# RDS

# AWS RDS

- Amazon RDS (Relational Database Service)
- AWS에서 제공하는 완전 관리형 관계형 데이터베이스 서비스
- 데이터베이스 설치, 운영, 확장, 패치, 백업 등을 자동 관리
- 사용자는 애플리케이션 개발에 집중 가능

- Amazon RDS는 관계형 데이터베이스를 클라우드에서 손쉽게 운영할 수 있도록 해주는 서비스!
- 서버 설치나 백업, 패치 같은 번거로운 작업→ AWS가!
- 개발자는 비즈니스 로직에 집중!

# 기존 데이터베이스의 한계

- 온프레미스 DB: 직접 설치·운영, 확장성 한계, 장애 복구 부담
- 오픈소스 DB: 무료이지만 성능·안정성 제약, 전문 튜닝 필요
- 상용 DB: 안정적이지만 라이선스 비용 ↑, 벤더 종속성

- RDS 엔진 종류
  - MySQL – 오픈소스, 대중적
  - PostgreSQL – 확장성·표준 SQL 강점
  - MariaDB – MySQL 기반 오픈소스
  - Oracle – 안정성·기업용 기능 풍부
  - MS SQL Server – Windows 생태계와 강력 통합

# 왜 AWS RDS인가?

- 인프라 관리 부담 감소
- 자동화 기능: 백업, 모니터링, 패치
- 고가용성 제공 (Multi-AZ 배포)
- 필요 시 수 분 내 확장 가능

- 안정적으로 DB사용는 것은 비용부담 발생! → AWS와 같은 관리형 서비스 도입!

# WHY Aurora RDS인가?

- 완전 관리형 관계형 DB엔진
- MySQL·PostgreSQL 호환 + 성능 극대화 → MySQL 대비 5배, PostgreSQL 대비 3배 성능
- RDS 엔진 대비 고성능·저비용
- 클라우드 네이티브 아키텍처

# Aurora RDS 특징

- 기존 코드·도구 그대로 사용 가능
- 용량 자동 증감 : 10GB부터 시작, 10GB단위로 용량 증가 (최대 128TB)
- 고성능 분산 스토리지: 6중 복제, 내결함성 제공
  - 내결함성: Fault Tolerance, 시스템 일부에 장애(결함)가 발생해도 전체 서비스가 멈추지 않고 계속 동작하는 능력
  - 각AZ마다 2개의 데이터 복제본 저장 X 최소 3개 이상의 AZ = 6중 복제!
- Aurora는 데이터를 10GB단위의 세그먼트로 쪼갬

→ 1TB면, 약100개의 세그먼트로 나눔

- 이 세그먼트안에 다시 데이터 블록(block,MB)단위로 복제 저장

# Quorum(쿼럼) 모델 : 내결함성, 고가용성

- 데이터를 3개 AZ, 총 6개 복제본에 분산 저장
- 쓰기 작업: 최소 4개 복제본이 응답해야 완료 → 2개까지 손실 OK
- 읽기 작업: 최소 3개 복제본이 응답해야 완료 → 3개까지 손실 OK

→Quorum(쿼럼) 모델 : 다수결의 원리와 비슷하게 여러 복제본 중 일정 개수 이상이 응답하면 작업을 성공적으로 처리하는 방식

→Quorum: 의사 결정에 필요한 최소 인원, 합의를 위한 최소 응답수

- Self-healing : 손실된 복제본은 자가 치유, 지속적으로 손실된 부분을 검사 후 복구

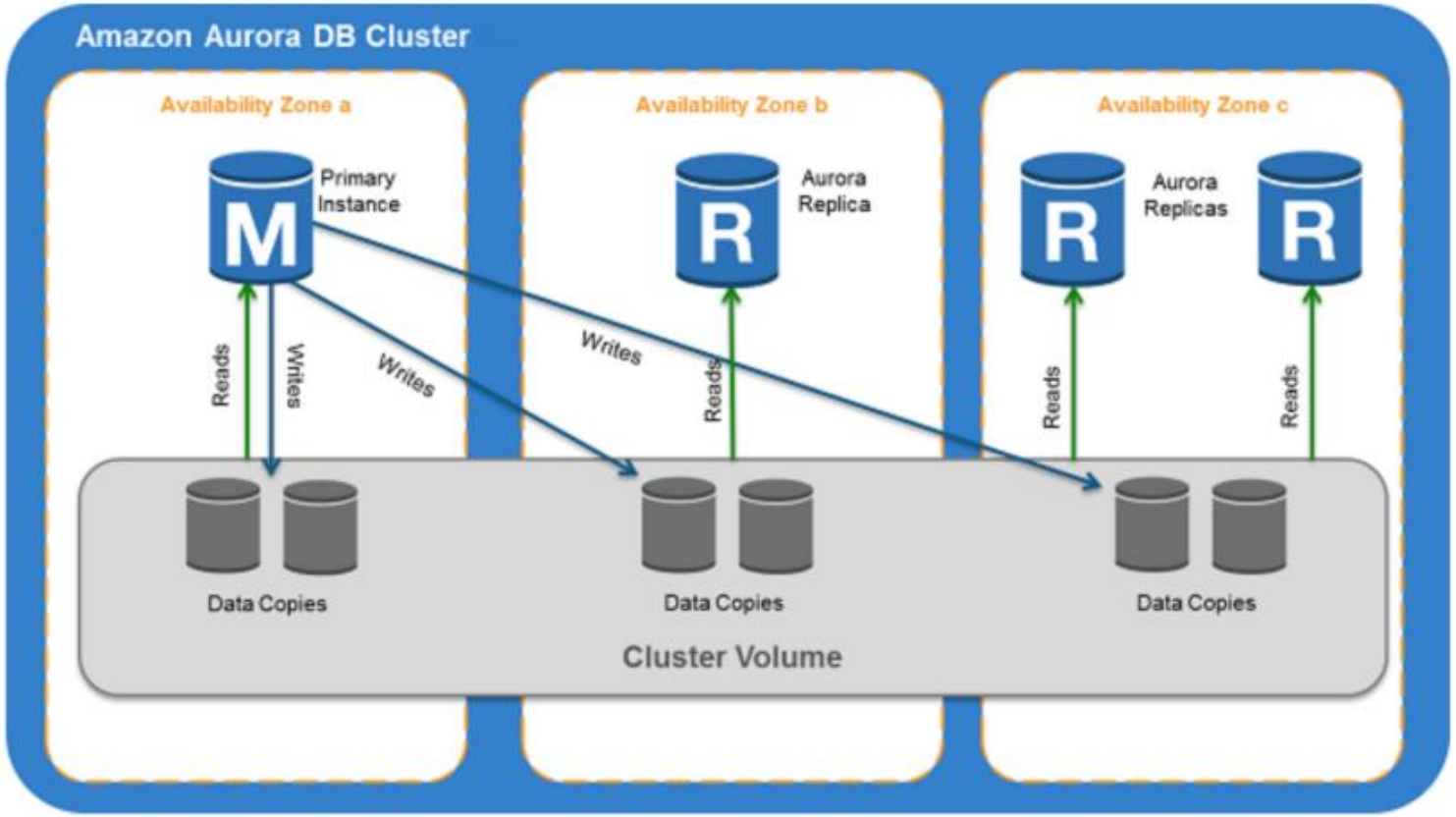
# Aurora의 두 가지 레벨

- 스토리지 레벨
  - 데이터가 저장되는 공간
  - Aurora는 6개 복제본을 유지
  - Quorum모델을 통해 내결함성과 고가용성 보장
- 컴퓨팅 레벨
  - 데이터를 읽고 쓰는 DB 인스턴스(Writer/Reader)
  - Single-Master: Writer 1개 + 여러 Reader
  - Multi-Master: 모든 인스턴스가 Writer 가능

# Single-Master

- 구성
  - Writer 인스턴스: 1개 (쓰기 전용)
  - Reader 인스턴스: 최대 15개 (읽기 전용 Replica)
- 동작방식
  - 쓰기 작업은 반드시 Writer에서만 수행
  - 읽기 작업은 Reader 인스턴스로 분산 처리
- 복제방식
  - Async 복제(비동기)로 Reader와 데이터 동기화
- 고가용성
  - Writer 장애 시 Reader 중 하나가 Writer로 자동 승격(Failover)
- 특징
  - 대부분의 워크로드(Workload, 처리해야 할 작업의 종류와 양)에 적합
  - 읽기 트래픽 확장에 강점

# Single-Master 아키텍처




https://jayendrapatil.com/aws-rds-aurora/?utm_source=chatgpt.com

# Aurora Global Database

- 전 세계 리전 간 데이터 공유
  - 1초 미만 지연 시간으로 다른 리전에서 읽기 가능
  - 다국적 서비스, 글로벌 사용자 대상 서비스에 최적화
- 재해 복구(Disaster Recovery) 용도 활용
  - Primary Region 장애 시, Secondary Region을 빠르게 승격(Failover) 가능
  - RPO(Recovery Point Objective, 복구 목표 시점): 1초 미만
  → 장애 시 데이터 손실 허용 범위.최대 1초치 데이터만 손실될 수 있다!
  - RTO(Recovery Time Objective, 복구 목표 시간): 1분 미만
  → 장애 발생 후 서비스가 복구되어 정상 운영까지 걸리는 시간
- 읽기 전용 확장성(Read Scalability)
  - 보조 리전(Secondary Region)마다 최대 16개의 Reader 인스턴스 생성 가능
  - 글로벌 서비스의 트래픽을 여러 리전에 분산 처리
- 제한 사항
  - Writer는 Primary Region에만 존재 (Single-Master 기반)
  - Region 간 복제는 비동기(Asynchronous) 복제 방식 → 약간의 지연 가능

# 병렬쿼리

- 온프레미스에서 Full Scan을 한다면? 많은 시간 소요!

- AWS Aurora : 스토리지에 분산 저장된 세그먼트를 스토리지 노트에서 직접 병렬 스캔
  - 빠름 : 수억 건 이상의 대규모 데이터를 처리할 때, 병렬적으로 분산 스캔 → 응답시간 단축
  - 부하 분산 : CPU·메모리를 쓰지 않고 스토리지 노드가 작업을 대신 수행.
  - OLAP(Online Analytical Processing)최적화 : 분석성 쿼리(집계, 리포트)에 매우 적합
  - Aurora MySQL 전용 (현재는 MySQL 5.6, 5.7 호환 버전에서만 제공).
  - 낮은 인스턴스 타입(t2, t3 계열)에서는 미지원.
  - 트랜잭션성 OLTP(Online Transaction Processing) 쿼리에는 큰 장점 없음 → 주로 OLAP/분석용.

# MySQL백업 (Binlog 기반)

- 데이터 변경(CRUD) → Binlog(Binary Log)에 기록
  예) INSERT INTO Orders VALUES (101, '상품A', 3);
  →Binlog에 INSERT 이벤트 기록됨
- Binlog는 SQL문 그대로 저장하는 게 아니라, 압축된 이진(Binary) 포맷으로 기록됨
- Replica DB는 Binlog를 읽고, 해당 이벤트를 재실행하여 Primary와 동기화
- 왜 이런식으로 할까?
  - 온프레미스는 스토리지를 별도로 구성하니깐!
  - 스토리지 통째 복제는 무겁고 포맷 의존적
  - Binlog 이벤트 복제는 가볍고 이식성 유연성 좋음!

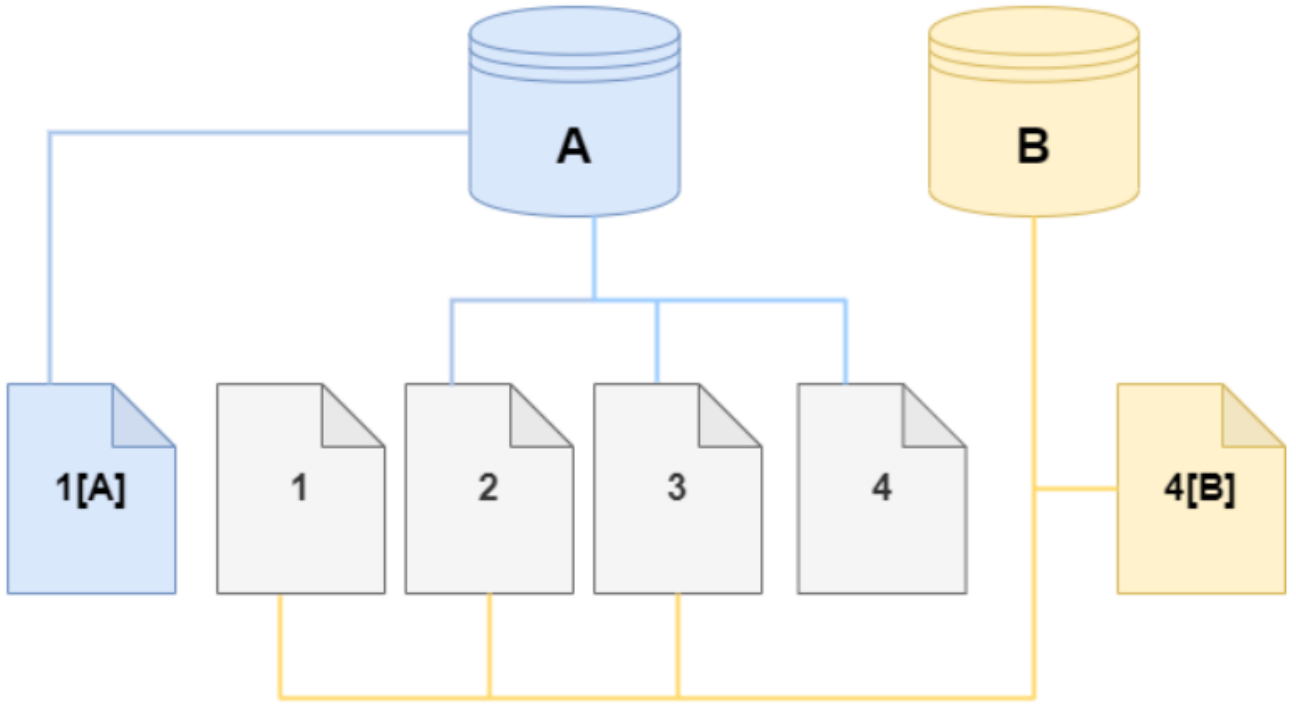
# Aurora 백업

- 스토리지 기반 백업
  - Aurora는 Binlog 대신 스토리지 레벨에서 자동 백업
  - 변경된 데이터 블록을 Amazon S3에 저장
  - PITR(Point-In-Time Recovery, 시점 복구) 지원
- 자동 백업
  - 기본 제공, 보관 기간 1~35일
  - Amazon S3에 저장 → 높은 내구성 (11 9’s)
- 수동 백업 (Snapshot)
  - 사용자가 특정 시점에 스냅샷 생성
  - 장기 보관 가능, 다른 리전으로 복사 가능
- 리전 간 백업
  - 스토리지는 리전 단위 리소스 → 다른 리전과 직접 공유 불가
  - Binlog 기반 Read Replica (MySQL 전통 복제 방식)
  - Aurora Global Database (Aurora 전용, 더 빠르고 효율적)

# Aurora Cloning

- 원리
  - 오리지널(Original) 클러스터를 그대로 두고
  - 클론(Clone) 클러스터를 새로 생성 → 하지만 데이터 전체를 복사하지 않음
  - 대신, 기존 스토리지 데이터를 공유
- Copy-on-Write 방식
  - 클론은 원본과 같은 스토리지를 바라봄
  - 새로운 데이터 변경이 발생하면?
    - 원본(Original)의 블록은 그대로 둠
    - 변경된 블록만 새로운 스토리지에 복사 후 기록
  - 결과: 공유 가능한 블록 = 그대로 사용 / 수정된 블록 = 분리 저장
- 장점
  - 속도: 수 테라급 DB도 몇 분 만에 클로닝 가능 (풀 카피 아님)
  - 비용 절감: 변경된 블록만 추가 저장 → 전체 용량 대비 매우 저렴
  - 유연성: 원본 클러스터를 지워도 클론은 독립적으로 동작

# Aurora Cloning




https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/Aurora.Managing.Clone.html?utm_source=chatgpt.com

# Aurora Backtrack

- 기존 DB를 특정 시점으로 되돌리는 기능
- 새로운 DB를 생성하지 않고, 기존 DB를 롤백
- 실수 복구에 최적화 (예: WHERE 없는 DELETE, 잘못된 UPDATE)
- MySQL 오픈소스에는 없는 기능 (Aurora MySQL 전용)
- Oracle DB의 Flashback Database와 유사
- 스냅샷 복구는 새 DB 생성 → 데이터 복원 → 느리고 복잡
- Backtrack은 클러스터 전체를 즉시 되돌림 → 빠르고 간편
- DBA 실수를 신속히 수정할 수 있음
- **PostgreSQL 버전에서는 지원하지 않음**

# Aurora Multi-Master

- Single-Master 한계
  - Writer는 1개뿐 → 쓰기 병목 발생
  - 장애 시 Failover 필요 → 수 초간 다운타임
- Multi-Master 필요성
  - 모든 인스턴스가 Writer 역할 수행
  - 무중단 고가용성 + 쓰기 확장성 확보

# Multi-Master

- 모든 Writer 인스턴스가 동일 Storage Volume 공유
- 모든 인스턴스가 읽기/쓰기 동시 처리 가능
- 최대 4개의 Writer 노드 지원 (현재 제한)
- 트랜잭션 충돌 시 Aurora가 조정 (첫 커밋 우선)
- 각 Writer 노드는 독립적 → 정지·재부팅·삭제가 다른 노드에 영향 없음
- Writer 장애 시 Failover 필요 없음 → 다른 Writer 즉시 처리
- 목표: 지속적인 가용성(Continuous Availability) 제공

# Multi-Master Best Practice

- Multi-tenant 서비스 (SaaS)
  - 고객사별 독립 데이터베이스 필요
  - 다운타임 없는 고가용성 중요
- Sharding 적용 대규모 애플리케이션
  - 사용자 수·데이터 크기 폭증 시 → 수평 확장 필수
  - Writer 노드를 나눠 샤드별 부하 분산 가능

- 즉! 모든 서비스에 필요한건 아니고! 극단적 확장성과 무중단이 필요한 경우 적합!

# Sharding(샤딩)이란?

- 큰 데이터베이스를 작은 단위(Shard)로 나눠 여러 DB 서버에 분산 저장하는 방식
- User ID 1~100만 → Shard A / User ID 100만~200만 → Shard B
- 특정 Shard에 부하 집중 시 Hotspot(특정 데이터 파티션에만 트래픽 집중되는 현상) 발생 단점 있음
- 학생이 너무 많아 한 반에서 수업이 불가능하면, 학번대로 1반,2반,3반으로 나누는 것 = Sharding

# Aurora Single-Master vs Multi-Master

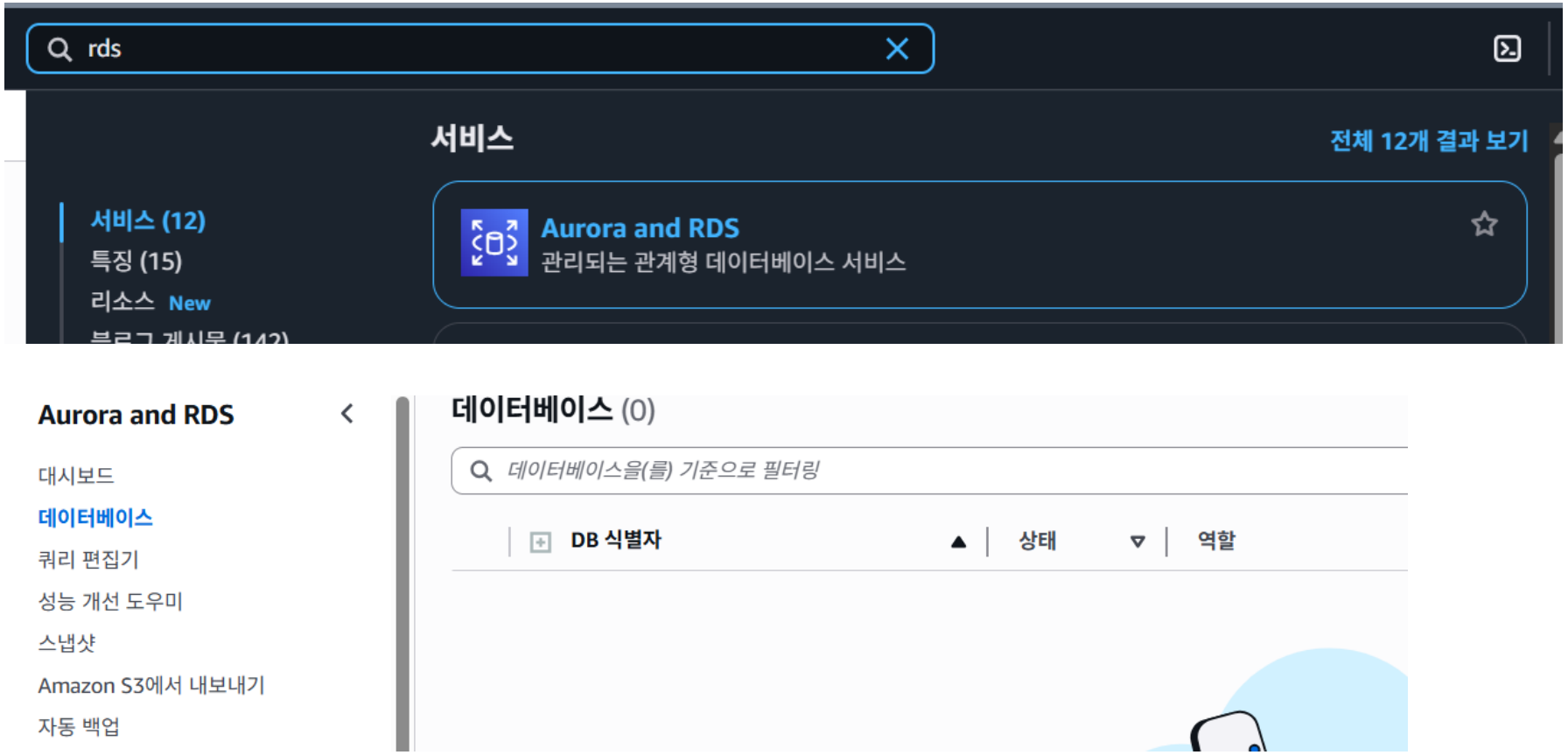
| 구분 | **Single-Master** | **Multi-Master** |
| --- | --- | --- |
| **Writer 개수** | 1개만 존재 | 최대 4개까지 가능 |
| **Failover** | Writer 장애 시 Replica를 승격 (수 초~수십 초 소요, 다운타임 발생) | 다른 Writer가 즉시 이어받음 (무중단) |
| **고가용성** | 읽기 고가용성은 Aurora Replica로 확보 가능, 쓰기 고가용성은 약함 | 읽기 + 쓰기 모두 고가용성 (Continuous Availability) |
| **적합한 환경** | 일반 웹/앱 서비스, 대부분의 SaaS 기본 구조 | 멀티테넌트 SaaS, Sharding 적용 대규모 애플리케이션, 금융/거래 시스템 |
| **확장성** | 읽기는 최대 15개 Aurora Replica로 확장 | 쓰기도 여러 Writer로 확장 가능 (수평적 확장) |
| **제한 사항** | Writer 하나가 병목이 될 수 있음 | Writer 간 동기화 및 충돌 관리 필요, 아키텍처 복잡 |

# Aurora Multi-Master 단점

- Aurora MySQL 전용
- Backtrack (시점 복원 기능), Parallel Query, Global Database 사용불가!
- 특정 라이터 인스턴스만 Binlog 활성화 가능: 모두다 안됨
- 클로닝도 일부 제약이 있음
- 아키텍쳐 복잡함! : 동일 Row에 대한 동시 Write 시 충돌 가능
  - 첫 번째 커밋 성공, 나머지는 에러 반환
- 충돌 처리 로직은 애플리케이션이 직접 고려해야 함

# 실습) Aurora DB를 생성해보자!

- AWS Console → Aurora and RDS → 데이터베이스



# Aurora RDS 생성

- 데이터베이스 생성 클릭
- 엔진 : Aurora (MySQL Compatible)
- 템플릿 : 개발/테스트
- DB클러스터 식별자 : 계정-rds 예)sgu-202500-rds
- 자격증명설정: 자체관리 체크
  - 마스터 사용자 이름 : admin, 마스터암호 입력 (잊어버리지않기!)
- 클러스터스토리지구성 : Aurora Standard
- 인스턴스 구성: 버스터블클래스 체크, db.t3.medium
- 퍼블릭엑세스 : 예
- 보안그룹 : 자신의 보안그룹
- Enhanced monitoring 활성화 : 체크 해제
- 로그 내보내기 : 에러로그 일반로그 체크

## 데이터베이스 생성 정보

### 데이터베이스 생성 방식 선택

표준 생성
가용성, 보안, 백업 및 유지 관리에 대한 옵션을 포함하여 모든 구성 옵션을 설정합니다.

### 엔진 옵션

**엔진 유형** 정보

Aurora (MySQL Compatible)

**엔진 버전**

Aurora MySQL 3.08.2 (compatible with MySQL 8.0.39) - 메이저 버전 8.0의 기본값

RDS 확장 지원 활성화 정보
Amazon RDS 확장 지원은 유료 오퍼링입니다. 이 옵션을 선택하면 해당 버전의 RDS 표준 지원 종료일 이후에 데이터베이스 메이저 버전을 실행하는 경우 오퍼링의 요금이 청구되는 데 동의하는
확인하세요.

### 템플릿

해당 사용 사례를 충족하는 샘플 템플릿을 선택하세요.

프로덕션
고가용성 및 빠르고 일관된 성능을 위해 기본값을 사용하세요.

개발/테스트
이 인스턴스는 프로덕션 환경 외부에서 개

### 설정

**DB 클러스터 식별자** 정보

DB 클러스터 이름을 입력합니다. 이름은 현재 AWS 리전에서 AWS 계정이 소유하는 모든 DB 클러스터에 대해 고유해야 합니다.

sgu-202500-rds

DB 클러스터 식별자는 대소문자를 구분하지 않지만 모두 소문자로 저장됩니다(예: "mydbcluster"). 제약 조건: 1~63자의 영숫자 또는 하이픈으로 구성되어야 합니다. 첫 번째 문자는 글자여야 합니


▼ 자격 증명 설정
마스터 사용자 이름 정보
DB 인스턴스의 마스터 사용자에 로그인 ID를 입력하세요.
admin
1~32자의 영숫자. 첫 번째 문자는 글자여야 합니다.
자격 증명 관리
AWS Secrets Manager를 사용하거나 마스터 사용자 자격 증명을 관리할 수 있습니다.
AWS Secrets Manager에서 관리 - 가장 뛰어난 안정성
RDS는 자동으로 암호를 생성하고 AWS Secrets Manager를 사용하여 전체 수명 주기 동안 암호를 관리합니다.
자체 관리
사용자가 암호를 생성하거나 RDS에서 암호를
암호 자동 생성
Amazon RDS에서 자동으로 암호를 생성하거나 사용자가 직접 암호를 지정할
마스터 암호 | 정보
암호 강도 강함
최소 제약 조건: 8자 이상의 인쇄 가능한 ASCII 문자를 사용합니다. / ' " @ 기호는 포
마스터 암호 확인 | 정보
클러스터 스토리지 구성 정보
애플리케이션의 요금 예측 가능성 및 요금 성능 요구 사항에 가장 적합한 Aurora DB 클러스터의 스토리지 구성을 선택하세요.
구성 옵션
데이터베이스 인스턴스, 스토리지 및 I/O 요금은 구성에 따라 달라집니다.자세히 알아보기
Aurora I/O 최적화
모든 애플리케이션에 대해 예측 가능한 가격. I/O 집약적 애플리케이션의 가격 대비 성능이 향상되었습니다(I/O 요금이 전체 데이터베이스 요금의 25% 이상).
읽기/쓰기 I/O 작업에 대한 추가 비용은 없습니다. DB 인스턴스 및 스토리지 가격에는 I/O 사용량이 포함됩니다.
Aurora Standard
I/O 사용량(I/O 비용)이 보통인 많은 애플리케이션에 대한
요청당 지불 I/O 요금이 적용됩니다. DB 인스턴스 및 스토리
인스턴스 구성
아래의 DB 인스턴스 구성 옵션은 위에서 선택한 엔진에서 지원하는 옵션으로 제한됩니다.
DB 인스턴스 클래스 | 정보
▼ 필터 숨기기
이전 세대 클래스 포함
서버리스 v2
메모리 최적화 클래스(r 클래스 포함)
버스터블 클래스(t 클래스 포함)
db.t3.medium
2 vCPUs 4 GiB RAM 네트워크: 최대 2,085Mbps

## 가용성 및 내구성

**다중 AZ 배포** 정보

- ○ 다른 AZ에 Aurora 복제본/리더 노드 생성(확장된 가용성에 권장)
  신속한 장애 조치 및 고가용성을 위해 Aurora 복제본 생성
- ◉ Aurora 복제본 생성하지 않음

## 연결 정보

**컴퓨팅 리소스**

이 데이터베이스의 컴퓨팅 리소스에 대한 연결을 설정할지를 선택합니다. 연결을 설정하면 컴퓨팅 리소스가 이 데이터베이스에 연결할 수 있도록 연결 설정이 자동으로 변경됩니다.

- ◉ EC2 컴퓨팅 리소스에 연결 안 함
  이 데이터베이스의 컴퓨팅 리소스에 대한 연결을 설정하지 않습니다. 나중에 컴퓨팅 리소스에 대한 연결을 수동으로 설정할 수 있습니다.
- ○ EC2 컴퓨팅 리소스에 연결
  이 데이터베이스의 EC2 컴퓨팅 리소스에 대한

**네트워크 유형** 정보

듀얼 스택 모드를 사용하려면 IPv6 CIDR 블록을 지정한 VPC의 서브넷과 연결해야 합니다.

- ◉ IPv4
  리소스는 IPv4 주소 지정 프로토콜을 통해서만 통신할 수 있습니다.

**Virtual Private Cloud(VPC)** 정보

VPC를 선택합니다. VPC는 이 DB 클라우드의 가상 네트워킹 환경을 정의합니다.

Default VPC (vpc-0352e69bb13e04b28)
4 서브넷, 4 가용 영역

해당 DB 서브넷 그룹이 있는 VPC만 나열됩니다.

**DB 서브넷 그룹** 정보

DB 서브넷 그룹을 선택합니다. DB 서브넷 그룹은 선택한 VPC에서 DB 클러스터가 어떤 서브넷과 IP 범위를 사용할 수 있는지를 정의합니다.

default-vpc-0352e69bb13e04b28
4 서브넷, 4 가용 영역

**퍼블릭 액세스** 정보

- ◉ 예
  RDS는 클러스터에 퍼블릭 IP 주소를 할당합니다. VPC 외부의 Amazon EC2 인스턴스 및 다른 리소스가 클러스터에 연결할 수 있습니다. VPC 내부의 리소스도 클러스터에 연결할 수 있습니다. 클러스
- ○ 아니요
  RDS는 클러스터에 퍼블릭 IP 주소를 할당하지 않습니다. VPC 내부의 Amazon EC2 인스턴스 및 다른 리소스만 클러스터에 연결할 수 있습니다. 클러스터에 연결할 수 있는 리소스를 지정하는 VPC 보

**VPC 보안 그룹(방화벽)** 정보

데이터베이스에 대한 액세스를 허용할 VPC 보안 그룹을 하나 이상 선택합니다. 보안 그룹 규칙이 적절한 수신 트래픽을 허용하는지 확인합니다.

- ◉ 기존 항목 선택
  기존 VPC 보안 그룹 선택
- ○ 새로 생성
  새 VPC 보안 그룹 생성

**기존 VPC 보안 그룹**

*하나 이상의 옵션 선택* ▼

sgu-202500-sg ✕

**가용 영역** 정보

기본 설정 없음 ▼

## 모니터링 정보

이 데이터베이스의 모니터링 도구를 선택합니다. Database Insights는 데이터베이스 플릿에 대한 Performance Insights 및 Enhanced Monitoring의 통합 보기를 제공
CloudWatch 요금을(를) 참조하세요.

Database Insights - 고급

- 15개월의 성능 기록 유지
- 플릿 수준 모니터링
- CloudWatch Application Signals와 통합

(●) Database Insights - 표준

---

### ▼ 추가 모니터링 설정

Enhanced Monitoring, CloudWatch Logs 및 DevOps Guru

#### 향상된 모니터링

☐ Enhanced monitoring 활성화

향상된 모니터링 지표를 활성화하면 다른 프로세스 또는 스레드에서 CPU를 사용하는 방법을 확인하려는 경우에 유용합니다.

#### 로그 내보내기

Amazon CloudWatch Logs로 게시할 로그 유형 선택

☐ 감사 로그
☑ 에러 로그
☑ 일반 로그
☐ iam-db-auth-error 로그
☐ instance 로그
☐ 느린 쿼리 로그

# Aurora RDS 생성 확인

- 클러스터와 라이터 인스턴스 생성 확인!