



AWS 이론 8

- Amazon RDS

Sung-Dong Kim,
School of Computer Engineering,
Hansung University

RDMS (1)

- Relational Database Management System
- 데이터 간 사전에 정의된 **relation**이 있고, 연관 관계가 있는 데이터 항목들의 모음
- 열/행 → table
- primary key, foreign key
- SQL query

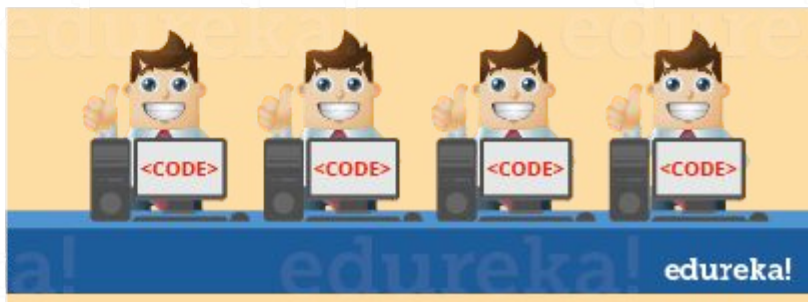
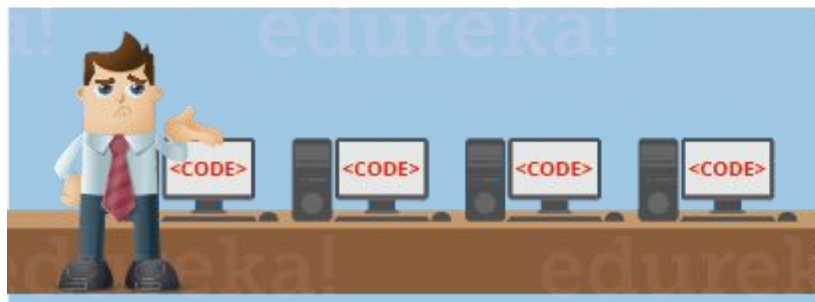
RDMS (2)

- Oracle, MySQL, MS SQL Server, PostgreSQL, MariaDB, ...
- AWS RDS (Relational Database Services)
 - Oracle
 - MySQL
 - MSSQL
 - Amazon Aurora

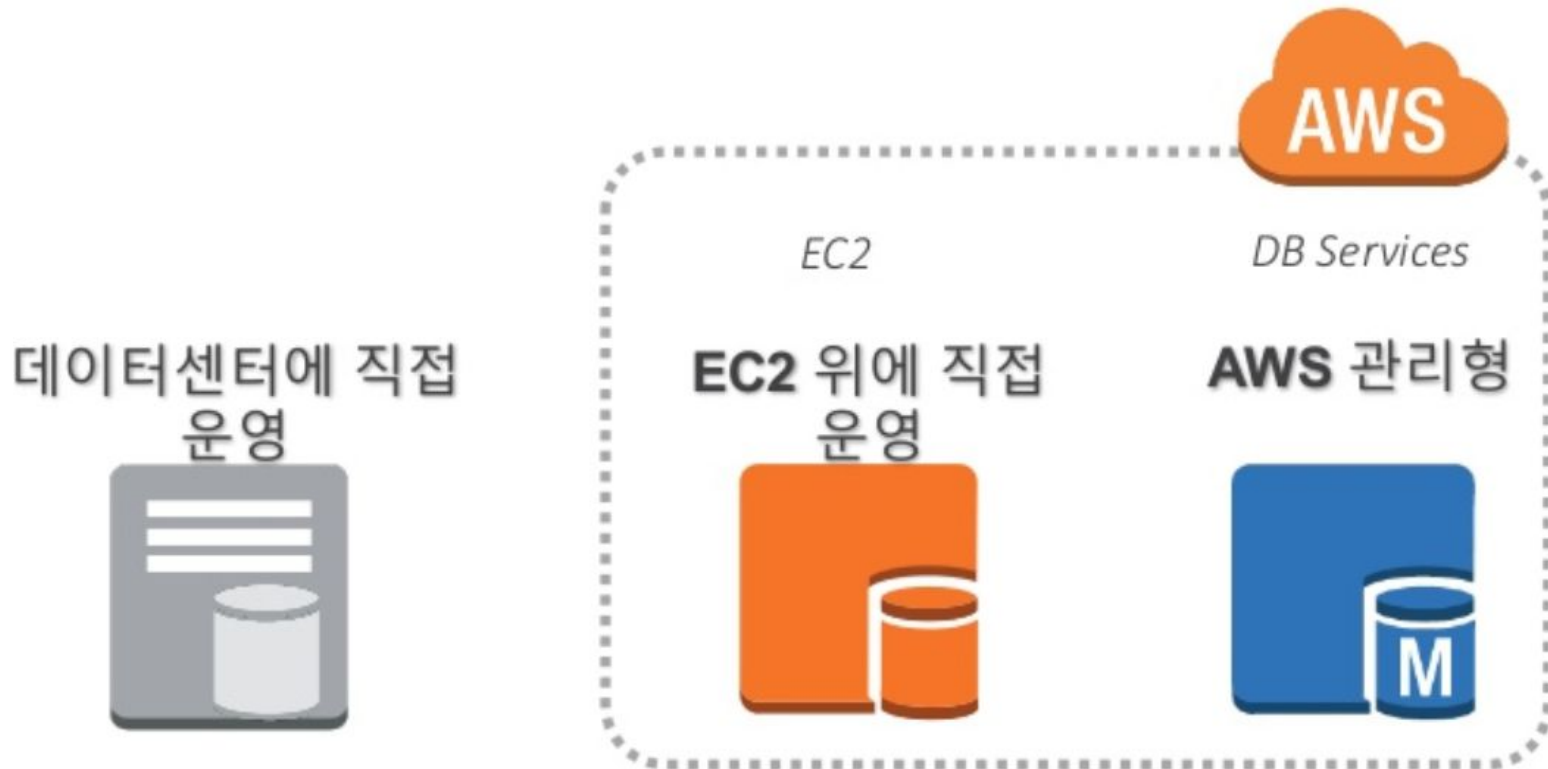
AWS RDS

- 클라우드에서 관계형 데이터베이스를 설정, 운영, 확장할 수 있는 서비스





AWS 데이터베이스 선택 (1)



AWS 데이터베이스 선택 (2)

직접 데이터베이스를 운영한다면?

App optimization

Scaling

High availability

Database backups

DB s/w patches

DB s/w installs

OS patches

OS installation

Server maintenance

Rack and stack

Power, HVAC, net



AWS 데이터베이스 선택 (3)

직접 Amazon EC2 에 데이터베이스를 운영한다면?

App optimization

Scaling

High availability

Database backups

DB s/w patches

DB s/w installs

OS patches



you

OS installation

Server maintenance

Rack and stack

Power, HVAC, net



AWS 데이터베이스 선택 (4)

AWS 관리형 데이터베이스를 선택한다면?



- Scaling
- High availability
- Database backups
- DB s/w patches
- DB s/w installs
- OS patches
- OS installation
- Server maintenance
- Rack and stack
- Power, HVAC, net

AWS 데이터베이스 선택 (5)

- 직접 운영: 하드웨어, 운영체제 등 관리, 직접 제어 가능
- EC2에 설치/운영: DB update, patch, backup 관리
- AWS 관리형 데이터베이스: AWS에서 모든 것을 자동 제공

AWS RDS 주요 특징

- 유연한 instance 및 storage 확장
 - 다양한 CPU/memory 옵션 제공
 - CloudWatch와 연계 → 트래픽에 따른 증설, 축소
- 쉽게 사용 가능한 백업 및 복원 기능: 자동 백업 설정
- 멀티 AZ를 통한 고가용성
- RDS 암호화: 보안성 강화
- Database migration 서비스

RDS (relational database service) (1)

- 관계 데이터베이스 서버의 구축/운용에 관한 서비스
- 쉬운 관리: 관리 콘솔, CLI, API 이용
- 가용성 및 내구성
 - 여러 AZ에서 DB instance 프로비저닝: instance 동기식 복제
 - 자동 백업, 스냅샷
- 보안: 네트워크 액세스 조정
- 쉬운 DB setup, 확장, 유지

RDS (relational database service) (2)

- 사용 사례
 - 웹 및 모바일 애플리케이션: 높은 처리량, 확장성, 가용성 만족
 - 모바일 및 온라인 게임
 - 전자 상거래 애플리케이션: 유연, 안전, 확장 가능성 제공
- 다양한 DB 엔진 제공

☒ Amazon Aurora

Amazon
Aurora

☐ MySQL



☐ MariaDB



☐ PostgreSQL



☐ Oracle

ORACLE®

☐ Microsoft SQL Server



DB instance

- Amazon RDS의 기본 빌딩 블록
- 클라우드에 있는 격리된 데이터베이스 환경
- 관리 콘솔, API, CLI를 이용해 생성, 수정, 관리
- 각 instance는 DB engine 실행
- VPC 서비스를 이용해 사설 클라우드에서 DB instance 실행

보안

- DB instance에 대한 access 제어
- DB 보안 그룹
 - VPC 외부의 DB instance에 대한 access 제어
- VPC 보안 그룹
 - VPC 내부의 DB instance에 대한 access 제어
- EC2 보안 그룹
 - EC2 instance에 대한 access 제어

모니터링

- DB instance의 성능, 상태 추적
- CloudWatch service 이용

AWS documentation

- https://docs.aws.amazon.com/ko_kr/AmazonRDS/latest/UserGuide/Welcome.html



AWS 실습

- RDS 1

Sung-Dong Kim,
School of Computer Engineering,
Hansung University

Create and Connect to MySQL DB

- What to do
 - 환경 (instance) 생성: MySQL DB를 실행하기 위한 환경
 - DB에 연결
 - DB instance 삭제

1단계: MySQL DB instance 만들기 (1)

- region 선택 (region마다 비용이 다름)
- 데이터베이스 생성 방식: 표준 생성/손쉬운 생성
- 엔진 선택: MySQL
- 템플릿: 개발/테스트, 프리 티어 - MySQL
- DB 인스턴스 크기
- 스토리지: 자동 조정 활성화

Amazon RDS



Dashboard

데이터베이스

성능 개선 도우미

스냅샷

Automated backups



Amazon Aurora

Amazon Aurora는 MySQL 및 PostgreSQL 호환 인
는 최대 64TB의 Auto Scaling 스토리지 용량, 3개
전용 복제본을 지원합니다. [자세히 알아보기](#)

데이터베이스 생성

또는, [S3에서 Aurora DB 클러스터 복원](#)

Amazon RDS



Dashboard

데이터베이스

성능 개선 도우미

스냅샷

Automated backups

예약 인스턴스

Proxies

서브넷 그룹

RDS > 데이터베이스

데이터베이스

☒ 그룹 리소스



수정

작업 ▼

S3에서 복원

데이터베이스 생성

🔍 데이터베이스 필터



DB 식별자



역할 ▼

엔진 ▼

리전 및 AZ ▼

인스턴스(를) 찾을

1단계: MySQL DB instance 만들기 (2)

- 가용성 및 내구성
- 연결: VPC
- 퍼블릭 액세스 - 허용
- 인증: 암호 인증
- 추가 구성

2단계: SQL client 다운로드

- 표준 SQL client를 이용하여 DB instance에 연결
 - MySQL Workbench
 - DB instance 생성한 컴퓨터에서 사용할 것 다운로드
 - <https://dev.mysql.com/downloads/workbench/>

3단계: MySQL 데이터베이스에 연결

- MySQL Workbench 시작
 - Database > Connect to Database
 - hostname: 관리 콘솔 → RDS → instance → 연결 & 보안:
엔드포인트

연결 & 보안

모니터링

연결 & 보안

엔드포인트 및 포트

엔드포인트




database-1.csofecurlmhw.ap-
northeast-2.rds.amazonaws.com

포트

3306

SCHEMAS



- ▶  innodb
- ▶  ksd_practice_db1
- ▶  sys

4단계: DB instance 삭제

- 사용하지 않는 instance 삭제 → 비용 부과하지 않도록 하는 것이 모범 사례
- 관리 콘솔 → RDS → instance
 - instance 선택 후 “삭제(delete)” 작업 수행

Source



- <https://aws.amazon.com/getting-started/tutorials/create-mysql-db/>

AWS Practice - Auto Scaling

Sung-Dong Kim,
School of Computer Engineering,
Hansung University

What to study

- ④ **EC2 Auto Scaling 시작하기**
- ④ **Launch Template 생성**
- ④ **Auto Scaling Group 생성/확인**
- ④ **Scaling Auto Scaling Group**



EC2 Auto Scaling 시작하기

☑ Auto Scaling Group 생성 전에,

- 몇개의 AZ?
- 어떤 resource를 이용할 것인지? - 보안 그룹, AMI, ...
- 용량 확장/축소 ? 일정한 개수의 서버를 항상 실행시킬 것인지?
- 애플리케이션의 성능에 관계된 지표는?



☑ 요약

- **launch template** 또는 launch configuration: EC2 instance를 정의하는 템플릿
- 하나의 instance를 가지는 Auto Scaling Group 생성
- 인스턴스를 종료시킨 후, 새로운 인스턴스가 생성되어 대체하는가를 확인
- 비용이 청구될 수 있음: free tier 기간 내이면 비용이 없음 (허용되는 인스턴스 이용시)
- default VPC를 이용 (처음 연습할 때)



Launch Template 생성

☑ 시작 템플릿 (launch template)

- EC2 인스턴스 → 시작 템플릿
- 리전 확인 (서울 - default VPC가 있어야 함)
- **[create launch template] (시작 템플릿 생성)**
 - 이름: ksd-templte-for-AS
 - Auto Scaling 지침 (guidance) → check
 - AMI: Amazon Linux 2 HVM
 - 인스턴스 유형: t2.micro
 - 키 페어: 자신의 것 선택
 - 보안 그룹: 사용하는 VPC의 보안 그룹 (default VPC)
 - **[시작 템플릿 생성]**

Auto Scaling Group

생성/확인

☑ EC2 대시보드 → Auto Scaling → Auto Scaling 그룹

☑ [Auto Scaling 그룹 생성]

- 이름: ksd-AS-group
- 시작 템플릿: ksd-template-for-AS (이전에 만든 것)
- VPC 확인: default
- 서브넷 선택: 여러 개 선택 가능 → 2개만 선택 (연습 - a, c)
- 고급 옵션 구성: 로드 밸런서 선택하지 않음, 모니터링 선택
- 그룹 크기: 크기만 1로 (추가로 2, 1, 3 도 연습)
- 조정 정책: 평균 CPU 사용률 50%
- 알림 추가: 수신 e-mail 등록
- 검토 후, [Auto Scaling Group 생성]

✓ Auto Scaling Group 확인

- 세부 정보
- 활동
- 인스턴스 관리

```
<html>
<head>
  <title>
  <meta name="description"
  <meta name="keywords"
  <style type="text/css"
html,body,div,h1,h2,h3
padding:0;
font-weight:100;
body {
background: #f0f0f0
font-size: 18px
color: #555
font-weight: 100;
```

Scaling Auto Scaling Group

Scaling Auto Scaling Group

- ④ EC2 대시보드 → Auto Scaling → Auto Scaling 그룹
- ④ Auto Scaling 그룹 생성 후, 목표 수의 instance를 시작함
- ④ 실행 중인 instance에 대해 주기적으로 상태를 확인함
- ④ instance가 비정상 상태이면 종료시키고 새로운 instance를 시작함
- ④ EC2 monitoring: CPU 사용률, 네트워크 사용률, 디스크 성능, 디스크 읽기/쓰기, ...



④ 인스턴스 1개 강제 종료 → 새로운 instance 자동 생성 확인

④ EC2 대시보드 → Auto Scaling → Auto Scaling 그룹

- 그룹 선택 → 세부 정보 → [편집]
 - 최소/최대 인스턴스 개수 조정: 2 → 1
 - instance 개수 확인
- 인스턴스 분리: 그룹 선택 → 인스턴스 관리
 - 인스턴스 선택 → 작업 → [분리]
- Auto Scaling 그룹에 인스턴스 추가
 - running 상태인 인스턴스 선택 → 작업 → instance 설정 → Auto Scaling 그룹에 추가



- ④ 조건: 두개의 instanc에서 실행되는 web applicatio이 있고, AS group의 CPU 사용률이 50% 정도를 유지하기 바람
- ④ 조정 정책을 생성하여 조건 충족
 - scale out: high demand peak 처리를 위해
 - scale in: low utilization 시기의 비용 절감을 위해
 - CloudWatch 지표 추적 → 알람
- ④ 조정 정책 유형
 - 대상 추적 조정 (target tracking scaling)
 - 단계 조정 (step scaling)
 - 단순 조정 (simple scaling)



☑ 대상 추적 조정

- 지표 선택
 - ASGAverageCPUUtilization - %
 - ASGAverageNetworkIn - # of bytes
 - ASGAverageNetworkOut - # of bytes
 - ALBRequestCounterTarget - # of requests
- monitoring - 1 min
- Auto Scaling 그룹 → 자동 크기 조정



☑ Auto Scaling Group 삭제 → EC2 instance 자동 삭제됨



✓ <https://docs.aws.amazon.com/autoscaling/ec2/userguide/what-is-amazon-ec2-auto-scaling.html>

AWS Practice – RDS 2

Sung-Dong Kim,
School of Computer Engineering,
Hansung University

Contents

- ▶ 사전 준비
- ▶ PHP가 있는 Apache web server 생성
 - ▶ Amazon Linux
 - ▶ EC2 instance
- ▶ MySQL DB 생성 및 연결
 - ▶ RDS MySQL instance
- ▶ 자원 삭제

사전 준비

사전 준비

- ▶ public/private subnet, 보안 그룹이 있는 VPC
 - ▶ private/public subnet을 포함하는 VPC 생성
 - ▶ 추가 subnet 생성
 - ▶ public Web server에 대해 VPC 보안 그룹 생성
 - ▶ private RDS DB instance에 대한 VPC 보안 그룹 생성
 - ▶ DB subnet group 만들기

사전 준비

- ▶ web server, DB instance가 동일한 VPC에서 실행됨
- ▶ web server에서만 DB instance 접속
 - ▶ private, public subnet이 모두 있는 VPC 생성
 - ▶ public subnet에서 web server 호스팅
 - ▶ private subnet에서 DB instance 호스팅
 - ▶ 보안 강화



Region



VPC

Availability Zone A



Private subnet

Security group
(rds-ec2-x)

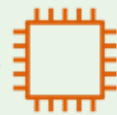


Amazon RDS DB instance



Public subnet

Security group
(ec2-rds-x)



Amazon EC2 instance
with web application

DB subnet group

Internet gateway



IPv4



Availability Zone B



Available private subnet



Available public subnet

사전 준비

- ▶ region: us-east (n. virginia) (DB 비용 고려)
- ▶ VPC 생성 - VPC만
 - ▶ 이름 태그 - ksd-db-vpc
 - ▶ CIDR: 10.0.0.0/16
- ▶ 서브넷 - 4개 (2개 public, 2개 private)
 - ▶ 가용 영역 = 2 (public = us-east-1a, private = us-east-1b)
 - ▶ CIDR: 10.0.1.0/24, 10.0.2.0/24, 10.0.3.0/24, 10.0.4.0/24

사전 준비

- ▶ 인터넷 게이트웨이 생성/연결
- ▶ NAT 게이트웨이 생성
- ▶ 라우팅 테이블
 - ▶ public subnet 연결 및 라우팅 편집
 - ▶ private subnet 연결 및 라우팅 편집

사전 준비

▶ 보안 그룹 생성

- ▶ web server 보안 그룹: ssh, http, https

- ▶ db server 보안 그룹

 - ▶ MYSQL/Autora

 - ▶ 소스: web server 보안 그룹 id 선택

사전 준비

▶ DB subnet 그룹 생성

- ▶ RDS → 서브넷 그룹 → [DB 서브넷 그룹 생성]
- ▶ 이름, 설명, VPC 선택
- ▶ 서브넷 추가
 - ▶ 2개 가용영역
 - ▶ 2개 private subnet의 CIDR

Web Server 만들기

Web Server 만들기

- ▶ AWS-Practice-LAMPWebServer 내용 참고
- ▶ VPC 선택
- ▶ public subnet 선택
- ▶ web server 보안 그룹 선택
- ▶ 자동 생성 public IP 활성화 or EIP 할당 및 연결

Web Server 만들기

- ▶ web server 설치 후,
- ▶ /var/www 아래 html의 소유권 변경
- ▶ /var/www 아래 inc 폴더 생성 및 소유권 변경
- ▶ /var/www/inc 폴더에 dbinfo.inc 파일 작성

```
<?php
define('DB_SERVER', 'endpoint');
define('DB_USERNAME', 'tutorial_user');
define('DB_PASSWORD', 'master password');
define('DB_DATABASE', 'sample');
?>
```

RDS instance 생성

RDS instance 생성

- ▶ AWS-Practice-RDS1 참고
- ▶ 표준 생성
- ▶ 연결
 - ▶ EC2 컴퓨팅 리소스에 연결 안 함
 - ▶ VPC 선택
 - ▶ 퍼블릭 액세스: 아니요
 - ▶ 보안 그룹: DB 보안 그룹
 - ▶ 가용 영역: us-east1-a1 (ksd-private-sbnet1)
- ▶ 추가 구성: 초기 데이터베이스 이름 - sample

RDS instance에 연결

RDS instance에 연결

- ▶ Web Server에서 /var/www/html → SamplePage.php 작성
 - ▶ https://drive.google.com/open?id=1CXfS9CQ1AyrZo1IWZTx-qoK-ZaFvc9m9ygF7MbRF_cA
- ▶ dbinfo.php 파일 수정
 - ▶ DB_SERVER → database instance 엔드 포인트
 - ▶ DB_USERNAME → admin
 - ▶ DB_PASSWORD
 - ▶ DB_DATABASE → sample

RDS instance에 연결

- ▶ 연결 확인 – **EC2 public IP**/SamplePage.php

자원 삭제

자원 삭제

- ▶ DB 인스턴스 삭제
- ▶ DB 서브넷 그룹 삭제
- ▶ 네트워크 자원 삭제: NAT, EIP, 서브넷, 인터넷게이트웨이, VPC
- ▶ EC2 인스턴스 삭제