RDS, Aurora & ElastiCache

일시	2023.03.26. (일)	작성자	김민경

1. RDS

1) 개요

-Relational Database Service(관계형 DB 서비스)의 약자
-SQL을 쿼리 언어로 사용
-클라우드의 RDS 서비스에 DB 생성, AWS가 DB 관리
-AWS가 관리하는 DB 엔진 유형

*SQL
:DB를 쿼리하는 구조화된 언어

• Postgres	• Oracle
• MySQL	• Microsoft SQL Server
• MariaDB	• Aurora

*** EC2 인스턴스 자체에 DB 서비스를 배포하지 x고 RDS를 사용하는 이유**

-RDS는 관리형 서비스

• DB 프로비저닝 & 기본 운영체제 패치 완전 자동화 • Point in Time Restore(지정시간 복구) • DB의 성능을 대시보디에서 모니터링 • 읽기 전용 복제본 활용해 읽기 성능 개선	• 재해 복구 목적으로 다중 AZ 설정 • 유지 관리 기간에 업그레이드 • 인스턴스 유형 늘려 수직 확장 or 읽기 전용 복제본 추가해 수평 확장 • 파일 스토리지는 EBS에 구성됨

-단점: RDS 인스턴스에 SSH 액세스x

*SSH (Secure Shell)
:원격 호스트에 접속하기 위해 사용되는 보안 프로토콜

2) Storage Auto Scaling

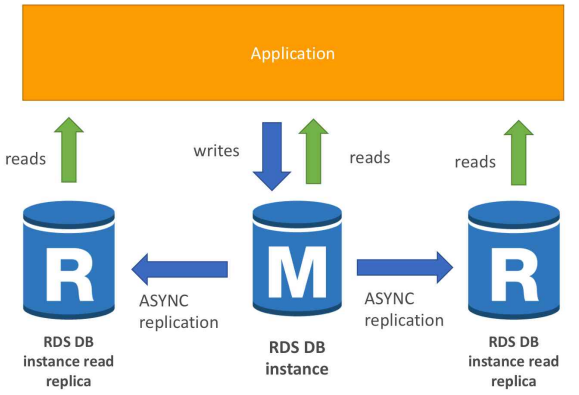
-RDS Storage Auto Scaling 기능 활성화 → RDS가 감지해 자동으로 스토리지 확장
∴확장할 스토리지 최대치 설정해야 함
-스토리지 자동 수정할 때

① 할당된 스토리지의 남은 공간이 10% 미만으로 떨어질 때
② 스토리지 부족 상태가 5분 이상 지속될 때
③ 지난 수정으로부터 6시간이 지나쓸 경우

-워크로드를 예측할 수 x는 app에 유용
-all RDS DB 엔진에서 지원되는 기능

3) RDS Read Replicas (RDS 읽기 전용 복제본)

(1) RDS Read Replicas for read scalability (읽기를 스케일링함)

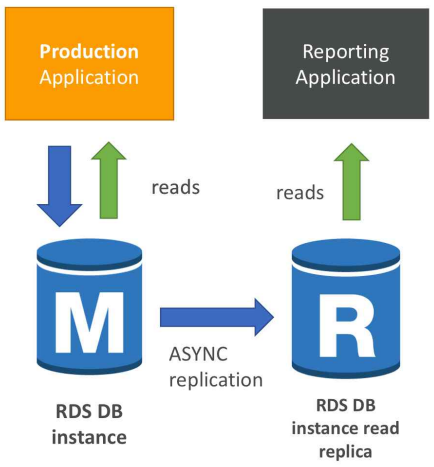


• app은 DB 인스턴스에 대해 읽기와 쓰기를 수행함
but 주된 DB 인스턴스가 너무 많은 요청을 받아
충분히 스케일링할 수 x어 읽기를 스케일링한다고 가정
• 이때 읽기 전용 복제본은 최대 5개까지 생성,
이들은 동일한 AZ or 리전으로 걸쳐 생성될 수 0
• 주된 RDS DB 인스턴스와 두 읽기 전용 복제본 사이에
비동기식 복제가 발생
(*비동기식 복제: 읽기가 일관적으로 유지된다는 것)
(app에서 데이터를 복제하기 전 읽기 전용 복제본을

	읽어들이면 all 데이터 얻을 수 0)

-읽기 전용 복제본을 DB로 승격 0

-Use Cases



• 가정: 데이터 기반으로 특정 보고 & 분석

-메인 RDS DB 인스턴스에 연결하면 오버로드 발생해 속도 느려짐

→읽기 전용 복제본 생성

→DB 인스턴스 & 읽기 전용 복제본 간 비동기식 복제

→읽기 전용 복제본에서 읽기 작업 & 분석 실시

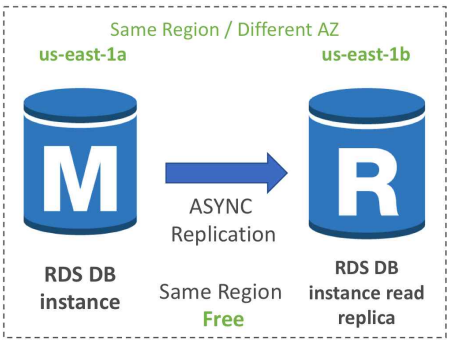
• 읽기 전용 복제본 있는 경우 SELECT 명령문에만 사용

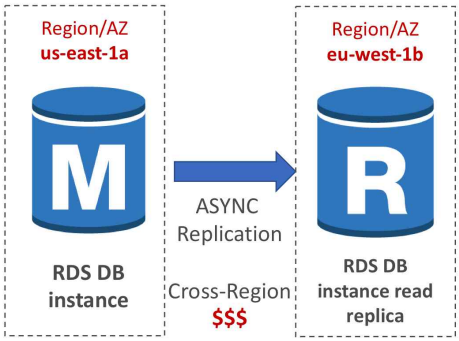
*SELECT
:SQL 키워드로 읽기를 의미

-DB 자체를 바꾸는 INSERT, UPDATE, DELETE 사용x

(∵읽기 전용 복제본은 only 읽기를 위한 것)

(2) RDS Read Replicas- Network Cost





• 읽기 전용 복제본이 다른 AZ 상이지만
동일한 리전 내에 있을 때 → 비용 발생x

• 복제 트래픽이 다른 AZ로 이동 시
⇒ 무료 (∵RDS⇒관리형 서비스)

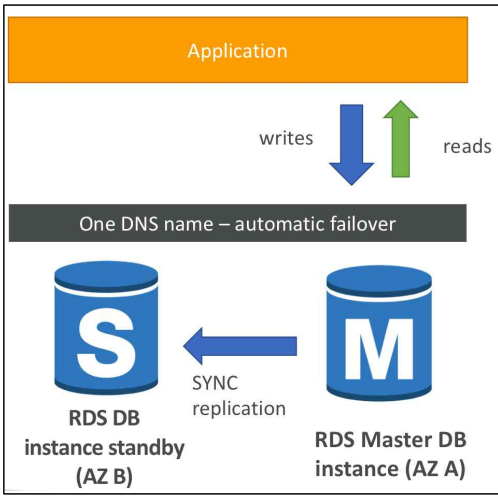
*원래 AZ 이동 시 비용 발생

• 다른 리전에 복제본이 존재
→ net에 대한 복제 비용 발생0

4) RDS Multi AZ (RDS 다중 AZ)

-주로 '재난 복구'에 사용

-동기식 복제



• 마스터 DB의 all 변경사항을 스탠바이 인스턴스로 복제

• 마스터 문제 발생 → 스탠바이 DB가 자동으로 새로운 마스터 됨
(∵하나의 DNS 이름을 가짐)
∴가용성 ⇑수 있음

• 스탠바이 DB ⇒단지 '대기' 목적 (읽거나 쓰기x)

• 무료

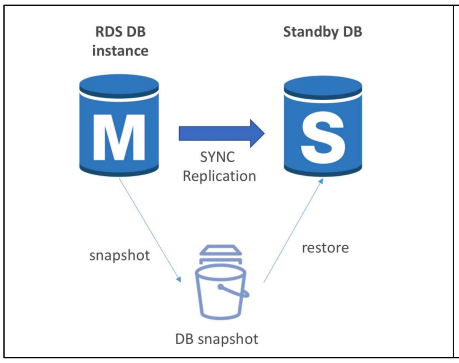
• 재해 복구를 대비해 읽기 전용 복제본을 다중 AZ로 설정 0

* RDS- From Single-AZ to Multi-AZ

-단일 AZ → 다중 AZ 전환 시 다운 타임x

(전환시 DB 중지 필요x)

-DB 수정 클릭하고 다중 AZ 기능 활성화하면 됨



① 기본 DB⇒ 자동으로 스냅샷 생성
② 이 스냅샷⇒ 새로운 스탠바이 DB에 복원됨
③ 복원되면 두 DB 간 동기화가 설정되므로 다중 AZ 설정 상태가 됨

5) RDS Custom (RDS 사용자 지정)

-Oracle & Microsoft SQL Server DB를 위한 것

-RDS에서는 OS or DB 사용자 지정 기능에 액세스x

but RDS Custom에서는 가능

-자동화 꺼놓는 것이 좋음

(RDS가 수시로 자동화, 유지관리 or 스케일링과 같은 수행하지 x도록)

-DB 스냅샷 만들어 놓는 것이 좋음

(∵복구 어려움)

RDS	• 전체 DB & OS 관리 • all 것 AWS에서 관리
RDS Custom	• Oracle & Microsoft SQL Server에만 사용 가능 • 기저 OS & DB에 대한 권리자 권한 전체를 가짐

2. Amazon Aurora

1) 개요

-AWS 독점 기술

-Postgre or MySQL DB에 호환

-스토리지 자동 확장(10GB-128TB) → 저장 디스크 신경x

-15개의 읽기 전용 복제본 가짐, MySQL DB는 5개, 복제 속도 빠름

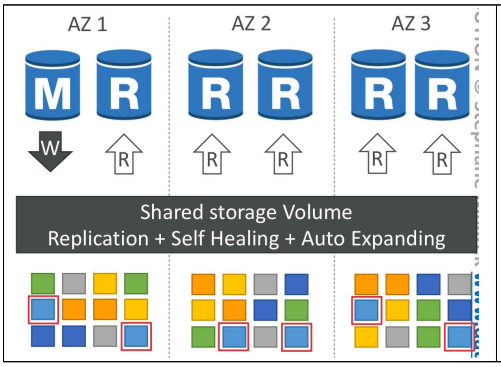
-자동 장애 조치

-HA

-RDS보다 비쌈

but 스케일링 측면에서 오히려 비용 절감0

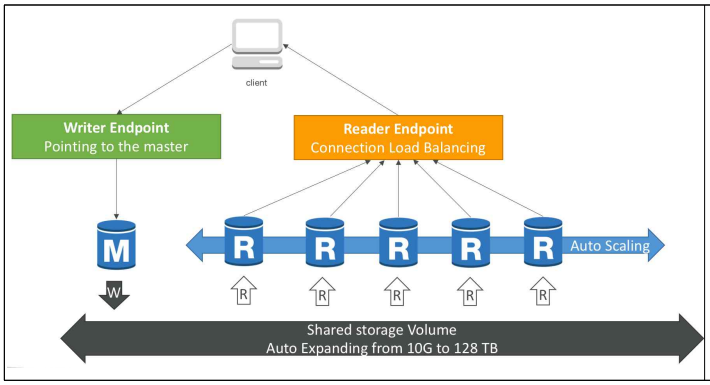
2) Aurora HA & Read Scaling



• 쓰기 요청할 때마다 6개의 복제본을 3개의 AZ에 걸쳐 저장
• 마스터 1개 ⇒ '쓰기'
• 마스터 장애 → 평균적 30s 안에 장애 조치 (장애 조치 빠름)
• 리전 간 복제 지원

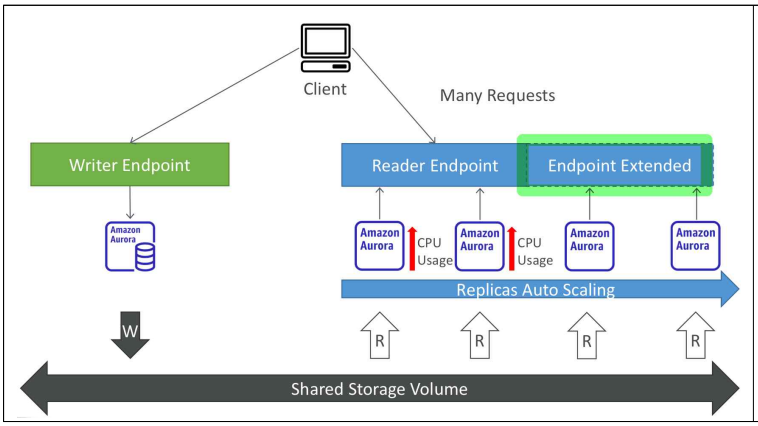
3) Aurora DB Cluster

• 클라이언트가 있을 때 수많은 인스턴스와 어떻게 접속하는지



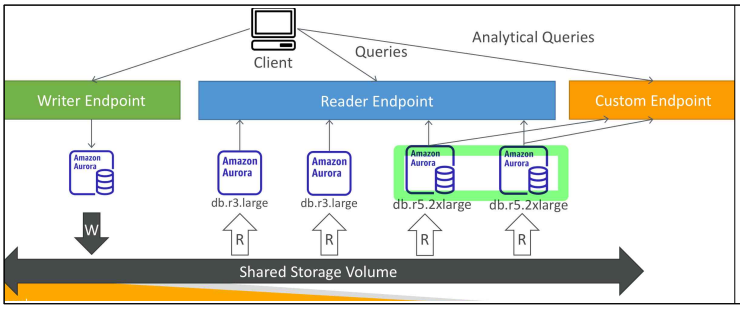
• 공유 스토리지 볼륨 ⇒ 자동확장
• Writer Endpoint ⇒ 변경 및 장애 조치,
항상 마스터 가리킴
• 읽기 전용 복제본 ⇒ 자동 스케일링,
15개까지 생성 가능
• Reader Endpoint
⇒ 클라이언트가 리더 엔드포인트 연결할 때마다
읽기 전용 복제본 中 하나에 연결(로드벨런싱)
(로드벨런싱⇒문장 레벨x, 연결 레벨에서 일어남)

4) Aurora Replicas - Auto Scaling (Aurora 복제본 자동 스케일링)



• Reader Endpoint에 요청 多
→자동 스케일링 설정해 Aurora 복제본 추가
→자동으로 Reader Endpoint 연장됨
→읽기가 좀 더 분산

5) Aurora Custom Endpoints (Aurora 사용자 지정 엔드포인트)

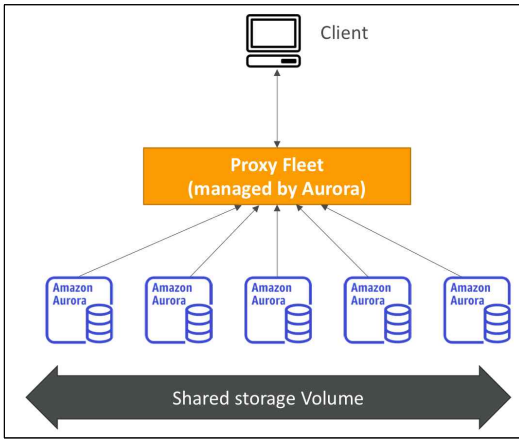


• 특정 오로라 인스턴스를 Custom Endpoint로 지정
• 특정 복제본에서 분석 쿼리 실행할 때 좋음
• Custom Endpoints로 지정 시
→Reader Endpoint 사용x

6) Aurora Serverless

-실제 사용을 기반한 자동 DB 인스턴스화 & 자동 스케일 가능
-비정기적, 간헐적, 예측할 수 x는 워크로드 있을 때 유용
-용량 계획x
-비용 효율(초당 지불)

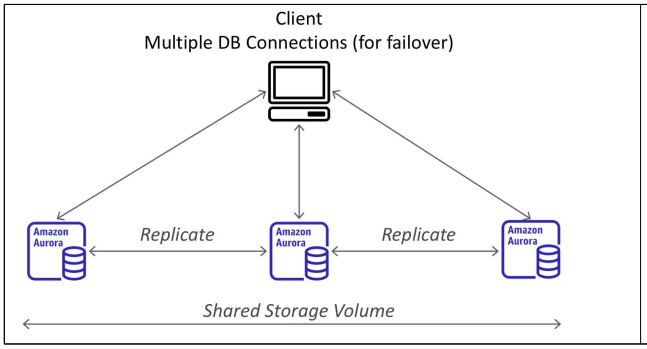
*serverless
:개발자가 서버 관리할 필요x



• 클라이언트 ⇒ 하위 Proxy Fleet과 소통
→백엔드에서 워크로드에 기반해 서버리스 방식으로
여러 오로라 인스턴스 생성
(∴용량 확보 필요x)

7) Aurora Multi-Master (Aurora 다중 마스터)

-writer 노드에 대한 즉각적인 장애 조치 원하는 경우 사용(writer 노드에 대한 HA 원할 때)

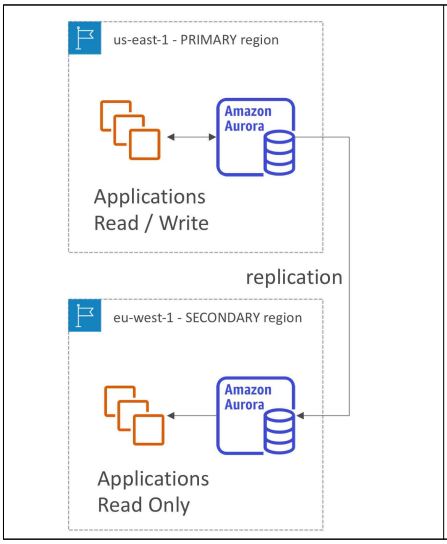


- all 인스턴스 간에 복제가 일어남
- all 인스턴스에서 쓰기 가능
- 특정 인스턴스 장애 → 장애 조치 통해 다른 인스턴스로 (라이터 노드에 대한 즉각적 장애 조치 가능)

8) Global Aurora

-Aurora 리전 간 읽기 전용 복제본 있을 때 재해 복구에도 유용하고 아주 간단히 구현 0

-최근 Aurora Global DB 만드는 것 권장



- all 읽기 및 쓰기가 발생하는 하나의 기본 리전 존재
- 복제 지연이 1s 미만인 보조 읽기 전용 리전을 5개까지 설정 가능
- 각 보조 지역마다 읽기 전용 복제본을 16개까지 생성 가능
- 한 리전 DB 장애 발생
  →복구 시간 목표(RTO) 1m 미만
- 리전에 걸쳐 데이터 복제⇒ 1s 미만

9) Aurora Machine Learning

-Aurora ⇒ ML 서비스와의 통합 지원

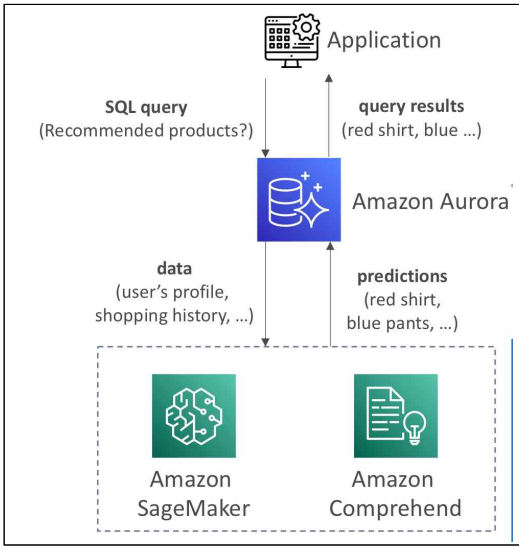
-ML 기반 예측을 SQL 인터페이스로 app에 적용

-지원하는 서비스

① Amazon SageMaker (ML 모델 구축하는 개발자 & 과학자 위한 서비스)

② Amazon Comprehend (자연어 처리 NLP(Natural Language Processing) 서비스)

-Use cases: 사기 탐지, 광고 타겟팅, 감정 분석, 제품 추천



- Aurora & ML 서비스 연결
  →app이 간단한 SQL 쿼리 실행(ex_추천 제품은 무엇인가?)
  →Aurora가 ML 서비스로 데이터 전송(ex_사용자 프로필)
  →ML 서비스는 Aurora로 직접 예측 반환(ex_빨간 셔츠와 파란 바지)
  →Aurora는 쿼리에 대한 결과를 app에 반환

3. RDS & Amazon Aurora

1) Backups

(1) RDS Backups

① *Automated Backup* (자동 백업)

-자동으로 매일 DB 유지 관리 시간에 DB 전체 백업

-5분마다 트랜잭션 로그도 백업됨(=가장 최신 백업이 5분 전, 5분 전 어떤 시점으로도 복구 가능)

-자동 백업 보유 기간: 1-35일 (비활성화 가능)

② *Manual DB Snapshots* (수동 DB 스냅샷 생성)

-사용자가 수동으로 트리거

-장점: 원하는 기간 동안 보유 가능

***비용 절감 방법**

-RDS DB를 매달 2시간만 사용할 예정이라면,
사용 후 스냅샷 만들고 원본 DB 삭제 → 스냅샷 복원
(∵DB 정지한다고 해도 스토리지 비용 지불해야 함)

(2) Aurora Backups

① *Automated Backup*

-1~35일 보유 가능

-비활성화 불가 (⇔RDS는 비활성화 가능)

-point-in-time recovery(지정시간 복구 기능): 정해진 시간 범위 내의 어느 시점으로도 복구 0

② *Manual DB Snapshots*

-사용자가 수동으로 트리거

-장점: 원하는 기간 동안 보유 가능

2) Restore options

• 스냅샷 or Amazon S3로 가능한 DB 복원 방법

(1) 복원 가능: RDS 및 Aurora 백업 or 스냅샷

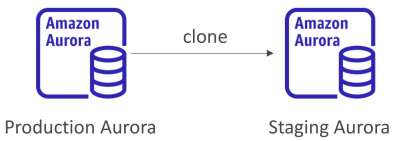
-자동 백업 or 수동 스냅샷 복원⇒ new DB 생거남

(2) S3로부터 MySQL RDS DB 복원

(3) S3로부터 MySQL Aurora 클러스터 복구

3) Aurora Database Cloning (Aurora DB 복제 기능)

-기존 DB로부터 new Aurora 클러스터 만들 수 0



• Production Aurora를 새로운 Aurora DB로 복제
ex_ Production Aurora에 영향 주지 x고,
Staging Aurora로 복제해 개발 or 테스트 수행 0

-스냅샷 만들고 복원하는 것보다 빠름

-비용 효율적

4) Security

(1) At-rest encrpytion (저장 데이터 암호화)

-RDS 및 Aurora DB에 저장된 데이터 암호화

-AWS KMS 사용해 마스터 & all 복제본 암호화 이루어짐, DB 처음 실행할 때 정의됨

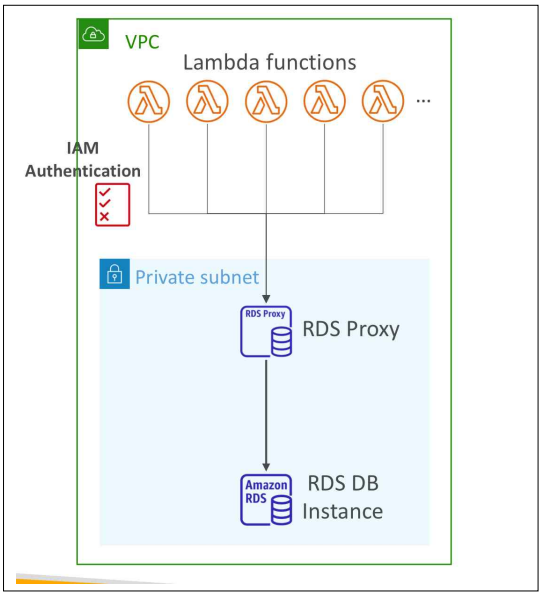
-주 DB 암호화 x → 읽기 전용 복제본 암호화 x

-암호화 x된 DB 암호화 시 ⇒ 암호화 x된 DB 스냅샷을 암호화된 DB 형태로 DB 스냅샷 복원해야 함

(2) In-flight encryption (전송 중 암호화)
-RDS 및 Aurora⇒ 기본적으로 전송 중 암호화 기능 갖춤
-AWS TLS 루트 인증서 사용
(3) IAM Authentication
ex_EC2 인스턴스에 IAM 역할이 있다면 사용자 이름 or 패스워드 x이 직접 DB 인증 0
(4) SG
-SG 사용해 DB에 대한 net 액세스 통제
(특정 포트, IP, SG 허용 or 차단)
(5) SSH 액세스 x
-RDS Custom은 예외
(6) Audit Logs (감사 로그)
-시간에 따라 RDS 및 Aurora에서 어떤 쿼리가 생성되는지 확인 시 활성화
-로그⇒ 시간 지나면 자동 삭제,
장기간 보관 원하면 CloudWatch Logs라는 전용 서비스로 연결

5) Amazon RDS Proxy

-app을 RDS DB 인스턴스에 일일이 연결하는 대신
프록시가 하나의 풀에 연결을 모음
→ RDS DB 인스턴스로 가는 연결 ⇓
-DB 리소스(CPU & RAM 등) 부담 ⇓ → DB 효율성 향상,
DB에 개방된 연결과 시간 초과 최소화
-서버리스, 오토 스케일, HA, Multi-AZ 지원
-RDS & Aurora 장애 조치 시간 ⇓
-코드 변경 x
-DB에 IAM 인증 강제,
자격 증명을 AWS Secrets Manager 서비스에 안전히 저장
-public access x(인터넷 통해 프록시 연결x, VPC 내에서만 액세스)



4. Amazon ElastiCache

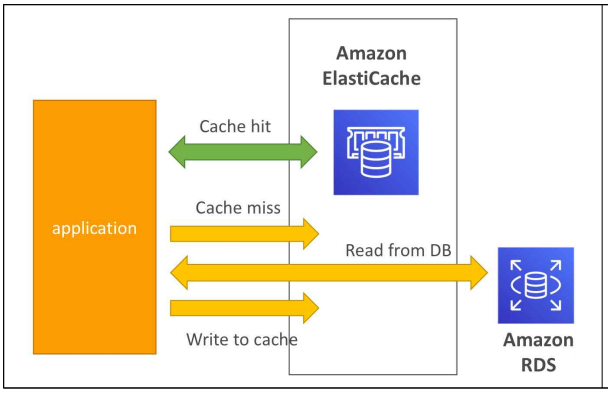
1) 개요

-Redis or Memcached 같은 캐시 기술 관리
-관계형 DB 관리
-읽기 집약적 워크로드 부하 ⇓
-일반적인 쿼리가 캐시되어 DB가 매번 쿼리 되지x (쿼리 결과 검색 시 사용)
-app에 관한 몇가지 어려운 코드 변경 요청 0

***캐시**
:높은 성능 & 낮은 지연시간 가진 인 메모리 DB

2) ElastiCache 사용을 위한 Architecture

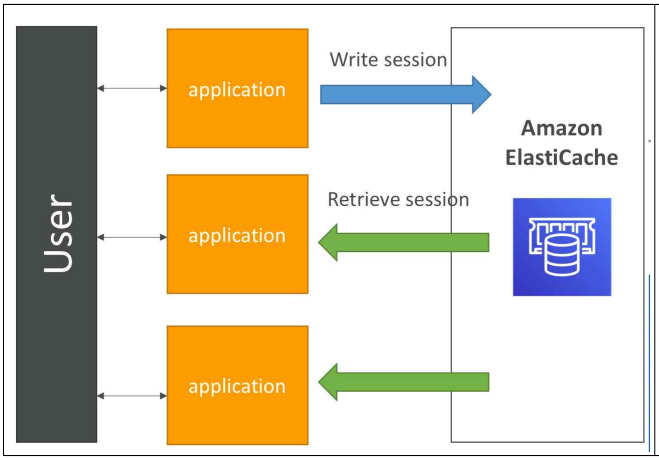
(1) DB Cache



- Cache miss
  ⇒DB에서 데이터 가져와 읽음,
  동일한 쿼리 발생 시 데이터를 캐시에 다시 기록
  (RDS DB에 부하 ⇓)
- Cache hit
  ⇒쿼리가 생성되어 ElastiCache에 저장됐는지 확인하는 것,
  ElastiCache에서 바로 응답 얻어 DB로 이동하는 동선⇓
- 캐시 무효화 전략 필요
  -가장 최근 데이터만 사용하는지 확인

(2) User Session Store

-사용자 세션 데이터를 ElastiCache에 저장해 app을 무상태로 만드는 것



• 사용자가 app의 all 계정에 로그인하면
app이 ElastiCache에 세션 데이터 기록
→사용자가 app의 다른 인스턴스로 리디렉션 되면
app은 ElastiCache에서 직접 세션 캐시를 검색
→so 사용자는 계속 로그인한 상태로
한 번 더 로그인할 필요x

3) Redis VS Memcached

Redis	Memcached
-HA -백업 & 복원 기능 0	-HA x -백업 & 복원 기능 x -데이터를 손실 할 수 있는 단순 분산 캐시

4) Cache Security

-all cache ⇒ IAM authentication 지원x
(AWS API 수준 보안에서만 사용(ex_캐시 생성, 삭제))

-Redis AUTH
:Redis 클러스터 생성 시 비밀번호 or 토큰 설정 (캐시에 들어갈 때 비번 사용),
전송 중 암호화 위해 SSL 보안 지원 0

-Memcached
:좀 더 높은 수준인 SASL 기반 인증 지원

*SSL
:Secure Sockets Layer(보안 소켓 계층),
웹사이트와 브라우저 사이(or 두 서버 사이)에 전송되는 데이터를 암호화하여 인터넷 연결 보호

5) Patterns for ElastiCache (캐싱 전략)

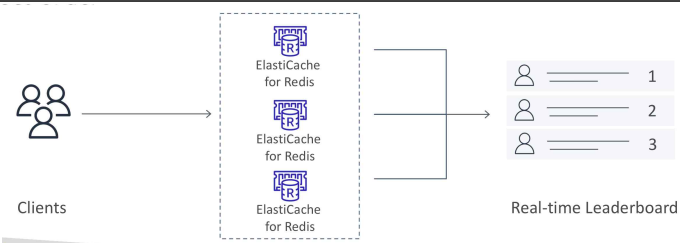
(1) Lazy Loading (지연 로딩)
-필요할 때만 데이터를 캐시에 로드
-장점: 요청한 데이터만 캐싱하므로 별도의 캐싱 작업 필요x
-단점: 최초 요청시 DB에 직접 쿼리 → 느림

(2) Write Through
-데이터 갱신 시 캐시 & DB all 업데이트
-장점: 데이터 갱신 끝나면 all 데이터 갱신됨 → 조회 시 항상 캐시를 읽어와 빠름
-단점: 데이터 갱신 시 DB, 캐시 ALL 업데이트하므로 느림

(3) Session Store
-세션 저장소로 사용0
-TTL 속성으로 세션 만료시킬 수0

*TTL (Time To Live)
:키가 만료될 때까지의 시간을 지정

6) Redis Use Case



• 게임 리더보드
• Redis로 ElastiCache와 상호작용할 때
클라이언트는 이 실시간 리더보드에 access 0,
app 측에서는 이 기능을 프로그래밍 할 필요x

5. 표준 포트 목록

중요한 포트:

- FTP: 21
- SSH: 22
- SFTP: 22 (SSH와 같음)
- HTTP: 80
- HTTPS: 443

vs RDS 데이터베이스 포트:

- PostgreSQL: 5432
- MySQL: 3306
- Oracle RDS: 1521
- MSSQL Server: 1433
- MariaDB: 3306 (MySQL과 같음)
- Aurora: 5432 (PostgreSQL와 호환될 경우) 또는 3306 (MySQL과 호환될 경우)