecificValue>
            <Value>6000</Value>
            <CacheClusterClass>cache.c1.xlarge</CacheClusterClass>
          </CacheNodeTypeSpecificValue>
          <CacheNodeTypeSpecificValue>
            <Value>7100</Value>
            <CacheClusterClass>cache.m1.large</CacheClusterClass>
          </CacheNodeTypeSpecificValue>
          <CacheNodeTypeSpecificValue>
            <Value>1300</Value>
            <CacheClusterClass>cache.m1.small</CacheClusterClass>
          </CacheNodeTypeSpecificValue>

...output omitted...

    </CacheClusterClassSpecificParameters>
  </DescribeCacheParametersResult>
  <ResponseMetadata>
    <RequestId>6d355589-af49-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeCacheParametersResponse>
```

자세한 내용은 DescribeCacheParameters 단원을 참조하십시오.

ElastiCache 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. 파라미터 값 변경이 파라미터 그룹에 적용되는 시기에 대한 자세한 내용은 Valkey 및 Redis OSS 파라미터 및 섹션을 참조하세요Memcached 특정 파라미터.

파라미터 그룹 수정(콘솔)

다음 절차에서는 ElastiCache 콘솔을 사용하여 `cluster-enabled` 파라미터 값을 변경하는 방법을 보여줍니다. 동일한 절차를 통해 모든 파라미터 값을 변경할 수 있습니다.

ElastiCache 콘솔을 사용하여 파라미터 값을 변경하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 사용 가능한 모든 파라미터 목록을 표시하려면 왼쪽 탐색 창에서 [Parameter Groups]를 선택합니다.
3. 파라미터 그룹의 이름 왼쪽에 있는 확인란을 선택하여 수정할 파라미터 그룹을 선택합니다.

   파라미터 그룹의 파라미터가 화면 하단에 나열됩니다. 모든 파라미터를 보려면 목록 페이지를 탐색해야 할 수도 있습니다.
4. 파라미터를 하나 이상 수정하려면 [Edit Parameters]를 선택합니다.
5. 파라미터 그룹 편집: 화면에서 왼쪽 및 오른쪽 화살표로 스크롤하여 `binding_protocol` 파라미터를 찾은 다음 값 열에 `ascii`를 입력합니다.
6. 변경 사항 저장을 선택합니다.
7. Memcached의 경우 변경한 파라미터의 이름을 찾으려면 섹션을 참조하세요Memcached 특정 파라미터. 다시 시작한 후 파라미터에 변경 사항이 발생하면 이 파라미터 그룹을 사용하는 모든 클러스터를 재부팅합니다. 자세한 내용은 클러스터 재부팅을 참조하세요.
8. Valkey 및 Redis 에서 변경한 파라미터의 이름을 찾으OSS려면 섹션을 참조하세요Valkey 및 Redis OSS 파라미터. Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터가 있고 다음 파라미터를 변경하는 경우 클러스터의 노드를 재부팅해야 합니다.

   - activerehashing

- 데이터베이스

자세한 내용은 노드 재부팅을 참조하세요.

> Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 파라미터 변경 사항
>
> Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터에서 다음 파라미터를 변경하는 경우 다음 단계를 따릅니다.
>
> - activerehashing
> - 데이터베이스
>
> 1. Redis를 사용하면 클러스터의 수동 백업을 연결할 수 있습니다. 수동 백업 지원을 참조하세요.
> 2. 클러스터를 삭제합니다. 클러스터 삭제를 참조하세요.
> 3. 변경한 파라미터 그룹 및 백업을 사용해 클러스터를 복원하여 새로운 클러스터를 시드합니다. 백업에서 새 캐시로 복원을 참조하세요.
>
> 다른 파라미터를 변경한 경우에는 필요하지 않습니다.

파라미터 그룹 수정(AWS CLI)

를 사용하여 파라미터의 값을 변경하려면 명령 을 AWS CLI사용합니다`modify-cache-parameter-group`.

Example

Memcached를 사용하여 변경하려는 파라미터의 이름과 허용 값을 찾으려면 섹션을 참조하세요. Memcached 특정 파라미터

다음 예제 코드에서는 `myMem14` 파라미터 그룹에서 두 파라미터 chunk_size 및 chunk_size_growth_fact의 값을 설정합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-cache-parameter-group \
```

```
    --cache-parameter-group-name myMem14 \
    --parameter-name-values \
        ParameterName=chunk_size,ParameterValue=96 \
        ParameterName=chunk_size_growth_fact,ParameterValue=1.5
```

Windows의 경우:

```
aws elasticache modify-cache-parameter-group ^
    --cache-parameter-group-name myMem14 ^
    --parameter-name-values ^
        ParameterName=chunk_size,ParameterValue=96 ^
        ParameterName=chunk_size_growth_fact,ParameterValue=1.5
```

이 명령의 출력은 다음과 같습니다.

```
{
    "CacheParameterGroupName": "myMem14"
}
```

Example

Valkey 및 Redis 에서 변경하려는 파라미터의 이름과 허용 값을 찾으OSS려면 섹션을 참조하세요. Valkey 및 Redis OSS 파라미터

다음 샘플 코드는 두 파라미터의 값을 설정하고 파라미터 그룹 에서 reserved-memory-percent 클러스터를 활성화합니다`myredis32-on-30`. 파라미터 그룹을 Valkey 또는 Redisreserved-memory-percent`30`(클러스터 모드 활성화됨) 클러스터OSS(복제 그룹)와 함께 사용할 수 `yes` 있도록 를 (30%)로 설정하고 클러스터를 활성화했습니다.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-cache-parameter-group \
    --cache-parameter-group-name myredis32-on-30 \
    --parameter-name-values \
        ParameterName=reserved-memory-percent,ParameterValue=30 \
        ParameterName=cluster-enabled,ParameterValue=yes
```

Windows의 경우:

```
aws elasticache modify-cache-parameter-group ^
```

```
    --cache-parameter-group-name myredis32-on-30 ^
    --parameter-name-values ^
        ParameterName=reserved-memory-percent,ParameterValue=30 ^
        ParameterName=cluster-enabled,ParameterValue=yes
```

이 명령의 출력은 다음과 같습니다.

```
{
    "CacheParameterGroupName": "my-redis32-on-30"
}
```

자세한 내용은 modify-cache-parameter-group 단원을 참조하십시오.

변경한 파라미터의 이름을 찾으려면 Valkey 및 Redis OSS 파라미터 섹션을 참조하세요.

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터가 있고 다음 파라미터를 변경하는 경우 클러스터의 노드를 재부팅해야 합니다.

- activerehashing
- 데이터베이스

자세한 내용은 노드 재부팅을 참조하세요.

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 파라미터 변경 사항

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터에서 다음 파라미터를 변경하는 경우 다음 단계를 따릅니다.

- activerehashing
- 데이터베이스

1. 클러스터의 수동 백업을 만듭니다. 수동 백업 지원을 참조하세요.
2. 클러스터를 삭제합니다. 클러스터 삭제를 참조하세요.
3. 변경한 파라미터 그룹 및 백업을 사용해 클러스터를 복원하여 새로운 클러스터를 시드합니다. 백업에서 새 캐시로 복원을 참조하세요.

다른 파라미터를 변경한 경우에는 필요하지 않습니다.

파라미터 그룹 수정(ElastiCache API)

를 사용하여 파라미터 그룹의 파라미터 값을 변경하려면 `ModifyCacheParameterGroup` 작업을 ElastiCache API사용합니다.

Example

Memcached를 사용하여 변경하려는 파라미터의 이름과 허용 값을 찾으려면 섹션을 참조하세요. Memcached 특정 파라미터

다음 예제 코드에서는 `myMem14` 파라미터 그룹에서 두 파라미터 chunk_size 및 chunk_size_growth_fact의 값을 설정합니다.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=ModifyCacheParameterGroup
   &CacheParameterGroupName=myMem14
   &ParameterNameValues.member.1.ParameterName=chunk_size
   &ParameterNameValues.member.1.ParameterValue=96
   &ParameterNameValues.member.2.ParameterName=chunk_size_growth_fact
   &ParameterNameValues.member.2.ParameterValue=1.5
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

Example

Valkey 및 Redis 에서 변경하려는 파라미터의 이름과 허용 값을 찾으OSS려면 섹션을 참조하세요. Valkey 및 Redis OSS 파라미터

다음 샘플 코드는 두 파라미터의 값을 설정하고 파라미터 그룹 에서 reserved-memory-percent 클러스터를 활성화합니다`myredis32-on-30`. 파라미터 그룹을 Valkey 또는 Redisreserved-memory-percent`30`(클러스터 모드 활성화됨) 클러스터OSS(복제 그룹)와 함께 사용할 수 `yes` 있도록 를 (30%)로 설정하고 클러스터를 활성화했습니다.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=ModifyCacheParameterGroup
   &CacheParameterGroupName=myredis32-on-30
   &ParameterNameValues.member.1.ParameterName=reserved-memory-percent
   &ParameterNameValues.member.1.ParameterValue=30
   &ParameterNameValues.member.2.ParameterName=cluster-enabled
```

```
&ParameterNameValues.member.2.ParameterValue=yes
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20150202T192317Z
&Version=2015-02-02
&X-Amz-Credential=<credential>
```

자세한 내용은 ModifyCacheParameterGroup 단원을 참조하십시오.

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터가 있고 다음 파라미터를 변경하는 경우 클러스터의 노드를 재부팅해야 합니다.

- activerehashing
- 데이터베이스

자세한 내용은 노드 재부팅 단원을 참조하십시오.

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 파라미터 변경 사항

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터에서 다음 파라미터를 변경하는 경우 다음 단계를 따릅니다.

- activerehashing
- 데이터베이스

1. 클러스터의 수동 백업을 만듭니다. 수동 백업 지원을 참조하세요.
2. 클러스터를 삭제합니다. 에서 클러스터 삭제 ElastiCache을 참조하세요.
3. 변경한 파라미터 그룹 및 백업을 사용해 클러스터를 복원하여 새로운 클러스터를 시드합니다. 백업에서 새 캐시로 복원을 참조하세요.

다른 파라미터를 변경한 경우에는 필요하지 않습니다.

ElastiCache 파라미터 그룹 삭제

ElastiCache 콘솔, AWS CLI또는 를 사용하여 사용자 지정 파라미터 그룹을 삭제할 수 있습니다 ElastiCache API.

파라미터 그룹이 클러스터와 연결된 경우 삭제할 수 없습니다. 또한 기본 파라미터 그룹도 삭제할 수 없습니다.

파라미터 그룹 삭제(콘솔)

다음 절차에서는 콘솔을 사용하여 ElastiCache 파라미터 그룹을 삭제하는 방법을 보여줍니다.

ElastiCache 콘솔을 사용하여 파라미터 그룹을 삭제하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 사용 가능한 모든 파라미터 목록을 표시하려면 왼쪽 탐색 창에서 [Parameter Groups]를 선택합니다.
3. 파라미터 그룹의 이름 왼쪽에 있는 확인란을 선택하여 삭제할 파라미터 그룹을 선택합니다.

   [Delete] 버튼이 활성화됩니다.
4. Delete(삭제)를 선택합니다.

   [Delete Parameter Groups] 확인 화면이 나타납니다.
5. 파라미터 그룹을 삭제하려면 [Delete Parameter Groups] 확인 화면에서 [Delete]를 추가합니다.

   파라미터 그룹을 유지하려면 [Cancel]을 선택합니다.

파라미터 그룹 삭제(AWS CLI)

를 사용하여 파라미터 그룹을 삭제하려면 명령 을 AWS CLI사용합니다`delete-cache-parameter-group`. 삭제할 파라미터 그룹의 경우 `--cache-parameter-group-name`으로 지정된 파라미터 그룹에는 클러스터를 연결할 수 없으며 기본 파라미터 그룹이 될 수도 없습니다.

다음 샘플 코드는 myMem14개의 파라미터 그룹을 삭제합니다.

Example

Linux, macOS, Unix의 경우:

```
aws elasticache delete-cache-parameter-group \
    --cache-parameter-group-name myRed28
```

Windows의 경우:

```
aws elasticache delete-cache-parameter-group ^
    --cache-parameter-group-name myRed28
```

자세한 내용은 delete-cache-parameter-group 단원을 참조하십시오.

파라미터 그룹 삭제(ElastiCache API)

를 사용하여 파라미터 그룹을 삭제하려면 DeleteCacheParameterGroup 작업을 ElastiCache API 사용합니다. 삭제할 파라미터 그룹의 경우 CacheParameterGroupName으로 지정된 파라미터 그룹에는 클러스터를 연결할 수 없으며 기본 파라미터 그룹이 될 수도 없습니다.

Example

Memcached를 사용하면 다음 샘플 코드가 myMem14개의 파라미터 그룹을 삭제합니다.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DeleteCacheParameterGroup
   &CacheParameterGroupName=myMem14
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

Example

다음 샘플 코드는 myRed28 파라미터 그룹을 삭제합니다.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DeleteCacheParameterGroup
   &CacheParameterGroupName=myRed28
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

자세한 내용은 DeleteCacheParameterGroup 단원을 참조하십시오.

엔진별 파라미터

Valkey 및 Redis OSS

Valkey 7.2 파라미터는 Redis OSS 7 파라미터와 동일합니다.

Valkey 또는 Redis OSS 클러스터에 대한 파라미터 그룹을 지정하지 않으면 엔진 버전에 적합한 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 ElastiCache 파라미터 그룹 생성 단원을 참조하십시오.

주제

- Valkey 및 Redis OSS 파라미터
- Memcached 특정 파라미터

Valkey 및 Redis OSS 파라미터

주제

Valkey 7.2 및 Redis OSS 7 파라미터 변경 사항

파라미터 그룹 패밀리: redis7

Redis OSS 7 기본 파라미터 그룹은 다음과 같습니다.

- `default.redis7` - Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터 및 복제 그룹에 이 파라미터 그룹 또는 이 파라미터 그룹에서 파생된 파라미터를 사용합니다.
- `default.redis7.cluster.on` - Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 및 복제 그룹에 이 파라미터 그룹 또는 이 파라미터 그룹에서 파생된 파라미터를 사용합니다.

Redis OSS 7에 추가된 파라미터는 다음과 같습니다.

이름	Details	설명
cluster-allow-pubsubshard-when-down	허용되는 값: yes, no 기본값: yes 유형: 문자열 수정 가능 여부: 예 변경 적용: 클러스터의 모든 노드에 즉시 적용됨	기본값인 yes로 설정하면 클러스터가 다운된 상태에서, 슬롯을 소유하고 있다고 판단되는 한 노드가 pubsub 샤드 트래픽을 처리할 수 있습니다.
cluster-preferred-endpoint-type	허용되는 값: ip, tls-dynamic 기본값: tls-dynamic 유형: 문자열 수정 가능 여부: 예 변경 적용: 클러스터의 모든 노드에 즉시 적용됨	이 값은 MOVED/ASKING 요청에 대해 반환되는 엔드포인트와 CLUSTER SLOTS 및 의 엔드포인트 필드를 제어합니다CLUSTER SHARDS. 값이 ip로 설정되면 노드는 자신의 IP 주소를 광고합니다. 값을 tls-dynamic으로 설정하면 이 활성화되면 encryption-in-transit 노드가 호스트 이름을, 그렇지 않으면 IP 주소를 알립니다.
latency-tracking	허용되는 값: yes, no 기본값: no 유형: 문자열 수정 가능 여부: 예 변경 적용: 클러스터의 모든 노드에 즉시 적용됨	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.
hash-max-listpack-entries	허용되는 값: 0+ 기본값: 512	데이터 세트를 압축하기 위한 해시 항목 최대 개수입니다.

이름	Details	설명
	유형: 정수 수정 가능 여부: 예 변경 적용: 클러스터의 모든 노드에 즉시 적용됨	
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 임계값입니다.
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	Details	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 임계값입니다.

Redis OSS 7에서 변경된 파라미터는 다음과 같습니다.

이름	Details	설명
activerehashing	수정 가능: no. Redis OSS 7에서 이 파라미터는 기본적으로 숨겨지고 활성화됩니다. 비활성화하려면 지원 사례를 생성해야 합니다.	수정 가능 여부는 '예'였습니다.

Redis OSS 7에서 제거된 파라미터는 다음과 같습니다.

이름	Details	설명
hash-max-ziplist-entries	허용되는 값: 0+ 기본값: 512 유형: 정수 수정 가능 여부: 예	작은 해시 인코딩을 표현하는 데 ziplist 대신 listpack 사용

이름	Details	설명
	변경 적용: 클러스터의 모든 노드에 즉시 적용됨	
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
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.

이름	Details	설명
`list-max-ziplist-size`	허용되는 값: 기본값: `-2` 유형: 정수 수정 가능 여부: 예 변경 적용: 클러스터의 모든 노드에 즉시 적용됨	내부 목록 노드당 허용되는 항목 개수입니다.

Redis OSS 6.x 파라미터 변경 사항

파라미터 그룹 패밀리: redis6.x

Redis OSS 6.x 기본 파라미터 그룹은 다음과 같습니다.

- `default.redis6.x` - Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터 및 복제 그룹에 이 파라미터 그룹 또는 이 파라미터 그룹에서 파생된 파라미터를 사용합니다.
- `default.redis6.x.cluster.on` - Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 및 복제 그룹에 이 파라미터 그룹 또는 이 파라미터 그룹에서 파생된 파라미터를 사용합니다.

> **Note**
>
> Redis OSS 엔진 버전 6.2에서는 r6gd 노드 패밀리가 와 함께 사용하도록 도입되었을 때 r6gd 노드 유형에서는 , volatile-lru 및 allkeys-lru max-memory 정책의 데이터 계층화 ElastiCache만 지원됩니다.

자세한 내용은 ElastiCache (Redis OSS) 버전 6.2(향상됨) 및 ElastiCache (Redis OSS) 버전 6.0(향상됨) 단원을 참조하세요.

Redis OSS 6.x에 추가된 파라미터는 다음과 같습니다.

Details	설명	
acl-pubsub-default (added in 6.2)	허용되는 값: resetchannels , allchannels 기본값: allchannels 유형: 문자열 수정 가능 여부: 예 변경 사항 적용: 클러스터에 연결된 기존 Redis OSS 사용자는 기존 권한을 계속 갖게 됩니다. 사용자를 업데이트하거나 클러스터를 재부팅하여 기존 Redis OSS 사용자를 업데이트합니다.	이 클러스터에 배포된 ACL 사용자의 기본 pubsub 채널 권한입니다.
cluster-allow-reads-when-down (added in 6.0)	기본값: 아니요 유형: 문자열 수정 가능 여부: 예 변경 적용: 클러스터의 모든 노드에 즉시 적용됨	예로 설정하면 RedisOSS(클러스터 모드 활성화됨) 복제 그룹은 노드가 프라이머리 쿼럼에 도달할 수 없는 경우에도 읽기 명령을 계속 처리합니다. 기본값인 no로 설정하면 복제 그룹이 모든 명령을 거부합니다. 노드 그룹이 3개 미만인 클러스터를 사용하거나 애플리케이션에서 기한 경과 읽기를 안전하게 처리할 수 있는 경우 이 값을 yes로 설정하는 것이 좋습니다.
tracking-table-max-keys (added in 6.0)	기본값: 1,000,000 유형: 숫자 수정 가능 여부: 예 변경 적용: 클러스터의 모든 노드에 즉시 적용됨	클라이언트 측 캐싱을 지원하기 위해 Redis는 어떤 클라이언트가 어떤 키에 액세스했는지 추적할 수 있도록 OSS 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. 이 파라미

Details	설명	
		터 값을 초과하면 클라이언트가 임의로 무효화 메시지를 전송합니다. 이 값은 충분한 수의 키를 계속 추적하면서 메모리 사용을 제한하도록 조정해야 합니다. 메모리 부족 조건에서도 키가 무효화됩니다.
`acllog-max-len (added in 6.0)`	기본값: 128 유형: 숫자 수정 가능 여부: 예 변경 적용: 클러스터의 모든 노드에 즉시 적용됨	이 값은 ACL 로그의 최대 항목 수에 해당합니다.
`active-expire-effort (added in 6.0)`	기본값: 1 유형: 숫자 수정 가능 여부: 예 변경 적용: 클러스터의 모든 노드에 즉시 적용됨	Redis는 두 가지 메커니즘을 통해 수명 시간을 초과한 키를 OSS 삭제합니다. 하나는 키가 액세스되고 만료된 것으로 확인된 경우입니다. 다른 하나는 정기적인 작업이 키를 샘플링하고 유지 시간(TTL)이 초과된 키를 만료시키는 경우입니다. 이 파라미터는 Redis가 정기 작업의 항목을 만료시키는 데 OSS 사용하는 노력의 양을 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. 절충은 지연 시간이 더 길CPU고 잠재적으로 더 높을 수 있습니다. 메모리 사용량이 높고 CPU 사용률 증가를 견딜 수 없는 한 1의 값을 사용하는 것이 좋습니다.

Details	설명	
`lazyfree-lazy-user-del (added in 6.0)`	기본값: 아니요 유형: 문자열 수정 가능 여부: 예 변경 적용: 클러스터의 모든 노드에 즉시 적용됨	값이 yes로 설정되면 `DEL` 명령이 `UNLINK` 명령과 동일하게 작동합니다.

Redis OSS 6.x에서 제거된 파라미터는 다음과 같습니다.

이름	Details	설명
`lua-replicate-commands`	허용되는 값: yes/no 기본값: yes 유형: boolean 수정 가능 여부: 예 변경 사항 적용: 즉시	Lua 스크립트에서 항상 Lua 효과 복제를 활성화하거나 활성화하지 않음

Redis OSS 5.0.3 파라미터 변경 사항

파라미터 그룹 Family: redis5.0

Redis OSS 5.0 기본 파라미터 그룹

- `default.redis5.0` - Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터 및 복제 그룹에 이 파라미터 그룹 또는 이 파라미터 그룹에서 파생된 파라미터를 사용합니다.
- `default.redis5.0.cluster.on` - Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 및 복제 그룹에 이 파라미터 그룹 또는 이 파라미터 그룹에서 파생된 파라미터를 사용합니다.

Redis OSS 5.0.3에 추가된 파라미터

이름	Details	설명
rename-commands	기본값: 없음 유형: 문자열 수정 가능 여부: 예 변경 적용: 클러스터의 모든 노드에 즉시 적용됨	이름이 변경된 Redis OSS 명령의 공백 구분 목록입니다. 다음은 이름 변경에 사용할 수 있는 제한된 명령의 목록입니다. APPEND AUTH BITCOUNT BITFIELD BITOP BITPOS BLPOP BRPOP BRPOPLPUSH BZPOPMIN BZPOPMAX CLIENT CLUSTER COMMAND DBSIZE DECR DECRBY DEL DISCARD DUMP ECHO EVAL EVALSHA EXEC EXISTS EXPIRE EXPIREAT FLUSHALL FLUSHDB GEOADD GEOHASH GEOPOS GEODIST GEORADIUS GEORADIUSBYMEMBER GET GETBIT GETRANGE GETSET HDEL HEXISTS HGET HGETALL HINCRBY HINCRBYFLOAT HKEYS HLEN HMGET HMSET HSET HSETNX HSTRLEN HVALS INCR INCRBY INCRBYFLOAT INFO KEYS LASTSAVE LINDEX LINSERT LLEN LPOP LPUSH LPUSHX LRANGE LREM LSET LTRIM MEMORY MGET MONITOR MOVE MSET MSETNX MULTI OBJECT PERSIST PEXPIRE PEXPIREAT PFADD PFCOUNT PFMERGE PING PSETEX PSUBSCRIBE PUBSUB PTTL PUBLISH PUNSUBSCRIBE RANDOMKEY READONLY READWRITE RENAME RENAMENX RESTORE ROLE RPOP RPOPLPUSH RPUSH RPUSHX SADD SCARD SCRIPT SDIFF SDIFFSTORE SELECT SET SETBIT SETEX SETNX SETRANGE SINTER SINTERSTORE SISMEMBER SLOWLOG SMEMBERS SMOVE SORT SPOP SRANDMEMBER SREM STRLEN SUBSCRIBE

이름	Details	설명
		SUNION SUNIONSTORE SWAPDB TIME TOUCH TTL TYPE UNSUBSCRIBE UNLINK UNWATCH WAIT WATCH ZADD ZCARD ZCOUNT ZINCRBY ZINTERSTO RE ZLEXCOUNT ZPOPMAX ZPOPMIN ZRANGE ZRANGEBYLEX ZREVRANGE BYLEX ZRANGEBYSCORE ZRANK ZREM ZREMRANGEBYLEX ZREMRANGEBYRANK ZREMRANGEBYSCORE ZREVRANGE ZREVRANGEBYSCORE ZREVRANK ZSCORE ZUNIONSTORE SCAN SSCAN HSCAN ZSCAN XINFO XADD XTRIM XDEL XRA NGE XREVRANGE XLEN XREAD XGROUP XREADGROUP XACK XCLAIM XPENDING GEORADIUS_RO GEORADIUSBYMEMBER_ RO LOLWUT XSETID SUBSTR

자세한 내용은 ElastiCache (Redis OSS) 버전 5.0.6(향상됨) 단원을 참조하십시오.

Redis OSS 5.0.0 파라미터 변경 사항

파라미터 그룹 Family: redis5.0

Redis OSS 5.0 기본 파라미터 그룹

- `default.redis5.0` - Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터 및 복제 그룹에 이 파라미터 그룹 또는 이 파라미터 그룹에서 파생된 파라미터를 사용합니다.
- `default.redis5.0.cluster.on` - Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 및 복제 그룹에 이 파라미터 그룹 또는 이 파라미터 그룹에서 파생된 파라미터를 사용합니다.

Redis OSS 5.0에 추가된 파라미터

이름	Details	설명
stream-node-max-bytes	허용되는 값: 0+ 기본값: 4096 유형: 정수 수정 가능 여부: 예 변경 사항 적용: 즉시	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.
stream-node-max-entries	허용되는 값: 0+ 기본값: 100 유형: 정수 수정 가능 여부: 예 변경 사항 적용: 즉시	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.
active-defrag-max-scan-fields	허용되는 값: 1~1000000 기본값: 1000 유형: 정수 수정 가능 여부: 예 변경 사항 적용: 즉시	기본 사전 스캔에서 처리될 최대 set/hash/zset/list 필드 수
lua-replicate-commands	허용되는 값: yes/no 기본값: yes 유형: boolean	Lua 스크립트에서 항상 Lua 효과 복제를 활성화하거나 활성화하지 않음

이름	Details	설명
	수정 가능 여부: 예 변경 사항 적용: 즉시	
`replica-ignore-maxmemory`	기본값: yes 유형: boolean 수정 가능 여부: 아니요	복제본이 기본 노드와 독립적인 항목을 유지하여 `maxmemory` 설정을 무시하는지 결정합니다.

RedisOSS는 커뮤니티 피드백에 대한 응답으로 엔진 버전 5.0에서 여러 파라미터의 이름을 변경했습니다. 자세한 내용은 Redis OSS 5의 새로운 기능을 참조하세요. 다음 표에는 새 이름 및 이러한 이름이 이전 버전과 매핑되는 방법이 나와 있습니다.

Redis OSS 5.0에서 파라미터 이름 변경

이름	Details	설명
`replica-lazy-flush`	기본값: yes 유형: boolean 수정 가능 여부: 아니요 이전 이름: slave-lazy-flush	복제본 동기화 동안 비동기식 flushDB를 수행합니다.
`client-output-buffer-limit-replica-hard-limit`	기본값: 값은 OSS 노드 유형별 파라미터 재정의를 참조하세요. 유형: 정수 수정 가능 여부: 아니요 이전 이름: client-output-buffer-limit-slave-hard-limit	Redis OSS 읽기 전용 복제본: 클라이언트의 출력 버퍼가 지정된 바이트 수에 도달하면 클라이언트 연결이 해제됩니다.

이름	Details	설명
client-output-buffer-limit-replica-soft-limit	기본값: 값은 OSS 노드 유형별 파라미터 재정의를 참조하세요. 유형: 정수 수정 가능 여부: 아니요 이전 이름: client-output-buffer-limit-slave-soft-limit	Redis OSS 읽기 전용 복제본의 경우: 클라이언트의 출력 버퍼가 지정된 바이트 수에 도달하면 클라이언트 연결이 끊어지지만 이 조건이 에 대해 지속되는 경우에만 연결이 해제됩니다client-output-buffer-limit-replica-soft-seconds .
client-output-buffer-limit-replica-soft-seconds	기본값: 60 유형: 정수 수정 가능 여부: 아니요 이전 이름: client-output-buffer-limit-slave-soft-seconds	Redis OSS 읽기 전용 복제본: 클라이언트의 출력 버퍼가 이 초보다 오래 client-output-buffer-limit-replica-soft-limit 바이트로 유지되면 클라이언트 연결이 해제됩니다.
replica-allow-chaining	기본값: 아니요 유형: 문자열 수정 가능 여부: 아니요 이전 이름: slave-allow-chaining	Redis의 읽기 전용 복제본에 읽기 전용 복제본이 있을 OSS 수 있는지 여부를 결정합니다.
min-replicas-to-write	기본값: 0 유형: 정수 수정 가능 여부: 예 이전 이름: min-slaves-to-write 변경 적용: 즉시	기본 노드가 클러스터에서 쓰기를 허용하기 위해 사용 가능해야 하는 최소 읽기 전용 복제본 수입니다. 사용 가능한 복제본 수가 이 수보다 떨어지면 기본 노드는 더 이상 쓰기 요청을 허용하지 않습니다. 이 파라미터 또는 min-replicas-max-lag 가 0인 경우 기본 노드는 복제본을 사용할 수 없더라도 항상 쓰기 요청을 수락합니다.

이름	Details	설명
`min-replicas-max-lag`	기본값: 10 유형: 정수 수정 가능 여부: 예 이전 이름: min-slaves-max-lag 변경 적용: 즉시	기본 노드가 읽기 전용 복제본에서 핑 요청을 수신해야 하는 시간(초)입니다. 이 시간까지 기본 노드가 핑을 수신하지 않으면 복제본을 더 이상 사용할 수 없는 것으로 간주합니다. 사용 가능한 복제본 수가 아래로 떨어지면 min-replicas-to-write기본 는 해당 시점에서 쓰기 수락을 중지합니다. 이 파라미터 또는 min-replicas-to-write 가 0인 경우 기본 노드는 사용 가능한 복제본이 없더라도 항상 쓰기 요청을 수락합니다.
`close-on-replica-write`	기본값: yes 유형: boolean 수정 가능 여부: 예 이전 이름: close-on-slave-write 변경 적용: 즉시	활성화하면 읽기 전용 복제본에 작성을 시도하는 클라이언트의 연결이 끊어집니다.

Redis OSS 5.0에서 제거된 파라미터

이름	Details	설명
`repl-timeout`	기본값: 60 수정 가능 여부: 아니요	이 버전에서는 파라미터를 사용할 수 없습니다.

Redis OSS 4.0.10 파라미터 변경 사항

파라미터 그룹 패밀리: redis4.0

Redis OSS 4.0.x 기본 파라미터 그룹

- `default.redis4.0` - Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터 및 복제 그룹에 이 파라미터 그룹 또는 이 파라미터 그룹에서 파생된 파라미터를 사용합니다.
- `default.redis4.0.cluster.on` - Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 및 복제 그룹에 이 파라미터 그룹 또는 이 파라미터 그룹에서 파생된 파라미터를 사용합니다.

Redis OSS 4.0.10에서 변경된 파라미터

이름	Details	설명
`maxmemory-policy`	허용되는 값: `allkeys-lru` , `volatile-lru` , **`allkeys-lfu`** , **`volatile-lfu`** , `allkeys-random` , `volatile-random` , `volatile-ttl` , `noeviction` 기본값: volatile-lru 유형: 문자열 수정 가능 여부: 예 변경 사항 적용: 즉시	`maxmemory-policy` 버전 2.6.13에는 섹션이 추가되었습니다. 버전 4.0.10에는 두 개의 새로운 허용 값이 추가되었습니다. 즉`allkeys-lfu` , 는 근사치를 사용하여 모든 키를 제거하며 LFU, `volatile-lfu` 는 만료 세트가 있는 키 LFU 중에서 근사치를 사용하여 제거됩니다. 버전 6.2에서는 데이터 계층화에 사용하기 위해 r6gd 노드 패밀리를 제공한 경우 `noeviction` ,`volatile-lru` 및 `allkeys-lru` 최대 메모리 정책만 r6gd 노드 유형을 통해 지원됩니다.

Redis OSS 4.0.10에 추가된 파라미터

이름	Details	설명
비동기 삭제 파라미터		
`lazyfree-lazy-eviction`	허용되는 값: yes/no 기본값: 아니요 유형: boolean 수정 가능 여부: 예	제거 시 비동기식 삭제를 수행합니다.

이름	Details	설명
	변경 사항 적용: 즉시	
`lazyfree-lazy-expire`	허용되는 값: yes/no 기본값: 아니요 유형: boolean 수정 가능 여부: 예 변경 사항 적용: 즉시	키 만료 시 비동기식 삭제를 수행합니다.
`lazyfree-lazy-server-del`	허용되는 값: yes/no 기본값: 아니요 유형: boolean 수정 가능 여부: 예 변경 사항 적용: 즉시	값을 업데이트하는 명령에 대해 비동기식 삭제를 수행합니다.
`slave-lazy-flush`	허용되는 값: N/A 기본값: 아니요 유형: boolean 수정 가능 여부: 아니요 변경 사항 적용: N/A	slave sync 동안 비동기식 flushDB를 수행합니다.
LFU 파라미터		

이름	Details	설명
lfu-log-factor	허용되는 값: 0보다 큰 모든 정수 기본값: 10 유형: 정수 수정 가능 여부: 예 변경 사항 적용: 즉시	키 카운터를 포화시키는 키 히트 수를 결정하는 로그 팩터를 설정합니다.
lfu-decay-time	허용되는 값: 모든 정수 기본값: 1 유형: 정수 수정 가능 여부: 예 변경 사항 적용: 즉시	키 카운터 감소에 소요된 시간입니다(분).
활성 조각 모음 파라미터		
activedefrag	허용되는 값: yes/no 기본값: 아니요 유형: boolean 수정 가능 여부: 예 변경 사항 적용: 즉시	활성 조각 모음 파라미터가 활성화됩니다.

이름	Details	설명
active-defrag-ignore-bytes	허용되는 값: 10485760~104857600 기본값: 104857600 유형: 정수 수정 가능 여부: 예 변경 사항 적용: 즉시	활성 조각 모음을 시작하는 조각의 최소 수입니다.
active-defrag-threshold-lower	허용되는 값: 1~100 기본값: 10 유형: 정수 수정 가능 여부: 예 변경 사항 적용: 즉시	활성 조각 모음을 시작하는 조각의 최소 비율입니다.
active-defrag-threshold-upper	허용되는 값: 1~100 기본값: 100 유형: 정수 수정 가능 여부: 예 변경 사항 적용: 즉시	최대 작업을 사용하는 조각의 최대 비율입니다.

이름	Details	설명
active-defrag-cycle-min	허용되는 값: 1~75 기본값: 25 유형: 정수 수정 가능 여부: 예 변경 사항 적용: 즉시	CPU 백분율 단위의 조각 모음에 대한 최소한의 노력.
active-defrag-cycle-max	허용되는 값: 1~75 기본값: 75 유형: 정수 수정 가능 여부: 예 변경 사항 적용: 즉시	CPU 백분율 단위 조각 모음에 대한 최대 노력입니다.
클라이언트 출력 버퍼 파라미터		
client-query-buffer-limit	허용되는 값: 1048576~1073741824 기본값: 1073741824 유형: 정수 수정 가능 여부: 예 변경 사항 적용: 즉시	단일 클라이언트 쿼리 버퍼의 최대 크기입니다.

이름	Details	설명
`proto-max-bulk-len`	허용되는 값: 1048576~536870912 기본값: 536870912 유형: 정수 수정 가능 여부: 예 변경 사항 적용: 즉시	단일 요소 요청의 최대 크기입니다.

Redis OSS 3.2.10 파라미터 변경 사항

파라미터 그룹 패밀리: redis3.2

ElastiCache (Redis OSS) 3.2.10 지원되는 추가 파라미터가 없습니다.

Redis OSS 3.2.6 파라미터 변경 사항

파라미터 그룹 패밀리: redis3.2

Redis OSS 3.2.6의 경우 추가 파라미터가 지원되지 않습니다.

Redis OSS 3.2.4 파라미터 변경 사항

파라미터 그룹 패밀리: redis3.2

Redis OSS 3.2.4부터는 두 가지 기본 파라미터 그룹이 있습니다.

- `default.redis3.2` - Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹을 생성하고 Redis OSS 3.2.4의 추가 기능을 계속 사용하려는 경우 Redis OSS 3.2.4를 실행할 때 이 파라미터 그룹 또는 이 그룹에서 파생된 파라미터 그룹을 지정합니다.
- `default.redis3.2.cluster.on` - Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹을 생성하려는 경우 이 파라미터 그룹 또는 이 그룹에서 파생된 파라미터 그룹을 지정합니다.

주제

- Redis OSS 3.2.4의 새 파라미터
- Redis OSS 3.2.4에서 변경된 파라미터(향상됨)

Redis OSS 3.2.4의 새 파라미터

파라미터 그룹 패밀리: redis3.2

Redis OSS 3.2.4의 경우 다음과 같은 추가 파라미터가 지원됩니다.

이름	Details	설명
`list-max-ziplist-size`	기본값: -2 유형: 정수 수정 가능 여부: 아니요	목록은 공간을 절약하기 위해 특별한 방법으로 인코딩됩니다. 내부 목록 노드 당 허용되는 항목 수는 요소의 최대 수 또는 최대 고정 크기로 지정할 수 있습니다. 최대 고정 크기의 경우 -5~-1을 사용합니다. • -5: 최대 크기: 64Kb - 일반 워크로드에 권장되지 않음 • -4: 최대 크기: 32Kb - 권장되지 않음 • -3: 최대 크기: 16Kb - 권장되지 않음 • -2: 최대 크기: 8Kb - 권장 • -1: 최대 크기: 4Kb - 권장 • 양수는 목록 노드당 정확히 그 수 만큼의 요소를 저장합니다.
`list-compress-depth`	기본값: 0 유형: 정수 수정 가능 여부: 예 변경 적용: 즉시	목록을 압축할 수도 있습니다. 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.

이름	Details	설명
		[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.
`cluster-enabled`	기본값: 아니요/예 * 유형: 문자열 수정 가능 여부: 아니요	클러스터 모드에서 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹인지(예) 비클러스터 모드에서 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹인지(아니요) 나타냅니다. 클러스터 모드의 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹은 최대 500개의 노드 그룹에 데이터를 분할할 수 있습니다. * Redis OSS 3.2.x에는 두 개의 기본 파라미터 그룹이 있습니다. • `default.redis3.2` - 기본 값 no. • `default.redis3.2.cluster.on` - 기본 값 `yes`. .

이름	Details	설명
cluster-require-full-coverage	기본값: 아니요 유형: boolean 수정 가능 여부: 예 변경 적용: 즉시	로 설정하면 클러스터 모드의 yesValkey 또는 RedisOSS(클러스터 모드 활성화됨) 노드가 해시 슬롯이 하나 이상 검색되지 않은 것으로 감지되면 쿼리 수락을 중지합니다(사용 가능한 노드가 쿼리를 제공하고 있지 않음). 클러스터가 부분적으로 가동 중지되면 클러스터를 사용할 수 없게 됩니다. 모든 슬롯이 다시 확인되는 즉시 자동으로 사용할 수 있게 됩니다. 그러나 때로는 확인된 일부 키스페이스의 쿼리를 계속해서 허용하는 작업 중인 클러스터의 하위 세트가 필요합니다. 이렇게 하려면 cluster-require-full-coverage 옵션을 no로 설정합니다.
hll-sparse-max-bytes	기본값: 3000 유형: 정수 수정 가능 여부: 예 변경 적용: 즉시	HyperLogLog 희소 표현 바이트 제한입니다. 제한은 16바이트 헤더를 포함합니다. 희소 표현을 HyperLogLog 사용하는 이 이 이 제한을 초과하면 조밀 표현으로 변환됩니다. 그 시점에서는 밀도가 높은 표현이 메모리 효율을 높이기 때문에 16000보다 큰 값은 권장하지 않습니다. PFADD 너무 많이 느려지지 않고 공간 효율적인 인코딩의 이점을 누리려면 약 3000의 값을 사용하는 것이 좋습니다. 즉, 희소 인코딩의 경우 O(N)입니다. 가 문제가 아니지만 공백CPU이이고 데이터 세트가 카디널리티가 0~15000 범위인 많은 HyperLogLogs 로 구성된 경우 값을 ~10000으로 올릴 수 있습니다.

이름	Details	설명
`reserved-memory-percent`	기본값: 25 유형: 정수 수정 가능 여부: 예 변경 적용: 즉시	비데이터 사용을 위해 예약된 노드의 메모리 비율입니다. 기본적으로 Redis OSS 데이터 풋프린트는 노드의 모든 메모리를 사용할 때까지 증가합니다. 이 경우 과도한 메모리 페이징으로 인해 노드 성능이 저하될 수 있습니다. 메모리를 예약하면 페이징 양을 줄이는 데 도움이 되도록 Redis 이외의 OSS 용도로 사용 가능한 메모리 중 일부를 따로 보관할 수 있습니다. 이 파라미터는 에 특정 ElastiCache되며 표준 Redis OSS 배포의 일부가 아닙니다. 자세한 내용은 `reserved-memory` 및 Valkey 및 Redis용 예약 메모리 관리 OSS 단원을 참조하세요.

Redis OSS 3.2.4에서 변경된 파라미터(향상됨)

파라미터 그룹 패밀리: redis3.2

Redis OSS 3.2.4의 경우 다음 파라미터가 변경되었습니다.

이름	Details	변경 사항
`activerehashing`	수정 가능 여부: 파라미터 그룹이 캐시 클러스터와 연결되어 있지 않은 경우 예이고, 그렇지 않으면 아니오입니다.	수정 가능 여부는 '아니요'였습니다.
`databases`	수정 가능 여부: 파라미터 그룹이 캐시 클러스터와 연결되어 있지 않은 경우 예이고, 그렇지 않으면 아니오입니다.	수정 가능 여부는 '아니요'였습니다.

이름	Details	변경 사항
appendonly	기본값: 꺼짐 수정 가능 여부: 아니요	이전 Redis OSS 버전에서 업그레이드하려면 먼저 appendonly 꺼야 합니다.
appendfsync	기본값: 꺼짐 수정 가능 여부: 아니요	이전 Redis OSS 버전에서 업그레이드하려면 먼저 appendfsync 꺼야 합니다.
repl-timeout	기본값: 60 수정 가능 여부: 아니요	현재 기본값을 60으로 수정할 수 없습니다.
tcp-keepalive	기본값: 300	기본값은 0입니다.
list-max-ziplist-entries		파라미터를 더 이상 사용할 수 없습니다.
list-max-ziplist-value		파라미터를 더 이상 사용할 수 없습니다.

Redis OSS 2.8.24(향상됨) 추가 파라미터

파라미터 그룹 패밀리: redis2.8

Redis OSS 2.8.24의 경우 추가 파라미터가 지원되지 않습니다.

Redis OSS 2.8.23(향상됨) 추가 파라미터

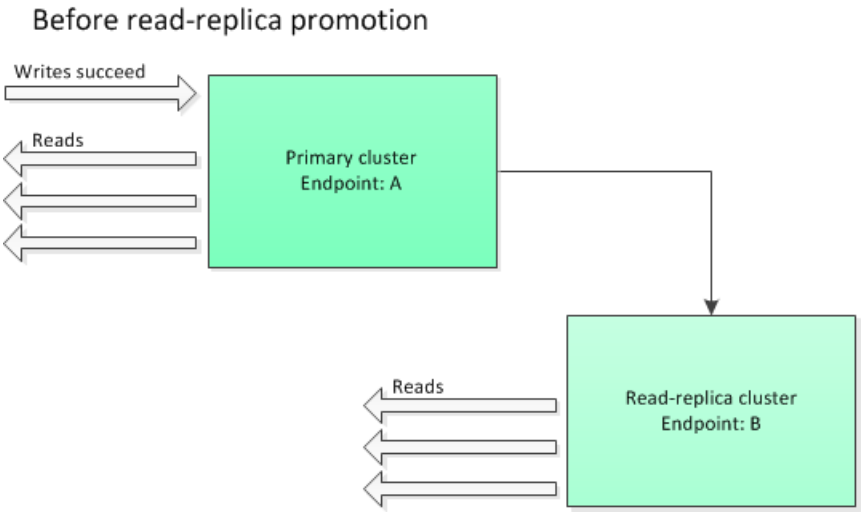
파라미터 그룹 패밀리: redis2.8

Redis OSS 2.8.23의 경우 다음과 같은 추가 파라미터가 지원됩니다.

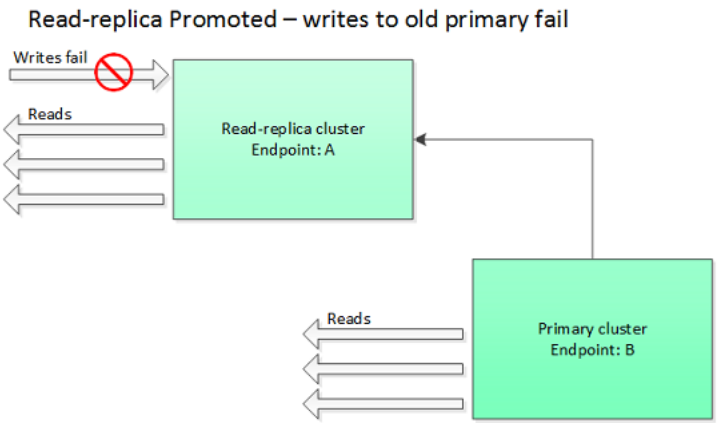
이름	Details	설명
`close-on-slave-write`	기본값: yes 유형: 문자열(yes/no) 수정 가능 여부: 예 변경 적용: 즉시	활성화하면 읽기 전용 복제본에 작성을 시도하는 클라이언트의 연결이 끊어집니다.

작동 방식 close-on-slave-write

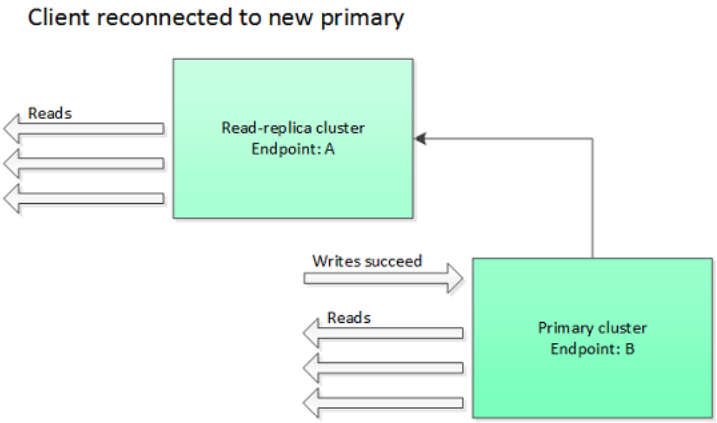
Amazon에서 이 `close-on-slave-write` 파라미터를 도입 ElastiCache 하면 읽기 전용 복제본을 프라이머리로 승격하여 프라이머리 노드 및 읽기 전용 복제본 노드 스왑 역할이 발생할 때 클러스터가 응답하는 방식을 더 잘 제어할 수 있습니다.



다중 AZ가 활성화된 복제 그룹 장애 조치 이외의 다른 이유로 읽기 전용 복제본 클러스터가 기본으로 승격되면 클라이언트는 엔드포인트 A에 계속 쓰려고 시도합니다. 그러나 엔드포인트 A가 읽기 전용 복제본의 엔드포인트이기 때문에 쓰기가 실패합니다. 이는 ElastiCache 도입 OSS 전 Redis의 동작`close-on-replica-write`과 를 비활성화한 경우의 동작입니다`close-on-replica-write`.



`close-on-replica-write`를 활성화하면 클라이언트가 읽기 전용 복제본에 쓰기를 시도할 때마다 클러스터와의 클라이언트 연결이 종료됩니다. 애플리케이션 로직이 연결 해제를 감지하고, DNS 테이블을 확인하고, 이제 엔드포인트 B가 되는 기본 엔드포인트에 다시 연결해야 합니다.



비활성화할 수 있는 경우 close-on-replica-write

`close-on-replica-write`를 비활성화했는데 장애가 발생한 클러스터에 쓴 경우 `close-on-replica-write`를 비활성화하는 이유는 무엇일까요?

앞서 언급했듯이 `close-on-replica-write`를 활성화하면 클라이언트가 읽기 전용 복제본에 쓰기를 시도할 때마다 클러스터와의 클라이언트 연결이 종료됩니다. 노드에 새로운 연결을 설정하는 것은 시간이 소요됩니다. 따라서 복제본에 대한 쓰기 요청의 결과로 연결을 끊고 다시 연결하면 동일한 연결을 통해 제공되는 읽기 요청의 지연 시간에도 영향을 미칩니다. 새로운 연결이 설정될 때까지 이 영향이 그대로 유지됩니다. 애플리케이션이 특별히 읽기 중심이거나 지연 시간에 매우 민감한 경우, 읽기 성능이 저하되지 않도록 클라이언트 연결을 유지하는 것이 좋습니다.

Redis OSS 2.8.22(향상됨) 추가 파라미터

파라미터 그룹 패밀리: redis2.8

Redis OSS 2.8.22의 경우 추가 파라미터가 지원되지 않습니다.

> **Important**
>
> - Redis OSS 버전 2.8.22부터 는 기본 클러스터와 복제본 클러스터에 `repl-backlog-size` 적용됩니다.
> - Redis OSS 버전 2.8.22부터는 `repl-timeout` 파라미터가 지원되지 않습니다. 변경되면 는 와 마찬가지로 기본값(60초)으로 ElastiCache 덮어씁니다`appendonly`.

다음 파라미터는 더 이상 지원되지 않습니다.

- appendonly
- appendfsync
- repl-timeout

Redis OSS 2.8.21 추가 파라미터

파라미터 그룹 패밀리: redis2.8

Redis OSS 2.8.21의 경우 추가 파라미터가 지원되지 않습니다.

Redis OSS 2.8.19 추가 파라미터

파라미터 그룹 패밀리: redis2.8

Redis OSS 2.8.19의 경우 추가 파라미터가 지원되지 않습니다.

Redis OSS 2.8.6 추가 파라미터

파라미터 그룹 패밀리: redis2.8

Redis OSS 2.8.6의 경우 다음과 같은 추가 파라미터가 지원됩니다.

이름	Details	설명
`min-slaves-max-lag`	기본값: 10 유형: 정수	기본 노드가 읽기 전용 복제본에서 핑 요청을 수신해야 하는 시간(초)입니다. 이 시간까지 기본 노드가

이름	Details	설명
	수정 가능 여부: 예 변경 적용: 즉시	핑을 수신하지 않으면 복제본을 더 이상 사용할 수 없는 것으로 간주합니다. 사용 가능한 복제본 수가 아래로 떨어지면 min-slaves-to-write 기본 는 해당 시점에서 쓰기 수락을 중지합니다. 이 파라미터 또는 min-slaves-to-write 가 0인 경우 기본 노드는 복제본을 사용할 수 없더라도 항상 쓰기 요청을 수락합니다.
`min-slaves-to-write`	기본값: 0 유형: 정수 수정 가능 여부: 예 변경 적용: 즉시	기본 노드가 클러스터에서 쓰기를 허용하기 위해 사용 가능해야 하는 최소 읽기 전용 복제본 수입니다. 사용 가능한 복제본 수가 이 수보다 떨어지면 기본 노드는 더 이상 쓰기 요청을 허용하지 않습니다. 이 파라미터 또는 min-slaves-max-lag 가 0인 경우 기본 노드는 복제본을 사용할 수 없는 경우에도 항상 쓰기 요청을 수락합니다.

이름	Details	설명
notify-keyspace-events	기본값: (빈 문자열) 유형: 문자열 수정 가능 여부: 예 변경 적용: 즉시	Redis가 클라이언트에 알릴 OSS 수 있는 키스페이스 이벤트의 유형 입니다. 각 이벤트 유형은 한 글자로 표현됩니다. • K - 접두사가 __keyspace@<db>__로 게시된 키스페이스 이벤트 • E - 접두사가 __keyevent@<db>__로 게시된 키-이벤트 이벤트 • g - , DEL, 등과 같은 일반적이고 비특정 명령입니다EXPIRERENAME. • $ - 문자열 명령 • l - 나열 명령 • s - 세트 명령 • h - 해시 명령 • z - 정렬된 세트 명령 • x - 만료된 이벤트(키가 만료될 때마다 생성되는 이벤트) • e - 제거된 이벤트(최대 메모리로 키가 제거될 때 생성되는 이벤트) • A - g$lshzxe의 별칭

이름	Details	설명
		이러한 이벤트 유형을 자유롭게 조합할 수 있습니다. 예를 들어, 는 Redis가 모든 이벤트 유형에 대한 알림을 게시OSS할 수 있음을 AKE 의미합니다. 오류 메시지가 발생할 수 있으므로 위에 나열된 문자 이외의 다른 문자로 시도하지 마십시오. 기본적으로 이 파라미터는 빈 문자열로 설정되어 있습니다. 즉, 키스페이스 이벤트 알림이 비활성화되어 있습니다.
`repl-backlog-size`	기본값: 1048576 유형: 정수 수정 가능 여부: 예 변경 적용: 즉시	기본 노드 백로그 버퍼의 크기(바이트)입니다. 백로그는 기본 노드의 데이터에 대한 업데이트를 레코딩하는 데 사용됩니다. 읽기 전용 복제본이 기본에 연결되면 기본 노드를 따라잡기 위해 백로그에서 데이터를 적용하는 부분적 동기화(psync)를 수행하려고 시도합니다. psync가 실패하면 전체 동기화가 필요합니다. 이 파라미터의 최소값은 16384입니다. **Note** Redis OSS 2.8.22부터 이 파라미터는 기본 클러스터와 읽기 전용 복제본에 적용됩니다.

이름	Details	설명
repl-backlog-ttl	기본값: 3600 유형: 정수 수정 가능 여부: 예 변경 적용: 즉시	기본 노드가 백로그 버퍼를 보관할 시간(초)입 니다. 마지막 복제본 노드가 연결 해제된 시점부터 백로그의 데이터는 repl-backlog-ttl이 만료될 때 까지 그대로 유지됩니다. 이 시간 안에 복제본이 기본 노드에 연결되지 않으면 기본이 백로그 버퍼를 해제합니다. 결국 복제본이 다시 연결되면 기본과 전체 동기화를 수행해야 합니다. 파라미터를 0으로 설정하면 백로그 버퍼가 절대 해제되지 않습니다.
repl-timeout	기본값: 60 유형: 정수 수정 가능 여부: 예 변경 적용: 즉시	다음에 대한 제한 시간(초)을 나타냅니다. • 읽기 전용 복제본의 관점에서 동기화 중 벌크 데이터 전송 • 복제본의 관점에서 기본 노드 제한 시간 • 기본 노드의 관점에서 복제본 제한 시간

Redis OSS 2.6.13 파라미터

파라미터 그룹 패밀리: redis2.6

Redis OSS 2.6.13는 에서 OSS 지원하는 Redis의 첫 번째 버전입니다 ElastiCache. 다음 표에는 에서 ElastiCache 지원하는 Redis OSS 2.6.13 파라미터가 나와 있습니다.

이름	Details	설명
`activerehashing`	기본값: yes 유형: 문자열(yes/no) 수정 가능 여부: 예 변경 적용: 생성 시	Redis의 활성 재해싱 기능을 사용할지 결정합니다. 기본 해시 테이블은 초당 10회 리해시되며, 각 리해시 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.
`appendonly`	기본값: 아니요 유형: 문자열 수정 가능 여부: 예 변경 적용: 즉시	Redis의 파일만 추가 기능()을 활성화 또는 비활성화합니다AOF. AOF 는 캐시의 데이터를 변경하는 모든 Redis OSS 명령을 캡처하며 특정 노드 장애로부터 복구하는 데 사용됩니다. 기본값은 가 아니며, 이는 가 꺼져 AOF 있음을 의미합니다. 를 활성화하려면 이 파라미터를 예로 설정합니다AOF. 자세한 내용은 장애 완화 단원을 참조하십시오. **Note** 캐시.t1.micro 및 캐시.t2.* 노드에는 파일만 추가(AOF)가 지원되지 않습니다. 이 유형의 노드에서는 `appendonly` 파라미터 값이 무시됩니다. **Note** 다중 AZ 복제 그룹의 경우 AOF는 허용되지 않습니다.

이름	Details	설명
appendfsync	기본값: everysec 유형: 문자열 수정 가능 여부: 예 변경 적용: 즉시	이 예로 설정되면 AOF 출력 버퍼appendonly 가 디스크에 기록되는 빈도를 제어합니다. • no - 버퍼가 필요에 따라 디스크로 플러시됩니다. • everysec - 버퍼가 1초에 한번씩 플러시됩니다. 이 값이 기본값입니다. • always - 클러스터의 데이터가 수정될 때마다 버퍼가 플러시됩니다. • 버전 2.8.22 이상에서는 appendfsync가 지원되지 않습니다.
client-output-buffer-limit-normal-hard-limit	기본값: 0 유형: 정수 수정 가능 여부: 예 변경 적용: 즉시	클라이언트의 출력 버퍼가 특정 바이트 수에 도달하면 클라이언트가 연결 해제됩니다. 기본값은 0입니다(하드 제한 없음).
client-output-buffer-limit-normal-soft-limit	기본값: 0 유형: 정수 수정 가능 여부: 예 변경 적용: 즉시	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	기본값: 0 유형: 정수 수정 가능 여부: 예 변경 적용: 즉시	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).

이름	Details	설명
client-output-buffer-limit-pubsub-hard-limit	기본값: 33554432 유형: 정수 수정 가능 여부: 예 변경 적용: 즉시	Redis 게시/구독 클라이언트의 경우: 클라이언트의 출력 버퍼가 지정된 OSS 바이트 수에 도달하면 클라이언트 연결이 해제됩니다.
client-output-buffer-limit-pubsub-soft-limit	기본값: 8388608 유형: 정수 수정 가능 여부: 예 변경 적용: 즉시	Redis OSS 게시/구독 클라이언트의 경우: 클라이언트의 출력 버퍼가 지정된 바이트 수에 도달하면 클라이언트 연결이 끊어지지만 이 조건이 에 대해 지속되는 경우에만 연결이 해제됩니다client-output-buffer-limit-pubsub-soft-seconds .
client-output-buffer-limit-pubsub-soft-seconds	기본값: 60 유형: 정수 수정 가능 여부: 예 변경 적용: 즉시	Redis OSS 게시/구독 클라이언트의 경우: 클라이언트의 출력 버퍼가 이 초보다 오래 client-output-buffer-limit-pubsub-soft-limit 바이트로 유지되면 클라이언트 연결이 해제됩니다.
client-output-buffer-limit-slave-hard-limit	기본값: 값은 OSS 노드 유형별 파라미터 재정의를 참조하세요. 유형: 정수 수정 가능 여부: 아니요	Redis OSS 읽기 전용 복제본: 클라이언트의 출력 버퍼가 지정된 바이트 수에 도달하면 클라이언트 연결이 해제됩니다.
client-output-buffer-limit-slave-soft-limit	기본값: 값은 OSS 노드 유형별 파라미터 재정의를 참조하세요. 유형: 정수 수정 가능 여부: 아니요	Redis OSS 읽기 전용 복제본의 경우: 클라이언트의 출력 버퍼가 지정된 바이트 수에 도달하면 클라이언트 연결이 끊어지지만 이 조건이 에 대해 지속되는 경우에만 연결이 해제됩니다client-output-buffer-limit-slave-soft-seconds .

이름	Details	설명
client-output-buffer-limit-slave-soft-seconds	기본값: 60 유형: 정수 수정 가능 여부: 아니요	Redis OSS 읽기 전용 복제본의 경우: 클라이언트의 출력 버퍼가 이 초보다 오래 client-output-buffer-limit-slave-soft-limit바이트로 유지되면 클라이언트 연결이 해제됩니다.
databases	기본값: 16 유형: 정수 수정 가능 여부: 아니요 변경 적용: 생성 시	데이터베이스가 분할되는 논리적 피티션의 수입니다. 이 값을 낮게 유지하는 것이 좋습니다. 이 값은 파라미터 그룹을 생성할 때 설정됩니다. 새 파라미터 그룹을 클러스터에 할당할 때 이전 파라미터 그룹과 새 파라미터 그룹에서 값이 동일해야 합니다.
hash-max-ziplist-entries	기본값: 512 유형: 정수 수정 가능 여부: 예 변경 적용: 즉시	해시에 사용되는 메모리 양을 결정합니다. 지정된 수보다 적은 수의 항목이 있는 해시는 공간을 절약하는 특수 인코딩을 사용하여 저장됩니다.
hash-max-ziplist-value	기본값: 64 유형: 정수 수정 가능 여부: 예 변경 적용: 즉시	해시에 사용되는 메모리 양을 결정합니다. 지정된 바이트 수보다 작은 항목이 있는 해시는 공간을 절약하는 특수 인코딩을 사용하여 저장됩니다.
list-max-ziplist-entries	기본값: 512 유형: 정수 수정 가능 여부: 예 변경 적용: 즉시	목록에 사용되는 메모리 양을 결정합니다. 지정된 수보다 적은 수의 항목이 있는 목록은 공간을 절약하는 특수 인코딩을 사용하여 저장됩니다.

이름	Details	설명
`list-max-ziplist-value`	기본값: 64 유형: 정수 수정 가능 여부: 예 변경 적용: 즉시	목록에 사용되는 메모리 양을 결정합니다. 지정된 바이트 수보다 작은 항목이 있는 목록은 공간을 절약하는 특수 인코딩을 사용하여 저장됩니다.
`lua-time-limit`	기본값: 5000 유형: 정수 수정 가능 여부: 아니요	가 스크립트를 중지하는 작업을 ElastiCache 수행하기 전에 Lua 스크립트의 최대 실행 시간을 밀리초 단위로 표시합니다. `lua-time-limit` 이 초과되면 모든 Redis OSS 명령은 양식 ____-BUSY의 오류를 반환합니다. 이 상태는 많은 필수 Redis OSS 작업에 간섭을 일으킬 수 있으므로 ElastiCache 는 먼저 SCRIPT KILL 명령을 실행합니다. 이 작업이 실패하면 Redis ElastiCache 를 강제로 다시 시작합니다OSS.
`maxclients` 이 값은 명시적으로 지정된 유형을 제외한 모든 인스턴스 유형에 적용됩니다.	기본값: 65000 유형: 정수 수정 가능 여부: 아니요	한 번에 연결할 수 있는 최대 클라이언트 수입니다.
	t2.medium 기본값: 20000 유형: 정수 수정 가능 여부: 아니요	
	t2.small 기본값: 20000 유형: 정수 수정 가능 여부: 아니요	

이름	Details	설명
	t2.micro 기본값: 20000 유형: 정수 수정 가능 여부: 아니요	
	t4g.micro 기본값: 20000 유형: 정수 수정 가능 여부: 아니요	
	t3.medium 기본값: 46000 유형: 정수 수정 가능 여부: 아니요	
	t3.small 기본값: 46000 유형: 정수 수정 가능 여부: 아니요	
	t3.micro 기본값: 20000 유형: 정수 수정 가능 여부: 아니요	
`maxmemory-policy`	기본값: volatile-lru 유형: 문자열 수정 가능 여부: 예 변경 적용: 즉시	최대 메모리 사용량에 도달했을 때 키에 대한 제거 정책입니다. 유효한 값은 `volatile-lru | allkeys-lru | volatile-random | allkeys-random | volatile-ttl | noeviction` 입니다. 자세한 내용은 Valkey 또는 Redis를 LRU 캐시 OSS로 사용을 참조하세요.

이름	Details	설명
maxmemory-samples	기본값: 3 유형: 정수 수정 가능 여부: 예 변경 적용: 즉시	(LRU) 및 time-to-live (TTL) 계산의 경우 least-recently-used 이 파라미터는 확인할 키의 샘플 크기를 나타냅니다. 기본적으로 Redis는 3개의 키를 OSS 선택하고 최근에 가장 적게 사용된 키를 사용합니다.
reserved-memory	기본값: 0 유형: 정수 수정 가능 여부: 예 변경 적용: 즉시	비데이터 사용을 위해 예약된 총 메모리(바이트)입니다. 기본적으로 Redis OSS 노드는 노드를 사용할 때까지 증가합니다maxmemory (참조OSS 노드 유형별 파라미터 재정의). 이 경우 과도한 메모리 페이징으로 인해 노드 성능이 저하될 수 있습니다. 메모리를 예약하면 페이징 양을 줄이는 데 도움이 되도록 Redis 이외의 OSS 용도로 사용 가능한 메모리 중 일부를 따로 보관할 수 있습니다. 이 파라미터는 에 특정 ElastiCache되며 표준 Redis OSS 배포의 일부가 아닙니다. 자세한 내용은 reserved-memory-percent 및 Valkey 및 Redis용 예약 메모리 관리 OSS 단원을 참조하세요.
set-max-intset-entries	기본값: 512 유형: 정수 수정 가능 여부: 예 변경 적용: 즉시	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.
slave-allow-chaining	기본값: 아니요 유형: 문자열 수정 가능 여부: 아니요	Redis의 읽기 전용 복제본에 읽기 전용 복제본이 있을 OSS 수 있는지 여부를 결정합니다.

이름	Details	설명
`slowlog-log-slower-than`	기본값: 10000 유형: 정수 수정 가능 여부: 예 변경 적용: 즉시	Redis OSS Slow Log 기능에 의해 로깅되는 명령의 최대 실행 시간은 마이크로초입니다.
`slowlog-max-len`	기본값: 128 유형: 정수 수정 가능 여부: 예 변경 적용: 즉시	Redis OSS 느린 로그의 최대 길이입니다.
`tcp-keepalive`	기본값: 0 유형: 정수 수정 가능 여부: 예 변경 적용: 즉시	이 값을 0이 아닌 값(N)으로 설정하면 연결이 되어있는지 확인하기 위해 노드 클라이언트가 N초마다 폴링됩니다. 기본 설정인 0을 사용하면 폴링이 발생하지 않습니다. **Important** 이 파라미터의 일부 측면이 Redis OSS 버전 3.2.4에서 변경되었습니다. Redis OSS 3.2.4에서 변경된 파라미터(향상됨)을 참조하세요.

이름	Details	설명
`timeout`	기본값: 0 유형: 정수 수정 가능 여부: 예 변경 적용: 즉시	제한 시간이 지나기 전에 노드가 대기하는 시간(초)입니다. 유효한 값: • `0` – 유휴 클라이언트 연결을 절대로 끊지 마십시오. • `1-19` – 잘못된 값입니다. • `>=20` – 유휴 클라이언트 연결을 끊기 전에 노드가 대기하는 시간(초)입니다.
`zset-max-ziplist-entries`	기본값: 128 유형: 정수 수정 가능 여부: 예 변경 적용: 즉시	정렬된 세트에 사용할 메모리 양을 결정합니다. 지정된 수보다 적은 수의 정렬된 세트는 공간을 절약하는 특수 인코딩을 사용하여 저장됩니다.
`zset-max-ziplist-value`	기본값: 64 유형: 정수 수정 가능 여부: 예 변경 적용: 즉시	정렬된 세트에 사용할 메모리 양을 결정합니다. 지정된 바이트 수보다 작은 항목이 있는 정렬된 세트는 공간을 절약하는 특수 인코딩을 사용하여 저장됩니다.

> **Note**
>
> Redis OSS 2.6.13 클러스터에 파라미터 그룹을 지정하지 않으면 기본 파라미터 그룹(`default.redis2.6`)이 사용됩니다. 기본 파라미터 그룹의 파라미터 값은 변경할 수 없지만 언제든지 사용자 지정 파라미터 그룹을 생성하고 클러스터에 할당할 수 있습니다.

OSS 노드 유형별 파라미터 재정의

대부분의 파라미터는 단일 값을 갖지만 일부 파라미터는 사용하는 노드 유형에 따라 다양한 값을 갖습니다. 다음 표에는 각 노드 유형에 대한 `maxmemory`, `client-output-buffer-limit-slave-hard-limit` 및 `client-output-buffer-limit-slave-soft-limit` 파라미터의 기본값이 나와 있습니다. `maxmemory`의 값은 노드에서 데이터 및 다른 용도에 사용할 수 있는 최대 바이트 수입니다. 자세한 내용은 사용할 수 있는 메모리를 참조하세요.

> **Note**
>
> `maxmemory` 파라미터는 수정할 수 없습니다.

노드 유형	Maxmemory	C lient-output-buffe r-limit-slave-hard-limit	C lient-output-buffe r-limit-slave-soft-limit
cache.t1.micro	142606336	14260633	14260633
cache.t2.micro	581959680	58195968	58195968
cache.t2.small	1665138688	166513868	166513868
cache.t2.medium	3461349376	346134937	346134937
cache.t3.micro	536870912	53687091	53687091
cache.t3.small	1471026299	147102629	147102629
cache.t3.medium	3317862236	331786223	331786223
cache.t4g.micro	536870912	53687091	53687091
cache.t4g.small	1471026299	147102629	147102629
cache.t4g.medium	3317862236	331786223	331786223
cache.m1.small	943718400	94371840	94371840
cache.m1.medium	3093299200	309329920	309329920

노드 유형	Maxmemory	C lient-output-buffe r-limit-slave-hard-limit	C lient-output-buffe r-limit-slave-soft-limit
cache.m1.large	7025459200	702545920	702545920
cache.m1.xlarge	14889779200	1488977920	1488977920
cache.m2.xlarge	17091788800	1709178880	1709178880
cache.m2.2xlarge	35022438400	3502243840	3502243840
cache.m2.4xlarge	70883737600	7088373760	7088373760
cache.m3.medium	2988441600	309329920	309329920
cache.m3.large	6501171200	650117120	650117120
cache.m3.xlarge	14260633600	1426063360	1426063360
cache.m3.2xlarge	29989273600	2998927360	2998927360
cache.m4.large	6892593152	689259315	689259315
cache.m4.xlarge	15328501760	1532850176	1532850176
cache.m4.2xlarge	31889126359	3188912636	3188912636
cache.m4.4xlarge	65257290629	6525729063	6525729063
cache.m4.10xlarge	166047614239	16604761424	16604761424
cache.m5.large	6854542746	685454275	685454275
cache.m5.xlarge	13891921715	1389192172	1389192172
cache.m5.2xlarge	27966669210	2796666921	2796666921
cache.m5.4xlarge	56116178125	5611617812	5611617812
cache.m5.12xlarge	168715971994	16871597199	16871597199

노드 유형	Maxmemory	C lient-output-buffe r-limit-slave-hard-limit	C lient-output-buffe r-limit-slave-soft-limit
cache.m5.24xlarge	337500562842	33750056284	33750056284
cache.m6g.large	6854542746	685454275	685454275
cache.m6g.xlarge	13891921715	1389192172	1389192172
cache.m6g.2xlarge	27966669210	2796666921	2796666921
cache.m6g.4xlarge	56116178125	5611617812	5611617812
cache.m6g.8xlarge	111325552312	11132555231	11132555231
cache.m6g.12xlarge	168715971994	16871597199	16871597199
cache.m6g.16xlarge	225000375228	22500037523	22500037523
cache.c1.xlarge	6501171200	650117120	650117120
cache.r3.large	14470348800	1468006400	1468006400
cache.r3.xlarge	30513561600	3040870400	3040870400
cache.r3.2xlarge	62495129600	6081740800	6081740800
cache.r3.4xlarge	126458265600	12268339200	12268339200
cache.r3.8xlarge	254384537600	24536678400	24536678400
cache.r4.large	13201781556	1320178155	1320178155
cache.r4.xlarge	26898228839	2689822883	2689822883
cache.r4.2xlarge	54197537997	5419753799	5419753799
cache.r4.4xlarge	108858546586	10885854658	10885854658
cache.r4.8xlarge	218255432090	21825543209	21825543209

노드 유형	Maxmemory	C lient-output-buffe r-limit-slave-hard-limit	C lient-output-buffe r-limit-slave-soft-limit
cache.r4.16xlarge	437021573120	43702157312	43702157312
cache.r5.large	14037181030	1403718103	1403718103
cache.r5.xlarge	28261849702	2826184970	2826184970
cache.r5.2xlarge	56711183565	5671118356	5671118356
cache.r5.4xlarge	113609865216	11360986522	11360986522
cache.r5.12xlarge	341206346547	34120634655	34120634655
cache.r5.24xlarge	682485973811	68248597381	68248597381
cache.r6g.large	14037181030	1403718103	1403718103
cache.r6g.xlarge	28261849702	2826184970	2826184970
cache.r6g.2xlarge	56711183565	5671118356	5671118356
cache.r6g.4xlarge	113609865216	11360986522	11360986522
cache.r6g.8xlarge	225000375228	22500037523	22500037523
cache.r6g.12xlarge	341206346547	34120634655	34120634655
cache.r6g.16xlarge	450000750456	45000075046	45000075046
cache.r6gd.xlarge	28261849702	2826184970	2826184970
cache.r6gd.2xlarge	56711183565	5671118356	5671118356
cache.r6gd.4xlarge	113609865216	11360986522	11360986522
cache.r6gd.8xlarge	225000375228	22500037523	22500037523
cache.r6gd.12xlarge	341206346547	34120634655	34120634655

노드 유형	Maxmemory	C lient-output-buffe r-limit-slave-hard-limit	C lient-output-buffe r-limit-slave-soft-limit
cache.r6gd.16xlarge	450000750456	45000075046	45000075046
cache.r7g.large	14037181030	1403718103	1403718103
cache.r7g.xlarge	28261849702	2826184970	2826184970
cache.r7g.2xlarge	56711183565	5671118356	5671118356
cache.r7g.4xlarge	113609865216	11360986522	11360986522
cache.r7g.8xlarge	225000375228	22500037523	22500037523
cache.r7g.12xlarge	341206346547	34120634655	34120634655
cache.r7g.16xlarge	450000750456	45000075046	45000075046
cache.m7g.large	6854542746	685454275	685454275
cache.m7g.xlarge	13891921715	1389192172	1389192172
cache.m7g.2xlarge	27966669210	2796666921	2796666921
cache.m7g.4xlarge	56116178125	5611617812	5611617812
cache.m7g.8xlarge	111325552312	11132555231	11132555231
cache.m7g.12xlarge	168715971994	16871597199	16871597199
cache.m7g.16xlarge	225000375228	22500037523	22500037523
cache.c7gn.large	3317862236	1403718103	1403718103
cache.c7gn.xlarge	6854542746	2826184970	2826184970
cache.c7gn.2xlarge	13891921715	5671118356	5671118356
cache.c7gn.4xlarge	27966669210	11360986522	11360986522

노드 유형	Maxmemory	C lient-output-buffe r-limit-slave-hard- limit	C lient-output-buffe r-limit-slave-soft- limit
cache.c7gn.8xlarge	56116178125	22500037523	22500037523
cache.c7gn.12xlarge	84357985997	34120634655	34120634655
cache.c7gn.16xlarge	113609865216	45000075046	45000075046

> **Note**
>
> 모든 현재 세대 인스턴스 유형은 VPC 기본적으로 Amazon Virtual Private Cloud에서 생성됩니다.
> T1 인스턴스는 다중 AZ를 지원하지 않습니다.
> T1 및 T2 인스턴스는 Redis OSS 를 지원하지 않습니다AOF.
> Redis OSS 구성 변수 `appendonly` 및 `appendfsync`는 Redis OSS 버전 2.8.22 이상에서는 지원되지 않습니다.

Memcached 특정 파라미터

Memcached

Memcached 클러스터에 파라미터 그룹을 지정하지 않으면 엔진 버전에 적절한 기본 파라미터 그룹이 사용됩니다. 기본 파라미터 그룹에서는 어떤 파라미터 값도 변경할 수 없습니다. 그러나 사용자 지정 파라미터 그룹을 생성하여 언제든지 클러스터에 할당할 수 있습니다. 자세한 내용은 ElastiCache 파라미터 그룹 생성 단원을 참조하십시오.

주제

- Memcached 1.6.17 변경 사항
- Memcached 1.6.6 추가 파라미터
- Memcached 1.5.10 파라미터 변경
- Memcached 1.4.34 추가 파라미터
- Memcached 1.4.33 추가 파라미터
- Memcached 1.4.24 추가 파라미터

- Memcached 1.4.14 추가 파라미터
- Memcached 1.4.5 지원 파라미터
- Memcached 연결 오버헤드
- Memcached 노드 유형별 파라미터

Memcached 1.6.17 변경 사항

Memcached 1.6.17부터는 `lru_crawler`, `lru`, `slabs` 관리 명령을 더 이상 지원하지 않습니다. 이러한 변경으로 인해 런타임에 명령을 통해 `lru_crawler`를 활성화하거나 비활성화할 수 없습니다. 사용자 지정 파라미터 그룹을 수정하여 `lru_crawler`를 활성화하거나 비활성화하세요.

Memcached 1.6.6 추가 파라미터

Memcached 1.6.6은 추가 파라미터를 지원하지 않습니다.

파라미터 그룹 패밀리: memcached1.6

Memcached 1.5.10 파라미터 변경

Memcached 1.5.10은 다음과 같은 추가 파라미터가 지원됩니다.

파라미터 그룹 Family: memcached1.5

이름	Details	설명
`no_modern`	기본값: 1 유형: boolean 수정 가능 여부: 예 허용된 값: 0,1 변경 적용: 시작 시	`slab_reassign` , `lru_maintainer_thread` , `lru_segmented` 및`maxconns_fast` 명령을 비활성화하기 위한 별칭입니다. Memcached 1.5 이상을 사용하는 경우 `no_modern` 도 hash_algorithm을 로 설정합니다`jenkins`. 또한 Memcached 1.5.10를 사용하는 경우 `inline_ascii_repon`

이름	Details	설명
		se 는 파라미터 에 의해 제어 됩니다parallelly . 즉, 가 no_modern 비활성화되면 inline_ascii_reponse 가 비활성화됩니다. Memcached 엔진 1.5.16부터 inline_as cii_response 파라미터가 더 이상 적용되지 않으므로 를 활 성화하거나 비활성화no_modern 해도 에 영향을 주지 않습니 다inline_ascii_reponse .
		no_modern 가 비활성화되면 , slab_reassign lru_maint ainer_thread , lru_segme nted , 가 maxconns_fast WILL 활성화됩니다. slab_auto move 및 hash_algorithm 파 라미터는 SWITCH 파라미터가 아 니기 때문에 파라미터 설정은 파라 미터 그룹의 구성을 기반으로 합니 다. 비활성화no_modern 하고 로 되돌 리려면 이 파라미터를 비활성화하 도록 사용자 지정 파라미터 그룹을 구성한 다음 이러한 변경 사항이 적 용되도록 재부팅해야 modern합니 다. **Note** 이 파라미터의 기본 구성 값은 2021년 8월 20일 현재 0에서 1로 변경되었습니다.

이름	Details	설명
		업데이트된 기본값은 2021년 8월 20일 이후 각 리전의 새 ElastiCache 사용자가 자동으로 선택합니다. 2021년 8월 20일 이전에 리전에 있는 기존 ElastiCache 사용자는 이 새로운 변경 사항을 적용하려면 사용자 지정 파라미터 그룹을 수동으로 수정해야 합니다.
`inline_ascii_resp`	기본값: 0 유형: boolean 수정 가능 여부: 예 허용된 값: 0,1 변경 적용: 시작 시	최대 24바이트를 사용하여 항목 내 VALUE 응답의 수치를 저장합니다. `get`, `faster` 세트ASCII에 대한 작은 속도 저하.

Memcached 1.5.10의 경우 다음과 같은 파라미터가 제거됩니다.

이름	Details	설명
`expirezero_does_not_evict`	기본값: 0 유형: boolean 수정 가능 여부: 예 허용된 값: 0,1 변경 적용: 시작 시	이 버전에서는 이제 지원하지 않습니다.
`modern`	기본값: 1	

이름	Details	설명
	유형: boolean 수정 가능 여부: 예 (no_modern 으로 설정하는 경우 재시작해야 함) 허용된 값: 0,1 변경 적용: 시작 시	이 버전에서는 이제 지원하지 않습니다. 이 버전부터는 시작할 때마다 항상 또는 재시작 시 기본적으로 no-modern 이 활성화됩니다.

Memcached 1.4.34 추가 파라미터

Memcached 1.4.34는 추가 파라미터를 지원하지 않습니다.

파라미터 그룹 패밀리: memcached1.4

Memcached 1.4.33 추가 파라미터

For Memcached 1.4.33은 다음과 같은 추가 파라미터가 지원됩니다.

파라미터 그룹 패밀리: memcached1.4

이름	Details	설명
modern	기본값: enabled 유형: boolean 수정 가능 여부: 예 변경 적용: 시작 시	여러 기능의 별칭입니다. modern을 활성화하는 것은 murmur3 해시 알고리즘을 사용하고 다음 명령을 사용하는 것과 같습니다. slab_reassign , slab_automove , lru_crawler , lru_maintainer , maxconns_fast 및 hash_algorithm=murmur3
watch	기본값: enabled	

이름	Details	설명
	유형: boolean 수정 가능 여부: 예 변경 적용: 즉시 사용자가 watcher_logbuf_size 및 worker_logbuf_size 한도에 도달하면 로그가 삭제될 수 있습니다.	로그 가져오기, 제거 또는 변형. 예를 들어 사용자가 watch를 켜면 get, set, delete 또는 update 발생 시 로그를 볼 수 있습니다.
idle_timeout	기본값: 0(비활성화) 유형: 정수 수정 가능 여부: 예 변경 적용: 시작 시	종료하라는 메시지가 표시되기 전에 클라이언트가 유휴 상태로 있을 수 있는 최소 시간(초)입니다. 값의 범위는 0~86400입니다.
track_sizes	기본값: 비활성화 유형: boolean 수정 가능 여부: 예 변경 적용: 시작 시	슬래브 그룹이 소비한 크기를 표시합니다. track_sizes 를 활성화하면 stats sizes_enable 을 실행할 필요 없이 stats sizes 를 실행할 수 있습니다.

이름	Details	설명
watcher_logbuf_size	기본값: 256(KB) 유형: 정수 수정 가능 여부: 예 변경 적용: 시작 시	watch 명령은 Memcached에 대한 스트림 로깅을 켭니다. 그러나 로깅 버퍼가 가득 찰 정도로 제거, 변형 또는 가져오기 비율이 높은 경우 watch에서 로그를 삭제할 수 있습니다. 이러한 상황에서 사용자는 로그 손실을 줄이기 위해 버퍼 크기를 늘릴 수 있습니다.
worker_logbuf_size	기본값: 64(KB) 유형: 정수 수정 가능 여부: 예 변경 적용: 시작 시	watch 명령은 Memcached에 대한 스트림 로깅을 켭니다. 그러나 버퍼가 가득 찰 정도로 제거, 변형 또는 가져오기 비율이 높으면 watch는 로그를 삭제할 수 있습니다. 이러한 상황에서 사용자는 로그 손실을 줄이기 위해 버퍼 크기를 늘릴 수 있습니다.
slab_chunk_max	기본값: 524288(바이트) 유형: 정수 수정 가능 여부: 예 변경 적용: 시작 시	슬래브의 최대 크기를 지정합니다. 슬래브 크기를 작게 설정하면 메모리를 더 효율적으로 사용합니다. slab_chunk_max 보다 큰 항목은 여러 슬래브로 분할됩니다.
lru_crawler metadump [all|1|2|3]	기본값: 비활성화 유형: boolean 수정 가능 여부: 예 변경 적용: 즉시	lru_crawler를 활성화하면 이 명령이 모든 키를 덤프합니다. all|1|2|3 - 모든 슬래브 또는 특정 슬래브 수 지정

Memcached 1.4.24 추가 파라미터

Memcached 1.4.24는 다음과 같은 추가 파라미터가 지원됩니다.

파라미터 그룹 패밀리: memcached1.4

이름	Details	설명
`disable_flush_all`	기본값: 0(비활성화) 유형: boolean 수정 가능 여부: 예 변경 적용: 시작 시	flush_all을 비활성화하려면 파라미터(`-F`)를 추가합니다. 프로덕션 인스턴스에서 전체 플러시를 실행할 수 없는 경우에 유용합니다. 값은 0, 1(값이 0일 때 사용자가 `flush_all` 을 수행할 수 있음)입니다.
`hash_algorithm`	기본값: jenkins 유형: 문자열 수정 가능 여부: 예 변경 적용: 시작 시	사용할 해시 알고리즘입니다. 허용되는 값은 murmur3 및 jenkins입니다.
`lru_crawler`	기본값: 0(비활성화) 유형: boolean 수정 가능 여부: 예 변경 적용: 재시작 후 **Note** 런타임 시 명령줄에서 `lru_crawler` 를 일시적으로 활성화할	만료된 항목의 슬래브 클래스를 삭제합니다. 백그라운드에서 실행되는 영향이 적은 프로세스입니다. 현재는 수동 명령을 사용하여 크롤링을 시작해야 합니다. 일시적으로 활성화하려면 명령줄에서 `lru_crawler enable` 을 실행합니다. `lru_crawler 1,3,5` 는 슬래브 클래스 1, 3 및 5를 크롤링하여 freelist에 추가할 만료 항목을 찾습니다.

이름	Details	설명
	수 있습니다. 자세한 내용은 설명 열을 참조하세요.	값: 0,1 **Note** 명령줄에서 `lru_crawler` 를 활성화하면 명령줄에서 비활성화하거나 다음에 재부팅될 때까지 크롤러가 활성화됩니다. 영구적으로 활성화하려면 파라미터 값을 수정해야 합니다. 자세한 내용은 ElastiCache 파라미터 그룹 수정 단원을 참조하십시오.
`lru_maintainer`	기본값: 0(비활성화) 유형: boolean 수정 가능 여부: 예 변경 적용: 시작 시	용량 LRUs 단위 간에 항목을 셔플링하는 백그라운드 스레드에 도달했습니다. 값: 0, 1

이름	Details	설명
`expirezero_does_not_evict`	기본값: 0(비활성화) 유형: boolean 수정 가능 여부: 예 변경 적용: 시작 시	`lru_maintainer` 와 함께 사용하면 만료 시간이 0인 항목을 제거할 수 없게 합니다. **Warning** 기타 제거할 수 있는 항목에 사용 가능한 메모리를 밀어낼 수 있습니다. `lru_maintainer` 를 무시하도록 설정할 수 있습니다.

Memcached 1.4.14 추가 파라미터

Memcached 1.4.14는 다음과 같은 추가 파라미터가 지원됩니다.

파라미터 그룹 패밀리: memcached1.4

Memcached 1.4.14에 추가된 파라미터

이름	설명
`config_max`	ElastiCache 구성 항목의 최대 수입니다.

이름	설명
config_size_max	구성 항목의 최대 크기(바이트)입니다.

이름	설명
hashpower_init	ElastiCache 해시 테이블의 초기 크기로, 2의 출력으로 표시됩니다. 기본값은 16(2^16) 또는 65536 키입니다.

이름	설명
maxconns_fast	최대 연결 한도에 도달했을 때 새 연결 요청을 처리하는 방식을 변경합니다. 이 파라미터를 0으로 설정하면 새 연결이 백로그 대기열에 추가되고 다른 연결이 끊길 때까지 대기합니다. 파라미터가 1로 설정된 경우 클라이언트에 오류를 ElastiCache 보내고 연결을 즉시 닫습니다.

이름	설명
slab_automove	슬래브 오토무브 알고리즘을 조정합니다. 이 파라미터를 0(영)으로 설정하면 오토무브 알고리즘이 비활성화됩니다. 1로 설정하면 ElastiCache 는 슬래브를 자동으로 이동하는 느리고 보수적인 접근 방식을 취합니다. 2로 설정하면 제거될 때마다 슬래브가 ElastiCache 공격적으로 이동합니다. (이 모드는 테스트 목적을 제외하고는 권장되지 않음)

이름	설명
slab_reassign	슬래브 재할당을 활성화하거나 비활성화합니다. 이 파라미터를 1로 설정하면 "슬래브 재할당" 명령을 사용하여 메모리를 수동으로 재할당할 수 있습니다.

Memcached 1.4.5 지원 파라미터

파라미터 그룹 패밀리: memcached1.4

Memcached 1.4.5는 다음과 같은 파라미터를 지원합니다.

Memcached 1.4.5에 추가된 파라미터

이름	Details	설명
`backlog_queue_limit`	기본값: 1024 유형: 정수 수정 가능 여부: 아니요	백 로그 대기열 제한입니다.
`binding_protocol`	기본값: 자동 유형: 문자열 수정 가능 여부: 예 변경 적용: 재시작 후	바인딩 프로토콜입니다. 허용 가능한 값은 `ascii` 및 `auto`입니다. `binding_protocol` 의 값을 수정하는 방법에 대한 지침은 ElastiCache 파라미터 그룹 수정을 참조하세요.
`cas_disabled`	기본값: 0(false) 유형: 부울 수정 가능 여부: 예 변경 적용: 재시작 후	1 (true)인 경우 확인 및 설정(CAS) 작업이 비활성화되고 저장된 항목은 CAS 활성화된 경우보다 8바이트를 적게 소비합니다.
`chunk_size`	기본값: 48 유형: 정수 수정 가능 여부: 예 변경 적용: 재시작 후	가장 작은 항목의 키, 값 및 플래그에 할당할 공간의 최소 크기(바이트)입니다.
`chunk_size_growth_factor`	기본값: 1.25 유형: float 수정 가능 여부: 예 변경 적용: 재시작 후	연속된 Memcached 청크 크기를 제어하는 성장 인자입니다. 각 청크는 이전 청크보다 `chunk_size_growth_factor` 배 더 큽니다.

이름	Details	설명
error_on_memory_exhausted	기본값: 0(false) 유형: 부울 수정 가능 여부: 예 변경 적용: 재시작 후	1(true)이면 항목을 저장할 메모리가 없으면 Memcached가 항목을 제거하는 대신 오류를 반환합니다.
large_memory_pages	기본값: 0(false) 유형: 부울 수정 가능 여부: 아니요	1 (true)인 경우 큰 메모리 페이지를 사용하려고 ElastiCache 합니다.
lock_down_paged_memory	기본값: 0(false) 유형: 부울 수정 가능 여부: 아니요	1 (true)인 경우 페이지가 지정된 모든 메모리 ElastiCache 를 잠급니다.
max_item_size	기본값: 1048576 유형: 정수 수정 가능 여부: 예 변경 적용: 재시작 후	클러스터에 저장할 수 있는 가장 큰 항목의 크기(바이트)입니다.
max_simultaneous_connections	기본값: 65000 유형: 정수 수정 가능 여부: 아니요	최대 동시 연결 수입니다.
maximize_core_file_limit	기본값: 0(false) 유형: 부울 수정 가능: 변경 적용: 재시작 후	1 (true)인 경우 ElastiCache 는 코어 파일 제한을 최대화합니다.

이름	Details	설명
memcached_connections_overhead	기본값: 100 유형: 정수 수정 가능 여부: 예 변경 적용: 재시작 후	Memcached 연결 및 기타 오버헤드에 예약된 메모리 양입니다. 이 파라미터에 대한 자세한 정보는 Memcached 연결 오버헤드를 참조하세요.
requests_per_event	기본값: 20 유형: 정수 수정 가능 여부: 아니요	지정된 연결의 이벤트 당 최대 요청 수입니다. 이 제한은 리소스 결핍을 막기 위해 필요합니다.

Memcached 연결 오버헤드

각 노드에서 항목을 저장하는 데 사용할 수 있는 메모리는 max_cache_memory 파라미터에 저장된 해당 노드의 총 사용 가능한 메모리에서 memcached_connections_overhead 파라미터에 저장된 연결에 사용하는 메모리와 기타 오버헤드를 뺀 값입니다. 예를 들어, cache.m1.small 유형의 노드에는 1300MB의 max_cache_memory가 있습니다. 기본 memcached_connections_overhead 값이 100MB이면 Memcached 프로세스는 항목을 저장하는 데 1200MB를 사용할 수 있습니다.

memcached_connections_overhead 파라미터의 기본값은 대부분의 사용 사례를 충족시키지만 연결 오버헤드에 필요한 할당량은 요청 빈도, 페이로드 크기 및 연결 수를 비롯한 여러 요소에 따라 달라질 수 있습니다.

애플리케이션에 맞게 memcached_connections_overhead 값을 변경할 수 있습니다. 예를 들어, memcached_connections_overhead 파라미터 값을 높이면 항목을 저장하는 데 사용할 수 있는 메모리 양이 줄어들어 연결 오버헤드에 더 큰 버퍼가 제공됩니다. memcached_connections_overhead 파라미터 값을 줄이면 항목을 저장하는 데 더 많은 메모리를 사용할 수 있지만 스왑 사용량 및 성능 저하 위험이 높아질 수 있습니다. 스왑 사용량이 늘고 성능 저하가 발생하면 memcached_connections_overhead 파라미터 값을 늘립니다.

> **Important**
>
> cache.t1.micro 노드 유형의 경우 memcached_connections_overhead 값은 다음과 같이 결정됩니다.

- 클러스터가 기본 파라미터 그룹을 사용하는 경우 ElastiCache 는 의 값을 13MB `memcached_connections_overhead`로 설정합니다.
- 클러스터가 사용자가 직접 생성한 파라미터 그룹을 사용하면 `memcached_connections_overhead` 값을 원하는 대로 설정할 수 있습니다.

Memcached 노드 유형별 파라미터

대부분의 파라미터는 단일 값을 갖지만 일부 파라미터는 사용하는 노드 유형에 따라 다양한 값을 갖습니다. 다음 표에는 각 노드 유형에 대한 `max_cache_memory` 및 `num_threads` 파라미터의 기본값이 나와 있습니다. 이 파라미터의 값은 수정할 수 없습니다.

노드 유형	max_cache_memory(MB)	num_threads
cache.t1.micro	213	1
cache.t2.micro	555	1
cache.t2.small	1588	1
cache.t2.medium	3301	2
cache.t3.micro	512	2
cache.t3.small	1402	2
cache.t3.medium	3364	2
cache.t4g.micro	512	2
cache.t4g.small	1402	2
cache.t4g.medium	3164	2
cache.m1.small	1301	1
cache.m1.medium	3350	1
cache.m1.large	7100	2

노드 유형	max_cache_memory(MB)	num_threads
cache.m1.xlarge	14600	4
cache.m2.xlarge	33800	2
cache.m2.2xlarge	30412	4
cache.m2.4xlarge	68000	16
cache.m3.medium	2850	1
cache.m3.large	6200	2
cache.m3.xlarge	13600	4
cache.m3.2xlarge	28600	8
cache.m4.large	6573	2
cache.m4.xlarge	11496	4
cache.m4.2xlarge	30412	8
cache.m4.4xlarge	62234	16
cache.m4.10xlarge	158355	40
cache.m5.large	6537	2
cache.m5.xlarge	13248	4
cache.m5.2xlarge	26671	8
cache.m5.4xlarge	53516	16
cache.m5.12xlarge	160900	48
cache.m5.24xlarge	321865	96
cache.m6g.large	6537	2

노드 유형	max_cache_memory(MB)	num_threads
cache.m6g.xlarge	13248	4
cache.m6g.2xlarge	26671	8
cache.m6g.4xlarge	53516	16
cache.m6g.8xlarge	107000	32
cache.m6g.12xlarge	160900	48
cache.m6g.16xlarge	214577	64
cache.c1.xlarge	6600	8
cache.r3.large	13800	2
cache.r3.xlarge	29100	4
cache.r3.2xlarge	59600	8
cache.r3.4xlarge	120600	16
cache.r3.8xlarge	120600	32
cache.r4.large	12590	2
cache.r4.xlarge	25652	4
cache.r4.2xlarge	51686	8
cache.r4.4xlarge	103815	16
cache.r4.8xlarge	208144	32
cache.r4.16xlarge	416776	64
cache.r5.large	13387	2
cache.r5.xlarge	26953	4

노드 유형	max_cache_memory(MB)	num_threads
cache.r5.2xlarge	54084	8
cache.r5.4xlarge	108347	16
cache.r5.12xlarge	325400	48
cache.r5.24xlarge	650869	96
cache.r6g.large	13387	2
cache.r6g.xlarge	26953	4
cache.r6g.2xlarge	54084	8
cache.r6g.4xlarge	108347	16
cache.r6g.8xlarge	214577	32
cache.r6g.12xlarge	325400	48
cache.r6g.16xlarge	429154	64
cache.c7gn.large	3164	2
cache.c7gn.xlarge	6537	4
cache.c7gn.2xlarge	13248	8
cache.c7gn.4xlarge	26671	16
cache.c7gn.8xlarge	53516	32
cache.c7gn.12xlarge	325400	48
cache.c7gn.16xlarge	108347	64

> **Note**
>
> 모든 T2 인스턴스는 Amazon Virtual Private Cloud(Amazon)에서 생성됩니다VPC.

크기 조정 ElastiCache

필요에 따라 ElastiCache 캐시를 확장할 수 있습니다. 서버리스 캐시와 자체 설계된 클러스터는 다양한 크기 조정 옵션을 제공합니다.

ElastiCache 서버리스 확장

ElastiCache Serverless는 워크로드 트래픽이 증가하거나 감소할 때 워크로드 트래픽을 자동으로 수용합니다. 각 ElastiCache 서버리스 캐시에 대해 는 , CPU메모리 및 네트워크와 같은 리소스의 사용률을 ElastiCache 지속적으로 추적합니다. 이러한 리소스가 제한되면 ElastiCache Serverless는 애플리케이션의 가동 중지 없이 새 샤드를 추가하고 새 샤드에 데이터를 다시 배포하여 확장합니다. 캐시 데이터 스토리지에 대한 지표와 컴퓨팅 사용에 대한 `ElastiCacheProcessingUnits` (ECPU)를 모니터링 CloudWatch 하여 `BytesUsedForCache` 에서 캐시가 사용하는 리소스를 모니터링할 수 있습니다.

비용 관리를 위한 규모 조정 한도 설정

캐시 데이터 스토리지에서 최대 사용량을 구성하고 캐시 비용을 제어하기 위해 캐시에 대해 ECPU/초를 구성하도록 선택할 수 있습니다. 이렇게 하면 캐시 사용량이 구성된 최댓값을 초과하지 않도록 할 수 있습니다.

크기 조정 최대값을 설정하면 캐시가 최대값에 도달하면 애플리케이션이 캐시 성능 저하를 경험할 수 있습니다. 캐시 데이터 스토리지 최대값을 설정하고 캐시 데이터 스토리지가 최대값에 도달하면 LRU ElastiCache 는 로직을 Time-To-Live 사용하여 (TTL)가 설정된 캐시의 데이터 제거를 시작합니다. 제거할 수 있는 데이터가 없는 경우 추가 데이터 쓰기 요청은 Out Of Memory(OOM) 오류 메시지를 받게 됩니다. ECPU/초 최대값을 설정하고 워크로드의 컴퓨팅 사용률이 이 값을 초과하면 ElastiCache 는 요청 제한을 시작합니다.

`BytesUsedForCache` 또는 에 최대 제한을 설정하는 경우 캐시가 이러한 제한에 가깝게 작동할 때 알림을 받을 수 있도록 최대 제한보다 낮은 값으로 CloudWatch 경보를 설정하는 `ElastiCacheProcessingUnits`것이 좋습니다. 설정한 최대 한도의 75%로 경보를 설정하는 것이 좋습니다. CloudWatch 경보 설정 방법에 대한 설명서를 참조하세요.

ElastiCache Serverless를 사용한 사전 크기 조정

ElastiCache 서버리스 사전 크기 조정

예열이라고도 하는 사전 크기 조정을 사용하면 ElastiCache 캐시에 지원되는 최소 제한을 설정할 수 있습니다. 초당 ElastiCache 처리 단위(ECPUs) 또는 데이터 스토리지에 대해 이러한 최소값을 설정할 수 있습니다. 이는 예상 조정 이벤트를 준비하는 데 유용할 수 있습니다. 예를 들어 게임 회사가 새 게임이 시작되는 첫 1분 내에 로그인이 5배 증가할 것으로 예상하는 경우 이러한 상당한 사용량 증가를 위해 캐시를 준비할 수 있습니다.

ElastiCache 콘솔, 또는 CLI를 사용하여 사전 크기 조정을 수행할 수 있습니다API. ElastiCache Serverless는 60분 이내에 캐시에서 사용 가능한 ECPUs/초를 업데이트하고 최소 한도 업데이트가 완료되면 이벤트 알림을 보냅니다.

사전 크기 조정 작동 방식

콘솔, CLI또는 를 통해 ECPUs/초 또는 데이터 스토리지에 대한 최소 제한이 업데이트되면 API1시간 이내에 새 제한을 사용할 수 있습니다. ElastiCache Serverless는 빈 캐시에서 30KECPUs/초를 지원하고 복제본에서 읽기 기능을 사용할 때는 최대 90KECPUs/초를 지원합니다. 는 10~12분마다 ECPUs/초를 두 배로 할 ElastiCache 수 있습니다. 이 조정 속도는 대부분의 워크로드에 충분합니다. 향후 스케일링 이벤트가 이 속도를 초과할 것으로 예상되는 경우, 최소 ECPUs/초를 최대 이벤트 최소 60분 전에 예상되는 최대 ECPUs/초로 설정하는 것이 좋습니다. 그렇지 않으면 애플리케이션의 지연 시간과 요청 제한이 증가할 수 있습니다.

최소 한도 업데이트가 완료되면 ElastiCache Serverless는 초ECPUs당 새 최소 또는 새 최소 스토리지에 대한 측정 작업을 시작합니다. 이는 애플리케이션이 캐시에서 요청을 실행하지 않거나 데이터 스토리지 사용량이 최소값 미만인 경우에도 발생합니다. 현재 설정에서 최소 한도를 낮추면 업데이트가 즉시 이루어지므로 ElastiCache Serverless는 새 최소 한도에서 즉시 측정이 시작됩니다.

> **Note**
>
> - 최소 사용량 한도를 설정하면 실제 사용량이 최소 사용량 한도보다 낮더라도 해당 한도에 대한 요금이 부과됩니다. ECPU 또는 최소 사용량 제한을 초과하는 데이터 스토리지 사용량에는 정규 요금이 부과됩니다. 예를 들어 최소 사용량 제한을 초ECPUs당 100,000으로 설정하면 사용량이 최소 설정보다 낮더라도 시간당 최소 1.224달러(us-east-1의 ECPU 요금 사용)가 부과됩니다.
> - ElastiCache Serverless는 캐시의 집계 수준에서 요청된 최소 규모를 지원합니다. ElastiCache Serverless는 또한 슬롯당 최대 30KECPUs/초(READONLY연결을 사용하여 복

제본에서 읽기를 사용하는 경우 90KECPUs/초)를 지원합니다. 모범 사례로 애플리케이션은 Valkey 또는 Redis OSS 슬롯 간 키 배포와 키 간 트래픽이 가능한 한 균일해야 합니다.

콘솔 및 를 사용하여 조정 제한 설정 AWS CLI

AWS 콘솔을 사용하여 조정 제한 설정

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서, 수정하려는 캐시에서 실행 중인 엔진을 선택합니다.
3. 선택한 엔진을 실행하는 캐시 목록이 표시됩니다.
4. 캐시 이름 왼쪽에 있는 라디오 버튼을 선택하여 수정할 캐시를 선택합니다.
5. 작업을 선택한 다음 수정을 선택합니다.
6. 사용량 한도에서 적절한 메모리 또는 컴퓨팅 한도를 설정합니다.
7. 변경 사항 미리보기를 클릭한 다음 변경 사항 저장을 클릭합니다.

를 사용하여 조정 제한 설정 AWS CLI

를 사용하여 조정 한도를 변경하려면 를 CLI modify-serverless-cache 사용합니다API.

Linux:

```
aws elasticache modify-serverless-cache --serverless-cache-name <cache name> \
--cache-usage-limits 'DataStorage={Minimum=10,Maximum=100,Unit=GB},
 ECPUPerSecond={Minimum=1000,Maximum=100000}'
```

Windows:

```
aws elasticache modify-serverless-cache --serverless-cache-name <cache name> ^
--cache-usage-limits 'DataStorage={Minimum=10,Maximum=100,Unit=GB},
 ECPUPerSecond={Minimum=1000,Maximum=100000}'
```

를 사용하여 조정 제한 제거 CLI

를 사용하여 조정 제한을 제거하려면 최소 및 최대 제한 파라미터를 0으로 CLI설정합니다.

Linux:

```
aws elasticache modify-serverless-cache --serverless-cache-name <cache name> \
--cache-usage-limits 'DataStorage={Minimum=0,Maximum=0,Unit=GB},
 ECPUPerSecond={Minimum=0,Maximum=0}'
```

Windows:

```
aws elasticache modify-serverless-cache --serverless-cache-name <cache name> ^
--cache-usage-limits 'DataStorage={Minimum=0,Maximum=0,Unit=GB},
 ECPUPerSecond={Minimum=0,Maximum=0}'
```

자체 설계된 클러스터 크기 조정

애플리케이션에서 처리해야 하는 데이터의 양은 거의 정적이 아닙니다. 비즈니스가 성장하거나 수요에서 일반적인 변동을 경험할 경우, 데이터의 양이 증가하거나 감소합니다. 캐시를 자체적으로 관리할 경우 최고의 수요에 대해 충분한 하드웨어를 프로비저닝해야 하므로, 비용이 많이 들 수 있습니다. Amazon을 사용하면 현재 수요에 맞게 확장 ElastiCache 하여 사용하는 만큼만 지불할 수 있습니다. 를 ElastiCache 사용하면 수요에 맞게 캐시를 확장할 수 있습니다.

> **Note**
>
> Valkey 또는 Redis OSS 클러스터가 하나 이상의 리전에 복제되는 경우 해당 리전은 순서대로 조정됩니다. 스케일 업 시 보조 리전은 먼저 스케일 업된 다음 기본 리전으로 스케일 업됩니다. 규모를 축소할 때 기본 리전이 먼저 되고 그 다음 보조 리전이 따릅니다.
> 엔진 버전을 업데이트할 때 순서는 보조 리전과 기본 리전 순입니다.

주제

- Memcached용 클러스터 크기 조정
- Valkey 또는 Redis용 클러스터 크기 조정OSS(클러스터 모드 비활성화됨)
- Valkey 또는 Redis용 복제본 노드 크기 조정OSS(클러스터 모드 비활성화됨)
- Valkey 또는 Redis에서 클러스터 크기 조정OSS(클러스터 모드 활성화됨)

Memcached용 클러스터 크기 조정

다음 표는 수행하려는 조정 작업에 대한 올바른 주제를 찾는 데 도움이 될 수 있습니다.

Memcached 클러스터 크기 조정

Memcached 클러스터는 1~60개의 노드로 구성됩니다. Memcached 클러스터를 스케일 아웃하고 스케일 인하는 것은 클러스터에서 노드를 추가 또는 제거하는 것만큼 쉽습니다.

Memcached 클러스터의 모든 노드로 데이터를 분할할 수 있으므로 더 많은 양의 메모리가 있는 노드 유형으로 스케일 업할 필요가 거의 없습니다. 그러나 Memcached 엔진이 데이터를 유지하지 않으므로 다른 노드 유형으로 조정할 경우 애플리케이션에서 채우지 않으면 새 클러스터가 비워집니다.

Memcached 클러스터 조정

작업	주제
확장	클러스터에 노드 추가
축소	클러스터에서 노드 삭제
노드 유형 변경	수직으로 Memcached 조정

Memcached 클러스터는 1~60개의 노드로 구성됩니다. Memcached 클러스터를 스케일 아웃하고 스케일 인하는 것은 클러스터에서 노드를 추가 또는 제거하는 것만큼 쉽습니다.

Memcached 클러스터의 모든 노드로 데이터를 분할할 수 있으므로 더 많은 양의 메모리가 있는 노드 유형으로 스케일 업할 필요가 거의 없습니다. 그러나 Memcached 엔진이 데이터를 유지하지 않으므로 다른 노드 유형으로 조정할 경우 애플리케이션에서 채우지 않으면 새 클러스터가 비워집니다.

주제

- 수평으로 Memcached 조정
- 수직으로 Memcached 조정

수평으로 Memcached 조정

Memcached 엔진은 여러 노드에 대한 데이터 분할을 지원합니다. 따라서 Memcached 클러스터는 쉽게 수평으로 조정됩니다. Memcached 클러스터에는 1~60개의 노드가 있을 수 있습니다. 수평으로 Memcached 클러스터를 조정하려면 노드를 추가 또는 제거하면 됩니다.

다음 항목은 노드를 추가 또는 제거하여 Memcached 클러스터를 스케일 아웃 또는 스케일 인하는 방법을 자세히 설명합니다.

- 클러스터에 노드 추가

- 클러스터에서 노드 삭제

Memcached 클러스터의 노드 수를 변경할 때마다 키스페이스의 일부라도 다시 매핑해야 정확한 노드에 매핑됩니다. Memcached 클러스터의 로드 밸런싱에 대한 자세한 내용은 효율적인 로드 밸런싱을 위해 ElastiCache 클라이언트 구성(Memcached) 섹션을 참조하세요.

Memcached 클러스터에 대해 Auto Discovery를 사용할 경우 노드를 추가 또는 제거할 때 애플리케이션에 있는 엔드포인트를 변경할 필요가 없습니다. 자동 검색에 대한 자세한 내용은 클러스터의 노드 자동 식별(Memcached) 섹션을 참조합니다. 자동 검색을 사용하지 않을 경우 Memcached 클러스터의 노드 수를 변경할 때마다 애플리케이션에 있는 엔드포인트를 업데이트해야 합니다.

수직으로 Memcached 조정

Memcached 클러스터를 위 또는 아래로 확장하거나 축소하는 경우 새 클러스터를 생성해야 합니다. Memcached 클러스터는 애플리케이션이 채울 때까지 항상 비어 있습니다.

> **Important**
>
> 소형 노드 유형으로 스케일 다운할 경우 소형 노드 유형이 데이터 및 오버헤드에 적합해야 합니다. 자세한 내용은 캐시 노드 크기 선택을 참조하세요.

주제

- 수직으로 Memcached 조정(콘솔)
- 수직으로 Memcached 조정(AWS CLI)
- 수직으로 Memcached 조정(ElastiCache API)

수직으로 Memcached 조정(콘솔)

다음 절차에서는 ElastiCache 콘솔을 사용하여 클러스터를 수직으로 확장하는 방법을 안내합니다.

Memcached 클러스터를 수직으로 조정하려면(콘솔)

1. 새 노드 유형으로 새 클러스터를 생성합니다. 자세한 내용은 Memcached 클러스터 생성(콘솔) 섹션을 참조하세요.
2. 애플리케이션에서 엔드포인트를 새 클러스터의 엔드포인트로 업데이트합니다. 자세한 내용은 클러스터의 엔드포인트 찾기(콘솔)(Memcached) 단원을 참조하십시오.
3. 이전 클러스터를 삭제합니다. 자세한 내용은 Memcached에서 새 노드 삭제를 참조하세요.

수직으로 Memcached 조정(AWS CLI)

다음 절차는 AWS CLI를 사용하여 Memcached 캐시 클러스터를 수직으로 조정하는 방법을 안내합니다.

Memcached 캐시 클러스터를 수직으로 조정하려면(AWS CLI)

1. 새 노드 유형으로 새 캐시 클러스터를 생성합니다. 자세한 내용은 를 사용하여 클러스터 생성을 참조하세요CLI.
2. 애플리케이션에서 엔드포인트를 새 클러스터의 엔드포인트로 업데이트합니다. 자세한 내용은 엔드포인트 찾기(AWS CLI) 단원을 참조하십시오.
3. 이전 캐시 클러스터를 삭제합니다. 자세한 내용은 AWS CLI 를 사용하여 ElastiCache 클러스터 삭제 단원을 참조하십시오.

수직으로 Memcached 조정(ElastiCache API)

다음 절차에서는 를 사용하여 Memcached 캐시 클러스터를 수직으로 확장하는 방법을 안내합니다 ElastiCache API.

Memcached 캐시 클러스터를 수직으로 확장하려면(ElastiCache API)

1. 새 노드 유형으로 새 캐시 클러스터를 생성합니다. 자세한 내용은 Memcached용 클러스터 생성(ElastiCache API) 단원을 참조하세요.
2. 애플리케이션에서 엔드포인트를 새 캐시 클러스터의 엔드포인트로 업데이트합니다. 자세한 내용은 엔드포인트 찾기(ElastiCache API) 단원을 참조하십시오.
3. 이전 캐시 클러스터를 삭제합니다. 자세한 내용은 사용 ElastiCache API 단원을 참조하십시오.

Valkey 또는 Redis용 클러스터 크기 조정OSS(클러스터 모드 비활성화됨)

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터는 샤드가 0개인 단일 노드 클러스터 또는 샤드가 1개인 다중 노드 클러스터일 수 있습니다. 단일 노드 클러스터에서는 읽기와 쓰기에 모두 사용되는 노드 1개를 사용합니다. 다중 노드 클러스터에는 읽기/쓰기 기본 노드인 노드 1개와 0~5개의 읽기 전용 복제본 노드가 있습니다.

주제

Valkey 또는 Redis OSS 클러스터 크기 조정

작업	Valkey 또는 RedisOSS(클러스터 모드 비활성화됨)	Valkey 또는 RedisOSS(클러스터 모드 활성화됨)
축소	ElastiCache 클러스터에서 노드 제거	Valkey 또는 Redis에서 클러스터 크기 조정OSS(클러스터 모드 활성화됨)
확장	클러스터에 노드 추가	Valkey 또는 Redis에 대한 온라인 리샤딩OSS(클러스터 모드 활성화됨)
노드 유형 변경	대형 노드 유형으로: • 단일 노드 Valkey 또는 Redis OSS 클러스터 확장 • 복제본을 사용하여 Valkey 또는 Redis OSS 클러스터 확장 소형 노드 유형으로: • 단일 노드 Valkey 또는 Redis OSS 클러스터 축소 •	노드 유형 수정하여 온라인 수직 조정

작업	Valkey 또는 RedisOSS(클러스터 모드 비활성화됨)	Valkey 또는 RedisOSS(클러스터 모드 활성화됨)
	복제본을 사용하여 Valkey 또는 Redis OSS 클러스터 축소	
노드 그룹의 수 변경	Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터에는 지원되지 않음	Valkey 또는 Redis에서 클러스터 크기 조정OSS(클러스터 모드 활성화됨)

목차

Valkey 또는 Redis에 대한 단일 노드 클러스터 크기 조정OSS(클러스터 모드 비활성화됨)

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 노드는 모든 캐시 데이터와 Valkey 또는 Redis OSS 오버헤드를 포함할 수 있을 만큼 충분히 커야 합니다. Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터의 데이터 용량을 변경하려면 세로로 확장하거나, 더 큰 노드 유형으로 확장하여 데이터 용량을 늘리거나, 더 작은 노드 유형으로 축소하여 데이터 용량을 줄여야 합니다.

ElastiCache 스케일 업 프로세스는 기존 데이터를 보존하기 위해 최선의 노력을 기울일 수 있도록 설계되었으며 Valkey 또는 Redis OSS 복제를 성공적으로 수행해야 합니다. Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터의 경우 Valkey 또는 Redis 에 충분한 메모리를 제공하는 것이 좋습니다 OSS.

여러 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터에서 데이터를 분할할 수 없습니다. 그러나 클러스터의 읽기 용량만 늘리거나 줄여야 하는 경우 복제본 노드가 있는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터를 생성하고 읽기 전용 복제본을 추가하거나 제거할 수 있습니다. 단일 노드 Valkey 또는 Redis OSS 캐시 클러스터를 기본 클러스터로 사용하여 복제본 노드가 있는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터를 생성하려면 섹션을 참조하세요Valkey(클러스터 모드 비활성화됨) 클러스터 생성(콘솔).

복제본이 있는 클러스터를 생성하면 읽기 전용 복제본을 추가하여 읽기 용량을 늘릴 수 있습니다. 나중에 필요한 경우 읽기 전용 복제본을 제거하여 읽기 용량을 줄일 수 있습니다. 자세한 내용은 읽기 용량 늘리기 또는 읽기 용량 줄이기을 참조하세요.

읽기 용량을 확장할 수 있을 뿐만 아니라 복제본이 있는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터는 다른 비즈니스 이점을 제공합니다. 자세한 내용은 고가용성을 위한 복제 그룹 사용 단원을 참조하십시오.

> **Important**
>
> 파라미터 그룹이 `reserved-memory`를 사용하여 Valkey 또는 Redis OSS 오버헤드용 메모리를 따로 설정하는 경우 크기 조정을 시작하기 전에 새 노드 유형에 맞는 메모리 양을 예약하는 사용자 지정 파라미터 그룹이 있는지 확인합니다. 또는 `reserved-memory-percent`를 사용하고 새 클러스터에 대해 해당 파라미터 그룹을 사용하도록 사용자 지정 파라미터 그룹을 수정할 수 있습니다.
>
> `reserved-memory-percent`를 사용할 경우에는 이렇게 하지 않아도 됩니다.
>
> 자세한 내용은 Valkey 및 Redis용 예약 메모리 관리 OSS 단원을 참조하십시오.

주제

- 단일 노드 Valkey 또는 Redis OSS 클러스터 확장
- 단일 노드 Valkey 또는 Redis OSS 클러스터 축소

단일 노드 Valkey 또는 Redis OSS 클러스터 확장

단일 노드 Valkey 또는 Redis OSS 클러스터를 스케일 업하면 ElastiCache 콘솔, AWS CLI또는 를 사용하는지 여부에 관계없이 가 다음 프로세스를 ElastiCache 수행합니다 ElastiCache API.

1. 새 노드 유형이 있는 새 캐시 클러스터가 동일한 가용 영역에서 기존 캐시 클러스터로 실행됩니다.
2. 기존 캐시 클러스터의 캐시 데이터가 새 캐시 클러스터로 복사됩니다. 이 프로세스의 기간은 노드 유형 및 캐시 클러스터에 있는 데이터의 양에 따라 달라집니다.
3. 새 캐시 클러스터를 사용하여 읽기 및 쓰기를 수행합니다. 새 캐시 클러스터의 엔드포인트가 이전 캐시 클러스터의 엔드포인트와 동일하므로 애플리케이션에 있는 엔드포인트를 업데이트할 필요가 없습니다. DNS 항목이 업데이트되는 동안 기본 노드에서 읽기 및 쓰기가 잠시 중단(몇 초)됩니다.
4. ElastiCache 는 이전 캐시 클러스터를 삭제합니다. 이전 노드에 대한 연결이 끊어지기 때문에 이전 노드의 읽기 및 쓰기가 잠깐(몇 초) 중단될 수 있습니다.

> **Note**
>
> r6gd 노드 유형을 실행하는 클러스터의 경우 r6gd 노드 패밀리 내의 노드 크기로만 조정할 수 있습니다.

다음 표와 같이 다음 유지 관리 기간에 예정된 엔진 업그레이드가 있는 경우 Valkey 또는 Redis OSS 스케일 업 작업이 차단됩니다. 유지 관리 기간에 대한 자세한 내용은 ElastiCache 클러스터 유지 관리 섹션을 참조하세요.

차단된 Valkey 또는 Redis OSS 작업

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

사용자를 차단하는 대기 중 작업이 있는 경우 다음 중 하나를 수행할 수 있습니다.

- 즉시 적용 확인란을 선택 취소하여 다음 유지 관리 기간에 대한 Valkey 또는 Redis OSS 스케일 업 작업을 예약합니다(CLI 사용: `--no-apply-immediately`, API 사용: `ApplyImmediately=false`).
- 다음 유지 관리 기간(또는 이후)까지 기다렸다가 Valkey 또는 Redis OSS 스케일 업 작업을 수행합니다.
- 즉시 적용 확인란을 선택한 상태에서 이 캐시 클러스터 수정에 Valkey 또는 Redis OSS 엔진 업그레이드를 추가합니다(CLI 사용: `--apply-immediately`, API 사용: `ApplyImmediately=true`). 이렇게 하면 스케일 업 작업의 차단이 해제되어 엔진 업그레이드가 즉시 수행됩니다.

ElastiCache 콘솔, 또는 를 사용하여 단일 노드 Valkey AWS CLI또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터를 확장할 수 있습니다 ElastiCache API.

> **Important**
>
> 파라미터 그룹이 `reserved-memory`를 사용하여 Valkey 또는 Redis OSS 오버헤드용 메모리를 따로 설정하는 경우 크기 조정을 시작하기 전에 새 노드 유형에 맞는 메모리 양을 예약하는 사용자 지정 파라미터 그룹이 있는지 확인합니다. 또는 `reserved-memory-percent`를 사용하고 새 클러스터에 대해 해당 파라미터 그룹을 사용하도록 사용자 지정 파라미터 그룹을 수정할 수 있습니다.
>
> `reserved-memory-percent`를 사용할 경우에는 이렇게 하지 않아도 됩니다.
>
> 자세한 내용은 Valkey 및 Redis용 예약 메모리 관리 OSS 단원을 참조하십시오.

Valkey 또는 Redis에 대한 단일 노드 클러스터 확장OSS(클러스터 모드 비활성화됨)(콘솔)

다음 절차에서는 ElastiCache 관리 콘솔을 사용하여 단일 노드 Valkey 또는 Redis OSS 클러스터를 확장하는 방법을 설명합니다. 이 프로세스 중에 Valkey 또는 Redis OSS 클러스터는 가동 중지 시간을 최소화하면서 요청을 계속 처리합니다.

단일 노드 Valkey 또는 Redis OSS 클러스터 확장(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 Valkey 또는 Redis OSS 클러스터를 선택합니다.
3. 클러스터 목록에서 확장하려는 클러스터를 선택합니다(클러스터링된 Valkey 또는 Redis OSS 엔진이 아닌 Valkey 또는 Redis OSS 엔진을 실행 중이어야 함).
4. 수정을 선택합니다.

5. [Modify Cluster] 마법사에서 다음을 수행합니다.

   a. [Node type] 목록에서 조정할 노드 유형을 선택합니다.

   b. `reserved-memory`를 사용하여 메모리를 관리할 경우 [Parameter Group] 목록에서 새 노드 유형에 대해 올바른 메모리 양을 예약하는 사용자 지정 파라미터 그룹을 선택합니다.

6. 스케일 업 프로세스를 즉시 수행하려면 [Apply immediately] 상자를 선택합니다. [Apply immediately] 상자를 선택하지 않으면 이 클러스터의 다음 유지 관리 기간 중 스케일 업 프로세스가 수행됩니다.

7. 수정을 선택합니다.

   이전 단계에서 [Apply immediately]를 선택한 경우 클러스터의 상태가 수정 중으로 변경됩니다. 상태가 사용 가능으로 변경되면 수정이 완료되고 새 클러스터의 사용을 시작할 수 있습니다.

단일 노드 Valkey 또는 Redis OSS 캐시 클러스터 확장(AWS CLI)

다음 절차에서는 를 사용하여 단일 노드 Valkey 또는 Redis OSS 캐시 클러스터를 확장하는 방법을 설명합니다 AWS CLI. 이 프로세스 중에 Valkey 또는 Redis OSS 클러스터는 가동 중지 시간을 최소화하면서 요청을 계속 처리합니다.

단일 노드 Valkey 또는 Redis OSS 캐시 클러스터 확장(AWS CLI)

1. 다음 파라미터로 `list-allowed-node-type-modifications` 명령을 실행 AWS CLI 하여 확장할 수 있는 노드 유형을 결정합니다.

   - `--cache-cluster-id`

   Linux, macOS, Unix의 경우:

   ```
   aws elasticache list-allowed-node-type-modifications \
        --cache-cluster-id my-cache-cluster-id
   ```

   Windows의 경우:

   ```
   aws elasticache list-allowed-node-type-modifications ^
        --cache-cluster-id my-cache-cluster-id
   ```

   위 명령의 출력은 다음과 같습니다(JSON 형식).

```
{
     "ScaleUpModifications": [
         "cache.m3.2xlarge",
         "cache.m3.large",
         "cache.m3.xlarge",
         "cache.m4.10xlarge",
         "cache.m4.2xlarge",
         "cache.m4.4xlarge",
         "cache.m4.large",
         "cache.m4.xlarge",
         "cache.r3.2xlarge",
         "cache.r3.4xlarge",
         "cache.r3.8xlarge",
         "cache.r3.large",
         "cache.r3.xlarge"
     ]
        "ScaleDownModifications": [
         "cache.t2.micro",
         "cache.t2.small ",
         "cache.t2.medium ",
            "cache.t1.small ",
     ],

 }
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.list-allowed-node-type-수정 AWS CLI 참조 의
.

2. 명령과 다음 파라미터를 사용하여 AWS CLI `modify-cache-cluster` 캐시 클러스터를 확장하고 더 큰 새 노드 유형을 지정하여 기존 캐시 클러스터를 수정합니다.

   - `--cache-cluster-id` - 확장할 캐시 클러스터의 이름입니다.
   - `--cache-node-type` - 캐시 클러스터를 조정할 새 노드 유형입니다. 이 값은 1단계의 `list-allowed-node-type-modifications` 명령에 의해 반환되는 노드 유형 중 하나여야 합니다.
   - `--cache-parameter-group-name` - [선택 사항] `reserved-memory`를 사용하여 클러스터의 예약된 메모리를 관리할 경우 이 파라미터를 사용합니다. 새 노드 유형에 대해 올바른 메모리 양을 예약하는 사용자 지정 캐시 파라미터 그룹을 지정합니다. `reserved-memory-percent`를 사용할 경우 이 파라미터를 생략할 수 있습니다.

- `--apply-immediately` - 스케일 업 프로세스가 즉시 적용되도록 합니다. 스케일 업 프로세스를 클러스터의 다음 유지 관리 기간으로 연기하려면 `--no-apply-immediately` 파라미터를 사용하세요.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-cache-cluster \
     --cache-cluster-id my-redis-cache-cluster \
     --cache-node-type cache.m3.xlarge \
     --cache-parameter-group-name redis32-m2-xl \
     --apply-immediately
```

Windows의 경우:

```
aws elasticache modify-cache-cluster ^
     --cache-cluster-id my-redis-cache-cluster ^
     --cache-node-type cache.m3.xlarge ^
     --cache-parameter-group-name redis32-m2-xl ^
     --apply-immediately
```

위 명령의 출력은 다음과 같습니다(JSON 형식).

```
{
     "CacheCluster": {
         "Engine": "redis",
         "CacheParameterGroup": {
             "CacheNodeIdsToReboot": [],
             "CacheParameterGroupName": "default.redis6.x",
             "ParameterApplyStatus": "in-sync"
         },
         "SnapshotRetentionLimit": 1,
         "CacheClusterId": "my-redis-cache-cluster",
         "CacheSecurityGroups": [],
         "NumCacheNodes": 1,
         "SnapshotWindow": "00:00-01:00",
         "CacheClusterCreateTime": "2017-02-21T22:34:09.645Z",
         "AutoMinorVersionUpgrade": true,
         "CacheClusterStatus": "modifying",
         "PreferredAvailabilityZone": "us-west-2a",
```

```
        "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
        "CacheSubnetGroupName": "default",
        "EngineVersion": "6.0",
        "PendingModifiedValues": {
            "CacheNodeType": "cache.m3.2xlarge"
        },
        "PreferredMaintenanceWindow": "tue:11:30-tue:12:30",
        "CacheNodeType": "cache.m3.medium",
         "DataTiering": "disabled"
    }
 }
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.modify-cache-cluster AWS CLI 참조 의 .

3. 를 사용한 경우 다음 파라미터와 함께 명령을 사용하여 AWS CLI `describe-cache-clusters` 새 캐시 클러스터의 상태를 `--apply-immediately`확인합니다. 상태가 사용 가능으로 변경되면 새로운 대형 캐시 클러스터의 사용을 시작할 수 있습니다.

   - `--cache-cache cluster-id` - 단일 노드 Valkey 또는 Redis OSS 캐시 클러스터의 이름입니다. 모든 캐시 클러스터 대신 특정 캐시 클러스터를 설명하려면 이 파라미터를 사용하세요.

```
aws elasticache describe-cache-clusters --cache-cluster-id my-redis-cache-cluster
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.describe-cache-clusters AWS CLI 참조 의 .

단일 노드 Valkey 또는 Redis OSS 캐시 클러스터 확장(ElastiCache API)

다음 절차에서는 를 사용하여 단일 노드 Valkey 또는 Redis OSS 캐시 클러스터를 확장하는 방법을 설명합니다 ElastiCache API. 이 프로세스 중에 Valkey 또는 Redis OSS 클러스터는 가동 중지 시간을 최소화하면서 요청을 계속 처리합니다.

단일 노드 Valkey 또는 Redis OSS 캐시 클러스터 확장(ElastiCache API)

1. 다음 파라미터로 `ListAllowedNodeTypeModifications` 작업을 실행 ElastiCache API하여 확장할 수 있는 노드 유형을 결정합니다.

   - `CacheClusterId` - 확장하려는 단일 노드 Valkey 또는 Redis OSS 캐시 클러스터의 이름입니다.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ListAllowedNodeTypeModifications
    &CacheClusterId=MyRedisCacheCluster
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.ListAllowedNodeTypeModifications Amazon ElastiCache API 참조 의 .

2. `ModifyCacheCluster` ElastiCache API 작업과 다음 파라미터를 사용하여 캐시 클러스터를 확장하고 더 큰 새 노드 유형을 지정하여 기존 캐시 클러스터를 수정합니다.

   - `CacheClusterId` - 확장할 캐시 클러스터의 이름입니다.
   - `CacheNodeType` - 캐시 클러스터를 조정할 새로운 대형 노드 유형입니다. 이 값은 이전 단계의 `ListAllowedNodeTypeModifications` 작업에서 반환된 노드 유형 중 하나여야 합니다.
   - `CacheParameterGroupName` - [선택 사항] `reserved-memory`를 사용하여 클러스터의 예약된 메모리를 관리할 경우 이 파라미터를 사용합니다. 새 노드 유형에 대해 올바른 메모리 양을 예약하는 사용자 지정 캐시 파라미터 그룹을 지정합니다. `reserved-memory-percent`를 사용할 경우 이 파라미터를 생략할 수 있습니다.
   - `ApplyImmediately` - 스케일 업 프로세스가 즉시 수행되도록 하려면 `true`로 설정합니다. 스케일 업 프로세스를 클러스터의 다음 유지 관리 기간으로 연기하려면 `ApplyImmediately=false`를 사용하세요.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyCacheCluster
    &ApplyImmediately=true
    &CacheClusterId=MyRedisCacheCluster
    &CacheNodeType=cache.m3.xlarge
    &CacheParameterGroupName redis32-m2-xl
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.ModifyCacheCluster Amazon ElastiCache API 참조 의 .

3. 를 사용한 경우 다음 파라미터로 `DescribeCacheClusters` 작업을 사용하여 새 캐시 클러스터의 ElastiCache API 상태를 `ApplyImmediately=true`확인합니다. 상태가 사용 가능으로 변경되면 새로운 대형 캐시 클러스터의 사용을 시작할 수 있습니다.

   - `CacheClusterId` - 단일 노드 Valkey 또는 Redis OSS 캐시 클러스터의 이름입니다. 모든 캐시 클러스터 대신 특정 캐시 클러스터를 설명하려면 이 파라미터를 사용하세요.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeCacheClusters
    &CacheClusterId=MyRedisCacheCluster
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

   자세한 내용을 알아보려면 다음 섹션을 참조하세요.DescribeCacheClusters Amazon ElastiCache API 참조 의 .

단일 노드 Valkey 또는 Redis OSS 클러스터 축소

다음 섹션에서는 단일 노드 Valkey 또는 Redis OSS 클러스터를 더 작은 노드 유형으로 축소하는 방법을 안내합니다. 새로운 소형 노드 유형이 모든 데이터와 Valkey 또는 Redis OSS 오버헤드를 수용할 수 있을 만큼 충분히 큰지 확인하는 것은 새로운 Valkey 또는 Redis OSS 클러스터의 장기적인 성공에 중요합니다. 자세한 내용은 Valkey 또는 Redis OSS 스냅샷을 생성하기에 충분한 메모리 확보 단원을 참조하십시오.

> **Note**
>
> r6gd 노드 유형을 실행하는 클러스터의 경우 r6gd 노드 패밀리 내의 노드 크기로만 조정할 수 있습니다.

주제

- 단일 노드 Valkey 또는 Redis OSS 클러스터 축소(콘솔)
- 단일 노드 Valkey 또는 Redis OSS 캐시 클러스터 축소(AWS CLI)
- 단일 노드 Valkey 또는 Redis OSS 캐시 클러스터 축소(ElastiCache API)

단일 노드 Valkey 또는 Redis OSS 클러스터 축소(콘솔)

다음 절차에서는 ElastiCache 콘솔을 사용하여 단일 노드 Valkey 또는 Redis OSS 클러스터를 더 작은 노드 유형으로 축소하는 방법을 안내합니다.

> **Important**
>
> 파라미터 그룹이 `reserved-memory`를 사용하여 Valkey 또는 Redis OSS 오버헤드용 메모리를 따로 설정하는 경우 크기 조정을 시작하기 전에 새 노드 유형에 맞는 메모리 양을 예약하는 사용자 지정 파라미터 그룹이 있는지 확인합니다. 또는 `reserved-memory-percent`를 사용하고 새 클러스터에 대해 해당 파라미터 그룹을 사용하도록 사용자 지정 파라미터 그룹을 수정할 수 있습니다.
>
> `reserved-memory-percent`를 사용할 경우에는 이렇게 하지 않아도 됩니다.
>
> 자세한 내용은 Valkey 및 Redis용 예약 메모리 관리 OSS 단원을 참조하십시오.

단일 노드 Valkey 또는 Redis OSS 클러스터 축소(콘솔)

1. 소형 노드 유형이 데이터 및 오버헤드 요구 사항에 적합한지 확인합니다.

2. 파라미터 그룹이 `reserved-memory`를 사용하여 Valkey 또는 Redis OSS 오버헤드용 메모리를 따로 설정하는 경우 새 노드 유형에 맞는 메모리 양을 따로 설정할 사용자 지정 파라미터 그룹이 있는지 확인합니다.

   또는 `reserved-memory-percent`를 사용하여 사용자 지정 파라미터 그룹을 수정할 수 있습니다. 자세한 내용은 Valkey 및 Redis용 예약 메모리 관리 OSS 단원을 참조하십시오.
3. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
4. 클러스터 목록에서 스케일 다운할 클러스터를 선택합니다. 이 클러스터는 클러스터링된 Valkey 또는 Redis OSS 엔진이 아닌 Valkey 또는 Redis OSS 엔진을 실행 중이어야 합니다.
5. 수정을 선택합니다.
6. [Modify Cluster] 마법사에서 다음을 수행합니다.

   a. [Node type] 목록에서 스케일 다운할 노드 유형을 선택합니다.

   b. `reserved-memory`를 사용하여 메모리를 관리할 경우 [Parameter Group] 목록에서 새 노드 유형에 대해 올바른 메모리 양을 예약하는 사용자 지정 파라미터 그룹을 선택합니다.
7. 스케일 다운 프로세스를 즉시 수행하려면 [Apply immediately] 확인란을 선택합니다. [Apply immediately] 확인란을 선택하지 않고 비워 두면 이 클러스터의 다음 유지 관리 기간 중 스케일 다운 프로세스가 수행됩니다.
8. 수정을 선택합니다.
9. 클러스터의 상태가 수정 중에서 사용 가능으로 변경되면 클러스터가 새 노드 유형으로 조정된 것입니다. 애플리케이션에서 엔드포인트를 업데이트할 필요가 없습니다.

단일 노드 Valkey 또는 Redis OSS 캐시 클러스터 축소(AWS CLI)

다음 절차에서는 를 사용하여 단일 노드 Valkey 또는 Redis OSS 캐시 클러스터를 축소하는 방법을 설명합니다 AWS CLI.

단일 노드 Valkey 또는 Redis OSS 캐시 클러스터를 축소하려면(AWS CLI)

1. 다음 파라미터로 `list-allowed-node-type-modifications` 명령을 실행 AWS CLI 하여 확장할 수 있는 노드 유형을 결정합니다.

   - `--cache-cluster-id`

   Linux, macOS, Unix의 경우:

```
aws elasticache list-allowed-node-type-modifications \
     --cache-cluster-id my-cache-cluster-id
```

Windows의 경우:

```
aws elasticache list-allowed-node-type-modifications ^
     --cache-cluster-id my-cache-cluster-id
```

위 명령의 출력은 다음과 같습니다(JSON 형식).

```
{
     "ScaleUpModifications": [
         "cache.m3.2xlarge",
         "cache.m3.large",
         "cache.m3.xlarge",
         "cache.m4.10xlarge",
         "cache.m4.2xlarge",
         "cache.m4.4xlarge",
         "cache.m4.large",
         "cache.m4.xlarge",
         "cache.r3.2xlarge",
         "cache.r3.4xlarge",
         "cache.r3.8xlarge",
         "cache.r3.large",
         "cache.r3.xlarge"
     ]
        "ScaleDownModifications": [
         "cache.t2.micro",
         "cache.t2.small ",
         "cache.t2.medium ",
            "cache.t1.small ",
     ],

 }
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.list-allowed-node-type-수정 AWS CLI 참조 의
.

2. 명령과 다음 파라미터를 사용하여 AWS CLI `modify-cache-cluster` 캐시 클러스터를 축소하고 새롭고 작은 노드 유형을 지정하여 기존 캐시 클러스터를 수정합니다.

- `--cache-cluster-id` - 축소할 캐시 클러스터의 이름입니다.
- `--cache-node-type` - 캐시 클러스터를 조정할 새 노드 유형입니다. 이 값은 1단계의 `list-allowed-node-type-modifications` 명령에 의해 반환되는 노드 유형 중 하나여야 합니다.
- `--cache-parameter-group-name` - [선택 사항] `reserved-memory`를 사용하여 클러스터의 예약된 메모리를 관리할 경우 이 파라미터를 사용합니다. 새 노드 유형에 대해 올바른 메모리 양을 예약하는 사용자 지정 캐시 파라미터 그룹을 지정합니다. `reserved-memory-percent`를 사용할 경우 이 파라미터를 생략할 수 있습니다.
- `--apply-immediately` - 축소 프로세스가 즉시 적용되도록 합니다. 스케일 업 프로세스를 클러스터의 다음 유지 관리 기간으로 연기하려면 `--no-apply-immediately` 파라미터를 사용하세요.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-cache-cluster \
     --cache-cluster-id my-redis-cache-cluster \
     --cache-node-type cache.m3.xlarge \
     --cache-parameter-group-name redis32-m2-xl \
     --apply-immediately
```

Windows의 경우:

```
aws elasticache modify-cache-cluster ^
     --cache-cluster-id my-redis-cache-cluster ^
     --cache-node-type cache.m3.xlarge ^
     --cache-parameter-group-name redis32-m2-xl ^
     --apply-immediately
```

위 명령의 출력은 다음과 같습니다(JSON 형식).

```
{
     "CacheCluster": {
         "Engine": "redis",
         "CacheParameterGroup": {
             "CacheNodeIdsToReboot": [],
             "CacheParameterGroupName": "default.redis6,x",
             "ParameterApplyStatus": "in-sync"
         },
```

```
          "SnapshotRetentionLimit": 1,
          "CacheClusterId": "my-redis-cache-cluster",
          "CacheSecurityGroups": [],
          "NumCacheNodes": 1,
          "SnapshotWindow": "00:00-01:00",
          "CacheClusterCreateTime": "2017-02-21T22:34:09.645Z",
          "AutoMinorVersionUpgrade": true,
          "CacheClusterStatus": "modifying",
          "PreferredAvailabilityZone": "us-west-2a",
          "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
          "CacheSubnetGroupName": "default",
          "EngineVersion": "6.0",
          "PendingModifiedValues": {
              "CacheNodeType": "cache.m3.2xlarge"
          },
          "PreferredMaintenanceWindow": "tue:11:30-tue:12:30",
          "CacheNodeType": "cache.m3.medium",
           "DataTiering": "disabled"
      }
  }
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.modify-cache-cluster AWS CLI 참조 의 .

3. 를 사용한 경우 다음 파라미터와 함께 명령을 사용하여 AWS CLI `describe-cache-clusters` 새 캐시 클러스터의 상태를 `--apply-immediately`확인합니다. 상태가 사용 가능으로 변경되면 새로운 대형 캐시 클러스터의 사용을 시작할 수 있습니다.

   - `--cache-cache cluster-id` - 단일 노드 Valkey 또는 Redis OSS 캐시 클러스터의 이름입니다. 모든 캐시 클러스터 대신 특정 캐시 클러스터를 설명하려면 이 파라미터를 사용하세요.

```
aws elasticache describe-cache-clusters --cache-cluster-id my-redis-cache-cluster
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.describe-cache-clusters AWS CLI 참조 의 .

단일 노드 Valkey 또는 Redis OSS 캐시 클러스터 축소(ElastiCache API)

다음 절차에서는 를 사용하여 단일 노드 Valkey 또는 Redis OSS 캐시 클러스터를 확장하는 방법을 설명합니다 ElastiCache API.

단일 노드 Valkey 또는 Redis OSS 캐시 클러스터를 축소하려면(ElastiCache API)

1. 다음 파라미터로 `ListAllowedNodeTypeModifications` 작업을 실행 ElastiCache API하여 확장할 수 있는 노드 유형을 결정합니다.

   - `CacheClusterId` - 축소하려는 단일 노드 Valkey 또는 Redis OSS 캐시 클러스터의 이름입니다.

   ```
   https://elasticache.us-west-2.amazonaws.com/
       ?Action=ListAllowedNodeTypeModifications
       &CacheClusterId=MyRedisCacheCluster
       &Version=2015-02-02
       &SignatureVersion=4
       &SignatureMethod=HmacSHA256
       &Timestamp=20150202T192317Z
       &X-Amz-Credential=<credential>
   ```

   자세한 내용을 알아보려면 다음 섹션을 참조하세요.ListAllowedNodeTypeModifications Amazon ElastiCache API 참조 의 .

2. `ModifyCacheCluster` ElastiCache API 작업과 다음 파라미터를 사용하여 캐시 클러스터를 확장하고 더 큰 새 노드 유형을 지정하여 기존 캐시 클러스터를 수정합니다.

   - `CacheClusterId` - 축소할 캐시 클러스터의 이름입니다.
   - `CacheNodeType` - 캐시 클러스터를 축소할 새롭고 더 작은 노드 유형입니다. 이 값은 이전 단계의 `ListAllowedNodeTypeModifications` 작업에서 반환된 노드 유형 중 하나여야 합니다.
   - `CacheParameterGroupName` - [선택 사항] `reserved-memory`를 사용하여 클러스터의 예약된 메모리를 관리할 경우 이 파라미터를 사용합니다. 새 노드 유형에 대해 올바른 메모리 양을 예약하는 사용자 지정 캐시 파라미터 그룹을 지정합니다. `reserved-memory-percent`를 사용할 경우 이 파라미터를 생략할 수 있습니다.
   - `ApplyImmediately` - 축소 프로세스가 즉시 수행되도록 하려면 `true`로 설정합니다. 스케일 업 프로세스를 클러스터의 다음 유지 관리 기간으로 연기하려면 `ApplyImmediately=false`를 사용하세요.

   ```
   https://elasticache.us-west-2.amazonaws.com/
       ?Action=ModifyCacheCluster
       &ApplyImmediately=true
   ```

```
    &CacheClusterId=MyRedisCacheCluster
    &CacheNodeType=cache.m3.xlarge
    &CacheParameterGroupName redis32-m2-xl
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.ModifyCacheCluster Amazon ElastiCache API 참조 의 .

3. 를 사용한 경우 다음 파라미터로 `DescribeCacheClusters` 작업을 사용하여 새 캐시 클러스터의 ElastiCache API 상태를 `ApplyImmediately=true`확인합니다. 상태가 사용 가능으로 변경되면 새롭고 더 작은 캐시 클러스터를 사용할 수 있습니다.

   - `CacheClusterId` - 단일 노드 Valkey 또는 Redis OSS 캐시 클러스터의 이름입니다. 모든 캐시 클러스터 대신 특정 캐시 클러스터를 설명하려면 이 파라미터를 사용하세요.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeCacheClusters
    &CacheClusterId=MyRedisCacheCluster
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.DescribeCacheClusters Amazon ElastiCache API 참조 의 .

Valkey 또는 Redis용 복제본 노드 크기 조정OSS(클러스터 모드 비활성화됨)

복제본 노드가 있는 Valkey 또는 Redis OSS 클러스터(API/에서 복제 그룹이라고 함CLI)는 자동 장애 조치가 활성화된 다중 AZ가 있는 복제를 통해 고가용성을 제공합니다. 복제본 노드가 있는 클러스터는 최대 6개의 Valkey 또는 Redis OSS 노드로 구성된 논리적 컬렉션으로, 기본 노드인 한 노드는 읽기 및 쓰기 요청을 모두 처리할 수 있습니다. 클러스터의 다른 모든 노드는 기본 노드의 읽기 전용 복제본입니다. 기본에 작성된 데이터는 클러스터의 모든 읽기 전용 복제본으로 비동기식으로 복제됩니다. Valkey 또는 RedisOSS(클러스터 모드 비활성화됨)는 여러 클러스터에서 데이터 파티셔닝을 지원하지 않으므로 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹의 각 노드에는 전체 캐시 데이터 세트가 포함됩니다. Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터는 최대 500개의 샤드에서 데이터 파티셔닝을 지원합니다.

클러스터의 데이터 용량을 변경하려면 대형 노드 유형으로 스케일 업하거나 소형 노드 유형으로 스케일 다운해야 합니다.

클러스터의 읽기 용량을 변경하려면 최대 5개의 추가 읽기 전용 복제본을 추가하거나 읽기 전용 복제본을 제거하세요.

ElastiCache 스케일 업 프로세스는 기존 데이터를 보존하기 위해 최선의 노력을 기울일 수 있도록 설계되었으며 Valkey 또는 Redis OSS 복제를 성공적으로 수행해야 합니다. 복제본이 있는 Valkey 또는 Redis OSS 클러스터의 경우 Valkey 또는 Redis 에 충분한 메모리를 제공하는 것이 좋습니다OSS.

주제

- 복제본을 사용하여 Valkey 또는 Redis OSS 클러스터 확장
- 복제본을 사용하여 Valkey 또는 Redis OSS 클러스터 축소
- 읽기 용량 늘리기
- 읽기 용량 줄이기

관련 항목

- 고가용성을 위한 복제 그룹 사용
- 복제: Valkey 및 Redis OSS 클러스터 모드 비활성화됨 vs. 활성화됨
- Valkey 및 Redis와 함께 다중 AZ ElastiCache 를 사용하여 의 가동 중지 시간 최소화 OSS
- Valkey 또는 Redis OSS 스냅샷을 생성하기에 충분한 메모리 확보

주제

- 복제본을 사용하여 Valkey 또는 Redis OSS 클러스터 확장
- 복제본을 사용하여 Valkey 또는 Redis OSS 클러스터 축소
- 읽기 용량 늘리기
- 읽기 용량 줄이기

복제본을 사용하여 Valkey 또는 Redis OSS 클러스터 확장

Amazon ElastiCache 은 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹 확장을 위한 콘솔CLI, 및 API 지원을 제공합니다.

스케일 업 프로세스가 시작되면 는 다음을 ElastiCache 수행합니다.

1. 새 노드 유형을 사용하여 복제 그룹을 시작합니다.
2. 현재 기본 노드에서 새 기본 노드로 모든 데이터를 복사합니다.
3. 새 읽기 전용 복제본을 새 기본 노드와 동기화합니다.
4. 새 노드를 가리키도록 DNS 항목을 업데이트합니다. 따라서 애플리케이션의 엔드포인트를 업데이트할 필요가 없습니다. Valkey 7.2 이상 또는 Redis OSS 5.0.5 이상의 경우 클러스터가 계속 온라인 상태를 유지하고 수신 요청을 처리하는 동안 자동 장애 조치 활성화 클러스터를 확장할 수 있습니다. Redis OSS 버전 4.0.10 이하에서는 DNS 항목이 업데이트되는 동안 기본 노드에서 이전 버전의 읽기 및 쓰기가 잠시 중단될 수 있습니다.
5. 이전 노드(CLI/API: 복제 그룹)를 삭제합니다. 이전 노드에 대한 연결이 끊어지기 때문에 이전 노드의 읽기 및 쓰기가 잠깐(몇 초) 중단될 수 있습니다.

이 프로세스의 기간은 노드 유형 및 클러스터에 있는 데이터의 양에 따라 달라집니다.

다음 표와 같이 클러스터의 다음 유지 관리 기간에 엔진 업그레이드가 예약된 경우 Valkey 또는 Redis OSS 스케일 업 작업이 차단됩니다.

차단된 Valkey 또는 Redis OSS 작업

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

사용자를 차단하는 대기 중 작업이 있는 경우 다음 중 하나를 수행할 수 있습니다.

- 즉시 적용 확인란을 선택 취소하여 다음 유지 관리 기간에 대한 Valkey 또는 Redis OSS 스케일 업 작업을 예약합니다(CLI 사용: `--no-apply-immediately`, API 사용: `ApplyImmediately=false`).
- 다음 유지 관리 기간(또는 이후)까지 기다렸다가 Valkey 또는 Redis OSS 스케일 업 작업을 수행합니다.
- 즉시 적용 확인란을 선택한 상태에서 이 캐시 클러스터 수정에 Valkey 또는 Redis OSS 엔진 업그레이드를 추가합니다(CLI 사용: `--apply-immediately`, API 사용: `ApplyImmediately=true`). 이렇게 하면 스케일 업 작업의 차단이 해제되어 엔진 업그레이드가 즉시 수행됩니다.

다음 섹션에서는 ElastiCache 콘솔, AWS CLI및 를 사용하여 복제본을 사용하여 Valkey 또는 Redis OSS 클러스터를 확장하는 방법을 설명합니다 ElastiCache API.

> **Important**
>
> 파라미터 그룹이 `reserved-memory`를 사용하여 Valkey 또는 Redis OSS 오버헤드용 메모리를 따로 설정하는 경우 크기 조정을 시작하기 전에 새 노드 유형에 맞는 메모리 양을 예약하는 사용자 지정 파라미터 그룹이 있는지 확인합니다. 또는 `reserved-memory-percent`를 사용하고 새 클러스터에 대해 해당 파라미터 그룹을 사용하도록 사용자 지정 파라미터 그룹을 수정할 수 있습니다.
>
> `reserved-memory-percent`를 사용할 경우에는 이렇게 하지 않아도 됩니다.
> 자세한 내용은 Valkey 및 Redis용 예약 메모리 관리 OSS 단원을 참조하십시오.

복제본을 사용하여 Valkey 또는 Redis OSS 클러스터 확장(콘솔)

대형 노드 유형으로 스케일 업하는 데 걸리는 시간은 노드 유형 및 현재 클러스터에 있는 데이터의 양에 따라 달라집니다.

다음 프로세스는 ElastiCache 콘솔을 사용하여 현재 노드 유형에서 더 큰 새 노드 유형으로 복제본을 사용하여 클러스터를 확장합니다. 이 프로세스 중에 DNS 항목이 업데이트되는 동안 기본 노드의 다른 버전에 대한 읽기 및 쓰기가 잠시 중단될 수 있습니다. 5.0.6 버전 이상에서 실행되는 노드의 경우 가동 중지 시간이 1초 미만이고 이전 버전의 경우 몇 초 미만일 수 있습니다.

복제본을 사용하여 Valkey 또는 Redis OSS 클러스터 확장(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 Valkey 클러스터 또는 Redis OSS 클러스터를 선택합니다.

3. 클러스터 목록에서 스케일 업할 클러스터를 선택합니다. 이 클러스터는 클러스터링된 Valkey 또는 Redis OSS 엔진이 아닌 Valkey 또는 Redis OSS 엔진을 실행 중이어야 합니다.
4. 수정을 선택합니다.
5. [Modify Cluster] 마법사에서 다음을 수행합니다.

    a. [Node type] 목록에서 조정할 노드 유형을 선택합니다. 모든 노드 유형을 축소할 수 있는 것은 아닙니다.

    b. `reserved-memory`를 사용하여 메모리를 관리할 경우 [Parameter Group] 목록에서 새 노드 유형에 대해 올바른 메모리 양을 예약하는 사용자 지정 파라미터 그룹을 선택합니다.
6. 스케일 업 프로세스를 즉시 수행하려면 [Apply immediately] 확인란을 선택합니다. [Apply immediately] 확인란을 선택하지 않고 비워 두면 이 클러스터의 다음 유지 관리 기간 중 스케일 업 프로세스가 수행됩니다.
7. 수정을 선택합니다.
8. 클러스터의 상태가 수정 중에서 사용 가능으로 변경되면 클러스터가 새 노드 유형으로 조정된 것입니다. 애플리케이션에서 엔드포인트를 업데이트할 필요가 없습니다.

Valkey 또는 Redis OSS 복제 그룹 확장(AWS CLI)

다음 절차는 AWS CLI를 사용하여 복제 그룹을 현재 노드 유형에서 새롭고 더 큰 노드 유형으로 조정합니다. 이 프로세스 중에 는 새 노드를 가리키도록 DNS 항목을 ElastiCache 업데이트합니다. 따라서 애플리케이션의 엔드포인트를 업데이트할 필요가 없습니다. Valkey 7.2 이상 또는 Redis OSS 5.0.5 이상의 경우 클러스터가 계속 온라인 상태를 유지하고 수신 요청을 처리하는 동안 자동 장애 조치 활성화 클러스터를 확장할 수 있습니다. 버전 4.0.10 이하에서는 DNS 항목이 업데이트되는 동안 기본 노드에서 이전 버전의 읽기 및 쓰기가 잠시 중단될 수 있습니다.

대형 노드 유형으로 스케일 업하는 데 걸리는 시간은 노드 유형 및 현재 캐시 클러스터에 있는 데이터의 양에 따라 달라집니다.

Valkey 또는 Redis OSS Replication 그룹 확장(AWS CLI)

1. 다음 파라미터로 AWS CLI `list-allowed-node-type-modifications` 명령을 실행하여 확장할 수 있는 노드 유형을 결정합니다.

    - `--replication-group-id` - 복제 그룹의 이름입니다. 모든 복제 그룹 대신 특정 복제 그룹을 설명하려면 이 파라미터를 사용하세요.

    Linux, macOS, Unix의 경우:

```
aws elasticache list-allowed-node-type-modifications \
     --replication-group-id my-repl-group
```

Windows의 경우:

```
aws elasticache list-allowed-node-type-modifications ^
     --replication-group-id my-repl-group
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
     "ScaleUpModifications": [
         "cache.m3.2xlarge",
         "cache.m3.large",
         "cache.m3.xlarge",
         "cache.m4.10xlarge",
         "cache.m4.2xlarge",
         "cache.m4.4xlarge",
         "cache.m4.large",
         "cache.m4.xlarge",
         "cache.r3.2xlarge",
         "cache.r3.4xlarge",
         "cache.r3.8xlarge",
         "cache.r3.large",
         "cache.r3.xlarge"
     ]
 }
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.list-allowed-node-type-수정 AWS CLI 참조 의
.

2. 다음 파라미터를 사용하여 `modify-replication-group` 명령을 사용하여 AWS CLI 현재 복제 그룹을 새 노드 유형으로 확장합니다.

   - `--replication-group-id` - 복제 그룹의 이름입니다.
   - `--cache-node-type` - 이 복제 그룹에 있는 캐시 클러스터의 새로운 대형 노드 유형입니다. 이 값은 이전 단계에서 `list-allowed-node-type-modifications` 명령으로 반환된 인스턴스 유형 중 하나여야 합니다.

- `--cache-parameter-group-name` - [선택 사항] `reserved-memory`를 사용하여 클러스터의 예약된 메모리를 관리할 경우 이 파라미터를 사용합니다. 새 노드 유형에 대해 올바른 메모리 양을 예약하는 사용자 지정 캐시 파라미터 그룹을 지정합니다. `reserved-memory-percent`를 사용할 경우 이 파라미터를 생략할 수 있습니다.
- `--apply-immediately` - 스케일 업 프로세스가 즉시 적용되도록 합니다. 스케일 업 작업을 다음 유지 관리 기간으로 연기하려면 `--no-apply-immediately`를 사용하세요.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-replication-group \
     --replication-group-id my-repl-group \
     --cache-node-type cache.m3.xlarge \
     --cache-parameter-group-name redis32-m3-2xl \
     --apply-immediately
```

Windows의 경우:

```
aws elasticache modify-replication-group ^
     --replication-group-id my-repl-group ^
     --cache-node-type cache.m3.xlarge ^
     --cache-parameter-group-name redis32-m3-2xl \
     --apply-immediately
```

이 명령의 출력은 다음과 같습니다(JSON 형식).

```
{
 "ReplicationGroup": {
  "Status": "available",
  "Description": "Some description",
  "NodeGroups": [{
   "Status": "available",
   "NodeGroupMembers": [{
     "CurrentRole": "primary",
     "PreferredAvailabilityZone": "us-west-2b",
     "CacheNodeId": "0001",
     "ReadEndpoint": {
      "Port": 6379,
      "Address": "my-repl-group-001.8fdx4s.0001.usw2.cache.amazonaws.com"
     },
```

```
      "CacheClusterId": "my-repl-group-001"
     },
     {
      "CurrentRole": "replica",
      "PreferredAvailabilityZone": "us-west-2c",
      "CacheNodeId": "0001",
      "ReadEndpoint": {
       "Port": 6379,
       "Address": "my-repl-group-002.8fdx4s.0001.usw2.cache.amazonaws.com"
      },
      "CacheClusterId": "my-repl-group-002"
     }
    ],
    "NodeGroupId": "0001",
    "PrimaryEndpoint": {
     "Port": 6379,
     "Address": "my-repl-group.8fdx4s.ng.0001.usw2.cache.amazonaws.com"
    }
   }],
   "ReplicationGroupId": "my-repl-group",
   "SnapshotRetentionLimit": 1,
   "AutomaticFailover": "disabled",
   "SnapshotWindow": "12:00-13:00",
   "SnapshottingClusterId": "my-repl-group-002",
   "MemberClusters": [
    "my-repl-group-001",
    "my-repl-group-002"
   ],
   "PendingModifiedValues": {}
  }
}
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.modify-replication-group AWS CLI 참조 의 .

3. `--apply-immediately` 파라미터를 사용한 경우 다음 파라미터와 함께 `describe-replication-group` 명령을 사용하여 AWS CLI 복제 그룹의 상태를 모니터링합니다. 상태가 여전히 를 수정하고 있는 동안 5.0.6 버전 이상에서 실행되는 노드의 경우 가동 중지 시간이 1초 미만이고 DNS 항목이 업데이트되는 동안 기본 노드에서 이전 버전의 경우 읽기 및 쓰기가 잠시 중단될 수 있습니다.

   - `--replication-group-id` - 복제 그룹의 이름입니다. 모든 복제 그룹 대신 특정 복제 그룹을 설명하려면 이 파라미터를 사용하세요.

Linux, macOS, Unix의 경우:

```
aws elasticache describe-replication-groups \
     --replication-group-id my-replication-group
```

Windows의 경우:

```
aws elasticache describe-replication-groups ^
     --replication-group-id my-replication-group
```

자세한 내용은 참조describe-replication-groups의 섹션을 참조하세요. AWS CLI

Valkey 또는 Redis OSS 복제 그룹 크기 조정(ElastiCache API)

다음 프로세스는 를 사용하여 복제 그룹을 현재 노드 유형에서 더 큰 새 노드 유형으로 확장합니다 ElastiCache API. Valkey 7.2 이상 또는 Redis OSS 5.0.5 이상의 경우 클러스터가 계속 온라인 상태를 유지하고 수신 요청을 처리하는 동안 자동 장애 조치 활성화 클러스터를 확장할 수 있습니다. 버전 Redis OSS 4.0.10 이하에서는 DNS 항목이 업데이트되는 동안 기본 노드에서 이전 버전에 대한 읽기 및 쓰기가 잠시 중단될 수 있습니다.

대형 노드 유형으로 스케일 업하는 데 걸리는 시간은 노드 유형 및 현재 캐시 클러스터에 있는 데이터의 양에 따라 달라집니다.

Valkey 또는 Redis OSS Replication Group 확장(ElastiCache API)

1. 다음 파라미터를 사용하여 `ListAllowedNodeTypeModifications` 작업을 사용하여 ElastiCache API 확장할 수 있는 노드 유형을 결정합니다.

   - `ReplicationGroupId` - 복제 그룹의 이름입니다. 모든 복제 그룹 대신 특정 복제 그룹을 설명하려면 이 파라미터를 사용하세요.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ListAllowedNodeTypeModifications
    &ReplicationGroupId=MyReplGroup
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
```

```
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.ListAllowedNodeTypeModifications Amazon ElastiCache API 참조의 .

2. ModifyRedplicationGroup ElastiCache API 작업을 사용하고 다음 파라미터를 사용하여 현재 복제 그룹을 새 노드 유형으로 확장합니다.

   - ReplicationGroupId - 복제 그룹의 이름입니다.
   - CacheNodeType - 이 복제 그룹에 있는 캐시 클러스터의 새로운 대형 노드 유형입니다. 이 값은 이전 단계의 ListAllowedNodeTypeModifications 작업에서 반환된 인스턴스 유형 중 하나여야 합니다.
   - CacheParameterGroupName - [선택 사항] reserved-memory를 사용하여 클러스터의 예약된 메모리를 관리할 경우 이 파라미터를 사용합니다. 새 노드 유형에 대해 올바른 메모리 양을 예약하는 사용자 지정 캐시 파라미터 그룹을 지정합니다. reserved-memory-percent를 사용할 경우 이 파라미터를 생략할 수 있습니다.
   - ApplyImmediately - 스케일 업 프로세스가 즉시 적용되도록 하려면 true로 설정합니다. 스케일 업 프로세스를 다음 유지 관리 기간으로 연기하려면 ApplyImmediately=false를 사용하세요.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyReplicationGroup
    &ApplyImmediately=true
    &CacheNodeType=cache.m3.2xlarge
    &CacheParameterGroupName=redis32-m3-2xl
    &ReplicationGroupId=myReplGroup
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &Version=2014-12-01
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.ModifyReplicationGroup Amazon ElastiCache API 참조 의 .

3. 를 사용한 경우 다음 파라미터로 DescribeReplicationGroups 작업을 사용하여 ElastiCache API 복제 그룹의 상태를 ApplyImmediately=true모니터링합니다. 상태가 수정 중에서 사용 가능으로 변경되면 스케일 업된 새 복제 그룹에 쓰기를 시작할 수 있습니다.

   - ReplicationGroupId - 복제 그룹의 이름입니다. 모든 복제 그룹 대신 특정 복제 그룹을 설명하려면 이 파라미터를 사용하세요.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeReplicationGroups
    &ReplicationGroupId=MyReplGroup
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

   자세한 내용을 알아보려면 다음 섹션을 참조하세요.DescribeReplicationGroups Amazon ElastiCache API 참조의 .

복제본을 사용하여 Valkey 또는 Redis OSS 클러스터 축소

다음 섹션에서는 복제본 노드가 있는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 캐시 클러스터를 더 작은 노드 유형으로 축소하는 방법을 안내합니다. 성공을 위해 새로운 소형 노드 유형이 모든 데이터와 오버헤드를 수용할 만큼 충분히 큰지 확인하는 것이 매우 중요합니다. 자세한 내용은 Valkey 또는 Redis OSS 스냅샷을 생성하기에 충분한 메모리 확보 단원을 참조하십시오.

> **Note**
>
> r6gd 노드 유형을 실행하는 클러스터의 경우 r6gd 노드 패밀리 내의 노드 크기로만 조정할 수 있습니다.

> **Important**
>
> 파라미터 그룹이 `reserved-memory`를 사용하여 Valkey 또는 Redis OSS 오버헤드용 메모리를 따로 설정하는 경우 크기 조정을 시작하기 전에 새 노드 유형에 맞는 메모리 양을 예약하는 사용자 지정 파라미터 그룹이 있는지 확인합니다. 또는 `reserved-memory-percent`를 사용하고 새 클러스터에 대해 해당 파라미터 그룹을 사용하도록 사용자 지정 파라미터 그룹을 수정할 수 있습니다.
>
> `reserved-memory-percent`를 사용할 경우에는 이렇게 하지 않아도 됩니다.
>
> 자세한 내용은 Valkey 및 Redis용 예약 메모리 관리 OSS 단원을 참조하십시오.

복제본을 사용하여 Valkey 또는 Redis OSS 클러스터 축소(콘솔)

다음 프로세스는 ElastiCache 콘솔을 사용하여 복제본 노드가 있는 Valkey 또는 Redis OSS 클러스터를 더 작은 노드 유형으로 확장합니다.

복제본 노드를 사용하여 Valkey 또는 Redis OSS 클러스터를 축소하려면(콘솔)

1. 소형 노드 유형이 데이터 및 오버헤드 요구 사항에 적합한지 확인합니다.
2. 파라미터 그룹이 `reserved-memory`를 사용하여 Valkey 또는 Redis OSS 오버헤드용 메모리를 따로 설정하는 경우 새 노드 유형에 맞는 메모리 양을 따로 설정할 사용자 지정 파라미터 그룹이 있는지 확인합니다.

   또는 `reserved-memory-percent`를 사용하여 사용자 지정 파라미터 그룹을 수정할 수 있습니다. 자세한 내용은 Valkey 및 Redis용 예약 메모리 관리 OSS 단원을 참조하십시오.

3. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
4. 클러스터 목록에서 스케일 다운할 클러스터를 선택합니다. 이 클러스터는 클러스터링된 Valkey 또는 Redis OSS 엔진이 아닌 Valkey 또는 Redis OSS 엔진을 실행 중이어야 합니다.
5. 수정을 선택합니다.
6. [Modify Cluster] 마법사에서 다음을 수행합니다.
   a. [Node type] 목록에서 스케일 다운할 노드 유형을 선택합니다.
   b. `reserved-memory`를 사용하여 메모리를 관리할 경우 [Parameter Group] 목록에서 새 노드 유형에 대해 올바른 메모리 양을 예약하는 사용자 지정 파라미터 그룹을 선택합니다.
7. 스케일 다운 프로세스를 즉시 수행하려면 [Apply immediately] 확인란을 선택합니다. [Apply immediately] 확인란을 선택하지 않고 비워 두면 이 클러스터의 다음 유지 관리 기간 중 스케일 다운 프로세스가 수행됩니다.
8. 수정을 선택합니다.
9. 클러스터의 상태가 수정 중에서 사용 가능으로 변경되면 클러스터가 새 노드 유형으로 조정된 것입니다. 애플리케이션에서 엔드포인트를 업데이트할 필요가 없습니다.

Valkey 또는 Redis OSS 복제 그룹 축소(AWS CLI)

다음 절차는 AWS CLI를 사용하여 복제 그룹을 현재 노드 유형에서 새롭고 더 작은 노드 유형으로 조정합니다. 이 프로세스 중에 는 새 노드를 가리키도록 DNS 항목을 ElastiCache 업데이트합니다. 따라서 애플리케이션의 엔드포인트를 업데이트할 필요가 없습니다. 위 Valkey 7.2 또는 Redis OSS 5.0.5 이상의 경우 클러스터가 계속 온라인 상태를 유지하고 수신 요청을 처리하는 동안 자동 장애 조치 활성화 클러스터를 확장할 수 있습니다. 버전 4.0.10 이하에서는 DNS 항목이 업데이트되는 동안 기본 노드에서 이전 버전의 읽기 및 쓰기가 잠시 중단될 수 있습니다.

그러나 읽기 전용 복제본 캐시 클러스터에서 읽기는 계속 중단되지 않습니다.

더 작은 노드 유형으로 축소하는 데 걸리는 시간은 노드 유형 및 현재 캐시 클러스터에 있는 데이터의 양에 따라 달라집니다.

Valkey 또는 Redis OSS Replication Group을 축소하려면(AWS CLI)

1. 다음 파라미터로 `list-allowed-node-type-modifications` 명령을 실행 AWS CLI 하여 확장할 수 있는 노드 유형을 결정합니다.

- `--replication-group-id` - 복제 그룹의 이름입니다. 모든 복제 그룹 대신 특정 복제 그룹을 설명하려면 이 파라미터를 사용하세요.

Linux, macOS, Unix의 경우:

```
aws elasticache list-allowed-node-type-modifications \
     --replication-group-id my-repl-group
```

Windows의 경우:

```
aws elasticache list-allowed-node-type-modifications ^
     --replication-group-id my-repl-group
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
     "ScaleDownModifications": [
         "cache.m3.2xlarge",
         "cache.m3.large",
         "cache.m3.xlarge",
         "cache.m4.10xlarge",
         "cache.m4.2xlarge",
         "cache.m4.4xlarge",
         "cache.m4.large",
         "cache.m4.xlarge",
         "cache.r3.2xlarge",
         "cache.r3.4xlarge",
         "cache.r3.8xlarge",
         "cache.r3.large",
         "cache.r3.xlarge"
     ]
 }
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.list-allowed-node-type-수정 AWS CLI 참조 의
.

2. 다음 파라미터를 사용하여 `modify-replication-group` 명령을 사용하여 AWS CLI 현재 복제 그룹을 새 노드 유형으로 확장합니다.

- `--replication-group-id` - 복제 그룹의 이름입니다.

- `--cache-node-type` - 이 복제 그룹에 있는 캐시 클러스터의 새롭고 더 작은 노드 유형입니다. 이 값은 이전 단계에서 `list-allowed-node-type-modifications` 명령으로 반환된 인스턴스 유형 중 하나여야 합니다.
- `--cache-parameter-group-name` - [선택 사항] `reserved-memory`를 사용하여 클러스터의 예약된 메모리를 관리할 경우 이 파라미터를 사용합니다. 새 노드 유형에 대해 올바른 메모리 양을 예약하는 사용자 지정 캐시 파라미터 그룹을 지정합니다. `reserved-memory-percent`를 사용할 경우 이 파라미터를 생략할 수 있습니다.
- `--apply-immediately` - 스케일 업 프로세스가 즉시 적용되도록 합니다. 스케일 업 작업을 다음 유지 관리 기간으로 연기하려면 `--no-apply-immediately`를 사용하세요.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-replication-group \
     --replication-group-id my-repl-group \
     --cache-node-type cache.t2.small  \
     --cache-parameter-group-name redis32-m3-2xl \
     --apply-immediately
```

Windows의 경우:

```
aws elasticache modify-replication-group ^
     --replication-group-id my-repl-group ^
     --cache-node-type cache.t2.small  ^
     --cache-parameter-group-name redis32-m3-2xl \
     --apply-immediately
```

이 명령의 출력은 다음과 같습니다(JSON 형식).

```
{"ReplicationGroup": {
         "Status": "available",
         "Description": "Some description",
         "NodeGroups": [
             {
                 "Status": "available",
                 "NodeGroupMembers": [
                     {
                         "CurrentRole": "primary",
                         "PreferredAvailabilityZone": "us-west-2b",
```

```
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Port": 6379,
                            "Address": "my-repl-
group-001.8fdx4s.0001.usw2.cache.amazonaws.com"
                        },
                        "CacheClusterId": "my-repl-group-001"
                    },
                    {
                        "CurrentRole": "replica",
                        "PreferredAvailabilityZone": "us-west-2c",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Port": 6379,
                            "Address": "my-repl-
group-002.8fdx4s.0001.usw2.cache.amazonaws.com"
                        },
                        "CacheClusterId": "my-repl-group-002"
                    }
                ],
                "NodeGroupId": "0001",
                "PrimaryEndpoint": {
                    "Port": 6379,
                    "Address": "my-repl-
group.8fdx4s.ng.0001.usw2.cache.amazonaws.com"
                }
            }
        ],
        "ReplicationGroupId": "my-repl-group",
        "SnapshotRetentionLimit": 1,
        "AutomaticFailover": "disabled",
        "SnapshotWindow": "12:00-13:00",
        "SnapshottingClusterId": "my-repl-group-002",
        "MemberClusters": [
            "my-repl-group-001",
            "my-repl-group-002",
        ],
        "PendingModifiedValues": {}
    }
}
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.modify-replication-group AWS CLI 참조 의 .

3. `--apply-immediately` 파라미터를 사용한 경우 다음 파라미터와 함께 `describe-replication-group` 명령을 사용하여 AWS CLI 복제 그룹의 상태를 모니터링합니다. 상태가 수정 중에서 사용 가능으로 변경되면 축소된 새 복제 그룹에 쓰기를 시작할 수 있습니다.

   - `--replication-group-id` - 복제 그룹의 이름입니다. 모든 복제 그룹 대신 특정 복제 그룹을 설명하려면 이 파라미터를 사용하세요.

   Linux, macOS, Unix의 경우:

   ```
   aws elasticache describe-replication-group \
        --replication-group-id my-replication-group
   ```

   Windows의 경우:

   ```
   aws elasticache describe-replication-groups ^
        --replication-group-id my-replication-group
   ```

   자세한 내용은 참조describe-replication-groups의 섹션을 참조하세요. AWS CLI

Valkey 또는 Redis OSS 복제 그룹 축소(ElastiCache API)

다음 프로세스는 를 사용하여 복제 그룹을 현재 노드 유형에서 더 작은 새 노드 유형으로 확장합니다 ElastiCache API. 이 프로세스 중에 는 새 노드를 가리키도록 DNS 항목을 ElastiCache 업데이트합니다. 따라서 애플리케이션의 엔드포인트를 업데이트할 필요가 없습니다. Valkey 7.2 이상 또는 Redis OSS 5.0.5 이상의 경우 클러스터가 계속 온라인 상태를 유지하고 수신 요청을 처리하는 동안 자동 장애 조치 활성화 클러스터를 확장할 수 있습니다. Redis OSS 버전 4.0.10 이하에서는 DNS 항목이 업데이트되는 동안 기본 노드에서 이전 버전의 읽기 및 쓰기가 잠시 중단될 수 있습니다. 그러나 읽기 전용 복제본 캐시 클러스터에서 읽기는 계속 중단되지 않습니다.

더 작은 노드 유형으로 축소하는 데 걸리는 시간은 노드 유형 및 현재 캐시 클러스터에 있는 데이터의 양에 따라 달라집니다.

Valkey 또는 Redis OSS Replication 그룹 축소(ElastiCache API)

1. 다음 파라미터를 사용하여 `ListAllowedNodeTypeModifications` 작업을 사용하여 ElastiCache API 축소할 수 있는 노드 유형을 결정합니다.

- `ReplicationGroupId` - 복제 그룹의 이름입니다. 모든 복제 그룹 대신 특정 복제 그룹을 설명하려면 이 파라미터를 사용하세요.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ListAllowedNodeTypeModifications
    &ReplicationGroupId=MyReplGroup
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.ListAllowedNodeTypeModifications Amazon ElastiCache API 참조의 .

2. `ModifyRedplicationGroup` ElastiCache API 작업을 사용하고 다음 파라미터를 사용하여 현재 복제 그룹을 새 노드 유형으로 확장합니다.

   - `ReplicationGroupId` - 복제 그룹의 이름입니다.
   - `CacheNodeType` - 이 복제 그룹에 있는 캐시 클러스터의 새롭고 더 작은 노드 유형입니다. 이 값은 이전 단계의 `ListAllowedNodeTypeModifications` 작업에서 반환된 인스턴스 유형 중 하나여야 합니다.
   - `CacheParameterGroupName` - [선택 사항] `reserved-memory`를 사용하여 클러스터의 예약된 메모리를 관리할 경우 이 파라미터를 사용합니다. 새 노드 유형에 대해 올바른 메모리 양을 예약하는 사용자 지정 캐시 파라미터 그룹을 지정합니다. `reserved-memory-percent`를 사용할 경우 이 파라미터를 생략할 수 있습니다.
   - `ApplyImmediately` - 스케일 업 프로세스가 즉시 적용되도록 하려면 `true`로 설정합니다. 축소 프로세스를 다음 유지 관리 기간으로 연기하려면 `ApplyImmediately=false`를 사용하세요.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyReplicationGroup
    &ApplyImmediately=true
    &CacheNodeType=cache.m3.2xlarge
    &CacheParameterGroupName=redis32-m3-2xl
    &ReplicationGroupId=myReplGroup
    &SignatureVersion=4
```

```
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &Version=2014-12-01
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.ModifyReplicationGroup Amazon ElastiCache API 참조의 .

3. 를 사용한 경우 다음 파라미터로 DescribeReplicationGroups 작업을 사용하여 ElastiCache API 복제 그룹의 상태를 ApplyImmediately=true모니터링합니다. 상태가 수정 중에서 사용 가능으로 변경되면 축소된 새 복제 그룹에 쓰기를 시작할 수 있습니다.

   - ReplicationGroupId - 복제 그룹의 이름입니다. 모든 복제 그룹 대신 특정 복제 그룹을 설명하려면 이 파라미터를 사용하세요.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeReplicationGroups
    &ReplicationGroupId=MyReplGroup
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.DescribeReplicationGroups Amazon ElastiCache API 참조 의 .

읽기 용량 늘리기

읽기 용량을 늘리려면 읽기 전용 복제본(최대 5개)을 Valkey 또는 Redis OSS 복제 그룹에 추가합니다.

ElastiCache 콘솔, 또는 를 사용하여 Valkey AWS CLI또는 Redis OSS 클러스터의 읽기 용량을 조정할 수 있습니다 ElastiCache API. 자세한 내용은 Valkey 또는 Redis에 대한 읽기 전용 복제본 추가OSS(클러스터 모드 비활성화됨) 단원을 참조하십시오.

읽기 용량 줄이기

읽기 용량을 줄이려면 복제본(API/에서 복제 그룹이라고 함)이 있는 Valkey 또는 Redis OSS 클러스터에서 하나 이상의 읽기 전용 복제본을 삭제합니다CLI. 클러스터가 자동 장애 조치가 활성화된 다중 AZ인 경우 먼저 다중 AZ를 비활성화해야 마지막 읽기 전용 복제본을 삭제할 수 있습니다. 자세한 내용은 복제 그룹 수정 단원을 참조하십시오.

자세한 내용은 Valkey 또는 Redis에 대한 읽기 전용 복제본 삭제OSS(클러스터 모드 비활성화됨) 단원을 참조하십시오.

Valkey 또는 Redis에서 클러스터 크기 조정OSS(클러스터 모드 활성화됨)

클러스터에 대한 수요가 변경되면 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 샤드 수를 변경하여 성능을 개선하거나 비용을 절감할 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, 및 지표를 모니터링하여 노드에 메모리 압력이 있는지 확인할 수 있습니다SwapUsageBytesUseForCache.
- CPU 또는 네트워크 병목 현상:

  클러스터에서 지연 시간/처리량 문제가 발생하면 문제를 해결하기 위해 확장이 필요할 수 있습니다.

  CPUUtilization, , , 및 지표를 모니터링하여 지연 시간 NetworkBytesIn NetworkBytesOut CurrConnections및 처리량 수준을 모니터링할 수 있습니다NewConnections.
- 클러스터가 과도하게 조정됨:

  축소와 같은 클러스터에 대한 현재 수요는 성능을 저하시키지 않고 비용을 줄입니다.

  클러스터의 사용을 모니터링하여 FreeableMemory, , , , , , 및 지표를 사용하여 안전하게 규모를 조정할 수 있는지 여부를 결정할 수 SwapUsage BytesUseForCache CPUUtilization NetworkBytesIn NetworkBytesOut CurrConnections있습니다NewConnections.

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. 발생하는 성능 저하의 정도는 일반적인 CPU 사용률과 데이터에 따라 달라집니다.

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터를 확장하는 방법에는 수평 및 수직 조정이라는 두 가지가 있습니다.

- 수평 확장에서는 노드 그룹(샤드)을 추가 또는 제거하여 복제 그룹 내 노드 그룹(샤드) 수를 변경할 수 있습니다. 온라인 리샤딩 프로세스를 통해 클러스터가 들어오는 요청을 계속 처리하는 동안 확장/축소할 수 있습니다.

  새 클러스터에서 이전 클러스터에서와 달리 슬롯을 구성합니다. 오프라인 방법에만 해당합니다.
- 수직 확장 - 노드 유형을 변경하여 클러스터의 크기를 조정합니다. 온라인 수직 확장을 통해 클러스터가 들어오는 요청을 계속 처리하는 동안 확장/축소할 수 있습니다.

크기 조정 또는 크기 축소를 통해 클러스터의 크기와 메모리 용량을 줄이는 경우 새 구성에 데이터와 Valkey 또는 Redis OSS 오버헤드를 위한 충분한 메모리가 있는지 확인합니다.

자세한 내용은 캐시 노드 크기 선택을 참조하세요.

목차

- Valkey 또는 Redis OSS 캐시 클러스터 축소(ElastiCache API)

Valkey 또는 Redis에 대한 오프라인 리샤딩OSS(클러스터 모드 활성화됨)

오프라인 리샤딩 재구성의 주요 이점은 복제 그룹에서 단순히 샤드를 추가 또는 제거하는 것 이상을 할 수 있다는 점입니다. 오프라인으로 리샤딩하고 리밸런싱할 때 복제 그룹의 샤드 수를 변경하는 것 외에도 다음을 수행할 수 있습니다.

> **Note**
>
> 데이터 계층화가 활성화된 Valkey 또는 Redis OSS 클러스터에서는 오프라인 리샤딩이 지원되지 않습니다. 자세한 내용은 의 데이터 계층화 ElastiCache 단원을 참조하십시오.

- 복제 그룹의 노드 유형을 변경합니다.
- 복제 그룹의 각 노드에 대한 가용 영역을 지정합니다.
- 최신 엔진 버전으로 업그레이드합니다.
- 각 샤드 내 복제 노드 수를 독립적으로 지정합니다.
- 각 샤드에 대한 키스페이스를 지정합니다.

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 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터를 오프라인으로 재구성하려면

1. 기존 Valkey 또는 Redis OSS 클러스터의 수동 백업을 생성합니다. 자세한 내용은 수동 백업 지원 단원을 참조하십시오.
2. 백업에서 복원해 새 클러스터를 생성합니다. 자세한 내용은 백업에서 새 캐시로 복원 단원을 참조하십시오.
3. 애플리케이션에서 엔드포인트를 새 클러스터의 엔드포인트로 업데이트합니다. 자세한 내용은 에서 연결 엔드포인트 찾기 ElastiCache 단원을 참조하십시오.

Valkey 또는 Redis에 대한 온라인 리샤딩OSS(클러스터 모드 활성화됨)

ElastiCache Valkey 7.2 이상 또는 Redis OSS 버전 3.2.10 이상에서 온라인 리샤딩 및 샤드 리밸런싱을 사용하면 가동 중지 없이 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터를 동적으로 확장할 수 있습니다. 이러한 접근 방식은 조정 또는 재분배 진행 중에도 클러스터에서 계속해서 요청을 처리할 수 있음을 의미합니다.

다음을 수행할 수 있습니다.

- 스케일 아웃 - Valkey 또는 Redis(클러스터 모드 활성화됨) 클러스터(복제 그룹)에 샤드OSS(노드 그룹)를 추가하여 읽기 및 쓰기 용량을 늘립니다.

  복제 그룹에 샤드를 하나 이상 추가하는 경우 각 샤드의 노드 수는 기존의 가장 작은 샤드에 있는 노드 수와 동일합니다.
- 스케일 인 - Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터에서 샤드를 제거하여 읽기 및 쓰기 용량을 줄이고 비용을 절감합니다.
- 리밸런싱 - 키스페이스를 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 샤드 간에 이동하여 키스페이스가 가능한 한 샤드 간에 균등하게 분산되도록 합니다.

다음을 수행할 수 없습니다.

- 독립적으로 샤드 구성:

  샤드의 키스페이스는 독립적으로 지정할 수 없습니다. 이렇게 하려면 오프라인 프로세스를 사용해야 합니다.

현재 ElastiCache 온라인 리샤딩 및 리밸런싱에는 다음 제한이 적용됩니다.

- 이러한 프로세스에는 Valkey 7.2 이상 또는 Redis OSS 엔진 버전 3.2.10 이상이 필요합니다. 엔진 버전 업그레이드에 대한 자세한 내용은 용 버전 관리 ElastiCache 섹션을 참조하세요.
- 슬롯 또는 키스페이스와 대용량 항목에 대한 제한 사항이 있습니다.

  샤드 내 키에 대용량 항목이 포함되어 있으면 확장 또는 재분배 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.
- 확장 시 새 샤드의 노드 수는 기존의 가장 작은 노드 수와 동일합니다.

- 확장 시 기존의 모든 샤드에 공통된 태그는 새 샤드로 복사됩니다.
- Global Data Store 클러스터를 확장할 때 ElastiCache 는 기존 노드 중 하나에서 새 노드(들)로 함수를 자동으로 복제하지 않습니다. 클러스터를 스케일 아웃한 후, 새 샤드에 함수를 로드하여 모든 샤드가 동일한 함수를 갖도록 하는 것이 좋습니다.

> **Note**
>
> 에서 ElastiCache Valkey 7.2 이상 및 Redis OSS 버전 7 이상: 클러스터를 확장할 때 ElastiCache 는 기존 노드(임의로 선택됨) 중 하나에 로드된 함수를 새 노드(들)에 자동으로 복제합니다. 애플리케이션이 함수 를 사용하는 경우 클러스터가 다른 샤드에 대해 다른 함수 정의로 끝나지 않도록 확장하기 전에 모든 함수를 모든 샤드에 로드하는 것이 좋습니다.

자세한 내용은 온라인 클러스터 크기 조정 단원을 참조하십시오.

AWS Management Console, 및 를 사용하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터를 수평으로 조정하거나 재조정 AWS CLI할 수 있습니다 ElastiCache API.

온라인 리샤딩을 사용하여 샤드 추가

AWS Management Console AWS CLI, 또는 ElastiCache 를 사용하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터에 샤드를 추가할 수 있습니다API. Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터에 샤드를 추가하면 기존 샤드의 모든 태그가 새 샤드에 복사됩니다.

샤드 추가(콘솔)

AWS Management Console 를 사용하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터에 샤드를 하나 이상 추가할 수 있습니다. 다음 절차에서는 이러한 프로세스를 설명합니다.

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터에 샤드를 추가하려면

1. 에서 ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 탐색 창에서 Valkey 클러스터 또는 Redis OSS 클러스터를 선택합니다.
3. 샤드를 추가할 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 클러스터 이름 왼쪽에 있는 상자가 아닌 이름을 찾아 선택합니다.

> **Tip**
>
> Valkey 또는 RedisOSS(클러스터 모드 활성화됨)가 모드 열에 클러스터형 Valkey 또는 클러스터형 RedisOSS를 표시합니다.

4. [Add shard]를 선택합니다.

   a. [Number of shards to be added]에서 이 클러스터에 추가할 샤드 수를 선택합니다.

   b. [Availability zone(s)]에서는 [No preference] 또는 [Specify availability zones]을 선택합니다.

   c. [Specify availability zones]를 선택한 경우 각 샤드의 각 노드에 대해 [Availability Zones] 목록에서 노드의 가용 영역을 선택합니다.

   d. 추가를 선택합니다.

샤드 추가(AWS CLI)

다음 프로세스에서는 를 사용하여 샤드를 추가하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 샤드를 재구성하는 방법을 설명합니다 AWS CLI.

`modify-replication-group-shard-configuration`에 다음 파라미터를 사용합니다.

파라미터

- `--apply-immediately` - 필수입니다. 즉시 시작할 샤드 재구성 작업을 지정합니다.
- `--replication-group-id` - 필수입니다. 샤드 재구성 작업을 수행할 복제 그룹(클러스터)을 지정합니다.
- `--node-group-count` - 필수입니다. 작업 완료 시 존재할 샤드(노드 그룹) 수를 지정합니다. 샤드를 추가하는 경우 `--node-group-count`의 값은 현재 샤드 수보다 커야 합니다.

  경우에 따라 `--resharding-configuration`을 사용해 복제 그룹의 각 노드에 대한 가용 영역을 지정할 수 있습니다.
- `--resharding-configuration` – 선택 사항입니다. 복제 그룹 내에 있는 각 샤드의 개별 노드에 대한 기본 가용 영역 목록입니다. 이 파라미터는 `--node-group-count`의 값이 현재 샤드 수보다 큰 경우에만 사용합니다. 샤드를 추가할 때 이 파라미터가 생략되면 Amazon ElastiCache 은 새 노드의 가용 영역을 선택합니다.

다음 예제에서는 이름이 인 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 샤드 4개를 통해 키스페이스를 재구성합니다`my-cluster`. 또한 이 예에서는 각 샤드 내 개별 노드에 대한 가용 영역을 지정합니다. 작업이 즉시 시작됩니다.

Example - 샤드 추가

Linux, macOS, Unix의 경우:

```
aws elasticache modify-replication-group-shard-configuration \
    --replication-group-id my-cluster \
    --node-group-count 4 \
    --resharding-configuration \
        "PreferredAvailabilityZones=us-east-2a,us-east-2c" \
        "PreferredAvailabilityZones=us-east-2b,us-east-2a" \
        "PreferredAvailabilityZones=us-east-2c,us-east-2d" \
        "PreferredAvailabilityZones=us-east-2d,us-east-2c" \
    --apply-immediately
```

Windows의 경우:

```
aws elasticache modify-replication-group-shard-configuration ^
    --replication-group-id my-cluster ^
    --node-group-count 4 ^
    --resharding-configuration ^
        "PreferredAvailabilityZones=us-east-2a,us-east-2c" ^
        "PreferredAvailabilityZones=us-east-2b,us-east-2a" ^
        "PreferredAvailabilityZones=us-east-2c,us-east-2d" ^
        "PreferredAvailabilityZones=us-east-2d,us-east-2c" ^
    --apply-immediately
```

자세한 내용은 AWS CLI 설명서의 modify-replication-group-shard-configuration을 참조하세요.

샤드 추가(ElastiCache API)

를 ElastiCache API 사용하여 `ModifyReplicationGroupShardConfiguration` 작업을 사용하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 샤드를 온라인으로 재구성할 수 있습니다.

`ModifyReplicationGroupShardConfiguration`에 다음 파라미터를 사용합니다.

파라미터

- `ApplyImmediately=true` - 필수입니다. 즉시 시작할 샤드 재구성 작업을 지정합니다.

- ReplicationGroupId - 필수입니다. 샤드 재구성 작업을 수행할 복제 그룹(클러스터)을 지정합니다.
- NodeGroupCount - 필수입니다. 작업 완료 시 존재할 샤드(노드 그룹) 수를 지정합니다. 샤드를 추가하는 경우 NodeGroupCount의 값은 현재 샤드 수보다 커야 합니다.

  경우에 따라 ReshardingConfiguration을 사용해 복제 그룹의 각 노드에 대한 가용 영역을 지정할 수 있습니다.
- ReshardingConfiguration – 선택 사항입니다. 복제 그룹 내에 있는 각 샤드의 개별 노드에 대한 기본 가용 영역 목록입니다. 이 파라미터는 NodeGroupCount의 값이 현재 샤드 수보다 큰 경우에만 사용합니다. 샤드를 추가할 때 이 파라미터가 생략되면 Amazon은 새 노드의 가용 영역을 ElastiCache 선택합니다.

다음 프로세스에서는 를 사용하여 샤드를 추가하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 샤드를 재구성하는 방법을 설명합니다 ElastiCache API.

Example - 샤드 추가

다음 예제에서는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터에 노드 그룹을 추가my-cluster하므로 작업이 완료되면 총 4개의 노드 그룹이 있습니다. 또한 이 예에서는 각 샤드 내 개별 노드에 대한 가용 영역을 지정합니다. 작업이 즉시 시작됩니다.

```
https://elasticache.us-east-2.amazonaws.com/
    ?Action=ModifyReplicationGroupShardConfiguration
    &ApplyImmediately=true
    &NodeGroupCount=4
    &ReplicationGroupId=my-cluster

 &ReshardingConfiguration.ReshardingConfiguration.1.PreferredAvailabilityZones.AvailabilityZone
east-2a

 &ReshardingConfiguration.ReshardingConfiguration.1.PreferredAvailabilityZones.AvailabilityZone
east-2c

 &ReshardingConfiguration.ReshardingConfiguration.2.PreferredAvailabilityZones.AvailabilityZone
east-2b

 &ReshardingConfiguration.ReshardingConfiguration.2.PreferredAvailabilityZones.AvailabilityZone
east-2a
```

```
 &ReshardingConfiguration.ReshardingConfiguration.3.PreferredAvailabilityZones.AvailabilityZone
east-2c

 &ReshardingConfiguration.ReshardingConfiguration.3.PreferredAvailabilityZones.AvailabilityZone
east-2d

 &ReshardingConfiguration.ReshardingConfiguration.4.PreferredAvailabilityZones.AvailabilityZone
east-2d

 &ReshardingConfiguration.ReshardingConfiguration.4.PreferredAvailabilityZones.AvailabilityZone
east-2c
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20171002T192317Z
    &X-Amz-Credential=<credential>
```

자세한 내용은 참조ModifyReplicationGroupShardConfiguration의 섹션을 참조하세요 ElastiCache API.

온라인 리샤딩을 사용하여 샤드 제거

AWS Management Console AWS CLI, 또는 를 사용하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터에서 샤드를 제거할 수 있습니다ElastiCache API.

주제

- 샤드 제거(콘솔)
- 샤드 제거(AWS CLI)
- 샤드 제거(ElastiCache API)

샤드 제거(콘솔)

다음 프로세스에서는 를 사용하여 샤드를 제거하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 샤드를 재구성하는 방법을 설명합니다 AWS Management Console.

복제 그룹에서 노드 그룹(샤드)을 제거하기 전에 모든 데이터가 나머지 샤드에 맞는지 ElastiCache 확인합니다. 데이터가 맞으면 요청된 대로 지정된 샤드가 복제 그룹에서 삭제됩니다. 데이터가 나머지 노드 그룹에 맞지 않으면 프로세스가 종료되고 복제 그룹은 요청이 작성되기 전과 동일한 노드 그룹 구성으로 남습니다.

AWS Management Console 를 사용하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터에서 하나 이상의 샤드를 제거할 수 있습니다. 복제 그룹에서 샤드를 모두 제거할 수는 없습니다. 대신 복제 그룹을 삭제해야 합니다. 자세한 내용은 복제 그룹 삭제 단원을 참조하십시오. 다음 절차는 샤드를 하나 이상 삭제하는 프로세스를 설명합니다.

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터에서 샤드를 제거하려면

1. 에서 ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 탐색 창에서 Valkey 클러스터 또는 Redis OSS 클러스터를 선택합니다.
3. 클러스터 이름 왼쪽의 상자가 아닌 샤드를 제거할 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 이름을 찾아 선택합니다.

   > **Tip**
   >
   > 샤드 열에서 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 값은 1 이상입니다.

4. 샤드 목록에서 삭제하고자 하는 각 샤드의 이름 왼쪽에 있는 상자를 선택합니다.
5. [Delete shard]를 선택합니다.

샤드 제거(AWS CLI)

다음 프로세스에서는 를 사용하여 샤드를 제거하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 샤드를 재구성하는 방법을 설명합니다 AWS CLI.

> **Important**
>
> 복제 그룹에서 노드 그룹(샤드)을 제거하기 전에 모든 데이터가 나머지 샤드에 맞는지 ElastiCache 확인합니다. 데이터가 맞으면 요청된 대로 지정된 샤드(`--node-groups-to-remove`)가 복제 그룹에서 삭제되고 해당 샤드의 키스페이스가 나머지 샤드로 매핑됩니다. 데이터가 나머지 노드 그룹에 맞지 않으면 프로세스가 종료되고 복제 그룹은 요청이 작성되기 전과 동일한 노드 그룹 구성으로 남습니다.

AWS CLI 를 사용하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터에서 하나 이상의 샤드를 제거할 수 있습니다. 복제 그룹에서 샤드를 모두 제거할 수는 없습니다. 대신 복제 그룹을 삭제해야 합니다. 자세한 내용은 복제 그룹 삭제 단원을 참조하십시오.

modify-replication-group-shard-configuration에 다음 파라미터를 사용합니다.

파라미터

- --apply-immediately - 필수입니다. 즉시 시작할 샤드 재구성 작업을 지정합니다.
- --replication-group-id - 필수입니다. 샤드 재구성 작업을 수행할 복제 그룹(클러스터)을 지정합니다.
- --node-group-count - 필수입니다. 작업 완료 시 존재할 샤드(노드 그룹) 수를 지정합니다. 샤드를 제거하는 경우 --node-group-count의 값은 현재 샤드 수보다 작아야 합니다.

- --node-groups-to-remove - --node-group-count가 노드 그룹(샤드)의 현재 수보다 작은 경우에만 필요합니다. 복제 그룹에서 IDs 제거할 샤드(노드 그룹) 목록입니다.

다음 절차는 샤드를 하나 이상 삭제하는 프로세스를 설명합니다.

Example - 샤드 제거

다음 예제에서는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 에서 두 개의 노드 그룹을 제거my-cluster하므로 작업이 완료될 때 총 두 개의 노드 그룹이 있습니다. 제거된 샤드의 키스페이스는 나머지 샤드 간에 균일하게 분배됩니다.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-replication-group-shard-configuration \
    --replication-group-id my-cluster \
    --node-group-count 2 \
    --node-groups-to-remove "0002" "0003" \
    --apply-immediately
```

Windows의 경우:

```
aws elasticache modify-replication-group-shard-configuration ^
    --replication-group-id my-cluster ^
    --node-group-count 2 ^
    --node-groups-to-remove "0002" "0003" ^
    --apply-immediately
```

샤드 제거(ElastiCache API)

를 ElastiCache API 사용하여 `ModifyReplicationGroupShardConfiguration` 작업을 사용하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 샤드를 온라인으로 재구성할 수 있습니다.

다음 프로세스에서는 를 사용하여 샤드를 제거하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 샤드를 재구성하는 방법을 설명합니다 ElastiCache API.

> **Important**
>
> 복제 그룹에서 노드 그룹(샤드)을 제거하기 전에 모든 데이터가 나머지 샤드에 맞는지 ElastiCache 확인합니다. 데이터가 맞으면 요청된 대로 지정된 샤드(`NodeGroupsToRemove`)가 복제 그룹에서 삭제되고 해당 샤드의 키스페이스가 나머지 샤드로 매핑됩니다. 데이터가 나머지 노드 그룹에 맞지 않으면 프로세스가 종료되고 복제 그룹은 요청이 작성되기 전과 동일한 노드 그룹 구성으로 남습니다.

를 ElastiCache API 사용하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터에서 하나 이상의 샤드를 제거할 수 있습니다. 복제 그룹에서 샤드를 모두 제거할 수는 없습니다. 대신 복제 그룹을 삭제해야 합니다. 자세한 내용은 복제 그룹 삭제 단원을 참조하십시오.

`ModifyReplicationGroupShardConfiguration`에 다음 파라미터를 사용합니다.

파라미터

- `ApplyImmediately=true` - 필수입니다. 즉시 시작할 샤드 재구성 작업을 지정합니다.
- `ReplicationGroupId` - 필수입니다. 샤드 재구성 작업을 수행할 복제 그룹(클러스터)을 지정합니다.
- `NodeGroupCount` - 필수입니다. 작업 완료 시 존재할 샤드(노드 그룹) 수를 지정합니다. 샤드를 제거하는 경우 `NodeGroupCount`의 값은 현재 샤드 수보다 작아야 합니다.
- `NodeGroupsToRemove` - `--node-group-count`가 노드 그룹(샤드)의 현재 수보다 작은 경우에만 필요합니다. 복제 그룹에서 IDs 제거할 샤드(노드 그룹) 목록입니다.

다음 절차는 샤드를 하나 이상 삭제하는 프로세스를 설명합니다.

Example - 샤드 제거

다음 예제에서는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 에서 두 개의 노드 그룹을 제거my-cluster하므로 작업이 완료될 때 총 두 개의 노드 그룹이 있습니다. 제거된 샤드의 키스페이스는 나머지 샤드 간에 균일하게 분배됩니다.

```
https://elasticache.us-east-2.amazonaws.com/
    ?Action=ModifyReplicationGroupShardConfiguration
    &ApplyImmediately=true
    &NodeGroupCount=2
    &ReplicationGroupId=my-cluster
    &NodeGroupsToRemove.member.1=0002
    &NodeGroupsToRemove.member.2=0003
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20171002T192317Z
    &X-Amz-Credential=<credential>
```

온라인 샤드 재분배

AWS Management Console AWS CLI, 또는 를 사용하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 샤드를 리밸런싱할 수 있습니다ElastiCache API.

주제

- 온라인 샤드 재분배(콘솔)
- 온라인 샤드 재분배(AWS CLI)
- 온라인 샤드 재분배(ElastiCache API)

온라인 샤드 재분배(콘솔)

다음 프로세스에서는 를 사용하여 샤드의 균형을 재조정하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 샤드를 재구성하는 방법을 설명합니다 AWS Management Console.

Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 샤드 간에 키스페이스를 리밸런싱하려면

1. 에서 ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 탐색 창에서 Valkey 클러스터 또는 Redis OSS 클러스터를 선택합니다.
3. 리밸런싱하려는 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 이름 왼쪽에 있는 상자가 아닌 이름을 선택합니다.

> **Tip**
>
> 샤드 열에서 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 값은 1 이상입니다.

4. [Rebalance]를 선택합니다.
5. 메시지가 나타나면 Rebalance를 선택합니다. 다음과 유사한 메시지가 표시될 수 있습니다.*Slots in the replication group are uniformly distributed. Nothing to do. (Service: AmazonElastiCache; Status Code: 400; Error Code: InvalidReplicationGroupState; Request ID: 2246cebd-9721-11e7-8d5b-e1b0f086c8cf)*. 취소하는 경우 취소를 선택합니다.

온라인 샤드 재분배(AWS CLI)

`modify-replication-group-shard-configuration`에 다음 파라미터를 사용합니다.

파라미터

- `-apply-immediately` - 필수입니다. 즉시 시작할 샤드 재구성 작업을 지정합니다.
- `--replication-group-id` - 필수입니다. 샤드 재구성 작업을 수행할 복제 그룹(클러스터)을 지정합니다.
- `--node-group-count` - 필수입니다. 클러스터 내 모드 샤드 간에 키스페이스를 재분배하려면 이 값이 현재 샤드 수와 동일해야 합니다.

다음 프로세스에서는 를 사용하여 샤드를 재조정하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 샤드를 재구성하는 방법을 설명합니다 AWS CLI.

Example - 클러스터에서 샤드 재분배

다음 예제에서는 슬롯이 가능한 한 균등하게 분산`my-cluster`되도록 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 슬롯을 재조정합니다. `--node-group-count`(4)의 값은 클러스터 내 현재 샤드 수입니다.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-replication-group-shard-configuration \
    --replication-group-id my-cluster \
```

```
    --node-group-count 4 \
    --apply-immediately
```

Windows의 경우:

```
aws elasticache modify-replication-group-shard-configuration ^
    --replication-group-id my-cluster ^
    --node-group-count 4 ^
    --apply-immediately
```

온라인 샤드 재분배(ElastiCache API)

를 ElastiCache API 사용하여 `ModifyReplicationGroupShardConfiguration` 작업을 사용하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 샤드를 온라인으로 재구성할 수 있습니다.

`ModifyReplicationGroupShardConfiguration`에 다음 파라미터를 사용합니다.

파라미터

- `ApplyImmediately=true` - 필수입니다. 즉시 시작할 샤드 재구성 작업을 지정합니다.
- `ReplicationGroupId` - 필수입니다. 샤드 재구성 작업을 수행할 복제 그룹(클러스터)을 지정합니다.
- `NodeGroupCount` - 필수입니다. 클러스터 내 모드 샤드 간에 키스페이스를 재분배하려면 이 값이 현재 샤드 수와 동일해야 합니다.

다음 프로세스에서는 를 사용하여 샤드의 균형을 재조정하여 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 샤드를 재구성하는 방법을 설명합니다 ElastiCache API.

Example - 클러스터 재분배

다음 예제에서는 슬롯이 가능한 한 균등하게 분산`my-cluster`되도록 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터의 슬롯을 재조정합니다. `NodeGroupCount`(4)의 값은 클러스터 내 현재 샤드 수입니다.

```
https://elasticache.us-east-2.amazonaws.com/
    ?Action=ModifyReplicationGroupShardConfiguration
    &ApplyImmediately=true
    &NodeGroupCount=4
    &ReplicationGroupId=my-cluster
```

```
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20171002T192317Z
    &X-Amz-Credential=<credential>
```

노드 유형 수정하여 온라인 수직 조정

Valkey 버전 7.2 이상 또는 Redis OSS 버전 3.2.10 이상에서 온라인 수직 조정을 사용하면 가동 중지 시간을 최소화하면서 Valkey 또는 Redis OSS 클러스터를 동적으로 확장할 수 있습니다. 이를 통해 Valkey 또는 Redis OSS 클러스터는 조정 중에도 요청을 제공할 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 의 데이터 계층화 ElastiCache 단원을 참조하십시오.

다음을 수행할 수 있습니다.

- 스케일 업 - 더 큰 노드 유형을 사용하도록 Valkey 또는 Redis OSS 클러스터의 노드 유형을 조정하여 읽기 및 쓰기 용량을 늘립니다.

  ElastiCache 는 온라인 상태를 유지하고 요청을 처리하는 동안 클러스터의 크기를 동적으로 조정합니다.
- 축소 - 더 작은 노드를 사용하도록 노드 유형을 조정하여 읽기 및 쓰기 용량을 줄입니다. 다시 말하지만, 는 온라인 상태를 유지하고 요청을 처리하는 동안 클러스터의 크기를 ElastiCache 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.
>
> - ENI (탄력적 네트워크 인터페이스) 용량이 충분한지 확인합니다. 축소된 경우, 작은 노드에 예상 트래픽을 흡수할 수 있는 충분한 메모리가 있어야 합니다.
>
>   메모리 관리 모범 사례에 대해서는 Valkey 및 Redis용 예약 메모리 관리 OSS를 참조합니다.

- 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.

목차

온라인 확장

주제

Valkey 또는 Redis OSS 캐시 클러스터 확장(콘솔)

다음 절차에서는 ElastiCache 관리 콘솔을 사용하여 Valkey 또는 Redis OSS 클러스터를 확장하는 방법을 설명합니다. 이 프로세스 중에 클러스터는 가동 중지 시간을 최소화하면서 요청을 계속 처리합니다.

Valkey 또는 Redis OSS 클러스터 확장(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 Valkey 클러스터 또는 Redis OSS 클러스터를 선택합니다.

3. 클러스터 목록에서 클러스터를 선택합니다.
4. 수정을 선택합니다.
5. [Modify Cluster] 마법사에서 다음을 수행합니다.
   - [Node type] 목록에서 조정할 노드 유형을 선택합니다. 확장하려면, 기존 노드보다 큰 노드 유형을 선택합니다.
6. 확장 프로세스를 즉시 수행하려면 즉시 적용 상자를 선택합니다. [Apply immediately] 상자를 선택하지 않으면 이 클러스터의 다음 유지 관리 기간 중 스케일 업 프로세스가 수행됩니다.
7. 수정을 선택합니다.

   이전 단계에서 [Apply immediately]를 선택한 경우 클러스터의 상태가 수정 중으로 변경됩니다. 상태가 사용 가능으로 변경되면 수정이 완료되고 새 클러스터의 사용을 시작할 수 있습니다.

Valkey 또는 Redis OSS 캐시 클러스터 확장(AWS CLI)

다음 절차에서는 를 사용하여 Valkey 또는 Redis OSS 캐시 클러스터를 확장하는 방법을 설명합니다 AWS CLI. 이 프로세스 중에 클러스터는 가동 중지 시간을 최소화하면서 요청을 계속 처리합니다.

Valkey 또는 Redis OSS 캐시 클러스터 확장(AWS CLI)

1. 다음 파라미터로 `list-allowed-node-type-modifications` 명령을 실행 AWS CLI 하여 확장할 수 있는 노드 유형을 결정합니다.

   Linux, macOS, Unix의 경우:

   ```
   aws elasticache list-allowed-node-type-modifications \
        --replication-group-id my-replication-group-id
   ```

   Windows의 경우:

   ```
   aws elasticache list-allowed-node-type-modifications ^
        --replication-group-id my-replication-group-id
   ```

   위 명령의 출력은 다음과 같습니다(JSON 형식).

   ```
   {
        "ScaleUpModifications": [
            "cache.m3.2xlarge",
            "cache.m3.large",
   ```

```
          "cache.m3.xlarge",
          "cache.m4.10xlarge",
          "cache.m4.2xlarge",
          "cache.m4.4xlarge",
          "cache.m4.large",
          "cache.m4.xlarge",
          "cache.r3.2xlarge",
          "cache.r3.4xlarge",
          "cache.r3.8xlarge",
          "cache.r3.large",
          "cache.r3.xlarge"
      ]
         "ScaleDownModifications": [
          "cache.t2.micro",
          "cache.t2.small ",
          "cache.t2.medium",
          "cache.t1.small "
      ],
}
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.list-allowed-node-type-수정 AWS CLI 참조 의
.

2. 명령과 다음 파라미터를 사용하여 AWS CLI `modify-replication-group` 더 큰 새 노드 유형으로 확장하도록 복제 그룹을 수정합니다.

   - `--replication-group-id` - 확장하는 복제 그룹의 이름입니다.
   - `--cache-node-type` - 캐시 클러스터를 조정할 새 노드 유형입니다. 이 값은 1단계의 `list-allowed-node-type-modifications` 명령에 의해 반환되는 노드 유형 중 하나여야 합니다.
   - `--cache-parameter-group-name` - [선택 사항] `reserved-memory`를 사용하여 클러스터의 예약된 메모리를 관리할 경우 이 파라미터를 사용합니다. 새 노드 유형에 대해 올바른 메모리 양을 예약하는 사용자 지정 캐시 파라미터 그룹을 지정합니다. `reserved-memory-percent`를 사용할 경우 이 파라미터를 생략할 수 있습니다.
   - `--apply-immediately` - 스케일 업 프로세스가 즉시 적용되도록 합니다. 스케일 업 프로세스를 클러스터의 다음 유지 관리 기간으로 연기하려면 `--no-apply-immediately` 파라미터를 사용하세요.

   Linux, macOS, Unix의 경우:

```
aws elasticache modify-replication-group  \
     --replication-group-id my-redis-cluster \
     --cache-node-type cache.m3.xlarge \
     --apply-immediately
```

Windows의 경우:

```
aws elasticache modify-replication-group ^
     --replication-group-id my-redis-cluster ^
     --cache-node-type cache.m3.xlarge ^
     --apply-immediately
```

위 명령의 출력은 다음과 같습니다(JSON 형식).

```
{
  "ReplicationGroup": {
        "Status": "modifying",
        "Description": "my-redis-cluster",
        "NodeGroups": [
            {
                "Status": "modifying",
                "Slots": "0-16383",
                "NodeGroupId": "0001",
                "NodeGroupMembers": [
                    {
                        "PreferredAvailabilityZone": "us-east-1f",
                        "CacheNodeId": "0001",
                        "CacheClusterId": "my-redis-cluster-0001-001"
                    },
                    {
                        "PreferredAvailabilityZone": "us-east-1d",
                        "CacheNodeId": "0001",
                        "CacheClusterId": "my-redis-cluster-0001-002"
                    }
                ]
            }
        ],
        "ConfigurationEndpoint": {
            "Port": 6379,
```

```
            "Address": "my-redis-
cluster.r7gdfi.clustercfg.use1.cache.amazonaws.com"
        },
        "ClusterEnabled": true,
        "ReplicationGroupId": "my-redis-cluster",
        "SnapshotRetentionLimit": 1,
        "AutomaticFailover": "enabled",
        "SnapshotWindow": "07:30-08:30",
        "MemberClusters": [
            "my-redis-cluster-0001-001",
            "my-redis-cluster-0001-002"
        ],
        "CacheNodeType": "cache.m3.xlarge",
         "DataTiering": "disabled"
        "PendingModifiedValues": {}
    }
}
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.modify-replication-group AWS CLI 참조 의 .

3. 를 사용한 경우 다음 파라미터와 함께 명령을 사용하여 AWS CLI `describe-cache-clusters` 캐시 클러스터의 상태를 `--apply-immediately`확인합니다. 상태가 사용 가능으로 변경되면 새롭고 더 큰 캐시 클러스터 노드를 사용할 수 있습니다.

Valkey 또는 Redis OSS 캐시 클러스터 확장(ElastiCache API)

다음 프로세스는 를 사용하여 캐시 클러스터를 현재 노드 유형에서 더 큰 새 노드 유형으로 확장합니다 ElastiCache API. 이 프로세스 중에 는 새 노드를 가리키도록 DNS 항목을 ElastiCache 업데이트합니다. 따라서 애플리케이션의 엔드포인트를 업데이트할 필요가 없습니다. Valkey 7.2 이상 Redis OSS 5.0.5 이상의 경우 클러스터가 계속 온라인 상태를 유지하고 수신 요청을 처리하는 동안 자동 장애 조치 활성화 클러스터를 확장할 수 있습니다. 버전 Redis OSS 4.0.10 이하에서는 DNS 항목이 업데이트되는 동안 기본 노드에서 이전 버전의 읽기 및 쓰기가 잠시 중단될 수 있습니다.

대형 노드 유형으로 스케일 업하는 데 걸리는 시간은 노드 유형 및 현재 캐시 클러스터에 있는 데이터의 양에 따라 달라집니다.

Valkey 또는 Redis OSS 캐시 클러스터 확장(ElastiCache API)

1. 다음 파라미터를 사용하여 `ListAllowedNodeTypeModifications` 작업을 사용하여 ElastiCache API 확장할 수 있는 노드 유형을 결정합니다.

- `ReplicationGroupId` - 복제 그룹의 이름입니다. 모든 복제 그룹 대신 특정 복제 그룹을 설명하려면 이 파라미터를 사용하세요.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ListAllowedNodeTypeModifications
    &ReplicationGroupId=MyReplGroup
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.ListAllowedNodeTypeModifications Amazon ElastiCache API 참조 의 .

2. `ModifyReplicationGroup` ElastiCache API 작업을 사용하고 다음 파라미터를 사용하여 현재 복제 그룹을 새 노드 유형으로 확장합니다.

- `ReplicationGroupId` - 복제 그룹의 이름입니다.
- `CacheNodeType` - 이 복제 그룹에 있는 캐시 클러스터의 새로운 대형 노드 유형입니다. 이 값은 이전 단계의 `ListAllowedNodeTypeModifications` 작업에서 반환된 인스턴스 유형 중 하나여야 합니다.
- `CacheParameterGroupName` - [선택 사항] `reserved-memory`를 사용하여 클러스터의 예약된 메모리를 관리할 경우 이 파라미터를 사용합니다. 새 노드 유형에 대해 올바른 메모리 양을 예약하는 사용자 지정 캐시 파라미터 그룹을 지정합니다. `reserved-memory-percent`를 사용할 경우 이 파라미터를 생략할 수 있습니다.
- `ApplyImmediately` - 스케일 업 프로세스가 즉시 적용되도록 하려면 `true`로 설정합니다. 스케일 업 프로세스를 다음 유지 관리 기간으로 연기하려면 `ApplyImmediately=false`를 사용하세요.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyReplicationGroup
    &ApplyImmediately=true
    &CacheNodeType=cache.m3.2xlarge
    &CacheParameterGroupName=redis32-m3-2xl
    &ReplicationGroupId=myReplGroup
    &SignatureVersion=4
```

```
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &Version=2014-12-01
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.ModifyReplicationGroup Amazon ElastiCache API 참조 의 .

3. 를 사용한 경우 다음 파라미터로 DescribeReplicationGroups 작업을 사용하여 ElastiCache API 복제 그룹의 상태를 ApplyImmediately=true모니터링합니다. 상태가 수정 중에서 사용 가능으로 변경되면 스케일 업된 새 복제 그룹에 쓰기를 시작할 수 있습니다.

   - ReplicationGroupId - 복제 그룹의 이름입니다. 모든 복제 그룹 대신 특정 복제 그룹을 설명하려면 이 파라미터를 사용하세요.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeReplicationGroups
    &ReplicationGroupId=MyReplGroup
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.DescribeReplicationGroups Amazon ElastiCache API 참조 의 .

온라인 축소

주제

- Valkey 또는 Redis OSS 캐시 클러스터 축소(콘솔)
- Valkey 또는 Redis OSS 캐시 클러스터 축소(AWS CLI)
- Valkey 또는 Redis OSS 캐시 클러스터 축소(ElastiCache API)

Valkey 또는 Redis OSS 캐시 클러스터 축소(콘솔)

다음 절차에서는 ElastiCache 관리 콘솔을 사용하여 Valkey 또는 Redis OSS 클러스터를 축소하는 방법을 설명합니다. 이 프로세스 중에 Valkey 또는 Redis OSS 클러스터는 가동 중지 시간을 최소화하면서 요청을 계속 처리합니다.

Valkey 또는 Redis OSS 클러스터 축소(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 Valkey 클러스터 또는 Redis OSS 클러스터를 선택합니다.
3. 클러스터 목록에서 원하는 클러스터를 선택합니다.
4. 수정을 선택합니다.
5. [Modify Cluster] 마법사에서 다음을 수행합니다.
   - [Node type] 목록에서 조정할 노드 유형을 선택합니다. 축소하려면, 기존 노드보다 작은 노드 유형을 선택합니다. 모든 노드 유형을 축소할 수 있는 것은 아닙니다.
6. 축소 프로세스를 즉시 수행하려면 즉시 적용 상자를 선택합니다. 즉시 적용 상자를 선택하지 않으면 이 클러스터의 다음 유지 관리 기간 중 축소 프로세스가 수행됩니다.
7. 수정을 선택합니다.

   이전 단계에서 [Apply immediately]를 선택한 경우 클러스터의 상태가 수정 중으로 변경됩니다. 상태가 사용 가능으로 변경되면 수정이 완료되고 새 클러스터의 사용을 시작할 수 있습니다.

Valkey 또는 Redis OSS 캐시 클러스터 축소(AWS CLI)

다음 절차에서는 를 사용하여 Valkey 또는 Redis OSS 캐시 클러스터를 축소하는 방법을 설명합니다 AWS CLI. 이 프로세스 중에 클러스터는 가동 중지 시간을 최소화하면서 요청을 계속 처리합니다.

Valkey 또는 Redis OSS 캐시 클러스터를 축소하려면(AWS CLI)

1. 다음 파라미터로 `list-allowed-node-type-modifications` 명령을 실행 AWS CLI 하여 확장할 수 있는 노드 유형을 결정합니다.

   Linux, macOS, Unix의 경우:

```
aws elasticache list-allowed-node-type-modifications \
     --replication-group-id my-replication-group-id
```

Windows의 경우:

```
aws elasticache list-allowed-node-type-modifications ^
     --replication-group-id my-replication-group-id
```

위 명령의 출력은 다음과 같습니다(JSON 형식).

```
{
     "ScaleUpModifications": [
         "cache.m3.2xlarge",
         "cache.m3.large",
         "cache.m3.xlarge",
         "cache.m4.10xlarge",
         "cache.m4.2xlarge",
         "cache.m4.4xlarge",
         "cache.m4.large",
         "cache.m4.xlarge",
         "cache.r3.2xlarge",
         "cache.r3.4xlarge",
         "cache.r3.8xlarge",
         "cache.r3.large",
         "cache.r3.xlarge"
     ]

        "ScaleDownModifications": [
         "cache.t2.micro",
         "cache.t2.small ",
         "cache.t2.medium ",
         "cache.t1.small"
     ]
}
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.list-allowed-node-type-수정 AWS CLI 참조 의
.

2. 명령과 다음 파라미터를 사용하여 AWS CLI `modify-replication-group` 복제 그룹을 수정하여 더 작은 새 노드 유형으로 축소합니다.

   - `--replication-group-id` - 축소하는 복제 그룹의 이름입니다.

- `--cache-node-type` - 캐시 클러스터를 조정할 새 노드 유형입니다. 이 값은 1단계의 `list-allowed-node-type-modifications` 명령에 의해 반환되는 노드 유형 중 하나여야 합니다.
- `--cache-parameter-group-name` - [선택 사항] `reserved-memory`를 사용하여 클러스터의 예약된 메모리를 관리할 경우 이 파라미터를 사용합니다. 새 노드 유형에 대해 올바른 메모리 양을 예약하는 사용자 지정 캐시 파라미터 그룹을 지정합니다. `reserved-memory-percent`를 사용할 경우 이 파라미터를 생략할 수 있습니다.
- `--apply-immediately` - 스케일 업 프로세스가 즉시 적용되도록 합니다. 축소 프로세스를 클러스터의 다음 유지 관리 기간으로 연기하려면 `--no-apply-immediately` 파라미터를 사용하세요.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-replication-group  \
     --replication-group-id my-redis-cluster \
     --cache-node-type cache.t2.micro \
     --apply-immediately
```

Windows의 경우:

```
aws elasticache modify-replication-group ^
     --replication-group-id my-redis-cluster ^
     --cache-node-type cache.t2.micro ^
     --apply-immediately
```

위 명령의 출력은 다음과 같습니다(JSON 형식).

```
{
  "ReplicationGroup": {
        "Status": "modifying",
        "Description": "my-redis-cluster",
        "NodeGroups": [
            {
                "Status": "modifying",
                "Slots": "0-16383",
                "NodeGroupId": "0001",
                "NodeGroupMembers": [
                    {
```

```
                        "PreferredAvailabilityZone": "us-east-1f",
                        "CacheNodeId": "0001",
                        "CacheClusterId": "my-redis-cluster-0001-001"
                    },
                    {
                        "PreferredAvailabilityZone": "us-east-1d",
                        "CacheNodeId": "0001",
                        "CacheClusterId": "my-redis-cluster-0001-002"
                    }
                ]
            }
        ],
        "ConfigurationEndpoint": {
            "Port": 6379,
            "Address": "my-redis-
cluster.r7gdfi.clustercfg.use1.cache.amazonaws.com"
        },
        "ClusterEnabled": true,
        "ReplicationGroupId": "my-redis-cluster",
        "SnapshotRetentionLimit": 1,
        "AutomaticFailover": "enabled",
        "SnapshotWindow": "07:30-08:30",
        "MemberClusters": [
            "my-redis-cluster-0001-001",
            "my-redis-cluster-0001-002"
        ],
        "CacheNodeType": "cache.t2.micro",
         "DataTiering": "disabled"
        "PendingModifiedValues": {}
    }
}
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.modify-replication-group AWS CLI 참조 의 .

3. 를 사용한 경우 다음 파라미터와 함께 명령을 사용하여 AWS CLI `describe-cache-clusters` 캐시 클러스터의 상태를 `--apply-immediately`확인합니다. 상태가 사용 가능으로 변경되면 새롭고 더 작은 캐시 클러스터 노드를 사용할 수 있습니다.

Valkey 또는 Redis OSS 캐시 클러스터 축소(ElastiCache API)

다음 프로세스는 를 사용하여 복제 그룹을 현재 노드 유형에서 더 작은 새 노드 유형으로 확장합니다 ElastiCache API. 이 프로세스 중에 Valkey 또는 Redis OSS 클러스터는 가동 중지 시간을 최소화하면서 요청을 계속 처리합니다.

더 작은 노드 유형으로 축소하는 데 걸리는 시간은 노드 유형 및 현재 캐시 클러스터에 있는 데이터의 양에 따라 달라집니다.

축소(ElastiCache API)

1. 다음 파라미터를 사용하여 `ListAllowedNodeTypeModifications` 작업을 사용하여 ElastiCache API 축소할 수 있는 노드 유형을 결정합니다.

   - `ReplicationGroupId` - 복제 그룹의 이름입니다. 모든 복제 그룹 대신 특정 복제 그룹을 설명하려면 이 파라미터를 사용하세요.

   ```
   https://elasticache.us-west-2.amazonaws.com/
       ?Action=ListAllowedNodeTypeModifications
       &ReplicationGroupId=MyReplGroup
       &Version=2015-02-02
       &SignatureVersion=4
       &SignatureMethod=HmacSHA256
       &Timestamp=20150202T192317Z
       &X-Amz-Credential=<credential>
   ```

   자세한 내용을 알아보려면 다음 섹션을 참조하세요.ListAllowedNodeTypeModifications Amazon ElastiCache API 참조 의 .

2. `ModifyReplicationGroup` ElastiCache API 작업을 사용하고 다음 파라미터를 사용하여 현재 복제 그룹을 새 노드 유형으로 축소합니다.

   - `ReplicationGroupId` - 복제 그룹의 이름입니다.
   - `CacheNodeType` - 이 복제 그룹에 있는 캐시 클러스터의 새롭고 더 작은 노드 유형입니다. 이 값은 이전 단계의 `ListAllowedNodeTypeModifications` 작업에서 반환된 인스턴스 유형 중 하나여야 합니다.
   - `CacheParameterGroupName` - [선택 사항] `reserved-memory`를 사용하여 클러스터의 예약된 메모리를 관리할 경우 이 파라미터를 사용합니다. 새 노드 유형에 대해 올바른 메모리 양을

예약하는 사용자 지정 캐시 파라미터 그룹을 지정합니다. `reserved-memory-percent`를 사용할 경우 이 파라미터를 생략할 수 있습니다.

- `ApplyImmediately` - 축소 프로세스가 즉시 적용되도록 하려면 `true`로 설정합니다. 축소 프로세스를 다음 유지 관리 기간으로 연기하려면 `ApplyImmediately=false`를 사용하세요.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyReplicationGroup
    &ApplyImmediately=true
    &CacheNodeType=cache.t2.micro
    &CacheParameterGroupName=redis32-m3-2xl
    &ReplicationGroupId=myReplGroup
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &Version=2014-12-01
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.ModifyReplicationGroup Amazon ElastiCache API 참조 의 .

Valkey 및 RedisJSON용 시작하기 OSS

ElastiCache 는 Valkey 및 Redis OSS 클러스터 내에서 복잡한 데이터 세트를 인코딩하는 간단하고 스키마 없는 방법인 네이티브 JavaScript 객체 표기법(JSON) 형식을 지원합니다. 클러스터 내에 JavaScript 객체 표기법(JSON) 형식을 사용하여 데이터를 기본적으로 저장하고 액세스할 수 있으며, 직렬화 및 역직렬화하기 위해 사용자 지정 코드를 관리할 필요 없이 해당 클러스터에 저장된 JSON 데이터를 업데이트할 수 있습니다.

에서 작동하는 애플리케이션에 Valkey 및 Redis OSS API 작업을 사용하는 것 외에도 이제 전체 객체를 조작할 필요 없이 JSON 문서의 특정 부분을 효율적으로 검색하고 업데이트할 JSON수 있습니다. 이렇게 하면 성능을 개선하고 비용을 절감할 수 있습니다. Goessner 스타일 `JSONPath` 쿼리를 사용하여 JSON 문서 콘텐츠를 검색할 수도 있습니다.

지원되는 엔진 버전으로 클러스터를 생성하면 JSON 데이터 유형 및 관련 명령을 자동으로 사용할 수 있습니다. API JSON 모듈의 버전 2와 RDB 호환되므로 기존 JSON기반 Valkey 및 Redis OSS 애플리케이션을 로 쉽게 마이그레이션할 수 있습니다 ElastiCache. 지원되는 명령에 대한 자세한 내용은 섹션을 참조하세요지원되는 Valkey 및 Redis OSS 명령.

JSON관련 지표 `JsonBasedCmds` 및 는 CloudWatch 에 `JsonBasedCmdsLatency` 통합되어 이 데이터 유형의 사용량을 모니터링합니다. 자세한 내용은 Valkey 및 Redis 에 대한 지표를 OSS참조하세요.

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

주제

JSON 데이터 유형 개요

ElastiCache 는 JSON 데이터 유형 작업을 위한 여러 Valkey 및 Redis OSS 명령을 지원합니다. 다음은 지원되는 JSON 데이터 형식의 개요와 명령의 세부 목록입니다.

용어

용어	설명
JSON 문서	JSON 키의 값을 나타냅니다.
JSON 값	전체 JSON 문서를 나타내는 루트를 포함하여 문서의 하위 집합을 나타냅니다. 값은 컨테이너 또는 컨테이너 내의 항목일 수 있습니다.
JSON 요소	JSON 값과 동일합니다.

지원되는 JSON 표준

JSON 형식은 RFC 7159 및 ECMA-404 JSON 데이터 교환 표준을 준수합니다. UTF-8 JSON 텍스트의 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 형식일 수 있습니다.

문서 크기 제한

JSON 문서는 빠른 액세스 및 수정에 최적화된 형식으로 내부적으로 저장됩니다. 이 형식은 일반적으로 동일한 문서의 동등하게 직렬화된 표현보다 어느 정도 더 많은 메모리를 소비하게 됩니다.

단일 JSON 문서의 메모리 소비량은 JSON 문자열이 아닌 인 메모리 데이터 구조의 크기인 64MB로 제한됩니다. `JSON.DEBUG MEMORY` 명령을 사용하여 JSON 문서에서 사용하는 메모리의 양을 확인할 수 있습니다.

JSON ACLs

- 기존 데이터 유형별 범주(@string, @hash 등)와 마찬가지로 JSON 명령 및 데이터에 대한 액세스 관리를 간소화하기 위해 새 범주 @json이 추가되었습니다. 다른 기존 Valkey 또는 Redis OSS 명령은 @json 범주의 구성원이 아닙니다. 모든 JSON 명령은 모든 키스페이스 또는 명령 제한 및 권한을 적용합니다.
- 새 JSON 명령을 포함하도록 업데이트된 기존 Valkey 및 Redis OSS ACL 범주는 @read, @write, @fast, @slow 및 @admin의 다섯 가지입니다. 다음 표는 적절한 범주에 대한 JSON 명령 매핑을 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 다른 JSON 객체 또는 배열 자체인 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 키 이름이 필요합니다. 일부 명령에는 path 인수도 있습니다. path 인수가 선택 사항이며 제공되지 않는 경우 기본적으로 루트로 설정됩니다.

표기법

- 필수 인수는 각괄호로 묶습니다. 예: <key>
- 선택적 인수는 대괄호로 묶습니다. 예: [path]
- 추가 선택적 인수는 줄임표('...')로 표시됩니다. 예: [json...]

경로 구문

Redis는 두 가지 종류의 경로 구문을 JSON 지원합니다.

- 향상된 구문 - 다음 표와 같이 Goessner 에서 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

다음 예제는 Goessner의 예제 XML 데이터를 기반으로 하며, 추가 필드를 추가하여 수정했습니다.

```
{ "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 멤버입니다.

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 숫자가 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.

예제

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.

접두사	설명
ERR	일반 오류.
LIMIT	크기 제한을 초과한 경우 발생하는 오류입니다. 예: 문서 크기 제한 또는 중첩 깊이 제한이 초과되었습니다.
NONEXISTENT	키 또는 경로가 존재하지 않습니다.

접두사	설명
OUTOFBOUNDARIES	배열 인덱스가 범위를 벗어났습니다.
SYNTAXERR	구문 오류.
WRONGTYPE	잘못된 값 유형.

JSON관련 지표

다음 JSON 정보 지표가 제공됩니다.

정보	설명
json_total_memory_bytes	JSON 객체에 할당된 총 메모리입니다.
json_num_documents	Valkey 또는 Redis의 총 문서 수입니다OSS.

핵심 지표를 쿼리하려면 다음 명령을 실행합니다.

```
info json_core_metrics
```

ElastiCache 와 Valkey 및 Redis의 OSS 상호 작용 방식 JSON

다음 섹션에서는 Valkey 및 Redis ElastiCache 가 JSON 데이터 유형과 OSS 상호 작용하는 방법을 설명합니다.

연산자 우선순위

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

최대 경로 중첩 제한 동작

ElastiCache (Redis OSS)의 최대 경로 중첩 제한은 128입니다. 따라서 `$.a.b.c.d...`와 같은 값은 128수준까지만 도달할 수 있습니다.

숫자 값 처리

JSON 에는 정수 및 부동 소수점 번호에 대한 별도의 데이터 유형이 없습니다. 모두 숫자라고 부릅니다.

숫자 표현:

입력 시 JSON 숫자가 수신되면 64비트 부호 있는 정수 또는 64비트 IEEE 이중 정밀도 부동 소수점이라는 두 가지 내부 바이너리 표현 중 하나로 변환됩니다. 원래 문자열 및 모든해당 서식은 보관되지 않습니다. 따라서 숫자가 JSON 응답의 일부로 출력되면 내부 바이너리 표현에서 일반 형식 지정 규칙을 사용하는 인쇄 가능한 문자열로 변환됩니다. 이러한 규칙은 수신된 문자열과 다른 문자열이 생성될 수 있습니다.

산술 명령어 `NUMINCRBY`과 `NUMMULTBY`:

- 두 숫자가 정수이고 결과가 범위를 벗어나는 경우 `int64`자동으로 64비트 IEEE 이중 정밀도 부동 소수점 번호가 됩니다.
- 숫자 중 하나 이상이 부동 소수점인 경우 결과는 64비트 IEEE 이중 정밀도 부동 소수점 번호입니다.
- 결과가 64비트 IEEE 이중 범위를 초과하는 경우 명령은 `OVERFLOW` 오류를 반환합니다.

사용할 수 있는 명령의 자세한 목록은 지원되는 Valkey 및 Redis OSS 명령 섹션을 참조하세요.

직접 배열 필터링

ElastiCache Valkey 또는 Redis를 사용하면 배열 객체를 직접 OSS 필터링할 수 있습니다.

와 같은 데이터 `[0,1,2,3,4,5,6]` 및 와 같은 경로 쿼리 `$[?(@<4)]`또는 와 같은 데이터 `{"my_key":[0,1,2,3,4,5,6]}` 및 와 같은 경로 쿼리의 경우 Valkey 또는 Redis를 `$.my_key[?(@<4)]` ElastiCache 사용하면 두 경우 모두 [1,2,3]을 반환OSS합니다.

배열 인덱싱 동작

ElastiCache Valkey 또는 Redis를 사용하면 배열에 대한 양수 인덱스와 음수 인덱스를 모두 OSS 사용할 수 있습니다. 길이가 5인 배열의 경우 0이 첫 번째 요소, 1이 두 번째 요소를 쿼리하는 식입니다. 음수는 배열의 끝에서 시작하므로 -1은 다섯 번째 요소를 쿼리하고, -2는 네 번째 요소를 쿼리하는 식입니다.

고객에 대해 예측 가능한 동작을 보장하기 위해 Valkey 또는 RedisOSS를 ElastiCache 사용하면 배열 인덱스가 아래로 또는 위로 라운딩되지 않으므로 길이가 5인 배열이 있는 경우 인덱스 5 이상 또는 -6 이하를 호출해도 결과가 생성되지 않습니다.

엄격한 구문 평가

MemoryDB는 경로의 하위 집합에 유효한 JSON 경로가 포함되어 있더라도 유효하지 않은 구문이 있는 경로를 허용하지 않습니다. 이는 고객에게 올바른 행동을 유지하는 것과 같습니다.

지원되는 Valkey 및 Redis OSS 명령

ElastiCache 는 다음 Valkey 및 Redis OSS JSON 명령을 지원합니다.

주제

- JSON.ARRAPPEND
- JSON.ARRINDEX
- JSON.ARRINSERT
- JSON.ARRLEN
- JSON.ARRPOP
- JSON.ARRTRIM
- JSON.CLEAR
- JSON.DEBUG
- JSON.DEL
- JSON.FORGET
- JSON.GET
- JSON.MGET
- JSON.NUMINCRBY
- JSON.NUMMULTBY
- JSON.OBJLEN
- JSON.OBJKEYS
- JSON.RESP
- JSON.SET
- JSON.STRAPPEND
- JSON.STRLEN
- JSON.TOGGLE

- JSON.TYPE

JSON.ARRAPPEND

하나 이상의 값을 경로의 배열 값에 추가합니다.

구문

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

- 키(필수) - JSON 문서 유형의 Valkey 또는 Redis OSS 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.
- SYNTAXERR 입력 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.
- SYNTAXERR 입력 json 인수 중 하나가 유효한 JSON 문자열이 아닌 경우 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 스칼라 JSON 값의 첫 번째 발생을 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 Valkey 또는 Redis OSS 키입니다.
- 경로(필수) - JSON 경로입니다.
- json-scalar(필수) - 검색 대상인 스칼라 값입니다. JSON 스칼라는 객체 또는 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 Valkey 또는 Redis OSS 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 Valkey 또는 Redis OSS 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 Valkey 또는 Redis OSS 키입니다.
- 경로(선택 사항) - JSON 경로입니다. 제공하지 않으면 기본적으로 루트로 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

- 팝업된 JSON 값을 나타내는 대량 문자열입니다.
- 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 Valkey 또는 Redis OSS 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 Valkey 또는 Redis OSS 키입니다.
- 경로(선택 사항) - JSON 경로입니다. 제공하지 않으면 기본적으로 root로 설정됩니다.

반환

- 정수, 삭제된 컨테이너의 수입니다.
- 삭제된 하나의 컨테이너에 대해 빈 배열 또는 객체 계정을 삭제합니다.
- 컨테이너가 아닌 값을 삭제하면 0이 반환됩니다.

예

```
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) 7
127.0.0.1:6379> JSON.CLEAR k1  $[*]
(integer) 4
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

보고서 정보 지원되는 하위 명령은 다음과 같습니다.

- MEMORY <key> [path] – 메모리 사용량을 JSON 값의 바이트 단위로 보고합니다. 경로는 제공되지 않으면 기본적으로 root로 설정됩니다.
- FIELDS <key> [path] – 지정된 문서 경로의 필드 수를 보고합니다. 경로는 제공되지 않으면 기본적으로 root로 설정됩니다. 컨테이너가 아닌 각 JSON 값은 하나의 필드로 계산됩니다. 객체 및 배열은 포함된 각 JSON 값에 대해 하나의 필드를 반복적으로 계산합니다. 루트 컨테이너를 제외한 각 컨테이너 값은 하나의 추가 필드로 계산됩니다.
- HELP – 명령의 도움말 메시지를 인쇄합니다.

구문

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

다음의 하위 명령에 따라 다릅니다.

MEMORY

- 경로가 향상된 구문인 경우
  - 각 경로에서 JSON 값의 메모리 크기(바이트)를 나타내는 정수 배열을 반환합니다.
  - Valkey 또는 Redis OSS 키가 없는 경우 빈 배열을 반환합니다.
- 경로가 제한된 구문인 경우
  - 정수, 메모리 크기 및 JSON 값을 바이트 단위로 반환합니다.
  - Valkey 또는 Redis OSS 키가 없는 경우 null을 반환합니다.

FIELDS

- 경로가 향상된 구문인 경우
  - 각 경로의 JSON 값 필드 수를 나타내는 정수 배열을 반환합니다.
  - Valkey 또는 Redis OSS 키가 없는 경우 빈 배열을 반환합니다.
- 경로가 제한된 구문인 경우
  - JSON 값의 필드 수인 정수를 반환합니다.

- Valkey 또는 Redis OSS 키가 없는 경우 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 경로에서 JSON 값을 삭제합니다. 경로가 루트인 경우 Valkey 또는 Redis 에서 키를 삭제하는 것과 같습니다OSS.

구문

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

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

반환

- 삭제된 요소 수입니다.
- Valkey 또는 Redis OSS 키가 없는 경우 0입니다.
- JSON 경로가 유효하지 않거나 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 이상의 경로JSON에서 직렬화된 를 반환합니다.

구문

```
JSON.GET <key>
[INDENT indentation-string]
[NEWLINE newline-string]
[SPACE space-string]
```

```
[NOESCAPE]
[path ...]
```

- 키(필수) - JSON 문서 유형의 Valkey 또는 Redis OSS 키입니다.
- INDENT/NEWLINE/SPACE(선택 사항) - 반환된 JSON 문자열의 형식, 즉 '정밀 인쇄'를 제어합니다. 각 문자열의 기본값은 빈 문자열이며, 모든 조합으로 재정의될 수 있습니다. 임의의 순서로 지정할 수 있습니다.
- NOESCAPE - 선택 사항으로, 레거시 호환성을 위해 존재할 수 있으며 다른 효과는 없습니다.
- 경로(선택 사항) - JSON 경로가 0개 이상이고 아무 것도 지정되지 않으면 루트가 기본값으로 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 경로JSONs에서 직렬화됩니다. 존재하지 않는 키 또는 JSON 경로에 대해 null을 반환합니다.

구문

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

- 키(필수) - 문서 유형의 하나 이상의 Valkey 또는 Redis OSS 키입니다.
- 경로(필수) - JSON 경로입니다.

반환

- 대량 문자열 배열 배열의 크기는 명령의 키 수와 같습니다. 배열의 각 요소에는 (a) 경로에 JSON 따라 직렬화된 가 있거나 (b) 키가 없거나, 경로가 문서에 없거나, 경로가 유효하지 않은 경우 null(구문 오류)이 채워집니다.
- 지정된 키 중 하나라도 존재하고 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 Valkey 또는 Redis OSS 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.
- OVERFLOW 결과가 64비트 IEEE 이중 범위를 벗어나는 경우 오류가 발생합니다.
- 문서 키가 없는 경우 NONEXISTENT입니다.

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.
- OVERFLOW 결과가 64비트 IEEE 이중 범위를 벗어나는 경우 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 Valkey 또는 Redis OSS 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.

- OVERFLOW 결과가 64비트 IEEE 이중 정밀도 부동 소수점 번호의 범위를 벗어나는 경우 오류가 발생합니다.
- 문서 키가 없는 경우 NONEXISTENT입니다.

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.
- OVERFLOW 결과가 64비트 IEEE 이중 범위를 벗어나는 경우 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.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 Valkey 또는 Redis OSS 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 Valkey 또는 Redis OSS 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 직렬화 프로토콜()의 지정된 경로에 JSON 값을 반환합니다RESP. 값이 컨테이너인 경우 응답은 RESP 배열 또는 중첩 배열입니다.

- JSON null은 RESP Null 대량 문자열에 매핑됩니다.
- JSON 부울 값은 해당 RESP Simple Strings에 매핑됩니다.
- 정수 번호는 RESP 정수에 매핑됩니다.
- 64비트 IEEE 이중 부동 소수점 번호는 RESP Bulk Strings에 매핑됩니다.

- JSON 문자열은 RESP 대량 문자열에 매핑됩니다.
- JSON 배열은 RESP 배열로 표시되며, 여기서 첫 번째 요소는 단순 문자열 [이고 배열의 요소는 그 뒤에 있습니다.
- JSON 객체는 RESP 배열로 표시되며, 여기서 첫 번째 요소는 단순 문자열 {, 그 뒤에는 키-값 페어가 있고, 각각은 RESP 대량 문자열입니다.

구문

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

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

반환

경로가 향상된 구문인 경우

- 배열의 배열입니다. 각 배열 요소는 한 경로에서 값의 RESP 형태를 나타냅니다.
- 문서 키가 없는 경우 빈 배열입니다.

경로가 제한된 구문인 경우

- 경로의 값 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

경로에서 JSON 값을 설정합니다.

경로가 객체 멤버를 호출하는 경우

- 상위 요소가 없는 경우 명령은 NONEXISTENT 오류를 반환합니다.
- 상위 요소가 존재하지만 객체가 아닌 경우 명령은 를 반환합니다ERROR.
- 상위 요소가 있고 객체인 경우
  - 멤버가 없으면 상위 객체가 경로의 마지막 하위 객체인 경우에만 새 멤버는 상위 객체에 추가됩니다. 그렇지 않으면 명령이 NONEXISTENT 오류를 반환합니다.
  - 멤버가 있는 경우 해당 값이 JSON 값으로 대체됩니다.

경로가 배열 인덱스를 호출하는 경우

- 상위 요소가 없는 경우 명령은 NONEXISTENT 오류를 반환합니다.
- 상위 요소가 존재하지만 배열이 아닌 경우 명령은 를 반환합니다ERROR.
- 상위 요소가 존재하지만 인덱스가 범위를 벗어나는 경우 명령은 OUTOFBOUNDARIES 오류를 반환합니다.
- 상위 요소가 존재하고 인덱스가 유효한 경우 요소가 새 JSON 값으로 대체됩니다.

경로에 객체 또는 배열이 필요한 경우 값(객체 또는 배열)이 새 JSON 값으로 바뀝니다.

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 Valkey 또는 Redis OSS 키입니다.
- 경로(필수) - JSON 경로입니다. 새 키의 경우 JSON 경로는 루트 “.”여야 합니다.
- NX(선택 사항) - 경로가 루트인 경우 키가 없는 경우에만 값을 설정합니다. 즉, 새 문서를 삽입합니다. 경로가 루트가 아니면 경로가 없는 경우에만 값을 설정합니다. 즉, 문서에 값을 삽입합니다.
- XX(선택 사항) - 경로가 루트인 경우 키가 있는 경우에만 값을 설정합니다. 즉, 기존 문서를 교체합니다. 경로가 루트가 아니면 경로가 있는 경우에만 값을 설정합니다. 즉, 기존 값을 업데이트합니다.

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

경로의 문자열에 JSON 문자열을 추가합니다.

구문

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

- 키(필수) - JSON 문서 유형의 Valkey 또는 Redis OSS 키입니다.
- 경로(선택 사항) - JSON 경로입니다. 제공하지 않으면 기본적으로 root로 설정됩니다.
- json_string(필수) - 문자열의 JSON 표현입니다. JSON 문자열은 따옴표로 표시해야 합니다. 예: '"문자열 예"'.

반환

경로가 향상된 구문인 경우

- 각 경로에서 문자열의 새 길이를 나타내는 정수 배열입니다.
- 경로의 값이 문자열이 아닌 경우 해당 반환 값은 null입니다.
- SYNTAXERR 입력 json 인수가 유효한 JSON 문자열이 아닌 경우의 오류입니다.
- 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.
- WRONGTYPE 입력 json 인수가 유효한 JSON 문자열이 아닌 경우의 오류입니다.
- 경로가 존재하지 않는 경우 NONEXISTENT 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 가져옵니다.

구문

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

- 키(필수) - JSON 문서 유형의 Valkey 또는 Redis OSS 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.
- 경로가 존재하지 않는 경우 NONEXISTENT 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 Valkey 또는 Redis OSS 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 NONEXISTENT입니다.

경로가 제한된 구문인 경우

- 결과 부울 값을 나타내는 문자열('참'/ '거짓')입니다.
- 문서 키가 없는 경우 NONEXISTENT입니다.
- 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 Valkey 또는 Redis OSS 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 유효하지 않거나 존재하지 않는 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
```

ElastiCache 리소스 태그 지정

클러스터 및 기타 ElastiCache 리소스를 관리하는 데 도움이 되도록 태그 형식으로 각 리소스에 자체 메타데이터를 할당할 수 있습니다. 태그를 사용하면 목적, 소유자 또는 환경 등 다양한 방식으로 AWS 리소스를 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 목적 또는 소유자 등 다양한 방식으로 AWS 리소스를 분류할 수 있습니다. 예를 들어 각 인스턴스의 소유자 및 사용자 그룹을 추적하는 데 도움이 되는 계정 ElastiCache 클러스터에 대한 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 에 의미가 없으며 문자열로 엄격하게 해석 ElastiCache 됩니다. 또한 태그는 리소스에 자동으로 배정되지 않습니다. 태그 키와 값을 편집할 수 있으며 언제든지 리소스에서 태그를 제거할 수 있습니다. 태그의 값을 null로 설정할 수 있습니다. 해당 리소스에 대해 키가 기존 태그와 동일한 태그를 추가하는 경우 새 값이 이전 값을 덮어씁니다. 리소스를 삭제하면 리소스 태그도 삭제됩니다. 또한 복제 그룹에서 태그를 추가하거나 삭제하면 해당 복제 그룹의 모든 노드에서 해당 태그가 추가되거나 제거됩니다.

AWS Management Console, 및 를 사용하여 태그를 사용할 수 AWS CLI있습니다 ElastiCache API.

를 사용하는 경우 AWS 계정에서 태그를 생성IAM, 편집 또는 삭제할 권한이 있는 사용자를 제어할 수 있습니다. 자세한 내용은 리소스 수준 권한 단원을 참조하십시오.

태그 지정이 가능한 리소스

계정에 이미 있는 대부분의 ElastiCache 리소스에 태그를 지정할 수 있습니다. 아래의 표에는 태그 지정을 지원하는 리소스가 나와 있습니다. 를 사용하는 경우 태그 편집기 를 사용하여 리소스에 태그를 적용할 AWS Management Console수 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 이름 태그에 따라 리소스를 구성할 수 있지만 이 태그는 ElastiCache 서비스에 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 ElastiCache API, AWS CLI또는 를 사용하는 경우 관련 ElastiCache API 작업의 Tags 파라미터를 사용하여 태그를 적용할 AWS SDK수 있습니다. 스크립트는 다음과 같습니다.

- `CreateServerlessCache`
- `CreateCacheCluster`
- `CreateReplicationGroup`
- `CopyServerlessCacheSnapshot`
- `CopySnapshot`
- `CreateCacheParameterGroup`
- `CreateCacheSecurityGroup`
- `CreateCacheSubnetGroup`
- `CreateServerlessCacheSnapshot`
- `CreateSnapshot`
- `CreateUserGroup`
- `CreateUser`

- `PurchaseReservedCacheNodesOffering`

다음 표에서는 태그를 지정할 수 있는 ElastiCache 리소스와 , ElastiCache API AWS CLI또는 를 사용하여 생성할 때 태그를 지정할 수 있는 리소스를 설명합니다 AWS SDK.

ElastiCache 리소스에 대한 지원 태그 지정

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

태그 지원	생성 시 태그 지정 지원
예	예

> **Note**
>
> 글로벌 데이터 스토어에는 태그를 지정할 수 없습니다.

생성 시 태깅을 지원하는 작업에 IAM 정책에 태그 기반 리소스 수준 권한을 적용하여 생성 시 리소스에 ElastiCache API 태깅할 수 있는 사용자 및 그룹에 대한 세분화된 제어를 구현할 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 단원을 참조하십시오.

결제를 위한 리소스 태그 지정에 대한 자세한 내용은 비용 할당 태그를 사용하여 비용을 모니터링합니다. 섹션을 참조하세요.

캐시 및 스냅샷에 태그 지정

태그 지정에는 요청 작업의 일부로 다음 규칙이 적용됩니다.

- CreateReplicationGroup:
  - `--primary-cluster-id` 및 `--tags` 파라미터가 요청에 포함되어 있으면 요청 태그가 복제 그룹에 추가되고 복제 그룹의 모든 캐시 클러스터에 전파됩니다. 기본 캐시 클러스터에 기존 태그가 있는 경우 모든 노드에서 일관된 태그를 유지하기 위해 이러한 기존 태그를 요청 태그로 덮어씁니다.

    요청 태그가 없는 경우 기본 캐시 클러스터 태그가 복제 그룹에 추가되고 모든 캐시 클러스터에 전파됩니다.
  - `--snapshot-name` 또는 `--serverless-cache-snapshot-name`이 제공된 경우:

    태그가 요청에 포함되어 있으면 복제 그룹에는 해당 태그로만 태그가 지정됩니다. 요청에 태그가 포함되어 있지 않은 경우 복제 그룹에 스냅샷 태그가 추가됩니다.
  - `--global-replication-group-id`이 제공된 경우:

태그가 요청에 포함되어 있으면 요청 태그가 복제 그룹에 추가되고 모든 캐시 클러스터에 전파됩니다.

- CreateCacheCluster :
  - `--replication-group-id`이 제공된 경우:

    태그가 요청에 포함되어 있으면 캐시 클러스터에는 해당 태그로만 태그가 지정됩니다. 요청에 태그가 포함되어 있지 않은 경우 캐시 클러스터는 기본 캐시 클러스터의 태그 대신 복제 그룹 태그를 상속합니다.
  - `--snapshot-name`이 제공된 경우:

    태그가 요청에 포함되어 있으면 캐시 클러스터에는 해당 태그로만 태그가 지정됩니다. 요청에 태그가 포함되어 있지 않은 경우 스냅샷 태그가 캐시 클러스터에 추가됩니다.
- CreateServerlessCache :
  - 태그가 요청에 포함되어 있으면 요청 태그만 서버리스 캐시에 추가됩니다.
- CreateSnapshot :
  - `--replication-group-id`가 제공된 경우:

    태그가 요청에 포함되어 있으면 요청 태그만 스냅샷에 추가됩니다. 요청에 태그가 포함되어 있지 않은 경우 복제 그룹 태그가 스냅샷에 추가됩니다.
  - `--cache-cluster-id`가 제공된 경우:

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

    태그가 복제 그룹 태그에서 전파됩니다.
- CreateServerlessCacheSnapshot :
  - 태그가 요청에 포함되어 있으면 요청 태그만 서버리스 캐시 스냅샷에 추가됩니다.
- CopySnapshot :
  - 태그가 요청에 포함되어 있으면 요청 태그만 스냅샷에 추가됩니다. 요청에 태그가 포함되어 있지 않은 경우 소스 스냅샷 태그가 복사된 스냅샷에 추가됩니다.
- CopyServerlessCacheSnapshot :
  - 태그가 요청에 포함되어 있으면 요청 태그만 서버리스 캐시 스냅샷에 추가됩니다.
- AddTagsToResource 및 RemoveTagsFromResource :

- 태그는 복제 그룹에서 추가/제거되며 작업은 복제 그룹의 모든 클러스터에 전파됩니다.

> **Note**
>
> AddTagsToResource 및 는 기본 파라미터 및 보안 그룹에 사용할 수 RemoveTagsFromResource 없습니다.

- IncreaseReplicaCount 및 ModifyReplicationGroupShardConfiguration:
  - 복제 그룹에 추가된 모든 새 클러스터에는 복제 그룹과 동일한 태그가 적용됩니다.

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 길이 - UTF-8의 유니코드 문자 128자.
- 최대 값 길이 - UTF-8의 유니코드 문자 256자.
- 는 태그에서 모든 문자를 ElastiCache 허용하지만 다른 서비스는 제한적일 수 있습니다. 서비스 전체에서 허용되는 문자는 문자, 숫자 및 UTF-8로 표시할 수 있는 공백과 + - = . _ : / @입니다.
- 태그 키와 값은 대/소문자를 구분합니다.
- `aws:` 접두사는 AWS 사용하도록 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 ElastiCache 리소스에 대한 자세한 내용은 섹션을 참조하세요태그 지정이 가능한 리소스.

리소스에 태그 지정 예제

- 태그를 사용하여 서버리스 캐시 생성. 이 예제에서는 Memcached를 엔진으로 사용합니다.

```
aws elasticache create-serverless-cache \
```

```
    --serverless-cache-name CacheName \
    --engine memcached
    --tags Key="Cost Center", Value="1110001" Key="project",Value="XYZ"
```

- 서버리스 캐시에 태그 추가

```
aws elasticache add-tags-to-resource \
--resource-name arn:aws:elasticache:us-east-1:111111222233:serverlesscache:my-cache \
--tags Key="project",Value="XYZ" Key="Elasticache",Value="Service"
```

- 복제 그룹에 태그 추가

```
aws elasticache add-tags-to-resource \
--resource-name arn:aws:elasticache:us-east-1:111111222233:replicationgroup:my-rg \
--tags Key="project",Value="XYZ" Key="Elasticache",Value="Service"
```

- 태그를 사용하여 캐시 클러스터 생성

```
aws elasticache create-cache-cluster \
--cluster-id testing-tags \
--cluster-description cluster-test \
--cache-subnet-group-name test \
--cache-node-type cache.t2.micro \
--engine valkey \
--tags Key="project",Value="XYZ" Key="Elasticache",Value="Service"
```

- 태그를 사용하여 캐시 클러스터 생성 이 예제에서는 Redis를 엔진으로 사용합니다.

```
aws elasticache create-cache-cluster \
--cluster-id testing-tags \
--cluster-description cluster-test \
--cache-subnet-group-name test \
--cache-node-type cache.t2.micro \
--engine valkey \
--tags Key="project",Value="XYZ" Key="Elasticache",Value="Service"
```

- 태그를 사용하여 서버리스 스냅샷 생성 이 예제에서는 Memcached를 엔진으로 사용합니다.

```
aws elasticache create-serverless-cache-snapshot \
--serverless-cache-name testing-tags \
--serverless-cache-snapshot-name bkp-testing-tags-scs \
```

```
--tags Key="work",Value="foo"
```

- 태그를 사용하여 스냅샷 생성

  스냅샷은 현재 Redis에서만 사용할 수 있습니다. 이 경우 요청에 태그를 추가하면 복제 그룹에 태그가 포함되어 있더라도 스냅샷은 요청 태그만 받습니다.

```
aws elasticache create-snapshot \
--replication-group-id testing-tags \
--snapshot-name bkp-testing-tags-rg \
--tags Key="work",Value="foo"
```

태그 기반 액세스 제어 정책 예제

1. 클러스터에 Project= 태그가 있는 경우에만 클러스터에 AddTagsToResource 작업을 허용합니다 XYZ.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticache:AddTagsToResource",
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Project": "XYZ"
                }
            }
        }
    ]
}
```

2. 복제 그룹에 Project 및 Service 태그가 포함되어 있고 Project 및 Service의 키가 서로 다른 경우에만 복제 그룹에서 RemoveTagsFromResource 작업을 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": "elasticache:RemoveTagsFromResource",
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Service": "Elasticache",
                    "aws:ResourceTag/Project": "XYZ"
                },
                "ForAnyValue:StringNotEqualsIgnoreCase": {
                    "aws:TagKeys": [
                        "Project",
                        "Service"
                    ]
                }
            }
        }
    ]
}
```

3. Project 및 Service의 태그가 서로 다른 경우에만 모든 리소스에 대한 AddTagsToResource 작업을 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticache:AddTagsToResource",
            "Resource": [
                "arn:aws:elasticache:*:*:*:*"
            ],
            "Condition": {
                "ForAnyValue:StringNotEqualsIgnoreCase": {
                    "aws:TagKeys": [
                        "Service",
                        "Project"
                    ]
                }
            }
        }
```

```
    ]
}
```

4. 요청에 `Tag Project=Foo`가 있는 경우 `CreateReplicationGroup` 작업을 거부합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "elasticache:CreateReplicationGroup",
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/Project": "Foo"
                }
            }
        }
    ]
}
```

5. 소스 스냅샷에 Project=XYZ 태그가 있고 요청 태그가 Service=Elasticache인 경우 `CopySnapshot` 작업 거부.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "elasticache:CopySnapshot",
            "Resource": [
                "arn:aws:elasticache:*:*:snapshot:*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Project": "XYZ",
                    "aws:RequestTag/Service": "Elasticache"
                }
            }
        }
    ]
```

```
}
```

6. 요청 태그 Project가 누락되었거나 Dev, QA 또는 Prod와 같지 않은 경우 CreateCacheCluster 작업은 거부됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
          {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*",
                "arn:aws:elasticache:*:*:securitygroup:*",
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ]
        },
        {
            "Effect": "Deny",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*"
            ],
            "Condition": {
                "Null": {
                    "aws:RequestTag/Project": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:AddTagsToResource"
            ],
            "Resource": "arn:aws:elasticache:*:*:cluster:*",
            "Condition": {
                "StringEquals": {
```

```
                    "aws:RequestTag/Project": [
                        "Dev",
                        "Prod",
                        "QA"
                    ]
                }
            }
        }
    ]
}
```

조건 키 관련 내용은 조건 키 사용 섹션을 참조하세요.

비용 할당 태그를 사용하여 비용을 모니터링합니다.

Amazon 의 리소스에 비용 할당 태그를 추가할 때 리소스 태그 값을 기준으로 송장의 비용을 그룹화하여 비용을 추적할 ElastiCache수 있습니다.

ElastiCache 비용 할당 태그는 사용자가 정의하고 ElastiCache 리소스와 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 자체 비용 구조를 반영하도록 AWS 청구서를 구성합니다. 이렇게 하려면 태그 키 값이 포함된 AWS 계정 청구서를 받으려면 가입하세요. 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 `ElastiCache` 및 `Asia Pacific (Singapore)` 값이 있을 수 있으며 다른 리소스에서 `ElastiCache` 및 `Europe (Frankfurt)` 값이 있을 수 있습니다. 그런 다음 총 ElastiCache 비용이 리전별로 구분되어 표시됩니다. 자세한 내용은 AWS Billing 사용 설명서의 비용 할당 태그 사용을 참조하세요.

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

ElastiCache 비용 할당 태그의 특성

- 비용 할당 태그는 CLI 및 API 작업에 로 지정된 ElastiCache 리소스에 적용됩니다ARN. 리소스 유형은 "클러스터"입니다.

  샘플ARN: `arn:aws:elasticache:<region>:<customer-id>:<resource-type>:<resource-name>`

  샘플 arn: `arn:aws:elasticache:us-west-2:1234567890: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.

- ElastiCache 리소스는 최대 50개의 태그를 가질 수 있습니다.

- 태그 세트의 값이 고유하지 않습니다. 예를 들어, 두 키 `Service`와 `Application`에 `ElastiCache` 값이 있는 태그 세트가 있을 수 있습니다.

AWS 는 태그에 의미론적 의미를 적용하지 않습니다. 태그는 문자열로 엄격하게 해석됩니다. AWS 는 ElastiCache 리소스에 태그를 자동으로 설정하지 않습니다.

를 사용하여 비용 할당 태그 관리 AWS CLI

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

비용 할당 태그는 ElastiCache 클러스터에 적용됩니다. 태그를 지정할 클러스터는 ARN (Amazon 리소스 이름)을 사용하여 지정됩니다.

샘플 arn: `arn:aws:elasticache:us-west-2:1234567890:cluster:my-cluster`

주제

를 사용하여 태그 나열 AWS CLI

를 사용하여 를 사용하여 기존 ElastiCache 리소스의 태그를 나열 AWS CLI 할 수 있습니다.list-tags-for-resource 작업.

다음 코드는 AWS CLI 를 사용하여 us-west-2 리전의 Memcached 클러스터에 태그를 나열`my-cluster`합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache list-tags-for-resource \
  --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster
```

Windows의 경우:

```
aws elasticache list-tags-for-resource ^
  --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster
```

다음 코드는 AWS CLI 를 사용하여 us-west-2 리전의 `my-cluster` 클러스터`my-cluster-001`에 있는 Valkey 또는 Redis OSS 노드의 태그를 나열합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache list-tags-for-resource \
  --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001
```

Windows의 경우:

```
aws elasticache list-tags-for-resource ^
  --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001
```

이 작업의 출력은 다음 리소스의 모든 태그 목록과 유사합니다.

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

리소스에 태그가 없는 경우 출력은 비어 있습니다 TagList.

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

자세한 내용은 의 섹션을 참조 AWS CLI 하세요. ElastiCache list-tags-for-resource.

를 사용하여 태그 추가 AWS CLI

AWS CLI 를 사용하여 를 사용하여 기존 ElastiCache 리소스에 태그를 추가할 수 있습니다.add-tags-to-resource CLI 작업. 리소스에 태그 키가 없으면 키와 값이 리소스에 추가됩니다. 리소스에 이미 키가 있는 경우 해당 키와 연결된 값이 새 값으로 업데이트됩니다.

다음 코드는 AWS CLI 를 사용하여 `Service` us-west-2 리전의 클러스터`my-cluster-001`에 있는 노드`my-cluster`에 `elasticache` `us-west-2` 각각 및 값과 `Region` 함께 및 키를 추가합니다.

Memcached

Linux, macOS, Unix의 경우:

```
aws elasticache add-tags-to-resource \
 --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster \
 --tags Key=Service,Value=elasticache \
        Key=Region,Value=us-west-2
```

Windows의 경우:

```
aws elasticache add-tags-to-resource ^
 --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster ^
 --tags Key=Service,Value=elasticache ^
        Key=Region,Value=us-west-2
```

Redis

Linux, macOS, Unix의 경우:

```
aws elasticache add-tags-to-resource \
 --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001 \
 --tags Key=Service,Value=elasticache \
        Key=Region,Value=us-west-2
```

Windows의 경우:

```
aws elasticache add-tags-to-resource ^
 --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001 ^
 --tags Key=Service,Value=elasticache ^
        Key=Region,Value=us-west-2
```

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

```
{
   "TagList": [
      {
         "Value": "elasticache",
         "Key": "Service"
      },
      {
         "Value": "us-west-2",
         "Key": "Region"
      }
   ]
}
```

자세한 내용은 의 섹션을 참조 AWS CLI 하세요. ElastiCache add-tags-to-resource.

작업을 사용하여 새 클러스터를 생성할 때 를 사용하여 클러스터에 태그를 AWS CLI 추가할 수도 있습니다.create-cache-cluster. ElastiCache 관리 콘솔을 사용하여 클러스터를 생성할 때는 태그를 추가할 수 없습니다. 클러스터가 생성된 후에는 콘솔을 사용하여 클러스터에 태그를 추가할 수 있습니다.

를 사용하여 태그 수정 AWS CLI

AWS CLI 를 사용하여 ElastiCache 클러스터의 태그를 수정할 수 있습니다.

태그를 수정하려면

- 사용 add-tags-to-resource 새 태그 및 값을 추가하거나 기존 태그와 연결된 값을 변경합니다.
- 사용 remove-tags-from-resource 리소스에서 지정된 태그를 제거합니다.

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

를 사용하여 태그 제거 AWS CLI

를 사용하여 AWS CLI 를 사용하여 기존 ElastiCache (Memcached) 클러스터에서 태그를 제거할 수 있습니다.remove-tags-from-resource 작업.

Memcached의 경우 다음 코드는 AWS CLI 를 사용하여 us-west-2 리전my-cluster의 클러스터my-cluster-001에 있는 노드Region에서 Service 및 키를 사용하여 태그를 제거합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache remove-tags-from-resource \
 --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster \
 --tag-keys PM Service
```

Windows의 경우:

```
aws elasticache remove-tags-from-resource ^
 --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster ^
 --tag-keys PM Service
```

Redis의 경우 다음 코드는 AWS CLI 를 사용하여 my-cluster us-west-2 리전의 클러스터my-cluster-001에 있는 노드Region에서 Service 및 키를 사용하여 태그를 제거합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache remove-tags-from-resource \
 --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001 \
 --tag-keys PM Service
```

Windows의 경우:

```
aws elasticache remove-tags-from-resource ^
 --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001 ^
 --tag-keys PM Service
```

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

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

자세한 내용은 의 섹션을 참조 AWS CLI 하세요. ElastiCache remove-tags-from-resource.

를 사용하여 비용 할당 태그 관리 ElastiCache API

를 ElastiCache API 사용하여 비용 할당 태그를 추가, 수정 또는 제거할 수 있습니다.

비용 할당 태그는 Memcached 클러스터 ElastiCache 에 대해 에 적용됩니다. 태그를 지정할 클러스터는 ARN (Amazon 리소스 이름)을 사용하여 지정됩니다.

샘플 arn: `arn:aws:elasticache:us-west-2:1234567890:cluster:my-cluster`

주제

- 를 사용하여 태그 나열 ElastiCache API
- 를 사용하여 태그 추가 ElastiCache API
- 를 사용하여 태그 수정 ElastiCache API
- 를 사용하여 태그 제거 ElastiCache API

를 사용하여 태그 나열 ElastiCache API

를 ElastiCache API 사용하여 를 사용하여 기존 리소스의 태그를 나열할 수 있습니다.ListTagsForResource 작업.

Memcached의 경우 다음 코드는 를 ElastiCache API 사용하여 us-west-2 리전의 리소스에 태그를 나열my-cluster합니다.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=ListTagsForResource
   &ResourceName=arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster
   &SignatureVersion=4
```

```
   &SignatureMethod=HmacSHA256
   &Version=2015-02-02
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

Redis의 경우 다음 코드는 를 ElastiCache API 사용하여 us-west-2 리전의 리소스에 태그를 나열`my-cluster-001`합니다.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=ListTagsForResource
   &ResourceName=arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Version=2015-02-02
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

를 사용하여 태그 추가 ElastiCache API

를 ElastiCache API 사용하여 를 사용하여 기존 ElastiCache 클러스터에 태그를 추가할 수 있습니다.AddTagsToResource 작업. 리소스에 태그 키가 없으면 키와 값이 리소스에 추가됩니다. 리소스에 이미 키가 있는 경우 해당 키와 연결된 값이 새 값으로 업데이트됩니다.

다음 코드는 를 ElastiCache API 사용하여 `Service` 각각 및 값과 `Region` 함께 `elasticache` 및 키를 추가합니다`us-west-2`. Memcached의 경우 리소스 에 적용됩니다`my-cluster`. Redis의 경우 us-west-2 리전의 리소스`my-cluster-001`에 적용됩니다.

Memcached

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=AddTagsToResource
   &ResourceName=arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Tags.member.1.Key=Service
   &Tags.member.1.Value=elasticache
   &Tags.member.2.Key=Region
   &Tags.member.2.Value=us-west-2
   &Version=2015-02-02
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

Redis

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=AddTagsToResource
   &ResourceName=arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Tags.member.1.Key=Service
   &Tags.member.1.Value=elasticache
   &Tags.member.2.Key=Region
   &Tags.member.2.Value=us-west-2
   &Version=2015-02-02
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

자세한 내용을 알아보려면 다음 섹션을 참조하세요.AddTagsToResource Amazon ElastiCache API 참조 의 .

를 사용하여 태그 수정 ElastiCache API

를 ElastiCache API 사용하여 ElastiCache 클러스터의 태그를 수정할 수 있습니다.

태그 값을 수정하려면

- 사용 AddTagsToResource 새 태그 및 값을 추가하거나 기존 태그의 값을 변경하는 작업입니다.
- 사용 RemoveTagsFromResource 리소스에서 태그를 제거합니다.

두 작업 중 하나의 출력은 지정된 리소스의 태그 목록과 값입니다.

사용 RemoveTagsFromResource 리소스에서 태그를 제거합니다.

를 사용하여 태그 제거 ElastiCache API

를 ElastiCache API 사용하여 를 사용하여 기존 ElastiCache (Memcached) 클러스터에서 태그를 제거할 수 있습니다.RemoveTagsFromResource 작업.

다음 코드는 ElastiCache API를 사용하여 us-west-2 `my-cluster` 리전의 클러스터`my-cluster-001`에 있는 노드`Region`에서 `Service` 및 키를 사용하여 태그를 제거합니다.

```
https://elasticache.us-west-2.amazonaws.com/
```

```
?Action=RemoveTagsFromResource
&ResourceName=arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&TagKeys.member.1=Service
&TagKeys.member.2=Region
&Version=2015-02-02
&Timestamp=20150202T192317Z
&X-Amz-Credential=<credential>
```

Amazon ElastiCache Well-Architected 렌즈 사용

이 섹션에서는 잘 설계된 ElastiCache 워크로드를 설계하기 위한 설계 원칙 및 지침 모음인 Amazon ElastiCache Well-Architected Lens에 대해 설명합니다.

- ElastiCache 렌즈는 AWS Well-Architected Framework 에 추가됩니다.
- 각 Pillar에는 아키텍처에 대한 토론을 시작하는 데 도움이 되는 일련의 질문이 있습니다 ElastiCache.
  - 각 질문에는 보고용 점수와 함께 여러 가지 주요 사례가 있습니다.
    - 필수 - 프로덕션 전 필요(따르지 않으면 위험도 높음)
    - 가장 좋음 - 고객이 될 수 있는 최상의 상태
    - 좋음 - 고객에게 권하는 상태(따르지 않으면 위험도 중간)
- Well-Architected 용어
  - 구성 요소 - 요구 사항에 따라 함께 제공하는 코드, 구성 및 AWS 리소스입니다. 구성 요소는 다른 구성 요소와 상호 작용하며 종종 마이크로 서비스 아키텍처의 서비스와 동일합니다.
  - 워크로드 - 함께 비즈니스 가치를 제공하는 구성 요소 세트입니다. 예를 들어, 마케팅 웹사이트, 전자 상거래 웹사이트, 모바일 앱 백엔드, 분석 플랫폼 등이 워크로드에 해당합니다.

> **Note**
>
> 이 가이드는 ElastiCache 서버리스 캐싱 및 새로운 Valkey 엔진에 대한 정보를 포함하도록 업데이트되지 않았습니다.

주제

- Amazon ElastiCache Well-Architected Lens Operational Excellence 기둥

- Amazon ElastiCache Well-Architected Lens Security Pillar
- Amazon ElastiCache Well-Architected 렌즈 신뢰성 기둥
- Amazon ElastiCache Well-Architected 렌즈 성능 효율성 기둥
- Amazon ElastiCache Well-Architected Lens Cost Optimization Pillar

Amazon ElastiCache Well-Architected Lens Operational Excellence 기둥

운영 우수성 요소는 비즈니스 가치를 제공하고 프로세스 및 절차를 지속적으로 개선하기 위한 시스템 운영 및 모니터링에 중점을 둡니다. 주요 주제에는 변경 자동화, 이벤트 대응, 일상 운영 관리를 위한 표준 정의 등이 포함됩니다.

주제

- OE 1: ElastiCache 클러스터에 의해 트리거되는 알림 및 이벤트에 대해 어떻게 이해하고 대응합니까?
- OE 2: 기존 ElastiCache 클러스터를 언제 어떻게 확장하나요?
- OE 3: ElastiCache 클러스터 리소스를 관리하고 클러스터를 유지 관리하려면 어떻게 해야 하나요 up-to-date?
- OE 4: ElastiCache 클러스터에 대한 클라이언트 연결을 어떻게 관리하나요?
- OE 5: 워크로드의 ElastiCache 구성 요소를 배포하려면 어떻게 해야 하나요?
- OE 6: 장애를 어떻게 대비하고 완화할 계획인가요?
- OE 7: Valkey 또는 Redis OSS 엔진 이벤트 문제를 어떻게 해결하나요?

OE 1: ElastiCache 클러스터에 의해 트리거되는 알림 및 이벤트에 대해 어떻게 이해하고 대응합니까?

질문 수준 소개: ElastiCache 클러스터를 작동할 때 특정 이벤트가 발생할 때 선택적으로 알림 및 알림을 받을 수 있습니다. ElastiCache기본적으로 는 장애 조치, 노드 교체, 조정 작업, 예약된 유지 관리 등과 같이 리소스와 관련된 이벤트를 기록합니다. 각 이벤트에는 날짜 및 시간, 소스 이름 및 소스 유형, 설명이 포함됩니다.

질문 수준의 이점: 클러스터에서 생성되는 경고를 트리거하는 이벤트의 근본 원인을 이해하고 관리할 수 있다면 더 효과적으로 운영하고 이벤트에 적절하게 대응할 수 있습니다.

- [필수] ElastiCache 콘솔ElastiCache 에서(리전 선택 후) 또는 Amazon Command Line Interface(AWS CLI) describe-events 명령 및 를 사용하여 에서 생성된 이벤트를 검토합니

다ElastiCache API. Amazon Simple Notification Service(Amazon)를 사용하여 중요한 클러스터 이벤트에 대한 알림을 보내 ElastiCache 도록 를 구성합니다SNS. 클러스터와 SNS 함께 Amazon을 사용하면 ElastiCache 이벤트에 대해 프로그래밍 방식으로 조치를 취할 수 있습니다.

- 이벤트에는 크게 현재 이벤트와 예정된 이벤트라는 두 가지 범주가 있습니다. 현재 이벤트 목록에는 리소스 생성 및 삭제, 조정 작업, 장애 조치, 노드 재부팅, 생성된 스냅샷, 클러스터의 파라미터 수정, CA 인증서 갱신, 장애 이벤트(클러스터 프로비저닝 실패 - VPC 또는 ENI-, 조정 실패 - 및 ENI스냅샷 실패)가 포함됩니다. 예정된 이벤트 목록에는 유지 관리 기간 동안 교체가 예정된 노드 및 교체 일정이 변경된 노드가 포함됩니다.
- 이러한 이벤트 중 일부에 즉시 대응할 필요는 없지만 먼저 모든 장애 이벤트를 살펴보는 것이 중요합니다.
  - ElastiCache:AddCacheNodeFailed
  - ElastiCache:CacheClusterProvisioningFailed
  - ElastiCache:CacheClusterScalingFailed
  - ElastiCache:CacheNodesRebooted
  - ElastiCache:SnapshotFailed (발키 또는 RedisOSS만 해당)
- [리소스]:
  - ElastiCache Amazon SNS 알림 관리
  - 이벤트 알림 및 Amazon SNS

- [최고] 이벤트에 대한 응답을 자동화하려면 SNS 및 Lambda Functions와 같은 AWS 제품 및 서비스 기능을 활용합니다. 모범 사례에 따라 작고 빈번하며 되돌릴 수 있는 변경을 코드로 적용하여 시간이 지남에 따라 운영을 발전시킵니다. Amazon CloudWatch 지표를 사용하여 클러스터를 모니터링해야 합니다.

  [리소스]: AWS Lambda 및 를 사용하는 사용 사례에 대해 Lambda, Amazon Route 53 및 Amazon을 사용하여 ElastiCache (Redis OSS)(클러스터 모드 비활성화됨) 읽기 전용 복제본 엔드포인트를 모니터링합니다SNSSNS.

OE 2: 기존 ElastiCache 클러스터를 언제 어떻게 확장하나요?

질문 수준 소개: ElastiCache 클러스터의 크기를 올바르게 조정하는 것은 기본 워크로드 유형이 변경될 때마다 평가해야 하는 밸런싱 작업입니다. 목표는 워크로드에 적합한 규모의 환경에서 운영하는 것입니다.

질문 수준의 이점: 리소스를 과도하게 사용하면 지연 시간이 늘어나고 전반적인 성능이 저하될 수 있습니다. 반면, 활용도가 낮으면 최적화되지 않은 비용으로 리소스를 과도하게 프로비저닝하게 될 수 있

습니다. 환경을 적절한 규모로 조정하면 성능 효율성과 비용 최적화 간의 균형을 맞출 수 있습니다. 리소스 사용률 초과 또는 미달을 해결하려면 를 두 가지 차원으로 확장 ElastiCache 할 수 있습니다. 노드 용량을 늘리거나 줄여서 수직으로 규모를 조정할 수 있습니다. 노드를 추가 및 제거하여 수평적으로 규모를 조정할 수도 있습니다.

- [필수] CPU 및 기본 노드의 네트워크 과다 사용은 읽기 작업을 오프로드하고 복제본 노드로 리디렉션하여 해결해야 합니다. 읽기 작업에 복제본 노드를 사용하여 프라이머리 노드 사용률을 줄입니다. 클러스터 모드가 비활성화된 경우 ElastiCache 리더 엔드포인트에 연결하거나 클러스터 모드가 활성화된 경우 READONLY 명령을 사용하여 Valkey 또는 Redis OSS 클라이언트 라이브러리에서 구성할 수 있습니다.

  [리소스]:

  - 에서 연결 엔드포인트 찾기 ElastiCache
  - 클러스터 적정 크기 조정
  - READONLY 명령
- [필수] CPU, 메모리 및 네트워크와 같은 중요 클러스터 리소스의 사용률을 모니터링합니다. 이러한 특정 클러스터 리소스의 사용률을 추적하여 규모 조정을 결정하고 규모 조정 작업의 유형을 결정하는 데 활용해야 합니다. ElastiCache (Redis OSS) 클러스터 모드가 비활성화된 경우 기본 및 복제본 노드는 수직으로 확장할 수 있습니다. 복제본 노드는 0개 노드에서 5개 노드까지 수평적으로 확장할 수도 있습니다. 클러스터 모드가 활성화된 경우 클러스터의 각 샤드에 동일하게 적용됩니다. 또한 샤드 수를 늘리거나 줄일 수 있습니다.

  [리소스]:

  - Amazon을 사용하여 ElastiCache (Redis OSS)를 사용하여 모범 사례 모니터링 CloudWatch
  - 스케일링 ElastiCache (RedisOSS) 클러스터
  - Memcached 클러스터의 ElastiCache 크기 조정
- [가장 좋음] 시간 경과에 따른 추세 모니터링은 특정 시점에 모니터링할 경우 눈에 띄지 않는 워크로드 변경을 감지하는 데 도움이 될 수 있습니다. 장기 추세를 감지하려면 지표를 사용하여 CloudWatch 더 긴 시간 범위를 스캔합니다. 장기 CloudWatch 지표 관찰을 통해 얻은 학습은 클러스터 리소스 사용률에 대한 예측에 도움이 됩니다. CloudWatch 데이터 포인트 및 지표는 최대 455일 동안 사용할 수 있습니다.

  [리소스]:

  - 지표를 사용한 CloudWatch 모니터링 ElastiCache (RedisOSS)
  - CloudWatch 지표를 사용하여 Memcached 모니터링
  - Amazon을 사용하여 ElastiCache (Redis OSS)를 사용하여 모범 사례 모니터링 CloudWatch

- [모범] ElastiCache 리소스를 로 생성하는 경우 템플릿을 사용하여 CloudFormation 변경을 수행하여 운영 일관성을 유지하고 관리되지 않는 구성 변경 및 스택 드리프트를 방지하는 CloudFormation 것이 좋습니다.

  [리소스]:

  - ElastiCache 에 대한 리소스 유형 참조 CloudFormation
- [최고] 클러스터 운영 데이터를 사용하여 조정 작업을 자동화하고 에서 임계값을 정의 CloudWatch 하여 경보를 설정합니다. CloudWatch 이벤트 및 Simple Notification Service(SNS)를 사용하여 Lambda 함수를 트리거하고 를 ElastiCache API 실행하여 클러스터를 자동으로 확장합니다. `EngineCPUUtilization` 지표가 장기간 80%에 도달하면 클러스터에 샤드를 추가하는 경우를 예로 들 수 있습니다. 또 다른 옵션은 메모리 기반 임계값에 `DatabaseMemoryUsedPercentages`를 사용하는 것입니다.

  [리소스]:

  - Amazon CloudWatch 경보 사용
  - Amazon CloudWatch 이벤트란 무엇입니까?
  - Amazon Simple Notification Service AWS Lambda 와 함께 사용
  - ElastiCache API 참조

OE 3: ElastiCache 클러스터 리소스를 관리하고 클러스터를 유지 관리하려면 어떻게 해야 하나요 up-to-date?

질문 수준 소개: 대규모로 운영할 때는 모든 리소스를 정확히 찾아 식별할 수 있어야 합니다 ElastiCache. 새 애플리케이션 기능을 롤아웃할 때는 개발, 테스트, 프로덕션 등 모든 ElastiCache 환경 유형에 클러스터 버전 대칭을 생성해야 합니다. 리소스 속성을 사용하면 새 기능을 배포하거나 새 보안 메커니즘을 활성화하는 등 다양한 운영 목표에 맞게 환경을 분리할 수 있습니다.

질문 수준의 이점: 개발, 테스트 및 프로덕션 환경을 분리하는 것이 운영 모범 사례입니다. 또한 잘 이해되고 문서화된 프로세스를 사용하여 환경 전반의 클러스터와 노드에 최신 소프트웨어 패치를 적용하는 것이 가장 좋습니다. 기본 ElastiCache 기능을 활용하면 엔지니어링 팀이 ElastiCache 유지 관리가 아닌 비즈니스 목표 달성에 집중할 수 있습니다.

- [최고] 사용 가능한 최신 엔진 버전에서 를 실행하고 셀프 서비스 업데이트가 사용 가능하게 되는 즉시 적용합니다. 클러스터의 지정된 유지 관리 기간 동안 기본 인프라를 ElastiCache 자동으로 업데이트합니다. 하지만 클러스터에서 실행 중인 노드는 셀프 서비스 업데이트를 통해 업데이트됩니다. 이러한 업데이트에는 보안 패치 또는 소프트웨어 소규모 업데이트의 두 가지 유형이 있습니다. 패치 유형과 적용 시기의 차이를 이해해야 합니다.

[리소스]:

- Amazon의 셀프 서비스 업데이트 ElastiCache
- Amazon ElastiCache Managed Maintenance and Service Updates 도움말 페이지

- [Best] 태그를 사용하여 ElastiCache 리소스를 구성합니다. 개별 노드가 아닌 복제 그룹에 태그를 사용합니다. 리소스를 쿼리할 때 표시되도록 태그를 구성하고 태그를 사용하여 검색을 수행하고 필터를 적용할 수 있습니다. 일반적인 태그 세트를 공유하는 리소스 모음을 쉽게 만들고 유지 관리하려면 리소스 그룹을 만들어야 합니다.

  [리소스]:

  - 태깅 모범 사례
  - ElastiCache 에 대한 리소스 유형 참조 CloudFormation
  - 파라미터 그룹

OE 4: ElastiCache 클러스터에 대한 클라이언트 연결을 어떻게 관리하나요?

질문 수준 소개: 대규모로 운영할 때는 클라이언트가 ElastiCache 클러스터와 연결되어 애플리케이션 운영 측면(예: 응답 시간)을 관리하는 방법을 이해해야 합니다.

질문 수준의 이점: 가장 적절한 연결 메커니즘을 선택하면 시간 초과와 같은 연결 오류로 인해 애플리케이션 연결이 끊기지 않습니다.

- [필수] 읽기와 쓰기 작업을 분리하고 복제본 노드에 연결하여 읽기 작업을 실행합니다. 하지만 쓰기와 읽기를 분리하면 Valkey 및 Redis OSS 복제의 비동기적 특성으로 인해 쓰기 직후 키를 읽을 수 없게 됩니다. WAIT 명령을 활용하여 실제 데이터 안전을 개선하고 전체 성능 비용으로 클라이언트에 응답하기 전에 복제본이 쓰기를 승인하도록 할 수 있습니다. 읽기 작업에 복제본 노드를 사용하는 것은 클러스터 모드 비활성화에 대한 ElastiCache 리더 엔드포인트를 사용하여 ElastiCache (Redis OSS) 클라이언트 라이브러리에서 구성할 수 있습니다. 클러스터 모드가 활성화된 경우 ElastiCache (Redis OSS) READONLY 명령을 사용합니다. 많은 ElastiCache (Redis OSS) 클라이언트 라이브러리의 경우 ElastiCache (Redis OSS)READONLY가 기본적으로 또는 구성 설정을 통해 구현됩니다.

  [리소스]:

  - 에서 연결 엔드포인트 찾기 ElastiCache
  - READONLY

- [필수] 연결 풀링을 사용합니다. TCP 연결을 설정하면 클라이언트와 서버 측 모두에서 CPU 시간이 많이 걸리고 풀링을 통해 TCP 연결을 재사용할 수 있습니다.

연결 오버헤드를 줄이려면 연결 풀링을 사용해야 합니다. 연결 풀을 사용하면 애플리케이션에서 연결 설정 비용 없이 '마음대로' 연결을 재사용하고 연결을 해제할 수 있습니다. 애플리케이션 환경에 사용할 수 있는 프레임워크를 사용하여 ElastiCache (Redis OSS) 클라이언트 라이브러리(지원되는 경우)를 통해 연결 풀링을 구현하거나 처음부터 구축할 수 있습니다.

- [가장 좋음] 클라이언트의 소켓 제한 시간이 최소 1초로 설정되어 있는지 확인합니다. 몇몇 클라이언트의 일반적인 기본값인 '없음'으로 설정되어 있으면 안 됩니다.
  - 제한 시간 값을 너무 낮게 설정하면 서버 로드가 높을 때 시간 초과가 발생할 수 있습니다. 너무 높게 설정하면 애플리케이션에서 연결 문제를 감지하는 데 시간이 오래 걸릴 수 있습니다.
  - 클라이언트 애플리케이션에서 연결 풀링을 구현하여 새 연결의 볼륨을 제어합니다. 이렇게 하면 연결을 열고 닫는 데 필요한 지연 시간과 CPU 사용률이 줄어들고 가 클러스터에서 활성화된 경우 TLS 핸드셰TLS이크를 수행할 수 있습니다.

  [리소스]: 고가용성을 위한 구성 ElastiCache (RedisOSS)
- [좋음] (사용 사례에서 가능한 경우) 파이프라이닝을 사용하면 성능이 크게 향상될 수 있습니다.
  - 파이프라이닝을 사용하면 애플리케이션 클라이언트와 클러스터 간의 왕복 시간(RTT)을 줄이고 클라이언트가 이전 응답을 아직 읽지 않은 경우에도 새 요청을 처리할 수 있습니다.
  - 파이프라이닝을 사용하면 응답/확인을 기다리지 않고 서버에 여러 명령을 보낼 수 있습니다. 파이프라이닝의 단점은 결국 모든 응답을 대량으로 가져올 때 끝에 가서야 잡을 수 있는 오류가 있을 수 있다는 점입니다.
  - 잘못된 요청을 생략하는 오류가 반환될 때 요청을 재시도하는 메서드를 구현합니다.

  [리소스]: 파이프라이닝

OE 5: 워크로드의 ElastiCache 구성 요소를 배포하려면 어떻게 해야 하나요?

질문 수준 소개: ElastiCache 환경은 AWS 콘솔을 통해 수동으로 배포하거나 , APIs, CLI툴킷 등을 통해 프로그래밍 방식으로 배포할 수 있습니다. 운영 우수성 모범 사례에서는 가능하면 코드를 통해 배포를 자동화하는 것을 권장합니다. 또한 ElastiCache 클러스터는 워크로드별로 격리하거나 비용 최적화를 위해 결합할 수 있습니다.

질문 수준의 이점: ElastiCache 환경에 가장 적합한 배포 메커니즘을 선택하면 시간이 지남에 따라 운영 우수성을 개선할 수 있습니다. 인적 오류를 최소화하고 반복성, 유연성 및 이벤트에 대한 응답 시간을 향상하기 위해서는 가능하면 코드로 작업을 수행하는 것이 좋습니다.

워크로드 격리 요구 사항을 이해하면 워크로드당 전용 ElastiCache 환경을 보유하거나 여러 워크로드를 단일 클러스터 또는 그 조합으로 결합하도록 선택할 수 있습니다. 장단점을 이해하면 운영 우수성과 비용 최적화 간의 균형을 맞추는 데 도움이 될 수 있습니다.

- [필수] 에서 사용할 수 있는 배포 옵션을 이해하고 가능하면 이러한 절차를 ElastiCache자동화합니다. 가능한 자동화 방법에는 CloudFormation, AWS CLI/SDK, 및 가 포함됩니다APIs.

  [리소스]:

  - Amazon ElastiCache 리소스 유형 참조
  - elasticache
  - Amazon ElastiCache API 참조
- [필수] 모든 워크로드에 대해 필요한 클러스터 격리 수준을 결정합니다.
  - [가장 좋음]: 높은 수준의 격리 - 워크로드와 클러스터를 1:1로 매핑합니다. 워크로드별로 ElastiCache 리소스의 액세스, 크기 조정, 크기 조정 및 관리를 완벽하게 제어할 수 있습니다.
  - [더 좋음]: 중간 수준의 격리 - 목적별로 M:1로 격리하지만 여러 워크로드 간에 공유될 수 있습니다(예: 워크로드 캐싱 전용 클러스터와 메시징 전용 클러스터).
  - [좋음]: 낮은 수준의 격리 - M:1 격리로 다용도로 사용하며 완전히 공유합니다. 공유 액세스가 허용되는 워크로드에 권장됩니다.

OE 6: 장애를 어떻게 대비하고 완화할 계획인가요?

질문 수준 소개: 운영 우수성에는 테스트 목적으로 잠재적 장애 원인을 제거하거나 완화할 수 있도록 정기적인 '사전' 연습을 수행하여 장애를 예측API하는 것이 포함됩니다. 는 시뮬레이션된 노드 장애 이벤트를 허용하는 장애 조치를 ElastiCache 제공합니다.

질문 수준의 이점: 장애 시나리오를 미리 테스트하면 장애 시나리오가 워크로드에 미치는 영향을 파악할 수 있습니다. 이를 통해 대응 절차와 그 효과를 안전하게 테스트할 수 있을 뿐만 아니라 팀이 대응 절차 실행에 익숙해질 수 있습니다.

[필수] 개발/테스트 계정에서 장애 조치 테스트를 정기적으로 수행합니다. TestFailover

OE 7: Valkey 또는 Redis OSS 엔진 이벤트 문제를 어떻게 해결하나요?

질문 수준 소개: Operational Excellence를 사용하려면 서비스 수준 및 엔진 수준 정보를 모두 조사하여 클러스터의 상태와 상태를 분석하는 기능이 필요합니다. ElastiCache 는 Amazon CloudWatch 및 Amazon Kinesis Data Firehose 모두에 Valkey 또는 Redis OSS 엔진 로그를 내보낼 수 있습니다.

질문 수준 이점: ElastiCache 클러스터에서 Valkey 또는 Redis OSS 엔진 로그를 활성화하면 클러스터의 상태와 성능에 영향을 미치는 이벤트에 대한 인사이트를 얻을 수 있습니다. Valkey 또는 Redis OSS 엔진 로그는 ElastiCache 이벤트 메커니즘을 통해 사용할 수 없는 데이터를 엔진에서 직접 제공합니다. ElastiCache 이벤트(이전 OE-1 참조)와 엔진 로그를 모두 주의 깊게 관찰하여 ElastiCache 서비스 관점과 엔진 관점에서 문제를 해결할 때 이벤트 순서를 결정할 수 있습니다.

- [필수] ElastiCache (Redis OSS) 6.2 이상부터 사용할 수 있는 Redis OSS 엔진 로깅 기능이 활성화되어 있는지 확인합니다. 클러스터 생성 중에 또는 생성 후 클러스터를 수정하여 이 작업을 수행할 수 있습니다.
  - Amazon CloudWatch Logs 또는 Amazon Kinesis Data Firehose가 Redis OSS 엔진 로그의 적절한 대상인지 확인합니다.
  - CloudWatch 또는 Kinesis Data Firehose 내에서 적절한 대상 로그를 선택하여 로그를 유지합니다. 클러스터가 여러 개 있는 경우, 문제 해결 시 데이터를 분리하는 데 도움이 되므로 클러스터마다 다른 대상 로그를 사용하는 것이 좋습니다.

  [리소스]:
  - 로그 전달: 로그 전달
  - 로깅 대상: Amazon CloudWatch Logs
  - Amazon CloudWatch Logs 소개: Amazon CloudWatch Logs란 무엇입니까?
  - Amazon Kinesis Data Firehose 소개: Amazon Kinesis Data Firehose란 무엇인가요?
- [최고] Amazon CloudWatch Logs를 사용하는 경우 Amazon CloudWatch Logs Insights를 활용하여 중요한 정보를 위해 Valkey 또는 Redis OSS 엔진 로그를 쿼리하는 것이 좋습니다.

  예를 들어 다음과 같이 가 LogLevel 'WARNING'인 이벤트를 반환하는 Valkey 또는 Redis OSS 엔진 로그가 포함된 CloudWatch 로그 그룹에 대한 쿼리를 생성합니다.

```
fields @timestamp, LogLevel, Message
| sort @timestamp desc
| filter LogLevel = "WARNING"
```

  [리소스]: CloudWatch Logs Insights를 사용한 로그 데이터 분석

Amazon ElastiCache Well-Architected Lens Security Pillar

보안 요소는 정보 및 시스템 보호에 중점을 둡니다. 주요 주제로는 데이터의 기밀성 및 무결성, 권한 기반의 관리를 통해 누가 무엇을 할 수 있는지 식별 및 관리하기, 시스템 보호하기, 보안 이벤트 탐지를 위한 제어 설정하기 등이 있습니다.

주제

SEC 1: ElastiCache 데이터에 대한 권한 있는 액세스를 제어하기 위해 어떤 조치를 취하고 있습니까?

질문 수준 소개: 모든 ElastiCache 클러스터는 VPC, 서버리스 함수(AWS Lambda) 또는 컨테이너(Amazon Elastic Container Service)의 Amazon Elastic Compute Cloud 인스턴스에서 액세스하도록 설계되었습니다. 가장 많이 발생하는 시나리오는 동일한 Amazon Virtual Private Cloud(Amazon Virtual Private Cloud) 내의 Amazon Elastic Compute Cloud 인스턴스에서 ElastiCache 클러스터에 액세스하는 것입니다. Amazon EC2 인스턴스에서 클러스터에 연결하려면 먼저 Amazon EC2 인스턴스가 클러스터에 액세스할 수 있도록 권한을 부여해야 합니다. 에서 실행되는 ElastiCache 클러스터에 액세스하려면 클러스터에 네트워크 수신을 부여VPC해야 합니다.

질문 수준의 이점: 클러스터로의 네트워크 수신은 VPC 보안 그룹을 통해 제어됩니다. 보안 그룹은 Amazon EC2 인스턴스가 수신 및 발신 트래픽을 제어할 수 있는 가상 방화벽 역할을 합니다. 인바운드 규칙은 인스턴스로 들어오는 트래픽을 제어하고 아웃바운드 규칙은 인스턴스에서 나가는 트래픽을 제어합니다. 의 경우 클러스터를 시작할 ElastiCache때 보안 그룹을 연결해야 합니다. 이렇게 하면 클러스터를 구성하는 모든 노드에 대해 인바운드 및 아웃바운드 트래픽 규칙이 적용됩니다. 또한 ElastiCache 는 VPC의 프라이빗 네트워킹을 통해서만 액세스할 수 있도록 프라이빗 서브넷에만 배포하도록 구성되어 있습니다.

- [필수] 클러스터와 연결된 보안 그룹은 클러스터에 대한 네트워크 수신 및 액세스를 제어합니다. 기본적으로 보안 그룹에는 인바운드 규칙이 정의되지 않으므로 에 대한 수신 경로가 없습니다 ElastiCache. 이를 활성화하려면 소스 IP 주소/범위를 지정하는 보안 그룹에 인바운드 규칙을 구성하고 트래픽과 ElastiCache 클러스터의 포트를 TCP 입력합니다(예: ElastiCache (Redis OSS)의 기본 포트 6379). VPC (0.0.0.0/0) 내의 모든 리소스와 같이 매우 광범위한 수신 소스를 허용하는 것이 가능하지만, 특정 보안 그룹과 연결된 Amazon EC2 인스턴스에서 실행되는 Valkey 또는 Redis OSS 클라이언트에 대한 인바운드 액세스 권한 부여와 같은 인바운드 규칙을 정의할 때는 최대한 세분화하는 것이 좋습니다.

  [리소스]:

  - 서브넷 및 서브넷 그룹
  - 클러스터 또는 복제 그룹에 액세스
  - 보안 그룹을 사용하여 리소스에 대한 트래픽 제어
  - Linux 인스턴스용 Amazon Elastic Compute Cloud 보안 그룹
- [필수] AWS Identity and Access Management 정책을 AWS Lambda 함수에 할당하여 ElastiCache 데이터에 액세스할 수 있습니다. 이 기능을 활성화하려면 `AWSLambdaVPCAccessExecutionRole` 권한으로 IAM 실행 역할을 생성한 다음 AWS Lambda 함수에 역할을 할당합니다.

  [리소스]: Amazon ElastiCache 에서 Amazon에 액세스하도록 Lambda 함수 구성VPC: 자습서: Amazon ElastiCache 에서 Amazon에 액세스하도록 Lambda 함수 구성 VPC

SEC 2: 애플리케이션이 네트워킹 기반 제어 이상의 제어에 ElastiCache 대한 추가 권한이 필요합니까?

질문 수준 소개: 개별 클라이언트 수준에서 ElastiCache (Redis OSS) 클러스터에 대한 액세스를 제한하거나 제어해야 하는 시나리오에서는 ElastiCache (Redis OSS) AUTH 명령을 통해 인증하는 것이 좋습니다. ElastiCache (Redis OSS) 인증 토큰은 선택적 사용자 및 사용자 그룹 관리를 통해 클라이언트가 명령 및 액세스 키를 실행하도록 허용하기 전에 ElastiCache (Redis OSS)에 암호가 필요하도록 활성화하여 데이터 영역 보안을 개선합니다.

질문 수준의 이점: 데이터를 안전하게 유지하기 위해 ElastiCache (Redis OSS)는 데이터의 무단 액세스로부터 보호하는 메커니즘을 제공합니다. 여기에는 권한 있는 명령을 수행하기 ElastiCache 전에 클라이언트가 에 연결하는 데 사용할 역할 기반 액세스 제어(RBAC) AUTH또는 AUTH 토큰(암호)을 적용하는 것이 포함됩니다.

- [Best] ElastiCache (Redis OSS) 6.x 이상의 경우 사용자 그룹, 사용자 및 액세스 문자열을 정의하여 인증 및 권한 부여 제어를 정의합니다. 사용자를 사용자 그룹에 할당한 다음, 클러스터에 사용자 그

룹을 할당합니다. 를 사용하려면 클러스터 생성 시 RBAC를 선택하고 전송 중 암호화를 활성화해야 합니다. 가 를 활용할 수 TLS 있도록 지원하는 Valkey 또는 Redis OSS 클라이언트를 사용하고 있는지 확인합니다RBAC.

[리소스]:

- RBAC에 대한 복제 그룹에 적용 ElastiCache (Redis OSS)
- 액세스 문자열을 사용하여 권한 지정
- ACL
- 지원되는 ElastiCache (Redis OSS) 버전

- [Best] 6.x 이전 ElastiCache (Redis OSS) 버전의 경우 강력한 토큰/암호를 설정하고 ElastiCache (Redis OSS) 에 대한 엄격한 암호 정책을 유지하는 것 외에도 암호/토큰을 교체하는 AUTH것이 좋습니다. 는 언제든지 최대 두(2) 개의 인증 토큰을 관리할 ElastiCache 수 있습니다. 인증 토큰 사용을 명시적으로 요구하도록 클러스터를 수정할 수도 있습니다.

  [리소스]: 기존 ElastiCache (Redis OSS) 클러스터의 AUTH 토큰 수정

SEC 3: 명령이 실수로 실행되어 데이터 손실 또는 장애가 발생할 위험이 있나요?

질문 수준 소개: 실수로 실행되거나 악의적인 공격자가 실행할 경우 작업에 부정적인 영향을 미칠 수 있는 Valkey 또는 Redis OSS 명령이 많이 있습니다. 이러한 명령은 성능 및 데이터 안전 관점에서 의도하지 않은 결과를 초래할 수 있습니다. 예를 들어 개발자는 FLUSHALL 개발 환경에서 명령을 정기적으로 호출할 수 있으며 실수로 인해 프로덕션 시스템에서 이 명령을 호출하려고 실수로 시도하여 실수로 데이터가 손실될 수 있습니다.

질문 수준 이점: ElastiCache (Redis OSS) 5.0.3부터 워크로드에 지장을 줄 수 있는 특정 명령의 이름을 바꿀 수 있습니다. 명령의 이름을 바꾸면 클러스터에서 실수로 명령이 실행되는 것을 방지할 수 있습니다.

- [필수]

  [리소스]:

  - ElastiCache (Redis OSS) 버전 5.0.3(사용되지 않음, 버전 5.0.6 사용)
  - Redis OSS 5.0.3 파라미터 변경 사항
  - Redis OSS 보안

SEC 4: 저장 데이터 암호화를 확인하려면 어떻게 해야 합니까? ElastiCache

질문 수준 소개: ElastiCache (Redis OSS)는 인 메모리 데이터 스토어이지만 클러스터의 표준 작업의 일부로 (스토리지에) 지속될 수 있는 모든 데이터를 암호화할 수 있습니다. 여기에는 Amazon S3에 기록된 정기 백업과 수동 백업뿐 아니라 동기화 및 스왑 작업의 결과로 디스크 스토리지에 저장된 데이터가 포함됩니다. M6g 및 R6g 패밀리의 인스턴스 유형에는 상시 켜져 있는 인메모리 암호화 기능도 있습니다.

질문 수준 이점: ElastiCache (Redis OSS)는 데이터 보안을 높이기 위해 저장 시 선택적 암호화를 제공합니다.

- [필수] 저장 시 암호화는 생성된 경우에만 ElastiCache 클러스터(복제 그룹)에서 활성화할 수 있습니다. 기존 클러스터를 수정하여 저장 데이터 암호화를 시작할 수 없습니다. 기본적으로 ElastiCache는 유휴 암호화에 사용되는 키를 제공하고 관리합니다.

  [리소스]:

  - 저장 시 암호화 제약 조건
  - 저장 데이터 암호화 활성화
- [최고] 메모리에 있는 동안 데이터를 암호화하는 Amazon EC2 인스턴스 유형(예: M6g 또는 R6g)을 활용합니다. 가능하면 저장 데이터 암호화를 위해 자체 키를 관리하는 것이 좋습니다. 보다 엄격한 데이터 보안 환경의 경우, AWS Key Management Service (KMS)를 사용하여 고객 마스터 키()를 자체 관리할 수 있습니다CMK. 와의 ElastiCache 통합을 통해 ElastiCache (Redis OSS) 클러스터의 저장 데이터 암호화에 사용되는 키를 생성, 소유 및 관리할 AWS Key Management Service수 있습니다.

  [리소스]:

  - 에서 고객 관리형 키 사용 AWS Key Management Service
  - AWS 키 관리 서비스
  - AWS KMS 개념

SEC 5: 를 사용하여 전송 중 데이터를 암호화하려면 어떻게 해야 하나요 ElastiCache?

질문 수준의 소개: 전송 중에 데이터가 손상되는 것을 방지하는 것은 일반적인 요구 사항입니다. 이는 분산 시스템의 구성 요소 내뿐만 아니라 애플리케이션 클라이언트와 클러스터 노드 간의 데이터를 나타냅니다. ElastiCache (Redis OSS)는 클라이언트와 클러스터 간 및 클러스터 노드 자체 간에 전송 중인 데이터를 암호화할 수 있도록 허용하여 이 요구 사항을 지원합니다. M6g 및 R6g 패밀리의 인스턴스 유형에는 상시 켜져 있는 인메모리 암호화 기능도 있습니다.

질문 수준 이점: Amazon 전송 ElastiCache 중 암호화는 한 위치에서 다른 위치로 전송 중일 때 가장 취약한 지점에서 데이터의 보안을 강화할 수 있는 선택적 기능입니다.

- [필수] 전송 중 암호화는 생성 시 ElastiCache (Redis OSS) 클러스터(복제 그룹)에서만 활성화할 수 있습니다. 데이터 암호화/복호화에 추가 처리가 필요하기 때문에 전송 중 암호화를 구현하면 성능에 어느 정도 영향을 미칠 수 있다는 점에 유의하시기 바랍니다. 영향을 이해하려면 를 활성화하기 전과 후에 워크로드를 벤치마킹하는 것이 좋습니다encryption-in-transit.

  [리소스]:

  - 전송 중 데이터 암호화 개요

SEC 6: 제어 영역 리소스에 대한 액세스를 어떻게 제한합니까?

질문 수준 소개: IAM ElastiCache (Redis OSS)에 대한 세분화된 액세스 제어를 정책 및 ARN 활성화하여 더 엄격한 제어를 통해 ElastiCache (Redis OSS) 클러스터의 생성, 수정 및 삭제를 관리할 수 있습니다.

질문 수준 이점: 복제 그룹, 노드 등과 같은 Amazon ElastiCache 리소스 관리는 IAM 정책에 따라 특정 권한이 있는 AWS 계정으로 제한되어 리소스의 보안 및 신뢰성을 개선할 수 있습니다.

- [필수] 사용자에게 특정 AWS Identity and Access Management정책을 할당하여 Amazon ElastiCache 리소스에 대한 AWS 액세스를 관리하여 클러스터에서 어떤 계정을 통해 어떤 작업을 수행할 수 있는지 더 세밀하게 제어할 수 있습니다.

  [리소스]:

  - 리소스에 ElastiCache 대한 액세스 권한 관리 개요
  - Amazon에 대한 자격 증명 기반 정책(IAM 정책) 사용 ElastiCache

SEC 7: 보안 이벤트를 어떻게 탐지하고 대응하나요?

질문 수준 소개: ElastiCache를 RBAC 활성화하여 배포하면 가 CloudWatch 지표를 내보내 사용자에게 보안 이벤트를 알립니다. 이러한 지표는 RBAC 사용자 연결 권한이 없는 인증, 액세스 키 또는 실행 명령에 실패한 시도를 식별하는 데 도움이 됩니다.

또한 AWS 제품 및 서비스 리소스는 배포를 자동화하고 이후 검토/감사를 위해 모든 작업 및 수정 사항을 로깅하여 전체 워크로드를 보호하는 데 도움이 됩니다.

질문 수준의 이점: 이벤트를 모니터링하면 조직이 요구 사항, 정책 및 절차에 따라 대응할 수 있습니다. 이러한 보안 이벤트에 대한 모니터링 및 대응을 자동화하면 전반적인 보안 태세가 강화됩니다.

- [필수] RBAC 인증 및 권한 부여 실패와 관련된 게시된 CloudWatch 지표를 숙지합니다.
  - AuthenticationFailures = Valkey 또는 Redis 인증 시도 실패 OSS
  - KeyAuthorizationFailures = 사용자가 권한 없이 키에 액세스하려는 시도 실패
  - CommandAuthorizationFailures = 권한 없이 명령을 실행하려는 사용자의 시도 실패

  [리소스]:
  - Valkey 또는 Redis에 대한 지표 OSS
- [가장 좋음] 이러한 지표에 대한 경고 및 알림을 설정하고 필요에 따라 대응하는 것이 좋습니다.

  [리소스]:
  - Amazon CloudWatch 경보 사용
- [Best] Valkey 또는 Redis OSS ACL LOG 명령을 사용하여 추가 세부 정보 수집

  [리소스]:
  - ACL LOG
- [최고] ElastiCache 배포 및 이벤트를 모니터링, 로깅 및 분석하는 것과 관련된 AWS 제품 및 서비스 기능을 숙지합니다.

  [리소스]:
  - 를 사용하여 Amazon ElastiCache API 통화 로깅 AWS CloudTrail
  - elasticache-redis-cluster-automatic-backup-check
  - CloudWatch 지표 사용 모니터링

Amazon ElastiCache Well-Architected 렌즈 신뢰성 기둥

신뢰성 요소는 의도한 기능을 수행하는 워크로드와 수요를 충족하기 위해 실패에서 빠르게 복구하는 방법에 중점을 둡니다. 주요 주제에는 분산 시스템 설계, 복구 계획 및 변화하는 요구 사항에 대한 적응이 포함됩니다.

주제

- REL 1: 고가용성(HA) 아키텍처 배포를 어떻게 지원하고 있습니까?
- REL 2: 를 사용하여 복구 시점 목표(RPOs)를 어떻게 충족하고 있나요ElastiCache?
- REL 3: 재해 복구(DR) 요구 사항을 어떻게 지원하나요?
- REL 4: 장애 조치를 효과적으로 계획하려면 어떻게 해야 하나요?
- REL 5: 구성 ElastiCache 요소가 확장되도록 설계되었습니까?

REL 1: 고가용성(HA) 아키텍처 배포를 어떻게 지원하고 있습니까?

질문 수준 소개: Amazon의 고가용성 아키텍처를 이해하면 가용성 이벤트 중에 복원력 있는 상태로 운영할 ElastiCache 수 있습니다.

질문 수준의 이점: 장애에 대한 복원력을 갖도록 ElastiCache 클러스터를 설계하면 ElastiCache 배포의 가용성을 높일 수 있습니다.

- [필수] ElastiCache 클러스터에 필요한 신뢰성 수준을 결정합니다. 완전히 일시적인 워크로드부터 미션 크리티컬 워크로드까지 워크로드마다 복원력에 대한 표준이 다릅니다. 개발, 테스트, 프로덕션 등 운영 환경의 각 유형에 대한 요구 사항을 정의합니다.

  캐싱 엔진: ElastiCache (Memcached) 대 ElastiCache (Redis OSS)

  1. ElastiCache (Memcached)는 복제 메커니즘을 제공하지 않으며 주로 임시 워크로드에 사용됩니다.
  2. ElastiCache (Redis OSS)는 아래에 설명된 HA 기능을 제공합니다.
- [최고] HA가 필요한 워크로드의 경우 샤드당 복제본이 최소 2개 이상 있는 클러스터 모드에서 ElastiCache (Redis OSS)를 사용합니다. 단 하나의 샤드만 필요한 처리량이 적은 워크로드의 경우에도 마찬가지입니다.
  1. 클러스터 모드를 활성화하면 다중 AZ가 자동으로 활성화됩니다.

     다중 AZ는 계획되거나 계획되지 않은 유지 관리 시 기본 노드에서 복제본으로 자동 장애 조치를 수행하고 AZ 장애를 완화하여 가동 중지 시간을 최소화합니다.
  2. 샤딩된 워크로드의 경우 Valkey 또는 Redis OSS 클러스터 프로토콜에 따라 쿼럼을 달성하기 위해 대부분의 프라이머리 노드를 사용할 수 있어야 하므로 최소 3개의 샤드가 장애 조치 이벤트 중에 더 빠른 복구를 제공합니다.
  3. 가용성 전체에서 두 개 이상의 복제본을 설정합니다.

     복제본이 두 개 있으면 읽기 확장성이 향상되고 하나의 복제본이 유지 관리되는 시나리오에서도 읽기 가용성이 향상됩니다.
  4. Graviton2 기반 노드 유형(대부분 리전의 기본 노드)을 사용합니다.

     ElastiCache (Redis OSS)는 이러한 노드에서 최적화된 성능을 추가했습니다. 따라서 복제 및 동기화 성능이 향상되어 전반적인 가용성이 향상됩니다.
  5. 예상 트래픽 피크를 처리하기 위해 모니터링 및 적절한 크기: 과부하 시 ElastiCache (Redis OSS) 엔진이 응답하지 않아 가용성에 영향을 미칠 수 있습니다. `BytesUsedForCache` `DatabaseMemoryUsagePercentage`는 메모리 사용량을 나타내는 좋은 지표인 반면

`ReplicationLag`는 쓰기 속도를 기반으로 하는 복제 상태를 나타내는 지표입니다. 이러한 지표를 사용하여 클러스터 규모 조정을 트리거할 수 있습니다.

6. 프로덕션 장애 조치 이벤트 API 전에 장애 조치로 테스트하여 클라이언트 측 복원력을 확인합니다.

[리소스]:

- 고가용성을 위해 ElastiCache (Redis OSS) 구성
- 고가용성을 위한 복제 그룹 사용

REL 2: 를 사용하여 복구 시점 목표(RPOs)를 어떻게 충족하고 있나요ElastiCache?

질문 수준 소개: 워크로드를 이해RPO하여 ElastiCache 백업 및 복구 전략에 대한 결정에 정보를 제공합니다.

질문 수준의 이점: 현재 위치 RPO 전략을 사용하면 재해 복구 시나리오 발생 시 비즈니스 연속성을 개선할 수 있습니다. 백업 및 복원 정책을 설계하면 ElastiCache 데이터의 복구 시점 목표(RPO)를 충족하는 데 도움이 될 수 있습니다. ElastiCache (Redis OSS)는 구성 가능한 보존 정책과 함께 Amazon S3에 저장되는 스냅샷 기능을 제공합니다. 이러한 스냅샷은 정의된 백업 기간 동안 생성되며 서비스에서 자동으로 처리됩니다. 워크로드에 추가적인 백업 세분화가 필요한 경우, 하루에 최대 20개의 수동 백업을 생성할 수 있습니다. 수동으로 생성한 백업에는 서비스 보존 정책이 없으며 무기한으로 보관할 수 있습니다.

- [필수] ElastiCache 배포RPO의 를 이해하고 문서화합니다.
  - Memcached는 백업 프로세스를 제공하지 않는다는 점에 유의하시기 바랍니다.
  - ElastiCache 백업 및 복원 기능의 기능을 검토합니다.
- [가장 좋음] 클러스터를 백업하기 위해 소통과 합의를 통해 프로세스를 마련합니다.
  - 필요에 따라 수동 백업을 시작합니다.
  - 자동 백업에 대한 보존 정책을 검토합니다.
  - 수동 백업은 무기한으로 보존된다는 점에 유의하시기 바랍니다.
  - 사용량이 적은 시간대에 자동 백업을 예약하세요.
  - 읽기 전용 복제본에 대해 백업 작업을 수행하여 클러스터 성능에 미치는 영향을 최소화합니다.
- [좋음] 의 예약된 백업 기능을 활용하여 정의된 기간 동안 데이터를 ElastiCache 정기적으로 백업합니다.
  - 백업에서 정기적으로 복원을 테스트합니다.
- [리소스]:

- Redis OSS
- 에 대한 백업 및 복원 ElastiCache (RedisOSS)
- 수동 백업 만들기
- 자동 백업 예약
- 백업 및 복원 ElastiCache (RedisOSS) 클러스터

REL 3: 재해 복구(DR) 요구 사항을 어떻게 지원하나요?

질문 수준 소개: 재해 복구는 모든 워크로드 계획의 중요한 요소입니다. ElastiCache (Redis OSS)는 워크로드 복원력 요구 사항에 따라 재해 복구를 구현할 수 있는 여러 옵션을 제공합니다. Amazon ElastiCache Global Datastore를 사용하면 한 리전의 ElastiCache (Redis OSS) 클러스터에 쓸 수 있고 다른 두 리전 간 복제본 클러스터에서 데이터를 읽을 수 있으므로 리전 간 지연 시간이 짧은 읽기 및 재해 복구가 가능합니다.

질문 수준의 이점: 다양한 재해 시나리오를 이해하고 계획하면 비즈니스 연속성을 보장할 수 있습니다. DR 전략은 비용, 성능에 미치는 영향 및 데이터 손실 가능성 사이에서 균형을 맞춰야 합니다.

- [필수] 워크로드 요구 사항에 따라 모든 ElastiCache 구성 요소에 대한 DR 전략을 개발하고 문서화합니다. ElastiCache 는 일부 사용 사례가 전적으로 임시적이고 DR 전략이 필요하지 않은 반면, 다른 사용 사례는 스펙트럼의 반대편에 있으며 매우 강력한 DR 전략이 필요하다는 점에서 고유합니다. 모든 옵션은 비용 최적화와 비교하여 평가해야 합니다. 복원력을 높이려면 더 많은 양의 인프라가 필요합니다.

  리전 수준 및 다중 리전 수준에서 사용할 수 있는 DR 옵션을 이해합니다.
  - AZ 장애를 방지하려면 다중 AZ 배포를 권장합니다. 다중 AZ 아키텍처에서 클러스터 모드가 활성화된 상태에서 최소 3개를 사용할 AZs 수 있는 상태로 배포해야 합니다.
  - 리전 장애를 방지하려면 글로벌 데이터 스토어를 사용하는 것이 좋습니다.
- [가장 좋음] 리전 수준의 복원력이 필요한 워크로드에 글로벌 데이터 스토어를 활성화합니다.
  - 기본 리전에서 성능 저하가 발생할 경우 보조 리전으로 장애 조치를 수행할 계획을 세웁니다.
  - 프로덕션 환경에서 장애 조치를 수행하기 전에 다중 리전 장애 조치 프로세스를 테스트합니다.
  - `ReplicationLag` 지표를 모니터링하여 장애 조치 이벤트 중 데이터 손실의 잠재적 영향을 파악합니다.
- [리소스]:
  - 장애 완화
  - 글로벌 데이터 스토어를 사용하여 AWS 리전 간 복제

- 선택적으로 클러스터 크기를 조정하여 백업에서 복원
- 다중 AZ를 사용하여 ElastiCache (Redis OSS)의 가동 중지 시간 최소화

REL 4: 장애 조치를 효과적으로 계획하려면 어떻게 해야 하나요?

질문 수준 소개: 자동 장애 조치로 다중 AZ를 활성화하는 것이 ElastiCache 모범 사례입니다. 경우에 따라 ElastiCache (Redis OSS)가 서비스 작업의 일부로 기본 노드를 대체합니다. 예를 들면 계획된 유지 관리 이벤트 및 드물지만 노드 장애나 가용 영역 문제가 발생하는 경우가 포함됩니다. 성공적인 장애 조치는 ElastiCache 및 클라이언트 라이브러리 구성 모두에 의존합니다.

질문 수준 이점: 특정 ElastiCache (Redis OSS) 클라이언트 라이브러리와 함께 ElastiCache 장애 조치에 대한 모범 사례를 따르면 장애 조치 이벤트 중에 발생할 수 있는 가동 중지 시간을 최소화하는 데 도움이 됩니다.

- [필수] 클러스터 모드가 비활성화된 경우, 클라이언트가 이전 프라이머리 노드와의 연결을 끊고 업데이트된 기본 엔드포인트 IP 주소를 사용하여 새 프라이머리 노드에 다시 연결해야 하는지 감지하도록 제한 시간을 사용합니다. 클러스터 모드가 활성화된 경우, 클라이언트 라이브러리가 기본 클러스터 토폴로지의 변경 사항을 감지합니다. 이는 ElastiCache (Redis OSS) 클라이언트 라이브러리의 구성 설정에 의해 가장 자주 수행되며, 이를 통해 빈도와 새로 고침 방법을 구성할 수도 있습니다. 각 클라이언트 라이브러리는 자체 설정을 제공하며 자세한 내용은 해당 설명서에서 확인할 수 있습니다.

  [리소스]:

  - 다중 AZ를 사용하여 ElastiCache (Redis OSS)의 가동 중지 시간 최소화
  - ElastiCache (Redis OSS) 클라이언트 라이브러리의 모범 사례를 검토합니다.
- [필수] 성공적인 장애 조치는 프라이머리 노드와 복제본 노드 간의 정상적인 복제 환경에 달려 있습니다. Valkey 및 Redis OSS 복제의 비동기 특성과 기본 노드와 복제 노드 간의 복제 지연을 보고하는 데 사용할 수 있는 CloudWatch 지표를 검토하고 이해합니다. 데이터 안전이 더 필요한 사용 사례의 경우 WAIT 명령을 활용하여 연결된 클라이언트에 응답하기 전에 복제본이 쓰기를 승인하도록 강제합니다.

  [리소스]:

  - Valkey 또는 Redis에 대한 지표 OSS
  - Amazon을 사용하여 ElastiCache (Redis OSS)를 사용하여 모범 사례 모니터링 CloudWatch
- [최고] 장애 조치 테스트 를 사용하여 ElastiCache 장애 조치 중에 애플리케이션의 응답성을 정기적으로 검증합니다API.

  [리소스]:

- Amazon에서 읽기 전용 복제본에 대한 자동 장애 조치 테스트 ElastiCache (RedisOSS)
- 자동 장애 조치 테스트

REL 5: 구성 ElastiCache 요소가 확장되도록 설계되었습니까?

질문 수준 소개: 확장 기능과 사용 가능한 배포 토폴로지를 이해하면 구성 요소는 변화하는 워크로드 요구 사항을 충족하도록 시간이 지남에 따라 조정할 수 있습니다 ElastiCache. ElastiCache 는 인/아웃(수평) 및 업/다운(수직)의 4방향 확장을 제공합니다.

질문 수준의 이점: ElastiCache 배포 모범 사례를 따르면 최대의 조정 유연성을 제공할 뿐만 아니라 장애의 영향을 최소화하기 위해 Well Architected 수평 조정 원칙을 충족할 수 있습니다.

- [필수] 클러스터 모드 활성화 토폴로지와 클러스터 모드 비활성화 토폴로지의 차이점을 이해합니다. 클러스터 모드를 활성화하면 시간이 지남에 따라 확장성 향상이 가능하므로 대부분의 경우에 클러스터 모드를 활성화하여 배포하는 것이 좋습니다. 클러스터 모드가 비활성화된 구성 요소는 읽기 전용 복제본을 추가하여 수평적으로 확장할 수 있는 기능이 제한됩니다.
- [필수] 규모 조정 시기 및 방법을 이해합니다.
  - 자세한 내용READIOPS: 복제본 추가
  - 자세한 내용WRITEOPS: 샤드 추가(스케일 아웃)
  - 네트워크 I/O 향상: 네트워크에 최적화된 인스턴스 사용, 스케일 업
- [최고] 클러스터 모드가 활성화된 구성 ElastiCache 요소를 배포합니다. 더 적고 더 큰 노드가 아닌 더 크고 작은 노드에 편향됩니다. 이는 노드 장애가 영향을 미치는 범위를 효과적으로 제한합니다.
- [가장 좋음] 규모 조정 이벤트 시 응답성을 높이기 위해 클러스터에 복제본을 포함합니다.
- [좋음] 클러스터 모드가 비활성화된 경우 읽기 전용 복제본을 활용하여 전체 읽기 용량을 늘립니다. ElastiCache 는 클러스터 모드가 비활성화된 상태에서 최대 5개의 읽기 전용 복제본과 수직 조정을 지원합니다.
- [리소스]:
  - 스케일링ElastiCache (RedisOSS) 클러스터
  - 온라인 스케일 업
  - ElastiCache Memcached 클러스터의 크기 조정

Amazon ElastiCache Well-Architected 렌즈 성능 효율성 기둥

성능 효율성 요소는 IT 및 컴퓨팅 리소스를 효율적으로 사용하는 데 중점을 둡니다. 주요 주제로는 워크로드 요구 사항을 기반으로 적절한 리소스 유형 및 크기 선택하기, 성능 모니터링하기, 비즈니스 요구 사항이 변화함에 따라 효율성을 유지하기 위해 정보에 입각하여 의사 결정 내리기가 있습니다.

주제

PE 1: Amazon ElastiCache 클러스터의 성능을 어떻게 모니터링하나요?

질문 수준의 소개: 기존 모니터링 지표를 이해하면 현재 사용률을 파악할 수 있습니다. 적절한 모니터링은 클러스터 성능에 영향을 미치는 잠재적 병목 현상을 식별하는 데 도움이 될 수 있습니다.

질문 수준의 이점: 클러스터와 관련된 지표를 이해하면 지연 시간을 줄이고 처리량을 높이는 데 도움이 되는 최적화 기법을 익힐 수 있습니다.

- [필수] 워크로드의 일부를 사용하여 기준 성능을 테스트합니다.
  - 로드 테스트와 같은 메커니즘을 사용하여 실제 워크로드의 성능을 모니터링해야 합니다.
  - 이러한 테스트를 실행하는 동안 CloudWatch 지표를 모니터링하여 사용 가능한 지표를 이해하고 성능 기준을 설정합니다.
- [Best] ElastiCache (Redis OSS) 워크로드의 경우 프로덕션 클러스터에서 차단 명령을 실행할 수 있는 사용자의 기능을 제한KEYS하려면 와 같이 계산 비용이 많이 드는 명령의 이름을 변경합니다.
  - ElastiCache 엔진 6.x를 실행하는 (Redis OSS) 워크로드는 역할 기반 액세스 제어를 활용하여 특정 명령을 제한할 수 있습니다. AWS 콘솔 또는 를 사용하여 사용자 및 사용자 그룹을 생성하고 사용자 그룹을 ElastiCache (Redis OSS) 클러스터에 CLI연결하여 명령에 대한 액세스를 제어할 수 있습니다. Redis OSS 6에서 RBAC 이 활성화되면 '-@위험'을 사용할 수 있으며 해당 사용자에 대해 KEYS, MONITORSORT, 등과 같은 값비싼 명령은 허용되지 않습니다.

  - 엔진 버전 5.x의 경우 ElastiCache (Redis OSS) 클러스터 `rename-commands` 파라미터 그룹의 파라미터를 사용하여 명령의 이름을 변경합니다.
- [더 좋음] 느린 쿼리를 분석하고 최적화 기법을 찾아봅니다.
  - ElastiCache (Redis OSS) 워크로드의 경우 느린 로그를 분석하여 쿼리에 대해 자세히 알아봅니다. 예를 들어 `valkey-cli slowlog get 10` 명령을 사용하여 지연 시간 임계값(기본값 10초)을 초과한 마지막 10개의 명령을 표시할 수 있습니다.
  - 복잡한 ElastiCache (Redis OSS) 데이터 구조를 사용하여 특정 쿼리를 더 효율적으로 수행할 수 있습니다. 예를 들어, 숫자 스타일 범위 조회의 경우 애플리케이션은 정렬된 세트를 사용하여 간단한 숫자 인덱스를 구현할 수 있습니다. 이러한 인덱스를 관리하면 데이터 세트에 대해 수행되는 스캔을 줄이고 더 높은 성능 효율성으로 데이터를 반환할 수 있습니다.
  - ElastiCache (Redis OSS) 워크로드의 경우 는 클라이언트 수 및 데이터 크기와 같은 사용자 정의 입력을 사용하여 다양한 명령의 성능을 테스트하기 위한 간단한 인터페이스를 `redis-benchmark` 제공합니다.
  - Memcached는 간단한 키 수준 명령만 지원하므로 클라이언트 쿼리를 처리하기 위해 키 공간을 반복하지 않도록 추가 키를 인덱스로 구축하는 것이 좋습니다.
- [리소스]:
  - CloudWatch 지표 사용 모니터링
  - Amazon CloudWatch 경보 사용
  - Valkey 및 Redis OSS 관련 파라미터
  - SLOWLOG
  - 벤치마크

PE 2: ElastiCache 클러스터 노드에 작업을 어떻게 배포하고 있나요?

질문 수준 소개: 애플리케이션이 Amazon ElastiCache 노드에 연결하는 방식은 클러스터의 성능과 확장성에 영향을 미칠 수 있습니다.

질문 수준의 이점: 클러스터에서 사용 가능한 노드를 적절히 사용하면 사용 가능한 리소스 전체에 작업이 분산됩니다. 다음 기법은 유휴 리소스를 방지하는 데도 도움이 됩니다.

- [필수] 클라이언트가 적절한 ElastiCache 엔드포인트에 연결하도록 합니다.
  - ElastiCache (Redis OSS)는 사용 중인 클러스터 모드를 기반으로 다양한 엔드포인트를 구현합니다. 클러스터 모드가 활성화된 경우 ElastiCache 는 구성 엔드포인트를 제공합니다. 클러스터 모드가 비활성화된 경우 는 일반적으로 쓰기에 사용되는 기본 엔드포인트와 복제본 간에 읽기의 균형을 맞추기 위한 리더 엔드포인트를 ElastiCache 제공합니다. 이러한 엔드포인트를 올바르게 구현

하면 성능이 향상되고 확장 작업이 더 쉬워집니다. 특별한 요구 사항이 없는 한 개별 노드 엔드포인트에 연결하지 마시기 바랍니다.

  - 다중 노드 Memcached 클러스터의 경우 Auto Discovery를 활성화하는 구성 엔드포인트를 ElastiCache 제공합니다. 캐시 노드 전체에 작업을 균등하게 분배하려면 해싱 알고리즘을 사용하는 것이 좋습니다. 많은 Memcached 클라이언트 라이브러리는 일관된 해싱을 구현합니다. 사용할 라이브러리의 설명서에서 일관적 해싱 지원 여부와 그 구현 방법을 참조하세요. 이러한 기능 구현에 대한 자세한 내용은 여기에서 확인할 수 있습니다.
- [더 좋음] 확장성을 개선하기 위해 활성화된 ElastiCache (Redis OSS) 클러스터 모드를 활용합니다.
  - ElastiCache (Redis OSS) (클러스터 모드 활성화됨) 클러스터는 온라인 조정 작업(out/in 및 up/down)을 지원하여 샤드 간에 데이터를 동적으로 분산하는 데 도움이 됩니다. 구성 엔드포인트를 사용하면 클러스터 인식 클라이언트가 클러스터 토폴로지의 변화에 맞게 조정할 수 있습니다.
  - (Redis OSS) ElastiCache (클러스터 모드 활성화됨) 클러스터에서 사용 가능한 샤드 간에 해시슬롯을 이동하여 클러스터의 균형을 재조정할 수도 있습니다. 이렇게 하면 사용 가능한 샤드에 작업을 더 효율적으로 분배할 수 있습니다.
- [더 좋음] 워크로드의 단축키를 식별하고 정정하기 위한 전략을 구현합니다.
  - 목록, 스트림, 세트 등과 같은 다차원 Valkey 또는 Redis OSS 데이터 구조의 영향을 고려합니다. 이러한 데이터 구조는 단일 노드에 있는 단일 키에 저장됩니다. 매우 큰 다차원 키는 다른 데이터 유형보다 더 많은 네트워크 용량과 메모리를 사용할 가능성이 있으며, 이로 인해 해당 노드가 불균형하게 사용될 수 있습니다. 가능하면 여러 개별 키로 데이터 액세스를 분산하도록 워크로드를 설계하세요.
  - 워크로드의 핫키는 사용 중인 노드의 성능에 영향을 미칠 수 있습니다. ElastiCache (Redis OSS) 워크로드의 경우 LFU 최대 메모리 정책이 있는지 `valkey-cli --hotkeys` 여부를 사용하여 핫키를 감지할 수 있습니다.
  - 여러 노드에 핫키를 복제하여 액세스를 더 균등하게 분산하는 것을 고려해 보세요. 이 접근 방식을 사용하려면 클라이언트가 여러 기본 노드에 쓸 것(Valkey 또는 Redis OSS 노드 자체는 이 기능을 제공하지 않음)과 원본 키 이름 외에도 읽을 키 이름 목록을 유지해야 합니다.
  - ElastiCache Valkey 7.2 이상 및 Redis OSS 버전 6 이상에서는 서버 지원 클라이언트 측 캐싱을 지원합니다. 이렇게 하면 애플리케이션이 로 다시 네트워크 호출하기 전에 키가 변경될 때까지 기다릴 수 있습니다ElastiCache.
- [리소스]:
  - 더 높은 가용성을 OSS 위해 Valkey 및 Redis ElastiCache 로 구성
  - 에서 연결 엔드포인트 찾기 ElastiCache
  - 로드 밸런싱 모범 사례
  - Valkey 또는 Redis에 대한 온라인 리샤딩OSS(클러스터 모드 활성화됨)

- Valkey 및 Redis의 클라이언트 측 캐싱 OSS

PE 3: 캐싱 워크로드의 경우, 캐시의 효율성과 성능을 어떻게 추적하고 보고하나요?

질문 수준 소개: 캐싱은 에서 일반적으로 발생하는 워크로드 ElastiCache 이며 캐시의 효율성과 성능을 관리하는 방법을 이해하는 것이 중요합니다.

질문 수준의 이점: 애플리케이션의 성능이 저하되었다는 징후가 보일 수 있습니다. 캐시별 지표를 사용하여 앱 성능을 높이는 방법을 결정하는 것은 캐시 워크로드에 매우 중요합니다.

- [필수] 시간에 따른 캐시 적중률을 측정하고 추적합니다. 캐시의 효율성은 '캐시 적중률'로 결정됩니다. 캐시 적중률이란 총 키 적중 수를 총 적중 수와 누락 수로 나눈 값을 말합니다. 비율이 1에 가까울수록 캐시의 효율성이 높은 것입니다. 캐시 적중률이 낮은 이유는 캐시 누락의 수가 많기 때문입니다. 요청된 키를 캐시에서 찾을 수 없을 때 캐시 누락이 발생합니다. 키가 캐시에 없는 이유는 키가 제거 또는 삭제되었거나, 만료되었거나, 존재한 적이 없기 때문입니다. 키가 캐시에 없는 이유를 이해하고 키를 캐시에 포함하는 데 적절한 전략을 개발하세요.

  [리소스]:

  - Valkey 및 Redis에 대한 지표 OSS
- [필수] 지연 시간 및 CPU 사용률 값과 함께 애플리케이션 캐시 성능을 측정하고 수집하여 또는 다른 애플리케이션 구성 요소를 조정 time-to-live해야 하는지 여부를 파악합니다. 는 각 데이터 구조에 대해 집계된 지연 시간에 대한 CloudWatch 지표 세트를 ElastiCache 제공합니다. 이러한 지연 시간 지표는 ElastiCache (Redis OSS) INFO 명령의 명령 통계를 사용하여 계산되며 네트워크 및 I/O 시간은 포함하지 않습니다. 이는 ElastiCache (Redis OSS)에서 작업을 처리하는 데 사용한 시간입니다.

  [리소스]:

  - Valkey 및 Redis에 대한 지표 OSS
  - Amazon을 사용하여 ElastiCache (Redis OSS)를 사용하여 모범 사례 모니터링 CloudWatch
- [가장 좋음] 필요에 맞는 캐싱 전략을 선택합니다. 캐시 적중률이 낮은 이유는 캐시 누락의 수가 많기 때문입니다. 워크로드가 캐시 누락이 적도록 설계된 경우(예: 실시간 통신), 캐싱 전략을 검토하고 메모리 및 성능 측정을 위한 쿼리 계측과 같이 워크로드에 가장 적합한 방법을 적용하는 것이 가장 좋습니다. 캐시를 채우고 유지 관리하기 위해 사용하는 실제 전략은 클라이언트가 캐싱해야 하는 데이터의 유형과 해당 데이터에 대한 액세스 패턴에 따라 달라집니다. 예를 들어 스트리밍 애플리케이션의 맞춤형 추천과 최신 뉴스 기사 모두에 동일한 전략을 사용할 가능성은 거의 없습니다.

  [리소스]:

  - Memcached에 대한 캐싱 전략

- 캐싱 모범 사례
- Amazon ElastiCache 백서를 사용한 대규모 성능

PE 4: 워크로드가 네트워킹 리소스 및 연결 사용을 어떻게 최적화하나요?

질문 수준 소개: ElastiCache (Redis OSS) 및 ElastiCache (Memcached)는 많은 애플리케이션 클라이언트에서 지원되며 구현은 다를 수 있습니다. 성능에 미치는 잠재적인 영향을 분석하려면 현재 사용 중인 네트워킹 및 연결 관리 방법을 이해해야 합니다.

질문 수준의 이점: 네트워킹 리소스를 효율적으로 사용하면 클러스터의 성능 효율성을 높일 수 있습니다. 다음 권장 사항을 적용하면 네트워킹 수요를 줄이고 클러스터 지연 시간 및 처리량을 개선할 수 있습니다.

- [필수] ElastiCache 클러스터에 대한 연결을 사전 예방적으로 관리합니다.
  - 애플리케이션의 연결 풀링은 연결을 열고 닫을 때 생성되는 클러스터의 오버헤드를 줄입니다. `CurrConnections` 및 를 CloudWatch 사용하여 Amazon의 연결 동작을 모니터링합니다`NewConnections`.
  - 상황에 따라 클라이언트 연결을 제대로 닫아 연결 누수를 방지합니다. 연결 관리 전략에는 사용하지 않는 연결을 올바르게 닫고 연결 시간 제한을 설정하는 것이 포함됩니다.
  - Memcached 워크로드의 경우, `memcached_connections_overhead`라는 연결 처리를 위해 예약된 메모리의 양을 구성할 수 있습니다.
- [더 좋음] 대용량 객체를 압축하여 메모리를 줄이고 네트워크 처리량을 개선합니다.
  - 데이터를 압축하면 필요한 네트워크 처리량(Gbps)을 줄일 수 있지만 애플리케이션에서 데이터를 압축 및 압축 해제하는 작업량이 늘어날 수 있습니다.
  - 압축은 키가 소비하는 메모리의 양도 줄여줍니다.
  - 애플리케이션 요구 사항에 따라 압축률과 압축 속도 간의 균형을 고려하세요.
- [리소스]:
  - ElastiCache (Redis OSS) - 글로벌 데이터 스토어
  - Memcached 특정 파라미터
  - ElastiCache (Redis OSS) 5.0.3은 I/O 처리를 개선하여 성능을 높입니다.
  - Valkey 및 Redis에 대한 지표 OSS
  - 고가용성을 위해 ElastiCache (Redis OSS) 구성

PE 5: 키 삭제 또는 제거를 어떻게 관리하나요?

질문 수준 소개: 워크로드는 요구 사항이 다르며 클러스터 노드가 메모리 소비 제한에 가까워질 때 예상되는 동작이 다릅니다. ElastiCache (Redis OSS)에는 이러한 상황을 처리하기 위한 다양한 정책이 있습니다.

질문 수준의 이점: 사용 가능한 메모리를 적절히 관리하고 제거 정책을 이해하면 인스턴스 메모리 제한을 초과했을 때 클러스터 동작을 인식하는 데 도움이 됩니다.

- [필수] 데이터 액세스를 계측하여 적용할 정책을 평가합니다. 클러스터에서 제거를 수행할지, 수행한다면 어떻게 수행할지 제어하는 데 적합한 최대 메모리 정책을 식별합니다.
  - 제거는 클러스터에서 최대 메모리가 소비되고 제거를 허용하는 정책이 있을 때 발생합니다. 이 상황에서 클러스터의 동작은 지정된 제거 정책에 따라 달라집니다. 이 정책은 ElastiCache (Redis OSS) 클러스터 파라미터 그룹에서 `maxmemory-policy`를 사용하여 관리할 수 있습니다.
  - 기본 정책은 만료 시간(TTL 값)이 설정된 키를 제거하여 메모리를 `volatile-lru` 확보합니다. 가장 자주 사용하지 않는 (LFU) 및 가장 최근에 사용하지 않는 (LRU) 정책은 사용량에 따라 키를 제거합니다.
  - Memcached 워크로드의 경우 각 노드에서 제거를 제어하는 기본 LRU 정책이 마련되어 있습니다. Amazon ElastiCache 클러스터의 퇴거 수는 Amazon 의 퇴거 지표를 사용하여 모니터링할 수 있습니다 CloudWatch.
- [더 좋음] 삭제 동작을 표준화하여 클러스터의 성능에 미치는 영향을 제어함으로써 예상치 못한 성능 병목 현상을 방지합니다.
  - ElastiCache (Redis OSS) 워크로드의 경우 클러스터에서 키를 명시적으로 제거할 때 `UNLINK`는 `DEL`다음과 같습니다. 지정된 키를 제거합니다. 그러나 이 명령은 다른 스레드에서 실제 메모리 회수를 수행하므로 `DEL`과는 달리 차단하지 않습니다. 실제 제거는 나중에 비동기적으로 수행됩니다.
  - ElastiCache (Redis OSS) 6.x 워크로드의 경우 `lazyfree-lazy-user-del` 파라미터를 사용하여 파라미터 그룹에서 `DEL` 명령 동작을 수정할 수 있습니다.
- [리소스]:
  - 파라미터 그룹을 사용하여 엔진 ElastiCache 파라미터 구성
  - UNLINK
  - 를 사용한 클라우드 재무 관리 AWS

PE 6: 의 데이터를 모델링하고 상호 작용하는 방법은 무엇입니까 ElastiCache?

질문 수준 소개: ElastiCache 는 사용되는 데이터 구조와 데이터 모델에 크게 의존하지만 기본 데이터 스토어(있는 경우)도 고려해야 합니다. 사용 가능한 ElastiCache (Redis OSS) 데이터 구조를 이해하고 필요에 가장 적합한 데이터 구조를 사용하고 있는지 확인합니다.

질문 수준의 이점: 의 데이터 모델링ElastiCache 에는 애플리케이션 사용 사례, 데이터 유형 및 데이터 요소 간의 관계를 비롯한 여러 계층이 있습니다. 또한 각 ElastiCache (Redis OSS) 데이터 유형 및 명령에는 잘 문서화된 자체 성능 서명이 있습니다.

- [가장 좋음] 가장 좋음은 의도하지 않은 데이터 덮어쓰기를 줄이는 것입니다. 중복되는 키 이름을 최소화하는 이름 지정 규칙을 사용하세요. 일반적으로 데이터 구조의 이름을 지정할 때는 `APPNAME:CONTEXT:ID`, `ORDER-APP:CUSTOMER:123` 등의 계층적 방법을 사용합니다.

  [리소스]:

  - 키 이름 지정
- [Best] ElastiCache (Redis OSS) 명령에는 Big O 표기법으로 정의된 시간 복잡성이 있습니다. 이때 명령의 시간 복잡도는 그 영향을 알고리즘/수학적으로 표현한 것입니다. 애플리케이션에 새 데이터 유형을 도입할 때는 관련 명령의 시간 복잡도를 주의 깊게 검토해야 합니다. 시간 복잡도가 O(1)인 명령은 시간이 일정하며 입력 크기에 의존하지 않지만 시간 복잡도가 O(N)인 명령은 시간이 선형이며 입력 크기의 영향을 받습니다. ElastiCache (Redis OSS)의 단일 스레드 설계로 인해 시간이 많이 걸리는 작업이 많으면 성능이 저하되고 작업 제한 시간이 초과될 수 있습니다.

  [리소스]:

  - 명령
- [최고] APIs 를 사용하여 클러스터의 데이터 모델을 GUI 확인할 수 있습니다.

  [리소스]:

  - Redis OSS Commander
  - Redis OSS 브라우저
  - Redsmin

PE 7: Amazon ElastiCache 클러스터에서 실행 속도가 느린 명령을 어떻게 로깅하나요?

질문 수준의 소개: 성능 튜닝은 오래 실행되는 명령의 캡처, 집계 및 알림에 유용합니다. 명령이 실행되는 데 걸리는 시간을 이해하여 성능이 저하되는 명령과 엔진의 최적 성능을 차단하는 명령을 결정할 수

있습니다. ElastiCache (Redis OSS)에는 이 정보를 Amazon CloudWatch 또는 Amazon Kinesis Data Firehose 에 전달할 수 있는 기능도 있습니다.

질문 수준의 이점: 영구적인 전용 위치에 로깅하고 느린 명령에 대한 알림 이벤트를 제공하면 상세한 성능 분석에 도움이 되고 자동화된 이벤트를 트리거하는 데 사용할 수 있습니다.

- [필수] 엔진 버전 6.0 이상을 실행하는 Amazon ElastiCache (Redis OSS)과 올바르게 구성된 파라미터 그룹 및 클러스터에서 활성화된 SLOWLOG 로깅.
  - 필수 파라미터는 엔진 버전 호환성이 Valkey 7.2 이상 또는 Redis OSS 버전 6.0 이상으로 설정된 경우에만 사용할 수 있습니다.
  - SLOWLOG 로깅은 명령의 서버 실행 시간이 지정된 값보다 오래 걸릴 때 발생합니다. 클러스터의 동작은 관련 파라미터 그룹 파라미터(`slowlog-log-slower-than` 및 `slowlog-max-len`)에 따라 달라집니다.
  - 변경 사항은 즉시 적용됩니다.
- [최고] CloudWatch 또는 Kinesis Data Firehose 기능을 활용합니다.
  - CloudWatch, CloudWatch Logs Insights 및 Amazon Simple Notification Services의 필터링 및 경보 기능을 사용하여 성능 모니터링 및 이벤트 알림을 달성합니다.
  - Kinesis Data Firehose의 스트리밍 기능을 사용하여 SLOWLOG 로그를 영구 스토리지에 아카이브하거나 자동 클러스터 파라미터 튜닝을 트리거합니다.
  - JSON 또는 일반 TEXT 형식이 요구 사항에 가장 적합한지 확인합니다.
  - CloudWatch 또는 Kinesis Data Firehose에 게시할 수 있는 IAM 권한을 제공합니다.
- [더 좋음] 기본값이 아닌 다른 값으로 `slowlog-log-slower-than`을 구성합니다.
  - 이 파라미터는 명령이 슬로우 실행 명령으로 기록되기 전에 Valkey 또는 Redis OSS 엔진 내에서 명령이 실행되는 기간을 결정합니다. 기본값은 1만 마이크로초(10밀리초)입니다. 일부 워크로드의 경우 기본값이 너무 높을 수 있습니다.
  - 애플리케이션 요구 사항 및 테스트 결과를 기반으로 워크로드에 더 적합한 값을 결정하세요. 그러나 값이 너무 낮으면 과도한 데이터가 생성될 수 있습니다.
- [더 좋음] `slowlog-max-len`의 기본값을 그대로 둡니다.
  - 이 파라미터는 주어진 시간에 Valkey 또는 Redis OSS 메모리에서 캡처되는 느리게 실행되는 명령 수의 상한을 결정합니다. 값이 0이면 캡처가 사실상 비활성화됩니다. 값이 클수록 더 많은 항목이 메모리에 저장되므로 중요한 정보가 검토되기 전에 제거될 가능성이 줄어듭니다. 기본값은 128입니다.

- 기본값은 대부분의 워크로드에 적합합니다. SLOWLOG 명령을 통해 valkey-cli에서 확장된 기간 내에 데이터를 분석해야 하는 경우 이 값을 늘리는 것이 좋습니다. 이렇게 하면 Valkey 또는 Redis OSS 메모리에 더 많은 명령을 유지할 수 있습니다.

  CloudWatch 로그 또는 Kinesis Data Firehose로 SLOWLOG 데이터를 내보내는 경우 데이터가 유지되고 ElastiCache 시스템 외부에서 분석할 수 있으므로 Valkey 또는 Redis OSS 메모리에 느리게 실행되는 명령을 많이 저장할 필요가 없습니다.

- [리소스]:
  - ElastiCache (Redis OSS) 캐시 클러스터에서 느린 로그를 켜려면 어떻게 해야 하나요?
  - 로그 전달
  - RedisOSS별 파라미터
  - https://aws.amazon.com/cloudwatch/Amazon CloudWatch
  - Amazon Kinesis Data Firehose

PE8: Auto Scaling은 ElastiCache 클러스터의 성능을 높이는 데 어떻게 도움이 됩니까?

질문 수준 소개: Valkey 또는 Redis OSS Auto Scaling 기능을 구현하면 구성 ElastiCache 요소가 시간이 지남에 따라 조정되어 원하는 샤드 또는 복제본을 자동으로 늘리거나 줄일 수 있습니다. 이는 대상 추적 또는 예약된 규모 조정 정책을 구현하여 수행할 수 있습니다.

질문 수준의 이점: 워크로드의 급증에 대한 이해 및 계획을 통해 캐싱 성능과 비즈니스 연속성을 향상시킬 수 있습니다. ElastiCache (Redis OSS) Auto Scaling은 CPU/Memory 사용률을 지속적으로 모니터링하여 클러스터가 원하는 성능 수준에서 작동하는지 확인합니다.

- [필수] ElastiCache (Redis)에 대한 클러스터를 시작할 때OSS:
  1. 클러스터 모드가 활성화되었는지 확인합니다.
  2. 인스턴스가 Auto Scaling을 지원하는 특정 유형 및 크기의 패밀리에 속하는지 확인합니다.
  3. 클러스터가 글로벌 데이터 스토어, Outposts 또는 로컬 영역에서 실행되고 있지 않은지 확인합니다.

  [리소스]:
  - Valkey 및 Redis에서 클러스터 크기 조정OSS(클러스터 모드 활성화됨)
  - 샤드에 Auto Scaling 사용
  - 복제본에 Auto Scaling 사용

- [가장 좋음] 워크로드에 읽기 작업이 많은지, 쓰기 작업이 많은지 파악하여 규모 조정 정책을 정의합니다. 최상의 성능을 위해서는 하나의 추적 지표만 사용하세요. Auto Scaling 정책은 목표에 도달하면 스케일 아웃이 수행되지만 모든 대상 추적 정책이 스케일 인 준비가 된 경우에만 스케일 인이 수행되므로 각 차원에 대해 여러 정책을 사용하지 않는 것이 좋습니다.

  [리소스]:

  - Auto Scaling 정책
  - 조정 정책 정의
- [가장 좋음] 시간 경과에 따른 성능 모니터링은 특정 시점에 모니터링할 경우 눈에 띄지 않는 워크로드 변경을 감지하는 데 도움이 될 수 있습니다. 4주 기간 동안 클러스터 사용률에 대한 해당 CloudWatch 지표를 분석하여 목표 값 임계값을 결정할 수 있습니다. 어떤 값을 선택할지 잘 모르는 경우 지원되는 최소 사전 정의 지표 값으로 시작하는 것이 좋습니다.

  [리소스]:

  - CloudWatch 지표 사용 모니터링
- [더 좋음] 클러스터가 규모 조정 정책을 개발하고 가용성 문제를 완화하는 데 필요한 샤드/복제본의 정확한 수를 파악하기 위해 예상되는 최소 및 최대 워크로드로 애플리케이션을 테스트하는 것이 좋습니다.

  [리소스]:

  - 확장 가능 목표 등록
  - 를 사용하여 확장 가능 대상 등록 AWS CLI

Amazon ElastiCache Well-Architected Lens Cost Optimization Pillar

비용 최적화 요소는 불필요한 비용을 피하는 데 중점을 둡니다. 주요 주제로는 자금이 어디에 사용되는지 이해하고 제어하기, 가장 적합한 노드 유형 선택하기(워크로드 요구 사항에 따라 데이터 계층화를 지원하는 인스턴스 사용), 적절한 수의 리소스 유형(읽기 전용 복제본 수), 시간 경과에 따른 지출 분석하기, 과도한 지출 없이 비즈니스 요구 사항을 충족할 수 있도록 확장하기 등이 있습니다.

주제

- COST 1: 리소스와 관련된 ElastiCache 비용을 어떻게 식별하고 추적하나요? 사용자가 리소스를 생성하고 생성된 리소스를 관리 및 폐기할 수 있도록 하는 메커니즘을 어떻게 개발하나요?
- COST 2: 지속적인 모니터링 도구를 사용하여 ElastiCache 리소스와 관련된 비용을 최적화하려면 어떻게 해야 하나요?

- COST 3: 데이터 계층화를 지원하는 인스턴스 유형을 사용해야 합니까? 데이터 계층화의 이점은 무엇인가요? 데이터 계층화 인스턴스를 사용하지 않는 경우는 언제인가요?

COST 1: 리소스와 관련된 ElastiCache 비용을 어떻게 식별하고 추적하나요? 사용자가 리소스를 생성하고 생성된 리소스를 관리 및 폐기할 수 있도록 하는 메커니즘을 어떻게 개발하나요?

질문 수준의 소개: 비용 지표를 이해하려면 소프트웨어 엔지니어링, 데이터 관리, 제품 소유자, 재무 및 리더십 등 여러 팀의 참여와 협업이 필요합니다. 비용의 주요 동인을 식별하려면 모든 관련 당사자가 서비스 사용 제어 레버와 비용 관리의 절충점을 이해해야 하며, 이것이 비용 최적화 노력의 성공을 좌우하는 경우가 많습니다. 개발에서 생산 및 폐기에 이르기까지 생성된 리소스를 추적하는 프로세스와 도구를 마련하면 와 관련된 비용을 관리하는 데 도움이 됩니다ElastiCache.

질문 수준의 이점: 워크로드와 관련된 모든 비용을 지속적으로 추적하려면 가 구성 요소 중 ElastiCache 하나로 포함된 아키텍처를 깊이 이해해야 합니다. 또한 사용량을 집계하여 예산과 비교할 수 있는 비용 관리 계획도 마련해 두어야 합니다.

- [필수] 조직의 ElastiCache 사용량에 대한 지표를 정의하고 추적하고 조치를 취할 수 있는 설립 전세기 중 하나를 사용하여 Cloud Center of Excellence(CCoE)를 도입합니다. 및 함수가 CCoE 있는 경우 와 관련된 비용을 읽고 추적하는 방법을 알고 있는지 확인합니다 ElastiCache. 리소스가 생성되면 IAM 역할 및 정책을 사용하여 특정 팀 및 그룹만 리소스를 인스턴스화할 수 있는지 확인합니다. 이를 통해 비용을 비즈니스 성과와 연관시키고 비용 관점에서 명확한 책임 범위를 설정할 수 있습니다.
  1. CCoE 는 다음과 같은 범주형 데이터 전반의 키 ElastiCache 사용에 대해 정기적으로 업데이트되는 비용 지표를 식별, 정의 및 게시해야 합니다.
     a. 사용된 노드 유형 및 속성: 표준 또는 메모리 최적화, 온디맨드 또는 예약 인스턴스, 리전 및 가용 영역
     b. 환경 유형: 무료, 개발, 테스트 및 프로덕션
     c. 백업 스토리지 및 보존 전략
     d. 리전 내 및 리전 간 데이터 전송
     e. Amazon Outposts에서 실행되는 인스턴스
  2. CCoE 는 조직의 소프트웨어 엔지니어링, 데이터 관리, 제품 팀, 재무 및 리더십 팀에서 비독점적으로 대리하는 교차 기능 팀으로 구성됩니다.

  [리소스]:

- 클라우드 혁신 센터 만들기

  - Amazon ElastiCache 요금
- [필수] 비용 할당 태그를 사용하여 낮은 수준의 세밀함으로 비용을 추적합니다. AWS Cost Management를 사용하여 시간 경과에 따른 AWS 비용 및 사용량을 시각화, 이해 및 관리할 수 있습니다.

  1. 태그를 사용하여 리소스를 구성하고 비용 할당 태그를 사용하여 세부 수준에서 AWS 비용을 추적할 수 있습니다. 비용 할당 태그를 활성화한 후 는 비용 할당 태그를 AWS 사용하여 비용 할당 보고서에서 리소스 비용을 구성하고 AWS 비용을 쉽게 분류하고 추적할 수 있습니다. AWS 는 AWS 생성된 태그와 사용자 정의 태그라는 두 가지 유형의 비용 할당 태그를 제공합니다. 는 생성된 태그를 AWS 정의, 생성 및 적용 AWS 하고 사용자 정의 태그를 정의, 생성 및 적용합니다. 두 유형의 태그 모두 개별적으로 활성화해야만 Cost Management나 비용 할당 보고서에 표시됩니다.
  2. 비용 할당 태그를 사용하여 자체 비용 구조를 반영하도록 AWS 청구서를 구성합니다. Amazon 의 리소스에 비용 할당 태그를 추가 ElastiCache하면 리소스 태그 값을 기준으로 송장의 비용을 그룹화하여 비용을 추적할 수 있습니다. 또한 태그를 결합하여 보다 세부적인 수준으로 비용을 추적하는 것을 고려해야 합니다.

  [리소스]:

  - AWS 비용 할당 태그 사용
  - 비용 할당 태그를 사용한 비용 모니터링
  - AWS Cost Explorer
- [최고] ElastiCache 비용을 조직 전체에서 도달하는 지표에 연결합니다.

  1. 비즈니스 지표와 지연 시간 같은 운영 지표를 고려해 보세요. 비즈니스 모델에서 모든 역할이 이해할 수 있는 개념은 무엇인가요? 지표는 조직 내에서 가능한 많은 역할이 이해할 수 있어야 합니다.
  2. 예 - 동시에 서비스되는 사용자, 작업 및 사용자당 최대 및 평균 지연 시간, 사용자 참여 점수, 사용자 재방문율/주, 세션 길이/사용자, 포기율, 캐시 적중률, 키 추적

  [리소스]:

  - CloudWatch 지표를 사용한 사용 모니터링
- [좋음] 를 사용하는 전체 워크로드의 지표 및 비용에 대한 아키텍처 및 운영 가시성을 유지합니다 up-to-date ElastiCache.

  1. 클라이언트에서 API Gateway, Redshift 및 보고 도구(예:)에 이르기까지 기술 세트의 전체 AWS 서비스 에코시스템에 속하는 ElastiCache 경향이 있는 전체 솔루션 에코시스템 QuickSight 을 이해합니다.

2. 클라이언트, 연결, 보안, 인메모리 작업, 스토리지, 리소스 자동화, 데이터 액세스 및 관리 등 솔루션의 구성 요소를 아키텍처 다이어그램에 매핑합니다. 각 계층은 전체 솔루션에 연결되며 전체 비용에 추가되거나 전체 비용을 관리하는 데 도움이 되는 고유한 요구 사항과 기능을 가지고 있습니다.
3. 다이어그램에는 컴퓨팅, 네트워킹, 스토리지, 수명 주기 정책, 지표 수집, 애플리케이션의 운영 및 기능 ElastiCache 요소 사용이 포함되어야 합니다.
4. 워크로드 요구 사항은 시간이 지남에 따라 변할 가능성이 높으므로 워크로드 비용 관리에서 선제적으로 대응하기 위해서는 기본 구성 요소와 주요 기능 목표에 대한 이해를 지속적으로 유지하고 문서화하는 것이 중요합니다.
5. 가시성, 책임성, 우선 순위 지정 및 리소스에 대한 경영진 지원은 에 대한 효과적인 비용 관리 전략을 수립하는 데 매우 중요합니다 ElastiCache.

COST 2: 지속적인 모니터링 도구를 사용하여 ElastiCache 리소스와 관련된 비용을 최적화하려면 어떻게 해야 하나요?

질문 수준 소개: ElastiCache 비용과 애플리케이션 성능 지표 간의 적절한 균형을 목표로 해야 합니다. AmazonCloudWatch 은 주요 운영 지표에 대한 가시성을 제공하여 필요에 따라 ElastiCache 리소스가 초과 또는 미달되었는지 평가하는 데 도움이 될 수 있습니다. 비용 최적화 관점에서는 과도 프로비저닝된 시기를 이해하고 운영, 가용성, 복원력 및 성능 요구 사항을 유지하면서 리소스 크기를 ElastiCache 조정할 수 있는 적절한 메커니즘을 개발할 수 있어야 합니다.

질문 수준의 이점: 워크로드 운영 요구 사항을 충족하기에 충분한 리소스를 프로비저닝하고, 낮은 리소스 활용도로 인해 비용이 최적화되지 않는 상태를 피하는 것이 이상적입니다. 장기간 동안 대용량 ElastiCache 리소스를 식별하고 운영하지 않도록 할 수 있어야 합니다.

- [필수] ElastiCache 클러스터 CloudWatch 를 모니터링하고 이러한 지표가 AWS Cost Explorer 대시보드와 어떤 관련이 있는지 분석하는 데 사용합니다.
  1. ElastiCache 는 호스트 수준 지표(예: CPU 사용량)와 캐시 엔진 소프트웨어에 고유한 지표(예: 캐시 가져오기 및 캐시 누락)를 모두 제공합니다. 이러한 지표는 60초 간격으로 각 캐시 노드에 대해 측정되어 게시됩니다.
  2. ElastiCache 성능 지표(CPUUtilization, EngineUtilization, SwapUsage CurrConnections및 Evictions)는 확장/축소(크거나 작은 캐시 노드 유형 사용) 또는 입출력(크거나 적은 샤드 추가)이 필요함을 나타낼 수 있습니다. 애플리케이션 성능 임계값을 충족하는 데 필요한 추가 비용과 최소 및 최대 시간을 추정하는 플레이북 매트릭스를 만들어 규모 조정에 대한 결정이 비용에 미치는 영향을 파악하세요.

[리소스]:

- CloudWatch 지표를 사용한 사용 모니터링
- 어떤 메트릭을 모니터링해야 합니까?
- Amazon ElastiCache 요금

- [필수] 백업 전략과 비용에 미치는 영향을 이해하고 문서화합니다.
  1. ElastiCache를 사용하면 백업이 내구성이 뛰어난 스토리지를 제공하는 Amazon S3에 저장됩니다. 장애 복구 능력과 관련하여 비용에 미치는 영향을 이해해야 합니다.
  2. 보존 한도가 지난 백업 파일을 삭제하는 자동 백업을 활성화합니다.

  [리소스]:

  - 자동 백업 예약
  - Amazon Simple Storage Service 요금

- [가장 좋음] 잘 이해되고 문서화된 워크로드의 비용을 관리하기 위한 의도적인 전략으로 인스턴스에 예약 노드를 사용합니다. 노드 유형과 예약 기간(1년 또는 3년)에 따라 예약 노드에 선결제 요금이 부과됩니다. 이 요금은 온디맨드 노드에서 발생하는 시간당 사용 요금보다 훨씬 낮습니다.
  1. 예약된 인스턴스 요구 사항을 추정하기에 충분한 데이터를 수집할 때까지 온디맨드 노드를 사용하여 ElastiCache 클러스터를 운영해야 할 수 있습니다. 요구 사항을 충족하는 데 필요한 리소스를 계획 및 문서화하고 인스턴스 유형(온디맨드 또는 예약형)의 예상 비용을 비교합니다.
  2. 사용 가능한 새 캐시 노드 유형을 정기적으로 평가하고 비용 및 운영 지표 관점에서 인스턴스 플릿을 새 캐시 노드 유형으로 마이그레이션하는 것이 합리적인지 평가합니다.

COST 3: 데이터 계층화를 지원하는 인스턴스 유형을 사용해야 합니까? 데이터 계층화의 이점은 무엇인가요? 데이터 계층화 인스턴스를 사용하지 않는 경우는 언제인가요?

질문 수준의 소개: 적절한 인스턴스 유형을 선택하면 성능 및 서비스 수준뿐만 아니라 재정에도 영향을 미칠 수 있습니다. 인스턴스 유형마다 관련된 비용이 다릅니다. 메모리의 모든 스토리지 요구 사항을 수용할 수 있는 대규모 인스턴스 유형을 하나 또는 몇 개 선택하는 것은 자연스러운 결정일 수 있습니다. 그러나 이는 프로젝트가 성숙해짐에 따라 비용에 상당한 영향을 미칠 수 있습니다. 올바른 인스턴스 유형을 선택했는지 확인하려면 ElastiCache 객체 유휴 시간을 정기적으로 검사해야 합니다.

질문 수준의 이점: 다양한 인스턴스 유형이 현재와 미래의 비용에 어떤 영향을 미치는지 명확히 이해해야 합니다. 사소하거나 주기적인 워크로드 변경으로 인해 과도한 비용 변동이 발생해서는 안 됩니다. 워크로드가 허용하는 경우 데이터 계층화를 지원하는 인스턴스 유형은 사용 가능한 스토리지당 더 나

은 가격을 제공합니다. 인스턴스당 사용 가능한 SSD 스토리지 데이터 계층화로 인해 인스턴스당 총 데이터가 훨씬 더 많이 지원됩니다.

- [필수] 데이터 계층화 인스턴스의 한계를 이해합니다.
  1. ElastiCache (Redis OSS) 클러스터에만 사용할 수 있습니다.
  2. 제한된 인스턴스 유형만 데이터 계층화를 지원합니다.
  3. ElastiCache (Redis OSS) 버전 6.2 이상만 지원됩니다.
  4. 큰 항목은 로 교체되지 않습니다SSD. 128MiB 이상의 객체는 메모리에 보관됩니다.

  [리소스]:
  - 데이터 계층화
  - Amazon ElastiCache 요금
- [필수] 워크로드가 데이터베이스의 몇 퍼센트에 정기적으로 액세스하는지 파악합니다.
  1. 데이터 계층화 인스턴스는 전체 데이터 세트의 일부에만 액세스하는 경우가 많지만 나머지 데이터에는 빠른 액세스가 필요한 워크로드에 적합합니다. 즉, 핫 데이터와 웜 데이터의 비율이 약 20:80 입니다.
  2. 객체 유휴 시간에 대한 클러스터 수준의 추적을 개발합니다.
  3. 500Gb가 넘는 데이터를 대규모로 구현하는 데 적합합니다.
- [필수] 특정 워크로드에서는 데이터 계층화 인스턴스가 선택 사항이 아니라는 점을 이해합니다.
  1. 덜 자주 사용되는 객체는 로컬 로 교체되므로 액세스하는 데 드는 성능 비용은 약간 있습니다SSD. 애플리케이션이 응답 시간에 민감한 경우 워크로드에 미치는 영향을 테스트하세요.
  2. 대부분 크기가 128MiB 이상인 대형 객체를 저장하는 캐시에는 적합하지 않습니다.

  [리소스]:
  - 제한 사항
- [가장 좋음] 예약 인스턴스 유형은 데이터 계층화를 지원합니다. 이를 통해 인스턴스당 데이터 스토리지 용량 측면에서 가장 낮은 비용이 보장됩니다.
  1. 요구 사항을 더 잘 이해할 때까지 비 데이터 계층화 인스턴스를 사용하여 ElastiCache 클러스터를 운영해야 할 수 있습니다.
  2. ElastiCache 클러스터 데이터 사용 패턴을 분석합니다.
  3. 객체 유휴 시간을 주기적으로 수집하는 자동화된 작업을 생성합니다.
  4. 대다수(약 80%)의 객체가 워크로드에 적합하다고 판단되는 기간 동안 유휴 상태인 경우, 조사 결과를 문서화하고 클러스터를 데이터 계층화를 지원하는 인스턴스로 마이그레이션하는 것을 제안하세요.

5. 사용 가능한 새 캐시 노드 유형을 정기적으로 평가하고 비용 및 운영 지표 관점에서 인스턴스 플릿을 새 캐시 노드 유형으로 마이그레이션하는 것이 합리적인지 평가합니다.

[리소스]:

- OBJECT IDLETIME
- Amazon ElastiCache 요금

를 사용한 일반적인 문제 해결 단계 및 모범 사례 ElastiCache

다음 주제에서는 사용 시 발생할 수 있는 오류 및 문제에 대한 문제 해결 조언을 제공합니다 ElastiCache. 여기에 나열되지 않은 문제를 발견하는 경우 이 페이지의 피드백 버튼을 사용하여 해당 문제를 보고할 수 있습니다.

일반적인 지원 질문에 대한 자세한 문제 해결 조언 및 답변은 AWS 지식 센터를 참조하세요.

주제

- 연결 문제
- Valkey 또는 Redis OSS 클라이언트 오류
- ElastiCache Serverless의 지연 시간 문제 해결
- ElastiCache Serverless의 제한 문제 해결
- 지속적인 연결 문제
- 관련 항목

연결 문제

ElastiCache 캐시에 연결할 수 없는 경우 다음 중 하나를 고려하세요.

1. 사용TLS: ElastiCache 엔드포인트에 연결하려고 할 때 연결이 끊긴 경우 클라이언트TLS에서 를 사용하지 않을 수 있습니다. ElastiCache Serverless를 사용하는 경우 전송 중 암호화가 항상 활성화됩니다. 클라이언트가 TLS를 사용하여 캐시에 연결하는지 확인합니다. TLS 활성화된 캐시 에 연결하는 방법에 대해 자세히 알아봅니다.
2. VPC: ElastiCache caches는 내에서만 액세스할 수 있습니다VPC. 캐시에 액세스하는 EC2 인스턴스와 ElastiCache 캐시가 동일한 에 생성되었는지 확인합니다VPC. 또는 EC2 인스턴스가 VPC 있는와 캐시를 생성하는 VPC 간에 VPC 피어링을 활성화해야 합니다.

3. 보안 그룹: ElastiCache 는 보안 그룹을 사용하여 캐시에 대한 액세스를 제어합니다. 다음을 고려하세요.
   a. ElastiCache 캐시에서 사용하는 보안 그룹이 EC2 인스턴스에서 인바운드 액세스를 허용하는지 확인합니다. 보안 그룹에서 인바운드 규칙을 올바르게 설정하는 방법을 알아보려면 여기를 참조하세요.
   b. ElastiCache 캐시에 사용되는 보안 그룹이 캐시의 포트(서버리스의 경우 6379 및 6380, 자체 설계의 경우 기본적으로 6379)에 대한 액세스를 허용하는지 확인합니다. 는 이러한 포트를 ElastiCache 사용하여 Valkey 또는 Redis OSS 명령을 수락합니다. 포트 액세스를 설정하는 방법에 대한 자세한 내용은 여기에서 확인하세요.

연결이 계속 어려운 경우 다른 단계는 섹션을 참조지속적인 연결 문제하세요.

Valkey 또는 Redis OSS 클라이언트 오류

ElastiCache Serverless는 Valkey 또는 Redis OSS 클러스터 모드 프로토콜을 지원하는 클라이언트를 통해서만 액세스할 수 있습니다. 자체 설계된 클러스터는 클러스터 구성에 따라 어느 모드에서든 클라이언트에서 액세스할 수 있습니다.

클라이언트에 오류가 발생하는 경우 다음을 고려하세요.

1. 클러스터 모드: SELECT 명령에 CROSSLOT 오류나 오류가 발생하는 경우 클러스터 프로토콜을 지원하지 않는 Valkey 또는 Redis OSS 클라이언트를 사용하여 클러스터 모드 활성화 캐시에 액세스하려고 할 수 있습니다. ElastiCache Serverless는 Valkey 또는 Redis OSS 클러스터 프로토콜을 지원하는 클라이언트만 지원합니다. “클러스터 모드 비활성화”(CMD)OSS에서 Valkey 또는 Redis를 사용하려면 자체 클러스터를 설계해야 합니다.
2. CROSSLOT 오류: `ERR CROSSLOT Keys in request don't hash to the same slot` 오류가 발생하는 경우 클러스터 모드 캐시의 동일한 슬롯에 속하지 않는 키에 액세스하려고 할 수 있습니다. 참고로 ElastiCache Serverless는 항상 클러스터 모드에서 작동합니다. 여러 키를 포함하는 다중 키 작업, 트랜잭션 또는 Lua 스크립트는 관련된 모든 키가 동일한 해시 슬롯에 있는 경우에만 허용됩니다.

Valkey 또는 Redis OSS 클라이언트 구성에 대한 추가 모범 사례는 이 블로그 게시물을 검토하세요.

ElastiCache Serverless의 지연 시간 문제 해결

워크로드에 지연 시간이 높게 나타나는 경우 `SuccessfulReadRequestLatency` 및 `SuccessfulWriteRequestLatency` 지표를 CloudWatch 분석하여 지연 시간이 ElastiCache

Serverless와 관련이 있는지 확인할 수 있습니다. 이러한 지표는 ElastiCache Serverless 내부인 지연 시간을 측정합니다. 클라이언트 측 지연 시간과 클라이언트와 ElastiCache Serverless 엔드포인트 간의 네트워크 트립 시간은 포함되지 않습니다.

클라이언트 측 지연 시간 문제 해결

클라이언트 측에서 지연 시간이 증가했지만 서버 측 지연 시간을 측정하는 CloudWatch `SuccessfulReadRequestLatency` 및 `SuccessfulWriteRequestLatency` 지표가 증가하지 않은 경우 다음을 고려하세요.

- 보안 그룹이 포트 6379 및 6380에 대한 액세스를 허용하는지 확인합니다. ElastiCache Serverless는 기본 엔드포인트에 6379 포트를 사용하고 리더 엔드포인트에 6380 포트를 사용합니다. 일부 클라이언트는 애플리케이션이 복제본에서 읽기 기능을 사용하지 않더라도 모든 새 연결에 대해 두 포트에 대한 연결을 설정합니다. 보안 그룹이 두 포트에 대한 인바운드 액세스를 허용하지 않는 경우 연결 설정이 더 오래 걸릴 수 있습니다. 여기에서 포트 액세스를 설정하는 방법에 대해 자세히 알아보세요.

서버 측 지연 시간 문제 해결

일부 변동성 및 간헐적 급증은 우려의 원인이 되어서는 안 됩니다. 그러나 `Average` 통계가 급격한 증가를 보이고 지속되면 AWS Health Dashboard 및 Personal Health Dashboard에서 자세한 내용을 확인해야 합니다. 필요한 경우 를 사용하여 지원 사례를 여는 것이 좋습니다 AWS Support.

지연 시간을 줄이기 위해 다음 모범 사례와 전략을 고려하세요.

- 복제본에서 읽기 활성화: 애플리케이션에서 허용하는 경우 Valkey 또는 Redis OSS 클라이언트에서 “복제본에서 읽기” 기능을 활성화하여 읽기를 확장하고 지연 시간을 줄이는 것이 좋습니다. 활성화되면 ElastiCache Serverless는 클라이언트와 동일한 가용 영역(AZ)에 있는 복제본 캐시 노드로 읽기 요청을 라우팅하려고 시도하므로 AZ 간 네트워크 지연 시간을 방지합니다. 클라이언트에서 복제본에서 읽기 기능을 활성화하면 애플리케이션이 데이터의 최종 일관성을 수락한다는 의미입니다. 키에 쓴 후 를 읽으려고 하면 애플리케이션은 한동안 이전 데이터를 수신할 수 있습니다.
- 애플리케이션이 캐시와 동일한 에 배포되었는지 확인합니다. 애플리케이션이 캐시AZs와 동일한 에 배포되지 않은 경우 클라이언트 측 지연 시간이 더 길어질 수 AZs 있습니다. 서버리스 캐시를 생성할 때 애플리케이션이 캐시에 액세스할 서브넷을 제공하고 ElastiCache Serverless는 해당 서브넷에 VPC 엔드포인트를 생성할 수 있습니다. 애플리케이션이 동일한 에 배포되었는지 확인합니다AZs. 그렇지 않으면 애플리케이션이 캐시에 액세스할 때 교차 AZ 홉이 발생하여 클라이언트 측 지연 시간이 길어질 수 있습니다.

- 연결 재사용: ElastiCache 서버리스 요청은 RESP 프로토콜을 사용하여 TLS 활성화된 TCP 연결을 통해 이루어집니다. 연결을 시작하는 데(구성된 경우 연결 인증 포함) 시간이 걸리므로 첫 번째 요청의 지연 시간이 일반적인 것보다 높습니다. 이미 초기화된 연결을 통한 요청은 ElastiCache의 일관된 짧은 지연 시간을 제공합니다. 따라서 연결 풀링을 사용하거나 기존 Valkey 또는 Redis OSS 연결을 재사용하는 것을 고려해야 합니다.
- 스케일링 속도: ElastiCache Serverless는 요청 속도가 증가함에 따라 자동으로 스케일링됩니다. ElastiCache Serverless가 확장하는 속도보다 빠른 요청 속도가 갑자기 크게 증가하면 일정 시간 동안 지연 시간이 늘어날 수 있습니다. ElastiCache Serverless는 일반적으로 지원되는 요청 속도를 빠르게 높여 요청 속도를 두 배로 늘리는 데 최대 10~12분이 걸릴 수 있습니다.
- 장기 실행 명령 검사: Lua 스크립트 또는 대규모 데이터 구조의 OSS 명령을 포함한 일부 Valkey 또는 Redis 명령이 장기간 실행될 수 있습니다. 이러한 명령을 식별하려면 에서 명령 수준 지표를 ElastiCache 게시합니다. ElastiCache Serverless를 사용하면 `BasedECPUs` 지표를 사용할 수 있습니다.
- 제한 요청: ElastiCache Serverless에서 요청이 제한되면 애플리케이션에서 클라이언트 측 지연 시간이 증가할 수 있습니다. ElastiCache Serverless에서 요청이 제한되면 `ThrottledRequests` ElastiCache Serverless 지표가 증가하는 것을 볼 수 있습니다. 제한 요청 문제를 해결하려면 아래 섹션을 검토하세요.
- 키 및 요청의 균일한 배포: 에서 Valkey 및 Redis ElastiCache 를 사용하면 슬롯당 키 또는 요청이 고르지 OSS않게 배포되면 핫 슬롯이 발생하여 지연 시간이 늘어날 수 있습니다. ElastiCache Serverless는 간단한 SET/GET 명령을 실행하는 워크로드에서 단일 슬롯에서 최대 30,000ECPUs/초(복제본에서 읽기 사용 시 90,000ECPUs/초)를 지원합니다. 슬롯 간 키 및 요청 배포를 평가하고 요청 속도가 이 제한을 초과하는 경우 균일한 배포를 확인하는 것이 좋습니다.

ElastiCache Serverless의 제한 문제 해결

서비스 지향 아키텍처 및 분산 시스템에서는 다양한 서비스 구성 요소에 의해 API 호출이 처리되는 속도를 제한하는 것을 제한이라고 합니다. 이렇게 하면 스파이크가 원활하게 발생하고, 구성 요소 처리량의 불일치를 제어하며, 예기치 않은 운영 이벤트가 발생할 때 예측 가능한 복구가 가능합니다. ElastiCache Serverless는 이러한 유형의 아키텍처를 위해 설계되었으며, 대부분의 Valkey 또는 Redis OSS 클라이언트는 제한된 요청에 대해 에 재시도를 내장했습니다. 어느 정도의 제한이 애플리케이션에 반드시 문제가 되는 것은 아니지만 데이터 워크플로의 지연 시간에 민감한 부분을 지속적으로 제한하면 사용자 경험에 부정적인 영향을 미치고 시스템의 전반적인 효율성이 떨어질 수 있습니다.

ElastiCache Serverless에서 요청이 제한되면 `ThrottledRequests` ElastiCache Serverless 지표가 증가하는 것을 볼 수 있습니다. 제한된 요청 수가 많을 경우 다음을 고려하세요.

- 확장 속도: ElastiCache 더 많은 데이터를 수집하거나 요청 속도를 높일수록 Serverless가 자동으로 확장됩니다. ElastiCache Serverless가 확장하는 속도보다 애플리케이션이 더 빠르게 확장되면 요청이 제한될 수 있고 ElastiCache Serverless는 워크로드에 맞게 확장됩니다. ElastiCache Serverless는 일반적으로 스토리지 크기를 빠르게 늘려 캐시의 스토리지 크기를 두 배로 늘리는 데 최대 10~12분이 걸릴 수 있습니다.
- 키 및 요청의 균일한 배포: 에서 Valkey 또는 Redis ElastiCache 를 사용하면 슬롯당 키 또는 요청이 고르지 OSS않게 배포되면 핫 슬롯이 발생할 수 있습니다. 핫 슬롯은 단순 SET/GET 명령을 실행하는 워크로드에서 단일 슬롯에 대한 요청 속도가 초ECPUs당 30,000을 초과하는 경우 요청을 제한할 수 있습니다.
- 복제본에서 읽기: 애플리케이션에서 허용하는 경우 “복제본에서 읽기” 기능을 사용하는 것이 좋습니다. 대부분의 Valkey 또는 Redis OSS 클라이언트는 읽기를 복제본 노드로 전달하도록 “스케일링”하도록 구성할 수 있습니다. 이 기능을 사용하면 읽기 트래픽을 확장할 수 있습니다. 또한 ElastiCache Serverless는 복제본 요청의 읽기를 애플리케이션과 동일한 가용 영역의 노드로 자동으로 라우팅하여 지연 시간을 줄입니다. 복제본에서 읽기가 활성화되면 단순 SETECPUsGET/ 명령이 있는 워크로드에 대해 단일 슬롯에서 최대 90,000/초를 달성할 수 있습니다.

지속적인 연결 문제

의 지속적 연결 문제를 해결하는 동안 다음 항목을 확인해야 합니다 ElastiCache.

주제

- 보안 그룹
- 네트워크 ACLs
- 라우팅 테이블
- DNS 해상도
- 서버 측 진단을 사용하여 문제 식별
- 네트워크 연결 검증
- 네트워크 관련 제한 사항
- CPU 사용량
- 서버 측에서 연결이 종료되는 경우
- Amazon EC2 인스턴스에 대한 클라이언트 측 문제 해결
- 단일 요청을 완료하는 데 걸리는 시간 분석

보안 그룹

보안 그룹은 ElastiCache 클라이언트(EC2인스턴스, AWS Lambda 함수, Amazon ECS 컨테이너 등) 및 ElastiCache 캐시를 보호하는 가상 방화벽입니다. 보안 그룹은 상태 유지(stateful) 방식으로 작동합니다. 즉, 들어오거나 나가는 트래픽이 허용되면 해당 트래픽에 대한 응답이 관련 보안 그룹의 컨텍스트에서 자동으로 승인됩니다.

상태 유지 기능을 사용하려면 보안 그룹이 권한이 부여된 모든 연결을 추적해야 하며, 이렇게 추적되는 연결에는 제한이 있습니다. 이 제한에 도달하면 새 연결이 실패합니다. 클라이언트 또는 ElastiCache 측에서 한도에 도달했는지 확인하는 방법에 대한 도움말은 문제 해결 섹션을 참조하세요.

클라이언트와 ElastiCache 클러스터에 동시에 단일 보안 그룹을 할당하거나 각각에 대해 개별 보안 그룹을 할당할 수 있습니다.

두 경우 모두 소스에서 ElastiCache 포트의 TCP 아웃바운드 트래픽과 동일한 포트의 인바운드 트래픽을 로 허용해야 합니다 ElastiCache. 기본 포트는 Memcached의 경우 11211이고 Valkey 또는 Redis 의 경우 6379입니다OSS. 기본적으로 보안 그룹은 모든 아웃바운드 트래픽을 허용합니다. 이 경우 대상 보안 그룹의 인바운드 규칙만 필요합니다.

자세한 내용은 Amazon 에서 ElastiCache 클러스터에 액세스하기 위한 액세스 패턴을 참조하세요 VPC.

네트워크 ACLs

네트워크 액세스 제어 목록(ACLs)은 상태 비저장 규칙입니다. 트래픽이 성공하려면 양방향(인바운드 및 아웃바운드)으로 허용되어야 합니다. 네트워크ACLs는 특정 리소스가 아닌 서브넷에 할당됩니다. 특히 클라이언트 리소스가 동일한 서브넷에 있는 경우 ElastiCache 및 클라이언트 리소스에 동일한 를 ACL 할당할 수 있습니다.

기본적으로 네트워크는 모든 추적을 ACLs 허용합니다. 그러나 트래픽을 거부하거나 허용하도록 네트워크 ACL을 사용자 지정할 수 있습니다. 또한 ACL 규칙 평가는 순차적입니다. 즉, 트래픽과 일치하는 숫자가 가장 낮은 규칙에서 허용하거나 거부합니다. Valkey 또는 Redis OSS 트래픽을 허용하는 최소 구성은 다음과 같습니다.

클라이언트 측 네트워크ACL:

- 인바운드 규칙:
- 규칙 번호: 모든 거부 규칙보다 번호가 작아야 합니다.
- 유형: 사용자 지정 TCP 규칙,

- 프로토콜: TCP
- 포트 범위: 1024-65535
- 소스: 0.0.0.0/0(또는 ElastiCache 클러스터 서브넷에 대한 개별 규칙 생성)
- 허용/거부: 허용

- 아웃바운드 규칙:
- 규칙 번호: 모든 거부 규칙보다 번호가 작아야 합니다.
- 유형: 사용자 지정 TCP 규칙,
- 프로토콜: TCP
- 포트 범위: 6379
- 소스: 0.0.0.0/0(또는 ElastiCache 클러스터 서브넷. 특정 를 사용하면 장애 조치 또는 클러스터 크기 조정 시 문제가 발생할 IPs 수 있음에 유의하세요.)
- 허용/거부: 허용

ElastiCache 네트워크ACL:

- 인바운드 규칙:
- 규칙 번호: 모든 거부 규칙보다 번호가 작아야 합니다.
- 유형: 사용자 지정 TCP 규칙,
- 프로토콜: TCP
- 포트 범위: 6379
- 소스: 0.0.0.0/0(또는 ElastiCache 클러스터 서브넷에 대한 개별 규칙 생성)
- 허용/거부: 허용

- 아웃바운드 규칙:
- 규칙 번호: 모든 거부 규칙보다 번호가 작아야 합니다.
- 유형: 사용자 지정 TCP 규칙,
- 프로토콜: TCP
- 포트 범위: 1024-65535
- 소스: 0.0.0.0/0(또는 ElastiCache 클러스터 서브넷. 특정 를 사용하면 장애 조치 또는 클러스터 크기 조정 시 문제가 발생할 IPs 수 있음에 유의하세요.)
- 허용/거부: 허용

자세한 내용은 네트워크 섹션을 ACLs참조하세요.

라우팅 테이블

네트워크 와 마찬가지로 ACLs각 서브넷에는 서로 다른 라우팅 테이블이 있을 수 있습니다. 클라이언트와 ElastiCache 클러스터가 서로 다른 서브넷에 있는 경우 라우팅 테이블에서 서로 연결할 수 있도록 허용해야 합니다.

여러 , VPCs동적 라우팅 또는 네트워크 방화벽을 포함하는 더 복잡한 환경은 문제를 해결하기 어려울 수 있습니다. 네트워크 연결 검증 섹션을 참조하여 네트워크 설정이 적절한지 확인하세요.

DNS 해상도

ElastiCache 는 DNS 이름을 기반으로 서비스 엔드포인트를 제공합니다. 사용 가능한 엔드포인트는 `Configuration`, `Primary`, `Reader` 및 `Node` 엔드포인트입니다. 자세한 내용은 연결 엔드포인트 찾기를 참조하세요.

장애 조치 또는 클러스터 수정의 경우 엔드포인트 이름에 연결된 주소가 변경될 수 있으며 자동으로 업데이트됩니다.

사용자 지정 DNS 설정(즉, VPC DNS 서비스를 사용하지 않음)은 ElastiCache제공된 DNS 이름을 인식하지 못할 수 있습니다. (다음과 `dig` 같이) 또는 와 같은 시스템 도구를 사용하여 시스템이 ElastiCache 엔드포인트를 성공적으로 해결할 수 있는지 확인합니다`nslookup`.

```
$ dig +short example.xxxxxx.ng.0001.use1.cache.amazonaws.com
example-001.xxxxxx.0001.use1.cache.amazonaws.com.
1.2.3.4
```

VPC DNS 또한 서비스를 통해 이름 확인을 강제로 적용할 수 있습니다.

```
$ dig +short example.xxxxxx.ng.0001.use1.cache.amazonaws.com @169.254.169.253
example-001.tihewd.0001.use1.cache.amazonaws.com.
1.2.3.4
```

서버 측 진단을 사용하여 문제 식별

CloudWatch ElastiCache 엔진의 지표 및 런타임 정보는 연결 문제의 잠재적 원인을 식별하기 위한 일반적인 소스 또는 정보입니다. 좋은 분석은 일반적으로 다음과 같은 항목으로 시작됩니다.

- CPU 사용량: Valkey 및 RedisOSS는 다중 스레드 애플리케이션입니다. 그러나 각 명령의 실행은 단일(주) 스레드에서 발생합니다. 이러한 이유로 는 지표 `CPUUtilization` 및 를 ElastiCache 제공합

니다EngineCPUUtilization. EngineCPUUtilization는 Valkey 또는 Redis OSS 프로세스 전용 CPU 사용률과 모든 의 사용량CPUUtilization을 제공합니다vCPUs. vCPU가 둘 이상인 노드는 일반적으로 CPUUtilization 및 에 대해 값이 다르EngineCPUUtilization며, 두 번째 노드는 일반적으로 더 높습니다. 높음은 완료하는 데 상당한 CPU 시간이 걸리는 요청 수 증가 또는 복잡한 작업으로 인해 발생할 EngineCPUUtilization 수 있습니다. 다음을 사용하여 둘 모두를 식별할 수 있습니다.

- 많은 수의 요청: EngineCPUUtilization 패턴과 일치하는 다른 지표의 증가 여부를 확인합니다. 유용한 지표는 다음과 같습니다.
  - CacheHits 및 CacheMisses: 성공한 요청 또는 캐시에서 유효한 항목을 찾지 못한 요청의 수입니다. 적중률에 비해 누락률이 높으면 애플리케이션에서 무의미한 요청에 시간과 리소스를 낭비하고 있는 것입니다.
  - SetTypeCmds 및 GetTypeCmds: EngineCPUUtilization과 상관 관계가 있는 이 두 지표는 SetTypeCmds에 의해 측정되는 쓰기 요청 또는 GetTypeCmds에 의해 측정되는 읽기 요청에 대한 로드가 얼마나 높은지 파악하는 데 도움이 될 수 있습니다 로드가 주로 읽기인 경우 여러 읽기 전용 복제본을 사용하면 여러 노드에 요청을 분산시켜 균형을 유지하고 기본 노드를 쓰기용으로 할당할 수 있습니다. 클러스터 모드 비활성화 클러스터에서는 ElastiCache 리더 엔드포인트를 사용하여 애플리케이션에서 추가 연결 구성을 생성하여 읽기 전용 복제본을 사용할 수 있습니다. 자세한 내용은 연결 엔드포인트 찾기를 참조하세요. 읽기 작업은 이 추가 연결에 제출되어야 합니다. 쓰기 작업은 일반적인 기본 엔드포인트를 통해 수행됩니다. 클러스터 모드가 활성화된 클러스터에서는 기본적으로 읽기 전용 복제본을 지원하는 라이브러리를 사용하는 것이 좋습니다. 올바른 플래그를 사용하면 라이브러리가 클러스터 토폴로지, 복제본 노드를 자동으로 검색하고, READONLY Valkey 또는 Redis OSS 명령을 통해 읽기 작업을 활성화하고, 읽기 요청을 복제본에 제출할 수 있습니다.
- 많은 수의 연결:
  - CurrConnections 및 NewConnections: CurrConnection은 데이터 포인트 컬렉션 순간에 설정된 연결 수이고, NewConnections는 이 기간에 생성된 연결 수를 보여줍니다.

    연결을 생성하고 처리하면 상당한 CPU 오버헤드가 발생합니다. 또한 새 연결을 생성하는 데 필요한 TCP3방향 핸드셰이크는 전체 응답 시간에 부정적인 영향을 미칩니다.

    NewConnections 분당 수천 개의 ElastiCache 노드는 연결이 최적의 상태가 아닌 몇 가지 명령으로만 생성되고 사용됨을 나타냅니다. 설정된 연결을 유지하고 새 작업에 이 연결을 다시 사용하는 것이 모범 사례입니다. 이것이 가능하려면 클라이언트 애플리케이션이 연결 풀링이나 영구 연결을 지원하고 적절하게 구현해야 합니다. 연결 풀링을 사용하는 경우 currConnections 수에는 큰 변화가 없으며 NewConnections는 가능한 낮아야 합니다. Valkey 및 Redis는 적은 수의 로 최적의 성능을 OSS 제공합니다currConnections. 수십 또는 수

백 currConnection 개 순으로 유지하면 클라이언트 버퍼 및 연결 서비스 CPU 주기와 같은 개별 연결을 지원하는 리소스 사용을 최소화할 수 있습니다.

- 네트워크 처리량:
  - 대역폭: ElastiCache 노드에 노드 크기에 비례하는 네트워크 대역폭이 있는지 확인합니다. 애플리케이션마다 특성이 다르기 때문에 워크로드에 따라 결과가 달라질 수 있습니다. 예를 들어 작은 요청의 비율이 높은 애플리케이션은 네트워크 처리량보다 CPU 사용량에 더 많은 영향을 미치는 경향이 있는 반면 키가 클수록 네트워크 사용률이 높아집니다. 그러므로 제한을 보다 정확하게 이해하기 위해 실제 워크로드로 노드를 테스트하는 것이 좋습니다.

    애플리케이션의 로드를 시뮬레이션하면 보다 정확한 결과를 얻을 수 있습니다. 그러나 벤치마크 도구가 제한에 대한 좋은 아이디어를 줄 수 있습니다.
  - 요청이 주로 읽기인 경우 읽기 작업에 복제본을 사용하면 기본 노드의 로드가 줄어듭니다. 사용 사례가 주로 쓰기인 경우 많은 복제본을 사용하면 네트워크 사용량이 크게 늘어납니다. 기본 노드에 한 바이트가 쓰여질 때마다 N바이트가 복제본으로 전송됩니다. 여기서 N은 복제본 수입니다. 쓰기 집약적 워크로드의 모범 사례는 클러스터 모드가 활성화된 ElastiCache (Redis OSS)를 사용하여 쓰기를 여러 샤드 간에 균형을 맞추거나 네트워크 기능이 더 많은 노드 유형으로 확장하는 것입니다.
  - 및 는 CloudWatchmetrics `NetworkBytesIn` 각각 노드로 들어오거나 노드에서 나가는 데이터의 양을 `NetworkBytesOut` 제공합니다. `ReplicationBytes`는 데이터 복제 전용 트래픽입니다.

  자세한 내용은 네트워크 관련 제한 사항 단원을 참조하십시오.
- 복잡한 명령: Redis OSS 명령은 단일 스레드에서 제공되므로 요청이 순차적으로 제공됩니다. 단일 느린 명령은 다른 요청 및 연결에 영향을 줄 수 있으며 시간 초과로 이어질 수 있습니다. 여러 값, 키 또는 데이터 유형에 작동하는 명령을 사용할 때 신중하게 사용해야 합니다. 연결은 파라미터의 수나 입력 또는 출력 값의 크기에 따라 차단되거나 종료될 수 있습니다.

  악명 높은 예는`KEYS` 명령입니다. 이 명령은 전체 키스페이스에 걸쳐 주어진 패턴을 검색하며, 실행 중에 다른 명령의 실행을 차단합니다. RedisOSS는 “Big O” 표기법을 사용하여 명령의 복잡성을 설명합니다.

  Keys 명령은 O(N) 시간 복잡도를 가지며 N은 데이터베이스의 키 수입니다. 따라서 키 수가 클수록 명령 속도가 느려집니다. `KEYS`는 여러 방식으로 문제를 일으킬 수 있습니다. 검색 패턴을 사용하지 않으면 이 명령은 사용 가능한 모든 키 이름을 반환합니다. 수 천개나 수 백만 개의 항목이 있는 데이터베이스에서는 엄청난 출력이 생성되어 네트워크 버퍼가 가득 차게 됩니다.

검색 패턴을 사용하는 경우 패턴과 일치하는 키만 클라이언트로 반환됩니다. 그러나 엔진은 여전히 전체 키스페이스에 걸쳐 키를 검색하며 명령을 완료하는 데 걸리는 시간은 동일합니다.

KEYS 명령의 대안은 SCAN 명령입니다. 이 명령은 키스페이스를 반복해서 처리하고 특정 수의 항목으로 반복을 제한하여 엔진의 장기적인 차단을 방지합니다.

Scan에는 반복 블록의 크기를 설정하는 데 사용되는 COUNT 파라미터가 있습니다. 기본값은 10(반복당 10개 항목)입니다.

데이터베이스의 항목 수에 따라, 작은 COUNT 값 블록이 전체 스캔을 완료하는 데 더 많은 반복을 필요로 하며 값이 클수록 각 반복마다 엔진을 더 오래 사용할 수 있습니다. 작은 count 값은 큰 데이터베이스에서 SCAN 속도를 느리게 만들며, 값이 클수록 KEYS에서 언급한 동일한 문제가 발생할 수 있습니다.

예를 들어, SCAN 명령을 count 값 10으로 실행하면 1백만 개의 키가 있는 데이터베이스에서 100,000번의 반복이 필요합니다. 평균 네트워크 왕복 시간이 0.5밀리초인 경우 요청을 전송하는 데 약 50,000밀리초(50초)가 소비됩니다.

반면에 count 값이 100,0000인 경우 단일 반복으로 충분하며 요청을 전송하는 데 단지 0.5밀리초가 소비됩니다. 그러나 명령이 모든 키스페이스의 처리를 마칠 때까지 엔진이 다른 작업에 대해 완전히 차단됩니다.

KEYS 외에도, 올바르게 사용하지 않으면 잠재적인 문제를 일으킬 수 있는 여러 다른 명령이 있습니다. 모든 명령의 목록과 각각의 시간 복잡성을 보려면 Valkey 및 Redis OSS 명령으로 이동합니다.

잠재적 문제의 예:

- Lua 스크립트: Valkey 및 Redis는 임베디드 Lua 해석기를 OSS 제공하여 서버 측에서 스크립트를 실행할 수 있습니다. Valkey 및 Redis의 Lua 스크립트OSS는 엔진 수준에서 실행되며 정의에 따라 원자적입니다. 즉, 스크립트가 실행되는 동안에는 다른 명령 또는 스크립트를 실행할 수 없습니다. Lua 스크립트는 엔진에서 직접 여러 명령, 의사 결정 알고리즘, 데이터 구문 분석 등을 실행할 수 있는 가능성을 제공합니다. 스크립트의 원자성과 애플리케이션을 오프로드할 수 있는 기능은 유혹적이지만 스크립트는 작은 작업에 신중하게 사용해야 합니다. 에서 Lua 스크립트의 ElastiCache실행 시간은 5초로 제한됩니다. 키스페이스에 기록되지 않은 스크립트는 5초 후에 자동으로 종료됩니다. 데이터 손상 및 불일치를 방지하기 위해 스크립트 실행이 5초 내에 완료되지 않았고 실행 중에 쓰기가 있는 경우 노드가 장애 조치됩니다. 트랜잭션은 Redis 에서 여러 관련 키 수정의 일관성을 보장하는 대안입니다OSS. 트랜잭션을 사용하면 명령 블록을 실행

하여 기존 키의 수정을 감시할 수 있습니다. 트랜잭션이 완료되기 전에 감시된 키가 변경되면 모든 수정 사항이 무시됩니다.

- 항목의 일괄 삭제: DEL 명령에는 삭제할 키 이름에 해당하는 파라미터 여러 개를 사용할 수 있습니다. 삭제 작업은 동기식이며 파라미터 목록이 크거나 큰 목록, 세트, 정렬된 세트 또는 해시(여러 하위 항목을 포함하는 데이터 구조)가 포함된 경우 상당한 CPU 시간이 걸립니다. 즉, 단일 키를 삭제하는 경우에도 요소가 많은 경우 상당한 시간이 걸릴 수 있습니다. 의 대안은 Redis 4 이후 사용할 수 UNLINK있는 비동기 명령인 OSS DEL입니다. 는 가능하면 DEL 보다 선호되어야 UNLINK 합니다. ElastiCache (Redis OSS) 6x부터 lazyfree-lazy-user-del 파라미터는 활성화 UNLINK 시DEL처럼 명령을 작동합니다. 자세한 내용은 Redis OSS 6.0 파라미터 변경 사항을 참조하세요.
- 여러 키에 작동하는 명령: DEL은 이전에 여러 인수를 허용하는 명령으로 언급되었으며 실행 시간은 이에 직접적으로 비례합니다. 하지만 RedisOSS는 비슷한 방식으로 작동하는 더 많은 명령을 제공합니다. 예를 들면 MSET 및 MGET를 사용하면 한 번에 여러 문자열 키를 삽입하거나 검색할 수 있습니다. 이러한 명령을 사용하면 개별 SET 또는 GET 명령을 여러 번 사용할 때 발생하는 네트워크 대기 시간을 줄일 수 있습니다. 그러나 광범위한 파라미터 목록은 CPU 사용량에 영향을 미칩니다.

  사용CPU률만으로는 연결 문제가 발생하지 않지만 여러 키를 통해 하나 또는 몇 개의 명령을 처리하는 데 너무 많은 시간을 할애하면 다른 요청이 실패하고 전체 CPU 사용률이 증가할 수 있습니다.

  키의 수와 해당 크기는 명령의 복잡도, 결과적으로 완료 시간에 영향을 미칩니다.

  여러 키에 작동할 수 있는 다른 명령의 예에는 HMGET, HMSET, MSETNX, PFCOUNT, PFMERGE, SDIFF, SDIFFSTORE, SINTER, SINTERSTORE, SUNION, SUNIONSTORE, TOUCH, ZDIFF, ZDIFFSTORE, ZINTER, ZINTERSTORE 등이 있습니다.
- 여러 데이터 유형에 작용하는 명령: Redis는 데이터 유형에 관계없이 하나 이상의 키에 작용하는 명령OSS도 제공합니다. ElastiCache (Redis OSS)는 이러한 명령을 모니터링하는 지표KeyBasedCmds를 제공합니다. 이 지표는 선택한 기간 동안 다음과 같은 명령의 실행을 합산합니다.
  - O(N) 복잡도:
    - KEYS
  - O(1)
    - EXISTS
    - OBJECT
    - PTTL

    - RANDOMKEY
    - TTL
    - TYPE
    - EXPIRE
    - EXPIREAT
    - MOVE
    - PERSIST
    - PEXPIRE
    - PEXPIREAT
    - UNLINK (O(N)를 사용하여 메모리를 회수합니다. 그러나 메모리 회수 작업은 분리된 스레드에서 발생하며 엔진을 차단하지 않습니다.
  - 데이터 유형에 따라 복잡도 시간이 다릅니다.
    - DEL
    - DUMP
    - RENAME은 복잡도가 O(1)인 명령으로 간주되지만 내부적으로 DEL을 실행합니다. 실행 시간은 이름이 바뀐 키의 크기에 따라 달라집니다.
    - RENAMENX
    - RESTORE
    - SORT
  - 큰 해시: 해시는 여러 키-값 하위 항목이 있는 단일 키를 허용하는 데이터 유형입니다. 각 해시는 4,294,967,295개 항목을 저장할 수 있으며 큰 해시에 대한 작업은 비용이 많이 들 수 있습니다. KEYS와 유사하게, 해시에는 O(N) 시간 복잡도를 갖는 HKEYS 명령이 있으며, N은 해시의 항목 수입니다. 장기 실행 명령을 방지하려면 HKEYS 대신 HSCAN을 사용해야 합니다. HDEL, HGETALL, HMGET, HMSET 및 HVALS는 큰 해시에서 주의해서 사용해야 하는 명령입니다.
- 기타 빅 데이터 구조: CPU 해시 외에도 다른 데이터 구조는 집약적일 수 있습니다. 집합, 목록, 정렬된 집합 및 Hyperloglog는 크기와 사용된 명령에 따라 처리하는 데 상당한 시간이 걸릴 수 있습니다. 이러한 명령에 대한 자세한 내용은 Valkey 및 Redis OSS 명령 을 참조하세요.

네트워크 연결 검증

DNS 해상도, 보안 그룹, 네트워크 ACLs및 라우팅 테이블과 관련된 네트워크 구성을 검토한 후 VPC Reachability Analyzer 및 시스템 도구를 사용하여 연결을 검증할 수 있습니다.

Reachability Analyzer는 네트워크 연결을 테스트하고 모든 요구 사항 및 권한이 충족되는지 확인합니다. 아래 테스트를 수행하려면 에서 사용할 수 있는 ElastiCache 노드 중 하나의 ENI ID(Elastic Network Interface Identification)가 필요합니다VPC. 다음을 수행하여 이 ID를 찾을 수 있습니다.

1. https://console.aws.amazon.com/ec2/v2/home으로 이동하시겠습니까?#NIC:
2. ElastiCache 클러스터 이름 또는 이전에 DNS 검증한 IP 주소를 기준으로 인터페이스 목록을 필터링합니다.
3. ENI ID를 기록하거나 저장합니다. 여러 인터페이스가 표시되는 경우 설명을 검토하여 해당 인터페이스가 올바른 ElastiCache 클러스터에 속하는지 확인하고 그 중 하나를 선택합니다.
4. 다음 단계를 진행합니다.
5. 집에서 https://console.aws.amazon.com/vpc/분석 경로를 생성하시겠습니까?#ReachabilityAnalyzer 다음 옵션을 선택합니다.

   - 소스 유형 : ElastiCache 클라이언트가 Amazon 인스턴스 또는 awsvpc 네트워크가 있는 Amazon 등 다른 서비스를 사용하는 경우 네트워크 인터페이스에서 실행되는 EC2 경우 인스턴스를 선택하고 해당 리소스 ID(EC2인스턴스 또는 ENI ID) AWS Lambda를 선택합니다. AWS Fargate ECS
   - 대상 유형 : 네트워크 인터페이스를 선택하고 목록에서 ElasticacheENI를 선택합니다.
   - 대상 포트 : ElastiCache (Redis OSS)의 경우 6379를 지정하고 ElastiCache (Memcached)의 경우 11211을 지정합니다. 이러한 포트는 기본 구성으로 정의된 포트이며 이 예에서는 포트가 변경되지 않는다고 가정합니다.
   - 프로토콜: TCP

분석 경로를 생성하고 결과를 몇 분 정도 기다립니다. 상태가 연결할 수 없음인 경우 분석 세부 정보를 열고 분석 탐색기에서 요청이 차단된 위치에 대한 자세한 정보를 검토합니다.

연결성 테스트가 통과되면 OS 수준에서 확인을 진행합니다.

ElastiCache 서비스 포트의 TCP 연결 검증: Amazon Linux에서 `Nping`는 패키지에서 사용할 수 `nmap` 있으며 ElastiCache 포트의 TCP 연결을 테스트하고 네트워크 왕복 시간을 제공하여 연결을 설정할 수 있습니다. 다음과 같이 이를 사용하여 ElastiCache 클러스터에 대한 네트워크 연결 및 현재 지연 시간을 검증합니다.

```
$ sudo nping --tcp -p 6379 example.xxxxxx.ng.0001.use1.cache.amazonaws.com

Starting Nping 0.6.40 ( http://nmap.org/nping ) at 2020-12-30 16:48 UTC
SENT (0.0495s) TCP ...
(Output suppressed )

Max rtt: 0.937ms | Min rtt: 0.318ms | Avg rtt: 0.449ms
Raw packets sent: 5 (200B) | Rcvd: 5 (220B) | Lost: 0 (0.00%)
Nping done: 1 IP address pinged in 4.08 seconds
```

기본적으로, nping은 5개의 프로브를 사이에 1초의 지연 시간을 두어 전송합니다. “-c” 옵션을 사용하여 프로브 수를 늘리고 “--delay“ 옵션을 사용하여 새 테스트를 전송하는 시간을 변경할 수 있습니다.

가 포함된 테스트가 nping 실패하고 VPC Reachability Analyzer 테스트가 통과된 경우 시스템 관리자에게 가능한 호스트 기반 방화벽 규칙, 비대칭 라우팅 규칙 또는 운영 체제 수준에서 발생할 수 있는 기타 제한을 검토하도록 요청합니다.

ElastiCache 콘솔에서 ElastiCache 클러스터 세부 정보에서 전송 중 암호화가 활성화되어 있는지 확인합니다. 전송 중 암호화가 활성화된 경우 다음 명령을 사용하여 TLS 세션을 설정할 수 있는지 확인합니다.

```
openssl s_client -connect example.xxxxxx.use1.cache.amazonaws.com:6379
```

연결 및 TLS 협상이 성공하면 광범위한 출력이 예상됩니다. 마지막 줄에서 사용할 수 있는 반환 코드를 확인하세요. 값이 0 (ok)여야 합니다. openssl이 다른 것을 반환하는 경우 https://www.openssl.org/docs/man1.0.2/man1/verify.html#DIAGNOSTICS에서 오류 이유를 확인합니다.

모든 인프라 및 운영 체제 테스트가 통과되었지만 애플리케이션이 여전히 에 연결할 수 없는 경우 애플리케이션 구성이 ElastiCache 설정을 준수하는지 ElastiCache확인합니다. 일반적인 실수는 다음과 같습니다.

- 애플리케이션이 ElastiCache 클러스터 모드를 지원하지 않으며 클러스터 모드 ElastiCache 가 활성화되어 있습니다.
- 애플리케이션이 TLS/를 지원하지 않으며 전송 중 암호화SSL ElastiCache 가 활성화되어 있습니다.
- 애플리케이션은 TLS/SSL를 지원하지만 올바른 구성 플래그 또는 신뢰할 수 있는 인증 기관이 없습니다.

네트워크 관련 제한 사항

- 최대 연결 수: 동시 연결에 대한 하드 제한이 있습니다. 각 ElastiCache 노드는 모든 클라이언트에서 최대 65,000개의 동시 연결을 허용합니다. 이 제한은 의 `CurrConnections` 지표를 통해 모니터링할 수 있습니다 CloudWatch. 그러나 클라이언트에는 아웃바운드 연결에 대한 제한이 있습니다. Linux의 경우 다음 명령을 사용하여 허용된 휘발성 포트 범위를 확인하세요.

  ```
  # sysctl net.ipv4.ip_local_port_range
  net.ipv4.ip_local_port_range = 32768 60999
  ```

  이전 예제에서는 동일한 소스에서 동일한 대상 IP(ElastiCache 노드) 및 포트로의 28231 연결이 허용됩니다. 다음 명령은 특정 ElastiCache 노드(IP 1.2.3.4)에 대한 연결 수를 보여줍니다.

  ```
  ss --numeric --tcp state connected "dst 1.2.3.4 and dport == 6379" | grep -vE
   '^State' | wc -l
  ```

  숫자가 너무 높으면 연결 요청을 처리하는 동안 시스템에 과부하가 걸릴 수 있습니다. 연결 처리를 개선하려면 연결 풀링이나 지속적인 연결과 같은 기술을 구현하는 것이 좋습니다. 가능한 경우 항상 연결 풀을 구성하여 최대 연결 수를 수백 개로 제한합니다. 또한 문제가 발생할 경우 연결 이탈을 방지하려면 시간 초과 또는 기타 연결 예외를 처리하는 백오프 로직이 필요합니다.

- 네트워크 트래픽 제한: CloudWatch Redis에 대한 다음 지표OSS를 확인하여 ElastiCache 노드에서 발생할 수 있는 네트워크 제한을 식별합니다.
  - `NetworkBandwidthInAllowanceExceeded/NetworkBandwidthOutAllowanceExceeded`: 처리량이 집계된 대역폭 제한을 초과하여 형성된 네트워크 패킷입니다.

    기본 노드에 한 바이트가 쓰여질 때마다 N개 복제본으로 복제됩니다. 여기서 N은 복제본 수입니다. 노드 유형이 작고, 여러 개의 복제본이 있으며 많은 쓰기 요청이 있는 클러스터는 복제 백로그를 처리하지 못할 수 있습니다. 이러한 경우 확장(노드 유형 변경), 스케일 아웃(클러스터 모드가 활성화된 클러스터에 샤드 추가), 복제본 수를 줄이기 또는 쓰기 수 최소화를 수행하는 것이 모범 사례입니다.
  - `NetworkConntrackAllowanceExceeded`: 노드에 할당된 모든 보안 그룹에서 추적되는 최대 연결 수를 초과했기 때문에 형성되는 패킷입니다. 이 기간 동안 새 연결이 실패할 가능성이 높습니다.
  - `NetworkPackets PerSecondAllowanceExceeded`: 초당 최대 패킷 수를 초과했습니다. 매우 크기가 작은 요청의 비율이 높은 워크로드는 최대 대역폭 전에 이 제한에 도달할 수 있습니다.

위의 지표는 네트워크 제한에 도달한 노드를 확인하는 이상적인 방법입니다. 그러나 제한은 네트워크 지표의 안정기로 식별할 수도 있습니다.

고평부가 장기간 관찰되면 복제 지연, 캐시에 사용되는 바이트 증가, 사용 가능한 메모리 삭제, 높은 스왑 및 CPU 사용량이 뒤따를 수 있습니다. Amazon EC2 인스턴스에는 ENA 드라이버 지표를 통해 추적할 수 있는 네트워크 제한도 있습니다. 향상된 네트워킹 지원이 포함된 Linux 인스턴스 및 ENA 드라이버 2.2.10 이상은 명령으로 제한 카운터를 검토할 수 있습니다.

```
# ethtool -S eth0 | grep "allowance_exceeded"
```

CPU 사용량

CPU 사용량 지표는 조사의 시작점이며, 다음 항목은 ElastiCache 측면에서 발생할 수 있는 문제를 좁히는 데 도움이 될 수 있습니다.

- Redis OSS SlowLogs: ElastiCache 기본 구성은 완료하는 데 10밀리초 이상 걸린 마지막 128개의 명령을 유지합니다. 느린 명령의 기록은 엔진 런타임 중에 유지되며 실패나 재시작 시 손실됩니다. 목록이 128개 항목에 도달하면 새 이벤트를 위한 공간을 확보하기 위해 이전 이벤트가 제거됩니다. 느린 이벤트 목록의 크기와 느린 것으로 간주되는 실행 시간은 사용자 지정 파라미터 그룹에서 `slowlog-max-len` 및 `slowlog-log-slower-than` 파라미터를 통해 수정할 수 있습니다. 슬로우 로그 목록을 검색하려면 엔진에 대해 `SLOWLOG GET 128`을 실행합니다. 여기서 128은 마지막 128개의 느린 명령을 보고합니다. 각 항목에는 다음과 같은 필드가 있습니다.

```
1) 1) (integer) 1 -----------> Sequential ID
   2) (integer) 1609010767 --> Timestamp (Unix epoch time)of the Event
   3) (integer) 4823378 -----> Time in microseconds to complete the command.
   4) 1) "keys" -------------> Command
      2) "*" ----------------> Arguments
   5) "1.2.3.4:57004"-> Source
```

위의 이벤트는 12월 26일 19:26:07에 발생했고UTC, 완료하는 데 4.8초(4.823ms)가 걸렸으며, 클라이언트 1.2.3.4에서 요청한 KEYS 명령으로 인해 발생했습니다.

Linux에서 타임스탬프를 date 명령으로 변환할 수 있습니다.

```
$ date --date='@1609010767'
Sat Dec 26 19:26:07 UTC 2020
```

Python 사용:

```
>>> from datetime import datetime
>>> datetime.fromtimestamp(1609010767)
datetime.datetime(2020, 12, 26, 19, 26, 7)
```

또는 Windows에서 PowerShell:

```
PS D:\Users\user> [datetimeoffset]::FromUnixTimeSeconds('1609010767')
DateTime      : 12/26/2020 7:26:07 PM
UtcDateTime
                : 12/26/2020 7:26:07 PM
LocalDateTime : 12/26/2020 2:26:07 PM
Date          : 12/26/2020 12:00:00 AM
Day           : 26
DayOfWeek
                : Saturday
DayOfYear     : 361
Hour          : 19
Millisecond   : 0
Minute        : 26
Month
                : 12
Offset        : 00:00:00Ticks         : 637446075670000000
UtcTicks
                : 637446075670000000
TimeOfDay     : 19:26:07
Year          : 2020
```

짧은 시간(1분 이하) 동안 발생한 많은 느린 명령이 우려의 이유입니다. 명령의 특성과 명령을 최적화할 수 있는 방법을 검토합니다(이전 예제 참조). O(1) 시간 복잡성이 있는 명령이 자주 보고되는 경우 앞서 CPU 언급한 사용량이 높은 다른 요인을 확인합니다.

- 지연 시간 지표: ElastiCache (Redis OSS)는 다양한 명령 클래스의 평균 지연 시간을 모니터링하는 CloudWatch 지표를 제공합니다. 데이터 포인트는 한 범주에 속하는 명령의 총 실행 횟수를 해당 기간의 총 실행 시간으로 나누어 계산합니다. 대기 시간 지표의 결과는 여러 명령의 집계임을 이해하는 것이 중요합니다. 단일 명령을 사용하면 지표에 큰 영향을 주지 않으며 시간 초과와 같은 예기치 않은 결과가 발생할 수 있습니다. 이러한 경우 느리 로그 이벤트가 보다 정확한 정보 소스가 될 수 있습니다. 다음 목록에는 사용 가능한 대기 시간 지표와 이에 영향을 주는 각 명령이 나와 있습니다.
  - EvalBasedCmdsLatency: Lua Script 명령 관련, `eval`, `evalsha`;

- GeoSpatialBasedCmdsLatency: geodist, geohash, geopos, georadius, georadiusbymember, geoadd;
- GetTypeCmdsLatency: 데이터 유형에 관계없이 명령 읽기
- HashBasedCmdsLatency: hexists, hget, hgetall, hkeys, hlen, hmget, hvals, hstrlen, hdel, hincrby, hincrbyfloat, hmset, hset, hsetnx;
- HyperLogLogBasedCmdsLatency: pfselftest, pfcount, pfdebug, pfadd, pfmerge;
- KeyBasedCmdsLatency: dump, , exists, keys, object, , , , , pttlrandomkeyttltype, , , delexpireexpireat, move, , persist, pexpire, pexpireat, rename, restoreK, renamenx, 등 다양한 데이터 유형에 따라 작동할 수 있는 명령 sort unlink
- ListBasedCmdsLatency: lindex, llen, lrange, blpop, brpop, brpoplpush, linsert, lpop, lpush, lpushx, lrem, lset, ltrim, rpop, rpoplpush, rpush, rpush;
- PubSubBasedCmdsLatency: 구독, 게시, pubsub, punsub, 구독, 구독 취소
- SetBasedCmdsLatency: scard, sdiff, sinter, sismember, smembers, srandmember, sunion, sadd, sdiffstore, sinterstore, smove, spop, srem, sunionstore;
- SetTypeCmdsLatency: 데이터 유형에 관계없이 명령 쓰기
- SortedSetBasedCmdsLatency: zcard, zcount, zrange, zrangebyscore, zrank, zrevrange, zrevrangebyscore, zrevrank, zscore, zrangebylex, zrevrangebylex, zlexcount, zadd. zincrby, zinterstore, zrem, zremrangebyrank, zremrangebyscore, zunionstore, zremrangebylex, zpopmax, zpopmin, bzpopmin, bzpopmax;
- StringBasedCmdsLatency: bitcount, get, getbit, getrange, mget, strlen, substr, bitpos, append, bitop, bitfield, decr, decrby, getset, incr, incrby, incrbyfloat, mset, msetnx, psetex, set, setbit, setex, setnx, setrange;
- StreamBasedCmdsLatency: xrange, xrevrange, xlen, xread, xpending, xinfo, xadd, xgroup, readgroup, xack, xclaim, xdel, xtrim, xsetid;

- Redis OSS 런타임 명령:
  - info commandstats: Redis OSS 엔진이 시작된 이후 실행된 명령 목록, 누적 실행 수, 총 실행 시간 및 명령당 평균 실행 시간을 제공합니다.
  - client list: 현재 연결된 클라이언트와 버퍼 사용량, 마지막으로 실행된 명령 등과 같은 관련 정보의 목록을 제공합니다.
- 백업 및 복제: 2.8.22 이전의 ElastiCache (Redis OSS) 버전은 포크 프로세스를 사용하여 백업을 생성하고 복제본과의 전체 동기화를 처리합니다. 이 방법은 쓰기 집약적인 사용 사례에서 상당한 메모리 오버헤드가 발생시킬 수 있습니다.

ElastiCache Redis OSS 2.8.22부터는 포크리스 백업 및 복제 방법을 AWS 도입했습니다. 새로운 방법은 실패를 방지하기 위해 쓰기를 지연시킬 수 있습니다. 두 방법 모두 CPU 사용률이 높아지고 응답 시간이 길어지므로 실행 중에 클라이언트 제한 시간이 발생할 수 있습니다. 백업 기간 중에 클라이언트 장애가 발생하거나 이 기간 중에 `SaveInProgress` 지표가 1이었는지 항상 확인하세요. 클라이언트 문제 또는 백업 실패의 가능성을 최소화하기 위해 사용률이 낮은 기간으로 백업 기간을 예약하는 것이 좋습니다.

서버 측에서 연결이 종료되는 경우

기본 ElastiCache (Redis OSS) 구성은 클라이언트 연결을 무기한으로 설정합니다. 그러나 경우에 따라 연결 종료가 필요할 수 있습니다. 예:

- 클라이언트 애플리케이션의 버그로 인해 연결이 손실되고 유휴 상태로 설정이 유지될 수 있습니다. 이것을 “연결 누수“라고 하며 결과적으로 `CurrConnections` 지표에서 관찰되는 설정된 연결의 수가 지속적으로 증가합니다. 이 동작은 클라이언트 또는 ElastiCache 측에서 포화될 수 있습니다. 클라이언트 측에서 즉각적인 수정이 불가능한 경우 일부 관리자는 ElastiCache 파라미터 그룹에서 “타임아웃” 값을 설정합니다. 시간 초과는 유휴 연결이 지속되도록 허용된 기간(초)입니다. 클라이언트가 해당 기간에 요청을 제출하지 않으면 Redis OSS 엔진은 연결이 제한 시간에 도달하는 즉시 연결을 종료합니다. 시간 초과 값이 작으면 불필요한 연결 끊기가 발생할 수 있으며 클라이언트가 연결을 올바르게 처리하고 다시 연결해야 하므로 지연이 발생합니다.
- 키를 저장하는 데 사용되는 메모리는 클라이언트 버퍼와 공유됩니다. 크기가 큰 요청이나 응답이 있는 느린 클라이언트는 버퍼를 처리하기 위해 상당한 양의 메모리를 요구할 수 있습니다. 기본 ElastiCache (Redis OSS) 구성은 일반 클라이언트 출력 버퍼의 크기를 제한하지 않습니다. `maxmemory` 제한에 도달하면 엔진은 버퍼 사용량을 충족하기 위해 항목을 제거하려고 시도합니다. 메모리가 매우 낮은 조건에서는 ElastiCache (Redis OSS)가 대용량 클라이언트 출력 버퍼를 사용하는 클라이언트의 연결을 해제하여 메모리를 확보하고 클러스터의 상태를 유지할 수 있습니다.

  사용자 지정 구성으로 클라이언트 버퍼의 크기를 제한할 수 있으며 이 제한에 도달한 클라이언트는 연결이 끊어집니다. 그러나 클라이언트는 예기치 않은 연결 해제를 처리할 수 있어야 합니다. 일반 클라이언트의 버퍼 크기를 처리하는 파라미터는 다음과 같습니다.

  - client-query-buffer-limit: 단일 입력 요청의 최대 크기
  - client-output-buffer-limit-normal-soft-limit: 클라이언트 연결에 대한 소프트 제한입니다. 가 에 정의된 시간보다 초 이상 소프트 제한을 초과 client-output-buffer-limitnormal-soft-seconds 하거나 하드 제한에 도달하면 연결이 종료됩니다.
  - client-output-buffer-limit-normal-soft-seconds: client-output-buffer-limit-를 초과하는 연결에 허용되는 시간normal-soft-limit

- client-output-buffer-limit-normal-hard-limit: 이 제한에 해당하는 연결이 즉시 종료됩니다.

일반 클라이언트 버퍼 외에도, 다음 옵션은 복제본 노드 및 Pub/Sub(게시/구독) 클라이언트에 대한 버퍼를 제어합니다.

- client-output-buffer-limit-replica-hard-limit;
- client-output-buffer-limit-replica-soft-seconds;
- client-output-buffer-limit-replica-hard-limit;
- client-output-buffer-limit-pubsub-soft-limit;
- client-output-buffer-limit-pubsub-soft-seconds;
- client-output-buffer-limit-pubsub-hard-limit;

Amazon EC2 인스턴스에 대한 클라이언트 측 문제 해결

클라이언트 측의 로드 및 응답성도 에 대한 요청에 영향을 미칠 수 있습니다 ElastiCache. EC2 간헐적인 연결 또는 제한 시간 문제를 해결하는 동안 인스턴스 및 운영 체제 제한을 주의 깊게 검토해야 합니다. 관찰할 몇 가지 핵심 사항:

- CPU:
  - EC2 인스턴스 CPU 사용량: CPU 이 포화되지 않았거나 100%에 가깝지 않은지 확인합니다. 과거 분석은 를 통해 수행할 수 CloudWatch있지만 데이터 포인트 세분화는 1분(세부 모니터링 활성화됨) 또는 5분이라는 점에 유의하세요.
  - 버스트 가능한 EC2 인스턴스를 사용하는 경우 CPU 크레딧 잔고가 고갈되지 않았는지 확인하세요. 이 정보는 `CPUCreditBalance` CloudWatch 지표에서 확인할 수 있습니다.
  - 사용량이 많으면 의 100% 사용률을 반영하지 않고 시간 초과가 발생할 CPU 수 있습니다 CloudWatch. 이러한 경우에는 `top`, `ps` 및 `mpstat` 같은 운영 체제 도구를 사용하여 실시간 모니터링을 수행해야 합니다.
- 네트워크
  - 네트워크 처리량이 인스턴스 용량에 따라 허용 가능한 값 이하인지 확인합니다. 자세한 내용은 Amazon EC2 인스턴스 유형을 참조하세요.
  - `ena` Enhanced Network 드라이버를 사용하는 인스턴스의 경우 ena statistics에서 시간 초과 또는 제한 초과를 확인합니다. 다음 통계는 네트워크 제한 도달 상태를 확인하는 데 유용합니다.
    - `bw_in_allowance_exceeded`/`bw_out_allowance_exceeded`: 과도한 인바운드 또는 아웃바운드 처리량으로 인해 형성된 패킷 수

- conntrack_allowance_exceeded: 보안 그룹 연결 추적 제한으로 인해 삭제된 패킷 수. 이 제한에 도달하면 새 연결이 실패합니다.
- linklocal_allowance_exceeded: VPC 를 NTP 통해 인스턴스 메타데이터에 대한 과도한 요청으로 인해 삭제된 패킷 수입니다DNS. 이 제한은 모든 서비스에 대해 초당 1024패킷입니다.
- pps_allowance_exceeded: 과도한 초당 패킷 수 비율로 인해 손실된 패킷 수. 네트워크 트래픽이 초당 수천 개 또는 수백만 개의 매우 작은 요청으로 구성된 경우 PPS 제한이 발생할 수 있습니다. ElastiCache 트래픽은 MGET 대신 와 같이 한 번에 여러 작업을 수행하는 파이프라인 또는 명령을 통해 네트워크 패킷을 더 잘 사용하도록 최적화할 수 있습니다GET.

단일 요청을 완료하는 데 걸리는 시간 분석

- 네트워크에서: Tcpdump 및 Wireshark (명령줄의 tshark)는 요청이 네트워크를 이동하는 데 걸린 시간을 이해하고, ElastiCache 엔진을 쳐서 반환을 받을 수 있는 편리한 도구입니다. 다음 예제에서는 다음과 같은 명령을 사용하여 생성한 단일 요청을 강조 표시합니다.

```
$ echo ping | nc example.xxxxxx.ng.0001.use1.cache.amazonaws.com 6379
+PONG
```

위의 명령과 병행하여 tcpdump가 실행 중이었고 반환되었습니다.

```
$ sudo tcpdump -i any -nn port 6379 -tt
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
1609428918.917869 IP 172.31.11.142.40966
    > 172.31.11.247.6379: Flags [S], seq 177032944, win 26883, options [mss
 8961,sackOK,TS val 27819440 ecr 0,nop,wscale 7], length 0
1609428918.918071 IP 172.31.11.247.6379 > 172.31.11.142.40966: Flags [S.], seq
 53962565, ack 177032945, win
    28960, options [mss 1460,sackOK,TS val 3788576332 ecr 27819440,nop,wscale 7],
 length 0
1609428918.918091 IP 172.31.11.142.40966 > 172.31.11.247.6379: Flags [.], ack 1, win
 211, options [nop,nop,TS val 27819440 ecr 3788576332], length 0
1609428918.918122
    IP 172.31.11.142.40966 > 172.31.11.247.6379: Flags [P.], seq 1:6, ack 1, win 211,
 options [nop,nop,TS val 27819440 ecr 3788576332], length 5: RESP "ping"
1609428918.918132 IP 172.31.11.142.40966 > 172.31.11.247.6379: Flags [F.], seq 6, ack
    1, win 211, options [nop,nop,TS val 27819440 ecr 3788576332], length 0
1609428918.918240 IP 172.31.11.247.6379 > 172.31.11.142.40966: Flags [.], ack 6, win
 227, options [nop,nop,TS val 3788576332 ecr 27819440], length 0
```

```
1609428918.918295
    IP 172.31.11.247.6379 > 172.31.11.142.40966: Flags [P.], seq 1:8, ack 7, win 227,
 options [nop,nop,TS val 3788576332 ecr 27819440], length 7: RESP "PONG"
1609428918.918300 IP 172.31.11.142.40966 > 172.31.11.247.6379: Flags [.], ack 8, win
    211, options [nop,nop,TS val 27819441 ecr 3788576332], length 0
1609428918.918302 IP 172.31.11.247.6379 > 172.31.11.142.40966: Flags [F.], seq 8, ack
 7, win 227, options [nop,nop,TS val 3788576332 ecr 27819440], length 0
1609428918.918307
    IP 172.31.11.142.40966 > 172.31.11.247.6379: Flags [.], ack 9, win 211, options
 [nop,nop,TS val 27819441 ecr 3788576332], length 0
^C
10 packets captured
10 packets received by filter
0 packets dropped by kernel
```

위의 출력에서 TCP3방향 핸드셰이크가 222마이크로초(918091~917869) 내에 완료되었고 ping 명령이 제출되어 173마이크로초(918295~918122) 내에 반환되었는지 확인할 수 있습니다.

요청부터 연결을 닫는 데까지 438마이크로초(918307 - 917869)가 걸렸습니다. 이러한 결과를 통해 네트워크 및 엔진 응답 시간이 양호하다고 확신할 수 있으며 조사를 다른 구성 요소에 집중할 수 있습니다.

- 운영 체제에서 Strace를 사용하여 OS 수준의 시간 간격을 식별할 수 있습니다. 실제 애플리케이션의 분석에는 보다 광범위하고 전문화된 애플리케이션 프로파일러 또는 디버거를 사용하는 것이 좋습니다. 다음 예제에서는 기본 운영 체제 구성 요소가 예상대로 작동하는지 여부만 보여 주며, 예상대로 작동하지 않는 경우 추가 조사가 필요할 수 있습니다. 동일한 Redis OSS PING 명령을 와 함께 사용하면 다음과 같은 strace 이점이 있습니다.

```
$ echo ping | strace -f -tttt -r -e trace=execve,socket,open,recvfrom,sendto
 nc example.xxxxxx.ng.0001.use1.cache.amazonaws.com (http://
example.xxxxxx.ng.0001.use1.cache.amazonaws.com/)
    6379
1609430221.697712 (+ 0.000000) execve("/usr/bin/nc", ["nc",
 "example.xxxxxx.ng.0001.use"..., "6379"], 0x7fffede7cc38 /* 22 vars */) = 0
1609430221.708955 (+ 0.011231) socket(AF_UNIX, SOCK_STREAM|SOCK_CLOEXEC|
SOCK_NONBLOCK, 0) = 3
1609430221.709084
    (+ 0.000124) socket(AF_UNIX, SOCK_STREAM|SOCK_CLOEXEC|SOCK_NONBLOCK, 0) = 3
1609430221.709258 (+ 0.000173) open("/etc/nsswitch.conf", O_RDONLY|O_CLOEXEC) = 3
1609430221.709637 (+ 0.000378) open("/etc/host.conf", O_RDONLY|O_CLOEXEC) = 3
1609430221.709923
    (+ 0.000286) open("/etc/resolv.conf", O_RDONLY|O_CLOEXEC) = 3
```

```
1609430221.711365 (+ 0.001443) open("/etc/hosts", O_RDONLY|O_CLOEXEC) = 3
1609430221.713293 (+ 0.001928) socket(AF_INET, SOCK_DGRAM|SOCK_CLOEXEC|SOCK_NONBLOCK,
 IPPROTO_IP) = 3
1609430221.717419
    (+ 0.004126) recvfrom(3, "\362|
\201\200\0\1\0\2\0\0\0\0\rnotls20201224\6tihew"..., 2048, 0, {sa_family=AF_INET,
 sin_port=htons(53), sin_addr=inet_addr("172.31.0.2")}, [28->16]) = 155
1609430221.717890 (+ 0.000469) recvfrom(3,
 "\204\207\201\200\0\1\0\1\0\0\0\0\rnotls20201224\6tihew"...,
    65536, 0, {sa_family=AF_INET, sin_port=htons(53),
 sin_addr=inet_addr("172.31.0.2")}, [28->16]) = 139
1609430221.745659 (+ 0.027772) socket(AF_INET, SOCK_STREAM, IPPROTO_TCP) = 3
1609430221.747548 (+ 0.001887) recvfrom(0, 0x7ffcf2f2ca50, 8192,
    0, 0x7ffcf2f2c9d0, [128]) = -1 ENOTSOCK (Socket operation on non-socket)
1609430221.747858 (+ 0.000308) sendto(3, "ping\n", 5, 0, NULL, 0) = 5
1609430221.748048 (+ 0.000188) recvfrom(0, 0x7ffcf2f2ca50, 8192, 0, 0x7ffcf2f2c9d0,
 [128]) = -1 ENOTSOCK
    (Socket operation on non-socket)
1609430221.748330 (+ 0.000282) recvfrom(3, "+PONG\r\n", 8192, 0, 0x7ffcf2f2c9d0,
 [128->0]) = 7
+PONG
1609430221.748543 (+ 0.000213) recvfrom(3, "", 8192, 0, 0x7ffcf2f2c9d0, [128->0]) = 0
1609430221.752110
    (+ 0.003569) +++ exited with 0 +++
```

위의 예제에서 명령을 완료하는 데 54밀리초 이상이 걸렸습니다(752110 - 697712 = 54398마이크로초).

nc를 인스턴스화하고 이름 확인(697712에서 717890으로)을 수행하는 데 약 20ms의 상당한 시간이 소요되었으며, 그 후 TCP 소켓을 생성하는 데 2ms(745659에서 747858로), 요청에 대한 응답을 제출하고 수신하기 위해 0.4ms(747858에서 748330으로)가 필요했습니다.

관련 항목

- the section called “모범 사례 및 캐싱 전략”

Amazon의 보안 ElastiCache

의 클라우드 보안 AWS 이 최우선 순위입니다. AWS 고객은 가장 보안에 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 클라우드 내 보안 및 클라우드의 보안으로 설명합니다.

- 클라우드 보안 - AWS 는 AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다. 는 안전하게 사용할 수 있는 서비스 AWS 도 제공합니다. 서드 파티 감사원은 정기적으로 AWS 규정 준수 프로그램의 일환으로 보안 효과를 테스트하고 검증합니다. Amazon 에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램 의 범위 내 서비스를 ElastiCache참조하세요. AWS
- 클라우드의 보안 - 사용자의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 귀하는 귀사의 데이터의 민감도, 귀사의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 Amazon 를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다 ElastiCache. 다음 주제에서는 보안 및 규정 준수 목표에 맞게 Amazon ElastiCache 을 구성하는 방법을 보여줍니다. 또한 Amazon ElastiCache 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법도 알아봅니다.

주제

Amazon의 데이터 보호 ElastiCache

AWS 공동 책임 모델 AWS ElastiCache ()의 데이터 보호에 적용됩니다ElastiCache. 이 모델에 설명된 대로 AWS 는 모든 AWS 클라우드를 실행하는 글로벌 인프라를 보호할 책임이 있습니다. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 이 콘텐츠에는 사용하는 AWS 서비스에 대한 보안 구성 및 관리 작업이 포함됩니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 섹션을 FAQ참조하세요.

데이터 보호를 위해 AWS 계정 자격 증명을 보호하고 AWS Identity and Access Management ()를 사용하여 개별 계정을 설정하는 것이 좋습니다IAM. 이러한 방식에서는 각 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다단계 인증(MFA)을 사용합니다.
- TLS 를 사용하여 AWS 리소스와 통신합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail.
- 암호화 AWS 솔루션과 서비스 내 AWS 모든 기본 보안 제어를 사용합니다.
- Amazon S3에 저장된 개인 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용합니다.

이름 필드와 같은 자유 형식 필드에 고객 계정 번호와 같은 중요 식별 정보를 절대 입력하지 마십시오. 여기에는 콘솔, 또는 를 사용하여 ElastiCache 또는 다른 AWS 서비스를 사용하는 경우가 포함됩니다 API AWS CLI AWS SDKs. ElastiCache 또는 다른 서비스에 입력하는 모든 데이터는 진단 로그에 포함되도록 픽업될 수 있습니다. 외부 서버에 URL를 제공하는 경우 해당 서버에 대한 요청을 검증URL하기 위해 에 보안 인증 정보를 포함하지 마세요.

주제

- Amazon의 데이터 보안 ElastiCache

Amazon의 데이터 보안 ElastiCache

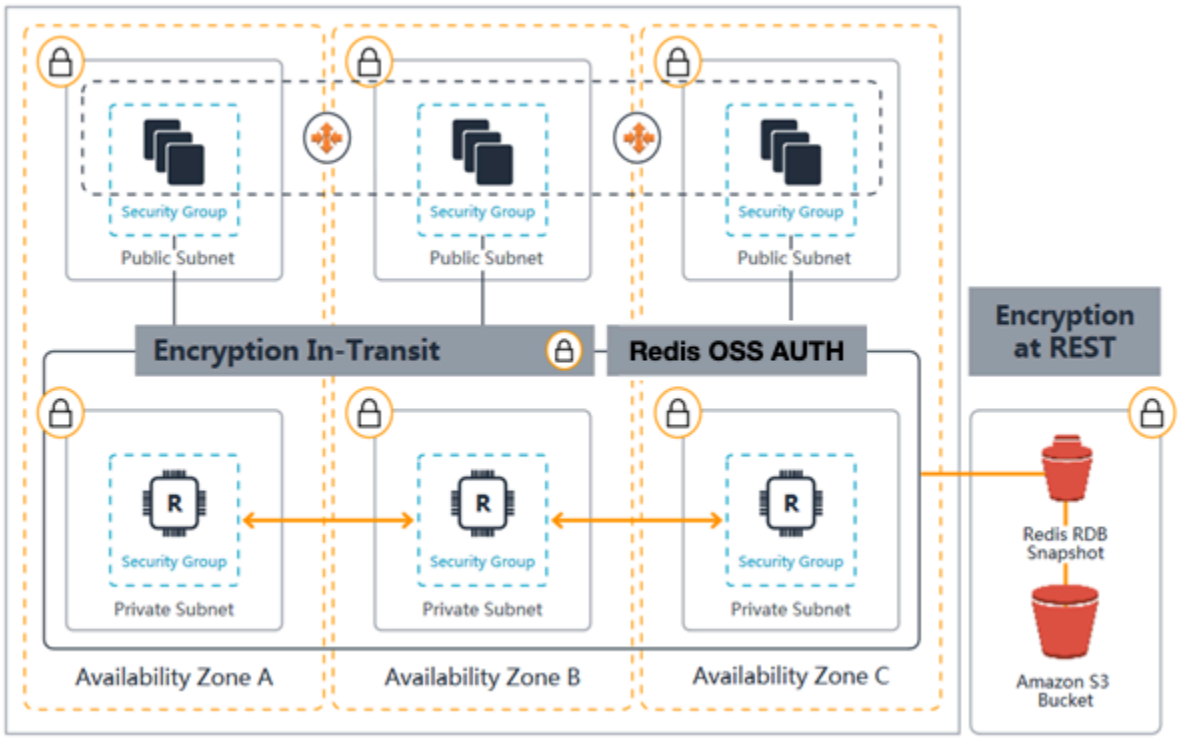
데이터를 안전하게 유지하기 위해 Amazon ElastiCache 과 Amazon은 서버에서 데이터에 대한 무단 액세스를 방지하는 메커니즘을 EC2 제공합니다.

Amazon ElastiCache (Memcached)은 Memcached 버전 1.6.12 이상을 실행하는 캐시의 데이터에 대한 암호화 기능을 제공합니다.

Amazon ElastiCache with Valkey and RedisOSS는 Valkey 7.2 이상 및 Redis OSS 버전 3.2.6(에 대해 예약됨EOL, Redis OSS 버전 수명 종료 일정 참조), 4.0.10 이상을 실행하는 캐시의 데이터에 대한 암호화 기능을 제공합니다. ElastiCache 또한 Amazon은 IAM 또는 Valkey 및 Redis OSS 를 사용하여 사용자 인증 AUTH및 역할 기반 액세스 제어()를 사용하여 사용자 작업 권한 부여를 지원합니다RBAC.

- 전송 중 암호화는 클러스터 내 노드 사이 또는 캐시와 애플리케이션 사이와 같이 한 위치에서 다른 위치로 데이터가 이동 중인 경우 항상 데이터를 암호화합니다.
- 미사용 데이터 암호화는 동기화 및 백업 작업 중 디스크 내 데이터를 암호화합니다.

ElastiCache 는 IAM 및 Valkey 및 Redis OSS AUTH 명령을 사용하여 사용자를 인증하고 역할 기반 액세스 제어()를 사용하여 사용자 작업을 승인합니다RBAC.



ElastiCache Valkey 및 Redis OSS 보안 다이어그램 사용

주제

- ElastiCache 전송 중 암호화(TLS)
- 의 저장 시 암호화 ElastiCache
- 인증 및 권한 부여

ElastiCache 전송 중 암호화(TLS)

데이터를 안전하게 유지하기 위해 Amazon ElastiCache 과 Amazon은 서버에서 데이터에 대한 무단 액세스를 방지하는 메커니즘을 EC2 제공합니다. 전송 중 암호화 기능을 제공하여 한 위치에서 다른 위치로 이동할 때 데이터를 보호하는 데 사용할 수 있는 도구를 ElastiCache 제공합니다.

모든 Valkey 또는 Redis OSS 서버리스 캐시에는 전송 중 암호화가 활성화되어 있습니다. 자체 설계된 클러스터의 경우 복제 그룹을 생성할 때 파라미터를 `true` (CLI:`--transit-encryption-enabled`)`TransitEncryptionEnabled`로 설정하여 복제 그룹에서 전송 중 암호화를 활성화할 수 있습니다. AWS Management Console, AWS CLI또는 를 사용하여 복제 그룹을 생성하든 관계없이 이 작업을 수행할 수 있습니다 ElastiCache API.

모든 서버리스 캐시에는 전송 중 암호화가 활성화되어 있습니다. 자체 설계된 클러스터의 경우 (CLI: `--transit-encryption-enabled`) 작업을 사용하여 캐시 클러스터를 생성할 때 파라미터를 `true` (: `CreateCacheClusterCLIcreate-cache-cluster`)`TransitEncryptionEnabled`로 설정하여 캐시 클러스터에서 전송 중 암호화를 활성화할 수 있습니다.

주제

전송 중 데이터 암호화 개요

Amazon 전송 ElastiCache 중 암호화는 한 위치에서 다른 위치로 전송 중일 때 가장 취약한 지점에서 데이터의 보안을 강화할 수 있는 기능입니다. 엔드포인트에서 데이터를 암호화 및 해독하기 위해 몇 가지 처리가 필요하기 때문에 전송 중 데이터 암호화를 활성화하면 성능에 어느 정도 영향이 있을 수 있습니다. 사용 사례에 대한 성능 영향을 파악하기 위해서는 전송 중 데이터 암호화를 사용한 상태와 사용하지 않은 상태에서 데이터를 벤치마크해야 합니다.

ElastiCache 전송 중 암호화는 다음 기능을 구현합니다.

- 암호화된 클라이언트 연결 - 캐시 노드에 대한 클라이언트 연결이 TLS 암호화됩니다.
- 암호화된 서버 연결 - 클러스터의 노드 간에 이동하는 데이터는 암호화됩니다.
- 서버 인증 - 클라이언트가 자신이 올바른 서버에 연결 중임을 인증할 수 있습니다.
- 클라이언트 인증 - Valkey 및 Redis OSS AUTH 기능을 사용하여 서버가 클라이언트를 인증할 수 있습니다.

전송 중 암호화 조건(Valkey 및 RedisOSS)

자체 설계된 클러스터 구현을 계획할 때는 Amazon 전송 ElastiCache 중 암호화에 대한 다음 제약 조건을 염두에 두어야 합니다.

- 전송 중 암호화는 Valkey 7.2 이상 및 Redis OSS 버전 3.2.6, 4.0.10 이상을 실행하는 복제 그룹에서 지원됩니다.
- 기존 클러스터에 대한 전송 중 암호화 설정 수정은 Valkey 7.2 이상 및 Redis OSS 버전 7 이상을 실행하는 복제 그룹에서 지원됩니다.
- 전송 중 암호화는 Amazon 에서 실행되는 복제 그룹에 대해서만 지원됩니다VPC.
- 전송 중 암호화는 M1, M2 노드 유형을 실행하는 복제 그룹에서는 지원되지 않습니다.

  자세한 내용은 지원되는 노드 유형 단원을 참조하십시오.
- 전송 중 데이터 암호화는 `TransitEncryptionEnabled` 파라미터를 명시적으로 `true`로 설정해 활성화합니다.
- 캐싱 클라이언트가 TLS 연결을 지원하고 클라이언트 구성에서 활성화했는지 확인합니다.
- 이전 TLS 1.0 및 TLS 1.1의 사용은 ElastiCache 버전 6 이상의 모든 AWS 리전에서 더 이상 사용되지 않습니다. ElastiCache 는 TLS 2025년 5월 8일까지 1.0 및 1.1을 계속 지원합니다. 고객은 해당 날짜 이전에 클라이언트 소프트웨어를 업데이트해야 합니다.

전송 중 암호화 조건(Memcached)

자체 설계된 클러스터 구현을 계획할 때는 Amazon 전송 ElastiCache 중 암호화에 대한 다음 제약 조건을 염두에 두어야 합니다.

- 전송 중 데이터 암호화는 Memcached 버전 1.6.12 이상 버전을 실행 중인 클러스터에서 지원됩니다.
- 전송 중 암호화는 전송 계층 보안(TLS) 버전 1.2 및 1.3을 지원합니다.
- 전송 중 암호화는 Amazon 에서 실행되는 클러스터에만 지원됩니다VPC.
- 전송 중 암호화는 M1, M2, M3, R3, T2 노드 유형을 실행하는 복제 그룹에서는 지원되지 않습니다.

  자세한 내용은 지원되는 노드 유형 단원을 참조하십시오.
- 전송 중 데이터 암호화는 `TransitEncryptionEnabled` 파라미터를 명시적으로 `true`로 설정해 활성화합니다.
- 클러스터를 생성하는 경우에만 클러스터에서 전송 중 데이터 암호화를 사용 설정할 수 있습니다. 클러스터를 수정하여 전송 중 데이터 암호화 켜기 및 끄기를 전환할 수 없습니다.
- 캐싱 클라이언트가 TLS 연결을 지원하고 클라이언트 구성에서 활성화했는지 확인합니다.

전송 중 데이터 암호화 모범 사례

- 엔드포인트에서 데이터를 암호화 및 해독하기 위해 처리가 필요하기 때문에 전송 중 데이터 암호화를 구현하면 성능이 저하될 수 있습니다. 이러한 암호화가 구현 성능에 미치는 영향을 확인하려면 전송 중 데이터 암호화와 데이터를 암호화하지 않은 경우를 비교해 벤치마크하세요.
- 새 연결을 생성하는 데 비용이 많이 들 수 있으므로 TLS 연결을 유지하여 전송 중 암호화의 성능 영향을 줄일 수 있습니다.

추가 Valkey 및 Redis OSS 옵션

Valkey 및 Redis에 사용할 수 있는 옵션에 대한 자세한 내용은 후속 링크를 OSS참조하세요.

- 의 저장 시 암호화 ElastiCache
- Valkey 및 Redis OSS AUTH 명령을 사용한 인증
- 역할 기반 액세스 제어(RBAC)
- Amazon VPCs 및 ElastiCache 보안
- Amazon용 자격 증명 및 액세스 관리 ElastiCache

Memcached에 대한 전송 중 암호화 활성화

AWS 관리 콘솔을 사용하여 Memcached 클러스터를 생성하는 경우 전송 중 데이터 암호화를 사용 설정하려면 다음을 선택합니다.

- 엔진으로 Memcached를 선택합니다.
- 엔진 버전 1.6.12 이상을 선택합니다.
- Encryption in transit(전송 중 데이터 암호화)에서 Enable(사용 설정)을 선택합니다.

프로세스는 섹션을 참조하세요 step-by-stepValkey 또는 Redis용 클러스터 생성 OSS.

전송 중 데이터 암호화 사용 설정

모든 서버리스 캐시에는 전송 중 암호화가 활성화되어 있습니다. 자체 설계된 클러스터에서 , AWS Management Console AWS CLI또는 를 사용하여 전송 중 암호화를 활성화할 ElastiCache 수 있습니다 API.

를 사용하여 전송 중 암호화 활성화 AWS Management Console

를 사용하여 새 자체 설계된 클러스터에 대한 전송 중 암호화 활성화 AWS Management Console

자체 클러스터를 설계할 때 '간편한 생성' 메서드를 사용하는 '개발 및 테스트' 및 '프로덕션' 구성에서는 전송 중 암호화가 활성화됩니다. 구성을 직접 선택할 때 다음과 같이 진행합니다.

- 다음 엔진 버전을 선택합니다. 3.2.6, 4.0.10 또는 이후 버전.
- 전송 중 암호화의 활성화 옵션 옆에서 확인란을 클릭합니다.

step-by-step 프로세스는 다음을 참조하세요.

- Valkey(클러스터 모드 비활성화됨) 클러스터 생성(콘솔)
- Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 생성(콘솔)

를 사용하여 기존 자체 설계된 클러스터에 대한 전송 중 암호화 활성화 AWS Management Console

전송 중 암호화 활성화는 두 단계 프로세스이므로, 먼저 전송 암호화 모드를 `preferred`(으)로 설정해야 합니다. 이 모드를 사용하면 Valkey 또는 Redis OSS 클라이언트가 암호화된 연결과 암호화되지 않은 연결을 모두 사용하여 연결할 수 있습니다. 암호화된 연결을 사용하도록 모든 Valkey 또는 Redis OSS 클라이언트를 마이그레이션한 후 클러스터 구성을 수정하여 전송 암호화 모드를 로 설정할 수 있

습니다`required`. 전송 암호화 모드를 `required`(으)로 설정하면 암호화되지 않은 모든 연결이 끊어지고 암호화된 연결만 허용됩니다.

전송 암호화 모드를기본 설정으로 설정

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 왼쪽에 있는 탐색 창에 나열된 리소스에서 ElastiCache Valkey 캐시 또는 Redis OSS 캐시를 선택합니다.
3. 업데이트할 캐시를 선택합니다.
4. 작업 드롭다운을 선택한 다음, 수정을 선택합니다.
5. 보안 섹션, 전송 중 암호화에서 활성화를 선택합니다.
6. 전송 암호화 모드로 선호를 선택합니다.
7. 변경 사항 미리 보기를 선택하여 변경 사항을 저장합니다.

암호화된 연결을 사용하도록 모든 Valkey 또는 Redis OSS 클라이언트를 마이그레이션한 후:

Transit 암호화 모드를필수로 설정

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 왼쪽에 있는 탐색 창에 나열된 리소스에서 ElastiCache Valkey 캐시 또는 Redis OSS 캐시를 선택합니다.
3. 업데이트할 캐시를 선택합니다.
4. 작업 드롭다운을 선택한 다음, 수정을 선택합니다.
5. 보안 섹션에서 전송 암호화 모드로 필수를 선택합니다.
6. 변경 사항 미리 보기를 선택하여 변경 사항을 저장합니다.

를 사용하여 전송 중 암호화 활성화 AWS CLI

를 사용하여 Valkey 또는 Redis OSS 복제 그룹을 생성할 때 전송 중 암호화를 활성화하려면 파라미터를 AWS CLI사용합니다`transit-encryption-enabled`.

Valkey 또는 Redis에 대해 새 자체 설계된 클러스터에서 전송 중 암호화 활성화OSS(클러스터 모드 비활성화됨)(CLI)

AWS CLI 작업`create-replication-group`과 다음 파라미터를 사용하여 전송 중 암호화가 활성화된 복제본을 사용하여 Valkey 또는 Redis OSS 복제 그룹을 생성합니다.

키 파라미터:

- **`--engine`**- `valkey` 또는 이어야 합니다`redis`.
- **`--engine-version`**—엔진이 Redis 인 경우 는 3.2.6, 4.0.10 이상이어야 OSS합니다.
- **`--transit-encryption-enabled`** - 필수입니다. 전송 중 데이터 암호화를 활성화하면 `--cache-subnet-group` 파라미터의 값도 제공해야 합니다.
- **`--num-cache-clusters`** - 1 이상이어야 합니다. 이 파라미터의 최대값은 6입니다.

자세한 내용은 다음 자료를 참조하세요.

- Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹을 처음부터 생성(AWS CLI)
- create-replication-group

Valkey 또는 Redis에 대해 새 자체 설계된 클러스터에서 전송 중 암호화 활성화OSS(클러스터 모드 활성화됨)(CLI)

AWS CLI 작업`create-replication-group`과 다음 파라미터를 사용하여 전송 중 암호화가 활성화된 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹을 생성합니다.

키 파라미터:

- **`--engine`**- `valkey` 또는 이어야 합니다`redis`.
- **`--engine-version`**—엔진이 Redis 인 경우 는 3.2.6, 4.0.10 이상이어야 OSS합니다.
- **`--transit-encryption-enabled`** - 필수입니다. 전송 중 데이터 암호화를 활성화하면 `--cache-subnet-group` 파라미터의 값도 제공해야 합니다.
- 다음 파라미터 세트 중 하나를 사용해 복제 그룹의 노드 그룹 구성을 지정합니다.
  - **`--num-node-groups`** - 이 복제 그룹의 샤드(노드 그룹) 수를 지정합니다. 이 파라미터의 최대값은 500입니다.

    `--replicas-per-node-group` - 각 노드 그룹의 복제 노드 수를 지정합니다. 여기서 지정된 값은 복제 그룹의 모든 샤드에 적용됩니다. 이 파라미터의 최대값은 5입니다.

- **`--node-group-configuration`** - 각 샤드의 구성을 독립적으로 지정합니다.

자세한 내용은 다음 자료를 참조하세요.

- 처음부터 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹 생성(AWS CLI)
- create-replication-group

AWS CLI을(를) 사용하여 기존 클러스터에 전송 중 암호화 활성화

전송 중 암호화 활성화는 두 단계 프로세스이므로, 먼저 전송 암호화 모드를 `preferred`(으)로 설정해야 합니다. 이 모드를 사용하면 Valkey 또는 Redis OSS 클라이언트가 암호화된 연결과 암호화되지 않은 연결을 모두 사용하여 연결할 수 있습니다. 암호화된 연결을 사용하도록 모든 Valkey 또는 Redis OSS 클라이언트를 마이그레이션한 후 클러스터 구성을 수정하여 전송 암호화 모드를 로 설정할 수 있습니다`required`. 전송 암호화 모드를 `required`(으)로 설정하면 암호화되지 않은 모든 연결이 끊어지고 암호화된 연결만 허용됩니다.

AWS CLI 작업 `modify-replication-group` 및 다음 파라미터를 사용하여 전송 중 암호화가 비활성화된 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹을 업데이트합니다.

전송 중 암호화를 활성화하려면

1. 다음 파라미터를 `preferred`사용하여 로 설정 transit-encryption-mode
   - **`--transit-encryption-enabled`** - 필수입니다.
   - **`--transit-encryption-mode`** - `preferred`(으)로 설정해야 합니다.
2. 다음 파라미터를 `required`사용하여 를 로 설정합니다 transit-encryption-mode.
   - **`--transit-encryption-enabled`** - 필수입니다.
   - **`--transit-encryption-mode`** - `required`(으)로 설정해야 합니다.

valkey-cli를 사용하여 전송 중 암호화를 사용하여 ElastiCache (Valkey) 또는 Amazon ElastiCache (RedisOSS)에 연결

전송 중 암호화로 활성화된 ElastiCache (Redis OSS) 캐시에서 데이터에 액세스하려면 Secure Socket Layer()로 작업하는 클라이언트를 사용합니다`SSL`. Amazon Linux 및 Amazon Linux 2에서 TLS/SSL와 함께 valkey-cli를 사용할 수도 있습니다. 클라이언트가 를 지원하지 않는 경우 클라이언트 호스트의 `stunnel` 명령을 사용하여 Redis OSS 노드에 대한 SSL 터널을 생성할 TLS수 있습니다.

Linux와의 암호화된 연결

valkey-cli를 사용하여 Amazon Linux 2023, Amazon Linux 2 또는 Amazon Linux에서 전송 중 암호화로 활성화된 Valkey 또는 Redis OSS 클러스터에 연결하려면 다음 단계를 따르세요.

1. valkey-cli 유틸리티를 다운로드하고 컴파일합니다. 이 유틸리티는 Valkey 소프트웨어 배포에 포함됩니다.
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 입력하여 클러스터 및 포트의 엔드포인트를 이 예제에 표시된 대로 바꿉니다.

   ```
   valkey-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://github.com/valkey-io/valkey/archive/refs/tags/7.2.6.tar.gz
   tar xvzf valkey-7.2.6.tar.gz
   cd valkey-7.2.6
   make distclean
   make valkey-cli BUILD_TLS=yes
   sudo install -m 755 src/valkey-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://github.com/valkey-io/valkey/archive/refs/tags/7.2.6.tar.gz
   ```

```
tar xvzf valkey-7.2.6.tar.gz
cd valkey-7.2.6
make valkey-cli CC=clang BUILD_TLS=yes
sudo install -m 755 src/valkey-cli /usr/local/bin/
```

Amazon Linux에서 다음 추가 단계를 실행해야 할 수도 있습니다.

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

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

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

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

stunnel을 사용한 암호화된 연결

valkey-cli를 사용하여 터널을 사용하여 전송 중 암호화로 활성화된 Redis OSS 클러스터에 연결하려면 다음 단계를 따르세요.

1. SSH 를 사용하여 클라이언트에 연결하고 를 설치합니다`stunnel`.

```
sudo yum install stunnel
```

2. 아래 제공된 출력을 템플릿으로 사용하여 다음 명령을 실행하여 파일을 `'/etc/stunnel/valkey-cli.conf'` 동시에 생성하고 편집하여 ElastiCache (Redis OSS) 클러스터 엔드포인트를 하나 이상의 연결 파라미터에 추가합니다.

```
vi /etc/stunnel/valkey-cli.conf
```

```
fips = no
setuid = root
setgid = root
pid = /var/run/stunnel.pid
debug = 7
delay = yes
options = NO_SSLv2
options = NO_SSLv3
[valkey-cli]
   client = yes
   accept = 127.0.0.1:6379
   connect = primary.ssltest.wif01h.use1.cache.amazonaws.com:6379
[valkey-cli-replica]
   client = yes
   accept = 127.0.0.1:6380
   connect = ssltest-02.ssltest.wif01h.use1.cache.amazonaws.com:6379
```

이 예에서는 구성 파일에 두 가지 연결, 즉 `valkey-cli`와 `valkey-cli-replica`가 있습니다. 파라미터는 다음과 같이 설정되어 있습니다.

- client는 이 stunnel 인스턴스를 클라이언트로 지정하기 위해 yes로 설정되어 있습니다.
- accept는 클라이언트 IP로 설정되어 있습니다. 이 예제에서 기본 는 포트 6379에서 Redis OSS 기본값 127.0.0.1로 설정됩니다. 복제본은 다른 포트를 호출해야 하며 6380으로 설정되어야 합니다. 휘발성 포트 1024~65535를 사용할 수 있습니다. 자세한 내용은 Amazon 사용 설명서의 Ephemeral 포트를 참조하세요. VPC
- 연결은 Redis OSS 서버 엔드포인트로 설정됩니다. 자세한 내용은 에서 연결 엔드포인트 찾기 ElastiCache 단원을 참조하십시오.

3. Start `stunnel`.

```
sudo stunnel /etc/stunnel/valkey-cli.conf
```

`netstat` 명령을 사용하여 터널이 시작되었는지 확인합니다.

```
sudo netstat -tulnp | grep -i stunnel

tcp        0      0 127.0.0.1:6379              0.0.0.0:*                   LISTEN
     3189/stunnel
```

```
tcp        0      0 127.0.0.1:6380              0.0.0.0:*                   LISTEN
     3189/stunnel
```

4. 터널의 로컬 엔드포인트를 사용하여 암호화된 Redis OSS 노드에 연결합니다.

   - ElastiCache (Redis OSS) 클러스터 생성 중에 AUTH 암호를 사용하지 않은 경우 이 예제에서는 valkey-cli를 사용하여 Amazon Linux에서 valkey-cli에 대한 전체 경로를 사용하여 ElastiCache (Redis OSS) 서버에 연결합니다.

     ```
     /home/ec2-user/redis-7.2.5/src/valkey-cli -h localhost -p 6379
     ```

     Redis OSS 클러스터 생성 중에 AUTH 암호를 사용한 경우 이 예제에서는 valkey-cli를 사용하여 Amazon Linux에서 valkey-cli에 대한 전체 경로를 사용하여 Redis OSS 서버에 연결합니다.

     ```
      /home/ec2-user/redis-7.2.5/src/valkey-cli -h localhost -p 6379 -a my-secret-
     password
     ```

   OR

   - 디렉터리를 redis-7.2.5로 변경하고 다음을 수행합니다.

     ElastiCache (Redis OSS) 클러스터 생성 중에 AUTH 암호를 사용하지 않은 경우 이 예제에서는 valkey-cli를 사용하여 Amazon Linux에서 valkey-cli에 대한 전체 경로를 사용하여 ElastiCache (Redis OSS) 서버에 연결합니다.

     ```
     src/valkey-cli -h localhost -p 6379
     ```

     Redis OSS 클러스터 생성 중에 AUTH 암호를 사용한 경우 이 예제에서는 valkey-cli를 사용하여 Amazon Linux에서 valkey-cli에 대한 전체 경로를 사용하여 Valkey 또는 Redis OSS 서버에 연결합니다.

     ```
     src/valkey-cli -h localhost -p 6379 -a my-secret-password
     ```

   이 예제에서는 Telnet을 사용하여 Valkey Redis OSS 서버에 연결합니다.

   ```
   telnet localhost 6379

   Trying 127.0.0.1...
   ```

```
Connected to localhost.
Escape character is '^]'.
auth MySecretPassword
+OK
get foo
$3
bar
```

5. SSL 터널을 중지하고 닫으려면 pkill 터널 프로세스가 진행됩니다.

```
sudo pkill stunnel
```

Python을 사용하여 자체 설계된 Redis OSS 클러스터에서 전송 중 암호화 활성화

다음 가이드에서는 원래 전송 중 암호화가 비활성화된 상태에서 생성된 Redis OSS 7.0 클러스터에서 전송 중 암호화를 활성화하는 방법을 보여줍니다. TCP 및 TLS 클라이언트는 이 프로세스 중에 가동 중지 없이 클러스터와 계속 통신합니다.

Boto3는 환경 변수에서 필요한 보안 인증 정보(aws_access_key_id, aws_secret_access_key 및 aws_session_token)를 가져옵니다. 이러한 보안 인증 정보는 이 가이드에 표시된 Python 코드를 처리하기 위해 python3를 실행할 동일한 bash 터미널에 미리 붙여넣어집니다. 아래 예제의 코드는 동일한 에서 시작된 EC2 인스턴스의 프로세스로, 해당 인스턴스에서 ElastiCache Redis OSS 클러스터를 생성하는 데 VPC 사용됩니다.

> **Note**
>
> - 다음 예제에서는 ElastiCache 관리 작업(클러스터 또는 사용자 생성)에 boto3를 사용하고 데이터 처리를 SDK 위해 redis-py/redis-py-cluster 를 사용합니다.
> - 클러스터 수정과 함께 온라인 TLS 마이그레이션을 사용하려면 boto3 버전(=~) 1.26.39 이상을 사용해야 합니다API.
> - ElastiCache 는 Valkey 버전 7.2 이상 또는 Redis OSS 버전 7.0 이상의 클러스터에 대해서만 온라인 TLS 마이그레이션을 지원합니다. 따라서 7.0 이전 OSS 버전의 Redis를 실행하는 클러스터가 있는 경우 클러스터의 Redis OSS 버전을 업그레이드해야 합니다. 버전 차이에 대한 자세한 내용은 Redis와의 주요 버전 동작 및 호환성 차이 OSS 섹션을 참조하세요.

주제

- ElastiCache Valkey 또는 Redis OSS 클러스터를 시작할 문자열 상수 정의

- 클러스터 구성을 위한 클래스를 정의합니다.
- 클러스터 자체를 나타내는 클래스를 정의합니다.
- (선택 사항) Valkey 또는 Redis OSS 클러스터에 대한 클라이언트 연결을 시연할 래퍼 클래스 생성
- 전송 중 데이터 암호화 구성 변경 프로세스를 시연하는 기본 함수 생성

ElastiCache Valkey 또는 Redis OSS 클러스터를 시작할 문자열 상수 정의

먼저 , 및 와 같이 ElastiCache 클러스터를 생성하는 데 필요한 AWS 엔터티의 이름을 포함하는 몇 가지 간단한 Python 문자열 상수`security-group`를`Cache Subnet group` 정의해 보겠습니다`default parameter group`. 이러한 모든 AWS 엔터티는 사용하려는 리전의 AWS 계정에 미리 생성해야 합니다.

```
#Constants definitions
SECURITY_GROUP = "sg-0492aa0a29c558427"
CLUSTER_DESCRIPTION = "This cluster has been launched as part of the online TLS
 migration user guide"
EC_SUBNET_GROUP = "client-testing"
DEFAULT_PARAMETER_GROUP_REDIS_7_CLUSTER_MODE_ENABLED = "default.redis7.cluster.on"
```

클러스터 구성을 위한 클래스를 정의합니다.

이제 클러스터의 구성을 나타내는 몇 가지 간단한 Python 클래스를 정의해 보겠습니다. 이 클래스는 Valkey 또는 Redis OSS 버전, 인스턴스 유형, 전송 중 암호화(TLS)의 활성화 또는 비활성화 여부와 같이 클러스터에 대한 메타데이터를 저장합니다.

```
#Class definitions

class Config:
    def __init__(
        self,
        instance_type: str = "cache.t4g.small",
        version: str = "7.0",
        multi_az: bool = True,
        TLS: bool = True,
        name: str = None,
    ):
        self.instance_type = instance_type
        self.version = version
        self.multi_az = multi_az
        self.TLS = TLS
```

```
        self.name = name or f"tls-test"

    def create_base_launch_request(self):
        return {
            "ReplicationGroupId": self.name,
            "TransitEncryptionEnabled": self.TLS,
            "MultiAZEnabled": self.multi_az,
            "CacheNodeType": self.instance_type,
            "Engine": "redis",
            "EngineVersion": self.version,
            "CacheSubnetGroupName": EC_SUBNET_GROUP ,
            "CacheParameterGroupName":
 DEFAULT_PARAMETER_GROUP_REDIS_7_CLUSTER_MODE_ENABLED ,
            "ReplicationGroupDescription": CLUSTER_DESCRIPTION,
            "SecurityGroupIds": [SECURITY_GROUP],
        }

class ConfigCME(Config):
    def __init__(
        self,
        instance_type: str = "cache.t4g.small",
        version: str = "7.0",
        multi_az: bool = True,
        TLS: bool = True,
        name: str = None,
        num_shards: int = 2,
        num_replicas_per_shard: int = 1,
    ):
        super().__init__(instance_type, version, multi_az, TLS, name)
        self.num_shards = num_shards
        self.num_replicas_per_shard = num_replicas_per_shard

    def create_launch_request(self) -> dict:
        launch_request = self.create_base_launch_request()
        launch_request["NumNodeGroups"] = self.num_shards
        launch_request["ReplicasPerNodeGroup"] = self.num_replicas_per_shard
        return launch_request
```

클러스터 자체를 나타내는 클래스를 정의합니다.

이제 ElastiCache Valkey 또는 Redis OSS 클러스터 자체를 나타내는 몇 가지 간단한 Python 클래스를 정의해 보겠습니다. 이 클래스에는 클러스터 생성 및 쿼리 ElastiCache와 같은 ElastiCache 관리 작업을 위해 boto3 클라이언트를 포함하는 클라이언트 필드가 있습니다API.

```
import botocore.config
import boto3

# Create boto3 client
def init_client(region: str = "us-east-1"):
    config = botocore.config.Config(retries={"max_attempts": 10, "mode": "standard"})
    init_request = dict()
    init_request["config"] = config
    init_request["service_name"] = "elasticache"
    init_request["region_name"] = region
    return boto3.client(**init_request)


class ElastiCacheClusterBase:
    def __init__(self, name: str):
        self.name = name
        self.elasticache_client = init_client()

    def get_first_replication_group(self):
        return self.elasticache_client.describe_replication_groups(
        ReplicationGroupId=self.name
        )["ReplicationGroups"][0]

    def get_status(self) -> str:
        return self.get_first_replication_group()["Status"]

    def get_transit_encryption_enabled(self) -> bool:
        return self.get_first_replication_group()["TransitEncryptionEnabled"]

    def is_available(self) -> bool:
        return self.get_status() == "available"

    def is_modifying(self) -> bool:
        return self.get_status() == "modifying"

    def wait_for_available(self):
        while True:
            if self.is_available():
                break
            else:
                time.sleep(5)

    def wait_for_modifying(self):
```

```
        while True:
            if self.is_modifying():
                break
            else:
                time.sleep(5)

    def delete_cluster(self) -> bool:
        self.elasticache_client.delete_replication_group(
            ReplicationGroupId=self.name, RetainPrimaryCluster=False
        )

    def modify_transit_encryption_mode(self, new_transit_encryption_mode: str):
        # generate api call to migrate the cluster to TLS preffered or to TLS required
            self.elasticache_client.modify_replication_group(
                ReplicationGroupId=self.name,
                TransitEncryptionMode=new_transit_encryption_mode,
                TransitEncryptionEnabled=True,
                ApplyImmediately=True,
            )
        self.wait_for_modifying()

 class ElastiCacheClusterCME(ElastiCacheClusterBase):
    def __init__(self, name: str):
        super().__init__(name)

    @classmethod
    def launch(cls, config: ConfigCME = None) -> ElastiCacheClusterCME:
        config = config or ConfigCME()
        print(config)
        new_cluster = ElastiCacheClusterCME(config.name)
        launch_request = config.create_launch_request()
        new_cluster.elasticache_client.create_replication_group(**launch_request)
        new_cluster.wait_for_available()
        return new_cluster

    def get_configuration_endpoint(self) -> str:
        return self.get_first_replication_group()["ConfigurationEndpoint"]["Address"]

#Since the code can throw exceptions, we define this class to make the code more
 readable and
#so we won't forget to delete the cluster
class ElastiCacheCMEManager:
    def __init__(self, config: ConfigCME = None):
        self.config = config or ConfigCME()
```

```
    def __enter__(self) -> ElastiCacheClusterCME:
        self.cluster = ElastiCacheClusterCME.launch(self.config)
        return self.cluster

    def __exit__(self, exc_type, exc_val, exc_tb):
        self.cluster.delete_cluster()
```

(선택 사항) Valkey 또는 Redis OSS 클러스터에 대한 클라이언트 연결을 시연할 래퍼 클래스 생성

이제 `redis-py-cluster` 클라이언트를 위한 래퍼 클래스를 생성해 보겠습니다. 이 래퍼 클래스는 클러스터를 일부 키로 미리 채운 다음 무작위로 반복되는 `get` 명령을 수행하는 것을 지원합니다.

> **Note**
>
> 이 단계는 선택 사항이지만 이후 단계에 나오는 기본 함수의 코드를 단순화합니다.

```
import redis
improt random
from time import perf_counter_ns, time


class DowntimeTestClient:
    def __init__(self, client):
        self.client = client

        # num of keys prefilled
        self.prefilled = 0
        # percent of get above prefilled
        self.percent_get_above_prefilled = 10 # nil result expected when get hit above
 prefilled
        # total downtime in nano seconds
        self.downtime_ns = 0
        # num of success and fail operations
        self.success_ops = 0
        self.fail_ops = 0
        self.connection_errors = 0
        self.timeout_errors = 0


    def replace_client(self, client):
```

```
        self.client = client

    def prefill_data(self, timelimit_sec=60):
        end_time = time() + timelimit_sec
        while time() < end_time:
            self.client.set(self.prefilled, self.prefilled)
            self.prefilled += 1

    # unsuccesful operations throw exceptions
    def _exec(self, func):
        try:
            start_ns = perf_counter_ns()
            func()
            self.success_ops += 1
            elapsed_ms = (perf_counter_ns() - start_ns) // 10 ** 6
            # upon succesful execution of func
            # reset random_key to None so that the next command
            # will use a new random key
            self.random_key = None

        except Exception as e:
            elapsed_ns = perf_counter_ns() - start_ns
            self.downtime_ns += elapsed_ns
            # in case of failure- increment the relevant counters so that we will keep
track
            # of how many connection issues we had while trying to communicate with
            # the cluster.
            self.fail_ops += 1
            if e.__class__ is redis.exceptions.ConnectionError:
                self.connection_errors += 1
            if e.__class__ is redis.exceptions.TimeoutError:
                self.timeout_errors += 1

    def _repeat_exec(self, func, seconds):
        end_time = time() + seconds
        while time() < end_time:
            self._exec(func)

    def _new_random_key_if_needed(self, percent_above_prefilled):
        if self.random_key is None:
            max = int((self.prefilled * (100 + percent_above_prefilled)) / 100)
            return random.randint(0, max)
        return self.random_key
```

```
    def _random_get(self):
        key = self._new_random_key_if_needed(self.percent_get_above_prefilled)
        result = self.client.get(key)
        # we know the key was set for sure only in the case key < self.prefilled
        if key < self.prefilled:
            assert result.decode("UTF-8") == str(key)


    def repeat_get(self, seconds=60):
        self._repeat_exec(self._random_get, seconds)

    def get_downtime_ms(self) -> int:
        return self.downtime_ns // 10 ** 6


    def do_get_until(self, cond_check):
        while not cond_check():
            self.repeat_get()
        # do one more get cycle once condition is met
        self.repeat_get()
```

전송 중 데이터 암호화 구성 변경 프로세스를 시연하는 기본 함수 생성

이제 다음을 수행하는 기본 함수를 정의해 보겠습니다.

1. boto3 ElastiCache client를 사용하여 클러스터를 생성합니다.
2. 를 사용하지 않고 명확한 연결로 클러스터에 TCP 연결할 `redis-py-cluster` 클라이언트를 초기화합니다TLS.
3. `redis-py-cluster` 클라이언트는 클러스터를 일부 데이터로 미리 채웁니다.
4. boto3 클라이언트는 no에서TLS TLS preferred로 TLS 마이그레이션을 트리거합니다.
5. 클러스터가 TLS 로 마이그레이션되는 동안 `Preferred` `redis-py-cluster` TCP 클라이언트는 마이그레이션이 완료될 때까지 클러스터에 반복 `get` 작업을 보냅니다.
6. 로 마이그레이션TLS`Preferred`이 완료되면 클러스터가 전송 중 암호화를 지원한다고 어설션합니다. 그런 다음 를 사용하여 클러스터에 연결할 `redis-py-cluster` 클라이언트를 생성합니다TLS.
7. 새 TLS 클라이언트와 이전 TCP 클라이언트를 사용하여 몇 가지 `get` 명령을 보냅니다.
8. boto3 클라이언트는 에서 TLS 필수로의 TLS 마이그레이션TLS`Preferred`을 트리거합니다.
9. 클러스터가 TLS 필수 redis-py-cluster로 마이그레이션되는 동안 TLS 클라이언트는 마이그레이션이 완료될 때까지 클러스터에 반복 `get` 작업을 보냅니다.

```
import redis

def init_cluster_client(
    cluster: ElastiCacheClusterCME, prefill_data: bool, TLS: bool = True) ->
 DowntimeTestClient:
    # we must use for the host name the cluster configuration endpoint.
    redis_client = redis.RedisCluster(
        host=cluster.get_configuration_endpoint(), ssl=TLS, socket_timeout=0.25,
 socket_connect_timeout=0.1
    )
    test_client = DowntimeTestClient(redis_client)
    if prefill_data:
        test_client.prefill_data()
    return test_client

if __name__ == '__main__':
    config = ConfigCME(TLS=False, instance_type="cache.m5.large")

    with ElastiCacheCMEManager(config) as cluster:
        # create a client that will connect to the cluster with clear tcp connection
        test_client_tcp = init_cluster_client(cluster, prefill_data=True, TLS=False)

       # migrate the cluster to TLS Preferred
        cluster.modify_transit_encryption_mode(new_transit_encryption_mode="preferred")

        # do repeated get commands until the cluster finishes the migration to TLS
 Preferred
        test_client_tcp.do_get_until(cluster.is_available)

       # verify that in transit encryption is enabled so that clients will be able to
 connect to the cluster with TLS
        assert cluster.get_transit_encryption_enabled() == True

       # create a client that will connect to the cluster with TLS connection.
        # we must first make sure that the cluster indeed supports TLS
        test_client_tls = init_cluster_client(cluster, prefill_data=True, TLS=True)

        # by doing get commands with the tcp client for 60 more seconds
       # we can verify that the existing tcp connection to the cluster still works
        test_client_tcp.repeat_get(seconds=60)

        # do get commands with the new TLS client for 60 more seconds
        test_client_tcp.repeat_get(seconds=60)
```

```
      # migrate the cluster to TLS required
       cluster.modify_transit_encryption_mode(new_transit_encryption_mode="required")

      # from this point the tcp clients will be disconnected and we must not use them
anymore.
      # do get commands with the TLS client until the cluster finishes migartion to
TLS required mode.
       test_client_tls.do_get_until(cluster.is_available)
```

전송 중 암호화 활성화 시 모범 사례

전송 중 암호화를 활성화하기 전에: 적절한 DNS 레코드 처리가 있는지 확인합니다.

> **Note**
>
> 이 프로세스에서 기존 엔드포인트를 변경 및 삭제합니다. 엔드포인트를 잘못 사용하면 Valkey 또는 Redis OSS 클라이언트가 이전 엔드포인트와 삭제된 엔드포인트를 사용하여 클러스터에 연결하지 못할 수 있습니다.

클러스터를 no-TLS에서 TLS-preferred로 마이그레이션하는 동안 이전 노드별 DNS 레코드가 유지되고 새 노드별 DNS 레코드가 다른 형식으로 생성됩니다. TLS-활성화된 클러스터는 클러스터와 non-TLS-enabled 다른 형식의 DNS 레코드를 사용합니다. 는 클러스터가 암호화 모드로 구성된 경우 두 DNS 레코드를 모두 ElastiCache 유지합니다. 애플리케이션과 다른 Valkey 또는 Redis OSS 클라이언트가 서로 전환할 수 있도록 선호됩니다. TLS마이그레이션 프로세스 중에 DNS 레코드가 다음과 같이 변경됩니다.

전송 중 암호화를 활성화할 때 발생하는 DNS 레코드의 변경 사항에 대한 설명

CME 클러스터의 경우

클러스터가 '전송 암호화 모드: 선호'로 설정된 경우:

- 활성화TLS되지 않은 클러스터의 원래 클러스터 엔드포인트는 활성 상태로 유지됩니다. 클러스터가 양식 TLS 암호화 모드 '없음'을 '선호됨'으로 재구성되면 가동 중지가 발생하지 않습니다.
- 클러스터가 TLS-기본 모드로 설정되면 새 TLS Valkey 또는 Redis OSS 엔드포인트가 생성됩니다. 이러한 새 엔드포인트는 이전 엔드포인트IPs와 동일하게 확인됩니다(비-TLS).
- 새 TLS Valkey 또는 Redis OSS 구성 엔드포인트는 ElastiCache 콘솔과 `describe-replication-group` 에 대한 응답에 노출됩니다API.

클러스터가 '전송 암호화 모드: 필수'로 설정된 경우:

- 활성화되지 않은TLS 이전 엔드포인트는 삭제됩니다. TLS 클러스터 엔드포인트의 가동 중지 시간은 없습니다.
- ElastiCache 콘솔 또는 `cluster-configuration-endpoint` 에서 새 `describe-replication-group` 를 검색할 수 있습니다API.

자동 장애 조치가 활성화되거나 자동 장애 조치가 비활성화된 CMD 클러스터의 경우

복제 그룹이 '전송 암호화 모드: 선호'로 설정된 경우:

- 활성화TLS되지 않은 클러스터의 원래 기본 엔드포인트 및 리더 엔드포인트는 활성 상태로 유지됩니다.
- 클러스터가 TLS `Preferred` 모드로 설정되면 새 TLS 프라이머리 및 리더 엔드포인트가 생성됩니다. 이 새 엔드포인트는 이전 엔드포인트(비)와 동일한 IP(들)로 확인됩니다TLS.
- 새 기본 엔드포인트 및 리더 엔드포인트는 ElastiCache 콘솔과 `describe-replication-group` 에 대한 응답으로 노출됩니다API.

복제 그룹이 '전송 암호화 모드: 필수'로 설정된 경우:

- 이전 비TLS 프라이머리 및 리더 엔드포인트가 삭제됩니다. TLS 클러스터 엔드포인트의 가동 중지 시간은 없습니다.
- 콘솔 또는 `describe-replication-group` 에서 ElastiCache 새 기본 및 리더 엔드포인트를 검색할 수 있습니다API.

DNS 레코드의 권장 사용량

CME 클러스터의 경우

- 애플리케이션 코드의 노드별 DNS 레코드 대신 클러스터 구성 엔드포인트를 사용합니다. 노드별 DNS 이름은 샤드를 추가하거나 제거할 때 변경될 수 있으므로 직접 사용하는 것은 권장되지 않습니다.
- 클러스터 구성 엔드포인트는 이 프로세스 중에 변경되므로 애플리케이션에서 클러스터 구성 엔드포인트를 하드코딩하지 마세요.
- 클러스터 구성 엔드포인트는 이 프로세스 중에 변경될 수 있으므로 애플리케이션에서 클러스터 구성 엔드포인트를 하드코딩하는 것은 좋지 않습니다. 전송 중 암호화가 완료되면 `describe-`

replication-group API (위에 표시된 대로(굵은 글씨체)) 로 클러스터 구성 엔드포인트를 쿼리하고 이 시점부터 응답으로 DNS 얻은 를 사용합니다.

자동 장애 조치가 활성화된 CMD 클러스터의 경우

- 이전 노드별 DNS 이름이 삭제되고 클러스터를 no-TLSto-TLSpreferred로 마이그레이션할 때 새 이름이 생성되므로 애플리케이션 코드의 노드별 DNS 이름 대신 기본 엔드포인트와 리더 엔드포인트를 사용합니다. 나중에 클러스터에 복제본을 추가할 수 있으므로 노드별 DNS 이름을 직접 사용하는 것은 권장되지 않습니다. 또한 자동 장애 조치가 활성화되면 기본 클러스터 및 복제본의 역할이 ElastiCache 서비스에 의해 자동으로 변경되며, 이러한 변경 사항을 추적하는 데 도움이 되도록 기본 엔드포인트 및 리더 엔드포인트를 사용하는 것이 좋습니다. 마지막으로 리더 엔드포인트를 사용하면 복제본의 읽기를 클러스터의 복제본 간에 균등하게 분산하는 데 도움이 됩니다.
- 애플리케이션에 프라이머리 엔드포인트와 리더 엔드포인트를 하드코딩하는 것은 TLS 마이그레이션 프로세스 중에 변경할 수 있기 때문에 잘못된 관행입니다. TLS-preferred로 마이그레이션 변경이 완료되면 를 사용하여 describe-replication-group 기본 엔드포인트 및 리더 엔드포인트를 쿼리API하고 이 시점부터 응답으로 DNS 얻은 를 사용합니다. 이렇게 하면 엔드포인트의 변경 사항을 동적으로 추적할 수 있습니다.

자동 장애 조치가 비활성화된 CMD 클러스터의 경우

- 애플리케이션 코드의 노드별 DNS 이름 대신 기본 엔드포인트 및 리더 엔드포인트를 사용합니다. 자동 장애 조치가 비활성화되면 자동 장애 조치가 활성화될 때 ElastiCache 서비스에 의해 자동으로 관리되는 크기 조정, 패치 적용, 장애 조치 및 기타 절차가 대신 수행됩니다. 이렇게 하면 여러 엔드포인트를 쉽게 수동으로 추적할 수 있습니다. 이전 노드별 DNS 이름이 삭제되고 클러스터를 no-TLStoTLS-preferred로 마이그레이션할 때 새 이름이 생성되므로 노드별 DNS 이름을 직접 사용하지 마세요. 이는 클라이언트가 TLS마이그레이션 중에 클러스터에 연결할 수 있도록 필수 사항입니다. 또한 리더 엔드포인트를 사용할 때 복제본 간에 읽기를 균등하게 분산하고 클러스터에서 복제본을 추가하거나 삭제할 때 DNS-레코드를 추적할 수 있습니다.
- 클러스터 구성 엔드포인트를 애플리케이션에 하드코딩하는 것은 TLS 마이그레이션 프로세스 중에 변경할 수 있으므로 잘못된 관행입니다.

전송 중 데이터 암호화 중: 마이그레이션 프로세스가 언제 완료되는지에 주의 기울이기

전송 중 데이터 암호화 모드 변경은 즉시 이루어지지 않으며 시간이 좀 걸릴 수 있습니다. 대규모 클러스터의 경우 특히 그렇습니다. 클러스터가 로 마이그레이션을 완료하는 경우에만 및 TCP TLS 연결을

모두 수락하고 제공할 수 TLS있습니다. 따라서 전송 중 암호화가 완료될 때까지 클러스터에 대한 TLS 연결을 설정하려고 시도하는 클라이언트를 생성해서는 안 됩니다.

전송 중 데이터 암호화가 성공적으로 완료되거나 실패했을 때 알림을 받는 방법에는 다음과 같은 여러 가지가 있습니다(위 코드 예에는 표시되지 않음).

- 암호화가 완료되면 SNS 서비스를 사용하여 알림 받기
- 암호화가 완료될 때 이벤트를 내보내`describe-events`API는 사용
- ElastiCache 콘솔에서 암호화가 완료되었다는 메시지 확인

또한 애플리케이션에 로직을 구현하여 암호화가 완료되었는지 확인할 수 있습니다. 위의 예에서는 클러스터가 마이그레이션을 완료하도록 하는 몇 가지 방법을 살펴보았습니다.

- 마이그레이션 프로세스가 시작될 때까지 대기(클러스터 상태가 '수정 중'으로 변경됨) 및 수정이 완료될 때까지 대기(클러스터 상태가 다시 '사용 가능'으로 변경됨)
- 를 쿼리하여 클러스터가 True로 `transit_encryption_enabled` 설정되었다고 주장합니다`describe-replication-group`API.

전송 중 데이터 암호화를 활성화한 후: 사용하는 클라이언트가 올바르게 구성되었는지 확인

클러스터가 TLS-선호 모드에 있는 동안 애플리케이션은 클러스터에 대한 TLS 연결을 열고 이러한 연결만 사용해야 합니다. 이렇게 하면 전송 중 데이터 암호화를 활성화할 때 애플리케이션이 가동 중지를 겪지 않습니다. SSL 섹션의 정보 명령을 사용하여 Valkey 또는 Redis OSS 엔진에 더 명확한 TCP 연결이 없는지 확인할 수 있습니다.

```
# SSL
ssl_enabled:yes
ssl_current_certificate_not_before_date:Mar 20 23:27:07 2017 GMT
ssl_current_certificate_not_after_date:Feb 24 23:27:07 2117 GMT
ssl_current_certificate_serial:D8C7DEA91E684163
tls_mode_connected_tcp_clients:0   (should be zero)
tls_mode_connected_tls_clients:100
```

Openssl(Memcached)을 사용하여 전송 중 암호화로 활성화된 노드에 연결

전송 중 암호화로 활성화된 ElastiCache (Memcached) 노드의 데이터에 액세스하려면 Secure Socket Layer()로 작업하는 클라이언트를 사용해야 합니다SSL. 또한 Amazon Linux 및 Amazon Linux 2에서 Openssl s_client를 사용할 수 있습니다.

Openssl s_client를 사용하여 Amazon Linux 2 또는 Amazon Linux에서 전송 중 데이터 암호화가 사용 설정된 Memcached 클러스터에 연결하려면 다음을 수행합니다.

```
/usr/bin/openssl s_client -connect memcached-node-endpoint:memcached-port
```

Java를 사용하여 TLS Memcached 클라이언트 생성

TLS 모드에서 클라이언트를 생성하려면 다음을 수행하여 적절한 로 클라이언트를 초기화합니다 SSLContext.

```
import java.security.KeyStore;
import javax.net.ssl.SSLContext;
import javax.net.ssl.TrustManagerFactory;
import net.spy.memcached.AddrUtil;
import net.spy.memcached.ConnectionFactoryBuilder;
import net.spy.memcached.MemcachedClient;
public class TLSDemo {
    public static void main(String[] args) throws Exception {
        ConnectionFactoryBuilder connectionFactoryBuilder = new
 ConnectionFactoryBuilder();
        // Build SSLContext
        TrustManagerFactory tmf =
 TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
        tmf.init((KeyStore) null);
        SSLContext sslContext = SSLContext.getInstance("TLS");
        sslContext.init(null, tmf.getTrustManagers(), null);
        // Create the client in TLS mode
        connectionFactoryBuilder.setSSLContext(sslContext);
        MemcachedClient client = new MemcachedClient(connectionFactoryBuilder.build(),
 AddrUtil.getAddresses("mycluster.fnjyzo.cfg.use1.cache.amazonaws.com:11211"));

        // Store a data item for an hour.
        client.set("theKey", 3600, "This is the data value");
    }
}
```

를 사용하여 TLS Memcached 클라이언트 생성 PHP

TLS 모드에서 클라이언트를 생성하려면 다음을 수행하여 적절한 로 클라이언트를 초기화합니다 SSLContext.

```
<?php
```

```
/**
 * Sample PHP code to show how to create a TLS Memcached client. In this example we
 * will use the Amazon ElastiCache Auto Descovery feature, but TLS can also be
 * used with a Static mode client.
 * See Using the ElastiCache Cluster Client for PHP (https://docs.aws.amazon.com/
AmazonElastiCache/latest/dg/AutoDiscovery.Using.ModifyApp.PHP.html) for more
 information
 * about Auto Discovery and persistent-id.
 */

/* Configuration endpoint to use to initialize memcached client.
 * this is only an example */
$server_endpoint = "mycluster.fnjyzo.cfg.use1.cache.amazonaws.com";

/* Port for connecting to the cluster.
 * This is only an example     */
$server_port = 11211;

/* Initialize a persistent Memcached client and configure it with the Dynamic client
 mode  */
$tls_client =  new Memcached('persistent-id');
$tls_client->setOption(Memcached::OPT_CLIENT_MODE, Memcached::DYNAMIC_CLIENT_MODE);

/* Add the memcached's cluster server/s */
$tls_client->addServer($server_endpoint, $server_port);

/* Configure the client to use TLS */
if(!$tls_client->setOption(Memcached::OPT_USE_TLS, 1)) {
    echo $tls_client->getLastErrorMessage(), "\n";
    exit(1);
}

/* Set your TLS context configurations values.
 * See MemcachedTLSContextConfig in memcached-api.php for all configurations */
$tls_config = new MemcachedTLSContextConfig();
$tls_config->hostname = '*.mycluster.fnjyzo.use1.cache.amazonaws.com';
$tls_config->skip_cert_verify = false;
$tls_config->skip_hostname_verify = false;

/* Use the created TLS context configuration object to create OpenSSL's SSL_CTX and set
 it to your client.
 * Note:  These TLS context configurations will be applied to all the servers connected
 to this client. */
```

```
$tls_client->createAndSetTLSContext((array)$tls_config);

/* test the TLS connection with set-get scenario: */

 /* store the data for 60 seconds in the cluster.
 * The client will decide which cache host will store this item.
 */
if($tls_client->set('key', 'value', 60)) {
    print "Successfully stored key\n";
} else {
    echo "Failed to set key: ", $tls_client->getLastErrorMessage(), "\n";
    exit(1);
}

/* retrieve the key */
if ($tls_client->get('key') === 'value') {
    print "Successfully retrieved key\n";
} else {
    echo "Failed to get key: ", $tls_client->getLastErrorMessage(), "\n";
    exit(1);
}
```

PHP 클라이언트 사용에 대한 자세한 내용은 섹션을 참조하세요PHP용 ElastiCache 클러스터 클라이언트 설치.

의 저장 시 암호화 ElastiCache

데이터를 안전하게 유지하기 위해 Amazon ElastiCache 과 Amazon S3는 캐시의 데이터에 대한 액세스를 제한하는 다양한 방법을 제공합니다. 자세한 내용은 Amazon VPCs 및 ElastiCache 보안 및 Amazon용 자격 증명 및 액세스 관리 ElastiCache 단원을 참조하세요.

ElastiCache 저장 시 암호화는 온디스크 데이터를 암호화하여 데이터 보안을 강화하는 기능입니다. 이 기능은 서버리스 캐시에서 항상 활성화되어 있습니다. 이 기능이 활성화되어 있으면 다음과 같은 항목이 암호화됩니다.

- 동기화, 백업 및 스왑 작업 중 디스크
- Amazon S3에 저장된 백업

데이터 계층화가 활성화된 클러스터의 SSDs (솔리드 스테이트 드라이브)에 저장된 데이터는 항상 암호화됩니다.

ElastiCache 는 저장 시 기본(서비스 관리형) 암호화와 AWS Key Management Service(KMS)에서 자체 대칭 고객 관리 AWS KMS형 키를 사용할 수 있는 기능을 제공합니다. 캐시가 백업되면 암호화 옵션에서 기본 암호화 키를 사용할지 또는 고객 관리 키를 사용할지 선택합니다. 자세한 내용은 저장 데이터 암호화 활성화 단원을 참조하십시오.

> **Note**
>
> 기본(서비스 관리형) 암호화는 GovCloud (미국) 리전에서 사용할 수 있는 유일한 옵션입니다.

> **Important**
>
> 기존 자체 설계된 Valkey 또는 Redis OSS 클러스터에서 at-Rest Encryption을 활성화하려면 복제 그룹에서 백업 및 복원을 실행한 후 기존 복제 그룹을 삭제해야 합니다.

저장 시 암호화는 캐시를 생성할 때만 캐시에 대해 활성화할 수 있습니다. 데이터를 암호화 및 해독하기 위해 몇 가지 처리가 필요하기 때문에 미사용 데이터 암호화를 활성화하면 이러한 작업 중 성능에 영향이 있을 수 있습니다. 사용 사례에 대한 성능 영향을 파악하기 위해서는 미사용 데이터 암호화를 사용한 상태와 사용하지 않은 상태에서 데이터를 벤치마크해야 합니다.

주제

미사용 데이터 암호화 조건

저장 ElastiCache 시 암호화 구현을 계획할 때는 저장 시 ElastiCache 암호화에 대한 다음 제약 조건을 염두에 두어야 합니다.

- 저장 시 암호화는 Valkey 7.2 이상 및 Redis OSS 버전(에 예약된 3.2.6EOL, Redis OSS 버전 수명 종료 일정 참조), 4.0.10 이상을 실행하는 복제 그룹에서 지원됩니다.
- 저장 시 암호화는 Amazon 에서 실행되는 복제 그룹에 대해서만 지원됩니다VPC.
- 유휴 데이터 암호화는 다음 노드 유형을 실행하는 복제 그룹에 대해서만 지원됩니다.
  - R6gd, R6g, R5, R4, R3
  - M6g, M5, M4, M3
  - T4g,T3, T2

  자세한 내용은 지원되는 노드 유형 섹션 참조
- 미사용 데이터 암호화는 `AtRestEncryptionEnabled` 파라미터를 명시적으로 `true`로 설정해 활성화합니다.
- 복제 그룹을 생성하는 경우에만 복제 그룹에 대해 미사용 데이터 암호화를 활성화할 수 있습니다. 복제 그룹을 수정하는 방법으로는 미사용 데이터 암호화 켜기 또는 끄기로 전환할 수 없습니다. 기존 복제 그룹에 대해 미사용 데이터 암호화를 구현하는 방법에 대한 자세한 내용은 저장 데이터 암호화 활성화 섹션을 참조하세요.
- 클러스터가 r6gd 패밀리의 노드 유형을 사용하는 경우 저장 시 암호화 활성화 여부와 관계없이 에 저장된 데이터가 암호화SSD됩니다.
- 저장 시 암호화에 고객 관리형 키를 사용하는 옵션은 AWS GovCloud (us-gov-east-1 및 us-gov-west-1) 리전에서 사용할 수 없습니다.
- 클러스터가 r6gd 패밀리의 노드 유형을 사용하는 경우 에 저장된 데이터는 선택한 고객 관리 AWS KMS형 키(또는 AWS GovCloud 리전의 서비스 관리형 암호화)로 암호화SSD됩니다.

- Memcached를 사용하면 서버리스 캐시에서만 저장 시 암호화가 지원됩니다.

- Memcached를 사용하는 경우 GovCloud (us-gov-east-1 및 -1 us-gov-west) 리전에서 AWS 유휴 암호화에 고객 관리형 키를 사용하는 옵션을 사용할 수 없습니다.

미사용 데이터 암호화를 구현하면 백업 및 노드 동기화 작업 중 성능이 저하될 수 있습니다. 이러한 암호화가 구현 성능에 미치는 영향을 확인하려면 미사용 데이터 암호화와 데이터를 암호화하지 않은 경우를 비교해 벤치마크하세요.

에서 고객 관리형 키 사용 AWS KMS

ElastiCache 는 저장 시 암호화를 위한 대칭 고객 관리 AWS KMS형 키(KMS 키)를 지원합니다. 고객 관리형 KMS 키는 AWS 계정에서 생성, 소유 및 관리하는 암호화 키입니다. 자세한 내용은 AWS KMS Key AWS Management Service 개발자 안내서의 키를 참조하세요. 키를 에서 AWS KMS 생성해야 에서 사용할 수 있습니다 ElastiCache.

루트 키를 생성하는 AWS KMS 방법을 알아보려면 Key AWS Management Service 개발자 안내서의 키 생성을 참조하세요.

ElastiCache 를 사용하면 와 통합할 수 있습니다 AWS KMS. 자세한 내용은 AWS Key Management Service 개발자 안내서의 권한 부여 사용을 참조하세요. Amazon과 의 ElastiCache 통합을 활성화하기 위해 고객 조치가 필요하지 않습니다 AWS KMS.

`kms:ViaService` 조건 키는 키(KMS 키) 사용을 AWS KMS 지정된 AWS 서비스의 요청으로 제한합니다. 와 `kms:ViaService` 함께 사용하려면 조건 키 값인 `elasticache.AWS_region.amazonaws.com` 및 에 두 ViaService 이름을 모두 ElastiCache포함합니다`dax.AWS_region.amazonaws.com`. 자세한 내용은 kms:ViaService를 참조하세요.

AWS CloudTrail 를 사용하여 Amazon이 사용자를 대신하여 ElastiCache 보내는 AWS Key Management Service 요청을 추적할 수 있습니다. 고객 관리형 키와 AWS Key Management Service 관련된 에 대한 모든 API 호출에는 해당 CloudTrail 로그가 있습니다. ListGrants KMS API 또한 호출을 호출하여 가 ElastiCache 생성하는 권한 부여를 볼 수 있습니다.

고객 관리형 키를 사용하여 복제 그룹을 암호화하면 복제 그룹에 대한 모든 백업이 다음과 같이 암호화됩니다.

- 자동 일일 백업은 클러스터와 연결된 고객 관리형 키를 사용하여 암호화됩니다.
- 복제 그룹을 삭제할 때 생성된 최종 백업은 복제 그룹에 연결된 고객 관리형 키를 사용하여 암호화됩니다.
- 수동으로 생성된 백업은 복제 그룹과 연결된 KMS 키를 사용하도록 기본적으로 암호화됩니다. 다른 고객 관리형 키를 선택하여 이를 재정의할 수 있습니다.

- 백업 복사는 기본적으로 소스 백업과 연결된 고객 관리형 키를 사용합니다. 다른 고객 관리형 키를 선택하여 이를 재정의할 수 있습니다.

> **Note**
>
> - 선택한 Amazon S3 버킷으로 백업을 내보낼 때 고객 관리형 키를 사용할 수 없습니다. 그러나 Amazon S3으로 내보낸 모든 백업은 서버 측 암호화를 사용하여 암호화됩니다. 백업 파일을 새 S3 객체에 복사하고 고객 관리형 KMS 키를 사용하여 암호화하거나, 키를 사용하여 기본 암호화로 설정된 다른 S3 버킷에 파일을 복사KMS하거나, 파일 자체에서 암호화 옵션을 변경할 수 있습니다.
> - 또한 고객 관리형 키를 사용하여 암호화에 고객 관리형 키를 사용하지 않는 복제 그룹에 대해 수동으로 생성한 백업을 암호화할 수도 있습니다. 이 옵션을 사용하면 원본 복제 그룹에서 데이터가 암호화되지 않더라도 Amazon S3에 저장된 백업 파일이 KMS 키를 사용하여 암호화됩니다.
>
> 백업에서 복원하면 새 복제 그룹을 생성할 때 사용할 수 있는 암호화 옵션과 유사한 암호화 옵션을 선택할 수 있습니다.

- 캐시를 암호화하는 데 사용한 키에 대해 키를 삭제하거나, 키를 비활성화하거나, 권한 부여를 취소하면 캐시를 복구할 수 없게 됩니다. 즉, 하드웨어 장애 후에는 수정하거나 복구할 수 없습니다. AWS KMS 는 최소 7일의 대기 기간이 지난 후에만 루트 키를 삭제합니다. 키를 삭제한 후 다른 고객 관리형 키를 사용하여 보관용 백업을 생성할 수 있습니다.
- 자동 키 교체는 루트 키의 AWS KMS 속성을 보존하므로 데이터 액세스 기능에 영향을 주지 않습니다 ElastiCache . 암호화된 Amazon ElastiCache 캐시는 새 루트 키를 생성하고 이전 키에 대한 참조를 업데이트하는 수동 키 교체를 지원하지 않습니다. 자세한 내용은 Key Management Service 개발자 안내서의 키 교체 AWS KMS를 참조하세요. AWS
- KMS 키를 사용하여 ElastiCache 캐시를 암호화하려면 캐시당 하나의 권한이 필요합니다. 이 권한은 캐시의 수명 기간 동안 사용됩니다. 또한 백업 생성 중에는 백업당 하나의 권한이 사용됩니다. 이 권한은 백업이 생성되면 폐기됩니다.
- 권한 부여 및 제한에 대한 AWS KMS 자세한 내용은 AWS 키 관리 서비스 개발자 안내서의 제한을 참조하세요.

저장 데이터 암호화 활성화

모든 서버리스 캐시에는 저장 시 암호화가 활성화되어 있습니다.

자체 설계된 클러스터를 생성할 때 `AtRestEncryptionEnabled` 파라미터를 `true`로 설정하여 저장 시 암호화를 활성화할 수 있습니다. 기존 복제 그룹에 대해서는 미사용 데이터 암호화를 설정할 수 없습니다.

ElastiCache 캐시를 생성할 때 저장 시 암호화를 활성화할 수 있습니다. AWS Management Console, AWS CLI또는 를 사용하여 할 수 있습니다 ElastiCache API.

캐시를 생성할 때 다음 옵션 중 하나를 선택할 수 있습니다.

- 기본값 – 이 옵션은 저장된 서비스 관리 암호화를 사용합니다.
- 고객 관리형 키 - 이 옵션을 사용하면 저장 시 암호화를 위해 의 키 ID/ARN AWS KMS를 제공할 수 있습니다.

루트 키를 생성하는 AWS KMS 방법을 알아보려면 Key AWS Management Service 개발자 안내서의 키 생성을 참조하세요.

목차

기존 자체 설계된 Valkey 또는 Redis OSS 클러스터에서 At-Rest 암호화 활성화

저장 시 암호화는 Valkey 또는 Redis OSS 복제 그룹을 생성할 때만 활성화할 수 있습니다. 미사용 데이터 암호화를 활성화하려는 기존 복제 그룹이 있으면 다음 작업을 수행하세요.

기존 복제 그룹에 대해 미사용 데이터 암호화를 활성화하려면

1. 기존 복제 그룹의 백업을 수동으로 생성합니다. 자세한 내용은 수동 백업 지원을 참조하세요.
2. 백업에서 복원하여 새 복제 그룹을 생성합니다. 새 복제 그룹에 대해 미사용 데이터 암호화를 활성화합니다. 자세한 내용은 백업에서 새 캐시로 복원을 참조하세요.
3. 새 복제 그룹을 가리키도록 애플리케이션에서 엔드포인트를 업데이트합니다.
4. 이전 복제 그룹을 삭제합니다. 자세한 내용은 에서 클러스터 삭제 ElastiCache 또는 복제 그룹 삭제을 참조하세요.

를 사용하여 휴지 시 암호화 활성화 AWS Management Console

서버리스 캐시에서 저장 시 암호화 활성화(콘솔)

모든 서버리스 캐시에는 저장 시 암호화가 활성화되어 있습니다. 기본적으로 AWS소유 KMS 키는 데이터를 암호화하는 데 사용됩니다. 자체 AWS KMS 키를 선택하려면 다음 항목을 선택합니다.

- 기본 설정 섹션을 펼칩합니다.
- 기본 설정 섹션에서 기본 설정 사용자 지정을 선택합니다.
- 보안 섹션에서 보안 설정 사용자 지정을 선택합니다.
- 암호화 키 설정에서 고객 관리를 CMK 선택합니다.
- AWS KMS 키 설정에서 키를 선택합니다.

자체 설계된 클러스터에 대해 저장 시 암호화 활성화(콘솔)

자체 캐시를 설계할 때 '간편한 생성' 메서드를 사용하는 '개발 및 테스트' 및 '프로덕션' 구성에서는 기본 키를 사용하여 저장 시 암호화가 활성화됩니다. 구성을 직접 선택할 때 다음과 같이 진행합니다.

- 엔진 버전으로 버전 3.2.6, 4.0.10 또는 이후 버전을 선택합니다.
- 저장 시 암호화의 활성화 옵션 옆에서 확인란을 클릭합니다.
- 기본 키 또는 고객 관리형 CMK를 선택합니다.

step-by-step 절차는 다음을 참조하세요.

- Valkey(클러스터 모드 비활성화됨) 클러스터 생성(콘솔)
- Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 생성(콘솔)

를 사용하여 휴지 시 암호화 활성화 AWS CLI

를 사용하여 Valkey 또는 Redis OSS 클러스터를 생성할 때 유휴 시 암호화를 활성화하려면 복제 그룹을 생성할 때 --at-rest-encryption-enabled 파라미터를 AWS CLI사용합니다.

Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터에서 At-Rest 암호화 활성화(CLI)

다음 작업은 기본 복제본 1개와 읽기 전용 복제본 2개, 노드 OSS 3개(--num-cache-clusters)`my-classic-rg`로 Valkey 또는 Redis(클러스터 모드 비활성화됨) 복제 그룹을 생성합니다. 이 복제 그룹(-at-rest-encryption-enabled-)에 대해 유휴 시 암호화가 활성화됩니다.

다음 파라미터와 해당 값은 복제 그룹에 대한 암호화를 활성화하는 데 필요합니다.

키 파라미터

- **`--engine`**- `valkey` 또는 이어야 합니다`redis`.
- **`--engine-version`**—엔진이 Redis 인 경우 는 3.2.6, 4.0.10 이상이어야 OSS합니다.
- **`--at-rest-encryption-enabled`** - 미사용 데이터 암호화를 활성화하기 위해 필요합니다.

Example 1: 복제본이 있는 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터

Linux, macOS, Unix의 경우:

```
aws elasticache create-replication-group \
    --replication-group-id my-classic-rg \
    --replication-group-description "3 node replication group" \
    --cache-node-type cache.m4.large \
    --engine redis \
    --at-rest-encryption-enabled \
    --num-cache-clusters 3
```

Windows의 경우:

```
aws elasticache create-replication-group ^
    --replication-group-id my-classic-rg ^
    --replication-group-description "3 node replication group" ^
    --cache-node-type cache.m4.large ^
    --engine redis ^
    --at-rest-encryption-enabled ^
    --num-cache-clusters 3 ^
```

추가 정보는 다음을 참조하세요.

- Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹을 처음부터 생성(AWS CLI)
- create-replication-group

클러스터 for Valkey 또는 Redis에서 저장 시 암호화 활성화OSS(클러스터 모드 활성화됨)(CLI)

다음 작업은 3개의 노드 그룹 또는 샤드OSS(--num-node-groups)를 `my-clustered-rg` 사용하여 Valkey 또는 Redis(클러스터 모드 활성화됨) 복제 그룹을 생성합니다. 각 노드에는 기본 복제본 1개와 읽기 전용 복제본 2개(--replicas-per-node-group)의 노드 3개가 있습니다. 이 복제 그룹(-at-rest-encryption-enabled-)에 대해 유휴 시 암호화가 활성화됩니다.

다음 파라미터와 해당 값은 복제 그룹에 대한 암호화를 활성화하는 데 필요합니다.

키 파라미터

- **`--engine`**- `valkey` 또는 이어야 합니다`redis`.
- **`--engine-version`**- 엔진이 Redis 인 경우 는 4.0.10 이상이어야 OSS합니다.
- **`--at-rest-encryption-enabled`** - 미사용 데이터 암호화를 활성화하기 위해 필요합니다.
- **`--cache-parameter-group`** - 이 클러스터 모드가 활성화된 복제 그룹을 만들려면 `default-redis4.0.cluster.on` 또는 이 클러스터에서 파생된 클러스터여야 합니다.

Example 2: Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터

Linux, macOS, Unix의 경우:

```
aws elasticache create-replication-group \
   --replication-group-id my-clustered-rg \
   --replication-group-description "redis clustered cluster" \
   --cache-node-type cache.m3.large \
   --num-node-groups 3 \
   --replicas-per-node-group 2 \
   --engine redis \
   --engine-version 6.2 \
   --at-rest-encryption-enabled \
   --cache-parameter-group default.redis6.x.cluster.on
```

Windows의 경우:

```
aws elasticache create-replication-group ^
   --replication-group-id my-clustered-rg ^
   --replication-group-description "redis clustered cluster" ^
   --cache-node-type cache.m3.large ^
   --num-node-groups 3 ^
```

```
    --replicas-per-node-group 2 ^
    --engine redis ^
    --engine-version 6.2 ^
    --at-rest-encryption-enabled ^
    --cache-parameter-group default.redis6.x.cluster.on
```

추가 정보는 다음을 참조하세요.

- 처음부터 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹 생성(AWS CLI)
- create-replication-group

참고

- Amazon VPCs 및 ElastiCache 보안
- Amazon용 자격 증명 및 액세스 관리 ElastiCache

인증 및 권한 부여

AWS Identity and Access Management(IAM)는 AWS 리소스에 대한 액세스를 안전하게 제어하는 데 도움이 되는 웹 서비스입니다. ElastiCache 는 IAM 및 Valkey 및 Redis OSS AUTH 명령을 사용하여 사용자를 인증하고 역할 기반 액세스 제어()를 사용하여 사용자 작업을 승인합니다RBAC.

주제

- 역할 기반 액세스 제어(RBAC)
- Valkey 및 Redis OSS AUTH 명령을 사용한 인증
- Valkey 또는 Redis OSS 캐시에서 ElastiCache 액세스 제어 비활성화

역할 기반 액세스 제어(RBAC)

에 설명된 대로 Valkey 및 Redis OSS AUTH 명령을 사용하여 사용자를 인증하는 대신 Valkey 7.2 이상 및 Redis OSS 6.0 이상Valkey 및 Redis OSS AUTH 명령을 사용한 인증에서 역할 기반 액세스 제어() 라는 기능을 사용할 수 있습니다RBAC. RBAC 또한 서버리스 캐시에 대한 액세스를 제어하는 유일한 방법입니다.

토큰이 인증된 경우 모든 인증된 클라이언트AUTH가 전체 캐시 액세스 권한을 갖는 Valkey 및 Redis와 달리 OSS 를 RBAC 사용하면 사용자 그룹을 통해 캐시 액세스를 제어할 수 있습니다. 이러한 사용자 그룹은 캐시에 대한 액세스를 구성하는 방법으로 설계되었습니다.

를 사용하면 다음 설명과 같이 사용자를 RBAC생성하고 액세스 문자열을 사용하여 특정 권한을 할당할 수 있습니다. 사용자를 특정 역할(관리자, 인적 자원)과 정렬된 사용자 그룹에 할당한 다음 하나 이상의 ElastiCache 캐시에 배포합니다. 이렇게 하면 동일한 Valkey 또는 Redis OSS 캐시 또는 캐시를 사용하여 클라이언트 간에 보안 경계를 설정하고 클라이언트가 서로의 데이터에 액세스하지 못하게 할 수 있습니다.

> **Note**
>
> Valkey 클러스터RBAC와 함께 를 사용하는 경우에도 엔진 “redis”를 사용자 및 사용자 그룹에 할당해야 합니다.

RBAC 는 Redis OSS 6ACL에서 의 도입을 지원하도록 설계되었습니다. ElastiCache Valkey 또는 Redis OSS 캐시RBAC와 함께 를 사용하는 경우 몇 가지 제한 사항이 있습니다.

- 액세스 문자열에는 암호를 지정할 수 없습니다. CreateUser 또는 ModifyUser 호출을 사용하여 암호를 설정합니다.
- 사용자 권한의 경우 on 및 `off`를 액세스 문자열의 일부로 전달합니다. 액세스 문자열에 둘 다 지정되지 않은 경우, 사용자에게 `off`가 할당되며 캐시에 대한 액세스 권한은 없습니다.
- 금지되거나 이름이 바뀐 명령은 사용할 수 없습니다. 금지되거나 이름이 바뀐 명령을 지정하면 예외가 발생합니다. 이름이 변경된 명령에 액세스 제어 목록(ACLs)을 사용하려면 명령의 원래 이름, 즉 이름이 변경되기 전에 명령의 이름을 지정합니다.
- `reset` 명령을 액세스 문자열의 일부로 사용할 수 없습니다. API 파라미터로 암호를 지정하면 ElastiCache (Redis OSS)가 암호를 관리합니다. 따라서 사용자의 모든 암호를 제거할 수 있는 `reset`을 사용할 수 없습니다.
- Redis OSS 6는 ACL LIST 명령을 도입합니다. 이 명령은 각 사용자에게 적용되는 ACL 규칙과 함께 사용자 목록을 반환합니다. ElastiCache (Redis OSS)는 `ACL LIST` 명령을 지원하지만 Redis와 마찬가지로 암호 해시를 지원하지 않습니다OSS. ElastiCache (Redis OSS)를 사용하면 describe-users 작업을 사용하여 액세스 문자열에 포함된 규칙을 포함하여 유사한 정보를 가져올 수 있습니다. 그러나 describe-users는 사용자 암호를 검색하지 않습니다.

  Valkey 및 Redis ElastiCache 에서 가 지원하는 다른 읽기 전용 명령에는 ACL WHOAMI, 및 ACL CATACLUSERS가 OSS 포함됩니다. ElastiCache Valkey 및 RedisOSS에서는 다른 쓰기 기반 ACL 명령을 지원하지 않습니다.
- 다음과 같은 제약이 적용됩니다.

Resource	최대 허용
사용자 그룹당 사용자	100
사용자 수	1000
사용자 그룹 수	100

ElastiCache (Redis OSS)와 RBAC 함께 를 사용하는 방법은 다음과 같습니다.

주제

- 액세스 문자열을 사용하여 권한 지정
- Valkey 또는 Redis를 ElastiCache 사용하여 의 캐시RBAC에 적용 OSS
- 에서 AUTH로 마이그레이션 RBAC
- 에서 RBAC로 마이그레이션 AUTH
- 사용자 암호 자동 교체
- IAM을 사용하는 인증

액세스 문자열을 사용하여 권한 지정

ElastiCache (Redis OSS) 캐시에 대한 권한을 지정하려면 AWS CLI 또는 를 사용하여 액세스 문자열을 생성하고 사용자에게 할당합니다 AWS Management Console.

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 섹션을 참조하세요ACL.

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.
- +@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 다음으로 제한됩니다.*## 1*.

+@category - 사용자의 액세스는 명령 범주로 제한됩니다.

사용자에게 액세스 문자열을 할당하는 방법에 대한 자세한 내용은 콘솔 및 를 사용하여 사용자 및 사용자 그룹 생성 CLI 섹션을 참조하세요.

기존 워크로드를 로 마이그레이션하는 경우 사용자 및 암호 ACL LIST해시를 제외하고 를 호출하여 액세스 문자열을 검색할 ElastiCache수 있습니다.

Redis OSS 버전 6.2 이상의 경우 다음 액세스 문자열 구문도 지원됩니다.

- &* - 사용 가능한 모든 채널에 대한 액세스 권한을 부여합니다.

Redis OSS 버전 7.0 이상의 경우 다음 액세스 문자열 구문도 지원됩니다.

- | - 하위 명령(예: “-config|set”)을 차단하는 데 사용 가능합니다.
- %R~<pattern> - 지정된 읽기 키 패턴을 추가합니다. 이는 일반 키 패턴과 유사하게 동작하지만 주어진 패턴과 일치하는 키에서 읽을 수 있는 권한만 부여합니다. 자세한 내용은 키 권한을 참조하세요.
- %W~<pattern> - 지정된 쓰기 키 패턴을 추가합니다. 이는 일반 키 패턴과 유사하게 동작하지만 주어진 패턴과 일치하는 키로 쓸 수 있는 권한만 부여합니다. 자세한 내용은 ACL 키 권한을 참조하세요.
- %RW~<pattern> - ~<pattern>의 다른 형태입니다.

- (<rule list>) - 규칙을 일치시킬 새 선택기를 생성합니다. 선택기는 사용자 권한 이후, 정의된 순서에 따라 평가됩니다. 명령이 사용자 권한과 일치하거나, 일치하는 선택기가 있으면 허용됩니다. 자세한 내용은 ACL 선택기를 참조하세요.
- clearselectors - 사용자에게 연결된 모든 선택기를 삭제합니다.

Valkey 또는 Redis를 ElastiCache 사용하여 의 캐시RBAC에 적용 OSS

Valkey 또는 Redis OSS 와 ElastiCache 함께 사용하려면 다음 단계를 RBAC수행합니다.

1. 하나 이상의 사용자를 생성합니다.
2. 사용자 그룹을 생성하고 사용자를 그룹에 추가합니다.
3. 전송 중 암호화가 활성화된 캐시에 사용자 그룹을 할당합니다.

이러한 단계는 다음에 자세히 설명되어 있습니다.

주제

- 콘솔 및 를 사용하여 사용자 및 사용자 그룹 생성 CLI
- 콘솔 및 를 사용하여 사용자 그룹 관리 CLI
- 서버리스 캐시에 사용자 그룹 할당
- 복제 그룹에 사용자 그룹 할당

콘솔 및 를 사용하여 사용자 및 사용자 그룹 생성 CLI

RBAC 사용자의 사용자 정보는 사용자 ID, 사용자 이름, 선택적으로 암호 및 액세스 문자열입니다. 액세스 문자열은 키와 명령에 대한 권한 수준을 제공합니다. 사용자 ID는 사용자마다 고유하며 사용자 이름은 엔진에 전달되는 이름입니다.

제공하는 사용자 권한이 사용자 그룹의 의도된 목적에 적합한지 확인합니다. 예를 들어, Administrators라는 사용자 그룹을 생성하는 경우 해당 그룹에 추가하는 모든 사용자는 키 및 명령에 대한 전체 액세스 권한으로 설정된 액세스 문자열을 가져야 합니다. e-commerce 사용자 그룹에 있는 사용자의 경우 액세스 문자열을 읽기 전용 액세스로 설정할 수 있습니다.

ElastiCache 는 사용자 ID와 사용자 이름으로 기본 사용자를 자동으로 구성"default"하고 모든 사용자 그룹에 추가합니다. 이 사용자는 수정하거나 삭제할 수 없습니다. 이 사용자는 이전 Redis OSS 버전의 기본 동작과 호환되도록 설계되었으며 모든 명령을 호출하고 모든 키에 액세스할 수 있는 액세스 문자열이 있습니다.

캐시에 적절한 액세스 제어를 추가하려면 이 기본 사용자를 활성화되지 않거나 강력한 암호를 사용하는 새 사용자로 바꿉니다. 기본 사용자를 변경하려면 사용자 이름이 `default`로 설정된 새 사용자를 생성합니다. 그런 다음 원래 기본 사용자를 새 사용자로 바꿉니다.

다음 절차에서는 원래 `default` 사용자를 수정된 액세스 문자열을 가진 다른 `default` 사용자로 바꾸는 방법을 보여 줍니다.

콘솔에서 기본 사용자 수정

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 왼쪽 탐색 창에서 사용자 그룹 관리를 선택합니다.
3. 사용자 그룹 ID에서 수정하려는 ID를 선택합니다. 확인란이 아니라 링크를 선택해야 합니다.
4. 수정을 선택합니다.
5. 수정 창에서 관리를 선택하고 사용자 이름에서 기본 사용자로 설정할 사용자를 선택합니다.
6. 선택을 선택합니다.
7. 수정을 선택합니다. 이렇게 하면 원래 기본 사용자가 가진 캐시에 대한 모든 기존 연결이 종료됩니다.

를 사용하여 기본 사용자를 수정하려면 AWS CLI

1. 다음 명령을 사용하여 사용자 이름이 `default`인 새 사용자를 생성합니다.

   Linux, macOS, Unix의 경우:

   ```
   aws elasticache create-user \
    --user-id "new-default-user" \
    --user-name "default" \
    --engine "REDIS" \
    --passwords "a-str0ng-pa))word" \
    --access-string "off +get ~keys*"
   ```

   Windows의 경우:

   ```
   aws elasticache create-user ^
    --user-id "new-default-user" ^
    --user-name "default" ^
    --engine "REDIS" ^
   ```

```
 --passwords "a-str0ng-pa))word" ^
 --access-string "off +get ~keys*"
```

2. 사용자 그룹을 생성하고 이전에 생성한 사용자를 추가합니다.

   Linux, macOS, Unix의 경우:

```
aws elasticache create-user-group \
  --user-group-id "new-group-2" \
  --engine "REDIS" \
  --user-ids "new-default-user"
```

   Windows의 경우:

```
aws elasticache create-user-group ^
  --user-group-id "new-group-2" ^
  --engine "REDIS" ^
  --user-ids "new-default-user"
```

3. 새 `default` 사용자로 원래 `default` 사용자를 바꿉니다.

   Linux, macOS, Unix의 경우:

```
aws elasticache modify-user-group \
    --user-group-id test-group \
    --user-ids-to-add "new-default-user" \
    --user-ids-to-remove "default"
```

   Windows의 경우:

```
aws elasticache modify-user-group ^
    --user-group-id test-group ^
    --user-ids-to-add "new-default-user" ^
    --user-ids-to-remove "default"
```

   이 수정 작업이 호출되면 원래 기본 사용자가 가진 캐시에 대한 모든 기존 연결이 종료됩니다.

사용자를 생성할 때 최대 두 개의 암호를 설정할 수 있습니다. 암호를 수정해도 캐시에 대한 모든 기존 연결은 유지됩니다.

특히 ElastiCache (Redis OSS)RBAC용 를 사용할 때 다음과 같은 사용자 암호 제약 조건에 유의하세요.

- 암호는 16~128자 길이의 인쇄 가능한 문자여야 합니다.
- 영숫자가 아닌 다음과 같은 문자는 허용되지 않습니다. `,` `"` `/` `@`.

콘솔 및 를 사용하여 사용자 관리 CLI

다음 절차에 따라 콘솔에서 사용자를 관리합니다.

콘솔에서 사용자를 관리하려면

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. Amazon ElastiCache 대시보드에서 사용자 관리 를 선택합니다. 다음과 같은 옵션을 사용할 수 있습니다.

   - 사용자 생성 - 사용자를 생성할 때 사용자 ID, 사용자 이름, 인증 모드, 액세스 문자열을 입력합니다. 액세스 문자열은 사용자에게 허용할 키와 명령에 대한 권한 수준을 설정합니다.

     사용자를 생성할 때 최대 두 개의 암호를 설정할 수 있습니다. 암호를 수정해도 캐시에 대한 모든 기존 연결은 유지됩니다.
   - 사용자 수정 - 사용자의 인증 설정을 업데이트하거나 액세스 문자열을 변경할 수 있습니다.
   - 사용자 삭제 - 계정이 속한 모든 사용자 관리 그룹에서 제거됩니다.

다음 절차에 따라 AWS CLI를 사용하여 사용자를 관리합니다.

를 사용하여 사용자를 수정하려면 CLI

- `modify-user` 명령을 사용하여 사용자의 암호를 업데이트하거나 사용자의 액세스 권한을 변경합니다.

  사용자를 수정하면 해당 사용자와 연결된 사용자 그룹이 업데이트되고, 해당 사용자 그룹과 연결된 캐시도 업데이트됩니다. 기존 연결은 모두 유지됩니다. 예를 들면 다음과 같습니다.

  Linux, macOS, Unix의 경우:

  ```
  aws elasticache modify-user \
    --user-id user-id-1 \
  ```

```
  --access-string "~objects:* ~items:* ~public:*" \
  --no-password-required
```

Windows의 경우:

```
aws elasticache modify-user ^
  --user-id user-id-1 ^
  --access-string "~objects:* ~items:* ~public:*" ^
  --no-password-required
```

> **Note**
>
> nopass 옵션을 사용하지 않는 것이 좋습니다. 사용해야 한다면 사용자 권한을 제한된 키 집합에 액세스할 수 있는 읽기 전용으로 설정하는 것이 좋습니다.

를 사용하여 사용자를 삭제하려면 CLI

- `delete-user` 명령을 사용하여 사용자를 삭제합니다. 계정이 삭제되고 해당 계정이 속한 모든 사용자 그룹에서 제거됩니다. 다음은 예입니다.

  Linux, macOS, Unix의 경우:

  ```
  aws elasticache delete-user \
    --user-id user-id-2
  ```

  Windows의 경우:

  ```
  aws elasticache delete-user ^
    --user-id user-id-2
  ```

사용자 목록을 보려면 describe-users 작업을 호출합니다.

```
aws elasticache describe-users
```

콘솔 및 를 사용하여 사용자 그룹 관리 CLI

다음과 같이 사용자 그룹을 생성하여 하나 이상의 캐시에 대한 사용자 액세스를 구성하고 제어할 수 있습니다.

다음 절차에 따라 콘솔을 사용하여 사용자 그룹을 관리합니다.

콘솔을 사용하여 사용자 그룹을 관리하려면

1. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. Amazon ElastiCache 대시보드에서 사용자 그룹 관리 를 선택합니다.

   다음 작업을 사용하여 새 사용자 그룹을 생성할 수 있습니다.

   - 생성 - 사용자 그룹을 생성할 때 사용자를 추가한 다음 캐시에 사용자 그룹을 할당합니다. 예를 들어 캐시에서 관리 역할을 가진 사용자의 `Admin` 사용자 그룹을 생성할 수 있습니다.

     > **Important**
     >
     > 사용자 그룹을 만들 때 기본 사용자를 포함해야 합니다.

   - 사용자 추가 - 사용자를 사용자 그룹에 추가합니다.
   - 사용자 제거 - 사용자 그룹에서 사용자를 제거합니다. 사용자 그룹에서 사용자를 제거하면 사용자가 가진 캐시에 대한 모든 기존 연결이 종료됩니다.
   - 삭제 - 사용자 그룹을 삭제하는 데 사용합니다. 그룹에 속한 사용자가 아닌 사용자 그룹 자체가 삭제된다는 것에 주의하세요.

   기존 사용자 그룹의 경우 다음 작업을 수행할 수 있습니다.

   - 사용자 추가 - 기존 사용자를 사용자 그룹에 추가합니다.
   - 사용자 삭제 - 사용자 그룹에서 기존 사용자를 제거합니다.

     > **Note**
     >
     > 사용자가 사용자 그룹에서 제거되지만 시스템에서 삭제되지는 않습니다.

다음 절차에 따라 를 사용하여 사용자 그룹을 관리합니다CLI.

를 사용하여 새 사용자 그룹을 생성하고 사용자를 추가하려면 CLI

- 다음과 같이 `create-user-group` 명령을 사용합니다.

  Linux, macOS, Unix의 경우:

```
aws elasticache create-user-group \
  --user-group-id "new-group-1" \
  --engine "REDIS" \
  --user-ids user-id-1, user-id-2
```

  Windows의 경우:

```
aws elasticache create-user-group ^
  --user-group-id "new-group-1" ^
  --engine "REDIS" ^
  --user-ids user-id-1, user-id-2
```

새 사용자를 추가하거나 를 사용하여 현재 멤버를 제거하여 사용자 그룹을 수정하려면 CLI

- 다음과 같이 `modify-user-group` 명령을 사용합니다.

  Linux, macOS, Unix의 경우:

```
aws elasticache modify-user-group --user-group-id new-group-1 \
--user-ids-to-add user-id-3 \
--user-ids-to-remove user-id-2
```

  Windows의 경우:

```
aws elasticache modify-user-group --user-group-id new-group-1 ^
--user-ids-to-add userid-3 ^
--user-ids-to-removere user-id-2
```

> **Note**
>
> 사용자 그룹에서 제거된 사용자에게 속하는 열려 있는 모든 연결이 이 명령으로 종료됩니다.

를 사용하여 사용자 그룹을 삭제하려면 CLI

- 다음과 같이 `delete-user-group` 명령을 사용합니다. 그룹에 속한 사용자가 아닌 사용자 그룹 자체가 삭제됩니다.

  Linux, macOS, Unix의 경우:

  ```
  aws elasticache delete-user-group /
     --user-group-id
  ```

  Windows의 경우:

  ```
  aws elasticache delete-user-group ^
     --user-group-id
  ```

사용자 그룹 목록을 보려면 describe-user-groups 작업을 호출할 수 있습니다.

```
aws elasticache describe-user-groups \
  --user-group-id test-group
```

서버리스 캐시에 사용자 그룹 할당

사용자 그룹을 생성하고 사용자를 추가한 후 구현의 마지막 단계는 서버리스 캐시에 사용자 그룹을 할당하는 RBAC 것입니다.

콘솔을 사용해 서버리스 캐시에 사용자 그룹 할당

를 사용하여 서버리스 캐시에 사용자 그룹을 추가하려면 다음을 AWS Management Console수행합니다.

- 클러스터 모드가 비활성화된 경우 Valkey(클러스터 모드 비활성화됨) 클러스터 생성(콘솔) 섹션을 참조하세요.
- 클러스터 모드가 활성화된 경우 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 생성(콘솔) 섹션을 참조하세요.

를 사용하여 서버리스 캐시에 사용자 그룹 할당 AWS CLI

다음 AWS CLI 작업은 값이 인 user-group-id 파라미터를 사용하여 서버리스 캐시를 생성합니다*my-user-group-id*. 서브넷 그룹 `sng-test`는 존재하는 서브넷 그룹으로 대체합니다.

키 파라미터

- **`--engine`** – `valkey` 또는 여야 합니다`redis`.
- **`--user-group-id`** - 이 값으로 사용자 그룹의 ID를 확인할 수 있으며, 이 그룹은 캐시에 대해 특정 액세스 권한을 보유한 사용자로 구성되어 있습니다.

Linux, macOS, Unix의 경우:

```
aws elasticache create-serverless-cache \
    --serverless-cache-name "new-serverless-cache" \
    --description "new-serverless-cache" \
    --engine "redis" \
    --user-group-id "new-group-1"
```

Windows의 경우:

```
aws elasticache create-serverless-cache ^
    --serverless-cache-name "new-serverless-cache" ^
    --description "new-serverless-cache" ^
    --engine "redis" ^
    --user-group-id "new-group-1"
```

다음 AWS CLI 작업은 값이 인 user-group-id 파라미터로 서버리스 캐시를 수정합니다*my-user-group-id*.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-serverless-cache \
    --serverless-cache-name serverless-cache-1 \
    --user-group-id "new-group-2"
```

Windows의 경우:

```
aws elasticache modify-serverless-cache ^
    --serverless-cache-name serverless-cache-1 ^
    --user-group-id "new-group-2"
```

캐시에 적용된 모든 수정 사항은 비동기식으로 업데이트됩니다. 이벤트를 보면서 진행 상태를 모니터링할 수 있습니다. 자세한 내용은 ElastiCache 이벤트 보기 단원을 참조하십시오.

복제 그룹에 사용자 그룹 할당

사용자 그룹을 생성하고 사용자를 추가한 후 구현의 마지막 단계는 복제 그룹에 사용자 그룹을 할당하는 RBAC 것입니다.

콘솔을 사용하여 복제 그룹에 사용자 그룹 할당

를 사용하여 복제에 사용자 그룹을 추가하려면 다음을 AWS Management Console수행합니다.

- 클러스터 모드가 비활성화된 경우 Valkey(클러스터 모드 비활성화됨) 클러스터 생성(콘솔) 섹션을 참조하세요.
- 클러스터 모드가 활성화된 경우 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 생성(콘솔) 섹션을 참조하세요.

를 사용하여 복제 그룹에 사용자 그룹 할당 AWS CLI

다음 AWS CLI 작업은 전송 중 암호화(TLS)가 활성화된 복제 그룹과 값이 인 user-group-ids 파라미터를 생성합니다*my-user-group-id*. 서브넷 그룹 `sng-test`는 존재하는 서브넷 그룹으로 대체합니다.

키 파라미터

- **`--engine`** – `valkey` 또는 여야 합니다`redis`.
- **`--engine-version`** - 6.0 이상이어야 합니다.
- **`--transit-encryption-enabled`** - 인증 및 사용자 그룹 연결에 필요합니다.
- **`--user-group-ids`** - 이 값으로 사용자 그룹의 ID를 확인할 수 있으며, 이 그룹은 캐시에 대해 특정 액세스 권한을 보유한 사용자로 구성되어 있습니다.
- **`--cache-subnet-group`** - 사용자 그룹 연결에 필요합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache create-replication-group \
    --replication-group-id "new-replication-group" \
    --replication-group-description "new-replication-group" \
    --engine "redis" \
    --cache-node-type cache.m5.large \
    --transit-encryption-enabled \
    --user-group-ids "new-group-1" \
```

```
    --cache-subnet-group "cache-subnet-group"
```

Windows의 경우:

```
aws elasticache create-replication-group ^
    --replication-group-id "new-replication-group" ^
    --replication-group-description "new-replication-group" ^
    --engine "redis" ^
    --cache-node-type cache.m5.large ^
    --transit-encryption-enabled ^
    --user-group-ids "new-group-1" ^
    --cache-subnet-group "cache-subnet-group"
```

다음 AWS CLI 작업은 전송 중 암호화(TLS)가 활성화된 복제 그룹과 값이 인 user-group-ids 파라미터를 수정합니다*my-user-group-id*.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-replication-group \
    --replication-group-id replication-group-1 \
    --user-group-ids-to-remove "new-group-1" \
    --user-group-ids-to-add "new-group-2"
```

Windows의 경우:

```
aws elasticache modify-replication-group ^
    --replication-group-id replication-group-1 ^
    --user-group-ids-to-remove "new-group-1" ^
    --user-group-ids-to-add "new-group-2"
```

응답의 PendingChanges에 주의하세요. 캐시에 적용된 모든 수정 사항은 비동기식으로 업데이트됩니다. 이벤트를 보면서 진행 상태를 모니터링할 수 있습니다. 자세한 내용은 ElastiCache 이벤트 보기 단원을 참조하십시오.

에서 AUTH로 마이그레이션 RBAC

에 설명된 AUTH 대로 를 사용하고 Valkey 및 Redis OSS AUTH 명령을 사용한 인증 있고 를 사용하여로 마이그레이션하려는 경우 다음 절차를 RBAC사용합니다.

콘솔을 RBAC 사용하여 에서 AUTH 로 마이그레이션하려면 다음 절차를 사용합니다.

콘솔을 RBAC 사용하여 Valkey 또는 Redis에서 OSS AUTH 로 마이그레이션하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 오른쪽 상단 모서리의 목록에서 수정하려는 캐시가 있는 AWS 리전을 선택합니다.
3. 탐색 창에서, 수정하려는 캐시에서 실행 중인 엔진을 선택합니다.

   선택한 엔진의 캐시 목록이 나타납니다.
4. 캐시 목록에서 수정할 캐시의 이름을 선택합니다.
5. 작업에서 수정을 선택합니다.

   수정 창이 나타납니다.
6. 액세스 제어에서 사용자 그룹 액세스 제어 목록을 선택합니다.
7. 사용자 그룹 액세스 제어 목록에서 사용자 그룹을 선택합니다.
8. 변경 사항 미리보기를 선택하면 나오는 다음 화면에서 수정을 선택합니다.

다음 절차에 따라 를 RBAC 사용하여 Valkey 또는 Redis에서 OSSAUTH로 마이그레이션합니다CLI.

를 RBAC 사용하여 에서 AUTH로 마이그레이션하려면 CLI

- 다음과 같이 `modify-replication-group` 명령을 사용합니다.

  Linux, macOS, Unix의 경우:

```
aws elasticache modify-replication-group --replication-group-id test \
    --auth-token-update-strategy DELETE \
    --user-group-ids-to-add user-group-1
```

  Windows의 경우:

```
aws elasticache modify-replication-group --replication-group-id test ^
    --auth-token-update-strategy DELETE ^
    --user-group-ids-to-add user-group-1
```

에서 RBAC로 마이그레이션 AUTH

를 사용하고 RBAC 있고 Redis OSS AUTH 로 마이그레이션하려는 경우 섹션을 참조하세요에서 RBAC로 마이그레이션 AUTH.

> **Note**
>
> ElastiCache 캐시에서 액세스 제어를 비활성화해야 하는 경우 를 통해 액세스 제어를 수행해야 합니다 AWS CLI. 자세한 내용은 the section called “Valkey 또는 Redis OSS 캐시에서 ElastiCache 액세스 제어 비활성화” 단원을 참조하십시오.

사용자 암호 자동 교체

를 사용하면 코드의 하드코딩된 보안 인증 정보(암호 포함)를 Secrets Manager에 API 직접 호출하여 프로그래밍 방식으로 보안 암호를 검색할 AWS Secrets Manager수 있습니다. 이렇게 하면 보안 암호가 해당 위치에 있지 않기 때문에 여러분의 코드를 검사하는 누군가에 의해 보안 암호가 손상되지 않도록 방지할 수 있습니다. 또한 사용자가 지정한 일정에 따라 Secrets Manager가 자동으로 보안 암호를 교체하도록 구성할 수 있습니다. 따라서 단기 보안 암호로 장기 보안 암호를 교체할 수 있어 손상 위험이 크게 줄어듭니다.

Secrets Manager를 사용하면 Secrets Manager가 제공하는 AWS Lambda 함수를 사용하여 ElastiCache (Redis OSS) 암호(즉, 암호)를 자동으로 교체할 수 있습니다.

에 대한 자세한 내용은 란 무엇입니까 AWS Secrets Manager?를 AWS Secrets Manager참조하세요.

가 보안 암호를 ElastiCache 사용하는 방법

Valkey 7.2에는 Redis OSS 7.0과 동등한 기능이 설정되어 있습니다. Redis OSS 6에서는 Valkey 또는 Redis OSS 클러스터를 보호하는 역할 기반 액세스 제어(RBAC) 방법을 ElastiCache 도입했습니다. 이 기능을 사용하면 실행 가능한 명령 및 액세스 가능한 키 측면에서 특정 연결을 제한할 수 있습니다. 를 사용하면 고객이 암호로 사용자를 생성하는 RBAC동안 암호 값을 일반 텍스트로 수동으로 입력해야 하며, 이는 작업자에게 표시됩니다.

Secrets Manager를 이용하면 애플리케이션이 Secrets Manager에서 암호를 가져와 애플리케이션 구성에 저장하므로 수동으로 입력할 필요가 없습니다. 이렇게 하는 방법에 대한 정보는 ElastiCache 사용자가 보안 암호와 연결되는 방법 단원을 참조하십시오.

보안 암호 사용 시 비용이 발생합니다. 요금 정보는 AWS Secrets Manager 요금을 참조하세요.

ElastiCache 사용자가 보안 암호와 연결되는 방법

Secrets Manager는 보안 암호의 SecretString 필드에 연결된 사용자에 대한 참조를 보관합니다. ElastiCache 측의 보안 암호에 대한 참조는 없습니다.

```
{
    "password": "strongpassword",
    "username": "user1",
    "user_arn": "arn:aws:elasticache:us-east-1:xxxxxxxxxx918:user:user1" //this is the
 bond between the secret and the user
}
```

Lambda 교체 함수

Secrets Manager 자동 암호 교체를 활성화하려면 수정 사용자와 상호 작용API하여 사용자의 암호를 업데이트하는 Lambda 함수를 생성합니다.

작동 방식에 대한 자세한 내용은 교체 작동 방식을 참조하세요.

> **Note**
>
> 일부 AWS 서비스의 경우 혼동된 대리자 시나리오를 방지하려면 aws:SourceArn 및 aws:SourceAccount 전역 조건 키를 모두 사용하는 것이 AWS 좋습니다. 그러나 교체 함수 정책에 aws:SourceArn 조건을 포함하는 경우 교체 함수는 해당 에서 지정한 보안 암호를 교체하는 데만 사용할 수 있습니다ARN. 여러 보안 암호에 대해 교체 기능을 사용할 수 있도록 컨텍스트 키 aws:SourceAccount만 포함하는 것이 좋습니다.

발생할 수 있는 모든 문제는 AWS Secrets Manager 교체 문제 해결을 참조하세요.

ElastiCache 사용자를 생성하고 Secrets Manager와 연결하는 방법

다음 단계에서는 사용자를 생성하고 Secrets Manager와 연결하는 방법을 보여줍니다.

1. 비활성 사용자 생성

   Linux, macOS, Unix의 경우:

   ```
   aws elasticache create-user \
    --user-id user1 \
   ```

```
 --user-name user1 \
 --engine "REDIS" \
 --no-password \ // no authentication is required
 --access-string "*off* +get ~keys*" // this disables the user
```

Windows의 경우:

```
aws elasticache create-user ^
 --user-id user1 ^
 --user-name user1 ^
 --engine "REDIS" ^
 --no-password ^ // no authentication is required
 --access-string "*off* +get ~keys*" // this disables the user
```

다음과 비슷한 응답이 나타납니다.

```
{
    "UserId": "user1",
    "UserName": "user1",
    "Status": "active",
    "Engine": "redis",
    "AccessString": "off ~keys* -@all +get",
    "UserGroupIds": [],
    "Authentication": {
        "Type": "no_password"
    },
    "ARN": "arn:aws:elasticache:us-east-1:xxxxxxxxxx918:user:user1"
}
```

2. 보안 암호 생성

Linux, macOS, Unix의 경우:

```
aws secretsmanager create-secret \
--name production/ec/user1 \
--secret-string \
'{
   "user_arn": "arn:aws:elasticache:us-east-1:123456xxxx:user:user1",
    "username":"user1"
 }'
```

Windows의 경우:

```
aws secretsmanager create-secret ^
--name production/ec/user1 ^
--secret-string ^
'{
   "user_arn": "arn:aws:elasticache:us-east-1:123456xxxx:user:user1",
    "username":"user1"
 }'
```

다음과 비슷한 응답이 나타납니다.

```
{
 "ARN": "arn:aws:secretsmanager:us-east-1:123456xxxx:secret:production/ec/user1-
eaFois",
 "Name": "production/ec/user1",
 "VersionId": "aae5b963-1e6b-4250-91c6-ebd6c47d0d95"
}
```

3. 암호를 교체하도록 Lambda 함수 구성

   a. 에 로그인 AWS Management Console 하고 에서 Lambda 콘솔을 엽니다. https://console.aws.amazon.com/lambda/

   b. 탐색 창에서 함수를 선택한 후, 생성해 둔 함수를 선택합니다. 왼쪽에 있는 확인란이 아닌, 함수 이름을 선택합니다.

   c. 구성 탭을 선택합니다.

   d. 일반 구성에서 편집을 선택한 다음, 제한 시간을 최소 12분으로 설정합니다.

   e. 저장(Save)을 선택합니다.

   f. 환경 변수를 선택한 후, 다음을 설정합니다.

      i. SECRETS_MANAGER_ENDPOINT – https://secretsmanager.**REGION**.amazonaws.com

      ii. SECRET_ARN – 2단계에서 생성한 보안 암호의 Amazon 리소스 이름(ARN)입니다.

      iii. USER_NAME – 사용자의 사용자 이름 ElastiCache ,

      iv. 저장(Save)을 선택합니다.

   g. 권한을 선택합니다.

   h. 실행 역할 에서 IAM 콘솔에서 볼 Lambda 함수 역할의 이름을 선택합니다.

   i. Lambda 함수에 다음 권한이 있어야 사용자를 수정하고 암호를 설정할 수 있습니다.

      ElastiCache

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Effect": "Allow",
        "Action": [
            "elasticache:DescribeUsers",
            "elasticache:ModifyUser"
        ],
        "Resource": "arn:aws:elasticache:us-east-1:xxxxxxxxxx918:user:user1"
        }
    ]
}
```

Secrets Manager

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret",
                "secretsmanager:PutSecretValue",
                "secretsmanager:UpdateSecretVersionStage"
            ],
            "Resource": "arn:aws:secretsmanager:us-
east-1:xxxxxxxxxxx:secret:XXXX"
        },
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetRandomPassword",
            "Resource": "*"
        }
    ]
}
```

4. Secrets Manager 보안 암호 교체 설정

   a. 를 사용하여 콘솔을 사용하여 Secrets Manager 보안 암호의 자동 교체 설정을 AWS Management Console참조하세요. AWS

교체 일정 설정에 대한 자세한 내용은 Secrets Manager 교체 일정 표현식을 참조하세요.

b. 를 사용하여 를 사용하기 위한 자동 교체 설정을 AWS CLI참조하세요. AWS Secrets ManagerAWS Command Line Interface

IAM을 사용하는 인증

주제

- 개요
- 제한 사항
- 설정
- Connecting

개요

IAM 인증을 사용하면 캐시가 Valkey 또는 Redis OSS 버전 7 이상을 사용하도록 구성된 경우 자격 증명을 OSS 사용하여 AWS IAM Valkey 또는 Redis ElastiCache 와의 연결을 인증할 수 있습니다. 그러면 보안 모델이 강화되고 여러 보안 관리 작업이 단순화됩니다. 또한 IAM 인증을 사용하여 최소 권한 원칙에 따라 각 개별 ElastiCache 캐시 및 ElastiCache 사용자에 대해 세분화된 액세스 제어를 구성할 수 있습니다. IAM Valkey 또는 Redis를 ElastiCache 사용한 에 대한 인증은 Valkey 또는 Redis OSS `AUTH` 또는 `HELLO` 명령에서 수명이 긴 ElastiCache 사용자 암호 대신 수명이 짧은 IAM 인증 토큰을 제공하는 방식으로 OSS 작동합니다. IAM 인증 토큰에 대한 자세한 내용은 AWS 일반 참조 가이드의 서명 버전 4 서명 프로세스와 아래 코드 예제를 참조하세요.

IAM 자격 증명 및 관련 정책을 사용하여 Valkey 또는 Redis OSS 액세스를 추가로 제한할 수 있습니다. 페더레이션 ID 공급자의 사용자에게 Valkey 또는 Redis OSS 캐시에 대한 액세스 권한을 직접 부여할 수도 있습니다.

와 함께 를 사용하려면 AWS IAM ElastiCache먼저 인증 모드가 로 설정된 ElastiCache 사용자를 생성해야 합니다IAM. 그런 다음 IAM 자격 증명을 생성하거나 재사용할 수 있습니다. 자격 IAM 증명은 ElastiCache 캐시 및 ElastiCache 사용자에게 `elasticache:Connect` 작업을 부여하기 위한 관련 정책이 필요합니다. 구성되면 IAM 사용자 또는 역할의 AWS 자격 증명을 사용하여 IAM 인증 토큰을 생성할 수 있습니다. 마지막으로 캐시에 연결할 때 Valkey 또는 Redis OSS Client에서 수명이 짧은 IAM 인증 토큰을 암호로 제공해야 합니다. 자격 증명 공급자를 지원하는 Valkey 또는 Redis OSS 클라이언트는 새 연결마다 임시 자격 증명을 자동으로 생성할 수 있습니다. ElastiCache 는 IAM활성화된 ElastiCache 사용자의 연결 요청에 대한 IAM 인증을 수행하고 를 사용하여 연결 요청을 검증합니다 IAM.

제한 사항

IAM 인증을 사용하는 경우 다음 제한이 적용됩니다.

- IAM Valkey 7.2 이상 및 Redis OSS 버전 7.0 이상 ElastiCache 에서 를 사용할 때 인증을 사용할 수 있습니다.
- IAM활성화된 ElastiCache 사용자의 경우 사용자 이름과 사용자 ID 속성이 동일해야 합니다.
- IAM 인증 토큰은 15분 동안 유효합니다. 장기 연결의 경우 자격 증명 공급자 인터페이스를 지원하는 Valkey 또는 Redis OSS 클라이언트를 사용하는 것이 좋습니다.
- Valkey 또는 Redis ElastiCache 를 사용한 에 대한 IAM 인증된 연결OSS은 12시간 후에 자동으로 연결이 해제됩니다. 새 IAM 인증 토큰을 사용하여 AUTH 또는 HELLO 명령을 전송하여 12시간 동안 연결을 연장할 수 있습니다.
- IAM `MULTI EXEC` 명령에서는 인증이 지원되지 않습니다.
- 현재 IAM 인증은 다음과 같은 전역 조건 컨텍스트 키를 지원합니다.
  - 서버리스 캐시에서 IAM 인증을 사용하는 경우 , `aws:VpcSourceIp`, `aws:SourceVpcaws:SourceVpce`, `aws:CurrentTime`, `aws:EpochTime`, 및 `aws:ResourceTag/%s` (관련 서버리스 캐시 및 사용자)가 지원됩니다.
  - 복제 그룹 `aws:SourceIp` 및 `aws:ResourceTag/%s` (연결된 복제 그룹 및 사용자의)에서 IAM 인증을 사용하는 경우 지원됩니다.

  전역 조건 컨텍스트 키에 대한 자세한 내용은 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

설정

IAM 인증을 설정하려면:

1. 캐시 생성

```
aws elasticache create-serverless-cache \
  --serverless-cache-name cache-01  \
  --description "ElastiCache IAM auth application" \
  --engine redis
```

2. 계정이 새 역할을 수임할 수 있도록 아래에 표시된 대로 역할에 대한 IAM 신뢰 정책 문서를 생성합니다. 정책을 trust-policy.json 파일에 저장합니다.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Principal": { "AWS": "arn:aws:iam::123456789012:root" },
        "Action": "sts:AssumeRole"
    }
}
```

3. 아래와 같이 IAM 정책 문서를 생성합니다. 정책을 policy.json 파일에 저장합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect" : "Allow",
      "Action" : [
        "elasticache:Connect"
      ],
      "Resource" : [
        "arn:aws:elasticache:us-east-1:123456789012:serverlesscache:cache-01",
        "arn:aws:elasticache:us-east-1:123456789012:user:iam-user-01"
      ]
    }
  ]
}
```

4. IAM 역할을 생성합니다.

```
aws iam create-role \
--role-name "elasticache-iam-auth-app" \
--assume-role-policy-document file://trust-policy.json
```

5. IAM 정책을 생성합니다.

```
aws iam create-policy \
  --policy-name "elasticache-allow-all" \
  --policy-document file://policy.json
```

6. IAM 정책을 역할에 연결합니다.

```
aws iam attach-role-policy \
 --role-name "elasticache-iam-auth-app" \
```

```
  --policy-arn "arn:aws:iam::123456789012:policy/elasticache-allow-all"
```

7. 새 IAM사용 가능 사용자를 생성합니다.

```
aws elasticache create-user \
  --user-name iam-user-01 \
  --user-id iam-user-01 \
  --authentication-mode Type=iam \
  --engine redis \
  --access-string "on ~* +@all"
```

8. 사용자 그룹을 생성하고 사용자를 연결합니다.

```
aws elasticache create-user-group \
  --user-group-id iam-user-group-01 \
  --engine redis \
  --user-ids default iam-user-01

aws elasticache modify-serverless-cache \
  --serverless-cache-name cache-01  \
  --user-group-id iam-user-group-01
```

Connecting

토큰을 암호로 연결

먼저 AWS SigV4 사전 서명된 요청 을 사용하여 수명이 짧은 IAM 인증 토큰을 생성해야 합니다. 그런 다음 아래 예제와 같이 Valkey 또는 Redis OSS 캐시에 연결할 때 IAM 인증 토큰을 암호로 제공합니다.

```
String userId = "insert user id";
String cacheName = "insert cache name";
boolean isServerless = true;
String region = "insert region";

// Create a default AWS Credentials provider.
// This will look for AWS credentials defined in environment variables or system
 properties.
AWSCredentialsProvider awsCredentialsProvider = new
 DefaultAWSCredentialsProviderChain();

// Create an IAM authentication token request and signed it using the AWS credentials.
```

```
// The pre-signed request URL is used as an IAM authentication token for ElastiCache
 (Redis OSS).
IAMAuthTokenRequest iamAuthTokenRequest = new IAMAuthTokenRequest(userId, cacheName,
 region, isServerless);
String iamAuthToken =
 iamAuthTokenRequest.toSignedRequestUri(awsCredentialsProvider.getCredentials());

// Construct Redis OSS URL with IAM Auth credentials provider
RedisURI redisURI = RedisURI.builder()
    .withHost(host)
    .withPort(port)
    .withSsl(ssl)
    .withAuthentication(userId, iamAuthToken)
    .build();

// Create a new Lettuce Redis OSS client
RedisClient client = RedisClient.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 PARAM_RESOURCE_TYPE = "ResourceType";
    private static final String RESOURCE_TYPE_SERVERLESS_CACHE = "ServerlessCache";
    private static final String ACTION_NAME = "connect";
    private static final String SERVICE_NAME = "elasticache";
    private static final long TOKEN_EXPIRY_SECONDS = 900;

    private final String userId;
    private final String cacheName;
    private final String region;
    private final boolean isServerless;

    public IAMAuthTokenRequest(String userId, String cacheName, String region, boolean
 isServerless) {
        this.userId = userId;
        this.cacheName = cacheName;
        this.region = region;
        this.isServerless = isServerless;
```

```
    }

    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(userId));
        if (isServerless) {
            request.addParameters(PARAM_RESOURCE_TYPE,
Collections.singletonList(RESOURCE_TYPE_SERVERLESS_CACHE));
        }
        return request;
    }

    private URI getRequestUri() {
        return URI.create(String.format("%s%s/", REQUEST_PROTOCOL, cacheName));
    }

    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 인증 자격 증명 공급자를 ElastiCache 사용하여 를 IAM 인증하는 방법을 보여줍니다.

```
String userId = "insert user id";
String cacheName = "insert cache name";
boolean isServerless = true;
String region = "insert region";

// Create a default AWS Credentials provider.
// This will look for AWS credentials 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 ElastiCache (Redis OSS).
IAMAuthTokenRequest iamAuthTokenRequest = new IAMAuthTokenRequest(userId, cacheName,
 region, isServerless);

// Create a Redis OSS credentials provider using IAM credentials.
RedisCredentialsProvider redisCredentialsProvider = new
 RedisIAMAuthCredentialsProvider(
    userId, iamAuthTokenRequest, awsCredentialsProvider);

// Construct Redis OSS URL with IAM Auth credentials provider
RedisURI redisURI = RedisURI.builder()
    .withHost(host)
    .withPort(port)
    .withSsl(ssl)
    .withAuthentication(redisCredentialsProvider)
    .build();

// Create a new Lettuce Redis OSS client
RedisClient client = RedisClient.create(redisURI);
client.connect();
```

다음은 필요한 경우 임시 보안 인증 정보를 자동 생성하기 위해 보안 인증 정보 공급자 IAMAuthTokenRequest에서 를 래핑하는 Lettuce Redis OSS 클라이언트의 예입니다.

```
public class RedisIAMAuthCredentialsProvider implements RedisCredentialsProvider {
    private static final long TOKEN_EXPIRY_SECONDS = 900;

    private final AWSCredentialsProvider awsCredentialsProvider;
    private final String userId;
    private final IAMAuthTokenRequest iamAuthTokenRequest;
    private final Supplier<String> iamAuthTokenSupplier;

    public RedisIAMAuthCredentialsProvider(String userId,
        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(userId, iamAuthTokenSupplier.get()));
    }

    private String getIamAuthToken() {
        return
 iamAuthTokenRequest.toSignedRequestUri(awsCredentialsProvider.getCredentials());
    }
}
```

Valkey 및 Redis OSS AUTH 명령을 사용한 인증

> **Note**
>
> AUTH 가 로 대체되었습니다the section called “역할 기반 액세스 제어(RBAC)”. 모든 서버리스 캐시는 인증RBAC에 를 사용해야 합니다.

Valkey 및 Redis OSS 인증 토큰 또는 암호를 사용하면 클라이언트가 명령을 실행OSS하도록 허용하기 전에 Valkey 및 Redis에 암호가 필요하므로 데이터 보안이 향상됩니다. AUTH 는 자체 설계된 클러스터에만 사용할 수 있습니다.

주제

Valkey 및 Redis를 ElastiCache 사용한 AUTH 의 개요 OSS

를 Valkey 또는 Redis OSS 클러스터 ElastiCache 와 AUTH 함께 사용하는 경우 몇 가지 개선 사항이 있습니다.

특히 를 사용할 때 이러한 AUTH 토큰 또는 암호 제약 조건에 유의하세요AUTH.

- 토큰 또는 암호는 16~128자 길이의 인쇄 가능한 문자여야 합니다.
- 영숫자 외의 특수 문자는 (!, &, #, $, ^, <, >, -)로 제한됩니다.
- AUTH 는 ElastiCache Valkey 또는 Redis OSS 클러스터에서 활성화된 전송 중 암호화에만 활성화할 수 있습니다.

강력한 토큰을 설정하려면 다음 사항을 요구하는 등 엄격한 암호 정책을 따르는 것이 좋습니다.

- 토큰 또는 암호에는 다음 문자 유형 중 세 가지 이상이 포함되어야 합니다.
  - 대문자
  - 소문자
  - Digits
  - 영숫자 이외의 문자(!, &, #, $, ^, <, >, -)
- 토큰 또는 암호에는 사전 단어 또는 약간 수정된 사전 단어가 포함되어서는 안 됩니다.
- 토큰 또는 암호는 최근에 사용한 토큰과 같거나 비슷해서는 안 됩니다.

Valkey 또는 Redis OSS 클러스터 ElastiCache 를 사용하여 에 인증 적용

사용자가 토큰으로 보호된 Valkey 또는 Redis OSS 서버에 토큰(암호)을 입력하도록 요구할 수 있습니다. 이렇게 하려면 복제 그룹 또는 클러스터를 생성할 때 파라미터`--auth-token`(API: `AuthToken`)를 올바른 토큰과 함께 포함시킵니다. 또한 복제 그룹 또는 클러스터에 대한 모든 후속 명령에 이를 포함시킵니다.

다음 AWS CLI 작업은 전송 중 암호화(TLS)가 활성화되고 AUTH 토큰이 인 복제 그룹을 생성합니다*This-is-a-sample-token*. 서브넷 그룹 `sng-test`는 존재하는 서브넷 그룹으로 대체합니다.

키 파라미터

- **`--engine`** – `valkey` 또는 이어야 합니다`redis`.
- **`--engine-version`** – 엔진이 Redis 인 경우 는 3.2.6, 4.0.10 이상이어야 OSS합니다.
- **`--transit-encryption-enabled`** - 인증 및 HIPAA 자격에 필요합니다.
- **`--auth-token`** - HIPAA 자격에 필요합니다. 이 값은 토큰으로 보호된 이 Valkey 또는 Redis OSS 서버의 올바른 토큰이어야 합니다.
- **`--cache-subnet-group`** – HIPAA 자격에 필요합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache create-replication-group \
    --replication-group-id authtestgroup \
    --replication-group-description authtest \
    --engine redis \
    --cache-node-type cache.m4.large \
    --num-node-groups 1 \
    --replicas-per-node-group 2 \
    --transit-encryption-enabled \
    --auth-token This-is-a-sample-token \
    --cache-subnet-group sng-test
```

Windows의 경우:

```
aws elasticache create-replication-group ^
    --replication-group-id authtestgroup ^
    --replication-group-description authtest ^
    --engine redis ^
    --cache-node-type cache.m4.large ^
```

```
    --num-node-groups 1 ^
    --replicas-per-node-group 2 ^
    --transit-encryption-enabled ^
    --auth-token This-is-a-sample-token ^
    --cache-subnet-group sng-test
```

기존 클러스터에서 AUTH 토큰 수정

인증을 더 쉽게 업데이트할 수 있도록 클러스터에 사용되는 AUTH 토큰을 수정할 수 있습니다. 엔진 버전이 Valkey 7.2 이상 또는 Redis 5.0.6 이상인 경우 이 수정을 수행할 수 있습니다. 전송 시 암호화도 활성화되어 있어야 ElastiCache 합니다.

인증 토큰을 수정하면 ROTATE 및 두 가지 전략이 지원됩니다SET. ROTATE 전략은 이전 AUTH 토큰을 유지하면서 서버에 추가 토큰을 추가합니다. 이 SET 전략은 단일 AUTH 토큰만 지원하도록 서버를 업데이트합니다. `--apply-immediately` 파라미터를 통해 이러한 수정 호출을 수행하면 변경 사항이 즉시 적용됩니다.

AUTH 토큰 교체

Valkey 또는 Redis OSS 서버를 새 AUTH 토큰 로 업데이트하려면 `--auth-token` 파라미터가 새 AUTH 토큰인 `ModifyReplicationGroup`API와 값이 `--auth-token-update-strategy` 인 를 호출합니다ROTATE. ROTATE 수정이 완료되면 클러스터는 `auth-token` 파라미터에 지정된 토큰 외에도 이전 AUTH 토큰을 지원합니다. AUTH 토큰 교체 전에 복제 그룹에 AUTH 토큰이 구성되지 않은 경우 클러스터는 인증 없이 연결을 지원하는 것 외에도 `--auth-token` 파라미터에 지정된 AUTH 토큰을 지원합니다. 업데이트 전략 을 사용하여 AUTH 토큰을 필수로 업데이트AUTH 토큰 설정하려면 섹션을 참조하세요SET.

> **Note**
>
> 이전에 AUTH 토큰을 구성하지 않은 경우 수정이 완료되면 클러스터는 인증 토큰 파라미터에 지정된 AUTH 토큰 외에 토큰을 지원하지 않습니다.

이미 두 개의 AUTH 토큰을 지원하는 서버에서 이 수정을 수행하면 이 작업 중에 가장 오래된 AUTH 토큰도 제거됩니다. 이를 통해 서버는 지정된 시간에 최대 2개의 최신 AUTH 토큰을 지원할 수 있습니다.

이때 클라이언트를 업데이트하여 최신 AUTH 토큰을 사용하도록 할 수 있습니다. 클라이언트가 업데이트된 후 AUTH 토큰 교체 SET 전략(다음 섹션에서 설명됨)을 사용하여 새 토큰만 사용할 수 있습니다.

다음 AWS CLI 작업은 복제 그룹을 수정하여 AUTH 토큰 를 교체합니다*This-is-the-rotated-token*.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-replication-group \
--replication-group-id authtestgroup \
--auth-token This-is-the-rotated-token \
--auth-token-update-strategy ROTATE \
--apply-immediately
```

Windows의 경우:

```
aws elasticache modify-replication-group ^
--replication-group-id authtestgroup ^
--auth-token This-is-the-rotated-token ^
--auth-token-update-strategy ROTATE ^
--apply-immediately
```

AUTH 토큰 설정

단일 필수 AUTH 토큰을 지원하도록 Valkey 또는 Redis OSS 서버를 업데이트하려면 마지막 AUTH 토큰과 동일한 값을 가진 `--auth-token` 파라미터로 `ModifyReplicationGroup` API 작업을 호출하고 값이 인 `--auth-token-update-strategy` 파라미터를 호출합니다`SET`. 이 SET 전략은 이전에 ROTATE 전략을 사용할 때AUTH부터 토큰 2개 또는 선택적 AUTH 토큰 1개가 있는 클러스터에서만 사용할 수 있습니다. 수정이 완료되면 서버는 인증 AUTH 토큰 파라미터에 지정된 토큰만 지원합니다.

다음 AWS CLI 작업은 복제 그룹을 수정하여 AUTH 토큰을 로 설정합니다*This-is-the-set-token*.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-replication-group \
--replication-group-id authtestgroup \
--auth-token This-is-the-set-token \
--auth-token-update-strategy SET \
--apply-immediately
```

Windows의 경우:

```
aws elasticache modify-replication-group ^
```

```
--replication-group-id authtestgroup ^
--auth-token This-is-the-set-token ^
--auth-token-update-strategy SET ^
--apply-immediately
```

기존 클러스터에서 인증 활성화

기존 Valkey 또는 Redis OSS 서버에서 인증을 활성화하려면 `ModifyReplicationGroup` API 작업을 호출합니다. `--auth-token` 파라미터를 새 토큰으로 `ModifyReplicationGroup` 호출하고 값을 `--auth-token-update-strategy` 로 호출합니다ROTATE.

ROTATE 수정이 완료되면 클러스터는 인증 없이 연결을 지원하는 것 외에도 `--auth-token` 파라미터에 지정된 AUTH 토큰을 지원합니다. AUTH 토큰을 사용하여 Valkey 또는 RedisOSS에 인증하도록 모든 클라이언트 애플리케이션이 업데이트되면 SET 전략을 사용하여 AUTH 토큰을 필요에 따라 표시합니다. 인증 활성화는 전송 중 암호화(TLS)가 활성화된 Valkey 및 Redis OSS 서버에서만 지원됩니다.

에서 RBAC로 마이그레이션 AUTH

에 설명된 대로 Valkey 또는 Redis OSS 역할 기반 액세스 제어(RBAC)로 사용자를 인증역할 기반 액세스 제어(RBAC)하고 로 마이그레이션하려는 경우 다음 절차를 AUTH사용합니다. 콘솔 또는 를 사용하여 마이그레이션할 수 있습니다CLI.

콘솔을 AUTH 사용하여 에서 RBAC 로 마이그레이션하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 오른쪽 상단 모서리의 목록에서 수정하려는 클러스터가 있는 AWS 리전을 선택합니다.
3. 탐색 창에서 수정하려는 클러스터에서 실행 중인 엔진을 선택합니다.

   선택한 엔진의 클러스터 목록이 나타납니다.
4. 클러스터 목록에서 수정할 클러스터의 해당 이름을 선택합니다.
5. 작업에서 수정을 선택합니다.

   수정 창이 나타납니다.
6. 액세스 제어 에서 Valkey AUTH 기본 사용자 액세스 또는 Redis OSS AUTH 기본 사용자 액세스를 선택합니다.
7. Valkey AUTH 토큰 또는 Redis OSS AUTH 토큰에서 새 토큰을 설정합니다.

8. 변경 사항 미리보기를 선택하면 나오는 다음 화면에서 수정을 선택합니다.

를 AUTH 사용하여 에서 RBAC로 마이그레이션하려면 AWS CLI

다음 명령 중 하나를 사용하여 Valkey 또는 Redis OSS 복제 그룹에 대한 새 선택적 AUTH 토큰을 구성합니다. 선택적 인증 토큰은 SET 다음 단계의 업데이트 전략을 사용하여 인증 토큰이 필요한 것으로 표시될 때까지 복제 그룹에 대한 인증되지 않은 액세스를 허용합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-replication-group \
    --replication-group-id test \
    --remove-user-groups \
    --auth-token This-is-a-sample-token \
    --auth-token-update-strategy ROTATE \
    --apply-immediately
```

Windows의 경우:

```
aws elasticache modify-replication-group ^
    --replication-group-id test ^
    --remove-user-groups ^
    --auth-token This-is-a-sample-token ^
    --auth-token-update-strategy ROTATE ^
    --apply-immediately
```

위 명령을 실행한 후 새로 구성된 선택적 AUTH 토큰을 사용하여 Valkey 또는 Redis OSS 애플리케이션을 업데이트하여 ElastiCache 복제 그룹에 인증할 수 있습니다. 인증 토큰 교체를 완료하려면 아래 후속 명령SET의 업데이트 전략을 사용합니다. 이렇게 하면 필요에 따라 선택적 AUTH 토큰에 표시됩니다. 인증 토큰 업데이트가 완료되면 복제 그룹 상태가 로 표시되고 이 복제 그룹에 대한 ACTIVE 모든 연결에 인증이 필요합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-replication-group \
   --replication-group-id test \
   --auth-token This-is-a-sample-token \
   --auth-token-update-strategy SET \
   --apply-immediately
```

Windows의 경우:

```
aws elasticache modify-replication-group ^
   --replication-group-id test ^
   --remove-user-groups ^
   --auth-token This-is-a-sample-token ^
   --auth-token-update-strategy SET ^
   --apply-immediately
```

자세한 내용은 Valkey 및 Redis OSS AUTH 명령을 사용한 인증 단원을 참조하십시오.

> **Note**
>
> ElastiCache 클러스터에서 액세스 제어를 비활성화해야 하는 경우 섹션을 참조하세요the section called “Valkey 또는 Redis OSS 캐시에서 ElastiCache 액세스 제어 비활성화”.

Valkey 또는 Redis OSS 캐시에서 ElastiCache 액세스 제어 비활성화

아래 지침에 따라 Valkey 또는 Redis OSS TLS지원 캐시에서 액세스 제어를 비활성화합니다. 캐시에는 AUTH 기본 사용자 액세스 또는 사용자 그룹 액세스 제어 목록()의 두 가지 구성 유형 중 하나가 있습니다RBAC. AUTH 구성으로 캐시가 생성된 경우 사용자 그룹을 제거하여 캐시를 RBAC 비활성화하려면 먼저 구성으로 변경해야 합니다. RBAC 구성으로 캐시가 생성된 경우 바로 비활성화할 수 있습니다.

로 구성된 Valkey 또는 Redis OSS 서버리스 캐시를 비활성화하려면 RBAC

1. 액세스 제어를 비활성화하려면 사용자 그룹을 제거합니다.

   ```
   aws elasticache modify-serverless-cache --serverless-cache-name <serverless-cache>
    --remove-user-group
   ```

2. (선택 사항) 서버리스 캐시에 연결된 사용자 그룹이 없는지 확인합니다.

   ```
   aws elasticache describe-serverless-caches --serverless-cache-name <serverless-
   cache>
   {
       "..."
       "UserGroupId": ""
       "..."
   }
   ```

AUTH 토큰으로 구성된 를 사용하여 Valkey 또는 Redis OSS 캐시를 비활성화하려면

1. AUTH 토큰을 로 변경RBAC하고 추가할 사용자 그룹을 지정합니다.

```
aws elasticache modify-replication-group --replication-group-id <replication-group-id-value> --auth-token-update-strategy DELETE --user-group-ids-to-add <user-group-value>
```

2. AUTH 토큰이 비활성화되었고 사용자 그룹이 추가되었는지 확인합니다.

```
aws elasticache describe-replication-groups --replication-group-id <replication-group-id-value>
{
    "..."
    "AuthTokenEnabled": false,
    "UserGroupIds": [
        "<user-group-value>"
    ]
    "..."
}
```

3. 액세스 제어를 비활성화하려면 사용자 그룹을 제거합니다.

```
aws elasticache modify-replication-group --replication-group-id <replication-group-value> --user-group-ids-to-remove <user-group-value>
{
    "..."
    "PendingModifiedValues": {
    "UserGroups": {
      "UserGroupIdsToAdd": [],
      "UserGroupIdsToRemove": [
        "<user-group-value>"
      ]
    }
    "..."
}
```

4. (선택 사항) 클러스터에 연결된 사용자 그룹이 없는지 확인합니다. `AuthTokenEnabled` 필드도 거짓으로 표시되어야 합니다.

```
aws elasticache describe-replication-groups --replication-group-id <replication-group-value>
```

```
"AuthTokenEnabled": false
```

로 구성된 Valkey 또는 Redis OSS 클러스터를 비활성화하려면 RBAC

1. 액세스 제어를 비활성화하려면 사용자 그룹을 제거합니다.

```
aws elasticache modify-replication-group --replication-group-id <replication-group-
value> --user-group-ids-to-remove <user-group-value>
{
    "..."
    "PendingModifiedValues": {
    "UserGroups": {
      "UserGroupIdsToAdd": [],
      "UserGroupIdsToRemove": [
        "<user-group-value>"
      ]
    }
    "..."
}
```

2. (선택 사항) 클러스터에 연결된 사용자 그룹이 없는지 확인합니다. `AuthTokenEnabled` 필드도 거짓으로 표시되어야 합니다.

```
aws elasticache describe-replication-groups --replication-group-id <replication-
group-value>
"AuthTokenEnabled": false
```

인터네트워크 트래픽 개인 정보

Amazon ElastiCache 은 다음 기법을 사용하여 캐시 데이터를 보호하고 무단 액세스로부터 보호합니다.

- Amazon VPCs 및 ElastiCache 보안은 설치에 필요한 보안 그룹 유형을 설명합니다.
- 사용자, 그룹 및 역할의 작업을 부여하고 제한하기 위한 Amazon용 자격 증명 및 액세스 관리 ElastiCache

주제

- Amazon VPCs 및 ElastiCache 보안

- ElastiCache API 및 인터페이스 VPC 엔드포인트(AWS PrivateLink)
- 서브넷 및 서브넷 그룹

Amazon VPCs 및 ElastiCache 보안

데이터 보안이 중요하기 때문에 는 데이터에 액세스할 수 있는 사용자를 제어할 수 있는 수단을 ElastiCache 제공합니다. 데이터에 대한 액세스를 제어하는 방법은 Amazon Virtual Private Cloud(AmazonVPC)에서 클러스터를 시작했는지 아니면 Amazon EC2-Classic에서 시작했는지 여부에 따라 달라집니다.

> **Important**
>
> ElastiCache 클러스터를 시작하기 위해 Amazon EC2-Classic 사용을 중단했습니다. 모든 현재 세대 노드는 Amazon Virtual Private Cloud에서만 시작됩니다.

Amazon Virtual Private Cloud(Amazon VPC) 서비스는 기존 데이터 센터와 매우 유사한 가상 네트워크를 정의합니다. Amazon을 구성할 때 IP 주소 범위를 선택하고, 서브넷을 생성하고, 라우팅 테이블, 네트워크 게이트웨이 및 보안 설정을 구성할 VPC 수 있습니다. 가상 네트워크에 캐시 클러스터를 추가하고 Amazon VPC 보안 그룹을 사용하여 캐시 클러스터에 대한 액세스를 제어할 수도 있습니다.

이 섹션에서는 Amazon 에서 ElastiCache 클러스터를 수동으로 구성하는 방법을 설명합니다VPC. 이 정보는 ElastiCache 및 Amazon이 함께 VPC 작동하는 방법을 더 깊이 이해하고자 하는 사용자를 위한 것입니다.

주제

- ElastiCache 및 Amazon 이해 VPCs
- Amazon에서 ElastiCache 캐시에 액세스하기 위한 액세스 패턴 VPC
- Virtual Private Cloud 생성(VPC)
- Amazon에서 실행되는 캐시에 연결 VPC

ElastiCache 및 Amazon 이해 VPCs

ElastiCache 는 Amazon Virtual Private Cloud(Amazon)와 완전히 통합됩니다VPC. ElastiCache 사용자에게는 다음과 같은 의미가 있습니다.

- AWS 계정이 EC2-VPC 플랫폼만 지원하는 경우 는 ElastiCache 항상 Amazon 에서 클러스터를 시작합니다VPC.
- 를 처음 사용하는 경우 AWS클러스터가 Amazon 에 배포됩니다VPC. 기본값VPC이 자동으로 생성됩니다.
- 기본값이 VPC 있고 클러스터를 시작할 때 서브넷을 지정하지 않으면 클러스터가 기본 Amazon 로 시작됩니다VPC.

자세한 내용은 지원되는 플랫폼 감지 및 기본 가 있는지 여부를 참조하세요VPC.

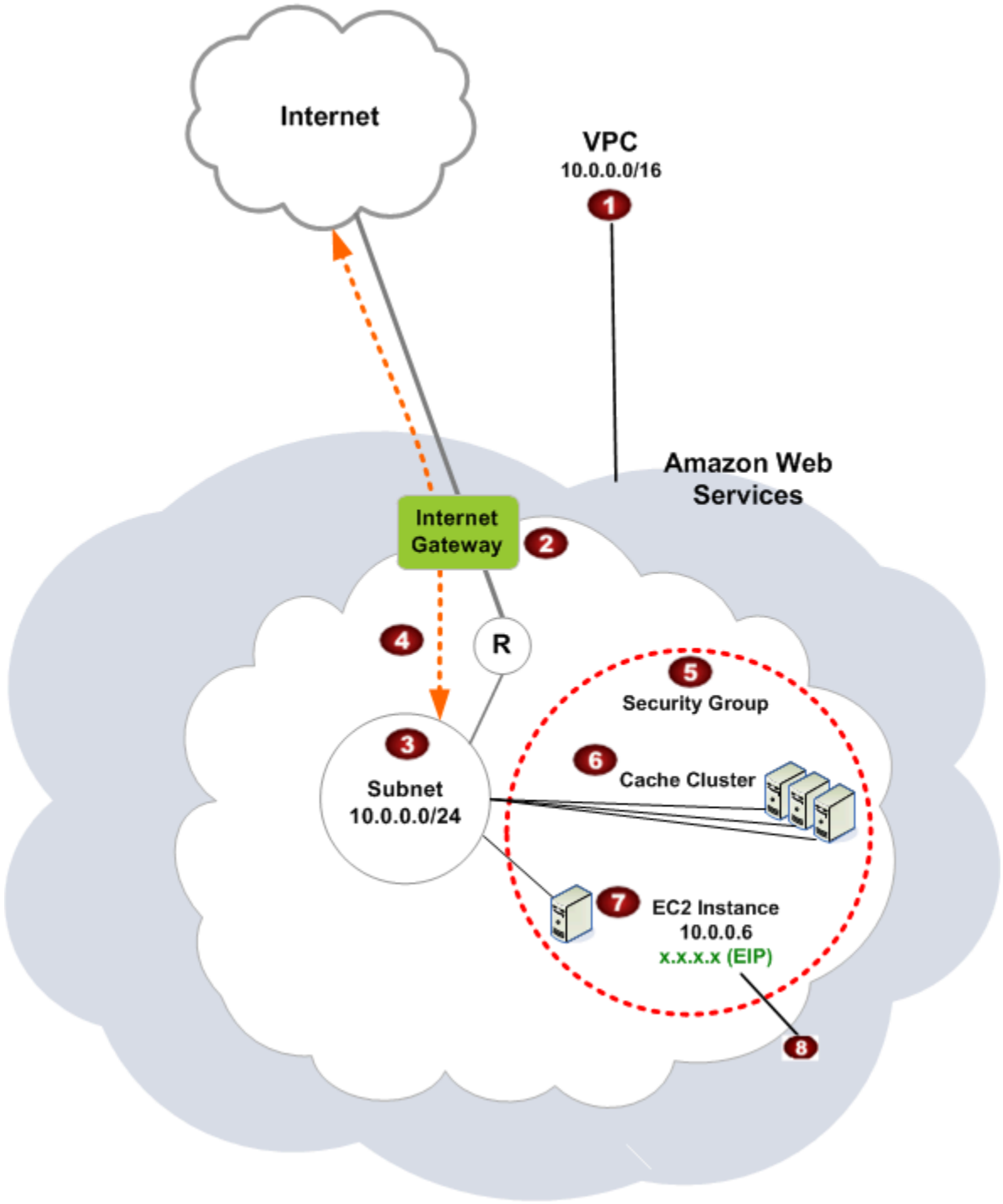
Amazon Virtual Private Cloud 를 사용하면 클라우드에서 기존 데이터 센터와 AWS 매우 유사한 가상 네트워크를 생성할 수 있습니다. IP 주소 범위 선택VPC, 서브넷 생성, 라우팅 테이블, 네트워크 게이트웨이 및 보안 설정 구성 등 Amazon 을 구성할 수 있습니다.

의 기본 기능은 가상 프라이빗 클라우드에서 ElastiCache 동일합니다. 클러스터가 Amazon 의 내부 또는 외부에 배포되었는지 여부에 관계없이 소프트웨어 업그레이드, 패치 적용, 장애 감지 및 복구를 ElastiCache 관리합니다VPC.

ElastiCache Amazon 외부에 배포된 캐시 노드VPC에는 엔드포인트/DNS이름이 확인되는 IP 주소가 할당됩니다. 이렇게 하면 Amazon Elastic Compute Cloud(AmazonEC2) 인스턴스와의 연결이 가능합니다. ElastiCache 클러스터를 Amazon VPC 프라이빗 서브넷으로 시작하면 모든 캐시 노드에 해당 서브넷 내의 프라이빗 IP 주소가 할당됩니다.

Amazon ElastiCache 의 개요 VPC

다음 다이어그램과 표는 Amazon 환경에서 시작된 ElastiCache 클러스터 및 Amazon EC2 인스턴스와 함께 Amazon VPC 환경을 설명합니다VPC.



1	Amazon은 자체 IP 주소 블록이 할당된 AWS 클라우드의 VPC 격리된 부분입니다.
2	

	인터넷 게이트웨이는 Amazon을 인터넷에 VPC 직접 연결하고 Amazon 외부에서 실행되는 Amazon Simple Storage Service(Amazon S3)와 같은 다른 AWS 리소스에 대한 액세스를 제공합니다VPC.
3	Amazon VPC 서브넷은 보안 및 운영 요구 사항에 따라 AWS 리소스를 격리할 수 VPC 있는 Amazon의 IP 주소 범위의 세그먼트입니다.
4	Amazon의 라우팅 테이블은 서브넷과 인터넷 간의 네트워크 트래픽을 VPC 지시합니다. Amazon에는 암시 라우터VPC가 있으며, 이 라우터는 R이 있는 원으로 이 다이어그램에 기호로 표시됩니다.
5	Amazon VPC 보안 그룹은 ElastiCache 클러스터 및 Amazon EC2 인스턴스의 인바운드 및 아웃바운드 트래픽을 제어합니다.
6	서브넷에서 ElastiCache 클러스터를 시작할 수 있습니다. 캐시 노드는 서브넷 주소 범위의 프라이빗 IP 주소를 가집니다.
7	서브넷에서 Amazon EC2 인스턴스를 시작할 수도 있습니다. 각 Amazon EC2 인스턴스에는 서브넷의 주소 범위에 속한 프라이빗 IP 주소가 있습니다. Amazon EC2 인스턴스는 동일한 서브넷의 모든 캐시 노드에 연결할 수 있습니다.
8	인터넷에서 Amazon EC2 인스턴스에 연결할 수 VPC 있으려면 탄력적 IP 주소라는 정적 퍼블릭 주소를 인스턴스에 할당해야 합니다.

사전 조건

Amazon 내에서 ElastiCache 클러스터를 생성하려면 VPCAmazon이 다음 요구 사항을 충족해야 VPC 합니다.

- Amazon은 전용이 아닌 Amazon EC2 인스턴스를 허용VPC해야 합니다. 전용 인스턴스 테넌시용으로 VPC 구성된 Amazon ElastiCache 에서는 를 사용할 수 없습니다.
- Amazon VPC. ElastiCache uses가 서브넷 그룹을 캐시하여 VPC 엔드포인트 또는 캐시 노드와 연결할 서브넷 및 해당 서브넷 내의 IP 주소를 선택하도록 하려면 캐시 서브넷 그룹을 정의해야 합니다.

- CIDR 각 서브넷의 블록은 유지 관리 활동 중에 ElastiCache 사용할 예비 IP 주소를 제공할 수 있을 만큼 커야 합니다.

라우팅 및 보안

Amazon에서 라우팅을 구성VPC하여 트래픽이 흐르는 위치(예: 인터넷 게이트웨이 또는 가상 프라이빗 게이트웨이)를 제어할 수 있습니다. 인터넷 게이트웨이를 사용하면 Amazon VPC 에서 실행되지 않는 다른 AWS 리소스에 직접 액세스할 수 있습니다VPC. 조직의 로컬 네트워크에 연결된 가상 프라이빗 게이트웨이만 사용하도록 선택한 경우 를 통해 인터넷 연결 트래픽을 라우팅VPN하고 로컬 보안 정책 및 방화벽을 사용하여 송신을 제어할 수 있습니다. 이 경우 인터넷을 통해 AWS 리소스에 액세스할 때 추가 대역폭 요금이 발생합니다.

Amazon VPC 보안 그룹을 사용하여 Amazon 에서 ElastiCache 클러스터와 Amazon EC2 인스턴스를 보호할 수 있습니다VPC. 보안 그룹은 서브넷 레벨이 아닌 인스턴스 레벨에서 방화벽처럼 작동합니다.

> **Note**
>
> 기본 IP 주소가 변경될 수 있으므로 DNS 이름을 사용하여 캐시 노드에 연결하는 것이 좋습니다.

Amazon VPC 설명서

AmazonVPC에는 Amazon 를 생성하고 사용하는 방법을 설명하는 자체 설명서 세트가 있습니다VPC. 다음 표에는 Amazon VPC 가이드에 대한 링크가 나와 있습니다.

설명	설명서
Amazon 사용을 시작하는 방법 VPC	Amazon 시작하기 VPC
를 VPC 통해 Amazon을 사용하는 방법 AWS Management Console	Amazon VPC 사용 설명서
모든 Amazon VPC 명령에 대한 전체 설명	Amazon EC2 명령줄 참조(Amazon VPC 명령은 Amazon EC2 참조에서 찾을 수 있음)
Amazon VPC API 작업, 데이터 유형 및 오류에 대한 전체 설명	Amazon EC2 API 참조(Amazon VPC API 작업은 Amazon EC2 참조에서 찾을 수 있음)

설명	설명서
선택적 IPsec VPN 연결 종료 시 게이트웨이를 구성해야 하는 네트워크 관리자를 위한 정보	란 무엇입니까 AWS Site-to-Site VPN?

Amazon Virtual Private Cloud에 대한 자세한 내용은 Amazon Virtual Private Cloud 섹션을 참조하세요.

Amazon에서 ElastiCache 캐시에 액세스하기 위한 액세스 패턴 VPC

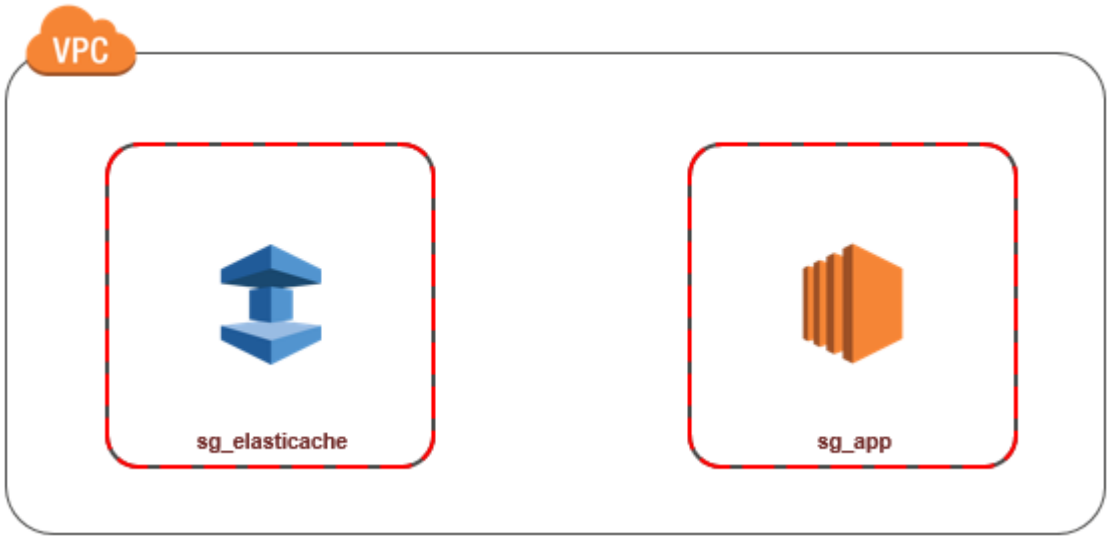
Amazon은 Amazon 의 캐시에 액세스하기 위해 다음 시나리오를 ElastiCache 지원합니다VPC.

목차

ElastiCache 캐시와 Amazon EC2 인스턴스가 동일한 Amazon에 있는 경우 캐시에 액세스 VPC

가장 일반적인 사용 사례는 EC2 인스턴스에 배포된 애플리케이션이 동일한 의 캐시에 연결해야 하는 경우입니다VPC.

다음은 이 시나리오를 설명하는 다이어그램입니다



동일한 의 EC2 인스턴스와 캐시 간의 액세스를 관리하는 가장 간단한 방법은 다음을 수행하는 VPC 것입니다.

1. 캐시에 대한 VPC 보안 그룹을 생성합니다. 이 보안 그룹을 사용해 캐시에 대한 액세스를 제한할 수 있습니다. 예를 들어 캐시를 생성할 때 캐시에 할당한 포트와 캐시TCP에 액세스하는 데 사용할 IP 주소를 사용하여 액세스를 허용하는 이 보안 그룹에 대한 사용자 지정 규칙을 생성할 수 있습니다.

   Memcached 캐시의 기본 포트는 11211입니다.

   Valkey 및 Redis OSS 캐시의 기본 포트는 입니다6379.
2. EC2 인스턴스(웹 및 애플리케이션 서버)에 대한 VPC 보안 그룹을 생성합니다. 이 보안 그룹은 필요한 경우 의 라우팅 테이블을 통해 인터넷에서 EC2 인스턴스에 대한 액세스를 허용할 수 VPC있습니다. 예를 들어 포트 22를 통해 EC2 인스턴스에 TCP 액세스할 수 있도록 이 보안 그룹에 규칙을 설정할 수 있습니다.
3. EC2 인스턴스에 대해 생성한 보안 그룹의 연결을 허용하는 사용자 지정 규칙을 캐시의 보안 그룹에 생성합니다. 그러면 보안 그룹의 모든 구성원이 캐시에 액세스할 수 있습니다.

> **Note**
>
> Local Zones를 사용할 계획이라면 활성화했는지 확인합니다. 해당 로컬 영역에서 서브넷 그룹을 생성하면 VPC가 해당 로컬 영역으로 확장되고 VPC는 서브넷을 다른 가용 영역의 서브넷으로 취급합니다. 모든 관련 게이트웨이 및 라우팅 테이블이 자동으로 조정됩니다.

다른 VPC 보안 그룹의 연결을 허용하는 규칙을 보안 그룹에 생성하려면

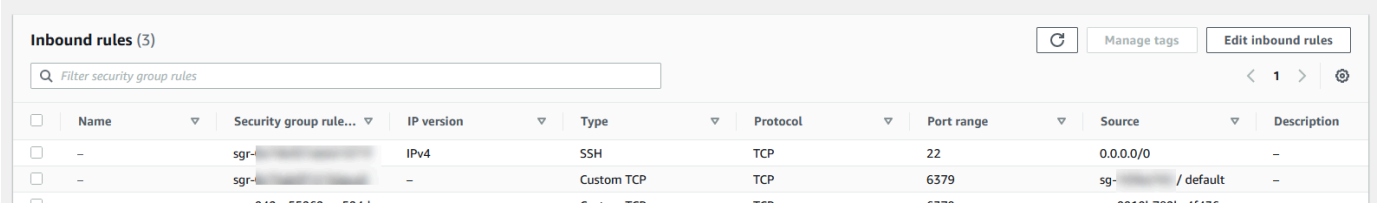
1. AWS 관리 콘솔에 로그인하고 https://console.aws.amazon.com/vpc 에서 Amazon VPC 콘솔을 엽니다.
2. 탐색 창에서 보안 그룹을 선택합니다.
3. 캐시에 사용할 보안 그룹을 선택하거나 만듭니다. [Inbound Rules]에서 [Edit Inbound Rules]를을 선택한 다음 [Add Rule][을 선택합니다. 이 보안 그룹은 다른 보안 그룹 멤버에 대한 액세스를 허용합니다.
4. 유형에서 사용자 지정 TCP 규칙 을 선택합니다.

   a. 포트 범위에서 캐시를 만들 때 사용한 포트를 지정합니다.

      Memcached 캐시의 기본 포트는 11211입니다.

      Valkey 및 Redis OSS 캐시 및 복제 그룹의 기본 포트는 입니다6379.

   b. [Source] 상자에 보안 그룹 ID를 입력합니다. 목록에서 Amazon EC2 인스턴스에 사용할 보안 그룹을 선택합니다.
5. 완료되면 [Save]를 선택합니다.



ElastiCache 캐시와 Amazon EC2 인스턴스가 다른 Amazon에 있을 때 캐시에 액세스 VPCs

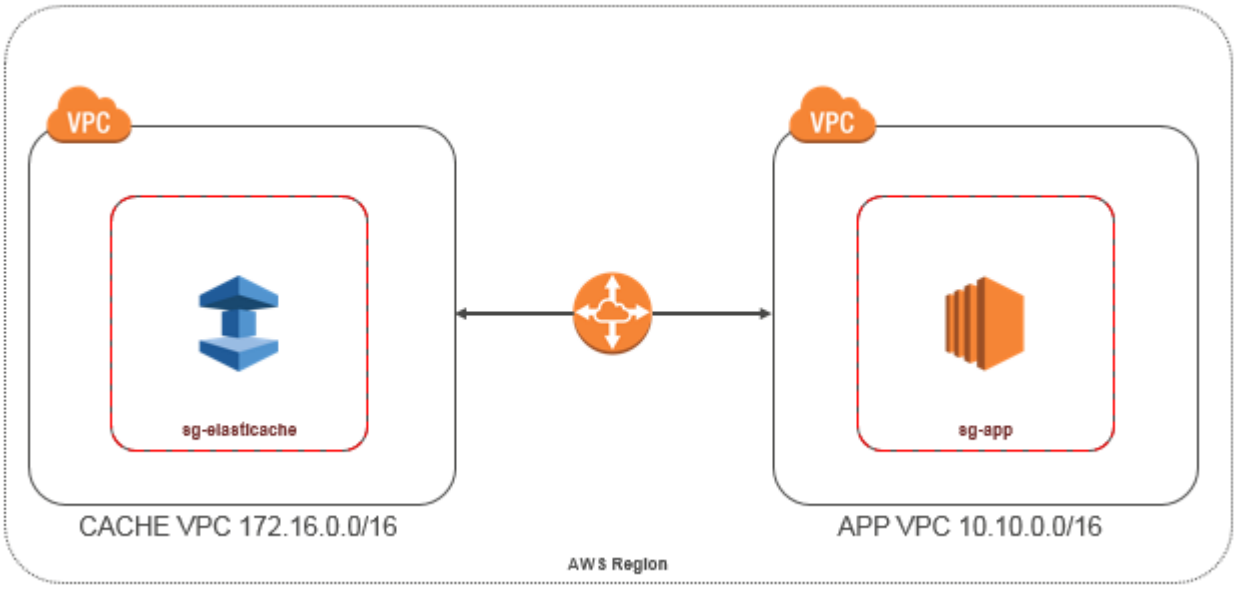
캐시가 액세스에 사용하는 VPC EC2 인스턴스와 다른 인스턴스에 있는 경우 캐시에 액세스하는 방법에는 여러 가지가 있습니다. 캐시와 EC2 인스턴스가 다르VPCs지만 동일한 리전에 있는 경우 VPC 피어링을 사용할 수 있습니다. 캐시와 EC2 인스턴스가 서로 다른 리전에 있는 경우 리전 간 VPN 연결을 생성할 수 있습니다.

주제

- ElastiCache 캐시와 Amazon EC2 인스턴스가 동일한 리전의 다른 Amazon에 있는 경우 캐시VPCs에 액세스
- ElastiCache 캐시와 Amazon EC2 인스턴스가 서로 다른 리전의 서로 다른 Amazon에 있을 때 캐시 VPCs에 액세스

ElastiCache 캐시와 Amazon EC2 인스턴스가 동일한 리전의 다른 Amazon에 있는 경우 캐시VPCs에 액세스

다음 다이어그램은 Amazon VPC 피어링 연결을 사용하여 동일한 리전의 다른 Amazon EC2 인스턴스 VPC에서 캐시에 액세스하는 방법을 보여줍니다.



동일한 리전 VPC 내 다른 Amazon의 Amazon EC2 인스턴스에서 액세스하는 캐시 - VPC 피어링 연결

VPC 피어링 연결은 프라이빗 IP 주소를 사용하여 둘 사이의 트래픽을 라우팅할 수 VPCs 있는 두 개 간의 네트워킹 연결입니다. 의 인스턴스는 동일한 네트워크 내에 있는 것처럼 서로 통신할 VPC 수 있습니다. 자체 Amazon 간에 VPCs피어링 연결을 생성하거나 단일 리전 내의 VPC 다른 AWS 계정에 Amazon과 VPC 피어링 연결을 생성할 수 있습니다. Amazon VPC 피어링에 대한 자세한 내용은 VPC 설명서를 참조하세요.

> **Note**
>
> DNS 에 적용된 구성에 VPCs따라 피어링된 에 대한 이름 확인이 실패할 수 있습니다 ElastiCache VPC. 이 문제를 해결하려면 DNS 호스트 이름과 DNS 해결에 대해 둘 다 활성화해야 VPCs 합니다. 자세한 내용은 VPC 피어링 연결에 대한 DNS 해상도 활성화를 참조하세요.

다른 Amazon VPC over 피어링의 캐시에 액세스하려면

1. 두 에 중복 IP 범위가 VPCs 없는지 확인합니다. 그렇지 않으면 피어링할 수 없습니다.

2. 두 를 피어링합니다VPCs. 자세한 내용은 Amazon VPC 피어링 연결 생성 및 수락을 참조하세요.
3. 라우팅 테이블을 업데이트합니다. 자세한 내용은 VPC 피어링 연결을 위한 라우팅 테이블 업데이트를 참조하세요.

   앞에 나온 다이어그램의 예제에 대한 라우팅 테이블은 다음과 같습니다. pcx-a894f1c1이 피어링 연결입니다.

Destination	Target
172.16.0.0/16	local
10.10.0.0/16	pcx-a894f1c1

Destination	Target
10.10.0.0/16	local
0.0.0.0/0	igw-bfdcccd8
172.16.0.0/16	pcx-a894f1c1

   VPC 라우팅 테이블

4. 피어링된 의 애플리케이션 보안 그룹에서 인바운드 연결을 허용하도록 ElastiCache 캐시의 보안 그룹을 수정합니다VPC. 자세한 내용은 참조 피어 VPC 보안 그룹 섹션을 참조하세요.

피어링 연결을 통해 캐시에 액세스하면 데이터 전송 비용이 추가로 발생합니다.

Transit Gateway 사용

전송 게이트웨이를 사용하면 동일한 AWS 리전에 VPCs 및 VPN 연결을 연결하고 이들 간에 트래픽을 라우팅할 수 있습니다. 전송 게이트웨이는 AWS 계정 간에 작동하며 AWS Resource Access Manager를 사용하여 전송 게이트웨이를 다른 계정과 공유할 수 있습니다. 전송 게이트웨이를 다른 AWS 계정과 공유한 후 계정 소유자는 해당 게이트웨이를 전송 게이트웨이VPCs에 연결할 수 있습니다. 두 계정의 사용자는 언제든지 연결을 삭제할 수 있습니다.

전송 게이트웨이에서 멀티캐스트를 활성화한 다음 도메인과 연결된 VPC 첨부 파일을 통해 멀티캐스트 원본에서 멀티캐스트 그룹 멤버로 멀티캐스트 트래픽을 전송할 수 있는 전송 게이트웨이 멀티캐스트 도메인을 생성할 수 있습니다.

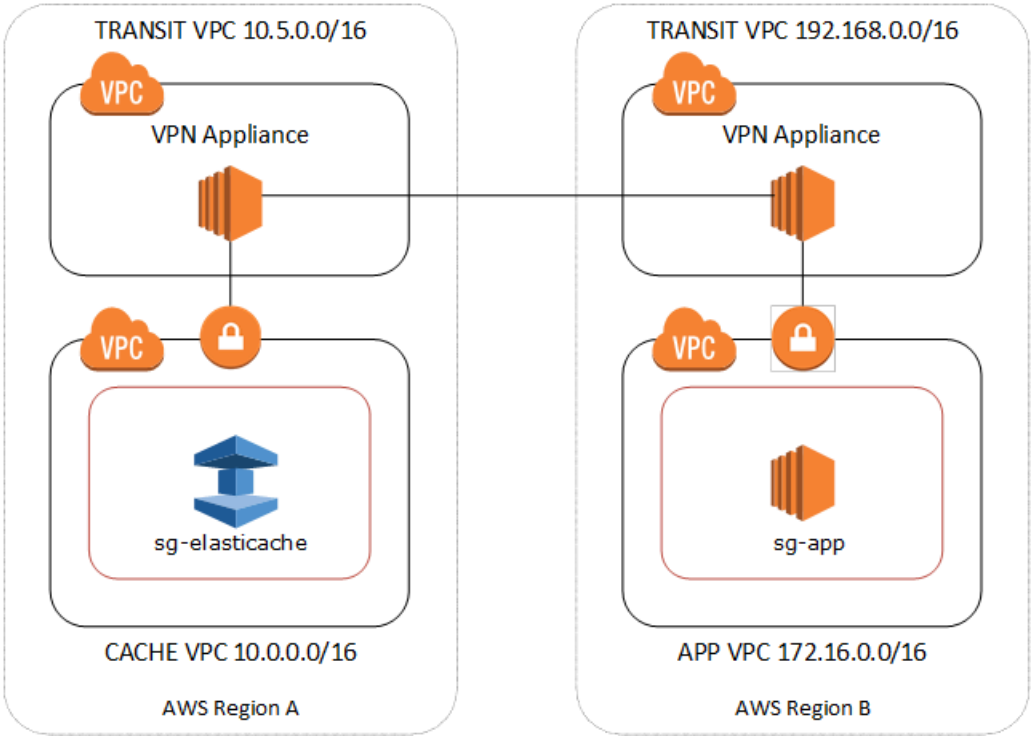
다른 AWS 리전의 전송 게이트웨이 간에 피어링 연결 연결을 생성할 수도 있습니다. 이렇게 하면 여러 리전의 Transit Gateway 연결 간에 트래픽을 라우팅할 수 있습니다.

자세한 내용은 전송 게이트웨이를 참조하세요.

ElastiCache 캐시와 Amazon EC2 인스턴스가 서로 다른 리전의 서로 다른 Amazon에 있을 때 캐시 VPCs에 액세스

전송 사용 VPC

VPC 피어링을 사용하는 대신 지리적으로 분산된 여러 네트워크VPCs와 원격 네트워크를 연결하는 또 다른 일반적인 전략은 글로벌 네트워크 전송 센터 역할을 VPC 하는 전송을 생성하는 것입니다. 전송은 네트워크 관리를 VPC 간소화하고 여러 VPCs 및 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.



서로 다른 리전VPCs에서 연결

Transit AmazonVPC이 설정되면 한 리전의 “스포크”VPC에 배포된 애플리케이션이 다른 리전VPC의 “스포크”에 있는 ElastiCache 캐시에 연결할 수 있습니다.

다른 AWS 리전 VPC 내의 다른 에서 캐시에 액세스하려면

1. Transit VPC 솔루션을 배포합니다. 자세한 내용은 AWS Transit Gateway를 참조하세요.
2. 앱 및 캐시의 VPC 라우팅 테이블을 업데이트VPCs하여 VGW (가상 프라이빗 게이트웨이) 및 VPN 어플라이언스를 통해 트래픽을 라우팅합니다. 경계 게이트웨이 프로토콜(BGP)을 사용한 동적 라우팅의 경우 경로가 자동으로 전파될 수 있습니다.

3. 애플리케이션 인스턴스 IP 범위에서 인바운드 연결을 허용하도록 ElastiCache 캐시의 보안 그룹을 수정합니다. 이 시나리오에는 애플리케이션 서버 보안 그룹을 참조할 수 없습니다.

여러 리전의 캐시에 액세스하면 네트워크 지연 시간이 생기고 리전 간 데이터 전송 비용이 추가로 발생합니다.

고객의 데이터 센터에서 실행되는 애플리케이션에서 ElastiCache 캐시에 액세스

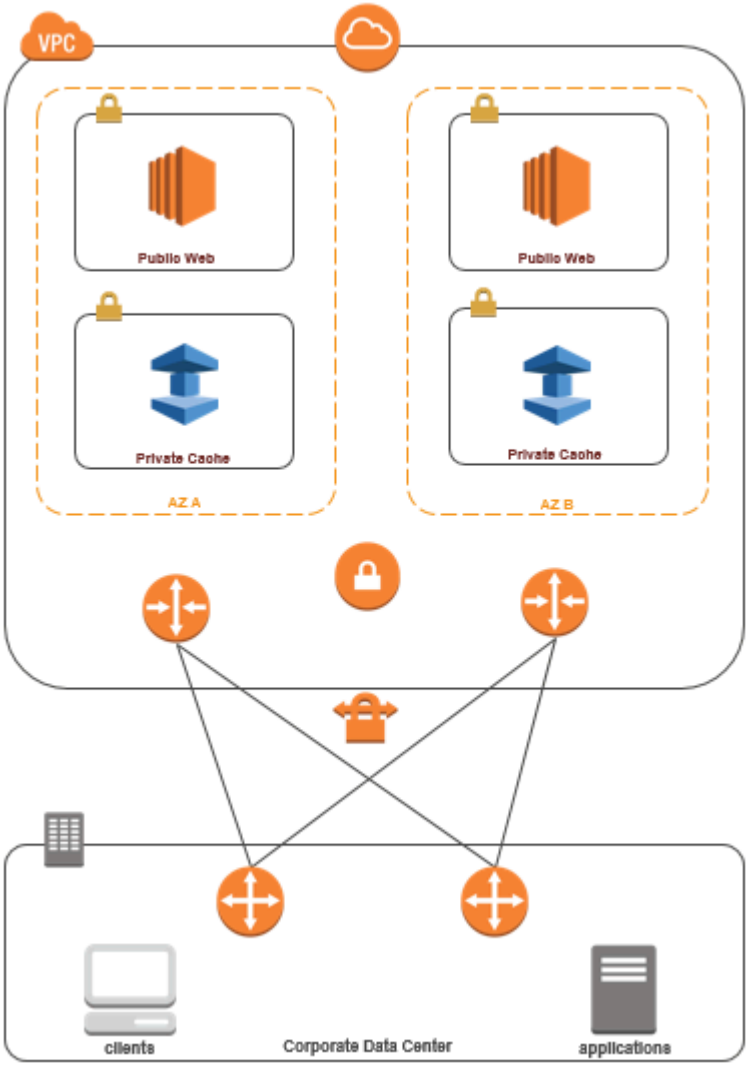
또 다른 가능한 시나리오는 고객 데이터 센터의 클라이언트 또는 애플리케이션이 의 ElastiCache 캐시에 액세스해야 할 수 있는 하이브리드 아키텍처입니다VPC. 이 시나리오는 VPN 또는 Direct Connect를 통해 고객과 VPC 데이터 센터 간에 연결이 있는 경우에도 지원됩니다.

주제

- VPN 연결을 사용하여 고객의 데이터 센터에서 실행되는 애플리케이션에서 ElastiCache 캐시에 액세스
- Direct Connect를 사용하여 고객의 데이터 센터에서 실행되는 애플리케이션에서 ElastiCache 캐시에 액세스

VPN 연결을 사용하여 고객의 데이터 센터에서 실행되는 애플리케이션에서 ElastiCache 캐시에 액세스

다음 다이어그램은 VPN 연결을 사용하여 회사 네트워크에서 실행되는 애플리케이션에서 ElastiCache 캐시에 액세스하는 방법을 보여줍니다.



를 통해 ElastiCache 데이터 센터에서 에 연결 VPN

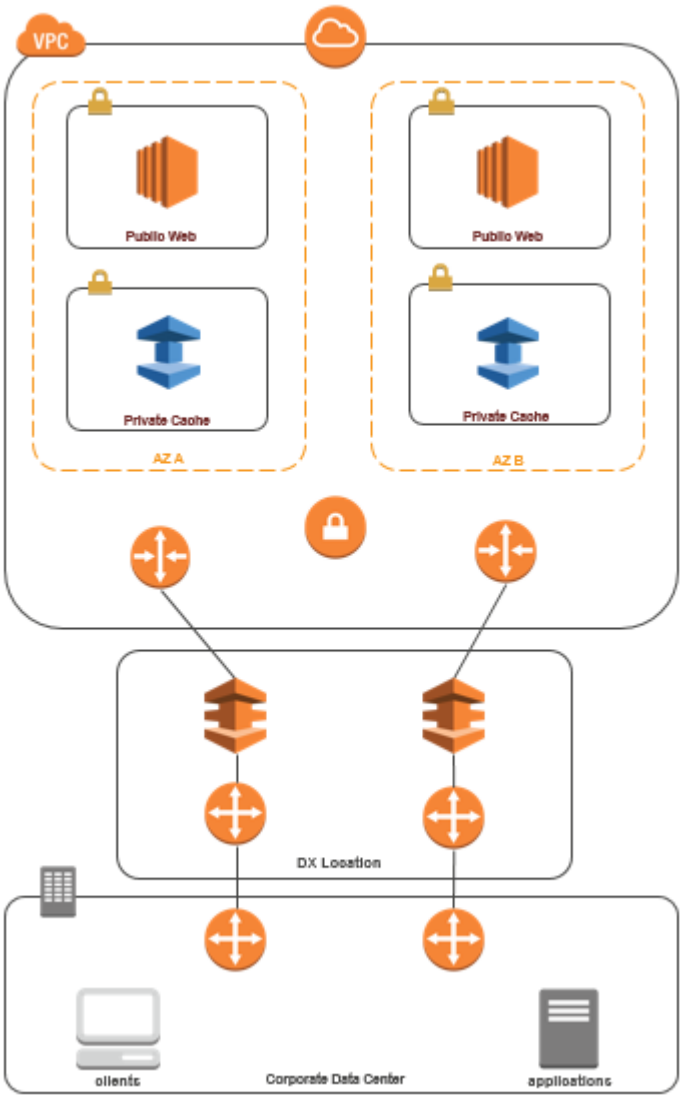
VPN 연결을 통해 VPC 온프레미스 애플리케이션에서 의 캐시에 액세스하려면

1. 하드웨어 Virtual Private Gateway를 에 추가하여 VPN 연결을 설정합니다VPC. 자세한 내용은 에 하드웨어 가상 프라이빗 게이트웨이 추가를 참조하세요VPC.
2. 온프레미스 애플리케이션 서버의 트래픽을 허용하도록 ElastiCache 캐시가 배포되는 서브넷의 VPC 라우팅 테이블을 업데이트합니다. 경로가 Dynamic Routing인 경우 경로가 자동으로 전파될 수 BGP 있습니다.
3. 온프레미스 애플리케이션 서버에서 인바운드 연결을 허용하도록 ElastiCache 캐시의 보안 그룹을 수정합니다.

VPN 연결을 통해 캐시에 액세스하면 네트워킹 지연 시간과 추가 데이터 전송 비용이 발생합니다.

Direct Connect를 사용하여 고객의 데이터 센터에서 실행되는 애플리케이션에서 ElastiCache 캐시에 액세스

다음 다이어그램은 Direct Connect를 사용하여 회사 네트워크에서 실행되는 애플리케이션에서 ElastiCache 캐시에 액세스하는 방법을 보여줍니다.



Direct Connect를 통해 데이터 센터에서 ElastiCache 에 연결

Direct Connect를 사용하여 네트워크에서 실행 중인 애플리케이션에서 ElastiCache 캐시에 액세스하려면

1. Direct Connect 연결을 설정합니다. 자세한 내용은 AWS Direct Connect 시작하기 섹션을 참조하세요.
2. 온프레미스 애플리케이션 서버에서 인바운드 연결을 허용하도록 ElastiCache 캐시의 보안 그룹을 수정합니다.

DX 연결을 통해 캐시에 액세스하면 네트워크 지연 시간이 생기고 데이터 전송 요금이 추가로 발생할 수 있습니다.

Virtual Private Cloud 생성(VPC)

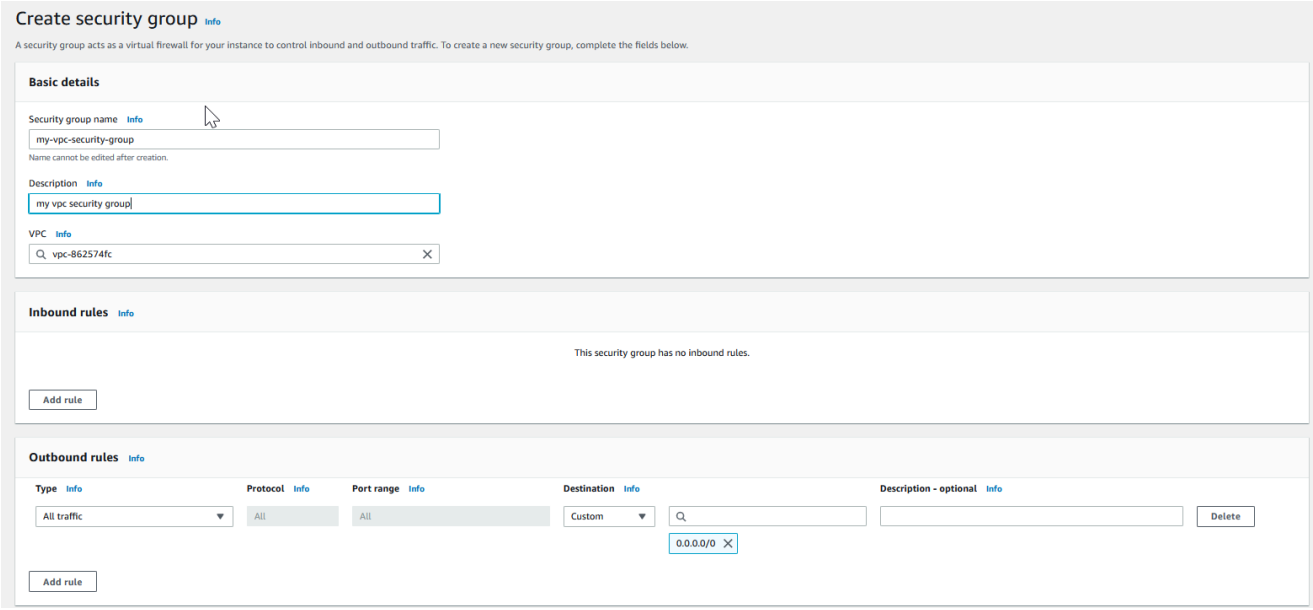
이 예제에서는 각 가용 영역에 대해 프라이빗 서브넷이 VPC 있는 Amazon을 생성합니다.

Amazon 생성VPC(콘솔)

1. AWS Management Console에 로그인하고 에서 Amazon VPC 콘솔을 엽니다https://console.aws.amazon.com/vpc/.
2. VPC 대시보드에서 생성을 VPC선택합니다.
3. 생성할 리소스에서 VPC 및 기타 를 선택합니다.
4. 가용 영역 수(AZs)에서 서브넷을 시작할 가용 영역 수를 선택합니다.
5. 퍼블릭 서브넷 수 에서 에 추가할 퍼블릭 서브넷 수를 선택합니다VPC.
6. 프라이빗 서브넷 수 에서 에 추가할 프라이빗 서브넷 수를 선택합니다VPC.

> **Tip**
> 서브넷 식별자(퍼블릭 서브넷, 프라이빗 서브넷)를 기록해 둡니다. 클러스터를 시작하고 Amazon EC2 인스턴스를 Amazon 에 추가할 때 나중에 이 정보가 필요합니다VPC.

7. Amazon VPC 보안 그룹을 생성합니다. 캐시 클러스터와 Amazon EC2 인스턴스에 이 그룹을 사용합니다.
   a. Amazon VPC Management 콘솔의 탐색 창에서 보안 그룹 을 선택합니다.
   b. 보안 그룹 생성을 선택합니다.
   c. 보안 그룹의 이름과 설명을 해당 상자에 입력합니다. VPC 상자에서 Amazon 의 식별자를 선택합니다VPC.



d. 원하는 대로 설정되었으면 [Yes, Create]를 선택합니다.

8. 보안 그룹의 네트워크 수신 규칙을 정의합니다. 이 규칙을 사용하면 Secure Shell()을 사용하여 Amazon EC2 인스턴스에 연결할 수 있습니다SSH.

   a. 탐색 목록에서 [Security Groups]를 선택합니다.

   b. 목록에서 보안 그룹을 찾아 선택합니다.

   c. [Security Group] 아래에서 [Inbound] 탭을 선택합니다. 새 규칙 생성 상자에서 SSH를 선택한 다음 규칙 추가 를 선택합니다.

   d. HTTP 액세스를 허용하려면 새 인바운드 규칙에 대해 다음 값을 설정합니다.

      - 유형: HTTP
      - 소스: 0.0.0.0/0

      [Apply Rule Changes]를 선택합니다.

이제 Amazon 에서 캐시 서브넷 그룹을 생성하고 캐시 클러스터를 시작할 준비가 되었습니다VPC.

- 서브넷 그룹 생성
- Memcached 클러스터 생성(콘솔).
- Valkey(클러스터 모드 비활성화됨) 클러스터 생성(콘솔).

Amazon에서 실행되는 캐시에 연결 VPC

이 예제는 Amazon 에서 Amazon EC2 인스턴스를 시작하는 방법을 보여줍니다VPC. 그런 다음 이 인스턴스에 로그인하고 Amazon 에서 실행 중인 ElastiCache 캐시에 액세스할 수 있습니다VPC.

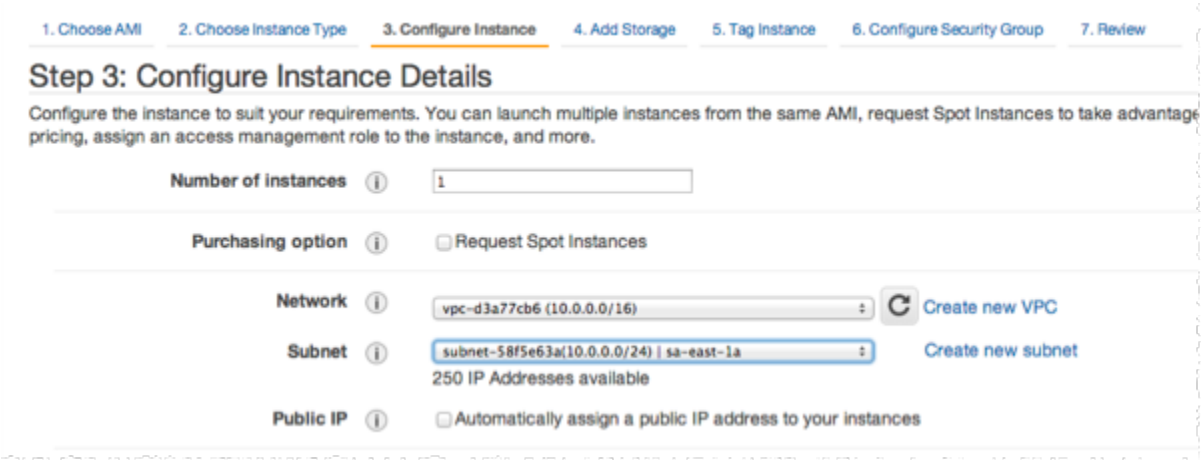
Amazon에서 실행되는 캐시에 연결VPC(콘솔)

이 예제에서는 Amazon 에서 Amazon EC2 인스턴스를 생성합니다VPC. 이 Amazon EC2 인스턴스를 사용하여 Amazon 에서 실행되는 캐시 노드에 연결할 수 있습니다VPC.

> **Note**
>
> Amazon 사용에 대한 자세한 내용은 Amazon EC2 설명서의 Amazon EC2 시작 안내서를 EC2 참조하세요.

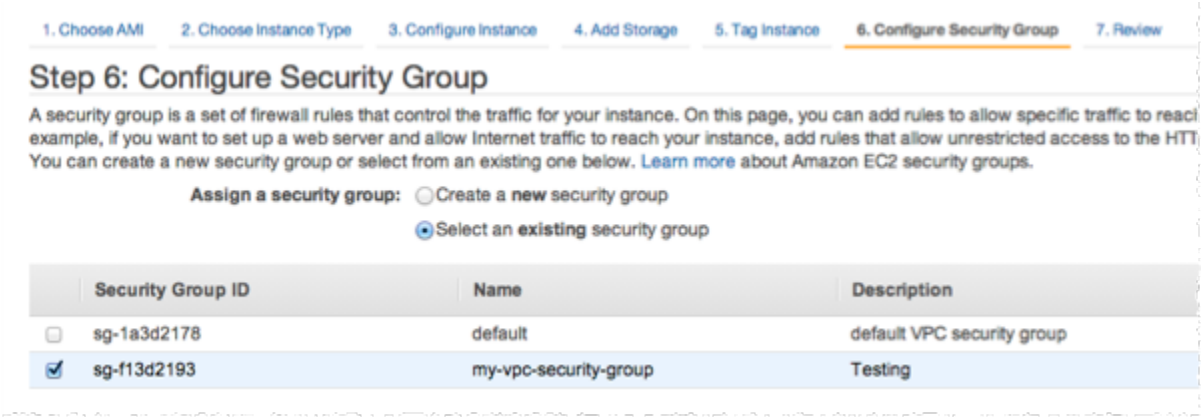
Amazon EC2 콘솔을 VPC 사용하여 Amazon에서 Amazon EC2 인스턴스를 생성하려면

1. 에 로그인 AWS Management Console 하고 에서 Amazon EC2 콘솔을 엽니다https://console.aws.amazon.com/ec2/.
2. 콘솔에서 [Launch Instance]를 선택하고 다음 단계를 따릅니다.
3. Amazon Machine Image 선택(AMI) 페이지에서 64비트 Amazon Linux 를 선택한 AMI다음 선택을 선택합니다.
4. 인스턴스 유형 선택 페이지에서 3. 인스턴스 구성을 선택합니다.
5. [Configure Instance Details] 페이지에서 다음과 같이 선택합니다.
   a. 네트워크 목록에서 Amazon 를 선택합니다VPC.
   b. [Subnet] 목록에서 퍼블릭 서브넷을 선택합니다.



원하는 대로 설정되었으면 4. 스토리지 추가를 선택합니다.

6. 스토리지 추가 페이지에서 5. 인스턴스 태그 지정을 선택합니다.
7. 태그 인스턴스 페이지에서 Amazon EC2 인스턴스의 이름을 입력한 다음 6을 선택합니다. 보안 그룹 구성을 선택합니다.
8. [Configure Security Group] 페이지에서 [Select an existing security group]을 선택합니다. 보안 그룹에 대한 자세한 내용은 Linux 인스턴스용 Amazon EC2 보안 그룹 섹션을 참조하세요.



Amazon VPC 보안 그룹의 이름을 선택한 다음 검토 및 시작을 선택합니다.

9. [Review Instance and Launch] 페이지에서 [Launch]를 선택합니다.
10. 기존 키 페어 선택 또는 새 키 페어 생성 창에서 이 인스턴스에서 사용할 키 페어를 지정합니다.

> **Note**
>
> 키 페어 관리에 대한 자세한 내용은 Amazon EC2 시작하기 안내서를 참조하세요.

11. Amazon EC2 인스턴스를 시작할 준비가 되면 시작을 선택합니다.

이제 방금 생성한 Amazon EC2 인스턴스에 탄력적 IP 주소를 할당할 수 있습니다. Amazon EC2 인스턴스에 연결하려면 이 IP 주소를 사용해야 합니다.

탄력적 IP 주소를 할당하려면(콘솔)

1. 에서 Amazon VPC 콘솔을 엽니다https://console.aws.amazon.com/vpc/.
2. 탐색 목록에서 탄력적 IPs을 선택합니다.
3. 탄력적 IP 주소 할당를 선택합니다.

4. [Allocate Elastic IP address] 대화 상자에서 기본 [Network Border Group]을 적용하고 [Allocate]를 선택합니다.
5. 목록에서 방금 할당한 탄력적 IP 주소를 선택하고 [Associate Address]를 선택합니다.
6. 주소 연결 대화 상자의 인스턴스 상자에서 시작한 Amazon EC2 인스턴스의 ID를 선택합니다.

   [Private IP address] 상자에서, 프라이빗 IP 주소를 가져올 상자를 선택한 다음 [Associate]를 선택합니다.

   이제 SSH를 사용하여 생성한 탄력적 IP 주소를 사용하여 Amazon EC2 인스턴스에 연결할 수 있습니다.

Amazon EC2 인스턴스에 연결하려면

- 명령 창을 엽니다. 명령 프롬프트에서 mykeypair.pem을 키 페어 파일 이름으로 바꾸고 54.207.55.251을 탄력적 IP 주소로 바꿔 다음 명령을 실행합니다.

```
ssh -i mykeypair.pem ec2-user@54.207.55.251
```

> **Important**
>
> 아직 Amazon EC2 인스턴스에서 로그아웃하지 마세요.

이제 ElastiCache 클러스터와 상호 작용할 준비가 되었습니다. Telnet 유틸리티를 설치하지 않았다면 설치한 후에 실행할 수 있습니다.

Telnet 설치 및 캐셔 클러스터(AWS CLI)와 상호 작용

- 명령 창을 엽니다. 명령 프롬프트에서 다음 명령을 실행합니다. 확인 프롬프트에 y를 입력합니다.

```
sudo yum install telnet
Loaded plugins: priorities, security, update-motd, upgrade-helper
Setting up Install Process
Resolving Dependencies
--> Running transaction check

...(output omitted)...
```

```
Total download size: 63 k
Installed size: 109 k
Is this ok [y/N]: y
Downloading Packages:
telnet-0.17-47.7.amzn1.x86_64.rpm                          |  63 kB     00:00

...(output omitted)...

Complete!
```

이제 Memcached 또는 RedisVPC를 사용하여 에 연결할 수 있습니다.

MemcachedVPC를 사용하여 에 연결

1. 에서 ElastiCache 콘솔로 이동하여 캐시 클러스터의 노드 중 하나에 대한 엔드포인트를 https://console.aws.amazon.com/elasticache/ 가져옵니다. 자세한 내용은 연결 엔드포인트 찾기를 참조하세요.
2. telnet을 사용하여 포트 11211을 통해 캐시 노드 엔드포인트에 연결합니다. 아래에 표시된 호스트 이름을 캐시 노드의 호스트 이름으로 바꿉니다.

   ```
   telnet my-cache-cluster.7wufxa.0001.use1.cache.amazonaws.com 11211
   ```

   이제 캐시 엔진에 연결되어 명령을 실행할 수 있습니다. 이 예에서는 캐시에 데이터 항목을 추가한 다음 즉시 가져옵니다. 마지막으로 캐시 노드에서 연결을 끊습니다.

   키 및 값을 저장하기 위해 다음 두 줄을 입력합니다.

   ```
   add mykey 0 3600 28
   This is the value for mykey
   ```

   캐시 엔진은 다음과 같이 응답합니다.

   ```
   OK
   ```

   mykey에 대한 값을 검색하려면 다음을 입력합니다.

   ```
   get mykey
   ```

캐시 엔진은 다음과 같이 응답합니다.

```
VALUE mykey 0 28
This is the value for my key
END
```

캐시 엔진에서 연결을 끊으려면 다음을 입력합니다.

```
quit
```

RedisVPC를 사용하여 에 연결

1. 에서 ElastiCache 콘솔로 이동하여 캐시 클러스터의 노드 중 하나에 대한 엔드포인트를 https://console.aws.amazon.com/elasticache/ 가져옵니다. 자세한 내용은 Redis용 연결 엔드포인트 찾기를 참조하세요.
2. telnet을 사용하여 포트 6379를 통해 캐시 노드 엔드포인트에 연결합니다. 아래에 표시된 호스트 이름을 캐시 노드의 호스트 이름으로 바꿉니다.

```
telnet my-cache-cluster.7wufxa.0001.use1.cache.amazonaws.com 6379
```

이제 캐시 엔진에 연결되어 명령을 실행할 수 있습니다. 이 예에서는 캐시에 데이터 항목을 추가한 다음 즉시 가져옵니다. 마지막으로 캐시 노드에서 연결을 끊습니다.

키 및 값을 저장하기 위해 다음 두 줄을 입력합니다.

```
set mykey myvalue
```

캐시 엔진은 다음과 같이 응답합니다.

```
OK
```

mykey에 대한 값을 검색하려면 다음을 입력합니다.

```
get mykey
```

캐시 엔진에서 연결을 끊으려면 다음을 입력합니다.

```
quit
```

3. 에서 ElastiCache 콘솔로 이동하여 캐시 클러스터의 노드 중 하나에 대한 엔드포인트를 https://console.aws.amazon.com/elasticache/ 가져옵니다. 자세한 내용은 Redis용 연결 엔드포인트 찾기를 참조하세요OSS.
4. telnet을 사용하여 포트 6379를 통해 캐시 노드 엔드포인트에 연결합니다. 아래에 표시된 호스트 이름을 캐시 노드의 호스트 이름으로 바꿉니다.

```
telnet my-cache-cluster.7wufxa.0001.use1.cache.amazonaws.com 6379
```

이제 캐시 엔진에 연결되어 명령을 실행할 수 있습니다. 이 예에서는 캐시에 데이터 항목을 추가한 다음 즉시 가져옵니다. 마지막으로 캐시 노드에서 연결을 끊습니다.

키 및 값을 저장하기 위해 다음을 입력합니다.

```
set mykey myvalue
```

캐시 엔진은 다음과 같이 응답합니다.

```
OK
```

mykey에 대한 값을 검색하려면 다음을 입력합니다.

```
get mykey
```

캐시 엔진은 다음과 같이 응답합니다.

```
get mykey
myvalue
```

캐시 엔진에서 연결을 끊으려면 다음을 입력합니다.

```
quit
```

> **Important**
>
> AWS 계정에 추가 요금이 발생하지 않도록 하려면 다음 예제를 시도한 후 더 이상 원하지 않는 리소스를 삭제 AWS 해야 합니다.

ElastiCache API 및 인터페이스 VPC 엔드포인트(AWS PrivateLink)

인터페이스 엔드포인트 를 생성하여 VPC 와 Amazon ElastiCache API 엔드포인트 간에 프라이빗 연결을 설정할 수 있습니다. VPC 인터페이스 엔드포인트는 로 구동됩니다AWS PrivateLink. AWS PrivateLink 를 사용하면 인터넷 게이트웨이, NAT 디바이스, VPN 연결 또는 AWS Direct Connect 연결 없이 Amazon ElastiCache API 작업에 비공개로 액세스할 수 있습니다.

의 인스턴스는 Amazon ElastiCache API 엔드포인트와 통신하는 데 퍼블릭 IP 주소가 필요하지 VPC 않습니다. 또한 사용 가능한 ElastiCache API 작업을 사용하는 데 퍼블릭 IP 주소가 필요하지 않습니다. VPC 와 Amazon 간의 트래픽 ElastiCache 은 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 내용은 Amazon VPC 사용 설명서의 인터페이스 VPC 엔드포인트(AWS PrivateLink)를 참조하세요.
- 작업에 대한 ElastiCache API 자세한 내용은 ElastiCache API 작업을 참조하세요.

인터페이스 VPC 엔드포인트를 생성한 후 엔드포인트에 프라이빗 DNS 호스트 이름을 활성화하면 기본 ElastiCache 엔드포인트(https://elasticache.*Region*.amazonaws.com)가 VPC 엔드포인트로 확인됩니다. 프라이빗 DNS 호스트 이름을 활성화하지 않으면 Amazon은 다음 형식으로 사용할 수 있는 DNS 엔드포인트 이름을 VPC 제공합니다.

```
VPC_Endpoint_ID.elasticache.Region.vpce.amazonaws.com
```

자세한 내용은 Amazon 사용 설명서의 인터페이스 VPC 엔드포인트(AWS PrivateLink)를 참조하세요. 는 내 모든 API 작업에 대한 호출을 ElastiCache 지원합니다VPC. VPC

> **Note**
>
> 프라이빗 DNS 호스트 이름은 의 VPC 엔드포인트 하나에 대해서만 활성화할 수 있습니다 VPC. 추가 VPC 엔드포인트를 생성하려면 프라이빗 DNS 호스트 이름을 비활성화해야 합니다.

VPC 엔드포인트에 대한 고려 사항

Amazon VPC ElastiCache API 엔드포인트에 대한 인터페이스 엔드포인트를 설정하기 전에 Amazon VPC 사용 설명서의 인터페이스 엔드포인트 속성 및 제한을 검토해야 합니다. Amazon ElastiCache 리소스 관리와 관련된 모든 ElastiCache API 작업은 를 VPC 사용하여 에서 사용할 수 있습니다 AWS PrivateLink.

VPC 엔드포인트 정책은 엔드포인트에 ElastiCache API 대해 지원됩니다. 기본적으로 엔드포인트를 ElastiCache API 통해 작업에 대한 전체 액세스가 허용됩니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC 엔드포인트를 사용하여 서비스에 대한 액세스 제어를 참조하세요.

에 대한 인터페이스 VPC 엔드포인트 생성 ElastiCache API

Amazon ElastiCache API VPC 콘솔 또는 를 사용하여 Amazon용 VPC 엔드포인트를 생성할 수 있습니다 AWS CLI. 자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트 생성을 참조하세요.

인터페이스 VPC 엔드포인트를 생성한 후 엔드포인트에 대한 프라이빗 DNS 호스트 이름을 활성화할 수 있습니다. 이 경우 기본 Amazon ElastiCache 엔드포인트(https://elasticache.*Region*.amazonaws.com)가 VPC 엔드포인트로 확인됩니다. 중국(베이징) 및 중국(닝샤) AWS 리전의 경우 베이징용 와 닝샤`elasticache.cn-north-1.amazonaws.com.cn`용 를 사용하여 VPC 엔드포인트`elasticache.cn-northwest-1.amazonaws.com.cn`로 API 요청할 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트를 통한 서비스 액세스를 참조하세요.

Amazon에 대한 VPC 엔드포인트 정책 생성 ElastiCache API

에 대한 액세스를 제어하는 엔드포인트에 VPC 엔드포인트 정책을 연결할 수 있습니다 ElastiCache API. 이 정책은 다음을 지정합니다.

- 작업을 수행할 수 있는 보안 주체.
- 수행할 수 있는 작업.
- 작업을 수행할 수 있는 리소스.

자세한 내용은 Amazon VPC 사용 설명서의 VPC 엔드포인트를 사용하여 서비스에 대한 액세스 제어를 참조하세요.

Example VPC Valkey 또는 Redis를 사용한 작업에 대한 ElastiCache API 엔드포인트 정책 OSS

다음은 에 대한 엔드포인트 정책의 예입니다 ElastiCache API. 엔드포인트에 연결되면 이 정책은 모든 리소스의 모든 보안 주체에 대해 나열된 ElastiCache API 작업에 대한 액세스 권한을 부여합니다.

```
{
 "Statement": [{
  "Principal": "*",
  "Effect": "Allow",
  "Action": [
   "elasticache:CreateCacheCluster",
   "elasticache:ModifyCacheCluster",
   "elasticache:CreateSnapshot"
  ],
  "Resource": "*"
 }]
}
```

Example VPC ElastiCache (Memcached) API 작업에 대한 엔드포인트 정책

다음은 에 대한 엔드포인트 정책의 예입니다 ElastiCache API. 엔드포인트에 연결되면 이 정책은 모든 리소스의 모든 보안 주체에 대해 나열된 ElastiCache API 작업에 대한 액세스 권한을 부여합니다.

```
{
 "Statement": [{
  "Principal": "*",
  "Effect": "Allow",
  "Action": [
   "elasticache:CreateCacheCluster",
   "elasticache:ModifyCacheCluster"
  ],
  "Resource": "*"
 }]
}
```

Example VPC 지정된 AWS 계정의 모든 액세스를 거부하는 엔드포인트 정책

다음 VPC 엔드포인트 정책이 AWS 계정을 거부합니다.*123456789012* 엔드포인트를 사용하여 리소스에 대한 모든 액세스. 이 정책은 다른 계정의 모든 작업을 허용합니다.

```
{
 "Statement": [{
   "Action": "*",
   "Effect": "Allow",
   "Resource": "*",
   "Principal": "*"
  },
  {
   "Action": "*",
   "Effect": "Deny",
   "Resource": "*",
   "Principal": {
    "AWS": [
     "123456789012"
    ]
   }
  }
 ]
}
```

서브넷 및 서브넷 그룹

서브넷 그룹은 Amazon Virtual Private Cloud() 환경에서 실행되는 자체 설계된 클러스터에 대해 지정할 수 있는 서브넷(일반적으로 프라이빗VPC) 모음입니다.

Amazon 에서 자체 설계된 클러스터를 생성하는 경우 서브넷 그룹을 사용해야 VPC합니다. ElastiCache 는 해당 서브넷 그룹을 사용하여 해당 서브넷 내에서 노드와 연결할 서브넷 및 IP 주소를 선택합니다.

ElastiCache 는 기본 서브넷 그룹을 제공하거나 새 IPv4 서브넷 그룹을 생성하도록 선택할 수 있습니다. 의 경우 IPv6 CIDR 블록을 사용하여 서브넷 그룹을 생성IPv6해야 합니다. 듀얼 스택 을 선택한 경우 IPv6 또는 중에서 검색 IP 유형을 선택해야 합니다IPv4.

ElastiCache Serverless는 서브넷 그룹 리소스를 사용하지 않고 생성 중에 직접 서브넷 목록을 가져옵니다.

이 섹션에서는 서브넷 및 서브넷 그룹을 생성하고 활용하여 ElastiCache 리소스에 대한 액세스를 관리하는 방법을 다룹니다.

Amazon VPC 환경에서 서브넷 그룹 사용에 대한 자세한 내용은 섹션을 참조하세요 ElastiCache 클러스터 또는 복제 그룹에 액세스.

주제

- 서브넷 그룹 생성
- 캐시에 서브넷 그룹 할당
- 서브넷 그룹 수정
- 서브넷 그룹 삭제

서브넷 그룹 생성

캐시 서브넷 그룹은 에서 캐시에 대해 지정할 수 있는 서브넷 모음입니다VPC. 에서 캐시를 시작할 때 캐시 서브넷 그룹을 선택해야 VPC합니다. 그런 다음 해당 캐시 서브넷 그룹을 ElastiCache 사용하여 해당 서브넷 내의 IP 주소를 캐시의 각 캐시 노드에 할당합니다.

새 서브넷 그룹을 생성할 때 사용 가능한 IP 주소의 수를 기록하세요. 서브넷에 무료 IP 주소가 거의 없는 경우 클러스터에 추가할 수 있는 노드의 수에 제약이 있을 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.

네트워크 유형IPV4으로 를 선택하면 기본 서브넷 그룹을 사용할 수 있거나 새 서브넷 그룹을 생성하도록 선택할 수 있습니다. 는 해당 서브넷 그룹을 ElastiCache 사용하여 해당 서브넷 내에서 서브넷 및 IP 주소를 선택하여 노드와 연결할 수 있습니다. 듀얼 스택 또는 를 선택하면 듀얼 스택 또는 IPV6 서브넷을 생성하라는 메시지가 IPV6표시됩니다. 네트워크 유형에 대한 자세한 내용은 네트워크 유형을 참조하세요. 자세한 내용은 에서 서브넷 생성을 참조하세요VPC.

다음 절차에서는 `mysubnetgroup` (콘솔), 및 라는 서브넷 그룹을 생성하는 방법을 보여줍니다 AWS CLI ElastiCache API.

서브넷 그룹 생성(콘솔)

다음 절차는 서브넷 그룹을 생성하는 방법을 보여줍니다(콘솔).

서브넷 그룹을 생성하려면(콘솔)

1. AWS 관리 콘솔에 로그인하고 에서 ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 탐색 목록에서 서브넷 그룹을 선택합니다.
3. [서브넷 그룹 생성]을 선택합니다.
4. 서브넷 그룹 생성 마법사에서 다음을 수행합니다. 원하는 대로 모두 설정되었으면 Create를 선택합니다.

    a. [Name] 상자에 서브넷 그룹의 이름을 입력합니다.

    b. [Description] 상자에 서브넷 그룹에 대한 설명을 입력합니다.

    c. VPC ID 상자에서 Amazon 를 선택합니다VPC.

    d. 기본적으로 모든 서브넷이 선택됩니다. 선택한 서브넷 패널에서 관리를 클릭하고 프라이빗 서브넷IDs의 가용 영역 또는 로컬 영역 및 를 선택한 다음 선택을 선택합니다.

5. 나타나는 확인 메시지에서 [Close]를 선택합니다.

새 서브넷 그룹이 ElastiCache 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-cache-subnet-group` 명령을 사용하여 서브넷 그룹을 생성합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache create-cache-subnet-group \
    --cache-subnet-group-name mysubnetgroup \
    --cache-subnet-group-description "Testing" \
    --subnet-ids subnet-53df9c3a
```

Windows의 경우:

```
aws elasticache create-cache-subnet-group ^
    --cache-subnet-group-name mysubnetgroup ^
    --cache-subnet-group-description "Testing" ^
    --subnet-ids subnet-53df9c3a
```

이 명령은 다음과 유사한 출력을 생성합니다.

```
{
    "CacheSubnetGroup": {
        "VpcId": "vpc-37c3cd17",
        "CacheSubnetGroupDescription": "Testing",
        "Subnets": [
            {
                "SubnetIdentifier": "subnet-53df9c3a",
                "SubnetAvailabilityZone": {
                    "Name": "us-west-2a"
                }
            }
        ],
        "CacheSubnetGroupName": "mysubnetgroup"
    }
}
```

자세한 내용은 AWS CLI 주제를 참조하세요.create-cache-subnet-group.

캐시에 서브넷 그룹 할당

서브넷 그룹을 생성한 후 Amazon 에서 캐시를 시작할 수 있습니다VPC. 추가 정보는 다음을 참조하세요.

- Memcached 클러스터 - Memcached 클러스터를 시작하려면 Memcached 클러스터 생성(콘솔) 섹션을 참조하세요. 7.a단계(고급 Memcached 설정)에서 VPC 서브넷 그룹을 선택합니다.
- 독립 실행형 Valkey 또는 Redis OSS 클러스터 - 단일 노드 Valkey 또는 Redis OSS 클러스터를 시작하려면 섹션을 참조하세요Valkey(클러스터 모드 비활성화됨) 클러스터 생성(콘솔). 7.a단계(고급 Redis OSS 설정)에서 VPC 서브넷 그룹을 선택합니다.
- Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹 - 에서 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹을 시작하려면 섹션을 VPC참조하세요처음부터 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 복제 그룹 생성. 7.b단계(고급 Redis OSS 설정)에서 VPC 서브넷 그룹을 선택합니다.
- Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 복제 그룹 - Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 생성(콘솔). 6.i단계(고급 Redis OSS 설정)에서 VPC 서브넷 그룹을 선택합니다.

서브넷 그룹 수정

서브넷 그룹의 설명을 수정하거나 서브넷 그룹과 IDs 연결된 서브넷 목록을 수정할 수 있습니다. 캐시가 현재 해당 서브넷을 사용하고 있는 경우 서브넷 그룹에서 서브넷 ID를 삭제할 수 없습니다.

다음 절차는 서브넷 그룹을 수정하는 방법을 보여줍니다.

서브넷 그룹 수정(콘솔)

서브넷 그룹을 수정하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 서브넷 그룹을 선택합니다.
3. 서브넷 그룹 목록에서 수정하려는 서브넷 그룹의 라디오 버튼을 선택하고 수정을 선택합니다.
4. 선택한 서브넷 패널에서 관리를 선택합니다.
5. 선택한 서브넷을 변경하고 선택을 클릭합니다.
6. 변경 사항 저장을 클릭해 저장합니다.

서브넷 그룹 수정(AWS CLI)

명령 프롬프트에서 `modify-cache-subnet-group` 명령을 사용하여 서브넷 그룹을 수정합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-cache-subnet-group \
    --cache-subnet-group-name mysubnetgroup \
    --cache-subnet-group-description "New description" \
    --subnet-ids "subnet-42df9c3a" "subnet-48fc21a9"
```

Windows의 경우:

```
aws elasticache modify-cache-subnet-group ^
    --cache-subnet-group-name mysubnetgroup ^
    --cache-subnet-group-description "New description" ^
    --subnet-ids "subnet-42df9c3a" "subnet-48fc21a9"
```

이 명령은 다음과 유사한 출력을 생성합니다.

```
{
    "CacheSubnetGroup": {
        "VpcId": "vpc-73cd3c17",
        "CacheSubnetGroupDescription": "New description",
        "Subnets": [
            {
                "SubnetIdentifier": "subnet-42dcf93a",
                "SubnetAvailabilityZone": {
                    "Name": "us-west-2a"
                }
            },
            {
                "SubnetIdentifier": "subnet-48fc12a9",
                "SubnetAvailabilityZone": {
                    "Name": "us-west-2a"
                }
            }
        ],
        "CacheSubnetGroupName": "mysubnetgroup"
    }
}
```

자세한 내용은 AWS CLI 주제를 참조하세요.modify-cache-subnet-group.

서브넷 그룹 삭제

서브넷 그룹이 더 이상 필요하지 않다고 판단되면 삭제할 수 있습니다. 현재 캐시에서 사용 중인 서브넷 그룹은 삭제할 수 없습니다.

다음 절차는 서브넷 그룹을 삭제하는 방법을 보여줍니다.

서브넷 그룹 삭제(콘솔)

서브넷 그룹을 삭제하는 방법

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 서브넷 그룹을 선택합니다.
3. 서브넷 그룹 목록에서 삭제할 서브넷 그룹을 선택한 다음 [Delete]를 선택합니다.
4. 이 작업을 확인하라는 메시지가 표시되면 텍스트 입력 필드에 서브넷 그룹 이름을 입력하고 삭제를 선택합니다.

서브넷 그룹 삭제(AWS CLI)

를 사용하여 다음 파라미터delete-cache-subnet-group로 명령을 AWS CLI호출합니다.

- `--cache-subnet-group-name` *`mysubnetgroup`*

Linux, macOS, Unix의 경우:

```
aws elasticache delete-cache-subnet-group \
    --cache-subnet-group-name mysubnetgroup
```

Windows의 경우:

```
aws elasticache delete-cache-subnet-group ^
    --cache-subnet-group-name mysubnetgroup
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS CLI 주제를 참조하세요.delete-cache-subnet-group.

Amazon용 자격 증명 및 액세스 관리 ElastiCache

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어하는 데 도움이 AWS 서비스 되는 입니다. IAM 관리자는 ElastiCache 리소스를 사용할 수 있는 인증(로그인) 및 권한 부여(권한 보유) 대상을 제어합니다. IAM 는 추가 비용 없이 사용할 수 AWS 서비스 있는 입니다.

주제

고객

AWS Identity and Access Management (IAM) 사용 방법은 에서 수행하는 작업에 따라 다릅니다 ElastiCache.

서비스 사용자 - ElastiCache 서비스를 사용하여 작업을 수행하는 경우 관리자는 필요한 자격 증명과 권한을 제공합니다. 더 많은 ElastiCache 기능을 사용하여 작업을 수행하면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방식을 이해하면 적절한 권한을 관리자에게 요청할 수 있습니다. 에서 기능에 액세스할 수 없는 경우 섹션을 ElastiCache참조하세요Amazon ElastiCache 자격 증명 및 액세스 문제 해결.

서비스 관리자 - 회사에서 ElastiCache 리소스를 담당하는 경우 에 대한 전체 액세스 권한이 있을 수 있습니다 ElastiCache. 서비스 사용자가 액세스해야 하는 ElastiCache 기능과 리소스를 결정하는 것은 사용자의 작업입니다. 그런 다음 IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 의 기본 개념을 이해합니다IAM. 회사에서 IAM를 사용하는 방법에 대한 자세한 내용은 섹션을 ElastiCache참조하세요Amazon의 ElastiCache 작업 방식 IAM.

IAM 관리자 - IAM 관리자인 경우 에 대한 액세스를 관리하기 위한 정책을 작성하는 방법에 대한 세부 정보를 알고 싶을 수 있습니다 ElastiCache. 에서 사용할 수 있는 자격 ElastiCache 증명 기반 정책 예제를 보려면 섹션을 IAM참조하세요Amazon의 자격 증명 기반 정책 예제 ElastiCache.

ID를 통한 인증

인증은 자격 증명 AWS 으로 에 로그인하는 방법입니다. 로 AWS 계정 루트 사용자, IAM 사용자로 또는 IAM 역할을 수임하여 인증(에 로그인 AWS)되어야 합니다.

자격 증명 소스를 통해 제공된 자격 증명을 사용하여 에 페더레이션 자격 증명 AWS 으로 로그인할 수 있습니다. AWS IAM Identity Center (IAM Identity Center) 사용자, 회사의 Single Sign-On 인증 및 Google 또는 Facebook 자격 증명은 페더레이션 자격 증명의 예입니다. 페더레이션 자격 증명으로 로그인하면 관리자가 이전에 IAM 역할을 사용하여 자격 증명 페더레이션을 설정합니다. 페더레이션을 사용하여 AWS 에 액세스하면 간접적으로 역할을 수임하게 됩니다.

사용자 유형에 따라 AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. 에 로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의 에 로그인하는 방법을 AWS 계정 AWS참조하세요.

AWS 프로그래밍 방식으로 에 액세스하는 경우 는 소프트웨어 개발 키트(SDK)와 명령줄 인터페이스(CLI)를 AWS 제공하여 자격 증명을 사용하여 요청에 암호화 방식으로 서명합니다. AWS 도구를 사용하지 않는 경우 직접 요청에 서명해야 합니다. 권장 방법을 사용하여 직접 요청에 서명하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 요청 서명을 참조하세요 AWS API.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어, 다중 인증(MFA)을 사용하여 계정의 보안을 강화하는 것이 AWS 좋습니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서의 다중 인증 및 사용 설명서의 다중 인증 사용(MFA) AWS을 참조하세요IAM.

AWS 계정 루트 사용자

를 생성하면 계정의 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 하나의 로그인 자격 증명으로 AWS 계정시작합니다. 이 자격 증명을 AWS 계정 루트 사용자라고 하며 계정을 생성하는 데 사용한 이메일 주소와 암호로 로그인하여 액세스합니다. 일상적인 작업에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 보안 인증 정보를 보호하고 루트 사용자만 수행할 수 있는 작업을 수행하는 데 사용합니다. 루트 사용자로 로그인해야 하는 작업의 전체 목록은 IAM 사용 설명서의 루트 사용자 보안 인증이 필요한 작업을 참조하세요.

페더레이션 자격 증명

가장 좋은 방법은 관리자 액세스가 필요한 사용자를 포함한 인간 사용자가 ID 공급자와의 페더레이션을 사용하여 임시 자격 증명을 사용하여 AWS 서비스 에 액세스하도록 요구하는 것입니다.

페더레이션 자격 증명은 엔터프라이즈 사용자 디렉터리, 웹 자격 증명 공급자, , AWS Directory Service Identity Center 디렉터리 또는 자격 증명 소스를 통해 제공된 자격 증명을 사용하여 AWS 서비스 에 액세스하는 모든 사용자의 사용자입니다. 페더레이션 자격 증명이 에 액세스하면 역할을 AWS 계정수임하고 역할은 임시 자격 증명을 제공합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center(을)를 사용하는 것이 좋습니다. IAM Identity Center에서 사용자 및 그룹을 생성하거나 및 애플리케이션에서 사용할 수 있도록 자체 자격 증명 소스의 사용자 AWS 계정 및 그룹 집합에 연결하고 동기화할 수 있습니다. IAM Identity Center에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 IAM Identity Center란 무엇입니까?를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 한 사람 또는 애플리케이션에 대한 특정 권한이 AWS 계정 있는 내 자격 증명입니다. 가능한 경우 암호 및 액세스 키와 같은 장기 보안 인증 정보가 있는 IAM 사용자를 생성하는 대신 임시 보안 인증 정보를 사용하는 것이 좋습니다. 그러나 IAM 사용자와 장기 보안 인증이 필요한 특정 사용 사례가 있는 경우 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 장기 보안 인증이 필요한 사용 사례에 대한 액세스 키 정기적으로 교체를 참조하세요.

IAM 그룹은 IAM 사용자 컬렉션을 지정하는 자격 증명입니다. 사용자는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어 라는 이름의 그룹을 지정IAMAdmins하고 해당 그룹에 IAM 리소스를 관리할 수 있는 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 보안 인증 정보를 가지고 있지만, 역할은 임시 보안 인증만 제공합니다. 자세한 내용은 IAM 사용 설명서의 (역할 대신) IAM 사용자를 생성할 시기를 참조하세요.

IAM 역할

IAM 역할은 특정 권한이 AWS 계정 있는 내 자격 증명입니다. IAM 사용자와 비슷하지만 특정 사용자와는 관련이 없습니다. IAM 역할을 전환 AWS Management Console 하여 에서 역할을 일시적으로 수임할 수 있습니다. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-console.html 또는 AWS API 작업을 호출 AWS CLI 하거나 사용자 지정 를 사용하여 역할을 수임할 수 있습니다URL. 역할 사용 방법에 대한 자세한 내용은 IAM 사용 설명서의 역할 수임 방법을 참조하세요.

IAM 임시 자격 증명이 있는 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 - 페더레이션 ID에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 페더레이션 ID가 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 역할에 대한 자세한 내용은 IAM 사용 설명서의 타사 자격 증명 공급자에 대한 역할 생성을 참조하세요. IAM Identity Center를 사용하는 경우 권한 세트를 구성합니다. 인증 후 자격 증명이 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 세트를 의 역할과 상호 연관시킵니다IAM. 권한 세트에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 세트를 참조하세요.
- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 특정 작업에 대해 일시적으로 다른 권한을 맡을 IAM 수 있습니다.
- 교차 계정 액세스 - IAM 역할을 사용하여 다른 계정의 누군가(신뢰할 수 있는 보안 주체)가 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 에서는 정책을 리소스에 직접 연결할 AWS 서비스수 있습니다(역할을 프록시로 사용하는 대신). 교차 계정 액세스에 대한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 에서 교차 계정 리소스 액세스를 IAM 참조하세요.
- 교차 서비스 액세스 - 일부 는 다른 에서 기능을 AWS 서비스 사용합니다 AWS 서비스. 예를 들어 서비스에서 호출할 때 해당 서비스가 Amazon에서 애플리케이션을 실행EC2하거나 Amazon S3에 객체를 저장하는 것이 일반적입니다. 서비스는직접적으로 호출하는 보안 주체의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.
  - 전달 액세스 세션(FAS) - IAM 사용자 또는 역할을 사용하여 에서 작업을 수행하면 보안 주체로 AWS간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS 는 를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대한 요청과 AWS 서비스함께 사용합니다. FAS 요청은 서비스가 다른 AWS 서비스 또는 리소스와의 상호 작용을 완료해야 하는 요청을 수신할 때만 수행됩니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 액세스 세션 전달을 참조하세요.
  - 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하기 위해 수임하는 IAM 역할입니다. IAM 관리자는 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다IAM. 자세한 내용은 IAM 사용 설명서의 에 권한을 위임할 역할 생성을 AWS 서비스 참조하세요.
  - 서비스 연결 역할 - 서비스 연결 역할은 에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은 에 표시 AWS 계정 되며 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할에 대한 권한을 볼 수 있지만 편집할 수는 없습니다.
- Amazon에서 실행되는 애플리케이션 EC2 - IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 AWS CLI 또는 AWS API 요청을 수행하는 애플리케이션의 임시 보안 인증을 관리할 수 있습니다. 이는

EC2 인스턴스 내에 액세스 키를 저장하는 것보다 좋습니다. EC2 인스턴스에 AWS 역할을 할당하고 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로파일을 생성합니다. 인스턴스 프로파일에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행 중인 프로그램이 임시 보안 인증을 가져올 수 있습니다. 자세한 내용은 IAM 사용 설명서EC2의 IAM 역할 사용을 참조하세요.

IAM 역할 또는 IAM 사용자를 사용할지 여부를 알아보려면 IAM 사용 설명서의 IAM 역할 생성 시기(사용자 대신)를 참조하세요.

정책을 사용한 액세스 관리

정책을 AWS 생성하고 AWS 자격 증명 또는 리소스에 연결하여 의 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결된 AWS 경우 권한을 정의하는 의 객체입니다. 는 보안 주체(사용자, 루트 사용자 또는 역할 세션)가 요청할 때 이러한 정책을 AWS 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는 지를 결정합니다. 대부분의 정책은 에 JSON 문서 AWS 로 저장됩니다. JSON 정책 문서의 구조 및 내용에 대한 자세한 내용은 IAM 사용 설명서의 JSON 정책 개요를 참조하세요.

관리자는 정책을 사용하여 AWS JSON 대상에 액세스할 수 있는 사용자를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

기본적으로, 사용자와 역할에는 어떠한 권한도 없습니다. 사용자에게 필요한 리소스에 대한 작업을 수행할 수 있는 권한을 부여하기 위해 IAM 관리자는 IAM 정책을 생성할 수 있습니다. 그런 다음 관리자는 IAM 정책을 역할에 추가하고 사용자는 역할을 수임할 수 있습니다.

IAM 정책은 작업을 수행하는 데 사용하는 방법에 관계없이 작업에 대한 권한을 정의합니다. 예를 들어, `iam:GetRole` 작업을 허용하는 정책이 있다고 가정합니다. 해당 정책을 사용하는 사용자는 AWS Management Console, AWS CLI또는 에서 역할 정보를 가져올 수 있습니다 AWS API.

보안 인증 기반 정책

자격 증명 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 자격 증명에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

보안 인증 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은 의 여러 사용자, 그룹 및 역할에 연결할 수 있는 독립 실행형 정책입니다 AWS 계정. 관리형 정책에는 AWS 관리형 정책 및 고객 관리형 정책이

포함됩니다. 관리형 정책 또는 인라인 정책 중에서 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책 및 인라인 정책 선택을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 문서입니다. 리소스 기반 정책의 예로는 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. 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 또는 가 포함될 수 있습니다 AWS 서비스.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책IAM에서는 의 AWS 관리형 정책을 사용할 수 없습니다.

액세스 제어 목록(ACLs)

액세스 제어 목록(ACLs)은 리소스에 액세스할 수 있는 권한이 있는 보안 주체(계정 멤버, 사용자 또는 역할)를 제어합니다. ACLs 는 리소스 기반 정책과 유사하지만 JSON 정책 문서 형식을 사용하지는 않습니다.

Amazon S3 AWS WAF및 AmazonVPC은 를 지원하는 서비스의 예입니다ACLs. 에 대한 자세한 내용은 Amazon Simple Storage Service 개발자 안내서의 액세스 제어 목록(ACL) 개요를 ACLs참조하세요.

기타 정책 타입

AWS 는 덜 일반적인 추가 정책 유형을 지원합니다. 이러한 정책 타입은 더 일반적인 정책 유형에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 - 권한 경계는 자격 증명 기반 정책이 IAM엔터티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 객체의 자격 증명 기반 정책과 그 권한 경계의 교집합입니다. `Principal` 필드에서 사용자나 역할을 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 내용은 IAM 사용 설명서의 IAM엔터티에 대한 권한 경계를 참조하세요.
- 서비스 제어 정책(SCPs) - 의 조직 또는 조직 단위(OU)에 대한 최대 권한을 지정하는 SCPs JSON 정책입니다 AWS Organizations. AWS Organizations 는 비즈니스가 소유 AWS 계정 한 여러 을 그룹화하고 중앙에서 관리하기 위한 서비스입니다. 조직의 모든 기능을 활성화하면 서비스 제어 정책(SCPs)을 계정의 일부 또는 전체에 적용할 수 있습니다. 는 각 를 포함하여 멤버 계정의 엔터

티에 대한 권한을 SCP 제한합니다 AWS 계정 루트 사용자. 조직 및 에 대한 자세한 내용은 AWS Organizations 사용 설명서의 서비스 제어 정책을 SCPs참조하세요.

- 세션 정책 – 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 보안 인증 기반 정책의 교차와 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 내용은 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. AWS 에서 여러 정책 유형이 관련될 때 요청을 허용할지 여부를 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

Amazon의 ElastiCache 작업 방식 IAM

IAM 를 사용하여 에 대한 액세스를 관리하기 전에 에서 사용할 수 있는 IAM 기능에 대해 ElastiCache 알아봅니다 ElastiCache.

IAM Amazon에서 사용할 수 있는 기능 ElastiCache

IAM 기능	ElastiCache 지원
ID 기반 정책	예
리소스 기반 정책	아니요
정책 작업	예
정책 리소스	예
정책 조건 키	예
ACLs	예
ABAC (정책의 태그)	예
임시 보안 인증	예

IAM 기능	ElastiCache 지원
보안 주체 권한	예
서비스 역할	예
서비스 연결 역할	예

ElastiCache 및 기타 AWS 서비스가 대부분의 IAM 기능을 어떻게 사용하는지 자세히 알아보려면 IAM 사용 설명서의 에서 AWS 를 사용하는 서비스를 IAM 참조하세요.

에 대한 자격 증명 기반 정책 ElastiCache

ID 기반 정책 지원: 예

자격 증명 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 자격 증명에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

IAM 자격 증명 기반 정책을 사용하면 허용되거나 거부된 작업 및 리소스와 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. 보안 인증 기반 정책에서는 보안 주체가 연결된 사용자 또는 역할에 적용되므로 보안 주체를 지정할 수 없습니다. JSON 정책에서 사용할 수 있는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

ElastiCache 자격 증명 기반 정책 예시

ElastiCache 자격 증명 기반 정책의 예를 보려면 섹션을 참조하세요Amazon의 자격 증명 기반 정책 예제 ElastiCache.

ElastiCache 내 리소스 기반 정책

리소스 기반 정책 지원: 아니요

리소스 기반 정책은 리소스에 연결하는 JSON 정책 문서입니다. 리소스 기반 정책의 예로는 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. 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 또는 가 포함될 수 있습니다 AWS 서비스.

교차 계정 액세스를 활성화하려면 리소스 기반 정책의 보안 주체로 전체 계정 또는 다른 계정의 IAM엔터티를 지정할 수 있습니다. 리소스 기반 정책에 크로스 계정 보안 주체를 추가하는 것은 트러스트 관계 설정의 절반밖에 되지 않는다는 것을 유념하십시오. 보안 주체와 리소스가 다른 에 있는 경우 신뢰할 수 AWS 계정있는 계정의 IAM 관리자는 보안 주체 엔터티(사용자 또는 역할)에게 리소스에 액세스할 수 있는 권한도 부여해야 합니다. 엔터티에 ID 기반 정책을 연결하여 권한을 부여합니다. 하지만 리소스 기반 정책이 동일 계정의 보안 주체에 액세스를 부여하는 경우, 추가 자격 증명 기반 정책이 필요하지 않습니다. 자세한 내용은 IAM 사용 설명서의 에서 크로스 계정 리소스 액세스를 IAM 참조하세요.

에 대한 정책 작업 ElastiCache

정책 작업 지원: 예

관리자는 정책을 사용하여 AWS JSON 대상에 액세스할 수 있는 사용자를 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. 정책 작업은 일반적으로 연결된 AWS API 작업과 이름이 동일합니다. 일치하는 API 작업이 없는 권한 전용 작업과 같은 몇 가지 예외가 있습니다. 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 일컬어 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하십시오.

ElastiCache 작업 목록을 보려면 서비스 승인 참조의 Amazon에서 정의한 작업을 ElastiCache 참조하세요.

의 정책 작업은 작업 앞에 다음 접두사를 ElastiCache 사용합니다.

```
elasticache
```

단일 문에서 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
      "elasticache:action1",
      "elasticache:action2"
         ]
```

와일드카드(*)를 사용하여 여러 작업을 지정할 수 있습니다. 예를 들어, Describe라는 단어로 시작하는 모든 태스크를 지정하려면 다음 태스크를 포함합니다.

```
"Action": "elasticache:Describe*"
```

ElastiCache 자격 증명 기반 정책의 예를 보려면 섹션을 참조하세요Amazon의 자격 증명 기반 정책 예제 ElastiCache.

에 대한 정책 리소스 ElastiCache

정책 리소스 지원: 예

관리자는 정책을 사용하여 AWS JSON 대상에 액세스할 수 있는 사용자를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

Resource JSON 정책 요소는 작업이 적용되는 객체를 지정합니다. 문장에는 Resource또는 NotResource요소가 반드시 추가되어야 합니다. 가장 좋은 방법은 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정하는 것입니다. 리소스 수준 권한이라고 하는 특정 리소스 유형을 지원하는 작업에 대해 이 태스크를 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

ElastiCache 리소스 유형 및 해당 의 목록을 보려면 서비스 승인 참조의 Amazon에서 정의한 리소스를 ElastiCache ARNs참조하세요. 각 리소스ARN의 를 지정할 수 있는 작업을 알아보려면 Amazon 에서 정의한 작업을 ElastiCache 참조하세요.

ElastiCache 자격 증명 기반 정책의 예를 보려면 섹션을 참조하세요Amazon의 자격 증명 기반 정책 예제 ElastiCache.

ElastiCache 정책 조건 키

서비스별 정책 조건 키 지원: 예

관리자는 정책을 사용하여 AWS JSON 대상에 액세스할 수 있는 사용자를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Condition` 요소(또는 `Condition` 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. `Condition` 요소는 옵션입니다. 같거나 작음과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다.

한 문에서 여러 `Condition` 요소를 지정하거나 단일 `Condition` 요소에서 여러 키를 지정하는 경우, AWS 는 논리적 `AND` 태스크를 사용하여 평가합니다. 단일 조건 키에 여러 값을 지정하는 경우 는 논리적 `OR` 작업을 사용하여 조건을 AWS 평가합니다. 명문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예를 들어 IAM 리소스에 사용자 IAM 이름으로 태그가 지정된 경우에만 사용자에게 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM 정책 요소: 변수 및 태그를 참조하세요.

AWS 는 전역 조건 키 및 서비스별 조건 키를 지원합니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

ElastiCache 조건 키 목록을 보려면 서비스 권한 부여 참조의 Amazon용 조건 키를 참조하세요 ElastiCache. 조건 키를 사용할 수 있는 작업 및 리소스를 알아보려면 Amazon 에서 정의한 작업을 ElastiCache 참조하세요.

ElastiCache 자격 증명 기반 정책의 예를 보려면 섹션을 참조하세요Amazon의 자격 증명 기반 정책 예제 ElastiCache.

의 액세스 제어 목록(ACLs) ElastiCache

지원 ACLs: 예

액세스 제어 목록(ACLs)은 리소스에 액세스할 수 있는 권한이 있는 보안 주체(계정 멤버, 사용자 또는 역할)를 제어합니다. ACLs 는 리소스 기반 정책과 유사하지만 JSON 정책 문서 형식을 사용하지는 않습니다.

를 사용한 속성 기반 액세스 제어(ABAC) ElastiCache

지원ABAC(정책의 태그): 예

속성 기반 액세스 제어(ABAC)는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. 에서는 AWS이러한 속성을 태그 라고 합니다. IAM 엔터티(사용자 또는 역할) 및 여러 AWS 리소스에 태그를

연결할 수 있습니다. 엔터티 및 리소스에 태그를 지정하는 것은 의 첫 번째 단계입니다ABAC. 그런 다음 보안 주체의 태그가 액세스하려는 리소스의 태그와 일치할 때 작업을 허용하는 ABAC 정책을 설계합니다.

ABAC 는 빠르게 성장하는 환경에서 유용하며 정책 관리가 번거로워지는 상황에 도움이 됩니다.

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.

에 대한 자세한 내용은 IAM 사용 설명서의 란 무엇입니까ABAC?를 ABAC참조하세요. 설정 단계가 포함된 자습서를 보려면 IAM 사용 설명서의 속성 기반 액세스 제어 사용(ABAC)을 ABAC참조하세요.

에서 임시 자격 증명 사용 ElastiCache

임시 자격 증명 지원: 예

임시 자격 증명을 사용하여 로그인할 때 작동하지 AWS 서비스 않는 경우도 있습니다. 임시 자격 증명으로 AWS 서비스 작업하는 경우를 비롯한 자세한 내용은 IAM 사용 설명서의 AWS 서비스 에서 작업하는 IAM 섹션을 참조하세요.

사용자 이름 및 암호를 제외한 방법을 AWS Management Console 사용하여 에 로그인하는 경우 임시 자격 증명을 사용합니다. 예를 들어 회사의 Single Sign-On(SSO) 링크를 AWS 사용하여 에 액세스하면 해당 프로세스가 임시 자격 증명을 자동으로 생성합니다. 또한 콘솔에 사용자로 로그인한 다음 역할을 전환할 때 임시 보안 인증을 자동으로 생성합니다. 역할 전환에 대한 자세한 내용은 IAM 사용 설명서의 역할(콘솔)로 전환을 참조하세요.

AWS CLI 또는 를 사용하여 임시 자격 증명을 수동으로 생성할 수 있습니다 AWS API. 그런 다음 이러한 임시 자격 증명을 사용하여 장기 액세스 키를 사용하는 대신 임시 자격 증명을 동적으로 생성하는 AWS. AWS recommends에 액세스할 수 있습니다. 자세한 내용은 의 임시 보안 자격 증명을 IAM참조하세요.

ElastiCache의 서비스 간 보안 주체 권한

전달 액세스 세션 지원(FAS): 예

IAM 사용자 또는 역할을 사용하여 에서 작업을 수행하는 경우 보안 주체로 AWS간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS 는 를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대한 요청과 AWS 서비스함께 사용합니

다. FAS 요청은 서비스가 다른 AWS 서비스 또는 리소스와의 상호 작용을 완료해야 하는 요청을 수신할 때만 수행됩니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 액세스 세션 전달을 참조하세요.

에 대한 서비스 역할 ElastiCache

서비스 역할 지원: 예

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하기 위해 수임하는 IAM 역할입니다. IAM 관리자는 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다IAM. 자세한 내용은 IAM 사용 설명서의 에 권한을 위임할 역할 생성을 AWS 서비스 참조하세요.

> **Warning**
>
> 서비스 역할에 대한 권한을 변경하면 ElastiCache 기능이 중단될 수 있습니다. 에서 지침을 ElastiCache 제공하는 경우에만 서비스 역할을 편집합니다.

에 대한 서비스 연결 역할 ElastiCache

서비스 링크 역할 지원: 예

서비스 연결 역할은 에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은 에 표시 AWS 계정 되며 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할에 대한 권한을 볼 수 있지만 편집할 수는 없습니다.

서비스 연결 역할 생성 또는 관리에 대한 자세한 내용은 AWS 에서 작동하는 서비스를 참조하세요IAM. 서비스 연결 역할 열에서 `Yes`(이)가 포함된 서비스를 테이블에서 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 Yes(네) 링크를 선택합니다.

Amazon의 자격 증명 기반 정책 예제 ElastiCache

기본적으로 사용자 및 역할에는 리소스를 생성하거나 수정할 ElastiCache 수 있는 권한이 없습니다. 또한 AWS Management Console, AWS Command Line Interface (AWS CLI) 또는 를 사용하여 작업을 수행할 수 없습니다 AWS API. 사용자에게 필요한 리소스에 대한 작업을 수행할 수 있는 권한을 부여하기 위해 IAM 관리자는 IAM 정책을 생성할 수 있습니다. 그런 다음 관리자는 IAM 정책을 역할에 추가하고 사용자는 역할을 수임할 수 있습니다.

이러한 예제 정책 문서를 사용하여 IAM 자격 증명 기반 JSON 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

각 리소스 유형에 대한 형식을 포함하여 ElastiCache에서 정의한 작업 및 리소스 유형에 ARNs 대한 자세한 내용은 서비스 승인 참조의 Amazon용 작업, 리소스 및 조건 키를 참조하세요 ElastiCache.

주제

- 정책 모범 사례
- ElastiCache 콘솔 사용
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용

정책 모범 사례

자격 증명 기반 정책은 계정에서 ElastiCache 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부를 결정합니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따릅니다.

- AWS 관리형 정책을 시작하고 최소 권한 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 사용 설명서의 AWS 관리형 정책 또는 AWS 작업 함수에 대한 관리형 정책을 참조하세요.
- 최소 권한 적용 - IAM 정책으로 권한을 설정할 때 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. 를 사용하여 권한을 적용하는 IAM 방법에 대한 자세한 내용은 IAM 사용 설명서의 에서 정책 및 권한을 IAM 참조하세요.
- IAM 정책의 조건을 사용하여 액세스 추가 제한 - 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어 정책 조건을 작성하여 를 사용하여 모든 요청을 전송하도록 지정할 수 있습니다SSL. AWS 서비스와 같은 특정 를 통해 서비스 작업을 사용하는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다 AWS CloudFormation. 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 사용하여 IAM 정책을 검증하여 안전하고 기능적인 권한을 보장합니다. IAM Access Analyzer는 정책이 정책 언어(JSON) 및 IAM 모범 사례를 준수하도록 새 정책 및 기존 IAM 정책을 검증합니다. IAM Access Analyzer는 안전하고 기능적인 정책을 작성하는 데 도움이 되는 100개 이상의 정책 확인 및 실행 가능한 권장 사항을 제공합니다. 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer 정책 검증을 참조하세요.
- 다중 인증 필요(MFA) - 에 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 MFA 위해 를 AWS 계정켭니다. API 작업을 호출할 MFA 때 를 요구하려면 정책에 MFA 조건을 추가합니다. 자세한 내용은 IAM 사용 설명서의 MFA-보호된 API 액세스 구성을 참조하세요.

의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 의 보안 모범 사례를 IAM IAM참조하세요.

ElastiCache 콘솔 사용

Amazon ElastiCache 콘솔에 액세스하려면 최소 권한 집합이 있어야 합니다. 이러한 권한을 통해 의 ElastiCache 리소스에 대한 세부 정보를 나열하고 볼 수 있어야 합니다 AWS 계정. 최소 필수 권한보다 더 제한적인 자격 증명 기반 정책을 만들면 콘솔이 해당 정책에 연결된 엔터티(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

AWS CLI 또는 에만 전화를 거는 사용자에 대해 최소 콘솔 권한을 허용할 필요는 없습니다 AWS API. 대신 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 허용합니다.

사용자와 역할이 ElastiCache 콘솔을 계속 사용할 수 있도록 하려면 `ConsoleAccess` 또는 `ReadOnly` AWS 관리형 정책도 엔터티에 연결합니다 ElastiCache. 자세한 내용은 IAM 사용 설명서의 사용자에게 권한 추가를 참조하세요.

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예제에서는 IAM 사용자가 사용자 ID에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 AWS CLI 또는 를 사용하여 프로그래밍 방식으로 이 작업을 완료할 수 있는 권한이 포함되어 있습니다 AWS API.

```
{
    "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": "*"
        }
    ]
}
```

Amazon ElastiCache 자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 ElastiCache 및 작업 시 발생할 수 있는 일반적인 문제를 진단하고 해결할 수 있습니다IAM.

주제

- 에서 작업을 수행할 권한이 없습니다. ElastiCache
- iam을 수행할 권한이 없습니다.PassRole
- 내 AWS 계정 외부의 사람이 내 ElastiCache 리소스에 액세스하도록 허용하고 싶습니다.

에서 작업을 수행할 권한이 없습니다. ElastiCache

에 작업을 수행할 권한이 없다고 AWS Management Console 표시되면 관리자에게 문의하여 지원을 받아야 합니다. 관리자는 사용자 이름과 비밀번호를 제공한 사람입니다.

다음 예제 오류는 `mateojackson` 사용자가 콘솔을 사용하여 가상 *`my-example-widget`* 리소스에 대한 세부 정보를 보려고 하지만 가상 `elasticache:`*`GetWidget`* 권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 elasticache:GetWidget on resource: my-example-widget
```

이 경우 Mateo는 *`my-example-widget`* 작업을 사용하여 `elasticache:`*`GetWidget`* 리소스에 액세스하도록 허용하는 정책을 업데이트하라고 관리자에게 요청합니다.

iam을 수행할 권한이 없습니다.PassRole

`iam:PassRole` 작업을 수행할 권한이 없다는 오류가 발생하면 역할을 에 전달할 수 있도록 정책을 업데이트해야 합니다 ElastiCache.

일부는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서비스에 전달할 수 있도록 AWS 서비스 허용합니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예제 오류는 이름이 인 IAM 사용자가 콘솔을 사용하여 에서 작업을 수행하려고 `marymajor` 할 때 발생합니다 ElastiCache. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 수 있는 권한을 가지고 있지 않습니다.

```
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 ElastiCache 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(ACLs)을 지원하는 서비스의 경우 이러한 정책을 사용하여 사용자에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하세요.

- 가 이러한 기능을 ElastiCache 지원하는지 알아보려면 섹션을 참조하세요Amazon의 ElastiCache 작업 방식 IAM.
- 소유 AWS 계정 한 의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 소유 AWS 계정 한 다른 의 IAM 사용자에게 액세스 권한 제공을 참조하세요.
- 타사에 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사 AWS 계정 소유 에 대한 액세스 권한 제공을 AWS 계정참조하세요.
- 자격 증명 페더레이션을 통해 액세스를 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부 인증 사용자(자격 증명 페더레이션)에 대한 액세스 제공을 참조하세요.

- 교차 계정 액세스를 위한 역할 및 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 에서 교차 계정 리소스 액세스를 IAM 참조하세요.

액세스 제어

요청을 인증하는 데 유효한 보안 인증 정보가 있을 수 있지만 권한이 없는 한 ElastiCache 리소스를 생성하거나 액세스할 수 없습니다. 예를 들어 ElastiCache 클러스터를 생성할 권한이 있어야 합니다.

다음 섹션에서는 에 대한 권한을 관리하는 방법을 설명합니다 ElastiCache. 먼저 개요를 읽어 보면 도움이 됩니다.

- ElastiCache 리소스에 대한 액세스 권한 관리 개요
- Amazon에 대한 자격 증명 기반 정책(IAM 정책) 사용 ElastiCache

ElastiCache 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 자격 증명(즉, 사용자, 그룹 및 역할)에 권한 정책을 연결할 수 있습니다. 또한 Amazon은 리소스에 권한 정책 연결을 ElastiCache 지원합니다.

> **Note**
>
> 계정 관리자 또는 관리자 사용자는 관리자 권한이 있는 사용자입니다. 자세한 내용은 IAM 사용 설명서의 IAM 모범 사례를 참조하세요.

액세스 권한을 제공하려면 사용자, 그룹 또는 역할에 권한을 추가하세요:

- 의 사용자 및 그룹 AWS IAM Identity Center:

  권한 세트를 생성합니다. AWS IAM Identity Center 사용 설명서의 권한 세트 생성의 지침을 따릅니다.
- 자격 증명 공급자를 IAM 통해 에서 관리되는 사용자:

  ID 페더레이션을 위한 역할을 생성합니다. IAM 사용 설명서의 타사 자격 증명 공급자(페더레이션)에 대한 역할 생성의 지침을 따릅니다.
- IAM 사용자:
  - 사용자가 맡을 수 있는 역할을 생성합니다. IAM 사용 설명서의 IAM 사용자 역할 생성의 지침을 따릅니다.
  - (권장되지 않음)정책을 사용자에게 직접 연결하거나 사용자를 사용자 그룹에 추가합니다. IAM 사용 설명서의 사용자(콘솔)에 권한 추가의 지침을 따릅니다.

주제

- Amazon ElastiCache 리소스 및 작업
- 리소스 소유권 이해
- 리소스 액세스 관리
- AWS Amazon용 관리형 정책 ElastiCache
- Amazon에 대한 자격 증명 기반 정책(IAM 정책) 사용 ElastiCache
- 리소스 수준 권한

- 조건 키 사용
- Amazon용 서비스 연결 역할 사용 ElastiCache
- ElastiCache API 권한: 작업, 리소스 및 조건 참조

Amazon ElastiCache 리소스 및 작업

ElastiCache 리소스 유형 및 해당 의 목록을 보려면 서비스 승인 참조의 Amazon에서 정의한 리소스를 ElastiCache ARNs참조하세요. 각 리소스ARN의 를 지정할 수 있는 작업을 알아보려면 Amazon 에서 정의한 작업을 ElastiCache 참조하세요.

리소스 소유권 이해

리소스 소유자는 리소스를 생성한 AWS 계정입니다. 즉, 리소스 소유자는 리소스를 생성하는 요청을 인증하는 보안 주체의 AWS 계정입니다. 보안 주체 엔터티는 루트 계정, IAM 사용자 또는 IAM 역할일 수 있습니다). 다음 예에서는 이러한 작동 방식을 설명합니다.

- AWS 계정의 루트 계정 자격 증명을 사용하여 캐시 클러스터를 생성한다고 가정해 보겠습니다. 이 경우 AWS 계정은 리소스의 소유자입니다. 에서 ElastiCache리소스는 캐시 클러스터입니다.
- AWS 계정에서 IAM 사용자를 생성하고 해당 사용자에게 캐시 클러스터를 생성할 수 있는 권한을 부여한다고 가정해 보겠습니다. 이러한 경우, 이 사용자가 캐시 클러스터를 생성할 수 있습니다. 하지만 사용자가 속한 AWS 계정은 캐시 클러스터 리소스를 소유합니다.
- AWS 계정에서 캐시 클러스터를 생성할 수 있는 권한이 있는 IAM 역할을 생성한다고 가정해 보겠습니다. 이러한 경우, 이 역할을 수임할 사람은 캐시 클러스터를 생성할 수 있습니다. 역할이 속한 AWS 계정은 캐시 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 Amazon 의 컨텍스트IAM에서 를 사용하는 방법에 대해 설명합니다 ElastiCache. IAM 서비스에 대한 자세한 정보는 제공하지 않습니다. 전체 IAM 설명서는 IAM 사용 설명서의 IAM?란 무엇입니까?를 참조하세요. IAM 정책 구문 및 설명에 대한 자세한 내용은 IAM 사용 설명서의 AWS IAM 정책 참조를 참조하세요.

IAM 자격 증명에 연결된 정책을 자격 증명 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)

자격 IAM 증명에 정책을 연결할 수 있습니다. 예를 들어 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 권한은 해당 사용자가 캐시 클러스터, 파라미터 그룹 또는 보안 그룹과 같은 ElastiCache 리소스를 생성할 수 있는 권한입니다.
- 권한 정책을 역할에 연결(계정 간 권한 부여) - 자격 증명 기반 권한 정책을 IAM 역할에 연결하여 계정 간 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 관리자는 IAM 역할을 생성하고 계정 A의 리소스에 대한 권한을 부여하는 권한 정책을 역할에 연결합니다.
  2. 계정 A 관리자는 계정 B를 역할을 수임할 보안 주체로 식별하는 역할에 신뢰 정책을 연결합니다.
  3. 그런 다음 계정 B 관리자는 계정 B의 모든 사용자에게 역할을 맡을 권한을 위임할 수 있습니다. 이렇게 하면 계정 B의 사용자가 계정 A의 리소스를 생성하거나 액세스할 수 있습니다. 경우에 따라 역할을 맡을 수 있는 AWS 서비스 권한을 부여할 수 있습니다. 이러한 접근 방식을 지원하려면, 신뢰 정책의 보안 주체는 AWS 서비스 보안 주체일 수도 있습니다.

  를 사용하여 권한을 위임IAM하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 액세스 관리를 참조하세요.

다음은 사용자가 AWS 계정에 대한 `DescribeCacheClusters` 작업을 수행할 수 있도록 허용하는 정책 예제입니다. 는 API 작업에 ARNs 대한 리소스를 사용하여 특정 리소스를 식별하는 ElastiCache 것도 지원합니다. (이러한 접근 방식을 리소스 수준 권한이라고도 합니다.)

```
{
   "Version": "2012-10-17",
   "Statement": [{
```

```
      "Sid": "DescribeCacheClusters",
      "Effect": "Allow",
      "Action": [
         "elasticache:DescribeCacheClusters"],
      "Resource": resource-arn
      }
   ]
}
```

에서 자격 증명 기반 정책을 사용하는 방법에 대한 자세한 내용은 섹션을 ElastiCache참조하세요Amazon에 대한 자격 증명 기반 정책(IAM 정책) 사용 ElastiCache. 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 Amazon ElastiCache 리소스(참조Amazon ElastiCache 리소스 및 작업)에 대해 서비스는 일련의 API 작업을 정의합니다(작업 참조). 이러한 API 작업에 대한 권한을 부여하려면 정책에서 지정할 수 있는 일련의 작업을 ElastiCache 정의합니다. 예를 들어 클러스터 리소스의 ElastiCache 경우 , 및 `CreateCacheClusterDeleteCacheCluster`라는 작업이 정의됩니다`DescribeCacheCluster`. API 작업을 수행하려면 둘 이상의 작업에 대한 권한이 필요할 수 있습니다.

다음은 가장 기본적인 정책 요소입니다:

- 리소스 - 정책에서 Amazon 리소스 이름(ARN)을 사용하여 정책이 적용되는 리소스를 식별합니다. 자세한 내용은 Amazon ElastiCache 리소스 및 작업 단원을 참조하십시오.
- 조치 – 조치 키워드를 사용하여 허용 또는 거부할 리소스 작업을 식별합니다. 예를 들어 지정된 에 따라 `elasticache:CreateCacheCluster` 권한`Effect`은 Amazon ElastiCache `CreateCacheCluster` 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 연결된 사용자는 암시적 보안 주체입니다. 리소스 기반 정책의 경우, 사용자, 계정, 서비스 또는 권한의 수신자인 기타 개체를 지정합니다(리소스 기반 정책에만 해당).

IAM 정책 구문 및 설명에 대한 자세한 내용은 IAM 사용 설명서의 AWS IAM 정책 참조를 참조하세요.

모든 Amazon ElastiCache API 작업을 보여주는 표는 섹션을 참조하세요ElastiCache API 권한: 작업, 리소스 및 조건 참조.

정책에서 조건 지정

권한을 부여하면 IAM 정책 언어를 사용하여 정책이 적용되는 조건을 지정할 수 있습니다. 예를 들어, 특정 날짜 이후에만 정책을 적용할 수 있습니다. 정책 언어로 조건을 지정하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 조건을 참조하세요.

조건을 표시하려면 미리 정의된 조건 키를 사용합니다. ElastiCache특정 조건 키를 사용하려면 섹션을 참조하세요조건 키 사용. 필요에 따라 사용할 수 있는 AWS전체 조건 키가 있습니다. AWS전체 키의 전체 목록은 IAM 사용 설명서의 사용 가능한 조건 키 섹션을 참조하세요.

AWS Amazon용 관리형 정책 ElastiCache

AWS 관리형 정책은 에서 생성하고 관리하는 독립 실행형 정책입니다 AWS. AWS 관리형 정책은 사용자, 그룹 및 역할에 권한 할당을 시작할 수 있도록 많은 일반적인 사용 사례에 대한 권한을 제공하도록 설계되었습니다.

AWS 관리형 정책은 모든 AWS 고객이 사용할 수 있으므로 특정 사용 사례에 대해 최소 권한 권한을 부여하지 않을 수 있다는 점에 유의하세요. 사용 사례에 고유한 고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다.

AWS 관리형 정책에 정의된 권한은 변경할 수 없습니다. 가 관리형 정책에 정의된 AWS 권한을 AWS 업데이트하면 정책이 연결된 모든 보안 주체 자격 증명(사용자, 그룹 및 역할)에 영향을 미칩니다. AWS 는 새 AWS 서비스 가 시작되거나 기존 서비스에 새 API 작업을 사용할 수 있게 되면 AWS 관리형 정책을 업데이트할 가능성이 높습니다.

자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책을 참조하세요.

AWS 관리형 정책: ElastiCacheServiceRolePolicy

IAM 엔터티 ElastiCacheServiceRolePolicy 에 연결할 수 없습니다. 이 정책은 가 사용자를 대신하여 작업을 ElastiCache 수행할 수 있도록 허용하는 서비스 연결 역할에 연결됩니다.

이 정책은 ElastiCache 가 캐시 관리에 필요한 경우 사용자를 대신하여 AWS 리소스를 관리할 수 있도록 허용합니다.

- `ec2` - VPC 엔드포인트(서버리스 캐시의 경우), 탄력적 네트워크 인터페이스(ENIs)(자체 설계된 클러스터의 경우) 및 보안 그룹을 포함하여 캐시 노드에 연결할 EC2 네트워킹 리소스를 관리합니다.
- `cloudwatch` – 서비스에서 지표 데이터를 로 내보냅니다 CloudWatch.
- `outposts` - AWS Outposts에서 캐시 노드 생성을 허용합니다.

정책은 IAM 콘솔 및 관리형 ElastiCacheServiceRolePolicy 정책 참조 가이드 ElastiCacheServiceRolePolicy 에서 찾을 수 있습니다. AWS

AWS 관리형 정책: AmazonElastiCacheFullAccess

`AmazonElastiCacheFullAccess` 정책을 IAM 자격 증명에 연결할 수 있습니다.

이 정책은 보안 주체가 AWS 관리 콘솔을 ElastiCache 사용하여 에 대한 전체 액세스를 허용합니다.

- `elasticache` - 모든 에 액세스합니다APIs.
- `iam` - 서비스 운영에 필요한 서비스 연결 역할을 생성합니다.
- `ec2` - 캐시 생성에 필요한 종속 EC2 리소스(VPC, 서브넷, 보안 그룹)를 설명하고 VPC 엔드포인트 생성 허용(서버리스 캐시의 경우).
- `kms` - 에 CMKs 대한 고객 관리형 사용을 허용합니다 encryption-at-rest.
- `cloudwatch` - 콘솔에 지표를 표시할 수 있도록 ElastiCache 지표에 대한 액세스를 허용합니다.
- `application-autoscaling` - 캐시에 대한 자동 크기 조정 정책을 설명하기 위한 액세스를 허용합니다.
- `logs` - 콘솔에서 로그 전송 기능용 로그 스트림을 채우는 데 사용됩니다.
- `firehose` - 콘솔에서 로그 전송 기능용 전송 스트림을 채우는 데 사용됩니다.
- `s3` - 콘솔에서 스냅샷 복원 기능용 S3 버킷을 채우는 데 사용됩니다.
- `outposts` - 콘솔에서 캐시 생성을 위해 AWS Outposts를 채우는 데 사용됩니다.
- `sns` - 콘솔에서 알림 기능에 대한 SNS 주제를 채우는 데 사용됩니다.

IAM 콘솔 및 관리형 AmazonElastiCacheFullAccess 정책 참조 가이드AmazonElastiCacheFullAccess에서 정책을 찾을 수 있습니다. AWS

AWS 관리형 정책: AmazonElastiCacheReadOnlyAccess

`AmazonElastiCacheReadOnlyAccess` 정책을 IAM 자격 증명에 연결할 수 있습니다.

이 정책은 보안 주체가 AWS 관리 콘솔을 ElastiCache 사용하여 에 대한 읽기 전용 액세스를 허용합니다.

- `elasticache` — 읽기 전용 `Describe` 에 액세스합니다APIs.

IAM 콘솔 및 관리형 AmazonElastiCacheReadOnlyAccess 정책 참조 가이드AmazonElastiCacheReadOnlyAccess에서 정책을 찾을 수 있습니다. AWS

ElastiCache AWS 관리형 정책에 대한 업데이트

이 서비스가 이러한 변경 사항을 추적하기 시작한 ElastiCache 이후 에 대한 AWS 관리형 정책 업데이트에 대한 세부 정보를 봅니다. 이 페이지의 변경 사항에 대한 자동 알림을 받으려면 ElastiCache 문서 기록 페이지에서 RSS 피드를 구독하세요.

변경 사항	설명	날짜
AmazonElastiCacheFullAccess - 기존 정책에 대한 업데이트	ElastiCache 는 서버리스 캐시를 관리하고 콘솔을 통해 모든 서비스 기능을 사용할 수 있는 새로운 권한을 추가했습니다.	2023년 11월 27일
ElastiCacheServiceRolePolicy -기존 정책 업데이트	ElastiCache 에 서버리스 캐시 리소스에 대한 VPC 엔드포인트 관리를 허용하는 새 권한이 추가되었습니다.	2023년 11월 27일
ElastiCache 변경 사항 추적 시작	ElastiCache 는 AWS 관리형 정책에 대한 변경 사항 추적을 시작했습니다.	2020년 2월 7일

Amazon에 대한 자격 증명 기반 정책(IAM 정책) 사용 ElastiCache

이 주제에서는 계정 관리자가 권한 정책을 자격 증명(즉, 사용자, 그룹 및 역할)에 연결할 수 있는 IAM 자격 증명 기반 정책의 예를 제공합니다.

> **Important**
>
> Amazon ElastiCache 리소스에 대한 액세스를 관리하기 위한 기본 개념 및 옵션을 설명하는 주제를 먼저 읽는 것이 좋습니다. 자세한 내용은 ElastiCache 리소스에 대한 액세스 권한 관리 개요 단원을 참조하십시오.

이 주제의 섹션에서는 다음 내용을 학습합니다.

- AWS Amazon용 관리형 정책 ElastiCache
- 고객 관리형 정책 예제

다음은 Redis 사용 시 권한 정책의 예입니다OSS.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "AllowClusterPermissions",
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateServerlessCache",
                "elasticache:CreateCacheCluster",
                "elasticache:DescribeServerlessCaches",
                "elasticache:DescribeReplicationGroups",
                "elasticache:DescribeCacheClusters",
                "elasticache:ModifyServerlessCache",
                "elasticache:ModifyReplicationGroup",
                "elasticache:ModifyCacheCluster"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowUserToPassRole",
            "Effect": "Allow",
            "Action": [ "iam:PassRole" ],
            "Resource": "arn:aws:iam::123456789012:role/EC2-roles-for-cluster"
        }
    ]
}
```

다음은 Memcached 사용 시 권한 정책의 예를 보여줍니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "AllowClusterPermissions",
        "Effect": "Allow",
        "Action": [
            "elasticache:CreateServerlessCache",
            "elasticache:CreateCacheCluster",
            "elasticache:DescribeServerlessCaches",
            "elasticache:DescribeCacheClusters",
            "elasticache:ModifyServerlessCache",
            "elasticache:ModifyCacheCluster"
        ],
        "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 문은 Amazon ElastiCache 작업(`elasticache:Create*`, `elasticache:Describe*`, `elasticache:Modify*`)에 대한 권한을 부여합니다.
- 두 번째 문은 `Resource` 값 끝에 지정된 IAM 역할 이름에 작업IAM(`iam:PassRole`)에 대한 권한을 부여합니다.

ID 기반 정책에서는 권한을 가질 보안 주체를 지정하지 않으므로 이 정책은 `Principal` 요소를 지정하지 않습니다. 정책을 사용자에게 연결할 경우, 사용자는 암시적인 보안 주체입니다. 권한 정책을 IAM 역할에 연결하면 역할의 신뢰 정책에 식별된 보안 주체가 권한을 얻습니다.

모든 Amazon ElastiCache API 작업과 해당 작업이 적용되는 리소스를 보여주는 표는 섹션을 참조하세요ElastiCache API 권한: 작업, 리소스 및 조건 참조.

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: 사용자가 직접 호출할 수 있도록 허용 IAM CreateServiceLinkedRole API 섹션을 참조하세요). 또는 ElastiCache 서비스 연결 역할을 생성했어야 합니다.

Amazon ElastiCache 콘솔을 사용하는 데 필요한 최소 권한과 결합하면 이 섹션의 정책 예제가 추가 권한을 부여합니다. 이 예제는 및 와 AWS SDKs도 관련이 있습니다 AWS CLI.

IAM 사용자 및 그룹 설정에 대한 지침은 IAM 사용 설명서의 첫 번째 IAM 사용자 및 관리자 그룹 생성을 참조하세요.

> **Important**
>
> 프로덕션에서 사용하기 전에 항상 IAM 정책을 철저히 테스트하세요. 단순해 보이는 일부 ElastiCache 작업은 ElastiCache 콘솔을 사용할 때 지원하는 다른 작업이 필요

할 수 있습니다. 예를 들어 는 ElastiCache 캐시 클러스터를 생성할 수 있는 권한을 `elasticache:CreateCacheCluster` 부여합니다. 그러나 이 작업을 수행하기 위해 ElastiCache 콘솔은 여러 `Describe` 및 `List` 작업을 사용하여 콘솔 목록을 채웁니다.

예시

- 예제 1: ElastiCache 리소스에 대한 사용자 읽기 전용 액세스 허용
- 예제 2: 사용자가 일반적인 ElastiCache 시스템 관리자 작업을 수행하도록 허용
- 예제 3: 사용자가 모든 ElastiCache API 작업에 액세스하도록 허용
- 예제 4: 사용자가 직접 호출할 수 있도록 허용 IAM CreateServiceLinkedRole API
- 예제 5: 사용자가 IAM 인증을 사용하여 서버리스 캐시에 연결하도록 허용

예제 1: ElastiCache 리소스에 대한 사용자 읽기 전용 액세스 허용

다음 정책은 사용자가 리소스를 나열할 수 있는 권한 ElastiCache 작업을 부여합니다. 일반적으로 이 유형의 권한 정책을 관리자 그룹에 연결합니다.

```
{
   "Version": "2012-10-17",
   "Statement":[{
      "Sid": "ECReadOnly",
      "Effect":"Allow",
      "Action": [
          "elasticache:Describe*",
          "elasticache:List*"],
      "Resource":"*"
      }
   ]
}
```

예제 2: 사용자가 일반적인 ElastiCache 시스템 관리자 작업을 수행하도록 허용

일반적인 시스템 관리자 작업으로는 리소스 수정이 있습니다. 시스템 관리자는 ElastiCache 이벤트에 대한 정보를 얻고자 할 수도 있습니다. 다음 정책은 이러한 일반적인 시스템 관리자 작업에 대한 ElastiCache 작업을 수행할 수 있는 권한을 사용자에게 부여합니다. 일반적으로 이 유형의 권한 정책을 시스템 관리자 그룹에 연결합니다.

```
{
```

```
   "Version": "2012-10-17",
   "Statement":[{
      "Sid": "ECAllowMutations",
      "Effect":"Allow",
      "Action":[
          "elasticache:Modify*",
          "elasticache:Describe*",
          "elasticache:ResetCacheParameterGroup"
      ],
      "Resource":"*"
      }
   ]
}
```

예제 3: 사용자가 모든 ElastiCache API 작업에 액세스하도록 허용

다음 정책은 사용자가 모든 ElastiCache 작업에 액세스할 수 있도록 허용합니다. 관리자 사용자에게만 이 유형의 권한 정책을 부여하는 것이 좋습니다.

```
{
   "Version": "2012-10-17",
   "Statement":[{
      "Sid": "ECAllowAll",
      "Effect":"Allow",
      "Action":[
          "elasticache:*"
      ],
      "Resource":"*"
      }
   ]
}
```

예제 4: 사용자가 직접 호출할 수 있도록 허용 IAM CreateServiceLinkedRole API

다음 정책은 사용자가 IAM `CreateServiceLinkedRole` 를 호출할 수 있도록 허용합니다API. 변형 ElastiCache 작업을 호출하는 사용자에게 이러한 유형의 권한 정책을 부여하는 것이 좋습니다.

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"CreateSLRAllows",
```

```
      "Effect":"Allow",
      "Action":[
        "iam:CreateServiceLinkedRole"
      ],
      "Resource":"*",
      "Condition":{
        "StringLike":{
          "iam:AWS ServiceName":"elasticache.amazonaws.com"
        }
      }
    }
  ]
}
```

예제 5: 사용자가 IAM 인증을 사용하여 서버리스 캐시에 연결하도록 허용

다음 정책은 모든 사용자가 2023-04-01에서 2023-06-30 사이의 IAM 인증을 사용하여 서버리스 캐시에 연결할 수 있도록 허용합니다.

```
{
  "Version" : "2012-10-17",
  "Statement" :
  [
    {
      "Effect" : "Allow",
      "Action" : ["elasticache:Connect"],
      "Resource" : [
        "arn:aws:elasticache:us-east-1:123456789012:serverlesscache:*"
      ],
      "Condition": {
        "DateGreaterThan": {"aws:CurrentTime": "2023-04-01T00:00:00Z"},
        "DateLessThan": {"aws:CurrentTime": "2023-06-30T23:59:59Z"}
      }
    },
    {
      "Effect" : "Allow",
      "Action" : ["elasticache:Connect"],
      "Resource" : [
        "arn:aws:elasticache:us-east-1:123456789012:user:*"
      ]
    }
  ]
}
```

리소스 수준 권한

IAM 정책에 리소스를 지정하여 권한 범위를 제한할 수 있습니다. 많은 ElastiCache API 작업은 작업의 동작에 따라 달라지는 리소스 유형을 지원합니다. 모든 IAM 정책 문은 리소스에서 수행되는 작업에 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. 많은 API 작업의 경우 리소스의 Amazon 리소스 이름(ARN) 또는 여러 리소스와 일치하는 ARN 패턴을 지정하여 사용자가 수정할 수 있는 리소스를 제한할 수 있습니다. 리소스별 권한을 제한하려면 를 기준으로 리소스를 지정합니다ARN.

ElastiCache 리소스 유형 및 해당 의 목록을 보려면 서비스 승인 참조의 Amazon에서 정의한 리소스를 ElastiCache ARNs참조하세요. 각 리소스ARN의 를 지정할 수 있는 작업을 알아보려면 Amazon 에서 정의한 작업을 ElastiCache 참조하세요.

예시

- 예제 1: 사용자가 특정 ElastiCache 리소스 유형에 대한 전체 액세스 허용
- 예제 2: 서버리스 캐시에 대한 사용자 액세스 거부

예제 1: 사용자가 특정 ElastiCache 리소스 유형에 대한 전체 액세스 허용

다음 정책은 서버리스 캐시 유형의 모든 리소스 유형을 명시적으로 허용합니다.

```
{
        "Sid": "Example1",
        "Effect": "Allow",
        "Action": "elasticache:*",
        "Resource": [
             "arn:aws:elasticache:us-east-1:account-id:serverlesscache:*"
        ]
}
```

예제 2: 서버리스 캐시에 대한 사용자 액세스 거부

다음 예제에서는 특정 서버리스 캐시에 대한 액세스를 명시적으로 거부합니다.

```
{
        "Sid": "Example2",
        "Effect": "Deny",
        "Action": "elasticache:*",
        "Resource": [
```

```
            "arn:aws:elasticache:us-east-1:account-id:serverlesscache:name"
        ]
}
```

조건 키 사용

IAM 정책이 적용되는 방식을 결정하는 조건을 지정할 수 있습니다. 에서 JSON 정책의 `Condition` 요소를 사용하여 요청 컨텍스트의 키를 정책에 지정한 키 값과 비교할 ElastiCache수 있습니다. 자세한 내용은 IAM JSON 정책 요소: 조건 을 참조하세요.

ElastiCache 조건 키 목록을 보려면 서비스 승인 참조의 Amazon용 조건 키를 참조하세요 ElastiCache.

글로벌 조건 키의 목록은 AWS 글로벌 조건 컨텍스트 키를 참조하세요.

조건 지정: 조건 키 사용

세분화된 제어를 구현하려면 특정 요청에서 개별 파라미터 세트를 제어하는 조건을 지정하는 IAM 권한 정책을 작성합니다. 그런 다음 IAM 콘솔을 사용하여 생성한 IAM 사용자, 그룹 또는 역할에 정책을 적용합니다.

조건을 적용하려면 IAM 정책 문에 조건 정보를 추가합니다. 다음 예제에서는 생성된 모든 자체 설계 캐시 클러스터가 `cache.r5.large` 노드 유형이 되도록 조건을 지정합니다.

다음은 Valkey 또는 Redis를 사용할 때 이 권한 정책의 예를 보여줍니다OSS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*",
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "StringEquals": {
                    "elasticache:CacheNodeType": [
                        "cache.r5.large"
                    ]
                }
            }
        }
    ]
}
```

다음은 Memcached를 사용할 때 이 권한 정책의 예를 보여줍니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*"
            ],
            "Condition": {
                "StringEquals": {
```

```
                    "elasticache:CacheNodeType": [
                        "cache.r5.large"
                    ]
                }
            }
        }
    ]
}
```

자세한 내용은 ElastiCache 리소스 태그 지정 단원을 참조하십시오.

정책 조건 연산자 사용에 대한 자세한 내용은 ElastiCache API 권한: 작업, 리소스 및 조건 참조 섹션을 참조하세요.

정책 예: 조건을 사용하여 세부적인 파라미터 제어 구현

이 섹션에서는 이전에 나열된 ElastiCache 파라미터에 대해 세분화된 액세스 제어를 구현하기 위한 정책의 예를 보여줍니다.

1. elasticache:MaximumDataStorage: 서버리스 캐시의 최대 데이터 스토리지를 지정합니다. 고객은 제공된 조건을 사용할 때 특정 양의 데이터보다 많이 저장할 수 있는 캐시는 생성할 수 없습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowDependentResources",
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateServerlessCache"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:serverlesscachesnapshot:*",
                "arn:aws:elasticache:*:*:snapshot:*",
                "arn:aws:elasticache:*:*:usergroup:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateServerlessCache"
            ],
            "Resource": [
```

```
                "arn:aws:elasticache:*:*:serverlesscache:*"
            ],
            "Condition": {
                "NumericLessThanEquals": {
                    "elasticache:MaximumDataStorage": "30"
                },
                "StringEquals": {
                    "elasticache:DataStorageUnit": "GB"
                }
            }
        }
    ]
}
```

2. elasticache:M aximumECPUPerSecond : 서버리스 캐시의 초ECPU당 최대값을 지정합니다. 제공된 조건을 사용하면 고객은 특정 초ECPUs당 수를 초과하여 실행할 수 있는 캐시를 생성할 수 없습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowDependentResources",
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateServerlessCache"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:serverlesscachesnapshot:*",
                "arn:aws:elasticache:*:*:snapshot:*",
                "arn:aws:elasticache:*:*:usergroup:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateServerlessCache"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:serverlesscache:*"
            ],
            "Condition": {
                "NumericLessThanEquals": {
```

```
                    "elasticache:MaximumECPUPerSecond": "100000"
                }
            }
        }
    ]
}
```

3. elasticache:CacheNodeType: 사용자가 생성할 수 있는 NodeType(s)를 지정합니다. 제공된 조건을 사용하여 고객은 노드 유형에 대해 단일 또는 범위 값을 지정할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*",
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "StringEquals": {
                    "elasticache:CacheNodeType": [
                        "cache.t2.micro",
                        "cache.t2.medium"
                    ]
                }
            }
```

```
        }
    ]
}
```

4. elasticache:CacheNodeType: Memcached를 사용하여 사용자가 생성할 수 있는 NodeType(s)를 지정합니다. 제공된 조건을 사용하여 고객은 노드 유형에 대해 단일 또는 범위 값을 지정할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*"
            ],
            "Condition": {
                "StringEquals": {
                    "elasticache:CacheNodeType": [
                        "cache.t2.micro",
                        "cache.t2.medium"
                    ]
                }
            }
        }
    ]
}
```

5. elasticache:NumNodeGroups: 노드 그룹이 20개 미만인 복제 그룹을 생성합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
             "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
             "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "NumericLessThanEquals": {
                    "elasticache:NumNodeGroups": "20"
                }
            }
        }
    ]
}
```

6. elasticache:ReplicasPerNodeGroup: 노드당 복제본을 5~10개로 지정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
```

```
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "NumericGreaterThanEquals": {
                    "elasticache:ReplicasPerNodeGroup": "5"
                },
                "NumericLessThanEquals": {
                    "elasticache:ReplicasPerNodeGroup": "10"
                }
            }
        }
    ]
}
```

7. elasticache:EngineVersion: 엔진 버전 5.0.6의 사용량을 지정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
     {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
           "Effect": "Allow",
```

```
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*",
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "StringEquals": {
                    "elasticache:EngineVersion": "5.0.6"
                }
            }
        }
    ]
}
```

8. elasticache:EngineVersion: Memcached 엔진 버전 1.6.6의 사용량 지정

```
{
    "Version": "2012-10-17",
    "Statement": [
     {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*"
            ],
            "Condition": {
                "StringEquals": {
                    "elasticache:EngineVersion": "1.6.6"
```

```
                }
            }
        }
    ]
}
```

9. elasticache:EngineType: Valkey 또는 Redis OSS 엔진만 사용하여 지정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*",
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "StringEquals": {
                    "elasticache:EngineType": "redis"
                }
            }
        }
    ]
}
```

10. elasticache:AtRestEncryptionEnabled: 암호화가 활성화된 상태에서만 복제 그룹을 생성하도록 지정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [

         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "Bool": {
                    "elasticache:AtRestEncryptionEnabled": "true"
                }
            }
        }
    ]
}
```

11. elasticache:TransitEncryptionEnabled

a. 를 사용하지 TLS 않을 때만 복제 그룹을 생성할 수 있도록 지정하려면 CreateReplicationGroup 작업에 `false` 대한 `elasticache:TransitEncryptionEnabled` 조건 키를 로 설정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "Bool": {
                    "elasticache:TransitEncryptionEnabled": "false"
                }
            }
        }
    ]
}
```

CreateReplicationGroup 작업의 정책`false`에서 `elasticache:TransitEncryptionEnabled` 조건 키가 로 설정된 경우 TLS가 사용되지 않는 경우에만 요청이 허용됩니다(즉, `CreateReplicationGroup` 요청에 로 설정된 `TransitEncryptionEnabled` 파라미터 `true` 또는 로 설정된 `TransitEncryptionMode` 파라미터가 포함되지 않는 경우에만 허용됩니다`required`.

b. 를 사용할 때만 복제 그룹을 생성할 수 있도록 지정하려면 CreateReplicationGroup 작업을 `true` 위한 `elasticache:TransitEncryptionEnabled` 컨디턴 키를 TLS로 설정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "Bool": {
                    "elasticache:TransitEncryptionEnabled": "true"
                }
            }
        }
    ]
}
```

CreateReplicationGroup 작업 정책`true`에서 `elasticache:TransitEncryptionEnabled` 조건 키가 로 설정된 경우 `CreateReplicationGroup` 요청에 로 설정된 `TransitEncryptionEnabled` 파라미터`true`와 `TransitEncryptionMode` 로 설정된 파라미터가 포함된 경우에만 요청이 허용됩니다`required`.

c. 를 TLS 사용할 때만 복제 그룹을 수정할 수 있도록 지정하려면 `ModifyReplicationGroup` 작업을 `elasticache:TransitEncryptionEnabled` `true` 로 설정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:ModifyReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
```

```
            ],
            "Condition": {
                "BoolIfExists": {
                    "elasticache:TransitEncryptionEnabled": "true"
                }
            }
        }
    ]
}
```

ModifyReplicationGroup 작업 정책`true`에서 `elasticache:TransitEncryptionEnabled` 조건 키가 로 설정된 경우 `ModifyReplicationGroup` 요청에 로 설정된 `TransitEncryptionMode` 파라미터가 포함된 경우에만 요청이 허용됩니다`required`. 로 설정된 `TransitEncryptionEnabled` 파라미터도 선택적으로 포함될 `true` 수 있지만 이 경우 를 활성화할 필요는 없습니다TLS.

12elasticache:AutomaticFailoverEnabled: 자동 장애 조치가 활성화된 상태에서만 복제 그룹을 생성하도록 지정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "Bool": {
```

```
                    "elasticache:AutomaticFailoverEnabled": "true"
                }
            }
        }
    ]
}
```

13elasticache:M ultiAZEnabled: 다중 AZ가 비활성화된 상태에서 복제 그룹을 생성할 수 없도록 지정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },
        {
            "Effect": "Deny",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*",
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "Bool": {
                    "elasticache:MultiAZEnabled": "false"
                }
            }
        }
    ]
}
```

14. elasticache:ClusterModeEnabled: 클러스터 모드가 활성화된 상태에서만 복제 그룹을 생성할 수 있도록 지정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "Bool": {
                    "elasticache:ClusterModeEnabled": "true"
                }
            }
        }
    ]
}
```

15. elasticache:AuthTokenEnabled: AUTH 토큰이 활성화된 상태에서만 복제 그룹을 생성할 수 있도록 지정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [

         {
            "Effect": "Allow",
```

```
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*",
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "Bool": {
                    "elasticache:AuthTokenEnabled": "true"
                }
            }
        }
    ]
}
```

16.elasticache:SnapshotRetentionLimit: 스냅샷을 유지할 일수(또는 최소/최대)를 지정합니다. 아래 정책은 최소 30일 동안 백업을 저장하도록 지정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [

         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
```

```
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup",
                "elasticache:CreateServerlessCache"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*",
                "arn:aws:elasticache:*:*:replicationgroup:*",
                "arn:aws:elasticache:*:*:serverlesscache:*"
            ],
            "Condition": {
                "NumericGreaterThanEquals": {
                    "elasticache:SnapshotRetentionLimit": "30"
                }
            }
        }
    ]
}
```

17.elasticache:KmsKeyId: 고객 관리 AWS KMS형 키의 사용량을 지정합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
        "Sid": "AllowDependentResources",
        "Effect": "Allow",
        "Action": [
            "elasticache:CreateServerlessCache"
        ],
        "Resource": [
            "arn:aws:elasticache:*:*:serverlesscachesnapshot:*",
            "arn:aws:elasticache:*:*:snapshot:*",
            "arn:aws:elasticache:*:*:usergroup:*"
        ]
    },
    {
```

```
        "Effect": "Allow",
        "Action": [
            "elasticache:CreateServerlessCache"
        ],
        "Resource": [
            "arn:aws:elasticache:*:*:serverlesscache:*"
        ],
        "Condition": {
            "StringEquals": {
                "elasticache:KmsKeyId": "my-key"
            }
        }
    }
  ]
}
```

18. elasticache:CacheParameterGroupName: 클러스터에 있는 조직의 특정 파라미터를 사용하여 기본이 아닌 파라미터 그룹을 지정합니다. 파라미터 그룹에 대한 이름 지정 패턴을 지정하거나 특정 파라미터 그룹 이름에 대한 블록 삭제를 지정할 수도 있습니다. 다음은 “my-org-param-group”만 사용하는 제약의 예입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [

         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
```

```
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*",
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "StringEquals": {
                    "elasticache:CacheParameterGroupName": "my-org-param-group"
                }
            }
        }
    ]
}
```

19elasticache:CacheParameterGroupName: Memcached를 사용하여 클러스터에 있는 조직의 특정 파라미터가 있는 기본이 아닌 파라미터 그룹을 지정합니다. 파라미터 그룹에 대한 이름 지정 패턴을 지정하거나 특정 파라미터 그룹 이름에 대한 블록 삭제를 지정할 수도 있습니다. 다음은 “my-org-param-group”만 사용하는 제약의 예입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [

         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*"
            ],
            "Condition": {
                "StringEquals": {
```

```
                    "elasticache:CacheParameterGroupName": "my-org-param-group"
                }
            }
        }
    ]
}
```

20elasticache:CreateCacheCluster: 요청 태그Project가 누락되었거나 Dev, QA 또는 와 같지 않은 경우 CreateCacheCluster 작업 거부Prod.

```
{
    "Version": "2012-10-17",
    "Statement": [
          {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*",
                "arn:aws:elasticache:*:*:securitygroup:*",
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ]
        },
        {
            "Effect": "Deny",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*"
            ],
            "Condition": {
                "Null": {
                    "aws:RequestTag/Project": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
```

```
                "elasticache:AddTagsToResource"
            ],
            "Resource": "arn:aws:elasticache:*:*:cluster:*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/Project": [
                        "Dev",
                        "Prod",
                        "QA"
                    ]
                }
            }
        }
    ]
}
```

21. elasticache:CacheNodeType: `CreateCacheCluster` `cacheNodeType` cache.r5.large 또는 cache.r6g.4xlarge 및 태그로 허용`Project=XYZ`.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateCacheCluster",
        "elasticache:CreateReplicationGroup"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:parametergroup:*",
        "arn:aws:elasticache:*:*:subnetgroup:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateCacheCluster"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:cluster:*"
      ],
      "Condition": {
        "StringEqualsIfExists": {
```

```
          "elasticache:CacheNodeType": [
            "cache.r5.large",
            "cache.r6g.4xlarge"
          ]
        },
        "StringEquals": {
          "aws:RequestTag/Project": "XYZ"
        }
      }
    }
  ]
}
```

22elasticache:CacheNodeType: `CreateCacheCluster` `cacheNodeType` cache.r5.large 또는 cache.r6g.4xlarge 및 태그로 허용`Project=XYZ`.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateCacheCluster"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:parametergroup:*",
        "arn:aws:elasticache:*:*:subnetgroup:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateCacheCluster"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:cluster:*"
      ],
      "Condition": {
        "StringEqualsIfExists": {
          "elasticache:CacheNodeType": [
            "cache.r5.large",
            "cache.r6g.4xlarge"
          ]
```

```
        },
        "StringEquals": {
          "aws:RequestTag/Project": "XYZ"
        }
      }
    }
  ]
}
```

> **Note**
>
> 태그와 다른 조건 키를 함께 적용하기 위한 정책을 생성할 때 `--tags` 파라미터가 있는 생성 요청에 대한 별도의 `elasticache:AddTagsToResource` 정책 요구 사항으로 인해 조건 키 요소에서 조건부 `IfExists`가 필요할 수 있습니다.

Amazon용 서비스 연결 역할 사용 ElastiCache

Amazon은 AWS Identity and Access Management (IAM) 서비스 연결 역할 을 ElastiCache 사용합니다. 서비스 연결 역할은 Amazon 과 같은 AWS 서비스에 직접 연결되는 고유한 유형의 IAM 역할입니다 ElastiCache. Amazon ElastiCache 서비스 연결 역할은 Amazon 에서 사전 정의합니다 ElastiCache. 서비스에서 클러스터를 대신하여 AWS 서비스를 호출하기 위해 필요한 모든 권한을 포함합니다.

서비스 연결 역할을 사용하면 필요한 권한을 수동으로 추가할 필요가 없으므로 Amazon을 ElastiCache 더 쉽게 설정할 수 있습니다. 역할은 AWS 이미 계정에 존재하지만 Amazon ElastiCache 사용 사례에 연결되어 있고 사전 정의된 권한이 있습니다. Amazon만 이러한 역할을 수임 ElastiCache 할 수 있으며 이러한 역할만 사전 정의된 권한 정책을 사용할 수 있습니다. 먼저 역할의 관련 리소스를 삭제해야만 역할을 삭제할 수 있습니다. 이렇게 하면 ElastiCache 리소스에 액세스하는 데 필요한 권한을 실수로 제거할 수 없으므로 Amazon 리소스를 보호할 수 있습니다.

서비스 연결 역할을 지원하는 다른 서비스에 대한 자세한 내용은 AWS 를 IAM 참조하고 서비스 연결 역할 열에서 예인 서비스를 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

목차

- 서비스 연결 역할 생성(IAM 콘솔)
- 서비스 연결 역할 생성(IAMCLI)
- 서비스 연결 역할 생성(IAM API)
- Amazon의 서비스 연결 역할에 대한 설명 편집 ElastiCache
  - 서비스 연결 역할 설명 편집(IAM 콘솔)
  - 서비스 연결 역할 설명 편집(IAM CLI)
  - 서비스 연결 역할 설명 편집(IAM API)
- Amazon의 서비스 연결 역할 삭제 ElastiCache
  - 서비스 연결 역할 정리
  - 서비스 연결 역할 삭제(IAM 콘솔)
  - 서비스 연결 역할 삭제(IAM CLI)
  - 서비스 연결 역할 삭제(IAM API)

Amazon에 대한 서비스 연결 역할 권한 ElastiCache

서비스 연결 역할을 생성하는데 필요한 권한

IAM엔터티가 AWS ServiceRoleForElastiCache 서비스 연결 역할을 생성하도록 허용하려면

해당 IAM엔터티에 대한 권한에 다음 정책 문을 추가합니다.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:CreateServiceLinkedRole",
        "iam:PutRolePolicy"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/elasticache.amazonaws.com/AWS
 ServiceRoleForElastiCache*",
    "Condition": {"StringLike": {"iam:AWS ServiceName": "elasticache.amazonaws.com"}}
}
```

IAM엔터티가 AWS ServiceRoleForElastiCache 서비스 연결 역할을 삭제하도록 허용하려면

해당 IAM엔터티에 대한 권한에 다음 정책 문을 추가합니다.

```
{
```

```
    "Effect": "Allow",
    "Action": [
        "iam:DeleteServiceLinkedRole",
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/elasticache.amazonaws.com/AWS
 ServiceRoleForElastiCache*",
    "Condition": {"StringLike": {"iam:AWS ServiceName": "elasticache.amazonaws.com"}}
}
```

또는 AWS 관리형 정책을 사용하여 Amazon 에 대한 전체 액세스를 제공할 수 있습니다 ElastiCache.

서비스 연결 역할 생성(IAM)

IAM 콘솔, CLI또는 를 사용하여 서비스 연결 역할을 생성할 수 있습니다API.

서비스 연결 역할 생성(IAM 콘솔)

IAM 콘솔을 사용하여 서비스 연결 역할을 생성할 수 있습니다.

서비스 연결 역할을 만들려면(콘솔 사용)

1. 에 로그인 AWS Management Console 하고 에서 IAM 콘솔을 엽니다https://console.aws.amazon.com/iam/.
2. IAM 콘솔의 탐색 창에서 역할 을 선택합니다. 그런 다음 [Create new role]을 선택합니다.
3. 신뢰할 수 있는 유형의 엔터티 선택 아래에서 AWS 서비스를 선택합니다.
4. 또는 사용 사례를 볼 서비스를 선택하고 를 선택합니다ElastiCache.
5. 다음: 권한을 선택합니다.
6. 정책 이름에서 이 역할에 `ElastiCacheServiceRolePolicy`가 있는지 확인합니다. 다음: 태그를 선택합니다.
7. 서비스 연결 역할에는 태그가 지원되지 않습니다. 다음: 검토를 선택합니다.
8. (선택 사항) [Role description]에서 새로운 서비스 연결 역할에 대한 설명을 편집합니다.
9. 역할을 검토한 다음 Create role을 선택합니다.

서비스 연결 역할 생성(IAMCLI)

의 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 생성하려면(CLI)

다음 작업을 사용합니다.

```
$ aws iam create-service-linked-role --aws-service-name elasticache.amazonaws.com
```

서비스 연결 역할 생성(IAM API)

IAM API 를 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 생성하려면(API)

를 사용합니다.CreateServiceLinkedRole API 호출. 요청 시 `elasticache.amazonaws.com` 서비스 이름을 지정합니다.

Amazon의 서비스 연결 역할에 대한 설명 편집 ElastiCache

Amazon ElastiCache 에서는 AWS ServiceRoleForElastiCache 서비스 연결 역할을 편집할 수 없습니다. 서비스 링크 역할을 생성한 후에는 다양한 개체가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. 그러나 를 사용하여 역할에 대한 설명을 편집할 수 있습니다IAM.

서비스 연결 역할 설명 편집(IAM 콘솔)

IAM 콘솔을 사용하여 서비스 연결 역할 설명을 편집할 수 있습니다.

서비스 연결 역할의 설명을 편집하려면(콘솔 사용)

1. IAM 콘솔의 탐색 창에서 역할 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)

의 IAM 작업을 사용하여 서비스 연결 역할 설명을 AWS Command Line Interface 편집할 수 있습니다.

서비스 연결 역할에 대한 설명을 변경하려면(CLI)

1. (선택 사항) 역할에 대한 현재 설명을 보려면 IAM 작업에 AWS CLI 를 사용합니다`get-role`.

Example

```
$ aws iam get-role --role-name AWS ServiceRoleForElastiCache
```

가 아닌 역할 이름을 사용하여 CLI 작업이 있는 역할을 ARN참조합니다. 예를 들어 역할에 가 있는 경우 역할을 로 ARN `arn:aws:iam::123456789012:role/myrole`참조하세요**myrole**.

2. 서비스 연결 역할의 설명을 업데이트하려면 IAM 작업 AWS CLI 용 를 사용합니다update-role-description.

   Linux, macOS, Unix의 경우:

```
$ aws iam update-role-description \
    --role-name AWS ServiceRoleForElastiCache \
    --description "new description"
```

   Windows의 경우:

```
$ aws iam update-role-description ^
    --role-name AWS ServiceRoleForElastiCache ^
    --description "new description"
```

서비스 연결 역할 설명 편집(IAM API)

IAM API 를 사용하여 서비스 연결 역할 설명을 편집할 수 있습니다.

서비스 연결 역할에 대한 설명을 변경하려면(API)

1. (선택 사항) 역할에 대한 현재 설명을 보려면 IAM API 작업을 사용합니다.GetRole.

   Example

```
https://iam.amazonaws.com/
   ?Action=GetRole
   &RoleName=AWS ServiceRoleForElastiCache
   &Version=2010-05-08
   &AUTHPARAMS
```

2. 역할의 설명을 업데이트하려면 IAM API 작업을 사용합니다.UpdateRoleDescription.

Example

```
https://iam.amazonaws.com/
   ?Action=UpdateRoleDescription
   &RoleName=AWS ServiceRoleForElastiCache
   &Version=2010-05-08
   &Description="New description"
```

Amazon의 서비스 연결 역할 삭제 ElastiCache

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.

Amazon ElastiCache 은 서비스 연결 역할을 삭제하지 않습니다.

서비스 연결 역할 정리

IAM 를 사용하여 서비스 연결 역할을 삭제하려면 먼저 역할에 연결된 리소스(클러스터 또는 복제 그룹)가 없는지 확인합니다.

서비스 연결 역할에 IAM 콘솔에 활성 세션이 있는지 확인하려면

1. 에 로그인 AWS Management Console 하고 에서 IAM 콘솔을 엽니다https://console.aws.amazon.com/iam/.
2. IAM 콘솔의 탐색 창에서 역할 을 선택합니다. 그런 다음 AWS ServiceRoleForElastiCache 역할의 이름(확인란 아님)을 선택합니다.
3. 선택한 역할의 요약 페이지에서 Access Advisor 탭을 선택합니다.
4. 액세스 관리자(Access Advisor) 탭에서 서비스 연결 역할의 최근 활동을 검토합니다.

필요한 Amazon ElastiCache 리소스를 삭제하려면 AWS ServiceRoleForElastiCache

- 클러스터를 삭제하려면 다음을 참조하세요.
  - 사용 AWS Management Console
  - AWS CLI 를 사용하여 ElastiCache 클러스터 삭제
  - 사용 ElastiCache API
- 복제 그룹을 삭제하려면 다음을 참조하세요.

- 복제 그룹 삭제(콘솔)
- 복제 그룹 삭제(AWS CLI)
- 복제 그룹 삭제(ElastiCache API)

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 Console 하고 에서 IAM 콘솔을 엽니다https://console.aws.amazon.com/iam/.
2. IAM 콘솔의 탐색 창에서 역할 을 선택합니다. 그런 다음 삭제할 역할의 이름이나 행이 아닌 이름 옆에 있는 확인란을 선택합니다.
3. 페이지 상단의 역할 작업에서 역할 삭제를 선택합니다.
4. 확인 대화 상자에서 선택한 각 역할이 서비스에 마지막으로 액세스한 시점을 보여주는 AWS 서비스 마지막 액세스 데이터를 검토합니다. 이를 통해 역할이 현재 활동 중인지를 확인할 수 있습니다. 계속 진행하려면 예, 삭제합니다(Yes, Delete)를 선택하여 삭제할 서비스 연결 역할을 제출합니다.
5. IAM 콘솔 알림을 시청하여 서비스 연결 역할 삭제 진행 상황을 모니터링합니다. IAM 서비스 연결 역할 삭제는 비동기식이므로 삭제를 위해 역할을 제출한 후 삭제 작업이 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)

의 IAM 작업을 사용하여 서비스 연결 역할을 AWS Command Line Interface 삭제할 수 있습니다.

서비스 연결 역할을 삭제하려면(CLI)

1. 삭제할 서비스 연결 역할의 이름을 모르는 경우 다음 명령을 입력합니다. 이 명령은 계정의 역할과 해당 Amazon 리소스 이름(ARNs)을 나열합니다.

   ```
   $ aws iam get-role --role-name role-name
   ```

   가 아닌 역할 이름을 사용하여 CLI 작업이 있는 역할을 ARN참조합니다. 예를 들어 역할에 ARN 이 있는 경우 역할을 라고 `arn:aws:iam::123456789012:role/myrole`합니다**`myrole`**.

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 API 를 사용하여 서비스 연결 역할을 삭제할 수 있습니다.

서비스 연결 역할을 삭제하려면(API)

1. 서비스 연결 롤에 대한 삭제 요청을 제출하려면 DeleteServiceLinkedRole. 요청에서 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 GetServiceLinkedRoleDeletionStatus. 요청에서 를 지정합니다`DeletionTaskId`.

   삭제 태스크는 `NOT_STARTED`, `IN_PROGRESS`, `SUCCEEDED` 또는 `FAILED` 상태일 수 있습니다. 삭제에 실패할 경우 문제를 해결할 수 있도록 실패 이유가 호출에 반환됩니다.

ElastiCache API 권한: 작업, 리소스 및 조건 참조

IAM 정책에 연결할 액세스 제어 및 쓰기 권한 정책을 설정할 때(ID 기반 또는 리소스 기반) 다음 표를 참조로 사용합니다. 이 표에는 각 Amazon ElastiCache API 작업과 작업을 수행할 수 있는 권한을 부여할 수 있는 해당 작업이 나열되어 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. 리소스가 없는 경우 정책의 ARN리소스는 와일드카드(*)입니다.

ElastiCache 정책에서 조건 키를 사용하여 조건을 표현할 수 있습니다. 적용되는 작업 및 리소스 유형과 함께 ElastiCache특정 조건 키 목록을 보려면 섹션을 참조하세요조건 키 사용. AWS전체 키의 전체 목록은 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

> **Note**
>
> 작업을 지정하려면 `elasticache:` 접두사와 API 작업 이름(예:)을 차례로 사용합니다`elasticache:DescribeCacheClusters`.

ElastiCache 작업 목록을 보려면 서비스 승인 참조의 Amazon에서 정의한 작업을 ElastiCache 참조하세요.

Amazon에 대한 규정 준수 검증 ElastiCache

타사 감사자는 , SOC, PCI연준 및 와 같은 여러 규정 준수 프로그램의 일환으로 AWS 서비스의 보안 RAMP및 AWS 규정 준수를 평가합니다HIPAA.

AWS 서비스 가 특정 규정 준수 프로그램의 범위 내에 있는지 알아보려면 AWS 서비스 규정 준수 프로그램 범위 참조하고 관심 있는 규정 준수 프로그램을 선택합니다. 일반 정보는 AWS 규정 준수 프로그램 참조하세요.

를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 내용은 의 보고서 다운로드 AWS Artifact.

를 사용할 때의 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률 및 규정에 따라 AWS 서비스 결정됩니다. 는 규정 준수를 지원하는 다음 리소스를 AWS 제공합니다.

- 보안 및 규정 준수 빠른 시작 안내서 - 이 배포 안내서에서는 아키텍처 고려 사항에 대해 설명하고 보안 및 규정 준수에 중점을 둔 에 기준 환경을 배포 AWS 하기 위한 단계를 제공합니다.

- Amazon Web Services의 HIPAA 보안 및 규정 준수를 위한 아키텍처 HIPAA- 이 백서에서는 기업이 AWS 를 사용하여 적격 애플리케이션을 생성하는 방법을 설명합니다.

> **Note**
>
> 모두 HIPAA 적합한 AWS 서비스 것은 아닙니다. 자세한 내용은 HIPAA 적격 서비스 참조를 참조하세요.

- AWS 규정 준수 리소스 - 이 워크북 및 가이드 모음은 해당 산업 및 위치에 적용될 수 있습니다.
- AWS 고객 규정 준수 가이드 - 규정 준수의 관점에서 공동 책임 모델을 이해합니다. 이 가이드는 여러 프레임워크(미국 국립표준기술연구소(), 결제카드 산업보안표준위원회(NIST), 국제표준화기구(ISO) 포함)의 보안 제어에 대한 지침을 보호하고 AWS 서비스 매핑하기 위한 모범 사례를 요약PCI 합니다.
- AWS Config 개발자 안내서의 규칙을 사용하여 리소스 평가 - 이 AWS Config 서비스는 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가합니다.
- AWS Security Hub - AWS 서비스 에서 보안 상태를 포괄적으로 볼 수 있습니다 AWS. Security Hub 는 보안 제어를 사용하여 AWS 리소스를 평가하고 보안 업계 표준 및 모범 사례에 대한 규정 준수를 확인합니다. 지원되는 서비스 및 제어 목록은 Security Hub 제어 참조를 참조하세요.
- Amazon GuardDuty - 의심스러운 활동 및 악의적인 활동이 있는지 환경을 모니터링하여 AWS 계정, 워크로드, 컨테이너 및 데이터에 대한 잠재적 위협을 AWS 서비스 탐지합니다. 는 특정 규정 준수 프레임워크에서 요구하는 침입 탐지 요구 사항을 충족DSS하여 PCI 와 같은 다양한 규정 준수 요구 사항을 해결하는 데 도움이 될 GuardDuty 수 있습니다.
- AWS Audit Manager - 이를 AWS 서비스 통해 AWS 사용량을 지속적으로 감사하여 규정 및 업계 표준 준수 및 위험을 관리하는 방법을 간소화할 수 있습니다.

추가 정보

AWS 클라우드 규정 준수에 대한 일반 정보는 다음을 참조하세요.

- FIPS 서비스별 엔드포인트
- 의 서비스 업데이트 ElastiCache
- AWS 클라우드 규정 준수
- 공동 책임 모델
- AWS PCI DSS 규정 준수 프로그램

Amazon의 복원력 ElastiCache

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 기반으로 구축됩니다. AWS 리전은 지연 시간이 짧고 처리량이 높으며 중복성이 높은 네트워킹과 연결된 물리적으로 분리되고 격리된 여러 가용 영역을 제공합니다. 가용 영역을 사용하면 중단 없이 가용 영역 간에 자동으로 장애 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 복수 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라 섹션을 참조하세요.

AWS 글로벌 인프라 외에도 Amazon ElastiCache 은 데이터 복원력 및 백업 요구 사항을 지원하는 데 도움이 되는 몇 가지 기능을 제공합니다.

주제

장애 완화

Amazon ElastiCache 구현을 계획할 때는 장애가 애플리케이션 및 데이터에 미치는 영향을 최소화하도록 계획을 세워야 합니다. 이 섹션의 항목은 애플리케이션 및 데이터를 장애로부터 보호하기 위해 취할 수 있는 접근 방식을 다룹니다.

주제

Memcached 실행 시 장애 완화

Memcached 엔진을 실행할 때 장애의 영향을 최소화하기 위한 다음과 같은 옵션이 있습니다. 장애 완화 계획에서 해결할 장애 유형에는 노드 장애와 가용 영역 장애의 두 유형이 있습니다.

노드 장애 완화

서버리스 캐시는 복제된 다중 AZ 아키텍처를 통해 노드 장애를 자동으로 완화하므로 노드 장애가 애플리케이션에 영향을 미치지 않도록 합니다. 자체 설계된 클러스터에서 노드 장애로 인한 영향을 완화하려면 캐시 데이터를 더 많은 노드로 분산합니다. 자체 설계된 클러스터는 복제를 지원하지 않으므로 노드 장애가 발생하면 항상 클러스터에서 일부 데이터가 손실됩니다.

Memcached 클러스터를 생성할 때 1~60개 이상의 노드를 사용하여 특별한 요청으로 생성할 수 있습니다. 더 많은 노드로 데이터를 분할하면 노드에 장애가 발생할 경우 데이터 손실이 줄어듭니다. 예를 들어, 10개의 노드에 데이터를 분할하면 단일 노드는 캐시된 데이터의 약 10%를 저장합니다. 이 경우, 노드 장애로 인해 대체 노드가 생성되고 프로비저닝될 때 대체해야 하는 캐시의 약 10%가 손실됩니다. 동일한 데이터가 3개의 대형 노드에 캐시된 경우 노드의 장애가 발생하면 캐시된 데이터의 약 33%가 손실됩니다.

Memcached 클러스터에서 노드 수를 지정하는 방법에 대한 자세한 내용은 Memcached 클러스터 생성 (콘솔) 섹션을 참조하세요.

가용 영역 장애 완화

서버리스 캐시는 복제된 다중 AZ 아키텍처를 통해 가용 영역 장애를 자동으로 완화하므로 AZ 장애가 애플리케이션에 영향을 미치지 않도록 합니다.

자체 설계된 클러스터에서 가용 영역 장애로 인한 영향을 완화하려면 가능한 한 많은 가용 영역에 노드를 배치합니다. 드물게 AZ 실패가 발생할 경우 다른 에 캐시된 데이터가 아니라 해당 AZ에 캐시된 데이터가 손실됩니다AZs.

그러면 여러 개의 노드가 필요한 이유는 무엇입니까?

내 리전에 가용 영역이 3개만 있는 경우 AZ 장애 시 내 데이터의 약 1/3이 손실되므로 3개가 넘는 노드가 필요한 이유는 무엇입니까?

매우 좋은 질문입니다. 노드와 가용 영역이라는 두 개의 구분된 장애 유형을 완화하려고 시도하고 있습니다. 맞습니다. 가용 영역에 데이터가 분산되어 있으며 영역 중 하나에 장애가 발생하면 보유한 노드 수와 상관없이 해당 AZ에 캐시된 데이터만 손실됩니다. 그러나 노드 장애 시 더 많은 노드가 있으면 손실된 데이터의 비율이 감소됩니다.

클러스터에 포함할 노드 수를 결정하기 위한 "마법 공식"은 없습니다. 데이터 손실의 영향과 장애 가능성 및 비용을 비교하여 자체적인 결론을 내려야 합니다.

Memcached 클러스터에서 노드 수를 지정하는 방법에 대한 자세한 내용은 Memcached 클러스터 생성 (콘솔) 섹션을 참조하세요.

리전 및 가용 영역에 대한 자세한 내용은 리전 및 가용 영역을 참조하세요.

Valkey 또는 Redis 실행 시 장애 완화 OSS

Valkey 또는 Redis OSS 엔진을 실행할 때 노드 또는 가용 영역 장애의 영향을 최소화하기 위한 다음 옵션이 있습니다.

노드 장애 완화

서버리스 캐시는 다중 AZ 아키텍처를 통해 노드 장애를 자동으로 완화하므로 노드 장애가 애플리케이션에 영향을 미치지 않도록 합니다. 개별 노드의 장애를 완화하려면 자체 설계된 클러스터를 적절하게 구성해야 합니다.

자체 설계된 클러스터에 대한 Valkey 또는 Redis OSS 노드 장애의 영향을 완화하기 위해 다음과 같은 옵션이 있습니다.

주제

장애 완화: Valkey 또는 Redis OSS 복제 그룹

Valkey 또는 Redis OSS 복제 그룹은 애플리케이션이 읽고 쓸 수 있는 단일 기본 노드와 읽기 전용 복제본 노드 1~5개로 구성됩니다. 기본 노드에 데이터가 작성될 때마다 읽기 전용 복제본 노드에서도 비동기식으로 업데이트됩니다.

읽기 전용 복제본 장애의 경우

1. ElastiCache 는 실패한 읽기 전용 복제본을 감지합니다.
2. ElastiCache 는 실패한 노드를 오프라인으로 전환합니다.
3. ElastiCache 는 동일한 AZ에서 대체 노드를 시작하고 프로비저닝합니다.
4. 새 노드가 기본 노드와 동기화됩니다.

이 기간 동안 애플리케이션에서는 다른 노드를 사용하여 계속 읽고 쓸 수 있습니다.

Valkey 또는 Redis OSS 다중 AZ

Valkey 또는 Redis OSS 복제 그룹에서 다중 AZ를 활성화할 수 있습니다. 다중 AZ를 활성화했는지 여부와 상관없이 장애가 있는 기본 노드가 자동으로 감지되고 대체됩니다. 다중 AZ가 활성화되었는지 여부에 따라 이러한 작동 방식이 달라집니다.

다중 AZ가 활성화된 경우

1. ElastiCache 는 기본 노드 장애를 감지합니다.
2. ElastiCache 는 복제 지연이 가장 적은 읽기 전용 복제본 노드를 기본 노드로 승격합니다.
3. 다른 복제본이 새 기본 노드와 동기화됩니다.

4. ElastiCache 는 실패한 기본 AZ에서 읽기 전용 복제본을 스핀업합니다.
5. 새 노드가 새로 승격된 기본 노드와 동기화됩니다.

복제본 노드에 장애 조치하는 것은 일반적으로 새 기본 노드를 생성하고 프로비저닝하는 것보다 빠릅니다. 즉, 애플리케이션에서는 다중 AZ가 활성화되지 않은 경우보다 더 빠르게 기본 노드에 대한 쓰기를 재개할 수 있습니다.

자세한 내용은 Valkey 및 Redis와 함께 다중 AZ ElastiCache 를 사용하여 의 가동 중지 시간 최소화 OSS 단원을 참조하십시오.

다중 AZ가 비활성화된 경우

1. ElastiCache 는 기본 장애를 감지합니다.
2. ElastiCache 는 기본 를 오프라인으로 전환합니다.
3. ElastiCache 는 장애가 발생한 기본 노드를 대체하기 위해 새 기본 노드를 생성하고 프로비저닝합니다.
4. ElastiCache 는 새 기본 를 기존 복제본 중 하나와 동기화합니다.
5. 동기화가 완료되면 새 노드가 클러스터의 기본 노드로 작동합니다.

이 프로세스의 1단계에서 4단계까지는 애플리케이션에서는 기본 노드에 쓸 수 없습니다. 그러나 애플리케이션에서는 복제본 노드에서 계속 읽을 수 있습니다.

추가 보호를 위해 다른 가용 영역()에서 복제 그룹의 노드를 시작하는 것이 좋습니다AZs. 이렇게 할 경우 AZ 장애는 해당 AZ의 노드에만 영향을 주며 다른 노드에는 영향을 주지 않습니다.

자세한 내용은 고가용성을 위한 복제 그룹 사용 단원을 참조하십시오.

가용 영역 장애 완화

서버리스 캐시는 복제된 다중 AZ 아키텍처를 통해 가용 영역 장애를 자동으로 완화하므로 AZ 장애가 애플리케이션에 영향을 미치지 않도록 합니다.

자체 설계된 클러스터에서 가용 영역 장애로 인한 영향을 완화하려면 가능한 한 많은 가용 영역에 샤드별 노드를 배치합니다.

샤드에 보유한 노드의 수와 상관없이 동일한 가용 영역에 노드가 모두 배치된 경우, 해당 AZ에 치명적인 장애가 발생하면 모든 샤드 데이터가 손실됩니다. 그러나 여러 에서 노드를 찾으면 AZ가 AZs실패하면 해당 AZ의 노드만 손실됩니다.

읽기 작업이 이제 더 적은 수의 노드에서 공유되므로 노드가 손실될 때마다 성능 저하를 경험할 수 있습니다. 노드가 대체될 때까지 이 성능 저하가 계속됩니다.

Valkey 또는 Redis OSS 노드의 가용 영역을 지정하는 방법에 대한 자세한 내용은 섹션을 참조하세요Valkey(클러스터 모드 비활성화됨) 클러스터 생성(콘솔).

리전 및 가용 영역에 대한 자세한 내용은 에 대한 리전 및 가용 영역 선택 ElastiCache 섹션을 참조하세요.

추천

추가 구성 없이 내결함성이 자동으로 향상되므로 자체 설계된 클러스터에서 서버리스 캐시를 생성하는 것이 좋습니다. 하지만 자체 설계된 클러스터를 생성할 때 계획해야 할 장애의 유형에는 개별 노드 장애와 광범위한 가용 영역 장애라는 2가지 유형이 있습니다. 가장 좋은 장애 완화 계획은 두 유형의 장애를 모두 해결하는 것입니다.

노드 장애로 인한 영향 최소화

Valkey 또는 Redis를 사용할 때 노드 장애의 영향을 최소화하려면 구현 시 각 샤드에 여러 노드를 사용하고 여러 가용 영역에 노드를 배포하는 것이 OSS좋습니다. 이 작업은 서버리스 캐시에서 자동으로 수행됩니다.

Valkey 또는 Redis 에서 자체 설계된 클러스터의 경우 기본 노드가 실패할 경우 ElastiCache 가 복제본에 자동으로 장애 조치되도록 복제 그룹에서 다중 AZ를 활성화하는 OSS것이 좋습니다.

Memcached를 실행하고 노드에 데이터를 분할할 때 더 많은 노드를 사용할수록 노드에 장애가 발생할 때 데이터 손실이 줄어듭니다.

가용 영역 장애의 영향 최소화

가용 영역 장애의 영향을 최소화하려면 가능한 한 여러 개의 서로 다른 가용 영역에서 노드를 시작하는 것이 좋습니다. 노드를 전체적으로 균등하게 분산AZs하면 AZ 실패가 발생할 가능성이 거의 없는 경우 영향을 최소화할 수 있습니다. 이 작업은 서버리스 캐시에서 자동으로 수행됩니다.

기타 주의 사항

Valkey 또는 Redis 를 실행하는 경우 위의 항목 OSS외에도 클러스터의 정기 백업을 예약하는 것이 좋습니다. 백업(스냅샷)은 장애 또는 손상이 발생할 경우 캐시를 복원하는 데 사용할 수 있는 .rdb 파일을 생성합니다. 자세한 내용은 스냅샷 및 복원 단원을 참조하십시오.

의 인프라 보안 AWS ElastiCache

관리형 서비스로서 는 AWS 아키텍처 센터 의 보안 및 규정 준수 섹션에 설명된 AWS 글로벌 네트워크 보안 절차에 의해 보호 AWS ElastiCache 됩니다.

AWS 게시된 API 호출을 사용하여 네트워크를 ElastiCache 통해 에 액세스합니다. 클라이언트는 전송 계층 보안(TLS) 1.2 이상을 지원해야 합니다. TLS 1.3 이상을 사용하는 것이 좋습니다. 또한 클라이언트는 Ephemeral Diffie-Hellman(PFS) 또는 Elliptic Curve Ephemeral Diffie-Hellman()과 같은 완벽한 순방향 보안(DHE)을 갖춘 암호 제품군을 지원해야 합니다ECDHE. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 액세스 키 ID와 IAM 보안 주체와 연결된 보안 액세스 키를 사용하여 요청에 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 보안 인증을 생성하여 요청에 서명할 수 있습니다.

의 서비스 업데이트 ElastiCache

ElastiCache 는 캐시, 클러스터 및 노드 플릿을 자동으로 모니터링하여 서비스 업데이트가 제공되면 이를 적용합니다. 서버리스 캐시에 대한 서비스 업데이트는 투명한 방식으로 자동 적용됩니다. 자체 설계된 클러스터의 경우 가 이러한 업데이트를 적용할 ElastiCache 수 있도록 사전 정의된 유지 관리 기간을 설정합니다. 그러나 경우에 따라 이 접근 방식이 너무 엄격하여 비즈니스 흐름이 제한될 수 있습니다.

서비스 업데이트를 통해 자체 설계된 클러스터에 적용할 업데이트와 시기를 제어할 수 있습니다. 선택한 ElastiCache 클러스터에 대한 이러한 업데이트의 진행 상황을 실시간으로 모니터링할 수도 있습니다.

주제

- 자체 설계된 클러스터에 대한 서비스 업데이트 관리

자체 설계된 클러스터에 대한 서비스 업데이트 관리

ElastiCache 자체 설계된 클러스터에 대한 서비스 업데이트는 정기적으로 릴리스됩니다. 이러한 서비스 업데이트에 대해 하나 이상의 적격 자체 설계 클러스터가 있는 경우 업데이트가 릴리스될 때 이메일, SNS, Personal Health Dashboard(PHD) 및 Amazon CloudWatch 이벤트를 통해 알림을 받게 됩니다. 업데이트는 ElastiCache 콘솔의 서비스 업데이트 페이지에도 표시됩니다. 이 대시보드를 사용하면 ElastiCache 플릿에 대한 모든 서비스 업데이트와 상태를 볼 수 있습니다. 서버리스 캐시에 대한 서비스 업데이트는 투명하게 적용되며 서비스 업데이트를 통해 관리할 수 없습니다.

자동 업데이트가 시작되기 전에 업데이트 시작 전에 업데이트 적용 시기를 제어합니다. ElastiCache 클러스터에 항상 up-to-date 최신 보안 패치가 적용되도록 가능한 한 빨리 유형 보안 업데이트 업데이트를 적용하는 것이 좋습니다.

다음 섹션에서는 다음과 같은 옵션에 대해 상세히 알아봅니다.

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 활성화된 경우 사용 가능해질 때 조치를 취하지 않도록 선택할 수 있습니다. ElastiCache 는 자동 업데이트 시작 날짜 이후 클러스터의 예정된 유지 관리 기간 중 하나에 업데이트를 적용하도록 예약할 것입니다. 업데이트의 각 단계에 대한 관련 알림을 받게 됩니다.

> **Note**
>
> 사용 가능(Available) 또는 예약됨(Scheduled) 상태인 서비스 업데이트만 적용할 수 있습니다.

서비스별 업데이트를 검토하고 해당 ElastiCache 클러스터에 적용하는 방법에 대한 자세한 내용은 섹션을 참조하세요콘솔을 사용한 서비스 업데이트 적용.

하나 이상의 ElastiCache 클러스터에서 새 서비스 업데이트를 사용할 수 있는 경우 ElastiCache 콘솔, API또는 를 사용하여 업데이트를 AWS CLI 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. 에 로그인 AWS Management Console 하고 에서 Amazon ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 탐색 창에서 서비스 업데이트(Service Updates)를 선택합니다.
3. 서비스 업데이트(Service Updates)에서 다음 내용을 볼 수 있습니다.

   - 서비스 업데이트 이름(Service update name): 서비스 업데이트의 고유 이름입니다.
   - 업데이트 유형(Update type): 보안 업데이트(security-update) 또는 엔진 업데이트(engine-update) 중 하나에 해당하는 서비스 업데이트 유형

- 업데이트 심각도: 업데이트 적용의 우선순위를 나타냅니다.
  - 중요: 이 업데이트를 즉시 적용하는 것이 좋습니다(14일 이내).
  - 중요: 비즈니스 흐름이 허용되는 즉시 이 업데이트를 적용하는 것이 좋습니다(30일 이내) .
  - 보통: 가능한 한 빨리 이 업데이트를 적용하는 것이 좋습니다(60일 이내).
  - 낮음: 가능한 한 빨리 이 업데이트를 적용하는 것이 좋습니다(90일 이내).
- 엔진 버전(Engine version): 업데이트 유형이 엔진 업데이트인 경우 업데이트되는 엔진 버전입니다.
- 릴리스 날짜: 업데이트가 릴리스되어 클러스터에 적용할 수 있게 된 날짜입니다.
- 권장 적용기한: ElastiCache 업데이트를 적용하기 위한 지침 날짜입니다.
- 상태: 업데이트의 상태로, 다음 중 하나에 해당합니다.
  - 사용 가능: 필요한 클러스터에 업데이트를 사용할 수 있습니다.
  - 완료(Complete): 업데이트가 적용되었습니다.
  - 취소됨: 업데이트가 취소되었으며 더 이상 필요하지 않습니다.
  - 만료됨: 업데이트를 더 이상 적용할 수 없습니다.

4. 서비스 업데이트의 세부 정보를 보려면 개별 업데이트(왼쪽에 있는 버튼이 아님)를 선택합니다.

   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 elasticache describe-service-updates --service-update-status
  available
  ```

  자세한 내용은 describe-service-updates를 참조하세요.
- 클러스터 목록에 서비스 업데이트를 적용하려면 다음 명령을 실행합니다.

  ```
  aws elasticache batch-apply-update-action --service-update
  ServiceUpdateNameToApply=sample-service-update --cluster-names cluster-1
  cluster2
  ```

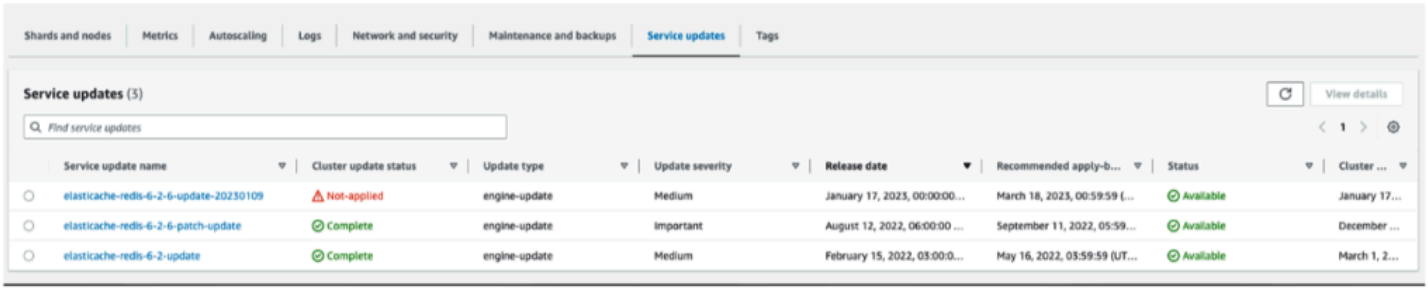
  자세한 내용은 batch-apply-update-action를 참조하세요.

AWS 콘솔을 사용하여 적용된 최신 서비스 업데이트가 있는지 확인

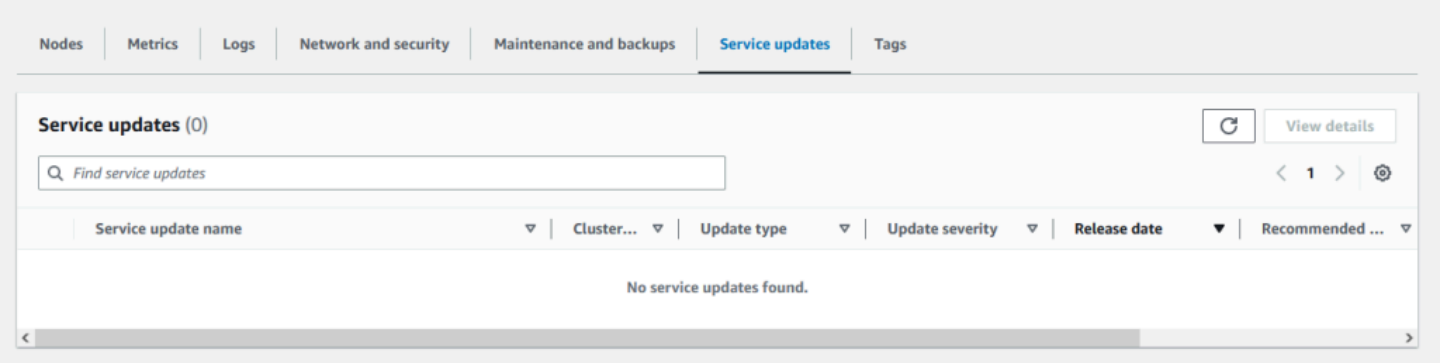
다음 단계에 따라 ElastiCache (Redis OSS) 클러스터가 최신 서비스 업데이트를 실행 중인지 확인할 수 있습니다.

1. Redis 클러스터 페이지에서 적용 가능한 OSS 클러스터를 선택합니다.
2. 탐색 창에서 서비스 업데이트를 선택하여 해당 클러스터에 해당하는 서비스 업데이트가 있는 경우 해당 업데이트를 확인합니다.

콘솔에 서비스 업데이트 목록이 표시되면 서비스 업데이트를 선택하고 지금 적용을 선택할 수 있습니다.



콘솔에 “서비스 업데이트를 찾을 수 없음”이 표시되면 ElastiCache (Redis OSS) 클러스터에 이미 최신 서비스 업데이트가 적용되었음을 의미합니다.



서비스 업데이트 중지

필요에 따라 클러스터에 대한 업데이트를 중지할 수 있습니다. 예를 들어 업데이트가 진행 중인 클러스터가 예기치 않게 급증하는 경우 업데이트를 중지할 수 있습니다. 또는 업데이트가 너무 오래 걸리고 피크 시간에 비즈니스 흐름을 방해하는 경우 업데이트를 중지할 수 있습니다.

중지 중 작업은 이들 노드와 아직 업데이트되지 않은 노드에 대한 모든 업데이트를 즉시 중지합니다. 진행 중 상태인 노드는 완료될 때까지 계속됩니다. 그러나, 업데이트 사용 가능 상태인 동일한 클러스터 내의 다른 노드에 대한 업데이트는 중단되며 중지 중 상태로 변경됩니다.

중지 중 워크플로가 완료되면 중지 중 상태인 노드는 중지됨 상태가 됩니다. 업데이트의 워크플로에 따라 일부 클러스터의 노드는 업데이트되지 않습니다. 다른 클러스터에는 업데이트된 노드와 여전히 업데이트 사용 가능 상태인 노드가 포함될 수 있습니다.

비즈니스 흐름 상 가능할 때 업데이트 프로세스를 완료할 수 있습니다. 이 경우, 업데이트를 완료할 해당 클러스터를 선택한 다음 지금 적용을 선택하세요. 자세한 내용은 서비스 업데이트 적용 단원을 참조하십시오.

콘솔 사용

ElastiCache 콘솔을 사용하여 서비스 업데이트를 중단할 수 있습니다. 다음에서는 이 작업을 수행하는 방법을 보여 줍니다.

- 선택한 클러스터에서 서비스 업데이트가 진행되면 ElastiCache 콘솔의 ElastiCache 대시보드 상단에 업데이트 보기/중지 탭이 표시됩니다.
- 업데이트를 중지하려면 Stop Update(업데이트 중지)를 선택합니다.
- 업데이트를 중지할 경우 클러스터를 선택하고 상태를 확인합니다. 중지 중 상태로 바뀌었다가 최종적으로 중지됨 상태가 됩니다.

사용 AWS CLI

AWS CLI를 사용해 서비스 업데이트를 중지할 수 있습니다. 다음 코드 예제에서는 이를 수행하는 방법을 보여줍니다.

복제 그룹의 경우 다음과 같이 합니다.

```
aws elasticache batch-stop-update-action --service-update-name sample-service-update --replication-group-ids my-replication-group-1 my-replication-group-2
```

캐시 클러스터의 경우 다음과 같이 합니다.

```
aws elasticache batch-stop-update-action --service-update-name sample-service-update --cache-cluster-ids my-cache-cluster-1 my-cache-cluster-2
```

자세한 내용은 BatchStopUpdateAction를 참조하세요.

일반적인 취약성 및 노출(CVE): 에서 해결된 보안 취약성 ElastiCache

일반적인 취약성 및 노출(CVE)은 공개적으로 알려진 사이버 보안 취약성에 대한 항목 목록입니다. 각 항목은 식별 번호, 설명 및 하나 이상의 퍼블릭 참조가 포함된 링크입니다. 이 페이지에서 에서 해결된 보안 취약성 목록을 찾을 수 있습니다 ElastiCache.

알려진 취약성으로부터 보호하려면 항상 최신 ElastiCache Valkey, Redis OSS 또는 ElastiCache Memcached 버전으로 업그레이드하는 것이 좋습니다. ElastiCache 서버리스 캐시를 작동할 때 CVE 수정 사항이 캐시에 자동으로 적용됩니다. Valkey 또는 Redis 를 사용하여 자체 설계된 클러스터를 작동할 때 는 PATCH 구성 요소를 OSS ElastiCache 노출합니다. 예를 들어 ElastiCache (Redis OSS) 버전 6.2.6을 사용할 때 메이저 버전은 6이고 마이너 버전은 2이고 패치 버전은 6입니다. PATCH 버전은 백워드 호환 버그 수정, 보안 수정 및 비기능적 변경에 사용됩니다.

다음 표를 사용하여 특정 버전의 ElastiCache Valkey 및 RedisOSS에 특정 보안 취약성에 대한 수정 사항이 있는지 확인할 수 있습니다. ElastiCache Valkey 또는 Redis OSS 클러스터가 보안 수정 없이 버전을 실행하는 경우 아래 표를 참조하고 조치를 취하세요. 수정 사항이 포함된 최신 ElastiCache Valkey 또는 Redis OSS 버전으로 업그레이드하거나 수정 사항이 포함된 버전을 사용하는 경우 를 참조하여 최신 서비스 업데이트를 적용해야 합니다자체 설계된 클러스터에 대한 서비스 업데이트 관리. 지원되는 ElastiCache 엔진 버전과 업그레이드 방법에 대한 자세한 내용은 섹션을 참조하세요의 엔진 버전 및 업그레이드 ElastiCache.

Note

- CVE 가 ElastiCache 버전에서 처리되는 경우 이는 최신 버전에서도 처리된다는 의미입니다. 예를 들어 취약성이 ElastiCache (Redis OSS) 버전 6.0.5에서 해결된 경우 버전 6.2.6, 7.0.7 및 7.1에 대해 이 작업이 계속됩니다.
- 다음 표의 별표(*)는 보안 취약성을 해결하려면 지정된 ElastiCache (Redis OSS) 버전을 실행하는 ElastiCache (Redis OSS) 클러스터에 최신 서비스 업데이트가 적용되어야 함을 나타냅니다. 클러스터가 실행 중인 ElastiCache (Redis OSS) 버전에 최신 서비스 업데이트가 적용되었는지 확인하는 방법에 대한 자세한 내용은 섹션을 참조하세요자체 설계된 클러스터에 대한 서비스 업데이트 관리.

ElastiCache (Redis OSS) 버전	CVEs 주소 지정됨
Redis OSS 6.0.5	CVE-2022-24735*, CVE-2022-24736*
Redis OSS 6.2.6	CVE-2022-24834*, CVE-2022-35977*, CVE-2022-36021*, CVE-2022-24735, CVE-2022-24736
Redis OSS 7.0.7	CVE-2023-41056*, CVE-2022-24834*, CVE-2022-35977, CVE-2022-36021, CVE-2022-24735, CVE-2022-24736
Redis OSS 7.1.0	CVE-2023-41056, CVE-2022-24834, CVE-2022-35977, CVE-2022-36021, CVE-2022-24735, CVE-2022-24736

Amazon에서 로깅 및 모니터링 ElastiCache

캐시를 관리하려면 캐시의 작동 방식을 이해해야 합니다. ElastiCache 는 캐시 성능을 모니터링하기 위해 Amazon CloudWatch Logs에 게시되는 지표를 생성합니다. 또한 캐시 리소스에 중대한 변경 사항이 발생할 때 이벤트를 ElastiCache 생성합니다(예: 새 캐시가 생성되거나 캐시가 삭제됨).

주제

- 를 사용하여 Amazon ElastiCache API 통화 로깅 AWS CloudTrail

Valkey 및 Redis에 대한 서버리스 지표 및 이벤트 OSS

ElastiCache 는 서버리스 캐시로 작업할 때 모니터링할 수 있는 다양한 지표와 이벤트를 제공합니다. 여기에는 Amazon 를 통해 수집할 수 있는 CloudWatch 지표, 명령 수준 지표 및 이벤트 로그가 포함됩니다 EventBridge.

주제

서버리스 캐시 지표

`AWS/ElastiCache` 네임스페이스에는 Valkey 또는 Redis OSS 서버리스 캐시에 대한 다음 CloudWatch 지표가 포함됩니다.

Valkey 또는 Redis의 지표 코드 OSS

지표	설명	단위
BytesUsedForCache	캐시에 저장된 데이터에서 사용되는 총 바이트 수	바이트
ElastiCacheProcessingUnits	캐시에서 실행된 요청에 사용된 총 ElastiCacheProcessingUnits (ECPUs) 수	개수
SuccessfulReadRequestLatency	성공적인 읽기 요청의 지연 시간	마이크로초
SuccessfulWriteRequestLatency	성공적인 쓰기 요청의 지연 시간	마이크로초
TotalCmdsCount	캐시에서 실행된 모든 명령의 총 개수	개수
CacheHitRate	캐시의 일치율 이는 `cache_hits` 및 `cache_misses` 통계를 사용하여 다음과 같은 방식으로 계산됩니다. `cache_hits /(cache_hits + cache_misses)`	%
CacheHits	캐시의 성공한 읽기 전용 키 조회수	개수
CurrConnections	캐시에 대한 클라이언트 연결 수	개수
ThrottledCmds	워크로드가 확장할 ElastiCache 수 있는 것보다 더 빠르게 확장되었기 ElastiCache 때문에 에서 제한한 요청 수입니다.	개수
NewConnections	이 기간에 서버에서 허용된 총 연결 수입니다.	개수

지표	설명	단위
CurrItems	캐시 항목 수입니다.	개수
CurrVolatileItems	를 사용하는 캐시의 항목 수입니다TTL.	개수
NetworkBytesIn	캐시로 전송된 총 바이트 수	바이트
NetworkBytesOut	캐시에서 나간 총 바이트 수	바이트
Evictions	캐시에서 제거된 키 수	개수
IamAuthenticationExpirations	만료된 IAM인증된 Valkey 또는 Redis OSS 연결의 총 수입니다. IAM을 사용하는 인증에 대한 자세한 내용은 사용 설명서를 참조하세요.	개수
IamAuthenticationThrottling	제한된 IAM인증된 Valkey 또는 Redis OSS AUTH 또는 HELLO 요청의 총 수입니다. IAM을 사용하는 인증에 대한 자세한 내용은 사용 설명서를 참조하세요.	개수
KeyAuthorizationFailures	사용자가 액세스 권한이 없는 키에 액세스한 실패한 시도의 총 수입니다. 무단 액세스 시도를 감지하려면 이에 대한 경보를 설정하는 것이 좋습니다.	개수
AuthenticationFailures	AUTH 명령을 OSS 사용하여 Valkey 또는 Redis에 인증하려는 총 실패한 시도 횟수입니다. 무단 액세스 시도를 감지하려면 이에 대한 경보를 설정하는 것이 좋습니다.	개수

지표	설명	단위
CommandAuthorizationFailures	사용자가 호출 권한이 없는 명령을 실행한 실패한 시도의 총 수입니다. 무단 액세스 시도를 감지하려면 이에 대한 경보를 설정하는 것이 좋습니다.	개수

명령 수준 지표

ElastiCache 는 또한 다음과 같은 명령 수준 지표를 내보냅니다. 각 명령 유형에 대해 총 명령 수와 해당 명령 유형에서 ECPUs 소비한 수를 ElastiCache 내보냅니다.

지표	설명	단위
EvalBasedCmds	캐시가 수신한 get 명령 수	개수
EvalBasedCmdsECPUs	ECPUs eval 기반 명령에서 사용됩니다.	개수
GeoSpatialBasedCmds	지리 기반 명령의 총 명령 수입니다. 이는 Valkey 또는 Redis OSS 명령 통계에서 파생됩니다. 모든 지리 유형의 명령(예: geoadd, geodist, geohash, geopos, georadius, georadius bymember)을 합산하여 도출됩니다.	개수
GeoSpatialBasedCmdsECPUs	ECPUs 지리 공간 기반 명령에서 사용됩니다.	개수
GetTypeCmds	읽기 전용 유형 명령의 총 수 이는 모든 읽기 전용 형식 OSS 명령(get, hget, scard, lrange 등)을 합산하여 Valkey 또는	개수

지표	설명	단위
	Redis 명령 통계에서 파생됩니다.	
GetTypeCmdsECPUs	ECPUs 읽기 명령에서 사용됩니다.	개수
HashBasedCmds	해시 기반 명령 총 수입니다. 이는 하나 이상의 해시(대상, hkey, hvals, hdel 등)에 대해 작동하는 모든 명령을 합산하여 Valkey 또는 Redis OSS 명령 통계에서 파생됩니다.	개수
HashBasedCmdsECPUs	ECPUs 해시 기반 명령에서 사용됩니다.	개수
HyperLogLogBasedCmds	HyperLogLog기반 명령의 총 수입니다. 이는 모든 pf 유형의 OSS 명령(pfadd, pfcount, pfmerge 등)을 합산하여 Valkey 또는 Redis 명령 통계에서 파생됩니다.	개수
HyperLogLogBasedCmdsECPUs	ECPUs HyperLogLog기반 명령에서 사용됩니다.	개수
JsonBasedCmds	읽기 및 쓰기 JSON 명령을 모두 포함한 총 명령 수입니다. 이는 JSON 키에 작용하는 모든 JSON 명령을 합산하여 Valkey 또는 Redis OSS 명령 통계에서 파생됩니다.	개수
JsonBasedCmdsECPUs	ECPUs 읽기 및 쓰기 JSON 명령을 포함한 모든 명령에서 사용됩니다.	개수

지표	설명	단위
JsonBasedGetCmds	JSON 읽기 전용 명령의 총 수입니다. 이는 JSON 키에 작용하는 모든 JSON 읽기 명령을 합산하여 Valkey 또는 Redis OSS 명령 통계에서 파생됩니다.	개수
JsonBasedGetCmdsECPUs	ECPUs JSON 읽기 전용 명령에서 사용됩니다.	개수
JsonBasedSetCmds	총 JSON 쓰기 명령 수입니다. 이는 JSON 키에 작용하는 모든 JSON 쓰기 명령을 합산하여 Valkey 또는 Redis OSS 명령 통계에서 파생됩니다.	개수
JsonBasedSetCmdsECPUs	ECPUs JSON 쓰기 명령에서 사용됩니다.	개수
KeyBasedCmds	키 기반 명령 총 수입니다. 이는 여러 데이터 구조(del, expire, rename 등)에서 하나 이상의 키에 대해 작동하는 모든 명령을 합산하여 Valkey 또는 Redis OSS 명령 통계에서 파생됩니다.	개수
KeyBasedCmdsECPUs	ECPUs 키 기반 명령에서 사용됩니다.	개수
ListBasedCmds	목록 기반 명령 총 수입니다. 이는 하나 이상의 목록(lindex, lrange, lpush, ltrim 등)에 따라 작동하는 모든 명령을 합산하여 Valkey 또는 Redis OSS 명령 통계에서 파생됩니다.	개수

지표	설명	단위
ListBasedCmdsECPUs	ECPUs 목록 기반 명령에서 사용됩니다.	개수
NonKeyTypeCmds	키 기반이 아닌 명령의 총 수입니다. 이는 acl, dbsize 또는 info와 같이 키에 대해 작동하지 않는 모든 명령을 합산하여 Valkey 또는 Redis OSS 명령 통계에서 파생됩니다.	개수
NonKeyTypeCmdsECPUs	ECPUs non-key-based 명령에 의해 사용됩니다.	개수
PubSubBasedCmds	pub/sub 기능의 명령 총 수입니다. 이는 pub/sub 기능에 사용되는 psubscribe, 게시, pubsub, punsub, ssubscribe, sunsubscribe, spublish, subscription 및 unsub와 같은 모든 명령을 합산하여 Valkey 또는 Redis OSS commandstatsstatistics에서 파생됩니다.	개수
PubSubBasedCmdsECPUs	ECPUs pub/sub 기반 명령에서 사용됩니다.	개수
SetBasedCmds	집합 기반 명령 총 수입니다. 이는 하나 이상의 세트(카드, sdiff, sadd, sunion 등)에 따라 작동하는 모든 명령을 합산하여 Valkey 또는 Redis OSS 명령 통계에서 파생됩니다.	개수
SetBasedCmdsECPUs	ECPUs 세트 기반 명령에서 사용됩니다.	개수

지표	설명	단위
SetTypeCmds	쓰기 유형의 총 명령 수입니다. 이는 데이터에서 작동하는 모든 유형의 명령(세트, hset, sadd, lpop 등)을 합산하여 Valkey 또는 Redis 명령 OSS 통계에서 파생됩니다.	개수
SetTypeCmdsECPUs	ECPUs 쓰기 명령에서 사용됩니다.	개수
SortedSetBasedCmds	정렬된 집합 기반 명령 총 수입니다. 이는 정렬된 세트(zcount, zrange, zrank, zadd 등) 중 하나 이상에 대해 작동하는 모든 명령을 합산하여 Valkey 또는 Redis OSS 명령 통계에서 파생됩니다.	개수
SortedSetBasedCmdsECPUs	ECPUs 정렬된 기반 명령에서 사용됩니다.	개수
StringBasedCmds	문자열 기반 명령 총 수입니다. 이는 하나 이상의 문자열(문자열, setex, setrange 등)에 따라 작동하는 모든 명령을 합산하여 Valkey 또는 Redis OSS 명령 통계에서 파생됩니다.	개수
StringBasedCmdsECPUs	ECPUs 문자열 기반 명령에서 사용됩니다.	개수

지표	설명	단위
StreamBasedCmds	총 스트림 기반 명령 수입니다. 이는 하나 이상의 스트림 데이터 유형(xrange, xlen, xadd, xdel 등)에 따라 작동하는 모든 명령을 합산하여 Valkey 또는 Redis OSS 명령 통계에서 파생됩니다.	개수
StreamBasedCmdsECPUs	ECPUs 스트림 기반 명령에서 사용됩니다.	개수

서버리스 캐시 이벤트

ElastiCache 는 서버리스 캐시와 관련된 이벤트를 기록합니다. 여기에는 이벤트 날짜 및 시간, 이벤트의 원본 이름 및 원본 유형, 이벤트 설명 등의 정보가 포함됩니다. ElastiCache 콘솔, AWS CLI describe-events 명령 또는 작업 을 사용하여 로그에서 이벤트를 쉽게 검색할 수 있습니다 ElastiCache API`DescribeEvents`.

Amazon 를 사용하여 ElastiCache 이벤트를 모니터링, 수집, 변환 및 작업하도록 선택할 수 있습니다 EventBridge. Amazon EventBridge https://docs.aws.amazon.com/eventbridge/최신/사용자 가이드/에서 자세히 알아보세요.

ElastiCache 이벤트 보기(콘솔)

ElastiCache 콘솔을 사용하여 이벤트를 보려면:

1. 에 로그인 AWS Management Console 하고 에서 ElastiCache 콘솔을 엽니다. https://console.aws.amazon.com/elasticache/
2. 사용 가능한 모든 이벤트의 목록을 보려면 탐색 창에서 이벤트를 선택합니다.
3. 이벤트 화면에서 목록의 각 행은 하나의 이벤트를 나타내며 이벤트 소스, 이벤트 유형, 이벤트 GMT 시간 및 이벤트 설명을 표시합니다. [Filter]를 사용하여 이벤트 목록에서 모든 이벤트를 볼지 특정 유형의 이벤트만 볼지를 지정할 수 있습니다.

ElastiCache 이벤트 보기(AWS CLI)

를 사용하여 ElastiCache 이벤트 목록을 생성하려면 describe-events 명령을 AWS CLI사용합니다. 선택적 파라미터를 사용하여 나열된 이벤트의 유형, 나열된 이벤트의 기간, 나열할 이벤트의 최대 수 등을 제어할 수 있습니다.

다음 코드는 최대 40개의 서버리스 캐시 이벤트를 나열합니다.

```
aws elasticache describe-events --source-type serverless-cache --max-items 40
```

다음 코드는 지난 24시간(1,440분) 동안 발생한 서버리스 캐시의 모든 이벤트를 나열합니다.

```
aws elasticache describe-events --source-type serverless-cache --duration 1440
```

서버리스 이벤트

이 섹션에서는 서버리스 캐시에서 수신할 수 있는 다양한 유형의 이벤트를 설명합니다.

서버리스 캐시 생성 이벤트

Detail-Type	설명	단위	소스	메시지
캐시 생성됨	캐시 ARN	생성	serverless-cache	<cache-name> 캐시가 생성되어 바로 사용할 수 있습니다.
캐시 생성됨	캐시 ARN 스냅샷 경과 시간	생성	serverless-cache	<cache-name> 캐시가 생성되고 스냅샷에서 데이터가 복원되었습니다. 캐시를 사용할 준비가 되었습니다.
캐시 생성 실패	캐시 ARN	실패	serverless-cache	<cache-name> 캐시 생성에 실패했습니다. VPC 엔드포인트를 생성하기에 충분한

Detail-Type	설명	단위	소스	메시지
				여유 IP 주소가 없습니다.
캐시 생성 실패	캐시 ARN	실패	serverless-cache	<cache-name> 캐시 생성에 실패했습니다. 요청에 잘못된 서브넷이 제공되었습니다.
캐시 생성 실패	캐시 ARN	실패	serverless-cache	<cache-name> 캐시 생성에 실패했습니다. VPC 엔드포인트를 생성하기 위한 할당량 한도에 도달했습니다.
캐시 생성 실패	캐시 ARN	실패	serverless-cache	<cache-name> 캐시 생성에 실패했습니다. VPC 엔드포인트를 생성할 권한이 없습니다.
캐시 생성 실패	캐시 ARN	실패	serverless-cache	<cache-name> 캐시 생성에 실패했습니다. 호환되지 않는 Valkey 또는 Redis OSS 버전이 있는 사용자가 사용자 그룹 <user-group-name>에 있습니다.

Detail-Type	설명	단위	소스	메시지
캐시 생성 실패	캐시 ARN 캐시 스냅샷 ARN	실패	serverless-cache	<cache-name> 캐시 생성에 실패했습니다. 제공된 사용자 그룹 <user-group-name>이 존재하지 않습니다.
캐시 생성 실패	캐시 ARN	실패	serverless-cache	<cache-name> 캐시 생성에 실패했습니다. <reason>으로 인해 스냅샷에서 데이터를 복원하지 못했습니다. 실패 이유: • S3에서 파일을 검색하지 못했습니다. • 예상 md5가 실제 md5와 일치하지 않습니다. • 제공된 RDB 파일에 지원되지 않는 버전이 있습니다.

서버리스 캐시 업데이트 이벤트(Valkey 또는 RedisOSS)

Detail-Type	리소스 목록	범주	소스	메시지
캐시 업데이트	캐시 ARN	구성 변경	serverless-cache	SecurityGroups 캐시 <cache-name>에 대해 업데이트되었습니다.
캐시 업데이트	캐시 ARN	구성 변경	serverless-cache	<cache-name> 캐시에 대한 태그가 업데이트되었습니다.
캐시 업데이트 실패	캐시 ARN	구성 변경	serverless-cache	<cache-name> 캐시를 업데이트하지 못했습니다. 호환되지 않는 Valkey 또는 Redis OSS 버전이 있는 사용자가 사용자 그룹 <user-group-name>에 있습니다.
캐시 업데이트 실패	캐시 ARN	구성 변경	serverless-cache	<cache-name> 캐시 업데이트에 SecurityGroups 실패했습니다.
캐시 업데이트 실패	캐시 ARN	구성 변경	serverless-cache	권한이 부족하여 캐시 <cache-name> SecurityGroups 업데이트에 실패했습니다.
캐시 업데이트 실패	캐시 ARN	구성 변경	serverless-cache	<cache-name> 캐시를 업데이트하지 못했습니다.

Detail-Type	리소스 목록	범주	소스	메시지
				SecurityGroups SecurityGroups 이 유효하지 않아 업데이트에 실패했습니다.

서버리스 캐시 삭제 이벤트(Valkey 또는 RedisOSS)

Detail-Type	리소스 목록	범주	소스	메시지
캐시 삭제	캐시 ARN	삭제	serverless-cache	<cache-name> 캐시가 삭제되었습니다.

서버리스 캐시 사용 제한 이벤트(Valkey 또는 RedisOSS)

Detail-Type	설명	단위	소스	메시지
캐시 업데이트	캐시 ARN	구성 변경	serverless-cache	<cache-name> 캐시에 대한 한도가 업데이트되었습니다.
캐시 한도 근접	캐시 ARN	알림	serverless-cache	슬롯 <X>가 슬롯당 제한인 32GB의 <Y>%를 초과하여 사용하고 있습니다. 슬롯 10이 슬롯당 제한인 32GB의 90%를 초과하여 사용하고 있습니다.

Detail-Type	설명	단위	소스	메시지
캐시 업데이트 실패	캐시 ARN	실패	serverless-cache	캐시가 삭제되어 <cache-name> 캐시에 대한 제한이 업데이트되지 못했습니다.
캐시 업데이트 실패	캐시 ARN	실패	serverless-cache	구성이 유효하지 않아 <cache-name> 캐시에 대한 한도가 업데이트되지 못했습니다.
캐시 업데이트 실패	캐시 ARN	실패	serverless-cache	현재 캐시된 데이터가 새 한도를 초과하여 <cache-name> 캐시 한도가 업데이트되지 못했습니다. 제한을 적용하기 전에 일부 데이터를 삭제합니다.

서버리스 캐시 스냅샷 이벤트(Valkey 또는 RedisOSS)

Detail-Type	Resources-list	범주	소스	메시지
스냅샷 생성됨	캐시 ARN 스냅샷 경과 시간	생성	serverless-cache-snapshot	<cache-name> 캐시용으로 생성된 <snapshot-name> 스냅샷입니다.
스냅샷 생성 실패	캐시 ARN	실패	serverless-cache-snapshot	<cache-name> 캐시용 스냅샷

Detail-Type	Resources-list	범주	소스	메시지
	스냅샷 ARN			생성에 실패했습니다. 고객 관리형 키 <key-id> <reason>으로 인해 <snapshot-name> 스냅샷 생성이 실패했습니다. 실패 이유 메시지: • 고객 관리형 키가 비활성화됨 • 고객 관리형 키를 찾을 수 없음 • 요청 제한 시간이 초과됨
스냅샷 생성 실패	캐시 ARN 스냅샷 ARN	실패	serverless-cache-snapshot	<cache-name> 캐시용 스냅샷 생성에 실패했습니다. <reason>으로 인해 <snapshot-name> 스냅샷 생성이 실패했습니다. 기본 이유: • 내부 오류 발생

Detail-Type	Resources-list	범주	소스	메시지
스냅샷 내보내기 작업 실패	스냅샷 ARN	실패	serverless-cache-snapshot	<cache-name> 캐시용 스냅샷 내보내기에 실패했습니다. ElastiCache 는 버킷에 대한 권한이 없으므로 스냅샷을 버킷 %s로 내보낼 수 없습니다.
스냅샷 내보내기 작업 실패	스냅샷 ARN	실패	serverless-cache-snapshot	<cache-name> 캐시용 스냅샷 내보내기에 실패했습니다. 버킷에 이미 동일한 이름의 객체가 있으므로 '%'의 버킷으로 스냅샷을 내보낼 수 없습니다.
스냅샷 내보내기 작업 실패	스냅샷 ARN	실패	serverless-cache-snapshot	<cache-name> 캐시용 스냅샷 내보내기에 실패했습니다. 버킷 소유자 계정 ID가 변경되었으므로 '%'의 버킷으로 스냅샷을 내보낼 수 없습니다.

Detail-Type	Resources-list	범주	소스	메시지
스냅샷 내보내기 작업 실패	스냅샷 ARN	실패	serverless-cache-snapshot	<cache-name> 캐시용 스냅샷 내보내기에 실패했습니다. S3 버킷에 액세스할 수 없으므로 '%'의 버킷으로 스냅샷을 내보낼 수 없습니다.
스냅샷 내보내기 작업 실패	스냅샷 ARN	실패	serverless-cache-snapshot	<cache-name> 캐시용 스냅샷 내보내기에 실패했습니다. 버킷에 액세스할 수 없으므로 '%'의 버킷으로 스냅샷을 내보낼 수 없습니다.
스냅샷 내보내기 작업 실패	스냅샷 ARN	실패	serverless-cache-snapshot	<cache-name> 캐시용 스냅샷 내보내기에 실패했습니다. 버킷이 존재하지 않으므로 '%'의 버킷으로 스냅샷을 내보낼 수 없습니다.

Detail-Type	Resources-list	범주	소스	메시지
스냅샷 내보내기 작업 실패	스냅샷 ARN	실패	serverless-cache-snapshot	<cache-name> 캐시용 스냅샷 내보내기에 실패했습니다. 소스 스냅샷 고객 관리형 키 % <reason> 과 함께 '%'의 버킷으로 내보낼 수 없습니다.
스냅샷 내보내기 작업 실패	스냅샷 ARN	실패	serverless-cache-snapshot	<cache-name> 캐시용 스냅샷 내보내기에 실패했습니다. 스냅샷을 '%'의 버킷으로 내보낼 수 없습니다.
스냅샷 복사 실패	스냅샷 ARN-1 스냅샷 ARN-2	실패	serverless-cache-snapshot	<snapshot-name> 스냅샷을 복사하지 못했습니다. 스냅샷 '%'를 소스 스냅샷 고객 관리형 키 <key-id> <reason-name> 와 함께 '%'의 스냅샷에 복사할 수 없습니다.

Detail-Type	Resources-list	범주	소스	메시지
스냅샷 복사 실패	스냅샷 ARN-1 스냅샷 ARN-2	실패	serverless-cache-snapshot	<snapshot-name> 스냅샷을 복사하지 못했습니다. 스냅샷 '%'를 타겟 스냅샷 고객 관리형 키 '%' '%'와 함께 스냅샷 '%'에 복사할 수 없습니다.

Valkey 및 Redis에 대한 자체 설계된 클러스터 지표 및 이벤트 OSS

ElastiCache 는 Valkey 및 Redis 로 작업할 때 자체 설계된 클러스터를 모니터링하기 위한 다양한 지표 및 이벤트를 제공합니다OSS. 여기에는 AWS CLI 및 Amazon Simple Notification Service()를 통해 사용할 수 있는 호스트 수준 지표, 명령 수준 지표 및 이벤트 로그가 포함됩니다SNS.

주제

- 자체 설계된 클러스터의 지표
- 자체 설계된 클러스터의 이벤트(Valkey 및 RedisOSS)

자체 설계된 클러스터의 지표

클러스터를 자체 설계할 때 호스트 수준 지표와 캐시 지표를 모두 포함하여 각 노드 수준에서 지표를 ElastiCache 내보냅니다.

호스트 수준 지표에 대한 자세한 내용은 호스트 수준 지표 섹션을 참조하세요.

노드 수준 지표에 대한 자세한 내용은 Valkey 및 Redis에 대한 지표 OSS 섹션을 참조하세요.

자체 설계된 클러스터의 이벤트(Valkey 및 RedisOSS)

ElastiCache 는 자체 설계된 캐시와 관련된 이벤트를 기록합니다. 자체 설계된 클러스터로 작업할 때 ElastiCache 콘솔, AWS CLI또는 Amazon Simple Notification Service()를 사용하여 클러스터 이벤트를 볼 수 있습니다SNS. 자체 설계된 클러스터 이벤트는 Amazon 에 게시되지 않습니다 EventBridge.

자체 설계된 클러스터 이벤트에는 이벤트 날짜 및 시간, 이벤트의 원본 이름 및 원본 유형, 이벤트 설명 등의 정보가 포함됩니다. ElastiCache 콘솔, AWS CLI describe-events 명령 또는 작업 을 사용하여 로그에서 이벤트를 쉽게 검색할 수 있습니다 ElastiCache API DescribeEvents.

ElastiCache 이벤트 보기(콘솔)

다음 절차는 ElastiCache 콘솔을 사용하여 이벤트를 표시합니다.

ElastiCache 콘솔을 사용하여 이벤트를 보려면

1. 에 로그인 AWS Management Console 하고 에서 ElastiCache 콘솔을 엽니다. https://console.aws.amazon.com/elasticache/
2. 사용 가능한 모든 이벤트의 목록을 보려면 탐색 창에서 이벤트를 선택합니다.
3. 이벤트 화면에서 목록의 각 행은 하나의 이벤트를 나타내며 이벤트 소스, 이벤트 유형, 이벤트 GMT 시간 및 이벤트 설명을 표시합니다. [Filter]를 사용하여 이벤트 목록에서 모든 이벤트를 볼지 특정 유형의 이벤트만 볼지를 지정할 수 있습니다.

ElastiCache 이벤트 보기(AWS CLI)

를 사용하여 ElastiCache 이벤트 목록을 생성하려면 describe-events 명령을 AWS CLI사용합니다. 선택적 파라미터를 사용하여 나열된 이벤트의 유형, 나열된 이벤트의 기간, 나열할 이벤트의 최대 수 등을 제어할 수 있습니다.

다음 코드는 최대 40개의 자체 설계된 클러스터 이벤트를 나열합니다.

```
aws elasticache describe-events --source-type cache-cluster --max-items 40
```

다음 코드는 지난 24시간(1,440분) 동안의 자체 설계된 캐시의 모든 이벤트를 나열합니다.

```
aws elasticache describe-events --source-type cache-cluster --duration 1440
```

자체 설계된 클러스터 이벤트

이 섹션에는 자체 설계된 클러스터에서 수신할 것으로 예상되는 이벤트 목록이 포함되어 있습니다.

다음 ElastiCache 이벤트는 Amazon SNS 알림을 트리거합니다. 이벤트 세부 정보에 대한 자세한 내용은 ElastiCache 이벤트 보기 섹션을 참조하세요.

이벤트 이름	메시지	설명
ElastiCache:AddCacheNodeComplete	ElastiCache:AddCacheNodeComplete : *cache-cluster*	캐시 노드가 캐시 클러스터에 추가되었고 사용할 준비가 되어 있습니다.
ElastiCache:사용 가능한 IP 주소가 충분하지 AddCacheNodeFailed 않아서	ElastiCache:AddCacheNodeFailed : *cluster-name*	사용 가능한 IP 주소가 충분하지 않아 캐시 노드를 추가하지 못했습니다.
ElastiCache:CacheClusterParametersChanged	ElastiCache:CacheClusterParametersChanged : *cluster-name*	하나 이상의 캐시 클러스터 파라미터가 변경되었습니다.
ElastiCache:CacheClusterProvisioningComplete	ElastiCache:CacheClusterProvisioningComplete *cluster-name-0001-005*	캐시 클러스터의 프로비저닝이 완료되어 캐시 클러스터에 있는 캐시 노드를 사용할 수 있습니다.
ElastiCache:호환되지 않는 네트워크 상태로 CacheClusterProvisioningFailed 인해	ElastiCache:CacheClusterProvisioningFailed : *cluster-name*	새 캐시 클러스터를 존재하지 않는 가상 프라이빗 클라우드()로 시작하려고 했습니다VPC.
ElastiCache:CacheClusterScalingComplete	CacheClusterScalingComplete : *cluster-name*	캐시 클러스터의 조정이 성공적으로 완료되었습니다.
ElastiCache:CacheClusterScalingFailed	ElastiCache:CacheClusterScalingFailed : *cluster-name*	캐시 클러스터에 대한 스케일업 작업이 실패했습니다.
ElastiCache:CacheClusterSecurityGroupModified	ElastiCache:CacheClusterSecurityGroupModified : *cluster-name*	다음 이벤트 중 하나가 발생했습니다. • 캐시 클러스터를 위한 승인된 캐시 보안 그룹이 수정되었습니다. •

이벤트 이름	메시지	설명
		캐시 클러스터와 연결된 캐시 EC2 보안 그룹에 대해 하나 이상의 새 보안 그룹이 승인되었습니다. • 캐시 클러스터와 연결된 캐시 EC2 보안 그룹에서 하나 이상의 보안 그룹이 취소되었습니다.
ElastiCache:CacheNodeReplaceStarted	`ElastiCache:CacheNodeReplaceStarted : cluster-name`	ElastiCache 에서 캐시 노드를 실행하는 호스트가 성능 저하되었거나 연결할 수 없음을 감지하여 캐시 노드를 교체하기 시작했습니다. **Note** 교체된 캐시 노드의 DNS 항목은 변경되지 않습니다. 대부분의 경우에 이 이벤트가 발생할 때 클라이언트의 서버 목록을 새로 고침하지 않아도 됩니다. 그러나 일부 캐시 클라이언트 라이브러리 ElastiCache 는 가 캐시 노드를 교체한 후에도 캐시 노드 사용을 중지할 수 있습니다. 이 경우 이 이벤트가 발생할 때 애플리케이션은 서버 목록을 새로 고쳐야 합니다.

이벤트 이름	메시지	설명
ElastiCache:CacheNodeReplaceComplete	`ElastiCache:CacheNodeReplaceComplete : cluster-name`	ElastiCache 에서 캐시 노드를 실행하는 호스트가 성능 저하되거나 연결할 수 없음을 감지하고 캐시 노드 교체를 완료했습니다. **Note** 교체된 캐시 노드의 DNS 항목은 변경되지 않습니다. 대부분의 경우에 이 이벤트가 발생할 때 클라이언트의 서버 목록을 새로 고침하지 않아도 됩니다. 그러나 일부 캐시 클라이언트 라이브러리 ElastiCache 는 가 캐시 노드를 교체한 후에도 캐시 노드 사용을 중지할 수 있습니다. 이 경우 이 이벤트가 발생할 때 애플리케이션은 서버 목록을 새로 고쳐야 합니다.
ElastiCache:CacheNodesRebooted	`ElastiCache:CacheNodesRebooted : cluster-name`	하나 이상의 캐시 노드가 재부팅되었습니다. 메시지(Memcached): `"Cache node %s shutdown"` , 두 번째 메시지: `"Cache node %s restarted"`

이벤트 이름	메시지	설명
ElastiCache:CertificateRenewalComplete (Valkey 또는 RedisOSS만 해당)	`ElastiCache:CertificateRenewalComplete`	Amazon CA 인증서가 성공적으로 갱신되었습니다.
ElastiCache:CreateReplicationGroupComplete	`ElastiCache:CreateReplicationGroupComplete : cluster-name`	복제 그룹이 성공적으로 생성되었습니다.
ElastiCache:DeleteCacheClusterComplete	`ElastiCache:DeleteCacheClusterComplete : cluster-name`	캐시 클러스터 및 연결된 모든 캐시 노드 삭제를 완료했습니다.
ElastiCache:FailoverComplete (Valkey 또는 RedisOSS만 해당)	`ElastiCache:FailoverComplete : mycluster`	복제본 노드에 대한 장애 조치가 성공했습니다.
ElastiCache:ReplicationGroupIncreaseReplicaCountFinished	`ElastiCache:ReplicationGroupIncreaseReplicaCountFinished : cluster-name-0001-005`	클러스터의 복제본 수가 증가했습니다.
ElastiCache:ReplicationGroupIncreaseReplicaCountStarted	`ElastiCache:ReplicationGroupIncreaseReplicaCountStarted : cluster-name-0003-004`	클러스터에 복제본을 추가하는 프로세스가 시작되었습니다.
ElastiCache:NodeReplacementCanceled	`ElastiCache:NodeReplacementCanceled : cluster-name`	교체가 예약되어 있는 클러스터의 노드가 더 이상 교체 예약이 되지 않습니다.

이벤트 이름	메시지	설명
ElastiCache:NodeReplacementRescheduled	`ElastiCache:NodeReplacementRescheduled : cluster-name`	이전에 교체가 예약되어 있는 클러스터의 노드가 알림에 설명된 새 기간 동안 교체가 예약됩니다. 수행할 수 있는 작업에 대한 자세한 내용은 노드 교체(Valkey 및 RedisOSS) 섹션을 참조하세요.
ElastiCache:NodeReplacementScheduled	`ElastiCache:NodeReplacementScheduled : cluster-name`	클러스터의 노드가 알림에 설명된 기간 동안 교체가 예약됩니다. 수행할 수 있는 작업에 대한 자세한 내용은 노드 교체(Valkey 및 RedisOSS) 섹션을 참조하세요.
ElastiCache:RemoveCacheNodeComplete	`ElastiCache:RemoveCacheNodeComplete : cluster-name`	캐시 노드가 캐시 클러스터에서 제거되었습니다.
ElastiCache:ReplicationGroupScalingComplete	`ElastiCache:ReplicationGroupScalingComplete : cluster-name`	복제 그룹에 대한 확장 작업이 성공적으로 완료되었습니다.
ElastiCache:ReplicationGroupScalingFailed	`"Failed applying modification to cache node type to %s."`	복제 그룹에 대한 확장 작업이 실패했습니다.
ElastiCache:ServiceUpdateAvailableForNode	`"Service update is available for cache node %s."`	노드에서 셀프 서비스 업데이트를 사용할 수 있습니다.

이벤트 이름	메시지	설명
ElastiCache:SnapshotComplete (Valkey 또는 RedisOSS만 해당)	`ElastiCache:SnapshotComplete : cluster-name`	캐시 스냅샷이 성공적으로 완료되었습니다.
ElastiCache:SnapshotFailed (발키 또는 RedisOSS만 해당)	`SnapshotFailed : cluster-name`	캐시 스냅샷이 실패했습니다. 원인에 대한 자세한 내용은 클러스터의 캐시 이벤트를 참조하세요. 스냅샷을 설명할 경우 `DescribeSnapshots` 를 참조하세요. 상태는 `failed`입니다.

Memcached에 대한 서버리스 지표 및 이벤트

이 섹션에는 서버리스 캐시로 작업할 때 모니터링할 수 있는 지표 및 이벤트에 대한 설명이 나와 있습니다.

주제

서버리스 캐시 지표

이 섹션에서는 Memcached 서버리스 캐시로 작업할 때 모니터링할 수 있는 지표 및 이벤트에 대해 설명합니다.

`AWS/ElastiCache` 네임스페이스에는 Memcached 서버리스 캐시에 대한 다음 CloudWatch 지표가 포함됩니다.

지표	설명	단위
BytesUsedForCache	캐시에 저장된 데이터에서 사용되는 총 바이트 수	바이트
ElastiCacheProcessingUnits	캐시에서 실행된 요청에 사용된 총 ElastiCacheProcessingUnits (ECPUs) 수	개수
SuccessfulReadRequestLatency	성공적인 읽기 요청의 지연 시간	마이크로초
SuccessfulWriteRequestLatency	성공적인 쓰기 요청의 지연 시간	마이크로초
TotalCmdsCount	캐시에서 실행된 모든 명령의 총 개수	개수
CurrConnections	캐시에 대한 클라이언트 연결 수	개수

지표	설명	단위
ThrottledCmds	워크로드가 확장할 ElastiCache 수 있는 것보다 더 빠르게 확장되었기 ElastiCache 때문에 에서 제한한 요청 수입니다.	개수
NewConnections	이 기간에 서버에서 허용된 총 연결 수입니다.	개수
CurrItems	캐시 항목 수입니다.	개수
NetworkBytesIn	캐시로 전송된 총 바이트 수	바이트
NetworkBytesOut	캐시에서 나간 총 바이트 수	바이트
Evictions	캐시에서 제거된 키 수	개수
Reclaimed	캐시에서 만료된 키 수	개수

명령 수준 지표

ElastiCache 는 또한 다음과 같은 Memcached 명령 수준 지표를 내보냅니다.

지표	설명	단위
CmdGet	캐시가 수신한 get 명령 수	개수
CmdSet	캐시가 수신한 set 명령 수	개수
CmdTouch	캐시가 수신한 터치 명령 수	개수
GetHits	요청한 키를 찾았을 때 캐시가 수신한 get 요청 수	개수
GetMisses	요청한 키를 찾지 못했을 때 캐시가 수신한 get 요청 수	개수

지표	설명	단위
IncrHits	요청한 키를 찾았을 때 캐시가 수신한 increment 요청 수	개수
IncrMisses	요청한 키를 찾지 못했을 때 캐시가 수신한 increment 요청 수	개수
DecrHits	요청한 키를 찾았을 때 캐시가 수신한 decrement 요청 수	개수
DecrMisses	요청한 키를 찾지 못했을 때 캐시가 수신한 decrement 요청 수	개수
DeleteHits	요청한 키를 찾았을 때 캐시가 수신한 delete 요청 수	개수
DeleteMisses	요청한 키를 찾지 못했을 때 캐시가 수신한 delete 요청 수	개수
TouchHits	새로운 만료 시간 지정 이후 사용한 적이 있는 키 수	개수
TouchMisses	사용한 적은 있지만 찾을 수 없는 키 수	개수
CasHits	요청한 키를 찾았고, CAS 값이 일치할 때 캐시가 수신한 CAS 요청 수	개수
CasMisses	요청한 키를 찾지 못했을 때 캐시가 수신한 CAS 요청 수	개수
CasBadval	CAS 값이 저장된 CAS 값과 일치하지 않을 때 캐시가 수신한 CAS 요청 수	개수
CmdFlush	캐시가 수신한 flush 명령 수	개수

서버리스 캐시 이벤트

ElastiCache 는 서버리스 캐시와 관련된 이벤트를 기록합니다. 여기에는 이벤트 날짜 및 시간, 이벤트의 원본 이름 및 원본 유형, 이벤트 설명 등의 정보가 포함됩니다. ElastiCache 콘솔, AWS CLI describe-events 명령 또는 작업 을 사용하여 로그에서 이벤트를 쉽게 검색할 수 있습니다 ElastiCache APIDescribeEvents.

Amazon 를 사용하여 ElastiCache 이벤트를 모니터링, 수집, 변환 및 작업하도록 선택할 수 있습니다 EventBridge. Amazon EventBridge https://docs.aws.amazon.com/eventbridge/최신/사용자 가이드/에서 자세히 알아보세요.

ElastiCache 이벤트 보기(콘솔)

ElastiCache 콘솔을 사용하여 이벤트를 보려면:

1. 에 로그인 AWS Management Console 하고 에서 ElastiCache 콘솔을 엽니다. https://console.aws.amazon.com/elasticache/
2. 사용 가능한 모든 이벤트의 목록을 보려면 탐색 창에서 이벤트를 선택합니다.
3. 이벤트 화면에서 목록의 각 행은 하나의 이벤트를 나타내며 이벤트 소스, 이벤트 유형, 이벤트 GMT 시간 및 이벤트 설명을 표시합니다. [Filter]를 사용하여 이벤트 목록에서 모든 이벤트를 볼지 특정 유형의 이벤트만 볼지를 지정할 수 있습니다.

ElastiCache 이벤트 보기(AWS CLI)

를 사용하여 ElastiCache 이벤트 목록을 생성하려면 describe-events 명령을 AWS CLI사용합니다. 선택적 파라미터를 사용하여 나열된 이벤트의 유형, 나열된 이벤트의 기간, 나열할 이벤트의 최대 수 등을 제어할 수 있습니다.

다음 코드는 최대 40개의 서버리스 캐시 이벤트를 나열합니다.

```
aws elasticache describe-events --source-type serverless-cache --max-items 40
```

다음 코드는 지난 24시간(1,440분) 동안 발생한 서버리스 캐시의 모든 이벤트를 나열합니다.

```
aws elasticache describe-events --source-type serverless-cache --duration 1440
```

서버리스 이벤트

이 섹션에서는 서버리스 캐시에서 수신할 수 있는 다양한 유형의 이벤트를 설명합니다.

서버리스 캐시 생성 이벤트

Detail-Type	설명	단위	소스	메시지
캐시 생성됨	캐시 ARN	생성	serverless-cache	<cache-name> 캐시가 생성되어 바로 사용할 수 있습니다.
캐시 생성 실패	캐시 ARN	실패	serverless-cache	<cache-name> 캐시 생성에 실패했습니다. VPC 엔드포인트를 생성하기에 충분한 여유 IP 주소가 없습니다.
캐시 생성 실패	캐시 ARN	실패	serverless-cache	<cache-name> 캐시 생성에 실패했습니다. 요청에 잘못된 서브넷이 제공되었습니다.
캐시 생성 실패	캐시 ARN	실패	serverless-cache	<cache-name> 캐시 생성에 실패했습니다. VPC 엔드포인트를 생성하기 위한 할당량 한도에 도달했습니다.
캐시 생성 실패	캐시 ARN	실패	serverless-cache	<cache-name> 캐시 생성에 실패했습니다. VPC 엔드포인트를 생

Detail-Type	설명	단위	소스	메시지
				성할 권한이 없습니다.

서버리스 캐시 업데이트 이벤트(Memcached)

Detail-Type	리소스 목록	범주	소스	메시지
캐시 업데이트	캐시 ARN	구성 변경	serverless-cache	SecurityGroups 캐시 <cache-name>에 대해 업데이트되었습니다.
캐시 업데이트	캐시 ARN	구성 변경	serverless-cache	<cache-name> 캐시에 대한 태그가 업데이트되었습니다.
캐시 업데이트 실패	캐시 ARN	구성 변경	serverless-cache	<cache-name> 캐시 업데이트에 SecurityGroups 실패했습니다.
캐시 업데이트 실패	캐시 ARN	구성 변경	serverless-cache	권한이 부족하여 캐시 <cache-name> SecurityGroups 업데이트에 실패했습니다.
캐시 업데이트 실패	캐시 ARN	구성 변경	serverless-cache	이 유효하지 않아 캐시 <cache-name> SecurityGroups 업데이트 SecurityGroups 에 실패했습니다.

서버리스 캐시 삭제 이벤트(Memcached)

Detail-Type	리소스 목록	범주	소스	메시지
캐시 삭제	캐시 ARN	삭제	serverless-cache	<cache-name> 캐시가 삭제되었습니다.

서버리스 캐시 사용 제한 이벤트(Memcached)

Detail-Type	설명	단위	소스	메시지
캐시 업데이트	캐시 ARN	구성 변경	serverless-cache	<cache-name> 캐시에 대한 한도가 업데이트되었습니다.
캐시 업데이트 실패	캐시 ARN	실패	serverless-cache	캐시가 삭제되어 <cache-name> 캐시에 대한 제한이 업데이트되지 못했습니다.
캐시 업데이트 실패	캐시 ARN	실패	serverless-cache	구성이 유효하지 않아 <cache-name> 캐시에 대한 한도가 업데이트되지 못했습니다.

서버리스 캐시 스냅샷 이벤트(Memcached)

Detail-Type	Resources-list	범주	소스	메시지
스냅샷 생성됨	캐시 ARN 스냅샷 경과 시간	생성	serverless-cache-snapshot	<cache-name> 캐시용으로 생성된 <snapshot-

Detail-Type	Resources-list	범주	소스	메시지
				name> 스냅샷입니다.
스냅샷 생성 실패	캐시 ARN 스냅샷 ARN	실패	serverless-cache-snapshot	<cache-name> 캐시용 스냅샷 생성에 실패했습니다. 고객 관리형 키 <key-id> <reason>으로 인해 <snapshot-name> 스냅샷 생성이 실패했습니다. 실패 이유 메시지: • 고객 관리형 키가 비활성화됨 • 고객 관리형 키를 찾을 수 없음 • 요청 제한 시간이 초과됨

Detail-Type	Resources-list	범주	소스	메시지
스냅샷 생성 실패	캐시 ARN 스냅샷 ARN	실패	serverless-cache-snapshot	<cache-name> 캐시용 스냅샷 생성에 실패했습니다. <reason>으로 인해 <snapshot-name> 스냅샷 생성이 실패했습니다. 기본 이유: • 내부 오류 발생
스냅샷 내보내기 작업 실패	스냅샷 ARN	실패	serverless-cache-snapshot	<cache-name> 캐시용 스냅샷 내보내기에 실패했습니다. ElastiCache 는 버킷에 대한 권한이 없으므로 스냅샷을 버킷 %s로 내보낼 수 없습니다.
스냅샷 내보내기 작업 실패	스냅샷 ARN	실패	serverless-cache-snapshot	<cache-name> 캐시용 스냅샷 내보내기에 실패했습니다. 버킷에 이미 동일한 이름의 객체가 있으므로 '%'의 버킷으로 스냅샷을 내보낼 수 없습니다.

Detail-Type	Resources-list	범주	소스	메시지
스냅샷 내보내기 작업 실패	스냅샷 ARN	실패	serverless-cache-snapshot	<cache-name> 캐시용 스냅샷 내보내기에 실패했습니다. 버킷 소유자 계정 ID가 변경되었으므로 '%'의 버킷으로 스냅샷을 내보낼 수 없습니다.
스냅샷 내보내기 작업 실패	스냅샷 ARN	실패	serverless-cache-snapshot	<cache-name> 캐시용 스냅샷 내보내기에 실패했습니다. S3 버킷에 액세스할 수 없으므로 '%'의 버킷으로 스냅샷을 내보낼 수 없습니다.
스냅샷 내보내기 작업 실패	스냅샷 ARN	실패	serverless-cache-snapshot	<cache-name> 캐시용 스냅샷 내보내기에 실패했습니다. 버킷에 액세스할 수 없으므로 '%'의 버킷으로 스냅샷을 내보낼 수 없습니다.

Detail-Type	Resources-list	범주	소스	메시지
스냅샷 내보내기 작업 실패	스냅샷 ARN	실패	serverless-cache-snapshot	<cache-name> 캐시용 스냅샷 내보내기에 실패했습니다. 버킷이 존재하지 않으므로 '%'의 버킷으로 스냅샷을 내보낼 수 없습니다.
스냅샷 내보내기 작업 실패	스냅샷 ARN	실패	serverless-cache-snapshot	<cache-name> 캐시용 스냅샷 내보내기에 실패했습니다. 소스 스냅샷 고객 관리형 키 % <reason> 과 함께 '%'의 버킷으로 내보낼 수 없습니다.
스냅샷 내보내기 작업 실패	스냅샷 ARN	실패	serverless-cache-snapshot	<cache-name> 캐시용 스냅샷 내보내기에 실패했습니다. 스냅샷을 '%'의 버킷으로 내보낼 수 없습니다.

Detail-Type	Resources-list	범주	소스	메시지
스냅샷 복사 실패	스냅샷 ARN-1 스냅샷 ARN-2	실패	serverless-cache-snapshot	<snapshot-name> 스냅샷을 복사하지 못했습니다. 스냅샷 '%'를 소스 스냅샷 고객 관리형 키 <key-id> <reason-name>와 함께 '%'의 스냅샷에 복사할 수 없습니다.
스냅샷 복사 실패	스냅샷 ARN-1 스냅샷 ARN-2	실패	serverless-cache-snapshot	<snapshot-name> 스냅샷을 복사하지 못했습니다. 스냅샷 '%'를 타겟 스냅샷 고객 관리형 키 '%' '%'와 함께 스냅샷 '%'에 복사할 수 없습니다.

를 사용하여 Amazon ElastiCache API 통화 로깅 AWS CloudTrail

Amazon ElastiCache 은 Amazon 의 사용자, 역할 또는 AWS 서비스에서 수행한 작업의 레코드를 제공하는 AWS CloudTrail서비스인 와 통합됩니다 ElastiCache. 는 Amazon ElastiCache 콘솔의 API 호출 및 Amazon 작업에 대한 코드 호출을 포함하여 Amazon에 대한 모든 호출을 ElastiCache API 이벤트 ElastiCache 로 CloudTrail 캡처합니다. 추적을 생성하는 경우 Amazon 에 대한 CloudTrail 이벤트를 포함하여 Amazon S3 버킷으로 이벤트를 지속적으로 전송할 수 있습니다 ElastiCache. 추적을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록 에서 최신 이벤트를 볼 수 있습니다. 에서 수집한 정보를 사용하여 Amazon 에 수행된 요청 CloudTrail, 요청이 수행된 ElastiCacheIP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

에 대한 자세한 내용은 AWS CloudTrail 사용 설명서 섹션을 CloudTrail참조하세요.

의 Amazon ElastiCache 정보 CloudTrail

CloudTrail 는 AWS 계정을 생성할 때 계정에서 활성화됩니다. Amazon 에서 활동이 발생하면 ElastiCache해당 활동은 CloudTrail 이벤트 기록 의 다른 AWS 서비스 이벤트와 함께 이벤트에 기록됩니다. AWS 계정에서 최근 이벤트를 보고 검색하고 다운로드할 수 있습니다. 자세한 내용은 이벤트 기록을 사용하여 CloudTrail 이벤트 보기를 참조하세요.

Amazon 에 대한 이벤트를 포함하여 AWS 계정의 이벤트에 대한 지속적인 기록을 위해 추적을 ElastiCache생성합니다. 추적을 사용하면 CloudTrail 가 Amazon S3 버킷에 로그 파일을 전달할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 리전에 추적이 적용됩니다. 추적은 AWS 파티션의 모든 리전에서 이벤트를 기록하고 지정한 Amazon S3 버킷에 로그 파일을 전달합니다. 또한 CloudTrail 로그에서 수집된 이벤트 데이터를 추가로 분석하고 이에 따라 작업하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음 자료를 참조하십시오.

- 추적 생성 개요
- CloudTrail 지원되는 서비스 및 통합
- 에 대한 Amazon SNS 알림 구성 CloudTrail
- 여러 리전에서 CloudTrail 로그 파일 수신 및 여러 계정에서 CloudTrail 로그 파일 수신

모든 Amazon ElastiCache 작업은 에 의해 기록 CloudTrail 되며 ElastiCache API 참조 에 문서화됩니다. 예를 들어 `CreateCacheCluster`를 호출`DescribeCacheCluster`하고 `ModifyCacheCluster` 작업을 수행하면 CloudTrail 로그 파일에 항목이 생성됩니다.

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. 보안 인증 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청이 루트 또는 IAM 사용자 자격 증명으로 이루어졌는지 여부.
- 역할 또는 페더레이션 사용자에 대한 임시 보안 보안 인증을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청을 했는지 여부.

자세한 내용은 CloudTrail userIdentity 요소를 참조하세요.

Amazon ElastiCache 로그 파일 항목 이해

추적은 사용자가 지정한 Amazon S3 버킷에 로그 파일로 이벤트를 전달할 수 있도록 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함됩니다. 이벤트는 모든 소스의 단일 요청을 나

타내며 요청된 작업, 작업 날짜 및 시간, 요청 파라미터 등에 대한 정보를 포함합니다. CloudTrail log 파일은 퍼블릭 API 호출의 순서가 지정된 스택 추적이 아니므로 특정 순서로 표시되지 않습니다.

다음 예제에서는 `CreateCacheCluster` 작업을 보여주는 CloudTrail 로그 항목을 보여줍니다.


```
{
    "eventVersion":"1.01",
    "userIdentity":{
        "type":"IAMUser",
        "principalId":"EXAMPLEEXAMPLEEXAMPLE",
        "arn":"arn:aws:iam::123456789012:user/elasticache-allow",
        "accountId":"123456789012",
        "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
        "userName":"elasticache-allow"
    },
    "eventTime":"2014-12-01T22:00:35Z",
    "eventSource":"elasticache.amazonaws.com",
    "eventName":"CreateCacheCluster",
    "awsRegion":"us-west-2",
    "sourceIPAddress":"192.0.2.01",
    "userAgent":"AWS CLI/ElastiCache 1.10 API 2014-12-01",
    "requestParameters":{
        "numCacheNodes":2,
        "cacheClusterId":"test-memcached",
        "engine":"memcached",
        "aZMode":"cross-az",
        "cacheNodeType":"cache.m1.small",

    },
    "responseElements":{
        "engine":"memcached",
        "clientDownloadLandingPage":"https://console.aws.amazon.com/elasticache/
home#client-download:",
        "cacheParameterGroup":{
            "cacheParameterGroupName":"default.memcached1.4",
            "cacheNodeIdsToReboot":{
            },
            "parameterApplyStatus":"in-sync"
        },
        "preferredAvailabilityZone":"Multiple",
        "numCacheNodes":2,
        "cacheNodeType":"cache.m1.small",

        "cacheClusterStatus":"creating",
```

```
        "autoMinorVersionUpgrade":true,
        "preferredMaintenanceWindow":"thu:05:00-thu:06:00",
        "cacheClusterId":"test-memcached",
        "engineVersion":"1.4.14",
        "cacheSecurityGroups":[
            {
                "status":"active",
                "cacheSecurityGroupName":"default"
            }
        ],
        "pendingModifiedValues":{
        }
    },
    "requestID":"104f30b3-3548-11e4-b7b8-6d79ffe84edd",
    "eventID":"92762127-7a68-42ce-8787-927d2174cde1"
}
```

다음 예제에서는 DescribeCacheCluster 작업을 보여주는 CloudTrail 로그 항목을 보여줍니다. 모든 Amazon ElastiCache Describe 호출(Describe*)의 경우 ResponseElements 섹션이 제거되고 로 표시됩니다null.

```
{
    "eventVersion":"1.01",
    "userIdentity":{
        "type":"IAMUser",
        "principalId":"EXAMPLEEXAMPLEEXAMPLE",
        "arn":"arn:aws:iam::123456789012:user/elasticache-allow",
        "accountId":"123456789012",
        "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
        "userName":"elasticache-allow"
    },
    "eventTime":"2014-12-01T22:01:00Z",
    "eventSource":"elasticache.amazonaws.com",
    "eventName":"DescribeCacheClusters",
    "awsRegion":"us-west-2",
    "sourceIPAddress":"192.0.2.01",
    "userAgent":"AWS CLI/ElastiCache 1.10 API 2014-12-01",
    "requestParameters":{
        "showCacheNodeInfo":false,
        "maxRecords":100
    },
    "responseElements":null,
    "requestID":"1f0b5031-3548-11e4-9376-c1d979ba565a",
```

```
    "eventID":"a58572a8-e81b-4100-8e00-1797ed19d172"
}
```

다음 예제는 ModifyCacheCluster 작업을 기록하는 CloudTrail 로그 항목을 보여줍니다.

```
{
    "eventVersion":"1.01",
    "userIdentity":{
        "type":"IAMUser",
        "principalId":"EXAMPLEEXAMPLEEXAMPLE",
        "arn":"arn:aws:iam::123456789012:user/elasticache-allow",
        "accountId":"123456789012",
        "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
        "userName":"elasticache-allow"
    },
    "eventTime":"2014-12-01T22:32:21Z",
    "eventSource":"elasticache.amazonaws.com",
    "eventName":"ModifyCacheCluster",
    "awsRegion":"us-west-2",
    "sourceIPAddress":"192.0.2.01",
    "userAgent":"AWS CLI/ElastiCache 1.10 API 2014-12-01",
    "requestParameters":{
        "applyImmediately":true,
        "numCacheNodes":3,
        "cacheClusterId":"test-memcached"
    },
    "responseElements":{
        "engine":"memcached",
        "clientDownloadLandingPage":"https://console.aws.amazon.com/elasticache/
home#client-download:",
        "cacheParameterGroup":{
            "cacheParameterGroupName":"default.memcached1.4",
            "cacheNodeIdsToReboot":{
            },
            "parameterApplyStatus":"in-sync"
        },
        "cacheClusterCreateTime":"Dec 1, 2014 10:16:06 PM",
        "preferredAvailabilityZone":"Multiple",
        "numCacheNodes":2,
        "cacheNodeType":"cache.m1.small",
        "cacheClusterStatus":"modifying",
        "autoMinorVersionUpgrade":true,
        "preferredMaintenanceWindow":"thu:05:00-thu:06:00",
```

```
        "cacheClusterId":"test-memcached",
        "engineVersion":"1.4.14",
        "cacheSecurityGroups":[
            {
                "status":"active",
                "cacheSecurityGroupName":"default"
            }
        ],
        "configurationEndpoint":{
            "address":"test-memcached.example.cfg.use1prod.cache.amazonaws.com",
            "port":11211
        },
        "pendingModifiedValues":{
            "numCacheNodes":3
        }
    },
    "requestID":"807f4bc3-354c-11e4-9376-c1d979ba565a",
    "eventID":"e9163565-376f-4223-96e9-9f50528da645"
}
```


Amazon ElastiCache 이벤트 SNS 모니터링

클러스터에 중요한 이벤트가 발생하면 는 특정 Amazon SNS 주제에 알림을 ElastiCache 보냅니다. 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

- ElastiCache Amazon SNS 알림 관리
- ElastiCache 이벤트 보기
- 이벤트 알림 및 Amazon SNS

ElastiCache Amazon SNS 알림 관리

Amazon Simple Notification Service(Amazon)를 사용하여 중요한 클러스터 이벤트에 대한 알림을 보내 ElastiCache 도록 를 구성할 수 있습니다SNS. 이 예제에서는 알림을 수신하도록 Amazon SNS 주제의 Amazon 리소스 이름(ARN)을 사용하여 클러스터를 구성합니다.

> **Note**
>
> - 이 주제에서는 Amazon에 가입SNS하고 Amazon SNS 주제를 설정하고 구독했다고 가정합니다. 이렇게 하는 방법에 대한 정보는 Amazon Simple Notification Service 개발자 안내서를 참조하세요.
> - 기본적으로 는 현재 지정된 그룹뿐만 아니라 리전의 모든 그룹에 `API modify-replication-group` 영향을 미칩니다. 리전의 특정 그룹을 다른 그룹과 다르게 구성하려면 `--notification-topic-arn` 옵션을 사용하여 해당 그룹에 대한 별도의 주제를 생성할 수 있습니다.

Amazon SNS 주제 추가

다음 섹션에서는 AWS 콘솔, AWS CLI또는 를 사용하여 Amazon SNS 주제를 추가하는 방법을 보여줍니다 ElastiCache API.

Amazon SNS 주제 추가(콘솔)

다음 절차에서는 클러스터에 대한 Amazon SNS 주제를 추가하는 방법을 보여줍니다. Valkey 또는 RedisOSS를 사용하여 2단계에서 복제 그룹에 대한 Amazon SNS 주제를 추가할 때 클러스터를 선택하는 대신 복제 그룹을 선택합니다. 그런 다음 동일한 나머지 단계를 따릅니다.

> **Note**
>
> 이 프로세스를 사용하여 Amazon SNS 주제를 수정할 수도 있습니다.

클러스터에 대한 Amazon SNS 주제를 추가하거나 수정하려면(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 클러스터 에서 Amazon SNS 주제를 추가하거나 수정할 클러스터를 선택합니다ARN.
3. 수정을 선택합니다.
4. SNS 알림 주제 의 클러스터 수정에서 추가할 SNS 주제를 선택하거나 수동 ARN 입력을 선택하고 Amazon SNS 주제ARN의 를 입력합니다.
5. 수정을 선택합니다.

Amazon SNS 주제 추가(AWS CLI)

클러스터에 대한 Amazon SNS 주제를 추가하거나 수정하려면 AWS CLI 명령 을 사용합니다`modify-cache-cluster`.

다음 코드 예제는 Amazon SNS 주제 arn을 my-cluster 에 추가합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-cache-cluster \
    --cache-cluster-id my-cluster \
    --notification-topic-arn arn:aws:sns:us-
west-2:123456789xxx:ElastiCacheNotifications
```

Windows의 경우:

```
aws elasticache modify-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --notification-topic-arn arn:aws:sns:us-west-2:123456789xx:ElastiCacheNotifications
```

자세한 내용은 modify-cache-cluster를 참조하세요.

Amazon SNS 주제 추가(ElastiCache API)

클러스터에 대한 Amazon SNS 주제를 추가하거나 수정하려면 다음 파라미터를 사용하여 `ModifyCacheCluster` 작업을 호출합니다.

- `CacheClusterId=my-cluster`
- `TopicArn=arn%3Aaws%3Asns%3Aus-west-2%3A565419523791%3AElastiCacheNotifications`

Example

```
https://elasticache.amazon.com/
    ?Action=ModifyCacheCluster
    &ApplyImmediately=false
    &CacheClusterId=my-cluster
    &NotificationTopicArn=arn%3Aaws%3Asns%3Aus-
west-2%3A565419523791%3AElastiCacheNotifications
    &Version=2014-12-01
    &SignatureVersion=4
```

```
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

자세한 내용은 ModifyCacheCluster를 참조하세요.

Amazon SNS 알림 활성화 및 비활성화

클러스터에 대해 알림을 켜거나 끌 수 있습니다. 다음 절차에서는 Amazon SNS 알림을 비활성화하는 방법을 보여줍니다.

Amazon SNS 알림 활성화 및 비활성화(콘솔)

를 사용하여 Amazon SNS 알림을 비활성화하려면 AWS Management Console

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. Memcached를 실행 중인 클러스터의 목록을 보려면 탐색 창에서 Memcached를 선택합니다.

   Valkey 또는 Redis를 실행하는 클러스터 목록을 보려면 OSS탐색 창에서 Valkey 또는 Redis를 OSS선택합니다.
3. 알림을 수정할 클러스터의 이름 왼쪽에 있는 확인란을 선택합니다.
4. 수정을 선택합니다.
5. SNS 알림 주제 아래의 클러스터 수정에서 알림 비활성화를 선택합니다.
6. 수정을 선택합니다.

Amazon SNS 알림 활성화 및 비활성화(AWS CLI)

Amazon SNS 알림을 비활성화하려면 다음 파라미터와 `modify-cache-cluster` 함께 명령을 사용합니다.

Linux, macOS, Unix의 경우:

```
aws elasticache modify-cache-cluster \
```

```
    --cache-cluster-id my-cluster \
    --notification-topic-status inactive
```

Windows의 경우:

```
aws elasticache modify-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --notification-topic-status inactive
```

Amazon SNS 알림 활성화 및 비활성화(ElastiCache API)

Amazon SNS 알림을 비활성화하려면 다음 파라미터를 사용하여 `ModifyCacheCluster` 작업을 호출합니다.

- `CacheClusterId=my-cluster`
- `NotificationTopicStatus=inactive`

이 호출은 다음과 비슷한 출력을 반환합니다.

Example

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyCacheCluster
    &ApplyImmediately=false
    &CacheClusterId=my-cluster
    &NotificationTopicStatus=inactive
    &Version=2014-12-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

ElastiCache 이벤트 보기

ElastiCache 는 클러스터 인스턴스, 보안 그룹 및 파라미터 그룹과 관련된 이벤트를 기록합니다. 여기에는 이벤트 날짜 및 시간, 이벤트의 원본 이름 및 원본 유형, 이벤트 설명 등의 정보가 포함됩니다. ElastiCache 콘솔, AWS CLI `describe-events` 명령 또는 작업 을 사용하여 로그에서 이벤트를 쉽게 검색할 수 있습니다 ElastiCache API`DescribeEvents`.

다음 절차에서는 지난 24시간(1,440분) 동안의 모든 ElastiCache 이벤트를 보는 방법을 보여줍니다.

ElastiCache 이벤트 보기(콘솔)

다음 절차는 ElastiCache 콘솔을 사용하여 이벤트를 표시합니다.

ElastiCache 콘솔을 사용하여 이벤트를 보려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 사용 가능한 모든 이벤트의 목록을 보려면 탐색 창에서 이벤트를 선택합니다.

   이벤트 화면에서 목록의 각 행은 하나의 이벤트를 나타내며 이벤트 소스, 이벤트 유형(캐시 클러스터 cache-parameter-group cache-security-group, 또는 cache-subnet-group), 이벤트 GMT 시간 및 이벤트 설명을 표시합니다.

   [Filter]를 사용하여 이벤트 목록에서 모든 이벤트를 볼지 특정 유형의 이벤트만 볼지를 지정할 수 있습니다.

ElastiCache 이벤트 보기(AWS CLI)

를 사용하여 ElastiCache 이벤트 목록을 생성하려면 명령 을 AWS CLI사용합니다`describe-events`. 선택적 파라미터를 사용하여 나열된 이벤트의 유형, 나열된 이벤트의 기간, 나열할 이벤트의 최대 수 등을 제어할 수 있습니다.

다음 코드는 최대 40개의 캐시 클러스터 이벤트를 나열합니다.

```
aws elasticache describe-events --source-type cache-cluster --max-items 40
```

다음 코드는 지난 24시간(1440분) 동안의 모든 이벤트를 나열합니다.

```
aws elasticache describe-events --source-type cache-cluster --duration 1440
```

describe-events 명령의 출력은 다음과 같습니다.

```
aws elasticache describe-events --source-type cache-cluster --max-items 40
{
    "Events": [
        {
            "SourceIdentifier": "my-mem-cluster",
            "SourceType": "cache-cluster",
            "Message": "Finished modifying number of nodes from 1 to 3",
            "Date": "2020-06-09T02:01:21.772Z"
        },
        {
            "SourceIdentifier": "my-mem-cluster",
            "SourceType": "cache-cluster",
            "Message": "Added cache node 0002 in availability zone us-west-2a",
            "Date": "2020-06-09T02:01:21.716Z"
        },
        {
            "SourceIdentifier": "my-mem-cluster",
            "SourceType": "cache-cluster",
            "Message": "Added cache node 0003 in availability zone us-west-2a",
            "Date": "2020-06-09T02:01:21.706Z"
        },
        {
            "SourceIdentifier": "my-mem-cluster",
            "SourceType": "cache-cluster",
            "Message": "Increasing number of requested nodes",
            "Date": "2020-06-09T01:58:34.178Z"
        },
        {
            "SourceIdentifier": "mycluster-0003-004",
            "SourceType": "cache-cluster",
            "Message": "Added cache node 0001 in availability zone us-west-2c",
            "Date": "2020-06-09T01:51:14.120Z"
        },
        {
            "SourceIdentifier": "mycluster-0003-004",
            "SourceType": "cache-cluster",
            "Message": "This cache cluster does not support persistence (ex:
 'appendonly').  Please use a different instance type to enable persistence.",
            "Date": "2020-06-09T01:51:14.095Z"
        },
        {
            "SourceIdentifier": "mycluster-0003-004",
```

```
            "SourceType": "cache-cluster",
            "Message": "Cache cluster created",
            "Date": "2020-06-09T01:51:14.094Z"
        },
        {
            "SourceIdentifier": "mycluster-0001-005",
            "SourceType": "cache-cluster",
            "Message": "Added cache node 0001 in availability zone us-west-2b",
            "Date": "2020-06-09T01:42:55.603Z"
        },
        {
            "SourceIdentifier": "mycluster-0001-005",
            "SourceType": "cache-cluster",
            "Message": "This cache cluster does not support persistence (ex:
 'appendonly').  Please use a different instance type to enable persistence.",
            "Date": "2020-06-09T01:42:55.576Z"
        },
        {
            "SourceIdentifier": "mycluster-0001-005",
            "SourceType": "cache-cluster",
            "Message": "Cache cluster created",
            "Date": "2020-06-09T01:42:55.574Z"
        },
        {
            "SourceIdentifier": "mycluster-0001-004",
            "SourceType": "cache-cluster",
            "Message": "Added cache node 0001 in availability zone us-west-2b",
            "Date": "2020-06-09T01:28:40.798Z"
        },
        {
            "SourceIdentifier": "mycluster-0001-004",
            "SourceType": "cache-cluster",
            "Message": "This cache cluster does not support persistence (ex:
 'appendonly').  Please use a different instance type to enable persistence.",
            "Date": "2020-06-09T01:28:40.775Z"
        },
        {
            "SourceIdentifier": "mycluster-0001-004",
            "SourceType": "cache-cluster",
            "Message": "Cache cluster created",
            "Date": "2020-06-09T01:28:40.773Z"
        }
    ]
```

```
}
```

사용 가능한 파라미터 및 허용된 파라미터 값과 같은 자세한 내용은 describe-events를 참조하세요.

ElastiCache 이벤트 보기(ElastiCache API)

를 사용하여 ElastiCache 이벤트 목록을 생성하려면 DescribeEvents 작업을 ElastiCache API사용합니다. 선택적 파라미터를 사용하여 나열된 이벤트의 유형, 나열된 이벤트의 기간, 나열할 이벤트의 최대 수 등을 제어할 수 있습니다.

다음 코드는 40개의 최신 cache-cluster 이벤트를 나열합니다.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeEvents
   &MaxRecords=40
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &SourceType=cache-cluster
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

다음 코드는 지난 24시간(1440분) 동안의 cache-cluster 이벤트를 나열합니다.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeEvents
   &Duration=1440
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &SourceType=cache-cluster
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

위 작업을 통해 다음과 비슷한 출력이 생성되어야 합니다.

```
<DescribeEventsResponse xmlns="http://elasticache.amazonaws.com/doc/2015-02-02/">
    <DescribeEventsResult>
        <Events>
```

```
            <Event>
                <Message>Cache cluster created</Message>
                <SourceType>cache-cluster</SourceType>
                <Date>2015-02-02T18:22:18.202Z</Date>
                <SourceIdentifier>mem01</SourceIdentifier>
            </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 DescribeEvents를 참조하세요.

이벤트 알림 및 Amazon SNS

ElastiCache 는 캐시 클러스터에서 중요한 이벤트가 발생할 때 Amazon Simple Notification Service(SNS)를 사용하여 메시지를 게시할 수 있습니다. 이 기능은 캐시 클러스터의 개별 캐시 노드 Endpoint에 연결된 클라이언트 머신의 서버 목록을 새로 고침하는 데 사용될 수 있습니다.

> **Note**
>
> 요금 및 Amazon SNS 설명서 링크에 대한 정보를 포함하여 Amazon Simple Notification Service(SNS)에 대한 자세한 내용은 Amazon SNS 제품 페이지를 참조하세요.

알림은 지정된 Amazon SNS 주제 에 게시됩니다. 다음은 알림에 대한 요구 사항입니다.

- ElastiCache 알림에 대해 하나의 주제만 구성할 수 있습니다.
- Amazon SNS 주제를 소유한 AWS 계정은 알림이 활성화된 캐시 클러스터를 소유한 계정과 동일해야 합니다.
- 게시하려는 Amazon SNS 주제는 암호화할 수 없습니다.

> **Note**
>
> 암호화된(휴지 중) Amazon SNS 주제를 클러스터에 연결할 수 있습니다. 그러나 ElastiCache 콘솔의 주제 상태는 비활성으로 표시되며, 가 메시지를 주제에 ElastiCache 푸시할 때 클러스터에서 주제의 연결을 효과적으로 해제합니다.

- Amazon SNS 주제는 ElastiCache 클러스터와 동일한 리전에 있어야 합니다.

ElastiCache 이벤트

다음 ElastiCache 이벤트는 Amazon SNS 알림을 트리거합니다. 이벤트 세부 정보에 대한 자세한 내용은 ElastiCache 이벤트 보기 섹션을 참조하세요.

이벤트 이름	메시지	설명
ElastiCache:AddCacheNodeComplete	`ElastiCache:AddCacheNodeComplete : cache-cluster`	캐시 노드가 캐시 클러스터에 추가되었고 사용할 준비가 되어 있습니다.
ElastiCache:사용 가능한 IP 주소가 충분하지 AddCacheNodeFailed 않아서	`ElastiCache:AddCacheNodeFailed : cluster-name`	사용 가능한 IP 주소가 충분하지 않아 캐시 노드를 추가하지 못했습니다.
ElastiCache:CacheClusterParametersChanged	`ElastiCache:CacheClusterParametersChanged : cluster-name`	하나 이상의 캐시 클러스터 파라미터가 변경되었습니다.
ElastiCache:CacheClusterProvisioningComplete	`ElastiCache:CacheClusterProvisioningComplete cluster-name-0001-005`	캐시 클러스터의 프로비저닝이 완료되어 캐시 클러스터에 있는 캐시 노드를 사용할 수 있습니다.
ElastiCache:호환되지 않는 네트워크 상태로 CacheClusterProvisioningFailed 인해	`ElastiCache:CacheClusterProvisioningFailed : cluster-name`	새 캐시 클러스터를 존재하지 않는 가상 프라이빗 클라우드()로 시작하려고 했습니다VPC.

이벤트 이름	메시지	설명
ElastiCache:CacheClusterScalingComplete	`CacheClusterScalingComplete : cluster-name`	캐시 클러스터의 조정이 성공적으로 완료되었습니다.
ElastiCache:CacheClusterScalingFailed	ElastiCache:CacheClusterScalingFailed : *cluster-name*	캐시 클러스터에 대한 스케일업 작업이 실패했습니다.
ElastiCache:CacheClusterSecurityGroupModified	`ElastiCache:CacheClusterSecurityGroupModified : cluster-name`	다음 이벤트 중 하나가 발생했습니다. • 캐시 클러스터를 위한 승인된 캐시 보안 그룹이 수정되었습니다. • 캐시 클러스터와 연결된 캐시 EC2 보안 그룹에 대해 하나 이상의 새 보안 그룹이 승인되었습니다. • 캐시 클러스터와 연결된 캐시 EC2 보안 그룹에서 하나 이상의 보안 그룹이 취소되었습니다.

이벤트 이름	메시지	설명
ElastiCache:CacheNodeReplaceStarted	`ElastiCache:CacheNodeReplaceStarted : cluster-name`	ElastiCache 에서 캐시 노드를 실행하는 호스트가 성능 저하되었거나 연결할 수 없음을 감지하여 캐시 노드를 교체하기 시작했습니다. **Note** 교체된 캐시 노드의 DNS 항목은 변경되지 않습니다. 대부분의 경우에 이 이벤트가 발생할 때 클라이언트의 서버 목록을 새로 고침하지 않아도 됩니다. 그러나 일부 캐시 클라이언트 라이브러리 ElastiCache 는 가 캐시 노드를 교체한 후에도 캐시 노드 사용을 중지할 수 있습니다. 이 경우 이 이벤트가 발생할 때 애플리케이션은 서버 목록을 새로 고쳐야 합니다.

이벤트 이름	메시지	설명
ElastiCache:CacheNodeReplaceComplete	`ElastiCache:CacheNodeReplaceComplete : cluster-name`	ElastiCache 에서 캐시 노드를 실행하는 호스트가 성능 저하되거나 연결할 수 없음을 감지하고 캐시 노드 교체를 완료했습니다. **Note** 교체된 캐시 노드의 DNS 항목은 변경되지 않습니다. 대부분의 경우에 이 이벤트가 발생할 때 클라이언트의 서버 목록을 새로 고침하지 않아도 됩니다. 그러나 일부 캐시 클라이언트 라이브러리 ElastiCache 는 가 캐시 노드를 교체한 후에도 캐시 노드 사용을 중지할 수 있습니다. 이 경우 이 이벤트가 발생할 때 애플리케이션은 서버 목록을 새로 고쳐야 합니다.
ElastiCache:CacheNodesRebooted	`ElastiCache:CacheNodesRebooted : cluster-name`	하나 이상의 캐시 노드가 재부팅되었습니다. 메시지(Memcached): `"Cache node %s shutdown"` , 두 번째 메시지: `"Cache node %s restarted"`

이벤트 이름	메시지	설명
ElastiCache:CertificateRenewalComplete (Valkey 또는 RedisOSS만 해당)	`ElastiCache:CertificateRenewalComplete`	Amazon CA 인증서가 성공적으로 갱신되었습니다.
ElastiCache:CreateReplicationGroupComplete	`ElastiCache:CreateReplicationGroupComplete : cluster-name`	복제 그룹이 성공적으로 생성되었습니다.
ElastiCache:DeleteCacheClusterComplete	`ElastiCache:DeleteCacheClusterComplete : cluster-name`	캐시 클러스터 및 연결된 모든 캐시 노드 삭제를 완료했습니다.
ElastiCache:FailoverComplete (Valkey 또는 RedisOSS만 해당)	`ElastiCache:FailoverComplete : mycluster`	복제본 노드에 대한 장애 조치가 성공했습니다.
ElastiCache:ReplicationGroupIncreaseReplicaCountFinished	`ElastiCache:ReplicationGroupIncreaseReplicaCountFinished : cluster-name-0001-005`	클러스터의 복제본 수가 증가했습니다.
ElastiCache:ReplicationGroupIncreaseReplicaCountStarted	`ElastiCache:ReplicationGroupIncreaseReplicaCountStarted : cluster-name-0003-004`	클러스터에 복제본을 추가하는 프로세스가 시작되었습니다.
ElastiCache:NodeReplacementCanceled	`ElastiCache:NodeReplacementCanceled : cluster-name`	교체가 예약되어 있는 클러스터의 노드가 더 이상 교체 예약이 되지 않습니다.

이벤트 이름	메시지	설명
ElastiCache:NodeReplacementRescheduled	`ElastiCache:NodeReplacementRescheduled : cluster-name`	이전에 교체가 예약되어 있는 클러스터의 노드가 알림에 설명된 새 기간 동안 교체가 예약됩니다. 수행할 수 있는 작업에 대한 자세한 내용은 노드 교체(Valkey 및 RedisOSS) 섹션을 참조하세요.
ElastiCache:NodeReplacementScheduled	`ElastiCache:NodeReplacementScheduled : cluster-name`	클러스터의 노드가 알림에 설명된 기간 동안 교체가 예약됩니다. 수행할 수 있는 작업에 대한 자세한 내용은 노드 교체(Valkey 및 RedisOSS) 섹션을 참조하세요.
ElastiCache:RemoveCacheNodeComplete	`ElastiCache:RemoveCacheNodeComplete : cluster-name`	캐시 노드가 캐시 클러스터에서 제거되었습니다.
ElastiCache:ReplicationGroupScalingComplete	`ElastiCache:ReplicationGroupScalingComplete : cluster-name`	복제 그룹에 대한 확장 작업이 성공적으로 완료되었습니다.
ElastiCache:ReplicationGroupScalingFailed	`"Failed applying modification to cache node type to %s."`	복제 그룹에 대한 확장 작업이 실패했습니다.
ElastiCache:ServiceUpdateAvailableForNode	`"Service update is available for cache node %s."`	노드에서 셀프 서비스 업데이트를 사용할 수 있습니다.

이벤트 이름	메시지	설명
ElastiCache:SnapshotComplete (Valkey 또는 RedisOSS만 해당)	`ElastiCache:SnapshotComplete : cluster-name`	캐시 스냅샷이 성공적으로 완료되었습니다.
ElastiCache:SnapshotFailed (발키 또는 RedisOSS만 해당)	`SnapshotFailed : cluster-name`	캐시 스냅샷이 실패했습니다. 원인에 대한 자세한 내용은 클러스터의 캐시 이벤트를 참조하세요. 스냅샷을 설명할 경우 `DescribeSnapshots` 를 참조하세요. 상태는 `failed`입니다.

관련 주제

- ElastiCache 이벤트 보기

로그 전달

> **Note**
>
> 느린 로그는 Valkey 7.x 이상 및 Redis OSS 캐시 클러스터 및 복제 그룹에서 엔진 버전 6.0 이상을 사용하여 지원됩니다.
> 엔진 로그는 엔진 버전 6.2 이상을 사용하는 Valkey 7.x 및 Redis OSS 캐시 클러스터 및 복제 그룹에 대해 지원됩니다.

로그 전송을 사용하면 두 대상 중 하나로 SLOWLOG 또는 엔진 로그를 스트리밍할 수 있습니다.

- Amazon Data Firehose
- Amazon CloudWatch Logs

를 사용하여 클러스터를 생성하거나 수정할 때 로그 전송을 활성화하고 구성합니다 ElastiCache APIs. 각 로그 항목은 JSON 또는 두 가지 형식 중 하나로 지정된 대상으로 전달됩니다TEXT.

고정된 수의 느린 로그 항목이 엔진에서 주기적으로 검색됩니다. 엔진 파라미터 `slowlog-max-len`에 지정된 값에 따라, 추가적인 슬로우 로그 항목이 대상에 전달되지 않을 수 있습니다.

AWS 콘솔 또는 수정 중 하나를 사용하여 언제든지 전송 구성을 변경하거나 로그 전송을 비활성화하도록 선택할 수 있습니다APIsmodify-cache-clustermodify-replication-group.

모든 로그 전달 수정에서 `apply-immediately` 파라미터를 설정해야 합니다.

> **Note**
>
> Amazon CloudWatch Data Firehose로 로그가 직접 전송되더라도 로그 전송이 활성화되면 Amazon Logs 요금이 적용됩니다. 자세한 내용은 Amazon CloudWatch 요금의 Vended Logs 섹션을 참조하세요.

슬로우 로그 항목의 내용

느린 로그에는 다음 정보가 포함되어 있습니다.

- CacheClusterId – 캐시 클러스터의 ID
- CacheNodeId - 캐시 노드의 ID
- Id - 모든 슬로우 로그 항목에 대한 고유한 프로그레시브 식별자입니다.
- Timestamp – 로그에 기록된 명령이 처리된 시간의 Unix 타임스탬프입니다.
- Duration – 실행에 걸린 시간(마이크로초)입니다.
- Command – 클라이언트에서 사용된 명령입니다. 예를 들어, `set foo bar` 여기서 `foo` 는 키이고 `bar`는 값입니다. 는 민감한 데이터가 노출되지 않도록 실제 키 이름과 값을 `(2 more arguments)`로 ElastiCache 대체합니다.
- ClientAddress – 클라이언트 IP 주소 및 포트
- ClientName – `CLIENT SETNAME` 명령을 통해 설정된 경우 클라이언트 이름

엔진 로그 항목의 내용

ElastiCache 엔진 로그에는 다음 정보가 포함되어 있습니다.

- CacheClusterId - 캐시 클러스터의 ID
- CacheNodeId - 캐시 노드의 ID
- 로그 수준 - `VERBOSE("-")`, , 중 하나를 LogLevel 수행할 수 있습니 다`NOTICE("*")WARNING("#")`.
- 시간 - 로깅된 메시지의 UTC 시간입니다. 시간은 `"DD MMM YYYY hh:mm:ss.ms UTC"`와 같은 형식입니다.
- 역할(Role) - 로그가 방출되는 노드의 역할입니다. 기본 의 경우 “M”, 복제본의 경우 “S”, RDB에서 AOF 작업하는 라이터 하위 프로세스의 경우 “C”, 감시의 경우 “X” 중 하나일 수 있습니다.
- 메시지 - 엔진 로그 메시지입니다.

로깅을 구성하기 위한 권한

IAM 사용자/역할 정책에 다음 IAM 권한을 포함해야 합니다.

- `logs:CreateLogDelivery`
- `logs:UpdateLogDelivery`
- `logs:DeleteLogDelivery`
- `logs:GetLogDelivery`
- `logs:ListLogDeliveries`

자세한 내용은 액세스 관리 개요: 권한 및 정책 섹션을 참조하세요.

로그 유형 및 로그 형식 지정

슬로우 로그

느린 로그는 JSON 및 를 모두 지원합니다. TEXT

다음은 JSON 형식 예제를 보여줍니다.

```
{
  "CacheClusterId": "logslowxxxxmsxj",
  "CacheNodeId": "0001",
  "Id": 296,
  "Timestamp": 1605631822,
  "Duration (us)": 0,
```

```
  "Command": "GET ... (1 more arguments)",
  "ClientAddress": "192.168.12.104:55452",
  "ClientName": "logslowxxxxmsxj##"
}
```

다음은 TEXT 형식 예제를 보여줍니다.

```
logslowxxxxmsxj,0001,1605631822,30,GET ... (1 more
 arguments),192.168.12.104:55452,logslowxxxxmsxj##
```

엔진 로그

엔진 로그는 JSON 및 를 모두 지원합니다. TEXT

다음은 JSON 형식 예제를 보여줍니다.

```
{
  "CacheClusterId": "xxxxxxxxxzy-engine-log-test",
  "CacheNodeId": "0001",
  "LogLevel": "VERBOSE",
  "Role": "M",
  "Time": "12 Nov 2020 01:28:57.994 UTC",
  "Message": "Replica is waiting for next BGSAVE before synchronizing with the primary.
 Check back later"
}
```

다음은 TEXT 형식 예제를 보여줍니다.

```
xxxxxxxxxxxzy-engine-log-test/0001:M 29 Oct 2020 20:12:20.499 UTC * A slow-running Lua
 script detected that is still in execution after 10000 milliseconds.
```

ElastiCache 로깅 대상

이 섹션에서는 ElastiCache 로그에 대해 선택할 수 있는 로깅 대상에 대해 설명합니다. 각 섹션에서는 대상 유형에 대한 로깅을 구성하기 위한 지침과 대상 유형과 관련된 모든 동작에 대한 정보를 제공합니다. 로깅 대상을 구성한 후 ElastiCache 로깅 구성에 사양을 제공하여 로깅을 시작할 수 있습니다.

주제

- Amazon CloudWatch Logs

- Amazon Data Firehose

Amazon CloudWatch Logs

- CloudWatch 로그가 전달될 로그 로그 그룹을 지정합니다.
- 여러 Valkey 또는 Redis OSS 클러스터 및 복제 그룹의 로그를 동일한 로그 그룹에 전달할 수 있습니다.
- 캐시 클러스터 또는 복제 그룹 내의 각 노드에 대해 새 로그 스트림이 생성되고 로그는 해당 로그 스트림으로 전달됩니다. 로그 스트림 이름에는 `elasticache/${engine-name}/${cache-cluster-id}/${cache-node-id}/${log-type}` 형식이 사용됩니다.

CloudWatch 로그에 로그를 게시할 수 있는 권한

CloudWatch 로그 로그 로그 그룹에 로그를 보내 ElastiCache 도록 를 구성하려면 다음 권한 설정이 있어야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "ElastiCacheLogging"
        },
        {
            "Sid": "ElastiCacheLoggingCWL",
            "Action": [
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups"
```

```
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

자세한 내용은 로그로 전송된 CloudWatch 로그를 참조하세요.

Amazon Data Firehose

- 로그를 전송할 Firehose 전송 스트림을 지정합니다.
- 여러 Valkey 또는 Redis OSS 클러스터 및 복제 그룹의 로그를 동일한 전송 스트림으로 전송할 수 있습니다.
- 캐시 클러스터 또는 복제 그룹 내에서 각 노드의 로그가 동일한 전송 스트림으로 전달됩니다. 각 로그 메시지에 포함된 `cache-cluster-id` 및 `cache-node-id`를 기반으로 서로 다른 캐시 노드의 로그 메시지를 구분할 수 있습니다.
- Firehose에 대한 로그 전송은 현재 아시아 태평양(오사카) 리전에서 사용할 수 없습니다.

Firehose에 로그를 게시할 수 있는 권한

Amazon Kinesis Data Firehose 전송 스트림 ElastiCache 으로 로그를 전송하도록 를 구성하려면 다음 권한이 있어야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries"
            ],
            "Resource": [
                "*"
```

```
            ],
            "Effect": "Allow",
            "Sid": "ElastiCacheLogging"
        },
        {
            "Sid": "ElastiCacheLoggingFHSLR",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Sid": "ElastiCacheLoggingFH",
            "Action": [
                "firehose:TagDeliveryStream"
            ],
            "Resource": "Amazon Kinesis Data Firehose delivery stream ARN",
            "Effect": "Allow"
        }
    ]
}
```

콘솔을 사용하여 로그 전달 지정

를 사용하여 의 단계에 따라 Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터를 생성Valkey(클러스터 모드 비활성화됨) 클러스터 생성(콘솔)하거나 의 단계에 따라 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터를 생성할 AWS Management Console 수 있습니다Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터 생성(콘솔). 두 경우 모두 다음을 수행하여 로그 전달을 구성합니다.

1. 고급 설정 에서 로그를 선택한 다음 느린 로그 또는 엔진 로그 를 확인합니다.
2. 로그 형식에서 텍스트 또는 를 선택합니다JSON.
3. 대상 유형에서 CloudWatch 로그 또는 Kinesis Firehose를 선택합니다.
4. 로그 대상 에서 새로 생성을 선택하고 Amazon S3 버킷 이름, CloudWatchLogs 로그 그룹 이름 또는 Kinesis Data Firehose 스트림 이름을 입력하거나 기존 선택을 선택한 다음 CloudWatch 로그 그룹 이름 또는 Kinesis Data Firehose 스트림 이름,

클러스터 수정 시:

로그 전달을 활성화/비활성화하거나 대상 유형, 형식 또는 대상을 변경할 수 있습니다.

1. 콘솔에 로그인하고 에서 ElastiCache 콘솔을 엽니다https://console.aws.amazon.com/elasticache/.
2. 탐색 창에서 Valkey 클러스터 또는 Redis OSS 클러스터를 선택합니다.
3. 클러스터 목록에서 수정할 클러스터를 선택합니다. 클러스터 이름을 선택합니다(그 옆의 확인란 아님).
4. 클러스터 이름 페이지에서 로그 탭을 선택합니다.
5. 슬로우 로그를 활성화/비활성화하려면 슬로우 로그 활성화 또는 슬로우 로그 비활성화를 선택합니다.
6. 엔진 로그를 사용 설정/사용 중지하려면 엔진 로그 사용 설정(Enable engine logs)또는엔진 로그 사용 중지(Disable engine logs) 중 하나를 선택합니다.
7. 구성을 변경하려면 슬로우 로그 수정(Modify slow logs) 또는 엔진 로그 수정(Modify engine logs) 중 하나를 선택합니다.
   - 대상 유형에서 CloudWatch 로그 또는 Kinesis Firehose를 선택합니다.
   - 로그 대상 에서 새로 생성을 선택하고 CloudWatchLogs 로그 그룹 이름 또는 Kinesis Data Firehose 스트림 이름을 입력합니다. 또는 기존 선택을 선택한 다음 CloudWatchLogs 로그 그룹 이름 또는 Kinesis Data Firehose 스트림 이름을 선택합니다.

를 사용하여 로그 전송 지정 AWS CLI

슬로우 로그

로그에 대한 느린 로그 전송을 사용하여 복제 그룹을 생성합니다 CloudWatch .

Linux, macOS, Unix의 경우:

```
aws elasticache create-replication-group \
    --replication-group-id test-slow-log \
    --replication-group-description test-slow-log \
    --engine redis \
    --cache-node-type cache.r5.large \
    --num-cache-clusters 2 \
    --log-delivery-configurations '{
        "LogType":"slow-log",
        "DestinationType":"cloudwatch-logs",
```

```
        "DestinationDetails":{
          "CloudWatchLogsDetails":{
            "LogGroup":"my-log-group"
          }
        },
        "LogFormat":"json"
      }'
```

Windows의 경우:

```
aws elasticache create-replication-group ^
    --replication-group-id test-slow-log ^
    --replication-group-description test-slow-log ^
    --engine redis ^
    --cache-node-type cache.r5.large ^
    --num-cache-clusters 2 ^
    --log-delivery-configurations '{
        "LogType":"slow-log",
        "DestinationType":"cloudwatch-logs",
        "DestinationDetails":{
          "CloudWatchLogsDetails":{
            "LogGroup":"my-log-group"
          }
        },
        "LogFormat":"json"
      }'
```

로그에 느린 로그를 전달하도록 복제 그룹 수정 CloudWatch

Linux, macOS, Unix의 경우:

```
aws elasticache modify-replication-group \
    --replication-group-id test-slow-log \
    --apply-immediately \
    --log-delivery-configurations '
    {
      "LogType":"slow-log",
      "DestinationType":"cloudwatch-logs",
      "DestinationDetails":{
        "CloudWatchLogsDetails":{

          "LogGroup":"my-log-group"
        }
```

```
      },
      "LogFormat":"json"
    }'
```

Windows의 경우:

```
aws elasticache modify-replication-group ^
    --replication-group-id test-slow-log ^
    --apply-immediately ^
    --log-delivery-configurations '
    {
      "LogType":"slow-log",
      "DestinationType":"cloudwatch-logs",
      "DestinationDetails":{
        "CloudWatchLogsDetails":{
          "LogGroup":"my-log-group"
        }
      },
      "LogFormat":"json"
    }'
```

슬로우 로그 전달을 비활성화하도록 복제 그룹 수정

Linux, macOS, Unix의 경우:

```
aws elasticache modify-replication-group \
    --replication-group-id test-slow-log \
    --apply-immediately \
    --log-delivery-configurations '
    {
      "LogType":"slow-log",
      "Enabled":false
    }'
```

Windows의 경우:

```
aws elasticache modify-replication-group ^
    --replication-group-id test-slow-log ^
    --apply-immediately ^
    --log-delivery-configurations '
    {
      "LogType":"slow-log",
```

```
      "Enabled":false
    }'
```

엔진 로그

로그에 엔진 로그를 전달하는 복제 그룹을 생성합니다 CloudWatch .

Linux, macOS, Unix의 경우:

```
aws elasticache create-replication-group \
    --replication-group-id test-slow-log \
    --replication-group-description test-slow-log \
    --engine redis \
    --cache-node-type cache.r5.large \
    --num-cache-clusters 2 \
    --log-delivery-configurations '{
        "LogType":"engine-log",
        "DestinationType":"cloudwatch-logs",
        "DestinationDetails":{
          "CloudWatchLogsDetails":{
            "LogGroup":"my-log-group"
          }
        },
        "LogFormat":"json"
      }'
```

Windows의 경우:

```
aws elasticache create-replication-group ^
    --replication-group-id test-slow-log ^
    --replication-group-description test-slow-log ^
    --engine redis ^
    --cache-node-type cache.r5.large ^
    --num-cache-clusters 2 ^
    --log-delivery-configurations '{
        "LogType":"engine-log",
        "DestinationType":"cloudwatch-logs",
        "DestinationDetails":{
          "CloudWatchLogsDetails":{
            "LogGroup":"my-log-group"
          }
        },
```

```
        "LogFormat":"json"
      }'
```

복제 그룹을 수정하여 Firehose에 엔진 로그 전송

Linux, macOS, Unix의 경우:

```
aws elasticache modify-replication-group \
    --replication-group-id test-slow-log \
    --apply-immediately \
    --log-delivery-configurations '
    {
      "LogType":"engine-log",
      "DestinationType":"kinesis-firehose",
      "DestinationDetails":{
      "KinesisFirehoseDetails":{
         "DeliveryStream":"test"
       }
     },
      "LogFormat":"json"
    }'
```

Windows의 경우:

```
aws elasticache modify-replication-group ^
    --replication-group-id test-slow-log ^
    --apply-immediately ^
    --log-delivery-configurations '
    {
      "LogType":"engine-log",
      "DestinationType":"kinesis-firehose",
      "DestinationDetails":{
      "KinesisFirehoseDetails":{
         "DeliveryStream":"test"
       }
      },
      "LogFormat":"json"
    }'
```

복제 그룹을 수정하여 엔진 형식으로 전환

Linux, macOS, Unix의 경우:

```
aws elasticache modify-replication-group \
    --replication-group-id test-slow-log \
    --apply-immediately \
    --log-delivery-configurations '
    {
       "LogType":"engine-log",
       "LogFormat":"json"
    }'
```

Windows의 경우:

```
aws elasticache modify-replication-group ^
    --replication-group-id test-slow-log ^
    --apply-immediately ^
    --log-delivery-configurations '
    {
       "LogType":"engine-log",
       "LogFormat":"json"
    }'
```

복제 그룹을 수정하여 엔진 로그 전달 사용 중지

Linux, macOS, Unix의 경우:

```
aws elasticache modify-replication-group \
    --replication-group-id test-slow-log \
    --apply-immediately \
    --log-delivery-configurations '
    {
      "LogType":"engine-log",
      "Enabled":false
    }'
```

Windows의 경우:

```
aws elasticache modify-replication-group ^
    --replication-group-id test-slow-log ^
    --apply-immediately ^
    --log-delivery-configurations '
    {
      "LogType":"engine-log",
```

```
    "Enabled":false
  }'
```

CloudWatch 지표 사용 모니터링

ElastiCache 는 클러스터를 모니터링할 수 있는 지표를 제공합니다. 를 통해 이러한 지표에 액세스할 수 있습니다 CloudWatch. 에 대한 자세한 내용은 설명서를 CloudWatch참조하세요. CloudWatch

ElastiCache 는 호스트 수준 지표(예: CPU 사용량)와 캐시 엔진 소프트웨어에 고유한 지표(예: 캐시 가져오기 및 캐시 누락)를 모두 제공합니다. 이러한 지표는 60초 간격으로 각 캐시 노드에 대해 측정되어 게시됩니다.

> **Important**
>
> 캐시 클러스터의 성능이 저하되기 시작하면 알림을 받을 수 있도록 특정 키 지표에 대한 CloudWatch 경보 설정을 고려해야 합니다. 자세한 내용은 이 안내서의 어떤 지표를 모니터링 해야 합니까? 섹션을 참조하세요.

주제

- 호스트 수준 지표
- Valkey 및 Redis에 대한 지표 OSS
- Memcached 지표
- 어떤 지표를 모니터링해야 합니까?
- 지표 통계 및 기간 선택
- CloudWatch 클러스터 및 노드 지표 모니터링

호스트 수준 지표

AWS/ElastiCache 네임스페이스에는 다음과 같이 개별 캐시 노드에 대한 호스트 수준 지표가 포함되어 있습니다. 이러한 지표는 60초 간격으로 각 캐시 노드에 대해 측정되어 게시됩니다.

참고 항목

- Valkey 및 Redis에 대한 지표 OSS

지표	설명	단위
CPUUtilization	전체 호스트의 CPU 사용률 백분율입니다. Valkey 및 RedisOSS는 단일 스레드이므로 EngineCPUUtilization 4개 이상의 노드에 대한 지표를 모니터링하는 것이 좋습니다 vCPUs.	%
CPUCreditBalance	인스턴스가 시작되거나 시작된 이후 발생한 적립 CPU 크레딧 수입니다. T2 Standard의 경우에는 누적된 시작 크레딧 수CPUCreditBalance도 포함됩니다. 크레딧은 획득 이후에 크레딧 밸런스에 누적되고, 소비 시 크레딧 밸런스에서 소멸됩니다. 크레딧 밸런스는 최대 한도(인스턴스 크기에 따라 결정)가 있습니다. 한도에 도달하면 새로 획득한 크레딧이 모두 삭제됩니다. T2 스탠다드의 경우 시작 크레딧은 한도에 포함되지 않습니다. 의 크레딧은 인스턴스가 기준 CPU 사용률을 초과하여 버스트하는 데 사용할 CPUCredit Balance 수 있습니다. CPU 크레딧 지표는 5분 간격으로만 사용할 수 있습니다. T2 성능 순간 확장 가능 인스턴스에는 이러한 지표를 사용할 수 없습니다.	크레딧(vCPU-분)
CPUCreditUsage	인스턴스에서 CPU 사용 목적으로 사용한 CPU 크레딧 수입니다. 1 CPU 크레딧은 1분 동안 100% 사용률로 실행되는 1vCPU 또는 vCPUs, 사용률 및 시간의 동등한 조합(예: 2분 동안 50% 사용률로 실행되는 1vCPU 또는 2분 동안 25% 사용률로 vCPUs 실행되는 2v)과 같습니다.	크레딧(vCPU-분)

지표	설명	단위
	CPU 크레딧 지표는 5분 간격으로만 사용할 수 있습니다. 5분 이상의 시간을 지정할 경우 Sum 통계 대신 Average 통계를 사용하세요. T2 성능 순간 확장 가능 인스턴스에는 이러한 지표를 사용할 수 없습니다.	
FreeableMemory	호스트에서 사용 가능한 메모리의 양입니다. 이는 OS가 무료로 보고하는 RAM, 버퍼 및 캐시에서 파생됩니다.	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 대기열에 추가되거나 손실된 패킷 수입니다.	개수
NetworkConntrackAllowanceExceeded	연결 추적에서 인스턴스의 최대값이 초과되어 새 연결을 설정하지 못했기 때문에 손실된 패킷 수입니다. 이로 인해 인스턴스의 수신 또는 송신 트래픽에 대한 패킷 손실이 발생할 수 있습니다.	개수

지표	설명	단위
NetworkBandwidthOutAllowanceExceeded	아웃바운드 집계 대역폭이 인스턴스의 최대값을 초과하여 대기열에 추가되거나 손실된 패킷 수입니다.	개수
NetworkPacketsPerSecondAllowanceExceeded	양방향 PPS(packets per second)가 인스턴스의 최대값을 초과하여 대기열에 있거나 삭제된 패킷 수입니다.	개수
NetworkMaxBytesIn	분당 수신된 바이트의 초당 최대 버스트입니다.	바이트
NetworkMaxBytesOut	분당 전송된 바이트의 초당 최대 버스트입니다.	바이트
NetworkMaxPacketsIn	분당 수신된 패킷의 초당 최대 버스트 수입니다.	개수
NetworkMaxPacketsOut	분당 전송된 패킷의 초당 최대 버스트입니다.	개수
SwapUsage	호스트에서 사용되는 스왑의 양입니다.	바이트

Valkey 및 Redis에 대한 지표 OSS

Amazon ElastiCache 네임스페이스에는 다음과 같은 Valkey 및 Redis OSS 지표가 포함됩니다. 이러한 지표는 Valkey 엔진을 사용할 때와 동일합니다.

ReplicationLag 및 를 제외하고 EngineCPUUtilization이러한 지표는 info 명령에서 파생됩니다. 각 지표는 캐시 노드 수준에서 계산됩니다.

info 명령에 대한 전체 설명서는 http://valkey.io/commands/info 참조하세요.

참고 항목

- 호스트 수준 지표

지표	설명	단위
ActiveDefragHits	활성 조각 모음 프로세스에서 수행된 분당 값 재할당 수입니다. 이는 active_defrag_hits의 통계에서 파생됩니다INFO.	숫자
AuthenticationFailures	AUTH 명령을 OSS 사용하여 Valkey 또는 Redis에 인증하려는 총 실패한 시도 횟수입니다. ACL LOG 명령을 사용하여 개별 인증 실패에 대한 자세한 정보를 찾을 수 있습니다. 무단 액세스 시도를 감지하려면 이에 대한 경보를 설정하는 것이 좋습니다.	개수
BytesUsedForCache	데이터 세트, 버퍼 등을 포함하여 모든 목적에 OSS 대해 Valkey 또는 Redis에서 할당한 총 바이트 수입니다.	바이트
	Dimension: Tier=Memory 를 사용하는 Valkey 또는 Redis OSS 클러스터의 경우의 데이터 계층화 ElastiCache: 메모리별 캐시에 사용되는 총 바이트 수입니다. 이는 used_memory의 통계 값입니다INFO.	바이트
	Dimension: Tier=SSD 를 사용하는 Valkey 또는 Redis OSS 클러스터의 경우의 데이터 계층화 ElastiCache: 에서 캐시에 사용하는 총 바이트 수입니다SSD.	바이트
BytesReadFromDisk	분당 디스크에서 읽은 총 바이트 수입니다. 의 데이터 계층화 ElastiCache를 사용하는 클러스터에서만 지원됩니다.	바이트
BytesWrittenToDisk	분당 디스크에 쓴 총 바이트 수입니다. 의 데이터 계층화 ElastiCache를 사용하는 클러스터에서만 지원됩니다.	바이트

지표	설명	단위
CacheHits	기본 사전의 성공한 읽기 전용 키 조회수입니다. 이는 keyspace_hits 의 통계에서 파생됩니다INFO.	개수
CacheMisses	기본 사전의 성공하지 못한 읽기 전용 키 조회수입니다. 이는 keyspace_misses 의 통계에서 파생됩니다INFO.	개수
CommandAuthorizationFailures	사용자가 호출 권한이 없는 명령을 실행한 실패한 시도의 총 수입니다. ACL LOG 명령을 사용하여 개별 인증 실패에 대한 자세한 정보를 찾을 수 있습니다. 무단 액세스 시도를 감지하려면 이에 대한 경보를 설정하는 것이 좋습니다.	개수
CacheHitRate	Valkey 또는 Redis OSS 인스턴스의 사용 효율성을 나타냅니다. 캐시 비율이 약 0.8보다 낮으면 상당한 양의 키가 제거되거나, 만료되거나, 존재하지 않음을 의미합니다. 이는 cache_hits 및 cache_misses 통계를 사용하여 다음과 같은 방식으로 계산됩니다. cache_hits / (cache_hits + cache_misses)	%
ChannelAuthorizationFailures	사용자가 액세스 권한이 없는 채널에 액세스 실패한 시도의 총 수입니다. ACL LOG 명령을 사용하여 개별 인증 실패에 대한 자세한 정보를 찾을 수 있습니다. 무단 액세스 시도를 감지하려면 이 지표에 대한 경보를 설정하는 것이 좋습니다.	개수
CurrConnections	읽기 전용 복제본의 연결을 제외한 클라이언트 연결 수. 각 경우에 클러스터를 모니터링하기 위해 연결 중 2~4개를 ElastiCache 사용합니다. 이는 connected_clients 의 통계에서 파생됩니다INFO.	개수

지표	설명	단위
CurrItems	캐시 항목 수입니다. 이는 keyspace 통계에서 파생되어 전체 키스페이스의 모든 키를 합산합니다.	개수
	의 데이터 계층화 ElastiCache를 사용하는 클러스터용 Dimension: Tier=Memory 입니다. 메모리에 있는 항목 수입니다.	개수
	의 데이터 계층화 ElastiCache를 사용하는 클러스터용 Dimension: Tier=SSD (solid state drives)입니다. 의 항목 수입니다SSD.	개수
CurrVolatileItems	ttl이 설정된 모든 데이터베이스의 총 키 수입니다. 이는 expires 통계에서 파생되어 모든 키를 전체 키스페이스의 ttl 세트와 합산합니다.	개수
DatabaseCapacityUsagePercentage	사용 중인 클러스터용 전체 데이터 용량의 백분율입니다. 데이터 계층형 인스턴스의 경우 지표는 로 계산되며(used_memory - mem_not_counted_for_evict + SSD used) / (maxmemory + SSD total capacity) , 여기서 used_memory 및 maxmemory 는 에서 가져옵니다INFO. 다른 모든 경우 지표는 를 사용하여 계산됩니다used_memory/maxmemory .	%

지표	설명	단위
DatabaseCapacityUsageCountedForEvictPercentage	오버헤드 및 에 사용되는 메모리를 제외하고 사용 중인 클러스터의 총 데이터 용량의 백분율입니다COB. 이 지표는 다음과 같이 계산됩니다. used_memory - mem_not_counted_for_evict/maxmemory 데이터 계층형 인스턴스에서 지표는 다음과 같이 계산됩니다. (used_memory + SSD used) / (maxmemory + SSD total capacity) 여기서 used_memory 및 maxmemory 는 에서 가져옵니다. INFO	%
DatabaseMemoryUsagePercentage	사용 중인 클러스터용 메모리의 백분율입니다. 이는 used_memory/maxmemory 의 를 사용하여 계산됩니다INFO.	%
DatabaseMemoryUsageCountedForEvictPercentage	오버헤드 및 에 사용되는 메모리를 제외하고 사용 중인 클러스터의 메모리 백분율입니다COB. 이는 used_memory-mem_not_counted_for_evict/maxmemory 의 를 사용하여 계산됩니다INFO.	%
DB0AverageTTL	INFO 명령 통계avg_ttlDBO에서 keyspace 를 노출합니다. 복제본은 키를 만료 처리하지 않고 프라이머리 노드에서 키를 만료 처리할 때까지 기다립니다. 기본 노드가 키가 만료되면(또는 로 인해 제거되면LRU) 모든 복제본으로 전송되는 DEL 명령을 합성합니다. 따라서 DB0AverageTTL는 복제본 노드의 경우 키가 만료되지 않아 를 추적하지 않기 때문에 0입니다TTL.	밀리초

지표	설명	단위
EngineCPUUtilization	Valkey 또는 Redis OSS 엔진 스레드의 CPU 사용률을 제공합니다. Valkey와 RedisOSS는 단일 스레드이므로 이 지표를 사용하여 프로세스 자체의 부하를 분석할 수 있습니다. EngineCPUUtilization 지표는 프로세스에 대한 보다 정확한 가시성을 제공합니다. 지표와 함께 사용할 수 있습니다. CPUUtilization 는 다른 운영 체제 및 관리 프로세스를 포함하여 서버 인스턴스의 CPU 사용률을 전체적으로 CPUUtilization 노출합니다. 4개 vCPUs 이상의 더 큰 노드 유형의 경우 EngineCPUUtilization 지표를 사용하여 조정을 위한 임계값을 모니터링하고 설정합니다. **Note** ElastiCache 호스트에서 백그라운드 프로세스는 호스트를 모니터링하여 관리형 데이터베이스 환경을 제공합니다. 이러한 백그라운드 프로세스는 CPU 워크로드의 상당 부분을 차지할 수 있습니다. 이는 가 2개 이상인 대규모 호스트에서는 중요하지 않습니다vCPUs. 하지만 2개 vCPUs 이하의 작은 호스트에 영향을 미칠 수 있습니다. EngineCPUUtilization 지표만 모니터링하는 경우 Valkey 또는 Redis에서 사용량이 많OSS고 백그라운드 모니터링 프로세스에서 CPU 사용량이 많은 호스트CPU가 과부하되는 상황을 알지 못합니다. 따라서 CPUUtilization 2개 vCPUs 이하의 호스트에 대한 지표를 모니터링하는 것이 좋습니다.	%

지표	설명	단위
Evictions	maxmemory 제한으로 인해 제거된 키 수입니다. 이는 evicted_keys 의 통계에서 파생됩니다INFO.	개수
GlobalDatastoreReplicationLag	보조 리전의 기본 노드와 기본 리전의 기본 노드 간의 지연입니다. 클러스터 모드가 활성화된 Valkey 또는 Redis OSS의 경우 지연은 샤드 간의 최대 지연을 나타냅니다.	초
IamAuthenticationExpirations	만료된 IAM인증된 Valkey 또는 Redis OSS 연결의 총 수입니다. IAM을 사용하는 인증에 대한 자세한 내용은 사용 설명서를 참조하세요.	개수
IamAuthenticationThrottling	제한된 IAM인증된 Valkey 또는 Redis OSS AUTH 또는 HELLO 요청의 총 수입니다. IAM을 사용하는 인증에 대한 자세한 내용은 사용 설명서를 참조하세요.	개수
IsMaster	노드가 현재 샤드/클러스터의 기본 노드인지 여부를 나타냅니다. 이 지표는 0(기본 노드 아님) 또는 1(기본 노드임)일 수 있습니다.	개수
KeyAuthorizationFailures	사용자가 액세스 권한이 없는 키에 액세스한 실패한 시도의 총 수입니다. ACL LOG 명령을 사용하여 개별 인증 실패에 대한 자세한 정보를 찾을 수 있습니다. 무단 액세스 시도를 감지하려면 이에 대한 경보를 설정하는 것이 좋습니다.	개수
KeysTracked	Valkey 또는 Redis 키 추적으로 추적되는 OSS 키 수를 의 백분율로 나타낸 것입니다tracking-table-max-keys . 키 추적은 클라이언트 측 캐싱을 지원하고 키가 수정된 경우, 클라이언트에 알리는 데 사용됩니다.	개수

지표	설명	단위
MemoryFragmentationRatio	Valkey 또는 Redis OSS 엔진의 메모리 할당 효율성을 나타냅니다. 특정 임계값은 다른 동작을 나타냅니다. 조각화를 1.0 이상으로 설정하는 것이 좋습니다. 이는 mem_fragmentation_ratio statistic 의 에서 계산됩니다INFO.	숫자
NewConnections	이 기간에 서버에서 허용된 총 연결 수입니다. 이는 total_connections_received 의 통계에서 파생됩니다INFO. **Note** ElastiCache (Redis OSS) 버전 5 이하를 사용하는 경우 이 지표에서 보고하는 연결 중 2~4개가 에서 클러스터를 모니터링하는 ElastiCache 데 사용됩니다. 그러나 ElastiCache (Redis OSS) 버전 6 이상을 사용하는 경우 에서 클러스터를 모니터링하는 ElastiCache 데 사용하는 연결은 이 지표에 포함되지 않습니다.	개수
NumItemsReadFromDisk	분당 디스크에서 검색된 총 항목 수입니다. 의 데이터 계층화 ElastiCache를 사용하는 클러스터에서만 지원됩니다.	개수
NumItemsWrittenToDisk	분당 디스크에 기록된 총 항목 수입니다. 의 데이터 계층화 ElastiCache를 사용하는 클러스터에서만 지원됩니다.	개수

지표	설명	단위
MasterLinkHealthStatus	이 상태에는 0 또는 1의 두 가지 값이 있습니다. 값 0은 기본 노드의 ElastiCache 데이터가 OSS의 Valkey 또는 Redis와 동기화되지 않았음을 나타냅니다EC2. 값 1은 데이터가 동기화되었음을 나타냅니다. 마이그레이션을 완료하려면 CompleteMigration API 작업을 사용합니다.	불
Reclaimed	키 만료 이벤트 총 수입니다. 이는 expired_keys 의 통계에서 파생됩니다INFO.	개수
ReplicationBytes	복제된 구성 노드의 경우 ReplicationBytes 가 기본 노드에서 모든 복제본에 전송하는 바이트 수를 보고합니다. 이 지표는 복제 그룹에 대한 쓰기 부하를 나타냅니다. 이는 master_repl_offset 의 통계에서 파생됩니다INFO.	바이트
ReplicationLag	이 지표는 읽기 복제본으로 실행되는 노드에 한해 적용됩니다. 기본 노드에서 변경 내용을 적용할 때 복제본에서 경과된 시간(초)을 나타냅니다. Valkey 7.2 이상 및 Redis OSS 엔진 버전 5.0.6 이상의 경우 지연을 밀리초 단위로 측정할 수 있습니다.	초
SaveInProgress	이 이진 지표는 백그라운드 저장(forked 또는 forkless)가 진행 중일 때마다 1을, 그렇지 않으면 0을 반환합니다. 백그라운드 저장 프로세스는 일반적으로 스냅샷 작업과 동기화 작업에 사용됩니다. 하지만 이 두 가지 작업은 성능 저하의 원인이 되기도 합니다. 이때는 SaveInProgress 지표를 사용하면 백그라운드 저장 프로세스로 인한 성능 저하 여부를 진단할 수 있습니다. 이는 rdb_bgsave_in_progress 의 통계에서 파생됩니다INFO.	불

지표	설명	단위
TrafficManagementActive	ElastiCache (Redis OSS)가 수신 명령, 모니터링 또는 복제에 할당된 트래픽을 조정하여 트래픽을 적극적으로 관리하고 있는지 여부를 나타냅니다. 트래픽은 Valkey 또는 Redis에서 처리할 수 있는 것보다 많은 명령이 노드로 전송될 때 관리OSS되며 엔진의 안정성과 최적의 작동을 유지하는 데 사용됩니다. 데이터 포인트가 1이면 노드가 제공되는 워크로드에 대해 적게 크기 조정되었음을 나타낼 수 있습니다. **Note** 이 지표가 활성 상태로 유지되면 클러스터를 평가하여 스케일 업 또는 스케일 아웃이 필요한지 결정하세요. 관련 지표에는 NetworkBandwidthOutAllowanceExceeded 및 EngineCPUUtilization 이 포함됩니다.	불

E ngineCPUUtilization 가용성

AWS 다음 나열된 리전은 지원되는 모든 노드 유형에서 사용할 수 있습니다.

리전	리전 이름
us-east-2	미국 동부(오하이오)
us-east-1	미국 동부(버지니아 북부)
us-west-1	미국 서부(캘리포니아 북부)
us-west-2	미국 서부(오레곤)
ap-northeast-1	아시아 태평양(도쿄)

리전	리전 이름
ap-northeast-2	아시아 태평양(서울)
ap-northeast-3	아시아 태평양(오사카)
ap-east-1	아시아 태평양(홍콩)
ap-south-1	아시아 태평양(뭄바이)
ap-southeast-1	아시아 태평양(싱가포르)
ap-southeast-2	아시아 태평양(시드니)
ap-southeast-3	아시아 태평양(자카르타)
ca-central-1	캐나다(중부)
cn-north-1	중국(베이징)
cn-northwest-2	중국(닝샤)
me-south-1	중동(바레인)
eu-central-1	유럽(프랑크푸르트)
eu-west-1	유럽(아일랜드)
eu-west-2	유럽(런던)
eu-west-3	EU(파리)
eu-south-1	유럽(밀라노)
af-south-1	아프리카(케이프타운)
eu-north-1	유럽(스톡홀름)
sa-east-1	남아메리카(상파울루)
us-gov-west-1	AWS GovCloud (미국 서부)

리전	리전 이름
us-gov-east-1	AWS GovCloud (미국 동부)

다음은 info commandstats에서 파생된 몇 가지 유형의 명령 모음입니다. 명령 통계 섹션에서는 호출 수, 이러한 명령에서 소비한 총 CPU 시간, 명령 실행당 CPU 소비한 평균을 포함하여 명령 유형에 따른 통계를 제공합니다. 각 명령 유형에 다음 행이 추가됩니다. `cmdstat_XXX: calls=XXX,usec=XXX,usec_per_call=XXX`

다음 대기 시간 지표는 의 명령 통계 통계를 사용하여 계산됩니다INFO. 이러한 지표는 `delta(usec)/delta(calls)` 방식으로 계산됩니다. `delta`는 1분 이내의 차이로 계산됩니다. 지연 시간은 명령을 처리하는 ElastiCache 데 걸리는 CPU 시간으로 정의됩니다. 데이터 계층화를 사용하는 클러스터의 경우 항목을 가져오는 데 걸리는 시간은 이러한 측정에 포함되지 SSD 않습니다.

사용 가능한 명령의 전체 목록은 Valkey 설명서의 명령을 참조하세요.

지표	설명	단위
`ClusterBasedCmds`	클러스트 기반 명령 총 수입니다. 이는 클러스터(`cluster slot`, `cluster info`등)에 작용하는 모든 명령을 합산하여 `commandstats` 통계에서 파생됩니다.	개수
`ClusterBasedCmdsLatency`	클러스터 기반 명령의 대기 시간입니다.	마이크로초
`EvalBasedCmds`	EVAL 기반 명령의 총 명령 수입니다. 이는 `commandstats` eval, 를 합산하여 통계에서 파생됩니다evalsha.	개수
`EvalBasedCmdsLatency`	eval 기반 명령의 대기 시간입니다.	마이크로초
`GeoSpatialBasedCmds`	geospatial 기반 명령의 총 명령 수입니다. 이는 `commandstats` 통계에서 파생됩니다. 이 수는 모든 geo 유형의 명령(geoadd, geodist, geohash, geopos, georadius 및 georadius bymember)을 합산하여 계산됩니다.	개수

지표	설명	단위
GeoSpatialBasedCmdsLatency	geospatial 기반 명령의 대기 시간입니다.	마이크로초
GetTypeCmds	read-only 유형 명령의 총 건수입니다. 이는 모든 read-only 유형 명령(get, commandstats , hget, scard lrange등)을 합산하여 통계에서 파생됩니다.	개수
GetTypeCmdsLatency	읽기 명령의 지연 시간.	마이크로초
HashBasedCmds	해시 기반 명령의 총 수입니다. 이는 하나 이상의 commandstats 해시(hget, , hkeys, hvals hdel등)에 대해 작동하는 모든 명령을 합산하여 통계에서 파생됩니다.	개수
HashBasedCmdsLatency	해시 기반 명령의 지연 시간.	마이크로초
HyperLogLogBasedCmds	HyperLogLog 기반 명령 총 건수입니다. 이는 모든 pf 유형의 명령(pfadd, commandstats pfcount, pfmerge등)을 합산하여 통계에서 파생됩니다.	개수
HyperLogLogBasedCmdsLatency	HyperLogLog기반 명령의 지연 시간입니다.	마이크로초
JsonBasedCmds	읽기 및 쓰기 JSON 명령을 모두 포함한 총 명령 수입니다. 이는 JSON 키에 따라 작동하는 모든 JSON 명령을 합산하여 commandstats 통계에서 파생됩니다.	개수
JsonBasedCmdsLatency	읽기 및 쓰기 JSON 명령을 포함한 모든 명령의 지연 시간입니다.	마이크로초

지표	설명	단위
JsonBasedGetCmds	JSON 읽기 전용 명령의 총 수입니다. 이는 JSON 키에 작용하는 모든 JSON 읽기 명령을 합산하여 commandstats 통계에서 파생됩니다.	개수
JsonBasedGetCmdsLatency	JSON 읽기 전용 명령의 지연 시간입니다.	마이크로초
JsonBasedSetCmds	총 JSON 쓰기 명령 수입니다. 이는 JSON 키에 작용하는 모든 JSON 쓰기 명령을 합산하여 commandstats 통계에서 파생됩니다.	개수
JsonBasedSetCmdsLatency	JSON 쓰기 명령의 지연 시간입니다.	마이크로초
KeyBasedCmds	키 기반 명령 총 수입니다. 이는 여러 데이터 구조(del, commandstats expirerename, 등)에서 하나 이상의 키에 대해 작동하는 모든 명령을 합산하여 통계에서 파생됩니다.	개수
KeyBasedCmdsLatency	키 기반 명령의 지연 시간.	마이크로초
ListBasedCmds	목록 기반 명령 총 수입니다. 이는 하나 이상의 목록(lindex, commandstats , lrange, lpush ltrim등)에 따라 작동하는 모든 명령을 합산하여 통계에서 파생됩니다.	개수
ListBasedCmdsLatency	목록 기반 명령의 지연 시간.	마이크로초
NonKeyTypeCmds	키 기반이 아닌 명령의 총 수입니다. 이는 commandstats 또는 와 같이 키에 대해 작동하지 않는 모든 명령을 합산하여 통계에서 파생됩니다acldbsizeinfo.	개수
NonKeyTypeCmdsLatency	명령의 지연 시간 non-key-based입니다.	마이크로초

지표	설명	단위
PubSubBasedCmds	pub/sub 기능의 명령 총 수입니다. 이는 psubscribe, , , , , publish, , , pubsub, , 와 같은 pub/sub 기능에 사용되는 모든 명령을 합산하여 commandstats 통계에서 파생됩니다punsubscribessubscribesunsubscribespublishsubscribeunsubscribe.	개수
PubSubBasedCmdsLatency	pub/sub 기반 명령의 대기 시간입니다.	마이크로초
SetBasedCmds	집합 기반 명령 총 수입니다. 이는 하나 이상의 세트(scard, commandstats , sdiff, sadd sunion등)에 대해 작동하는 모든 명령을 합산하여 통계에서 파생됩니다.	개수
SetBasedCmdsLatency	집합 기반 명령의 지연 시간.	마이크로초
SetTypeCmds	write 유형의 총 명령 건수입니다. 이는 데이터에서 작동하는 모든 mutative 유형의 명령(set, commandstats , hset, sadd lpop등)을 합산하여 통계에서 파생됩니다.	개수
SetTypeCmdsLatency	쓰기 명령의 지연 시간.	마이크로초
SortedSetBasedCmds	정렬된 집합 기반 명령 총 수입니다. 이는 하나 이상의 정렬된 세트(zcount, commandstats , zrange, zrank zadd등)에 대해 작동하는 모든 명령을 합산하여 통계에서 파생됩니다.	개수
SortedSetBasedCmdsLatency	정렬 기반 명령의 지연 시간.	마이크로초

지표	설명	단위
StringBasedCmds	문자열 기반 명령 총 수입니다. 이는 하나 이상의 문자열(strlen, commandstats , setex setrange등)에 대해 작동하는 모든 명령을 합산하여 통계에서 파생됩니다.	개수
StringBasedCmdsLatency	문자열 기반 명령 지연 시간	마이크로초
StreamBasedCmds	총 스트림 기반 명령 수입니다. 이는 하나 이상의 스트림 데이터 유형(xrange, commandstats , xlen, xadd xdel등)에 따라 작동하는 모든 명령을 합산하여 통계에서 파생됩니다.	개수
StreamBasedCmdsLatency	스트림 기반 명령의 지연 시간.	마이크로초

Memcached 지표

AWS/ElastiCache 네임스페이스에는 다음 Memcached 지표가 포함되어 있습니다.

AWS/ElastiCache namespace에는 Memcached stats 명령에서 파생된 다음 지표가 포함됩니다. 각 지표는 캐시 노드 수준에서 계산됩니다.

참고 항목

- 호스트 수준 지표

지표	설명	단위
BytesReadIntoMemcached	캐시 노드가 네트워크에서 읽어온 바이트 수	바이트
BytesUsedForCacheItems	캐시 항목을 저장하는 데 사용된 바이트 수	바이트

지표	설명	단위
BytesWrittenOutFromMemcached	캐시 노드가 네트워크로 작성한 바이트 수	바이트
CasBadval	Cas 값이 저장된 Cas 값과 일치하지 않는 캐시가 수신한 요청의 수CAS(확인 및 설정)입니다.	개수
CasHits	요청한 키를 찾았고, CAS 값이 일치할 때 캐시가 수신한 CAS 요청 수	개수
CasMisses	요청한 키를 찾지 못했을 때 캐시가 수신한 CAS 요청 수	개수
CmdFlush	캐시가 수신한 flush 명령 수	개수
CmdGet	캐시가 수신한 get 명령 수	개수
CmdSet	캐시가 수신한 set 명령 수	개수
CurrConnections	동시에 캐시에 연결된 연결 수 집계. ElastiCache 는 2~3개의 연결을 사용하여 클러스터를 모니터링합니다. 위의 내용 외에도 memcached는 노드 유형에 사용되는 스레드의 두 배와 동일한 수의 내부 연결을 만듭니다. 다양한 노드 유형에 대한 스레드 수는 해당하는 파라미터 그룹의 Nodetype Specific Parameters 에서 확인할 수 있습니다. 총 연결 수는 클라이언트 연결, 모니터링을 위한 연결 및 위에 언급된 내부 연결의 합계입니다.	개수
CurrItems	현재 캐시에 저장된 항목 수 집계	개수
DecrHits	요청한 키를 찾았을 때 캐시가 수신한 decrement 요청 수	개수

지표	설명	단위
DecrMisses	요청한 키를 찾지 못했을 때 캐시가 수신한 decrement 요청 수	개수
DeleteHits	요청한 키를 찾았을 때 캐시가 수신한 delete 요청 수	개수
DeleteMisses	요청한 키를 찾지 못했을 때 캐시가 수신한 delete 요청 수	개수
Evictions	새로운 쓰기 공간 확보를 위해 캐시가 제거한 만료 이전 항목 수	개수
GetHits	요청한 키를 찾았을 때 캐시가 수신한 get 요청 수	개수
GetMisses	요청한 키를 찾지 못했을 때 캐시가 수신한 get 요청 수	개수
IncrHits	요청한 키를 찾았을 때 캐시가 수신한 increment 요청 수	개수
IncrMisses	요청한 키를 찾지 못했을 때 캐시가 수신한 increment 요청 수	개수
Reclaimed	새로운 쓰기 공간 확보를 위해 캐시가 제거한 만료 항목 수	개수

Memcached 1.4.14에서는 다음과 같은 지표가 추가 제공됩니다.

지표	설명	단위
BytesUsedForHash	현재 해시 테이블에서 사용 중인 바이트 수	바이트
CmdConfigGet	누적된 config get 요청 수	개수
CmdConfigSet	누적된 config set 요청 수	개수

지표	설명	단위
CmdTouch	누적된 touch 요청 수	개수
CurrConfig	현재 저장된 구성 수	개수
EvictedUnfetched	설정 후 접촉되지 않은 가장 최근에 사용한 캐시(LRU)에서 제거된 유효한 항목의 수입니다.	개수
ExpiredUnfetched	설정 후 접촉하지 LRU 않은 에서 회수된 만료된 항목의 수입니다.	개수
SlabsMoved	이동한 슬래브 페이지 총 수	개수
TouchHits	새로운 만료 시간 지정 이후 사용한 적이 있는 키 수	개수
TouchMisses	사용한 적은 있지만 찾을 수 없는 항목 수	개수

AWS/ElastiCache namespace에는 다음과 같이 계산된 캐시 수준 지표가 포함됩니다.

지표	설명	단위
NewConnections	캐시가 수신한 새로운 연결 수. 이는 일정 기간 동안 total_connections의 변화를 기록하여 memcached total_connections 통계로부터 파생됩니다. 에 대해 예약된 연결로 인해 이 값은 항상 1 이상입니다 ElastiCache.	개수
NewItems	캐시에 저장된 새로운 항목 수. 이는 일정 기간 동안 total_items의 변화를 기록하여 memcached total_items 통계로부터 파생됩니다.	개수
UnusedMemory	데이터에서 사용하지 않는 메모리 크기. 이는 limit_maxbytes에서 바이트를 빼 Memcached 통계 limit_maxbytes 및 바이트로부터 파생됩니다.	바이트

지표	설명	단위
	Memcached 오버헤드는 데이터에서 사용하는 메모리 외에 메모리를 사용하기 때문에 추가 데이터에 사용할 수 있는 메모리 양으로 간주해서는 UnusedMemory 안 됩니다. 아직 미사용 메모리가 있는데도 불구하고 데이터를 제거해야 하는 경우가 발생할 수도 있습니다. 자세한 내용은 Memcached 항목 메모리 사용 섹션을 참조하세요.	

어떤 지표를 모니터링해야 합니까?

다음 CloudWatch 지표는 ElastiCache 성능에 대한 좋은 통찰력을 제공합니다. 대부분의 경우 성능 문제가 발생하기 전에 수정 조치를 취할 수 있도록 이러한 지표에 대한 CloudWatch 경보를 설정하는 것이 좋습니다.

모니터링할 지표

- CPUUtilization
- EngineCPUUtilization
- SwapUsage (발키 및 Redis OSS)
- Evictions
- CurrConnections
- 메모리(Valkey 및 RedisOSS)
- 네트워크
- 지연 시간
- 복제
- 트래픽 관리(Valkey 및 RedisOSS)

CPUUtilization

이는 백분율(%)로 보고된 호스트 수준 지표입니다. 자세한 내용은 호스트 수준 지표 단원을 참조하십시오.

Valkey 및 Redis OSS

2 vCPUs 이하인 작은 노드 유형의 경우 `CPUUtilization` 지표를 사용하여 워크로드를 모니터링합니다.

일반적으로 임계값을 사용 가능한 의 90%로 설정하는 것이 좋습니다CPU. Valkey와 RedisOSS는 모두 단일 스레드이므로 실제 임계값은 노드 총 용량의 일부로 계산해야 합니다. 2개의 코어가 있는 노드 유형을 사용하는 경우를 예로 들어보겠습니다. 이 경우 의 임계값은 90/2 또는 45%CPUUtilization입니다.

사용 중인 캐시 노드에 있는 코어 개수에 따라 임계값을 결정해야 합니다. 이 임계값을 초과하고, 주된 워크로드가 읽기 요청에서 비롯되는 경우에는 읽기 전용 복제본을 추가하여 캐시 클러스터를 스케일 아웃합니다. 주된 워크로드가 쓰기 요청에서 비롯되는 경우에는 클러스터 구성에 따라 다음을 권장합니다.

- Valkey 또는 RedisOSS(클러스터 모드 비활성화됨) 클러스터: 더 큰 캐시 인스턴스 유형을 사용하여 확장합니다.
- Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 클러스터: 샤드를 더 추가하여 쓰기 워크로드를 더 많은 프라이머리 노드에 분산합니다.

> **Tip**
>
> 호스트 수준 지표 를 사용하는 대신 `CPUUtilization`Valkey 및 Redis OSS 사용자는 Valkey 또는 Redis OSS 엔진 코어의 사용량 비율을 `EngineCPUUtilization`보고하는 지표 를 사용할 수 있습니다. 노드에서 이 지표를 사용할 수 있는지 확인하고 자세한 내용은 Valkey 및 Redis 에 대한 지표를 OSS참조하세요.

4vCPUs 개 이상의 더 큰 노드 유형의 경우 Valkey 또는 Redis OSS 엔진 코어의 사용량 비율을 보고하는 `EngineCPUUtilization` 지표를 사용할 수 있습니다. 노드에서 이 지표를 사용할 수 있는지 확인하고 자세한 내용은 Redis 지표를 OSS참조하세요.

Memcached

Memcached는 다중 스레드이므로 이 지표가 90%에 이를 수 있습니다. 이 임계값을 초과하는 경우 더 큰 캐시 노드 유형을 사용하여 캐시 클러스터를 확장하거나 더 많은 캐시 노드를 추가하여 확장합니다.

EngineCPUUtilization

4vCPUs 개 이상의 더 큰 노드 유형의 경우 Redis OSS 엔진 코어의 사용량 비율을 보고하는 `EngineCPUUtilization` 지표를 사용할 수 있습니다. 노드에서 이 지표를 사용할 수 있는지 확인하고 자세한 내용은 Valkey 및 Redis 에 대한 지표를 OSS참조하세요.

자세한 내용은 Amazon 를 사용하여 Amazon(Redis)을 사용한 모범 사례 모니터링 CPUs 섹션을 참조하세요. ElastiCache OSS CloudWatch

SwapUsage (발키 및 Redis OSS)

이는 바이트로 보고된 호스트 수준 지표입니다. 자세한 내용은 호스트 수준 지표 단원을 참조하십시오.

`FreeableMemory` CloudWatch 지표가 0에 가깝거나(즉, 100MB 미만) `SwapUsage` 지표보다 큰 `FreeableMemory` 지표는 노드가 메모리 압력 상태임을 나타냅니다. 이러한 상황이 발생하면 다음 주제를 참조하세요.

- Valkey 또는 Redis OSS 스냅샷을 생성하기에 충분한 메모리 확보
- Valkey 및 Redis용 예약 메모리 관리 OSS

Evictions

이것은 캐시 엔진 지표입니다. 애플리케이션 요구 사항에 따라 이 지표에 대한 경보 임계값을 결정하는 것이 좋습니다.

Memcached를 사용하고 선택한 임계값을 초과하는 경우 더 큰 노드 유형을 사용하여 클러스터를 확장하거나 노드를 더 추가하여 스케일 아웃합니다.

CurrConnections

이것은 캐시 엔진 지표입니다. 애플리케이션 요구 사항에 따라 이 지표에 대한 경보 임계값을 결정하는 것이 좋습니다.

의 수가 증가하면 애플리케이션의 문제를 나타낼 CurrConnections 수 있습니다. 이 문제를 해결하려면 애플리케이션 동작을 조사해야 합니다.

자세한 내용은 Amazon 를 사용하여 Amazon(Redis)을 사용한 모범 사례 모니터링의 연결 섹션을 참조하세요. ElastiCache OSS CloudWatch

메모리(Valkey 및 RedisOSS)

메모리는 Valkey 및 Redis 의 핵심 요소입니다OSS. 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.

자세한 내용은 Amazon 을 사용하여 Amazon(Redis)을 사용한 모범 사례 모니터링의 메모리 섹션을 참조하세요. ElastiCache OSS CloudWatch

네트워크

클러스터의 네트워크 대역폭 용량을 결정하는 요인 중 하나는 선택한 노드 유형입니다. 노드의 네트워크 용량에 대한 자세한 내용은 Amazon ElastiCache 요금 섹션을 참조하세요.

자세한 내용은 Amazon 를 사용하여 Amazon(Redis)을 사용하여 모범 사례 모니터링의 네트워크 섹션을 참조하세요. ElastiCache OSS CloudWatch

지연 시간

데이터 구조당 집계된 지연 시간을 제공하는 CloudWatch 지표 세트를 사용하여 명령의 지연 시간을 측정할 수 있습니다. 이러한 지연 시간 지표는 Valkey INFO 명령의 `commandstats` 통계를 사용하여 계산됩니다.

자세한 내용은 Amazon 를 사용하여 Amazon을 사용한 모범 사례 모니터링의 지연 시간 섹션을 참조하세요. ElastiCache CloudWatch

복제

복제되는 데이터의 볼륨은 `ReplicationBytes` 지표를 통해 확인할 수 있습니다. 이 지표는 복제 그룹에 대한 쓰기 로드를 나타내지만 복제 상태에 대한 자세한 정보는 제공하지 않습니다. 이 목적을 위해 `ReplicationLag` 지표를 사용할 수 있습니다.

자세한 내용은 Amazon 을 사용하여 Amazon(Redis)을 사용한 모범 사례 모니터링의 복제 섹션을 참조하세요. ElastiCache OSS CloudWatch

트래픽 관리(Valkey 및 RedisOSS)

ElastiCache (Redis OSS)는 Valkey 또는 Redis 에서 처리할 수 있는 것보다 더 많은 수신 명령이 노드로 전송될 때 노드에 대한 트래픽을 자동으로 관리합니다OSS. 이는 엔진의 최적 운영 및 안정성을 유지하기 위한 것입니다.

노드에서 트래픽이 활발하게 관리되는 경우 `TrafficManagementActive` 지표는 데이터 포인트 1을 방출합니다. 이는 노드가 제공되는 워크로드에 대해 적게 크기 조정되었음을 나타냅니다. 이 지표가 1인 상태로 오래 유지되면 클러스터를 평가하여 스케일 업 또는 스케일 아웃이 필요한지 결정하세요.

자세한 내용은 지표 페이지의 `TrafficManagementActive` 지표를 참조하세요.

지표 통계 및 기간 선택

CloudWatch 에서 각 지표에 대한 통계 및 기간을 선택할 수 있지만 모든 조합이 유용한 것은 아닙니다. 예를 들어 의 평균, 최소 및 최대 통계CPUUtilization는 유용하지만 합계 통계는 유용하지 않습니다.

모든 ElastiCache 샘플은 각 개별 캐시 노드에 대해 60초 동안 게시됩니다. 60초 기간 동안 캐시 노드 지표는 단일 샘플만 포함할 수 있습니다.

캐시 노드 지표를 검색하는 방법에 대한 자세한 내용은 CloudWatch 클러스터 및 노드 지표 모니터링를 참조하세요.

CloudWatch 클러스터 및 노드 지표 모니터링

ElastiCache 및 CloudWatch 는 통합되므로 다양한 지표를 수집할 수 있습니다. 를 사용하여 이러한 지표를 모니터링할 수 있습니다 CloudWatch.

> **Note**
>
> 다음 예제에서는 CloudWatch 명령줄 도구가 필요합니다. 개발자 도구를 다운로드하는 CloudWatch 및 에 대한 자세한 내용은 CloudWatch 제품 페이지를 참조하세요.

다음 절차에서는 를 CloudWatch 사용하여 지난 한 시간 동안의 캐시 클러스터에 대한 스토리지 공간 통계를 수집하는 방법을 보여줍니다.

> **Note**
>
> 아래 예제에 나온 `StartTime` 및 `EndTime` 값은 설명을 돕기 위해 지정되었습니다. 따라서 캐시 노드의 올바른 시작 및 종료 시간 값으로 대체해야 합니다.

ElastiCache 제한에 대한 자세한 내용은 의 AWS 서비스 제한을 참조하세요 ElastiCache.

CloudWatch 클러스터 및 노드 지표 모니터링(콘솔)

캐시 클러스터에 대한 CPU 사용률 통계를 수집하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.

2. 지표를 확인할 캐시 노드를 선택합니다.

> **Note**
>
> 20개보다 많은 노드를 선택하면 콘솔에 지표가 표시되지 않습니다.

a. AWS 관리 콘솔의 캐시 클러스터 페이지에서 하나 이상의 캐시 클러스터의 이름을 클릭합니다.

   캐시 클러스터의 세부 정보 페이지가 나타납니다.

b. 창 맨 위의 [Nodes] 탭을 클릭합니다.

c. 세부 정보 창의 [Nodes] 탭에서 지표를 확인할 캐시 노드를 선택합니다.

   사용 가능한 CloudWatch 지표 목록이 콘솔 창 하단에 표시됩니다.

d. CPU 사용률 지표를 클릭합니다.

   CloudWatch 콘솔이 열리고 선택한 지표가 표시됩니다. 통계 및 기간 드롭다운 목록 상자와 시간 범위 탭을 사용하여 표시되는 지표를 변경할 수 있습니다.

를 사용하여 CloudWatch 클러스터 및 노드 지표 모니터링 CloudWatch CLI

캐시 클러스터에 대한 CPU 사용률 통계를 수집하려면

- Linux, macOS, Unix의 경우:

```
aws cloudwatch get-metric-statistics \
    --namespace AWS/ElastiCache \
    --metric-name CPUUtilization \
    --dimensions='[{"Name":"CacheClusterId","Value":"test"},
{"Name":"CacheNodeId","Value":"0001"}]' \
    --statistics=Average \
    --start-time 2018-07-05T00:00:00 \
    --end-time 2018-07-06T00:00:00 \
    --period=3600
```

  Windows의 경우:

```
aws cloudwatch get-metric-statistics ^
```

```
    --namespace AWS/ElastiCache ^
    --metric-name CPUUtilization ^
    --dimensions='[{"Name":"CacheClusterId","Value":"test"},
{"Name":"CacheNodeId","Value":"0001"}]' ^
    --statistics=Average ^
    --start-time 2018-07-05T00:00:00 ^
    --end-time 2018-07-06T00:00:00 ^
    --period=3600
```

를 사용하여 CloudWatch 클러스터 및 노드 지표 모니터링 CloudWatch API

캐시 클러스터에 대한 CPU 사용률 통계를 수집하려면

- 다음 파라미터GetMetricStatistics로 를 CloudWatch API 호출합니다(시작 및 종료 시간은 예시로만 표시되므로 적절한 시작 및 종료 시간을 직접 대체해야 합니다).

  - Statistics.member.1=Average
  - Namespace=AWS/ElastiCache
  - StartTime=2013-07-05T00:00:00
  - EndTime=2013-07-06T00:00:00
  - Period=60
  - MeasureName=CPUUtilization
  - Dimensions=CacheClusterId=mycachecluster,CacheNodeId=0002

  Example

```
http://monitoring.amazonaws.com/
    ?Action=GetMetricStatistics
    &SignatureVersion=4
    &Version=2014-12-01
    &StartTime=2018-07-05T00:00:00
    &EndTime=2018-07-06T23:59:00
    &Period=3600
    &Statistics.member.1=Average
    &Dimensions.member.1="CacheClusterId=mycachecluster"
    &Dimensions.member.2="CacheNodeId=0002"
    &Namespace=&AWS;/ElastiCache
    &MeasureName=CPUUtilization
```

```
    &Timestamp=2018-07-07T17%3A48%3A21.746Z
    &AWS;AccessKeyId=<&AWS; Access Key ID>
    &Signature=<Signature>
```

에 대한 할당량 ElastiCache

AWS 계정에는 각 AWS 서비스에 대해 이전에 한도라고 하는 기본 할당량이 있습니다. 다르게 표시되지 않는 한 리전별로 각 할당량이 적용됩니다. 일부 할당량에 대한 증가를 요청할 수 있으며 다른 할당량은 늘릴 수 없습니다.

에 대한 할당량을 보려면 Service Quotas 콘솔을 ElastiCache엽니다. 탐색 창에서 AWS 서비스를 선택하고 를 선택합니다ElastiCache.

할당량 증가를 요청하려면 Service Quotas 사용 설명서의 할당량 증가 요청을 참조하십시오. Service Quotas에서 아직 할당량을 사용할 수 없는 경우 한도 증가 양식을 사용합니다.

AWS 계정에는 와 관련된 다음과 같은 할당량이 있습니다 ElastiCache.

Resource	기본값
리전별 서버리스 캐시	40
캐시당 하루당 서버리스 스냅샷, Redis	24
리전당 노드	300
클러스터당 노드, Memcached	60
인스턴스 유형별 클러스터당 노드, Valkey 또는 RedisOSS(클러스터 모드 활성화됨)	90
샤드, Valkey 또는 Redis당 노드OSS(클러스터 모드 비활성화됨)	6
리전당 파라미터 그룹	300
리전당 보안 그룹	50
리전당 서브넷 그룹	300
서브넷 그룹당 서브넷 수	20
사용자 그룹당 사용자, Redis	100

Resource	기본값
최대 사용자 수, Redis	1000
최대 사용자 그룹 수, Redis	100

레퍼런스

이 섹션의 주제에서는 Amazon ElastiCache API 및 ElastiCache 섹션 작업을 다룹니다 AWS CLI. 또한 여기에는 일반적인 오류 메시지와 서비스 알림에 관한 설명이 포함되어 있습니다.

- 사용 ElastiCache API
- ElastiCache API 참조
- ElastiCache AWS CLI 참조의 섹션
- Amazon ElastiCache 오류 메시지
- 알림

사용 ElastiCache API

이 섹션에서는 ElastiCache 작업을 사용하고 구현하는 방법에 대한 작업 지향적 설명을 제공합니다. 이러한 작업에 대한 전체 설명은 Amazon ElastiCache API 참조 를 참조하세요.

주제

- 쿼리 사용 API
- 사용 가능한 라이브러리
- 애플리케이션 문제 해결

쿼리 사용 API

쿼리 파라미터

HTTP 쿼리 기반 요청은 HTTP 동사 GET 또는 POST 및 라는 쿼리 파라미터를 사용하는 HTTP 요청입니다`Action`.

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. 의 값 *n* 는 1부터 시작하는 정수입니다.

쿼리 요청 인증

쿼리 요청은 로만 보낼 수 HTTPS 있으며 모든 쿼리 요청에 서명을 포함해야 합니다. 이 섹션에서는 서명을 작성하는 방법을 설명합니다. 아래 절차에 설명된 방법은 서명 버전 4라고 합니다.

다음은 AWS에 대한 요청을 인증하는 데 사용되는 기본 단계입니다. 이는 사용자가 에 등록되어 AWS 있고 액세스 키 ID 및 보안 액세스 키가 있다고 가정합니다.

쿼리 인증 절차

1. 발신자는 에 대한 요청을 구성합니다 AWS.
2. 발신자는 이 주제의 다음 섹션에 정의된 대로 요청 서명인 -SHA1 해시 함수를 사용하여 해시 기반 메시지 인증 코드(HMAC)에 대한 키 지정 해싱을 계산합니다.
3. 요청 발신자는 요청 데이터, 서명 및 액세스 키 ID(사용된 보안 액세스 키의 키 식별자)를 로 전송합니다 AWS.
4. AWS 는 액세스 키 ID를 사용하여 보안 액세스 키를 조회합니다.
5. AWS 는 요청의 서명을 계산하는 데 사용된 것과 동일한 알고리즘을 사용하여 요청 데이터와 보안 액세스 키에서 서명을 생성합니다.
6. 서명이 일치하는 경우, 요청이 인증되는 것으로 간주됩니다. 서명이 일치하지 않을 경우, 요청이 삭제되고 AWS 에서 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. 파라미터 이름별로 UTF-8 쿼리 문자열 구성 요소를 자연 바이트 정렬로 정렬합니다. 파라미터는 POST 본문에서 GET URI 또는 본문에서 가져올 수 있습니다(콘텐츠 유형이 애플리케이션인 경우/x-www-form-urlencoded).

   b. URL 다음 규칙에 따라 파라미터 이름과 값을 인코딩합니다.

i. RFC 3986에서 정의한 예약되지 않은 문자는 URL 인코딩하지 마세요. 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. 백분율 인코딩 확장 UTF-8자, %XY%ZA... 형식.

iv. 공백 문자는 %20(일반 인코딩 구조인 +가 아님)으로 퍼센트 인코딩합니다.

c. 파라미터 값이 비어 있더라도 인코딩된 파라미터 이름과 인코딩된 값을 등호(=)(ASCII 문자 61)로 구분합니다.

d. 이름-값 페어를 앰퍼샌드(&)(ASCII 코드 38)로 구분합니다.

2. 다음 의사 문법에 따라 서명할 문자열을 생성합니다("\n"은 ASCII 새 줄을 나타냄).

```
StringToSign = HTTPVerb + "\n" +
ValueOfHostHeaderInLowercase + "\n" +
HTTPRequestURI + "\n" +
CanonicalizedQueryString <from the preceding step>
```

구성 HTTPRequestURI 요소는 쿼리 문자열을 포함하지 않는 URI 최대 의 HTTP 절대 경로 구성 요소입니다. HTTPRequestURI 가 비어 있는 경우 슬래시(/)를 사용합니다.

3. 방금 생성한 문자열, 보안 액세스 키를 키로, SHA256 또는 RFC 해시 알고리즘SHA1으로 HMAC 2104를 준수합니다.

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://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeCacheClusters
    &CacheClusterIdentifier=myCacheCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-12-01
```

이전 쿼리 문자열의 경우 다음 문자열을 통해 HMAC 서명을 계산합니다.

```
GET\n
    elasticache.amazonaws.com\n
    Action=DescribeCacheClusters
    &CacheClusterIdentifier=myCacheCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-12-01
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE%2F20140523%2Fus-west-2%2Felasticache
%2Faws4_request
    &X-Amz-Date=20141201T223649Z
    &X-Amz-SignedHeaders=content-type%3Bhost%3Buser-agent%3Bx-amz-content-sha256%3Bx-
amz-date
        content-type:
        host:elasticache.us-west-2.amazonaws.com
        user-agent:CacheServicesAPICommand_Client
    x-amz-content-sha256:
    x-amz-date:
```

이 결과는 다음의 서명된 요청입니다.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeCacheClusters
    &CacheClusterIdentifier=myCacheCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-12-01
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20141201/us-west-2/elasticache/aws4_request
    &X-Amz-Date=20141201T223649Z
    &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

AWS 는 쿼리 APIs 대신 언어별 를 사용하여 애플리케이션을 빌드하려는 소프트웨어 개발자를 위한 소프트웨어 개발 키트(SDKs)를 제공합니다API. 이러한 함수SDKs는 인증 요청, 재시도 요청 및 오류 처리와 같은 기본 함수(에 포함되지 않음APIs)를 제공하므로 더 쉽게 시작할 수 있습니다. SDKs 및 추가 리소스는 다음 프로그래밍 언어에 사용할 수 있습니다.

- Java
- Windows 및 .NET
- PHP
- Python
- Ruby

다른 언어에 대한 자세한 내용은 샘플 코드 및 라이브러리를 참조하세요.

애플리케이션 문제 해결

ElastiCache 는 와 상호 작용하는 동안 문제를 해결하는 데 도움이 되는 구체적이고 설명적인 오류를 제공합니다 ElastiCache API.

오류 검색

일반적으로 사용자는 시간을 소비하여 결과를 처리하기 전에 애플리케이션이 먼저 해당 요청으로 오류가 발생되는지 여부를 확인하려고 합니다. 오류가 발생했는지 확인하는 가장 쉬운 방법은 의 응답에서 `Error` 노드를 찾는 것입니다 ElastiCache API.

XPath 구문은 `Error` 노드의 존재를 검색하는 간단한 방법과 오류 코드 및 메시지를 검색하는 쉬운 방법을 제공합니다. 다음 코드 조각은 Perl 및 XML::XPath 모듈을 사용하여 요청 중에 오류가 발생했는지 확인합니다. 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

의 문제를 진단하고 해결하려면 다음 프로세스를 사용하는 것이 좋습니다 ElastiCache API.

- ElastiCache 가 올바르게 실행 중인지 확인합니다.

  이렇게 하려면 브라우저 창을 열고 ElastiCache 서비스(예:)에 쿼리 요청을 제출하면 됩니다 https://elasticache.amazonaws.com. MissingAuthenticationTokenException 또는 500 내부 서버 오류는 서비스를 사용할 수 있고 요청에 응답하는지 확인합니다.
- 요청 구조 확인.

  각 ElastiCache 작업에는 참조 에 ElastiCache API 참조 페이지가 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.

  ElastiCache 에는 다른 사람들이 그 과정에서 경험한 문제에 대한 솔루션을 검색할 수 있는 토론 포럼이 있습니다. 포럼을 보려면 다음 사이트를 참조하세요.

  https://forums.aws.amazon.com/ .

ElastiCache 명령줄 인터페이스 설정

이 섹션은 명령줄 도구 실행을 위한 필수 조건, 명령줄 도구를 구할 수 있는 위치, 도구 및 환경 설정 방법을 설명하고 도구 사용의 몇몇 일반적인 예를 포함하고 있습니다.

용 AWS CLI 로 이동하는 경우에만 이 주제의 지침을 따르세요 ElastiCache.

> **Important**
>
> Amazon ElastiCache Command Line Interface(CLI)는 API 버전 2014-09-30 이후의 ElastiCache 개선 사항을 지원하지 않습니다. 명령줄에서 최신 ElastiCache 기능을 사용하려면 AWS 명령줄 인터페이스 를 사용합니다.

주제

- 사전 조건
- 명령줄 도구 얻기

- 도구 설정
- 도구에 대한 자격 증명 제공
- 환경 변수

사전 조건

이 문서에서는 Linux/UNIX 또는 Windows 환경에서 작업할 수 있다고 가정합니다. Amazon ElastiCache 명령줄 도구는 UNIX기반 환경인 Mac OS X에서도 작동하지만 이 안내서에는 특정 Mac OS X 지침이 포함되어 있지 않습니다.

하나의 규칙으로서 모든 명령줄 텍스트 앞에 일반적인 **PROMPT>** 명령줄 프롬프트가 나옵니다. 머신의 실제 명령줄 프롬프트는 다를 수 있습니다. 또한 를 사용하여 Linux/UNIX특정 명령을 **$** 표시하고 Windows별 명령을 **C:\>** 표시합니다. 명령의 결과인 출력 예는 접두사 없이 그 후에 즉시 표시됩니다.

Java 런타임 환경

이 설명서에 사용된 명령줄 도구를 실행하려면 Java 버전 5 이상이 있어야 합니다. JRE 또는 JDK 설치가 허용됩니다. Linux/UNIX 및 Windows를 포함한 다양한 플랫폼에 JREs 대한 보기 및 다운로드는 Java SE 다운로드를 참조하세요.

Java Home 변수 설정

명령줄 도구는 Java 런타임을 찾기 위해 환경 변수(JAVA_HOME)를 사용합니다. 이 환경 변수는 라는 하위 디렉터리가 포함된 디렉터리의 전체 경로로 설정해야 합니다. bin 하위 디렉터리에는 차례로 실행 파일java(Linux 및 UNIX) 또는 java.exe (Windows) 실행 파일이 포함됩니다.

Java Home 변수를 설정하려면

1. Java Home 변수를 설정합니다.
   - Linux 및 에서 다음 명령을 UNIX입력합니다.

     ```
     $ export JAVA_HOME=<PATH>
     ```

   - Windows에서 다음 명령을 입력합니다.

     ```
     C:\> set JAVA_HOME=<PATH>
     ```

2. **$JAVA_HOME/bin/java -version**을 실행하고 출력을 확인하여 경로 설정을 확인합니다.

- Linux/에서는 다음과 유사한 출력이 UNIX표시됩니다.

```
$ $JAVA_HOME/bin/java -version
java version "1.6.0_23"
Java(TM) SE Runtime Environment (build 1.6.0_23-b05)
Java HotSpot(TM) Client VM (build 19.0-b09, mixed mode, sharing)
```

- Windows에서 다음과 유사한 출력을 확인할 수 있습니다.

```
C:\> %JAVA_HOME%\bin\java -version
java version "1.6.0_23"
Java(TM) SE Runtime Environment (build 1.6.0_23-b05)
Java HotSpot(TM) Client VM (build 19.0-b09, mixed mode, sharing)
```

명령줄 도구 얻기

명령줄 도구는 ElastiCache 개발자 도구 웹 사이트 에서 ZIP 파일로 사용할 수 있습니다. 이러한 도구는 Java로 작성되며 Windows 2000/XP/Vista/Windows 7, Linux/ UNIX및 Mac 용 쉘 스크립트를 포함합니다OSX. ZIP 파일은 자체 포함되어 있으므로 설치할 필요가 없습니다. zip 파일을 다운로드하고 로컬 시스템의 디렉터리에 압축을 풀기만 하면 됩니다.

도구 설정

명령줄 도구는 환경 변수(AWS_ELASTICACHE_HOME)에 따라 지원 라이브러리를 찾습니다. 이 환경 변수를 먼저 설정해야 도구를 사용할 수 있습니다. 환경 변수를 명령줄 도구의 압축을 푼 디렉토리 경로로 설정합니다. 이 디렉터리의 이름은 ElastiCacheCli-A.B.nnnn(A, B 및 n은 버전/릴리스 번호)이며 bin 및 lib라는 하위 디렉터리가 포함되어 있습니다.

AWS_ELASTICACHE_HOME 환경 변수를 설정하려면

- 명령줄 창을 열고 다음 명령 중 하나를 입력하여 AWS_ELASTICACHE_HOME 환경 변수를 설정합니다.

  - Linux 및 에서 다음 명령을 UNIX입력합니다.

```
$ export &AWS;_ELASTICACHE_HOME=<path-to-tools>
```

- Windows에서 다음 명령을 입력합니다.

```
C:\> set &AWS;_ELASTICACHE_HOME=<path-to-tools>
```

도구를 더 쉽게 사용하려면 도구의 BIN 디렉터리를 시스템에 추가하는 것이 좋습니다PATH. 이 가이드의 나머지 부분에서는 BIN 디렉터리가 시스템 경로에 있다고 가정합니다.

도구의 BIN 디렉터리를 시스템 경로에 추가하려면

- 다음 명령을 입력하여 도구의 BIN 디렉터리를 시스템에 추가합니다PATH.
  - Linux 및 에서 다음 명령을 UNIX입력합니다.

```
$ export PATH=$PATH:$&AWS;_ELASTICACHE_HOME/bin
```

  - Windows에서 다음 명령을 입력합니다.

```
C:\> set PATH=%PATH%;%&AWS;_ELASTICACHE_HOME%\bin
```

> **Note**
>
> Windows 환경 변수는 명령 창을 닫으면 재설정됩니다. 이러한 변수를 영구적으로 설정할 수 있습니다. 자세한 내용은 현재 Windows 버전의 설명서를 참조하세요.

> **Note**
>
> 경로에 공백이 포함된 경우 다음 예에서처럼 큰따옴표로 묶어야 합니다.
> "C:\Program Files\Java"

도구에 대한 자격 증명 제공

명령줄 도구에는 AWS 계정과 함께 제공된 AWS 액세스 키 및 보안 액세스 키가 필요합니다. 명령줄을 사용하거나 로컬 시스템에 위치하는 자격 증명 파일에서 이러한 키를 얻을 수 있습니다.

배포에는 정보로 편집해야 하는 템플릿 파일 ${AWS_ELASTICACHE_HOME}/credential-file-path.template가 포함됩니다. 다음은 템플릿 파일의 콘텐츠입니다.

```
AWS AccessKeyId=<Write your AWS access ID>
AWS SecretKey=<Write your AWS secret key>
```

Important

에서 자격 증명 파일의 소유자로 권한을 UNIX제한합니다.

```
$ chmod 600 <the file created above>
```

보안 인증 파일 설정을 사용하면 ElastiCache 도구가 정보를 찾을 수 있도록 AWS_CREDENTIAL_FILE 환경 변수를 설정해야 합니다.

AWS_CREDENTIAL_FILE 환경 변수를 설정하려면

1. 환경 변수를 설정합니다.
   - Linux 및 에서 다음 명령을 사용하여 변수를 UNIX업데이트합니다.

     ```
     $ export &AWS;_CREDENTIAL_FILE=<the file created above>
     ```
   - Windows에서는 다음 명령을 사용하여 변수를 설정합니다.

     ```
     C:\> set &AWS;_CREDENTIAL_FILE=<the file created above>
     ```
2. 설정한 것이 제대로 작동하는지 확인하려면 다음 명령을 실행합니다.

   ```
   elasticache --help
   ```

   모든 ElastiCache 명령에 대한 사용 페이지가 표시됩니다.

환경 변수

환경 변수는 스크립팅, 기본값 구성 또는 기본값 임시 재정의에 유용할 수 있습니다.

AWS_CREDENTIAL_FILE 환경 변수 외에도 ElastiCache 명령줄 인터페이스에 포함된 대부분의 API 도구는 다음 변수를 지원합니다.

- EC2_REGION - 사용할 AWS 리전입니다.
- AWS_ELASTICACHE_URL - 서비스 호출에 URL 사용할 입니다. EC2_REGION가 지정되거나 -- 리전 파라미터가 전달되는 경우 다른 리전 엔드포인트를 지정할 필요가 없습니다.

다음 예제에서는 환경 변수 EC2_REGION를 설정하여 API 도구에서 사용하는 리전을 구성하는 방법을 보여줍니다.

Linux, OS X 또는 Unix

```
$ export EC2_REGION=us-west-1
```

Windows

```
$ set EC2_REGION=us-west-1
```

Amazon ElastiCache 오류 메시지

Amazon 에서 다음 오류 메시지를 반환합니다 ElastiCache. ElastiCache, 기타 AWS 서비스 또는 Valkey, Redis OSS또는 Memcached에서 반환하는 다른 오류 메시지가 표시될 수 있습니다. 이외의 소스의 오류 메시지에 대한 설명은 오류 메시지를 생성하는 소스의 설명서를 ElastiCache참조하세요.

- Cluster node quota exceeded
- Customer's node quota exceeded
- Manual snapshot quota exceeded
- Insufficient cache cluster capacity

오류 메시지: 클러스터 노드 할당량이 초과되었습니다. 각 클러스터는 이 리전에서 최대 %n개의 노드를 가질 수 있습니다.

원인: 클러스터를 생성 또는 수정하려고 시도하여 클러스터에 %n개가 넘는 노드가 있습니다.

솔루션: 클러스터에 %n개가 넘는 노드가 있지 않도록 요청을 변경하세요. 또는 %n개 이상의 노드가 필요한 경우 Amazon ElastiCache Node 요청 양식 을 사용하여 요청합니다.

자세한 내용은 의 Amazon ElastiCache 제한을 참조하세요Amazon Web Services 일반 참조.

오류 메시지: 고객 노드 할당량이 초과되었습니다. 이 리전에서 최대 %n개의 노드가 있을 수 있습니다. 또는 이 리전에서 이미 %s개의 노드 할당량에 도달했습니다.

원인: 클러스터를 생성 또는 수정하려고 시도하여 계정에 이 리전의 모든 클러스터에 대해 %n개가 넘는 노드가 있습니다.

솔루션: 이 계정에 대한 모든 클러스터의 리전에 있는 총 노드가 %n개를 초과하지 않도록 요청을 변경하세요. 또는 %n개 이상의 노드가 필요한 경우 Amazon ElastiCache Node 요청 양식 을 사용하여 요청합니다.

자세한 내용은 의 Amazon ElastiCache 제한을 참조하세요Amazon Web Services 일반 참조.

오류 메시지: 24시간 내에 수행된 이 클러스터에 대한 수동 스냅샷의 최대 수가 도달했습니다. 개의 할당량에 도달했거나24시간 내에 수행된 이 노드에 대한 수동 스냅샷의 최대 수가 %n개의 할당량에 도달했습니다.

원인: 24시간의 기간 내에 허용된 수동 스냅샷의 최대 수에 이미 도달했을 때 클러스터의 수동 스냅샷을 수행하려고 시도했습니다.

솔루션: 24시간을 기다린 후 클러스터의 다른 수동 스냅샷을 시도하세요. 또는 지금 수동 스냅샷을 수행해야 하는 경우 클러스터의 다른 노드와 같이 동일한 데이터가 있는 다른 노드의 스냅샷을 수행하세요.

오류 메시지: InsufficientCacheClusterCapacity

원인: 현재 AWS 에 요청에 대한 서비스를 제공할 수 있을 만큼 온디맨드 용량이 충분하지 않습니다.

해결 방법:

- 몇 분 정도 기다린 후 다시 요청을 제출합니다. 용량은 자주 변할 수 있습니다.
- 노드 또는 샤드(노드 그룹) 수가 줄어든 새 요청을 제출하세요. 예를 들어 단일 요청을 통해 노드 15개를 시작하는 경우 노드 5개의 요청 3개 또는 노드 1개 대신 요청 15개를 시도합니다.
- 클러스터를 시작하고 있는 경우 가용 영역을 지정하지 않고 새 요청을 제출하세요.

- 클러스터를 시작하고 있는 경우 이후의 단계에서 확장할 수 있는 다른 노드 유형을 사용하여 새 요청을 제출하세요. 자세한 내용은 크기 조정 ElastiCache 단원을 참조하십시오.

알림

이 주제에서는 관심 있을 수 있는 ElastiCache 알림을 다룹니다. 알림은 대부분 일시적이며, 솔루션을 찾아 구현할 때까지만 지속되는 상황이나 이벤트입니다. 알림에는 일반적으로 시작 날짜와 해결 날짜가 있으며, 그 이후에는 알림이 더 이상 관련되지 않습니다. 알림은 사용자과 관련이 있거나 관련이 없을 수 있습니다. 클러스터의 성능을 향상하는 구현 지침을 따르는 것이 좋습니다.

알림은 새롭거나 향상된 ElastiCache 특성 또는 기능을 알리지 않습니다.

일반 ElastiCache 알림

현재 엔진별이 아닌 미해결 ElastiCache 알림은 없습니다.

ElastiCache (Memcached) 알림

다음 ElastiCache 알림은 Memcached 엔진에만 해당됩니다.

ElastiCache (Memcached)별 알림

- 알림: 세분화 결함을 유발하는 Memcached LRU 크롤러

알림: 세분화 결함을 유발하는 Memcached LRU 크롤러

⚠ 알림 날짜: 2017년 2월 28일
경우에 따라 클러스터가 Memcached LRU Crawler의 분할 오류와 함께 불안정성을 표시할 수 있습니다. 이는 일정 기간 동안 존재하는 Memcached 엔진 내부의 문제입니다. 이 문제는 기본적으로 LRU 크롤러가 활성화되었을 때 Memcached 1.4.33에서 명백해졌습니다.
이 문제가 발생하는 경우 수정이 있을 때까지 LRU크롤러를 비활성화하는 것이 좋습니다. 이렇게 하려면 명령줄에서 `lru_crawler disable`을 사용하거나 `lru_crawler` 파라미터 값을 수정(선호됨)하세요.
해결된 날짜:
해결 방법:

ElastiCache (Redis OSS) 특정 알림

현재 미해결 ElastiCache (Redis OSS) 알림이 없습니다.

ElastiCache 문서 기록

- API 버전: 2015-02-02
- 최신 설명서 업데이트: 2023년 11월 27일

다음 표에서는 2018년 3월 이후 ElastiCache 사용 설명서의 각 릴리스에 대한 중요한 변경 사항을 설명합니다. 이 설명서의 업데이트에 대한 알림을 받으려면 RSS 피드를 구독하세요.

최근 ElastiCache 업데이트

변경 사항	설명	날짜
Valkey ElastiCache 에서 에 대한 지원	ElastiCache 는 이제 Valkey 를 지원합니다. Valkey 7.2.6은 Redis OSS 7.2와 호환됩니다. 자세한 내용은 Valkey 단원을 참조하세요.	2024년 10월 8일
유연한 예약 노드 크기 조정	ElastiCache 는 이제 유연한 예약 노드 크기 를 지원합니다. 자세한 내용은 Amazon ElastiCache 요금 섹션을 참조하세요.	2024년 10월 1일
ElastiCache (Redis OSS) 추가 C7gn 노드 크기에 대한 지원이 추가되었습니다.	ElastiCache (Redis OSS)에 추가 C7gn 노드 크기에 대한 지원이 추가되었습니다.	2024년 1월 10일
ElastiCache (Redis OSS)는 이제 서버리스 캐시 생성을 지원합니다.	이제 서버리스 캐시를 생성하여 캐시 관리를 간소화하고 가장 까다로운 애플리케이션을 지원하도록 즉시 규모 조정할 수 있습니다. 자세한 내용은 배포 옵션 간 선택을 참조하세요. 이 기능의 일부로 서버리스 캐시를 관리형 VPC 엔드포인트와 연결할 수 AmazonElastiCacheF	2023년 11월 27일

	ullAccess 있도록 ElastiCacheService RolePolicy 및 에 새 권한이 추가되었습니다. 또한 AmazonElastiCacheF ullAccess 정책을 사용하여 수정된 콘솔 환경을 지원하는 권한이 추가되었습니다.	
ElastiCache 이제 (Memcached)에서 서버리스 캐시 생성을 지원합니다.	이제 서버리스 캐시를 생성하여 캐시 관리를 간소화하고 가장 까다로운 애플리케이션을 지원하도록 즉시 규모 조정할 수 있습니다. 자세한 내용은 배포 옵션 간 선택을 참조하세요. 이 기능의 일부로 서버리스 캐시를 관리형 VPC 엔드포인트와 연결할 수 AmazonElastiCacheF ullAccess 있도록 ElastiCacheService RolePolicy 및 에 새 권한이 추가되었습니다. 또한 AmazonElastiCacheF ullAccess 정책을 사용하여 수정된 콘솔 환경을 지원하는 권한이 추가되었습니다.	2023년 11월 27일
ElastiCache 이제 (Redis OSS)에서 클러스터 모드 수정 지원	이제 클러스터 모드 비활성화(CMD)에서 클러스터 모드 활성화()로 클러스터를 마이그레이션할 수 있습니다CME. 자세한 정보는 클러스터 모드 수정을 참조하세요.	2023년 5월 11일

ElastiCache (Redis OSS)에서 이제 전송 중 암호화 설정 수정 지원	이제 OSS 클러스터를 재구축하거나 재프로비저닝하거나 애플리케이션 가용성에 영향을 주지 않고 Redis 클러스터의 TLS 구성을 변경할 수 있습니다. 자세한 내용은 기존 클러스터에 전송 중 데이터 암호화 활성화를 참조하세요.	2022년 12월 28일
ElastiCache (Redis OSS)는 이제 를 사용하여 사용자 인증 지원 IAM	IAM 인증을 사용하면 자격 증명을 사용하여 ElastiCache AWS IAM (Redis OSS)에 대한 연결을 인증할 수 있습니다. 그러면 보안 모델이 강화되고 여러 보안 관리 작업이 단순화됩니다. 자세한 내용은 를 사용한 인증을 참조하세요IAM.	2022년 11월 16일
ElastiCache (Redis OSS)에서 이제 Redis OSS 7 지원	이 릴리스는 Amazon ElastiCache (Redis OSS)에 Redis OSS 함수, ACL 개선 사항 및 Sharded Pub/Sub와 같은 몇 가지 새로운 기능을 제공합니다. 자세한 내용은 ElastiCache (Redis OSS) 버전 7.0을 참조하세요.	2022년 11월 8일

ElastiCache 이제 (Redis OSS)에서 지원 IPV6	ElastiCache 는 Internet Protocol 버전 4 및 6(IPv4 및 IPv6)을 지원하므로 IPv4 연결만 허용하거나 연결만 허용하거나 IPv4 IPv6 및 IPv6 연결(이중 스택)을 모두 허용하도록 클러스터를 구성할 수 있습니다. IPv6 는 Nitro 시스템 에 구축된 모든 인스턴스에서 Redis OSS 엔진 버전 6.2 이상을 사용하는 워크로드에 대해 지원됩니다. 를 ElastiCache 통한 액세스에는 추가 요금이 부과되지 않습니다IPv6. 자세한 내용은 네트워크 유형 선택을 참조하세요.	2022년 11월 7일
ElastiCache 이제 (Memcached)에서 를 지원합니다. IPV6	ElastiCache 는 Internet Protocol 버전 4 및 6(IPv4 및 IPv6)을 지원하므로 IPv4 연결만 허용하거나 연결만 허용하거나 IPv4 IPv6 및 IPv6 연결(이중 스택)을 모두 허용하도록 클러스터를 구성할 수 있습니다. IPv6 는 Nitro 시스템 에 구축된 모든 인스턴스에서 Memcached 엔진 버전 1.6.6 이상을 사용하는 워크로드에 대해 지원됩니다. 를 ElastiCache 통한 액세스에는 추가 요금이 부과되지 않습니다IPv6. 자세한 내용은 네트워크 유형 선택을 참조하세요.	2022년 11월 7일

ElastiCache (Memcached)가 이제 전송 중 암호화를 지원합니다.	전송 중 데이터 암호화는 선택적 기능으로, 가장 취약한 지점 즉, 한 위치에서 다른 위치로 데이터를 전송할 때 데이터의 보안을 강화할 수 있습니다. Memcached 버전 1.6.12 이상에서 지원됩니다. 자세한 내용은 ElastiCache 전송 중 암호화(TLS)를 참조하세요.	2022년 5월 26일
ElastiCache (Redis OSS)는 이제 네이티브 JavaScript 객체 표기법(JSON) 형식을 지원합니다.	네이티브 JavaScript 객체 표기법(JSON) 형식은 Redis OSS 클러스터 내에서 복잡한 데이터 세트를 인코딩하는 간단하고 스키마 없는 방법입니다. Redis OSS 클러스터 내에 JavaScript 객체 표기법(JSON) 형식을 사용하여 데이터를 기본적으로 저장하고 액세스할 수 있으며, 사용자 지정 코드를 관리하여 직렬화 및 역직렬화할 필요 없이 해당 클러스터에 저장된 JSON 데이터를 업데이트할 수 있습니다. 자세한 내용은 시작하기를 참조하세요 JSON.	2022년 5월 25일

ElastiCache 이제 에서 지원 PrivateLink	AWS PrivateLink 를 사용하면 인터넷 게이트웨이, NAT 디바이스, VPN 연결 또는 AWS Direct Connect 연결 없이 비공개로 작업에 액세스할 ElastiCache API 수 있습니다. 자세한 내용은 Redis용 Amazon ElastiCache API 및 인터페이스 VPC 엔드포인트(AWS PrivateLink) OSS 또는 Memcached용 Amazon ElastiCache API 및 인터페이스 VPC 엔드포인트(AWS PrivateLink)를 참조하세요.	2022년 1월 24일
ElastiCache (Redis OSS)에서 이제 Redis OSS 6.2 및 데이터 계층화 지원	Amazon ElastiCache (Redis OSS)은 Amazon 에서 지원하는 다음 버전의 Redis OSS 엔진을 도입합니다 ElastiCache. ElastiCache (Redis OSS) 6.2에는 8 vCPUs 개 이상의 x86 노드 유형 또는 4개 vCPUs 이상의 Graviton2 노드 유형을 사용하는 TLS지원 클러스터에 대한 성능 개선이 포함되어 있습니다. ElastiCache (Redis OSS)도 데이터 계층화를 도입합니다. 저렴한 방법으로 데이터 계층화를 사용하여 클러스터를 최대 수백 테라바이트 용량으로 확장할 수 있습니다. 자세한 내용은 ElastiCache (Redis OSS) 버전 6.2(향상) 및 데이터 계층화를 참조하세요.	2021년 11월 23일

Auto Scaling 지원	ElastiCache (Redis OSS)는 이제 Auto Scaling 을 지원합니다. ElastiCache (Redis OSS) Auto Scaling은 ElastiCache (Redis OSS) 서비스에서 원하는 샤드 또는 복제본을 자동으로 늘리거나 줄이는 기능입니다. ElastiCache 는 Application Auto Scaling 서비스를 활용하여 이 기능을 제공합니다. 자세한 내용은 Auto Scaling ElastiCache (RedisOSS) 클러스터 를 참조하세요.	2021년 8월 19일
Redis OSS Slow 로그 전송 지원	ElastiCache 이제 에서 Amazon Data Firehose 또는 Amazon CloudWatch Logs의 두 대상 중 하나로 Redis를 스트리밍OSSSLOWLOG할 수 있습니다. 자세한 내용은 로그 전달을 참조하세요.	2021년 4월 22일
리소스 태그 지정 및 조건 키 지원	ElastiCache 는 이제 클러스터 및 기타 ElastiCache 리소스를 관리하는 데 도움이 되는 태그 지정을 지원합니다. 자세한 내용은 ElastiCache 리소스 태그 지정을 참조하세요. 에서는 조건 키에 대한 지원 ElastiCache 도 소개합니다. IAM 정책이 적용되는 방식을 결정하는 조건을 지정할 수 있습니다. 자세한 내용은 조건 키 사용을 참조하세요.	2021년 4월 7일

리소스 태그 지정 및 조건 키 지원	ElastiCache 는 이제 클러스터 및 기타 ElastiCache 리소스를 관리하는 데 도움이 되는 태그 지정을 지원합니다. 자세한 내용은 ElastiCache 리소스 태그 지정을 참조하세요. 에서는 조건 키에 대한 지원 ElastiCache 도 제공합니다. IAM 정책이 적용되는 방식을 결정하는 조건을 지정할 수 있습니다. 자세한 내용은 조건 키 사용을 참조하세요.	2021년 4월 7일
ElastiCache 이제 AWS Outposts에서 를 사용할 수 있습니다.	AWS Outpost는 네이티브 AWS 서비스, 인프라 및 운영 모델을 거의 모든 데이터 센터, 코로케이션 공간 또는 온프레미스 시설에 제공합니다. Outposts ElastiCache 에 를 배포하여 클라우드와 마찬가지로 온프레미스에서 캐시를 설정, 운영 및 사용할 수 있습니다. 자세한 내용은 Redis용 Outposts 사용 OSS 또는 Memcached용 Outposts 사용을 참조하세요.	2020년 10월 8일

ElastiCache 이제 Redis OSS 6 지원	Amazon ElastiCache (Redis OSS)은 Amazon 에서 지원하는 다음 버전의 Redis OSS 엔진을 도입합니다 ElastiCache. 이 버전에는 역할 기반 액세스 제어를 사용한 사용자 인증, 버전 관계없는 지원, 클라이언트 측 캐싱과 상당한 작동 성능 향상이 포함되어 있습니다. 자세한 내용은 ElastiCache (Redis OSS) 버전 6.0(고급)을 참조하세요.	2020년 10월 7일
ElastiCache 이제 에서 로컬 영역 지원	로컬 영역은 사용자와 지리적으로 가까운 AWS 리전의 확장입니다. 새 서브넷을 생성하고 로컬 영역에 할당하여 상위 AWS 리전에서 로컬 영역으로 가상 프라이빗 클라우드(VPC)를 확장할 수 있습니다. 자세한 내용은 Local Zones 사용을 참조하세요.	2020년 9월 25일
ElastiCache (Redis OSS)는 이제 최대 500개의 노드 또는 500개의 샤드까지 Redis OSS 클러스터 환경 확장을 지원합니다.	Redis OSS 클러스터 모드를 사용하면 여러 샤드에서 데이터를 분할하는 데 사용할 수 있는 구성을 사용할 수 있으며 확장성, 성능 및 가용성이 향상됩니다. 이 기능은 모든 AWS 리전의 Amazon ElastiCache (Redis OSS) 버전 5.0.6 이상과 모든 기존 및 신규 ElastiCache (Redis OSS) 클러스터 환경에서 사용할 수 있습니다. 자세한 내용은 Redis OSS 노드 및 샤드를 참조하세요.	2020년 8월 13일

ElastiCache 는 이제 리소스 수준 권한을 지원합니다.	이제 AWS Identity and Access Management (IAM) 정책에 ElastiCache 리소스를 지정하여 사용자 권한 범위를 제한할 수 있습니다. 자세한 내용은 리소스 수준 권한 섹션을 참조하세요.	2020년 8월 12일
ElastiCache (Redis OSS)에 Amazon CloudWatch 지표 추가	ElastiCache (Redis OSS)는 이제 PubSubCmds 및 를 포함한 새 CloudWatch 지표를 지원합니다HyperLogLogBasedCmds . 전체 목록은 Redis 지표를 참조하세요OSS.	2020년 6월 10일
ElastiCache 이제 에서 ElastiCache 클러스터의 자동 업데이트를 지원합니다.	Amazon은 ElastiCache 이제 서비스 업데이트의 '추천 적용 기한'이 지난 후 ElastiCache 클러스터의 자동 업데이트를 지원합니다. ElastiCache 는 유지 관리 기간을 사용하여 해당 클러스터의 자동 업데이트를 예약합니다. 자세한 내용은 셀프 서비스 업데이트를 참조하세요.	2020년 5월 13일

ElastiCache (Redis OSS)에서 이제 Global Datastore for Redis 지원 OSS	Global Datastore for Redis OSS 기능은 AWS 리전 간에 완전 관리형, 빠르고 안정적이며 안전한 복제를 제공합니다. 이 기능을 사용하면 ElastiCache (Redis OSS)에 대한 리전 간 읽기 전용 복제본 클러스터를 생성하여 AWS 리전 간 지연 시간이 짧은 읽기 및 재해 복구를 활성화할 수 있습니다. 글로벌 데이터 스토어를 생성, 수정 및 설명할 수 있습니다. 글로벌 데이터 스토어에서 AWS 리전을 추가하거나 제거하고 글로벌 데이터 스토어 내에서 AWS 리전을 프라이머리로 승격할 수도 있습니다. 자세한 내용은 Global Datastore를 사용하여 AWS 리전 간 복제를 참조하세요.	2020년 3월 16일
ElastiCache (Redis OSS)는 이제 Redis OSS 버전 5.0.6을 지원합니다.	자세한 내용은 ElastiCache (Redis OSS) 버전 5.0.6(고급)을 참조하세요.	2019년 12월 18일
ElastiCache 이제 Amazon에서 T3-Standard 캐시 노드 지원	이제 Amazon 에서 차세대 범용 버스트 가능 T3-Standard 캐시 노드를 시작할 수 있습니다 ElastiCache. Amazon EC2의 T3-Standard 인스턴스는 누적 크레딧이 소진될 때까지 언제든지 CPU 사용량을 버스트할 수 있는 기능과 함께 기준 수준의 CPU 성능을 제공합니다. 자세한 내용은 지원되는 노드 유형 섹션을 참조하세요.	2019년 11월 12일

Amazon은 ElastiCache 이제 기존 ElastiCache (Redis OSS) 서버의 AUTH 토큰 수정을 지원합니다.	ElastiCache (Redis OSS) 5.0.6을 사용하면 이제 새 토큰을 설정하고 교체하여 인증 토큰을 수정할 수 있습니다. 이제는 사용 중인 활성 토큰도 수정할 수 있습니다. 또한 이전에 인증 토큰 없이 전송 중 데이터 암호화가 활성화되도록 설정된 기존 클러스터에 새로운 토큰을 추가할 수 있습니다. 이러한 2단계 프로세스를 통해 클라이언트 요청을 중단시키지 않고 토큰을 설정 및 교체할 수 있습니다. 이 기능은 현재 에서 지원되지 않습니다 AWS CloudFormation. 자세한 내용은 Redis OSS AUTH 명령을 사용한 사용자 인증을 참조하세요.	2019년 10월 30일
Amazon은 ElastiCache 이제 AmazonOSS의 Redis에서 온라인 데이터 마이그레이션을 지원합니다. EC2	이제 온라인 마이그레이션을 사용하여 AmazonOSS의 자체 호스팅 Redis에서 Amazon 로 데이터를 마이그레이션EC2할 수 있습니다 ElastiCache. 자세한 내용은 로 온라인 마이그레이션을 ElastiCache 참조하세요.	2019년 10월 28일

ElastiCache (Redis OSS)는 Redis OSS 클러스터 모드에 대한 온라인 수직 조정을 도입합니다.	이제 샤딩된 Redis OSS 클러스터를 온디맨드로 확장하거나 축소할 수 있습니다. ElastiCache (Redis OSS)는 노드 유형을 변경하여 클러스터의 크기를 조정하지만 클러스터는 계속 온라인 상태를 유지하고 수신 요청을 처리합니다. 자세한 내용은 노드 유형 수정하여 온라인 수직 확장을 참조하세요.	2019년 8월 20일
ElastiCache (Redis OSS)를 사용하면 이제 사용자가 Amazon ElastiCache (Redis OSS) 클러스터에 단일 리더 엔드포인트를 사용할 수 있습니다.	이 기능을 사용하면 모든 읽기 트래픽을 단일 클러스터 수준 엔드포인트를 통해 ElastiCache (Redis OSS) 클러스터로 전달하여 로드 밸런싱 및 더 높은 가용성을 활용할 수 있습니다. 자세한 내용은 연결 엔드포인트 찾기를 참조하세요.	2019년 6월 13일
ElastiCache 이제 (Redis OSS) 사용자가 자신의 일정에 따라 서비스 업데이트를 적용할 수 있습니다.	이 기능을 활용하면 유지 관리 기간 뿐만 아니라 선택한 시간에 제공되는 서비스 업데이트를 적용할 수 있습니다. 이렇게 하면 특히 비즈니스 흐름이 가장 많을 때 서비스 중단이 최소화되고 클러스터가 ElastiCache지원되는 규정 준수 프로그램에 있는 경우 규정 준수를 유지할 수 있습니다. 자세한 내용은 Amazon의 셀프 서비스 업데이트 ElastiCache 및 Amazon 에 대한 규정 준수 검증 ElastiCache을 참조하세요.	2019년 6월 4일

ElastiCache 표준 예약 인스턴스 제공: 부분 선불, 전체 선불 및 사전 불가.	예약 인스턴스는 ElastiCache 인스턴스 유형 및 AWS 리전을 기반으로 1년 또는 3년 동안 Amazon 인스턴스를 예약할 수 있는 유연성을 제공합니다. 자세한 내용은 예약 노드로 비용 관리를 참조하세요.	2019년 1월 18일
ElastiCache (Redis OSS) Redis OSS 클러스터당 최대 250개의 노드 지원	노드 또는 샤드 제한은 ElastiCache (Redis OSS) 클러스터당 최대 250개까지 늘릴 수 있습니다. 자세한 내용은 샤드를 참조하세요.	2018년 11월 19일
ElastiCache 모든 T2 노드에서 자동 장애 조치 및 백업 및 복원에 대한 (Redis OSS) 지원	ElastiCache (Redis OSS)는 모든 T2 노드에서 오토페일오버, 스냅샷 생성, 백업 및 복원에 대한 지원을 도입합니다. 자세한 내용은 ElastiCache (Redis OSS) Backup and Restore and Snapshot을 참조하세요.	2018년 11월 19일
ElastiCache M5 및 R5 노드에 대한 (Redis OSS) 지원	ElastiCache (Redis OSS)는 이제 AWS Nitro 시스템을 기반으로 M5 및 R5 노드, 범용 및 메모리 최적화 인스턴스 유형을 지원합니다. 자세한 내용은 지원되는 노드 유형 섹션을 참조하세요.	2018년 10월 23일

동적으로 변경되는 읽기 전용 복제본 지원	ElastiCache (Redis OSS)는 클러스터 가동 중지 없이 모든 클러스터에서 읽기 전용 복제본을 추가하고 제거할 수 있는 지원을 추가했습니다. 이 릴리스의 이러한 변경 사항 및 기타 변경 사항에 대한 자세한 내용은 ElastiCache (Redis OSS) 사용 설명서의 복제본 수 변경을 참조하세요. 참조IncreaseReplicaCount의 DecreaseReplicaCount 및 도 참조하세요. ElastiCache API	2018년 9월 17일
연준RAMP 규정 준수 인증	ElastiCache (Redis OSS)는 이제 연준RAMP 규정 준수 인증을 받았습니다. 자세한 내용은 Amazon 에 대한 규정 준수 검증 ElastiCache을 참조하세요.	2018년 8월 30일
Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 엔진 업그레이드	Amazon ElastiCache (Redis OSS)은 Valkey 또는 RedisOSS(클러스터 모드 활성화됨) 엔진 버전 업그레이드에 대한 지원을 추가했습니다. 자세한 내용은 엔진 버전 업그레이드를 참조하세요.	2018년 8월 20일
PCI DSS 규정 준수 인증	ElastiCache (Redis OSS)는 이제 PCI DSS 규정 준수 인증을 받았습니다. 자세한 내용은 Amazon 에 대한 규정 준수 검증 ElastiCache을 참조하세요.	2018년 7월 5일

ElastiCache (Redis OSS) 4.0.10 지원	ElastiCache (Redis OSS)는 이제 암호화와 온라인 클러스터 크기 조정을 모두 단일 버전으로 포함하는 Redis OSS 4.0.10을 지원합니다. 자세한 내용은 ElastiCache (Redis OSS) 버전 4.0.10(고급)을 참조하세요.	2018년 6월 14일
사용 설명서 재구성	이제 단일 ElastiCache 사용 설명서가 재구성되어 RedisOSS(ElastiCache (Redis OSS) 사용 설명서) 및 Memcached ((Memcached) 사용 설명서 ElastiCache)에 대한 별도의 사용 설명서가 있습니다. AWS CLI 명령 참조: elasticache 섹션 및 Amazon ElastiCache API 참조의 설명서 구조는 변경되지 않습니다.	2018년 4월 20일
E ngineCPUUtilization 지표 지원	ElastiCache (Redis OSS)는 현재 사용 중인 의 용량 비율을 `EngineCPUUtilization` 보고하는 새 지표 CPU를 추가했습니다. 자세한 내용은 Redis 지표를 참조하세요OSS.	2018년 9월 4일

다음 표에서는 2018년 3월 이전의 ElastiCache 사용 설명서의 중요한 변경 사항을 설명합니다.

변경 사항	설명	변경 날짜
아시아 태평양(오사카-로컬) 리전 지원	ElastiCache 아시아 태평양(오사카-로컬) 리전에 대한 지원이 추가되었습니다. 아시아 태평양(오사카) 리전은 현재 단일 가용 영역을 지원하고 있으며 초대를 통해서만 이루어집니다. 자세한 내용은 다음 자료를 참조하세요.	2018년 2월 12일

변경 사항	설명	변경 날짜
	• 지원되는 리전 • 지원되는 캐시 노드 유형	
EU(파리) 지원	ElastiCache EU(파리) 리전에 대한 지원이 추가되었습니다. 자세한 내용은 다음 자료를 참조하세요. • 지원되는 리전 • 지원되는 캐시 노드 유형	2017년 12월 18일
중국(닝샤) 리전 지원	Amazon은 중국(닝샤) 리전에 대한 지원을 ElastiCache 추가했습니다. 자세한 내용은 다음 자료를 참조하세요. • 지원되는 리전 • 지원되는 캐시 노드 유형	2017년 12월 11일
서비스 연결 역할 지원	이번 서비스 연결 역할()에 대한 지원이 ElastiCache 추가되었습니다SLR. 자세한 내용은 다음 자료를 참조하세요. • Amazon용 서비스 연결 역할 사용 ElastiCache • 권한 설정(신규 ElastiCache 사용자만 해당)	2017년 12월 7일

변경 사항	설명	변경 날짜
R4 노드 유형 지원	추가된 이 릴리스는 에서 지원하는 모든 AWS 리전의 R4 노드 유형을 ElastiCache 지원합니다 ElastiCache. R4 노드 유형을 온디맨드 또는 예약 캐시 노드로 구입할 수 있습니다. 자세한 내용은 다음 자료를 참조하세요. • 지원되는 캐시 노드 유형 • Memcached 노드 유형별 파라미터 • OSS 노드 유형별 파라미터 재정의	2017년 11월 20일
ElastiCache (Redis OSS) 3.2.10 및 온라인 리샤딩 지원	Amazon ElastiCache (Redis OSS)은 ElastiCache (Redis OSS) 3.2.10에 대한 지원을 추가합니다. ElastiCache (Redis OSS)는 수신 I/O 요청을 계속 처리하는 동안 클러스터에서 샤드를 추가하거나 제거하기 위한 온라인 클러스터 크기 조정도 도입합니다. 자세한 내용은 다음 자료를 참조하세요. • 온라인 클러스터 크기 조정 • Valkey 또는 Redis에 대한 온라인 리샤딩OSS(클러스터 모드 활성화됨)	2017년 11월 9일
HIPAA 자격	ElastiCache 이제 클러스터에서 암호화가 활성화되면 (Redis OSS) 버전 3.2.6의 HIPAA 적합성이 인증됩니다. 자세한 내용은 다음 자료를 참조하세요. • Amazon에 대한 규정 준수 검증 ElastiCache • Amazon의 데이터 보안 ElastiCache	2017년 11월 2일

변경 사항	설명	변경 날짜
ElastiCache (Redis OSS) 3.2.6 및 암호화 지원	ElastiCache 는 두 가지 암호화 기능을 포함하는 ElastiCache (Redis OSS) 3.2.6에 대한 지원을 추가합니다. • 전송 중 데이터 암호화는 클러스터 내 노드 사이 또는 클러스터와 애플리케이션 사이와 같이 전송 중인 경우 항상 데이터를 암호화합니다. • 미사용 데이터 암호화는 동기화 및 백업 작업 중 디스크 내 데이터를 암호화합니다. 자세한 내용은 다음 자료를 참조하세요. • Amazon의 데이터 보안 ElastiCache • 지원되는 엔진 및 버전	2017년 10월 25일
연결 패턴 항목	ElastiCache 설명서는 Amazon 에서 클러스터에 액세스 ElastiCache하기 위한 다양한 패턴을 다루는 주제를 추가합니다VPC. 자세한 내용은 사용 설명서Amazon에서 ElastiCache 캐시에 액세스하기 위한 액세스 패턴 VPC의 섹션을 참조하세요. ElastiCache	2017년 4월 24일
Memcached 1.4.34 지원	ElastiCache 는 이전 Memcached 버전에 여러 수정 사항을 통합하는 Memcached 버전 1.4.34을 지원합니다. 자세한 내용은 의 Memcached에서 Memcached 1.4.34 릴리스 정보를 참조하세요GitHub.	2017년 10월 4일

변경 사항	설명	변경 날짜
자동 장애 조치 테스트 지원	ElastiCache 는 복제를 지원하는 Redis OSS 클러스터에서 자동 장애 조치 테스트를 지원합니다. 자세한 내용은 다음 자료를 참조하세요. • 자동 장애 조치 테스트(출처: ElastiCache 사용 설명서). • TestFailover ElastiCache API 참조 의 . • AWS CLI 참조의 test-failover	2017년 4월 4일
향상된 Redis OSS 복원	ElastiCache 는 클러스터 크기 조정을 사용하여 향상된 Redis OSS 백업 및 복원을 추가합니다. 이 기능은 백업 생성에 사용되는 클러스터와 샤드 수가 다른 클러스터에 백업을 복원하도록 지원합니다. (API 및 의 경우 CLI이 기능은 다른 수의 샤드가 아닌 다른 수의 노드 그룹을 복원할 수 있습니다.) 이 업데이트는 다양한 Redis OSS 슬롯 구성도 지원합니다. 자세한 내용은 백업에서 새 캐시로 복원 단원을 참조하십시오.	2017년 3월 15일
새 Redis OSS 메모리 관리 파라미터	ElastiCache 는 새 Redis OSS 파라미터`reserved-memory-percent` 인 를 추가하므로 예약 메모리를 더 쉽게 관리할 수 있습니다. 이 파라미터는 ElastiCache (Redis)의 모든 버전에서 사용할 수 있습니다OSS. 자세한 내용은 다음 자료를 참조하세요. • Valkey 및 Redis용 예약 메모리 관리 OSS • Redis OSS 3.2.4의 새 파라미터	2017년 3월 15일

변경 사항	설명	변경 날짜
Memcached 1.4.33 지원	ElastiCache 는 Memcached 버전 1.4.33에 대한 지원을 추가합니다. 자세한 내용은 다음 자료를 참조하세요. • Memcached 버전 1.4.33 • Memcached 1.4.33 추가 파라미터	2016년 12월 20일
EU 서부(런던) 리전 지원	ElastiCache 는 EU(런던) 리전에 대한 지원을 추가합니다. 현재는 노드 유형 T2 및 M4만 지원됩니다. 자세한 내용은 다음 자료를 참조하세요. • 지원되는 리전 • 지원되는 캐시 노드 유형	2016년 12월 13일
캐나다(몬트리올) 리전 지원	ElastiCache 는 캐나다(몬트리올) 리전에 대한 지원을 추가합니다. 현재 이 AWS 리전에서는 노드 유형 M4 및 T2만 지원됩니다. 자세한 내용은 다음 자료를 참조하세요. • 지원되는 리전 • 지원되는 캐시 노드 유형	2016년 12월 8일
M4 및 R3 노드 유형 지원	ElastiCache 는 남아메리카(상파울루) 리전의 R3 및 M4 노드 유형과 중국(베이징) 리전의 M4 노드 유형에 대한 지원을 추가합니다. 자세한 내용은 다음 자료를 참조하세요. • 지원되는 리전 • 지원되는 캐시 노드 유형	2016년 11월 1일

변경 사항	설명	변경 날짜
미국 동부 2(오하이오) 리전 지원	ElastiCache 는 M4, T22 및 R3 노드 유형이 있는 미국 동부(오하이오) 리전(us-east-2)에 대한 지원을 추가합니다. R3 자세한 내용은 다음 자료를 참조하세요. • 지원되는 리전 • 지원되는 캐시 노드 유형	2016년 10월 17일
Redis OSS 클러스터 지원	ElastiCache 는 Redis OSS 클러스터(향상됨)에 대한 지원을 추가합니다. Redis OSS 클러스터를 사용하는 고객은 최대 15개의 샤드(노드 그룹)로 데이터를 분할할 수 있습니다. 각 샤드는 샤드당 읽기 전용 복제본이 최대 5개인 복제를 지원합니다. Redis OSS 클러스터 자동 장애 조치 시간은 이전 버전의 시간보다 약 1/4 정도 깁니다. 이 릴리스에는 업계의 용도에 맞게 용어를 사용하도록 다시 고안된 관리 콘솔에 포함되어 있습니다. 자세한 내용은 다음 자료를 참조하세요. • Memcached와 Redis 비교 OSS • ElastiCache 구성 요소 및 기능 - 노드, 샤드, 클러스터 및 복제에 대한 섹션을 참조하세요. • ElastiCache 용어	2016년 10월 12일

변경 사항	설명	변경 날짜
M4 노드 유형 지원	ElastiCache 는 에서 지원하는 대부분의 AWS 리전에서 M4 노드 유형 패밀리에 대한 지원을 추가합니다ElastiCache. M4 노드 유형을 온디맨드 또는 예약 캐시 노드로 구입할 수 있습니다. 자세한 내용은 다음 자료를 참조하세요. • 지원되는 캐시 노드 유형 • Memcached 노드 유형별 파라미터 • OSS 노드 유형별 파라미터 재정의	2016년 8월 3일
뭄바이 리전 지원	ElastiCache 는 아시아 태평양(뭄바이) 리전에 대한 지원을 추가합니다. 자세한 내용은 다음 자료를 참조하세요. • 지원되는 캐시 노드 유형 • Memcached 노드 유형별 파라미터 • OSS 노드 유형별 파라미터 재정의	2016년 6월 27일
스냅샷 내보내기	ElastiCache 는 외부에서 액세스할 수 있도록 Redis OSS 스냅샷을 내보낼 수 있는 기능을 추가합니다ElastiCache. 자세한 내용은 다음 자료를 참조하세요. • 백업 내보내기 Amazon ElastiCache 사용 설명서의 • CopySnapshot Amazon ElastiCache API 참조의	2016년 5월 26일
노드 유형 확장	ElastiCache 는 Redis OSS 노드 유형을 확장하는 기능을 추가합니다. 자세한 내용은 크기 조정 ElastiCache 단원을 참조하십시오.	2016년 3월 24일
간편한 엔진 업그레이드	ElastiCache 는 Redis OSS 캐시 엔진을 쉽게 업그레이드할 수 있는 기능을 추가합니다. 자세한 내용은 용 버전 관리 ElastiCache 단원을 참조하십시오.	2016년 3월 22일

변경 사항	설명	변경 날짜
R3 노드 유형 지원	ElastiCache 는 중국(베이징) 리전 및 남아메리카(상파울루) 리전의 R3 노드 유형에 대한 지원을 추가합니다. 자세한 내용은 지원되는 캐시 노드 유형을 참조하세요.	2016년 3월 16일
Lambda 함수를 ElastiCache 사용하여 액세스	Amazon ElastiCache 에서 액세스하도록 Lambda 함수를 구성하는 방법에 대한 자습서가 추가되었습니다VPC. 자세한 내용은 기타 ElastiCache 자습서 및 동영상 단원을 참조하십시오.	2016년 2월 12일
Redis OSS 2.8.24 지원	ElastiCache 는 Redis OSS 버전 2.8.24에 대한 지원을 추가하고 Redis OSS 2.8.23 이후 개선 사항을 추가했습니다. 개선 사항에는 버그 수정 및 불량 메모리 액세스 주소의 로깅에 대한 지원이 포함됩니다. 자세한 내용은 다음 자료를 참조하세요. • ElastiCache (Redis OSS) 버전 2.8.24(향상됨) • Redis OSS 2.8 릴리스 정보	2016년 1월 20일
아시아 태평양(서울) 리전 지원	ElastiCache 는 t2, m3 및 r3 노드 유형이 있는 아시아 태평양(서울)(ap-northeast-2) 리전에 대한 지원을 추가합니다.	2016년 1월 6일
Amazon ElastiCache 콘솔이 변경됩니다.	최신 Redis OSS 버전은 더 우수하고 안정적인 사용자 환경을 제공하기 때문에 Redis OSS 버전 2.6.13, 2.8.6 및 2.8.19은 더 이상 ElastiCache 관리 콘솔에 나열되지 않습니다. 기타 옵션과 자세한 내용은 지원되는 엔진 및 버전 섹션을 참조하세요.	2015년 12월 15일

변경 사항	설명	변경 날짜
Redis OSS 2.8.23 지원.	ElastiCache 는 Redis OSS 버전 2.8.23에 대한 지원을 추가하고 Redis OSS 2.8.22 이후 개선 사항을 추가했습니다. 개선 사항에는 버그 수정과 새 파라미터 `close-on-slave-write` 에 대한 지원도 포함됩니다. 이 파라미터가 활성화되면 읽기 전용 복제본에 쓰려고 시도하는 클라이언트를 연결 해제합니다. 자세한 내용은 ElastiCache (Redis OSS) 버전 2.8.23(향상됨) 단원을 참조하십시오.	2015년 11월 13일
Redis OSS 2.8.22 지원.	ElastiCache 는 OSS 버전 2.8.22 이후 향상된 기능 및 개선 사항을 ElastiCache 추가하여 Redis 버전 2.8.21에 대한 지원을 추가합니다. 개선 사항에는 다음이 포함됩니다. • 사용 가능한 메모리가 부족하여 forked 저장이 실패할 수 있는 경우 저장을 성공하게 해주는 forkless 저장 프로세스 구현 • 추가 CloudWatch 지표 — SaveInProgress 및 ReplicationBytes. • 부분 동기화를 활성화하려면 `repl-backlog-size` 이제 Redis OSS 파라미터가 모든 클러스터에 적용됩니다. 전체 변경 사항 목록과 자세한 내용은 ElastiCache (Redis OSS) 버전 2.8.22(향상됨) 섹션을 참조하세요. 이 설명서 릴리스에는 설명서의 재구성 및 ElastiCache 명령줄 인터페이스(CLI) 설명서의 제거가 포함되어 있습니다. 명령줄 사용에 대해서는 의 AWS 명령줄을 참조하세요 ElastiCache.	2015년 9월 28일

변경 사항	설명	변경 날짜
Memcached 1.4.28 지원	ElastiCache 는 버전 1.4.24 이후 Memcached 버전 1.4.14 및 Memcached 개선 사항에 대한 지원을 추가합니다. 이 릴리스에서는 가장 최근에 사용되지 않은 (LRU) 캐시 관리를 백그라운드 작업으로, 진킨 또는 잡음3을 해싱 알고리즘으로 선택, 새 명령 및 기타 버그 수정으로 지원합니다. 자세한 내용은 Memcached 릴리스 정보를 참조하세요. .	2015년 8월 27일
PHP 5.6을 사용한 Memcached Auto Discovery 지원	이 Amazon 릴리스에서는 PHP 버전 5.6용 Memcached Auto Discovery 클라이언트에 대한 지원을 ElastiCache 추가합니다. 자세한 내용은 에 대한 ElastiCache 클러스터 클라이언트의 소스 코드 컴파일 PHP 단원을 참조하십시오.	2015년 7월 29일
Redis OSS 2.8.21 지원	ElastiCache 는 버전 2.8.21 이후 Redis OSS 버전 2.8.19 및 Redis OSS 개선 사항에 대한 지원을 추가합니다. 이 Redis OSS 릴리스에는 여러 버그 수정이 포함되어 있습니다. 자세한 내용은 Redis OSS 2.8 릴리스 정보 섹션을 참조하세요.	2015년 7월 29일
새 주제: 외부 ElastiCache 에서 액세스 AWS	외부에서 ElastiCache 리소스에 액세스하는 방법에 대한 새 주제가 추가되었습니다 AWS. 자세한 내용은 외부 ElastiCache 에서 액세스를 참조하세요 AWS.	2015년 7월 9일
노드 대체 메시지 추가	ElastiCache 는 예약된 노드 교체 NodeReplacementScheduled:, ElastiCache: ElastiCacheNodeReplacementRescheduled및 ElastiCache:와 관련된 세 가지 메시지를 추가합니다NodeReplacementCanceled. 노드 교체가 예약될 때 수행할 수 있는 자세한 내용과 작업은 ElastiCache의 섹션을 참조하세요이벤트 알림 및 Amazon SNS.	2015년 6월 11일

변경 사항	설명	변경 날짜
Redis OSS v. 2.8.19 지원.	ElastiCache 는 OSS 버전 2.8.6 이후 Redis 버전 2.8.19 및 Redis OSS 개선 사항에 대한 지원을 추가합니다. 여기에는 다음에 대한 지원이 포함되어 있습니다. • Redis OSS 명령 , PFCOUNT및 PFADD를 사용하는 HyperLogLog 데이터 구조입니다PFMERGE. • 새 명령 ZRANGEBYLEX, ZLEXCOUNT및 를 사용한 렉시코그래픽 범위 쿼리입니다ZREMRANGEBYLEX. • 백그라운드 저장(bgsave) 하위 프로세스가 예기치 않게 종료될 SYNC 때 기본 노드에 장애가 발생하여 기본 노드가 복제본 노드로 오래된 데이터를 보내는 것을 방지하는 여러 버그 수정을 도입했습니다. 에 대한 자세한 내용은 Redis OSS 새 데이터 구조: 를 HyperLogLog HyperLogLog참조하세요. PFADD, 및 에 대한 자세한 내용은 Redis OSS 설명서를 PFCOUNT PFMERGE참조하고 를 클릭합니다HyperLogLog.	2015년 3월 11일
비용 할당 태그 지원	ElastiCache 는 비용 할당 태그에 대한 지원을 추가합니다. 자세한 내용은 비용 할당 태그를 사용하여 비용을 모니터링합니다. 단원을 참조하십시오.	2015년 2월 9일
AWS GovCloud (미국 서부) 리전 지원	ElastiCache 는 AWS GovCloud (미국 서부) (us-gov-west-1) 리전에 대한 지원을 추가합니다.	2015년 1월 29일

변경 사항	설명	변경 날짜
유럽(프랑크푸르트) 리전 지원	ElastiCache 는 유럽(프랑크푸르트)(eu-central-1) 리전에 대한 지원을 추가합니다.	2015년 1월 19일
Redis OSS 복제 그룹에 대한 다중 AZ 지원	ElastiCache 는 기본 노드의 다중 AZ에 대한 지원을 Redis OSS 복제 그룹의 읽기 전용 복제본에 추가합니다. 는 복제 그룹의 상태를 ElastiCache 모니터링합니다. 기본 에 장애가 발생하면 는 자동으로 복제본을 기본 로 ElastiCache 승격한 다음 복제본을 대체합니다. 자세한 내용은 Valkey 및 Redis와 함께 다중 AZ ElastiCache 를 사용하여 의 가동 중지 시간 최소화 OSS 단원을 참조하십시오.	2014년 10월 24일
AWS CloudTrail 지원되는 API 통화 로깅	ElastiCache 는 를 사용하여 모든 ElastiCache API 통화를 로깅 AWS CloudTrail 하는 데 대한 지원을 추가합니다. 자세한 내용은 를 사용하여 Amazon ElastiCache API 통화 로깅 AWS CloudTrail 단원을 참조하십시오.	2014년 9월 15일
새로운 인스턴스 크기 지원	ElastiCache 는 추가 범용(T2) 인스턴스에 대한 지원을 추가합니다. 자세한 내용은 파라미터 그룹을 사용하여 엔진 ElastiCache 파라미터 구성 단원을 참조하십시오.	2014년 9월 11일
Memcached에서 유연한 노드 배치 지원	ElastiCache 는 여러 가용 영역에서 Memcached 노드를 생성하는 데 대한 지원을 추가합니다.	2014년 7월 23일
새로운 인스턴스 크기 지원	ElastiCache 는 추가 범용(M3) 인스턴스 및 메모리 최적화(R3) 인스턴스에 대한 지원을 추가합니다. 자세한 내용은 파라미터 그룹을 사용하여 엔진 ElastiCache 파라미터 구성 단원을 참조하십시오.	2014년 7월 1일
PHP 자동 검색	PHP 버전 5.5 자동 검색에 대한 지원이 추가되었습니다.	2014년 5월 13일

변경 사항	설명	변경 날짜
Redis OSS 클러스터의 백업 및 복원	이 릴리스에서 는 고객이 Redis OSS 클러스터의 스냅샷을 생성하고 이러한 스냅샷을 사용하여 새 클러스터를 생성할 수 있도록 ElastiCache 허용합니다. 백업은 특정 시점에 클러스터의 복사본이며 클러스터 메타데이터와 Redis OSS 캐시의 모든 데이터로 구성됩니다. 백업은 Amazon S3에 저장되며 고객은 스냅샷의 데이터를 언제든지 새로운 클러스터에 복원할 수 있습니다. 자세한 내용은 스냅샷 및 복원 단원을 참조하십시오.	2014년 4월 24일
Redis OSS 2.8.6	ElastiCache 는 Redis OSS 2.6.13 외에도 Redis OSS 2.8.6을 지원합니다. Redis OSS 2.8.6을 사용하면 부분 재동기화를 지원하여 읽기 전용 복제본의 복원력과 내결함성을 개선하고 항상 사용할 수 있어야 하는 사용자 정의 최소 읽기 전용 복제본 수를 개선할 수 있습니다. 또한 Redis OSS 2.8.6은 서버에서 발생하는 이벤트를 클라이언트에 알릴 수 publish-and-subscribe있는 에 대한 완전한 지원을 제공합니다.	2014년 3월 13일

변경 사항	설명	변경 날짜
Redis OSS 캐시 엔진	ElastiCache 는 Memcached 외에도 Redis OSS 캐시 엔진 소프트웨어를 제공합니다. 현재 Redis를 사용하는 고객은 ElastiCache Redis OSS 스냅샷 파일의 기존 데이터가 포함된 새 Redis OSS 캐시 클러스터를 '시드'하여 관리형 ElastiCache 환경으로 쉽게 마이그레이션할 OSS 수 있습니다. Redis OSS 복제 기능을 지원하기 위해 는 ElastiCache API 이제 복제 그룹을 지원합니다. 고객은 기본 Redis OSS 캐시 노드로 복제 그룹을 생성하고 기본 노드의 캐시 데이터와 자동으로 동기화되는 하나 이상의 읽기 전용 복제본 노드를 추가할 수 있습니다. 읽기 집중형 애플리케이션의 경우 읽기 전용 복제본으로 로드를 덜어 기본 노드의 로드를 줄일 수 있습니다. 또한 읽기 전용 복제본은 기본 캐시 노드 실패 시 데이터 손실을 방지할 수 있습니다.	2013년 9월 3일
기본 Amazon Virtual Private Cloud 지원(VPC)	이 릴리스에서는 ElastiCache 가 Amazon Virtual Private Cloud()와 완전히 통합됩니다VPC. 신규 고객의 경우 캐시 클러스터는 VPC 기본적으로 Amazon에서 생성됩니다. 자세한 내용은 Amazon VPCs 및 ElastiCache 보안 단원을 참조하십시오.	2013년 1월 8일
PHP 캐시 노드 자동 검색 지원	캐시 노드 Auto Discovery는 초기 릴리스부터 Java 프로그램에 대한 지원을 제공했습니다. 이 릴리스에서는 캐시 노드 자동 검색 지원을 에 ElastiCache 제공합니다PHP.	2013년 1월 2일

변경 사항	설명	변경 날짜
Amazon Virtual Private Cloud 지원(VPC)	이 릴리스에서는 ElastiCache 클러스터를 Amazon Virtual Private Cloud()에서 시작할 수 있습니다 VPC. 기본적으로 신규 고객의 캐시 클러스터는 Amazon에서 VPC 자동으로 생성되며, 기존 고객은 원하는 속도로 Amazon으로 마이그레이션할 수 VPC 있습니다. 자세한 내용은 Amazon VPCs 및 ElastiCache 보안 단원을 참조하십시오.	2012년 20월 12일
캐시 노드 자동 검색 및 새로운 캐시 엔진 버전	ElastiCache 는 클라이언트 프로그램이 클러스터의 모든 캐시 노드를 자동으로 결정하고 이러한 모든 노드에 대한 연결을 시작하고 유지하는 기능인 캐시 노드 자동 검색을 제공합니다. 이 릴리스는 또한 새로운 캐시 엔진 버전인 Memcached 버전 1.4.14를 제공합니다. 이 새 캐시 엔진은 개선된 슬래브 균형 다시 맞추기 기능, 중요한 성능 및 확장성 향상 및 여러 개의 버그 수정을 제공합니다. 여러 가지 새로운 캐시 파라미터를 구성할 수 있습니다. 자세한 내용은 파라미터 그룹을 사용하여 엔진 ElastiCache 파라미터 구성 단원을 참조하십시오.	2012년 11월 28일
새로운 캐시 노드 유형	이 릴리스에서는 캐시 노드 유형 4개를 추가로 제공합니다.	2012년 11월 13일
예약 캐시 노드	이 릴리스는 예약된 캐시 노드에 대한 지원을 추가합니다. q	2012년 4월 5일
새 안내서	Amazon ElastiCache 사용 설명서 의 첫 번째 릴리스입니다.	2011년 8월 22일

AWS 용어집

최신 AWS 용어는 AWS 용어집 참조 의 AWS 용어집을 참조하세요.

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