Module 3 - AWS 통합형 서비스 (1, 2, 3, 4, 5)

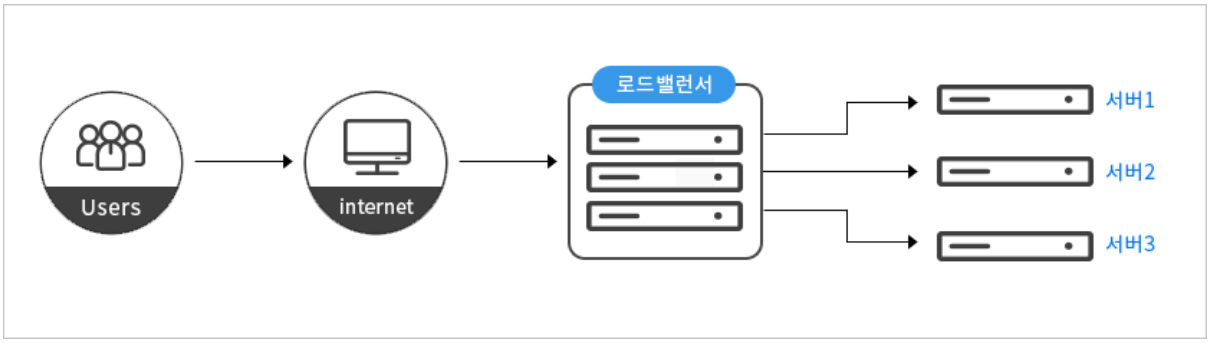
Created At	@Dec 17, 2020 9:26 AM
Person	민 경환
Tags	
Updated At	@Dec 18, 2020 3:12 PM
마지막 수정	민 경환
발표일	
발표자	
분류	스터디
작성자	민 경환
참석 인원	
태그	AWS Cloud Practitioner Essentials
프로젝트	

Application Load Balancer

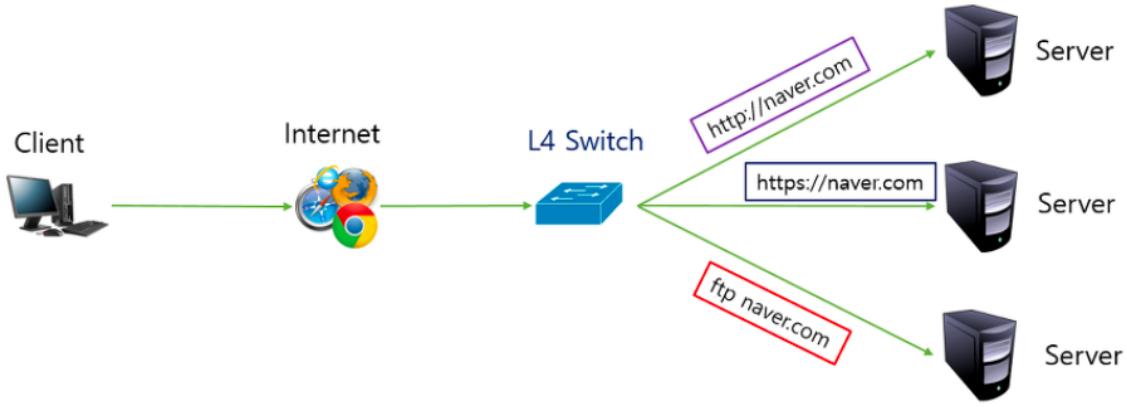
- Elastic Load Balancing 서비스의 일부로 도입된 두 번째 유형의 Load Balancer
- Elastic Load Balancing
    1. Classic Load Balancer (L4/L7)
    2. Application Load Balancer (L7)
    3. Network Load Balacner (L4)

Load Balancer

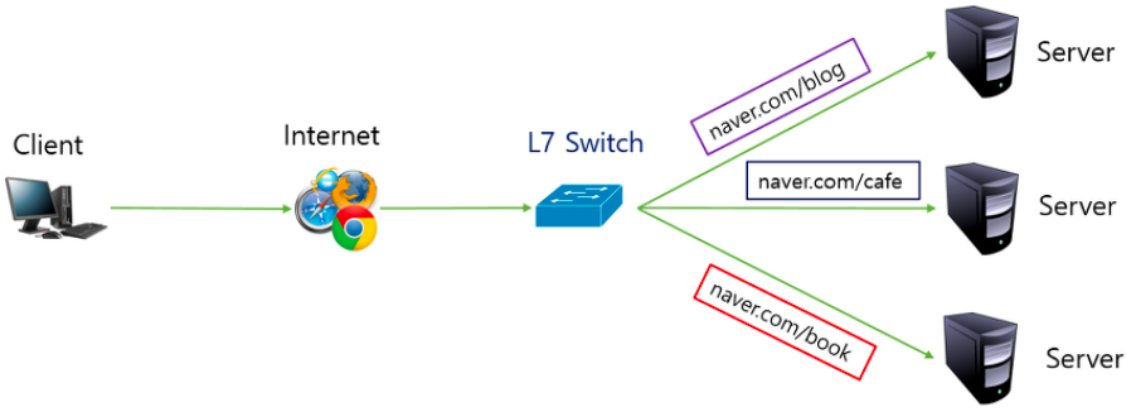
- 여러개의 서버에 균등하게 트래픽을 분산시켜주는 역할



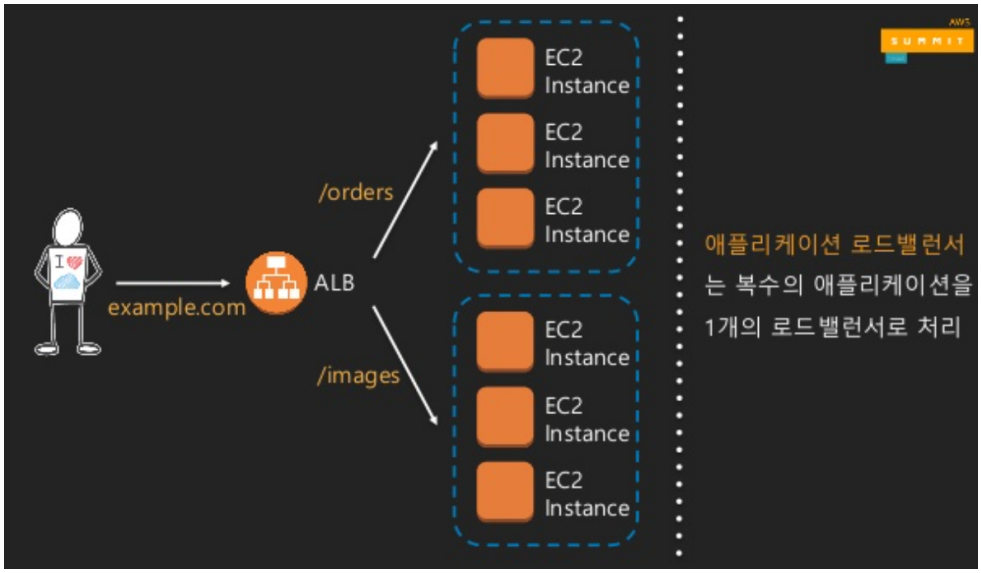
- L4 스위치
  - TCP/UDP 포트 정보를 확인하여 적절한 서버로 패킷 전송



- L7 스위치
  - L4 스위치의 단점을 극복하기 위해 포트 + 데이터 페이로드를 이용하여 서버로 패킷 전송
  - Application Load Balancer (이하 ALB) 가 여기에 해당

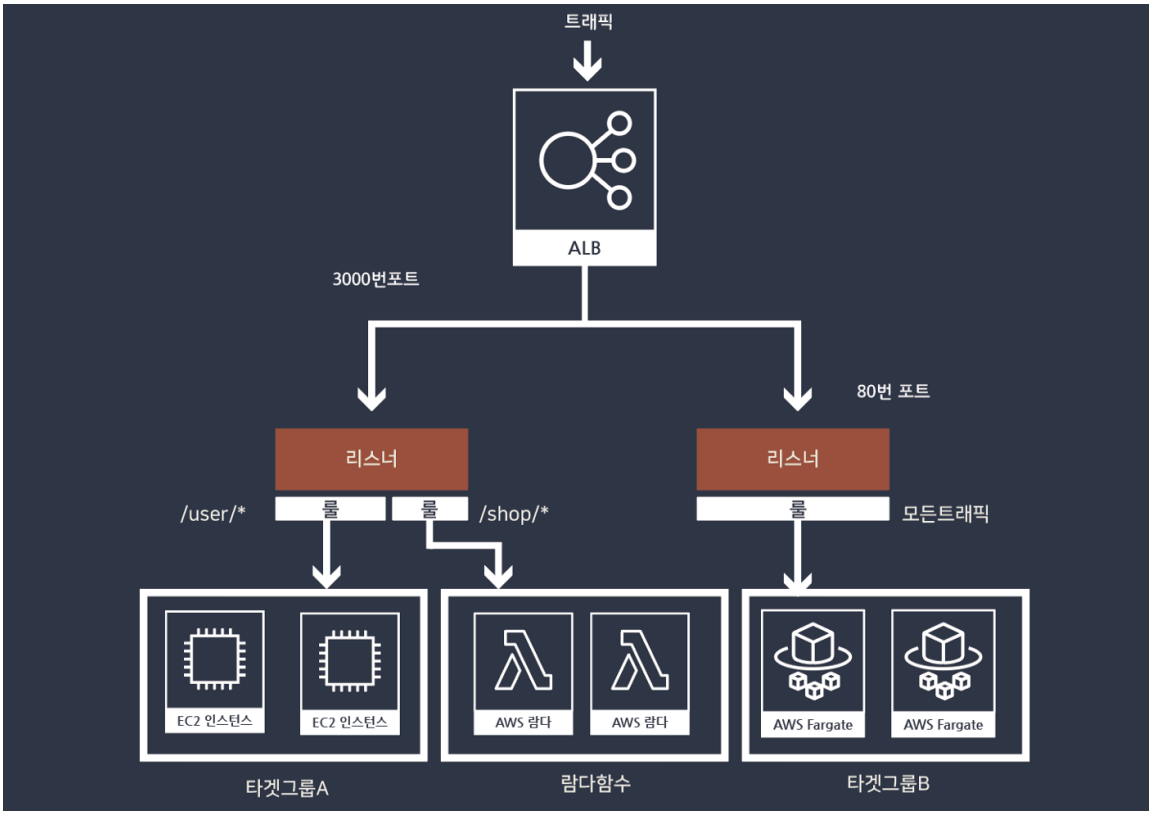


- 향상된 기능
  1. 지원되는 프로토콜
     - HTTP, HTTPS, HTTP/2 프로토콜 및 WebSocket 지원
  2. CloudWatch 지표
     - 로드 밸런서 및 대상에 대한 데이터 포인트를 Amazon CloudWatch에 게시
     - 요청이 로드 밸런서를 통과하는 경우에만 CloudWatch에 지표를 보고
     - 지정된 기간 동안 로드 밸런서에 대한 정상 상태 대상의 총 수를 모니터링
  3. 액세스 로그
     - 전송된 요청에 대한 자세한 정보
       - 요청을 받은 시간
       - 클라이언트의 IP 주소
       - 지연 시간
       - 요청 경로 및 서버 응답 등
     - WebSocket 연결의 연결 세부 정보
  4. 상태확인
     - 등록된 인스턴스로 요청을 주기적으로 전송하여 세부적인 상태를 확인
- 추가기능
  1. 경로 기반 라우팅
     - URL에 기반하여 대상 그룹으로 요청을 전달하는 규칙을 통해 리스너를 생성
       - `/orders` 로 들어오는 모든 요청은 order 서버로 라우팅
       - `/images` 로 들어오는 모든 요청은 image 서버로 라우팅



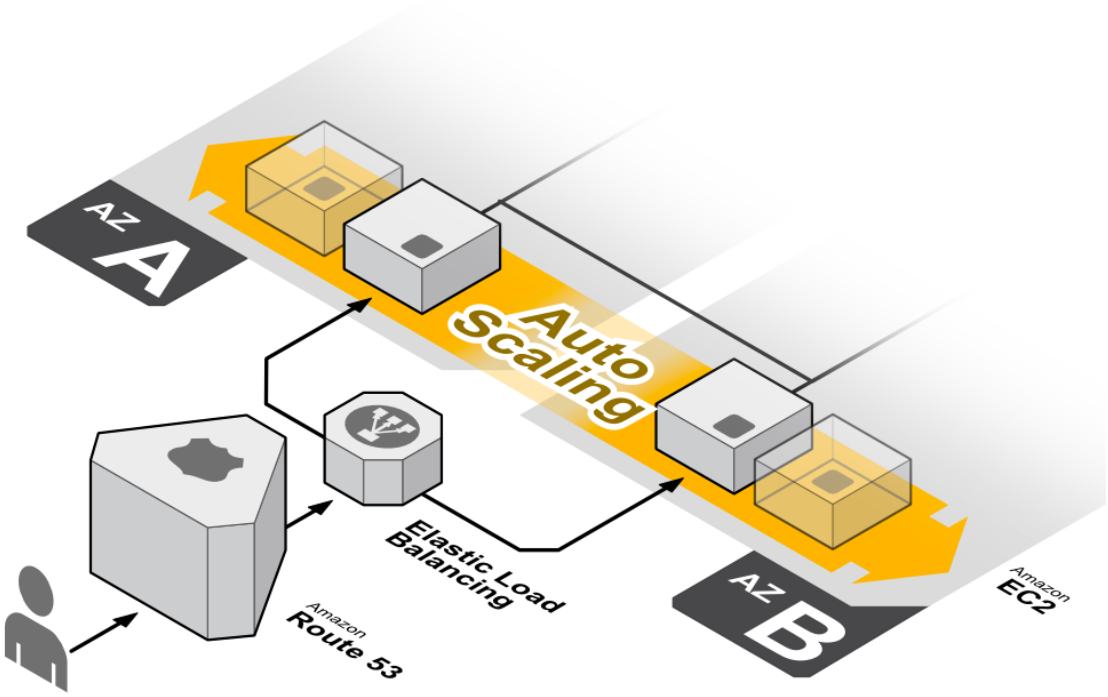
2. 호스팅 기반 라우팅
    - 호스트 헤더에 지정된 도메인 이름을 기반으로 들어오는 트래픽을 라우팅
        - `order.example.com` 에 대한 요청은 하나의 대상 그룹으로
        - `mobile.example.com` 에 대한 요청은 다른 그룹
3. 네이티브 IPv6 지원
4. AWS WAF (Web Application Firewall)
    - 선택적으로 웹 응용 프로그램의 특정 경로에 대한 접근을 허용하거나 거부설정 가능
5. 동적포트
6. 삭제 보호 및 요청 추적

- 주요 용어

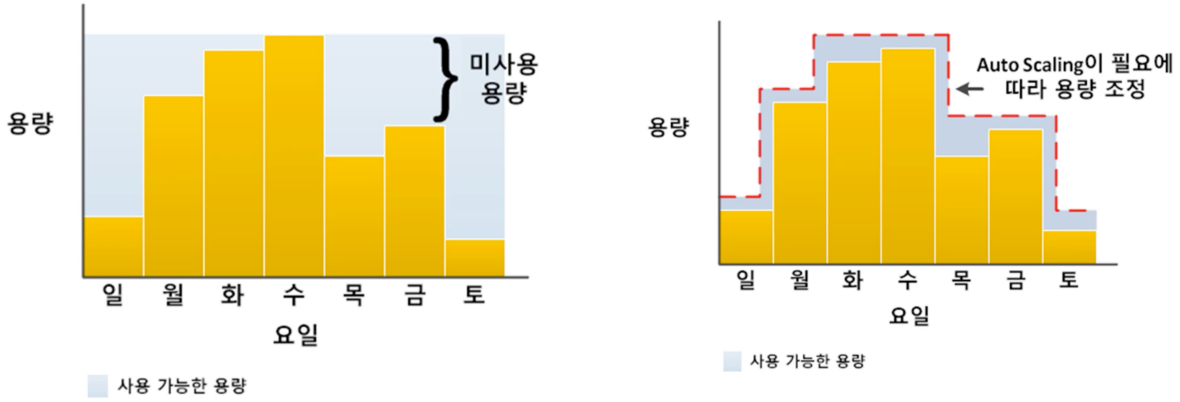


1. 리스너
    - 프로토콜 및 포트를 사용하여 클라이언트의 연결 요청을 확인
    - 각 리스너에 대한 기본 룰(규칙)을 정의
        - 리스너 규칙을 생성할 때 대상 그룹 및 조건을 지정
        - 규칙 조건이 충족되면 해당하는 대상 그룹으로 트래픽이 전달
2. 대상
    - 설정된 리스너 규칙에 따른 목적지
    - EC2 인스턴스, 마이크로서비스 애플리케이션 등
3. 대상 그룹
    - 대상에 대한 논리적인 그룹
    - 프로토콜과 포트 번호를 사용하여 EC2 인스턴스 같은 하나 이상의 등록된 대상으로 요청을 라우팅

Auto Scaling



- 지정한 조건에 따라 EC2 인스턴스를 추가하거나 제거하는 기능

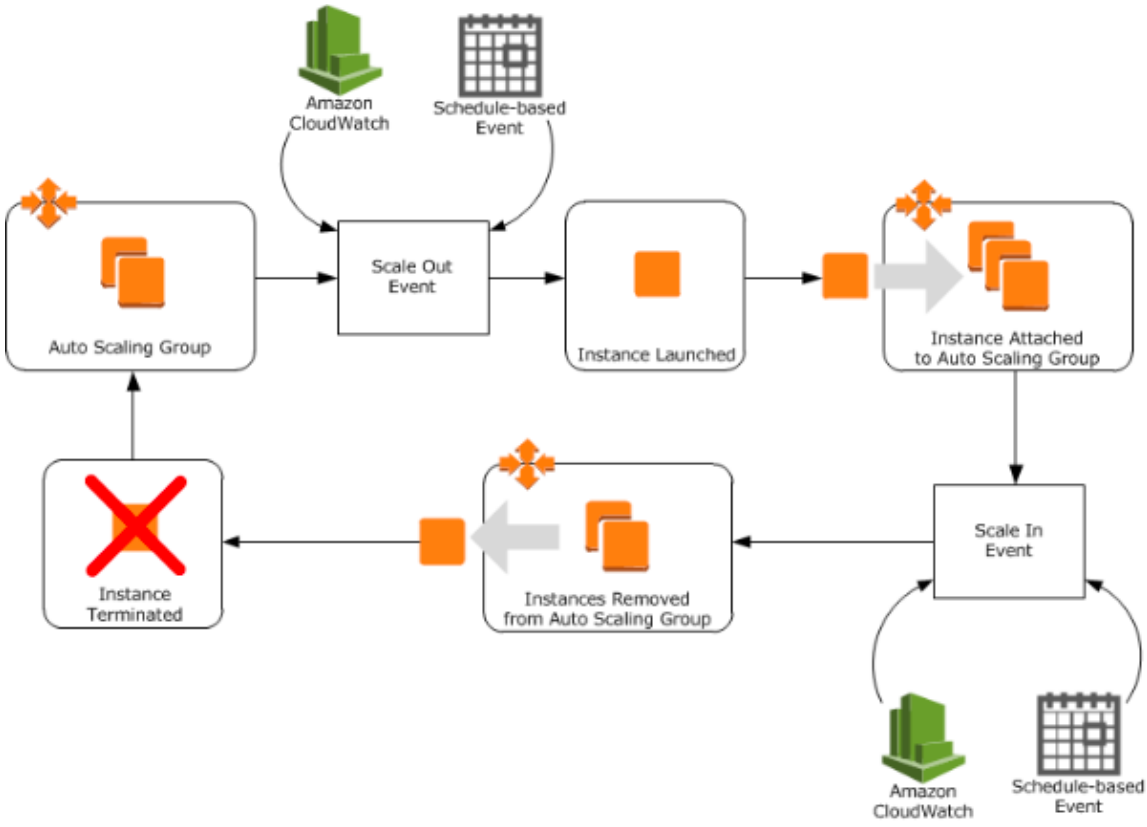


- 확장 및 축소
  - 사용자가 정의한 조건 (CPU 사용률 80% 이상) 이나 스케줄을 기준으로 EC2 인스턴스 수를 자동으로 조정 가능
    - `Scaling OUT` - Auto Scaling 이 더 많은 인스턴스를 추가하는 것
    - `Scaling IN` - Auto Scaling 이 인스턴스를 종료하는 것
- 구성요소
  1. 시작 구성 - 무엇을 배포할 것인지 정의

- EC2 인스턴스를 구성할 때 지정해야하는 모든 사항을 정의
  - Amazon 머신 이미지(AMI)의 ID
  - 인스턴스 유형 및 키 페어
  - 하나 이상의 보안 그룹
  - 인스턴스에 대한 추가 EBS 볼륨 또는 인스턴스 스토어 볼륨의 ID

2. Auto Scaling 그룹 - 배포 위치를 정의
   - VPC 및 서브넷
   - 로드 밸런서
   - 최소, 최대 인스턴스
   - 원하는 용량
3. Auto Scaling 정책 - 배포 시기를 정의
   - EC2 인스턴스를 시작하거나 종료할 시기를 저장하는 방법
     - e.g) 매주 금요일 오후 3시에 Auto Scaling 이 동작하게

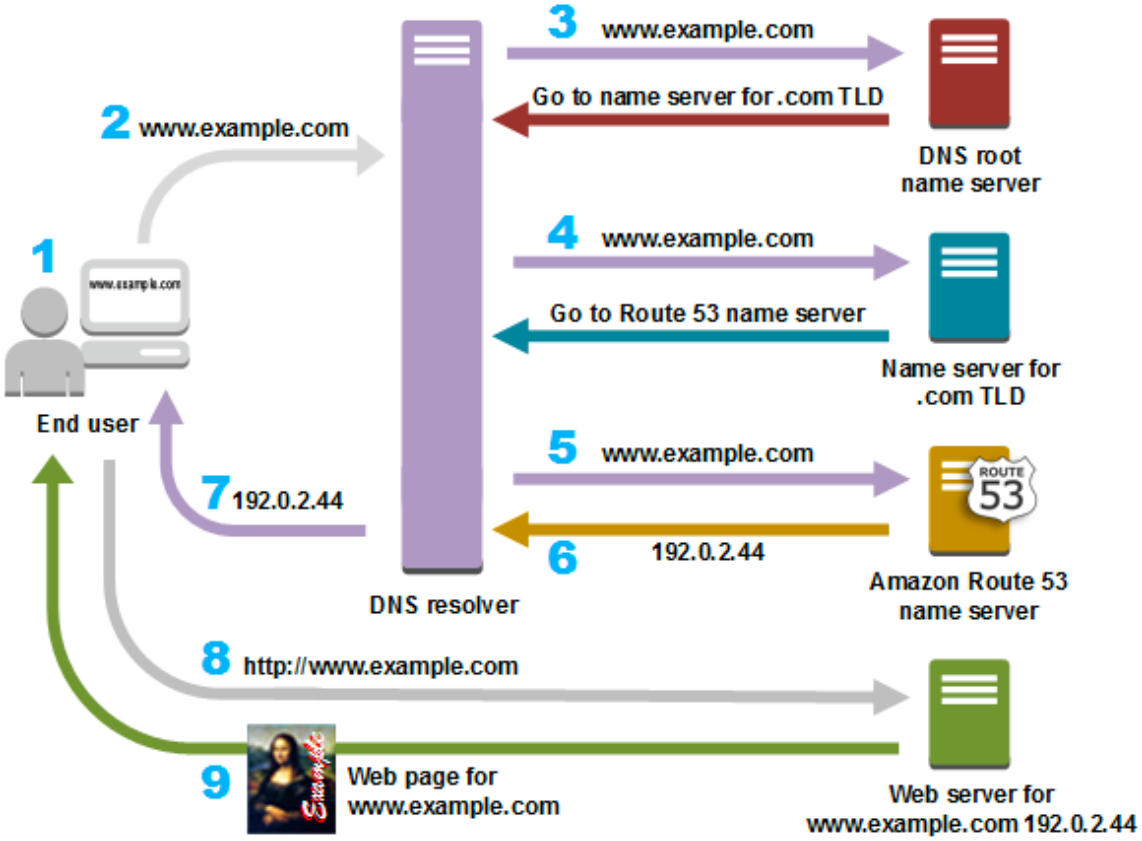
- 동적 Auto Scaling



1. 변화에 대응하여 Auto Scaling 그룹의 용량을 조정하는 방법
2. EC2 인스턴스 혹은 로드 밸런서의 성능 정보를 기반으로 CloudWatch 경보 작성
3. 경보를 트리거하는 데 사용되는 지표는 Auto Scaling 그룹의 모든 인스턴스에서 보내는 지표를 집계
    - 하나는 60% CPU이고 다른 하나는 40% CPU인 두 개의 인스턴스가 있는 Auto Scaling 그룹이 있다면 평균적으로 50% CPU
4. 조정 정책이 실행될 때 최소 및 최대 크기 제한을 벗어나지 않도록 생성하거나 삭제
    - 그룹의 최대 용량은 3이고, 현재 용량은 2이며, 조정 정책은 인스턴스 3개를 추가 → 1개만 추가
    - 그룹의 최소 용량은 2이고, 현재 용량은 3이며, 조정 정책은 인스턴스 2개를 제거 → 1개만 제거

Amazon Route 53

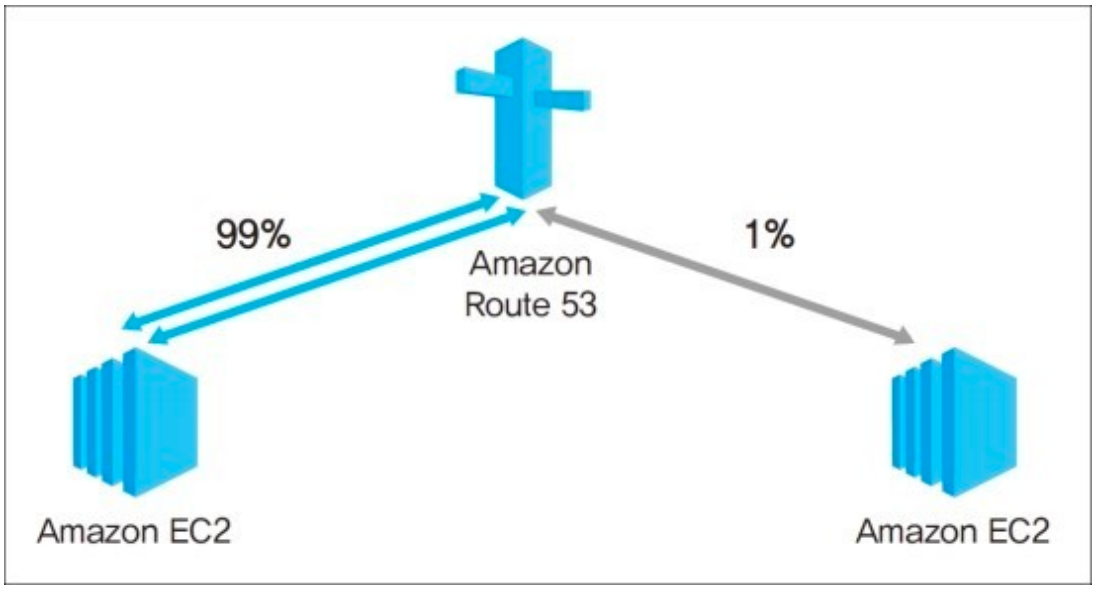
- 글로벌 고가용성 `DNS`(Domain Name System) 서비스
    - *가용성 - 시스템이 서비스를 정상적으로 제공할 수 있는 상태
- `www.example.com`과 같은 이름을 `192.0.2.44` 와 같은 컴퓨터 간 연결에 사용되는 숫자 IP 주소로 변환
- Amazon Route 53이 도메인의 트래픽을 라우팅하는 방법



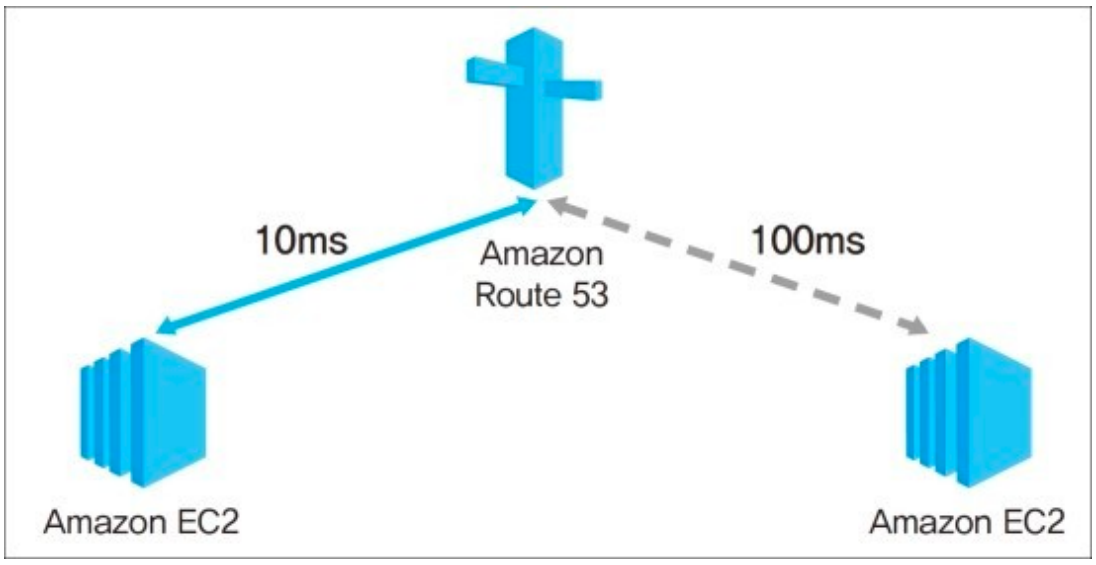
1. 사용자가 웹 브라우저에 접속한다
2. 도메인을 입력한다 e.g.) www.example.com
3. www.example.com 요청을 DNS 루트 이름 서버로 전달
4. www.example.com 요청을 .com 도메인의 이름 서버로 전달, 이후 .com 도메인의 이름 서버는 example.com 도메인과 연관된 4개의 Route 53 이름 서버의 이름을 사용하여 요청에 응답
5. Route 53 이름 서버 하나를 선택하여 www.example.com 에 대한 요청을 해당 이름 서버에 전달
6. Route 53 이름 서버는 example.com 호스팅 영역에서 www.example.com 레코드를 찾아 웹 서버의 IP 주소 192.0.2.44 등 연관된 값을 받아 반환

- 호스팅 영역
  1. Route 53 을 사용할 때 가장 먼저 생성, DNS 데이터가 유지되는 곳
  2. 호스팅 영역을 만들면 메인 서버 4개가 생성
- 주요 특징 및 기능
  1. 단순 라우팅

2. 지리적 위치
    - DNS 쿼리가 발생하는 위치를 기반으로 트래픽을 제공하는 리소스를 선택
    - e.g.) 유럽에서 발생하는 모든 쿼리를 프랑크푸르트 리전으로
3. 장애 조치
    - 특정 리소스가 정상일 경우 해당 리소스로 트래픽을 라우팅
    - 특정 리소스가 비정상일 경우 다른 리소스로 트래픽을 라우팅
4. 가중치 기반 라운드 로빈



    - 각 리소스로 라우팅되는 트래픽 비율을 선택
5. 지연 시간 기반

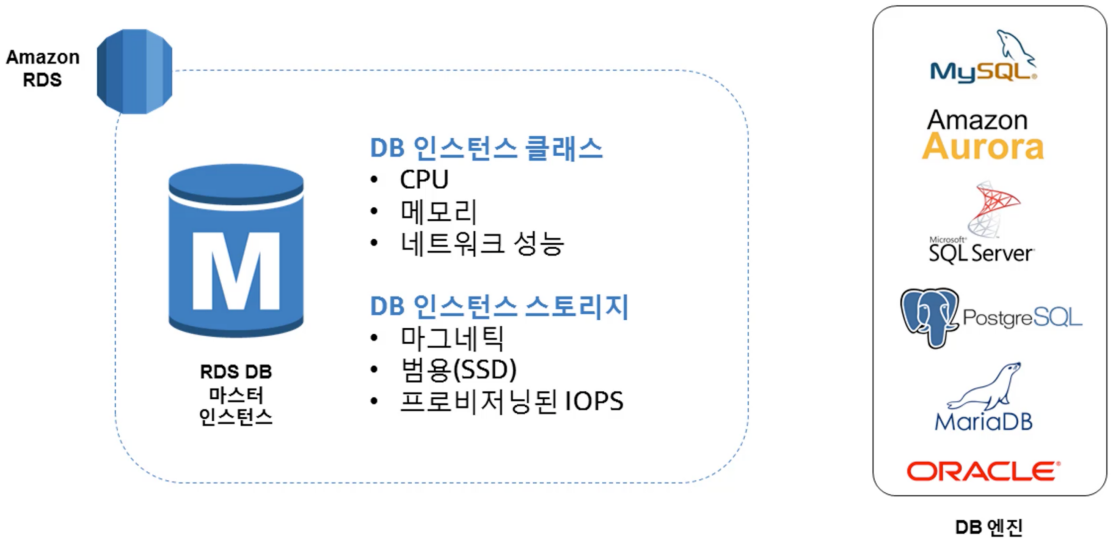


- 도메인 하나에 각 지역별로 가장 빠른 곳으로 연결해주는 서비스
- 사용자가 전세계 어디서든 가장 반응이 빠른 네임서버로 연결

6. 다중 응답
    - 하나의 DNS 명에 대해서 여러개의 IP 주소를 반환
    - 서버의 상태를 체크해서 장애 상태인 경우에는 제외 복구시 추가

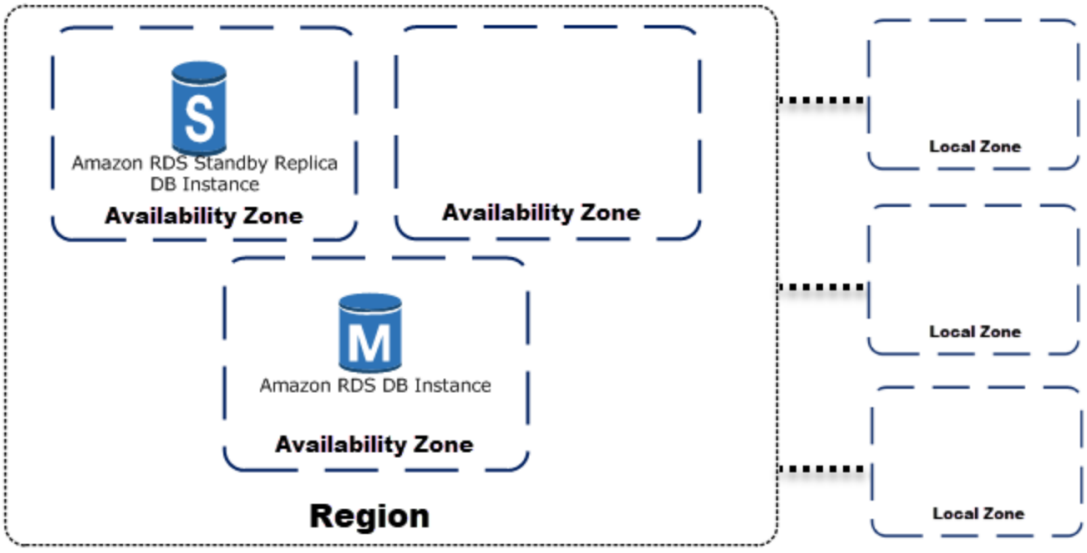
Amazon Relational Database Services (RDS)

- AWS 클라우드에서 관계형 데이터베이스를 더 쉽게 설치, 운영 및 확장할 수 있는 웹 서비스
- RDS 가 필요한 이유
    1. [기존] CPU, 메모리, 스토리지 및 IOPS가 모두 한데 묶여 제공 → 독립적으로 확장 가능
    2. 백업, 소프트웨어 패치, 자동 장애 감지 및 복구를 관리
    3. 필요할 때 자동화된 백업을 수행하거나 고유한 백업 스냅샷 생성 및 복원 가능
    4. 친숙한 데이터베이스 제품 사용 가능 (6가지)
        - MySQL, Oracle, SQL Server, PostgreSQL, MariaDB, Amazon Aurora
    5. 시간 소모적인 관리 작업을 자동화
- Amazon RDS DB 인스턴스

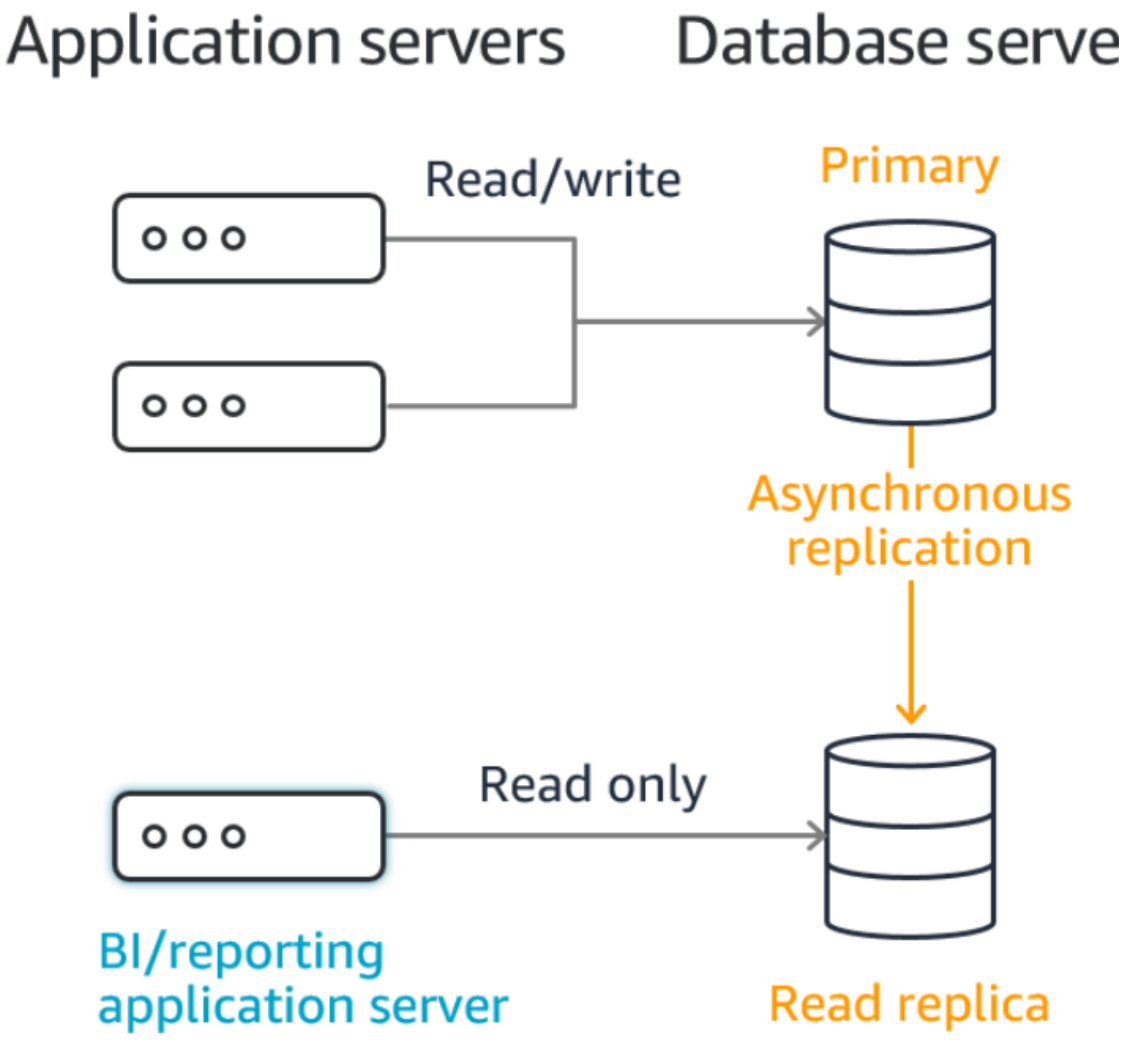


1. 클라우드에서 실행하는 격리된 데이터베이스 환경, RDS 의 기본 구성 요소
2. CPU, 메모리, 스토리지 및 네트워킹 용량 설정 가능
3. 데이터베이스에 따라 적절한 리소스 조합을 선택할 수 있는 유연성을 제공

- 다중 AZ 배포하여 고가용성 데이터베이스 인스턴스 구성가능



  - 서로 다른 가용 영역에 동기식 예비 복제본 유지
- Amazon RDS 읽기 전용 복제본 생성 지원



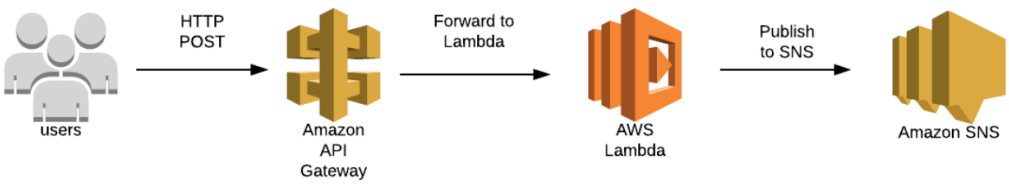
1. 읽기 전용 복제본으로 읽기 쿼리를 라우팅하여 원본 DB 인스턴스의 로드를 줄일 수 있음
2. 단일 DB 인스턴스 용량의 한도 이상으로 탄력적으로 확장 가능
3. 읽기 전용 복제본을 마스터 상태로 승격가능

AWS Lambda

- 서버를 프로비저닝하거나 관리하지 않고도 코드를 실행할 수 있도록 지원하는 컴퓨팅 서비스
  - *프로비저닝 - 사용자의 요구에 맞게 시스템 자원을 할당, 배치, 배포해 두었다가 필요 시 시스템을 즉시 사용할 수 있는 상태로 미리 준비해 두는 것

- 사용하는 컴퓨팅에 대해서만 비용을 지불 (코드가 실행되지 않을 때는 비용을 지불하지 않음)
- AWS 서비스에서 코드를 자동으로 트리거하도록 설정하거나 웹 또는 모바일 앱에서 직접 호출 가능
- 다양한 관리 작업을 제공하는 고가용성 컴퓨팅 인프라에서 코드를 실행
    1. 서버 및 운영체제 유지 관리
    2. 용량 프로비저닝 및 Auto Scaling
    3. 코드 모니터링, 로깅
- Node.js, Java, C#, Python 등 다양한 프로그래밍 언어 지원
- 이벤트 중심의 컴퓨팅에 사용하면 효과적
    1. e.g) S3 버킷에 대한 변경사항 및 이벤트에 대응하여 코드를 실행

    2. e.g) Amazon API Gateway 를 사용한 HTTP 요청에 응답 가능



- 사용 사례
    1. 실시간 파일 처리
    2. 실시간 스트림 처리
    3. 추출, 변환, 로드
    4. IoT 백엔드
    5. 모바일 백엔드
    6. 웹 애플리케이션