ElastiCache Memcached 사용 설명서

아마존 ElastiCache

API 버전 2015-02-02

아마존 ElastiCache: ElastiCache Memcached 사용 설명서

Table of Contents

멤캐시드의 ElastiCache 용도는 무엇입니까? 1
서버리스 캐시 1
자체 설계된 클러스터 1
작동 방식 2
캐시 및 캐싱 엔진 2
사용 사례 7
인 메모리 데이터 스토어 7
ElastiCache 고객 추천사 8
ElastiCache for Memcached 리소스 9
구현 관리용 도구 11
AWS Management Console 사용 11
AWS CLI 사용 11
AWS SDK 사용 11
ElastiCache API 사용 11
다음 사항도 참조하십시오. 11
배포 옵션 간 선택 12
Memcached 캐시와 Redis 자체 설계된 캐시 비교 13
ElastiCache for Memcached 시작하기 18
설정 18
가입하세요 AWS 계정 18
관리자 액세스 권한이 있는 사용자 생성 19
프로그래밍 방식 액세스 권한 부여 20
권한 설정 21
EC2 설정 22
네트워크 액세스 권한 부여 23
캐시 생성 23
서버리스 캐시 생성 23
데이터 읽기 및 쓰기 25
(선택 사항) 정리 29
다음 단계 30
자습서: Amazon VPC에서 Amazon ElastiCache에 액세스하도록 Lambda 함수 구성 31
1단계: ElastiCache 캐시 생성 31
2단계: Lambda 함수 생성 33
3단계: Lambda 함수 테스트 37

자습서 및 동영상 39
동영상 40
리전 및 가용 영역 선택 45
가용 영역 고려 사항 46
지원되는 리전 및 엔드포인트 48
노드 찾기 53
로컬 영역 사용 53
로컬 영역 활성화 54
Outposts 사용 54
Memcached 콘솔에서 Outposts 사용 55
CLI를 통한 아웃포스트 사용 AWS 56
자체 ElastiCache 클러스터 설계 58
ElastiCache Memcached 구성 요소 및 기능용 59
노드 59
클러스터 60
AWS 지역 및 가용 영역 61
엔드포인트 62
파라미터 그룹 62
보안 63
서브넷 그룹 63
이벤트 63
클러스터 관리 63
네트워크 유형 선택 65
데이터 계층화 67
Auto Discovery 67
클러스터 준비 108
클러스터 생성 114
클러스터 세부 정보 보기 117
클러스터 수정 122
클러스터 재부팅 125
클러스터에 노드 추가 127
클러스터에서 노드 제거 133
대기 중인 노드 추가 또는 삭제 작업 취소 139
클러스터 삭제 140
클러스터에 액세스 142
연결 엔드포인트 찾기 148

노드 관리 154
ElastiCache 노드 상태 보기 155
노드에 연결 161
지원되는 노드 유형 163
노드 교체 172
예약 노드 174
이전 세대 노드 마이그레이션 185
함께 일하기 ElastiCache 187
스냅샷 및 복원 187
제약 조건 188
자동 백업 예약 189
수동 백업 지원 190
최종 백업 생성 192
백업 설명 194
백업 복사 195
백업에서 복원 197
백업 삭제 198
백업 태그 지정 199
엔진 버전 및 업그레이드 200
지원되는 Memcached 버전 201
엔진 버전 및 업그레이드 205
엔진 버전 업그레이드 방법 206
모범 사례 및 캐싱 전략 207
Memcached 클라이언트 사용 모범 사례 207
지원되는 Memcached 명령 214
캐싱 전략 215
자체 설계된 클러스터 관리 220
유지 관리 관리 중 221
파라미터 그룹을 사용해 엔진 파라미터 구성 223
ElastiCache Memcached를 위한 스케일링 264
멤캐시드 ElastiCache 스케일링 264
비용 관리를 위한 규모 조정 한도 설정 264
서버리스를 통한 사전 스케일링 ElastiCache 264
콘솔을 사용한 규모 조정 제한 설정 및 AWS CLI 265
Memcached ElastiCache 자체 설계 클러스터를 위한 확장 267
ElastiCache 리소스에 태그 지정 271

태그를 사용한 비용 모니터링 ... 278
AWS CLI를 사용하여 태그 관리 ... 279
ElastiCache API를 사용한 태그 관리 ... 283
Amazon ElastiCache의 Well-Architected Lens ... 285
운영 우수성 요소 ... 286
보안 요소 ... 293
신뢰성 요소 ... 299
성능 효율성 요소 ... 304
비용 최적화 요소 ... 314
문제 해결 ... 319
연결 문제 ... 320
Redis 클라이언트 오류 ... 320
서버리스의 ElastiCache 높은 지연 시간 문제 해결 ... 321
서버리스의 스로틀링 문제 해결 ElastiCache ... 322
관련 항목 ... 323
추가 문제 해결 단계 ... 323
보안 그룹 ... 324
네트워크 ACL ... 324
라우팅 테이블 ... 326
DNS 확인 ... 326
서버 측 진단을 사용하여 문제 식별 ... 327
네트워크 연결 검증 ... 332
네트워크 관련 제한 사항 ... 334
CPU 사용량 ... 335
서버 측에서 연결이 종료되는 경우 ... 338
Amazon EC2 인스턴스에 대한 클라이언트 측 문제 해결 ... 339
단일 요청을 완료하는 데 걸리는 시간 분석 ... 340
보안 ... 344
데이터 보호 ... 344
Amazon ElastiCache의 데이터 보안 ... 345
인터네트워크 트래픽 개인 정보 ... 354
Amazon VPC 및 ElastiCache 보안 ... 354
Amazon ElastiCache API 및 인터페이스 VPC 엔드포인트(AWS PrivateLink) ... 376
서브넷 및 서브넷 그룹 ... 379
ID 및 액세스 관리 ... 387
고객 ... 387

ID를 통한 인증 388
정책을 사용한 액세스 관리 391
아마존이 IAM과 협력하는 ElastiCache 방식 393
자격 증명 기반 정책 예시 400
문제 해결 403
액세스 제어 404
액세스 관리 개요 405
규정 준수 확인 434
추가 정보 435
복원성 436
장애 완화 436
인프라 보안 438
서비스 업데이트 438
서비스 업데이트 관리 439
로깅 및 모니터링 445
서버리스 지표 및 이벤트 445
서버리스 지표 445
서버리스 이벤트 448
자체 설계된 지표 및 이벤트 452
자체 설계된 클러스터 지표 452
자체 설계된 클러스터 이벤트 453
사용량 모니터링 460
Amazon SNS 이벤트 모니터링 472
AWS CloudTrail을 사용하여 Amazon ElastiCache API 호출 로깅 488
CloudTrail의 Amazon ElastiCache 정보 488
Amazon ElastiCache 로그 파일 항목 이해 489
할당량 494
Reference 495
ElastiCache API 사용 495
Query API 사용 495
사용 가능한 라이브러리 499
애플리케이션 문제 해결 499
ElastiCache용 AWS CLI 설정 500
필수 조건 501
명령줄 도구 얻기 502
도구 설정 502

도구에 대한 자격 증명 제공 503
환경 변수 504
오류 메시지 505
알림 506
일반 ElastiCache 알림 507
ElastiCache for Memcached 알림 507
문서 기록 508
AWS 용어집 521
..... dxxii

멤캐시드를 ElastiCache 위한 아마존이란 무엇입니까?

ElastiCache Memcached용 Amazon 사용 설명서에 오신 것을 환영합니다. ElastiCache Amazon은 클라우드에서 분산된 인메모리 데이터 스토어 또는 캐시 환경을 쉽게 설정, 관리 및 확장할 수 있게 해주는 웹 서비스입니다. 확장 가능하고 비용 효율적인 고성능 캐싱 솔루션을 제공합니다. 또한 분산된 캐시 환경의 배포 및 관리와 관련된 복잡성을 해소할 수 있습니다.

Amazon은 두 가지 ElastiCache 형식으로 운영할 수 있습니다. 서버리스 캐시로 시작하거나 자체 캐시 클러스터를 설계하도록 선택할 수 있습니다.

> **Note**
>
> Amazon은 Redis 엔진과 Memcached 엔진 모두에서 ElastiCache 작동합니다. 관심 있는 엔진에 대해 설명한 가이드를 사용하세요. 사용하고 싶은 엔진을 결정하기 어렵다면 이 가이드의 Memcached 캐시와 Redis 자체 설계된 캐시 비교 섹션을 참조하세요.

서버리스 캐시

ElastiCache for Memcached는 애플리케이션에 Memcached 기반 캐시를 추가하고 운영하는 작업을 간소화하는 서버리스 캐싱을 제공합니다. ElastiCache Memcached Serverless의 경우 1분 이내에 고가용성 캐시를 생성할 수 있으며 인스턴스를 프로비저닝하거나 노드 또는 클러스터를 구성할 필요가 없습니다. 개발자는 ElastiCache 콘솔, SDK 또는 CLI를 사용하여 캐시 이름을 지정하여 서버리스 캐시를 생성할 수 있습니다.

ElastiCache 또한 서버리스를 사용하면 캐싱 용량을 계획하고 관리할 필요가 없습니다. ElastiCache 애플리케이션에서 사용하는 캐시 메모리와 컴퓨팅을 지속적으로 모니터링하고 애플리케이션 요구 사항에 맞게 용량을 자동으로 확장합니다. ElastiCache 기본 캐시 인프라 및 소프트웨어를 추상화하여 개발자에게 간단한 엔드포인트 경험을 제공합니다. ElastiCache 하드웨어 프로비저닝, 모니터링, 노드 교체 및 소프트웨어 패치를 자동으로 투명하게 관리하므로 캐시를 운영하는 대신 애플리케이션 개발에 집중할 수 있습니다.

ElastiCache Memcached 서버리스의 경우 Memcached 1.6 이상 버전과 호환됩니다.

Memcached 클러스터를 위한 자체 설계 ElastiCache

Memcached 클러스터를 세밀하게 제어해야 하는 경우, Memcached 클러스터를 사용하여 ElastiCache 자체 Memcached 클러스터를 설계할 수 있습니다. ElastiCache ElastiCache 클러스터의 노드 유형,

노드 수, 가용 영역 전반의 노드 배치를 선택하여 노드 기반 클러스터를 운영할 수 있습니다. AWS ElastiCache 는 완전 관리형 서비스이므로 클러스터의 하드웨어 프로비저닝, 모니터링, 노드 교체 및 소프트웨어 패치를 자동으로 관리합니다.

Memcached ElastiCache 클러스터에 맞게 자체 설계하면 클러스터를 보다 유연하게 제어할 수 있습니다. 예를 들어 필요에 따라 단일 AZ 가용성 또는 교차 AZ 가용성으로 클러스터를 운영하도록 선택할 수 있습니다. 클러스터를 직접 설계할 때는 애플리케이션에 필요한 만큼 캐시 용량이 충분하도록 노드 유형과 수를 올바르게 선택해야 합니다. 또한 Memcached 클러스터에 새 소프트웨어 패치를 적용할 시기를 선택할 수 있습니다.

ElastiCache Memcached에 맞게 직접 설계할 때는 Memcached 1.4 이상을 실행하도록 선택할 수 있습니다.

작동 방식

여기에서 Memcached 배포의 주요 구성 요소에 ElastiCache 대한 개요를 확인할 수 있습니다.

캐시 및 캐싱 엔진

캐시는 캐시된 데이터를 저장하는 데 사용할 수 있는 인메모리 데이터 저장소입니다. 일반적으로 애플리케이션은 응답 시간을 최적화하기 위해 자주 액세스하는 데이터를 캐시에 캐시합니다. ElastiCache for Memcached는 서버리스 클러스터와 자체 설계 클러스터라는 두 가지 배포 옵션을 제공합니다. 배포 옵션 간 선택 섹션 참조

> **Note**
>
> Amazon은 Redis 엔진과 Memcached 엔진 모두에서 ElastiCache 작동합니다. 관심 있는 엔진에 대해 설명한 가이드를 사용하세요. 사용하고 싶은 엔진을 결정하기 어렵다면 이 가이드의 Memcached 캐시와 Redis 자체 설계된 캐시 비교 섹션을 참조하세요.

주제

- 멤캐시드의 작동 방식 ElastiCache
- 차원 사용

멤캐시드의 작동 방식 ElastiCache

ElastiCache 멤캐시드 서버리스용.

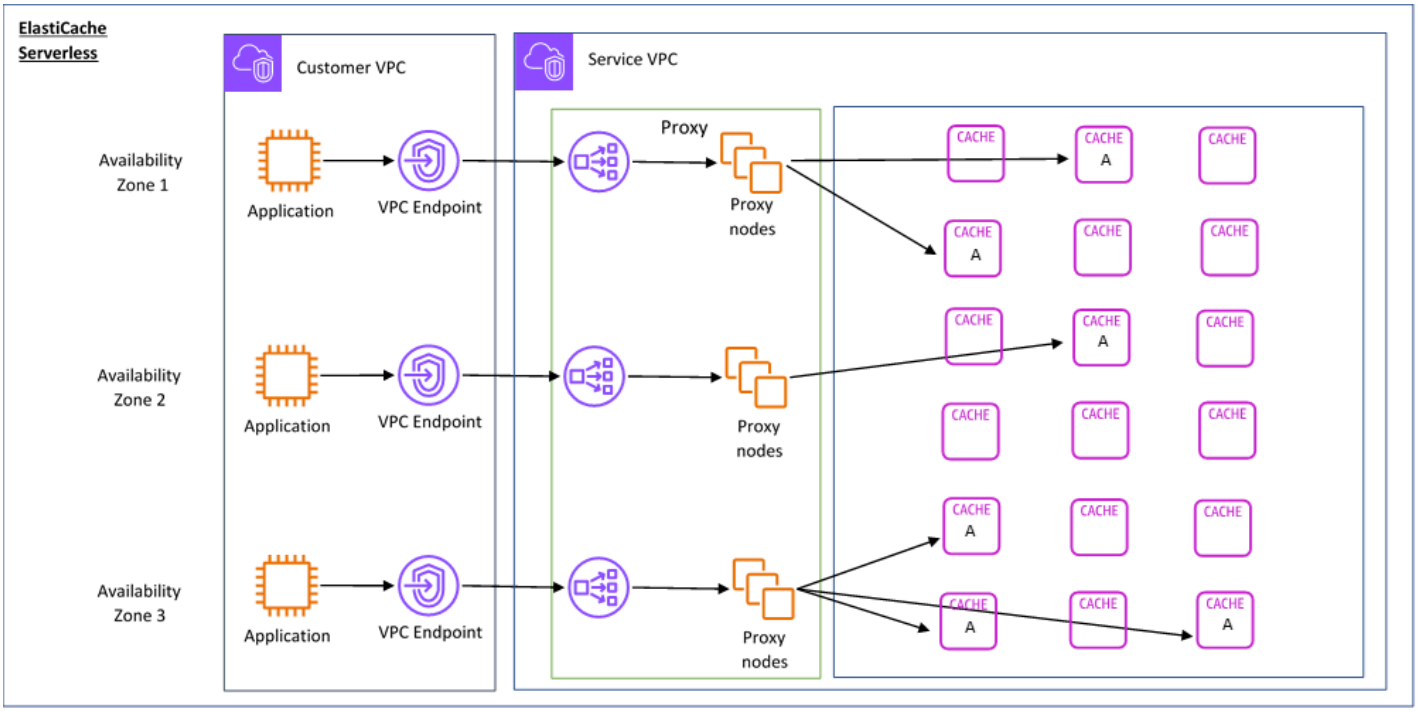
ElastiCache Memcached Serverless의 경우 용량 계획, 하드웨어 관리 또는 클러스터 설계에 대한 걱정 없이 캐시를 생성할 수 있습니다. 캐시 이름만 입력하면 Memcached 클라이언트에서 캐시 액세스를 시작하도록 구성할 수 있는 단일 엔드포인트를 받게 됩니다.

> **Note**
>
> ElastiCache Memcached 서버리스의 경우 TLS를 지원하는 Memcached 클라이언트와만 호환됩니다.

주요 이점

- 용량 계획 없음: ElastiCache 서버리스를 사용하면 용량을 계획할 필요가 없습니다. ElastiCache 서버리스는 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.
- Pay-per-use: ElastiCache 서버리스를 사용하면 캐시에 저장된 데이터와 워크로드에서 사용한 컴퓨팅에 대한 비용을 지불하면 됩니다. 차원 사용 섹션을 참조하십시오.
- 고가용성: ElastiCache 서버리스는 고가용성을 위해 여러 가용 영역 (AZ) 에 데이터를 자동으로 복제합니다. 기본 캐시 노드를 자동으로 모니터링하고 장애 발생 시 이를 대체합니다. 여기서는 모든 캐시에 대해 99.99%의 가용성 SLA를 제공합니다.
- 자동 소프트웨어 업그레이드: ElastiCache 서버리스는 애플리케이션의 가용성에 영향을 주지 않고 캐시를 최신 마이너 및 패치 소프트웨어 버전으로 자동 업그레이드합니다. 새 Memcached 메이저 버전이 ElastiCache 출시되면 알림을 보내드립니다.
- 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 서버리스의 작동 방식을 보여줍니다.



새 서버리스 캐시를 생성할 때 VPC에서 선택한 서브넷에 가상 사설 클라우드 (VPC) 엔드포인트를 ElastiCache 생성합니다. 애플리케이션은 이러한 VPC 엔드포인트를 통해 캐시에 연결할 수 있습니다.

ElastiCache 서버리스를 사용하면 애플리케이션이 연결되는 단일 DNS 엔드포인트를 받게 됩니다. 엔드포인트에 대한 새 연결을 요청하면 ElastiCache 서버리스는 프록시 레이어를 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. 이를 통해 ElastiCache 애플리케이션의 가용성에 영향을 미치거나 연결을 재설정하지 않고도 지능적으로 로드 밸런싱을 수행하고, 새 캐시 노드를 확장 및 추가하고, 장애 발생 시 캐시 노드를 교체하고, 캐시 노드의 소프트웨어를 업데이트할 수 있습니다.

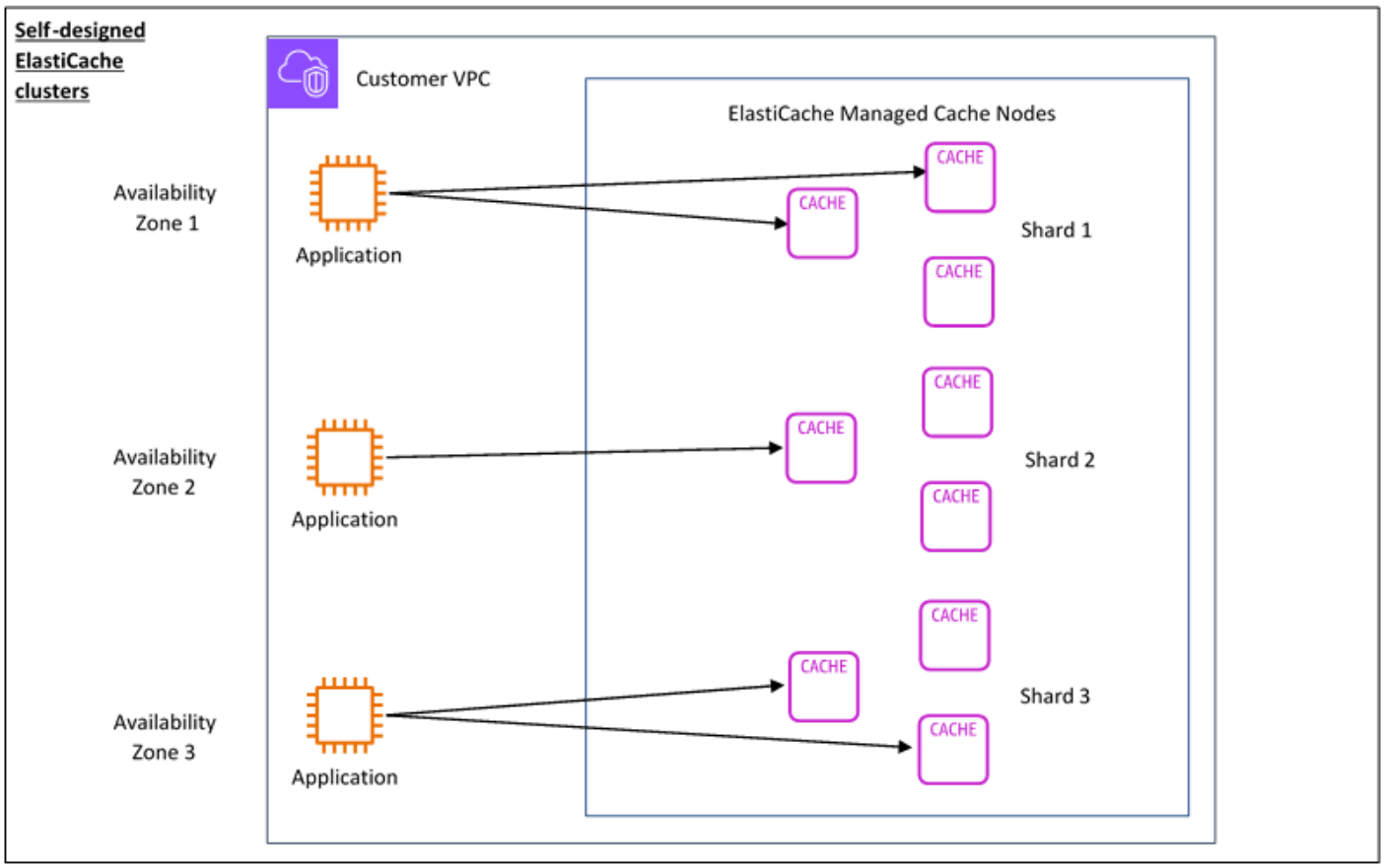
자체 설계된 클러스터 ElastiCache

ElastiCache 클러스터의 캐시 노드 패밀리, 크기 및 노드 수를 선택하여 자체 클러스터를 설계할 수 있습니다. 자체 클러스터를 설계하면 클러스터의 구성 및 규모 조정을 보다 세밀하게 제어할 수 있습니다.

주요 이점

- 자체 클러스터 설계: 를 사용하면 자체 클러스터를 설계하고 캐시 노드를 배치할 위치를 선택할 수 있습니다. ElastiCache 예를 들어 고가용성을 낮은 지연 시간과 절충하려는 애플리케이션을 보유하고 있는 경우 단일 AZ에 캐시 노드를 배포하도록 선택할 수 있습니다. 또는 다중 AZ에 노드가 있는 클러스터를 설계하여 고가용성을 달성할 수도 있습니다.
- 세밀한 제어: 자체 클러스터를 설계할 때 캐시의 설정을 더 세밀하게 조정할 수 있습니다. 예를 들어 파라미터 그룹을 사용해 엔진 파라미터 구성를 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. 노드를 추가하여 수평적으로 규모를 조정할 수도 있습니다.

다음 다이어그램은 ElastiCache 자체 설계된 클러스터의 작동 방식을 보여줍니다.



차원 사용

두 가지 배포 옵션으로 ElastiCache 배포할 수 있습니다. ElastiCache 서버리스를 배포할 때는 GB-시간 단위로 저장된 데이터에 대한 사용량과 ElastiCache 처리 장치 (ECPU) 에서 컴퓨팅에 대한 사용량

을 지불합니다. Memcached 클러스터를 자체 ElastiCache 설계하기로 선택한 경우 캐시 노드 사용량을 시간당 지불합니다. 요금 관련 세부 사항은 여기를 참조하세요.

데이터 스토리지

기가바이트-시간 (GB-시간) 단위로 청구되는 ElastiCache 서버리스에 저장된 데이터에 대한 요금을 지불합니다. ElastiCache 서버리스는 캐시에 저장된 데이터를 지속적으로 모니터링하여 분당 여러 번 샘플링하고 시간당 평균을 계산하여 캐시의 데이터 스토리지 사용량을 GB-시간 단위로 결정합니다. 각 ElastiCache 서버리스 캐시는 저장된 최소 1GB의 데이터에 대해 측정됩니다.

ElastiCache 처리 장치 (ECPU)

vCPU 시간과 전송된 데이터를 모두 포함하는 단위인 ElastiCache 프로세싱 유닛 (ECPU) 에서 애플리케이션이 ElastiCache 서버리스에서 실행하는 요청에 대한 비용을 지불합니다.

- 단순 읽기 및 쓰기에는 전송되는 데이터 1킬로바이트(KB)당 ECPU 1개가 필요합니다. 예를 들어 최대 1KB의 데이터를 전송하는 GET 명령은 1개의 ECPU를 사용합니다. 3.2KB의 데이터를 전송하는 SET 요청은 3.2ECPU를 사용합니다.
- 여러 항목에서 작동하는 명령은 그에 비례하여 더 많은 ECPU를 사용합니다. 예를 들어 애플리케이션이 항목 3개에 대해 멀티젯을 수행하는 경우 3개의 ECPU를 사용하게 됩니다.
- 더 많은 항목에서 작동하고 더 많은 데이터를 전송하는 명령은 두 차원 중 더 높은 차원을 기준으로 ECPU를 사용합니다. 예를 들어 애플리케이션이 GET 명령을 사용하여 항목 3개를 검색하고 3.2KB의 데이터를 전송하는 경우 3.2ECPU를 사용하게 됩니다. 또는 2KB의 데이터만 전송하는 경우 3개의 ECPU를 사용하게 됩니다.

ElastiCache 서버리스는 워크로드에 소비되는 eCPU를 이해하는 데 도움이 `ElastiCacheProcessingUnits` 되는 새로운 메트릭을 생성합니다.

노드 시간

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

일반적인 ElastiCache 사용 사례 및 ElastiCache 활용 방법

최신 뉴스 또는 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 쿼리나 원격 API 호출이라 할지라도 인 메모리 캐시에서 플랫 키를 검색하는 것보다 현저히 느려집니다. 현저히 느려지면 고객이 다른 곳으로 떠나는 경향이 있습니다.

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

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 수행하는 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 들

어, 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 다음 자료를 참조하세요.

- ElastiCache for Memcached 사용 설명서의 캐싱 전략

ElastiCache 고객 추천사

Airbnb, PBS, Esri 등의 기업이 어떻게 Amazon ElastiCache를 활용해 고객 환경을 개선하여 비즈니스 성장을 도모하고 있는지에 관해서는 Amazon ElastiCache 추천사를 참조하세요.

다른 ElastiCache 고객 사용 사례는 튜토리얼 동영상에서 보실 수 있습니다.

ElastiCache for Memcached 리소스

먼저 다음의 섹션을 읽은 다음, 필요에 따라 참조하는 것이 좋습니다.

- 서비스 하이라이트 및 요금 - 제품 세부 정보 페이지는 일반적인 ElastiCache 제품 개요, 서비스 하이라이트 및 요금 정보를 제공합니다.
- ElastiCache 비디오 - ElastiCache 동영상 섹션에는 Amazon ElastiCache for Memcached를 소개하고, 일반 사용 사례를 다루며, ElastiCache for Memcached를 사용하여 지연 시간을 줄이고 애플리케이션의 처리량을 향상하는 방법을 설명하는 비디오가 포함되어 있습니다.
- 시작하기 - Amazon ElastiCache for Memcached 시작하기 섹션에는 캐시 클러스터 생성, 캐시 클러스터에 대한 액세스 인증, 캐시 노드 연결 및 캐시 클러스터 삭제 과정을 안내하는 예제가 포함되어 있습니다.
- 규모에 따른 성능 - Amazon ElastiCache의 규모에 따른 성능 백서에서는 규모에 따라 애플리케이션이 원활하게 수행할 수 있는 캐싱 전략에 대해 설명합니다.

AWS Command Line Interface (AWS CLI)를 사용할 경우 시작하는 데 도움이 되도록 다음 문서를 사용할 수 있습니다.

- AWS Command Line Interface 설명서

  이 섹션에서는 AWS CLI 다운로드, 시스템에서 작동 중인 AWS CLI 가져오기 및 AWS 자격 증명 제공에 대한 정보를 제공합니다.
- ElastiCache용 AWS CLI 문서

  이 별도의 문서는 구문 및 예제를 포함한 모든 ElastiCache용 AWS CLI 명령을 다룹니다.

ElastiCache API를 널리 사용되는 다양한 프로그래밍 언어로 사용하도록 애플리케이션 프로그램을 작성할 수 있습니다. 다음과 같은 몇 가지 리소스가 있습니다.

- Amazon Web Services용 도구

  Amazon Web Services는 ElastiCache for Memcached에 대한 지원과 더불어 다양한 SDK(소프트웨어 개발 키트)를 제공합니다. Java, .NET, PHP, Ruby 및 기타 언어를 사용하여 ElastiCache에 대해 코딩할 수 있습니다. 이러한 SDK는 ElastiCache에 대한 요청을 포맷하고 응답을 분석하며 재시도 논리 및 오류 처리를 제공함으로써 애플리케이션 개발을 상당히 간소화할 수 있습니다.
- ElastiCache API 사용

AWS SDK를 사용하지 않으려는 경우 Query API를 직접 사용하면서 ElastiCache와 상호 작용할 수 있습니다. 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 포함한 모든 ElastiCache API 작업을 다룹니다.

구현 관리용 도구

ElastiCache 클러스터에 대한 Amazon EC2 인스턴스 액세스를 허용하면 ElastiCache 클러스터를 관리할 수 있는 네 가지 수단(AWS Management Console, AWS CLI for ElastiCache, AWS SDK for ElastiCache 및 ElastiCache API)이 제공됩니다.

AWS Management Console 사용

가장 간편한 Amazon ElastiCache for Memcached 관리 방법은 AWS Management Console을 사용하는 것입니다. 콘솔을 사용하면 캐시 클러스터를 생성하고 캐시 노드를 추가 및 제거하며 코드를 작성하지 않고도 다른 관리 작업을 수행할 수 있습니다. 콘솔은 캐시 엔진 활동, 메모리 및 CPU 사용률과 기타 지표를 보여주는 CloudWatch의 캐시 노드 성능 그래프도 제공합니다. 자세한 내용은 이 사용 설명서의 특정 항목을 참조하세요.

AWS CLI 사용

AWS Command Line Interface(AWS CLI) for ElastiCache를 사용해도 됩니다. AWS CLI를 사용하면 캐시 클러스터 시작 또는 중지와 같은 한 번에 한 작업을 쉽게 수행할 수 있습니다. 선택한 스크립팅 언어에서 ElastiCache용 AWS CLI 명령을 호출하여 반복 작업을 자동화할 수도 있습니다. AWS CLI에 대한 자세한 내용은 사용 설명서 및 AWS CLI 명령 참조를 참조하세요.

AWS SDK 사용

애플리케이션에서 ElastiCache에 액세스하려면 AWS SDK(소프트웨어 개발 키트) 중 하나를 사용할 수 있습니다. SDK는 ElastiCache API 호출을 래핑하고 ElastiCache API의 하위 수준 세부 정보로부터 애플리케이션을 보호합니다. 사용자는 자격 증명을 제공하고 SDK 라이브러리가 인증 및 요청 서명을 담당합니다. AWS SDK 사용에 대한 자세한 내용은 Amazon Web Services용 도구를 참조하십시오.

ElastiCache API 사용

또한 ElastiCache 웹 서비스 API에 대해 직접 애플리케이션 코드를 작성할 수 있습니다. API를 사용할 때 HTTP 요청을 구성하고 인증하고 ElastiCache의 결과를 분석하고 오류를 처리하기 위해 필요한 코드를 작성해야 합니다. API에 대한 자세한 내용은 을 참조하세요ElastiCache API 사용

다음 사항도 참조하십시오.

Amazon ElastiCache for Memcached 배포 관리에 대한 자세한 내용은 다음을 참조하세요.

- 함께 일하기 ElastiCache

- 인터네트워크 트래픽 개인 정보
- Amazon ElastiCache에서 로깅 및 모니터링

배포 옵션 간 선택

Amazon ElastiCache에는 다음과 같은 2가지 배포 옵션이 있습니다.

- 서버리스 캐시
- 자체 클러스터 설계

서버리스 캐시

Amazon ElastiCache 서버리스는 캐시 생성을 간소화하고 고객의 가장 까다로운 애플리케이션을 지원하도록 즉시 규모를 조정할 수 있습니다. ElastiCache 서버리스를 사용하면 캐시 클러스터 용량을 프로비저닝, 계획 및 관리할 필요 없이 1분 이내에 가용성과 확장성이 뛰어난 캐시를 생성할 수 있습니다. ElastiCache 서버리스는 3개의 가용 영역에 데이터를 자동으로 중복 저장하고 99.99%의 가용성 서비스 수준에 관한 계약(SLA)을 제공합니다. ElastiCache는 여러 AZ에서 자동 데이터 복제 기능을 제공하므로 복제본을 수동으로 관리할 필요가 없으며, 복제본을 동기화된 상태로 유지하기 위한 사용자 지정 소프트웨어를 수동으로 관리할 필요가 없습니다.

자체 설계된 클러스터

ElastiCache for Memcached 클러스터를 세밀하게 제어해야 하는 경우 ElastiCache를 사용하여 자체 Memcached 클러스터를 설계할 수 있습니다. ElastiCache를 사용하면 클러스터의 AWS 가용 영역 전반에서 노드 유형, 노드 수, 노드 배치를 선택하여 노드 기반 클러스터를 운영할 수 있습니다. ElastiCache는 완전관리형 서비스이므로 클러스터의 하드웨어 프로비저닝, 모니터링, 노드 교체 및 소프트웨어 패치를 자동으로 관리합니다.

배포 옵션 간 선택

다음과 같은 경우 서버리스 캐싱을 선택합니다.

- 새 워크로드 또는 알려지지 않은 워크로드용 새 캐시를 생성하는 경우
- 애플리케이션 트래픽이 예측 불가능한 경우
- 캐시를 가장 쉽게 시작하는 방법을 찾고 있는 경우

다음과 같은 경우 자체 ElastiCache 클러스터 설계를 선택합니다.

- 이미 ElastiCache 서버리스를 실행하고 있으며 Memcached를 실행하는 노드 유형, 노드 수, 노드 배치를 보다 세밀하게 제어하고자 하는 경우
- 애플리케이션 트래픽이 크게 변동할 것으로 예상하지 않거나 애플리케이션 트래픽 최고점 및 최저점을 정확하게 예측할 수 있는 경우
- 비용 제어를 위해 용량 요구 사항을 예측할 수 있는 경우

관련 항목:

- Memcached 구현용 자체 ElastiCache 클러스터 설계 및 관리

Memcached 캐시와 Redis 자체 설계된 캐시 비교

ElastiCache Amazon은 멤캐시 및 레디 스 캐시 엔진을 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 조건에 해당하는 경우 ElastiCache Redis용 버전이 포함된 Redis를 선택하십시오.

- ElastiCache Redis 버전 7.0의 경우 (고급)

Redis 함수, 샤딩된 Pub/Sub 또는 Redis ACL 개선 사항을 사용하고 싶습니다. 자세한 내용은 Redis 버전 7.0(향상된 버전)을 참조하세요.

- ElastiCache 레디스 버전 6.2용 (고급)

  r6gd 노드 유형을 사용하여 메모리와 SSD 간에 데이터를 계층화할 수 있어야 합니다. 자세한 내용은 데이터 암호화를 참조하세요.

- ElastiCache 레디스 버전 6.0용 (고급)

  역할 기반 액세스 제어로 사용자를 인증하려는 경우

  자세한 내용은 Redis 버전 6.0(향상된 버전)을 참조하세요.

- ElastiCache 레디스 버전 5.0.0의 경우 (고급)

  생산자가 실시간으로 새 항목을 추가하고 소비자가 차단 또는 비 차단 방식으로 메시지를 사용할 수 있도록 지원하는 로그 데이터 구조인 Redis 스트림을 사용할 수 있습니다.

  자세한 내용은 Redis 버전 5.0.0(확장)을 참조하세요.

- ElastiCache 레디스 버전 4.0.10의 경우 (고급)

  암호화 및 Redis(클러스터 모드 활성화됨) 클러스터에서 샤드의 동적인 추가 또는 제거를 지원합니다.

  자세한 내용은 Redis 버전 4.0.10(확장)을 참조하세요.

다음 버전은 더 이상 사용되지 않거나 수명이 다했거나 곧 종료될 예정입니다.

- ElastiCache 레디스 버전 3.2.10의 경우 (고급)

  Redis(클러스터 모드 활성화됨) 클러스터에서 샤드를 동적으로 추가 또는 제거하는 기능을 지원합니다.

  > **Important**
  >
  > 현재 ElastiCache Redis 3.2.10은 암호화를 지원하지 않습니다.

  자세한 내용은 다음을 참조하십시오.

  - Redis 버전 3.2.10(확장)

  - Redis에 대한 온라인 리샤딩 모범 사례에 대한 자세한 내용은 다음 자료를 참조하세요.
    - 모범 사례: 온라인 리샤딩
    - Redis(클러스터 모드 활성화됨)를 위한 온라인 리샤딩 및 샤드 재분배
  - Redis 클러스터 조정에 대한 자세한 내용은 조정 섹션을 참조하세요.

- ElastiCache 레디스 버전 3.2.6의 경우 (고급)

  이전 Redis 버전의 기능과 다음 기능이 필요한 경우 Redis 3.2.6을 선택하십시오 ElastiCache .
  - 전송 중 데이터 암호화. 자세한 내용은 ElastiCache Amazon용 Redis 전송 중 암호화를 참조하십시오.
  - 미사용 데이터 암호화. 자세한 ElastiCache 내용은 Amazon의 Redis 저장 중 암호화를 참조하십시오.
- ElastiCache Redis용 (클러스터 모드 활성화) 버전 3.2.4

  다음 기능 이외에 Redis 2.8.x의 기능이 필요하면 Redis 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 Redis의 경우 (비클러스터형 모드) 2.8.x 및 3.2.4 (고급)

  다음과 같은 경우 Redis 2.8.x 또는 Redis 3.2.4(비클러스터 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, Redis(클러스터 모드 비활성화됨) 및 Redis(클러스터 모드 활성화됨) 비교 요약

	Memcached	Redis(클러스터 모드 비활성화됨)	Redis(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)	운송 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			
FedRAMP	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
게시/구독 기능	아니요	예	예

	Memcached	Redis(클러스터 모드 비활성화됨)	Redis(클러스터 모드 활성화됨)
정렬된 세트	아니요	예	예
백업 및 복원	서버리스 Memcached에만 해당되며, 자체 설계된 Memcached 클러스터에는 해당되지 않습니다.	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 예정인 버전은 제외됩니다.

클러스터에 대한 엔진을 선택한 후 해당 엔진의 최신 버전을 사용하는 것이 좋습니다. 자세한 ElastiCache 내용은 Memcached 버전 지원 또는 Redis 버전 지원을 참조하십시오. ElastiCache

Amazon ElastiCache for Memcached 시작하기

이 섹션의 여러 주제에서는 ElastiCache 콘솔을 사용하여 Memcached 서버리스 캐시를 생성하는 것부터 액세스 권한을 부여하고 연결하며 마지막으로 삭제하는 것까지 일련의 과정을 설명합니다.

주제

- 설정
- 1단계: 캐시 생성
- 2단계: 캐시에 데이터 읽기 및 쓰기
- 3단계: (선택 사항) 정리
- 다음 단계
- 자습서: Amazon VPC에서 Amazon ElastiCache에 액세스하도록 Lambda 함수 구성
- ElastiCache 자습서 및 동영상

설정

설정하려면 ElastiCache:

주제

- 가입하세요 AWS 계정
- 관리자 액세스 권한이 있는 사용자 생성
- 프로그래밍 방식 액세스 권한 부여
- 권한 설정 (신규 ElastiCache 사용자만 해당)
- EC2 설정
- Amazon VPC 보안 그룹에서 캐시에 대한 네트워크 액세스 권한 부여

가입하세요 AWS 계정

계정이 없는 경우 다음 단계를 완료하여 계정을 만드세요. AWS 계정

가입하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 여세요.

2. 온라인 지시 사항을 따르세요.

   등록 절차 중에는 전화를 받고 키패드로 인증 코드를 입력하는 과정이 있습니다.

   에 AWS 계정가입하면 AWS 계정 루트 사용자a가 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스 액세스 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS 가입 절차가 완료된 후 확인 이메일을 보냅니다. 언제든지 https://aws.amazon.com/으로 가서 내 계정(My Account)을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

등록한 AWS 계정후에는 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 보호하고 AWS IAM Identity Center활성화하고 생성하십시오 AWS 계정 루트 사용자.

보안을 유지하세요. AWS 계정 루트 사용자

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

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

   지침은 IAM 사용 설명서의 AWS 계정 루트 사용자 (콘솔) 에 대한 가상 MFA 디바이스 활성화를 참조하십시오.

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 ID 소스로 사용하는 방법에 대한 자습서는 사용 설명서의 기본값으로 IAM Identity Center 디렉터리사용자 액세스 구성을 참조하십시오. IAM Identity Center 디렉터리 AWS IAM Identity Center

관리 액세스 권한이 있는 사용자로 로그인

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

  IAM Identity Center 사용자를 사용하여 로그인하는 데 도움이 필요하면 사용 설명서의 AWS 액세스 포털에 로그인을 참조하십시오.AWS 로그인

추가 사용자에게 액세스 권한 할당

1. IAM Identity Center에서 최소 권한 적용 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

사용자가 AWS 외부 사용자와 상호 작용하려는 경우 프로그래밍 방식의 액세스가 필요합니다. AWS Management Console프로그래밍 방식의 액세스 권한을 부여하는 방법은 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 (IAM Identity Center가 관리하는 사용자)	임시 자격 증명을 사용하여 AWS CLI, AWS SDK 또는 API에 대한 프로그래밍 요청에 서명할 수 있습니다. AWS	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • AWS CLI 에 대한 내용은 사용 설명서의 AWS CLI사용을 AWS IAM Identity Center위한 구성을 참조하십시오.AWS Command Line Interface • AWS SDK, 도구 및 AWS API의 경우 AWS SDK 및 도

프로그래밍 방식 액세스가 필요한 사용자는 누구인가요?	To	액세스 권한을 부여하는 사용자
		구 참조 안내서의 IAM ID 센터 인증을 참조하십시오.
IAM	임시 자격 증명을 사용하여 AWS CLI, AWS SDK 또는 API에 대한 프로그래밍 방식 요청에 서명할 수 있습니다. AWS	IAM 사용 설명서의 AWS 리소스와 함께 임시 자격 증명 사용의 지침을 따르십시오.
IAM	(권장되지 않음) 장기 자격 증명을 사용하여 AWS CLI, AWS SDK 또는 API에 대한 프로그래밍 요청에 서명할 수 있습니다. AWS	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • 에 대한 내용은 사용 설명서의 IAM 사용자 자격 증명을 사용한 인증을 참조하십시오. AWS CLIAWS Command Line Interface • AWS SDK 및 도구의 경우 SDK 및 도구 참조 안내서의 장기 자격 증명을 사용한 인증을 참조하십시오.AWS • AWS API의 경우 IAM 사용 설명서의 IAM 사용자의 액세스 키 관리를 참조하십시오.

관련 항목:

- IAM 사용 설명서의 IAM이란 무엇입니까?
- AWS 일반 참조의 보안 자격 증명.AWS

권한 설정 (신규 ElastiCache 사용자만 해당)

액세스 권한을 제공하려면 사용자, 그룹 또는 역할에 권한을 추가하세요:

- AWS IAM Identity Center다음 지역의 사용자 및 그룹:

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

- ID 제공자를 통해 IAM에서 관리되는 사용자:

  ID 페더레이션을 위한 역할을 생성합니다. IAM 사용 설명서의 서드 파티 자격 증명 공급자의 역할 만들기(연합)의 지침을 따르세요.
- IAM 사용자:
  - 사용자가 맡을 수 있는 역할을 생성합니다. IAM 사용 설명서에서 IAM 사용자의 역할 생성의 지침을 따르세요.
  - (권장되지 않음)정책을 사용자에게 직접 연결하거나 사용자를 사용자 그룹에 추가합니다. IAM 사용 설명서에서 사용자(콘솔)에 권한 추가의 지침을 따르세요.

Amazon은 사용자를 대신하여 리소스를 프로비저닝하고 다른 AWS 리소스 및 서비스에 액세스할 수 있도록 서비스 연결 역할을 ElastiCache 생성하고 사용합니다. 서비스 연결 역할을 ElastiCache 생성하려면 이름이 지정된 -managed 정책을 사용하십시오. AWS`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 수 있는 권한이 `iam:createServiceLinkedRole` 있거나 서비스 연결 역할을 생성했는지 확인하세요. ElastiCache

자세한 내용은 다음을 참조하십시오.

- 새 정책 생성(IAM)
- AWS 관리형 정책(Amazon ElastiCache용)
- Amazon ElastiCache에 대해 서비스 연결 역할 사용

EC2 설정

캐시에 연결할 EC2 인스턴스를 설정해야 합니다.

- 아직 EC2 인스턴스가 없는 경우, EC2 시작하기에서 EC2 인스턴스 설정 방법을 알아보세요.
- EC2 인스턴스는 동일한 VPC에 있어야 하며 캐시와 동일한 보안 그룹 설정이어야 합니다. 기본적으로 ElastiCache Amazon은 기본 VPC에 캐시를 생성하고 기본 보안 그룹을 사용합니다. 이 자습서를 따르려면 EC2 인스턴스가 기본 VPC에 있고 기본 보안 그룹이 있는지 확인합니다.

Amazon VPC 보안 그룹에서 캐시에 대한 네트워크 액세스 권한 부여

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

1. AWS Command Line Interface 로그인하고 Amazon EC2 콘솔을 엽니다.
2. 탐색 창의 [Network & Security] 아래에서 [Security Groups]를 선택합니다.
3. 보안 그룹 목록에서 Amazon VPC를 위한 보안 그룹을 선택합니다. ElastiCache 사용할 보안 그룹을 생성하지 않은 경우 이 보안 그룹의 이름은 default로 지정됩니다.
4. 인바운드 탭을 선택한 후 다음과 같이 하세요.

    a. 편집을 선택합니다.
    b. 규칙 추가를 선택합니다.
    c. 유형 열에서 사용자 지정 TCP 규칙을 선택합니다.
    d. 포트 범위 상자에 11211을 입력합니다.
    e. 소스 상자에서 포트 범위(0.0.0.0/0)를 가진 위치 무관을 선택하면 Amazon VPC 내에서 시작한 Amazon EC2 인스턴스를 캐시에 연결할 수 있습니다.
    f. ElastiCache 서버리스를 사용하는 경우 규칙 추가를 선택하여 다른 규칙을 추가하십시오.
    g. [Type] 열에서 [Custom TCP rule]을 선택합니다.
    h. 포트 범위 상자에 11212를 입력합니다.
    i. 소스 상자에서 포트 범위(0.0.0.0/0)를 가진 위치 무관을 선택하면 Amazon VPC 내에서 시작한 Amazon EC2 인스턴스를 캐시에 연결할 수 있습니다.
    j. 저장을 선택합니다.

1단계: 캐시 생성

시작하려는 캐시는 샌드박스에서 실행되지 않고 활성화됩니다. 삭제하기 전까지 캐시에 해당하는 표준 ElastiCache 사용 요금이 청구됩니다. 여기에 설명된 연습을 한 번에 끝내고 연습을 마칠 때 캐시를 삭제하면 총 청구 비용이 가장 적게 듭니다(일반적으로 1 USD 미만). ElastiCache 사용 요금에 대한 자세한 내용은 Amazon ElastiCache를 참조하세요.

서버리스 캐시 생성

AWS Management Console

ElastiCache 콘솔을 사용하여 새 캐시를 만들려면 다음과 같이 하세요.

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
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 '활성' 상태가 되면 캐시에 데이터를 쓰고 읽을 수 있습니다.

AWS CLI를 사용하여 새 캐시를 생성하려면 다음과 같이 하세요.

다음 AWS CLI 예제에서는 create-serverless-cache를 사용하여 새 캐시를 만듭니다.

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.

ElastiCache에서 캐시 생성이 완료되었는지 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 2단계: 캐시에 데이터 읽기 및 쓰기 섹션으로 이동합니다.

2단계: 캐시에 데이터 읽기 및 쓰기

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

기본적으로 기본 VPC에 캐시를 ElastiCache 생성합니다. 기본 VPC에서도 EC2 인스턴스를 생성해야 캐시에 연결할 수 있습니다.

구성

시작하기 전에 액세스에 사용할 수 있는 올바른 포트가 있는지 확인하세요.

기본 포트: 11211

읽기 최적화된 포트: 11212

서버리스 Memcached 캐시는 동일한 호스트 이름을 가진 두 개의 포트를 광고합니다. 기본 포트는 OSS Memcached와 동일한 일관성을 보장하면서 쓰기 및 읽기를 허용합니다. 읽기 최적화 포트를 통해 쓰기가 가능하고, 지연 시간이 짧고 최종적으로 일관성이 유지되는 읽기가 가능합니다.

캐시 엔드포인트 확인

AWS Management Console

콘솔을 사용하여 캐시의 엔드포인트를 찾으려면: ElastiCache

1. https://console.aws.amazon.com/elasticache/ 에서 AWS Management Console 로그인하고 아마존 ElastiCache 콘솔을 엽니다.
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 describe-serverless-caches 명령을 사용하여 새 캐시의 엔드포인트를 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
```

OpenSSL을 사용하여 연결

OpenSSL을 사용하여 연결하는 방법에 대한 내용은 ElastiCache 전송 중 암호화 (TLS) 섹션을 참조하세요.

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 참조

```
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 참조

`npm i memcache-client`를 통해 설치합니다.

애플리케이션에서 다음과 같이 Memcache 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-memcache 참조

```
// 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 참조

```
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/dalli 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 클라이언트를 사용하여 연결 () EnyimMemcachedCore

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

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

이제 3단계: (선택 사항) 정리 섹션으로 이동합니다.

3단계: (선택 사항) 정리

AWS Management Console 사용

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. AWS Management Console에 로그인하고 https://console.aws.amazon.com/elasticache/에서 Amazon ElastiCache 콘솔을 엽니다.
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 콘솔에서는 캐시를 한 번에 하나씩만 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 삭제 중 상태로 전환되면 그 즉시 요금 발생이 중지됩니다.

AWS CLI 사용

다음 코드는 my-cache 캐시를 삭제합니다.

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

delete-serverless-cache CLI 작업은 서버리스 캐시 1개만 삭제합니다. 여러 캐시를 삭제하려면 삭제할 서버리스 캐시별로 delete-serverless-cache를 직접 호출합니다. 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 delete-serverless-cache에 대한 AWS CLI 섹션을 참조하세요.

이제 다음 단계 섹션으로 이동합니다.

다음 단계

ElastiCache에 관한 자세한 내용은 다음 섹션을 참조하세요.

- 함께 일하기 ElastiCache
- ElastiCache Memcached를 위한 스케일링
- ElastiCache에 대한 할당량
- ElastiCache 모범 사례 및 캐싱 전략
- ElastiCache 이벤트 보기

자습서: Amazon VPC에서 Amazon ElastiCache에 액세스하도록 Lambda 함수 구성

이 자습서에서는 다음 작업을 수행합니다.

- us-east-1 리전 내 기본 Amazon Virtual Private Cloud(VPC)에서 Amazon ElastiCache 캐시를 생성합니다.
- ElastiCache 캐시에 액세스하기 위한 Lambda 함수를 생성합니다. Lambda 함수를 생성할 때 Amazon VPC 및 VPC 보안 그룹에 서브넷 ID를 제공하여 Lambda 함수가 VPC의 리소스에 액세스할 수 있도록 합니다. 이 자습서의 설명에서는 Lambda 함수가 UUID를 생성하여 이를 캐시에 기록한 다음, 캐시에서 이를 검색합니다.
- Lambda 함수를 수동으로 간접 호출하고 VPC의 ElastiCache 캐시에 액세스했는지 확인합니다.

> **Important**
>
> 이 자습서에서는 us-east-1 리전 내 계정에서 기본 Amazon VPC를 사용합니다. Amazon VPC에 대한 자세한 내용은 Amazon VPC를 시작하는 방법 및 Amazon VPC 사용 설명서를 참조하세요.

주제

- 1단계: ElastiCache 캐시 생성
- 2단계: Lambda 함수 생성
- 3단계: Lambda 함수 테스트

시작하기

1단계: ElastiCache 캐시 생성

1단계: ElastiCache 캐시 생성

이 단계에서는 AWS CLI를 사용하여 us-east-1 리전 내 계정에서 기본 Amazon Virtual Private Cloud 내에 Amazon ElastiCache 캐시를 생성합니다. ElastiCache 콘솔 또는 API를 사용하여 ElastiCache 서버리스 캐시를 생성하는 방법에 대한 자세한 내용은 ElastiCache for Memcached 사용 설명서의 클러스터 생성 섹션을 참조하세요.

AWS Management Console

다음 AWS CLI CLI 명령을 실행하여 us-east-1 리전 내 기본 VPC에서 Memcached 클러스터 서버리스 캐시를 새로 생성합니다.

Linux

```
aws elasticache create-serverless-cache \
--serverless-cache-name serverlessCacheForLambda \
--region us-east-1 \
--engine memcached
```

Windows

```
aws elasticache create-serverless-cache ^
--serverless-cache-name serverlessCacheForLambda ^
--region us-east-1 ^
--engine memcached
```

상태 필드 값은 CREATING으로 설정됩니다. ElastiCache가 클러스터 생성을 완료하기까지 몇 분 정도 걸릴 수 있습니다.

ElastiCache에서 캐시 생성이 완료되었는지 확인하려면 describe-serverless-caches 명령을 사용합니다.

Linux

```
aws elasticache describe-serverless-caches \
--serverless-cache-name serverlessCacheforLambda \
--region us-east-1
```

Windows

```
aws elasticache describe-serverless-caches ^
--serverless-cache-name serverlessCacheforLambda ^
--region us-east-1
```

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

새 캐시를 생성한 후에는 2단계: Lambda 함수 생성 섹션으로 이동합니다.

다음 단계:

2단계: Lambda 함수 생성

2단계: Lambda 함수 생성

이 단계에서는 다음 작업을 수행합니다.

1. 제공된 샘플 코드를 사용하여 Lambda 함수 배포 패키지를 생성합니다.
2. IAM 역할(실행 역할)을 생성합니다. 배포 패키지를 업로드할 때 이 역할을 지정해야 Lambda가 역할을 수임하고 사용자 대신 함수를 실행할 수 있습니다. 권한 정책이 AWS Lambda에 탄력적 네트워크 인터페이스, 즉, ENI를 설정할 수 있는 권한을 부여하여 Lambda 함수가 VPC 내 리소스에 액세스할 수 있습니다. 이 예제에서는 Lambda 함수가 VPC 내 ElastiCache 클러스터에 액세스합니다.
3. 배포 패키지를 업로드하여 Lambda 함수를 생성합니다.

다음 단계

2.1단계: 배포 패키지 생성

2.1단계: 배포 패키지 생성

현재 Lambda 함수의 예제 코드는 Python에서만 제공됩니다.

Python

다음의 예제 Python 코드는 항목을 읽어서 ElastiCache 클러스터에 기록합니다. 코드를 복사한 후 `app.py` 파일에 저장합니다. 코드의 `elasticache_config_endpoint` 값을 1단계에서 복사한 엔드포인트 주소로 바꿔야 합니다.

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

elasticache_config_endpoint = "serverlesscacheforlambda-
ces85m.serverless.use1.cache.amazonaws.com"
target_port = 11211

context = ssl.create_default_context()
```

```
memcached_client = Client((elasticache_config_endpoint, target_port),
 tls_context=context)

def lambda_handler(event, context):

    # create a random UUID - this will be the sample element we add to the cache
    uuid_in = uuid.uuid4().hex

    # put the UUID to the cache
    memcached_client.set("uuid", uuid_in, expire=500, noreply=False)

    # get the item (UUID) from the cache
    result = memcached_client.get("uuid")
    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 Memcached.")
    else:
        raise Exception(f"Bad value retrieved. Expected {uuid_in}, got
 {decoded_result}")

    return "Fetched value from Memcached"
```

이 코드는 Python pymemcache 라이브러리를 사용하여 항목을 캐시에 넣고 검색합니다. pymemcache가 포함된 배포 패키지를 만들려면 다음 단계를 수행합니다.

1. app.py 소스 코드 파일이 포함된 프로젝트 디렉터리에서 package 폴더를 만들어 pymemacache 라이브러리를 설치합니다.

   ```
   mkdir package
   ```

2. pip를 사용하여 pymemcache를 설치합니다.

   ```
   pip install --target ./package pymemcache
   ```

3. pymemcache 라이브러리가 포함된 .zip 파일을 생성합니다. Linux 및 macOS에서 다음 명령을 실행합니다. Windows에서 선호하는 zip 유틸리티를 사용하여 루트에 pymemache 라이브러리가 포함된 .zip 파일을 생성합니다.

   ```
   cd package
   ```

```
zip -r ../my_deployment_package.zip .
```

4. 함수 코드를 .zip 파일에 추가합니다. Linux 및 macOS에서 다음 명령을 실행합니다. Windows에서 선호하는 zip 유틸리티를 사용하여 .zip 파일의 루트에 app.py를 추가합니다.

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

다음 단계

2.2단계: IAM 역할 생성(실행 역할)

2.2단계: IAM 역할 생성(실행 역할)

이 단계에서는 사전 정의된 다음 역할 유형 및 액세스 정책을 사용하여 AWS IAM(Identity and Access Management) 역할을 생성합니다.

- AWS Lambda 유형의 AWS 서비스 역할 - 이 역할은 AWS Lambda에 역할을 수임할 권한을 부여합니다.
- AWSLambdaVPCAccessExecutionRole - 이는 역할에 연결하는 액세스 권한 정책입니다. AWS Lambda가 ENI를 관리하는 데 필요한 EC2 작업에 대한 권한을 정책이 부여합니다. 이 AWS 관리형 정책은 IAM 콘솔에서 볼 수 있습니다.

IAM 사용자 역할에 대한 자세한 내용은 IAM 사용 설명서의 역할(위임 및 페더레이션)을 참조하세요.

다음 절차에 따라 IAM 역할을 생성합니다.

IAM(실행) 역할을 생성하려면

1. AWS 관리 콘솔에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 역할을 선택하고 역할 생성을 선택합니다.
   - 신뢰할 수 있는 엔터티 유형에서 AWS 서비스를 선택한 다음, 사용 사례 아래에서 Lambda를 선택합니다. 이렇게 하면 AWS Lambda 서비스에 역할 수임 권한이 부여됩니다. 다음을 선택합니다.
   - 권한 추가에서 **AWSLambdaVPCAccessExecutionRole**을 검색한 후 라이브러리 옆의 확인란을 선택합니다.
   - 다음을 선택합니다.

- 역할 이름에는 AWS 계정 내에서 고유한 이름을 사용해야 합니다(예: lambda-vpc-execution-role).
- 역할 생성을 선택합니다.

3. 역할 ARN을 복사합니다. Lambda 함수를 생성하는 다음 단계에서 필요합니다.

다음 단계

2.3단계: 배포 패키지 업로드(Lambda 함수 생성)

2.3단계: 배포 패키지 업로드(Lambda 함수 생성)

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

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

`role` 옵션의 경우 2.2단계에서 생성한 실행 역할의 ARN을 사용합니다. `vpc-config`에서 기본 VPC의 서브넷과 기본 VPC의 보안 그룹 ID를 쉼표로 구분한 목록을 입력합니다. 이러한 값은 Amazon VPC 콘솔에서 확인할 수 있습니다. 기본 VPC의 서브넷을 확인하려면 VPC를 선택한 다음 AWS 계정의 기본 VPC를 선택합니다. 이 VPC의 보안 그룹을 확인하려면 보안에서 보안 그룹을 선택합니다. us-east-1 리전이 선택되어 있어야 합니다.

Linux, macOS, Unix의 경우:

```
aws lambda create-function \
--function-name AccessMemcached  \
--region us-east-1 \
--zip-file fileb://my_deployment_package.zip \
--role arn:aws:iam::123456789012:role/lambda-vpc-execution-role \
--handler app.lambda_handler \
--runtime python3.11  \
--timeout 30 \
--vpc-config SubnetIds=comma-separated-vpc-subnet-ids,SecurityGroupIds=default-security-group-id \
```

Windows의 경우:

```
aws lambda create-function ^
--function-name AccessMemcached  ^
```

```
--region us-east-1 ^
--zip-file fileb://path-to/my_deployment_package.zip ^
--role arn:aws:iam::123456789012:role/lambda-vpc-execution-role ^
--handler app.lambda_handler ^
--runtime python3.11  ^
--timeout 30 ^
--vpc-config SubnetIds=comma-separated-vpc-subnet-ids,SecurityGroupIds=default-
security-group-id ^
```

.zip 파일을 동일한 AWS 리전의 Amazon S3 버킷에 업로드한 다음, 앞의 명령에서 버킷 및 객체 이름을 지정할 수도 있습니다. 다음과 같이, `--zip-file` 파라미터를 `--code` 파라미터로 교체해야 합니다.

```
--code S3Bucket=bucket-name,S3Key=zip-file-object-key
```

AWS Lambda 콘솔을 사용하여 Lambda 함수를 생성할 수도 있습니다. 함수를 생성할 때 Lambda의 VPC를 선택하고, 제공된 필드에서 서브넷과 보안 그룹을 선택합니다.

다음 단계

3단계: Lambda 함수 테스트

3단계: Lambda 함수 테스트

이 단계에서는 `invoke` 명령을 사용하여 Lambda 함수를 수동으로 호출합니다. Lambda 함수는 실행될 때 UUID를 생성하고 이를 Lambda 코드에 지정된 ElastiCache 클러스터에 기록합니다. 그런 다음 Lambda 함수는 캐시에서 해당 항목을 검색합니다.

1. AWS Lambda `invoke` 명령을 사용하여 Lambda 함수(`AccessMemCache`)를 호출합니다.

   Linux, macOS, Unix의 경우:

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

   Windows의 경우:

   ```
   aws lambda invoke ^
   --function-name AccessMemCache  ^
   ```

```
--region us-east-1 ^
output.txt
```

2. 다음과 같이 Lambda 함수가 성공적으로 실행되었는지 확인합니다.

   - output.txt 파일을 검토합니다.
   - CloudWatch 콘솔을 열고 함수의 로그 그룹(/aws/lambda/AccessMemcached)을 선택하여 CloudWatch Logs에서 결과를 확인합니다. 로그 스트림의 출력은 다음과 유사해야 합니다.

     ```
     Success: Inserted 05fcf2e4d6c942209acc89ea79b5b15e. Fetched
      05fcf2e4d6c942209acc89ea79b5b15e from Memcached.
     ```

   - AWS Lambda 콘솔에서 결과를 검토합니다.

ElastiCache 자습서 및 동영상

다음은 Amazon ElastiCache 사용자가 관심을 기울이는 작업을 해결하는 자습서입니다.

- ElastiCache 동영상
- 자습서: Amazon VPC에서 Amazon ElastiCache에 액세스하도록 Lambda 함수 구성

ElastiCache 동영상

다음으로, 기본 및 고급 Amazon ElastiCache 개념을 학습할 수 있는 동영상을 찾을 수 있습니다. AWS 교육에 대한 자세한 내용은 AWS Training & Certification을 참조하세요.

주제

- 입문자용 동영상
- 고급 동영상

입문자용 동영상

다음 동영상은 Amazon ElastiCache에 대해 소개합니다.

주제

- AWS re:Invent 2020: What's new in Amazon ElastiCache
- AWS re:Invent 2019: What's new in Amazon ElastiCache
- AWS re:Invent 2017: What's new in Amazon ElastiCache
- DAT204 - Building Scalable Applications on AWS NoSQL Services (re:Invent 2015)
- DAT207 - Accelerating Application Performance with Amazon ElastiCache (AWS re:Invent 2013)

AWS re:Invent 2020: What's new in Amazon ElastiCache

AWS re:Invent 2020: What's new in Amazon ElastiCache

AWS re:Invent 2019: What's new in Amazon ElastiCache

AWS re:Invent 2019: What's new in Amazon ElastiCache

AWS re:Invent 2017: What's new in Amazon ElastiCache

AWS re:Invent 2017: What's new in Amazon ElastiCache

DAT204 - Building Scalable Applications on AWS NoSQL Services (re:Invent 2015)

이 세션에서는 NoSQL 데이터베이스에 대한 이점을 설명하고, AWS에서 제공한 기본 NoSQL 서비스(Amazon DynamoDB 및 Amazon ElastiCache)에 대해 알아보겠습니다. 그런 다음, 두 선도적인 고객인 Expedia와 Mapbox로부터 사용 사례와 아키텍처 문제점 및 설계 패턴과 모범 사례를 비롯한 AWS NoSQL 서비스를 사용하여 이를 해결한 방법에 대해 듣습니다. 이 세션을 통해 NoSQL과 NoSQL의 강력한 기능을 잘 이해하여 데이터베이스 문제점을 확실하게 처리할 준비를 갖추게 됩니다.

DAT204 - Building Scalable Applications on AWS NoSQL Services (re:Invent 2015)

DAT207 - Accelerating Application Performance with Amazon ElastiCache (AWS re:Invent 2013)

이 동영상에서는 애플리케이션 성능의 속도를 높이기 위해 Amazon ElastiCache를 사용하여 인 메모리 캐시를 쉽게 배포하는 방법에 대해 알아봅니다. Amazon ElastiCache를 사용하여 애플리케이션 지연 시간을 개선하고 데이터베이스 서버에 대한 부담을 줄이는 방법을 알아봅니다. 또한 애플리케이션이 성장함에 따라 쉽게 관리 및 조정할 수 있는 캐싱 레이어를 구축하는 방법도 알아봅니다. 이 세션 중 캐싱을 활성화하여 혜택을 받을 수 있는 다양한 시나리오 및 사용 사례를 살펴보고, Amazon ElastiCache에서 제공한 기능을 설명합니다.

DAT207 - Accelerating Application Performance with Amazon ElastiCache (re:Invent 2013)

고급 동영상

다음 동영상은 고급 Amazon ElastiCache 주제를 다룹니다.

주제

- Design for success with Amazon ElastiCache best practices (re:Invent 2020)
- Supercharge your real-time apps with Amazon ElastiCache (re:Invent 2019)
- 모범 사례: migrating Redis clusters from Amazon EC2 to ElastiCache (re:Invent 2019)
- Scaling a Fantasy Sports Platform with Amazon ElastiCache & Amazon Aurora STP11 (re:Invent 2018)
- Reliable & Scalable Redis in the Cloud with Amazon ElastiCache (re:Invent 2018)
- ElastiCache Deep Dive: Design Patterns for In-Memory Data Stores (re:Invent 2018)
- DAT305 - Amazon ElastiCache Deep Dive (re:Invent 2017)
- DAT306 - Amazon ElastiCache Deep Dive (re:Invent 2016)
- DAT407 - Amazon ElastiCache Deep Dive (re:Invent 2015)
- SDD402 - Amazon ElastiCache Deep Dive (re:Invent 2014)
- DAT307 - Deep Dive into Amazon ElastiCache Architecture and Design Patterns (re:Invent 2013)

Design for success with Amazon ElastiCache best practices (re:Invent 2020)

Redis를 기반으로 구축된 비즈니스 크리티컬 실시간 애플리케이션이 폭발적으로 증가함에 따라 가용성, 확장성 및 보안이 가장 중요한 고려 사항이 되었습니다. 온라인 확장/축소, 다중 AZ 배포 간 고가용성 및 보안 구성에서 성공하기 위해 Amazon ElastiCache를 설정하는 모범 사례를 알아봅니다.

Design for success with Amazon ElastiCache best practices (re:Invent 2020)

Supercharge your real-time apps with Amazon ElastiCache (re:Invent 2019)

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 위해 완전 관리형 인 메모리 Amazon ElastiCache를 사용하여 실시간 애플리케이션의 성능을 향상시킬 수 있는 방법을 알아봅니다.

Supercharge your real-time apps with Amazon ElastiCache (re:Invent 2019:)

모범 사례: migrating Redis clusters from Amazon EC2 to ElastiCache (re:Invent 2019)

Redis 클러스터를 직접 관리하는 것은 어려울 수 있습니다. 하드웨어를 프로비저닝하고, 소프트웨어를 패치하고, 데이터를 백업하고, 워크로드를 지속적으로 모니터링해야 합니다. 새로 출시된 Amazon ElastiCache용 온라인 마이그레이션 기능을 사용하면 이제 클러스터 모드가 비활성화된 상태에서 Amazon EC2의 자체 호스팅된 Redis에서 완전관리형 Amazon ElastiCache로 데이터를 쉽게 이전할 수 있습니다. 이 세션에서는 새로운 온라인 마이그레이션 도구에 대해 알아보고 데모를 살펴보며, 더 중요한 것은 Amazon ElastiCache로 원활하게 마이그레이션할 수 있게 해주는 실무 모범 사례를 알아봅니다.

모범 사례: migrating Redis clusters from Amazon EC2 to ElastiCache (re:Invent 2019)

Scaling a Fantasy Sports Platform with Amazon ElastiCache & Amazon Aurora STP11 (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 대담에서 Dream11의 CTO인 Amit Sharma가 이 기업이 Amazon Aurora 및 Amazon ElastiCache를 사용하여 30초 응답 시간 범위 내에서 3배까지 증가할 수 있는 플래시 트래픽을 처리하는 방법을 설명합니다. 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: Scaling a Fantasy Sports Platform with Amazon ElastiCache & Amazon Aurora (STP11)

Scaling a Fantasy Sports Platform with Amazon ElastiCache & Amazon Aurora STP11 (re:Invent 2018)

Reliable & Scalable Redis in the Cloud with Amazon ElastiCache (re:Invent 2018)

이 세션에서는 Redis 호환 서비스인 Amazon ElastiCache for Redis의 기능과 향상된 기능에 대해 다룹니다. Redis 5, 확장성 및 성능 개선 사항, 보안 및 규정 준수 등과 같은 주요 기능을 다룹니다. 또한 예정된 기능 및 고객 사례 연구에 대해서도 논의합니다.

Reliable & Scalable Redis in the Cloud with Amazon ElastiCache (re:Invent 2018)

ElastiCache Deep Dive: Design Patterns for In-Memory Data Stores (re:Invent 2018)

이 세션에서는 Amazon ElastiCache의 디자인 및 아키텍처에 대해 살펴볼 수 있는 배경 지식 자료를 제공합니다. Redis 및 Memcached 솔루션으로 일반적인 설계 패턴을 살펴보고, 고객이 인 메모리 데이터 처리에 이런 패턴을 사용하여 지연 시간을 단축하고 애플리케이션 처리량을 개선한 방법을 알아봅니다. ElastiCache 모범 사례, 설계 패턴 및 안티 패턴을 검토합니다.

ElastiCache Deep Dive: Design Patterns for In-Memory Data Stores (re:Invent 2018)

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

Amazon ElastiCache의 설계 및 아키텍처에 대한 뒷이야기를 살펴봅니다. Memcached 및 Redis 솔루션으로 일반적인 설계 패턴을 살펴보고, 고객이 인 메모리 작업에 이런 패턴을 사용하여 지연 시간을 단축하고 애플리케이션 처리량을 개선한 방법을 알아봅니다. 이 비디오 중에는 ElastiCache 모범 사례, 설계 패턴 및 안티 패턴을 검토합니다.

비디오는 다음의 내용을 소개합니다.

- ElastiCache for Redis 온라인 리샤딩
- ElastiCache 보안 및 암호화
- ElastiCache for Redis 버전 3.2.10

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

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

Amazon ElastiCache의 설계 및 아키텍처에 대한 뒷이야기를 살펴봅니다. Memcached 및 Redis 솔루션으로 일반적인 설계 패턴을 살펴보고, 고객이 인 메모리 작업에 이런 패턴을 사용하여 지연 시간을 단축하고 애플리케이션 처리량을 개선한 방법을 알아봅니다. 이 세션 중에는 ElastiCache 모범 사례, 설계 패턴 및 안티 패턴을 검토합니다.

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

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

Amazon ElastiCache의 설계 및 아키텍처에 대한 뒷이야기를 살펴봅니다. Memcached 및 Redis 솔루션의 일반적인 설계 패턴을 살펴보고, 고객이 인 메모리 작업에 이런 패턴을 사용하여 애플리케이션에 대한 개선된 지연 시간 및 처리량을 실현한 방법을 알아봅니다. 이 세션 중에는 Amazon ElastiCache와 관련된 모범 사례, 설계 패턴 및 안티 패턴을 검토합니다.

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 엔진, 요구 사항에 적합한 엔진을 결정하는 데 도움이 되는 패턴, 일관적 해싱 및 기타 빠르고 확장 가능한 애플리케이션을 구축하는 수단을 살펴봅니다. Adobe의 최고 책임 과학자인 Frank Wiebe는 Adobe에서 Amazon ElastiCache를 사용하여 고객 경험을 향상하고 비즈니스를 확장하는 방법을 자세히 설명합니다.

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

DAT307 - Deep Dive into Amazon ElastiCache Architecture and Design Patterns (re:Invent 2013)

이 동영상에서는 캐싱, 캐싱 전략, 확장, 모니터링을 살펴봅니다. 또한 Memcached 및 Redis 엔진을 비교합니다. 이 세션 중에는 Amazon ElastiCache와 관련된 모범 사례 및 설계 패턴도 검토합니다.

DAT307 - Deep Dive into Amazon ElastiCache Architecture and Design Patterns (AWS re:Invent 2013).

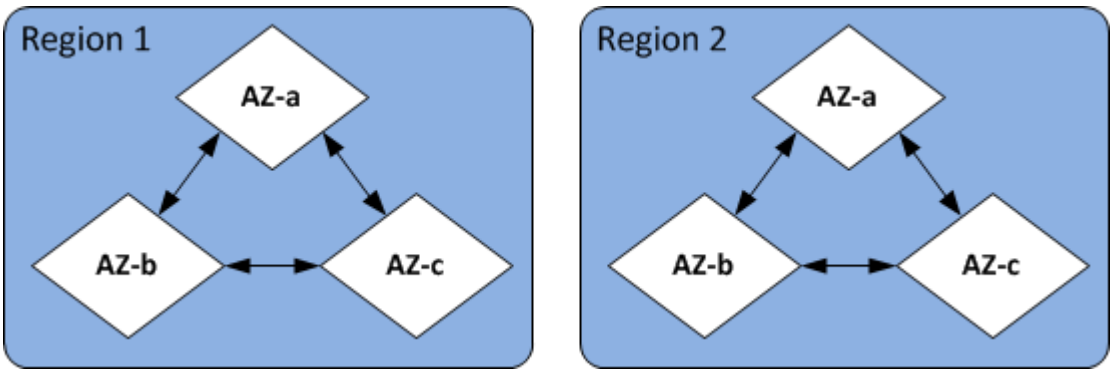
리전 및 가용 영역 선택

AWS 클라우드 컴퓨팅 리소스는 가용성이 높은 데이터 센터 시설에 보관됩니다. 추가 확장성 및 안정성을 제공하기 위해 이러한 데이터 센터 시설은 여러 물리적 위치에 배치됩니다. 이러한 위치는 리전 및 가용 영역으로 분류됩니다.

AWS 지역은 크고 여러 지리적 위치에 널리 분산되어 있습니다. 가용 영역은 다른 가용 영역의 장애로부터 격리되도록 설계된 AWS 지역 내 개별 위치입니다. 이들은 같은 AWS 지역의 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 지역에서만 실행됩니다.

특정 리전에서 클러스터를 생성하거나 사용하려면 해당하는 리전 서비스 엔드포인트를 사용하세요. 서비스 엔드포인트는 지원되는 리전 및 엔드포인트 섹션을 참조하세요.



리전 및 가용 영역

주제

- 가용 영역 고려 사항
- 지원되는 리전 및 엔드포인트
- 노드 찾기
- ElastiCache에서 로컬 영역 사용
- Outposts 사용

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 걸친 고가용성 캐시를 생성합니다. 캐시를 생성할 때 다른 가용 영역 및 동일한 VPC의 서브넷을 ElastiCache 지정하거나 기본 VPC에서 서브넷을 자동으로 선택할 수 있습니다.

Memcached 클러스터를 위한 자체 클러스터 설계 ElastiCache

Memcached 클러스터에는 최대 300개의 노드가 있을 수 있습니다. Memcached 클러스터에 노드를 만들거나 추가할 때 모든 노드에 대해 단일 가용 영역을 지정하거나, 모든 노드에 대해 단일 가용 영역을 선택하거나, 각 노드에 대해 가용 영역을 지정하거나, 각 노드에 대해 가용 영역을 ElastiCache 선택하도록 허용할 수 있습니다. ElastiCache 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 해당 노드가 여러 가용 영역에 분산되도록 하려면 다양한 가용 영역에 새 노드를 만들 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 `cross-az`로 설정합니다.
   - `NumCacheNodes`(CLI: `--num-cache-nodes`)를 현재 활성 캐시 노드 수와 생성하려는 새 캐시 노드 수를 더한 값으로 설정합니다.
   - `NewAvailabilityZones`(CLI: `--new-availability-zones`)를 새 캐시 노드를 생성할 영역의 목록으로 설정합니다. 각 새 노드의 가용 영역을 ElastiCache 결정하려면 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.

   - `NumCacheNodes`(CLI: `--num-cache-nodes`)를 이 수정이 적용된 후 필요한 활성 캐시 노드의 수로 설정합니다.
   - `CacheNodeIdsToRemove`(CLI: `--nodes-to-remove`)를 클러스터에서 제거할 캐시 노드의 목록으로 설정합니다.

     나열된 캐시 노드 ID 수는 현재 활성 노드 수에서 `NumCacheNodes`의 값을 뺀 값과 같아야 합니다.
   - (선택 사항) `ApplyImmediately`(CLI: `--apply-immediately`)를 true로 설정합니다.

     `ApplyImmediately`(CLI: `--apply-immediately`)를 true로 설정하지 않은 경우 노드 삭제는 다음 유지 관리 기간에 수행됩니다.

지원되는 리전 및 엔드포인트

ElastiCache Amazon은 여러 AWS 지역에서 사용할 수 있습니다. 즉, 요구 사항을 충족하는 위치에서 ElastiCache 클러스터를 시작할 수 있습니다. 예를 들어, 고객과 가장 가까운 AWS 지역에서 시작하거나 특정 지역에서 시작하여 특정 법적 요구 사항을 충족할 수 있습니다. AWS

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 (단`us-west-1`, 데이터가 두 개의 가용 영역에 복제되는 경우 제외) 에 데이터를 자동으로 복제합니다. 자체 ElastiCache 클러스터를 설계할 때는 내결함성을 확보하기 위해 여러 AZ에서 노드를 시작하도록 선택할 수 있습니다. 리전 및 가용 영역에 대한 자세한 내용은 이 주제의 맨 위에 있는 리전 및 가용 영역 선택 섹션을 참조하십시오.

지원되는 ElastiCache 지역

리전 이름/리전	Endpoint	프로토콜	
US East (Ohio) Region `us-east-2`	`elasticache.us-east-2.amazonaws.com`	HTTPS	
리전 - 미국 동부(버지니아 북부) `us-east-1`	`elasticache.us-east-1.amazonaws.com`	HTTPS	
US West (N. California) Region `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	

리전 이름/리전	Endpoint	프로토콜	
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	
Asia Pacific (Tokyo) Region 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	
Asia Pacific (Singapore) Region ap-southeast-1	elasticache.ap-southeast-1.amazonaws.com	HTTPS	

리전 이름/리전	Endpoint	프로토콜	
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	
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	

리전 이름/리전	Endpoint	프로토콜	
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	
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	

리전 이름/리전	Endpoint	프로토콜	
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	
AWS GovCloud (미국) 사용에 대한 자세한 내용은 AWS GovCloud (미국) 지역의 서비스를 참조하십시오. ElastiCache ElastiCache			

일부 리전은 노드 유형의 하위 집합을 지원합니다. AWS 지역별 지원되는 노드 유형 표는 을 참조하십시오AWS 리전별로 지원되는 노드 유형.

지역별 AWS 제품 및 서비스 표는 지역별 제품 및 서비스를 참조하십시오.

노드 찾기

ElastiCache Amazon은 단일 또는 다중 가용 영역 (AZ) 에 클러스터의 모든 노드를 찾는 것을 지원합니다. 또한 여러 AZ에 노드를 배치하기로 선택한 경우 (권장), ElastiCache 각 노드에 대해 AZ를 선택하거나 대신 선택할 수 ElastiCache 있습니다.

여러 AZ에서 노드를 찾으면 AZ 하나에서 정전과 같은 장애가 발생할 경우 전체 시스템이 실패하는 경우가 없어집니다.

기존 클러스터를 수정할 때 노드를 추가거나 클러스터를 생성할 때 각 노드에 대한 AZ를 지정할 수 있습니다. 자세한 내용은 다음을 참조하세요.

- 클러스터 생성
- 클러스터 수정 ElastiCache
- 클러스터에 노드 추가

ElastiCache에서 로컬 영역 사용

로컬 영역은 사용자와 지리적으로 가까운 AWS 리전의 확장입니다. 새 서브넷을 만들고 로컬 영역에 할당하여 상위 AWS 리전에서 Local Zones로 Virtual Private Cloud(VPC)를 확장할 수 있습니다. 로컬 영역에 서브넷을 생성하면 VPC도 해당 로컬 영역으로 확장됩니다. 로컬 영역의 서브넷은 VPC의 다른 서브넷과 동일하게 작동합니다.

Local Zones를 사용하면 사용자에게 가까운 여러 위치에서 ElastiCache 클러스터와 같은 리소스를 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 코드 뒤에 위치를 나타내는 식별자를 붙여 표시됩니다(예: `us-west-2-lax-1a`).

현재 사용 가능한 Local Zones는 `us-west-2-lax-1a` 및 `us-west-2-lax-1b`입니다.

ElastiCache for Local Zones에는 다음과 같은 제한 사항이 적용됩니다.

- 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 Local Zones 활성화를 참조하세요.
2. 로컬 영역에서 서브넷을 만듭니다.

   자세한 내용은 Amazon VPC 사용 설명서의 VPC에서 서브넷 만들기를 참조하세요.
3. 로컬 영역에서 ElastiCache 서브넷 그룹을 생성합니다.

   ElastiCache 서브넷 그룹을 생성할 때 로컬 영역에 대한 가용 영역 그룹을 선택합니다.

   자세한 내용은 ElastiCache 사용 설명서의 서브넷 그룹 생성을 참조하세요.
4. 로컬 영역에서 ElastiCache 서브넷을 사용하는 ElastiCache for Memcached 클러스터를 생성합니다.

   자세한 내용은 Memcached 클러스터 생성(콘솔) 섹션을 참조하세요.

Outposts 사용

AWS Outposts는 AWS 인프라, 서비스, API 및 도구를 고객 사업장으로 확장하는 완전 관리형 서비스입니다. AWS Outposts는 AWS 관리형 인프라에 대한 로컬 액세스를 제공함으로써 고객이 AWS 지역에서와 동일한 프로그래밍 인터페이스를 사용하여 온프레미스에서 애플리케이션을 구축하고 실행하는 동시에 로컬 컴퓨팅 및 스토리지 리소스를 사용하여 지연 시간을 줄이고 로컬 데이터 처리 요구 사항을 줄일 수 있도록 합니다. Outpost는 고객 사이트에 배포되는 AWS 컴퓨팅 및 스토리지 용량 풀입니다. AWS AWS 지역의 일부로서 이 용량을 운영, 모니터링 및 관리합니다. Outpost에서 서브넷을 생성하고 클러스터와 같은 AWS ElastiCache 리소스를 생성할 때 서브넷을 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.
> - ElastiCache on Outposts는 CoIP를 지원하지 않습니다.
> - ElastiCache for Outposts는 cn-north-1, cn-northwest-1 및 ap-northeast-3 지역에서는 지원되지 않습니다.

Memcached 콘솔에서 Outposts 사용

1. https://console.aws.amazon.com/elasticache/ 에서 AWS Management Console 로그인하고 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 Memcached 캐시를 선택합니다.
3. Memcached 캐시 생성을 선택합니다.
4. 클러스터 설정에서 자체 캐시 및 클러스터 캐시 설계를 선택합니다. 클러스터 모드를 비활성화됨으로 설정된 상태로 둡니다. 그런 다음 캐시의 이름과 선택적 설명을 생성합니다.
5. 위치에서는 온-프레미스를 선택합니다.
6. 온-프레미스 섹션에는 Outpost ID 필드가 표시됩니다. 클러스터가 실행될 위치의 ID를 입력합니다.

   클러스터 설정 아래의 모든 추가 설정을 기본값으로 유지할 수 있습니다.
7. 연결에서 새 서브넷 그룹 생성을 선택하고 VPC ID를 입력합니다. 나머지는 기본값으로 두고 다음을 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. Memcached 설정에서 다음을 수행합니다.

a. 이름: Memcached 클러스터의 이름을 입력합니다.

b. 설명 - Memcached 클러스터에 대한 설명을 입력합니다.

c. 엔진 버전 호환성: 엔진 버전은 Outpost 지역을 기반으로 합니다. AWS

d. 포트: 기본 포트 11211을 적용합니다. 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. 드롭다운 목록에서 Outposts를 선택한 다음 이 클러스터에 사용할 사용 가능한 노드 유형을 선택합니다. 그런 다음 저장을 선택합니다.

g. 노드 수: 클러스터에 필요한 노드의 수를 입력합니다.

2. 연결 상태:

a. 서브넷 그룹: 목록에서 새로 생성을 선택합니다.

- 이름: 서브넷 그룹의 이름을 입력합니다.
- 설명: 서브넷 그룹에 대한 설명을 입력합니다.
- VPC ID: VPC ID는 Outpost VPC와 일치해야 합니다.
- 가용 영역 또는 Outpost: 사용 중인 Outpost를 선택합니다.
- 서브넷 ID: Outpost에 사용할 수 있는 서브넷 ID를 선택합니다. 사용 가능한 서브넷 ID가 없는 경우 서브넷을 생성해야 합니다. 자세한 내용은 서브넷 생성을 참조하세요.

b. 생성을 선택합니다.

Outpost 클러스터 세부 정보 보기

Memcached 목록 페이지에서 AWS Outpost에 속하는 클러스터를 선택하고 클러스터 세부 정보를 볼 때 다음 사항을 참고하십시오.

- 가용 영역: ARN (Amazon 리소스 이름) 과 리소스 번호를 사용하여 AWS Outpost를 나타냅니다.
- 전초 기지 이름: 전초 기지의 이름. AWS

CLI를 통한 아웃포스트 사용 AWS

AWS Command Line Interface (AWS CLI) 를 사용하여 명령줄에서 여러 AWS 서비스를 제어하고 스크립트를 통해 이를 자동화할 수 있습니다. AWS CLI를 임시 (일회성) 작업에 사용할 수 있습니다.

다운로드 및 구성 AWS CLI

윈도우, macOS 또는 리눅스에서 AWS CLI 실행됩니다. 다음 절차에 따라 다운로드 및 구성합니다.

CLI를 다운로드, 설치 및 구성하려면

1. AWS 명령줄 인터페이스 웹 페이지에서 AWS CLI를 다운로드하십시오.
2. 사용 설명서의 AWS CLI 설치 및 AWS CLI 구성 지침을 따르십시오.AWS Command Line Interface

Outposts와 함께 AWS CLI 사용

다음 CLI 작업을 사용하여 Outposts를 사용하는 캐시 클러스터를 생성합니다.

- create-cache-cluster— 이 작업을 사용할 경우 `outpost-mode` 매개 변수는 캐시 클러스터의 노드를 단일 Outpost에서 생성할지 아니면 여러 Outposts에서 생성할지를 지정하는 값을 받아들입니다.

> **Note**
>
> 현재,`single-outpost` 모드만 지원됩니다.

```
aws elasticache create-cache-cluster \
   --cache-cluster-id cache cluster id \
   --outpost-mode single-outpost \
```

Memcached 구현용 자체 ElastiCache 클러스터 설계 및 관리

ElastiCache 클러스터를 세밀하게 제어해야 하는 경우 자체 클러스터를 설계할 수 있습니다. ElastiCache를 사용하면 클러스터의 AWS 가용 영역 전반에서 노드 유형, 노드 수, 노드 배치를 선택하여 노드 기반 클러스터를 운영할 수 있습니다. ElastiCache는 완전관리형 서비스이므로 클러스터의 하드웨어 프로비저닝, 모니터링, 노드 교체 및 소프트웨어 패치를 자동으로 관리합니다.

설정 방법에 대한 자세한 설명은 설정 섹션을 참조하세요. 노드나 클러스터의 관리, 업데이트 또는 삭제에 대한 세부 사항은 노드 관리 섹션을 참조하세요. 자체 ElastiCache 클러스터를 설계할 때 Amazon ElastiCache 배포의 주요 구성 요소의 개요를 볼 수 있는 주요 개념을 참조하세요.

주제

- ElastiCache Memcached 구성 요소 및 기능용
- 클러스터 관리
- 노드 관리

ElastiCache Memcached 구성 요소 및 기능용

다음은 Amazon ElastiCache for Memcached 배포의 주요 구성 요소에 대한 개요를 확인할 수 있습니다.

주제

- ElastiCache 노드
- ElastiCache Memcached 클러스터의 경우
- AWS 지역 및 가용 영역
- ElastiCache Memcached 엔드포인트의 경우
- ElastiCache 파라미터 그룹
- ElastiCache 보안
- ElastiCache 서브넷 그룹
- ElastiCache Memcached 이벤트의 경우

ElastiCache 노드

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

노드는 안전한 네트워크에 연결된 RAM의 크기가 고정된 청크입니다. 각 노드는 Memcached의 인스턴스를 실행합니다. 필요한 경우 클러스터의 노드를 다른 인스턴스 유형으로 스케일 업하거나 스케일 다운할 수 있습니다. 자세한 설명은 ElastiCache Memcached를 위한 스케일링 섹션을 참조하세요.

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

노드 사용에 대한 비용만 지불하는 pay-as-you-go 기준으로 노드를 구매할 수 있습니다. 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 시간 동안 실행할 예약 노드를 여러 개 구매하고 가끔 pay-as-you-go 노드를 추가해야 하는 시간에 대비해 노드를 구매할 수 있습니다. 예약 노드에 대한 자세한 내용은 ElastiCache 예약 노드 섹션을 참조하세요.

Memcached 엔진은 Auto Discovery를 지원합니다. Auto Discovery는 클라이언트 프로그램이 캐시 클러스터의 모든 노드를 자동으로 식별하고 이 모든 노드에 대한 연결을 시작하고 유지 관리할 수 있는 기능입니다. Auto Discovery 기능 덕분에 애플리케이션에서는 개별 노드에 수동으로 연결할 필요가 없습니다. 그 대신에 애플리케이션은 구성 엔드포인트에 연결됩니다. 구성 엔드포인트 DNS 항목에는 캐시 노드 엔드포인트 각각에 대한 CNAME 항목이 포함되어 있습니다. 따라서 구성 엔드포인트에 연결되는 즉시 애플리케이션은 클러스터의 모든 노드에 관한 정보를 얻게 되어 모든 노드에 연결할 수 있습니다. 애플리케이션에서 각각의 캐시 노드 엔드포인트를 하드 코딩할 필요가 없습니다. 자세한 내용은 Auto Discovery를 참조하세요.

노드에 대한 자세한 내용은 노드 관리 섹션을 참조하세요.

ElastiCache Memcached 클러스터의 경우

Memcached 클러스터는 ElastiCache 노드 하나 이상의 논리적 그룹입니다. 데이터는 Memcached 클러스터의 노드로 분할됩니다.

대부분의 ElastiCache 작업은 클러스터를 대상으로 합니다.

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

자세한 내용은 다음 관련 항목을 참조하세요.

- 클러스터 관리 및 노드 관리

  클러스터, 노드 및 관련 작업에 대한 정보입니다.
- AWS 서비스 제한: 아마존 ElastiCache

  최대 노드 또는 클러스터 수와 같은 ElastiCache 한도에 대한 정보.

  이러한 한도를 초과해야 하는 경우 Amazon ElastiCache 캐시 노드 요청 양식을 사용하여 요청하십시오.
- 장애 완화

  클러스터의 내결함성 향상에 관한 정보입니다.

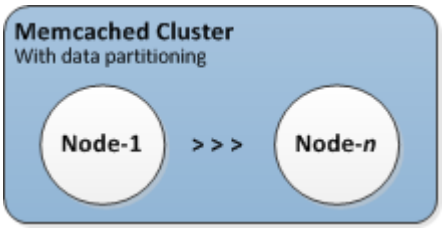
일반적인 클러스터 구성

Memcached는 AWS 지역별로 고객당 최대 300개의 노드를 지원하며, 각 클러스터에는 1~60개의 노드가 있습니다. 사용자는 데이터를 Memcached 클러스터의 노드에 두루 분할합니다.

Memcached 엔진을 실행하면 클러스터를 1~60개의 노드로 구성할 수 있습니다. 사용자는 데이터베이스를 노드에 두루 분할합니다. 애플리케이션은 각 노드의 엔드포인트를 읽고 씁니다. 자세한 내용은 Auto Discovery를 참조하세요.

내결함성을 높이려면 클러스터 지역 내의 다양한 가용 영역 (AZ) 에 Memcached 노드를 배치하십시오. AWS 이렇게 하면 한 AZ에서 발생한 오류가 전체 클러스터 및 애플리케이션에 미치는 영향을 최대한 줄일 수 있습니다. 자세한 설명은 장애 완화 섹션을 참조하세요.

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 섹션을 참조하세요. 다음 다이어그램에는 단일 노드 및 다중 노드 Memcached 클러스터의 예시가 있습니다.

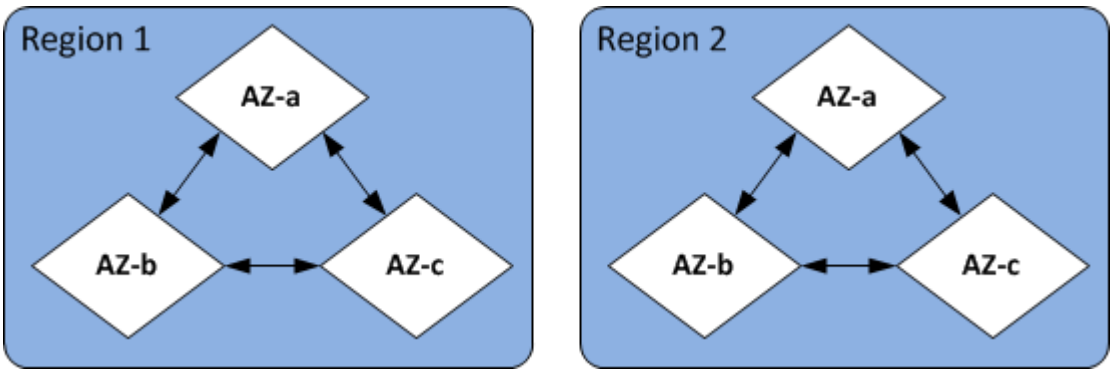


AWS 지역 및 가용 영역

ElastiCache Memcached용 Amazon은 전 세계 여러 AWS 지역에서 사용할 수 있습니다. 따라서 비즈니스 요구 사항을 충족하는 위치에서 ElastiCache 클러스터를 시작할 수 있습니다. 예를 들어 고객과 가장 가까운 AWS 지역에서 시작하거나 특정 법적 요구 사항을 충족하기 위해 시작할 수 있습니다.

기본적으로 AWS SDK AWS CLI, ElastiCache API 및 ElastiCache 콘솔은 미국 서부 (오레곤) 지역을 참조합니다. 새 AWS 지역으로 가용성이 ElastiCache 확대됨에 따라 해당 AWS 지역의 새 엔드포인트를 HTTP 요청, AWS SDK 및 콘솔에서도 사용할 수 있습니다. AWS CLI ElastiCache

각 AWS 지역은 다른 지역과 완전히 격리되도록 설계되었습니다. AWS 각 리전 내에는 가용 영역이 여러 개 있습니다. 서로 다른 가용 영역에서 노드를 시작하면 가능한 최고 수준의 내결함성을 갖출 수 있습니다. AWS 지역 및 가용 영역에 대한 자세한 내용은 을 참조하십시오리전 및 가용 영역 선택.



에서 지원하는 AWS 지역 ElastiCache 및 해당 엔드포인트에 대한 자세한 내용은 을 참조하십시오지원되는 리전 및 엔드포인트.

ElastiCache Memcached 엔드포인트의 경우

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

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 이름이 지정된 컬렉션입니다. 이를 통해 해당 클러스터에 있는 모든 노드가 정확히 동일한 방법으로 구성되게 할 수 있습니다.

지원되는 파라미터의 목록, 해당 기본값 및 수정할 수 있는 사항은 DescribeEngineDefaultParameters 섹션(describe-engine-default-parameters)을 참조하세요.

ElastiCache 파라미터 그룹에 대한 자세한 내용은 을 참조하십시오. 파라미터 그룹을 사용해 엔진 파라미터 구성

ElastiCache 보안

보안 강화를 위해 ElastiCache 노드 액세스는 화이트리스트 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 VPC (가상 사설 클라우드) 환경에서 시작됩니다. 서브넷 그룹을 사용하여 특정 서브넷에서 실행 중인 Amazon EC2 인스턴스에서의 액세스 권한을 클러스터에 부여할 수 있습니다. Amazon VPC 외부에서 클러스터를 실행하도록 선택할 경우 보안 그룹을 생성하여 특정 Amazon EC2 보안 그룹 내에서 실행되는 Amazon EC2 인스턴스를 승인할 수 있습니다.

ElastiCache 서브넷 그룹

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

Amazon VPC에서 클러스터를 생성하는 경우 캐시 서브넷 그룹을 지정해야 합니다. ElastiCache 해당 캐시 서브넷 그룹을 사용하여 해당 서브넷 내에서 캐시 노드와 연결할 서브넷 및 IP 주소를 선택합니다.

Amazon VPC 환경에서 캐시 서브넷 그룹 사용에 대한 자세한 내용은 Amazon VPC 및 ElastiCache 보안, 액세스 권한 부여 및 서브넷 및 서브넷 그룹 섹션을 참조하세요.

ElastiCache Memcached 이벤트의 경우

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

ElastiCache 이벤트에 대한 자세한 내용은 을 참조하십시오ElastiCache 이벤트에 대한 Amazon SNS 모니터링.

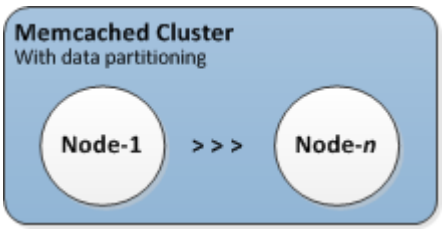
클러스터 관리

클러스터는 하나 이상의 캐시 노드 모음으로서, 이 모든 노드는 Memcached 캐시 엔진 소프트웨어의 인스턴스 하나를 실행합니다. 클러스터를 만들 때 모든 노드에서 사용할 엔진과 버전을 지정합니다.

다음 다이어그램은 일반적인 Memcached 클러스터를 나타낸 것입니다. Memcached 클러스터에는 데이터를 수평으로 분할하는 1~60개의 노드가 있습니다.

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

일반적인 Memcached 클러스터는 다음과 같습니다.



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 ElastiCache API 및 AWS CLI 명령과 상호 작용할 때 특정 클러스터를 지정합니다. 클러스터 식별자는 특정 AWS 지역의 해당 고객에 대해 고유해야 합니다.

ElastiCache 여러 엔진 버전을 지원합니다. 특별한 이유가 없으면 최신 버전을 사용하는 것이 좋습니다.

ElastiCache 클러스터는 Amazon EC2 인스턴스를 사용하여 액세스할 수 있도록 설계되었습니다. Amazon VPC 서비스 기반의 Virtual Private Cloud(VPC)에서 클러스터를 시작하는 경우 AWS밖에서 액세스할 수 있습니다. 자세한 정보는 Amazon VPC의 ElastiCache 캐시에 액세스하기 위한 액세스 패턴을 참조하세요.

지원되는 Memcached 버전 목록은 Memcached 버전 지원을 ElastiCache 참조하십시오.

네트워크 유형 선택

ElastiCache는 인터넷 프로토콜 버전 4(IPv4) 및 6(IPv6)를 지원하므로 클러스터가 다음을 수락하도록 구성할 수 있습니다.

- IPv4 연결만 가능.
- IPv6 연결만 가능.
- IPv4 및 IPv6 연결 모두(듀얼 스택).

IPv6는 Nitro 시스템에 빌드된 모든 인스턴스에서 Memcached 엔진 버전 1.6.6 이상을 사용하는 워크로드에 지원됩니다. IPv6을 통해 ElastiCache에 액세스하는 데 대한 추가 요금은 없습니다.

> **Note**
>
> IPv6/듀얼 스택이 제공되기 전에 생성된 클러스터의 마이그레이션은 지원되지 않습니다. 새로 생성된 클러스터에서의 네트워크 유형 간 전환도 지원되지 않습니다.

네트워크 유형에 맞는 서브넷 구성

Amazon VPC에서 클러스터를 생성하는 경우 서브넷 그룹을 지정해야 합니다. ElastiCache는 해당 서브넷 그룹을 사용하여 노드에 연결된 서브넷 내의 서브넷 및 IP 주소를 선택합니다. ElastiCache 클러스터가 듀얼 스택 모드에서 작동하려면 IPv4 및 IPv6 주소가 모두 할당된 듀얼 스택 서브넷이 필요하고, IPv6 전용으로 작동하려면 IPv6 전용 서브넷이 필요합니다.

듀얼 스택 사용

캐시 클러스터를 생성하고 네트워크 유형으로 듀얼 스택을 선택할 때는 IP Discovery 유형으로 IPv4 또는 IPv6을 지정해야 합니다. ElastiCache는 네트워크 유형 및 IP 검색을 IPv6으로 기본 설정하지만 이는 변경할 수 있습니다. 자동 검색을 사용하는 경우, 선택된 IP 유형의 IP 주소만 Memcached 클라이언트로 반환됩니다.

기존의 모든 클라이언트와의 하위 호환성을 유지하기 위해 IP Discovery가 도입되어, 이를 통해 검색 프로토콜에서 광고할 IP 유형(IPv4 또는 IPv6)을 선택할 수 있습니다. 이렇게 하면 자동 검색이 한 가지 IP 유형으로만 제한되지만, 가동 중단 없이 IPv4에서 IPv6 검색 IP 유형으로 마이그레이션(또는 롤백)할 수 있는 자동 검색 덕분에 이중 스택의 유용성이 유지됩니다.

TLS 활성화 듀얼 스택 ElastiCache 클러스터

ElastiCache 클러스터에 TLS가 활성화된 경우, 클러스터 검색 함수 `config get cluster`는 IP 대신 호스트 이름을 반환합니다. IP 대신 호스트 이름을 사용하여 ElastiCache 클러스터에 연결하고 TLS 핸드셰이크를 수행합니다. 따라서, 클라이언트가 IP Discovery 파라미터의 영향을 받지 않습니다. TLS 활성화 클러스터의 경우, IP Discovery 파라미터는 선호 IP 프로토콜에 영향을 끼치지 않습니다. 대신 클라이언트가 DNS 호스트 이름을 확인할 때 어떤 IP 프로토콜을 선호하는지에 따라, 사용되는 IP 프로토콜이 결정됩니다.

DNS 호스트 이름을 확인할 때 IP 프로토콜 기본 설정을 구성하는 방법에 대한 예제는 TLS를 지원하는 이중 스택 클러스터 ElastiCache 을 참조하세요.

AWS Management Console 사용

AWS Management Console을 사용하여 캐시 클러스터를 생성할 때는 연결에서 네트워크 유형으로 IPv4, IPv6, 듀얼 스택 중에 선택합니다. 듀얼 스택을 선택한 경우 검색 IP 유형으로 IPv6 또는 IPv4 중에 선택해야 합니다.

자세한 내용은 Memcached 클러스터 생성(콘솔) 섹션을 참조하세요.

CLI 사용

CLI를 사용하여 캐시 클러스터를 생성할 때는 create-cache-cluster 명령을 사용하고 `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
```

데이터 계층화

복제 그룹을 구성하고 r6gd 패밀리의 노드 유형을 사용하는 클러스터는 메모리와 로컬 SSD(solid state drives) 스토리지 간에 데이터를 계층화합니다. 데이터 계층화는 데이터를 메모리에 저장하는 것 외에도 각 클러스터 노드에서 저렴한 SSD(solid state drives)를 활용하여 Redis 워크로드에 대한 새로운 가격 대비 성능 옵션을 제공합니다. 데이터 계층화는 전체 데이터 세트의 최대 20%까지 정기적으로 액세스하는 워크로드와 SSD에서 데이터에 액세스할 때 추가 지연 시간을 허용할 수 있는 애플리케이션에 이상적입니다.

데이터 계층화가 적용된 클러스터에서는 저장하는 모든 항목의 마지막 액세스 시간을 ElastiCache 모니터링합니다. 가용 메모리 (DRAM) 가 완전히 소모되면 LRU (Least-Recest Use) 알고리즘을 ElastiCache 사용하여 자주 액세스하지 않는 항목을 메모리에서 SSD로 자동으로 이동합니다. 이후에 SSD의 데이터에 액세스하면 요청을 처리하기 전에 ElastiCache 자동으로 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 사용하면 키 자체는 항상 메모리에 남아 있지만 LRU는 메모리 대 디스크의 값 배치를 제어합니다. 일반적으로 데이터 계층화를 사용하는 경우 키 크기가 값 크기보다 작은 것이 좋습니다.

데이터 계층화는 애플리케이션 워크로드에 미치는 성능 영향을 최소화하도록 설계되었습니다. 예를 들어 500바이트 문자열 값을 가정하면 SSD에 저장된 데이터에 대한 요청 및 메모리의 데이터에 대한 요청과 비교하여 평균 300마이크로초의 지연 시간을 추가로 기대할 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 계층화는에서 지원되는 모든 Redis 명령 및 데이터 구조와 호환됩니다. ElastiCache 이 기능을 사용하려면 클라이언트 측 변경 사항이 필요하지 않습니다.

클러스터의 노드를 자동으로 식별

Memcached 엔진을 실행하는 클러스터에 대해 ElastiCache는 Auto Discovery를 지원합니다. 자동 검색은 클라이언트 프로그램이 캐시 클러스터의 모든 노드를 자동으로 식별하고 이러한 모든 노드와의 연결을 시작하고 유지하는 기능입니다.

> **Note**
>
> Amazon ElastiCache Memcached에서 실행되는 캐시 클러스터를 위해 Auto Discovery가 추가되었습니다.

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 IPv4 또는 IPv6 주소 중 선택된 것만 반환합니다. 자세한 정보는 네트워크 유형 선택을 참조하세요.

클러스터에 있는 모든 캐시 노드는 모든 다른 노드에 대한 메타데이터 목록을 유지 관리합니다. 이 메타데이터는 클러스터에서 노드가 추가되거나 제거될 때마다 업데이트됩니다.

주제

- Auto Discovery의 이점
- Auto Discovery 작동 방법
- Auto Discovery 사용
- 캐시 노드에 수동으로 연결
- Auto Discovery를 클라이언트 라이브러리에 추가
- Auto Discovery가 있는 ElastiCache 클라이언트

Auto Discovery의 이점

Auto Discovery는 다음 혜택을 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.

모든 ElastiCache Memcached 캐시 클러스터에서 Auto Discovery가 활성화됩니다. 이 기능을 사용하기 위해 캐시 노드 중 어느 것도 재부팅할 필요가 없습니다.

Auto Discovery 작동 방법

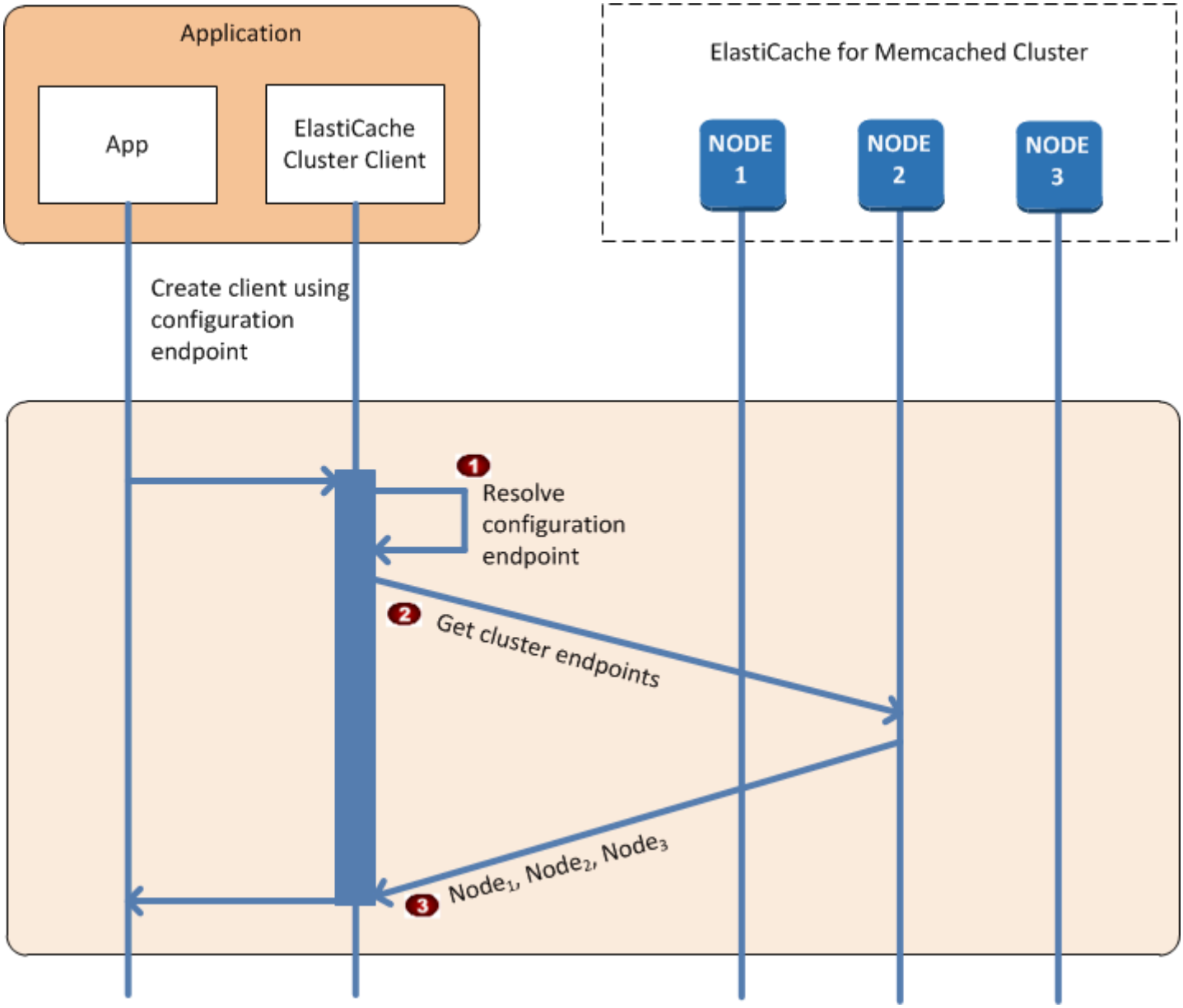
주제

- 캐시 노드에 연결
- 일반적인 클러스터 작업
- 기타 작업

이 섹션은 클라이언트 애플리케이션이 ElastiCache 클러스터 클라이언트를 사용하여 캐시 노드 연결을 관리하고 캐시의 데이터와 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

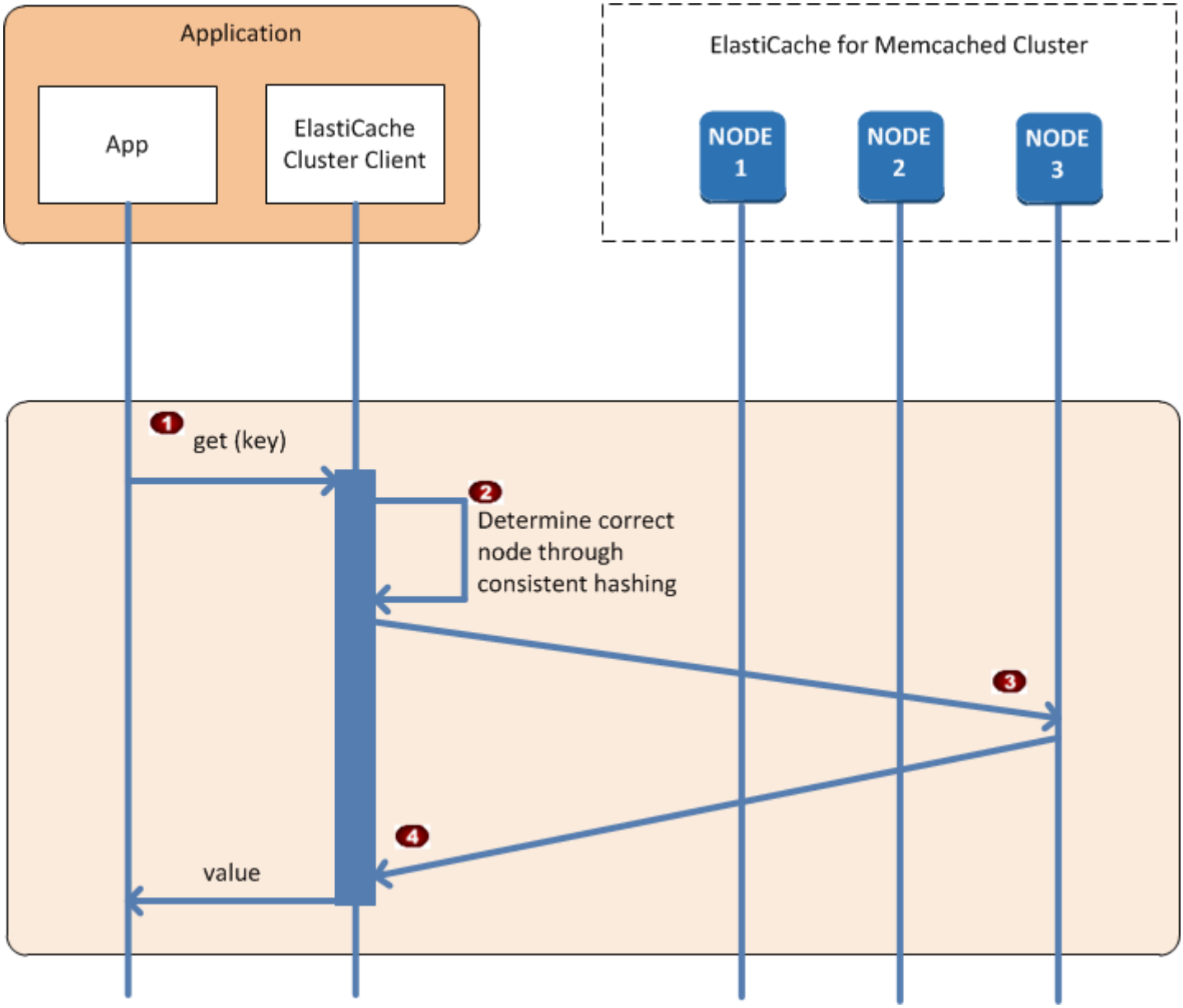
- 애플리케이션이 구성 Endpoint의 DNS 이름을 확인합니다. 구성 Endpoint가 모든 캐시 노드의 CNAME 항목을 유지 관리하기 때문에 DNS 이름은 노드 중 하나에 대해 확인되며 해당 클라이언트가 해당 노드에 연결합니다.
- 클라이언트는 모든 다른 노드에 대한 구성 정보를 요청합니다. 각 노드가 클러스터에 있는 모든 노드에 대한 구성 정보를 유지 관리하므로 요청 시 모든 노드가 구성 정보를 클라이언트에 전달할 수 있습니다.

- 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 연결되었을 때 ElastiCache 클러스터 클라이언트는 개별 데이터 항목을 저장해야 하는 노드 및 나중에 그러한 데이터 항목에 대해 쿼리되어야 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 ElastiCache가 해당 노드의 작동을 중지하고 대체합니다. 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 Discovery를 사용하려면 다음 단계를 따릅니다.

- 1단계: 구성 Endpoint 확보
- 2단계: ElastiCache 클러스터 클라이언트 다운로드
- 3단계: 애플리케이션 프로그램 수정

1단계: 구성 Endpoint 확보

클라이언트 프로그램은 클러스터에 연결하기 위해 클러스터 구성 Endpoint를 알아야 합니다. 클러스터 엔드포인트 찾기(콘솔) 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 ElastiCache용 AWS CLI를 사용하여 엔드포인트 찾기

Linux, macOS 또는 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"
        }
    ]
}
```

2단계: ElastiCache 클러스터 클라이언트 다운로드

Auto Discovery를 이용하려면 클라이언트 프로그램에서 ElastiCache 클러스터 클라이언트를 사용해야 합니다. ElastiCache 클러스터 클라이언트는 Java, PHP 및 .NET에 사용할 수 있으며 모든 캐시 노드를 검색하고 연결하는 데 필요한 모든 로직을 포함합니다.

ElastiCache 클러스터 클라이언트를 다운로드하려면

1. AWS 관리 콘솔에 로그인하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. ElastiCache 콘솔에서 ElastiCache Cluster Client(ElastiCache 클러스터 클라이언트)를 선택한 후 다운로드를 선택합니다.

Java용 ElastiCache 클러스터 클라이언트의 소스 코드는 https://github.com/amazonwebservices/aws-elasticache-cluster-client-memcached-for-java에서 사용할 수 있습니다. 이 라이브러리는 일반적으로 많이 사용되는 Spymemcached 클라이언트를 기반으로 합니다. ElastiCache 클러스터 클라이언트는 Amazon Software License(https://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**
>
> PHP용 ElastiCache 클러스터 클라이언트를 사용하려면 먼저 클라이언트를 Amazon EC2 인스턴스에 설치해야 합니다. 자세한 내용은 PHP용 ElastiCache 클러스터 클라이언트 설치 섹션을 참조하세요.
> TLS 지원 클라이언트의 경우 PHP 버전 7.4 이상의 바이너리를 다운로드합니다.
> .NET용 ElastiCache 클러스터 클라이언트를 사용하려면 먼저 클라이언트를 Amazon EC2 인스턴스에 설치해야 합니다. 자세한 내용은 .NET용 ElastiCache 클러스터 클라이언트 설치 섹션을 참조하세요.

3단계: 애플리케이션 프로그램 수정

Auto Discovery를 사용하도록 애플리케이션 프로그램을 수정합니다. 다음 섹션에서는 Java, PHP 및 .NET용 ElastiCache를 사용하는 방법을 보여줍니다.

> **Important**
>
> 클러스터의 구성 엔드포인트를 지정할 때 다음과 같이 엔드포인트의 주소에 ".cfg"를 포함해야 합니다. ".cfg"를 포함하지 않은 엔드포인트 또는 CNAME을 사용하지 마십시오.
>
> ```
> "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 데이터 항목을 캐시에 추가하는 방법을 설명합니다. 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");
    }
}
```

PHP용 ElastiCache 클러스터 클라이언트 사용

아래 프로그램은 ElastiCache 클러스터 클라이언트를 사용하여 클러스터 구성 엔드포인트에 연결하고 데이터 항목을 캐시에 추가하는 방법을 설명합니다. 프로그램은 Auto Discovery를 사용하여 추가 개입 없이 클러스터에 있는 모든 노드에 연결합니다.

PHP용 ElastiCache 클러스터 클라이언트를 사용하려면 먼저 클라이언트를 Amazon EC2 인스턴스에 설치해야 합니다. 자세한 내용은 PHP용 ElastiCache 클러스터 클라이언트 설치 단원을 참조하십시오.

```
<?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');
 ?>
```

TLS가 활성화된 ElastiCache 클러스터 클라이언트를 사용하는 방법의 예제는 PHP 및 Memcached에서 전송 시 암호화 사용 섹션을 참조하세요.

.NET용 ElastiCache 클러스터 클라이언트 사용

> **Note**
>
> ElastiCache .NET 클러스터 클라이언트는 2022년 5월부터 사용 중단되었습니다.

ElastiCache용 .NET 클라이언트는 https://github.com/awslabs/elasticache-cluster-config-net에 있는 오픈 소스입니다.

.NET 애플리케이션은 대개 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 데이터 항목을 캐시에 추가하는 방법을 설명합니다. 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
```

캐시 노드에 수동으로 연결

클라이언트 프로그램이 Auto Discovery를 사용하지 않는 경우 프로그램은 수동으로 캐시 노드 각각에 연결할 수 있습니다. 이 방법이 Memcached 클라이언트의 기본 방법입니다.

AWS 관리 콘솔에서 캐시 노드 호스트 이름 및 포트 번호 목록을 확보할 수 있습니다. 또한 다음과 같이 AWS CLI 파라미터를 포함하여 `aws elasticache describe-cache-clusters --show-cache-node-info` 명령을 사용할 수 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.

Auto Discovery를 클라이언트 라이브러리에 추가

Auto Discovery의 구성 정보는 각 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 ElastiCache에 의해 Memcached ASCII 및 바이너리 프로토콜에 추가되었으며 ElastiCache 클러스터 클라이언트에서 구현됩니다. 다른 클라이언트 라이브러리에 Auto Discovery를 사용하려면 해당 라이브러리가 `config` 명령을 지원하도록 확장될 필요가 있습니다.

> **Note**
>
> 다음 설명은 ASCII 프로토콜에 관련한 것이지만 `config` 명령은 ASCII 및 바이너리 모두를 지원합니다. 바이너리 프로토콜을 사용하여 Auto Discovery 지원을 추가하려면 ElastiCache 클러스터 클라이언트에 대한 소스 코드를 참조하세요.

구문

```
config [sub-command] [key]
```

옵션

이름	설명	필수
`sub-command`	캐시 노드와 상호 작용하는 데 사용되는 하위 명령입니다. Auto Discovery의 경우 이 하위 명령은 `get`입니다.	예
`key`		예

이름	설명	필수
	클러스터 구성이 저장되어 있는 키입니다. Auto Discovery의 경우 이 키 이름은 `cluster`입니다.	

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**
>
> "AmazonElastiCache:cluster" 키에는 클러스터 구성 정보가 상주하므로 이 키를 함부로 변경하지 마십시오. 이 키를 덮어쓰면 ElastiCache가 자동으로 올바르게 구성 정보를 업데이트하기 전에 클라이언트가 짧은 시간 동안(15초 미만) 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 사설 IP 주소 모두를 표시됩니다. CNAME은 항상 있으며 사설 IP 주소가 사용 불가능한 경우 표시되지 않지만 파이프 문자 "|"는 여전히 프린트됩니다.

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.

Auto Discovery가 있는 ElastiCache 클라이언트

이 섹션에서는 ElastiCache PHP 및 .NET 클라이언트의 설치 및 구성에 대해 설명합니다.

주제

- 클러스터 클라이언트 설치 및 컴파일
- ElastiCache 클라이언트 구성

클러스터 클라이언트 설치 및 컴파일

이 섹션에서는 PHP 및 .NET Amazon ElastiCache 자동 검색 클러스터 클라이언트의 설치, 구성 및 컴파일을 다룹니다.

주제

- .NET용 ElastiCache 클러스터 클라이언트 설치

- PHP용 ElastiCache 클러스터 클라이언트 설치
- PHP용 ElastiCache 클러스터 클라이언트에 대한 소스 코드 컴파일

.NET용 ElastiCache 클러스터 클라이언트 설치

> **Note**
>
> ElastiCache .NET 클러스터 클라이언트는 2022년 5월부터 사용 중단되었습니다.

ElastiCache .NET 클러스터 클라이언트 코드는 https://github.com/awslabs/elasticache-cluster-config-net에서 오픈 소스로 찾을 수 있습니다.

이 섹션에서는 Amazon EC2 인스턴스에서 ElastiCache 클러스터 클라이언트에 대한 .NET 구성 요소를 설치, 업데이트 및 제거하는 방법에 대해 설명합니다. 자동 검색에 대한 자세한 내용은 자동 검색 섹션을 참조하세요. 클라이언트를 사용할 샘플 .NET 코드는 DotNET으로 자동 검색 섹션을 참조하세요.

주제

- .NET 설치
- ElastiCache용 ElastiCache .NET 클러스터 클라이언트 다운로드
- NuGet으로 AWS 어셈블리 설치

.NET 설치

ElastiCache용 AWS .NET SDK를 사용하려면 .NET 3.5 이상이 설치되어 있어야 합니다. .NET 3.5 이상이 설치되어 있지 않은 경우 http://www.microsoft.com/net에서 최신 버전을 다운로드하여 설치할 수 있습니다.

ElastiCache용 ElastiCache .NET 클러스터 클라이언트 다운로드

ElastiCache .NET 클러스터 클라이언트를 다운로드하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 [ElastiCache Cluster Client]를 클릭합니다.
3. [Download ElastiCache Memcached Cluster Client] 목록에서 [.NET]을 선택한 다음 [Download]를 클릭합니다.

NuGet으로 AWS 어셈블리 설치

NuGet은 .NET 플랫폼용 패키지 관리 시스템입니다. NuGet은 어셈블리 종속성을 인식하며 모든 필수 파일을 자동으로 설치합니다. NuGet이 설치된 어셈블리는 `Program Files`와 같은 중앙 위치가 아닌 솔루션에 저장되므로 호환성 문제 없이 애플리케이션에 특정한 버전을 설치할 수 있습니다.

NuGet 설치

MSDN의 Installation Gallery(https://visualstudiogallery.msdn.microsoft.com/27077b70-9dad-4c64-adcf-c7cf6bc9970c 참조)에서 NuGet을 설치할 수 있습니다. Visual Studio 2010 이상을 사용하는 경우 NuGet이 자동으로 설치됩니다.

[Solution Explorer] 또는 [Package Manager Console]에서 NuGet을 사용할 수 있습니다.

Solution Explorer에서 NuGet 사용

Visual Studio 2010의 Solution Explorer에서 NuGet을 사용하려면

1. [Tools] 메뉴에서 [Library Package Manager]를 선택합니다.
2. [Package Manager Console]을 클릭합니다.

Visual Studio 2012 또는 Visual Studio 2013의 Solution Explorer에서 NuGet을 사용하려면

1. [Tools] 메뉴에서 [NuGet Package Manager]를 선택합니다.
2. [Package Manager Console]을 클릭합니다.

명령줄에서 다음에 표시된 대로 `Install-Package`를 사용하여 어셈블리를 설치할 수 있습니다.

```
Install-Package Amazon.ElastiCacheCluster
```

AWSSDK 및 AWS.Extensions 어셈블리 등 NuGet을 통해 사용할 수 있는 모든 패키지에 대한 페이지를 보려면 http://www.nuget.org의 NuGet 웹 사이트를 참조하세요. 각 패키지의 페이지에는 콘솔을 사용하여 패키지를 설치하기 위한 샘플 명령줄과 NuGet을 통해 사용할 수 있는 패키지의 이전 버전 목록이 포함되어 있습니다.

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 노드를 자동으로 식별 섹션을 참조하세요. 클라이언트를 사용할 샘플 PHP 코드는 PHP용 ElastiCache 클러스터 클라이언트 사용 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

올바른 버전의 PHP용 ElastiCache 클러스터 클라이언트를 사용하는지 확인하려면 Amazon EC2 인스턴스에 설치된 PHP 버전을 알아야 합니다. Amazon EC2 인스턴스에서 실행되는 Linux 버전이 64비트인지 32비트인지도 알아야 합니다.

Amazon EC2 인스턴스에 설치되어 있는 PHP 버전을 확인하려면

- 명령 프롬프트에서 다음 명령을 실행합니다.

```
php -v
```

PHP 버전은 다음 예와 같이 출력에 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 PHP용 ElastiCache 클러스터 클라이언트에 대한 소스 코드 컴파일 섹션을 참조하세요.

Amazon EC2 AMI 아키텍처(64비트 또는 32비트)를 확인하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 인스턴스 목록에서 Amazon EC2 인스턴스를 클릭합니다.
3. Description 탭에서 AMI: 필드를 찾습니다. 64비트 인스턴스는 설명에 `x86_64`를 포함해야 하며 32비트 인스턴스는 이 필드에 `i386` 또는 `i686`를 포함합니다.

이제 ElastiCache 클러스터 클라이언트를 다운로드할 준비가 되었습니다.

PHP용 ElastiCache 클러스터 클라이언트를 다운로드하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. ElastiCache 콘솔에서 ElastiCache Cluster Client(ElastiCache 클러스터 클라이언트)를 선택합니다.
3. ElastiCache Memcached 클러스터 클라이언트 다운로드 목록에서 PHP 버전 및 AMI 아키텍처에 맞는 ElastiCache 클러스터 클라이언트를 선택한 다음 다운로드 버튼을 선택합니다.

   TLS 지원 클라이언트의 경우 PHP 버전 7.4 이상의 바이너리를 다운로드합니다.

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

- 신규 사용자를 위한 PHP 7.x~8.x 설치
- 신규 사용자를 위한 PHP 5.x 설치

신규 사용자를 위한 PHP 7.x~8.x 설치

주제

- Amazon Linux 2 AMI에서 PHP 7.x~8.x 설치
- Amazon Linux 201609 AMI에서 PHP 7.x~8.x 설치(To install PHP 7.x - 8.x on an Amazon Linux 201609 AMI)
- SUSE Linux AMI에서 PHP 7.x~8.x 설치(To install PHP 7.x - 8.x on an SUSE Linux 15 AMI)
- Ubuntu 22.04 AMI에서 PHP 7.x~8.x 설치(To install PHP 7.x - 8.x on an Ubuntu 22.04 AMI)

Amazon Linux 2 AMI에서 PHP 7.x~8.x 설치

> **Note**
>
> 필요한 경우 *PHP-7.x*를 사용 중인 버전으로 바꿉니다.

1. AMI에서 새 인스턴스를 시작합니다.
2. 다음 명령을 실행합니다.

   ```
   sudo yum install gcc-c++ zlib-devel
   ```

3. `amazon-linux-extras`를 사용하여 PHP 7.x를 설치합니다.

   Amazon Linux 2를 사용하면 Extras Library를 사용하여 인스턴스에 애플리케이션 및 소프트웨어 업데이트를 설치할 수 있습니다. 이러한 소프트웨어 업데이트를 주제라고 합니다. 특정 버전의 주제를 설치하거나 버전 정보를 생략하여 최신 버전을 사용할 수 있습니다. 자세한 내용은 Extras Library(Amazon Linux 2)를 참조하세요.

   이렇게 하려면 다음 단계를 따르세요.

   a. 먼저 amazon-linux-extras가 설치되어 있는지 확인합니다.

b. 설치되어 있지 않은 경우에는 다음 명령을 사용하여 설치합니다.

```
sudo yum install -y amazon-linux-extras
```

c. PHP 7.x 주제를 아마존 Amazon Linux 2 시스템에서 사용할 수 있는지 확인합니다.

```
sudo amazon-linux-extras | grep php
```

d. 출력에서 모든 PHP 7 주제를 검토하고 원하는 버전을 선택합니다.

```
sudo amazon-linux-extras enable php7.x
```

e. 리포지토리에서 PHP 패키지를 설치합니다. 예:

```
sudo yum clean metadata

sudo yum install php php-devel
```

4. Amazon ElastiCache 클러스터 클라이언트를 다운로드합니다.

   - https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.

     ElastiCache 대시보드에서 ElastiCache 클러스터 클라이언트로 이동한 다음 원하는 PHP7 버전을 선택합니다.

   - 명령줄에서 `PHP-7.X`를 원하는 PHP 버전으로 바꾸고 `ARCH`를 원하는 아키텍처(X86 또는 arm)로 바꿉니다. PHP >= 7.4의 경우 OpenSSL을 원하는 OpenSSL 버전(openssl1.1 또는 openssl3)으로 바꿉니다. PHP > 7.4를 사용하는 경우 OpenSSL 접미사를 제거합니다.

     ```
     wget https://elasticache-downloads.s3.amazonaws.com/ClusterClient/PHP-7.X/
     latest-64bit-<ARCH>-<OpenSSL>
     ```

5. `tar -zxvf`를 사용하여 다운로드한 파일의 압축을 풉니다.

   ```
   tar -zxvf latest-64bit-<ARCH>-<OpenSSL>
   ```

6. 루트 권한으로 추출된 아티팩트 파일 `amazon-elasticache-cluster-client.so`를 `/usr/lib64/php/modules`로 복사합니다.

   ```
   sudo mv amazon-elasticache-cluster-client.so /usr/lib64/php/modules/
   ```

7. `extension=amazon-elasticache-cluster-client.so`를 `/etc/php.ini` 파일에 추가합니다.

8. PHP 7.4 이상이 설치된 ElastiCache 클러스터 클라이언트를 다운로드한 경우 OpenSSL 1.1.x 이상을 설치합니다. OpenSSL 1.1.1 설치 지침입니다.

```
sudo yum -y update
sudo yum install -y make gcc perl-core pcre-devel wget zlib-devel
wget https://www.openssl.org/source/openssl-1.1.1c.tar.gz
tar xvf openssl-1.1.1c.tar.gz
cd openssl-1.1.1c
./config
make
sudo make install
sudo ln -s /usr/local/lib64/libssl.so.1.1 /usr/lib64/libssl.so.1.1
```

Amazon Linux 201609 AMI에서 PHP 7.x~8.x 설치(To install PHP 7.x - 8.x on an Amazon Linux 201609 AMI)

> **Note**
>
> 필요한 경우 *php7.x*를 사용 중인 버전으로 바꿉니다.

1. AMI에서 새 인스턴스를 시작합니다. 자세한 내용은 Amazon EC2 사용 설명서의 1단계: 인스턴스 시작을 참조하세요.
2. 다음 명령을 실행합니다.

```
sudo yum install gcc-c++
```

3. PHP를 설치합니다.

```
sudo yum install php7.x
```

4. Amazon ElastiCache 클러스터 클라이언트를 다운로드합니다.

```
wget https://elasticache-downloads.s3.amazonaws.com/ClusterClient/PHP-7.x/
latest-64bit
```

5. `latest-64bit`를 추출합니다.

```
tar -zxvf latest-64bit
```

6. 루트 권한으로 추출된 아티팩트 파일 amazon-elasticache-cluster-client.so를 /usr/lib64/php/7.x/modules/로 복사합니다.

```
sudo mv artifact/amazon-elasticache-cluster-client.so /usr/lib64/php/7.x/modules/
```

7. 50-memcached.ini 파일을 생성합니다.

```
echo "extension=amazon-elasticache-cluster-client.so" | sudo tee --append /
etc/php-7.x.d/50-memcached.ini
```

8. Apache 서버를 시작하거나 다시 시작합니다.

```
sudo /etc/init.d/httpd start
```

SUSE Linux AMI에서 PHP 7.x~8.x 설치(To install PHP 7.x - 8.x on an SUSE Linux 15 AMI)

> **Note**
>
> 필요한 경우 *php7.x*를 사용 중인 버전으로 바꿉니다.

1. AMI에서 새 인스턴스를 시작합니다.
2. 다음 명령을 실행합니다.

```
sudo zypper refresh
sudo zypper update -y
sudo zypper install gcc
```

3. PHP를 설치합니다.

```
sudo yum install php7.x
```

또는

```
sudo zypper addrepo //download.opensuse.org/repositories/devel:/languages:/php/
openSUSE_Leap_15.3/ php
```

4. Amazon ElastiCache 클러스터 클라이언트를 다운로드하고 <ARCH>를 원하는 아키텍처(X86 또는 arm)로 바꿉니다. SUSE 15에는 OpenSSL1.1이 내장되어 있으므로 PHP >= 7.4의 경우 OpenSSL1을 사용하여 클라이언트 바이너리를 선택합니다. PHP < 7.4를 사용하는 경우 OpenSSL 접미사를 제거합니다.

```
wget https://elasticache-downloads.s3.amazonaws.com/ClusterClient/PHP-7.x/
latest-64bit-<ARCH>-openssl1.1
```

5. `latest-64bit`를 추출합니다.

```
tar -zxvf latest-64bit-<ARCH>-openssl1.1
```

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
```

Ubuntu 22.04 AMI에서 PHP 7.x~8.x 설치(To install PHP 7.x - 8.x on an Ubuntu 22.04 AMI)

> **Note**
>
> 필요한 경우 *php7.x*를 사용 중인 버전으로 바꿉니다.

1. AMI에서 새 인스턴스를 시작합니다.
2. 다음 명령을 실행합니다.

```
sudo apt-get update
sudo apt-get install gcc g++ make zlib1g zlib1g-dev
```

3. PHP를 설치합니다.

   a. PHP 8.1용 설치 지침입니다.

```
sudo apt install php8.1-cli php8.1-dev
```

   b. PHP 7.4용 설치 지침입니다.

```
sudo apt -y install software-properties-common
sudo add-apt-repository ppa:ondrej/php
sudo apt-get update
sudo apt -y install php7.4
```

4. Amazon ElastiCache 클러스터 클라이언트를 다운로드하고 <ARCH>를 원하는 아키텍처(X86 또는 arm)로 바꿉니다. Ubuntu 22.04에는 OpenSSL3가 내장되어 있으므로 PHP >= 7.4의 경우 OpenSSL3를 사용하여 클라이언트 바이너리를 선택합니다. PHP < 7.4를 사용하는 경우 OpenSSL 접미사를 제거합니다.

```
wget https://elasticache-downloads.s3.amazonaws.com/ClusterClient/PHP-7.x/
latest-64bit-<ARCH>-openssl3
```

5. 최신 64비트를 추출합니다.

```
tar -zxvf latest-64bit-<ARCH>-openssl3
```

6. 루트 권한으로 추출된 아티팩트 파일 `amazon-elasticache-cluster-client.so`를 php 확장 디렉터리 `/usr/lib/php/20190902`로 복사합니다. 이 확장 디렉터리가 없으면 `php -i | grep extension_dir`을 실행하여 찾을 수 있습니다.
7. `extension=amazon-elasticache-cluster-client.so` 라인을 `/etc/php/7.x/cli/php.ini` 파일에 삽입합니다.

신규 사용자를 위한 PHP 5.x 설치

주제

- Amazon Linux AMI 2014.03(64비트 및 32비트)에서 PHP 5를 설치하려면
- Red Hat Enterprise Linux 7.0 AMI(64비트 및 32비트)에서 PHP 5를 설치하려면
- Ubuntu 서버 14.04 LTS AMI(64비트 및 32비트)에서 PHP 5를 설치하려면
- SUSE Linux 엔터프라이즈 서버 11 AMI(64비트 또는 32비트)용 PHP 5를 설치하려면
- 기타 Linux 배포

Amazon Linux AMI 2014.03(64비트 및 32비트)에서 PHP 5를 설치하려면

1. Amazon Linux 인스턴스(64비트 또는 32비트)를 시작하여 인스턴스에 로그인합니다.
2. PHP 종속 파일을 설치합니다.

```
$ sudo yum install gcc-c++ php php-pear
```

3. Amazon EC2 인스턴스 및 PHP 버전용 올바른 php-memcached 패키지를 다운로드합니다. 자세한 내용은 설치 패키지 다운로드 섹션을 참조하세요.
4. php-memcached를 설치합니다. URI는 설치 패키지에 대한 다운로드 경로여야 합니다.

```
$ sudo pecl install <package download path>
```

다음은 PHP 5.4, 64비트 Linux용 샘플 설치 명령입니다. 이 샘플에 사용된 *X.Y.Z*를 실제 버전 번호로 교체해야 합니다.

```
$ sudo pecl install /home/AmazonElastiCacheClusterClient-X.Y.Z-PHP54-64bit.tgz
```

> **Note**
>
> 최신 버전의 설치 아티팩트를 사용하세요.

5. 루트/sudo 권한으로 /etc/php.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/
php.d/memcached.ini
```

6. Apache 서버를 시작하거나 다시 시작합니다.

```
sudo /etc/init.d/httpd start
```

Red Hat Enterprise Linux 7.0 AMI(64비트 및 32비트)에서 PHP 5를 설치하려면

1. Red Hat Enterprise Linux 인스턴스(64비트 또는 32비트)를 시작하여 인스턴스에 로그인합니다.
2. PHP 종속 파일을 설치합니다.

```
sudo yum install gcc-c++ php php-pear
```

3. Amazon EC2 인스턴스 및 PHP 버전용 올바른 php-memcached 패키지를 다운로드합니다. 자세한 내용은 설치 패키지 다운로드 섹션을 참조하세요.
4. php-memcached를 설치합니다. URI는 설치 패키지에 대한 다운로드 경로여야 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 14.04 LTS AMI(64비트 및 32비트)에서 PHP 5를 설치하려면

1. Ubuntu Linux 인스턴스(64비트 또는 32비트)를 시작하여 인스턴스에 로그인합니다.
2. PHP 종속 파일을 설치합니다.

```
sudo apt-get update
sudo apt-get install gcc g++ php5 php-pear
```

3. Amazon EC2 인스턴스 및 PHP 버전용 올바른 php-memcached 패키지를 다운로드합니다. 자세한 내용은 설치 패키지 다운로드 섹션을 참조하세요.
4. php-memcached를 설치합니다. URI는 설치 패키지에 대한 다운로드 경로여야 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 않으면 다운로드된 *.tgz 파일에서 PHP 클라이언트 아티팩트 amazon-elasticache-cluster-client.so를 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/sudo 권한으로 /etc/php5/cli/conf.d 디렉터리에 이름이 memcached.ini인 새 파일을 추가하고 파일에 "extension=<absolute path to amazon-elasticache-cluster-client.so>"를 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 11 AMI(64비트 또는 32비트)용 PHP 5를 설치하려면

1. SUSE Linux 인스턴스(64비트 또는 32비트)를 시작하여 인스턴스에 로그인합니다.
2. PHP 종속 파일을 설치합니다.

```
$ sudo zypper install gcc php53-devel
```

3. Amazon EC2 인스턴스 및 PHP 버전용 올바른 php-memcached 패키지를 다운로드합니다. 자세한 내용은 설치 패키지 다운로드 섹션을 참조하세요.
4. php-memcached를 설치합니다. URI는 설치 패키지에 대한 다운로드 경로여야 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 중인 PHP가 지원되는 버전인지 확인하세요. 5.3~5.5의 버전이 지원됩니다.

기타 Linux 배포

일부 시스템(특히 CentOS7 및 Red Hat Enterprise Linux (RHEL) 7.1)에서는 libsasl2.so.3이 libsasl2.so.2로 대체되었습니다. 이러한 시스템에서 ElastiCache 클러스터 클라이언트를 로드하면 libsasl2.so.2를 찾아 로드하려고 시도하지만 실패합니다. 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 이상 제거
- 이전 버전의 PHP 5 제거

이전 버전의 PHP 7 이상 제거

이전 버전의 PHP 7 이상 제거

1. 이전에 설치 지침에 표시된 대로 해당하는 PHP 라이브러리 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.

PHP용 ElastiCache 클러스터 클라이언트에 대한 소스 코드 컴파일

이 섹션에서는 PHP용 ElastiCache 클러스터 클라이언트에 대한 소스 코드를 획득 및 컴파일하는 방법을 다룹니다.

aws-elasticache-cluster-client-libmemcached 및 aws-elasticache-cluster-client-memcached-for-php의 두 패키지는 GitHub에서 가져와 컴파일해야 합니다.

주제

- libmemcached 라이브러리 컴파일
- PHP용 ElastiCache Memcached 자동 검색 클라이언트 컴파일

libmemcached 라이브러리 컴파일

필수 구성 요소 라이브러리

- OpenSSL 1.1.0 이상(./configure --disable-tls에서 TLS 지원을 사용 중지하지 않는 경우).
- SASL(libsasl2, `./configure --disable-sasl`에서 SASL 지원을 사용 중지하지 않는 경우).

aws-elasticache-cluster-client-libmemcached 라이브러리를 컴파일하려면 다음을 수행하세요.

1. Amazon EC2 인스턴스 시작
2. 라이브러리 종속 항목을 설치합니다.
   - Amazon Linux 201509 AMI / Amazon Linux 2 AMI에서

```
sudo yum -y update
sudo yum install gcc gcc-c++ autoconf libevent-devel make perl-core pcre-devel
 wget zlib-devel
// Install OpenSSL 1.1.1
wget https://www.openssl.org/source/openssl-1.1.1c.tar.gz
tar xvf openssl-1.1.1c.tar.gz
cd openssl-1.1.1c
./config
make
sudo make install
sudo ln -s /usr/local/lib64/libssl.so.1.1 /usr/lib64/libssl.so.1.1
```

   - Ubuntu 14.04 AMI의 경우(OpenSSL >= 1.1과 함께 제공되는 Ubuntu 버전에는 필요하지 않음)

```
sudo apt-get update

sudo apt-get install libevent-dev gcc g++ make autoconf libsasl2-dev

// Install OpenSSL 1.1.1
wget https://www.openssl.org/source/openssl-1.1.1c.tar.gz
tar xvf openssl-1.1.1c.tar.gz
cd openssl-1.1.1c
./config
make
sudo make install
sudo ln -s /usr/local/lib/libssl.so.1.1 /usr/lib/x86_64-linux-gnu/libssl.so.1.1
```

3. 리포지토리를 가져오고 코드를 컴파일합니다.

```
git clone https://github.com/awslabs/aws-elasticache-cluster-client-
libmemcached.git
cd aws-elasticache-cluster-client-libmemcached
touch configure.ac aclocal.m4 configure Makefile.am Makefile.in
mkdir BUILD
cd BUILD
../configure --prefix=<libmemcached-install-directory> --with-pic --disable-sasl
```

`../configure` 실행으로 `libssl`(OpenSSL 라이브러리)을 찾지 못한 경우 `PKG_CONFIG_PATH` 환경 변수를 조정해야 할 수 있습니다.

```
PKG_CONFIG_PATH=/path/to/ssl/lib/pkgconfig ../configure --prefix=<libmemcached-
install-directory> --with-pic --disable-sasl
```

또는 TLS를 사용하지 않는 경우 다음을 실행하여 사용 중지할 수 있습니다.

```
make
sudo make install
../configure –prefix=<libmemcached-install-directory> --with-pic --disable-sasl --
disable-tls
```

PHP용 ElastiCache Memcached 자동 검색 클라이언트 컴파일

다음 섹션에서는 ElastiCache Memcached Auto Discovery 클라이언트를 컴파일하는 방법을 설명합니다.

주제

- PHP 7 이상용 ElastiCache Memcached 클라이언트 컴파일
- PHP 5용 ElastiCache Memcached 클라이언트 컴파일

PHP 7 이상용 ElastiCache Memcached 클라이언트 컴파일

PHP-7.x를 사용 중인 버전으로 바꿉니다.

PHP를 설치합니다.

```
sudo yum install -y amazon-linux-extras
sudo amazon-linux-extras enable php7.x
```

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
phpize
mkdir BUILD
CD BUILD
../configure --with-libmemcached-dir=<libmemcached-install-directory> --disable-
memcached-sasl
```

../configure 실행으로 libssl(OpenSSL 라이브러리)을 찾지 못한 경우 OpenSSL의.PC 파일 디렉토리에 대한 PKG_CONFIG_PATH 환경 변수를 조정해야 할 수 있습니다.

```
PKG_CONFIG_PATH=/path/to/ssl/lib/pkgconfig ../configure --with-libmemcached-dir=<path
 to libmemcached build directory> --disable-memcached-sasl
```

또는 TLS를 사용하지 않는 경우 다음을 실행하여 사용 중지할 수 있습니다.

```
make
make install
```

```
../configure --with-libmemcached-dir=<path to libmemcached build directory> --disable-
memcached-sasl --disable-memcached-tls
```

> **Note**
>
> 다양한 Linux 플랫폼으로 이식될 수 있도록 libmemcached 라이브러리를 PHP 바이너리에 정적으로 연결할 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 5용 ElastiCache Memcached 클라이언트 컴파일

`aws-elasticache-cluster-client-memcached-for-php/` 폴더에서 다음 명령을 실행하여 `aws-elasticache-cluster-client-memcached-for-php`를 컴파일합니다.

```
git clone  https:////github.com/awslabs/aws-elasticache-cluster-client-memcached-for-
php/tree/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 Memcached와 프로토콜이 호환됩니다. 현재 기존 Memcached 환경과 함께 사용하고 있는 코드, 애플리케이션 및 인기 도구는 서비스와 원활하게 연동합니다.

이 섹션은 ElastiCache에 있는 캐시 노드에 연결하기 위한 특정 고려 사항을 설명합니다.

주제

- 노드 엔드포인트 및 포트 번호 찾기
- 자동 검색 사용을 위한 연결
- DNS 이름 및 기본 IP

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 로그인한 후 클러스터에서 실행 중인 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 이름 및 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 노드를 자동으로 식별 섹션을 참조하세요.

> **Note**
>
> 이때, Auto Discovery는 Memcached를 실행하는 캐시 클러스터에 대해서만 사용할 수 있습니다.

DNS 이름 및 기본 IP

클라이언트는 캐시 데이터를 보관 중인 서버의 주소 및 포트 번호를 포함하는 서버 목록을 유지 관리합니다. ElastiCache를 사용할 때 DescribeCacheClusters API(또는 describe-cache-clusters 명령줄 유틸리티)는 서버 목록에 사용할 수 있는 정규화된 DNS 항목과 포트 번호를 반환합니다.

> **Important**
>
> 중요한 것은 클라이언트 애플리케이션이 캐시 노드 Endpoint에 연결을 시도할 때 캐시 노드의 DNS 이름을 자주 확인하도록 구성되어 있다는 점입니다.

VPC 설치

ElastiCache는 실패 시 캐시 노드가 복구될 때 캐시 노드의 DNS 이름과 IP 주소가 모두 동일하게 유지되도록 합니다.

비 VPC 설치

ElastiCache는 실패 시 캐시 노드가 복구될 때 캐시 노드의 DNS 이름이 변경되지 않도록 하지만, 캐시 노드의 기본 IP 주소는 변경될 수 있습니다.

클라이언트 라이브러리는 대부분 기본적으로 영구 캐시 노드 연결을 지원합니다. ElastiCache를 사용할 때 영구 캐시 노드 연결을 사용하는 것이 좋습니다. 클라이언트 측 DNS 캐싱은 클라이언트 라이브러리, 언어 런타임 또는 클라이언트 운영 체제 등 여러 위치에서 발생할 수 있습니다. 캐시 노드의 IP 주소를 자주 확인하도록 하려면 각 계층의 애플리케이션 구성을 검토해야 합니다.

클러스터 준비

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

ElastiCache 클러스터는 AWS CloudFormation을 사용하여 생성할 수도 있습니다. 자세한 정보는 AWS Cloud Formation 사용 설명서의 AWS::ElastiCache::CacheCluster 섹션을 참조하세요. 여기에는 이 접근 방식을 구현하는 방법에 대한 지침도 포함되어 있습니다.

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

주제

- 요구 사항 결정
- 노드 크기 선택

요구 사항 결정

준비

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

ElastiCache 서버리스 또는 인스턴스 기반 서비스를 사용하시겠습니까?

서버리스 캐시를 사용하려면 VPC, 서브넷, 보안 그룹을 올바르게 구성했는지 확인하기만 하면 됩니다. 자세한 내용은 Amazon VPC의 ElastiCache 캐시에 액세스하기 위한 액세스 패턴를 참조하세요. 인스턴스 기반을 사용하려는 ElastiCache 경우 계속 읽어보세요.

- 필요한 노드 인스턴스 유형은 무엇입니까?

  인스턴스 노드 유형 선택에 도움이 필요한 경우 Memcached 노드 크기 선택를 참조하세요.

- Amazon VPC를 기반으로 Virtual Private Cloud(VPC)에서 클러스터를 시작할 예정입니까?

  > **Important**
  >
  > VPC에서 클러스터를 시작하는 경우 클러스터를 생성하기 전에 동일한 VPC에서 서브넷 그룹을 생성해야 합니다. 자세한 설명은 서브넷 및 서브넷 그룹 섹션을 참조하세요.
  > ElastiCache Amazon EC2를 AWS 사용하여 내부에서 액세스할 수 있도록 설계되었습니다. 하지만 Amazon VPC 기반의 VPC에서 시작하고 클러스터가 VPC에 있는 경우 AWS 밖에서 액세스 권한을 제공할 수 있습니다. 자세한 설명은 Amazon VPC의 ElastiCache 캐시에 액세스하기 위한 액세스 패턴 섹션을 참조하세요.

- 파라미터 값을 사용자 지정해야 합니까?

  그렇다면 사용자 지정 파라미터 그룹을 만듭니다. 자세한 설명은 파라미터 그룹 생성 섹션을 참조하세요.

- 자체 VPC 보안 그룹을 생성해야 합니까?

  자세한 내용은 VPC의 보안을 참조하세요.

- 어떤 방법으로 내결함성을 구현하시겠습니까?

  자세한 설명은 장애 완화 섹션을 참조하세요.

주제

- 메모리 및 프로세서 요구 사항
- Memcached 클러스터 구성
- 조정 요구 사항
- 액세스 요구 사항
- 리전, 가용 영역 및 로컬 영역 요구 사항

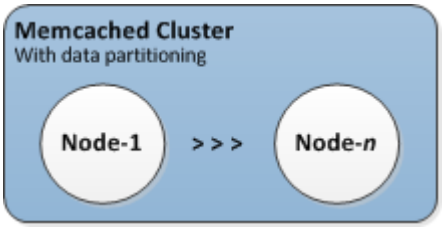
메모리 및 프로세서 요구 사항

Amazon의 기본 구성 요소는 ElastiCache 노드입니다. 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 클러스터의 경우 1~60개의 노드로 구성됩니다. Memcached 클러스터의 데이터는 클러스터의 노드로 분할됩니다. 애플리케이션은 엔드포인트라는 네트워크 주소를 사용하여 Memcached 클러스터에 연결됩니다. Memcached 클러스터의 각 노드에는 애플리케이션이 특정 노드에(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, 클러스터의 노드를 자동으로 식별에서 읽을 노드 또는 쓸 노드에 대한 결정을 남깁니다.



자세한 설명은 클러스터 관리 섹션을 참조하세요.

조정 요구 사항

더 크고 새로운 노드 유형으로 새 클러스터를 생성하여 모든 클러스터를 조정할 수 있습니다. Memcached 클러스터를 확장하면 새 클러스터가 비워집니다.

Amazon ElastiCache for Memcached 클러스터는 확장하거나 축소할 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 ElastiCache Memcached를 위한 스케일링을 참조하세요.

액세스 요구 사항

Amazon ElastiCache 클러스터는 Amazon EC2 인스턴스에서 액세스할 수 있도록 설계되어 있습니다. ElastiCache 클러스터에 대한 네트워크 액세스는 클러스터를 생성한 계정으로 제한됩니다. 따라서 Amazon EC2 인스턴스에서 클러스터에 액세스하려면 먼저 Amazon EC2 인스턴스가 클러스터에 액세스하도록 승인해야 합니다. 이를 수행하는 단계는 EC2-VPC로 시작했는지, EC2-Classic으로 시작했는지에 따라 다릅니다.

클러스터를 EC2-VPC로 시작한 경우 클러스터에 대한 네트워크 진입을 허용해야 합니다. 클러스터를 EC2-Classic으로 시작한 경우 인스턴스와 연결된 Amazon Elastic Compute Cloud 보안 그룹에 보안 그룹에 대한 액세스 권한을 부여해야 합니다 ElastiCache . 자세한 지침은 이 가이드의 클러스터에 액세스를 참조하세요.

리전, 가용 영역 및 로컬 영역 요구 사항

ElastiCache Amazon은 모든 AWS 지역을 지원합니다. 애플리케이션과 가까운 AWS 지역에 ElastiCache 클러스터를 배치하면 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 다음 자료를 참조하십시오.

- 리전 및 가용 영역
- 로컬 영역
- 장애 완화

노드 크기 선택

클러스터에 대해 선택하는 노드 크기는 비용, 성능, 내결함성에 영향을 미칩니다.

Memcached 노드 크기 선택

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 뺀 각 노드의 RAM 용량을 클러스터에 있는 노드 수에 곱하는 방식으로 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. 이러한 경우, ElastiCache가 오류 캐시 노드를 대체하기 위해 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 USD	6	41.1	12	635 USD

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 USD	3	42.12	6	442 USD
* 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. 옵션별 비용을 비교하려면 Amazon ElastiCache 요금을 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. ElastiCache Auto Discovery 기능을 사용하면 클라이언트 프로그램이 클러스터의 모든 노드를 자동으로 식별하여 엔드포인트를 쉽게 관리할 수 있습니다. 자세한 내용은 클러스터의 노드를 자동으로 식별 섹션을 참조하세요.

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 CloudWatch에 게시된 ElastiCache 지표를 사용하여 메모리 사용량, CPU 사용률 및 캐시 적중률을 모니터링합니다. ElastiCache용 CloudWatch 지표에 대한 자세한 내용은 CloudWatch 지표를 사용한 사용량 모니터링 섹션을 참조하세요. 프로덕션 및 대규모 워크로드에는 R5 노드의 성능과 RAM 비용 가치가 가장 우수합니다.

클러스터의 적중률이 기대에 미치지 못하면 간편하게 노드를 더 추가하여 클러스터의 총 가용 메모리를 늘릴 수 있습니다.

클러스터가 CPU의 제약을 받지만 적중률은 충분할 경우 컴퓨팅 파워가 더 큰 노드 유형으로 새로운 클러스터를 설정해 보십시오.

클러스터 생성

다음 예제는 AWS Management Console, AWS CLI 및 ElastiCache API를 사용하여 클러스터를 생성하는 방법을 보여줍니다.

Memcached 클러스터 생성(콘솔)

Memcached 엔진을 사용하는 경우 Amazon은 여러 노드에 걸쳐 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. Auto Discovery에 대한 자세한 내용은 클러스터의 노드를 자동으로 식별 섹션을 참조하세요.

Memcached 클러스터를 생성하려면 1단계: 캐시 생성의 단계를 수행하세요.

클러스터 상태가 사용 가능이 되면 클러스터에 Amazon EC2 액세스 권한을 부여하고 클러스터에 연결하며 사용할 수 있습니다. 자세한 정보는 클러스터에 액세스 및 캐시 노드에 수동으로 연결 섹션을 참조하세요.

> **Important**
>
> 클러스터를 사용할 수 있게 되면 클러스터를 적극 사용하지 않더라도 클러스터가 활성화되어 있는 매 시간 또는 60분 미만 단위로 비용이 청구됩니다. 이 클러스터의 요금 발생을 중지하려면 클러스터를 삭제해야 합니다. 클러스터 삭제 섹션을 참조하십시오.

클러스터 생성(AWS CLI)

를 사용하여 클러스터를 생성하려면 명령을 사용하십시오 AWS CLI. `create-cache-cluster`

> **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. 클러스터 삭제 섹션을 참조하십시오.

Memcached 캐시 클러스터 생성(AWS CLI)

다음 CLI 코드를 통해 노드가 3개인 Memcached 캐시 클러스터가 생성됩니다.

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
```

클러스터 생성 (ElastiCache API)

ElastiCache API를 사용하여 클러스터를 생성하려면 `CreateCacheCluster` 작업을 사용하십시오.

> **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. 클러스터 삭제 섹션을 참조하십시오.

멤캐싱 캐시 클러스터 (ElastiCache API) 생성

다음 코드는 노드 3개 (API) 가 있는 Memcached 클러스터를 만듭니다. ElastiCache

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>
```

클러스터 세부 정보 보기

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

클러스터 세부 정보 보기(콘솔)

ElastiCache 콘솔, ElastiCache용 AWS CLI 또는 ElastiCache API를 사용하여 Memcached 클러스터의 세부 정보를 볼 수 있습니다.

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

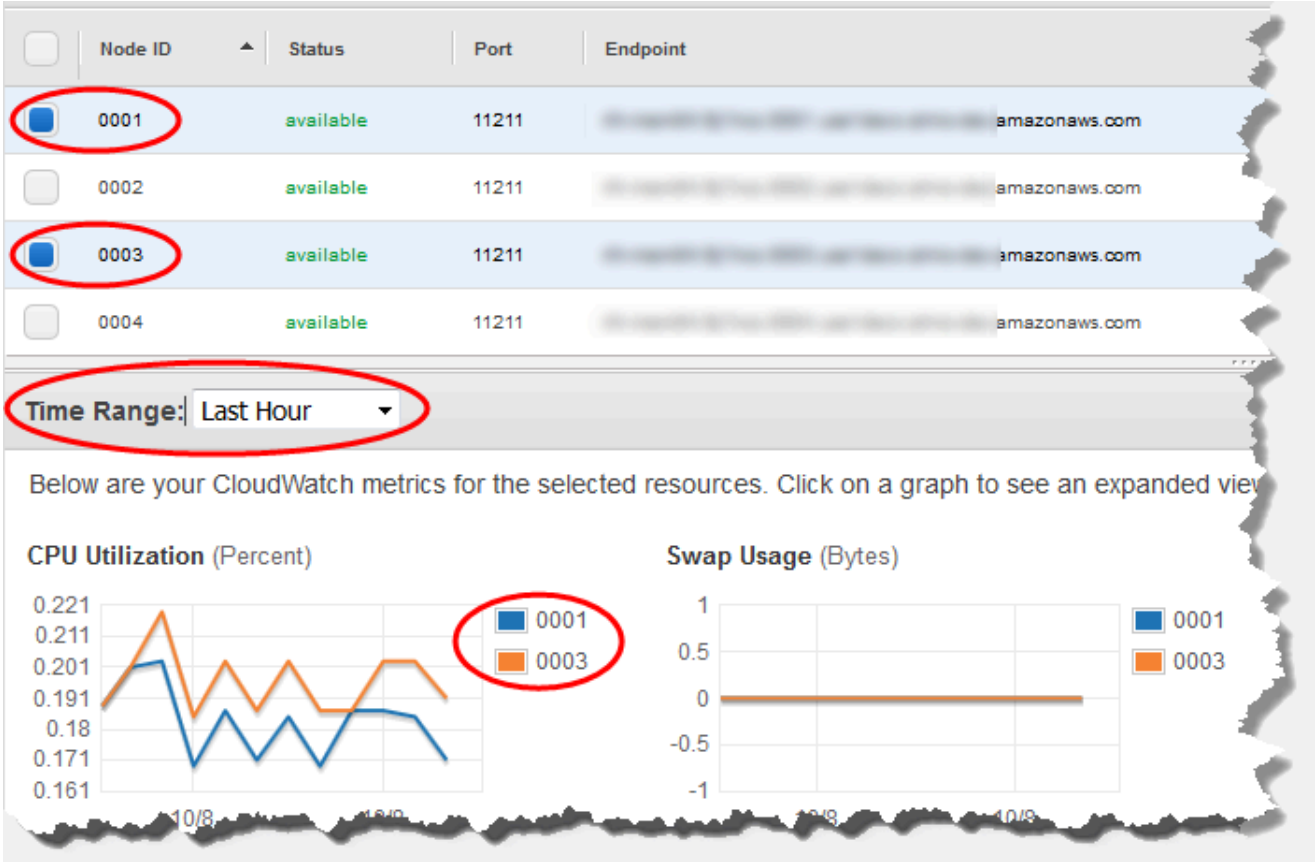
Memcached 클러스터의 세부 정보를 보려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/elasticache/에서 Amazon ElastiCache 콘솔을 엽니다.
2. 상단 오른쪽 모서리의 목록에서 원하는 AWS 리전을 선택합니다.
3. ElastiCache 콘솔 대시보드에서 Memcached를 선택합니다. 그러면 버전에 관계없이 Memcached를 실행하는 모든 클러스터 목록이 표시됩니다.
4. 클러스터의 세부 정보를 보려면 클러스터 이름의 왼쪽에 있는 상자를 선택합니다.
5. 노드 정보를 보려면

    a. 클러스터의 이름을 선택합니다.

    b. [Nodes] 탭을 선택합니다.

    c. 노드 1개 이상의 지표를 보려면 노드 ID 왼쪽의 상자를 선택한 후 시간 범위 목록에서 지표의 시간 범위를 선택합니다. 여러 노드를 선택하면 중첩 그래프가 생성됩니다.



Memcached 노드 2개에 대한 지난 1시간의 지표

클러스터 세부 정보 보기(AWS CLI)

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

다음 코드는 `my-cluster`의 세부 정보를 나열합니다.

```
aws elasticache describe-cache-clusters --cache-cluster-id my-cluster
```

다음 코드는 클러스터 최대 25개의 세부 정보를 나열합니다.

```
aws elasticache describe-cache-clusters --max-items 25
```

Example

Linux, macOS, Unix의 경우:

```
aws elasticache describe-cache-clusters \
    --cache-cluster-id my-cluster \
    --show-cache-node-info
```

Windows의 경우:

```
aws elasticache describe-cache-clusters ^
    --cache-cluster-id my-cluster ^
    --show-cache-node-info
```

이 작업은 다음과 유사한 출력을 생성합니다(JSON 형식).

```
{
    "CacheClusters": [
        {
            "Engine": "memcached",
            "CacheNodes": [
                {
                    "CacheNodeId": "0001",
                    "Endpoint": {
                        "Port": 11211,
                        "Address": "my-cluster.7ef-
example.0001.usw2.cache.amazonaws.com"
                    },
                    "CacheNodeStatus": "available",
                    "ParameterGroupStatus": "in-sync",
                    "CacheNodeCreateTime": "2016-09-21T16:28:28.973Z",
                    "CustomerAvailabilityZone": "us-west-2b"
                },
                {
                    "CacheNodeId": "0002",
                    "Endpoint": {
                        "Port": 11211,
                        "Address": "my-cluster.7ef-
example.0002.usw2.cache.amazonaws.com"
                    },
                    "CacheNodeStatus": "available",
                    "ParameterGroupStatus": "in-sync",
                    "CacheNodeCreateTime": "2016-09-21T16:28:28.973Z",
```

```
                    "CustomerAvailabilityZone": "us-west-2b"
                },
                {
                    "CacheNodeId": "0003",
                    "Endpoint": {
                        "Port": 11211,
                        "Address": "my-cluster.7ef-
example.0003.usw2.cache.amazonaws.com"
                    },
                    "CacheNodeStatus": "available",
                    "ParameterGroupStatus": "in-sync",
                    "CacheNodeCreateTime": "2016-09-21T16:28:28.973Z",
                    "CustomerAvailabilityZone": "us-west-2b"
                }
            ],
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "CacheParameterGroupName": "default.memcached1.4",
                "ParameterApplyStatus": "in-sync"
            },
            "CacheClusterId": "my-cluster",
            "PreferredAvailabilityZone": "us-west-2b",
            "ConfigurationEndpoint": {
                "Port": 11211,
                "Address": "my-cluster.7ef-example.cfg.usw2.cache.amazonaws.com"
            },
            "CacheSecurityGroups": [],
            "CacheClusterCreateTime": "2016-09-21T16:28:28.973Z",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterStatus": "available",
            "NumCacheNodes": 3,
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "CacheSubnetGroupName": "default",
            "EngineVersion": "1.4.24",
            "PendingModifiedValues": {},
            "PreferredMaintenanceWindow": "sat:09:00-sat:10:00",
            "CacheNodeType": "cache.m3.medium"
```

```
        }
    ]
}
```

자세한 내용은 ElastiCache용 AWS CLI 항목 describe-cache-clusters를 참조하세요.

클러스터 세부 정보 보기(ElastiCache API)

ElastiCache API DescribeCacheClusters 작업을 사용하여 클러스터의 세부 정보를 볼 수 있습니다. 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 정보는 ElastiCache API 참조 항목 DescribeCacheClusters을 참조하세요.

클러스터 수정 ElastiCache

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

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

클러스터의 파라미터를 변경하면 변경 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 시점을 확인하려면 Memcached 특정 파라미터 및 에 대한 테이블의 변경 적용 열을 참조하세요. 클러스터 재부팅에 관한 자세한 내용은 클러스터 재부팅 섹션을 참조하세요.

사용 AWS Management Console

클러스터를 수정하려면

1. 에 AWS Management Console 로그인하고 https://console.aws.amazon.com/elasticache/ 에서 ElastiCache 콘솔을 엽니다.
2. 오른쪽 상단의 목록에서 수정하려는 클러스터가 위치한 AWS 지역을 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.

   - 엔진 버전 호환성
   - VPC 보안 그룹
   - Parameter Group
   - 유지 관리 기간
   - SNS 알림에 대한 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.

7. 수정을 선택합니다.

사용: AWS CLI

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

> **Important**
>
> 새로운 엔진 버전으로 업그레이드할 수 있습니다. 이에 대한 자세한 내용은 엔진 버전 및 업그레이드 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 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 AWS CLI for ElastiCache 항목을 참조하십시오 `modify-cache-cluster`.

ElastiCache API 사용

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

> **Important**
>
> 새로운 엔진 버전으로 업그레이드할 수 있습니다. 이에 대한 자세한 내용은 엔진 버전 및 업그레이드 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 ElastiCache API 참조 항목을 참조하십시오 `ModifyCacheCluster`.

클러스터 재부팅

일부 변경 사항은 클러스터를 재부팅해야 적용됩니다. 예를 들어, 일부 파라미터는 파라미터 그룹의 파라미터 값을 변경할 경우 재부팅해야 변경 사항이 적용됩니다.

클러스터를 재부팅할 때 클러스터가 모든 데이터를 플러시하고 해당 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 CLI 또는ElastiCache API를 사용하여 클러스터를 재부팅할 수 있습니다. ElastiCache 콘솔, AWS CLI 또는 ElastiCache API 중 어느 것을 사용하건 단일 클러스터 재부팅만 시작할 수 있습니다. 여러 클러스터를 재부팅하려면 프로세스나 작업에서 반복해야 합니다.

AWS Management Console 사용

ElastiCache 콘솔을 사용하여 클러스터를 재부팅할 수 있습니다.

클러스터를 재부팅하려면(콘솔)

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 상단 오른쪽 모서리의 목록에서 원하는 AWS 리전을 선택합니다.
3. 탐색 창에서 재부팅하려는 클러스터에서 실행 중인 엔진을 선택합니다.

   선택한 엔진을 실행하는 클러스터 목록이 표시됩니다.
4. 클러스터 이름 왼쪽의 버튼을 선택하여 재부팅할 클러스터를 선택합니다.

   작업을 선택한 다음 재부팅을 선택합니다.

   클러스터를 2개 이상 선택하면 Reboot(재부팅) 버튼이 비활성화됩니다.

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.

AWS CLI 사용

클러스터를 재부팅하려면(AWS CLI) `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.

ElastiCache API 사용

ElastiCache API를 사용하여 클러스터를 재부팅하려면 `RebootCacheCluster` 작업을 사용하세요.

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.

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 섹션을 참조하세요.

ElastiCache 관리 콘솔, AWS CLI 또는 ElastiCache API를 사용하여 클러스터에 노드를 추가할 수 있습니다.

AWS Management Console 사용

주제

- 클러스터에 노드를 추가하려면(콘솔)

클러스터에 노드를 추가하려면(콘솔)

다음 절차에 따라 클러스터에 노드를 추가할 수 있습니다.

1. AWS Management Console에 로그인하고 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.
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.

시나리오	대기 중 작업	신규 요청	결과
			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.

AWS CLI 사용

AWS CLI를 사용하여 클러스터에 노드를 추가하려면 다음 파라미터와 함께 AWS CLI 작업 `modify-cache-cluster`를 사용하세요.

- `--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 ElastiCache가 파라미터 `cache-`

`node-ids-to-remove` 및 클러스터에서 제거할 노드 목록을 필요로 합니다. 자세한 내용은 AWS CLI 사용 섹션을 참조하세요.

- `--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",

    }
}
```

자세한 내용은 AWS CLI 항목 modify-cache-cluster를 참조하세요.

ElastiCache API 사용

클러스터에 노드를 추가하려면(ElastiCache API)

- 다음 파라미터와 함께 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 ElastiCache가 클러스터에서 제거할 노드 목록과 함께 파라미터 CacheNodeIdsToRemove를 필요로 합니다(ElastiCache API 사용 참조).
  - 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 정보는 ElastiCache API 항목 ModifyCacheCluster를 참조하세요.

클러스터에서 노드 제거

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 섹션을 참조하세요.

AWS Management Console, AWS CLI 또는 ElastiCache API를 사용하여 클러스터에서 노드를 삭제할 수 있습니다.

AWS Management Console 사용

클러스터 에서 노드를 제거하려면(콘솔)

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 상단 오른쪽 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 노드를 한 번에 하나만 삭제할 수 있으므로 노드를 여러 개 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.

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 요청

시나리오	대기 중 작업	신규 요청	결과
			이 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 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.

AWS CLI 사용

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

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

   - `--cache-cluster-id` 노드를 제거할 캐시 클러스터의 ID입니다.
   - `--num-cache-nodes` `--num-cache-nodes` 파라미터는 수정이 적용된 후 이 클러스터에 포함할 노드 수를 지정합니다.
   - `--cache-node-ids-to-remove` 이 클러스터에서 제거할 노드 ID 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.

   다음 예제에서는 my-cluster 클러스터에서 노드 0001을 즉시 제거합니다.

   Linux, macOS, Unix의 경우:

   ```
   aws elasticache modify-cache-cluster \
       --cache-cluster-id my-cluster \
       --num-cache-nodes 2 \
       --cache-node-ids-to-remove 0001 \
       --region us-east-2 \
       --apply-immediately
   ```

   Windows의 경우:

   ```
   aws elasticache modify-cache-cluster ^
       --cache-cluster-id my-cluster ^
       --num-cache-nodes 2 ^
       --cache-node-ids-to-remove 0001 ^
       --region us-east-2 ^
       --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-east-2b",
        "ConfigurationEndpoint": {
            "Port": 11211,
            "Address": "rlh-mem000.7ef-example.cfg.usw2.cache.amazonaws.com"
        },
        "CacheSecurityGroups": [],
        "CacheClusterCreateTime": "2016-09-21T16:28:28.973Z", 9dcv5r
        "AutoMinorVersionUpgrade": true,
        "CacheClusterStatus": "modifying",
        "NumCacheNodes": 3,
        "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
        "SecurityGroups": [
            {
                "Status": "active",
                "SecurityGroupId": "sg-dbe93fa2"
            }
        ],
        "CacheSubnetGroupName": "default",
        "EngineVersion": "1.4.24",
        "PendingModifiedValues": {
            "NumCacheNodes": 2,
            "CacheNodeIdsToRemove": [
                "0001"
            ]
        },
        "PreferredMaintenanceWindow": "sat:09:00-sat:10:00",
        "CacheNodeType": "cache.m3.medium",

    }
}
```

자세한 내용은 AWS CLI 항목 describe-cache-cluster 및 modify-cache-cluster를 참조하세요.

ElastiCache API 사용

ElastiCache API를 사용하여 노드를 제거하려면 다음과 같이 캐시 클러스터 ID 및 제거할 노드 목록과 함께 ModifyCacheCluster API 작업을 호출하세요.

- CacheClusterId 노드를 제거할 캐시 클러스터의 ID입니다.
- NumCacheNodes NumCacheNodes 파라미터는 수정이 적용된 후 이 클러스터에 포함할 노드 수를 지정합니다.
- CacheNodeIdsToRemove.member.n 클러스터에서 제거할 노드 ID 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 ElastiCache API 항목 ModifyCacheCluster를 참조하세요.

대기 중인 노드 추가 또는 삭제 작업 취소

변경 사항을 즉시 적용하지 않도록 선택하면 다음 번 유지 관리 기간에 수행할 때까지 작업이 [pending] 상태가 됩니다. 대기 중인 작업을 취소할 수 있습니다.

대기 중인 작업을 취소하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 상단 오른쪽 모서리의 목록에서 대기 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.

클러스터 삭제

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

AWS Management Console 사용

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. AWS Management Console에 로그인하고 https://console.aws.amazon.com/elasticache/에서 Amazon ElastiCache 콘솔을 엽니다.
2. ElastiCache 콘솔 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 콘솔에서는 클러스터를 한 번에 하나씩만 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 사용

다음 코드는 캐시 클러스터 `my-cluster`를 삭제합니다.

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

`delete-cache-cluster` CLI 작업은 캐시 클러스터를 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 ElastiCache용 AWS CLI 항목 delete-cache-cluster를 참조하세요.

ElastiCache API 사용

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 클러스터를 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 정보는 ElastiCache API 참조 항목 DeleteCacheCluster을 참조하세요.

클러스터에 액세스

Amazon ElastiCache 인스턴스는 Amazon EC2 인스턴스를 통해 액세스하도록 설계되었습니다.

Amazon Virtual Private Cloud(Amazon VPC)에서 ElastiCache 인스턴스를 시작한 경우에는 동일한 Amazon VPC에 있는 Amazon EC2 인스턴스에서 ElastiCache 인스턴스에 액세스할 수 있습니다. 또는 VPC 피어링을 사용하여 다른 Amazon VPC의 Amazon EC2에서 ElastiCache 인스턴스에 액세스할 수 있습니다.

EC2 Classic에서 ElastiCache 인스턴스를 시작하면, 인스턴스와 연결된 Amazon EC2 보안 그룹의 캐시 보안 그룹에 대한 액세스를 허용하여 EC2 인스턴스가 클러스터에 액세스하는 것을 허용합니다. 기본적으로 클러스터에 대한 액세스는 클러스터를 시작한 계정으로 제한됩니다.

주제

- 클러스터에 액세스 권한 부여

클러스터에 액세스 권한 부여

클러스터를 EC2-VPC로 시작한 경우

Amazon Virtual Private Cloud(Amazon VPC)로 클러스터를 시작한 경우 동일한 Amazon VPC에서 실행 중인 Amazon EC2 인스턴스에서만 ElastiCache 클러스터에 연결할 수 있습니다. 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 Local Zones로 확장되고 VPC는 서브넷을 다른 가용 영역에 있는 서브넷으로 처리하고 관련 게이트웨이, 라우팅 테이블 및 기타 보안 그룹 고려 사항이 자동으로 조정됩니다.

Amazon VPC 보안 그룹에서 클러스터로의 네트워크 진입을 허용하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창의 [Network & Security] 아래에서 [Security Groups]를 선택합니다.
3. 보안 그룹 목록에서 Amazon VPC를 위한 보안 그룹을 선택합니다. ElastiCache 사용을 위한 보안 그룹을 생성하지 않는 한 이 보안 그룹의 이름은 default로 지정됩니다.

4. [Inbound] 탭을 선택하고 다음을 수행합니다.

   a. 편집을 선택합니다.

   b. 규칙 추가를 선택합니다.

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

   d. [Port range] 상자에 클러스터 노드의 포트 번호를 입력합니다. 이 번호는 클러스터를 시작할 때 지정한 번호와 동일해야 합니다. Memcached의 기본 포트는 **11211** 입니다.

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

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

   f. 저장을 선택합니다.

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

외부 AWS에서 ElastiCache 리소스에 액세스

Amazon ElastiCache는 클라우드 기반 인메모리 키-값 저장소를 지원하는 AWS 서비스입니다. 이 서비스는 AWS 내에서만 액세스하도록 설계되었습니다. 그러나 ElastiCache 클러스터가 VPC 내에서 호스팅되는 경우 Network Address Translation(NAT) 인스턴스를 사용하여 외부 액세스 권한을 제공할 수 있습니다.

요구 사항

AWS 외부에서 ElastiCache 리소스에 액세스하려면 다음 요구 사항이 충족되어야 합니다.

- 클러스터는 VPC에 상주해야 하며 NAT(Network Address Translation) 인스턴스를 통해 이 클러스터에 액세스해야 합니다. 이 요구 사항에는 예외가 없습니다.
- 클러스터와 동일한 VPC에서 NAT 인스턴스를 시작해야 합니다.
- 클러스터와 다른 퍼블릭 서브넷에서 NAT 인스턴스를 시작해야 합니다.
- 탄력적 IP 주소(EIP)를 NAT 인스턴스와 연결해야 합니다. iptables의 포트 전달 기능은 NAT 인스턴스의 포트를 VPC에 있는 캐시 노드 포트로 전달하는 데 사용됩니다.

고려 사항

ElastiCache 외부에서 ElastiCache 리소스에 액세스할 때 다음 사항을 고려해야 합니다.

- 클라이언트가 NAT 인스턴스의 캐시 포트 및 EIP에 연결합니다. NAT 인스턴스의 포트 전달이 적절한 캐시 클러스터 노드로 트래픽을 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 인스턴스가 클라이언트와 여러 클러스터 간의 프록시로 작동하고 있습니다. 프록시를 추가하면 캐시 클러스터 성능에 영향을 줍니다. NAT 인스턴스를 통해 액세스하는 캐시 클러스터 수에 따라 영향이 커집니다.
- 클라이언트에서 NAT 인스턴스로 향하는 트래픽은 암호화되지 않으므로 NAT 인스턴스를 통해 민감한 데이터를 전송하면 안 됩니다.
- NAT 인스턴스로 인해 다른 인스턴스를 유지하는 부담이 커집니다.

- NAT 인스턴스가 단일 장애 지점으로 사용됩니다. VPC에서 고가용성 NAT를 설정하는 방법에 대한 자세한 정보는 Amazon VPC NAT 인스턴스의 고가용성: 예를 참조하십시오.

AWS 외부에서 ElastiCache 리소스에 액세스하는 방법

다음 절차에서는 NAT 인스턴스를 사용하여 ElastiCache 리소스에 연결하는 방법을 보여 줍니다.

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 반대 방향의 NAT가 필요합니다.

```
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*
- 신뢰할 수 있는 클라이언트의 IP 주소가 198.51.100.27입니다.
- NAT 인스턴스의 탄력적 IP 주소가 203.0.113.73입니다.
- NAT 인스턴스의 보안 그룹이 sg-ce56b7a9입니다.

NAT 인스턴스를 사용하여 ElastiCache 리소스에 연결하려면 다음과 같이 하세요.

1. 퍼블릭 서브넷이 아닌 캐시 클러스터와 동일한 VPC에 NAT 인스턴스를 만듭니다.

   기본적으로 VPC 마법사가 cache.m1.small 노드 유형을 시작합니다. 필요에 따라 노드 크기를 선택해야 합니다. AWS 외부에서 ElastiCache에 액세스하려면 EC2 NAT AMI를 사용해야 합니다.

NAT 인스턴스 생성에 대한 자세한 내용은 AWS VPC 사용 설명서의 NAT 인스턴스를 참조하세요.

2. 캐시 클러스터 및 NAT 인스턴스의 보안 그룹 규칙을 만듭니다.

   NAT 인스턴스 보안 그룹과 클러스터 인스턴스에 다음과 같은 규칙을 지정해야 합니다.

   - 인바운드 규칙 2개
     - 1개는 신뢰할 수 있는 클러스터에서 NAT 인스턴스로부터 전달된 각 캐시 포트(11211 - 11213)로 TCP 연결을 허용합니다.
     - 두 번째 포트는 신뢰할 수 있는 클라이언트에 대한 SSH 액세스를 허용합니다.

     NAT 인스턴스 보안 그룹 - 인바운드 규칙

유형	프로토콜	포트 범위	소스(Source)
사용자 지정 TCP 규칙	TCP	11211-11213	198.51.100.27-32
SSH	TCP	22	198.51.100.27-32

   - 아웃바운드 규칙은 캐시 포트(11211)와의 TCP 연결을 허용합니다.

     NAT 인스턴스 보안 그룹 - 아웃바운드 규칙

유형	프로토콜	포트 범위	대상
사용자 지정 TCP 규칙	TCP	11211	sg-ce56b7a9(클러스터 인스턴스 보안 그룹)

   - 인바운드 규칙은 NAT 인스턴스에서 캐시 포트(11211)로 TCP 연결을 허용하는 클러스터의 보안 그룹에 적용됩니다.

     클러스터 인스턴스 보안 그룹 - 인바운드 규칙

유형	프로토콜	포트 범위	소스(Source)
사용자 지정 TCP 규칙	TCP	11211	sg-bd56b7da(NAT 보안 그룹)

3. 규칙을 검증합니다.

- 신뢰할 수 있는 클라이언트가 NAT 인스턴스로 SSH할 수 있음을 확인합니다.
- 신뢰할 수 있는 클라이언트가 NAT 인스턴스에서 클러스터에 연결할 수 있음을 확인합니다.

4. NAT 인스턴스에 iptables 규칙을 추가합니다.

   클러스터에 있는 노드마다 NAT 테이블에 iptables 규칙을 추가해야 NAT 인스턴스에서 클러스터 노드로 캐시 포트를 전달할 수 있습니다. 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
   ```

5. 신뢰할 수 있는 클라이언트가 클러스터에 연결할 수 있음을 확인합니다.

   신뢰할 수 있는 클라이언트는 NAT 인스턴스와 연결된 EIP 및 적합한 클러스터 노드에 해당하는 클러스터 포트에 연결해야 합니다. 예를 들어, PHP의 연결 문자열은 다음과 같습니다.

   ```
   $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
   ```

6. 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.

- Amazon VPC의 ElastiCache 캐시에 액세스하기 위한 액세스 패턴
- 고객의 데이터 센터에서 실행되는 애플리케이션의 ElastiCache 캐시에 액세스
- NAT 인스턴스
- ElastiCache 클라이언트 구성
- Amazon VPC NAT 인스턴스의 고가용성: 예

연결 엔드포인트 찾기

애플리케이션에서 엔드포인트를 사용하여 클러스터에 연결합니다. 엔드포인트는 노드나 클러스터의 고유한 주소입니다.

사용할 엔드포인트

Memcached를 사용하는 ElastiCache 서버리스 캐시의 경우 콘솔에서 클러스터 엔드포인트 DNS와 포트를 가져오기만 하면 됩니다.

AWS CLI에서 `describe-serverless-caches` 명령을 사용하여 엔드포인트 정보를 획득합니다.

Linux

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

Windows

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

위 작업의 출력은 다음과 유사해야 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.

다음 섹션에서는 실행 중인 엔진에 필요한 엔드포인트를 찾는 방법을 안내합니다.

클러스터 엔드포인트 찾기(콘솔)

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 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 엔드포인트의 CNAME을 만들기로 선택하면 CNAME에 `.cfg.`를 포함해야 자동 검색 클라이언트가 CNAME을 구성 엔드포인트로 인식합니다.

엔드포인트 찾기(AWS CLI)

Amazon ElastiCache용 AWS CLI를 사용하여 노드 및 클러스터의 엔드포인트를 찾을 수 있습니다.

주제

- 노드 및 클러스터의 엔드포인트 찾기(AWS CLI)

노드 및 클러스터의 엔드포인트 찾기(AWS CLI)

AWS CLI 사용하여 `describe-cache-clusters` 명령으로 클러스터 및 해당 노드의 엔드포인트를 찾을 수 있습니다. 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 같습니다(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",

        }
    ]
}
```

Important

Memcached 구성 엔드포인트의 CNAME을 만들기로 선택하면 CNAME에 `.cfg.`를 포함해야 자동 검색 클라이언트가 CNAME을 구성 엔드포인트로 인식합니다. 예를 들어, `mycluster.cfg.local`가 `session.save_path` 파라미터의 php.ini 파일에 있습니다.

자세한 내용은 describe-cache-clusters 섹션을 참조하세요.

엔드포인트 찾기(ElastiCache API)

Amazon ElastiCache API를 사용하여 노드 및 클러스터의 엔드포인트를 찾을 수 있습니다.

주제

- 노드 및 클러스터의 엔드포인트 찾기(ElastiCache API)

노드 및 클러스터의 엔드포인트 찾기(ElastiCache API)

ElastiCache API를 사용하여 `DescribeCacheClusters` 작업으로 클러스터 및 해당 노드의 엔드포인트를 찾을 수 있습니다. Memcached 클러스터의 경우 명령이 구성 엔드포인트를 반환합니다. 또한 선택적 파라미터 `ShowCacheNodeInfo`를 포함할 경우 작업이 클러스터에 있는 개별 노드의 엔드포인트를 반환합니다.

Example

다음 명령은 Memcached 클러스터 mycluster의 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 엔드포인트의 CNAME을 만들기로 선택하면 CNAME에 `.cfg.`를 포함해야 자동 검색 클라이언트가 CNAME을 구성 엔드포인트로 인식합니다. 예를 들어, `mycluster.cfg.local`가 `session.save_path` 파라미터의 php.ini 파일에 있습니다.

노드 관리

노드는 Amazon ElastiCache 배포의 가장 작은 구성 요소입니다. 안전한 네트워크 부착 RAM의 크기가 고정된 청크입니다. 각 노드는 클러스터가 생성되거나 마지막으로 수정되었을 때 선택한 엔진을

실행합니다. 각 노드에는 고유한 DNS(Domain Name Service) 이름 및 포트가 있습니다. 여러 유형의 ElastiCache 노드가 지원되며, 연결된 메모리 양과 컴퓨팅 파워는 각각 다릅니다.

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

주제

- ElastiCache 노드 상태 보기
- 노드에 연결
- 지원되는 노드 유형
- 노드 교체
- ElastiCache 예약 노드
- 이전 세대 노드 마이그레이션

노드와 관련된 몇 가지 중요한 작업은 다음과 같습니다.

- 클러스터에 노드 추가
- 클러스터에서 노드 제거
- ElastiCache Memcached를 위한 스케일링
- 연결 엔드포인트 찾기

ElastiCache 노드 상태 보기

ElastiCache 콘솔을 사용하여 ElastiCache 노드 상태에 빠르게 액세스할 수 있습니다. 노드의 상태는 ElastiCache 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 표에는 해당 ElastiCache 노드에 대한 요금 청구 여부도 나와 있습니다.

Type	청구	설명
`available`	청구	ElastiCache 노드는 정상이며 사용 가능합니다.

Type	청구	설명
`creating`	미청구	ElastiCache 노드가 생성되고 있습니다. 생성 중인 노드에는 액세스할 수 없습니다.
`deleting`	미청구	ElastiCache 노드가 삭제되고 있습니다.
`modifying`	청구	고객의 ElastiCache 노드 수정 요청으로 인해 노드가 수정되고 있습니다.

Type	청구	설명
`updating`	청구	업데이트 중 상태는 Amazon ElastiCache 노드에 다음 중 하나 이상이 해당됨을 나타냅니다. • 서비스 업데이트의 일환으로 ElastiCache 노드에 패치가 적용되고 있습니다. 서비스 업데이트에 대한 자세한 내용은 Amazon ElastiCache 관리형 유지 관리 및 서비스 업데이트 도움말 페이지를 참조하십시오. • VPC 보안 그룹이 클러스터를 업데이트하고 있습니다. ElastiCache • ElastiCache 클러스터가 확장 또는 축소되고 있습니다. • 클러스터의 로그 전달 구성을 수정하고 있습니다. ElastiCache • ElastiCache 노드에 대한 삭제 작업이 보류 중입니다. • 를 사용하여 ElastiCache Redis용 비밀번호를 업데이트/교체 중입니다. AWS Secrets Manager

Type	청구	설명
`rebooting cache cluster nodes`	청구	ElastiCache 노드를 재부팅해야 하는 고객 요청 또는 Amazon ElastiCache 프로세스로 인해 노드가 재부팅되고 있습니다.
`incompatible_parameters`	미청구	노드의 파라미터 그룹에 지정된 파라미터가 노드와 호환되지 않기 때문에 Amazon에서 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 해당 ElastiCache 노드의 이벤트 목록을 확인하십시오.
`incompatible_network`	미청구	호환되지 않는 네트워크 상태는 Amazon 노드가 다음 중 하나 이상에 해당함을 나타냅니다. ElastiCache • 노드가 시작된 서브넷에는 사용 가능한 IP 주소가 없습니다. ElastiCache • ElastiCache 서브넷 그룹에 언급된 서브넷은 Amazon VPC (가상 사설 클라우드)에 더 이상 존재하지 않습니다.

Type	청구	설명
`restore_failed`	미청구	복원 실패 상태는 Amazon 노드가 다음 중 하나에 해당함을 나타냅니다. ElastiCache • 인스턴스 용량 부족이 반복되어 노드 교체가 실패했습니다. 이는 일반적으로 이전 세대 노드를 다음과 같은 이전 세대 노드를 실행할 때 발생합니다. end-of-life 하지만 지정된 가용 영역에서 요청을 처리할 온디맨드 용량이 충분하지 AWS 않은 경우 현재 세대 노드를 교체하는 경우에도 문제가 발생할 수 있습니다. 이러한 노드를 수정하거나 제거하는 방법에 대한 자세한 내용은 을 참조하십시오이전 세대 노드 마이그레이션. • 지정된 RDB 스냅샷을 복원하지 못했습니다. • ElastiCache 클러스터 AWS 계정이 일시 중지되었습니다. • 노드에 장애가 발생하여 복구할 수 없습니다.
`snapshotting`	청구	ElastiCache ElastiCache Redis용 노드의 스냅샷을 만들고 있습니다.

콘솔로 ElastiCache 노드 상태 보기

콘솔로 ElastiCache 노드의 상태를 보려면:

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

를 사용하여 ElastiCache 노드 상태 보기 AWS CLI

를 사용하여 ElastiCache 노드와 해당 상태 정보를 보려면 `describe-cache-cluster` 명령을 사용합니다. AWS CLI예를 들어, 다음 AWS CLI 명령은 각 ElastiCache 노드를 표시합니다.

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

API를 통해 ElastiCache 노드 상태 보기

Amazon ElastiCache API를 사용하여 ElastiCache 노드의 상태를 보려면 `ShowCacheNodeInfo` 플래그와 `DescribeCacheClusteroperation` 함께 를 호출하여 개별 캐시 노드에 대한 정보를 검색하십시오.

노드에 연결

Memcached 클러스터에 연결하기 전에 노드의 엔드포인트가 있어야 합니다. 엔드포인트를 찾으려면 다음을 참조하세요.

- 클러스터 엔드포인트 찾기(콘솔)
- 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. 선택한 연결 유틸리티를 사용하여 Amazon EC2 인스턴스에 연결하세요.

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

2. telnet 유틸리티를 다운로드하여 Amazon EC2 인스턴스에 설치합니다. Amazon EC2 인스턴스 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 프롬프트에서 다음 명령을 입력하고 노드의 엔드포인트를 이 예제에 표시된 것으로 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

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

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 유형을 참조하세요.

- 범용:
  - 현재 세대:

    M6g 노드 유형(Memcached 엔진 버전 1.5.16 이상에만 사용 가능)

    `cache.m6g.large`, `cache.m6g.xlarge`, `cache.m6g.2xlarge`, `cache.m6g.4xlarge`, `cache.m6g.8xlarge`, `cache.m6g.12xlarge`, `cache.m6g.16xlarge`

    > **Note**
    >
    > 리전 사용 가능성은 AWS 리전별로 지원되는 노드 유형 섹션을 참조하세요.

    M5 노드 유형: `cache.m5.large`, `cache.m5.xlarge`, `cache.m5.2xlarge`, `cache.m5.4xlarge`, `cache.m5.12xlarge`, `cache.m5.24xlarge`

    M4 노드 유형: `cache.m4.large`, `cache.m4.xlarge`, `cache.m4.2xlarge`, `cache.m4.4xlarge`, `cache.m4.10xlarge`

    T4g 노드 유형(Memcached 엔진 버전 1.5.16 이상에만 사용 가능).

    `cache.t4g.micro`, `cache.t4g.small`, `cache.t4g.medium`

    T3 노드 유형: `cache.t3.micro`, `cache.t3.small`, `cache.t3.medium`

    T2 노드 유형: `cache.t2.micro`, `cache.t2.small`, `cache.t2.medium`
  - 이전 세대: (권장하지 않음. 기존 클러스터는 계속 지원되지만, 이러한 유형에서는 새 클러스터 생성이 지원되지 않습니다.)

    T1 노드 유형: `cache.t1.micro`

    M1 노드 유형: `cache.m1.small`, `cache.m1.medium`, `cache.m1.large`, `cache.m1.xlarge`

M3 노드 유형: `cache.m3.medium`, `cache.m3.large`, `cache.m3.xlarge`, `cache.m3.2xlarge`

- 컴퓨팅 최적화:
  - 이전 세대: (권장하지 않음)

    C1 노드 유형: `cache.c1.xlarge`
- 메모리 최적화:
  - 현재 세대:

    (R6g 노드 유형은 Memcached 엔진 버전 1.5.16 이상에만 사용 가능합니다.)

    R6g 노드 유형: `cache.r6g.large`, `cache.r6g.xlarge`, `cache.r6g.2xlarge`, `cache.r6g.4xlarge`, `cache.r6g.8xlarge`, `cache.r6g.12xlarge`, `cache.r6g.16xlarge`

    > **Note**
    > 리전 사용 가능성은 AWS 리전별로 지원되는 노드 유형 섹션을 참조하세요.

    R5 노드 유형: `cache.r5.large`, `cache.r5.xlarge`, `cache.r5.2xlarge`, `cache.r5.4xlarge`, `cache.r5.12xlarge`, `cache.r5.24xlarge`

    R4 노드 유형: `cache.r4.large`, `cache.r4.xlarge`, `cache.r4.2xlarge`, `cache.r4.4xlarge`, `cache.r4.8xlarge`, `cache.r4.16xlarge`
  - 이전 세대: (권장하지 않음)

    M2 노드 유형: `cache.m2.xlarge`, `cache.m2.2xlarge`, `cache.m2.4xlarge`

    R3 노드 유형: `cache.r3.large`, `cache.r3.xlarge`, `cache.r3.2xlarge`, `cache.r3.4xlarge`, `cache.r3.8xlarge`
- 네트워크 최적화:
  - 현재 세대:

    (C7gn 노드 유형은 Memcached 엔진 버전 1.6.6 이상에만 사용 가능합니다.)

C7gn 노드 유형: `cache.c7gn.large`, `cache.c7gn.xlarge`, `cache.c7gn.2xlarge`, `cache.c7gn.4xlarge`, `cache.c7gn.8xlarge`, `cache.c7gn.12xlarge`, `cache.c7gn.16xlarge`

현재 세대

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
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

인스턴스 유형	지원되는 최소 Memcached 버전	기준 대역폭(Gbps)	버스트 대역폭(Gbps)
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
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

인스턴스 유형	지원되는 최소 Memcached 버전	기준 대역폭(Gbps)	버스트 대역폭(Gbps)
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

메모리 최적화

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
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

인스턴스 유형	지원되는 최소 버전	기준 대역폭(Gbps)	버스트 대역폭(Gbps)
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

네트워크 최적화

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

인스턴스 유형	지원되는 최소 버전	기준 대역폭(Gbps)	버스트 대역폭(Gbps)
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

AWS 리전별로 지원되는 노드 유형

지원되는 노드 유형은 AWS 지역마다 다를 수 있습니다. 자세한 내용은 Amazon ElastiCache 요금을 참조하십시오.

성능 순간 확장 가능 인스턴스

Amazon에서 버스트 가능한 범용 T4g, T3 표준 및 T2 표준 캐시 노드를 시작할 수 있습니다. ElastiCache 이러한 노드는 기준 수준의 CPU 성능과 더불어 누적된 크레딧이 소진될 때까지 언제든 CPU 사용량을 순간 확장할 수 있는 기능을 제공합니다. CPU 크레딧은 1분 동안 CPU 코어의 전체 성능을 제공합니다.

ElastiCacheAmazon의 T4g, T3 및 T2 노드는 표준으로 구성되며 평균 CPU 사용률이 인스턴스의 기준 성능보다 지속적으로 낮은 워크로드에 적합합니다. 기준 이상으로 순간 확장하려면 노드는 CPU 크레딧 밸런스에 누적한 크레딧을 사용합니다. 누적된 크레딧에서 노드가 부족한 경우, 성능이 점진적으로 기준 성능 수준으로 저하됩니다. 이렇게 점진적으로 저하되면 누적된 CPU 크레딧 밸런스가 고갈될 때 노드에 급격한 성능 저하가 발생하지 않습니다. 자세한 내용은 Amazon EC2 사용 설명서의 성능 순간 확장 가능 인스턴스에 대한 CPU 크레딧 및 기준 성능 섹션을 참조하세요.

다음 표에는 성능 순간 확장 가능 노드 유형과 시간당 CPU 크레딧이 획득되는 속도가 나와 있습니다. 또한 노드가 누적할 수 있는 최대 획득 CPU 크레딧 개수와 노드당 vCPU 개수도 보여줍니다. 또한 기준 성능 수준을 전체 코어 성능의 백분율로 제공합니다(단일 vCPU 사용).

시간당 지급되는 CPU 크레딧	누적 가능한 최대 지급된 크레딧*	vCPU	vCPU당 기준 성능	메모리(GiB)	네트워크 성능
12	288	2	10%	0.5	최대 5 기가비트

시간당 지급되는 CPU 크레딧	누적 가능한 최대 지급된 크레딧*	vCPU	vCPU당 기준 성능	메모리(GiB)	네트워크 성능
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
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 vCPU 단위로 계산됩니다. vCPU가 1개 이상인 일부 노드 크기입니다. vCPU 백분율에 vCPU 개수를 곱하여 노드의 기준 CPU 사용률을 계산합니다.

다음 CPU 크레딧 지표는 T3 및 T4g 성능 순간 확장 가능 인스턴스에 사용할 수 있습니다.

> **Note**
>
> T2 성능 순간 확장 가능 인스턴스에는 이러한 지표를 사용할 수 없습니다.

- `CPUCreditUsage`
- `CPUCreditBalance`

이 지표에 대한 자세한 내용은 CPU 크레딧 지표 섹션을 참조하세요.

또한 다음 사항을 숙지해야 합니다.

- 모든 현재 세대 노드 유형은 기본적으로 Amazon VPC를 기반으로 Virtual Private Cloud(VPC)에서 생성됩니다.

관련 정보

- Amazon ElastiCache 제품 특징 및 세부 정보
- Memcached 노드 유형별 파라미터

노드 교체

Amazon ElastiCache for Memcached는 인스턴스에 원활하게 적용되는 패치 및 업그레이드를 통해 플릿을 주기적으로 업그레이드합니다. 하지만 기본 호스트에 필수 OS 업데이트를 적용하기 위해 ElastiCache for Memcached 노드를 다시 시작해야 하는 경우가 있습니다. 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**
>
> Amazon ElastiCache에서 자동으로 생성된 교체 캐시 노드는 IP 주소가 다를 수 있습니다. 애플리케이션 구성을 검토하여 캐시 노드가 적절한 IP 주소와 연결되어 있는지 확인해야 합니다.

다음 목록은 ElastiCache에서 Memcached 노드 하나의 대체를 예약할 경우 취할 수 있는 조치를 보여줍니다.

- 아무 작업 안 함 - 아무 작업도 하지 않으면 ElastiCache에서 예약대로 노드를 대체합니다. ElastiCache에서 자동으로 노드를 새 노드로 대체하면 새 노드가 처음에는 비어 있습니다.
- 유지 관리 기간 변경 - 예약된 유지 관리 이벤트의 경우 ElastiCache에서 이메일 또는 알림 이벤트를 수신합니다. 이 경우 예약된 대체 시간 전에 유지 관리 기간을 변경하면 이제 노드가 새 시간에 대체됩니다. 자세한 내용은 클러스터 수정 ElastiCache 섹션을 참조하세요.

  > **Note**
  >
  > 유지 관리 기간을 이동해 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 유지 관리 관리 중 섹션을 참조하세요.

- 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 클러스터에서 노드 제거 섹션을 참조하세요.
  2. 클러스터에 새 노드를 추가합니다. 지침은 클러스터에 노드 추가 섹션을 참조하세요.
  3. 이 클러스터에서 자동 검색 기능을 사용하지 않으려면 애플리케이션을 확인하고 기존 노드의 모든 엔드포인트 인스턴스를 새로운 노드 엔드포인트로 대체합니다.

ElastiCache 예약 노드

하나 이상의 노드를 예약하여 비용을 줄일 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. 요금에 대한 자세한 정보는 Amazon ElastiCache 요금을 참조하세요.

> **Note**
>
> 예약된 노드는 유연하지 않으며 사용자가 예약하는 정확한 인스턴스 유형에만 적용됩니다.

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. 요금에 대한 자세한 정보는 Amazon ElastiCache 요금을 참조하세요.

AWS 리전, 노드 유형 및 기간은 구매 시 선택해야 하며 나중에 변경할 수 없습니다.

AWS Management Console, AWS CLI 또는 ElastiCache API를 사용하여 사용 가능한 예약 노드 제품을 나열하고 구입할 수 있습니다.

예약 노드에 대한 자세한 내용은 Amazon ElastiCache 예약 노드를 참조하세요.

주제

- 표준 예약 노드 제품
- 이전 예약 노드 제품
- 예약 노드 제품에 대한 정보 가져오기
- 예약 노드 구입
- 예약 노드에 대한 정보 가져오기

표준 예약 노드 제품

Amazon ElastiCache에서 표준 예약 인스턴스(RI)를 구매할 때는 예약 노드 인스턴스의 기간 동안 특정 노드 인스턴스 유형 및 AWS 리전에 대해 할인 요금을 이용하는 약정을 구매하는 것입니다. Amazon ElastiCache 예약 노드 인스턴스를 사용하려면 온디맨드 인스턴스와 똑같은 방법으로 새 ElastiCache 노드 인스턴스를 생성해야 합니다.

새 노드 인스턴스는 예약 노드 인스턴스의 사양과 정확히 일치해야 합니다. 새로운 노드 인스턴스의 사양이 계정의 기존 예약 노드 인스턴스와 일치하면 예약 인스턴스에 제공되는 할인 요금이 청구됩니다. 그렇지 않으면 노드 인스턴스에 대해 온디맨드 요금이 청구됩니다. 이 표준 RI는 R5 및 M5 인스턴스 패밀리부터 사용할 수 있습니다.

> **Note**
>
> 다음에 논의되는 세 가지 제공 유형은 1년과 3년 단위로 사용할 수 있습니다.

제공 유형

선수금 없음 RI는 선결제 금액 없이 예약된 ElastiCache 인스턴스에 대한 액세스를 제공합니다. 선결제 없음 예약 ElastiCache 인스턴스는 사용되는지 여부와 상관없이 사용 기간 동안 매시간마다 할인된 시간당 요금이 청구됩니다.

부분 선결제 RI의 경우 예약된 ElasticCache 인스턴스의 일부를 먼저 결제해야 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.

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 전체 기간에 적용됩니다.	예상되는 예약 노드 사용률이 3년 약정 기준으로 79% 이상인 경우 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 요	항상 실행할 계획이라면 전체 비용이 가장

제공 유형	선결제 비용	사용료	이점
		금이 청구됩니다. 노드가 실행되지 않을 때는 시간당 요금이 청구되지 않습니다. 모든 제공 유형 중에서 시간당 요금이 가장 높지만 예약 노드가 실행되고 있을 때만 요금이 적용됩니다.	높습니다. 그러나, 이는 3년 약정 기간의 약 15% 이상 예약 노드를 자주 사용하지 않는 경우 전체 비용이 가장 낮습니다.
온디맨드 사용(예약 노드 없음)	없음	시간당 요금이 가장 높습니다. 노드가 실행 중일 때마다 적용됩니다.	시간당 비용이 가장 높습니다.

자세한 내용은 Amazon ElastiCache 요금을 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 ElastiCache API를 사용하여 사용 가능한 예약 노드 제품의 요금과 정보를 확인하는 방법을 보여줍니다.

주제

- 예약 노드 제품에 대한 정보 가져오기(콘솔)
- 예약 노드 제품에 대한 정보 가져오기(AWS CLI)
- 예약 노드 제품에 대한 정보 가져오기(ElastiCache API)

예약 노드 제품에 대한 정보 가져오기(콘솔)

AWS Management Console을 사용하여 사용 가능한 예약 클러스터 상품의 요금과 그 밖의 정보를 가져오려면 다음 절차를 따르십시오.

사용 가능한 예약 노드 제품에 대한 정보를 가져오려면

1. AWS Management Console에 로그인하고 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(Engine)에서 Memcached를 선택합니다.
5. 사용 가능한 제품을 확인하려면 다음 옵션 중에서 선택합니다.

   - 노드 유형(Node Type)
   - 기간
   - 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)

사용 가능한 예약 노드 제품의 요금과 그 밖의 정보를 가져오려면 명령 프롬프트에 다음 명령을 입력합니다.

```
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": "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 AWS CLI 참조에서 describe-reserved-cache-nodes-offerings를 참조하세요.

예약 노드 제품에 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 ElastiCache API 참조에서 DescribeReservedCacheNodesOfferings를 참조하세요.

예약 노드 구입

다음 예제에서는 AWS Management Console, AWS CLI 및 ElastiCache API를 사용하여 예약 노드 제품을 구입하는 방법을 설명합니다.

> **Important**
>
> 이 섹션의 예제에 따라 예약할 수 없는 AWS 계정에 요금이 부과됩니다.

주제

- 예약 노드 구매(콘솔)
- 예약 노드 구입(AWS CLI)
- 예약 노드 구입(ElastiCache API)

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 Console을 사용하여 제품 ID로 예약 노드 제품을 구입합니다.

예약 노드를 구입하려면

1. AWS Management Console에 로그인하고 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(Engine)에서 Memcached를 선택합니다.
5. 사용 가능한 제품을 확인하려면 다음 옵션 중에서 선택합니다.
   - 노드 유형(Node Type)
   - 기간
   - 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 선택합니다.

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 또는 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 AWS CLI 참조에서 purchase-reserved-cache-nodes-offering을 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 ElastiCache API 참조에서 PurchaseReservedCacheNodesOffering을 참조하세요.

예약 노드에 대한 정보 가져오기

AWS Management Console, AWS CLI 및 ElastiCache API를 사용하여 구입한 예약 노드에 대한 정보를 가져올 수 있습니다.

주제

- 예약 노드에 대한 정보 가져오기(콘솔)
- 예약 노드에 대한 정보 가져오기(AWS CLI)
- 예약 노드에 대한 정보 가져오기(ElastiCache API)

예약 노드에 대한 정보 가져오기(콘솔)

다음 절차에서는 AWS Management Console을 사용하여 구입한 예약 노드에 대한 정보를 가져오는 방법을 설명합니다.

구입한 예약 노드에 대한 정보를 가져오려면

1. AWS Management Console에 로그인하고 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 계정의 예약 노드에 대한 정보를 가져오려면 명령 프롬프트에 명령을 입력하세요.

```
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",

    "Duration": "31536000",
```

```
    "ProductDescription": "memcached",
    "OfferingType": "Medium Utilization",
    "MaxRecords": 0
}
```

자세한 내용은 AWS CLI 참조에서 describe--reserved-cache-nodes를 참조하세요.

예약 노드에 대한 정보 가져오기(ElastiCache API)

AWS 계정의 예약 노드에 대한 정보를 가져오려면 DescribeReservedCacheNodes 작업을 호출하세요.

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 ElastiCache API 참조에서 DescribeReservedCacheNodes를 참조하세요.

이전 세대 노드 마이그레이션

이전 세대 노드는 단계적으로 제거되고 있는 노드 유형입니다. 이전 세대 노드 유형을 사용하는 기존 클러스터가 없는 경우 ElastiCache는 해당 노드 유형의 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. ElastiCache에서 지원되는 노드 유형 및 이전 세대 노드 유형의 전체 목록에 대해서는 지원되는 노드 유형 섹션을 참조하세요.

Memcached 클러스터에서 노드 마이그레이션

ElastiCache for Memcached를 다른 노드 유형으로 마이그레이션하려면 새 클러스터를 생성해야 하며, 이 클러스터는 애플리케이션이 채울 수 있도록 항상 비어 있는 상태로 시작해야 합니다.

ElastiCache 콘솔을 사용하여 ElastiCache for 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 클러스터 엔드포인트 찾기(콘솔) 섹션을 참조하세요.
- 이전 클러스터를 삭제합니다. 자세한 내용은 클러스터 삭제 단원을 참조하십시오.

함께 일하기 ElastiCache

이 섹션에서는 ElastiCache 구현의 다양한 구성 요소를 관리하는 방법에 대한 세부 정보를 찾을 수 있습니다.

주제

- 스냅샷 및 복원
- 엔진 버전 및 업그레이드
- ElastiCache 모범 사례 및 캐싱 전략
- 자체 설계된 클러스터 관리
- ElastiCache Memcached를 위한 스케일링
- ElastiCache 리소스에 태그 지정
- Amazon ElastiCache의 Well-Architected Lens 사용
- 일반적인 문제 해결 단계 및 모범 사례
- 추가 문제 해결 단계

스냅샷 및 복원

서버리스 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 서버리스 Memcached 캐시를 생성하고 백업 데이터로 채워 데이터를 복원할 수 있습니다. 를 사용하면 ElastiCache, AWS Command Line Interface (AWS CLI) 및 API를 사용하여 백업을 관리할 수 있습니다. AWS Management Console ElastiCache

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

- 백업 제약 조건
- 자동 백업 예약

- 수동 백업 지원
- 최종 백업 생성
- 백업 설명
- 백업 복사
- 백업에서 새 캐시로 복원
- 백업 삭제
- 백업 태그 지정

백업 제약 조건

백업을 계획하거나 만들려는 경우 다음 제약 조건을 고려해야 합니다.

- 백업 및 복원은 Redis 또는 서버리스 Memcached에서 실행되는 캐시에만 지원됩니다.
- 연속 24시간 동안에는 서버리스 캐시당 최대 24개의 수동 백업을 생성할 수 있습니다.
- 백업 프로세스 중에는 서버리스 캐시에서 다른 API 또는 CLI 작업을 실행할 수 없습니다.

자동 백업 예약

모든 Memcached 서버리스 캐시에 대해 자동 백업을 활성화할 수 있습니다. 자동 백업이 활성화되면 캐시의 백업이 매일 ElastiCache 생성됩니다. 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 정보는 백업에서 새 캐시로 복원을 참조하세요.

자동 백업을 ElastiCache 예약하면 언제부터 백업 생성이 시작됩니다. 가장 편리한 시간에 백업 기간을 설정할 수 있습니다. 백업 기간을 지정하지 않으면 백업 기간이 자동으로 ElastiCache 할당됩니다.

Memcached Serverless 캐시를 생성할 때 ElastiCache 콘솔, 또는 API를 사용하여 자동 백업을 활성화하거나 비활성화할 수 있습니다. AWS CLI ElastiCache 이 작업은 고급 Memcached 설정 섹션에서 자동 백업 활성화 상자를 선택하여 수행할 수 있습니다.

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 CLI, 또는 ElastiCache API를 사용하여 캐시의 수동 백업을 생성할 수 있습니다.

수동 백업 생성(콘솔)

캐시의 백업을 생성하려면 다음과 같이 하세요(콘솔).

1. AWS Management Console 로그인하고 https://console.aws.amazon.com/ec2/ 에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창에서 Memcached 캐시를 선택합니다.
3. 백업하려는 캐시 이름 왼쪽에 있는 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 캐시를 수동으로 백업합니다. AWS CLI

를 사용하여 캐시의 수동 백업을 생성하려면 다음 매개 변수와 함께 `create-serverless-snapshot` AWS CLI 작업을 사용하십시오. AWS CLI

- `--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
```

관련 주제

자세한 내용은 AWS CLI 명령 참조의 create-snapshot을 참조하세요.

최종 백업 생성

ElastiCache 콘솔 AWS CLI, 또는 ElastiCache API를 사용하여 최종 백업을 생성할 수 있습니다.

최종 백업 생성(콘솔)

콘솔을 사용하여 Memcached 서버리스 캐시를 삭제할 때 최종 백업을 생성할 수 있습니다. ElastiCache

캐시를 삭제할 때 최종 백업을 생성하려면 삭제 대화 상자의 백업 생성에서 [예] 를 선택하고 백업 이름을 지정합니다.

관련 주제

- AWS Management Console 사용

최종 백업 생성(AWS CLI)

를 사용하여 캐시를 삭제할 때 최종 백업을 생성할 수 AWS CLI있습니다.

주제

- 서버리스 캐시를 삭제하는 경우

서버리스 캐시를 삭제하는 경우

최종 백업을 생성하려면 다음 매개 변수와 함께 `delete-serverless-cache` AWS CLI 작업을 사용하십시오.

- `--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 AWS CLI 명령 참조의 delete-serverless-cache를 참조하세요.

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. https://console.aws.amazon.com/elasticache/ 에서 AWS Management Console 로그인하고 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 목록을 표시하고 필요에 따라 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 명령 참조의 설명 서버리스 캐시 스냅샷을 참조하십시오. AWS CLI

백업 복사

자동 또는 수동으로 백업을 생성했을 때 모든 백업 복사본을 생성할 수 있습니다.

다음 단계에서는 백업을 복사하는 방법을 보여 줍니다.

백업 복사(콘솔)

백업을 복사하려면(콘솔)

1. https://console.aws.amazon.com/elasticache/ 에서 AWS Management Console 로그인하고 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.

백업에서 새 캐시로 복원

기존 백업을 새 서버리스 캐시 .

서버리스 캐시로 백업 복원(콘솔)

서버리스 캐시로 백업을 복원하려면 다음과 같이 하세요(콘솔).

1. AWS Management Console 로그인하고 https://console.aws.amazon.com/elasticache/ 에서 콘솔을 엽니다. ElastiCache
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.

서버리스 캐시로 백업 복원(AWS CLI)

서버리스 캐시로 백업을 복원하려면 다음과 같이 하세요(AWS CLI).

다음 AWS CLI 예제는 백업에서 데이터를 사용하여 `create-serverless-cache` 새 캐시를 생성하고 데이터를 가져옵니다.

Linux, macOS, Unix의 경우:

Windows의 경우:

```
aws elasticache create-serverless-cache \
    --serverless-cache-name CacheName \
    --engine memcached
    --snapshot-arns-to-restore Snapshot-ARN
```

Windows의 경우

```
aws elasticache create-serverless-cache ^
    --serverless-cache-name CacheName ^
    --engine memcached ^
    --snapshot-arns-to-restore Snapshot-ARN
```

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 백업이 자동 생성되었는지 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 CLI, 또는 ElastiCache API를 사용하여 백업을 삭제할 수 있습니다.

백업 삭제(콘솔)

다음 절차는 ElastiCache 콘솔을 사용하여 백업을 삭제합니다.

백업 삭제

1. https://console.aws.amazon.com/elasticache/ 에서 AWS Management Console 로그인하고 ElastiCache 콘솔을 엽니다.
2. 탐색 창에서 백업을 선택합니다.

   백업 화면에 백업 목록이 나타납니다.
3. 삭제할 백업의 이름 왼쪽에 있는 상자를 선택합니다.
4. 삭제를 선택합니다.
5. 이 백업을 삭제하려면 [Delete Backup] 확인 화면에서 [Delete]를 선택합니다. 상태가 [deleting]으로 변경됩니다.

서버리스 백업 삭제(AWS CLI)

다음 매개 변수와 함께 스냅샷 삭제 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 AWS CLI 명령 참조의 delete-serverless-cache-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 정보는 태그 지정이 가능한 리소스을 참조하세요.

비용 할당 태그는 청구서의 비용을 태그 값별로 그룹화하여 여러 AWS 서비스의 비용을 추적하는 방법입니다. 비용 할당 태그에 대해 자세히 알아보려면 비용 할당 태그 사용을 참조하세요.

ElastiCache 콘솔 AWS CLI, 또는 ElastiCache API를 사용하여 백업에 비용 할당 태그를 추가, 나열, 수정, 제거 또는 복사할 수 있습니다. 자세한 내용은 비용 할당 태그를 사용하여 비용을 모니터링합니다.을(를) 참조하세요.

엔진 버전 및 업그레이드

이 섹션에서는 지원되는 Memcached 엔진 버전과 이를 업그레이드하는 방법을 설명합니다.

주제

- 지원되는 ElastiCache for Memcached 버전
- 엔진 버전 및 업그레이드
- 엔진 버전 업그레이드 방법

지원되는 ElastiCache for Memcached 버전

ElastiCache는 다음 Memcached 버전과 새 버전으로의 업그레이드를 지원합니다. 새 버전으로 업그레이드할 때 충족되지 않을 경우 업그레이드가 실패하는 조건에 주의를 기울이십시오.

ElastiCache for Memcached 버전

- 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 for Memcached에 Memcached 버전 1.6.22에 대한 지원이 추가되었습니다. 새로운 기능은 없지만 Memcached 1.6.18의 버그 수정 및 누적 업데이트가 포함되어 있습니다.

자세한 내용은 GitHub의 Memcached에서 ReleaseNotes1622를 참조하세요.

Memcached 버전 1.6.17

ElastiCache for Memcached에 Memcached 버전 1.6.17에 대한 지원이 추가되었습니다. 새로운 기능은 없지만 Memcached 1.6.17의 버그 수정 및 누적 업데이트가 포함되어 있습니다.

자세한 내용은 GitHub의 Memcached에서 ReleaseNotes1617을 참조하세요.

Memcached 버전 1.6.12

ElastiCache for Memcached에 Memcached 버전 1.6.12 및 전송 중 암호화 지원이 추가되었습니다. Memcached 1.6.6부터의 버그 해결 및 누적 업데이트도 포함되었습니다.

자세한 내용은 GitHub의 Memcached에서 ReleaseNotes1612를 참조하세요.

Memcached 버전 1.6.6

ElastiCache for Memcached에 Memcached 버전 1.6.6에 대한 지원이 추가되었습니다. 새로운 기능은 없지만 Memcached 1.5.16의 버그 수정 및 누적 업데이트가 포함되어 있습니다. ElastiCache for Memcached에는 Extstore에 대한 지원이 포함되어 있지 않습니다.

자세한 내용은 GitHub의 Memcached에서 ReleaseNotes166을 참조하세요.

Memcached 버전 1.5.16

ElastiCache for Memcached에 Memcached 버전 1.5.16에 대한 지원이 추가되었습니다. 새로운 기능은 없지만 Memcached 1.5.14 및 Memcached 1.5.15의 버그 수정 및 누적 업데이트가 포함되어 있습니다.

자세한 내용은 GitHub의 Memcached에서 Memcached 1.5.16 릴리스 정보를 참조하세요.

Memcached 버전 1.5.10

ElastiCache for Memcached 버전 1.5.10은 다음 Memcached 기능을 지원합니다.

- 자동화된 슬래브 재분배 기능.
- `murmur3` 알고리즘으로 더 빠른 해시 테이블 조회.
- 세분화된 LRU 알고리즘.
- background-reclaim 메모리에 대한 LRU 크롤러.
- `--enable-seccomp`: 컴파일 시간 옵션.

또한 `no_modern` 및 `inline_ascii_resp` 파라미터를 도입합니다. 자세한 내용은 Memcached 1.5.10 파라미터 변경 섹션을 참조하세요.

Memcached용 ElastiCache 버전 1.4.34부터 추가된 Memcached 개선 사항에는 다음이 포함됩니다.

- ASCII multigets, CVE-2017-9951 및 `metadumper`에 대한 크롤링 한도와 같은 누적 방식 수정.
- 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 GitHub의 Memcached에서 Memcached 1.5.10 릴리스 정보를 참조하세요.

Memcached 버전 1.4.34

ElastiCache for Memcached 버전 1.4.34는 버전 1.4.33에 대해 새 기능이 추가되지 않았습니다. 버전 1.4.34는 일반적인 릴리스보다 큰 버그 수정 릴리스입니다.

자세한 내용은 GitHub의 Memcached에서 Memcached 1.4.34 릴리스 정보를 참조하세요.

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 통한 LRU(가장 오랫동안 사용되지 않음) 관리.
- 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 for Memcached가 지원하는 초기 엔진 및 버전이었습니다.

엔진 버전 및 업그레이드

메이저 버전은 API 비호환 변경 사항을 위한 것이고 마이너 버전은 이전 버전과 호환되는 방식으로 추가된 새로운 기능을 위한 것입니다. 패치 버전은 이전 버전과 호환되는 버그 수정 및 비기능 변경을 위한 것입니다.

ElastiCache 서버리스용 버전 관리

ElastiCache 서버리스는 애플리케이션에 영향을 미치거나 가동 중지를 일으키지 않고 최신 마이너 및 패치 소프트웨어 버전을 캐시에 자동으로 적용합니다. 여러분은 아무 작업도 수행할 필요가 없습니다.

새 메이저 버전이 사용 가능하면 ElastiCache 서버리스는 콘솔에서 알림을 보내고 EventBridge에서 이벤트를 보냅니다. 콘솔, CLI 또는 API를 사용하여 캐시를 수정하고 최신 엔진 버전을 선택하여 캐시를 최신 메이저 버전으로 업그레이드할 수 있습니다.

자체 설계된 ElastiCache 클러스터의 버전 관리

자체 설계된 ElastiCache 클러스터로 작업하는 경우 ElastiCache에서 지원되는 새 버전으로 캐시 클러스터를 실행하는 소프트웨어를 업그레이드할 때 제어할 수 있습니다. 캐시를 사용 가능한 최신 메이저, 마이너, 패치 버전으로 업그레이드할 시기를 제어할 수 있습니다. 클러스터 또는 복제 그룹을 수정하고 새 엔진 버전을 지정하여 엔진 버전 업그레이드를 시작합니다.

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

버전 업그레이드에는 약간의 호환성 위험이 있을 수 있으므로 업그레이드가 자동으로 이루어지지 않기 때문에 업그레이드는 사용자가 시작해야 합니다.

최신 Memcached 버전으로 업그레이드하려면 사용하려는 새 엔진 버전을 지정하여 캐시 클러스터를 수정합니다. 최신 Memcached 버전으로의 업그레이드는 안전하지 않은 프로세스로, 데이터가 손상되고 콜드 캐시로 시작합니다. 자세한 정보는 클러스터 수정을 참조하세요.

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 섹션을 참조하세요.

자체 설계된 클러스터를 사용할 때의 업그레이드 고려 사항

> **Note**
>
> 다음 고려 사항은 자체 설계된 클러스터를 업그레이드할 때만 적용됩니다. 이 내용은 ElastiCache 서버리스에는 적용되지 않습니다.

자체 설계된 클러스터를 업그레이드할 때만 다음 내용을 고려합니다.

- 엔진 버전 관리는 패치 발생 방법을 최대한 제어할 수 있도록 설계되었습니다. 그러나 ElastiCache는 시스템 또는 캐시 소프트웨어에 심각한 보안 취약성이 발견되는 등 발생할 가능성이 거의 없는 이벤트의 경우 사용자를 대신하여 클러스터에 패치를 적용할 수 있는 권한을 보유합니다.
- Memcached 엔진은 지속성을 지원하지 않으므로 Memcached 엔진 버전 업그레이드는 항상 클러스터에서 모든 캐시 데이터를 지우는 방해가 되는 프로세스입니다.

엔진 버전 업그레이드 방법

클러스터로 버전 업그레이드를 시작하려면 이를 수정하고 새 엔진 버전을 지정합니다. ElastiCache 콘솔, AWS CLI 또는 ElastiCache API를 사용하여 이 작업을 수행할 수 있습니다.

- AWS Management Console을 사용하려면 콘솔을 통한 클러스터 수정을 참조하세요.
- AWS CLI를 사용하려면 CLI를 사용한 클러스터 수정을 참조하세요.
- ElastiCache API를 사용하려면 API를 통한 클러스터 수정을 참조하세요.

엔진 버전 업그레이드 방법

클러스터로 버전 업그레이드를 시작하려면 이를 수정하고 새 엔진 버전을 지정합니다. ElastiCache 콘솔, AWS CLI 또는 ElastiCache API를 사용하여 이 작업을 수행할 수 있습니다.

- AWS Management Console 사용 시에는 사용 AWS Management Console 섹션을 참조하세요.
- AWS CLI 사용 시에는 사용: AWS CLI 섹션을 참조하세요.
- ElastiCache API를 사용하려면 ElastiCache API 사용 섹션을 참조하세요.

ElastiCache 모범 사례 및 캐싱 전략

아래에서 Amazon에 대한 권장 모범 사례를 확인할 수 ElastiCache 있습니다. 다음 모범 사례를 준수하면 클러스터의 성능과 신뢰성을 향상시킬 수 있습니다.

주제

- Memcached 클라이언트 사용 모범 사례
- 지원되는 Memcached 명령
- 캐싱 전략

Memcached 클라이언트 사용 모범 사례

일반적으로 사용되는 오픈 소스 Memcached 클라이언트 라이브러리로 ElastiCache 리소스와 상호작용하는 모범 사례를 알아보려면 다음 주제를 참조하세요.

주제

- 효율적인 로드 밸런싱을 위해 ElastiCache 클라이언트 구성
- IPv6 클라이언트 예시

효율적인 로드 밸런싱을 위해 ElastiCache 클라이언트 구성

> **Note**
>
> 이 섹션의 내용은 자체 설계된 다중 노드 Memcached 클러스터에 적용됩니다.

여러 ElastiCache Memcached 노드를 효과적으로 사용하려면 캐시 키를 노드에 분산시킬 수 있어야 합니다. n 노드 사용 클러스터를 로드 밸런스하는 간단한 방법은 객체 키의 해시를 계산하고 n - hash(key) mod n으로 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 2개로 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, PHP 또는 .NET으로 작업하는 경우 Amazon ElastiCache 클라이언트 라이브러리 중 하나를 사용하는 것이 좋습니다.

Java를 사용한 일관적 해싱

ElastiCache Memcached Java 클라이언트는 일관적 해싱 기능이 내장된 오픈 소스 spymemcached Java 클라이언트를 기반으로 합니다. 일관적 해싱을 구현하는 KetamaConnectionFactory 클래스가 라이브러리에 포함되어 있습니다. 기본적으로 spymemcached에서는 일관적 해싱이 해제됩니다.

자세한 내용은 KetamaConnectionFactory에서 KetamaConnectionFactory 설명서를 참조하세요.

PHP를 사용한 일관적 해싱

ElastiCache Memcached PHP 클라이언트는 기본 제공 Memcached PHP 라이브러리 주변의 래퍼입니다. 기본적으로 Memcached PHP 라이브러리에서는 일관적 해싱이 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.php에서 Memcached PHP의 런타임 구성 설명서를 참조하세요. 특히 `memcached.sess_consistent_hash` 파라미터에 유의하시기 바랍니다.

.NET을 사용한 일관적 해싱

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-memcachedlocator에 있는 `memcached/locator` 설명서를 참조하세요.

IPv6 클라이언트 예시

> **Note**
>
> 이 섹션의 내용은 자체 설계 Memcached 클러스터에 적용됩니다.

ElastiCache 오픈 소스 Memcached와 호환됩니다. 즉, IPv6 연결을 지원하는 Memcached용 오픈 소스 클라이언트는 Memcached 클러스터에 사용할 수 있는 IPv6에 연결할 수 있어야 합니다. ElastiCache 또한 다음 클라이언트는 지원되는 모든 네트워크 유형 구성에서 작동한다고 특별 검증되었습니다.

다음은 일반적으로 사용되는 오픈 소스 클라이언트 라이브러리를 사용하여 IPv6 지원 리소스와 상호 작용하는 모범 사례입니다. ElastiCache 상호 작용에 대한 기존 모범 사례에서 리소스용 클라이언트 구성에 ElastiCache 대한 권장 사항을 확인할 수 있습니다. ElastiCache 하지만 IPv6 활성화 리소스와 상호 작용할 때 주의해야 할 점이 몇 가지 있습니다.

검증된 클라이언트

검증된 클라이언트:

- AWS ElastiCache Php용 Memcached 클러스터 클라이언트 — 버전 *3.6.2
- AWS ElastiCache 자바용 Memcached 클러스터 클라이언트 — Github의 최신 마스터

듀얼 스택 클러스터에 선호되는 프로토콜 구성

Memcached 클러스터의 경우, 클라이언트가 클러스터 내 노드에 연결하는 데 사용할 프로토콜을 IP Discovery 파라미터로 제어할 수 있습니다. IP Discovery 파라미터는 IPv4 또는 IPv6로 설정할 수 있습니다.

IP Discovery 파라미터는 config get 클러스터 출력에 사용되는 IP 프로토콜을 제어합니다. 그러면 Memcached 클러스터의 자동 검색을 지원하는 클라이언트가 사용하는 IP 프로토콜이 결정됩니다. ElastiCache

IP Discovery를 변경해도 연결된 클라이언트는 가동 중지되지 않습니다. 하지만 변경 사항이 전파되려면 다소 시간이 소요됩니다.

Java의 경우 `getAvailableNodeEndPoints` 출력을, Php의 경우 `getServerList` 출력을 모니터링합니다. 이들 함수의 출력에서, 업데이트된 프로토콜을 사용하는 클러스터 내 모든 노드에 해당하는 IP를 보고하면, 변경 사항이 완전히 전파된 것입니다.

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.

TLS를 지원하는 이중 스택 클러스터 ElastiCache

ElastiCache 클러스터에 TLS가 활성화된 경우 클러스터 검색 함수 는 IP 대신 호스트 이름을 `config get cluster`반환합니다. 그러면 IP 대신 호스트 이름을 사용하여 ElastiCache 클러스터에 연결하고 TLS 핸드셰이크를 수행합니다. 따라서 클라이언트가 IP Discovery 파라미터의 영향을 받지 않습니다. TLS 활성화 클러스터의 경우, IP Discovery 파라미터는 선호 IP 프로토콜에 영향을 끼치지 않습니다. 대신 클라이언트가 DNS 호스트 이름을 확인할 때 어떤 IP 프로토콜을 선호하는지에 따라, 사용되는 IP 프로토콜이 결정됩니다.

Java 클라이언트

IPv4와 IPv6를 모두 지원하는 Java 환경에서 연결할 때, Java는 기본적으로 이전 버전과의 호환성을 위해 IPv6보다 IPv4를 선호합니다. 하지만 JVM 인수를 통해 IP 프로토콜 기본 설정을 구성할 수 있습니다. IPv4를 선호하려면 JVM에 `-Djava.net.preferIPv4Stack=true`를, IPv6를 선호하려면 `-Djava.net.preferIPv6Stack=true`를 설정합니다. `-Djava.net.preferIPv4Stack=true`를 설정하면 JVM이 더 이상 IPv6 연결을 만들지 않습니다.

호스트 수준 기본 설정

일반적으로 클라이언트 또는 클라이언트 런타임에 IP 프로토콜 기본 설정을 위한 구성 옵션이 제공되지 않으면, DNS 확인을 수행할 때 IP 프로토콜은 호스트의 구성을 기반으로 작동합니다. 기본적으로, 대부분의 호스트는 IPv4보다 IPv6를 선호하지만 이 기본 설정은 호스트 수준에서 구성할 수 있습니다. 이는 클러스터에 대한 요청뿐 아니라 해당 호스트의 모든 DNS 요청에도 영향을 미칩니다. ElastiCache

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. 클러스터에 연결할 때 IPv4를 선호하도록 구성을 업데이트하려면 ElastiCache 클러스터 IP를 포함하는 CIDR 범위의 우선 순위를 기본 IPv6 연결의 우선 순위보다 높게 업데이트하십시오. IPv6 연결의 우선 순위는 기본적으로 40입니다. 예를 들어, 클러스터가 CIDR이 172.31.0.0:0/16인 서브넷에 위치할 때는 아래 구성으로 인해 클라이언트는 해당 클러스터에 IPv4 연결을 선호하게 됩니다.

```
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.conf에 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 범위에 IPv6 연결보다 IPv4 연결을 선호하도록 기본 설정 정책이 업데이트됩니다. 예를 들어, 클러스터가 CIDR이 172.31.0.0:0/16인 서브넷에 위치할 때 `netsh interface ipv6 set prefix ::ffff:172.31.0.0:0/112 100 15`를 실행하면 다음 우선 순위 테이블로 인해 클라이언트는 해당 클러스터에 IPv4 연결을 선호하게 됩니다.

```
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
```

지원되는 Memcached 명령

ElastiCache Memcached용 서버리스는 다음을 제외하고 오픈 소스 memcached 1.6의 모든 memcached 명령을 지원합니다.

- 클라이언트를 연결하려면 TLS가 필요하므로 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 LRU 크롤러 설정 수정
  - watch - memcached 서버 로그 감시
  - verbosity - 서버 로그 수준 구성
  - me- 메타 디버그 (me) 명령은 지원되지 않습니다.

캐싱 전략

다음 항목에서는 캐시를 채우고 유지 관리하기 위한 전략을 확인할 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
- 라이트-스루
- TTL 추가
- 관련 주제

지연 로딩

이름에서 알 수 있듯이 지연 로딩은 필요할 때에만 데이터를 캐시에 로드하는 캐싱 전략입니다. 다음 설명과 같이 작동합니다.

Amazon ElastiCache는 액세스하는 애플리케이션과 데이터 스토어(데이터베이스) 사이에 위치하는 인메모리 키/값 저장소입니다. 애플리케이션에서 데이터를 요청할 때마다 ElastiCache 캐시에 먼저 요청합니다. 데이터가 캐시에 있으며 최신 상태인 경우 ElastiCache가 데이터를 애플리케이션에 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 TTL 추가 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. TTL(Time to Live) 값을 추가하면 낭비되는 공간을 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"})
```

TTL 추가

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 TTL(Time to Live) 값을 추가하면 각 전략의 이점을 얻을 수 있습니다. 동시에 추가 데이터로 캐시를 복잡하게 만들지 않을 수 있습니다.

TTL(Time To Live)은 키가 만료될 때까지의 시간(초)을 지정하는 정수 값입니다. 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 또는 Memcached `set`명령을 참조하세요.

TTL 의사 코드 예제

다음 코드는 TTL을 통한 라이트-스루 로직의 의사(pseudo) 코드 예제입니다.

```
// ****************************************
// 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 TTL을 통한 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.
// 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
- ElastiCache Memcached를 위한 스케일링

자체 설계된 클러스터 관리

이 섹션에는 자체 설계된 클러스터를 관리하는 데 도움이 되는 주제가 포함되어 있습니다.

> **Note**
>
> 이러한 주제는 ElastiCache 서버리스에는 적용되지 않습니다.

주제

- 유지 관리 관리 중
- 파라미터 그룹을 사용해 엔진 파라미터 구성

유지 관리 관리 중

모든 클러스터에는 시스템 변경 내용이 적용되는 주 단위 유지 관리 기간이 있습니다. 클러스터를 생성하거나 수정할 때 선호하는 유지 관리 기간을 지정하지 않는 경우, 해당 지역의 유지 관리 기간 내에 무작위로 선택한 요일에 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	Asia Pacific (Tokyo) Region	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
ap-south-1	Asia Pacific (Mumbai) Region	17:30~1:30 UTC
ap-southeast-1	Asia Pacific (Singapore) Region	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

리전 코드	리전 이름	리전 유지 관리 기간
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) 지역	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

클러스터의 유지 관리 기간 변경

유지 관리 기간은 사용률이 가장 낮은 시간에 할당되어야 하므로 수시로 수정되어야 할 수 있습니다. 클러스터를 수정하여 요청한 유지 관리 활동이 이루어지는 기간을 최대 24시간까지 지정할 수 있습니다. 이 시간 동안 사용자가 요청한 지연된 또는 대기 중인 클러스터 수정이 발생합니다.

> **Note**
>
> Apply now 상자를 사용하여 노드 유형 수정 및/또는 엔진 업그레이드를 즉시 적용하려면 Apply now 상자를 AWS Management Console 선택하십시오. 그러지 않으면 이러한 수정 사항은 다음 예약된 유지 관리 기간에 적용됩니다. API를 사용하려면 modify-replication-group또는 을 참조하십시오 modify-cache-cluster.

추가 정보

유지 관리 기간 및 노드 대체에 대한 자세한 내용은 다음을 참조하세요.

- ElastiCache 유지 관리 —유지 관리 및 노드 교체에 대한 FAQ
- 노드 교체 - 노드 교체 관리

- 클러스터 수정 ElastiCache - 클러스터의 유지 관리 기간 변경

파라미터 그룹을 사용해 엔진 파라미터 구성

Amazon ElastiCache는 파라미터를 사용하여 노드 및 클러스터의 런타임 속성을 제어합니다. 일반적으로 최신 엔진 버전에는 새로운 기능을 기원하는 추가 파라미터가 포함됩니다. 파라미터가 정리된 표는 Memcached 특정 파라미터 섹션을 참조하세요.

`maxmemory`와 같은 일부 파라미터 값은 엔진 및 노드 유형에 의해 결정됩니다. 노드 유형별 파라미터 값의 표는 Memcached 노드 유형별 파라미터 섹션을 참조하세요.

> **Note**
>
> Memcached 특정 파라미터 목록은 Memcached 특정 파라미터를 참조하세요.

주제

- 파라미터 관리
- 캐시 파라미터 그룹 티어
- 파라미터 그룹 생성
- 이름별로 파라미터 그룹 목록 조회
- 파라미터 그룹의 값 목록 조회
- 파라미터 그룹 수정
- 파라미터 그룹 삭제
- Memcached 특정 파라미터

파라미터 관리

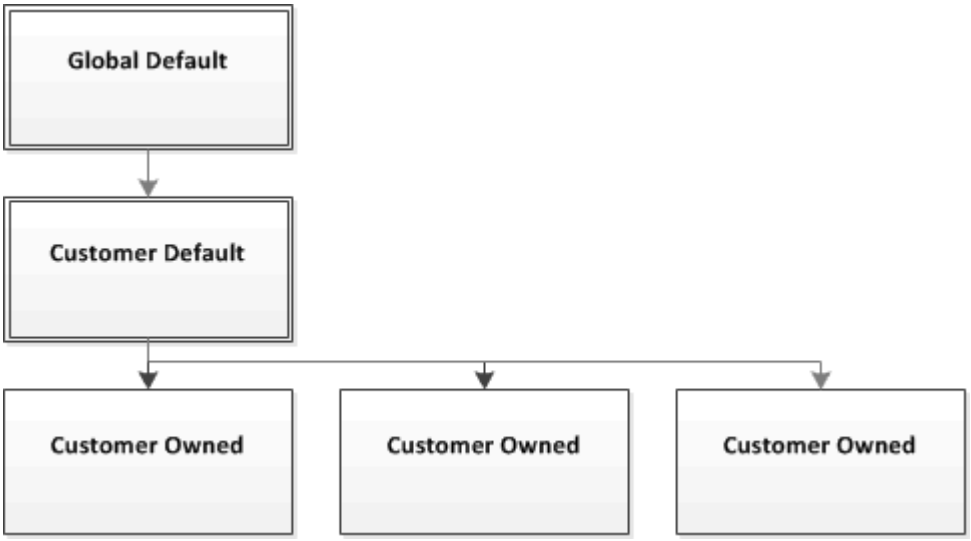
더욱 쉬운 파라미터 관리를 위해 파라미터를 명명된 파라미터 그룹으로 그룹화합니다. 파라미터 그룹은 시작하는 동안 엔진 소프트웨어에 전달되는 파라미터의 특정 값 조합을 나타냅니다. 이 값은 각 노드의 엔진 프로세서가 런타임에 작동하는 방식을 결정합니다. 특정 파라미터 그룹의 파라미터 값은 해당 파라미터가 속한 클러스터와 상관없이 그룹과 연결된 모든 노드에 적용됩니다.

클러스터 성능을 미세 조정하려면 일부 파라미터 값을 수정하거나 클러스터의 파라미터 그룹을 변경할 수 있습니다.

- 기본 파라미터 그룹을 수정하거나 삭제할 수 없습니다. 사용자 지정 파라미터 값이 필요하면 사용자 지정 파라미터 그룹을 생성해야 합니다.
- 파라미터 그룹 패밀리와 할당할 클러스터는 호환 가능해야 합니다. 예를 들어 클러스터에서 Memcached 버전 1.4.8을 실행 중이라면 Memcached 1.4 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.
- 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.

캐시 파라미터 그룹 티어

Amazon ElastiCache에는 다음과 같이 세 가지 캐시 파라미터 그룹 티어가 있습니다.



Amazon ElastiCache 파라미터 그룹 티어

전역 기본값

해당 리전의 모든 Amazon ElastiCache 고객을 위한 최상위 루트 파라미터 그룹입니다.

전역 기본 캐시 파라미터 그룹:

- ElastiCache용으로 예약되어 있으며 고객이 사용할 수 없습니다.

고객 기본값

고객의 사용을 위해 생성된 전역 기본 캐시 파라미터 그룹의 사본입니다.

고객 기본 캐시 파라미터 그룹:

- ElastiCache가 생성하고 소유합니다.
- 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. 자세한 내용은 Memcached 특정 파라미터 섹션을 참조하세요.

파라미터 그룹 생성

기본값에서 변경하려는 파라미터 값이 하나 이상이면 새 파라미터 그룹을 생성해야 합니다. ElastiCache 콘솔, AWS CLI 또는 ElastiCache API를 사용하여 파라미터 그룹을 생성할 수 있습니다.

파라미터 그룹 생성(콘솔)

다음 절차는 ElastiCache 콘솔을 사용하여 파라미터 그룹을 생성하는 방법을 보여줍니다.

ElastiCache 콘솔을 사용하여 파라미터 그룹을 생성하려면

1. AWS Management Console에 로그인하고 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 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 파라미터 그룹 수정 섹션을 참조하세요.

파라미터 그룹 생성(AWS CLI)

AWS CLI를 사용하여 파라미터 그룹을 생성하려면 이 파라미터와 함께 `create-cache-parameter-group` 명령을 사용합니다.

- `--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 memcached1.4를 템플릿으로 사용하여 myMem14라는 파라미터 그룹을 생성합니다.

Linux, macOS 또는 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"
    }
}
```

파라미터 그룹을 생성하면 패밀리의 기본값이 부여됩니다. 기본값을 변경하려면 파라미터 그룹을 수정해야 합니다. 자세한 내용은 파라미터 그룹 수정 섹션을 참조하세요.

자세한 내용은 `create-cache-parameter-group` 섹션을 참조하세요.

파라미터 그룹 생성(ElastiCache API)

ElastiCache API를 사용하여 파라미터 그룹을 생성하려면 이 파라미터와 함께 `CreateCacheParameterGroup` 작업을 사용합니다.

- `ParameterGroupName` - 파라미터 그룹의 이름입니다.

  파라미터 그룹 명명 제약 조건은 다음과 같습니다.

  - ASCII 문자로 시작해야 합니다.
  - ASCII 문자, 숫자 및 하이픈만 포함할 수 있습니다.
  - 1-255자여야 합니다.
  - 하이픈 2개가 연속될 수 없습니다.

  - 끝에 하이픈이 올 수 없습니다.
- `CacheParameterGroupFamily` - 파라미터 그룹의 엔진 및 버전 패밀리입니다. 예: `memcached1.4`.
- `Description` - 사용자가 정의한 파라미터 그룹에 대한 설명입니다.

Example

다음 예제에서는 memcached1.4를 템플릿으로 사용하여 myMem14라는 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>
```

파라미터 그룹을 생성하면 패밀리의 기본값이 부여됩니다. 기본값을 변경하려면 파라미터 그룹을 수정해야 합니다. 자세한 내용은 파라미터 그룹 수정 섹션을 참조하세요.

자세한 내용은 `CreateCacheParameterGroup` 섹션을 참조하세요.

이름별로 파라미터 그룹 목록 조회

ElastiCache 콘솔, AWS CLI 또는 ElastiCache API를 사용하여 파라미터 그룹을 나열할 수 있습니다.

이름별로 파라미터 그룹 목록 조회(콘솔)

다음 절차에서는 ElastiCache 콘솔을 사용하여 파라미터 그룹 목록을 보는 방법을 설명합니다.

ElastiCache 콘솔을 사용하여 파라미터 그룹을 나열하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 사용 가능한 모든 파라미터 목록을 표시하려면 왼쪽 탐색 창에서 [Parameter Groups]를 선택합니다.

이름으로 파라미터 그룹 나열(AWS CLI)

AWS CLI를 사용하여 파라미터 그룹의 목록을 생성하려면 `describe-cache-parameter-groups` 명령을 사용합니다. 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 코드에는 myMem14 파라미터 그룹이 나열되어 있습니다.

Linux, macOS 또는 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 최대 10개의 파라미터 그룹을 나열합니다.

```
aws elasticache describe-cache-parameter-groups --max-records 10
```

이 명령의 JSON 출력은 각 파라미터 그룹의 이름, 패밀리, 설명 및 redis5.6의 경우 파라미터 그룹이 글로벌 데이터 스토어(IsGlobal)의 일부인지 여부를 나열하는 것과 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 섹션을 참조하세요.

이름으로 파라미터 그룹 나열(ElastiCache API)

ElastiCache API를 사용하여 파라미터 그룹의 목록을 생성하려면 `DescribeCacheParameterGroups` 작업을 사용합니다. 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 코드에는 myMem14 파라미터 그룹이 나열되어 있습니다.

```
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 그룹의 이름, 패밀리, 설명 및 redis5.6의 경우 파라미터 그룹이 글로벌 데이터 스토어(IsGlobal)의 일부인지 여부를 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>
```

자세한 내용은 DescribeCacheParameterGroups 섹션을 참조하세요.

파라미터 그룹의 값 목록 조회

ElastiCache 콘솔, AWS CLI 또는 ElastiCache API를 사용하여 파라미터 및 파라미터 그룹의 값을 나열할 수 있습니다.

파라미터 그룹의 값 목록 조회(콘솔)

다음 절차에서는 ElastiCache 콘솔을 사용하여 파라미터 및 파라미터 그룹의 값을 나열하는 방법을 보여줍니다.

ElastiCache 콘솔을 사용하여 파라미터 그룹의 파라미터 및 그 값을 나열하려면

1. AWS Management Console에 로그인하고 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)

AWS CLI를 사용하여 파라미터 그룹의 파라미터 및 그 값을 나열하려면 `describe-cache-parameters` 명령을 사용합니다.

Example

다음 샘플 코드에는 myMem14 파라미터 그룹의 모든 파라미터 및 그 값이 나열되어 있습니다.

Linux, macOS 또는 Unix의 경우:

```
aws elasticache describe-cache-parameters \
    --cache-parameter-group-name myMem14
```

Windows의 경우:

```
aws elasticache describe-cache-parameters ^
    --cache-parameter-group-name myMem14
```

자세한 내용은 describe-cache-parameters 섹션을 참조하세요.

파라미터 그룹 값 나열(ElastiCache API)

ElastiCache API를 사용하여 파라미터 그룹의 파라미터 및 그 값을 나열하려면 DescribeCacheParameters 작업을 사용합니다.

Example

다음 샘플 코드에는 myMem14 파라미터 그룹의 모든 파라미터가 나열되어 있습니다.

```
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>
```

자세한 내용은 `DescribeCacheParameters` 섹션을 참조하세요.

파라미터 그룹 수정

> **Important**
>
> 어떤 기본 파라미터 그룹도 수정할 수 없습니다.

파라미터 그룹의 일부 파라미터 값을 수정할 수 있습니다. 이 파라미터 값은 파라미터 그룹과 연결된 클러스터에 적용됩니다. 변경한 파라미터 값이 파라미터 그룹에 적용되는 시점에 관한 자세한 내용은 Memcached 특정 파라미터 섹션을 참조하세요.

파라미터 그룹 수정(콘솔)

다음 절차는 ElastiCache 콘솔을 사용하여 `binding_protocol` 파라미터 값을 변경하는 방법을 보여줍니다. 동일한 절차를 통해 모든 파라미터 값을 변경할 수 있습니다.

ElastiCache 콘솔을 사용하여 파라미터 값을 변경하려면

1. AWS Management Console에 로그인하고 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. 파라미터 그룹 편집: 화면에서 왼쪽 및 오른쪽 화살표로 스크롤하여 `cluster-enabled` 파라미터를 찾은 다음 값 열에 `yes`를 입력합니다.
7. 변경 사항 저장을 선택합니다.
8. 변경한 파라미터의 이름을 찾으려면 Memcached 특정 파라미터 섹션을 참조하세요. 다시 시작한 후 파라미터에 변경 사항이 발생하면 이 파라미터 그룹을 사용하는 모든 클러스터를 재부팅합니다. 자세한 내용은 클러스터 재부팅을 참조하세요.

파라미터 그룹 수정(AWS CLI)

AWS CLI를 사용하여 파라미터 값을 변경하려면 `modify-cache-parameter-group` 명령을 사용합니다.

Example

변경하려는 파라미터의 이름과 허용되는 값을 찾으려면 Memcached 특정 파라미터 섹션을 참조하세요.

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"
}
```

자세한 내용은 modify-cache-parameter-group 섹션을 참조하세요.

파라미터 그룹 수정(ElastiCache API)

ElastiCache API를 사용하여 파라미터 그룹의 파라미터 값을 변경하려면 ModifyCacheParameterGroup 작업을 사용합니다.

Example

변경하려는 파라미터의 이름과 허용되는 값을 찾으려면 Memcached 특정 파라미터 섹션을 참조하세요.

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>
```

자세한 내용은 ModifyCacheParameterGroup 섹션을 참조하세요.

파라미터 그룹 삭제

ElastiCache 콘솔, AWS CLI 또는 ElastiCache API를 사용하여 사용자 지정 파라미터 그룹을 삭제할 수 있습니다.

파라미터 그룹이 클러스터와 연결된 경우 삭제할 수 없습니다. 또한 기본 파라미터 그룹도 삭제할 수 없습니다.

파라미터 그룹 삭제(콘솔)

다음 절차는 ElastiCache 콘솔을 사용하여 파라미터 그룹을 삭제하는 방법을 보여줍니다.

ElastiCache 콘솔을 사용하여 파라미터 그룹을 삭제하려면

1. AWS Management Console에 로그인하고 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. 삭제를 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)

`delete-cache-parameter-group`를 사용하여 파라미터 그룹을 삭제하려면 AWS CLI 명령을 사용합니다. 삭제할 파라미터 그룹의 경우 `--cache-parameter-group-name`으로 지정된 파라미터 그룹에는 클러스터를 연결할 수 없으며 기본 파라미터 그룹이 될 수도 없습니다.

다음 샘플 코드에서는 myMem14 파라미터 그룹을 삭제합니다.

Example

Linux, macOS 또는 Unix의 경우:

```
aws elasticache delete-cache-parameter-group \
    --cache-parameter-group-name myMem14
```

Windows의 경우:

```
aws elasticache delete-cache-parameter-group ^
    --cache-parameter-group-name myMem14
```

자세한 내용은 delete-cache-parameter-group 섹션을 참조하세요.

파라미터 그룹 삭제(ElastiCache API)

ElastiCache API를 사용하여 파라미터 그룹을 삭제하려면 DeleteCacheParameterGroup 작업을 사용합니다. 삭제할 파라미터 그룹의 경우 CacheParameterGroupName으로 지정된 파라미터 그룹에는 클러스터를 연결할 수 없으며 기본 파라미터 그룹이 될 수도 없습니다.

Example

다음 샘플 코드에서는 myMem14 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>
```

자세한 내용은 DeleteCacheParameterGroup 섹션을 참조하세요.

Memcached 특정 파라미터

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 파라미터 그룹 생성 섹션을 참조하세요.

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 , slab_automove , lru_crawler , lru_maintainer , maxconns_fast 명령 비활성화를 위한 별칭. No modern은 hash_algorithm을 jenkins로 설정하고 ASCII VALUE의 인라이닝을 허용합니다. memcached 1.5 이상에 적용됩니다. 현대로 되돌리려면 이 파라미터를 사용 중지한 다음 다시 실행해야 합니다. 그러면 자동으로 slab_reassign , slab_automove , lru_crawler , lru_maintainer 및 maxconns_fast 가 활성화됩니다. **Note** 이 파라미터의 기본 구성 값은 2021년 8월 20일 현재 0에서 1로 변경되었습니다. 업데이트된 기본값은 2021년 8월 20일 이후 각 리전의 새로운 ElastiCache 사용자에 의해 자동으로 선택됩니다. 2021년 8월 20일 이전의 해당 리전에서는 기존 ElastiCache 사용자가 사용자 지정 파라미터 그룹을 수동으로 수정해야 이 새로

이름	Details	설명
		운 변경 사항이 적용됩니다.
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. ASCII get, faster 세트의 속도가 약간 느려집니다.

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 유형: boolean 수정 가능 여부: 예 (no_modern 으로 설정하는 경우 재시작해야 함) 허용된 값: 0,1	이 버전에서는 이제 지원하지 않습니다. 이 버전부터는 시작할 때마다 항상 또는 재시작 시 기본적으로 no-modern 이 활성화됩니다.

이름	Details	설명
	변경 적용: 시작 시	

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. `modern`을 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 유형: boolean 수정 가능 여부: 예 변경 적용: 즉시 사용자가 `watcher_logbuf_size` 및 `worker_lo`	로그 가져오기, 제거 또는 변형. 예를 들어 사용자가 `watch`를 켜면 `get`, `set`, `delete` 또는 `update` 발생 시 로그를 볼 수 있습니다.

이름	Details	설명
	gbuf_size 한도에 도달하면 로그가 삭제될 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.
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.

이름	Details	설명
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	flush_all을 비활성화하려면 파라미터(-F)를 추가합니다. 프로덕션 인

이름	Details	설명
	수정 가능 여부: 예 변경 적용: 시작 시	스턴스에서 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.

이름	Details	설명
`lru_crawler`	기본값: 0(비활성화) 유형: boolean 수정 가능 여부: 예 변경 적용: 재시작 후 **Note** 런타임 시 명령줄에서 `lru_crawler` 를 일시적으로 활성화할 수 있습니다. 자세한 내용은 설명 열을 참조하세요.	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. 값: 0,1 **Note** 명령줄에서 `lru_crawler` 를 활성화하면 명령줄에서 비활성화하거나 다음에 재부팅될 때까지 크롤러가 활성화됩니다. 영구적으로 활성화하려면 파라미터 값을 수정해야 합니다. 자세한 내용은 파라미터 그룹 수정 섹션을 참조하세요.

이름	Details	설명
`lru_maintainer`	기본값: 0(비활성화) 유형: boolean 수정 가능 여부: 예 변경 적용: 시작 시	용량에 도달할 때 LRU 간에 항목을 셔플링한 백그라운드 스레드입니다. 값: 0, 1
`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 처음 크기이며 2의 거듭 제곱으로 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. 파라미터를 1로 설정하면 ElastiCache가 클라이언트에 오류를 전송하고 즉시 연결을 끊습니다.

이름	설명
slab_automove	슬래브 오토무브 알고리즘을 조정합니다. 이 파라미터를 0(영)으로 설정하면 오토무브 알고리즘이 비활성화됩니다. 1로 설정하면 ElastiCache가 자동으로 슬래브를 이동하는 데 느리고 보수적인 접근 방식을 취합니다. 2로 설정하면 제거할 때마다 ElastiCache가 적극적으로 슬래브를 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 auto입니다. binding_protocol 의 값을 수정하는 방법에 대한 지침은 파라미터 그룹 수정을 참조하세요.
cas_disabled	기본값: 0(false) 유형: 부울 수정 가능 여부: 예 변경 적용: 재시작 후	1(true)이면 확인 및 설정(CAS) 작업이 비활성화되고 저장된 항목이 CAS가 활성화된 경우보다 8바이트 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)이면 ElastiCache가 더 큰 메모리 페이지를 사용하고자 합니다.
lock_down_paged_memory	기본값: 0(false) 유형: 부울 수정 가능 여부: 아니요	1(true)이면 ElastiCache가 페이징된 모든 메모리를 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)이면 ElastiCache가 핵심 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 사용하면 ElastiCache는 `memcached_connections_overhead` 값을 13MB로 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 VPC)에서 생성됩니다.

ElastiCache Memcached를 위한 스케일링

멤캐시드 ElastiCache 스케일링

ElastiCache 서버리스는 워크로드 트래픽이 증가하거나 감소할 때 워크로드 트래픽을 자동으로 수용합니다. 각 ElastiCache 서버리스 캐시에 대해 CPU, 메모리, 네트워크 등의 리소스 사용률을 ElastiCache 지속적으로 추적합니다. 이러한 리소스가 제한되면 ElastiCache 서버리스는 애플리케이션을 중단하지 않고 새 샤드를 추가하고 새 샤드에 데이터를 재분배하여 규모를 축소합니다. 캐시 데이터 스토리지에 대한 `BytesUsedForCache` 메트릭과 `ElastiCacheProcessingUnits` 컴퓨팅 사용량에 대한 지표 (ECPU) 를 CloudWatch 모니터링하여 캐시에서 소비되는 리소스를 모니터링할 수 있습니다.

비용 관리를 위한 규모 조정 한도 설정

캐시 데이터 스토리지와 초당 ECPU의 최대 사용량을 모두 구성하여 캐시 비용을 제어할 수 있습니다. 이렇게 하면 캐시 사용량이 구성된 최댓값을 초과하지 않도록 할 수 있습니다.

규모 조정 최댓값을 설정하면 캐시가 최댓값에 도달할 경우 애플리케이션의 캐시 성능이 저하될 수 있습니다. 캐시 데이터 스토리지의 최대값을 설정하고 캐시 데이터 스토리지가 최대값에 ElastiCache 도달하면 LRU 로직을 사용하여 캐시에 있는 데이터를 제거하기 시작합니다. 초당 최대 ECPU를 설정하고 워크로드의 컴퓨팅 사용률이 이 값을 초과하면 Memcached 요청 병목 현상이 시작됩니다. ElastiCache

`BytesUsedForCache`또는 `ElastiCacheProcessingUnits` 에 최대 한도를 설정하는 경우 최대 한도보다 낮은 값으로 CloudWatch 경보를 설정하여 캐시가 이 한도에 가깝게 작동할 때 알림을 받을 수 있도록 하는 것이 좋습니다. 설정한 최대 한도의 75%로 경보를 설정하는 것이 좋습니다. CloudWatch 알람 설정 방법에 대한 설명서를 참조하십시오.

서버리스를 통한 사전 크기 조정 ElastiCache

ElastiCache 서버리스 사전 스케일링

사전 크기 조정 (사전 워밍이라고도 함) 을 사용하여 캐시에 지원되는 최소 제한을 설정할 수 있습니다. ElastiCache 초당 ElastiCache 처리 단위 (ECPU) 또는 데이터 스토리지에 대해 이러한 최소값을 설정할 수 있습니다. 이는 예상 규모 조정 이벤트에 대비하는 데 유용할 수 있습니다. 예를 들어 게임 회사가 새 게임이 출시된 후 첫 1분 이내에 로그인 수가 5배 증가할 것으로 예상한다면 사용량이 크게 급증하는 상황에 대비해 캐시를 준비할 수 있습니다.

ElastiCache 콘솔, CLI 또는 API를 사용하여 사전 조정을 수행할 수 있습니다. ElastiCache 서버리스는 60분 이내에 캐시에서 사용 가능한 ECPU/초를 업데이트하고 최소 한도 업데이트가 완료되면 이벤트 알림을 보냅니다.

사전 스케일링 작동 방식

콘솔, CLI 또는 API를 통해 초당 ECPU 또는 데이터 스토리지의 최소 한도를 업데이트하면 1시간 이내에 새 한도를 사용할 수 있습니다. ElastiCache 서버리스는 빈 캐시에서 초당 30K ECPU를 지원하고 복제본에서 읽기 기능을 사용할 경우 초당 최대 90K ECPU를 지원합니다. ElastiCache 10~12분마다 ECPU/초를 두 배로 늘릴 수 있습니다. 이 스케일링 속도는 대부분의 워크로드에 충분합니다. 예정된 스케일링 이벤트가 이 속도를 초과할 것으로 예상되는 경우 최소 ECPU/초를 피크 이벤트가 발생하기 최소 60분 전에 예상되는 최대 ECPU/초로 설정하는 것이 좋습니다. 그렇지 않으면 애플리케이션의 지연 시간이 증가하고 요청 전송이 제한될 수 있습니다.

최소 한도 업데이트가 완료되면 ElastiCache 서버리스는 새로운 초당 최소 ECPU 또는 새로운 최소 스토리지에 대한 측정을 시작합니다. 이는 애플리케이션이 캐시에서 요청을 실행하지 않거나 데이터 스토리지 사용량이 최소 이하인 경우에도 발생합니다. 현재 설정에서 최소 한도를 낮추면 업데이트가 즉시 적용되므로 ElastiCache 서버리스는 새로운 최소 한도에서 즉시 측정을 시작합니다.

> **Note**
>
> - 최소 사용량 한도를 설정하면 실제 사용량이 최소 사용량 한도보다 낮더라도 해당 한도에 대한 요금이 부과됩니다. 최소 사용량 한도를 초과하는 ECPU 또는 데이터 스토리지 사용량에는 일반 요금이 부과됩니다. 예를 들어 초당 100,000ECPU의 최소 사용량 한도를 설정하면 사용량이 설정된 최소값보다 낮더라도 시간당 최소 1.224 USD의 요금이 부과됩니다 (us-east-1의 ECPU 가격 사용).
> - ElastiCache 서버리스는 캐시의 전체 수준에서 요청된 최소 규모를 지원합니다. ElastiCache 또한 서버리스는 슬롯당 초당 최대 30K ECPU를 지원합니다 (READONLY 연결을 사용하여 복제본에서 읽기 기능을 사용하는 경우 초당 90K ECPU). 가장 좋은 방법은 애플리케이션에서 Redis 슬롯을 통한 키 분배와 키 간의 트래픽을 최대한 균일하게 하는 것입니다.

콘솔을 사용하여 스케일링 제한을 설정하고 AWS CLI

AWS 콘솔을 사용한 규모 조정 제한 설정

1. https://console.aws.amazon.com/elasticache/ 에서 AWS Management Console 로그인하고 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 제한에서 적절한 메모리 또는 컴퓨팅 한도를 설정합니다.
7. 변경 사항 미리보기를 클릭한 다음 변경 사항 저장을 클릭합니다.

를 사용하여 스케일링 제한을 설정합니다. AWS CLI

CLI를 사용하여 스케일링 제한을 변경하려면 API를 사용하십시오. modify-serverless-cache

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}'
```

CLI를 사용하여 규모 조정 제한 제거

CLI를 사용하여 조정 제한을 제거하려면 최소 및 최대 제한 매개변수를 0으로 설정합니다.

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}'
```

Memcached ElastiCache 자체 설계 클러스터를 위한 확장

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. ElastiCache Amazon을 사용하면 현재 수요에 맞게 규모를 조정하고 사용한 만큼만 비용을 지불할 수 있습니다. ElastiCache 수요에 맞춰 캐시를 확장할 수 있습니다.

다음은 수행하려는 조정 작업에 대한 올바른 주제를 찾는 데 도움이 됩니다.

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 클러스터에 60개 이상의 노드가 필요하거나 한 AWS 지역에 총 300개 이상의 노드가 필요한 경우 https://aws.amazon.com/contact-us/ /에서 ElastiCache 한도 증가 요청 양식을 작성하십시오. elasticache-node-limit-request

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.

Memcached 클러스터에 60개 이상의 노드가 필요하거나 한 AWS 지역에 총 300개 이상의 노드가 필요한 경우 https://aws.amazon.com/contact-us/ /에서 ElastiCache 한도 증가 요청 양식을 작성하십시오. elasticache-node-limit-request

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 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 노드를 자동으로 식별 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)

- Memcached를 수직으로 확장 (API) ElastiCache

수직으로 Memcached 조정(콘솔)

다음 절차는 콘솔을 사용하여 클러스터를 수직으로 확장하는 과정을 안내합니다. ElastiCache

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 설명은 클러스터 엔드포인트 찾기(콘솔) 섹션을 참조하세요.
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 CLI로 클러스터 생성을 참조하세요.
2. 애플리케이션에서 엔드포인트를 새 클러스터의 엔드포인트로 업데이트합니다. 자세한 설명은 엔드포인트 찾기(AWS CLI) 섹션을 참조하세요.
3. 이전 캐시 클러스터를 삭제합니다. 자세한 설명은 AWS CLI 사용 섹션을 참조하세요.

Memcached를 수직으로 확장 (API) ElastiCache

다음 절차는 API를 사용하여 Memcached 캐시 클러스터를 수직으로 확장하는 방법을 안내합니다. ElastiCache

Memcached 캐시 클러스터를 수직으로 확장하는 방법 (API) ElastiCache

1. 새 노드 유형으로 새 캐시 클러스터를 생성합니다. 자세한 정보는 클러스터 생성 (ElastiCache API)를 참조하세요.
2. 애플리케이션에서 엔드포인트를 새 캐시 클러스터의 엔드포인트로 업데이트합니다. 자세한 설명은 엔드포인트 찾기(ElastiCache API) 섹션을 참조하세요.

3. 이전 캐시 클러스터를 삭제합니다. 자세한 내용은 ElastiCache API 사용을(를) 참조하세요.

ElastiCache 리소스에 태그 지정

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

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

태그 기본 사항

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

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

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

AWS Management Console, AWS CLI 및 ElastiCache API를 사용하여 태그 관련 작업을 수행할 수 있습니다.

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

태그 지정이 가능한 리소스

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

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

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

- `CreateServerlessCache`
- `CreateCacheCluster`
- `CreateCacheParameterGroup`
- `CreateCacheSecurityGroup`
- `CreateCacheSubnetGroup`
- `PurchaseReservedCacheNodesOffering`

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

ElastiCache 리소스에 대한 태그 지정 지원

태그 지원	생성 시 태그 지정 지원
예	예
예	예

태그 지원	생성 시 태그 지정 지원
예	예
예	예
예	예
예	예

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

자세한 내용은 리소스에 태그 지정 예제 섹션을 참조하세요.

결제를 위한 리소스 태그 지정에 대한 자세한 내용은 비용 할당 태그를 사용하여 비용을 모니터링합니다. 섹션을 참조하세요.

태그 제한

태그에 적용되는 기본 제한은 다음과 같습니다.

- 리소스당 최대 태그 수 – 50개
- 각 리소스에 대해 각 태그 키는 고유하며 하나의 값만 가질 수 있습니다.
- 최대 키 길이는 유니코드 문자(UTF-8) 128자입니다.
- 최대 값 길이는 유니코드 문자(UTF-8) 256자입니다.
- ElastiCache는 태그에 모든 문자를 사용할 수 있지만, 다른 서비스에는 제한이 적용될 수 있습니다. 서비스에서 허용되는 문자는 UTF-8로 표현할 수 있는 문자, 숫자 및 공백과 특수 문자 + - = . _ : / @ 입니다.
- 태그 키와 값은 대/소문자를 구분합니다.

- `aws:` 접두사는 AWS용으로 예약되어 있습니다. 태그에 이 접두사가 있는 태그 키가 있는 경우 태그의 키 또는 값을 편집하거나 삭제할 수 없습니다. `aws:` 접두사가 지정된 태그는 리소스당 태그 수 제한에 포함되지 않습니다.

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

태그를 지정할 수 있는 ElastiCache 리소스에 대한 자세한 내용은 태그 지정이 가능한 리소스 섹션을 참조하세요.

리소스에 태그 지정 예제

- 태그를 사용한 서버리스 캐시 생성

```
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"
```

- 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 memcached \
--tags Key="project",Value="XYZ" Key="Elasticache",Value="Service"
```

태그 기반 액세스 제어 정책 예제

1. 클러스터에 Project=XYZ 태그가 있는 경우에만 클러스터에 대한 AddTagsToResource 작업을 허용합니다.

```
{
    "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. 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 ElastiCache에서 리소스에 비용 할당 태그를 추가하면 인보이스 비용을 리소스 태그 값으로 그룹화하여 비용을 추적할 수 있습니다.

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

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

또한 태그를 결합하여 보다 세부적인 수준으로 비용을 추적할 수 있습니다. 예를 들어, 리전별 서비스 비용을 추적하려면 `Service` 및 `Region` 태그 키를 사용할 수 있습니다. 하나의 리소스에서 `ElastiCache` 및 `Asia Pacific (Singapore)` 값이 있을 수 있으며 다른 리소스에서 `ElastiCache` 및 `Europe (Frankfurt)` 값이 있을 수 있습니다. 리전별로 구분된 총 ElastiCache 비용을 볼 수 있습니다. 자세한 내용은 AWS Billing 사용 설명서의 비용 할당 태그 사용을 참조하세요.

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

ElastiCache 비용 할당 태그 특성

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

  샘플 ARN: `arn:aws:elasticache:<region>:<customer-id>:<resource-type>:<resource-name>`

  Memcached: 태그가 클러스터에만 적용됩니다.

  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 보유할 수 있습니다.

- 태그 세트의 값이 고유하지 않습니다. 예를 들어, 두 키 `Service`와 `Application`에 `ElastiCache` 값이 있는 태그 세트가 있을 수 있습니다.

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

AWS CLI를 사용하여 비용 할당 태그 관리

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

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

샘플 arn: `arn:aws:elasticache:us-west-2:1234567890:cluster:my-cluster`

샘플 arn: `arn:aws:elasticache:us-west-2:1234567890:cluster:my-cluster`

주제

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

AWS CLI를 사용하여 태그 나열

list-tags-for-resource 작업을 사용하여 기존 ElastiCache 리소스에서 태그를 나열하는 데 AWS CLI를 사용할 수 있습니다.

다음 코드는 AWS CLI를 사용하여 us-west-2 리전에 있는 Memcached 클러스터 `my-cluster`의 태그를 나열합니다.

Linux, macOS 또는 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
```

이 작업의 출력은 다음 리소스의 모든 태그 목록과 유사합니다.

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

리소스에 태그가 없으면 출력은 빈 TagList가 됩니다.

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

자세한 내용은 ElastiCache list-tags-for-resource에 대한 AWS CLI 섹션을 참조하세요.

AWS CLI를 사용하여 태그 추가

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

다음 코드는 AWS CLI를 사용하여 us-west-2 리전에 있는 클러스터 `my-cluster`의 에 각각 `elasticache` 및 `us-west-2` 값을 갖는 `Service` 및 `Region` 키를 추가합니다.

Linux, macOS 또는 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
```

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

```
{
   "TagList": [
      {
         "Value": "elasticache",
         "Key": "Service"
      },
      {
         "Value": "us-west-2",
         "Key": "Region"
      }
   ]
}
```

자세한 내용은 ElastiCache add-tags-to-resource에 대한 AWS CLI 섹션을 참조하세요.

AWS CLI을 통해 create-cache-cluster 작업을 사용하여 새 클러스터를 생성할 때 클러스터에 태그를 추가할 수도 있습니다. ElastiCache 관리 콘솔을 사용하는 경우 클러스터를 생성할 때 태그를 추가할 수 없습니다. 클러스터가 생성된 후에는 콘솔을 사용하여 클러스터에 태그를 추가할 수 있습니다.

AWS CLI를 사용하여 태그 수정

AWS CLI를 사용하여 ElastiCache for Memcached 클러스터의 태그를 수정할 수 있습니다.

태그를 수정하려면

- add-tags-to-resource를 사용하여 새 태그 및 값을 추가하거나 기존 태그에 연결된 값을 변경합니다.
- remove-tags-from-resource를 사용하여 리소스에서 지정된 태그를 제거합니다.

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

AWS CLI를 사용하여 태그 제거

remove-tags-from-resource 작업을 사용하여 기존 ElastiCache for Memcached 클러스터에서 태그를 제거하는 데 AWS CLI를 사용할 수 있습니다.

다음 코드에서는 AWS CLI를 사용하여 us-west-2 리전에 있는 클러스터 `my-cluster`의 에서 `Service` 및 `Region` 키를 갖는 태그를 제거합니다.

Linux, macOS 또는 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
```

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

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

자세한 내용은 ElastiCache remove-tags-from-resource에 대한 AWS CLI 섹션을 참조하세요.

ElastiCache API를 사용하여 비용 할당 태그 관리

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

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

샘플 arn: `arn:aws:elasticache:us-west-2:1234567890:cluster:my-cluster`

주제

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

ElastiCache API를 사용하여 태그 나열

ListTagsForResource 작업을 사용하여 기존 리소스에서 태그를 나열하는 데 ElastiCache API를 사용할 수 있습니다.

다음 코드는 ElastiCache API를 사용하여 us-west-2 리전에 있는 리소스 `my-cluster`의 태그를 나열합니다.

```
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>
```

ElastiCache API를 사용하여 태그 추가

AddTagsToResource 작업을 사용하여 기존 ElastiCache 클러스터에 태그를 추가하는 데 ElastiCache API를 사용할 수 있습니다. 리소스에 태그 키가 없으면 키와 값이 리소스에 추가됩니다. 리소스에 이미 키가 있는 경우 해당 키와 연결된 값이 새 값으로 업데이트됩니다.

다음 코드는 ElastiCache API를 사용하여 us-west-2 리전의 리소스 `my-cluster`에 각각 `elasticache` 및 `us-west-2` 값을 갖는 `Service` 및 `Region` 키를 추가합니다.

```
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>
```

자세한 내용은 Amazon ElastiCache API 참조에서 AddTagsToResource 섹션을 참조하세요.

ElastiCache API를 사용하여 태그 수정

ElastiCache API를 사용하여 ElastiCache 클러스터의 태그를 수정할 수 있습니다.

태그 값을 수정하려면

- AddTagsToResource 작업을 사용하여 새 태그와 값을 추가하거나 기존 태그의 값을 변경합니다.
- RemoveTagsFromResource를 사용하여 리소스에서 태그를 제거합니다.

두 작업 중 하나의 출력은 지정된 리소스의 태그 목록과 값입니다.

RemoveTagsFromResource를 사용하여 리소스에서 태그를 제거합니다.

ElastiCache API를 사용하여 태그 제거

RemoveTagsFromResource 작업을 사용하여 기존 ElastiCache for Memcached 클러스터에서 태그를 제거하는 데 ElastiCache API를 사용할 수 있습니다.

다음 코드에서는 ElastiCache API를 사용하여 us-west-2 리전에 있는 클러스터 `my-cluster`에서 `Service` 및 `Region` 키를 갖는 태그를 제거합니다.

```
https://elasticache.us-west-2.amazonaws.com/
```

```
?Action=RemoveTagsFromResource
&ResourceName=arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&TagKeys.member.1=Service
&TagKeys.member.2=Region
&Version=2015-02-02
&Timestamp=20150202T192317Z
&X-Amz-Credential=<credential>
```

Amazon ElastiCache의 Well-Architected Lens 사용

이 섹션에서는 효율적으로 아키텍팅된 ElastiCache 워크로드를 설계하기 위한 설계 원칙과 지침을 모은 Amazon ElastiCache Well-Architected Lens에 대해 설명합니다.

- ElastiCache Lens는 AWS Well-Architected Framework에 포함됩니다.
- 요소마다 ElastiCache 아키텍처에 대한 논의를 시작하는 데 도움이 되는 일련의 질문이 있습니다.
  - 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 사항을 충족하는 코드, 구성 및 AWS 리소스. 구성 요소는 다른 구성 요소와 상호 작용하며 종종 마이크로 서비스 아키텍처의 서비스와 동일합니다.
  - 워크로드 - 함께 비즈니스 가치를 제공하는 구성 요소의 집합. 예를 들어, 마케팅 웹사이트, 전자 상거래 웹사이트, 모바일 앱 백엔드, 분석 플랫폼 등이 워크로드에 해당합니다.

주제

- Amazon ElastiCache Well-Archited Lens 운영 우수성 요소
- Amazon ElastiCache의 Well-Architected Lens 보안 요소
- Amazon ElastiCache의 Well-Architected Lens 신뢰성 요소
- Amazon ElastiCache Well-Architected Lens 성능 효율성 요소
- Amazon ElastiCache의 Well-Architected Lens 비용 최적화 요소

Amazon ElastiCache Well-Archited Lens 운영 우수성 요소

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 클러스터에서 트리거되는 경고 및 이벤트를 어떻게 이해하고 이에 대응하나요?
- OE 2: 기존 ElastiCache 클러스터를 언제, 어떻게 확장하나요?
- OE 3: ElastiCache 클러스터 리소스를 관리하고 클러스터를 최신 상태로 유지하려면 어떻게 해야 하나요?
- OE 4: ElastiCache 클러스터에 대한 클라이언트의 연결을 어떻게 관리하나요?
- OE 5: 워크로드에 대해 ElastiCache 구성 요소를 어떻게 배포하나요?
- OE 6: 장애를 어떻게 대비하고 완화할 계획인가요?
- OE 7: Redis 엔진 이벤트 문제를 어떻게 해결하나요?

OE 1: ElastiCache 클러스터에서 트리거되는 경고 및 이벤트를 어떻게 이해하고 이에 대응하나요?

질문 수준의 소개: ElastiCache 클러스터를 운영할 때 선택적으로 특정 이벤트 발생 시 알림 및 경고를 받을 수 있습니다. ElastiCache는 기본적으로 장애 조치, 노드 교체, 규모 조정 작업, 예약된 유지 관리 등 리소스와 관련된 이벤트를 로깅합니다. 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 콘솔에서 리전을 선택한 후 또는 Amazon Command Line Interface(AWS CLI) describe-events 명령과 ElastiCache API를 사용하여 ElastiCache에서 생성된 이벤트를 검토합니다. Amazon Simple Notification Service(Amazon SNS)를 사용하여 중요한 클러스터 이벤트에 대해 알림을 보내도록 ElastiCache를 구성합니다. Amazon SNS를 클러스터와 함께 사용하면 프로그래밍 방식으로 ElastiCache 이벤트에 대한 조치를 취할 수 있습니다.
  - 이벤트에는 크게 현재 이벤트와 예정된 이벤트라는 두 가지 범주가 있습니다. 현재 이벤트 목록에는 리소스 생성 및 삭제, 규모 조정 작업, 장애 조치, 노드 재부팅, 생성된 스냅샷, 클러스터의 파라미터 수정, CA 인증서 갱신, 장애 이벤트(클러스터 프로비저닝 실패 - VPC 또는 ENI-, 규모 조정

실패 - ENI- 및 스냅샷 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(Redis만 해당)
- [리소스]:
  - ElastiCache Amazon SNS 알림 관리
  - 이벤트 알림 및 Amazon SNS

- [가장 좋음] 이벤트에 대한 응답을 자동화하려면 SNS와 Lambda 함수 등의 AWS 제품 및 서비스 기능을 활용합니다. 모범 사례에 따라 작고 빈번하며 되돌릴 수 있는 변경을 코드로 적용하여 시간이 지남에 따라 운영을 발전시킵니다. 클러스터를 모니터링하려면 Amazon CloudWatch 지표를 사용해야 합니다.

  [리소스]: AWS Lambda, Amazon Route 53 및 Amazon SNS를 사용하여 Amazon ElastiCache for Redis(클러스터 모드 비활성화됨)에서 Lambda 및 SNS를 사용하는 사용 사례에 대해 읽기 전용 복제본 엔드포인트를 모니터링합니다.

OE 2: 기존 ElastiCache 클러스터를 언제, 어떻게 확장하나요?

질문 수준의 소개: ElastiCache 클러스터의 크기를 적절하게 조정하는 것은 기본 워크로드 유형이 변경될 때마다 평가해야 하는 균형 조정 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 있습니다. 환경을 적절한 규모로 조정하면 성능 효율성과 비용 최적화 간의 균형을 맞출 수 있습니다. ElastiCache는 두 가지 차원으로 스케일 인을 수행하여 리소스의 과도하거나 저조한 사용 문제를 해결할 수 있습니다. 노드 용량을 늘리거나 줄여서 수직으로 규모를 조정할 수 있습니다. 노드를 추가 및 제거하여 수평적으로 규모를 조정할 수도 있습니다.

- [필수] 프라이머리 노드에서 CPU 및 네트워크의 과도한 사용은 읽기 작업을 복제본 노드로 오프로드하고 리디렉션하여 해결해야 합니다. 읽기 작업에 복제본 노드를 사용하여 프라이머리 노드 사용률

을 줄입니다. 클러스터 모드가 비활성화된 경우 ElastiCache 리더 엔드포인트에 연결하거나 클러스터 모드가 활성화된 경우 Redis READONLY 명령을 사용하여 Redis 클라이언트 라이브러리에서 이를 구성할 수 있습니다.

[리소스]:

- 연결 엔드포인트 찾기
- 클러스터 적정 크기 조정
- Redis READONLY 명령

- [필수] CPU, 메모리, 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 for Redis 클러스터 모드가 비활성화된 경우 프라이머리 및 복제본 노드를 수직으로 규모 조정할 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]:

  - Amazon CloudWatch를 사용하는 Amazon ElastiCache for Redis 모니터링 모범 사례
  - ElastiCache for Redis 클러스터 크기 조정
  - ElastiCache for Memcached 클러스터 크기 조정
- [가장 좋음] 시간 경과에 따른 추세 모니터링은 특정 시점에 모니터링할 경우 눈에 띄지 않는 워크로드 변경을 감지하는 데 도움이 될 수 있습니다. 장기적인 추세를 감지하려면 CloudWatch 지표를 사용하여 더 긴 시간 범위를 스캔하세요. 장기간의 CloudWatch 지표를 관찰하면서 얻은 교훈은 클러스터 리소스 사용률에 대한 예측에 도움이 될 것입니다. CloudWatch 데이터 포인트 및 지표는 최대 455일 동안 사용할 수 있습니다.

  [리소스]:

  - CloudWatch 지표를 사용한 ElastiCache for Redis 모니터링
  - CloudWatch 지표를 사용한 Memcached 모니터링
  - Amazon CloudWatch를 사용하는 Amazon ElastiCache for Redis 모니터링 모범 사례
- [가장 좋음] CloudFormation으로 ElastiCache 리소스를 생성하는 경우 CloudFormation 템플릿을 사용하여 변경을 수행함으로써 운영 일관성을 유지하고 관리되지 않는 구성 변경 사항 및 스택 편차를 방지하는 것이 가장 좋습니다.

  [리소스]:

  - CloudFormation에 대한 ElastiCache 리소스 유형 참조

- [가장 좋음] 클러스터 운영 데이터를 사용하여 규모 조정 작업을 자동화하고 CloudWatch에서 임계값을 정의하여 경고를 설정합니다. CloudWatch Events 및 Simple Notification Service(SNS)를 사용하여 Lambda 함수를 트리거하고 ElastiCache API를 실행하여 클러스터의 크기를 자동으로 조정합니다. `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 Events란?
  - Amazon Simple Notification Service에서 AWS Lambda 사용
  - ElastiCache API 참조

OE 3: ElastiCache 클러스터 리소스를 관리하고 클러스터를 최신 상태로 유지하려면 어떻게 해야 하나요?

질문 수준의 소개: 규모에 맞게 운영하려면 모든 ElastiCache 리소스를 정확히 찾아내고 식별할 수 있어야 합니다. 새 애플리케이션 기능을 배포할 때는 개발, 테스트, 프로덕션 등 모든 ElastiCache 환경 유형에서 클러스터 버전 대칭을 만들어야 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. 네이티브 ElastiCache 기능을 활용하면 엔지니어링 팀이 ElastiCache 유지 관리가 아닌 비즈니스 목표를 달성하는 데 집중할 수 있습니다.

- [가장 좋음] 사용 가능한 최신 엔진 버전에서 실행하고 셀프 서비스 업데이트가 출시되는 즉시 적용합니다. ElastiCache는 클러스터의 지정된 유지 관리 기간 동안 기본 인프라를 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]:

  - Amazon ElastiCache의 셀프 서비스 업데이트
  - Amazon ElastiCache 관리형 유지 관리 및 서비스 업데이트 도움말 페이지

- [가장 좋음] 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
  - CloudFormation에 대한 ElastiCache 리소스 유형 참조
  - 파라미터 그룹

OE 4: ElastiCache 클러스터에 대한 클라이언트의 연결을 어떻게 관리하나요?

질문 수준의 소개: 규모에 맞게 운영할 때는 클라이언트가 ElastiCache 클러스터와 연결하여 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. 그러나 쓰기와 읽기를 분리하면 Redis 복제의 비동기적 특성으로 인해 쓰기 직후에 키를 읽을 수 없게 된다는 점에 유의하시기 바랍니다. WAIT 명령을 사용하면 실제 데이터 안전성을 향상하고 복제본이 클라이언트에 응답하기 전에 쓰기를 확인하도록 강제할 수 있지만, 전반적인 성능 저하가 발생할 수 있습니다. 클러스터 모드가 비활성화된 경우, ElastiCache 리더 엔드포인트를 사용하여 ElastiCache for Redis 클라이언트 라이브러리에서 읽기 작업에 복제본 노드를 사용하도록 구성할 수 있습니다. 클러스터 모드가 활성화된 경우, ElastiCache for Redis READONLY 명령을 사용합니다. 많은 ElastiCache for Redis 클라이언트 라이브러리에서 ElastiCache for Redis READONLY는 기본적으로 또는 구성 설정을 통해 구현됩니다.

  [리소스]:

  - 연결 엔드포인트 찾기
  - READONLY
- [필수] 연결 풀링을 사용합니다. TCP 연결을 설정하면 클라이언트와 서버 측 모두에서 CPU 시간이 소모되며 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. ElastiCache for Redis

클라이언트 라이브러리(지원되는 경우)를 통해 애플리케이션 환경에 사용 가능한 프레임워크를 사용하여 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 클러스터에서 TLS가 활성화된 경우 연결을 열고 닫으며 TLS 핸드셰이크를 수행하는 데 필요한 지연 시간과 CPU 사용률이 줄어듭니다.

  [리소스]: 더 높은 가용성을 위해 Amazon ElastiCache for Redis 구성
- [좋음] (사용 사례에서 가능한 경우) 파이프라이닝을 사용하면 성능이 크게 향상될 수 있습니다.
  - 파이프라이닝을 사용하면 애플리케이션 클라이언트와 클러스터 간의 왕복 시간(RTT)이 줄어들고 클라이언트가 아직 이전 응답을 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: 워크로드에 대해 ElastiCache 구성 요소를 어떻게 배포하나요?

질문 수준의 소개: ElastiCache 환경은 AWS 콘솔을 통해 수동으로 배포하거나 API, CLI, 툴킷 등을 통해 프로그래밍 방식으로 배포할 수 있습니다. 운영 우수성 모범 사례에서는 가능하면 코드를 통해 배포를 자동화하는 것을 권장합니다. 또한 ElastiCache 클러스터를 워크로드별로 분리하거나 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 사용하거나 여러 워크로드를 단일 클러스터 또는 이들의 조합으로 결합할 수 있습니다. 장단점을 이해하면 운영 우수성과 비용 최적화 간의 균형을 맞추는 데 도움이 될 수 있습니다.

- [필수] ElastiCache에서 사용할 수 있는 배포 옵션을 이해하고 가능하면 이러한 절차를 자동화합니다. 가능한 자동화 수단으로는 CloudFormation, AWS CLI/SDK 및 API가 있습니다.

  [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, 규모 조정 및 관리를 가장 세밀하게 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 수준의 소개: 운영 우수성에는 잠재적 장애 원인을 식별하여 제거하거나 완화할 수 있도록 정기적으로 '사전 검토' 훈련을 수행하여 장애를 예측하는 것이 포함됩니다. ElastiCache는 테스트 목적으로 노드 장애 이벤트를 시뮬레이션할 수 있는 장애 조치 API를 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 및 테스트 계정에서 정기적으로 장애 조치 테스트를 수행합니다. TestFailover

OE 7: Redis 엔진 이벤트 문제를 어떻게 해결하나요?

질문 수준의 소개: 운영 우수성을 위해서는 서비스 수준 및 엔진 수준 정보를 모두 조사하여 클러스터의 상태를 분석할 수 있어야 합니다. Amazon ElastiCache for Redis는 Amazon CloudWatch와 Amazon Kinesis Data Firehose에 Redis 엔진 로그를 보낼 수 있습니다.

질문 수준의 이점: Amazon ElastiCache for Redis 클러스터에서 Redis 엔진 로그를 활성화하면 클러스터의 상태 및 성능에 영향을 미치는 이벤트를 파악할 수 있습니다. Redis 엔진 로그는 ElastiCache 이벤트 메커니즘을 통해 사용할 수 없는 데이터를 Redis 엔진에서 직접 제공합니다. ElastiCache 이벤트(위 OE-1 참조)와 Redis 엔진 로그를 주의 깊게 관찰하면 문제 해결 시 ElastiCache 서비스 관점과 Redis 엔진 관점에서 이벤트 순서를 판단할 수 있습니다.

- [필수] Redis 엔진 로깅 기능이 활성화되었는지 확인합니다. 이 기능은 ElastiCache for Redis 6.2부터 사용할 수 있습니다. 클러스터 생성 중에 또는 생성 후 클러스터를 수정하여 이 작업을 수행할 수 있습니다.
  - Amazon CloudWatch Logs 또는 Amazon Kinesis Data Firehose가 Redis 엔진 로그의 적절한 대상인지 확인합니다.
  - 로그를 유지하려면 CloudWatch 또는 Kinesis Data Firehose에서 적절한 대상 로그를 선택합니다. 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 무엇인가요?
  - Amazon Kinesis Data Firehose 소개: Amazon Kinesis Data Firehose란 무엇인가요?
- [가장 좋음] Amazon CloudWatch Logs를 사용하는 경우 Amazon CloudWatch 로그 인사이트를 활용하여 Redis 엔진 로그에서 중요한 정보를 쿼리하는 것을 고려합니다.

  예를 들어, 다음과 같이 LogLevel이 'WARNING'인 이벤트를 반환하는 Redis 엔진 로그가 포함된 CloudWatch 로그 그룹에 대한 쿼리를 생성합니다.

```
fields @timestamp, LogLevel, Message
| sort @timestamp desc
| filter LogLevel = "WARNING"
```

  [리소스]: CloudWatch 로그 인사이트를 사용한 로그 데이터 분석

Amazon ElastiCache의 Well-Architected Lens 보안 요소

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 승인된 액세스를 제어하기 위해 어떤 조치를 취하고 있나요?
- SEC 2: 네트워킹 기반 제어를 넘어 ElastiCache에 대한 추가 권한 부여가 애플리케이션에 필요한가요?

- SEC 3: 명령이 실수로 실행되어 데이터 손실이나 장애가 발생할 위험이 있나요?
- SEC 4: ElastiCache를 사용하여 저장 데이터를 암호화하려면 어떻게 해야 하나요?
- SEC 5: ElastiCache를 사용하여 전송 중인 데이터를 어떻게 암호화하나요?
- SEC 6: 컨트롤 플레인 리소스에 대한 액세스를 어떻게 제한하나요?
- SEC 7: 보안 이벤트를 어떻게 감지하고 이에 대응하나요?

SEC 1: ElastiCache 데이터에 대한 승인된 액세스를 제어하기 위해 어떤 조치를 취하고 있나요?

질문 수준의 소개: 모든 ElastiCache 클러스터는 VPC, 서버리스 함수(AWS Lambda) 또는 컨테이너(Amazon Elastic Container Service)의 Amazon Elastic Compute Cloud 인스턴스에서 액세스할 수 있도록 설계되었습니다. 가장 자주 접하는 시나리오는 동일한 Amazon Virtual Private Cloud(Amazon VPC) 내의 Amazon Elastic Compute Cloud 인스턴스에서 ElastiCache 클러스터에 액세스하는 것입니다. Amazon EC2 인스턴스에서 클러스터에 연결하려면 Amazon EC2 인스턴스가 클러스터에 액세스하도록 권한을 부여해야 합니다. VPC에서 실행 중인 ElastiCache 클러스터에 액세스하려면 클러스터에 네트워크 수신을 허용해야 합니다.

질문 수준의 이점: 클러스터로의 네트워크 수신은 VPC 보안 그룹을 통해 제어됩니다. 보안 그룹은 Amazon EC2 인스턴스에 대한 수신 및 발신 트래픽을 제어하는 가상 방화벽 역할을 합니다. 인바운드 규칙은 인스턴스로 들어오는 트래픽을 제어하고 아웃바운드 규칙은 인스턴스에서 나가는 트래픽을 제어합니다. ElastiCache의 경우, 클러스터를 시작할 때 보안 그룹을 연결해야 합니다. 이렇게 하면 클러스터를 구성하는 모든 노드에 대해 인바운드 및 아웃바운드 트래픽 규칙이 적용됩니다. 또한 ElastiCache는 VPC의 프라이빗 네트워킹을 통해서만 액세스할 수 있게 하기 위해 프라이빗 서브넷에만 배포되도록 구성되어 있습니다.

- [필수] 클러스터와 연결된 보안 그룹은 클러스터에 대한 네트워크 수신 및 액세스를 제어합니다. 기본적으로 보안 그룹에는 인바운드 규칙이 정의되어 있지 않으므로 ElastiCache로의 수신 경로가 없습니다. 이를 활성화하려면 소스 IP 주소/범위, TCP 유형 트래픽 및 ElastiCache 클러스터용 포트(예: ElastiCache for Redis의 기본 포트 6379)를 지정하여 보안 그룹에 인바운드 규칙을 구성합니다. VPC 내의 모든 리소스(0.0.0.0/0)와 같이 매우 광범위한 수신 소스 세트를 허용할 수 있지만, 특정 보안 그룹과 연결된 Amazon EC2 인스턴스에서 실행 중인 Redis 클라이언트에 대한 인바운드 액세스만 승인하는 등 인바운드 규칙을 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 데이터 액세스를 허용할 수 있습니다. 이 기능을 활성화하려면 `AWSLambdaVPCAccessExecutionRole` 권한이 있는 IAM 실행 역할을 생성한 다음, 해당 역할을 AWS Lambda 함수에 할당해야 합니다.

  [리소스]: Amazon VPC에서 Amazon ElastiCache에 액세스하도록 Lambda 함수 구성: 자습서: Amazon VPC에서 Amazon ElastiCache에 액세스하도록 Lambda 함수 구성

SEC 2: 네트워킹 기반 제어를 넘어 ElastiCache에 대한 추가 권한 부여가 애플리케이션에 필요한가요?

질문 수준의 소개: 개별 클라이언트 수준에서 ElastiCache for Redis 클러스터에 대한 액세스를 제한하거나 제어해야 하는 시나리오에서는 ElastiCache for Redis AUTH 명령을 통해 인증하는 것이 좋습니다. 선택적인 사용자 및 사용자 그룹 관리 기능이 포함된 ElastiCache for Redis 인증 토큰을 사용하면 클라이언트가 명령 및 액세스 키를 실행하도록 허용하기 전에 ElastiCache for Redis가 암호를 요구하므로 데이터 영역 보안을 개선할 수 있습니다.

질문 수준의 이점: 데이터를 안전하게 유지하기 위해 ElastiCache for Redis는 데이터에 대한 무단 액세스를 방지하는 메커니즘을 제공합니다. 여기에는 인증된 명령을 수행하기 전에 클라이언트가 ElastiCache에 연결할 때 역할 기반 액세스 제어(RBAC) AUTH 또는 AUTH 토큰(암호)을 사용하도록 강제하는 것이 포함됩니다.

- [가장 좋음] ElastiCache for Redis 6.x 이상의 경우, 사용자 그룹, 사용자 및 액세스 문자열을 정의하여 인증 및 권한 부여 제어를 정의합니다. 사용자를 사용자 그룹에 할당한 다음, 클러스터에 사용자 그룹을 할당합니다. RBAC를 사용하려면 클러스터 생성 시 RBAC를 선택하고 전송 중 암호화를 활성화해야 합니다. RBAC를 활용할 수 있으려면 TLS를 지원하는 Redis 클라이언트를 사용해야 합니다.

  [리소스]:

  - ElastiCache for Redis에 대한 복제 그룹에 RBAC 적용
  - 액세스 문자열을 사용하여 권한 지정
  - ACL
  - 지원되는 ElastiCache for Redis 버전

- [가장 좋음] 6.x 이전 버전의 ElastiCache for Redis의 경우, 강력한 토큰/암호를 설정하고 ElastiCache for Redis AUTH에 대해 엄격한 암호 정책을 유지하는 것 외에도 암호/토큰을 교체하는 것이 가장 좋습니다. ElastiCache는 언제든지 최대 2개의 인증 토큰을 관리할 수 있습니다. 인증 토큰 사용을 명시적으로 요구하도록 클러스터를 수정할 수도 있습니다.

  [리소스]: 기존 ElastiCache for Redis 클러스터에서 AUTH 토큰 수정

SEC 3: 명령이 실수로 실행되어 데이터 손실이나 장애가 발생할 위험이 있나요?

질문 수준의 소개: 실수로 실행하거나 악의적인 공격자가 실행할 경우 운영에 부정적인 영향을 미칠 수 있는 Redis 명령이 많이 있습니다. 이러한 명령은 성능 및 데이터 안전 관점에서 의도하지 않은 결과를 초래할 수 있습니다. 예를 들어 개발자가 개발 환경에서 일상적으로 FLUSHALL 명령을 호출할 수 있는데, 실수로 인해 프로덕션 시스템에서 실수로 이 명령을 호출하려고 시도하여 우발적으로 데이터가 손실될 수 있습니다.

질문 수준의 이점: ElastiCache for Redis 5.0.3부터 워크로드에 지장을 줄 수 있는 특정 명령의 이름을 변경할 수 있습니다. 명령의 이름을 바꾸면 클러스터에서 실수로 명령이 실행되는 것을 방지할 수 있습니다.

- [필수]

  [리소스]:

  - ElastiCache for Redis 버전 5.0.3(사용 중단, 버전 5.0.6 사용)
  - Redis 5.0.3 파라미터 변경 사항
  - Redis 보안

SEC 4: ElastiCache를 사용하여 저장 데이터를 암호화하려면 어떻게 해야 하나요?

질문 수준의 소개: ElastiCache for Redis는 인메모리 데이터 스토어이지만 클러스터의 표준 작업의 일부로 스토리지에 유지될 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 수준의 이점: ElastiCache for Redis는 데이터 보안을 강화하기 위해 저장 시 암호화를 선택적으로 제공합니다.

- [필수] 저장 데이터 암호화는 ElastiCache 클러스터(복제 그룹)를 생성할 때만 클러스터에서 활성화할 수 있습니다. 기존 클러스터를 수정하여 저장 데이터 암호화를 시작할 수 없습니다. 기본적으로 ElastiCache는 저장 데이터 암호화에 사용되는 키를 제공하고 관리합니다.

  [리소스]:

  - 저장 데이터 암호화 조건
  - 저장 데이터 암호화 활성화
- [가장 좋음] 메모리에 있는 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(CMK)를 자체 관리할 수 있습니다. AWS Key Management Service와의 ElastiCache 통합을 통해 ElastiCache for Redis 클러스터의 저장 데이터를 암호화하는 데 사용되는 키를 생성, 소유 및 관리할 수 있습니다.

  [리소스]:

  - AWS Key Management Service에서 고객 관리형 키 사용
  - AWS Key Management Service
  - AWS KMS 개념

SEC 5: ElastiCache를 사용하여 전송 중인 데이터를 어떻게 암호화하나요?

질문 수준의 소개: 전송 중에 데이터가 손상되는 것을 방지하는 것은 일반적인 요구 사항입니다. 이는 분산 시스템의 구성 요소 내 데이터뿐만 아니라 애플리케이션 클라이언트와 클러스터 노드 간의 데이터를 나타냅니다. ElastiCache for Redis는 클라이언트와 클러스터 간 및 클러스터 노드 간에 전송 중 데이터를 암호화할 수 있도록 하여 이러한 요구 사항을 지원합니다. M6g 및 R6g 패밀리의 인스턴스 유형에는 상시 켜져 있는 인메모리 암호화 기능도 있습니다.

질문 수준의 이점: Amazon ElastiCache의 전송 중 데이터 암호화는 선택적 기능으로, 가장 취약한 지점 즉, 한 위치에서 다른 위치로 데이터를 전송할 때 데이터의 보안을 강화합니다.

- [필수] 전송 중 암호화는 ElastiCache for Redis 클러스터(복제 그룹) 생성 시에만 클러스터에서 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 전송 중 암호화를 활성화하기 전과 후에 워크로드를 벤치마킹하는 것이 좋습니다.

  [리소스]:

  - 전송 중 데이터 암호화 개요

SEC 6: 컨트롤 플레인 리소스에 대한 액세스를 어떻게 제한하나요?

질문 수준의 소개: IAM 정책과 ARN을 통해 ElastiCache for Redis에 대한 세밀한 액세스 제어가 가능하므로 ElastiCache for Redis 클러스터의 생성, 수정 및 삭제를 보다 엄격하게 관리할 수 있습니다.

질문 수준의 이점: 복제 그룹, 노드 등 Amazon ElastiCache 리소스의 관리를 IAM 정책에 따라 특정 권한이 있는 AWS 계정으로 제한하여 리소스의 보안과 신뢰성을 개선할 수 있습니다.

- [필수] 특정 AWS Identity and Access Management 정책을 AWS 사용자에게 할당하여 Amazon ElastiCache 리소스에 대한 액세스를 관리함으로써 어떤 계정이 클러스터에서 어떤 작업을 수행할 수 있는지 더 세밀하게 제어합니다.

  [리소스]:

  - ElastiCache 리소스에 대한 액세스 권한 관리 개요
  - Amazon ElastiCache에 대한 자격 증명 기반 정책(IAM 정책) 사용

SEC 7: 보안 이벤트를 어떻게 감지하고 이에 대응하나요?

질문 수준의 소개: ElastiCache는 RBAC를 활성화한 상태로 배포할 경우 CloudWatch 지표를 내보내 사용자에게 보안 이벤트를 알립니다. 이러한 지표는 실패한 인증 시도, 키 액세스 시도, RBAC 사용자 연결 권한이 없는 명령 실행 시도를 식별하는 데 도움이 됩니다.

또한 AWS 제품 및 서비스 리소스는 배포를 자동화하고 추후 검토 및 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 관련하여 게시된 CloudWatch 지표를 숙지합니다.
  - AuthenticationFailures = Redis에 대한 인증 시도 실패
  - KeyAuthorizationFailures = 사용자가 권한 없이 키에 액세스하려는 시도 실패
  - CommandAuthorizationFailures = 사용자가 권한 없이 명령을 실행하려는 시도 실패

  [리소스]:

  - Redis 지표
- [가장 좋음] 이러한 지표에 대한 경고 및 알림을 설정하고 필요에 따라 대응하는 것이 좋습니다.

  [리소스]:

  - Amazon CloudWatch 경보 사용
- [가장 좋음] Redis ACL LOG 명령을 사용하여 추가 세부 정보를 수집합니다.

  [리소스]:

  - ACL 로그
- [가장 좋음] ElastiCache 배포 및 이벤트의 모니터링, 로깅 및 분석과 관련된 AWS 제품 및 서비스 기능을 숙지합니다.

  [리소스]:

  - AWS CloudTrail을 사용하여 Amazon ElastiCache API 호출 로깅
  - elasticache-redis-cluster-automatic-backup-check
  - CloudWatch 지표를 사용한 사용량 모니터링

Amazon ElastiCache의 Well-Architected Lens 신뢰성 요소

주제

- REL 1: 고가용성(HA) 아키텍처 배포를 어떻게 지원하고 있나요?
- REL 2: ElastiCache를 사용하여 Recovery Point Objective(RPO)를 어떻게 달성하고 있나요?
- REL 3: 재해 복구(DR) 요구 사항을 어떻게 지원하나요?
- REL 4: 장애 조치를 어떻게 효과적으로 계획하나요?
- REL 5: ElastiCache 구성 요소가 규모를 조정하도록 설계되었나요?

REL 1: 고가용성(HA) 아키텍처 배포를 어떻게 지원하고 있나요?

질문 수준의 소개: Amazon ElastiCache의 고가용성 아키텍처를 이해하면 가용성 이벤트 발생 시에도 탄력적인 상태로 운영할 수 있습니다.

질문 수준의 이점: 장애에 대한 복원력을 갖추도록 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: Memcached와 ElastiCache for Redis

1. Memcached는 복제 메커니즘을 제공하지 않으며 주로 임시 워크로드에 사용됩니다.
2. ElastiCache for Redis는 아래에 설명된 HA 기능을 제공합니다.

- [가장 좋음] HA가 필요한 워크로드의 경우 샤드당 최소 2개의 복제본이 있는 클러스터 모드에서 ElastiCache for Redis를 사용합니다. 처리량에 대한 요구 사항이 적어 단 하나의 샤드만 필요한 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. Redis Cluster Protocol에서는 쿼럼을 달성하기 위해 대부분의 프라이머리 노드를 사용할 수 있어야 하므로 샤딩된 워크로드의 경우 최소 3개의 샤드가 장애 조치 이벤트 시 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.

     Amazon ElastiCache for Redis는 이러한 노드에 최적화된 성능을 추가했습니다. 따라서 복제 및 동기화 성능이 향상되어 전반적인 가용성이 향상됩니다.
  5. 예상되는 트래픽 피크에 대처하기 위한 모니터링 및 적절한 크기 조정: 로드가 심한 경우 ElastiCache for Redis 엔진이 응답하지 않아 가용성에 영향을 미칠 수 있습니다. `BytesUsedForCache` 및 `DatabaseMemoryUsagePercentage`는 메모리 사용량을 나타내는 좋은 지표이며 `ReplicationLag`는 쓰기 속도를 기반으로 복제 상태를 나타내는 지표입니다. 이러한 지표를 사용하여 클러스터 규모 조정을 트리거할 수 있습니다.
  6. 프로덕션 장애 조치 이벤트 전에 장애 조치 API로 테스트하여 클라이언트 측 복원력을 보장합니다.

  [리소스]:

  - 더 높은 가용성을 위해 Amazon ElastiCache for Redis 구성
  - 고가용성을 위한 복제 그룹 사용

REL 2: ElastiCache를 사용하여 Recovery Point Objective(RPO)를 어떻게 달성하고 있나요?

질문 수준의 소개: 워크로드 RPO를 이해하여 ElastiCache 백업 및 복구 전략에 대한 의사 결정을 내립니다.

질문 수준의 이점: RPO 전략을 수립하면 재해 복구 시나리오 발생 시 비즈니스 연속성을 개선할 수 있습니다. 백업 및 복원 정책을 설계하면 ElastiCache 데이터에 대한 Recovery Point Objective(RPO)를 달성하는 데 도움이 될 수 있습니다. ElastiCache for Redis는 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 배포의 RPO를 이해하고 문서화합니다.
  - Memcached는 백업 프로세스를 제공하지 않는다는 점에 유의하시기 바랍니다.
  - ElastiCache 백업 및 복원 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] ElastiCache의 예약 백업 기능을 활용하여 지정된 기간 동안 데이터를 정기적으로 백업합니다.
  - 백업에서 정기적으로 복원을 테스트합니다.
- [리소스]:
  - Redis
  - ElastiCache for Redis 백업 및 복원
  - 수동 백업 만들기
  - 자동 백업 예약
  - ElastiCache Redis 클러스터 백업 및 복원

REL 3: 재해 복구(DR) 요구 사항을 어떻게 지원하나요?

질문 수준의 소개: 재해 복구는 모든 워크로드 계획에서 중요한 측면입니다. ElastiCache for Redis는 워크로드 복원력 요구 사항에 따라 재해 복구를 구현하기 위한 여러 옵션을 제공합니다. Amazon ElastiCache for Redis 글로벌 데이터 스토어를 사용하면 한 리전에서 ElastiCache for Redis 클러스터에 데이터를 쓰고 다른 두 리전 간 복제본 클러스터에서 데이터를 읽을 수 있으므로 읽기 지연 시간이 짧고 리전 전체에서 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 사항을 기반으로 모든 ElastiCache 구성 요소에 대한 DR 전략을 개발하고 문서화합니다. ElastiCache가 독특한 점은 일부 사용 사례는 완전히 일시적이어서 DR 전략이 필요하지 않은 반면, 어떤 사용 사례는 스펙트럼의 반대편에 있어서 매우 강력한 DR 전략이 필요하다는 점입니다. 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. 최소 3개의 AZ를 사용할 수 있는 다중 AZ 아키텍처에서 클러스터 모드를 활성화하여 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 사용한 AWS 리전 간 복제
  - 선택적으로 클러스터 크기를 조정하여 백업에서 복원
  - 다중 AZ로 ElastiCache for Redis의 가동 중지 시간 최소화

REL 4: 장애 조치를 어떻게 효과적으로 계획하나요?

질문 수준의 소개: 자동 장애 조치가 포함된 다중 AZ를 활성화하는 것이 ElastiCache의 모범 사례입니다. 경우에 따라 ElastiCache for Redis는 서비스 운영의 일부로 프라이머리 노드를 대체합니다. 예를

들면 계획된 유지 관리 이벤트 및 드물지만 노드 장애나 가용 영역 문제가 발생하는 경우가 포함됩니다. 성공적인 장애 조치는 ElastiCache와 클라이언트 라이브러리 구성에 달려 있습니다.

질문 수준의 이점: 특정 ElastiCache for Redis 클라이언트 라이브러리와 함께 ElastiCache 장애 조치의 모범 사례를 따르면 장애 조치 이벤트 중에 발생할 수 있는 가동 중지 시간을 최소화할 수 있습니다.

- [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 주로 ElastiCache for Redis 클라이언트 라이브러리의 구성 설정을 통해 설정할 수 있으며, 구성 설정에서 새로 고침 빈도와 방법도 구성할 수 있습니다. 각 클라이언트 라이브러리는 자체 설정을 제공하며 자세한 내용은 해당 설명서에서 확인할 수 있습니다.

  [리소스]:

  - 다중 AZ로 ElastiCache for Redis의 가동 중지 시간 최소화
  - ElastiCache for Redis 클라이언트 라이브러리의 모범 사례를 검토하세요.
- [필수] 성공적인 장애 조치는 프라이머리 노드와 복제본 노드 간의 정상적인 복제 환경에 달려 있습니다. Redis 복제의 비동기적 특성 및 프라이머리 노드와 복제본 노드 간의 복제 지연을 보고하는 데 사용할 수 있는 CloudWatch 지표를 검토하고 이해합니다. 더 높은 데이터 안전성이 필요한 사용 사례의 경우 Redis WAIT 명령을 활용하여 연결된 클라이언트에 응답하기 전에 복제본이 쓰기를 강제로 확인하도록 합니다.

  [리소스]:

  - Redis 지표
  - Amazon CloudWatch를 사용하는 Amazon ElastiCache for Redis 모니터링 모범 사례
- [가장 좋음] ElastiCache 테스트 장애 조치 API를 사용하여 장애 조치 중에 애플리케이션의 응답성을 정기적으로 검증합니다.

  [리소스]:

  - Amazon ElastiCache for Redis의 읽기 전용 복제본에 대한 자동 장애 조치 테스트
  - 자동 장애 조치 테스트

REL 5: ElastiCache 구성 요소가 규모를 조정하도록 설계되었나요?

질문 수준의 소개: 규모 조정 기능과 사용 가능한 배포 토폴로지를 이해하면 변화하는 워크로드 요구 사항에 맞게 ElastiCache 구성 요소를 시간이 지남에 따라 조정할 수 있습니다. ElastiCache는 스케일 인/아웃(수평)과 스케일 업/다운(수직)의 4방향 규모 조정을 제공합니다.

질문 수준의 이점: ElastiCache 배포의 모범 사례를 따르면 규모 조정의 유연성이 극대화될 뿐만 아니라 수평적 규모 조정이라는 Well-Architected 원칙을 충족하여 장애의 영향을 최소화할 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.
  - READIOPS 향상: 복제본 추가
  - WRITEOPS 향상: 샤드 추가(스케일 아웃)
  - 네트워크 I/O 향상: 네트워크에 최적화된 인스턴스 사용, 스케일 업
- [가장 좋음] 클러스터 모드를 활성화한 상태로 ElastiCache 구성 요소를 배포합니다. 더 적은 수의 더 큰 노드보다는 더 많은 수의 더 작 은 노드를 사용합니다. 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. ElastiCache는 클러스터 모드가 비활성화된 상태에서 최대 5개의 읽기 전용 복제본과 수직적 규모 조정을 지원합니다.
- [리소스]:
  - ElastiCache for Redis 클러스터 크기 조정
  - 온라인 스케일 업
  - ElastiCache for Memcached 클러스터 크기 조정

Amazon ElastiCache Well-Architected Lens 성능 효율성 요소

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?
- PE 2: ElastiCache 클러스터 노드에 작업을 어떻게 분배하고 있나요?
- PE 3: 캐싱 워크로드의 경우, 캐시의 효율성과 성능을 어떻게 추적하고 보고하나요?
- PE 4: 워크로드가 네트워킹 리소스 및 연결 사용을 어떻게 최적화하나요?
- PE 5: 키 삭제 또는 제거를 어떻게 관리하나요?
- PE 6: ElastiCache에서 어떻게 데이터를 모델링하고 데이터와 상호 작용하나요?
- PE 7: Amazon ElastiCache 클러스터에서 느리게 실행되는 명령을 어떻게 로깅하나요?
- PE8: Auto Scaling은 ElastiCache 클러스터의 성능을 높이는 데 어떻게 도움이 되나요?

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.
- [가장 좋음] ElastiCache for Redis 워크로드의 경우, 사용자가 프로덕션 클러스터에서 차단 명령을 실행하지 못하도록 KEYS와 같이 계산 비용이 많이 드는 명령의 이름을 변경하세요.
  - 엔진 6.x를 실행하는 ElastiCache for Redis 워크로드는 역할 기반 액세스 제어를 활용하여 특정 명령을 제한할 수 있습니다. AWS 콘솔이나 CLI를 사용하여 사용자 및 사용자 그룹을 생성하고 사용자 그룹을 ElastiCache for Redis 클러스터에 연결하여 명령에 대한 액세스를 제어할 수 있습니다. Redis 6에서는 RBAC가 활성화되면 '-@dangerous'를 사용할 수 있으며, 이는 해당 사용자가 KEYS, MONITOR, SORT 등과 같이 비용이 많이 드는 명령을 사용하는 것을 허용하지 않습니다.
  - 엔진 버전 5.x의 경우 Amazon ElastiCache for Redis 클러스터 파라미터 그룹의 `rename-commands` 파라미터를 사용하여 명령의 이름을 변경합니다.
- [더 좋음] 느린 쿼리를 분석하고 최적화 기법을 찾아봅니다.
  - ElastiCache for Redis 워크로드의 경우 느린 로그를 분석하여 쿼리에 대해 자세히 알아보세요. 예를 들어 `redis-cli slowlog get 10` 명령을 사용하여 지연 시간 임계값(기본값 10초)을 초과한 마지막 10개의 명령을 표시할 수 있습니다.

  - 복잡한 ElastiCache for Redis 데이터 구조를 사용하면 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 for Redis 워크로드에서 `redis-benchmark`는 클라이언트 수, 데이터 크기와 같이 사용자가 정의한 입력을 사용하여 다양한 명령의 성능을 테스트할 수 있는 간단한 인터페이스를 제공합니다.
  - Memcached는 간단한 키 수준 명령만 지원하므로 클라이언트 쿼리를 처리하기 위해 키 공간을 반복하지 않도록 추가 키를 인덱스로 구축하는 것이 좋습니다.
- [리소스]:
  - CloudWatch 지표를 사용한 사용량 모니터링
  - CloudWatch 지표를 사용한 사용량 모니터링
  - Amazon CloudWatch 경보 사용
  - Redis 특정 파라미터
  - SLOWLOG
  - Redis 벤치마크

PE 2: ElastiCache 클러스터 노드에 작업을 어떻게 분배하고 있나요?

질문 수준의 소개: 애플리케이션이 Amazon ElastiCache 노드에 연결하는 방식이 클러스터의 성능 및 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 연결되도록 합니다.
  - Amazon ElastiCache for Redis는 사용 중인 클러스터 모드에 따라 각기 다른 엔드포인트를 구현합니다. 클러스터 모드가 활성화된 경우 ElastiCache는 구성 엔드포인트를 제공합니다. 클러스터 모드가 비활성화된 경우 ElastiCache는 일반적으로 쓰기에 사용되는 기본 엔드포인트와 복제본 간에 읽기 밸런싱을 위한 리더 엔드포인트를 제공합니다. 이러한 엔드포인트를 올바르게 구현하면 성능이 향상되고 확장 작업이 더 쉬워집니다. 특별한 요구 사항이 없는 한 개별 노드 엔드포인트에 연결하지 마시기 바랍니다.
  - 다중 노드 Memcached 클러스터의 경우 ElastiCache는 자동 검색을 지원하는 구성 엔드포인트를 제공합니다. 캐시 노드 전체에 작업을 균등하게 분배하려면 해싱 알고리즘을 사용하는 것이 좋습니다. 많은 Memcached 클라이언트 라이브러리는 일관된 해싱을 구현합니다. 사용할 라이브러리

의 설명서에서 일관적 해싱 지원 여부와 그 구현 방법을 참조하세요. 이러한 기능 구현에 대한 자세한 내용은 여기에서 확인할 수 있습니다.

- [더 좋음] 워크로드의 단축키를 식별하고 정정하기 위한 전략을 구현합니다.
  - 목록, 스트림, 세트 등과 같은 다차원 Redis 데이터 구조의 영향을 고려하세요. 이러한 데이터 구조는 단일 노드에 있는 단일 Redis 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 for Redis 워크로드의 경우, LFU 최대 메모리 정책이 설정되어 있다면 `redis-cli --hotkeys`를 사용하여 단축키를 감지할 수 있습니다.
  - 여러 노드에 핫키를 복제하여 액세스를 더 균등하게 분산하는 것을 고려해 보세요. 이 방법을 사용하려면 클라이언트가 여러 기본 노드에 쓰고(Redis 노드 자체에서는 이 기능을 제공하지 않음) 원래 키 이름 외에도 읽을 키 이름 목록을 유지 관리해야 합니다.
  - ElastiCache for Redis 버전 6는 서버의 도움을 받는 클라이언트 측 캐싱을 지원합니다. 이렇게 하면 애플리케이션이 키 변경을 기다린 후 ElastiCache로 다시 네트워크를 호출할 수 있습니다.
- [리소스]:
  - 더 높은 가용성을 위해 Amazon ElastiCache for Redis 구성
  - 연결 엔드포인트 찾기
  - 로드 밸런싱 모범 사례
  - Redis의 클라이언트 측 캐싱

PE 3: 캐싱 워크로드의 경우, 캐시의 효율성과 성능을 어떻게 추적하고 보고하나요?

질문 수준의 소개: 캐싱은 ElastiCache에서 흔히 볼 수 있는 워크로드이므로 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]:

- [필수] 지연 시간 및 CPU 사용률 값과 함께 애플리케이션 캐시 성능을 측정 및 수집하여 TTL(Time To Live) 또는 기타 애플리케이션 구성 요소를 조정해야 하는지 파악합니다. ElastiCache는 각 데이터 구조에 집계된 지연 시간에 대한 CloudWatch 지표 세트를 제공합니다. 이러한 지연 시간 지표는 ElastiCache for Redis INFO 명령의 commandstats 통계를 사용하여 계산되며 네트워크 및 I/O 시간은 포함되지 않습니다. 이는 ElastiCache for Redis가 작업을 처리하는 데 소요되는 시간일 뿐입니다.

  [리소스]:

  - Amazon CloudWatch를 사용하는 Amazon ElastiCache for Redis 모니터링 모범 사례
- [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]:

  - 캐싱 전략
  - 캐싱 모범 사례
  - Amazon ElastiCache를 통한 규모에 따른 성능 백서

PE 4: 워크로드가 네트워킹 리소스 및 연결 사용을 어떻게 최적화하나요?

질문 수준의 소개: ElastiCache for Redis와 Memcached는 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 사전에 관리합니다.
  - 애플리케이션의 연결 풀링은 연결을 열고 닫을 때 생성되는 클러스터의 오버헤드를 줄입니다. `CurrConnections` 및 `NewConnections`를 사용하여 Amazon CloudWatch의 연결 동작을 모니터링하세요.

  - 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]:
  - Amazon ElastiCache for Redis - 글로벌 데이터 스토어
  - Memcached 특정 파라미터
  - I/O 처리를 개선하여 성능을 향상하는 Amazon ElastiCache for Redis 5.0.3
  - 더 높은 가용성을 위해 Amazon ElastiCache for Redis 구성

PE 5: 키 삭제 또는 제거를 어떻게 관리하나요?

질문 수준의 소개: 워크로드는 요구 사항이 각기 다르며 클러스터 노드가 메모리 소비 제한에 근접했을 때 예상되는 동작이 각기 다릅니다. Amazon ElastiCache for Redis에는 이러한 상황을 처리하기 위해 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 for Redis 클러스터 파라미터 그룹에서 `maxmemory-policy`를 사용하여 관리할 수 있습니다.
  - 기본 정책인 `volatile-lru`는 만료 시간(TTL 값)이 설정된 키를 제거하여 메모리를 확보합니다. 가장 낮은 빈도로 사용(LFU) 및 가장 오래 전에 사용(LRU) 정책은 사용 현황에 따라 키를 제거합니다.

  - Memcached 워크로드의 경우, 각 노드의 제거를 제어하는 기본 LRU 정책이 있습니다. Amazon CloudWatch의 제거 지표를 사용하여 Amazon ElastiCache 클러스터의 제거 수를 모니터링할 수 있습니다.
- [더 좋음] 삭제 동작을 표준화하여 클러스터의 성능에 미치는 영향을 제어함으로써 예상치 못한 성능 병목 현상을 방지합니다.
  - ElastiCache for Redis 워크로드의 경우, 클러스터에서 키를 명시적으로 제거하면 `UNLINK`는 마치 `DEL`처럼 지정된 키를 제거합니다. 그러나 이 명령은 다른 스레드에서 실제 메모리 회수를 수행하므로 `DEL`과는 달리 차단하지 않습니다. 실제 제거는 나중에 비동기적으로 수행됩니다.
  - ElastiCache for Redis 6.x 워크로드의 경우, `lazyfree-lazy-user-del` 파라미터를 사용하여 파라미터 그룹에서 `DEL` 명령의 동작을 수정할 수 있습니다.
- [리소스]:
  - 파라미터 그룹을 사용해 엔진 파라미터 구성
  - UNLINK
  - AWS를 통한 클라우드 재무 관리

PE 6: ElastiCache에서 어떻게 데이터를 모델링하고 데이터와 상호 작용하나요?

질문 수준의 소개: ElastiCache는 사용되는 데이터 구조 및 데이터 모델과 관련해서 애플리케이션에 크게 의존하지만 데이터 스토어가 있는 경우 기본 데이터 스토어도 고려해야 합니다. 사용 가능한 ElastiCache for Redis 데이터 구조를 이해하고 필요에 가장 적합한 데이터 구조를 사용하고 있는지 확인하세요.

질문 수준의 이점: ElastiCache의 데이터 모델링에는 애플리케이션 사용 사례, 데이터 유형 및 데이터 요소 간 관계를 비롯한 여러 계층이 있습니다. 또한 각 ElastiCache for Redis 데이터 유형 및 명령에는 잘 문서화된 고유한 성능 시그니처가 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
- [가장 좋음] ElastiCache for Redis 명령에는 Big O 표기법으로 정의된 시간 복잡도가 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 선형이

며 입력 크기의 영향을 받습니다. ElastiCache for Redis의 단일 스레드 설계로 인해 시간 복잡도가 높은 걸리는 작업을 대량으로 수행하면 성능이 저하되고 작업 시간 초과가 발생할 수 있습니다.

[리소스]:

- 명령

- [가장 좋음] API를 사용하여 클러스터의 데이터 모델에 대한 GUI 가시성을 확보합니다.

  [리소스]:

  - Redis Commander
  - Redis Browser
  - Redsmin

PE 7: Amazon ElastiCache 클러스터에서 느리게 실행되는 명령을 어떻게 로깅하나요?

질문 수준의 소개: 성능 튜닝은 오래 실행되는 명령의 캡처, 집계 및 알림에 유용합니다. 명령을 실행하는 데 걸리는 시간을 이해하면 저조한 성능을 초래하는 명령과 엔진이 최적의 성능을 발휘하지 못하게 하는 명령을 파악할 수 있습니다. Amazon ElastiCache for Redis에는 이 정보를 Amazon CloudWatch 또는 Amazon Kinesis Data Firehose에 전달하는 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 for Redis를 사용하고, 파라미터 그룹을 올바르게 구성하며, 클러스터에서 SLOWLOG 로깅을 활성화합니다.
  - 필수 파라미터는 엔진 버전 호환성이 Redis 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.
- [가장 좋음] CloudWatch 또는 Kinesis Data Firehose 기능을 활용합니다.
  - CloudWatch, CloudWatch 로그 인사이트 및 Amazon Simple Notification Services의 필터링 및 경보 기능을 사용하여 성능을 모니터링하고 이벤트 알림을 받습니다.
  - Kinesis Data Firehose의 스트리밍 기능을 사용하여 SLOWLOG 로그를 영구 스토리지에 보관하거나 자동화된 클러스터 파라미터 튜닝을 트리거합니다.

  - JSON과 일반 텍스트 형식 중에서 요구 사항에 가장 적합한 형식을 결정하세요.
  - 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 느리게 실행되는 명령으로 로깅되기 전에 Redis 엔진 내에서 명령을 실행할 수 있는 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 Redis 메모리에 캡처되는 느리게 실행되는 명령 수의 상한선을 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 redis-cli에서 더 오랜 기간 동안 데이터를 분석해야 하는 경우 이 값을 높이는 것이 좋습니다. 이렇게 하면 더 많은 명령을 Redis 메모리에 유지할 수 있습니다.

    SLOWLOG 데이터를 CloudWatch Logs 또는 Kinesis Data Firehose로 내보내는 경우, 데이터가 유지되고 ElastiCache 시스템 외부에서 분석할 수 있으므로 느리게 실행되는 대량의 명령을 Redis 메모리에 저장할 필요가 줄어듭니다.
- [리소스]:
  - ElastiCache for Redis 캐시 클러스터에서 Redis 느린 로그를 켜려면 어떻게 해야 하나요?
  - 로그 전달
  - Redis 특정 파라미터
  - https://aws.amazon.com/cloudwatch/Amazon CloudWatch
  - Amazon Kinesis Data Firehose

PE8: Auto Scaling은 ElastiCache 클러스터의 성능을 높이는 데 어떻게 도움이 되나요?

질문 수준의 소개: Redis Auto Scaling 기능을 구현하면 ElastiCache 구성 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 급증을 이해하고 이에 대한 계획을 세우면 향상된 캐싱 성능과 비즈니스 연속성을 보장할 수 있습니다. ElastiCache for Redis Auto Scaling은 CPU/메모리 사용률을 지속적으로 모니터링하여 클러스터가 원하는 성능 수준으로 작동하는지 확인합니다.

- [필수] ElastiCache for Redis용 클러스터를 시작할 경우:
  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]:

  - Redis(클러스터 모드 활성화됨)에서 클러스터 조정
  - 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. 4주간의 클러스터 사용률에 대한 CloudWatch 지표를 분석하여 대상 값 임계값을 확인할 수 있습니다. 어떤 값을 선택할지 잘 모르는 경우 지원되는 최소 사전 정의 지표 값으로 시작하는 것이 좋습니다.

  [리소스]:

  - CloudWatch 지표를 사용한 사용량 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

- 확장 가능 목표 등록

Amazon ElastiCache의 Well-Architected Lens 비용 최적화 요소

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: ElastiCache 리소스와 관련된 비용을 어떻게 식별하고 추적하나요? 사용자가 리소스를 생성하고 생성된 리소스를 관리 및 폐기할 수 있도록 하는 메커니즘을 어떻게 개발하나요?
- COST 2: ElastiCache 리소스와 관련된 비용을 최적화하는 데 도움이 되는 지속적 모니터링 도구를 어떻게 사용하나요?
- COST 3: 데이터 계층화를 지원하는 인스턴스 유형을 사용해야 하나요? 데이터 계층화의 이점은 무엇인가요? 데이터 계층화 인스턴스를 사용하지 않는 경우는 언제인가요?

COST 1: ElastiCache 리소스와 관련된 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. 개발부터 프로덕션 및 사용 중지에 이르기까지 생성된 리소스를 추적할 수 있는 프로세스와 도구를 갖추면 ElastiCache와 관련된 비용을 관리하는 데 도움이 됩니다.

질문 수준의 이점: 워크로드와 관련된 모든 비용을 지속적으로 추적하려면 ElastiCache를 구성 요소 중 하나로 포함하는 아키텍처에 대한 심층적인 이해가 필요합니다. 또한 사용량을 집계하여 예산과 비교할 수 있는 비용 관리 계획도 마련해 두어야 합니다.

- [필수] 클라우드 혁신 센터(CCoE)를 설립하고, 설립 규범 중 하나로 CCoE가 조직의 ElastiCache 사용과 관련된 지표의 정의, 추적 및 조치 수행을 담당하도록 합니다. CCoE가 존재하고 운영되는 경우 CCoE가 ElastiCache와 관련된 비용을 읽고 추적하는 방법을 알고 있는지 확인하세요. 리소스가 생성되면 IAM 역할 및 정책을 사용하여 특정 팀 및 그룹만 리소스를 인스턴스화할 수 있는지 검증합니

다. 이를 통해 비용을 비즈니스 성과와 연관시키고 비용 관점에서 명확한 책임 범위를 설정할 수 있습니다.

1. CCoE는 다음과 같은 범주형 데이터에서 주요 ElastiCache 사용량을 기준으로 정기적(월간)으로 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. CCoE는 조직 내 소프트웨어 엔지니어링, 데이터 관리, 제품 팀, 재무 및 리더십 팀 등 다양한 팀으로 구성됩니다.

[리소스]:

- 클라우드 혁신 센터 만들기
- Amazon ElastiCache 요금

- [필수] 비용 할당 태그를 사용하여 낮은 수준의 세밀함으로 비용을 추적합니다. AWS Cost Management를 사용하면 시간 경과에 따른 AWS 비용 및 사용량을 시각화하고 이해하며 관리할 수 있습니다.
  1. 태그를 이용하여 리소스를 정리하고, 비용 할당 태그를 이용하여 AWS 비용을 더 자세히 추적합니다. 비용 할당 태그를 활성화하면, AWS는 비용 할당 태그를 이용해 비용 할당 보고서의 리소스 비용을 정리하기 때문에 사용자는 쉽게 AWS 비용을 분류하고 추적할 수 있습니다. AWS는 AWS 생성 태그 및 사용자 정의 태그라는 두 가지 유형의 비용 할당 태그를 제공합니다. AWS는 사용자를 위해 AWS 생성 태그를 정의, 생성 및 적용하며, 사용자는 사용자 정의 태그를 정의, 생성 및 적용합니다. 두 유형의 태그 모두 개별적으로 활성화해야만 Cost Management나 비용 할당 보고서에 표시됩니다.
  2. 비용 할당 태그를 사용하여 자신만의 비용 구조를 반영하도록 AWS 청구서를 구성합니다. Amazon ElastiCache에서 리소스에 비용 할당 태그를 추가하면 인보이스 비용을 리소스 태그 값으로 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
- [가장 좋음] ElastiCache 비용을 조직 전체에 적용되는 지표와 연계합니다.
  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 사용량 모니터링
- [좋음] ElastiCache를 사용하는 전체 워크로드의 지표 및 비용에 대해 최신 아키텍처 가시성과 운영 가시성을 유지합니다.
  1. 전체 솔루션 에코시스템을 이해하세요. ElastiCache는 클라이언트부터 API Gateway, Redshift 및 예를 들면 보고 도구용 QuickSight에 이르기까지 기술 세트의 전체 AWS 서비스 에코시스템에 속하는 경향이 있습니다.
  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 애플리케이션의 운영 및 기능적 ElastiCache 요소뿐만 아니라 컴퓨팅, 네트워킹, 스토리지, 수명 주기 정책, 지표 수집의 사용을 포함해야 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, 책임, 우선순위 지정 및 리소스에 대한 경영진의 지원은 ElastiCache에 대한 효과적인 비용 관리 전략을 수립하는 데 매우 중요합니다.

COST 2: ElastiCache 리소스와 관련된 비용을 최적화하는 데 도움이 되는 지속적 모니터링 도구를 어떻게 사용하나요?

질문 수준의 소개: ElastiCache 비용과 애플리케이션 성능 지표 간의 적절한 균형을 맞추는 것을 목표로 해야 합니다. Amazon CloudWatch는 요구 사항에 따라 ElastiCache 리소스의 활용도가 과도한지, 저조한지 평가하는 데 도움이 되는 주요 운영 지표에 대한 가시성을 제공합니다. 비용 최적화 관점에

서 보면 오버프로비저닝되는 시기를 이해하고 운영, 가용성, 복원력 및 성능 요구 사항을 유지하면서 ElastiCache 리소스 크기를 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. 과도하게 큰 ElastiCache 리소스가 장기간 운영되는 것을 식별하고 이를 방지할 수 있어야 합니다.

- [필수] CloudWatch를 사용하여 ElastiCache 클러스터를 모니터링하고 이러한 지표가 AWSCost Explorer 대시보드와 어떤 관련이 있는지 분석합니다.
  1. ElastiCache는 호스트 수준 지표(예: CPU 사용) 및 캐시 엔진 소프트웨어별 지표(예: 캐시가 얻은 것과 잃은 것) 모두를 제공합니다. 이러한 지표는 60초 간격으로 각 캐시 노드에 대해 측정되어 게시됩니다.
  2. ElastiCache 성능 지표(CPUUtilization, EngineUtilization, SwapUsage, CurrConnections, Evictions)를 확인하여 스케일 업/다운(더 큰/작은 캐시 노드 유형 사용) 또는 스케일 인/아웃(샤드 추가/축소)이 필요하다는 것을 알 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 사용량 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. 예약 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 하나요? 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 나은 가격을 제공합니다. 인스턴스별로 사용 가능한 SSD 스토리지 때문에 데이터 계층화 인스턴스는 인스턴스 기능당 훨씬 더 높은 총 데이터 용량을 지원합니다.

- [필수] 데이터 계층화 인스턴스의 한계를 이해합니다.
  1. ElastiCache for Redis 클러스터에만 사용할 수 있습니다.
  2. 제한된 인스턴스 유형만 데이터 계층화를 지원합니다.
  3. ElastiCache for Redis 버전 6.2 이상만 지원됩니다.
  4. 대형 항목은 SSD로 교체되지 않습니다. 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. 자주 사용하지 않는 객체는 로컬 SSD로 교체되므로 액세스 빈도가 낮은 객체에는 약간의 성능 비용이 부과됩니다. 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 데이터 계층화 인스턴스가 아닌 인스턴스를 사용하여 ElastiCache 클러스터를 운영해야 할 수도 있습니다.
  2. ElastiCache 클러스터의 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

일반적인 문제 해결 단계 및 모범 사례

주제

- 연결 문제
- Redis 클라이언트 오류
- 서버리스의 ElastiCache 높은 지연 시간 문제 해결
- 서버리스의 스로틀링 문제 해결 ElastiCache
- 관련 항목

연결 문제

ElastiCache 캐시에 연결할 수 없는 경우 다음 중 하나를 고려해 보십시오.

1. TLS 사용: ElastiCache 엔드포인트에 연결하려고 할 때 연결이 중단되는 경우 클라이언트에서 TLS를 사용하고 있지 않을 수 있습니다. ElastiCache 서버리스를 사용하는 경우 전송 중 암호화가 항상 활성화됩니다. 클라이언트가 TLS를 사용하여 캐시에 연결하고 있는지 확인하십시오. TLS 지원 캐시에 연결하는 방법에 대한 자세한 내용은 여기를 참조하십시오.
2. VPC: ElastiCache 캐시는 VPC 내에서만 액세스할 수 있습니다. 캐시에 액세스하는 EC2 인스턴스와 캐시가 동일한 ElastiCache VPC에 생성되었는지 확인하십시오. 또는 EC2 인스턴스가 있는 VPC와 캐시를 생성하는 VPC 간의 VPC 피어링을 활성화해야 합니다.
3. 보안 그룹: 보안 그룹을 ElastiCache 사용하여 캐시에 대한 액세스를 제어합니다. 다음을 고려하세요.
   a. ElastiCache 캐시에 사용되는 보안 그룹이 EC2 인스턴스에서의 인바운드 액세스를 허용하는지 확인하십시오. 보안 그룹에서 인바운드 규칙을 올바르게 설정하는 방법을 알아보려면 여기를 참조하십시오.
   b. 캐시에서 사용하는 보안 그룹이 ElastiCache 캐시 포트에 대한 액세스를 허용하는지 확인하십시오 (서버리스의 경우 6379 및 6380, 자체 설계의 경우 기본적으로 6379). ElastiCache 이러한 포트를 사용하여 Redis 명령을 수락합니다. 여기에서 포트 액세스를 설정하는 방법에 대해 자세히 알아보세요.

Redis 클라이언트 오류

ElastiCache 서버리스는 Redis 클러스터 모드 프로토콜을 지원하는 Redis 클라이언트를 통해서만 액세스할 수 있습니다. 자체 설계된 클러스터는 클러스터 구성에 따라 어느 모드에서든 Redis 클라이언트에서 액세스할 수 있습니다.

클라이언트에서 Redis 오류가 발생하는 경우 다음 사항을 고려하십시오.

1. 클러스터 모드: SELECT Redis 명령에 CROSSLOT 오류나 오류가 발생하는 경우 Redis 클러스터 프로토콜을 지원하지 않는 Redis 클라이언트에서 클러스터 모드가 활성화된 캐시에 액세스하려고 할 수 있습니다. ElastiCache 서버리스는 Redis 클러스터 프로토콜을 지원하는 Redis 클라이언트만 지원합니다. Redis를 “클러스터 모드 비활성화" (CMD) 에서 사용하려면 자체 클러스터를 설계해야 합니다.
2. CROSSLOT **`ERR CROSSLOT Keys in request don't hash to the same slot`** 오류: 오류가 발생하는 경우 클러스터 모드 캐시에서 동일한 슬롯에 속하지 않는 키에 액세스하려고 할 수

있습니다. 다시 말씀드리지만, ElastiCache 서버리스는 항상 클러스터 모드에서 작동합니다. 여러 키가 포함된 다중 키 작업, 트랜잭션 또는 Lua 스크립트는 관련된 모든 키가 동일한 해시 슬롯에 있는 경우에만 허용됩니다.

Redis 클라이언트 구성과 관련된 추가 모범 사례는 이 블로그 게시물을 검토하세요.

서버리스의 ElastiCache 높은 지연 시간 문제 해결

워크로드에 지연 시간이 긴 것으로 보이는 경우 CloudWatch `SuccessfulReadRequestLatency` 및 `SuccessfulWriteRequestLatency` 지표를 분석하여 지연 시간이 ElastiCache 서버리스와 관련이 있는지 확인할 수 있습니다. 이러한 지표는 ElastiCache 서버리스 내부의 지연 시간을 측정합니다. 클라이언트측 지연 시간과 클라이언트와 ElastiCache 서버리스 엔드포인트 간의 네트워크 트립 시간은 포함되지 않습니다.

클라이언트측 지연 문제 해결

클라이언트측 지연 시간이 길어졌으나 이에 상응하는 CloudWatch `SuccessfulReadRequestLatency` 증가와 서버 측 지연 시간을 측정하는 `SuccessfulWriteRequestLatency` 지표가 없는 경우 다음 사항을 고려해 보십시오.

- 보안 그룹이 포트 6379 및 6380에 대한 액세스를 허용하는지 확인하십시오. ElastiCache 서버리스는 6379 포트를 기본 엔드포인트로 사용하고 6380 포트를 리더 엔드포인트로 사용합니다. 애플리케이션에서 Read from Replica 기능을 사용하지 않는 경우에도 일부 클라이언트는 새로 연결할 때마다 두 포트에 모두 연결을 설정합니다. 보안 그룹이 두 포트에 대한 인바운드 액세스를 허용하지 않는 경우 연결 설정 시간이 더 오래 걸릴 수 있습니다. 포트 액세스를 설정하는 방법에 대한 자세한 내용은 여기를 참조하십시오.

서버 측 지연 문제 해결

일부 변동성과 가끔 발생하는 스파이크는 걱정할 필요가 없습니다. 그러나 `Average` 통계가 급격히 증가하고 계속 유지되는 경우 Personal Health Dashboard에서 AWS Health Dashboard 자세한 내용을 확인해야 합니다. 필요한 경우 를 통해 지원 사례를 여는 것을 고려해 보십시오. AWS Support

다음과 같은 모범 사례와 전략을 고려하여 지연 시간을 줄이십시오.

- 복제본에서 읽기 활성화: 애플리케이션에서 허용하는 경우 Redis 클라이언트에서 '복제본에서 읽기' 기능을 활성화하여 읽기를 확장하고 지연 시간을 줄이는 것이 좋습니다. 활성화되면 ElastiCache 서버리스는 읽기 요청을 클라이언트와 동일한 가용 영역 (AZ) 에 있는 복제 캐시 노드로 라우팅하여 AZ 간 네트워크 지연 시간을 방지합니다. 참고로, 클라이언트에서 Read from Replica 기능을 활성화

하면 애플리케이션이 최종 데이터 일관성을 수용한다는 의미입니다. 키에 쓴 후 읽기를 시도하면 애플리케이션이 한동안 오래된 데이터를 수신할 수 있습니다.

- 애플리케이션이 캐시와 동일한 AZ에 배포되었는지 확인: 애플리케이션이 캐시와 동일한 AZ에 배포되지 않으면 클라이언트 측 지연 시간이 더 길어질 수 있습니다. 서버리스 캐시를 생성할 때 애플리케이션이 캐시에 액세스할 서브넷을 제공할 수 있으며, ElastiCache 서버리스는 해당 서브넷에 VPC 엔드포인트를 생성합니다. 애플리케이션이 동일한 AZ에 배포되었는지 확인하십시오. 그렇지 않으면 애플리케이션이 캐시에 액세스할 때 AZ 간 홉이 발생하여 클라이언트 측 지연 시간이 길어질 수 있습니다.
- 연결 재사용: ElastiCache 서버리스 요청은 RESP 프로토콜을 사용하는 TLS 지원 TCP 연결을 통해 이루어집니다. 연결 시작 (구성된 경우 연결 인증 포함) 에는 시간이 걸리므로 첫 번째 요청의 지연 시간이 평소보다 깁니다. 이미 초기화된 연결을 통한 요청은 ElastiCache 일관되게 낮은 지연 시간을 제공합니다. 이러한 이유로 연결 풀링을 사용하거나 기존 Redis 연결을 재사용하는 것을 고려해야 합니다.
- 속도 조정: ElastiCache 서버리스는 요청 속도가 증가함에 따라 자동으로 확장됩니다. ElastiCache 서버리스가 확장되는 속도보다 빠른 속도로 요청 속도가 갑자기 크게 증가하면 한동안 지연 시간이 길어질 수 있습니다. ElastiCache 서버리스는 일반적으로 지원되는 요청 속도를 빠르게 높일 수 있으며, 요청 속도를 두 배로 늘리는 데 최대 10~12분이 걸립니다.
- 장기 실행 명령 검사: Lua 스크립트 또는 대규모 데이터 구조의 명령을 비롯한 일부 Redis 명령은 오래 실행될 수 있습니다. 이러한 명령을 식별하기 위해 명령어 수준 ElastiCache 메트릭을 게시합니다. ElastiCache 서버리스를 사용하면 메트릭을 BasedECPUs 사용할 수 있습니다.
- 제한된 요청: ElastiCache 서버리스에서 요청이 제한되면 애플리케이션에서 클라이언트 측 지연 시간이 증가할 수 있습니다. 서버리스에서 요청이 병목 현상이 발생하면 ElastiCache 서버리스 지표가 증가하는 것을 확인할 수 있습니다. ThrottledRequests ElastiCache 병목 현상이 발생한 요청의 문제를 해결하려면 아래 섹션을 검토하십시오.
- 키와 요청의 균일한 배포: Redis의 ElastiCache 경우 슬롯당 키 또는 요청이 고르지 않게 분산되면 핫 슬롯이 생성되어 지연 시간이 길어질 수 있습니다. ElastiCache 서버리스는 간단한 SET/GET 명령을 실행하는 워크로드에서 단일 슬롯에서 초당 최대 30,000ECPU (복제본에서 읽기 사용 시 초당 90,000 ECPU) 를 지원합니다. 슬롯 전반에 걸친 키 및 요청 분배를 평가하고 요청 비율이 이 한도를 초과하는 경우 균일하게 분배되도록 하는 것이 좋습니다.

서버리스의 스로틀링 문제 해결 ElastiCache

서비스 지향 아키텍처 및 분산 시스템에서는 다양한 서비스 구성 요소가 API 호출을 처리하는 속도를 제한하는 것을 제한이라고 합니다. 이렇게 하면 스파이크가 완화되고 구성 요소 처리량 불일치를 제어하며 예상치 못한 운영 이벤트가 발생할 경우 보다 예측 가능한 복구가 가능합니다. ElastiCache 서버

리스는 이러한 유형의 아키텍처에 맞게 설계되었으며, 대부분의 Redis 클라이언트에는 병목 현상이 발생한 요청에 대한 재시도 기능이 내장되어 있습니다. 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.

서버리스에서 요청이 병목 현상이 발생하면 ElastiCache 서버리스 지표가 증가하는 것을 확인할 수 있습니다. `ThrottledRequests` ElastiCache 요청 수가 제한되는 경우가 많으면 다음 사항을 고려하십시오.

- 조정 속도: ElastiCache 서버리스는 더 많은 데이터를 수집하거나 요청 비율이 증가하면 자동으로 확장됩니다. 서버리스가 확장되는 속도보다 빠르게 애플리케이션이 확장되는 경우 ElastiCache 서버리스가 워크로드에 맞게 확장되는 동안 요청이 병목 현상을 겪을 수 있습니다. ElastiCache ElastiCache 서버리스는 일반적으로 스토리지 크기를 빠르게 늘릴 수 있으며, 캐시의 스토리지 크기를 두 배로 늘리는 데 최대 10~12분이 소요됩니다.
- 키와 요청의 균일한 배포: Redis의 ElastiCache 경우 슬롯당 키 또는 요청이 고르지 않게 분산되면 핫 슬롯이 생성될 수 있습니다. 간단한 SET/GET 명령을 실행하는 워크로드에서 단일 슬롯에 대한 요청 속도가 초당 30,000 ECPU를 초과하는 경우 핫 슬롯으로 인해 요청이 제한될 수 있습니다.
- 복제본에서 읽기: 애플리케이션에서 허용하는 경우 “복제본에서 읽기” 기능을 사용하는 것이 좋습니다. 대부분의 Redis 클라이언트는 읽기를 복제본 노드로 직접 보내도록 '읽기를 확장'하도록 구성할 수 있습니다. 이 기능을 사용하면 읽기 트래픽을 확장할 수 있습니다. 또한 ElastiCache 서버리스는 복제본 요청의 읽기를 애플리케이션과 동일한 가용 영역의 노드로 자동 라우팅하므로 지연 시간이 줄어듭니다. Read from Replica를 활성화하면 간단한 SET/GET 명령으로 워크로드에 대해 단일 슬롯에서 초당 최대 90,000 ECPU를 달성할 수 있습니다.

관련 항목

- 추가 문제 해결 단계
- the section called “모범 사례 및 캐싱 전략”

추가 문제 해결 단계

지속적 연결 문제를 해결하는 동안 다음 항목을 확인해야 합니다 ElastiCache.

주제

- 보안 그룹
- 네트워크 ACL

- 라우팅 테이블
- DNS 확인
- 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 (EC2 인스턴스, AWS Lambda 함수, Amazon ECS 컨테이너 등)와 캐시를 보호하는 가상 방화벽입니다. ElastiCache 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 측에서 한도에 도달했는지 확인하는 방법에 대한 도움말은 문제 해결 섹션을 참조하십시오. ElastiCache

클라이언트와 ElastiCache 클러스터에 단일 보안 그룹을 동시에 할당하거나 각각에 대해 개별 보안 그룹을 할당할 수 있습니다.

두 경우 모두 소스에서 나오는 ElastiCache 포트의 TCP 아웃바운드 트래픽과 동일한 포트의 인바운드 트래픽을 허용해야 합니다. ElastiCache 기본 포트는 Memcached의 경우 11211, Redis의 경우 6379입니다. 기본적으로 보안 그룹은 모든 아웃바운드 트래픽을 허용합니다. 이 경우 대상 보안 그룹의 인바운드 규칙만 필요합니다.

자세한 내용은 Amazon VPC의 ElastiCache 클러스터에 액세스하기 위한 액세스 패턴을 참조하십시오.

네트워크 ACL

네트워크 액세스 제어 목록(ACL)은 무상태 규칙입니다. 트래픽이 성공하려면 양방향(인바운드 및 아웃바운드)으로 허용되어야 합니다. 네트워크 ACL은 특정 리소스가 아닌 서브넷에 할당됩니다. 특히 클라

이언트 리소스가 동일한 서브넷에 있는 경우 동일한 ElastiCache ACL과 클라이언트 리소스를 할당할 수 있습니다.

기본적으로 네트워크 ACL은 모든 트래픽을 허용합니다. 그러나 트래픽을 거부하거나 허용하도록 네트워크 ACL을 사용자 지정할 수 있습니다. 또한 ACL 규칙의 평가는 순차적입니다. 즉, 트래픽과 일치하는 가장 빠른 번호의 규칙이 트래픽을 허용하거나 거부합니다. Redis 트래픽을 허용하는 최소 구성은 다음과 같습니다.

클라이언트 측 네트워크 ACL:

- 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 클러스터 서브넷에 대한 개별 규칙 생성) ElastiCache
- 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 클러스터 서브넷) ElastiCache 특정 IP를 사용하면 장애 조치 또는 클러스터 규모 조정 시 문제가 발생할 수 있다는 점에 유의하세요.
- 허용/거부: 허용

ElastiCache 네트워크 ACL:

- 인바운드 규칙:
- 규칙 번호: 모든 거부 규칙보다 번호가 작아야 합니다.
- 유형: 사용자 지정 TCP 규칙
- 프로토콜: TCP
- 포트 범위: 6379

- 출처: 0.0.0.0/0 (또는 클러스터 서브넷에 대한 개별 규칙 생성) ElastiCache
- 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 클러스터 서브넷) ElastiCache 특정 IP를 사용하면 장애 조치 또는 클러스터 규모 조정 시 문제가 발생할 수 있다는 점에 유의하세요.
- 허용/거부: 허용

자세한 내용은 네트워크 ACL을 참조하세요.

라우팅 테이블

네트워크 ACL과 마찬가지로 각 서브넷에는 서로 다른 라우팅 테이블이 있을 수 있습니다. 클라이언트와 ElastiCache 클러스터가 서로 다른 서브넷에 있는 경우 라우팅 테이블을 통해 서로 연결할 수 있는지 확인하십시오.

여러 VPC, 동적 라우팅 또는 네트워크 방화벽이 관련된 복잡한 환경에서는 문제 해결이 어려울 수 있습니다. 네트워크 연결 검증 섹션을 참조하여 네트워크 설정이 적절한지 확인하세요.

DNS 확인

ElastiCache 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 않는 경우) 은 ElastiCache 제공된 DNS 이름을 인식하지 못할 수 있습니다. 시스템이 `dig` (아래 그림 참조) 또는 같은 시스템 도구를 사용하여 ElastiCache 엔드포인트를 성공적으로 해결할 수 있는지 확인하세요. `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 서비스를 통해 이름 확인을 강제할 수도 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 메트릭과 런타임 정보는 연결 문제의 잠재적 원인을 식별하기 위한 공통 소스 또는 정보입니다. 좋은 분석은 일반적으로 다음과 같은 항목으로 시작됩니다.

- CPU 사용량: Redis는 다중 스레드 애플리케이션입니다. 그러나 각 명령의 실행은 단일(주) 스레드에서 발생합니다. 이러한 이유로 `CPUUtilization` 메트릭과 `EngineCPUUtilization`를 ElastiCache 제공합니다. `EngineCPUUtilization`Redis 프로세스 전용 CPU 사용률과 모든 `CPUUtilization` vCPU의 사용량을 제공합니다. 두 개 이상의 vCPU가 있는 노드는 대개 `CPUUtilization` 및 `EngineCPUUtilization`의 값이 서로 다르며, 일반적으로 두 번째 값이 더 큽니다. 높은 `EngineCPUUtilization` 값은 많은 수의 요청이나 완료하는 데 상당한 양의 CPU 시간이 필요한 복잡한 작업으로 인해 발생할 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`: `EngineCPUUtilization`과 상관 관계가 있는 이 두 지표는 `SetTypeCmds`에 의해 측정되는 쓰기 요청 또는 `GetTypeCmds`에 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 모드가 비활성화된 클러스터에서는 리더 엔드포인트를 사용하여 애플리케이션에서 추가 연결 구성을 생성하여 읽기 전용 복제본을 사용할 수 있습니다. ElastiCache 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 토폴로지와 복제본 노드

를 자동으로 검색하고, READONLY Redis 명령을 통해 읽기 작업을 활성화하고, 복제본에 읽기 요청을 제출할 수 있습니다.

- 많은 수의 연결:
  - `CurrConnections` 및 `NewConnections`: `CurrConnection`은 데이터 포인트 컬렉션 순간에 설정된 연결 수이고, `NewConnections`는 이 기간에 생성된 연결 수를 보여줍니다.

    연결 생성 및 처리에서는 상당한 CPU 오버헤드가 발생합니다. 또한 새 연결을 생성하는 데 필요한 TCP 3방향 핸드셰이크는 전체 응답 시간에 부정적인 영향을 미칩니다.

    ElastiCache 노드가 `NewConnections` 분당 수천 개라는 것은 몇 개의 명령만으로 연결이 생성되고 사용된다는 것을 나타내며, 이는 최적이 아닙니다. 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 `NewConnections`는 가능한 낮아야 합니다. Redis는 작은 수의 currConnections에서 최적의 성능을 제공합니다. currConnections를 대략 수십 개나 수백 개로 유지하면 클라이언트 버퍼와 같은 개별 연결을 지원하기 위한 리소스 사용량과 연결을 제공하기 위한 CPU 사이클을 최소화할 수 있습니다.
- 네트워크 처리량:
  - 대역폭 결정: ElastiCache 노드의 네트워크 대역폭은 노드 크기에 비례합니다. 애플리케이션마다 특성이 다르기 때문에 워크로드에 따라 결과가 달라질 수 있습니다. 예를 들어 크기가 작은 요청의 비율이 높은 애플리케이션은 네트워크 처리량보다 CPU 사용량에 더 많은 영향을 미치는 경향이 있으며, 키가 커질수록 네트워크 사용률도 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 Redis를 사용하여 ElastiCache 여러 샤드에 걸쳐 쓰기를 분산하거나 더 많은 네트워크 기능을 갖춘 노드 유형으로 확장할 수 있는 Redis를 사용하는 것입니다.
  - CloudWatchmetrics `NetworkBytesIn`및 는 각각 노드로 들어오거나 나가는 데이터의 양을 `NetworkBytesOut` 제공합니다. `ReplicationBytes`데이터 복제 전용 트래픽입니다.

자세한 정보는 네트워크 관련 제한 사항을 참조하세요.

- 복잡한 명령: Redis 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 예는KEYS 명령입니다. 이 명령은 전체 키스페이스에 걸쳐 주어진 패턴을 검색하며, 실행 중에 다른 명령의 실행을 차단합니다. Redis는 “Big O” 표기법을 사용하여 명령의 복잡도를 설명합니다.

  Keys 명령은 O(N) 시간 복잡도를 가지며 N은 데이터베이스의 키 수입니다. 따라서 키 수가 클수록 명령 속도가 느려집니다. KEYS는 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 복잡도를 보려면 https://redis.io/commands로 이동하세요.

  잠재적 문제의 예:

- Lua 스크립트: Redis는 서버 측에서 스크립트를 실행할 수 있도록 임베디드 Lua 인터프리터를 제공합니다. Redis의 Lua 스크립트는 엔진 수준에서 실행되며 정의에 따라 원자적입니다. 즉, 한 스크립트가 실행되는 동안 다른 명령이나 스크립트가 실행될 수 없습니다. Lua 스크립트는 Redis 엔진에서 직접 다중 명령, 의사 결정 알고리즘, 데이터 구문 분석 등을 실행할 수 있는 길을 열어줍니다. 스크립트의 원자성과 애플리케이션을 오프로드할 수 있는 기능은 유혹적이지만 스크립트는 작은 작업에 신중하게 사용해야 합니다. ElastiCache에서는 Lua 스크립트의 실행 시간이 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 Redis에서 여러 관련 키 수정 사항의 일관성을 보장하는 다른 방법입니다. 트랜잭션을 사용하면 명령 블록을 실행하여 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, 집합, 정렬된 집합 또는 해시(여러 하위 항목을 포함하는 데이터 구조)가 포함된 경우 상당한 CPU 시간을 소비합니다. 즉, 단일 키를 삭제하는 경우에도 요소가 많은 경우 상당한 시간이 걸릴 수 있습니다. DEL의 대안은 Redis 4부터 사용할 수 있는 비동기 명령인 UNLINK입니다. 가능한 경우 항상 DEL 대신 UNLINK를 사용해야 합니다. Redis 6x부터 이 `lazyfree-lazy-user-del` 매개 변수를 사용하면 DEL 명령이 활성화되었을 때와 같이 UNLINK 동작합니다. ElastiCache 자세한 내용은 Redis 6.0 파라미터 변경 사항 섹션을 참조하세요.
- 여러 키에 작동하는 명령: DEL은 이전에 여러 인수를 허용하는 명령으로 언급되었으며 실행 시간은 이에 직접적으로 비례합니다. 그러나 Redis는 유사하게 작동하는 많은 다른 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 파라미터의 광범위한 목록은 CPU 사용량에 영향을 미칩니다.

  CPU 사용률만으로는 연결 문제의 원인이라고 할 수 없지만 여러 키에 대해 하나 또는 몇 개의 명령을 처리하는 데 너무 많은 시간이 소비되면 다른 요청이 실패하고 전반적인 CPU 사용률이 높아질 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 작동하는 명령: Redis는 데이터 유형에 관계없이 하나 이상의 키에 작동하는 명령도 제공합니다. ElastiCache for Redis는 이러한 명령을 KeyBasedCmds 모니터링하기 위한 메트릭을 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 해

시의 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: 해시 외에도 다른 데이터 구조가 CPU 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 https://redis.io/commands를 참조하세요.

네트워크 연결 검증

DNS 확인, 보안 그룹, 네트워크 ACL 및 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 테스트에는 VPC에서 사용할 수 있는 ElastiCache 노드 중 하나의 ENI ID (엘라스틱 네트워크 인터페이스 식별) 가 필요합니다. 다음을 수행하여 이 ID를 찾을 수 있습니다.

1. https://console.aws.amazon.com/ec2/v2/home?#NIC로 이동합니다.
2. ElastiCache 클러스터 이름 또는 이전 DNS 검증에서 가져온 IP 주소를 기준으로 인터페이스 목록을 필터링합니다.
3. ENI ID를 적어 두거나 따로 저장하세요. 여러 인터페이스가 표시되는 경우 설명을 검토하여 해당 인터페이스가 올바른 ElastiCache 클러스터에 속하는지 확인하고 그 중 하나를 선택합니다.
4. 다음 단계를 진행합니다.
5. https://console.aws.amazon.com/vpc/home? ReachabilityAnalyzer #에서 분석 경로를 생성하고 다음 옵션을 선택합니다.

   - 소스 유형: ElastiCache 클라이언트가 Amazon EC2 인스턴스에서 실행되는 경우 인스턴스를 선택하고, 다른 서비스 (예: awsvpc 네트워크를 사용하는 AWS Fargate Amazon ECS 등) 및 해당 리소스 ID (EC2 인스턴스 또는 ENI ID) 를 사용하는 경우 네트워크 인터페이스를 선택합니다. AWS Lambda
   - 대상 유형: 네트워크 인터페이스를 선택하고 목록에서 ElastiCache ENI를 선택합니다.
   - 대상 포트: Redis의 경우 6379, Memcached의 경우 ElastiCache 11211을 지정합니다. ElastiCache 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 포트에서 TCP 연결을 확인하는 방법: Amazon Linux에서는 패키지로 Nping nmap 제공되며 포트에서 TCP 연결을 테스트할 수 있을 뿐만 아니라 연결을 설정하는 데 필요한 네트워크 왕복 시간을 제공할 수 있습니다. ElastiCache 이를 사용하여 다음과 같이 ElastiCache 클러스터의 네트워크 연결과 현재 지연 시간을 검증할 수 있습니다.

```
$ 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.

nping을 사용하는 테스트는 실패하고 VPC Reachability Analyzer 테스트는 통과한다면 시스템 관리자에게 가능한 호스트 기반 방화벽 규칙, 비대칭 라우팅 규칙 또는 운영 체제 수준의 다른 모든 가능한 제한을 검토하도록 요청하세요.

ElastiCache 콘솔에서 ElastiCache 클러스터 세부 정보에 전송 중 암호화가 활성화되어 있는지 확인합니다. 전송 중 암호화가 활성화되어 있으면 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 많은 출력이 나타날 수 있습니다. 마지막 줄에서 사용할 수 있는 반환 코드를 확인하세요. 값이 0 (ok)여야 합니다. openssl이 다른 값을 반환하면 https://www.openssl.org/docs/man1.0.2/man1/verify.html#DIAGNOSTICS에서 오류의 원인을 확인하세요.

모든 인프라 및 운영 체제 테스트를 통과했지만 애플리케이션이 여전히 연결할 ElastiCache 수 없는 경우 애플리케이션 구성이 설정과 호환되는지 확인하십시오. ElastiCache 일반적인 실수는 다음과 같습니다.

- 애플리케이션이 클러스터 모드를 지원하지 않으며 ElastiCache 클러스터 모드가 ElastiCache 활성화되어 있습니다.
- 애플리케이션이 TLS/SSL을 지원하지 않으며 전송 중 암호화가 활성화되어 ElastiCache 있습니다.
- 애플리케이션은 TLS/SSL을 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 한도는 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 예에서는 동일한 소스에서 동일한 대상 IP (ElastiCache 노드) 및 포트로 28231개의 연결이 허용됩니다. 다음 명령은 특정 ElastiCache 노드 (IP 1.2.3.4) 에 대해 존재하는 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: Redis에 대한 다음 CloudWatch 지표를 확인하여 노드에서 발생할 수 있는 네트워크 한도를 식별하십시오. ElastiCache

  - `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.

안정기가 오랜 기간 동안 관찰되면 이후에 복제 지연, 캐시에 사용되는 바이트 수 증가, 여유 메모리 하락, 높은 스왑 및 CPU 사용량이 뒤따를 가능성이 높습니다. 또한 Amazon EC2 인스턴스에는 ENA 드라이버 지표를 통해 추적할 수 있는 네트워크 제한이 있습니다. 향상된 네트워킹 지원과 ENA 드라이버 2.2.10 이상을 사용하는 Linux 인스턴스에서는 다음 명령을 사용하여 제한 카운터를 검토할 수 있습니다.

```
# ethtool -S eth0 | grep "allowance_exceeded"
```

CPU 사용량

CPU 사용량 지표는 조사의 출발점이며, 다음 항목은 ElastiCache 측면에서 발생할 수 있는 문제의 범위를 좁히는 데 도움이 될 수 있습니다.

- Redis SlowLogs: ElastiCache 기본 구성에는 완료하는 데 10밀리초 이상 걸린 마지막 128개 명령이 보존됩니다. 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 128을 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 UTC 기준으로 12월 26일 19:26:07에 발생했고 완료하는 데 4.8초(4,823밀리초)가 걸렸으며 클라이언트 1.2.3.4에서 요청된 KEYS 명령에 의해 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 다음과 같은 Windows에서 사용할 수도 있습니다. 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 복잡도가 있는 명령이 자주 보고되는 경우 앞서 언급한 높은 CPU 사용량에 대한 다른 요인을 확인하세요.

- 지연 시간 지표: ElastiCache for Redis는 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 스크립트 명령 관련,,; `eval` `evalsha`
  - GeoSpatialBasedCmdsLatency: `geodist`, `geohash`, `geopos`, `georadius`, `georadiusbymember`, `geoadd`;
  - GetTypeCmdsLatency: 데이터 유형에 관계없이 명령을 읽습니다.
  - HashBasedCmdsLatency: `hexists`, `hget`, `hgetall`, `hkeys`, `hlen`, `hmget`, `hvals`, `hstrlen`, `hdel`, `hincrby`, `hincrbyfloat`, `hmset`, `hset`, `hsetnx`;
  - HyperLogLogBasedCmdsLatency: `pfselftest`, `pfcount`, `pfdebug`, `pfadd`, `pfmerge`;
  - KeyBasedCmdsLatency: 다양한 데이터 유형에 사용할 수 있는 명령: `dumpexists`,`keys`,`object`,`pttl`,`randomkey`,`ttl`, `typedel`,`expire`,`expireat`,`move`,`persist`,`pexpire`,`pexpireat`, `renamerenamenx`,`restoreK`,`sort`,`unlink`,
  - ListBasedCmdsLatency: 인덱스, 렌즈, 오렌지, 블팝, 브로퍼, 브로폴푸시, 인서트, 루프, 푸시, 플럭스, lem, ltrim, rpop, rpush, rpush, rpush, rpush, rpush
  - PubSubBasedCmdsLatency: 구독, 게시, 게시, 구독 취소, 구독 취소
  - SetBasedCmdsLatency: `scard`, `sdiff`, `sinter`, `sismember`, `smembers`, `srandmember`, `sunion`, `sadd`, `sdiffstore`, `sinterstore`, `smove`, `spop`, `srem`, `sunionstore`;
  - SetTypeCmdsLatency: 데이터 유형에 관계없이 명령을 작성합니다.
  - 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 런타임 명령:
  - info commandstats: Redis 엔진이 시작된 이후 실행된 명령, 누적 실행 횟수, 총 실행 시간 및 명령당 평균 실행 시간으로 구성된 목록을 제공합니다.
  - client list: 현재 연결된 클라이언트와 버퍼 사용량, 마지막으로 실행된 명령 등과 같은 관련 정보의 목록을 제공합니다.
- 백업 및 복제: ElastiCache 2.8.22 이전의 Redis 버전에서는 포크 프로세스를 사용하여 백업을 생성하고 복제본과의 전체 동기화를 처리합니다. 이 방법은 쓰기 집약적인 사용 사례에서 상당한 메모리 오버헤드가 발생시킬 수 있습니다.

  ElastiCache Redis 2.8.22부터 포크 없는 백업 및 복제 방법이 도입되었습니다. AWS 새로운 방법은 실패를 방지하기 위해 쓰기를 지연시킬 수 있습니다. 두 방법 모두 CPU 사용률을 높일 수 있고 응답 시간이 길어지게 만들 수 있으므로 실행 중에 클라이언트 시간 초과가 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

Redis 구성의 기본값은 ElastiCache 클라이언트 연결을 무기한 설정된 상태로 유지합니다. 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 동작으로 인해 클라이언트 또는 측에서 포화 상태가 발생할 수 있습니다. ElastiCache 클라이언트 측에서 즉각적인 수정이 불가능한 경우 일부 관리자는 ElastiCache 파라미터 그룹에 “timeout” 값을 설정합니다. 시간 초과는 유휴 연결이 지속되도록 허용된 기간(초)입니다. 클라이언트가 이 기간 동안 요청을 제출하지 않으면 Redis 엔진은 연결이 시간 초과 값에 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. Redis 구성의 기본값은 ElastiCache 일반 클라이언트 출력 버퍼의 크기를 제한하지 않습니다. `maxmemory` 제한에 도달하면 엔진은 버퍼 사용량을 충족하기 위해 항목을 제거하려고 시도합니다. 메모리가 극도로 부족한 상황에서 ElastiCache for Redis는 메모리를 확보하고 클러스터의 상태를 유지하기 위해 대용량 클라이언트 출력 버퍼를 사용하는 클라이언트의 연결을 끊을 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 연결의 소프트 리밋. 에 정의된 시간 (초) 보다 오랫동안 소프트 한도를 normal-soft-seconds 초과하거나 하드 제한에 client-output-buffer-limit 도달하면 연결이 종료됩니다.
- client-output-buffer-limit-normal-soft-seconds: -를 초과하는 연결에 허용되는 시간 client-output-buffer-limit; normal-soft-limit
- client-output-buffer-limit-normal-hard-limit: 이 한도에 도달하면 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 부하 및 응답 속도도 요청에 영향을 줄 수 있습니다. ElastiCache 간헐적인 연결성 또는 시간 초과 문제를 해결하는 동안 EC2 인스턴스 및 운영 체제 제한을 신중하게 검토해야 합니다. 관찰할 몇 가지 핵심 사항:

- CPU:
  - EC2 인스턴스 CPU 사용량: CPU가 포화되거나 100%에 가까워지지 않았는지 확인합니다. 기록 분석은 다음을 통해 CloudWatch 수행할 수 있지만 데이터 포인트의 세분성은 1분 (세부 모니터링이 활성화된 경우) 또는 5분이라는 점에 유의하십시오.
  - 버스트 가능 EC2 인스턴스를 사용하는 경우 CPU 크레딧 밸런스가 고갈되지 않았는지 확인하세요. 이 정보는 지표에서 `CPUCreditBalance` CloudWatch 확인할 수 있습니다.

- 짧은 기간 동안 CPU 사용량이 높으면 사용률 100% 를 반영하지 않고 시간 초과가 발생할 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 EC2인스턴스 유형 참조
  - 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: 인스턴스 메타 데이터, VPC DNS를 통한 NTP에 대한 과도한 요청으로 인해 손실된 패킷 수. 이 제한은 모든 서비스에 대해 초당 1024패킷입니다.
    - pps_allowance_exceeded: 과도한 초당 패킷 수 비율로 인해 손실된 패킷 수. 네트워크 트래픽이 초당 수천 또는 수백만 개의 매우 작은 요청으로 구성된 경우 PPS 제한에 도달할 수 있습니다. ElastiCache 대신 여러 작업을 한 번에 수행하는 파이프라인이나 명령을 통해 네트워크 패킷을 더 잘 활용하도록 트래픽을 최적화할 수 있습니다. MGET GET

단일 요청을 완료하는 데 걸리는 시간 분석

- On the network: Tcpdump 및 Wireshark (명령줄의 tshark) 는 요청이 네트워크를 통과하여 ElastiCache 엔진에 도달하고 반환되는 데 걸린 시간을 파악할 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 TCP 3방향 핸드셰이크가 완료되는 데 222마이크로초(918091 - 917869)가 걸렸고 ping 명령이 제출되어 반환되는 데 173마이크로초(918295 - 918122)가 걸렸음을 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. strace와 함께 동일한 Redis PING 명령을 사용하여 얻은 결과:

```
$ 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 이름을 확인하는 데 약 20밀리초(697712에서 717890)의 상당한 시간이 걸렸으며, 그 후에 TCP 소켓을 생성하는 데 2밀리초(745659에서 747858), 요청을 제출하고 응답을 받는 데 0.4밀리초(747858에서 748330)가 필요했습니다.

Amazon ElastiCache의 보안

AWS에서는 클라우드 보안을 가장 중요하게 생각합니다. 여러분은 AWS 고객으로서 보안에 민감한 기관의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

보안은 AWS과 사용자의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드 내 보안 및 클라우드의 보안으로 설명합니다.

- 클라우드의 보안 - AWS는 AWS클라우드에서 AWS서비스를 실행하는 인프라를 보호합니다.AWS는 또한 안전하게 사용할 수 있는 서비스를 제공합니다. 서드 파티 감사원은 정기적으로 AWS 규제 준수 프로그램의 일환으로 보안 효과를 테스트하고 검증합니다. Amazon ElastiCache에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램별 범위 내 AWS 서비스를 참조하세요.
- 클라우드 내 보안: 귀하의 책임은 귀하가 사용하는 AWS 서비스에 의해 결정됩니다. 또한 귀하는 귀사의 데이터의 민감도, 귀사의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 Amazon ElastiCache를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 주제에서는 보안 및 규정 준수 목적에 맞게 Amazon ElastiCache를 구성하는 방법을 보여줍니다. 또한 Amazon ElastiCache 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법을 알아봅니다.

주제

- Amazon ElastiCache의 데이터 보호
- 인터네트워크 트래픽 개인 정보
- Amazon용 ID 및 액세스 관리 ElastiCache
- Amazon에 대한 규정 준수 검증 ElastiCache
- Amazon ElastiCache의 복원성
- AWS ElastiCache의 인프라 보안
- 서비스 업데이트 ElastiCache

Amazon ElastiCache의 데이터 보호

AWS 공동 책임 모델은 AWS ElastiCache(ElastiCache)의 데이터 보호에 적용됩니다. 이 모델에서 설명하는 것처럼 AWS은(는) 모든 AWS 클라우드를 실행하는 글로벌 인프라를 보호할 책임이 있습니다. 이 인프라에서 호스팅되는 콘텐츠에 대한 통제를 유지하는 것은 사용자의 책임입니다. 이 콘텐츠에는

사용하는 AWS 서비스에 대한 보안 구성 및 관리 작업이 포함됩니다. 데이터 프라이버시에 대한 자세한 설명은 데이터 프라이버시 FAQ를 참조하세요.

데이터를 보호하려면 AWS 계정 자격 증명을 보호하고 AWS Identity and Access Management(IAM)을 사용해 개별 계정을 설정하는 것이 좋습니다. 이러한 방식에서는 각 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용합니다.
- TLS를 사용하여 AWS 리소스와 통신합니다.
- AWS CloudTrail로 API 및 사용자 활동 로깅을 설정합니다.
- AWS 암호화 솔루션을 AWS 서비스 내의 모든 기본 보안 컨트롤과 함께 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 마세요. 여기에는 콘솔, API, AWS CLI 또는 AWS SDK를 사용하여 ElastiCache 또는 다른 AWS 서비스를 처리하는 경우가 포함됩니다. ElastiCache 또는 기타 서비스에 입력하는 모든 데이터를 선택하여 진단 로그에 포함시킬 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명 정보를 URL에 포함시키지 마세요.

주제

- Amazon ElastiCache의 데이터 보안

Amazon ElastiCache의 데이터 보안

데이터를 안전하게 보관하기 위해 Amazon ElastiCache 및 Amazon EC2에서는 서버의 데이터에 대한 무단 액세스를 방지하는 메커니즘을 제공합니다.

Amazon ElastiCache for Memcached는 Memcached 버전 1.6.12 이상을 실행하는 캐시에서 데이터 암호화를 제공하는 기능을 갖추고 있습니다.

- 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.

주제

- ElastiCache 전송 중 암호화 (TLS)
- ElastiCache에서 저장 시 암호화

ElastiCache 전송 중 암호화 (TLS)

데이터를 안전하게 유지하기 위해 Amazon ElastiCache 및 Amazon EC2는 서버의 데이터에 대한 무단 액세스를 방지하는 메커니즘을 제공합니다. 전송 중 암호화 기능을 제공함으로써 데이터가 한 위치에서 다른 위치로 이동할 때 데이터를 보호하는 데 사용할 수 있는 도구를 ElastiCache 제공합니다.

모든 서버리스 캐시에는 전송 중 암호화가 활성화되어 있습니다. 자체 설계된 클러스터의 경우 `CreateCacheCluster`(CLI: `create-cache-cluster`) 작업을 사용하여 캐시 클러스터를 생성할 때 `TransitEncryptionEnabled` 파라미터를 `true`(CLI: `--transit-encryption-enabled`)로 설정하여 캐시 클러스터에서 전송 중 암호화를 활성화할 수 있습니다.

주제

- 전송 중 데이터 암호화 개요
- 전송 중 데이터 암호화 조건
- 전송 중 데이터 암호화 모범 사례
- 전송 중 데이터 암호화 사용 설정
- Openssl을 사용하여 전송 중 데이터 암호화가 사용 설정된 노드에 연결
- Java를 사용하여 TLS Memcached 클라이언트 생성
- PHP를 사용하여 TLS Memcached 클라이언트 생성

전송 중 데이터 암호화 개요

Amazon ElastiCache 전송 중 암호화는 데이터가 한 위치에서 다른 위치로 전송될 때 가장 취약한 시점에서 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 연결은 TLS로 암호화됩니다.
- 암호화된 서버 연결 - 클러스터의 노드 간에 이동하는 데이터는 암호화됩니다.

- 서버 인증 - 클라이언트가 자신이 올바른 서버에 연결 중임을 인증할 수 있습니다.

전송 중 데이터 암호화 조건

자체 설계된 클러스터 구현을 계획할 때는 Amazon ElastiCache 전송 중 암호화에 대한 다음과 같은 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 데이터 암호화는 Amazon VPC에서 실행 중인 클러스터에 대해서만 지원됩니다.
- M1, M2, M3, R3, T2 노드 유형을 실행하는 복제 그룹에는 전송 중 암호화가 지원되지 않습니다.

  자세한 정보는 지원되는 노드 유형을 참조하세요.
- 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.
- 캐시 클라이언트가 TLS 연결을 지원하고 클라이언트 구성에서 TLS 연결을 활성화합니다.

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.
- 새로운 연결을 생성하면 비용이 높아질 수 있기 때문에 TLS 연결을 지속해 전송 중 데이터 암호화가 성능에 미치는 영향을 줄일 수 있습니다.

전송 중 데이터 암호화 사용 설정

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.

step-by-step 프로세스에 대한 자세한 내용은 Memcached 클러스터 생성 (콘솔) 을 참조하십시오.

Openssl을 사용하여 전송 중 데이터 암호화가 사용 설정된 노드에 연결

전송 중 암호화가 활성화된 Memcached 노드의 데이터에 액세스하려면 SSL (Secure Socket Layer)을 사용하는 클라이언트를 사용해야 합니다. ElastiCache 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 모드로 클라이언트를 생성하려면 다음을 수행하여 적절한 SSLContext를 통해 클라이언트를 초기화합니다.

```
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");
    }
```

```
}
```

PHP를 사용하여 TLS Memcached 클라이언트 생성

TLS 모드로 클라이언트를 생성하려면 다음을 수행하여 적절한 SSLContext를 통해 클라이언트를 초기화합니다.

```
<?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/mem-ug/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 사용 방법에 관한 자세한 정보는 PHP용 ElastiCache 클러스터 클라이언트 설치 섹션을 참조하세요.

ElastiCache에서 저장 시 암호화

데이터를 안전하게 보관하기 위해 Amazon ElastiCache 및 Amazon S3에서는 캐시의 데이터에 대한 액세스를 제한하는 다른 방식을 제공합니다. 자세한 정보는 Amazon VPC 및 ElastiCache 보안 및 Amazon용 ID 및 액세스 관리 ElastiCache 섹션을 참조하십시오.

- 동기화, 스왑 작업 중 디스크

ElastiCache는 기본(서비스 관리형) 저장 시 암호화와 더불어 AWS Key Management Service(KMS)에서 자체 대칭 고객 관리형 AWS KMS 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 섹션을 참조하세요.

> **Note**
>
> GovCloud(미국) 리전에서 사용할 수 있는 유일한 옵션은 기본(서비스 관리형) 암호화입니다.

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
- AWS KMS에서 고객 관리형 키 사용
- 저장 데이터 암호화 활성화
- 참고

미사용 데이터 암호화 조건

ElastiCache의 미사용 데이터 암호화를 구현하기 위해 계획하는 경우에는 ElastiCache의 미사용 데이터 암호화에 대한 다음 제약을 염두에 둬야 합니다.

- 저장 시 암호화는 서버리스 캐시에서만 지원됩니다.
- AWS GovCloud(us-gov-east-1 및 us-gov-west-1) 리전에서는 저장된 데이터 암호화를 위한 고객 관리형 키를 사용할 수 없습니다.

AWS KMS에서 고객 관리형 키 사용

ElastiCache는 저장 시 암호화에 대한 대칭 고객 관리형 AWS KMS 키(KMS 키)를 지원합니다. 고객 관리형 KMS 키는 사용자가 생성, 소유 및 관리하는 AWS 계정의 암호화 키입니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 AWS KMS 키를 참조하세요. ElastiCache와 함께 사용하기 전에 AWS KMS에서 키를 생성해야 합니다.

AWS KMS 루트 키 생성 방법을 알아보려면 AWS Key Management Service 개발자 안내서에서 키 생성을 참조하세요.

ElastiCache를 사용하면 AWS KMS와 통합할 수 있습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 권한 부여 사용을 참조하세요. Amazon ElastiCache와 AWS KMS의 통합을 활성화하기 위해 고객이 별도로 취해야 할 조치는 없습니다.

`kms:ViaService` 조건 키는 AWS KMS 키(KMS 키)의 사용을 지정된 AWS 서비스로부터의 요청으로 제한합니다. ElastiCache에서 `kms:ViaService`를 사용하려면 조건 키 `elasticache.AWS_region.amazonaws.com` 및 `dax.AWS_region.amazonaws.com` 모두에 ViaService 이름을 포함시켜야 합니다. 자세한 내용은 kms:ViaService 섹션을 참조하세요.

AWS CloudTrail을 사용하여 Amazon ElastiCache가 사용자 대신 AWS Key Management Service에 전송하는 요청을 추적할 수 있습니다. 고객 관리형 키와 관련된 AWS Key Management Service에 대한 모든 API 호출에는 해당 CloudTrail 로그가 있습니다. ListGlants KMS API 호출을 통해 ElastiCache가 생성하는 그랜트를 볼 수도 있습니다.

- 캐시를 암호화하는 데 사용한 키에 대해 키를 삭제하거나, 키를 비활성화하거나, 권한 부여를 취소하면 캐시를 복구할 수 없게 됩니다. 즉, 하드웨어 장애 후 이를 수정하거나 복구할 수 없습니다. AWS KMS에서는 최소 7일 이상의 대기 기간이 지난 후에만 루트 키를 삭제합니다. 키를 삭제한 후 다른 고객 관리형 키를 사용하여 보관용 백업을 생성할 수 있습니다.
- 자동 키 교체 기능은 AWS KMS 루트 키의 속성을 그대로 보존하기 때문에 키가 교체되더라도 ElastiCache 데이터에 대한 액세스 권한에는 아무런 영향도 끼치지 않습니다. 암호화된 Amazon ElastiCache 캐시는 새로운 루트 키 생성 및 기존 키에 대한 모든 참조를 업데이트하는 수동 키 교체를 지원하지 않습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 AWS KMS 키 교체를 참조하세요.
- KMS 키를 사용하여 ElastiCache 캐시를 암호화하려면 캐시당 1개의 권한이 필요합니다. 이 권한은 캐시의 수명 기간 동안 사용됩니다.
- AWS KMS 그랜트 및 제한에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서에서 제한을 참조하세요.

저장 데이터 암호화 활성화

모든 서버리스 캐시에는 저장 시 암호화가 활성화되어 있습니다.

ElastiCache 캐시를 생성할 때 저장 시 암호화를 활성화할 수 있습니다. AWS Management Console, AWS CLI 또는 ElastiCache API를 사용해 이 작업을 수행할 수 있습니다.

캐시를 생성할 때 다음 옵션 중 하나를 선택할 수 있습니다.

- 기본값 – 이 옵션은 저장된 서비스 관리 암호화를 사용합니다.
- 고객 관리형 키 – 이 옵션을 통해 저장된 데이터 암호화에 대한 AWS KMS의 키 ID/ARN을 제공합니다.

AWS KMS 루트 키 생성 방법을 알아보려면 AWS Key Management Service 개발자 안내서에서 키 생성을 참조하세요.

목차

- AWS Management Console를 사용하여 미사용 데이터 암호화 활성화

AWS Management Console를 사용하여 미사용 데이터 암호화 활성화

서버리스 캐시에서 저장 시 암호화 활성화(콘솔)

모든 서버리스 캐시에는 저장 시 암호화가 활성화되어 있습니다. 기본적으로 AWS 소유의 KMS 키가 데이터를 암호화하는 데 사용됩니다. 자체 AWS KMS 키를 선택하려면 다음과 같이 진행합니다.

- 기본 설정 섹션을 펼칩합니다.
- 기본 설정 섹션에서 기본 설정 사용자 지정을 선택합니다.
- 보안 섹션에서 보안 설정 사용자 지정을 선택합니다.
- 암호화 키 설정에서 고객 관리형 CMK를 선택합니다.
- AWS KMS 키 설정에서 키를 선택합니다.

참고

- Amazon VPC 및 ElastiCache 보안
- Amazon용 ID 및 액세스 관리 ElastiCache

인터네트워크 트래픽 개인 정보

Amazon ElastiCache는 다음 기술을 사용하여 캐시 데이터 보안을 유지하고 무단 액세스로부터 보호합니다.

- Amazon VPC 및 ElastiCache 보안은 설치에 필요한 보안 그룹 유형을 설명합니다.
- 사용자, 그룹 및 역할의 작업을 부여하고 제한하기 위한 Amazon용 ID 및 액세스 관리 ElastiCache

Amazon VPC 및 ElastiCache 보안

데이터 보안이 중요하기 때문에 ElastiCache는 데이터를 액세스할 수 있는 대상을 제어하는 수단을 제공합니다. 데이터에 대한 액세스를 제어하는 방법은 클러스터를 Amazon Virtual Private Cloud(Amazon VPC)에서 시작했는지 또는 Amazon EC2-Classic에서 시작했는지에 따라 다릅니다.

> **Important**
>
> ElastiCache 클러스터 시작을 위해 Amazon EC2-Classic을 사용하던 방식은 사용이 중지되었습니다. 모든 현재 세대 노드는 Amazon Virtual Private Cloud에서만 시작됩니다.

Amazon Virtual Private Cloud(Amazon VPC) 서비스는 기존 데이터 센터와 매우 유사한 가상 네트워크를 정의합니다. Amazon VPC를 구성할 때 그 IP 주소 범위를 선택하고 서브넷을 생성하고 라우팅 테이블, 네트워크 게이트웨이 및 보안 설정을 구성할 수 있습니다. 또한 Amazon VPC 보안 그룹을 사용하여 가상 네트워크에 캐시 클러스터를 추가하고 캐시 클러스터에 대한 액세스 권한을 제어할 수 있습니다.

이 섹션에서는 Amazon VPC에서 ElastiCache 클러스터를 수동으로 구성하는 방법을 설명합니다. 이 정보는 ElastiCache 및 Amazon VPC가 연동되는 방식을 더 깊이 이해하고자 하는 사용자를 대상으로 합니다.

주제

- ElastiCache 및 Amazon VPC 이해
- Amazon VPC의 ElastiCache 캐시에 액세스하기 위한 액세스 패턴
- Virtual Private Cloud(VPC) 생성
- Amazon VPC에서 실행 중인 캐시에 연결

ElastiCache 및 Amazon VPC 이해

ElastiCache는 Amazon Virtual Private Cloud(Amazon VPC)와 완벽하게 통합됩니다. ElastiCache 사용자의 경우 이는 다음을 의미합니다.

- AWS 계정에서 EC2-VPC 플랫폼만 지원하는 경우 ElastiCache는 항상 Amazon VPC에서 클러스터를 시작합니다.
- AWS를 처음 이용하는 경우 클러스터가 Amazon VPC에 배포됩니다. 기본 VPC가 자동으로 생성됩니다.
- 기본 VPC가 있고 클러스터를 시작할 때 서브넷을 지정하지 않으면 클러스터가 기본 Amazon VPC에서 시작합니다.

자세한 정보는 Detecting Your Supported Platforms and Whether You Have a Default VPC를 참조하세요.

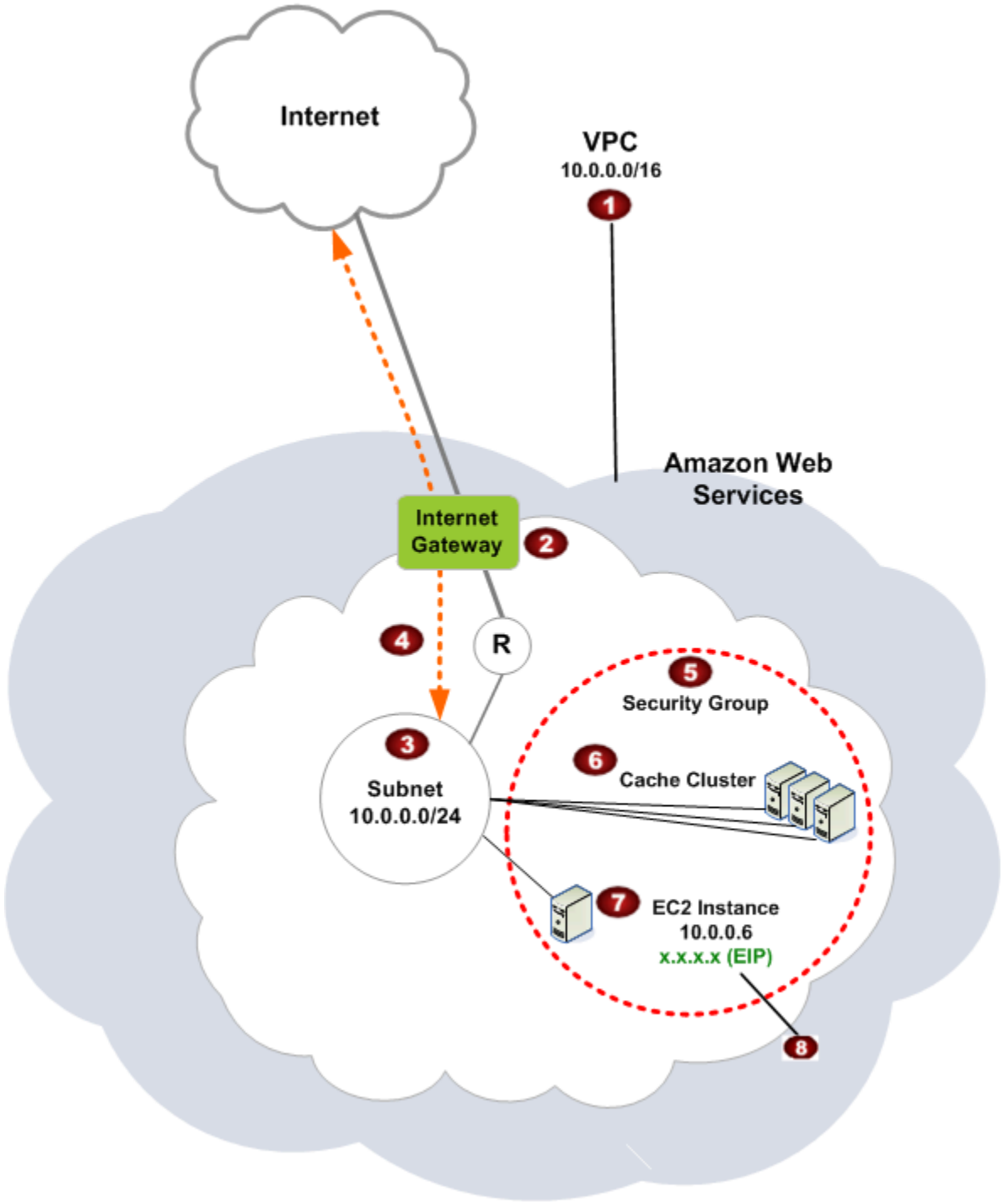
Amazon Virtual Private Cloud를 사용하면 기존의 데이터 센터와 매우 유사한 AWS 클라우드에 가상 네트워크를 생성할 수 있습니다. IP 주소 범위 선택, 서브넷 생성 및 라우팅 테이블, 네트워크 게이트웨이, 보안 설정 구성을 포함하여 Amazon VPC를 구성할 수 있습니다.

ElastiCache의 기본 기능은 가상 프라이빗 클라우드에서 동일합니다. ElastiCache는 클러스터가 Amazon VPC의 내부 또는 외부에 배포되는 여부와 관계없이 소프트웨어 업그레이드, 패치 적용, 장애 탐지 및 복구를 관리합니다.

Amazon VPC 외부에 배포된 ElastiCache 캐시 노드에는 엔드포인트/DNS 이름이 확인되는 IP 주소가 할당됩니다. 이는 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스에서 연결을 제공합니다. ElastiCache 클러스터를 Amazon VPC 프라이빗 서브넷으로 실행할 때 모든 캐시 노드는 그 서브넷 내의 프라이빗 IP 주소에 할당됩니다.

Amazon VPC의 ElastiCache 개요

다음 다이어그램과 표에는 Amazon VPC에서 시작되는 ElastiCache 클러스터 및 Amazon EC2 인스턴스와 함께 Amazon VPC 환경에 대한 설명이 나와 있습니다.



1	Amazon VPC는 자체 IP 주소 블록이 할당된 AWS 클라우드의 격리된 부분입니다.
2	

	인터넷 게이트웨이는 Amazon VPC를 인터넷에 직접 연결하고 Amazon VPC 외부에서 실행되는 Amazon Simple Storage Service(Amazon S3)와 같은 다른 AWS 리소스에 대한 액세스를 제공합니다.
3	Amazon VPC 서브넷은 보안 및 운영상의 필요에 따라 AWS 리소스를 격리할 수 있는 Amazon VPC의 IP 주소 범위 세그먼트입니다.
4	Amazon VPC의 라우팅 테이블은 서브넷과 인터넷 간 네트워크 트래픽을 지시합니다. Amazon VPC에는 이 다이어그램에서 원 안에 R로 표시된 라우터가 내재되어 있습니다.
5	Amazon VPC 보안 그룹은 ElastiCache 클러스터와 Amazon EC2 인스턴스의 인바운드 및 아웃바운드 트래픽을 제어합니다.
6	서브넷에서 ElastiCache 클러스터를 실행할 수 있습니다. 캐시 노드는 서브넷 주소 범위의 프라이빗 IP 주소를 가집니다.
7	또한 서브넷에서 Amazon EC2 인스턴스를 시작할 수 있습니다. 각각의 Amazon EC2 인스턴스는 서브넷 주소 범위의 프라이빗 IP 주소를 가집니다. Amazon EC2 인스턴스를 동일한 서브넷의 모든 캐시 노드에 연결할 수 있습니다.
8	Amazon VPC의 Amazon EC2 인스턴스를 인터넷에 연결하려면 인스턴스에 탄력적 IP 주소라는 고정 퍼블릭 주소를 할당해야 합니다.

사전 조건

Amazon VPC에 ElastiCache 클러스터를 생성하려면 Amazon VPC가 다음 요구 사항을 충족해야 합니다.

- Amazon VPC는 비전용 Amazon EC2 인스턴스를 허용해야 합니다. 전용 인스턴스 테넌시로 구성된 Amazon VPC에서는 ElastiCache를 사용할 수 없습니다.
- Amazon VPC에 대해 캐시 서브넷 그룹을 정의해야 합니다. ElastiCache는 캐시 서브넷 그룹을 사용하여 VPC 엔드포인트 또는 캐시 노드에 연결된 서브넷 내의 서브넷 및 IP 주소를 선택할 수 있습니다.

- 각 서브넷의 CIDR 블록은 유지 관리 작업에서 ElastiCache에 사용할 여분의 IP 주소를 제공할 수 있을 만큼 충분히 커야 합니다.

라우팅 및 보안

Amazon VPC에서 라우팅을 구성하여 트래픽 흐름(예: 인터넷 게이트웨이, 가상 프라이빗 게이트웨이)을 제어할 수 있습니다. 인터넷 게이트웨이를 통해 Amazon VPC를 Amazon VPC에서 실행되지 않는 다른 AWS 리소스에 직접 액세스할 수 있습니다. 조직의 로컬 네트워크에 연결된 가상 사설 게이트웨이만을 사용하도록 선택한 경우, VPN을 통해 인터넷 바운드 트래픽을 라우팅하고 출구를 제어하기 위한 로컬 보안 정책 및 방화벽을 사용할 수 있습니다. 이 경우 인터넷을 통해 AWS 리소스에 액세스할 때 대역폭 요금이 추가로 부과됩니다.

Amazon VPC 보안 그룹을 사용하여 Amazon VPC에서 ElastiCache 클러스터 및 Amazon EC2 인스턴스를 보호할 수 있습니다. 보안 그룹은 서브넷 레벨이 아닌 인스턴스 레벨에서 방화벽처럼 작동합니다.

> **Note**
>
> 기본 IP 주소가 변경될 수 있으므로 캐시 노드에 연결할 때 DNS 이름을 사용하는 것이 좋습니다.

Amazon VPC 설명서

Amazon VPC에는 Amazon VPC를 생성하고 사용하는 방법을 설명하는 자체 문서 세트가 있습니다. 다음 테이블은 Amazon VPC 지침의 링크를 제공합니다.

설명	설명서
Amazon VPC 사용을 시작하는 방법	Amazon VPC 시작하기
AWS Management Console을 통해 Amazon VPC를 사용하는 방법	Amazon VPC User Guide
모든 Amazon VPC 명령의 전체 설명	Amazon EC2 명령줄 레퍼런스(Amazon VPC 명령은 Amazon EC2 참조에서 찾을 수 있음)
Amazon VPC API 작업, 데이터 형식 및 오류의 전체 설명	Amazon EC2 API 참조(Amazon VPC API 작업은 Amazon EC2 참조에서 찾을 수 있음)

설명	설명서
선택적인 IPsec VPN 연결 사용자 측의 게이트웨이를 구성하는 데 필요한 네트워크 관리자를 위한 정보	AWS Site-to-Site VPN이란 무엇입니까?

Amazon Virtual Private Cloud에 대한 자세한 내용은 Amazon Virtual Private Cloud 섹션을 참조하세요.

Amazon VPC의 ElastiCache 캐시에 액세스하기 위한 액세스 패턴

Amazon은 Amazon VPC의 캐시에 액세스하는 다음 시나리오를 ElastiCache 지원합니다.

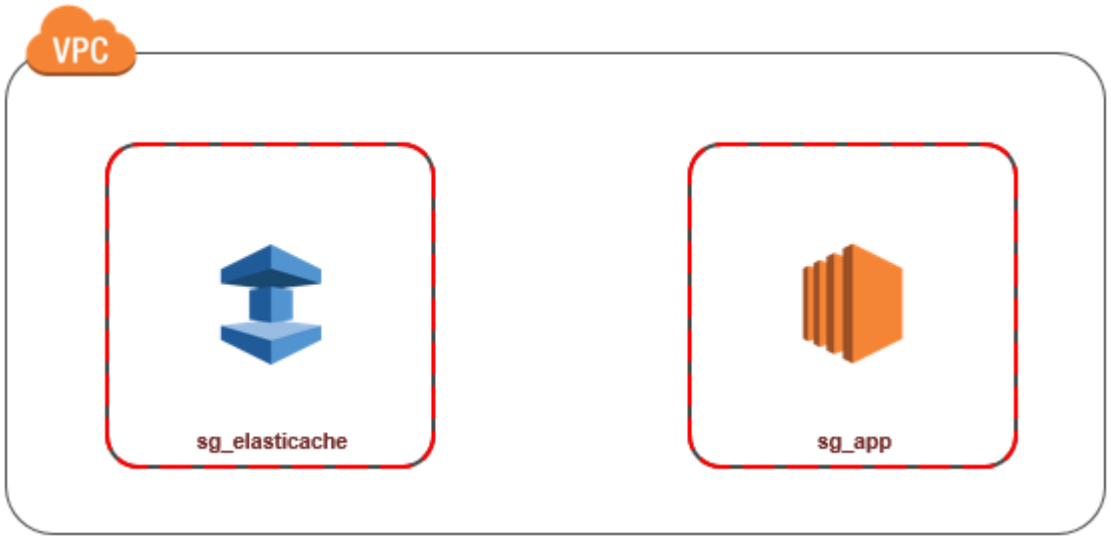
목차

- ElastiCache 캐시와 Amazon EC2 인스턴스가 동일한 Amazon VPC에 있을 때 캐시에 액세스
- ElastiCache 캐시와 Amazon EC2 인스턴스가 서로 다른 Amazon VPC에 있을 때 캐시에 액세스
  - ElastiCache 캐시와 Amazon EC2 인스턴스가 같은 지역의 서로 다른 Amazon VPC에 있을 때 캐시에 액세스
    - Transit Gateway 사용
  - ElastiCache 캐시와 Amazon EC2 인스턴스가 서로 다른 지역의 서로 다른 Amazon VPC에 있을 때 캐시에 액세스
    - 전송 VPC 사용
- 고객의 데이터 센터에서 실행되는 애플리케이션의 ElastiCache 캐시에 액세스
  - VPN 연결을 사용하여 고객의 데이터 센터에서 실행되는 애플리케이션의 ElastiCache 캐시에 액세스
  - Direct Connect를 사용하여 고객 데이터 센터에서 실행되는 애플리케이션의 ElastiCache 캐시에 액세스

ElastiCache 캐시와 Amazon EC2 인스턴스가 동일한 Amazon VPC에 있을 때 캐시에 액세스

가장 일반적인 사용 사례는 EC2 인스턴스에 배포된 애플리케이션이 동일한 VPC에 있는 캐시에 연결해야 하는 경우입니다.

다음은 이 시나리오를 설명하는 다이어그램입니다



동일한 VPC에서 EC2 인스턴스와 캐시 간 액세스를 관리하는 가장 간단한 방법은 다음과 같습니다.

1. 캐시에서 VPC 보안 그룹을 생성합니다. 이 보안 그룹을 사용해 캐시에 대한 액세스를 제한할 수 있습니다. 예를 들어 캐시를 만들 때 할당한 포트와 캐시에 액세스할 때 이용할 IP 주소를 사용해 TCP 액세스를 허용하는 이 보안 그룹에 대해 사용자 지정 규칙을 만들 수 있습니다.

   Memcached 캐시의 기본 포트는 11211입니다.
2. EC2 인스턴스(웹 및 애플리케이션 서버)의 VPC 보안 그룹을 만듭니다. 이 보안 그룹은 필요할 경우 VPC의 라우팅 테이블을 통한 EC2 인스턴스 액세스를 허용할 수 있습니다. 예를 들어, 이 보안 그룹에서 TCP가 포트 22를 통해 EC2 인스턴스에 액세스하도록 허용하는 규칙을 설정할 수 있습니다.
3. 캐시의 보안 그룹에서 EC2 인스턴스에 대해 생성한 보안 그룹으로부터의 연결을 허용하는 사용자 지정 규칙을 만듭니다. 그러면 보안 그룹의 모든 구성원이 캐시에 액세스할 수 있습니다.

> **Note**
>
> Local Zones를 사용할 계획이라면 활성화했는지 확인합니다. 해당 로컬 영역에 서브넷 그룹을 생성하면 VPC가 해당 로컬 영역으로 확장되고 VPC에서 해당 서브넷을 다른 가용 영역의 서브넷처럼 처리합니다. 모든 관련 게이트웨이 및 라우팅 테이블이 자동으로 조정됩니다.

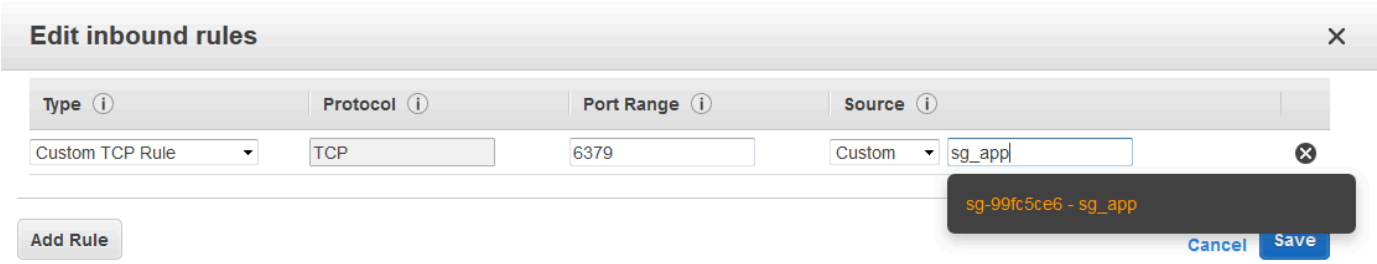
VPC 보안 그룹에서 다른 보안 그룹으로부터의 연결을 허용하는 규칙을 만들려면

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. [Type]에서 [Custom TCP Rule]을 선택합니다.

   a. 포트 범위에서 캐시를 만들 때 사용한 포트를 지정합니다.

      Memcached 캐시의 기본 포트는 11211입니다.

   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 서로 다른 Amazon VPC에 있을 때 캐시에 액세스

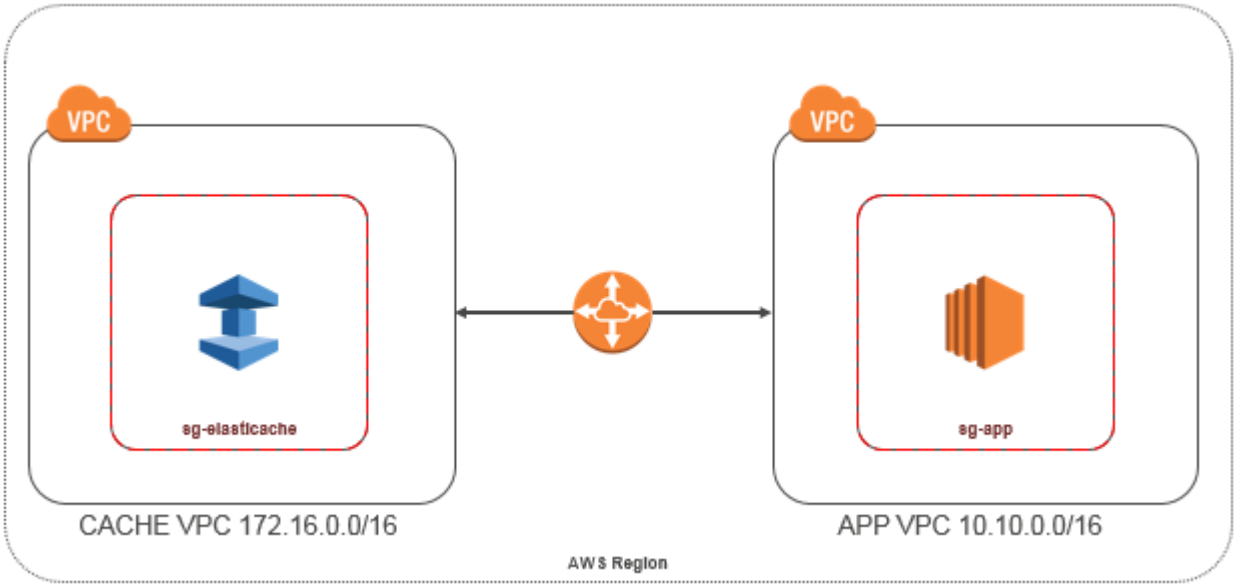
캐시가 액세스에 사용할 EC2 인스턴스와 다른 VPC에 있는 경우 여러 방법으로 캐시에 액세스할 수 있습니다. 캐시와 EC2 인스턴스가 서로 다른 VPC에 있지만 리전은 동일하면 VPC 피어링을 사용할 수 있습니다. 캐시와 EC2 인스턴스가 서로 다른 리전에 있으면 리전 간에 VPN 연결을 만들 수 있습니다.

주제

- ElastiCache 캐시와 Amazon EC2 인스턴스가 같은 지역의 서로 다른 Amazon VPC에 있을 때 캐시에 액세스
- ElastiCache 캐시와 Amazon EC2 인스턴스가 서로 다른 지역의 서로 다른 Amazon VPC에 있을 때 캐시에 액세스

ElastiCache 캐시와 Amazon EC2 인스턴스가 같은 지역의 서로 다른 Amazon VPC에 있을 때 캐시에 액세스

다음 다이어그램에서는 Amazon VPC 피어링 연결을 사용하여 동일한 리전의 서로 다른 Amazon VPC에 있는 Amazon EC2 인스턴스에서 캐시에 액세스하는 과정을 보여 줍니다.



동일한 리전 내의 서로 다른 Amazon VPC에 있는 Amazon EC2 인스턴스가 액세스하는 캐시 - VPC 피어링 연결

VPC 피어링 연결은 프라이빗 IP 주소를 사용하여 두 VPC 간에 트래픽을 라우팅할 수 있도록 하기 위한 두 VPC 사이의 네트워킹 연결입니다. 동일한 네트워크에 속하는 경우와 같이 VPC의 인스턴스가 서로 통신할 수 있습니다. 자신의 Amazon VPC 간에 또는 단일 지역 내 다른 AWS 계정의 Amazon VPC와 VPC 간 VPC 피어링 연결을 생성할 수 있습니다. Amazon VPC 피어링에 대한 자세한 내용은 VPC 설명서를 참조하세요.

> **Note**
>
> VPC에 적용된 구성에 따라 피어링된 VPC의 DNS 이름 확인이 실패할 수 있습니다. ElastiCache 이를 해결하려면 DNS 호스트 이름과 DNS 확인에 대해 두 VPC를 모두 사용 설정해야 합니다. 자세한 정보는 VPC 피어링 연결에 대해 DNS 확인 사용 설정을 참조하세요.

피어링으로 서로 다른 Amazon VPC에 있는 캐시에 액세스하려면 다음과 같이 하세요.

1. 두 VPC에 겹치는 IP 범위가 없거나 이 VPC를 피어링할 수 없어야 합니다.

2. 두 VPC를 피어링합니다. 자세한 정보는 Amazon VPC 피어링 연결 생성 및 수락을 참조하세요.
3. 라우팅 테이블을 업데이트합니다. 자세한 내용은 VPC 피어링 연결을 위한 라우팅 테이블 업데이트를 참조하세요.

   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 VPC의 애플리케이션 보안 그룹으로부터의 인바운드 연결을 허용하도록 ElastiCache 캐시의 보안 그룹을 수정합니다. 자세한 내용은 피어 VPC 보안 그룹 참조를 참조하세요.

피어링 연결을 통해 캐시에 액세스하면 데이터 전송 비용이 추가로 발생합니다.

Transit Gateway 사용

트랜짓 게이트웨이를 사용하면 동일한 AWS 지역의 VPC와 VPN 연결을 연결하고 이들 간에 트래픽을 라우팅할 수 있습니다. 트랜짓 게이트웨이는 여러 AWS 계정에서 작동하며, AWS Resource Access Manager를 사용하여 트랜짓 게이트웨이를 다른 계정과 공유할 수 있습니다. 트랜짓 게이트웨이를 다른 AWS 계정과 공유한 후 계정 소유자는 자신의 VPC를 트랜짓 게이트웨이에 연결할 수 있습니다. 두 계정의 사용자는 언제든지 연결을 삭제할 수 있습니다.

Transit Gateway에서 멀티캐스트를 활성화한 다음 도메인과 연결된 VPC 연결을 통해 멀티캐스트 소스에서 멀티캐스트 그룹 멤버로 멀티캐스트 트래픽을 보낼 수 있는 Transit Gateway 멀티캐스트 도메인을 생성할 수 있습니다.

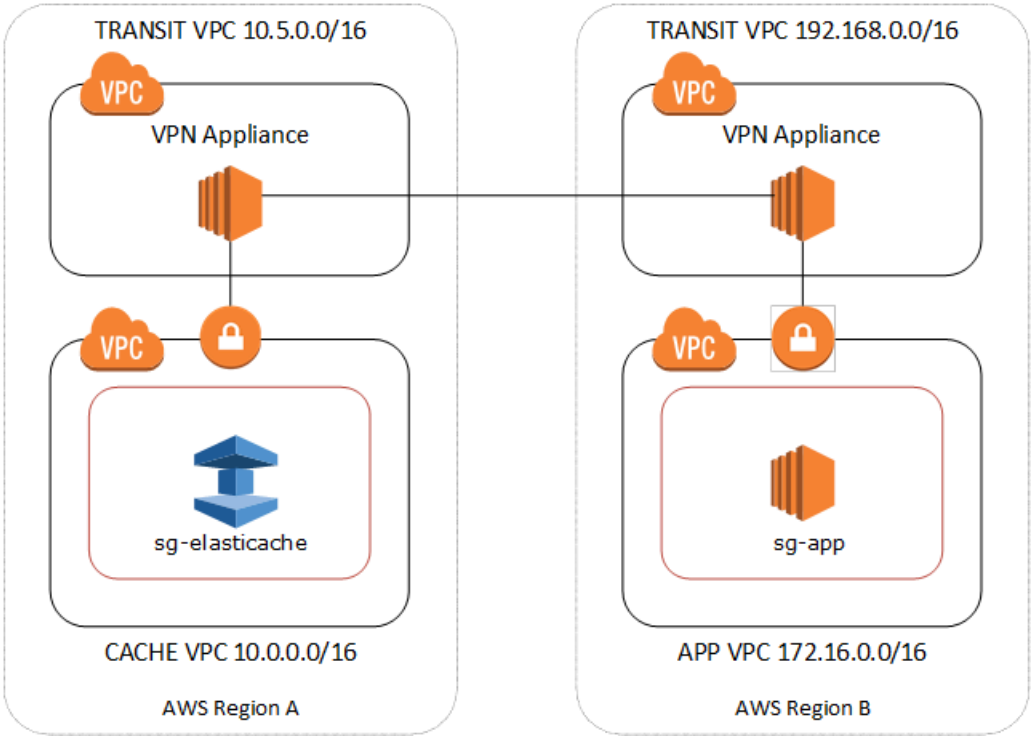
또한 여러 지역의 트랜짓 게이트웨이 간에 피어링 연결 연결을 생성할 수 있습니다. AWS 이렇게 하면 여러 리전의 Transit Gateway 연결 간에 트래픽을 라우팅할 수 있습니다.

자세한 내용은 전송 게이트웨이를 참조하십시오.

ElastiCache 캐시와 Amazon EC2 인스턴스가 서로 다른 지역의 서로 다른 Amazon VPC에 있을 때 캐시에 액세스

전송 VPC 사용

VPC 피어링을 사용하는 대신, 지리적으로 떨어져 있는 여러 VPC와 원격 네트워크를 연결하는 또 다른 일반적인 전략은 글로벌 네트워크 전송 센터 역할을 하는 전송 VPC를 만드는 것입니다. 전송 VPC는 네트워크 관리를 간소화하고 여러 VPC와 원격 네트워크를 연결하는 데 필요한 연결 수를 최소화합니다. 이 디자인은 기존의 방식대로 공동 배치 전송 허브에 실제 존재를 만들거나 물리적 네트워크 장비를 배포하지 않고 가상으로 구현되므로 시간, 노력, 비용을 아낄 수 있습니다.



다른 리전의 다른 VPC를 지나는 연결

Transit Amazon VPC가 설정되면 한 지역의 “스포크” VPC에 배포된 애플리케이션을 다른 지역의 “스포크” VPC에 있는 ElastiCache 캐시에 연결할 수 있습니다.

다른 지역 내 다른 VPC의 캐시에 액세스하는 방법 AWS

1. 전송 VPC 솔루션을 배포합니다. 자세한 내용은 AWS Transit Gateway를 참조하세요.
2. 앱 및 캐시 VPC에서 VPC 라우팅 테이블을 업데이트하여 VGW(가상 프라이빗 게이트웨이) 및 VPN 어플라이언스를 통해 트래픽을 라우팅합니다. BGP(Border Gateway Protocol)를 사용하는 동적 라우팅의 경우 라우팅이 자동으로 전파됩니다.

3. 애플리케이션 인스턴스 IP 범위에서의 인바운드 연결을 허용하도록 ElastiCache 캐시의 보안 그룹을 수정하십시오. 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 애플리케이션의 ElastiCache 캐시에 액세스

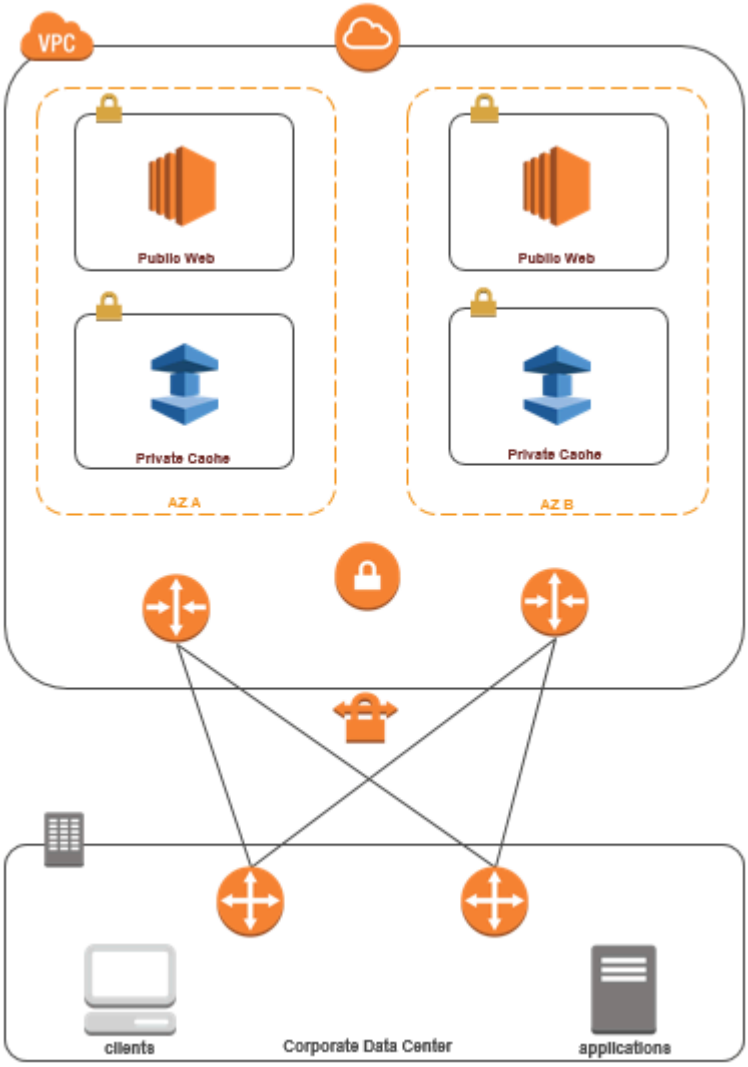
또 다른 가능한 시나리오는 고객 데이터 센터의 클라이언트 또는 애플리케이션이 VPC의 ElastiCache 캐시에 액세스해야 하는 하이브리드 아키텍처입니다. VPN이나 Direct Connect를 통해 고객의 VPC와 데이터 센터 간에 연결을 제공하여 이 시나리오도 지원됩니다.

주제

- VPN 연결을 사용하여 고객의 데이터 센터에서 실행되는 애플리케이션의 ElastiCache 캐시에 액세스
- Direct Connect를 사용하여 고객 데이터 센터에서 실행되는 애플리케이션의 ElastiCache 캐시에 액세스

VPN 연결을 사용하여 고객의 데이터 센터에서 실행되는 애플리케이션의 ElastiCache 캐시에 액세스

다음 다이어그램은 VPN 연결을 사용하여 회사 네트워크에서 실행되는 애플리케이션에서 ElastiCache 캐시에 액세스하는 방법을 보여줍니다.



VPN을 ElastiCache 통해 데이터 센터에서 연결

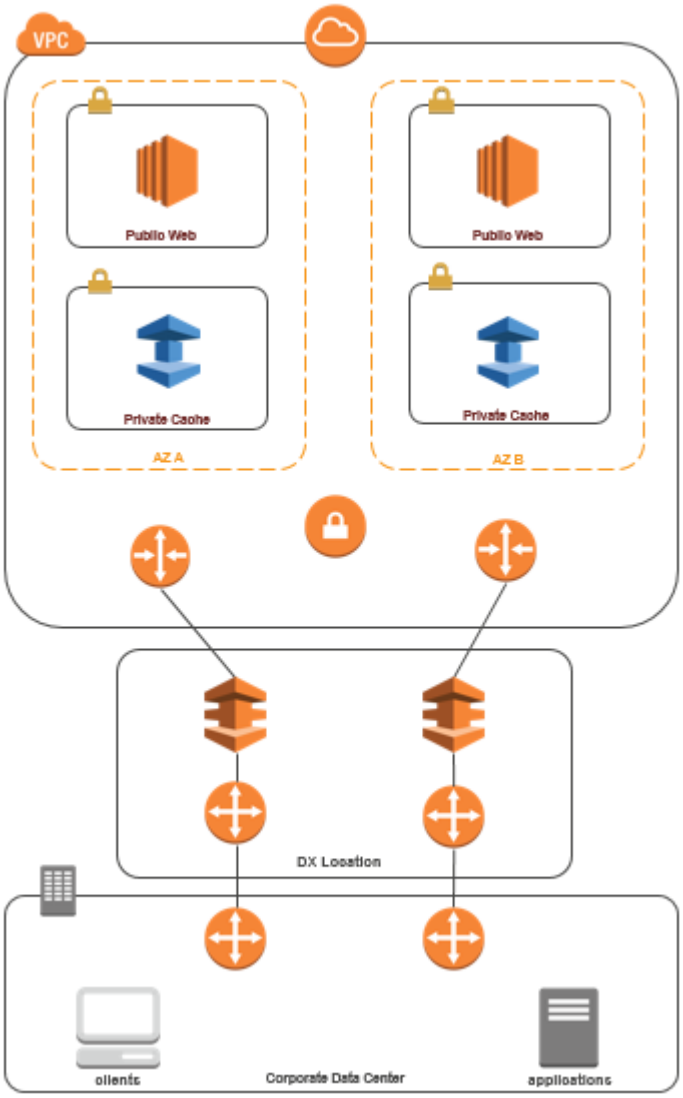
VPN 연결을 통해 온프레미스 애플리케이션에서 VPC의 캐시에 액세스하려면 다음과 같이 하세요.

1. 하드웨어 가상 프라이빗 게이트웨이를 VPC에 추가하여 VPN 연결을 설정합니다. 자세한 내용은 VPC에 하드웨어 가상 프라이빗 게이트웨이 추가를 참조하세요.
2. ElastiCache 캐시가 배포된 서브넷의 VPC 라우팅 테이블을 업데이트하여 온프레미스 애플리케이션 서버의 트래픽을 허용하십시오. BGP를 사용하는 동적 라우팅의 경우 라우팅이 자동으로 전파될 수 있습니다.
3. 온프레미스 애플리케이션 서버로부터의 인바운드 연결을 허용하도록 ElastiCache 캐시의 보안 그룹을 수정하십시오.

VPN 연결을 통해 캐시에 액세스하면 네트워크 지연 시간이 생기고 데이터 전송 비용이 추가로 발생합니다.

Direct Connect를 사용하여 고객 데이터 센터에서 실행되는 애플리케이션의 ElastiCache 캐시에 액세스

다음 다이어그램은 Direct Connect를 사용하여 회사 네트워크에서 실행되는 응용 프로그램에서 ElastiCache 캐시에 액세스하는 방법을 보여 줍니다.



Direct Connect를 통해 데이터 센터에서 연결 ElastiCache

Direct Connect를 사용하여 네트워크에서 실행 중인 응용 프로그램의 ElastiCache 캐시에 액세스하려면

1. Direct Connect 연결을 설정합니다. 자세한 내용은 AWS Direct Connect로 시작하기를 참조하십시오.
2. 온프레미스 애플리케이션 서버로부터의 인바운드 연결을 허용하도록 ElastiCache 캐시의 보안 그룹을 수정하십시오.

DX 연결을 통해 캐시에 액세스하면 네트워크 지연 시간이 생기고 데이터 전송 요금이 추가로 발생할 수 있습니다.

Virtual Private Cloud(VPC) 생성

이 예제에서는 각 가용 영역에 대해 프라이빗 서브넷이 있는 Amazon VPC를 생성합니다.

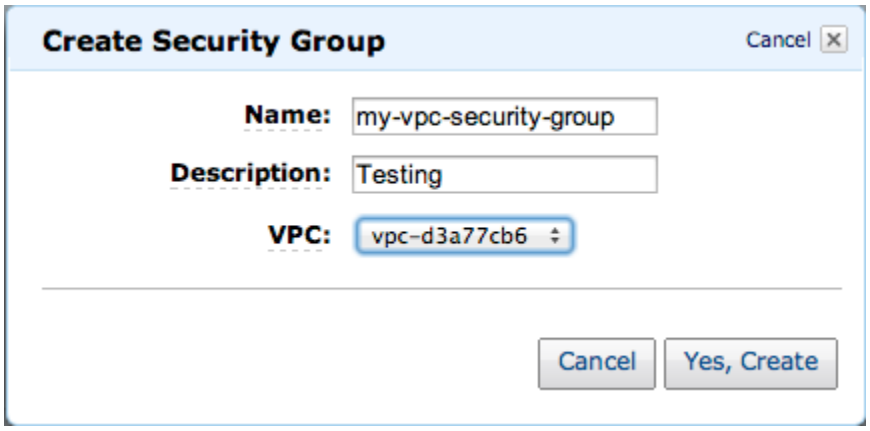
Amazon VPC 생성(콘솔)

1. AWS 관리 콘솔에 로그인한 다음 https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
2. VPC 대시보드에서 VPC 생성(Create VPC)을 선택합니다.
3. 생성할 리소스(Resources to create)에서 VPC 등(VPC and more)을 선택합니다.
4. 가용 영역(AZ) 수(Number of Availability Zones(AZs))에서 서브넷을 시작할 가용 영역 수를 선택합니다.
5. 퍼블릭 서브넷 수(Number of public subnets)에서 VPC 추가할 퍼블릭 서브넷 수를 선택합니다.
6. 프라이빗 서브넷 수(Number of private subnets)에서 VPC 추가할 프라이빗 서브넷 수를 선택합니다.

> **Tip**
>
> 서브넷 식별자(퍼블릭 서브넷, 프라이빗 서브넷)를 기록해 둡니다. 추후 클러스터를 시작하고 Amazon VPC에 Amazon EC2 인스턴스를 추가하는 경우 이 정보가 필요합니다.

7. Amazon VPC 보안 그룹을 생성합니다. 캐시 클러스터 및 Amazon EC2 인스턴스에 이 그룹을 사용합니다.

   a. Amazon VPC 관리 콘솔의 탐색 창에서 보안 그룹을 선택합니다.
   b. 보안 그룹 생성을 선택합니다.
   c. 보안 그룹의 이름과 설명을 해당 상자에 입력합니다. VPC 상자에서 Amazon VPC의 식별자를 선택합니다.



d. 원하는 대로 설정되었으면 [Yes, Create]를 선택합니다.

8. 보안 그룹의 네트워크 수신 규칙을 정의합니다. 이 규칙을 사용하면 SSH(Secure Shell)를 사용하여 Amazon EC2 인스턴스에 연결할 수 있습니다.

   a. 탐색 목록에서 [Security Groups]를 선택합니다.

   b. 목록에서 보안 그룹을 찾아 선택합니다.

   c. [Security Group] 아래에서 [Inbound] 탭을 선택합니다. [Create a new rule] 상자에서 [SSH]를 선택한 다음 [Add Rule]을 선택합니다.

   d. 새 인바운드 규칙에 다음 값을 설정하여 HTTP 액세스를 허용합니다.

      - 유형: HTTP
      - 소스: 0.0.0.0/0

      [Apply Rule Changes]를 선택합니다.

이제 캐시 서브넷 그룹을 생성하고 Amazon VPC에서 캐시 클러스터를 시작할 수 있습니다.

- 서브넷 그룹 생성
- Memcached 클러스터 생성(콘솔).

Amazon VPC에서 실행 중인 캐시에 연결

이 예에서는 Amazon VPC에서 Amazon EC2 인스턴스를 시작하는 방법을 보여 줍니다. 이 인스턴스에 로그인하여 Amazon VPC에서 실행 중인 ElastiCache 캐시에 액세스할 수 있습니다.

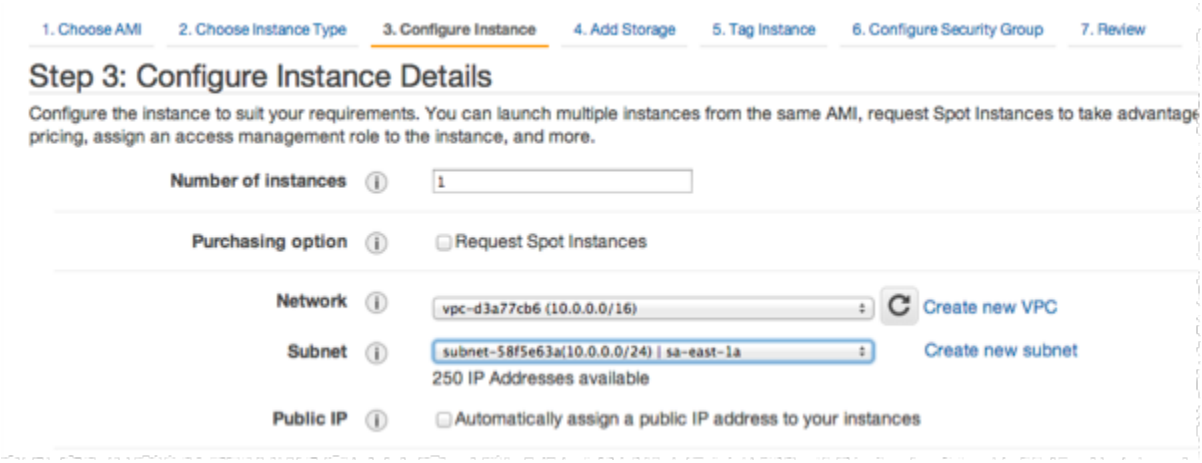
Amazon VPC에서 실행 중인 캐시에 연결(콘솔)

이 예에서는 Amazon VPC에서 Amazon EC2 인스턴스를 생성합니다. 이 Amazon EC2 인스턴스를 사용하여 Amazon VPC에서 실행 중인 캐시 노드에 연결할 수 있습니다.

> **Note**
>
> Amazon EC2 사용에 대한 자세한 내용은 Amazon EC2 설명서에서 Amazon EC2 시작 안내서를 참조하세요.

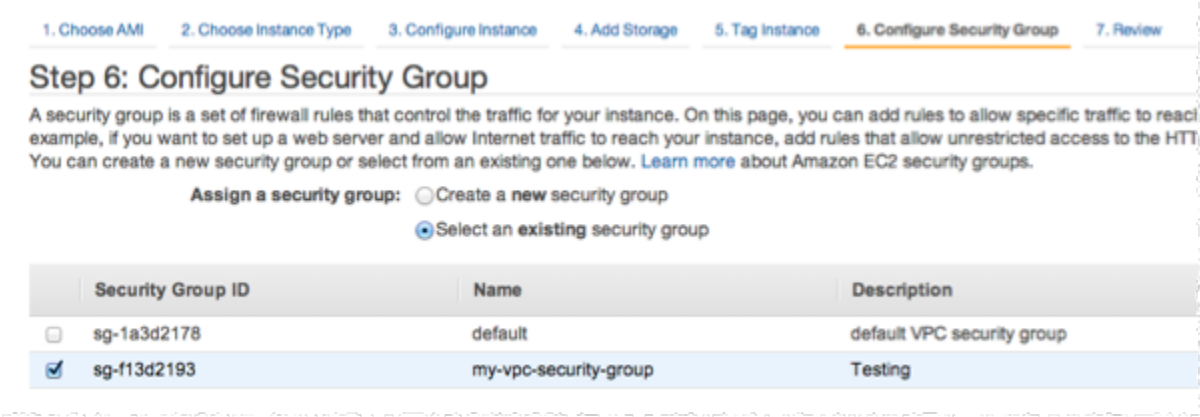
Amazon EC2 콘솔을 사용하여 Amazon VPC에 Amazon EC2 인스턴스를 생성하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 콘솔에서 [Launch Instance]를 선택하고 다음 단계를 따릅니다.
3. [Choose an Amazon Machine Image (AMI)] 페이지에서 64비트 Amazon Linux AMI를 선택한 다음 [Select]를 선택합니다.
4. 인스턴스 유형 선택 페이지에서 3. 인스턴스 구성을 선택합니다.
5. [Configure Instance Details] 페이지에서 다음과 같이 선택합니다.
   a. 네트워크 목록에서 Amazon VPC를 선택합니다.
   b. [Subnet] 목록에서 퍼블릭 서브넷을 선택합니다.



원하는 대로 설정되었으면 4. 스토리지 추가를 선택합니다.

6. 스토리지 추가 페이지에서 5. 인스턴스 태그 지정을 선택합니다.
7. 인스턴스 태그 지정 페이지에서 Amazon EC2 인스턴스 이름을 입력한 다음 6. 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 그룹을 참조하세요.



Amazon VPC 보안 그룹 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 페어에 대한 자세한 내용은 Amazon EC2 시작 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. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
2. 탐색 목록에서 [Elastic IPs]를 선택합니다.
3. 탄력적 IP 주소 할당을 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 방금 생성한 탄력적 IP 주소로 Amazon EC2 인스턴스에 연결할 수 있습니다.

Amazon EC2 인스턴스에 연결

- 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 마십시오.

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

1. 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!
```

2. https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔로 이동하고 캐시 클러스터의 노드 중 하나에 대한 엔드포인트를 가져옵니다. 자세한 내용은 Memcached에 대한 연결 엔드포인트 찾기를 참조하세요.
3. 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
```

> **Important**
>
> AWS 계정에 추가 요금이 발생하는 것을 방지하기 위해 이 예제를 실행한 후에 더 이상 필요하지 않은 AWS 리소스를 삭제하세요.

Amazon ElastiCache API 및 인터페이스 VPC 엔드포인트(AWS PrivateLink)

인터페이스VPC 엔드포인트를 생성하여 VPC와 Amazon ElastiCache API 엔드포인트 간에 프라이빗 연결을 설정할 수 있습니다. 인터페이스 엔드포인트는 AWS PrivateLink에 의해 구동됩니다. AWS PrivateLink를 사용하면 인터넷 게이트웨이, NAT 디바이스, VPN 연결 또는 AWS Direct Connect 연결 없이도 Amazon ElastiCache API 작업에 프라이빗 액세스할 수 있습니다.

VPC에 있는 인스턴스는 퍼블릭 IP 주소가 없어도 Amazon ElastiCache API 엔드포인트와 통신할 수 있습니다. 또한 인스턴스는 퍼블릭 IP 주소가 없어도 사용 가능한 ElastiCache API 작업을 모두 사용할 수 있습니다. VPC와 Amazon ElastiCache 간의 트래픽은 Amazon 네트워크를 벗어나지 않습니다. 각 인터페이스 엔드포인트는 서브넷에서 하나 이상의 탄력적 네트워크 인터페이스로 표현됩니다. 탄력적 네트워크 인터페이스에 대한 자세한 내용은 Amazon EC2 사용 설명서의 탄력적 네트워크 인터페이스를 참조하십시오.

- VPC 엔드포인트에 대한 자세한 정보는 Amazon VPC 사용 설명서의 인터페이스 VPC 엔드포인트(AWS PrivateLink)를 참조하세요.
- ElastiCache API 작업에 관한 자세한 정보는 ElastiCache API 작업을 참조하세요.

인터페이스 VPC 엔드포인트를 생성한 후 엔드포인트에 대해 프라이빗 DNS 호스트 이름을 사용 설정하는 경우 기본 ElastiCache 엔드포인트(https://athena.*Region*.amazonaws.com)는 VPC 엔드포인트로 확인됩니다. 프라이빗 DNS 호스트 이름을 활성화하지 않은 경우, Amazon VPC는 다음 형식으로 사용할 수 있는 DNS 엔드포인트를 제공합니다.

```
VPC_Endpoint_ID.elasticache.Region.vpce.amazonaws.com
```

자세한 정보는 Amazon VPC 사용 설명서에서 인터페이스 VPC 엔드포인트(AWS PrivateLink)를 참조하세요. ElastiCache는 VPC 내부에 있는 모든 API 작업에 대한 호출 수행을 지원합니다.

> **Note**
>
> 프라이빗 DNS 호스트 이름은 VPC에 있는 하나의 VPC 엔드포인트에 대해서만 사용 설정할 수 있습니다. 추가 VPC 엔드포인트를 생성하려는 경우 프라이빗 DNS 호스트 이름을 사용 중지해야 합니다.

VPC 엔드포인트 고려 사항

Amazon ElastiCache API 엔드포인트에 대한 인터페이스 VPC 엔드포인트를 설정하기 전에 Amazon VPC 사용 설명서에서 인터페이스 엔드포인트 속성 및 제한 사항을 검토해야 합니다. Amazon ElastiCache 리소스 관리와 관련된 모든 ElastiCache API 작업은 AWS PrivateLink를 통해 VPC에서 사용할 수 있습니다.

VPC 엔드포인트 정책은 ElastiCache API 엔드포인트에 대해 지원됩니다. 기본적으로, 엔드포인트를 통해 ElastiCache API 작업에 대한 전체 액세스가 허용됩니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC 엔드포인트를 통해 서비스에 대한 액세스 제어를 참조하세요.

ElastiCache API에 대한 인터페이스 VPC 엔드포인트 생성

Amazon VPC 콘솔 또는 AWS CLI를 사용하여 Amazon ElastiCache API에 대한 VPC 엔드포인트를 생성할 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트 생성을 참조하세요.

인터페이스 VPC 엔드포인트를 생성한 후 엔드포인트에 대한 프라이빗 DNS 호스트 이름을 사용할 수 있습니다. 이렇게 하면 기본 Amazon ElastiCache 엔드포인트(https://elasticache.*Region*.amazonaws.com)는 VPC 엔드포인트로 확인됩니다. 중국(베이징) 및 중국(닝샤) AWS 리전의 경우 베이징에 대해 `elasticache.cn-north-1.amazonaws.com.cn` 및 닝샤에 대해 `elasticache.cn-northwest-1.amazonaws.com.cn`을 사용하여 VPC 엔드포인트를 통해 API 요청을 수행할 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트를 통해 서비스 액세스를 참조하세요.

Amazon ElastiCache API에 대한 VPC 엔드포인트 정책 생성

ElastiCache API에 대한 액세스를 제어하는 VPC 엔드포인트에 엔드포인트 정책을 연결할 수 있습니다. 이 정책은 다음을 지정합니다.

- 태스크를 수행할 수 있는 보안 주체.
- 수행할 수 있는 작업입니다.
- 태스크를 수행할 있는 리소스.

자세한 정보는 Amazon VPC 사용 설명서의 VPC 엔드포인트를 통해 서비스에 대한 액세스 제어를 참조하세요.

Example ElastiCache API 작업에 대한 VPC 엔드포인트 정책

다음은 ElastiCache API에 대한 엔드포인트 정책의 예입니다. 이 정책은 엔드포인트에 연결될 때 모든 리소스의 모든 보안 주체에 대한 액세스 권한을 나열된 ElastiCache API 작업에 부여합니다.

```
{
 "Statement": [{
  "Principal": "*",
  "Effect": "Allow",
  "Action": [
   "elasticache:CreateCacheCluster",
   "elasticache:ModifyCacheCluster"
  ],
  "Resource": "*"
 }]
}
```

Example 지정된 AWS 계정의 모든 액세스를 거부하는 VPC 엔드포인트 정책

다음 VPC 엔드포인트 정책은 AWS 계정 *123456789012*가 엔드포인트를 사용하는 리소스에 대한 모든 액세스를 거부합니다. 이 정책은 다른 계정의 모든 작업을 허용합니다.

```
{
 "Statement": [{
   "Action": "*",
   "Effect": "Allow",
   "Resource": "*",
   "Principal": "*"
  },
  {
   "Action": "*",
   "Effect": "Deny",
   "Resource": "*",
   "Principal": {
```

```
    "AWS": [
     "123456789012"
    ]
   }
  }
 ]
}
```

서브넷 및 서브넷 그룹

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

Amazon VPC에서 자체 설계된 클러스터를 생성하는 경우 서브넷 그룹을 사용해야 합니다. ElastiCache는 해당 서브넷 그룹을 사용하여 노드에 연결된 서브넷 내의 서브넷 및 IP 주소를 선택합니다.

ElastiCache는 기본 IPv4 서브넷 그룹을 제공하거나 선택하여 새 서브넷 그룹을 생성할 수 있습니다. IPv6의 경우 IPv6 CIDR 블록이 있는 서브넷 그룹을 생성해야 합니다. 듀얼 스택을 선택한 경우 검색 IP 유형으로 IPv6 또는 IPv4 중에 선택해야 합니다.

ElastiCache 서버리스는 서브넷 그룹 리소스를 사용하지 않고 생성 과정에서 서브넷 목록을 직접 가져옵니다.

이 섹션에서는 서브넷과 서브넷 그룹을 생성 및 활용하여 ElastiCache 리소스에 대한 액세스를 관리하는 방법을 알아봅니다.

Amazon VPC 환경에서 서브넷 그룹 사용에 대한 자세한 내용은 클러스터에 액세스 섹션을 참조하세요.

주제

- 서브넷 그룹 생성
- 캐시에 서브넷 그룹 할당
- 서브넷 그룹 수정
- 서브넷 그룹 삭제

서브넷 그룹 생성

캐시 서브넷 그룹은 VPC에서 캐시에 대해 지정할 수 있는 서브넷 모음입니다. VPC에서 캐시를 시작할 때 캐시 서브넷 그룹을 선택해야 합니다. 그러면 ElastiCache가 해당 캐시 서브넷 그룹을 사용하여 해당 서브넷의 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 유형으로 IPv4를 선택하면 기본 서브넷 그룹을 선택하거나 새 서브넷 그룹을 생성할 수 있습니다. ElastiCache는 해당 서브넷 그룹을 사용하여 노드에 연결된 서브넷 내의 서브넷 및 IP 주소를 선택합니다. 듀얼 스택 또는 IPv6을 선택하면 듀얼 스택 또는 IPv6 서브넷을 생성하라는 메시지가 표시됩니다. 네트워크 유형에 대한 자세한 내용은 네트워크 유형을 참조하세요. 자세한 정보는 VPC에서 서브넷 생성을 참조하세요.

다음 절차는 `mysubnetgroup`이라는 서브넷 그룹을 콘솔, AWS CLI 및 ElastiCache API를 통해 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 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
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 VPC를 선택합니다.

    d. 기본적으로 모든 서브넷이 선택됩니다. 선택한 서브넷 패널에서 관리를 클릭하고 프라이빗 서브넷의 가용 영역 또는 로컬 영역 및 ID를 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 항목 create-cache-subnet-group를 참조하세요.

캐시에 서브넷 그룹 할당

서브넷 그룹을 생성한 후 Amazon VPC에서 캐시를 시작할 수 있습니다. 추가 정보는 다음을 참조하세요.

- Memcached 클러스터 - Memcached 클러스터를 시작하려면 Memcached 클러스터 생성(콘솔) 섹션을 참조하세요. 7.a 단계(고급 Memcached 설정)에서 VPC 서브넷 그룹을 선택합니다.

서브넷 그룹 수정

서브넷 그룹의 설명을 수정하거나 서브넷 그룹과 연관된 서브넷 ID의 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. AWS Management Console에 로그인하고 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 항목 modify-cache-subnet-group를 참조하세요.

서브넷 그룹 삭제

서브넷 그룹이 더 이상 필요하지 않다고 판단되면 삭제할 수 있습니다. 현재 캐시에서 사용 중인 경우 서브넷 그룹을 삭제할 수 없습니다.

다음 절차는 서브넷 그룹을 삭제하는 방법을 보여줍니다.

서브넷 그룹 삭제(콘솔)

서브넷 그룹을 삭제하는 방법

1. AWS Management Console에 로그인하고 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)

AWS CLI를 사용하여 다음 파라미터로 delete-cache-subnet-group 명령을 호출할 수 있습니다.

- `--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 항목 delete-cache-subnet-group를 참조하세요.

Amazon용 ID 및 액세스 관리 ElastiCache

AWS Identity and Access Management (IAM) 은 관리자가 리소스에 대한 액세스를 안전하게 제어할 수 AWS 서비스 있도록 AWS 도와줍니다. IAM 관리자는 리소스를 사용할 수 있는 인증 (로그인) 및 권한 부여 (권한 보유) 를 받을 수 있는 사용자를 제어합니다. ElastiCache IAM은 추가 AWS 서비스 비용 없이 사용할 수 있습니다.

주제

- 고객
- ID를 통한 인증
- 정책을 사용한 액세스 관리
- 아마존이 IAM과 협력하는 ElastiCache 방식
- Amazon ElastiCache의 자격 증명 기반 정책 예제
- Amazon ElastiCache ID 및 액세스 문제 해결
- 액세스 제어
- ElastiCache 리소스에 대한 액세스 권한 관리 개요

고객

사용하는 방식 AWS Identity and Access Management (IAM) 은 수행하는 작업에 따라 다릅니다. ElastiCache

서비스 사용자 - ElastiCache 서비스를 사용하여 작업을 수행하는 경우 관리자가 필요한 자격 증명과 권한을 제공합니다. 더 많은 ElastiCache 기능을 사용하여 작업을 수행함에 따라 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방식을 이해하면 적절한 권한을 관리자에게 요청할 수 있습니다. 에서 ElastiCache 기능에 액세스할 수 없는 경우 을 참조하십시오Amazon ElastiCache ID 및 액세스 문제 해결.

서비스 관리자 — 회사에서 ElastiCache 리소스를 담당하고 있다면 전체 액세스 권한이 있을 것입니다 ElastiCache. 서비스 사용자가 액세스해야 하는 ElastiCache 기능과 리소스를 결정하는 것은 여러분의 몫입니다. 그런 다음, IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해하십시오. 회사에서 IAM을 어떻게 사용할 수 있는지 자세히 ElastiCache 알아보려면 을 참조하십시오아마존이 IAM과 협력하는 ElastiCache 방식.

IAM 관리자 — IAM 관리자라면 액세스 관리를 위한 정책을 작성하는 방법에 대해 자세히 알고 싶을 것입니다. ElastiCache IAM에서 사용할 수 있는 ElastiCache ID 기반 정책의 예를 보려면 을 참조하십시오. Amazon ElastiCache의 자격 증명 기반 정책 예제

ID를 통한 인증

인증은 ID 자격 증명을 AWS 사용하여 로그인하는 방법입니다. IAM 사용자로 인증 (로그인 AWS) 하거나 IAM 역할을 맡아 인증 (로그인) 해야 합니다. AWS 계정 루트 사용자

ID 소스를 통해 제공된 자격 증명을 사용하여 페더레이션 ID로 로그인할 수 있습니다. AWS AWS IAM Identity Center (IAM ID 센터) 사용자, 회사의 싱글 사인온 인증, Google 또는 Facebook 자격 증명이 페더레이션 ID의 예입니다. 페더레이션 ID로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 페더레이션을 사용하여 액세스하는 경우 AWS 간접적으로 역할을 맡게 됩니다.

사용자 유형에 따라 AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. 로그인에 대한 자세한 내용은 AWS 로그인 사용 설명서의 내 로그인 방법을 참조하십시오. AWS AWS 계정

AWS 프로그래밍 방식으로 액세스하는 경우 자격 증명을 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트 (SDK) 와 명령줄 인터페이스 (CLI) 를 AWS 제공합니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. 권장 방법을 사용하여 직접 요청에 서명하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 AWS API 요청 서명을 참조하십시오.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어, AWS 계정의 보안을 강화하기 위해 다단계 인증 (MFA) 을 사용할 것을 권장합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서의 다중 인증 및 IAM 사용 설명서의 AWS에서 다중 인증(MFA) 사용을 참조하십시오.

AWS 계정 루트 사용자

계정을 AWS 계정만들 때는 먼저 계정의 모든 AWS 서비스 리소스에 대한 완전한 액세스 권한을 가진 하나의 로그인 ID로 시작합니다. 이 ID를 AWS 계정 루트 사용자라고 하며, 계정을 만들 때 사용한 이메일 주소와 비밀번호로 로그인하여 액세스할 수 있습니다. 일상적인 태스크에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 보안 인증 정보를 보호하고 루트 사용자만 수행할 수 있는 태스크를 수행하는 데 사용하세요. 루트 사용자로 로그인해야 하는 전체 작업 목록은 IAM 사용 설명서의 루트 사용자 보안 인증이 필요한 작업을 참조하십시오.

페더레이션 자격 증명

가장 좋은 방법은 관리자 액세스가 필요한 사용자를 비롯한 수동 AWS 서비스 사용자가 ID 공급자와의 페더레이션을 사용하여 임시 자격 증명을 사용하여 액세스하도록 하는 것입니다.

페더레이션 ID는 기업 사용자 디렉토리, 웹 ID 공급자, Identity Center 디렉터리의 사용자 또는 ID 소스를 통해 제공된 자격 증명을 사용하여 액세스하는 AWS 서비스 모든 사용자를 말합니다. AWS Directory Service페더레이션 ID에 AWS 계정액세스하면 이들이 역할을 맡고 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 자체 ID 소스의 사용자 및 그룹 집합에 연결하고 동기화하여 모든 사용자 및 애플리케이션에서 사용할 수 있습니다. AWS 계정 IAM Identity Center에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 IAM Identity Center란 무엇입니까?를 참조하십시오.

IAM 사용자 및 그룹

IAM 사용자는 단일 사용자 또는 애플리케이션에 대한 특정 권한을 AWS 계정 가진 사용자 내 자격 증명입니다. 가능하면 암호 및 액세스 키와 같은 장기 보안 인증이 있는 IAM 사용자를 생성하는 대신 임시 보안 인증을 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 보안 인증이 필요한 특정 사용 사례가 있는 경우, 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 장기 보안 인증이 필요한 사용 사례의 경우 정기적으로 액세스 키 교체를 참조하십시오.

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, IAMAdmins라는 그룹이 있고 이 그룹에 IAM 리소스를 관리할 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 IAM 사용자를 만들어야 하는 경우(역할이 아님)를 참조하십시오.

IAM 역할

IAM 역할은 특정 권한을 가진 사용자 AWS 계정 내의 자격 증명입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 역할을 AWS Management Console 전환하여 에서 일시적으로 IAM 역할을 맡을 수 있습니다. AWS CLI 또는 AWS API 작업을 호출하거나 사용자 지정 URL을 사용하여 역

할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 내용은 IAM 사용 설명서의 IAM 역할 사용을 참조하십시오.

임시 보안 인증이 있는 IAM 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 서드 파티 ID 공급자의 역할 생성 단원을 참조하십시오. IAM Identity Center를 사용하는 경우, 권한 집합을 구성합니다. 인증 후 ID가 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 세트를 IAM의 역할과 연관짓습니다. 권한 세트에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 세트를 참조하십시오.
- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 태스크에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 크로스 계정 액세스 - IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 AWS 서비스경우에는 역할을 프록시로 사용하는 대신 정책을 리소스에 직접 연결할 수 있습니다. 계정 간 액세스에 대한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 교차 계정 리소스 액세스를 참조하십시오.
- 서비스 간 액세스 — 일부는 다른 기능을 사용합니다. AWS 서비스 AWS 서비스예를 들어 서비스에서 직접적 호출을 수행하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 직접적으로 호출하는 보안 주체의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 태스크를 수행할 수 있습니다.
  - 순방향 액세스 세션 (FAS) — IAM 사용자 또는 역할을 사용하여 작업을 수행하는 경우 보안 AWS 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 전화를 거는 주체의 권한을 다운스트림 AWS 서비스서비스에 AWS 서비스 요청하기 위한 요청과 결합하여 사용합니다. FAS 요청은 다른 서비스 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 서비스가 수신한 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.
  - 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 태스크를 수행하기 위해 맡는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하십시오.
  - 서비스 연결 역할 — 서비스 연결 역할은 에 연결된 서비스 역할의 한 유형입니다. AWS 서비스서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은 사용자에게 AWS 계정 표시되며 해당 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

- Amazon EC2에서 실행되는 애플리케이션 — IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 API 요청을 AWS CLI 하는 애플리케이션의 임시 자격 증명을 관리할 수 있습니다. AWS 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS 역할을 할당하고 모든 애플리케이션에서 사용할 수 있게 하려면 인스턴스에 연결된 인스턴스 프로필을 생성합니다. 인스턴스 프로파일에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증을 얻을 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하십시오.

IAM 역할을 사용할지 또는 IAM 사용자를 사용할지를 알아보려면 IAM 사용 설명서의 IAM 역할(사용자 대신)을 생성하는 경우를 참조하십시오.

정책을 사용한 액세스 관리

정책을 생성하고 이를 AWS ID 또는 리소스에 AWS 연결하여 액세스를 제어할 수 있습니다. 정책은 ID 또는 리소스와 연결될 때 AWS 해당 권한을 정의하는 객체입니다. AWS 주도자 (사용자, 루트 사용자 또는 역할 세션) 가 요청할 때 이러한 정책을 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는 지를 결정합니다. 대부분의 정책은 JSON 문서로 AWS 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 내용은 IAM 사용 설명서의 JSON 정책 개요를 참조하십시오.

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

기본적으로, 사용자와 역할에는 어떠한 권한도 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 수임할 수 있습니다.

IAM 정책은 작업을 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, `iam:GetRole` 작업을 허용하는 정책이 있다고 가정합니다. 해당 정책을 사용하는 사용자는 AWS Management Console, AWS CLI, 또는 AWS API에서 역할 정보를 가져올 수 있습니다.

보안 인증 기반 정책

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 태스크를 수행할 수 있는지를 제어합니다. ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하십시오.

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 내 여러 사용자, 그룹 및 역할에 연결할

수 있는 독립형 정책입니다. AWS 계정관리형 정책에는 AWS 관리형 정책과 고객 관리형 정책이 포함됩니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책과 인라인 정책의 선택을 참조하십시오.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우, 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 태스크를 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 연동 사용자 등이 포함될 수 있습니다. AWS 서비스

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. IAM의 AWS 관리형 정책은 리소스 기반 정책에 사용할 수 없습니다.

액세스 제어 목록(ACL)

액세스 제어 목록(ACL)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACLs는 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

ACL을 지원하는 서비스의 예로는 아마존 S3와 아마존 VPC가 있습니다. AWS WAF ACL에 대해 자세히 알아보려면 Amazon Simple Storage Service 개발자 가이드의 ACL(액세스 제어 목록) 개요를 참조하십시오.

기타 정책 타입

AWS 일반적이지 않은 추가 정책 유형을 지원합니다. 이러한 정책 타입은 더 일반적인 정책 타입에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 자격 증명 기반 정책에 따라 IAM 엔터티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 개체의 보안 인증 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 IAM 엔터티에 대한 권한 경계를 참조하십시오.
- 서비스 제어 정책 (SCP) - SCP는 조직 또는 조직 단위 (OU) 에 대한 최대 권한을 지정하는 JSON 정책입니다. AWS Organizations AWS Organizations 사업체가 소유한 여러 AWS 계정 개를 그룹화하고 중앙에서 관리하는 서비스입니다. 조직에서 모든 기능을 활성화할 경우, 서비스 제어 정책

(SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 각 항목을 포함하여 구성원 계정의 엔티티에 대한 권한을 제한합니다. AWS 계정 루트 사용자조직 및 SCP에 대한 자세한 내용은 AWS Organizations 사용 설명서의 SCP 작동 방식을 참조하십시오.

- 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 참조하십시오.

여러 정책 타입

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 여러 정책 유형이 관련된 경우 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하십시오.

아마존이 IAM과 협력하는 ElastiCache 방식

IAM을 사용하여 액세스를 ElastiCache 관리하기 전에 어떤 IAM 기능을 사용할 수 있는지 알아보십시오. ElastiCache

Amazon에서 사용할 수 있는 IAM 기능 ElastiCache

IAM 특성	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 특성	ElastiCache 지원
보안 주체 권한	예
서비스 역할	예
서비스 연결 역할	예

ElastiCache 및 기타 AWS 서비스가 대부분의 IAM 기능과 어떻게 작동하는지 자세히 알아보려면 IAM 사용 설명서의 IAM과 함께 작동하는AWS 서비스를 참조하십시오.

ID 기반 정책은 다음과 같습니다. ElastiCache

보안 인증 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 태스크를 수행할 수 있는지를 제어합니다. ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하십시오.

IAM ID 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스뿐 아니라 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 사용하는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하십시오.

ElastiCache 자격 증명 기반 정책 예시

ElastiCache ID 기반 정책의 예를 보려면 을 참조하십시오. Amazon ElastiCache의 자격 증명 기반 정책 예제

ElastiCache 내 리소스 기반 정책

리소스 기반 정책 지원	아니요

리소스 기반 정책은 리소스에 연결하는 JSON 정책 문서입니다. 리소스 기반 정책의 예는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이

러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우, 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 태스크를 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 연동 사용자 등이 포함될 수 있습니다. AWS 서비스

교차 계정 액세스를 활성화하려는 경우, 전체 계정이나 다른 계정의 IAM 개체를 리소스 기반 정책의 보안 주체로 지정할 수 있습니다. 리소스 기반 정책에 크로스 계정 보안 주체를 추가하는 것은 트러스트 관계 설정의 절반밖에 되지 않는다는 것을 유념하십시오. 보안 주체와 리소스가 다른 AWS 계정경우 신뢰할 수 있는 계정의 IAM 관리자는 보안 주체 개체 (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 IAM의 교차 계정 리소스 액세스를 참조하십시오.

에 대한 정책 조치 ElastiCache

정책 작업 지원	예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 `Action`요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 태스크를 설명합니다. 정책 작업은 일반적으로 관련 AWS API 작업과 이름이 같습니다. 일치하는 API 작업이 없는 권한 전용 작업 같은 몇 가지 예외도 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 조치 목록을 보려면 서비스 승인 ElastiCache 참조의 Amazon이 정의한 작업을 참조하십시오.

정책 조치 중 조치 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 ID 기반 정책의 예를 보려면 을 참조하십시오. Amazon ElastiCache의 자격 증명 기반 정책 예제

에 대한 정책 리소스 ElastiCache

정책 리소스 지원	예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

Resource JSON 정책 요소는 작업이 적용되는 하나 이상의 개체를 지정합니다. 문장에는 Resource또는 NotResource요소가 반드시 추가되어야 합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 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 ARN 목록을 보려면 서비스 인증 참조의 ElastiCache Amazon이 정의한 리소스를 참조하십시오. 각 리소스의 ARN을 지정할 수 있는 작업에 대해 알아보려면 Amazon에서 정의한 작업을 참조하십시오. ElastiCache

ElastiCache ID 기반 정책의 예를 보려면 을 참조하십시오. Amazon ElastiCache의 자격 증명 기반 정책 예제

ElastiCache 정책 조건 키

서비스별 정책 조건 키 지원	예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Condition` 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` 연산을 사용하여 조건을 AWS 평가합니다. 명문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예컨대, IAM 사용자에게 IAM 사용자 이름으로 태그가 지정된 경우에만 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM 정책 요소: 변수 및 태그를 참조하십시오.

AWS 글로벌 조건 키 및 서비스별 조건 키를 지원합니다. 모든 AWS 글로벌 조건 키를 보려면 IAM 사용 AWS 설명서의 글로벌 조건 컨텍스트 키를 참조하십시오.

ElastiCache 조건 키 목록을 보려면 서비스 인증 ElastiCache 참조의 Amazon용 조건 키를 참조하십시오. 조건 키를 사용할 수 있는 작업 및 리소스를 알아보려면 Amazon에서 정의한 작업을 참조하십시오 ElastiCache.

ElastiCache ID 기반 정책의 예를 보려면 을 참조하십시오. Amazon ElastiCache의 자격 증명 기반 정책 예제

의 액세스 제어 목록 (ACL) ElastiCache

ACL 지원	예

ACL(액세스 통제 목록)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACLs는 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

다음을 통한 속성 기반 액세스 제어 (ABAC) ElastiCache

ABAC 지원(정책의 태그)	예

ABAC(속성 기반 액세스 제어)는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. AWS에서는 이러한 속성을 태그라고 합니다. IAM 개체 (사용자 또는 역할) 및 여러 AWS 리소스에 태그를 첨부할 수 있습니다. ABAC의 첫 번째 단계로 개체 및 리소스에 태그를 지정합니다. 그런 다음 보안 주체의 태그가 액세스하려는 리소스의 태그와 일치할 때 작업을 허용하도록 ABAC 정책을 설계합니다.

ABAC는 빠르게 성장하는 환경에서 유용하며 정책 관리가 번거로운 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.

ABAC에 대한 자세한 정보는 IAM 사용 설명서의 ABAC란 무엇입니까?를 참조하십시오. ABAC 설정 단계가 포함된 자습서를 보려면 IAM 사용 설명서의 속성 기반 액세스 제어(ABAC) 사용을 참조하십시오.

임시 자격 증명 사용: ElastiCache

임시 보안 인증 지원	예

임시 자격 증명을 사용하여 로그인하면 작동하지 AWS 서비스 않는 것도 있습니다. 임시 자격 증명을 사용하는 방법을 AWS 서비스 비롯한 추가 정보는 IAM 사용 설명서의 IAM과AWS 서비스 연동되는 내용을 참조하십시오.

사용자 이름과 암호를 제외한 다른 방법을 AWS Management Console 사용하여 로그인하면 임시 자격 증명을 사용하는 것입니다. 예를 들어 회사의 SSO (Single Sign-On) 링크를 AWS 사용하여 액세스하는 경우 이 프로세스에서 자동으로 임시 자격 증명을 생성합니다. 또한 콘솔에 사용자로 로그인한 다음 역할을 전환할 때 임시 보안 인증을 자동으로 생성합니다. 역할 전환에 대한 자세한 내용은 IAM 사용 설명서의 역할로 전환(콘솔)을 참조하십시오.

또는 API를 사용하여 임시 자격 증명을 수동으로 생성할 수 있습니다 AWS CLI . AWS 그런 다음 해당 임시 자격 증명을 사용하여 액세스할 수 AWS있습니다. AWS 장기 액세스 키를 사용하는 대신 임시 자

격 증명을 동적으로 생성할 것을 권장합니다. 자세한 정보는 IAM의 임시 보안 자격 증명 섹션을 참조하십시오.

ElastiCache의 서비스 간 보안 주체 권한

전달 액세스 세션(FAS) 지원	예

IAM 사용자 또는 역할을 사용하여 작업을 수행하는 AWS경우 보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 전화를 거는 주체의 권한을 다운스트림 서비스에 AWS 서비스 요청하라는 요청과 결합하여 사용합니다. AWS 서비스 FAS 요청은 다른 서비스 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 서비스가 수신한 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

서비스 역할: ElastiCache

서비스 역할 지원	예

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하십시오.

> **Warning**
>
> 서비스 역할의 권한을 변경하면 ElastiCache 기능이 중단될 수 있습니다. 서비스 역할을 편집하기 위한 지침이 ElastiCache 제공되는 경우에만 서비스 역할을 편집하십시오.

서비스 연결 역할은 다음과 같습니다. ElastiCache

서비스 링크 역할 지원	예

서비스 연결 역할은 에 연결된 서비스 역할 유형입니다. AWS 서비스서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은 사용자에게 AWS 계정 표시되며 해

당 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

서비스 연결 역할 생성 또는 관리에 대한 자세한 내용은 IAM으로 작업하는AWS 서비스를 참조하십시오. 서비스 연결 역할 열에서 Yes(이)가 포함된 서비스를 테이블에서 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 Yes(네) 링크를 선택합니다.

Amazon ElastiCache의 자격 증명 기반 정책 예제

기본적으로 사용자 및 역할은 ElastiCache 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 또한 AWS Management Console, AWS Command Line Interface(AWS CLI) 또는 AWS API를 사용하여 작업을 수행할 수 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 수임할 수 있습니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하십시오.

각 리소스 유형에 대한 ARN 형식을 포함하여 ElastiCache에서 정의한 작업 및 리소스 유형에 대한 자세한 내용은 서비스 승인 참조의 Amazon ElastiCache의 작업, 리소스 및 조건 키를 참조하세요.

주제

- 정책 모범 사례
- ElastiCache 콘솔 사용
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용

정책 모범 사례

ID 기반 정책에 따라 계정에서 사용자가 ElastiCache 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따릅니다.

- AWS 관리형 정책으로 시작하고 최소 권한을 향해 나아가기: 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. 관리형 정책은 AWS 계정에서 사용할 수 있습니다. 사용 사례에 고유한 AWS고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다. 자세한 정보는 IAM 사용 설명서의 AWS 관리형 정책 또는 직무에 대한 AWS 관리형 정책 섹션을 참조하십시오.

- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서의 IAM의 정책 및 권한을 참조하십시오.
- IAM 정책의 조건을 사용하여 액세스 추가 제한: 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어 SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. 특정 AWS 서비스(예: AWS CloudFormation)을(를) 통해 사용되는 경우에만 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하십시오.
- IAM Access Analyzer를 통해 IAM 정책을 검증하여 안전하고 기능적인 권한 보장 – IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 IAM 모범 사례가 정책에서 준수되도록 신규 및 기존 정책을 검증합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 권장 사항을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 정보는 IAM 사용 설명서의 IAM Access Analyzer 정책 검증을 참조하십시오.
- 다중 인증(MFA) 필요: AWS 계정 계정에 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 설정합니다. API 작업을 직접적으로 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 정보는 IAM 사용 설명서의 MFA 보호 API 액세스 구성 섹션을 참조하십시오.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례 섹션을 참조하십시오.

ElastiCache 콘솔 사용

Amazon ElastiCache 콘솔에 액세스하려면 최소한의 권한 집합이 있어야 합니다. 이러한 권한은 AWS 계정에서 ElastiCache 리소스에 대한 세부 정보를 나열하고 볼 수 있도록 허용해야 합니다. 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 AWS API만 호출하는 사용자에게 최소 콘솔 권한을 허용할 필요가 없습니다. 그 대신, 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

사용자와 역할이 ElastiCache 콘솔을 여전히 사용할 수 있도록 하려면 ElastiCache `ConsoleAccess` 또는 `ReadOnly` AWS 관리형 정책을 엔터티에 추가합니다. 자세한 내용은 IAM 사용 설명서의 사용자에게 권한 추가를 참조하세요.

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예시는 IAM 사용자가 자신의 사용자 자격 증명에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 AWS CLI나 AWS API를 사용하여 프로그래밍 방식으로 이 작업을 완료할 수 있는 권한이 포함됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon ElastiCache ID 및 액세스 문제 해결

다음 정보를 사용하면 및 IAM을 사용할 때 발생할 수 있는 일반적인 문제를 ElastiCache 진단하고 해결하는 데 도움이 됩니다.

주제

- 저는 다음과 같은 작업을 수행할 권한이 없습니다. ElastiCache
- 저는 IAM을 수행할 권한이 없습니다. PassRole
- 내 AWS 계정 외부의 사용자가 내 ElastiCache 리소스에 액세스할 수 있도록 허용하고 싶습니다.

저는 다음과 같은 작업을 수행할 권한이 없습니다. ElastiCache

작업을 수행할 권한이 없다는 AWS Management Console 메시지가 표시되면 관리자에게 도움을 요청해야 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.

저는 IAM을 수행할 권한이 없습니다. PassRole

작업을 수행할 권한이 없다는 오류가 발생하는 경우 역할을 넘길 수 있도록 정책을 업데이트해야 합니다. `iam:PassRole` ElastiCache

일부 AWS 서비스 서비스에서는 새 서비스 역할 또는 서비스 연결 역할을 만드는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예제 오류는 이라는 IAM 사용자가 콘솔을 사용하여 작업을 `marymajor` 수행하려고 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 필요하면 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

내 AWS 계정 외부의 사용자가 내 ElastiCache 리소스에 액세스할 수 있도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하십시오.

- 이러한 기능의 ElastiCache 지원 여부를 알아보려면 을 참조하십시오아마존이 IAM과 협력하는 ElastiCache 방식.
- 소유한 리소스에 대한 액세스 권한을 AWS 계정 부여하는 방법을 알아보려면 IAM 사용 설명서에서 자신이 소유한 다른 AWS 계정 IAM 사용자에게 액세스 권한 제공을 참조하십시오.
- 제3자에게 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사 AWS 계정AWS 계정 소유에 대한 액세스 제공을 참조하십시오.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(자격 증명 페더레이션)을 참조하십시오.
- 교차 계정 액세스에 대한 역할 사용과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 교차 계정 리소스 액세스를 참조하십시오.

액세스 제어

요청을 인증하는 데 필요한 유효한 자격 증명을 보유할 수 있지만 권한이 없으면 리소스를 생성하거나 액세스할 ElastiCache 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
- Amazon ElastiCache에 대한 자격 증명 기반 정책(IAM 정책) 사용

ElastiCache 리소스에 대한 액세스 권한 관리 개요

모든 AWS 리소스는 AWS 계정의 소유이고, 리소스 생성 또는 리소스 액세스 권한은 권한 정책에 따라 결정됩니다. 계정 관리자는 IAM 자격 증명(사용자, 그룹 및 역할)에 권한 정책을 연결할 수 있습니다. 또한 Amazon ElastiCache에서는 리소스에 권한 정책 연결을 지원합니다.

> **Note**
>
> 계정 관리자 또는 관리자 사용자는 관리자 권한이 있는 사용자입니다. 자세한 설명은 IAM 사용자 가이드의 IAM 모범 사례 섹션을 참조하세요.

액세스 권한을 제공하려면 사용자, 그룹 또는 역할에 권한을 추가하십시오:

- AWS IAM Identity Center의 사용자 및 그룹:

  권한 세트를 생성합니다. AWS IAM Identity Center 사용 설명서의 권한 세트 생성의 지침을 따르세요.
- ID 공급자를 통해 IAM에서 관리되는 사용자:

  ID 페더레이션을 위한 역할을 생성합니다. IAM 사용 설명서의 서드 파티 자격 증명 공급자의 역할 만들기(연합)의 지침을 따르세요.
- IAM 사용자:
  - 사용자가 맡을 수 있는 역할을 생성합니다. IAM 사용 설명서에서 IAM 사용자의 역할 생성의 지침을 따르세요.
  - (권장되지 않음)정책을 사용자에게 직접 연결하거나 사용자를 사용자 그룹에 추가합니다. IAM 사용 설명서에서 사용자(콘솔)에 권한 추가의 지침을 따르세요.

주제

- Amazon ElastiCache 리소스 및 작업
- 리소스 소유권 이해
- 리소스 액세스 관리
- AWS 관리형 정책(Amazon ElastiCache용)
- Amazon ElastiCache에 대한 자격 증명 기반 정책(IAM 정책) 사용
- 리소스 수준 권한

- 조건 키 사용
- Amazon ElastiCache에 대해 서비스 연결 역할 사용
- ElastiCache API 권한: 작업, 리소스, 조건 참조

Amazon ElastiCache 리소스 및 작업

ElastiCache 리소스 유형 및 해당 ARN 목록을 보려면 서비스 인증 참조에 나와 있는 Amazon ElastiCache에서 정의한 리소스를 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 Amazon ElastiCache에서 정의한 작업을 참조하세요.

리소스 소유권 이해

리소스 소유자는 리소스를 생성한 AWS 계정입니다. 즉, 리소스 소유자는 리소스를 생성하는 요청을 인증하는 보안 주체 엔터티의 AWS 계정입니다. 보안주체 엔터티는 루트 계정, IAM 사용자 또는 IAM 역할일 수 있습니다. 다음 예에서는 이러한 작동 방식을 설명합니다.

- AWS 계정의 루트 계정 자격 증명을 사용하여 캐시 클러스터를 생성한다고 가정합니다. 이 경우, AWS 계정이 리소스의 소유자입니다. ElastiCache에서 리소스는 캐시 클러스터입니다.
- AWS 계정에서 IAM 사용자를 생성하고 해당 사용자에게 캐시 클러스터 생성 권한을 부여한다고 가정합니다. 이러한 경우, 이 사용자가 캐시 클러스터를 생성할 수 있습니다. 하지만 캐시 클러스터 리소스는 사용자가 속한 AWS 계정의 소유입니다.
- AWS 계정에서 IAM 역할을 생성하고 해당 역할에게 캐시 클러스터 생성 권한을 부여한다고 가정합니다. 이러한 경우, 이 역할을 수임할 사람은 캐시 클러스터를 생성할 수 있습니다. 캐시 클러스터 리소스는 역할이 속한 AWS 계정의 소유입니다.

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 ElastiCache의 맥락에서 IAM을 사용하는 방법에 대해 설명하며, IAM 서비스에 대한 자세한 정보는 다루지 않습니다. IAM 설명서 전체 내용은 IAM 사용 설명서의 IAM이란 무엇입니까? 섹션을 참조하세요. IAM 정책 구문과 설명에 대한 자세한 내용은 IAM 사용 설명서의 AWS IAM 정책 참조 섹션을 참조하세요.

IAM 보안 인증에 연결된 정책을 보안 인증 기반 정책(IAM 정책)이라고 합니다. 리소스에 연결된 정책을 리소스 기반 정책이라고 합니다.

주제

- 자격 증명 기반 정책(IAM 정책)
- 정책 요소 지정: 작업, 효과, 리소스, 보안 주체
- 정책에서 조건 지정

자격 증명 기반 정책(IAM 정책)

정책을 IAM ID에 연계할 수 있습니다. 예를 들면, 다음을 수행할 수 있습니다:

- 계정 내 사용자 또는 그룹에 권한 정책 연결 - 계정 관리자는 권한을 부여하기 위해 특정 사용자에 연결된 권한 정책을 사용할 수 있습니다. 이 경우 해당 사용자는 캐시 클러스터, 파라미터 그룹 또는 보안 그룹과 같은 ElastiCache 리소스를 생성할 수 있습니다.
- 역할에 권한 정책 연결(교차 계정 권한 부여) - 자격 증명 기반 권한 정책을 IAM 역할에 연결하여 교차 계정 권한을 부여할 수 있습니다. 예를 들어 계정 A의 관리자는 다음과 같이 다른 AWS 계정(예: 계정 B) 또는 AWS 서비스에 교차 계정 권한을 부여할 역할을 생성할 수 있습니다.
  1. 계정 A 관리자는 IAM 역할을 생성하고 계정 A의 리소스에 대한 권한을 부여하는 역할에 권한 정책을 연결합니다.
  2. 계정 A 관리자는 계정 B를 역할을 수임할 보안 주체로 식별하는 역할에 신뢰 정책을 연결합니다.
  3. 계정 B 관리자는 계정 B의 사용자에게 역할을 수임할 권한을 위임할 수 있습니다. 그러면 계정 B의 사용자가 계정 A에서 리소스를 생성하거나 액세스할 수 있습니다. 일부 경우에는, 역할에 할당할 AWS 서비스 권한을 부여하려고 할 수 있습니다. 이러한 접근 방식을 지원하려면, 신뢰 정책의 보안 주체는 AWS 서비스 보안 주체일 수도 있습니다.

  IAM을 사용하여 권한을 위임하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 액세스 관리 섹션을 참조하세요.

다음은 사용자가 AWS 계정에 대해 `DescribeCacheClusters` 작업을 수행할 수 있도록 허용하는 정책 예제입니다. 또한 ElastiCache는 API 작업에 대한 리소스 ARN을 사용하여 특정 리소스를 식별할 수 있도록 지원합니다. (이러한 접근 방식을 리소스 수준 권한이라고도 합니다.)

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Sid": "DescribeCacheClusters",
```

```
      "Effect": "Allow",
      "Action": [
         "elasticache:DescribeCacheClusters"],
      "Resource": resource-arn
      }
   ]
}
```

ElastiCache에서 자격 증명 기반 정책을 사용하는 방법에 대한 자세한 내용은 Amazon ElastiCache에 대한 자격 증명 기반 정책(IAM 정책) 사용 섹션을 참조하세요. 사용자, 그룹, 역할 및 권한에 대한 자세한 내용은 IAM 사용 설명서의 자격 증명(사용자, 그룹 및 역할)을 참조하세요.

정책 요소 지정: 작업, 효과, 리소스, 보안 주체

각 Amazon ElastiCache 리소스(Amazon ElastiCache 리소스 및 작업 참조)에 대해 서비스는 일련의 API 작업을 정의합니다(작업 참조). 이러한 API 작업에 대한 권한을 부여하기 위해 ElastiCache에서는 정책에서 지정할 수 있는 일련의 작업을 정의합니다. 예를 들어 ElastiCache 스냅샷 리소스에 대해 `CreateCacheCluster`, `DeleteCacheCluster` 및 `DescribeCacheCluster` 작업을 정의합니다. API 작업을 실시하려면 둘 이상의 작업에 대한 권한이 필요할 수 있습니다.

다음은 가장 기본적인 정책 요소입니다.

- 리소스 – 정책에서 Amazon 리소스 이름(ARN)을 사용하여 정책을 적용할 리소스를 식별합니다. 자세한 내용은 Amazon ElastiCache 리소스 및 작업 섹션을 참조하세요.
- 조치 – 조치 키워드를 사용하여 허용 또는 거부할 리소스 작업을 식별합니다. 예를 들면, 지정된 `Effect`에 따라 `elasticache:CreateCacheCluster` 권한은 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 연결되는 사용자는 암시적인 보안 주체입니다. 리소스 기반 정책의 경우, 사용자, 계정, 서비스 또는 권한의 수신자인 기타 개체를 지정합니다(리소스 기반 정책에만 해당).

IAM 정책 구문과 설명에 대한 자세한 내용은 IAM 사용 설명서의 AWS IAM 정책 참조 섹션을 참조하세요.

모든 Amazon ElastiCache API 작업을 보여 주는 표는 ElastiCache API 권한: 작업, 리소스, 조건 참조 섹션을 참조하세요.

정책에서 조건 지정

권한을 부여할 때 IAM 정책 언어를 사용하여 정책이 적용되는 조건을 지정할 수 있습니다. 예를 들어, 특정 날짜 이후에만 정책을 적용할 수 있습니다. 정책 언어에서의 조건 지정에 관한 자세한 내용은 IAM 사용 설명서의 조건을 참조하세요.

조건을 표시하려면 미리 정의된 조건 키를 사용합니다. ElastiCache별 조건 키를 사용하려면 조건 키 사용 섹션을 참조하세요. 필요에 따라 사용할 수 있는 AWS 차원의 조건 키가 있습니다. AWS 전체 키의 전체 목록은 IAM 사용 설명서의 사용 가능한 조건 키를 참조하세요.

AWS 관리형 정책(Amazon ElastiCache용)

AWS 관리형 정책은 AWS에서 생성되고 관리되는 독립형 정책입니다. AWS 관리형 정책은 사용자, 그룹 및 역할에 권한 할당을 시작할 수 있도록 많은 일반 사용 사례에 대한 권한을 제공하도록 설계되었습니다.

AWS 관리형 정책은 모든 AWS 고객이 사용할 수 있기 때문에 특정 사용 사례에 대해 최소 권한을 부여하지 않을 수 있습니다. 사용 사례에 고유한 고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다.

AWS 관리형 정책에서 정의한 권한은 변경할 수 없습니다. AWS에서 AWS 관리형 정책에 정의된 권한을 업데이트할 경우 정책이 연결되어 있는 모든 보안 주체 엔터티(사용자, 그룹 및 역할)에도 업데이트가 적용됩니다. 새로운 AWS 서비스를 시작하거나 새로운 API 작업을 기존 서비스에 이용하는 경우 AWS가 AWS 관리형 정책을 업데이트할 가능성이 높습니다.

자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책을 참조하십시오.

AWS 관리형 정책: ElastiCacheServiceRolePolicy

ElastiCacheServiceRolePolicy를 IAM 엔터티에 연결할 수 없습니다. 이 정책은 ElastiCache가 사용자를 대신하여 작업을 수행할 수 있도록 서비스 연결 역할에 연결됩니다.

이 정책은 ElastiCache가 사용자를 대신하여 캐시 관리에 필요한 만큼 AWS 리소스를 관리할 수 있도록 허용합니다.

- `ec2` - VPC 엔드포인트(서버리스 캐시용), 탄력적 네트워크 인터페이스(ENI)(자체 설계된 클러스터용) 및 보안 그룹을 포함하여 캐시 노드에 연결할 EC2 네트워킹 리소스를 관리합니다.
- `cloudwatch` - 서비스에서 CloudWatch로 지표 데이터를 내보냅니다.
- `outposts` - AWS Outposts에 캐시 노드를 생성할 수 있습니다.

ElastiCacheServiceRolePolicy 정책은 IAM 콘솔 및 AWS 관리형 정책 참조 가이드의 ElastiCacheServiceRolePolicy에서 확인할 수 있습니다.

AWS 관리형 정책: AmazonElastiCacheFullAccess

`AmazonElastiCacheFullAccess` 정책을 IAM ID에 연결할 수 있습니다.

이 정책이 있으면 보안 주체는 AWS 관리 콘솔을 사용하여 ElastiCache에 완전히 액세스할 수 있습니다.

- `elasticache` - 전체 API에 액세스합니다.
- `iam` - 서비스 운영에 필요한 서비스 연결 역할을 생성합니다.
- `ec2` - 캐시 생성에 필요한 종속 EC2 리소스(VPC, 서브넷, 보안 그룹)를 설명하고 VPC 엔드포인트(서버리스 캐시용) 생성을 허용합니다.
- `kms` - 저장 시 암호화에 고객 관리형 CMK를 사용할 수 있습니다.
- `cloudwatch` - 지표에 대한 액세스를 허용하여 콘솔에 ElastiCache 지표를 표시할 수 있습니다.
- `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 생성용 AWS Outposts를 채우는 데 사용됩니다.
- `sns` - 콘솔에서 알림 기능용 SNS 주제를 채우는 데 사용됩니다.

AmazonElastiCacheFullAccess 정책은 IAM 콘솔 및 AWS 관리형 정책 참조 가이드의 AmazonElastiCacheFullAccess에서 확인할 수 있습니다.

AWS 관리형 정책: AmazonElastiCacheReadOnlyAccess

`AmazonElastiCacheReadOnlyAccess` 정책을 IAM ID에 연결할 수 있습니다.

이 정책이 있으면 보안 주체는 AWS 관리 콘솔을 사용하여 ElastiCache에 대해 읽기 전용으로 액세스할 수 있습니다.

- `elasticache` - 읽기 전용 `Describe` API에 액세스할 수 있습니다.

AmazonElastiCacheReadOnlyAccess 정책은 IAM 콘솔 및 AWS 관리형 정책 참조 가이드의 AmazonElastiCacheReadOnlyAccess에서 확인할 수 있습니다.

AWS 관리형 정책에 대한 ElastiCache 업데이트

이 서비스에서 이러한 변경 내용을 추적하기 시작한 이후 발생한 ElastiCache의 AWS 관리형 정책 업데이트에 대한 세부 정보를 확인합니다. 이 페이지의 변경 사항에 대한 자동 알림을 받으려면 ElastiCache 문서 기록 페이지에서 RSS 피드를 구독합니다.

변경 사항	설명	날짜
AmazonElastiCacheFullAccess - 기존 정책에 대한 업데이트	ElastiCache는 서버리스 캐시를 관리하고 콘솔을 통해 모든 서비스 기능을 사용할 수 있도록 하는 새로운 권한을 추가했습니다.	2023년 11월 27일
ElastiCacheServiceRolePolicy - 기존 정책에 대한 업데이트	ElastiCache는 서버리스 캐시 리소스의 VPC 엔드포인트를 관리할 수 있는 새로운 권한을 추가했습니다.	2023년 11월 27일
ElastiCache에서 변경 사항 추적 시작	ElastiCache에서 AWS 관리형 정책에 대한 변경 사항 추적을 시작했습니다.	2020년 2월 7일

Amazon ElastiCache에 대한 자격 증명 기반 정책(IAM 정책) 사용

이 항목에서는 계정 관리자가 IAM 자격 증명(사용자, 그룹, 역할)에 권한 정책을 연결할 수 있는 자격 증명 기반 정책의 예를 제공합니다.

> **Important**
>
> Amazon ElastiCache 리소스 액세스를 관리하기 위한 기본 개념과 옵션을 설명하는 주제를 먼저 읽어 보는 것이 좋습니다. 자세한 내용은 ElastiCache 리소스에 대한 액세스 권한 관리 개요 섹션을 참조하세요.

이 주제의 섹션에서는 다음 내용을 학습합니다.

- AWS 관리형 정책(Amazon ElastiCache용)
- 고객 관리형 정책 예제

다음은 권한 정책의 예입니다.

```
{
```

```
    "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 명령문은 Amazon ElastiCache 작업(`elasticache:Create*`, `elasticache:Describe*`, `elasticache:Modify*`)에 대한 권한을 부여합니다.
- 두 번째 명령문은 `Resource` 값의 끝에 지정된 IAM 역할 이름에서 IAM 작업(`iam:PassRole`)에 대한 권한을 부여합니다.

자격 증명 기반 정책에서는 권한을 가질 보안 주체를 지정하지 않으므로 이 정책은 `Principal` 요소를 지정하지 않습니다. 정책을 사용자에게 연결할 경우, 사용자는 암시적인 보안 주체입니다. IAM 역할에 권한 정책을 연결하면 역할의 신뢰 정책에서 식별된 보안 주체가 권한을 얻습니다.

모든 Amazon ElastiCache API 작업과 해당 작업이 적용되는 리소스를 보여주는 표는 ElastiCache API 권한: 작업, 리소스, 조건 참조 섹션을 참조하세요.

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 예

제 4: 사용자에게 IAM CreateServiceLinkedRole API 호출 허용 섹션을 참조하세요). 또는 ElastiCache 서비스 연결 역할을 생성해야 합니다.

Amazon ElastiCache 콘솔을 사용하는 데 필요한 최소 권한과 함께 이 섹션의 예제 정책을 사용할 경우 추가 권한이 부여됩니다. 예제는 AWS SDK 및 AWS CLI와도 관련이 있습니다.

IAM 사용자 및 그룹을 설정하는 것에 대한 지침은 IAM 사용 설명서의 IAM 사용자와 관리자 그룹 처음 만들기 섹션을 참조하세요.

> **Important**
>
> 프로덕션에서 IAM 정책을 사용하기 전에 항상 철저히 테스트하세요. 간단해 보이는 일부 ElastiCache 작업에서 ElastiCache 콘솔을 사용할 때 다른 작업이 해당 작업을 지원해야 할 수도 있습니다. 예를 들어, `elasticache:CreateCacheCluster`가 ElastiCache 캐시 클러스터를 생성하기 위한 권한을 부여합니다. 그러나 이 작업을 수행하기 위해 ElastiCache 콘솔에서 여러 `Describe` 및 `List` 작업을 사용하여 콘솔 목록을 채웁니다.

예제

- 예제 1: 사용자에게 ElastiCache 리소스에 대한 읽기 전용 액세스 허용
- 예제 2: 사용자가 일반 ElastiCache 시스템 관리자 작업을 수행하도록 허용
- 예제 3: 사용자가 모든 ElastiCache API 작업에 액세스하도록 허용
- 예제 4: 사용자에게 IAM CreateServiceLinkedRole API 호출 허용

예제 1: 사용자에게 ElastiCache 리소스에 대한 읽기 전용 액세스 허용

다음 정책은 사용자에게 리소스를 나열하도록 허용하는 ElastiCache 작업에 대한 권한을 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 일반 ElastiCache 시스템 관리자 작업을 수행하도록 허용

일반적인 시스템 관리자 작업으로는 리소스 수정이 있습니다. 시스템 관리자가 ElastiCache 이벤트에 대한 정보를 보고 싶어할 수도 있습니다. 다음 정책은 이러한 일반 시스템 관리자 작업을 위한 ElastiCache 작업을 수행할 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 작업을 호출할 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: 사용자에게 IAM CreateServiceLinkedRole API 호출 허용

다음 정책은 사용자가 IAM `CreateServiceLinkedRole` API를 호출하도록 허용합니다. 변화하기 쉬운 ElastiCache 작업을 호출하는 사용자에게 이 유형의 권한 정책을 부여하는 것이 좋습니다.

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"CreateSLRAllows",
      "Effect":"Allow",
      "Action":[
        "iam:CreateServiceLinkedRole"
      ],
      "Resource":"*",
      "Condition":{
        "StringLike":{
          "iam:AWSServiceName":"elasticache.amazonaws.com"
        }
      }
    }
  ]
}
```

리소스 수준 권한

IAM 정책에 리소스를 지정하여 권한의 범위를 제한할 수 있습니다. 많은 ElastiCache API 작업은 작업의 동작에 따라 달라지는 리소스 유형을 지원합니다. 각 IAM 정책 구문은 리소스에 대해 수행되는 작업에 대한 권한을 부여합니다. 지명된 리소스에서 이루어지는 작업이 아니거나 모든 리소스에 대해 그 작업을 수행할 수 있도록 권한을 부여하는 경우, 정책에서 해당 리소스의 값은 와일드카드(*)가 됩니다. 대부분의 API 작업에서는 리소스의 Amazon 리소스 이름(ARN) 또는 복수의 리소스에 맞는 ARN 패턴을 지정함으로써 사용자 수정이 가능한 리소스를 제한할 수 있습니다. 리소스별 권한을 제한하려면 ARN으로 리소스를 지정하세요.

ElastiCache 리소스 유형 및 해당 ARN 목록을 보려면 서비스 인증 참조에 나와 있는 Amazon ElastiCache에서 정의한 리소스를 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 Amazon ElastiCache에서 정의한 작업을 참조하세요.

예제

- 예제 1: 사용자에게 특정 ElastiCache 리소스 유형에 대한 모든 액세스 권한 허용
- 예제 2: 서버리스 캐시에 대한 사용자 액세스 거부

예제 1: 사용자에게 특정 ElastiCache 리소스 유형에 대한 모든 액세스 권한 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. ElastiCache에서 JSON 정책의 `Condition` 요소를 사용하여 요청 컨텍스트의 키를 정책에서 지정한 키 값과 비교할 수 있습니다. 자세한 정보는 IAM JSON 정책 요소: 조건을 참조하세요.

ElastiCache 조건 키 목록을 보려면 서비스 승인 참조의 Amazon ElastiCache용 조건 키를 참조하세요.

글로벌 조건 키의 목록은 AWS 글로벌 조건 컨텍스트 키를 참조하세요.

조건 지정: 조건 키 사용

세분화된 제어를 구현하려면 특정 요청에 대한 개별 파라미터 집합을 제어하는 조건을 지정하는 IAM 권한 정책을 작성합니다. 그런 다음 IAM 콘솔을 사용하여 만드는 IAM 사용자, 그룹 또는 역할에 정책을 적용합니다.

조건을 적용하려면 IAM 정책 설명에 조건 정보를 추가합니다. 다음 예제에서는 생성된 모든 자체 설계 캐시 클러스터가 `cache.r5.large` 노드 유형이 되도록 조건을 지정합니다.

```
{
    "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 태그 기반 액세스 제어 정책 예제 섹션을 참조하세요.

정책 조건 연산자 사용에 대한 자세한 내용은 ElastiCache API 권한: 작업, 리소스, 조건 참조 섹션을 참조하세요.

정책 예: 조건을 사용하여 세부적인 파라미터 제어 구현

이 섹션에서는 이전에 나열된 ElastiCache 파라미터에 대한 세분적인 액세스 제어를 구현하기 위한 정책 예를 보여 줍니다.

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:MaximumECPUPerSecond: 서버리스 캐시의 초당 최대 ECPU 값을 지정합니다. 고객은 제공된 조건을 사용할 때 초당 특정 수의 ECPU보다 많이 실행할 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:MaximumECPUPerSecond": "100000"
                }
            }
        }
    ]
}
```

3. elasticache:CacheNodeType: 사용자가 생성할 수 있는 NodeType을 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"
                    ]
                }
            }
        }
    ]
}
```

4. elasticache:EngineVersion: 엔진 버전 1.6.6의 사용을 지정합니다.

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

5. elasticache:KmsKeyId: 고객 관리형AWS KMS 키의 사용을 사용합니다.

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

6. elasticache:CacheParameterGroupName: 클러스터에서 조직의 특정 파라미터가 있는 기본값이 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"의 사용을 제한하는 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"
                }
            }
        }
```

```
    ]
}
```

7. elasticache:CreateCacheCluster: 요청 태그 `CreateCacheCluster`가 누락되었거나 `Project`, `Dev` 또는 `QA`와 같지 않은 경우 `Prod` 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"
                    ]
                }
            }
        }
    ]
}
```

8. elasticache:CacheNodeType: cacheNodeType cache.r5.large 또는 cache.r6g.4xlarge 및 Project=XYZ 태그를 사용하여 CreateCacheCluster를 허용합니다.

```
{
  "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.

Amazon ElastiCache에 대해 서비스 연결 역할 사용

Amazon ElastiCache는 AWS Identity and Access Management(IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 Amazon ElastiCache와 같은 AWS 서비스에 직접 연결된 고유한 유형의 IAM 역할입니다. Amazon ElastiCache 서비스 연결 역할은 Amazon ElastiCache에 의해 미리 정의됩니다. 서비스에서 클러스터를 대신하여 AWS 서비스를 호출하기 위해 필요한 모든 권한을 포함합니다.

필요한 권한을 수동으로 추가할 필요가 없으므로 서비스 연결 역할은 Amazon ElastiCache를 더 쉽게 설정할 수 있습니다. 이 역할은 AWS 계정에 이미 속해 있으나 Amazon ElastiCache 사용 사례에 연결되어 있으며 사전 정의된 권한을 가지고 있습니다. Amazon ElastiCache만 이러한 역할을 맡을 수 있고 이러한 역할만 사전 정의된 권한 정책을 사용할 수 있습니다. 먼저 역할의 관련 리소스를 삭제해야만 역할을 삭제할 수 있습니다. 이렇게 하면 Amazon ElastiCache 리소스에 대한 필수 액세스 권한을 부주의로 삭제할 수 없기 때문에 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용은 IAM으로 작업하는 AWS 서비스를 참조하고 서비스 연결 역할(Service-Linked Role) 열에 예(Yes)가 있는 서비스를 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 링크가 있는 예를 선택합니다.

목차

- Amazon ElastiCache에 대한 서비스 연결 역할 권한
  - 서비스 연결 역할을 생성하는데 필요한 권한
- 서비스 연결 역할 생성(IAM)
  - 서비스 연결 역할 생성(IAM 콘솔)

- 서비스 연결 역할 생성(IAM CLI)
- 서비스 연결 역할 생성(IAM API)
- Amazon ElastiCache에 대한 서비스 연결 역할의 설명 편집
  - 서비스 연결 역할 설명 편집(IAM 콘솔)
  - 서비스 연결 역할 설명 편집(IAM CLI)
  - 서비스 연결 역할 설명 편집(IAM API)
- Amazon ElastiCache에 대한 서비스 연결 역할 삭제
  - 서비스 연결 역할 정리
  - 서비스 연결 역할 삭제(IAM 콘솔)
  - 서비스 연결 역할 삭제(IAM CLI)
  - 서비스 연결 역할 삭제(IAM API)

Amazon ElastiCache에 대한 서비스 연결 역할 권한

서비스 연결 역할을 생성하는데 필요한 권한

IAM 엔터티가 AWS ServiceRoleForElastiCache 서비스 연결 역할을 생성하도록 허용하려면 다음과 같이 하세요.

IAM 개체에 대한 권한에 다음 정책 설명을 추가합니다.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:CreateServiceLinkedRole",
        "iam:PutRolePolicy"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/elasticache.amazonaws.com/
AWSServiceRoleForElastiCache*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "elasticache.amazonaws.com"}}
}
```

IAM 엔터티가 AWS ServiceRoleForElastiCache 서비스 연결 역할을 삭제하도록 허용하려면 다음과 같이 하세요.

IAM 개체에 대한 권한에 다음 정책 설명을 추가합니다.

```
{
```

```
    "Effect": "Allow",
    "Action": [
        "iam:DeleteServiceLinkedRole",
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/elasticache.amazonaws.com/
AWSServiceRoleForElastiCache*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "elasticache.amazonaws.com"}}
}
```

또는 AWS 관리형 정책을 사용하여 Amazon ElastiCache에 대한 전체 액세스 권한을 제공할 수 있습니다.

서비스 연결 역할 생성(IAM)

IAM 콘솔, CLI 또는 API를 사용하여 서비스 연결 역할을 생성할 수 있습니다.

서비스 연결 역할 생성(IAM 콘솔)

IAM 콘솔을 사용하여 서비스 연결 역할을 생성할 수 있습니다.

서비스 연결 역할을 만들려면(콘솔 사용)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. IAM 콘솔의 탐색 창에서 역할(Roles)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 서비스 선택에서 ElastiCache를 선택합니다.
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 역할 생성을 선택합니다.

서비스 연결 역할 생성(IAM CLI)

AWS Command Line Interface에서 IAM 작업을 사용하여 서비스 연결 역할을 생성할 수 있습니다. 이 역할에는 서비스가 역할을 수임하는 데 필요한 신뢰 정책 및 인라인 정책이 포함될 수 있습니다.

서비스 연결 역할을 만드는 방법(CLI)

다음 작업을 사용합니다.

```
$ aws iam create-service-linked-role --aws-service-name elasticache.amazonaws.com
```

서비스 연결 역할 생성(IAM API)

IAM API를 사용하여 서비스 연결 역할을 생성할 수 있습니다. 이 역할에는 서비스가 역할을 수임하는 데 필요한 신뢰 정책 및 인라인 정책이 포함될 수 있습니다.

서비스 연결 역할을 만들려면(API 사용)

CreateServiceLinkedRole API 호출을 사용합니다. 요청 시 `elasticache.amazonaws.com` 서비스 이름을 지정합니다.

Amazon ElastiCache에 대한 서비스 연결 역할의 설명 편집

Amazon ElastiCache에서는 AWSServiceRoleForElastiCache 서비스 연결 역할을 편집하도록 허용하지 않습니다. 서비스 연결 역할을 생성한 후에는 다양한 객체가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. 하지만 IAM을 사용하여 역할의 설명을 편집할 수 있습니다.

서비스 연결 역할 설명 편집(IAM 콘솔)

IAM 콘솔을 사용하여 서비스 연결 역할의 설명을 편집할 수 있습니다.

서비스 연결 역할의 설명을 편집하려면(콘솔 사용)

1. IAM 콘솔의 탐색 창에서 역할을 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)

AWS Command Line Interface의 IAM 작업을 사용하여 서비스 연결 역할의 설명을 편집할 수 있습니다.

서비스 연결 역할의 설명을 변경하려면(CLI 사용)

1. (선택 사항) 역할에 대한 현재 설명을 보려면 IAM 작업 `get-role`에 대한 AWS CLI를 사용하세요.

Example

```
$ aws iam get-role --role-name AWSServiceRoleForElastiCache
```

CLI 작업에서 역할을 참조하려면 ARN이 아니라 역할 이름을 사용해야 합니다. 예를 들어 어떤 역할의 ARN이 `arn:aws:iam::123456789012:role/myrole`인 경우 참조할 역할은 **myrole**입니다.

2. 서비스 연결 역할의 설명을 업데이트하려면 IAM 작업 update-role-description에 대한 AWS CLI를 사용하세요.

Linux, macOS, Unix의 경우:

```
$ aws iam update-role-description \
    --role-name AWSServiceRoleForElastiCache \
    --description "new description"
```

Windows의 경우:

```
$ aws iam update-role-description ^
    --role-name AWSServiceRoleForElastiCache ^
    --description "new description"
```

서비스 연결 역할 설명 편집(IAM API)

IAM API를 사용하여 서비스 연결 역할의 설명을 편집할 수 있습니다.

서비스 연결 역할의 설명을 변경하려면(API 사용)

1. (선택 사항) 역할의 현재 설명을 보려면 IAM API 작업 GetRole을 사용하세요.

Example

```
https://iam.amazonaws.com/
   ?Action=GetRole
   &RoleName=AWSServiceRoleForElastiCache
   &Version=2010-05-08
   &AUTHPARAMS
```

2. 역할 설명을 업데이트하려면 IAM API 작업 UpdateRoleDescription을 사용하세요.

Example

```
https://iam.amazonaws.com/
   ?Action=UpdateRoleDescription
   &RoleName=AWSServiceRoleForElastiCache
   &Version=2010-05-08
   &Description="New description"
```

Amazon ElastiCache에 대한 서비스 연결 역할 삭제

서비스 연결 역할이 필요한 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제하는 것이 좋습니다. 따라서 적극적으로 모니터링하거나 유지하지 않는 미사용 엔터티가 없도록 합니다. 단, 삭제 전에 서비스 연결 역할을 정리해야 합니다.

Amazon ElastiCache에서는 서비스 연결 역할을 자동으로 삭제하지 않습니다.

서비스 연결 역할 정리

IAM을 사용하여 서비스 연결 역할을 삭제하기 전에 먼저 역할에 해당 역할과 연결된 리소스(클러스터)가 없는지 확인합니다.

IAM 콘솔에서 서비스 연결 역할에 활성 세션이 있는지 확인하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. IAM 콘솔의 탐색 창에서 역할(Roles)을 선택합니다. 그런 다음 AWSServiceRoleForElastiCache 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.

AWS ServiceRoleForElastiCache가 필요한 Amazon ElastiCache 리소스를 삭제하려면 다음과 같이 하세요.

- 클러스터를 삭제하려면 다음을 참조하세요.
  - AWS Management Console 사용
  - AWS CLI 사용
  - ElastiCache API 사용

서비스 연결 역할 삭제(IAM 콘솔)

IAM 콘솔을 사용하여 서비스 연결 역할을 삭제할 수 있습니다.

서비스 연결 역할을 삭제하는 방법(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. IAM 콘솔의 탐색 창에서 역할을 선택합니다. 그런 다음 삭제할 역할의 이름이나 행이 아닌 이름 옆에 있는 확인란을 선택합니다.
3. 페이지 상단의 역할 작업에서 역할 삭제를 선택합니다.
4. 확인 대화 상자가 나타나면 서비스 마지막 액세스 데이터를 검토합니다. 이 데이터는 선택한 각 역할이 AWS 서비스를 마지막으로 액세스한 일시를 보여 줍니다. 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 보고 서비스 연결 역할 삭제 진행 상황을 모니터링합니다. IAM 서비스 연결 역할 삭제는 비동기이므로 삭제할 역할을 제출한 후에 삭제 태스크가 성공하거나 실패할 수 있습니다. 태스크에 실패할 경우 알림의 세부 정보 보기 또는 리소스 보기를 선택하면 삭제 실패 이유를 확인할 수 있습니다.

서비스 연결 역할 삭제(IAM CLI)

AWS Command Line Interface에서 IAM 작업을 사용하여 서비스 연결 역할을 삭제할 수 있습니다.

서비스 연결 역할을 삭제하는 방법(CLI)

1. 삭제할 서비스 연결 역할의 이름을 모르는 경우 다음 명령을 입력합니다. 이 명령은 계정의 역할과 해당 Amazon 리소스 이름(ARN)을 나열합니다.

   ```
   $ aws iam get-role --role-name role-name
   ```

   CLI 작업에서 역할을 참조하려면 ARN이 아니라 역할 이름을 사용해야 합니다. 예를 들어 역할의 ARN이 `arn:aws:iam::123456789012:role/myrole`인 경우 해당 역할을 **myrole**으로 참조합니다.
2. 서비스 연결 역할이 사용되지 않거나 연결된 리소스가 없는 경우에는 서비스 연결 역할을 삭제할 수 없으므로 삭제 요청을 제출해야 합니다. 이러한 조건이 충족되지 않으면 요청이 거부될 수 있습

니다. 삭제 태스크 상태를 확인하려면 응답의 deletion-task-id(을)를 캡처해야 합니다. 다음을 입력하여 서비스 연결 역할 삭제 요청을 제출합니다.

```
$ aws iam delete-service-linked-role --role-name role-name
```

3. 다음을 입력하여 삭제 작업의 상태를 확인합니다.

```
$ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
```

삭제 태스크는 NOT_STARTED, IN_PROGRESS, SUCCEEDED 또는 FAILED 상태일 수 있습니다. 삭제에 실패할 경우 문제를 해결할 수 있도록 실패 이유가 호출에 반환됩니다.

서비스 연결 역할 삭제(IAM API)

IAM API를 사용하여 서비스 연결 역할을 삭제할 수 있습니다.

서비스 연결 역할(API)을 삭제하는 방법

1. 서비스 연결 역할 삭제 요청을 제출하려면 DeleteServiceLinkedRole을 호출합니다. 요청에 역할 이름을 지정합니다.

   서비스 연결 역할이 사용되지 않거나 연결된 리소스가 없는 경우에는 서비스 연결 역할을 삭제할 수 없으므로 삭제 요청을 제출해야 합니다. 이러한 조건이 충족되지 않으면 요청이 거부될 수 있습니다. 삭제 태스크 상태를 확인하려면 응답의 DeletionTaskId(을)를 캡처해야 합니다.
2. 삭제 상태를 확인하려면 GetServiceLinkedRoleDeletionStatus을 호출합니다. 요청에 DeletionTaskId(을)를 지정합니다.

   삭제 태스크는 NOT_STARTED, IN_PROGRESS, SUCCEEDED 또는 FAILED 상태일 수 있습니다. 삭제에 실패할 경우 문제를 해결할 수 있도록 실패 이유가 호출에 반환됩니다.

ElastiCache API 권한: 작업, 리소스, 조건 참조

IAM 정책(보안 인증 기반 또는 리소스 기반)에 연결할 액세스 제어 및 쓰기 권한 정책을 설정할 때 다음 표를 참조로 사용하세요. 이 표에는 각 Amazon ElastiCache API 작업과 작업 수행 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. 리소스 ARN이 없는 경우, 정책의 리소스는 와일드카드(*)입니다.

ElastiCache 정책의 조건 키를 사용하여 조건을 표현할 수 있습니다. ElastiCache특정 조건 키 목록과 해당 조건 키가 적용되는 작업 및 리소스 유형을 보려면 을 참조하십시오조건 키 사용. AWS-wide 키의 전체 목록은 IAM 사용 설명서의 AWS 글로벌 조건 컨텍스트 키를 참조하십시오.

> **Note**
>
> 작업을 지정하려면 `elasticache:` 접두사 다음에 API 작업 명칭을 사용합니다(예: `elasticache:DescribeCacheClusters`).

ElastiCache 조치 목록을 보려면 서비스 승인 ElastiCache 참조의 Amazon이 정의한 작업을 참조하십시오.

Amazon에 대한 규정 준수 검증 ElastiCache

제3자 감사자는 SOC, PCI, FedRAMP, HIPAA와 같은 여러 AWS 규정 준수 프로그램의 일환으로 AWS 서비스의 보안 및 규정 준수를 평가합니다.

특정 규정 준수 프로그램의 범위 내에 AWS 서비스 있는지 알아보려면AWS 서비스 규정 준수 프로그램의 범위별 범위 내 규정 준수 프로그램별 규정을 선택하십시오. 일반 정보는 AWS 규정 준수 프로그램AWS 보증 프로그램 규정AWS 참조하십시오.

를 사용하여 AWS Artifact타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 의 보고서 https://docs.aws.amazon.com/artifact/latest/ug/downloading-documents.html 참조하십시오 AWS Artifact.

사용 시 규정 준수 AWS 서비스 책임은 데이터의 민감도, 회사의 규정 준수 목표, 관련 법률 및 규정에 따라 결정됩니다. AWS 규정 준수에 도움이 되는 다음 리소스를 제공합니다.

- 보안 및 규정 준수 퀵 스타트 가이드 - 이 배포 가이드에서는 아키텍처 고려 사항을 설명하고 보안 및 규정 준수에 AWS 중점을 둔 기본 환경을 배포하기 위한 단계를 제공합니다.

- Amazon Web Services의 HIPAA 보안 및 규정 준수를 위한 설계 — 이 백서에서는 기업이 HIPAA 적격 애플리케이션을 만드는 AWS 데 사용할 수 있는 방법을 설명합니다.

> **Note**
>
> 모든 AWS 서비스 사람이 HIPAA 자격을 갖춘 것은 아닙니다. 자세한 내용은 HIPAA 적격 서비스 참조를 참조하십시오.

- AWS 규정 준수 리소스AWS — 이 워크북 및 가이드 모음은 해당 산업 및 지역에 적용될 수 있습니다.
- AWS 고객 규정 준수 가이드 — 규정 준수의 관점에서 공동 책임 모델을 이해하십시오. 이 가이드에서는 보안을 유지하기 위한 모범 사례를 AWS 서비스 요약하고 여러 프레임워크 (미국 표준 기술 연구소 (NIST), 결제 카드 산업 보안 표준 위원회 (PCI), 국제 표준화기구 (ISO) 등) 에서 보안 제어에 대한 지침을 매핑합니다.
- 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— 이를 AWS 서비스 통해 내부 AWS보안 상태를 포괄적으로 파악할 수 있습니다. Security Hub는 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 참조하십시오.
- Amazon GuardDuty — 환경에 의심스럽고 악의적인 활동이 있는지 AWS 계정모니터링하여 워크로드, 컨테이너 및 데이터에 대한 잠재적 위협을 AWS 서비스 탐지합니다. GuardDuty 특정 규정 준수 프레임워크에서 요구하는 침입 탐지 요구 사항을 충족하여 PCI DSS와 같은 다양한 규정 준수 요구 사항을 해결하는 데 도움이 될 수 있습니다.
- AWS Audit Manager— 이를 AWS 서비스 통해 AWS 사용량을 지속적으로 감사하여 위험을 관리하고 규정 및 업계 표준을 준수하는 방법을 단순화할 수 있습니다.

추가 정보

AWS 클라우드 규정 준수에 대한 일반 정보는 다음을 참조하십시오.

- 서비스별 FIPS 엔드포인트
- 서비스 업데이트 ElastiCache
- AWS 클라우드 규정 준수
- 공동 책임 모델

- AWS PCI DSS 규정 준수 프로그램

Amazon ElastiCache의 복원성

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다. AWS 리전은 물리적으로 분리되고 격리된 다수의 가용 영역을 제공하며 이러한 가용 영역은 짧은 지연 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워크에 연결되어 있습니다. 가용 영역을 사용하면 중단 없이 가용 영역 간에 자동으로 장애 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

AWS 글로벌 인프라 외에도 Amazon ElastiCache는 데이터 복원성과 백업 요구 사항을 지원하는 다양한 기능을 제공합니다.

주제

- 장애 완화

장애 완화

Amazon ElastiCache 구현을 계획할 때는 장애가 애플리케이션과 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
- 추천

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 완화하

려면 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 노드로 클러스터를 생성할 수 있으며, 특별 요청에 따라 그 이상도 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 클러스터에 60개 이상의 노드가 필요하거나 한 AWS 지역에 총 300개 이상의 노드가 필요한 경우 https://aws.amazon.com/contact-us/elasticache-node-limit-request/ 에서 ElastiCache 한도 증가 요청 양식을 작성하십시오.

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. 드물지만 AZ 장애가 발생할 경우 해당 AZ에 캐시된 데이터가 손실되며 다른 AZ에 캐시된 데이터는 손실되지 않습니다.

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.

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

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. AZ에 노드를 균등하게 분산하면, 드물지만 AZ 장애가 발생할 경우 영향을 최소화합니다. 이 작업은 서버리스 캐시에서 자동으로 수행됩니다.

AWS ElastiCache의 인프라 보안

관리형 서비스인 AWS ElastiCache는 AWS 아키텍처 센터의 보안 및 규정 준수 섹션에 설명된 AWS 글로벌 네트워크 보안 절차에 의해 보호됩니다.

AWS에서 게시한 API 직접 호출을 사용하여 네트워크를 통해 ElastiCache에 액세스합니다. 클라이언트가 전송 계층 보안(TLS) 1.2 이상을 지원해야 합니다. TLS 1.3 이상을 권장합니다. 클라이언트는 Ephemeral Diffie-Hellman(DHE) 또는 Elliptic Curve Ephemeral Diffie-Hellman(ECDHE)과 같은 PFS(전달 완전 보안, Perfect Forward Secrecy)가 포함된 암호 제품군도 지원해야 합니다. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 자격 증명 및 IAM 보안 주체와 관련된 비밀 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)을 사용하여 임시 보안 자격 증명을 생성하여 요청에 서명할 수 있습니다.

서비스 업데이트 ElastiCache

ElastiCache 캐시, 클러스터 및 노드를 자동으로 모니터링하여 서비스 업데이트가 제공되는 대로 적용합니다. 서버리스 캐시에 대한 서비스 업데이트는 투명한 방식으로 자동 적용됩니다. 자체 설계된 클러

스터의 경우 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.

서비스 업데이트 관리

ElastiCache 자체 설계된 클러스터에 대한 서비스 업데이트는 정기적으로 릴리스됩니다. 해당 서비스 업데이트에 대해 적격한 자체 설계 클러스터가 하나 이상 있는 경우 업데이트가 릴리스되면 이메일, SNS, PHD (Personal Health Dashboard) 및 Amazon CloudWatch 이벤트를 통해 알림을 받게 됩니다. 업데이트는 콘솔의 서비스 업데이트 페이지에도 표시됩니다. ElastiCache 이 대시보드를 사용하면 ElastiCache 플릿에 대한 모든 서비스 업데이트와 해당 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 보안 업데이트 유형의 모든 업데이트를 가능한 한 빨리 적용하는 것이 좋습니다.

다음 섹션에서는 다음과 같은 옵션에 대해 상세히 알아봅니다.

주제

- 서비스 업데이트 적용
- 콘솔을 사용하여 최신 서비스 업데이트가 적용되었는지 확인 AWS
- 서비스 업데이트 중지

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 업데이트가 제공되면 아무 조치도 취하지 않도록 선택할 수 있습니다. ElastiCache 자동 업데이트 시작일 이후 클러스터의 예정된 유지 관리 기간 중 하나에 업데이트를 적용하도록 예약합니다. 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 클러스터에 적용하는 방법에 대한 자세한 내용은 을 참조하십시오. ElastiCache 콘솔을 사용한 서비스 업데이트 적용

하나 이상의 ElastiCache 클러스터에 새 서비스 업데이트를 사용할 수 있는 경우 ElastiCache 콘솔, API를 사용하거나 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 AWS Management Console 로그인하고 https://console.aws.amazon.com/elasticache/ 에서 아마존 ElastiCache 콘솔을 엽니다.
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 신청 날짜: 업데이트 적용 기한 ElastiCache 안내 날짜.
   - 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 을 참조하십시오 describe-service-updates.
- 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
```

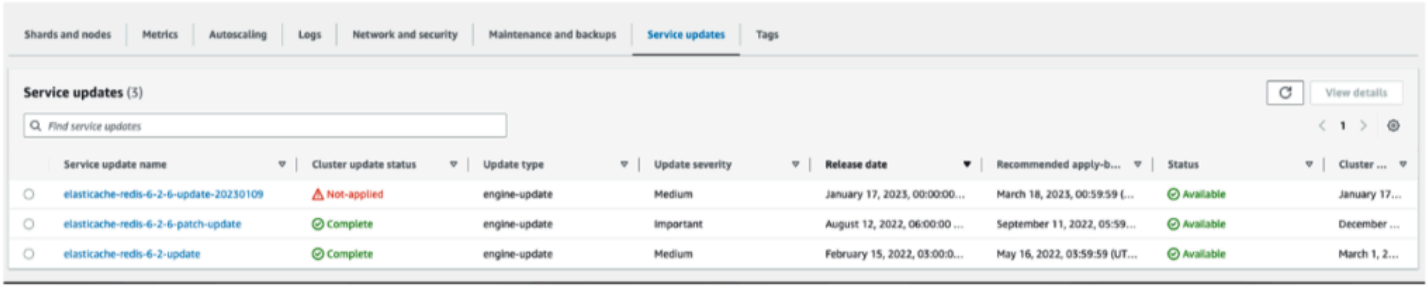
자세한 내용은 을 참조하십시오 batch-apply-update-action.

콘솔을 사용하여 최신 서비스 업데이트가 적용되었는지 확인 AWS

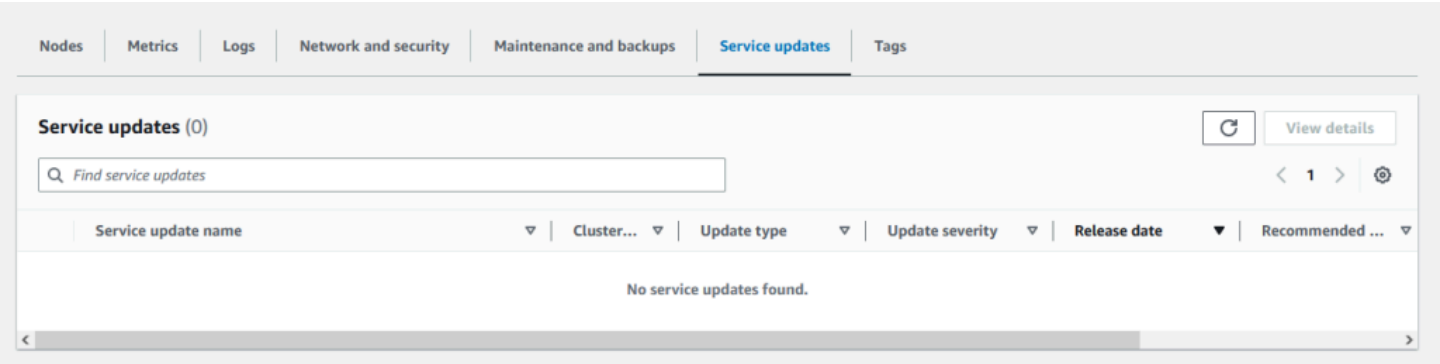
다음 단계에 따라 ElastiCache Redis용 클러스터가 최신 서비스 업데이트를 실행하고 있는지 확인할 수 있습니다.

1. Redis 클러스터 페이지에서 해당 클러스터를 선택합니다.
2. 탐색 창에서 서비스 업데이트를 선택하여 해당 클러스터에 해당하는 서비스 업데이트 (있는 경우)를 확인합니다.

콘솔에 서비스 업데이트 목록이 표시되면 서비스 업데이트를 선택하고 지금 적용을 선택할 수 있습니다.



콘솔에 “서비스 업데이트를 찾을 수 없음”이 표시되면 Redis ElastiCache 클러스터용 클러스터에 이미 최신 서비스 업데이트가 적용된 것입니다.



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 정보는 서비스 업데이트 적용을 참조하세요.

콘솔 사용

ElastiCache 콘솔을 사용하여 서비스 업데이트를 중단할 수 있습니다. 다음에서는 이 작업을 수행하는 방법을 보여 줍니다.

- 선택한 클러스터에서 서비스 업데이트가 진행되면 ElastiCache 콘솔에 대시보드 상단에 업데이트 보기/중지 탭이 표시됩니다. ElastiCache
- 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 을 참조하십시오 BatchStopUpdateAction.

Amazon ElastiCache에서 로깅 및 모니터링

캐시를 관리하려면 캐시의 작동 방식을 이해해야 합니다. ElastiCache는 Amazon CloudWatch Logs에 게시되는 지표를 생성하여 캐시 성능을 모니터링합니다. 또한 ElastiCache는 캐시 리소스에 중대한 변경 사항이 발생할 경우(예: 새 캐시 생성 또는 캐시 삭제) 이벤트를 생성합니다.

주제

- 서버리스 지표 및 이벤트
- 자체 설계된 클러스터 지표 및 이벤트
- AWS CloudTrail을 사용하여 Amazon ElastiCache API 호출 로깅

- AWS CloudTrail을 사용하여 Amazon ElastiCache API 호출 로깅

서버리스 지표 및 이벤트

이 섹션에는 서버리스 캐시로 작업할 때 모니터링할 수 있는 지표 및 이벤트에 대한 설명이 나와 있습니다.

주제

- 서버리스 캐시 지표
- 서버리스 캐시 이벤트

서버리스 캐시 지표

`AWS/ElastiCache` 네임스페이스에는 Memcached 서버리스 캐시에 대한 다음의 CloudWatch 지표가 포함되어 있습니다.

지표	설명	단위
BytesUsedForCache	캐시에 저장된 데이터에서 사용되는 총 바이트 수	바이트
ElastiCacheProcessingUnits	캐시에서 실행된 요청이 사용한 총 ElastiCacheProcessingUnits(ECPU) 수	개수

지표	설명	단위
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
ThrottledCmds	워크로드가 ElastiCache가 규모를 조정할 수 있는 속도보다 빠르게 확장되어 ElastiCache에서 병목 현상이 발생한 요청 수	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

또한 ElastiCache는 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
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

지표	설명	단위
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

ElastiCache는 서버리스 캐시와 관련된 이벤트를 로그합니다. 여기에는 이벤트 날짜 및 시간, 이벤트의 원본 이름 및 원본 유형, 이벤트 설명 등의 정보가 포함됩니다. ElastiCache 콘솔, AWS CLI describe-events 명령 또는 ElastiCache API 작업 `DescribeEvents`를 사용하여 로그에서 이벤트를 쉽게 검색할 수 있습니다.

Amazon EventBridge를 사용하여 ElastiCache 이벤트를 모니터링하고, 수집하고, 변환하고, 조치를 취하도록 선택할 수 있습니다. Amazon EventBridge https://docs.aws.amazon.com/eventbridge/latest/userguide/에서 자세히 알아보세요.

ElastiCache 이벤트 보기(콘솔)

ElastiCache 콘솔을 사용하여 이벤트를 보려면 다음과 같이 하세요.

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 사용 가능한 모든 이벤트의 목록을 보려면 탐색 창에서 이벤트를 선택합니다.
3. 이벤트 화면에서 목록의 각 행은 하나의 이벤트를 나타내며, 이벤트 소스, 이벤트 유형, 이벤트의 GMT 시간 및 이벤트 설명이 표시됩니다. [Filter]를 사용하여 이벤트 목록에서 모든 이벤트를 볼지 특정 유형의 이벤트만 볼지를 지정할 수 있습니다.

ElastiCache 이벤트 보기(AWS CLI)

AWS CLI를 사용하여 ElastiCache 이벤트의 목록을 생성하려면 describe-events 명령을 사용합니다. 선택적 파라미터를 사용하여 나열된 이벤트의 유형, 나열된 이벤트의 기간, 나열할 이벤트의 최대 수 등을 제어할 수 있습니다.

다음 코드는 최대 40개의 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(Source)	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. 사용 가능한 IP 주소가 부족하여 VPC 엔드포인트를 생성할 수 없습니다.
캐시 생성 실패	캐시 ARN	실패	serverless-cache	<cache-name> 캐시 생성에 실패했습니다. 요청에

Detail-Type	설명	단위	소스(Source)	메시지
				잘못된 서브넷이 제공되었습니다.
캐시 생성 실패	캐시 ARN	실패	serverless-cache	<cache-name> 캐시 생성에 실패했습니다. VPC 엔드포인트 생성을 위한 할당량 한도에 도달했습니다.
캐시 생성 실패	캐시 ARN	실패	serverless-cache	<cache-name> 캐시 생성에 실패했습니다. VPC 엔드포인트를 생성할 수 있는 권한이 없습니다.

서버리스 캐시 업데이트 이벤트

Detail-Type	리소스 목록	범주	소스(Source)	메시지
캐시 업데이트	캐시 ARN	구성 변경	serverless-cache	<cache-name> 캐시에 대한 SecurityGroups가 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 업데이트

Detail-Type	리소스 목록	범주	소스(Source)	메시지
				하지 못했습니다. SecurityGroups를 업데이트하지 못했습니다.
캐시 업데이트 실패	캐시 ARN	구성 변경	serverless-cache	<cache-name> 캐시를 업데이트하지 못했습니다. 권한이 충분하지 않아 SecurityGroups를 업데이트하지 못했습니다.
캐시 업데이트 실패	캐시 ARN	구성 변경	serverless-cache	<cache-name> 캐시를 업데이트하지 못했습니다. SecurityGroups가 잘못되어 SecurityGroups를 업데이트하지 못했습니다.

서버리스 캐시 삭제 이벤트

Detail-Type	리소스 목록	범주	소스(Source)	메시지
캐시 삭제	캐시 ARN	삭제	serverless-cache	<cache-name> 캐시가 삭제되었습니다.

서버리스 캐시 사용 제한 이벤트

Detail-Type	설명	단위	소스(Source)	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 잘못되어 <cache-name> 캐시에 대한 한도가 업데이트되지 못했습니다.

자체 설계된 클러스터 지표 및 이벤트

이 섹션은 자체 설계된 클러스터를 사용할 때 예상할 수 있는 지표, 이벤트 및 로그에 대해 설명합니다.

주제

- 자체 설계된 클러스터의 지표
- 자체 설계된 클러스터용 이벤트
- CloudWatch 지표를 사용한 사용량 모니터링
- ElastiCache 이벤트에 대한 Amazon SNS 모니터링

자체 설계된 클러스터의 지표

클러스터를 자체 설계하면 ElastiCache는 호스트 수준 지표와 캐시 지표를 모두 포함하여 각 노드 수준에서 지표를 내보냅니다.

Memcached의 호스트 수준 지표에 대한 자세한 내용은 호스트 수준 지표 섹션을 참조하세요.

Memcached의 노드 수준 지표에 대한 자세한 내용은 Memcached 지표 섹션을 참조하세요.

자체 설계된 클러스터용 이벤트

ElastiCache는 직접 설계된 캐시와 관련된 이벤트를 로그합니다. 자체 설계된 클러스터를 사용할 때는 ElastiCache 콘솔에서 AWS CLI 또는 Amazon Simple Notification Service(SNS)를 사용하여 클러스터 이벤트를 볼 수 있습니다. 자체 설계된 클러스터 이벤트는 Amazon EventBridge에 게시되지 않습니다.

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 ElastiCache API 작업 DescribeEvents를 사용하여 로그에서 이벤트를 쉽게 검색할 수 있습니다.

ElastiCache 이벤트 보기(콘솔)

다음 절차는 ElastiCache 콘솔을 사용하여 이벤트를 표시합니다.

ElastiCache 콘솔을 사용하여 이벤트를 보려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 사용 가능한 모든 이벤트의 목록을 보려면 탐색 창에서 이벤트를 선택합니다.
3. 이벤트 화면에서 목록의 각 행은 하나의 이벤트를 나타내며, 이벤트 소스, 이벤트 유형, 이벤트의 GMT 시간 및 이벤트 설명이 표시됩니다. [Filter]를 사용하여 이벤트 목록에서 모든 이벤트를 볼지 특정 유형의 이벤트만 볼지를 지정할 수 있습니다.

ElastiCache 이벤트 보기(AWS CLI)

AWS CLI를 사용하여 ElastiCache 이벤트의 목록을 생성하려면 describe-events 명령을 사용합니다. 선택적 파라미터를 사용하여 나열된 이벤트의 유형, 나열된 이벤트의 기간, 나열할 이벤트의 최대 수 등을 제어할 수 있습니다.

다음 코드는 최대 40개의 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.
무료 IP 주소가 부족함으로 인한 ElastiCache:AddCacheNodeFailed	`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.
호환되지 않는 네트워크 상태로 인한 ElastiCache:CacheClusterProvisioningFailed	`ElastiCache:CacheClusterProvisioningFailed : cluster-name`	존재하지 않는 Virtual Private Cloud(VPC)에서 새로운 캐시 클러스터를 실행하려고 시도했습니다.
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:CacheClusterSecurityGrou`	다음 이벤트 중 하나가 발생했습니다.

이벤트 이름	메시지	설명
	pModified : *cluster-name*	• 캐시 클러스터를 위한 승인된 캐시 보안 그룹이 수정되었습니다. • 하나 이상의 새로운 EC2 보안 그룹이 캐시 클러스터와 연결된 캐시 보안 그룹 중 하나에서 승인되었습니다. • 하나 이상의 EC2 보안 그룹이 캐시 클러스터와 연결된 캐시 보안 그룹 중 하나에서 승인이 취소되었습니다.

이벤트 이름	메시지	설명
ElastiCache:CacheNodeReplaceStarted	`ElastiCache:CacheNodeReplaceStarted : cluster-name`	ElastiCache가 캐시 노드를 실행하는 호스트 성능이 저하되었거나 연결되지 않음을 감지하여 캐시 노드 교체를 시작했습니다. **Note** 교체된 캐시 노드의 DNS 항목은 변경되지 않습니다. 대부분의 경우에 이 이벤트가 발생할 때 클라이언트의 서버 목록을 새로 고침하지 않아도 됩니다. 하지만 일부 캐시 클라이언트 라이브러리는 ElastiCache가 캐시 노드를 교체한 후에도 캐시 노드 사용을 중단할 수 있습니다. 이 경우에 애플리케이션은 이 이벤트가 발생할 때 서버 목록을 새로 고침해야 합니다.

이벤트 이름	메시지	설명
ElastiCache:CacheNodeReplaceComplete	`ElastiCache:CacheNodeReplaceComplete : cluster-name`	ElastiCache가 캐시 노드를 실행하는 호스트 성능이 저하되었거나 연결되지 않음을 감지하여 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. 하지만 일부 캐시 클라이언트 라이브러리는 ElastiCache가 캐시 노드를 교체한 후에도 캐시 노드 사용을 중단할 수 있습니다. 이 경우에 애플리케이션은 이 이벤트가 발생할 때 서버 목록을 새로 고침해야 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(Redis만 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(Redis만 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 Memcached용 캐시 노드 교체를 참조하세요.
ElastiCache:NodeReplacementScheduled	`ElastiCache:NodeReplacementScheduled : cluster-name`	클러스터의 노드가 알림에 설명된 기간 동안 교체가 예약됩니다. 수행할 수 있는 작업에 대한 자세한 내용은 Memcached용 캐시 노드 교체를 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(Redis만 해당)	`ElastiCache:SnapshotComplete : cluster-name`	캐시 스냅샷이 성공적으로 완료되었습니다.
ElastiCache:SnapshotFailed(Redis만 해당)	`SnapshotFailed : cluster-name`	캐시 스냅샷이 실패했습니다. 원인에 대한 자세한 내용은 클러스터의 캐시 이벤트를 참조하세요. 스냅샷을 설명할 경우 `DescribeSnapshots` 를 참조하세요. 상태는 `failed`입니다.

CloudWatch 지표를 사용한 사용량 모니터링

ElastiCache는 클러스터를 모니터링할 수 있는 지표를 제공합니다. CloudWatch를 통해 이러한 지표에 액세스할 수 있습니다. CloudWatch에 대한 자세한 내용은 CloudWatch 설명서를 참조하세요.

ElastiCache는 호스트 수준 지표(예: CPU 사용) 및 캐시 엔진 소프트웨어별 지표(예: 캐시가 얻은 것과 잃은 것) 모두를 제공합니다. 이러한 지표는 60초 간격으로 각 캐시 노드에 대해 측정되어 게시됩니다.

> **Important**
>
> 특정 키 지표에 CloudWatch 경보를 설정하면 캐시 클러스터의 성능이 나빠지기 시작하면 알림을 받을 수 있습니다. 자세한 내용은 이 가이드의 어떤 지표를 모니터링해야 합니까?을 참조하세요.

주제

- 호스트 수준 지표
- Memcached 지표
- 어떤 지표를 모니터링해야 합니까?
- CloudWatch 클러스터 및 노드 지표 모니터링

호스트 수준 지표

AWS/ElastiCache 네임스페이스에는 다음과 같이 개별 캐시 노드에 대한 호스트 수준 지표가 포함되어 있습니다.

참고 항목

- Memcached 지표

지표	설명	단위
CPUUtilization	전체 호스트에 대한 CPU 사용률 비율입니다.	%
CPUCreditBalance	시작 이후 인스턴스가 누적한 획득 CPU 크레딧 수입니다. T2 스탠다드의 경우 CPUCreditBalance에 누적된 시작 크레딧 수도 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. CPUCreditBalance의 크레딧은 인스턴스가 기준 CPU 사용률 이상으로 버스터를 하는 데 소비할 수 있습니다. CPU 크레딧 지표는 5분 간격으로만 제공됩니다.	크레딧(vCPU-분)
CPUCreditUsage	CPU 사용률을 위해 인스턴스에서 소비되는 CPU 크레딧의 수입니다. CPU 크레딧 하나는 1분 동안 100%의 사용률로 실행되는 vCPU 1개 또는 이와 동등한 vCPU, 사용률 및 시간의 조합과 동일합니다(예를 들어 2분 동안 50%의 사용	크레딧(vCPU-분)

지표	설명	단위
	률로 실행되는 vCPU 1개 또는 2분 동안 25%의 사용률로 실행되는 vCPU 2개). CPU 크레딧 지표는 5분 간격으로만 제공됩니다. 5분 이상의 시간을 지정할 경우 Sum 통계 대신 Average 통계를 사용하세요.	
FreeableMemory	호스트에서 사용 가능한 메모리의 양입니다. 이 값은 OS가 여유 공간으로 보고한 RAM, 버퍼, 캐시에서 나왔습니다.	바이트
NetworkBytesIn	호스트가 네트워크에서 읽어온 바이트 수입니다.	바이트
NetworkBytesOut	인스턴스가 모든 네트워크 인터페이스에서 보낸 바이트 수입니다.	바이트
NetworkPacketsIn	인스턴스가 모든 네트워크 인터페이스에서 받은 패킷 수입니다. 이 지표는 단일 인스턴스에서 수신 트래픽의 볼륨을 패킷 수 기준으로 식별합니다.	개수
NetworkPacketsOut	인스턴스가 모든 네트워크 인터페이스에서 보낸 패킷 수입니다. 이 지표는 단일 인스턴스에서 발신 트래픽의 볼륨을 패킷 수 기준으로 식별합니다.	개수
NetworkBandwidthInAllowanceExceeded	인바운드 집계 대역폭이 인스턴스의 최대값을 초과하여 형성된 패킷 수입니다.	개수
NetworkConntrackAllowanceExceeded	연결 추적이 인스턴스의 최대값을 초과하여 새 연결을 설정할 수 없기 때문에 형성된 패킷 수입니다. 이로 인해 인스턴스의 수신 또는 송신 트래픽에 대한 패킷 손실이 발생할 수 있습니다.	개수
NetworkBandwidthOutAllowanceExceeded	아웃바운드 집계 대역폭이 인스턴스의 최대값을 초과하여 형성된 패킷 수입니다.	개수

지표	설명	단위
Network Packets Per Second Allowance Exceeded	양방향 PPS(packets per second)가 인스턴스의 최대값을 초과하여 형성된 패킷 수입니다.	개수
NetworkMaxBytesIn	1분마다 수신된 바이트의 최대 버스트입니다.	바이트
NetworkMaxBytesOut	1분마다 전송된 바이트의 최대 버스트입니다.	바이트
NetworkMaxPacketsIn	1분마다 수신된 패킷의 최대 버스트입니다.	개수
NetworkMaxPacketsOut	1분마다 전송된 패킷의 최대 버스트입니다.	개수
SwapUsage	호스트에서 사용되는 스왑의 양입니다.	바이트

Memcached 지표

AWS/ElastiCache 네임스페이스에는 다음 Memcached 지표가 포함되어 있습니다.

AWS/ElastiCache 네임스페이스에는 Memcached stats 명령에서 파생된 다음 메트릭이 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
BytesWrittenOutFromMemcached	캐시 노드가 네트워크로 작성한 바이트 수	바이트

지표	설명	단위
CasBadval	CAS(Check And Set) 값이 저장된 CAS 값과 일치하지 않을 때 캐시가 수신한 CAS 요청 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
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 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
DecrMisses	요청한 키를 찾지 못했을 때 캐시가 수신한 decrement 요청 수	개수

지표	설명	단위
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
CmdTouch	누적된 touch 요청 수	개수
CurrConfig	현재 저장된 구성 수	개수

지표	설명	단위
EvictedUnfetched	LRU(Least Recently Used) 캐시에서 설정 이후 한 번도 사용하지 않아서 제거된 유효 항목 수	개수
ExpiredUnfetched	LRU에서 설정 이후 한 번도 사용하지 않아서 다시 회수된 만료 항목 수	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 네임스페이스에는 다음과 같은 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. 연결이 a용으로 예약되어 있기 때문에 이 값은 항상 1 이상이어야 합니다. 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. Memcached 오버헤드는 데이터에 사용되는 메모리 외에 메모리를 사용하므로 이 값을 추가 데이터에 사용할 수 있는 메모리 양으로 UnusedMemory 간주해서는 안 됩니다. 아직 미	바이트

지표	설명	단위
	사용 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 정보를 제공합니다. 대부분의 경우 이러한 지표에 대해 성능 문제가 생기기 전에 수정 조치를 취할 수 있도록 CloudWatch 경보를 설정하는 것이 좋습니다.

모니터링할 지표

- CPUUtilization
- SwapUsage
- Evictions
- CurrConnections

CPUUtilization

이는 비율(%)로 보고된 호스트 수준 지표입니다. 자세한 내용은 호스트 수준 지표 섹션을 참조하세요.

Memcached는 다중 스레드이므로 이 지표가 90%에 이를 수 있습니다. 이 임계값을 초과할 경우 대형 캐시 노드 유형을 사용하여 캐시 클러스터를 스케일 업하거나 더 많은 캐시 노드를 추가하여 스케일 아웃합니다.

SwapUsage

이는 바이트로 보고된 호스트 수준 지표입니다. 자세한 내용은 호스트 수준 지표 섹션을 참조하세요.

`FreeableMemory` CloudWatch 지표가 0에 가깝거나(즉, 100MB 미만) `SwapUsage` 지표가 `FreeableMemory` 지표보다 큰 경우 노드가 메모리 부족 상태임을 나타냅니다. 초과하면 ConnectionOverhead 파라미터 값을 높이는 것이 좋습니다.

Evictions

이것은 캐시 엔진 지표입니다. 애플리케이션 요구 사항에 따라 이 지표에 대한 경보 임계값을 결정하는 것이 좋습니다.

선택한 임계값을 초과할 경우 대형 노드 유형을 사용하여 클러스터를 스케일 업하거나 더 많은 노드를 추가하여 스케일 아웃합니다.

CurrConnections

이것은 캐시 엔진 지표입니다. 애플리케이션 요구 사항에 따라 이 지표에 대한 경보 임계값을 결정하는 것이 좋습니다.

CurrConnections 개수가 증가하는 것은 애플리케이션에 문제가 있음을 나타내며 이 문제를 해결하려면 애플리케이션 동작을 확인해야 합니다.

CloudWatch 클러스터 및 노드 지표 모니터링

ElastiCache와 CloudWatch가 서로 통합되어 있어서 다양한 지표를 수집할 수 있습니다. CloudWatch를 사용하여 이러한 지표를 모니터링할 수 있습니다.

> **Note**
>
> 다음 예제를 실행하려면 CloudWatch 명령줄 도구가 필요합니다. CloudWatch에 대해 자세한 내용을 알아보고 개발자 도구를 다운로드하려면 CloudWatch 제품 페이지를 참조하세요.

다음 절차에서는 CloudWatch를 사용하여 지난 시간 캐시 클러스터의 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 한도에 대한 자세한 내용은 ElastiCache에 대한 AWS 서비스 한도를 참조하세요.

CloudWatch 클러스터 및 노드 지표 모니터링(콘솔)

캐시 클러스터의 CPU 사용률 통계를 수집하려면

1. AWS Management Console에 로그인하고 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 Management Console의 캐시 클러스터 페이지에서 하나 이상의 캐시 클러스터 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 나타납니다.

d. [CPU Utilization] 지표를 클릭합니다.

   선택한 지표가 표시된 CloudWatch 콘솔이 열립니다. 통계 및 기간 드롭다운 목록 상자와 시간 범위 탭을 사용하여 표시되는 지표를 변경할 수 있습니다.

CloudWatch CLI를 사용하여 CloudWatch 클러스터 및 노드 지표 모니터링

캐시 클러스터의 CPU 사용률 통계를 수집하려면

- Linux, macOS 또는 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
```

CloudWatch API를 사용하여 CloudWatch 클러스터 및 노드 지표 모니터링

캐시 클러스터의 CPU 사용률 통계를 수집하려면

- CloudWatch API GetMetricStatistics를 다음 파라미터와 함께 호출합니다. 시작 및 종료 시간은 예제와 같이 표시되며 적절한 시작 및 종료 시간으로 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>
```

ElastiCache 이벤트에 대한 Amazon SNS 모니터링

클러스터에서 중요 이벤트가 발생하면 ElastiCache는 특정 Amazon SNS 주제에 알림을 전송합니다. 이러한 예에는 노드 추가 실패, 노드 추가 성공, 보안 그룹 수정 등이 있습니다. 주요 이벤트를 모니터링하면 클러스터의 현재 상태를 파악할 수 있으며, 이벤트에 따라 교정 작업을 수행할 수도 있습니다.

주제

- ElastiCache Amazon SNS 알림 관리
- ElastiCache 이벤트 보기
- 이벤트 알림 및 Amazon SNS

ElastiCache Amazon SNS 알림 관리

Amazon Simple Notification Service(Amazon SNS)를 사용하여 중요한 클러스터 이벤트에 대해 알림을 보내도록 ElastiCache를 구성할 수 있습니다. 이러한 예에서는 Amazon SNS 항목의 ARN(Amazon 리소스 이름)으로 클러스터를 구성하여 알림을 받습니다.

> **Note**
>
> 이 항목에서는 Amazon SNS에 가입했으며 Amazon SNS 주제를 설정 및 구독했다고 가정합니다. 이렇게 하는 방법에 대한 정보는 Amazon Simple Notification Service 개발자 안내서를 참조하세요.

Amazon SNS 주제 추가

다음 섹션은 AWS 콘솔, AWS CLI 또는 ElastiCache API를 사용하여 Amazon SNS 주제를 추가하는 방법을 보여줍니다.

Amazon SNS 주제 추가(콘솔)

다음 절차는 클러스터에 대해 Amazon SNS 주제를 추가하는 방법을 보여줍니다.

> **Note**
>
> 이 프로세스는 Amazon SNS 주제를 수정하는 데에도 사용할 수 있습니다.

클러스터에 대해 Amazon SNS 주제를 추가 또는 수정하려면(콘솔)

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. 클러스터에서 Amazon SNS 주제 ARN을 추가 또는 수정할 클러스터를 선택합니다.
3. 수정을 선택합니다.

4. 클러스터 수정의 SNS 알림에 대한 주제에서 추가하려는 SNS 주제를 선택하거나 수동 ARN 입력을 선택하고 Amazon SNS 주제의 ARN을 입력합니다.
5. 수정을 선택합니다.

Amazon SNS 주제 추가(AWS CLI)

클러스터에 대해 Amazon SNS 주제를 추가 또는 수정하려면 AWS CLI 명령 `modify-cache-cluster`를 사용합니다.

다음 코드 예제는 Amazon SNS 주제 arn을 my-cluster에 추가합니다.

Linux, macOS 또는 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 대해 Amazon SNS 주제를 추가 또는 수정하려면 다음 파라미터와 함께 `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 절차는 Amazon SNS 알림을 비활성화하는 방법을 보여줍니다.

Amazon SNS 알림 활성화 및 비활성화(콘솔)

AWS Management Console을 사용하여 Amazon SNS 알림을 비활성화하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/elasticache/에서 ElastiCache 콘솔을 엽니다.
2. Memcached를 실행 중인 클러스터의 목록을 보려면 탐색 창에서 Memcached를 선택합니다.
3. 알림을 수정할 클러스터의 이름 왼쪽에 있는 확인란을 선택합니다.
4. 수정을 선택합니다.
5. 클러스터 수정의 SNS 알림에 대한 주제에서 알림 비활성화를 선택합니다.
6. 수정을 선택합니다.

Amazon SNS 알림 활성화 및 비활성화(AWS CLI)

Amazon SNS 알림을 비활성화하려면 다음 파라미터와 함께 `modify-cache-cluster` 명령을 사용합니다.

Linux, macOS 또는 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 파라미터와 함께 `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

ElastiCache는 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 ElastiCache API 작업 `DescribeEvents`를 사용하여 로그에서 이벤트를 쉽게 검색할 수 있습니다.

다음 절차는 지난 24시간(1440분) 동안의 모든 ElastiCache 이벤트를 보는 방법을 보여줍니다.

ElastiCache 이벤트 보기(콘솔)

다음 절차는 ElastiCache 콘솔을 사용하여 이벤트를 표시합니다.

ElastiCache 콘솔을 사용하여 이벤트를 보려면

1. AWS Management Console에 로그인하고 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 이벤트 하나를 나타내며 이벤트 소스, 이벤트 유형(cache-cluster, cache-parameter-group, cache-security-group 또는 cache-subnet-group), 이벤트의 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)

AWS CLI를 사용하여 ElastiCache 이벤트의 목록을 생성하려면 `describe-events` 명령을 사용합니다. 선택적 파라미터를 사용하여 나열된 이벤트의 유형, 나열된 이벤트의 기간, 나열할 이벤트의 최대 수 등을 제어할 수 있습니다.

다음 코드는 최대 40개의 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)

ElastiCache API를 사용하여 ElastiCache 이벤트의 목록을 생성하려면 DescribeEvents 작업을 사용합니다. 선택적 파라미터를 사용하여 나열된 이벤트의 유형, 나열된 이벤트의 기간, 나열할 이벤트의 최대 수 등을 제어할 수 있습니다.

다음 코드는 40개의 최신 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

ElastiCache는 캐시 클러스터에서 중요 이벤트가 발생하는 경우 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**
>
> 이용 요금 정보 및 Amazon SNS 설명서 링크 등 Amazon Simple Notification Service(SNS)에 대한 자세한 내용은 Amazon SNS 제품 페이지를 참조하세요.

알림은 지정된 Amazon SNS 주제에 대해 게시됩니다. 다음은 알림에 대한 요구 사항입니다.

- ElastiCache 알림에 대해 주제 하나만 구성할 수 있습니다.
- Amazon SNS 주제를 소유한 AWS 계정은 알림이 활성화된 캐시 클러스터를 소유하는 계정과 동일해야 합니다.
- 게시할 Amazon SNS 주제는 암호화할 수 없습니다.

> **Note**
>
> 암호화된(미사용 데이터) Amazon SNS 주제를 클러스터에 연결할 수 있지만, ElastiCache 콘솔의 주제 상태가 비활성으로 표시되어 ElastiCache가 메시지를 주제로 푸시할 때 클러스터에서 주제가 실제로 연결 해제됩니다.

- Amazon SNS 주제는 ElastiCache 클러스터와 같은 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.
무료 IP 주소가 부족함으로 인한 ElastiCache:AddCacheNodeFailed	`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.
호환되지 않는 네트워크 상태로 인한 ElastiCache:CacheClusterProvisioningFailed	`ElastiCache:CacheClusterProvisioningFailed : cluster-name`	존재하지 않는 Virtual Private Cloud(VPC)에서 새로운 캐시 클러스터를 실행하려고 시도했습니다.
ElastiCache:CacheClusterScalingComplete	`CacheClusterScalingComplete : cluster-name`	캐시 클러스터의 조정이 성공적으로 완료되었습니다.
ElastiCache:CacheClusterScalingFailed	ElastiCache:CacheClusterScalingFailed : *cluster-name*	캐시 클러스터에 대한 스케일업 작업이 실패했습니다.

이벤트 이름	메시지	설명
ElastiCache:CacheClusterSecurityGroupModified	`ElastiCache:CacheClusterSecurityGroupModified : cluster-name`	다음 이벤트 중 하나가 발생했습니다. • 캐시 클러스터를 위한 승인된 캐시 보안 그룹이 수정되었습니다. • 하나 이상의 새로운 EC2 보안 그룹이 캐시 클러스터와 연결된 캐시 보안 그룹 중 하나에서 승인되었습니다. • 하나 이상의 EC2 보안 그룹이 캐시 클러스터와 연결된 캐시 보안 그룹 중 하나에서 승인이 취소되었습니다.

이벤트 이름	메시지	설명
ElastiCache:CacheNodeReplaceStarted	`ElastiCache:CacheNodeReplaceStarted : cluster-name`	ElastiCache가 캐시 노드를 실행하는 호스트 성능이 저하되었거나 연결되지 않음을 감지하여 캐시 노드 교체를 시작했습니다. **Note** 교체된 캐시 노드의 DNS 항목은 변경되지 않습니다. 대부분의 경우에 이 이벤트가 발생할 때 클라이언트의 서버 목록을 새로 고침하지 않아도 됩니다. 하지만 일부 캐시 클라이언트 라이브러리는 ElastiCache가 캐시 노드를 교체한 후에도 캐시 노드 사용을 중단할 수 있습니다. 이 경우에 애플리케이션은 이 이벤트가 발생할 때 서버 목록을 새로 고침해야 합니다.

이벤트 이름	메시지	설명
ElastiCache:CacheNodeReplaceComplete	`ElastiCache:CacheNodeReplaceComplete : cluster-name`	ElastiCache가 캐시 노드를 실행하는 호스트 성능이 저하되었거나 연결되지 않음을 감지하여 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. 하지만 일부 캐시 클라이언트 라이브러리는 ElastiCache가 캐시 노드를 교체한 후에도 캐시 노드 사용을 중단할 수 있습니다. 이 경우에 애플리케이션은 이 이벤트가 발생할 때 서버 목록을 새로 고침해야 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(Redis만 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(Redis만 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 Memcached용 캐시 노드 교체를 참조하세요.
ElastiCache:NodeReplacementScheduled	`ElastiCache:NodeReplacementScheduled : cluster-name`	클러스터의 노드가 알림에 설명된 기간 동안 교체가 예약됩니다. 수행할 수 있는 작업에 대한 자세한 내용은 Memcached용 캐시 노드 교체를 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(Redis만 해당)	`ElastiCache:SnapshotComplete : cluster-name`	캐시 스냅샷이 성공적으로 완료되었습니다.
ElastiCache:SnapshotFailed(Redis만 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

AWS CloudTrail을 사용하여 Amazon ElastiCache API 호출 로깅

Amazon ElastiCache는 Amazon ElastiCache에서 사용자, 역할 또는 AWS 서비스가 수행한 작업에 대한 레코드를 제공하는 서비스인 AWS CloudTrail과 통합됩니다. CloudTrail은 Amazon ElastiCache 콘솔의 호출과 Amazon ElastiCache API 작업에 대한 코드 호출을 포함하여 Amazon ElastiCache에 대한 모든 API 호출을 이벤트로 캡처합니다. 추적을 생성하면 Amazon ElastiCache 이벤트를 포함한 CloudTrail 이벤트를 지속적으로 Amazon S3 버킷에 배포할 수 있습니다. 추적을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 Amazon ElastiCache에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

CloudTrail의 Amazon ElastiCache 정보

CloudTrail은 계정 생성 시 AWS 계정에서 사용되도록 설정됩니다. Amazon ElastiCache에서 활동이 수행되면 해당 활동은 이벤트 기록에서 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩

니다. AWS 계정에서 최신 이벤트를 확인, 검색 및 다운로드할 수 있습니다. 자세한 내용은 CloudTrail 이벤트 기록을 사용하여 이벤트 보기를 참조하세요.

Amazon ElastiCache에 대한 이벤트를 포함하여 AWS 계정의 이벤트의 지속적인 레코드의 경우, 추적을 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 리전에 추적이 적용됩니다. 추적은 AWS 파티션에 있는 모든 리전의 이벤트를 로깅하고 지정된 Amazon S3 버킷으로 로그 파일을 전송합니다. 또는 CloudTrail 로그에서 수집된 이벤트 데이터를 추가 분석 및 처리하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음 자료를 참조하세요.

- 추적 생성 개요
- CloudTrail 지원 서비스 및 통합
- CloudTrail에 대한 Amazon SNS 알림 구성
- 여러 리전에서 CloudTrail 로그 파일 받기 및 여러 계정에서 CloudTrail 로그 파일 받기

모든 Amazon ElastiCache 작업은 CloudTrail에서 로깅되며 ElastiCache API Reference에서 문서화됩니다. 예를 들어 `CreateCacheCluster`, `DescribeCacheCluster`, `ModifyCacheCluster` 작업을 호출하면 CloudTrail 로그 파일에 항목이 생성됩니다.

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. 자격 증명 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 IAM 사용자 보안 인증 정보로 했는지 여부.
- 역할 또는 연합된 사용자에 대한 임시 보안 자격 증명을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청했는지 여부.

자세한 내용은 CloudTrail userIdentity 요소를 참조하세요.

Amazon ElastiCache 로그 파일 항목 이해

추적이란 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 입력할 수 있게 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스의 단일 요청을 나타내며 요청된 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보를 포함하고 있습니다. CloudTrail 로그 파일은 퍼블릭 API 호출에 대한 순서 지정된 스택 추적이 아니기 때문에 특정 순서로 표시되지 않습니다.

다음은 `CreateCacheCluster` 작업을 보여 주는 CloudTrail 로그 항목이 나타낸 예제입니다.

```
{
    "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"
}
```

다음은 DescribeCacheCluster 작업을 보여주는 CloudTrail 로그 항목이 나타낸 예제입니다. 모든 Amazon ElastiCache Describe 호출(Describe*)의 경우 ResponseElements 섹션이 제거되며 null로 표시됩니다.

```
{
    "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"
}
```

ElastiCache에 대한 할당량

AWS 계정에는 각 AWS 서비스에 대한 기본 할당량(이전에는 제한이라고 함)이 있습니다. 다르게 표시되지 않는 한 리전별로 각 할당량이 적용됩니다. 일부 할당량에 대한 증가를 요청할 수 있으며 다른 할당량은 늘릴 수 없습니다.

ElastiCache에 대한 할당량을 보려면 Service Quotas 콘솔을 엽니다. 탐색 창에서 AWS 서비스를 선택하고 ElastiCache를 선택합니다.

할당량 증가를 요청하려면 Service Quotas 사용 설명서의 할당량 증가 요청을 참조하세요. Service Quotas에서 아직 할당량을 사용할 수 없는 경우 한도 증가 양식을 사용합니다.

AWS 계정에는 ElastiCache와 관련하여 다음과 같은 할당량이 있습니다.

Resource	기본값
리전별 서버리스 캐시	40
리전당 노드	300
클러스터당 노드	40
리전당 파라미터 그룹	300
리전당 보안 그룹	50
리전당 서브넷 그룹	300
서브넷 그룹당 서브넷 수	20

Reference

이 섹션의 본 주제에서는 Amazon ElastiCache API 및 AWS CLI의 ElastiCache 섹션 관련 작업을 다룹니다. 또한 여기에는 일반적인 오류 메시지와 서비스 알림에 관한 설명이 포함되어 있습니다.

- ElastiCache API 사용
- ElastiCache API 참조
- AWS CLI 참조의 ElastiCache 섹션
- Amazon ElastiCache 오류 메시지
- 알림

ElastiCache API 사용

이 섹션에서는 ElastiCache 작업 사용 방법 및 구현에 관해 작업 중심으로 설명합니다. 이러한 작업에 대한 자세한 설명은 Amazon ElastiCache API 참조를 참조하세요.

주제

- Query API 사용
- 사용 가능한 라이브러리
- 애플리케이션 문제 해결

Query API 사용

쿼리 파라미터

HTTP 쿼리 기반 요청은 GET 또는 POST와 같은 HTTP 동사와 `Action` 쿼리 매개 변수를 사용하는 HTTP 요청입니다.

각 쿼리 요청은 인증 및 작업을 처리할 수 있도록 일부 공통 파라미터를 포함해야 합니다.

일부 작업은 파라미터의 목록을 허용합니다. 이러한 목록은 `param.`*`n`* 표기법을 사용하여 지정됩니다. *`n`*의 값은 1부터 시작하는 정수입니다.

쿼리 요청 인증

HTTPS를 통해서만 쿼리 요청을 보낼 수 있으며 모든 쿼리 요청에는 서명이 포함되어야 합니다. 이 섹션에서는 서명을 작성하는 방법을 설명합니다. 아래 절차에 설명된 방법은 서명 버전 4라고 합니다.

다음은 AWS에 대한 요청을 인증하는 데 사용되는 기본 단계입니다. 이 경우 사용자가 AWS에 등록되어 있으며 액세스 키 ID 및 보안 액세스 키를 가지고 있다고 가정합니다.

쿼리 인증 절차

1. 발신자가 AWS에 대한 요청을 구성합니다.
2. 발신자가 이 항목의 다음 섹션에 정의된 방법으로 SHA-1 해시 기능을 사용하는 HMAC(Hash-based Message Authentication Code)에 대한 키 해싱인 요청 서명을 계산합니다.
3. 요청의 발신자가 요청 데이터, 서명 및 액세스 키 ID(사용된 보안 액세스 키의 키 식별자)를 AWS로 보냅니다.
4. AWS는 액세스 키 ID를 사용하여 보안 액세스 키를 찾습니다.
5. AWS는 요청의 서명 계산에 사용된 동일한 알고리즘을 사용하여 요청 데이터 및 보안 액세스 키에서 서명을 생성합니다.
6. 서명이 일치하는 경우 요청이 인증되는 것으로 간주됩니다. 서명이 일치하지 않을 경우 요청이 삭제되고 AWS에서 오류 응답을 반환합니다.

> **Note**
>
> `Timestamp` 매개 변수가 요청에 포함된 경우 요청에 대해 계산된 서명은 그 매개 변수 값보다 15분 후에 만료됩니다.
> `Expires` 매개 변수가 요청에 포함된 경우 그 서명은 `Expires` 매개 변수에 의해 지정된 시간에 만료됩니다.

요청 서명을 계산하려면

1. 정규화된 쿼리 문자열을 만듭니다. 이 절차의 뒷부분에서 필요합니다.

   a. UTF-8 쿼리 문자열 구성 요소를 매개 변수 이름의 일반 바이트 순서로 정렬합니다. 이 매개 변수는 GET URI 또는 POST 요청 본문(Content-Type이 application/x-www-form-urlencoded일 경우)의 내용이 사용될 수 있습니다.

   b. 다음 규칙에 따라 매개 변수 이름과 값을 URL 인코딩합니다.

i. RFC 3986에 정의된 예약되지 않은 모든 문자는 URL 인코딩하지 않습니다. 이러한 예약되지 않은 문자는 A~Z, a~z, 0~9, 하이픈(-), 밑줄(_), 마침표(.) 및 물결표(~)입니다.

ii. %XY와 같이 모든 기타 문자를 퍼센트 인코딩합니다(여기서 X 및 Y는 16진 문자 0~9 및 대문자 A~F).

iii. 확장된 UTF-8 문자는 %XY%ZA... 형식으로 퍼센트 인코딩합니다.

iv. 공백 문자는 %20(일반 인코딩 구조인 +가 아님)으로 퍼센트 인코딩합니다.

c. 매개 변수 값이 비어있는 경우에도 인코딩된 매개 변수 이름을 인코딩된 매개 변수 값과 등호(=)(ASCII 문자 61)로 구별합니다.

d. 앰퍼샌드(&)(ASCII 코드 38)로 이름-값 쌍을 구별합니다.

2. 다음의 의사(pseudo) 문법("\n"은 ASCII 줄 바꿈을 나타냄)에 따라 서명할 문자열을 만듭니다.

```
StringToSign = HTTPVerb + "\n" +
ValueOfHostHeaderInLowercase + "\n" +
HTTPRequestURI + "\n" +
CanonicalizedQueryString <from the preceding step>
```

HTTPRequestURI 구성 요소는 URI의 HTTP 절대 경로 구성 요소이고 쿼리 문자열은 포함하지는 않습니다. HTTPRequestURI가 비어있는 경우 슬래시(/)를 사용합니다.

3. 사용자의 보안 액세스 키를 키로, SHA256 또는 SHA1을 해시 알고리즘으로 하여 방금 만든 문자열로 RFC 2104 호환 HMAC를 계산합니다.

   자세한 내용은 https://www.ietf.org/rfc/rfc2104.txt를 참조하세요.

4. 결과 값을 base64로 변환합니다.

5. 요청에서 `Signature` 매개 변수 값을 값으로 포함합니다.

예를 들어, 다음은 샘플 요청입니다(줄 바꿈이 명확성을 위해 추가됨).

```
https://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

AWS는 Query API 대신 언어별 API를 사용하여 애플리케이션을 구축하려는 소프트웨어 개발자를 위한 소프트웨어 개발 키트(SDK)를 제공합니다. 이러한 SDK는 보다 쉽게 시작하도록 요청 인증, 요청 재시도 및 오류 처리 같은 기본 기능(API에는 포함되지 않음)을 제공합니다. SDK 및 추가 리소스는 다음 프로그래밍 언어에 대해 제공됩니다.

- Java
- Windows 및 .NET
- PHP
- Python
- Ruby

다른 언어에 대한 자세한 내용은 샘플 코드 및 라이브러리를 참조하세요.

애플리케이션 문제 해결

ElastiCache는 ElastiCache API와 상호 작용하는 동안 발생하는 문제를 해결할 때 도움이 되도록 구체적이고 서술적인 오류를 제공합니다.

오류 검색

일반적으로 사용자는 시간을 소비하여 결과를 처리하기 전에 애플리케이션이 먼저 해당 요청으로 오류가 발생되는지 여부를 확인하려고 합니다. 오류 발생 여부를 확인하는 가장 쉬운 방법은 ElastiCache API의 응답에서 `Error` 노드를 찾는 것입니다.

XPath 구문은 `Error` 노드의 발생뿐만 아니라 오류 코드 및 메시지를 쉽게 검색할 수 있는 간단한 방법을 제공합니다. 다음 코드 조각에서는 요청 중에 오류가 발생했는지 여부를 파악하기 위해 Perl 및 XML::Xpath 모듈을 사용합니다. 오류가 발생되면 코드는 응답에 첫 번째 오류 코드와 메시지를 인쇄합니다.

```
use XML::XPath;
my $xp = XML::XPath->new(xml =>$response);
if ( $xp->find("//Error") )
{print "There was an error processing your request:\n", " Error code: ",
$xp->findvalue("//Error[1]/Code"), "\n", " ",
$xp->findvalue("//Error[1]/Message"), "\n\n"; }
```

문제 해결 팁

다음 절차를 통해 ElastiCache API의 문제를 진단하고 해결하는 것이 좋습니다.

- ElastiCache가 올바르게 실행되는지 확인합니다.

  이렇게 하려면, 브라우저 창을 열고 ElastiCache 서비스(https://elasticache.amazonaws.com 등)에 쿼리 요청을 제출하면 됩니다. MissingAuthenticationTokenException 또는 500 내부 서버 오류가 발생하는 경우 서비스가 사용 가능하고 요청에 응답하는지 확인할 수 있습니다.
- 요청 구조 확인.

  각 ElastiCache 작업에 대한 참조 페이지는 ElastiCache API 참조에 있습니다. 파라미터를 올바르게 사용하고 있는지 여부를 다시 확인합니다. 어떤 문제가 발생할 수 있을 지에 대해 미리 알아보려면 샘플 요청이나 사용자 시나리오를 살펴보고 이러한 샘플이 유사한 작업을 하고 있는지 확인하세요.
- 포럼 확인.

  ElastiCache에는 그 과정에서 다른 사람들이 경험한 문제에 대한 해결책을 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.

ElastiCache용 AWS CLI로 이동하는 경우에만 이 주제의 지침을 따르십시오.

> **Important**
>
> Amazon ElastiCache 명령줄 인터페이스(CLI)는 API 버전 2014-09-30 이후의 ElastiCache 개선 사항을 지원하지 않습니다. 명령줄에서 최신 ElastiCache 기능을 사용하려면 AWS 명령줄 인터페이스를 사용합니다.

주제

- 필수 조건
- 명령줄 도구 얻기

- 도구 설정
- 도구에 대한 자격 증명 제공
- 환경 변수

필수 조건

이 문서는 Linux/UNIX 또는 Windows 환경에서 작업할 수 있음을 가정합니다. 또한 Amazon ElastiCache 명령줄 도구는 UNIX 기반 환경인 Mac OS X에서도 작동하지만 이 설명서에는 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 Linux/UNIX 고유 명령을 표시하기 위해서는 **$** 를, Windows 고유 명령에 대해서는 **C:\>** 을 사용합니다. 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 플랫폼 용도의 JRE를 살펴보고 다운로드하려면 Java SE Downloads를 참조하세요.

Java Home 변수 설정

명령줄 도구는 Java 런타임을 찾기 위해 환경 변수(JAVA_HOME)를 사용합니다. 이 환경 변수는 실행 가능한 `java`(Linux 및 UNIX) 또는 `java.exe`(Windows) 실행 파일을 차례로 포함하고 있는 `bin`이라는 하위 디렉토리가 있는 디렉토리의 전체 경로로 설정되어야 합니다.

Java Home 변수를 설정하려면

1. Java Home 변수를 설정합니다.
   - Linux 및 UNIX에서 다음 명령을 입력합니다.

     ```
     $ 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/UNIX에서 다음과 유사한 출력을 확인할 수 있습니다.

```
$ $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 사이트에서 ZIP 파일로 제공합니다. 이러한 도구는 Java로 작성되었으며 Windows 2000/XP/Vista/Windows 7, Linux/UNIX 및 Mac OSX에 대한 셸 스크립트를 포함하고 있습니다. ZIP 파일은 자체 포함되어 있고 설치가 필요 없으므로 간단히 해당 Zip 파일을 다운로드하여 로컬 머신의 디렉토리에 압축을 풉니다.

도구 설정

명령줄 도구는 지원 라이브러리를 찾기 위해 환경 변수(AWS_ELASTICACHE_HOME)를 사용합니다. 이 환경 변수를 먼저 설정해야 도구를 사용할 수 있습니다. 환경 변수를 명령줄 도구의 압축을 푼 디렉토리 경로로 설정합니다. 이 디렉토리 이름은 ElastiCacheCli-A.B.nnnn(A, B 및 n은 버전/릴리스 번호)이며 bin 및 lib라는 하위 디렉토리를 포함하고 있습니다.

AWS_ELASTICACHE_HOME 환경 변수를 설정하려면

- 명령줄 창을 열고 다음 명령 중 하나를 입력하여 AWS_ELASTICACHE_HOME 환경 변수를 설정합니다.
  - Linux 및 UNIX에서 다음 명령을 입력합니다.

```
$ export &AWS;_ELASTICACHE_HOME=<path-to-tools>
```

- Windows에서 다음 명령을 입력합니다.

```
C:\> set &AWS;_ELASTICACHE_HOME=<path-to-tools>
```

도구 사용을 좀 더 쉽게 하려면 도구의 BIN 디렉토리를 시스템 경로에 추가하는 것이 좋습니다. 이 설명서의 나머지 부분은 BIN 디렉토리가 시스템 경로에 있음을 가정합니다.

도구의 BIN 디렉토리를 시스템 경로에 추가하려면

- 다음 명령을 입력하여 도구의 BIN 디렉토리를 시스템 경로에 추가합니다.
  - Linux 및 UNIX에서 다음 명령을 입력합니다.

```
$ 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 제공되는 AWS 액세스 키 및 보안 액세스 키가 필요합니다. 명령줄을 사용하거나 로컬 시스템에 위치하는 자격 증명 파일에서 이러한 키를 얻을 수 있습니다.

배포 파일에는 개별 정보를 사용하여 수정해야 하는 템플릿 파일인 ${AWS_ELASTICACHE_HOME}/credential-file-path.template가 포함되어 있습니다. 다음은 템플릿 파일의 콘텐츠입니다.

```
AWSAccessKeyId=<Write your AWS access ID>
AWSSecretKey=<Write your AWS secret key>
```

Important

UNIX에서는 자격 증명 파일의 소유자로 권한을 제한합니다.

```
$ chmod 600 <the file created above>
```

자격 증명 파일 설정을 사용하여 AWS_CREDENTIAL_FILE 환경 변수를 설정해야 ElastiCache 도구가 정보를 찾을 수 있습니다.

AWS_CREDENTIAL_FILE 환경 변수를 설정하려면

1. 환경 변수를 설정합니다.
   - Linux 및 UNIX에서는 다음 명령을 사용하여 변수를 업데이트합니다.

     ```
     $ 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 사용법 페이지가 표시됩니다.

환경 변수

환경 변수는 스크립팅, 기본값 구성 또는 기본값 임시 재정의에 유용할 수 있습니다.

AWS_CREDENTIAL_FILE 환경 변수뿐만 아니라 ElastiCache 명령줄 인터페이스와 함께 포함된 API 도구는 대부분 다음 변수를 지원하지 않습니다.

- EC2_REGION - 사용할 AWS 리전입니다.
- AWS_ELASTICACHE_URL - 서비스 호출에 사용할 URL입니다. EC2_REGION이 지정되어 있거나 --region 파라미터가 전달되는 경우 다른 리전 엔드포인트를 지정할 필요가 없습니다.

다음 예제에서는 EC2_REGION이라는 환경 변수를 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

다음 오류 메시지가 Amazon ElastiCache에 의해 반환됩니다. ElastiCache, 기타 AWS 서비스 또는 Memcached에 의해 반환되는 다른 오류 메시지를 받을 수 있습니다. ElastiCache가 아닌 소스의 오류 메시지에 대한 설명은 오류 메시지를 생성하는 소스의 설명서를 참조하세요.

- Cluster node quota exceeded
- Customer's node 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 %n 이상의 노드가 필요한 경우 Amazon ElastiCache 노드 요청 양식을 사용하여 요청하세요.

자세한 내용은 Amazon Web Services 일반 참조에서 Amazon ElastiCache 제한 사항을 참조하세요.

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 %n 이상의 노드가 필요한 경우 Amazon ElastiCache 노드 요청 양식을 사용하여 요청하세요.

자세한 내용은 Amazon Web Services 일반 참조에서 Amazon ElastiCache 제한 사항을 참조하세요.

오류 메시지: InsufficientCacheClusterCapacity

원인: 현재 AWS에 요청에 대한 서비스를 제공할 수 있을 만큼 온디맨드 용량이 충분하지 않습니다.

솔루션:

- 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 ElastiCache Memcached를 위한 스케일링 섹션을 참조하세요.

알림

이 주제에서는 관심을 가질 수 있는 ElastiCache 알림을 다룹니다. 알림은 대부분 일시적이며, 솔루션을 찾아 구현할 때까지만 지속되는 상황이나 이벤트입니다. 알림에는 일반적으로 시작 날짜와 해결 날

짜가 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 새로운 또는 향상된 ElastiCache 기능을 소개하지 않습니다.

일반 ElastiCache 알림

현재 엔진별로 분류되지 않은 미해결 ElastiCache 알림은 없습니다.

ElastiCache for Memcached 알림

다음 ElastiCache 알림은 Memcached 엔진에 관한 것입니다.

ElastiCache for Memcached 관련 알림

- 알림: 세분화 결함을 유발하는 Memcached LRU 크롤러

알림: 세분화 결함을 유발하는 Memcached LRU 크롤러

⚠ 알림 날짜: 2017년 2월 28일
일부 환경에서는 클러스터에 Memcached LRU 크롤러의 세분화 결함으로 인한 불안정성이 표시될 수 있습니다. 이는 일정 기간 동안 존재하는 Memcached 엔진 내부의 문제입니다. 이 문제는 LRU 크롤러가 기본적으로 활성화된 Memcached 1.4.33에서 명백하게 드러납니다.
이 문제를 겪고 있는 경우 수정될 때까지 LRU 크롤러를 비활성화하는 것이 좋습니다. 이렇게 하려면 명령줄에서 `lru_crawler disable`을 사용하거나 `lru_crawler` 파라미터 값을 수정(선호됨)하세요.
해결된 날짜:
해결:

문서 기록

- API 버전: 2015-02-02
- 최신 설명서 업데이트: 2023년 11월 27일

다음 표에는 2018년 3월 이후 ElastiCache Memcached 사용 설명서 사용 설명서의 각 릴리스에서 변경된 주요 내용이 설명되어 있습니다. 이 설명서에 대한 업데이트 알림을 받으려면 RSS 피드에 가입하면 됩니다.

최신 ElastiCache Memcached 업데이트

변경 사항	설명	날짜
멤캐시드 ElastiCache 1.6.22에 대한 지원	ElastiCache 멤캐시드의 경우 이제 멤캐시드 1.6.22를 지원합니다. 버전 1.6.18의 누적된 버그 수정도 포함됩니다. 자세한 내용은 Memcached 버전 1.6.22를 참조하세요.	2024년 1월 10일
ElastiCache Memcached의 경우 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 있는 새로운 권한이 `ElastiCacheServiceRolePolicy` 및 `AmazonElastiCacheFullAccess` 에 추가되었습니다. 또한 `AmazonElastiCacheFullAccess` 정책을 사용하여 수정된 콘솔	2023년 11월 27일

	환경을 지원하는 권한이 추가되었습니다.	
멤캐시드 ElastiCache 1.6.17에 대한 지원	ElastiCache 멤캐시드의 경우 이제 멤캐시드 1.6.17을 지원합니다. 여기에는 Memcached 버전 1.6.12의 누적 버그 수정이 포함되어 있습니다. 자세한 내용은 Memcached 버전 1.6.17을 참조하십시오.	2023년 1월 18일
ElastiCache Memcached의 경우 이제 IPV6를 지원합니다.	ElastiCache 인터넷 프로토콜 버전 4 및 6 (IPv4 및 IPv6) 을 지원하므로 IPv4 연결만 허용하거나 IPv6 연결만 허용하거나 IPv4 및 IPv6 연결 모두 허용 (이중 스택) 하도록 클러스터를 구성할 수 있습니다. IPv6는 Nitro 시스템에 빌드된 모든 인스턴스에서 Memcached 엔진 버전 1.6.6 이상을 사용하는 워크로드에 지원됩니다. IPv6를 통한 액세스에 대한 추가 요금은 없습니다. ElastiCache 자세한 내용은 네트워크 유형 선택을 참조하세요.	2022년 11월 7일
ElastiCache Memcached의 경우 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 참조하십시오.	2022년 5월 26일

ElastiCache 이제 지원됩니다. PrivateLink	AWS PrivateLink 인터넷 게이트웨이, NAT 디바이스, VPN 연결 또는 Direct AWS Connect 연결 없이 ElastiCache API 작업에 비공개로 액세스할 수 있습니다. 자세한 내용은 Redis용 Amazon ElastiCache API 및 인터페이스 VPC 엔드포인트 (AWS PrivateLink) 또는 ElastiCache Amazon API용 및 Memcached의 인터페이스 VPC 엔드포인트 () 를 참조하십시오. AWS PrivateLink	2022년 1월 24일
리소스 태그 지정 및 조건 키 지원	ElastiCache 이제 클러스터와 기타 리소스를 관리하는 데 도움이 되는 태그 지정이 지원됩니다. ElastiCache 자세한 내용은 리소스 태그 지정을 참조하십시오. ElastiCache ElastiCache 조건 키에 대한 지원도 소개합니다. IAM 정책이 적용되는 방식을 결정하는 조건을 지정할 수 있습니다. 자세한 내용은 조건 키 사용을 참조하세요.	2021년 4월 7일
Memcached 1.6.6 지원	ElastiCache 멤캐시드의 경우 이제 멤캐시드 1.6.6을 지원합니다. Memcached 1.5.16 버전의 누적된 버그 수정도 포함됩니다. 자세한 내용은 Memcached 버전 1.6.6 섹션을 참조하세요.	2020년 11월 12일

ElastiCache 이제 AWS Outposts에서 사용할 수 있습니다	AWS Outposts는 거의 모든 데이터 센터, 코로케이션 공간 또는 온프레미스 시설에 네이티브 AWS 서비스, 인프라 및 운영 모델을 제공합니다. ElastiCache Outposts에 배포하여 클라우드에서와 마찬가지로 온프레미스에서 캐시를 설정, 운영 및 사용할 수 있습니다. 자세한 내용은 Redis에 대한 Outposts 사용 또는 Memcached에 대한 Outposts 사용을 참조하세요.	2020년 10월 8일
ElastiCache 이제 Local Zones를 지원합니다.	로컬 영역은 지리적으로 사용자와 가까운 AWS 지역의 확장입니다. 새 서브넷을 만들고 로컬 영역에 할당하여 상위 AWS 지역의 모든 가상 사설 클라우드 (VPC) 를 Local Zones로 확장할 수 있습니다. 자세한 내용은 Local Zones 사용을 참조하세요.	2020년 9월 25일
ElastiCache 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 이제 클러스터 자동 업데이트 지원 ElastiCache	Amazon은 ElastiCache 이제 서비스 업데이트의 “권장 적용 날짜”가 지난 후에 ElastiCache 클러스터의 자동 업데이트를 지원합니다. ElastiCache 유지 관리 기간을 사용하여 해당 클러스터의 자동 업데이트 일정을 잡습니다. 자세한 내용은 셀프 서비스 업데이트를 참조하세요.	2020년 5월 13일
Amazon은 ElastiCache 이제 T3 표준 캐시 노드를 지원합니다.	이제 Amazon에서 차세대 범용 버스트 가능 T3-Standard 캐시 노드를 시작할 수 있습니다. ElastiCache Amazon EC2의 T3 표준 인스턴스는 기준 수준의 CPU 성능과 더불어 누적된 크레딧이 소진될 때까지 언제든 CPU 사용량을 순간 확장할 수 있는 기능을 제공합니다. 자세한 내용은 지원되는 노드 유형 섹션을 참조하세요.	2019년 11월 12일
ElastiCache Memcached의 경우 이제 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. 그 덕분에 특히 피크 비즈니스 흐름 중에 서비스 중단을 최소화하고 보안 업데이트 준수 상태를 유지하는 데 도움이 됩니다. 자세한 내용은 ElastiCacheAmazon의 셀프 서비스 업데이트를 참조하십시오.	2019년 10월 9일

멤캐시드 ElastiCache 1.5.16에 대한 지원	ElastiCache 멤캐시드의 경우 이제 멤캐시드 1.5.16을 지원합니다. Memcached 1.5.14 및 Memcached 1.5.15 버전의 누적된 버그 수정도 포함됩니다. 자세한 내용은 Memcached 버전 1.5.16 섹션을 참조하세요.	2019년 9월 6일
ElastiCache 표준 예약 인스턴스 상품: 부분 선결제, 전액 선결제, 선결제 없음.	예약 인스턴스를 사용하면 인스턴스 유형 및 AWS 지역에 따라 Amazon ElastiCache 인스턴스를 1년 또는 3년 약정으로 예약할 수 있는 유연성을 제공합니다. 자세한 내용은 예약 노드로 비용 관리를 참조하세요.	2019년 1월 18일
멤캐시드 ElastiCache 1.5.10에 대한 지원	ElastiCache 멤캐시드의 경우 이제 멤캐시드 1.5.10을 지원합니다. no_modern 및 inline_ascii_resp 파라미터에 대한 지원이 포함됩니다. 자세한 내용은 Memcached 버전 1.5.10 섹션을 참조하세요.	2018년 11월 14일
사용 설명서 재구성	이제 단일 ElastiCache 사용 설명서가 재구성되어 Redis (Redis 사용 설명서용) 및 Memcached (Memcached 사용 ElastiCache 설명서용) 에 대한 별도의 사용 설명서가 있습니다. ElastiCache AWS CLI 명령 참조: Elasticache 섹션과 Amazon ElastiCache API 참조의 문서 구조는 변경되지 않습니다.	2018년 4월 20일

다음 표에는 2018년 3월 이전의 ElastiCache Memcached 사용 설명서 사용 설명서의 중요한 변경 내용이 설명되어 있습니다.

변경 사항	설명	변경 날짜
아시아 태평양(오사카-로컬) 리전 지원	ElastiCache 아시아 태평양 (오사카-로컬) 지역에 대한 지원이 추가되었습니다. 아시아 태평양(오사카) 리전은 현재 단일 가용 영역을 지원하고 있으며 초대를 통해서만 이루어집니다. 자세한 내용은 다음 자료를 참조하십시오. • 지원되는 리전 • 지원되는 캐시 노드 유형	2018년 2월 12일
EU(파리) 지원	ElastiCache EU (파리) 지역에 대한 지원이 추가되었습니다. 자세한 내용은 다음 자료를 참조하십시오. • 지원되는 리전 • 지원되는 캐시 노드 유형	2017년 12월 18일
중국(닝샤) 리전 지원	Amazon은 중국 (닝샤) 지역에 대한 지원을 ElastiCache 추가했습니다. 자세한 내용은 다음 자료를 참조하십시오. • 지원되는 리전 • 지원되는 캐시 노드 유형	2017년 12월 11일
서비스 연결 역할 지원	이번 릴리스에서는 서비스 연계 역할 (SLR) 에 대한 지원이 ElastiCache 추가되었습니다. 자세한 내용은 다음 자료를 참조하십시오. • Amazon ElastiCache에 대해 서비스 연결 역할 사용	2017년 12월 7일

변경 사항	설명	변경 날짜
	• 권한 설정 (신규 ElastiCache 사용자만 해당)	
R4 노드 유형 지원	이번 릴리스에서는 에서 지원하는 모든 AWS 지역에 지원 R4 노드 유형이 ElastiCache 추가되었습니다. ElastiCache R4 노드 유형을 온디맨드 또는 예약 캐시 노드로 구입할 수 있습니다. 자세한 내용은 다음 자료를 참조하십시오. • 지원되는 캐시 노드 유형 • Memcached 노드 유형별 파라미터	2017년 11월 20일
연결 패턴 항목	ElastiCache 설명서에는 Amazon VPC의 ElastiCache 클러스터에 액세스하기 위한 다양한 패턴을 다루는 항목이 추가되었습니다. 자세한 내용은 ElastiCache 사용 설명서의 Amazon VPC의 ElastiCache 캐시에 액세스하기 위한 액세스 패턴를 참조하십시오.	2017년 4월 24일
Memcached 1.4.34 지원	ElastiCache 이전 Memcached 버전에 대한 여러 수정 사항이 포함된 Memcached 버전 1.4.34에 대한 지원을 추가합니다. 자세한 내용은 Memcached on의 Memcached 1.4.34 릴리스 노트를 참조하십시오. GitHub	2017년 10월 4일
Memcached 1.4.33 지원	ElastiCache Memcached 버전 1.4.33에 대한 지원을 추가합니다. 자세한 내용은 다음 자료를 참조하십시오. • Memcached 버전 1.4.33 • Memcached 1.4.33 추가 파라미터	2016년 12월 20일

변경 사항	설명	변경 날짜
EU 서부(런던) 리전 지원	ElastiCache EU (런던) 지역에 대한 지원을 추가합니다. 현재는 노드 유형 T2 및 M4만 지원됩니다. 자세한 내용은 다음 자료를 참조하십시오. • 지원되는 리전 • 지원되는 캐시 노드 유형	2016년 12월 13일
캐나다(몬트리올) 리전 지원	ElastiCache 캐나다 (몬트리올) 지역에 대한 지원을 추가합니다. 현재 이 지역에서는 노드 유형 M4 및 T2만 지원됩니다. AWS 자세한 내용은 다음 자료를 참조하십시오. • 지원되는 리전 • 지원되는 캐시 노드 유형	2016년 12월 8일
M4 및 R3 노드 유형 지원	ElastiCache 남아메리카 (상파울루) 지역의 R3 및 M4 노드 유형과 중국 (베이징) 지역의 M4 노드 유형에 대한 지원을 추가합니다. 자세한 내용은 다음 자료를 참조하십시오. • 지원되는 리전 • 지원되는 캐시 노드 유형	2016년 11월 1일
미국 동부 2(오하이오) 리전 지원	ElastiCache M4, T2 및 R3 노드 유형을 사용하여 미국 동부 (오하이오) 지역 (us-east-2) 에 대한 지원을 추가합니다. 자세한 내용은 다음 자료를 참조하십시오. • 지원되는 리전 • 지원되는 캐시 노드 유형	2016년 10월 17일

변경 사항	설명	변경 날짜
M4 노드 유형 지원	ElastiCache 에서 지원하는 대부분의 AWS 지역에서 M4 제품군 노드 유형에 대한 지원을 추가합니다. ElastiCache M4 노드 유형을 온디맨드 또는 예약 캐시 노드로 구입할 수 있습니다. 자세한 내용은 다음 자료를 참조하십시오. • 지원되는 캐시 노드 유형 • Memcached 노드 유형별 파라미터	2016년 8월 3일
뭄바이 리전 지원	ElastiCache 아시아 태평양 (뭄바이) 지역에 대한 지원을 추가합니다. 자세한 내용은 다음 자료를 참조하십시오. • 지원되는 캐시 노드 유형 • Memcached 노드 유형별 파라미터	2016년 6월 27일
R3 노드 유형 지원	ElastiCache 중국 (베이징) 지역 및 남미 (상파울루) 지역의 R3 노드 유형에 대한 지원을 추가합니다. 자세한 내용은 지원되는 캐시 노드 유형을 참조하세요.	2016년 3월 16일
Lambda 함수를 ElastiCache 사용하여 액세스하기	Amazon VPC에서 액세스할 수 있도록 ElastiCache Lambda 함수를 구성하는 방법에 대한 자습서가 추가되었습니다. 자세한 설명은 ElastiCache 자습서 및 동영상 섹션을 참조하세요.	2016년 2월 12일
아시아 태평양(서울) 리전 지원	ElastiCache t2, m3 및 r3 노드 유형을 사용하여 아시아 태평양 (서울) (ap-북동부-2) 지역에 대한 지원을 추가합니다.	2016년 1월 6일

변경 사항	설명	변경 날짜
Memcached 1.4.28 지원	ElastiCache Memcached 버전 1.4.24에 대한 지원과 버전 1.4.14 이후의 Memcached 개선 사항을 추가합니다. 이 릴리스에서는 LRU(가장 오랫동안 사용되지 않음) 캐시 관리에 대한 지원이 백그라운드 작업으로 추가되고 jenkins 또는 murmur3 선택이 해싱 알고리즘, 새로운 명령 및 기타 버그 수정으로 추가됩니다. 자세한 내용은 Memcached 릴리스 정보를 참조하세요. .	2015년 8월 27일
PHP 5.6을 사용하는 Memcached Auto Discovery 지원	Amazon의 이번 릴리스에는 PHP 버전 5.6용 Memcached 자동 검색 클라이언트에 대한 지원이 ElastiCache 추가되었습니다. 자세한 설명은 PHP용 ElastiCache 클러스터 클라이언트에 대한 소스 코드 컴파일 섹션을 참조하세요.	2015년 7월 29일
새 주제: 외부에서 액세스 ElastiCache AWS	외부에서 ElastiCache 리소스에 액세스하는 방법에 대한 새 항목이 추가되었습니다 AWS. 자세한 내용은 ElastiCache 외부에서 액세스를 참조하십시오 AWS.	2015년 7월 9일
노드 대체 메시지 추가	ElastiCache 예약된 노드 교체와 관련된 세 개의 메시지,ElastiCache:NodeReplacementScheduled,ElastiCache:NodeReplacementRescheduled, 를 추가합니다ElastiCache. NodeReplacementCanceled 노드 교체가 예정된 경우 취할 수 있는 자세한 내용 및 조치는 ElastiCache 이벤트 알림 및 Amazon SNS 's를 참조하십시오.	2015년 6월 11일
비용 할당 태그 지원	ElastiCache 비용 할당 태그에 대한 지원을 추가합니다. 자세한 설명은 비용 할당 태그를 사용하여 비용을 모니터링합니다. 섹션을 참조하세요.	2015년 2월 9일

변경 사항	설명	변경 날짜
AWS GovCloud (미국 서부) 지역 지원	ElastiCache AWS GovCloud (미국 서부) (us-gov-west-1) 지역에 대한 지원을 추가합니다.	2015년 1월 29일
유럽(프랑크푸르트) 리전 지원	ElastiCache 유럽 (프랑크푸르트) (eu-central-1) 지역에 대한 지원을 추가합니다.	2015년 1월 19일
AWS CloudTrail API 호출 로깅이 지원됩니다.	ElastiCache 모든 ElastiCache API 호출을 AWS CloudTrail 기록하는 데 사용할 수 있는 지원을 추가합니다. 자세한 설명은 AWS CloudTrail을 사용하여 Amazon ElastiCache API 호출 로깅 섹션을 참조하세요.	2014년 9월 15일
새로운 인스턴스 크기 지원	ElastiCache 추가 범용 (T2) 인스턴스에 대한 지원을 추가합니다. 자세한 설명은 파라미터 그룹을 사용해 엔진 파라미터 구성 섹션을 참조하세요.	2014년 9월 11일
Memcached에서 유연한 노드 배치 지원	ElastiCache 여러 가용 영역에 Memcached 노드를 생성하기 위한 지원을 추가합니다.	2014년 7월 23일
새로운 인스턴스 크기 지원	ElastiCache 범용 (M3) 인스턴스 및 메모리 최적화 (R3) 인스턴스에 대한 지원을 추가합니다. 자세한 설명은 파라미터 그룹을 사용해 엔진 파라미터 구성 섹션을 참조하세요.	2014년 7월 1일
PHP 자동 검색	PHP 버전 5.5에 대한 지원이 추가되었습니다.	2014년 5월 13일
기본 Amazon Virtual Private Cloud(VPC) 지원	이번 릴리스에서는 Amazon VPC (가상 사설 클라우드) 와 완벽하게 ElastiCache 통합됩니다. 새로운 고객의 경우 캐시 클러스터가 기본적으로 Amazon VPC에 생성됩니다. 자세한 설명은 Amazon VPC 및 ElastiCache 보안 섹션을 참조하세요.	2013년 1월 8일

변경 사항	설명	변경 날짜
캐시 노드 Auto Discovery를 위한 PHP 지원	캐시 노드 Auto Discovery는 초기 릴리스부터 Java 프로그램에 대한 지원을 제공했습니다. 이번 릴리스에서는 PHP에 캐시 노드 자동 검색 지원을 ElastiCache 제공합니다.	2013년 1월 2일
Amazon Virtual Private Cloud(VPC) 지원	이번 릴리스에서는 Amazon VPC (가상 사설 클라우드) 에서 ElastiCache 클러스터를 시작할 수 있습니다. 기본적으로 새로운 고객의 캐시 클러스터가 자동으로 Amazon VPC에 생성됩니다. 기존 고객은 자신의 속도로 Amazon VPC로 마이그레이션할 수 있습니다. 자세한 설명은 Amazon VPC 및 ElastiCache 보안 섹션을 참조하세요.	2012년 20월 12일
캐시 노드 자동 검색 및 새로운 캐시 엔진 버전	ElastiCache 클라이언트 프로그램이 클러스터의 모든 캐시 노드를 자동으로 확인하고 이러한 모든 노드에 대한 연결을 시작 및 유지 관리하는 캐시 노드 자동 검색 기능을 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 설명은 파라미터 그룹을 사용해 엔진 파라미터 구성 섹션을 참조하세요.	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.	2012년 4월 5일
새 안내서	Amazon ElastiCache 사용 설명서의 첫 번째 릴리스입니다.	2011년 8월 22일

AWS 용어집

최신 AWS 용어는 참조의 AWS 용어집을 참조하십시오.AWS 용어집

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