클라우드 컴퓨팅 소개 [AWS Cloud]

목차

프로페셔널

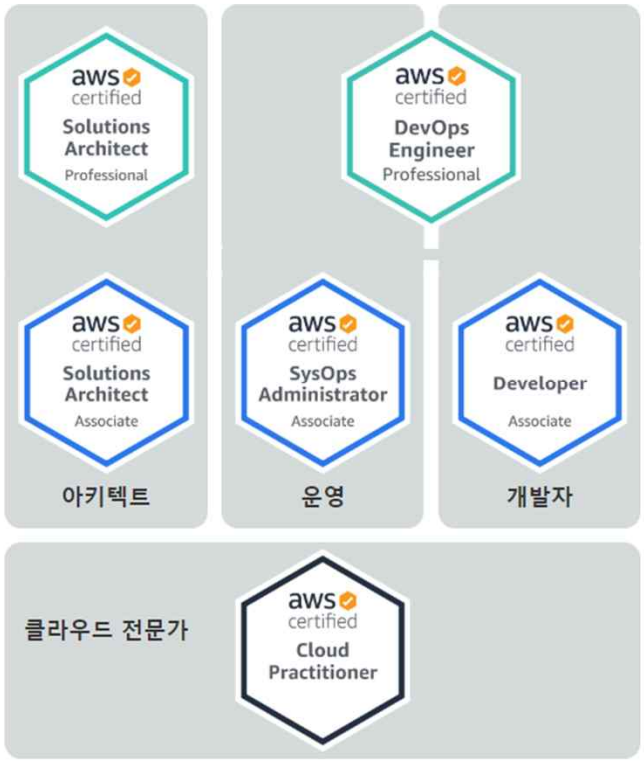
2년 의 AWS 클라우드를 사용한 설계, 운영 및 솔루션 문제 해결과 관련된 포괄적인 경험

어소시에이트

1년의 문제 해결 및 AWS 클라우드를 사용한 솔루션 구현 경험

기초

6개월의 기초 AWS 클라우드 및 업계 지식



전문 분야

시험 가이드에 명시된 대로 전문 분야 도메인에 대한 기술적 AWS 클라우드 경험

Before we start

1. Introduction of AWS

Server - Client Model

- **서버** : 서비스를 제공하는 컴퓨터 시스템
  AWS 에서의 가상 서버인 Amazon Elastic Cloud Compute(**Amazon EC2**)

- **클라이언트** : 사람이 컴퓨터 서버에 요청을 보내기 위해 상호 작용하는 웹 브라우저 또는 데스크톱 애플리케이션

AWS (Amazon Web Service)

- 클라우드 컴퓨팅은 인터넷(Network) 기술을 이용하여 내-외부 고객들에게 확장성(Scalable) 있고 탄력적(Elastic)인 IT 서비스가 제공되는 방식

- Cloud is a Style of Computing where Scalable and Elastic IT-related capabilities are provided as a service to customer using Internet technologies

AWS (Amazon Web Service)

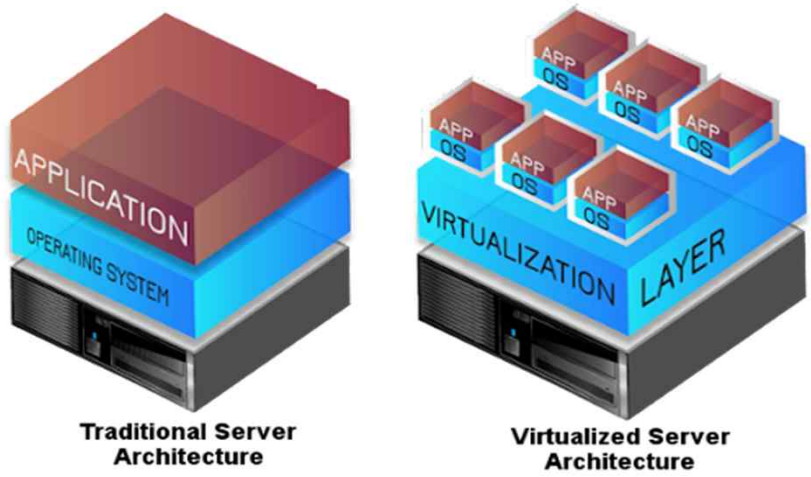
- **클라우드 컴퓨팅이전의 시스템 운영 방식 (On-Premise)**
  - 서비스 사용 용량을 비즈니스 시작 이전에 이론적으로 산정
  - 충분한 자원(CPU, Memory, Storage, Network…) 용량 산정 필요

AWS (Amazon Web Service)

- **AWS와 함께하는 클라우드 컴퓨팅 (Cloud Computing)**
  - 서버 / 데이터베이스 / 스토리지 / 상위 수준의 애플리케이션
  - 몇 초 만에 시작 가능
  - 임시적이고 삭제 가능한 것으로 취급
  - 비유연성 및 제약으로부터 자유로움

AWS (Amazon Web Service)

- **가상화(Virtualization)**
  - 물리적인 컴포넌트(Components, HW장치)를 논리적인 객체로 추상화 하는 것을 의미
  - 가상화의 대상이 되는 컴퓨팅 자원은 프로세서(CPU), 메모리(Memory), 스토리지(Storage), 네트워크(Network), 그래픽 처리 장치(GPU)가 있으며, 이들로 구성된 서버나 장치들을 가상화 함으로써 높은 수준의 자원 사용률과 분산 처리 능력을 얻을 수 있음



AWS (Amazon Web Service)

[특성]

- **On-Demand Self-Service: 주문형 셀프서비스**
  - 사용자가 별도의 기술습득 없이 필요할 때 온라인으로 즉시 사용
- **Broad Network Access: 광대역 액서스**
  - 네트워크를 통한 컴퓨팅 자원 접근(Any time, Any place, Any device)
- **Resource Pooling: 자원 공동관리**
  - 다중 임대 모델을 통한 자원 할당(Multi-Tenancy)
- **Rapid Elasticity: 요구탄력성**
  - 비즈니스 상황에 따른 컴퓨팅자원의 탄력적 사용(Flexibility, Scalability)
- **Measured Service: 도수제**
  - 서비스를 사용한 만큼 비용 지불(Pay-Per-Use, Pay as you go)

AWS (Amazon Web Service)

- **AWS Cloud Computing Benefits**

  - **자본 비용(선행 비용)을 가변 비용으로 대체**
  - 선행 비용은 데이터 센터, 물리적 서버 등 미리 투자를 해야 사용할 수 있는 리소스를 사용하는 경우 발생
  - 가변 비용의 경우 어떻게 사용할지 결정하기도 전에 데이터 센터와 서버에 대규모로 투자하는 대신, 사용하는 컴퓨팅 리소스에 대해서만 비용을 지불
    → 비용을 절감하면서 혁신적 솔루션 구현

  - **규모의 경제로 얻게 되는 혁신**
  - 클라우드 컴퓨팅을 사용하면 소유하고 있는 인프라에서 작업을 수행할 때보다 가변 비용이 낮음
  - 클라우드를 사용하는 고객이 많아질 수록 Amazon Web Services 와 같은 공급자는 더 높은 규모의 경제를 달성
- → 따라서 사용량에 따라 지불하는 방식의 요금이 더 낮아짐

AWS (Amazon Web Service)

- **필요한 용량을 추정할 필요 없음**

- 탄력적 인프라 설계 가능
- 혁신적인 신규 서비스/제품
- 여러 리전에 배포함으로 더 빠르게 고객에게 서비스 가능
- 고객이 원하는 속도로 서비스 이용 가능
- 워크로드가 증가하면 확장, 불필요한 리소스를 종료 → Auto Scaling 사용

AWS (Amazon Web Service)

- **AWS Cloud Computing Benefits**
  - **속도 및 민첩성 개선**
  - 전 세계에 서비스를 배포하여 즉각적인 글로벌 접근성을 제공
  - 새로운 리소스 배포 시 빠른 가용성 가능
  - 모든 인프라를 코드 형태의 템플릿으로 운영하면서, 운영자에 의한 실수를 최소화 → AWS CloudFormation
  - 낮은 비용으로 새로운 비즈니스 요구사항을 실험 → 새로운 비즈니스 진출

  - **데이터 센터 운영 및 유지 관리에 비용 투자 불필요**
  - 인프라가 아니라 비즈니스를 차별화하는 프로젝트에 집중
  - 클라우드 컴퓨팅을 사용하면 수많은 서버를 관리하느라 시간을 허비하지 않고 고객에 더욱 집중

  - **몇 분만에 전 세계에 배포**
  - 클릭 몇 번으로 세계 곳곳의 여러 리전에 어플리케이션을 손쉽게 배포
- → 다시 말해 최소 비용으로 간단하게 고객에게 더 짧은 지연 시간과 더 나은 경험을 제공

AWS (Amazon Web Service)

- **Cloud Computing Limitations**
  - Internet Access(No Internet = No Cloud)
  - Security(How do you know?)
  - Privacy(What legislation or regulations?)
  - Vendor Lock-in(Application migration may be impossible)

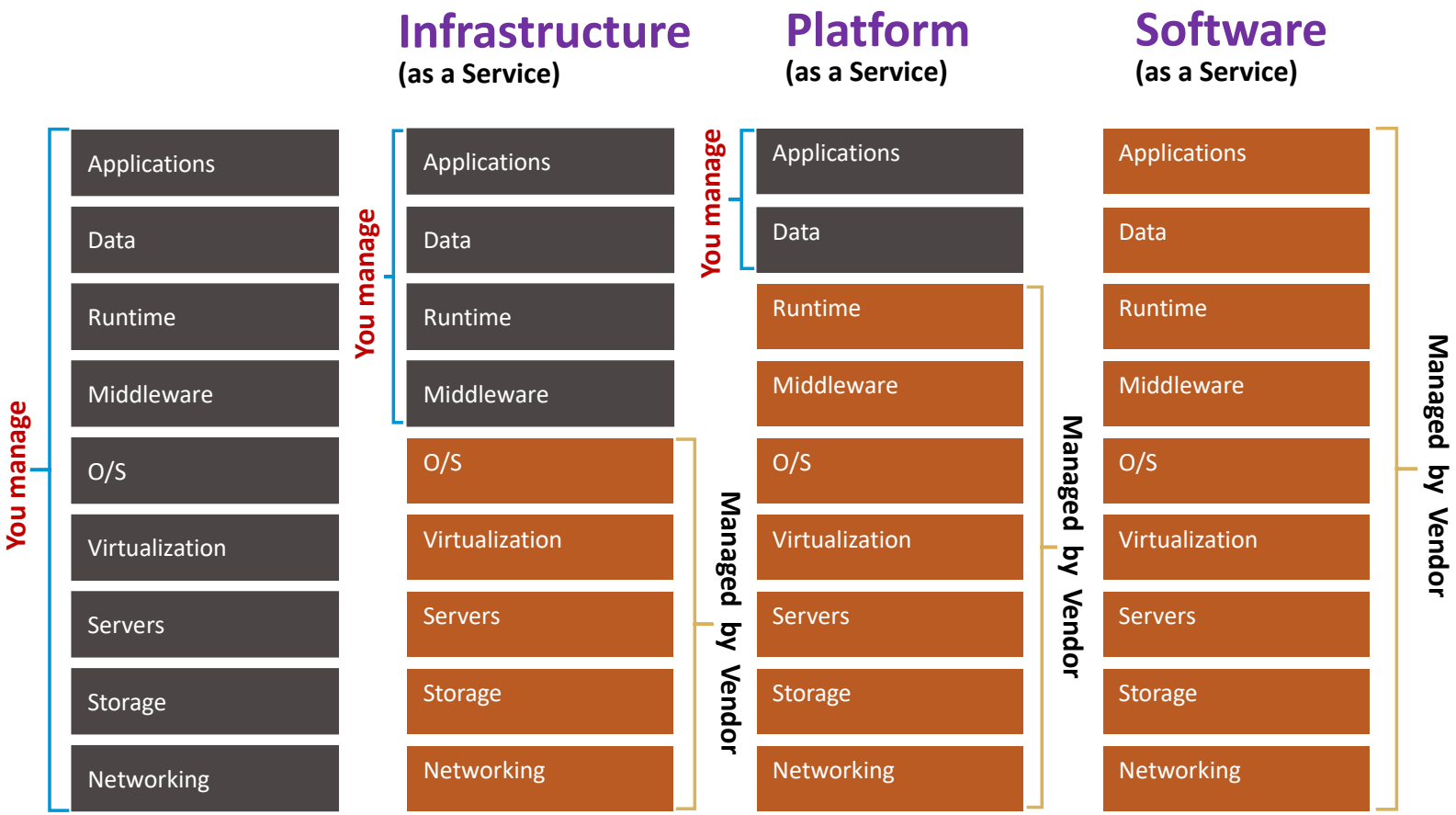
AWS (Amazon Web Service)

Cloud Computing Service Model

INFRASTRUCTURE AS A SERVICE(IAAS)	PLATFORM AS A SERVICE(PAAS)	SOFTWARE AS A SERVICE(SAAS)
Infrastructure as a Service(IaaS)는 클라우드 IT의 기본 빌딩 블록을 포함하고 일반적으로 네트워킹 기능, 컴퓨터(가상 또는 전용 하드웨어) 및 데이터 스토리지 공간을 제공 Infrastructure as a Service(IaaS)는 IT 리소스에 대해 가장 높은 수준의 유연성과 관리 제어를 제공하며 오늘날 많은 IT 부서와 개발자에게 익숙한 기존 IT 리소스와 가장 비슷	Platform as a Service(PaaS)를 사용하면 조직은 기본 인프라(일반적으로 하드웨어와 운영 체제)를 관리할 필요가 없어 애플리케이션 개발과 관리에 집중 즉, 애플리케이션 실행과 관련된 리소스 구매, 용량 계획, 소프트웨어 유지 관리, 패치 또는 다른 모든 획일적인 작업에 대한 부담을 덜어 더욱 효율적	Software as a Service는 서비스 제공업체에 의해 실행되고 관리되는 완전한 제품을 고객에게 제공 대부분의 경우 Software as a Service라고 하면 최종 사용자 애플리케이션을 의미 SaaS 오퍼링을 사용하면 서비스가 어떻게 유지 관리되는지 또는 기본 인프라가 어떻게 관리되는지 생각할 필요가 없으며 소프트웨어 이 특정 부분을 어떻게 사용 할지 고려 SaaS 애플리케이션의 일반적인 예로는 이메일 제품용 추가 기능을 관리할 필요가 없고 이메일 프로그램이 실행되는 서버 및 운영 체제를 유지 관리하지 않고 이메일을 보내고 받을 수 있는 웹 기반 이메일 같은 서비스

AWS (Amazon Web Service)

- Cloud Computing Service Model



AWS (Amazon Web Service)

[Cloud Computing Deployment Models]

. public

- 클라우드 기반 애플리케이션은 클라우드 상에 완전히 배포되며 애플리케이션의 모든 부분이 클라우드에서 실행

- 클라우드의 애플리케이션은 클라우드에서 생성되었거나 클라우드 컴퓨팅의 이점을 활용하기 위해 기존 인프라에서 클라우드로 이전(migration) 되어 서비스

- 클라우드 기반 애플리케이션은 낮은 수준의 인프라상에 구축할 수 있고 또는 주요 인프라를 관리, 설계 및 확장할 필요가 없는 높은 수준의 서비스를 사용 가능

AWS (Amazon Web Service)

. Hybrid

- 하이브리드 배포는 클라우드 기반 리소스와 클라우드에 위치하지 않은 기존 리소스 간에 인프라와 애플리케이션을 연결하는 방식

- 클라우드와 기존 온 프레미스 인프라 간에 가장 일반적인 하이브리드 배포 방법은 클라우드 리소스를 내부 시스템에 연결하면서 조직의 인프라를 클라우드로 확장하는 모델

AWS (Amazon Web Service)

. private

- 가상화 및 리소스 관리 도구를 사용하여 온프레미스에 리소스를 배포하는 것을 "프라이빗 클라우드"

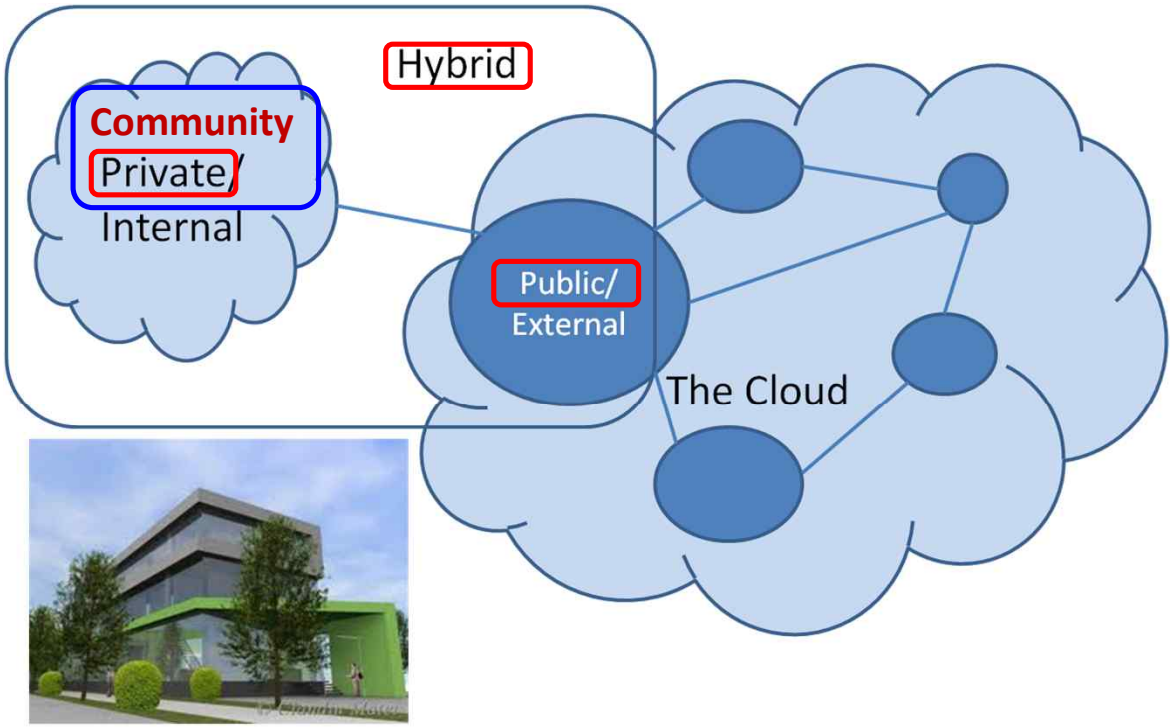
- 온 프레미스 배포는 클라우드 컴퓨팅이 가진 많은 장점을 제공하지는 않지만 전용 리소스를 제공하는 온프레미스 기능이 필요할 때가 있음

- 대부분의 경우 온 프레미스 배포 모델은 리소스 활용도를 높이기 위해 애플리케이션 관리 및 가상화 기술을 사용한다는 점에서 레거시 IT 인프라와 동일

https://aws.amazon.com/ko/types-of-cloud-computing/?WICC-N=tile&tile=types_of_cloud

AWS (Amazon Web Service)

- **Cloud Computing Deployment Models**



On Premise / Internal **Off Premise / External**

2. Global infrastructure

Global infrastructure

Magic Quadrant for Cloud Infrastructure and Platform Service(2020)

https://pages.awscloud.com/GLOBAL-multi-DL-gartner-mq-cips-2020-learn.html

Global infrastructure

- AWS 글로벌 클라우드 인프라는 업계에서 가장 안전하고 광범위하고 안정적인 클라우드 플랫폼으로, 전 세계 데이터 센터를 통해 완전한 기능을 갖춘 200가지가 넘는 서비스를 제공

- 클릭 한 번으로 전 세계 모든 위치에 애플리케이션 워크로드를 배포하거나 한 자릿수 밀리 초의 지연 시간으로 최종 사용자에게 더 가까운 위치에 특정 애플리케이션을 배포해야 하는 경우 언제 어디서나 필요할 때 AWS의 글로벌 인프라를 사용 가능

- AWS는 전 세계적으로 수백만 명의 활동 고객과 수만 개의 파트너로 이루어진 가장 큰 규모의 가장 역동적인 에코시스템을 갖추고 있으며, 스타트 업, 엔터프라이즈, 공공 부문의 조직을 비롯해 규모에 상관없이 거의 모든 산업의 고객이 AWS에서 다양한 사용 사례를 운영

Global infrastructure

AWS Global inframap

- 모든 리소스를 보관하는 거대한 데이터 센터 하나를 보유하는 것은 바람직하지 않음
- AWS는 25개 지리적 리전에 걸쳐 80개의 가용 영역을 보유하고 있으며 앞으로 호주, 인도, 인도네시아, 스페인과 스위스에 5개의 AWS 리전과 15개의 가용 영역을 추가할 계획

- **보안**
- **가용성**
- **성능**
- **국제적 입지**
- **확장성**
- **유연성**

https://aws.amazon.com/ko/about-aws/global-infrastructure/?hp=tile&tile=map

Global infrastructure

리전(Region)

- 리전은 AWS의 서비스가 위치하는 물리적인 장소로 전세계적으로 리전들이 분포
- 각 리전에는 애플리케이션을 실행하는 데 필요한 컴퓨팅, 스토리지 및 기타 모든 서비스가 구비된 다양한 데이터 센터가 존재
- 리전은 AWS에서 제어하는 고속 광섬유 네트워크를 통해 다른 리전에 연결
- 각 리전들은 다른 리전과 격리되도록 설계되어 있어 이를 통해 가장 강력한 내결함성 및 안정성 보장

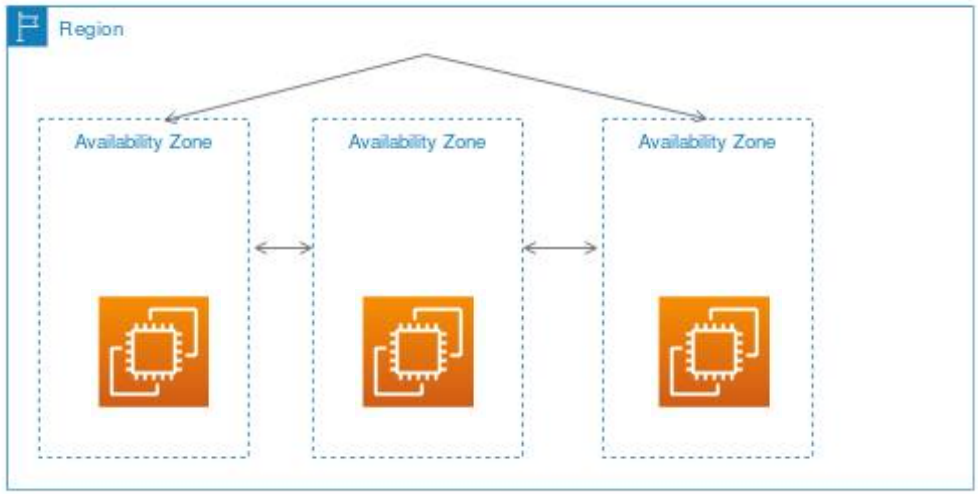
AWS 리전 선택시 고려사항

1) 규정 준수
2) 근접성
3) 기능 가용성
4) 요금

Global infrastructure

가용영역(Availability Zone, AZ)

- 각 리전은 가용 영역이라고 알려진 격리된 위치를 여러 개 보유
- 일반적으로 리전에는 가용 역역이 보통 두개 이상(Seoul 4개) 구성되어 있으며, 가용 영역 간에는 고속의 망으로 연결
- 복수의 가용 영역에 걸쳐 인스턴스를 배포했을 때 하나의 인스턴스에 장애가 발생한 경우를 대비하여, 다른 가용 영역의 인스턴스가 장애가 발생한 인스턴스 관련 요청을 처리할 수 있도록 애플리케이션을 설계 권장
- 인스턴스를 실행할 때 사용자가 직접 가용 영역을 선택하거나 AWS가 사용자를 위해 가용 영역을 선택



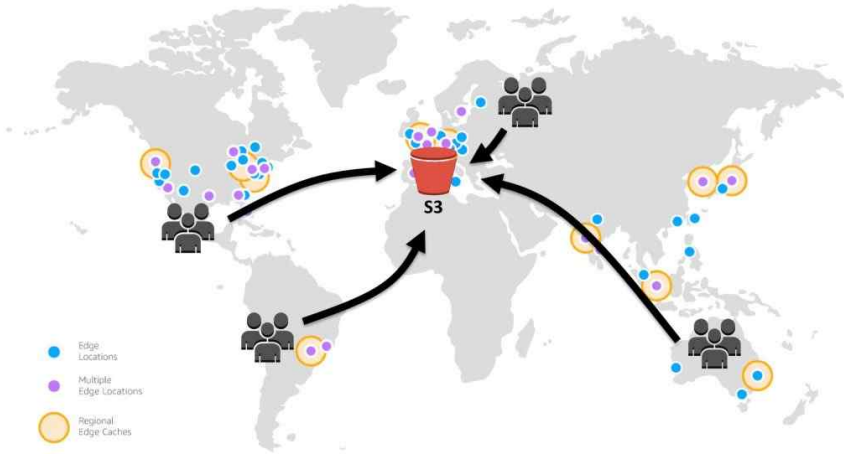
https://docs.aws.amazon.com/ko_kr/AWSEC2/latest/UserGuide/using-regions-availability-zones.html#concepts-regions

Global infrastructure

https://aws.amazon.com/ko/blogs/korea/amazon-s3-amazon-cloudfront-a-match-made-in-the-cloud/?nc1=b_rp

엣지 로케이션(Edge Location)

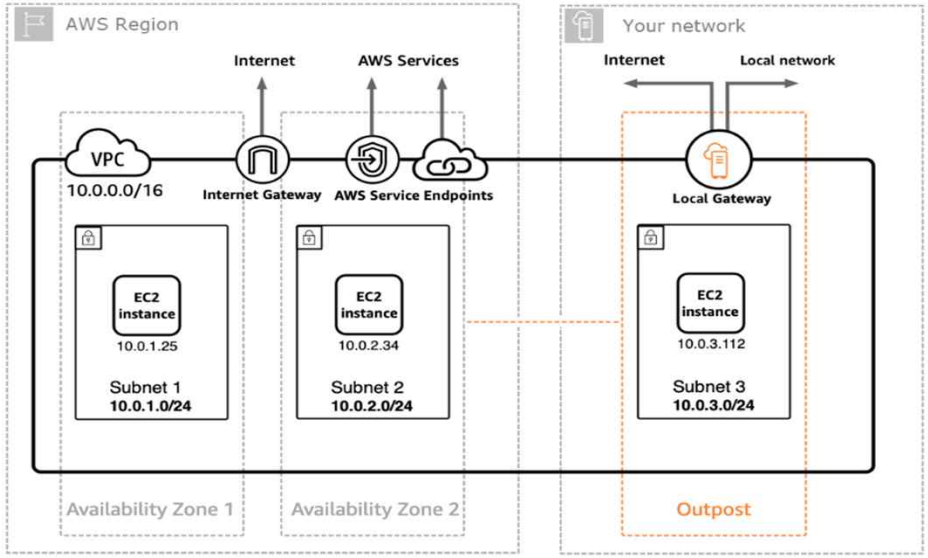
- Amazon CloudFront : 짧은 지연 시간과 빠른 전송 속도로 데이터, 동영상, 애플리케이션 및 API를 전 세계 고객에게 안전하게 전송하는 고속 콘텐츠 전송 네트워크(Contents Delevery Network , CDN) 서비스로 콘텐츠에 대한 사용자의 요청은 가장 가까운 엣지 로케이션으로 자동 라우팅 되어 빠르게 최종 사용자에게 전송
- 네트워크 및 애플리케이션 계층 DDoS 공격을 비롯해 여러 유형의 공격으로부터 보호하기 위해 **CloudFront, AWS Shield, AWS Web Application Firewall(WAF) Route 53, Lambda@Edge 컴퓨팅**과 완벽하게 통합되어 필드 수준 암호화 및 HTTPS 지원을 포함한 대부분의 고급 보안 기능을 제공
- 엣지 네트워킹 위치에 구성되며, AWS 네트워크 백본을 통해 전 세계적으로 확장 및 연결되어
  이를 통해 사용자에게 보다 안전하고 뛰어난 성능과 가용성을 보장 지원



Global infrastructure

Amazon Outposts

- AWS Outposts는 일관된 하이브리드 환경을 위해 동일한 AWS 인프라, AWS 서비스, API 및 도구를 모든 데이터 센터, 코-로케이션 공간, 온프레미스 시설에 제공하는 완전 관리형 서비스
- AWS가 사용자의 데이터 센터 내부에 정상적으로 작동하는 소형 리전을 기본적으로 구성
- 리전에서 제공되는 모든 AWS 서비스에 액세스하여 익숙한 AWS 서비스 및 도구를 사용해서 온프레미스 애플리케이션을 구축, 관리 및 확장



- https://www.wisen.co.kr/pages/blog/blog-detail.html?idx=11971

AWS 서비스로의 접근 방법

AWS의 모든 것은 API 호출을 통해서 이루어진다.

1) AWS Management Console

- 웹 브라우저 기반으로 시각적이며 이해하기 쉬운 형태
- 작업을 수행하는 프로세스를 단순화할 수 있는 마법사(Wizard) 및 자동화된 워크플로가 포함

2) AWS 명령줄 인터페이스 (Command Line Interface , CLI)

- 터미널을 사용하여 API 호출
- API 호출을 스크립트 또는 프로그래밍 할 수 있는 도구 사용 가능
- 스크립트를 사용하면 실행을 예약 하거나 다른 프로세스에서 호출되는 방식으로 자동 실행 가능

3) AWS 소프트웨어 개발 키트 (Software Develop Kit, SDK)

- 다양한 프로그래밍 언어를 통해 AWS 리소스와 상호 작용
- AWS를 사용하는 프로그램을 만들고 수동으로 작업할 필요 없음.

4) 기타 도구

- **AWS CloudFormation**
  - 코드형 인프라 도구로 CloudFormation 템플릿이라고 하는 JSON 또는 YAML 텍스트 기반 문서를 사용하여 선언적인 방식으로 스토리지, 데이터베이스, 분석, 기계 학습 등의 다양한 AWS 리소스를 정의
  - 리소스를 안전하고 반복 가능한 방식으로 프로비저닝
  - 스택을 관리할 때 수행해야 할 적절한 작업을 결정하고 오류를 감지하면 변경 사항을 자동으로 롤백

AWS 주요 서비스

컴퓨팅 서비스

✓ **Amazon EC2(Elastic Compute Cloud)**
- 다양한 형태의 타입과 서비스에 따라 적합한 사양을 선택 가능
- 사용량만큼 비용을 지불하는 컴퓨팅 서비스

✓ **Amazon Auto Scaling**
- 서버의 특정 조건에 따라 서버를 추가/삭제할 수 있게 해주는 서비스
- 서버 사용량이 많은 경우 추가생성, 사용하지 않는 경우 서버를 자동 삭제

✓ **Amazon Lightsail**
- 간단한 가상화 프라이빗 서버가 필요한 개발자에게 웹사이트와 웹 애플리케이션을 배포하고 관리하는 기능을 저렴함 비용으로 제공

✓ **Amazon WorkSpaces**
- 데스크탑 가상화 서비스
- 사내 PC를 가상화로 구성, 문서 및 데이터를 서버에서 보관/관리하는 서비스

AWS 주요 서비스

네트워크 서비스

✓ **Amazon Route 53**
- 가용성과 확장성이 우수한 클라우드 기반의 DNS 웹서비스
- 사용자의 요청을 AWS에서 실행되는 다양한 인프라에 효과적으로 연결
- 사용자를 AWS 외부의 인프라로 전달하는 서비스도 사용 가능

✓ **Amazon VPC(Virtual Private Cloud)**
- 가상 사설 네트워크 인프라를 클라우드 내에 생성/구성
- 네트워크를 이용한 접근제어, DHCP, VPN, IGW, Peering 제공

✓ **Amazon Direct Connect**
- 기존 On-Premise의 인프라와 AWS를 연결하는 전용선을 구성
- 낮은 지연시간으로 데이터 및 정보를 공유할 수 있게 하는 서비스

✓ **Amazon ELB(Elastic Load Balancing)**
- 기존에 사용하던 Load Balance 서비스와 비슷
- 웹서버 및 각종 서버에 사용량과 접속자가 많은 경우 트래픽에 대한 부하분산

AWS 주요 서비스

스토리지 서비스

✓ **Amazon S3**
- 여러가지 용도로 사용할 수 있는 범용적인 스토리지 서비스
- 데이터 보관 이외에도 정적 웹호스팅 및 다양한 형태의 서비스로 활용 가능

✓ **Amazon Glacier**
- 사용 빈도가 높지 않은 데이터를 저렴한 비용으로 장기 보관할 수 있게 해주는 서비스
- 가격이 저렴하고 무제한으로 데이터를 보관할 수 있는 장점

✓ **Amazon EBS(Elastic Block Storage)**
- 빠른 속도록 데이터를 저장, 보관할 수 있는 서비스
- 주로 서버에 디스크로 추가하여 데이터를 보관, 제공할 수 있음

✓ **Amazon Storage Gateway**
- On-premise에 있는 데이터를 클라우드로 저장 보관하기 위한 연결 Gateway 서비스

✓ **Amazon Snowball**
- Import/Export 서비스를 통해 대량의 데이터를 AWS로 이전할 때 네트워크로 전송하지 않고 디스크나 스토리지에 저장하여 물리적으로 전달

AWS 주요 서비스

데이터베이스 서비스

✓ **Amazon RDS(Relational Database Service)**

- 관계형 데이터베이스 서비스인 MySQL, MSSQL, Oracle, MariaDB, PostresSQL 등 RDBMS 서비스를 사용자가 직접 관리하지 않고 Amazon에서 제공하는 서비스를 이용하여 데이터베이스를 운영

✓ **Amazon DynamoDB**

- NoSQL 용 서비스로 대량의 데이터를 손쉽게 저장 가능
- 저장된 데이터를 추가 분석 서비스와 연계 활용할 수 있도록 확장할 수 있는 서비스

✓ **Amazon ElastiCache**

- In-Memory 기반의 Cache 서비스
- 빠른 속도를 필요로 하는 서비스와 연계하여 높은 응답속도와 신뢰성을 필요로 하는 서비스에 적합

 # 이외에도 40개가 넘는 고유 서비스를 포함하여 175개가 넘는 기능의 서비스를 제공하고 있으며, 지금 이 시간에도 매일 새로운 서비스를 만들고 있음

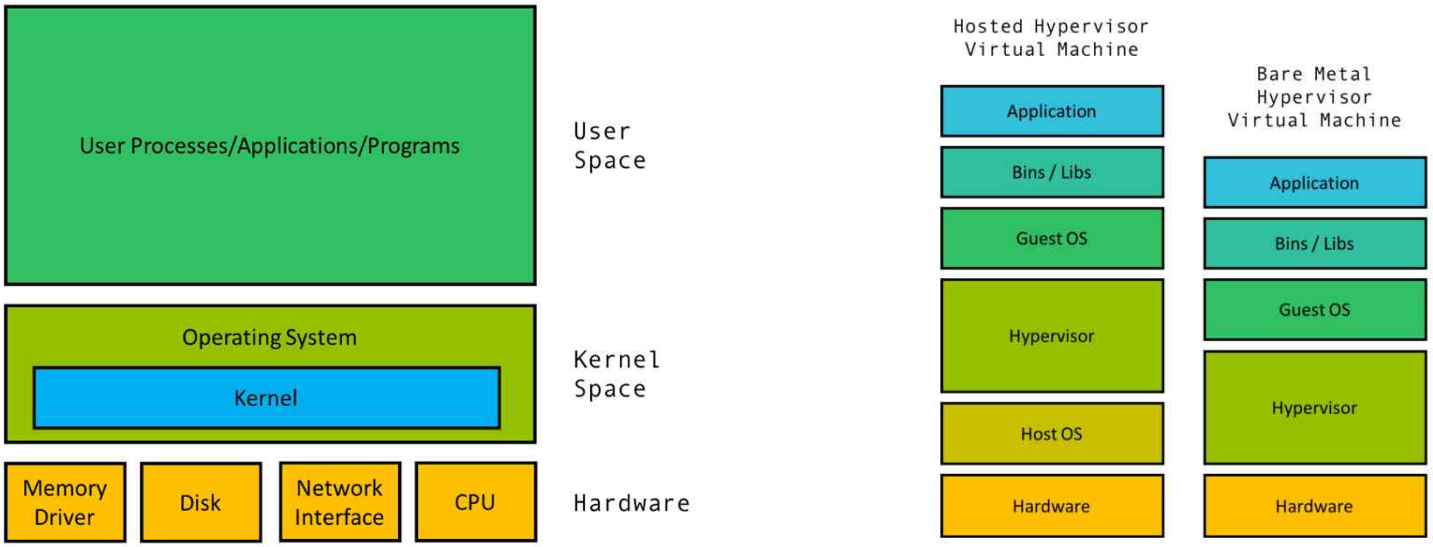
3.
Elastic Compute Cloud(EC2)

Amazon Elastic Compute Cloud(Amazon EC2)

가상 머신으로의 EC2

- EC2는 가상화 기술을 사용하여 AWS에서 관리하는 물리 서버 위에서 실행
- EC2인스턴스를 가동할 때는 전체 호스트를 직접 소유할 필요 없다. → 여러 인스턴스가 하나의 물리 서버 공유
- 하드웨어를 공유 = 멀티태넌시 (Multitenancy)
- 하이퍼바이저는 이러한 멀티 태넌시 조정하고 가상 머신을 서로 독립적으로 분리하는 책임 → AWS에서 관리

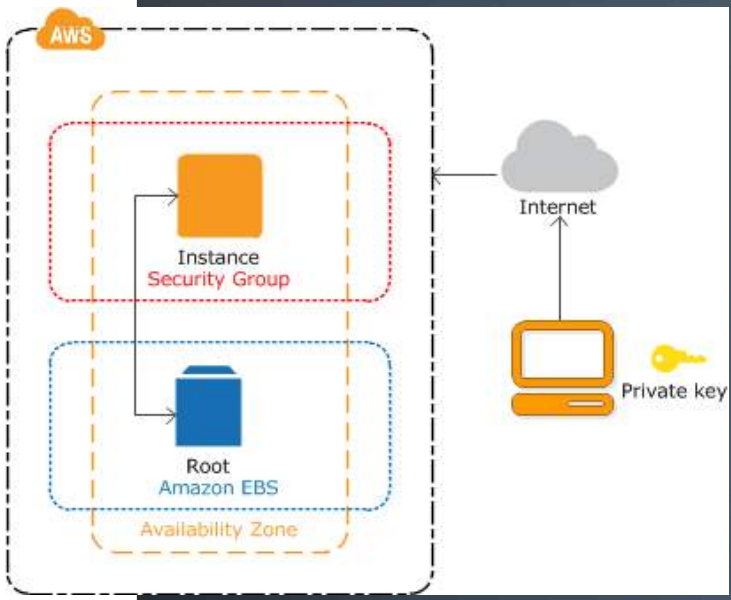
→ EC2는 탁월한 수준의 유연성과 제어 기능



https://www.freecodecamp.org/news/demystifying-containers-101-a-deep-dive-into-container-technology-for-beginners-d7b60d8511c1/

Amazon Elastic Compute Cloud(Amazon EC2)

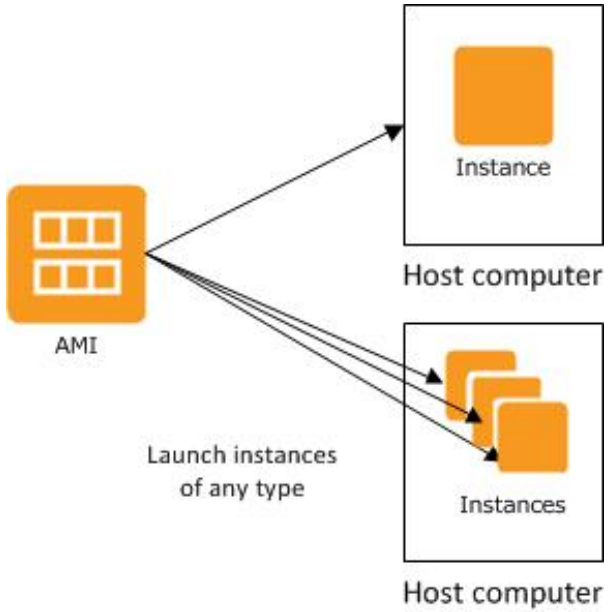
- **Amazon Elastic Compute Cloud (Amazon EC2)란?**
  - Amazon EC2는 AWS 클라우드에서 확장 가능 컴퓨팅 용량을 제공
  - 하드웨어에 선 투자할 필요가 없어 더 빠르게 애플리케이션을 개발하고 배포
  - AWS 데이터센터 구축 / 데이터 센터 보호 / 서버 구매 / 서버 설치 / 운영체제 설치
  - 사용자가 원하는 EC2인스턴스를 요청하면 몇 분 안에 바로 사용 가능
  - 워크 로드 실행이 완료되면 인스턴스 사용을 중지하고 더 이상 원하지 않으면 종료(Terminate, 삭제) 가능
  - 인스턴스가 실행 중 일 때 사용한 컴퓨팅 시간 만큼만 과금 (인스턴스가 중지 또는 종료된 상태에서는 과금 되지 않음)



Amazon Elastic Compute Cloud(Amazon EC2)

Amazon EC2 인스턴스와 AMI

- AMI (Amazon Machine Image) : 소프트웨어 구성이 기재된 템플릿 (예: 운영 체제, 애플리케이션 서버, 애플리케이션)
- 인스턴스(Instance) 는 AMI의 사본으로, 클라우드에서 실행되는 가상 서버
- 하나의 AMI로 여러 인스턴스 실행 가능
- 인스턴스는 중단하거나 최대 절전 모드로 전환하거나 종료할 때까지 또는 오류가 발생하지 않는 한 계속 실행
- 인스턴스가 실패하면 AMI에서 새로 실행 가능



https://docs.aws.amazon.com/ko_kr/AWSEC2/latest/UserGuide/ec2-instances-and-amis.html

Amazon Elastic Compute Cloud(Amazon EC2)

- **AWS Marketplace**
  - AWS 아키텍처에서 실행되는 타사 소프트웨어를 검색, 배포, 관리하는 단계를 간소화하는 큐레이트 된 디지털 카탈로그
  - 광범위한 솔루션을 빠르고 안전하게 배포하고 총 소유 비용을 절감
  - Marketplace에서 제공하는 타사 애플리케이션을 실행하는 데 필요한 기본 인프라를 빌드, 설치, 유지 관리할 필요 없이 필요할 경우 바로 수천의 소프트웨어 판매자로부터 신속하게 제품을 구매하고 사용할 수 있는 원 클릭 배포와 같은 옵션을 사용
  - 고객이 이미 소유하고 있는 연간 라이선스를 사용하도록 허용
  - 온 디맨드 및 종량제 옵션도 제공
  - 공급업체가 고객이 자사 제품을 실험하고 배울 수 있도록 무료 평가판 또는 Quick Start 플랜을 제공

Amazon Elastic Compute Cloud(Amazon EC2)

- **Amazon EC2 인스턴스 타입 및 유형**
  - 인스턴스 유형에 따라 인스턴스에 사용되는 호스트 컴퓨터의 하드웨어가 기본적으로 결정
  - 각 인스턴스 유형은 서로 다른 컴퓨팅 및 메모리 기능을 제공
  - 각 Amazon EC2 인스턴스 유형은 인스턴스 패밀리 형태로 분류되어 특정 유형의 작업에 최적화될 수 있도록 구성

인스턴스 타입	설 명
범용	• 균형 있는 컴퓨팅, 메모리 및 네트워킹 리소스를 제공하며, 다양한 여러 워크로드에 사용 • 이 인스턴스는 웹 서버 및 코드 리포지토리와 같이 이러한 리소스를 동등한 비율로 사용하는 애플리케이션에 적합
컴퓨팅 최적화	• 컴퓨팅 최적화 인스턴스는 고성능 프로세서를 활용하는 컴퓨팅 집약적인 애플리케이션에 적합 • 배치 처리 워크로드, 미디어 트랜스코딩, 고성능 웹 서버, HPC(고성능 컴퓨팅), 과학적 모델링, 전용 게임 서버 및 광고 서버 엔진, 기계 학습 추론 및 기타 컴퓨팅 집약적인 애플리케이션에 매우 적합
메모리 최적화	• 메모리에서 대규모 데이터 세트를 처리하는 워크로드를 위한 빠른 성능을 제공하기 위해 설계 • 고성능 데이터베이스일 수도 있고 방대한 양의 비정형 데이터의 실시간 처리가 필요한 워크로드
가속화된 컴퓨팅 최적화	• 하드웨어 액셀러레이터 또는 코프로세서를 사용하여 부동 소수점 수 계산이나 그래픽 처리, 데이터 패턴 일치 등의 기능을 CPU에서 실행되는 소프트웨어보다 훨씬 효율적으로 수행 • 그래픽 애플리케이션, 게임 스트리밍, 애플리케이션 스트리밍에 유리
스토리지 최적화	• 대규모 데이터 세트에 대한 순차적 읽기 및 쓰기 액세스가 많이 필요한 워크로드를 위해 설계 • 적합한 워크로드의 예로는 분산 파일 시스템, 데이터 웨어하우징 애플리케이션, 고빈도 온라인 트랜잭션 처리(OLTP) 시스템 등 • 스토리지 최적화 인스턴스는 지연 시간이 짧은 임의 IOPS를 애플리케이션에 제공하도록 설계

Amazon Elastic Compute Cloud(Amazon EC2)

Amazon EC2 요금

요금 선택	설 명
온디맨드 인스턴스	• 시간당 또는 초당 컴퓨팅 파워에 대한 비용을 지불 • 장기 약정이나 선결제 없음 • 애플리케이션 수요에 따라 컴퓨팅 파워를 변경가능 하며 사용한 인스턴스에 대해 지정된 시간당 요금만 지불 • 중단할 수 없는 불규칙한 단기 워크로드가 있는 애플리케이션에 매우 적합 • 온디맨드 인스턴스는 1년 이상 지속되는 워크로드에는 권장하지 않는다. → 이러한 워크로드는 예약 인스턴스를 사용하면 비용 절감 효과가 더 크기 때문
예약 인스턴스	• 예약 인스턴스는 1년 또는 3년 약정으로, 온디맨드 인스턴스 요금과 비교하여 할인 혜택(최대 75%)을 제공 • 특정 가용 영역에 지정하면 용량 예약이 제공되므로 필요할 때 예약한 인스턴스를 반드시 시작 가능 • 수요가 꾸준한 애플리케이션, 예약 용량이 필요할 수 있는 애플리케이션 적합
Amazon EC2 Saving Plans	• Savings Plans에서는 1년 또는 3년 동안 특정 양의 컴퓨팅 파워를 사용하기로 약정(시간당 요금을 기준으로 측정)하면, AWS 컴퓨팅 사용량 요금을 최대 72% 절감할 수 있는 유연한 요금 모델 • AWS Fargate 및 AWS Lambda 사용량에도 적용 • 약정을 초과한 사용량에 대해서는 일반 온디맨드 요금이 부과 → AWS Cost Explorer
스팟 인스턴스	• 시작 및 종료 시간이 자유롭거나 중단을 견딜 수 있는 워크로드에 적합 • 스팟 인스턴스는 미사용 Amazon EC2 컴퓨팅 용량을 사용하며 온디맨드 요금의 최대 90%까지 비용을 절감 • 스팟 인스턴스를 시작한 후 용량을 더 이상 사용할 수 없거나 스팟 인스턴스에 대한 수요가 늘면 인스턴스가 중단 • 시작 및 종료 시간이 자유로운 애플리케이션, 컴퓨팅 가격이 매우 저렴해야만 수익이 나는 애플리케이션
전용 호스트	• 전용 호스트는 사용자 전용의 Amazon EC2 인스턴스 용량을 갖춘 물리적 서버 • Amazon EC2 옵션 중에서 전용 호스트가 가장 비용이 많이 듭니다.

Amazon Elastic Compute Cloud (Amazon EC2)

AWS EC2 상태

- Pending: 인스턴스가 구동하기 위해 준비중인 상태, 요금 미청구
- Running: 인스턴스를 실행하고 사용할 준비가 된 상태, 요금 청구
- stopping: 인스턴스가 중지 모드로 전환되려는 상태, 요금 미 청구
- Shutting-down: 인스턴스가 종료할 준비중인 상태, 요금 미청구
- Terminated: 인스턴스가 종료된 상태, 요금 미청구

Elastic IP

- EC2에 설정되는 네트워크 인터페이스의 공인 IP
- EC2가 기본적으로 갖는 Public IP와 Private IP 등과는 구분해야 함
- Elastic IP를 사용하면 EC2로 하여금 중지되었다가 다시 시작하더라도 고정된 공인 IP를 사용하게 할 수 있음
- Elastic IP는 계정 내 리전당 최대 5개까지 보유 가능하며 그 이상 필요 시 AWS에 요청해야 함
- 프리티어의 경우, 사용 중이 아닐 땐 요금이 부과됨

https://aws.amazon.com/ko/ec2/pricing/

Amazon Elastic Compute Cloud (Amazon EC2)

Key Pair

- EC2는 SSH 접속 시 Key Pair라고 하는 퍼블릭 키 암호화 기법을 사용하여 접속하며 로그인 정보를 암호화 및 해독함
- Key Pair 분실 시 접속 불가
- 또한 접속 시 OS별로 접속 name이 다름(Linux: ec2-user 등)

Batch Group

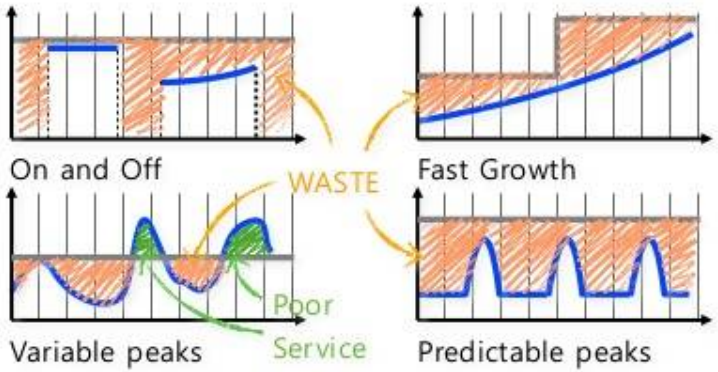
- 클러스터: 단일 AZ 내에 있는 인스턴스의 논리적 그룹. 인스턴스를 AZ 내에서 근접하게 배치함. 결합된 노드간 낮은 지연 시간의 네트워크 달성 가능
- 파티션: 인스턴스가 담긴 그룹을 논리 세그먼트로 나누어 각 파티션에 배치함. 최대 7개의 파티션을 가질 수 있으며, 각 파티션은 자체 랙 세트를 보유하고 자체 네트워크와 전원을 보유함. 애플리케이션에 대한 상관 관계가 있는 하드웨어 장애 가능성을 줄이는 데 도움
- 분산: 파티션이 논리 세그먼트로 분리된 인스턴스 그룹인 것과 달리 분산은 '인스턴스' 개체 하나가 자체 랙에 분산 배치됨. AZ 당 최대 7개의 인스턴스 배치 가능

Amazon Elastic Compute Cloud (Amazon EC2)c

https://aws.amazon.com/ko/ec2/pricing/

Amazon Elastic Compute Cloud (Amazon EC2)

전통적 IT 리소스 사용 패턴



- **Amazon EC2**
확장성(scalability) /
탄력성(elastic ability)

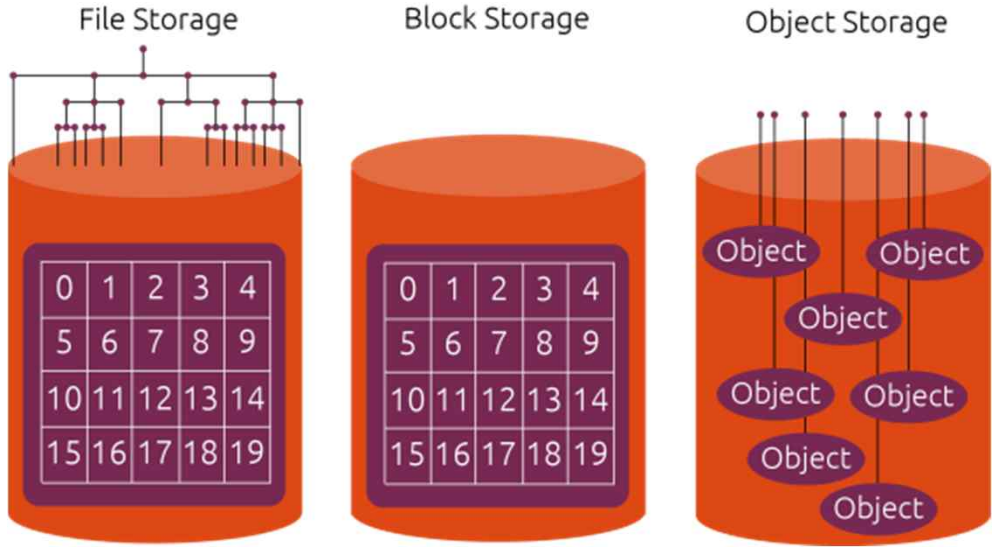
- 확장성을 위해서는 필요한 리소스만으로 시작하고 확장 및 축소를 통해 수요 변화에 자동으로 대응하도록 아키텍처를 설계

- → 이러한 리소스의 탄력적인 확장을 자동으로 수행 → Amazon Ec2 Auto Scaling

4. Elastic Block Storage(EBS)

블록 수준 스토리지

- **블록 스토리지 란?**
  - 블록 스토리지는 파일(디스크의 블록에 저장되는 일련의 바이트로 구성) 의 형태로 데이터를 저장
  - Block 단위로 I/O를 진행
  - 파일이 업데이트되면 변경된 부분의 블록만 업데이트
- → 데이터베이스, 엔터프라이즈 소프트웨어 등에서 효율적인 I/O 를 제공하는 스토리지 유형



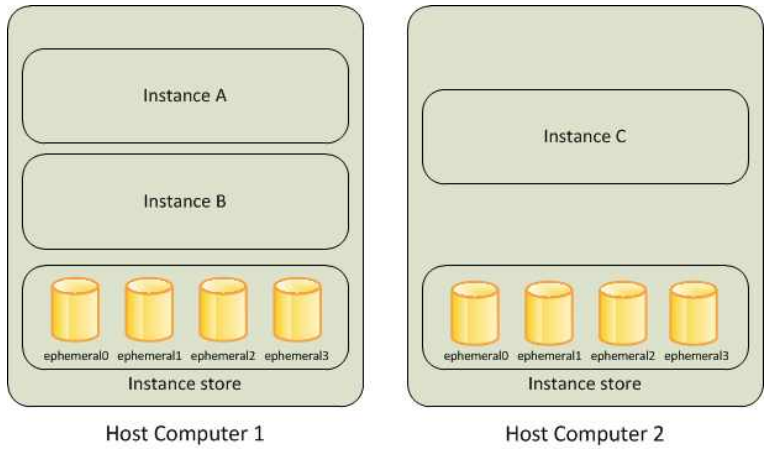
EC2 인스턴스 스토어 볼륨(EC2 Instance store volume)

EC2 인스턴스 스토어 볼륨(EC2 Instance store volume)이란?

- Amazon EC2 인스턴스에 임시 블록 수준 스토리지를 제공
- EC2 인스턴스 종류에 따라 지원 가능
- EC2 인스턴스가 실행되는 물리 호스트에 연결된 스토리지 공간을 사용

→EC2 인스턴스가 재가동 시 늘 동일한 호스트에 배치되지 않기 때문에 인스턴스와 동일한 수명을 가진다.

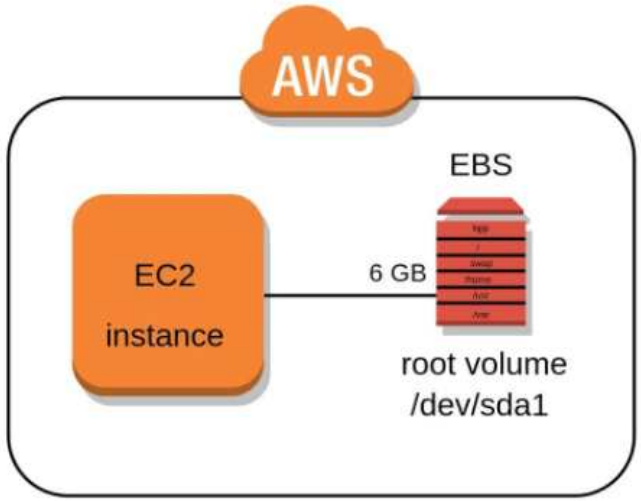
→데이터 휘발성 (EC2 인스턴스 중지하거나 종료되면 모든 데이터 삭제)



Amazon Elastic Block Store (EBS)

Amazon Elastic Block Store (EBS)란?

- Amazon EC2 인스턴스에서 사용할 수 있는 블록 수준 스토리지 볼륨을 제공하는 서비스
- 애플리케이션의 기본 스토리지로 쓰거나 시스템 드라이브용으로 쓰기 적합
- 인스턴스 생성 시 루트 디바이스 볼륨이 생성되며 사용 중에는 언 마운트 할 수 없음
- 또한 인스턴스는 여러 볼륨을 마운트 할 수 있고, 추가 볼륨에 대해서는 사용 중 이라도 마운트/언 마운트가 가능
- 볼륨은 여러 인스턴스에 마운트 할 수 있음
- EBS는 동일한 AZ 내에서 다른 인스턴스에 연결 가능함
- 인스턴스 스토어 볼륨과는 달리 EBS 기반 인스턴스는 중지/재시작이 가능함
- 사용중인 EBS더라도 볼륨 유형과 사이즈를 변경할 수 있음 (사이즈 축소는 불가)



Amazon Elastic Block Store (EBS)

- **EBS 볼륨 유형**
  - Amazon EBS는 다음의 볼륨 유형을 제공하고 이러한 볼륨 유형은 성능 특성과 가격이 다르므로 애플리케이션의 필요에 맞게 스토리지 성능과 비용을 조정

    - SSD(Solid-State Drive) — 주요 성능 특성이 IOPS인 작은 I/O 크기의 읽기/쓰기 작업을 자주 처리하는 트랜잭션 워크로드에 최적화

    - HDD(Hard Disk Drive) — 주요 성능 특성이 처리량인 대규모 스트리밍 워크로드에 최적화

    - 이전 세대 — 데이터를 자주 액세스하지 않고 성능이 중요하지 않은 소규모 데이터 세트가 있는 워크로드에 사용할 수 있는 하드 디스크 드라이브. 대신 현재 세대의 볼륨 유형을 고려하는 것이 좋음

  - 인스턴스 구성, I/O 특성 및 워크로드 요구량 등 여러 가지 요인이 EBS 볼륨의 성능에 영향을 미칠 수 있음.
  - EBS 볼륨에서 프로 비저닝 된 IOPS를 완전히 사용하려면 EBS 최적화 인스턴스를 사용

Amazon Elastic Block Store (EBS)

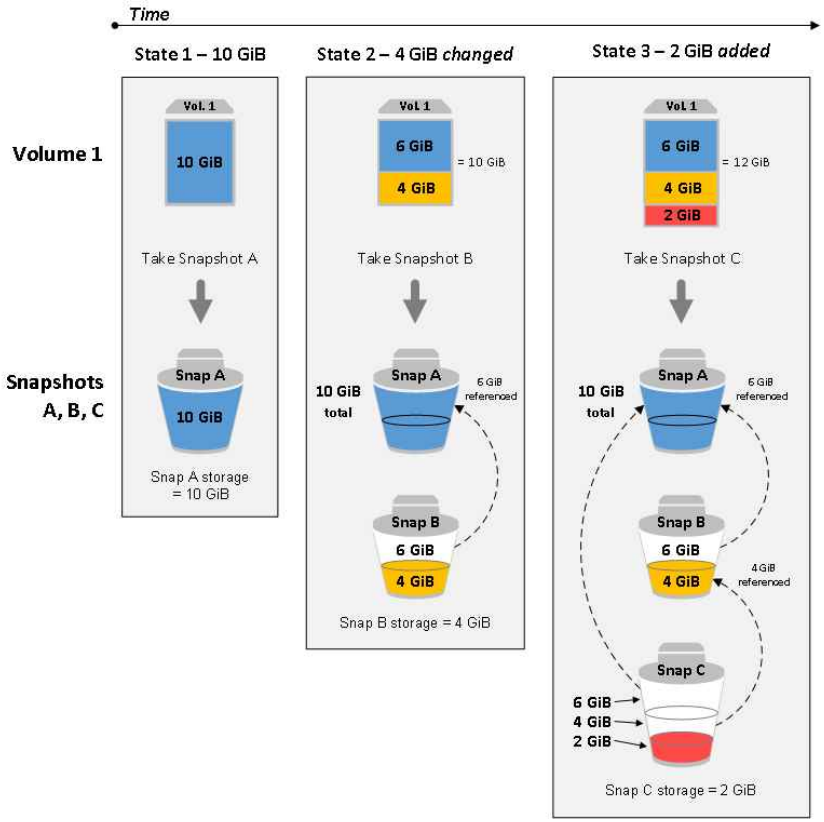
Amazon EBS 스냅샷(snapshot)

- 스냅샷을 이용하여 EBS 볼륨의 데이터를 S3에 저장할 수 있음
- 증분식 백업이기 때문에 스냅샷은 하나이며, 마지막 스냅샷 이후 변경된 블록만 추가적으로 저장됨
- 각 스냅샷에는 스냅샷을 만든 시점의 데이터를 새 EBS 볼륨에 복원하는데 필요한 정보가 들어 있음
- EBS의 스냅샷은 S3에 저장되며 S3에 저장된 스냅샷으로 EBS 볼륨 복구 가능
- 암호화된 볼륨의 스냅샷은 자동으로 암호화 됨
- 암호화된 스냅샷에서 생성되는 볼륨은 자동으로 암호화 됨
- 암호화되지 않은 EBS에서 암호화된 스냅샷은 생성할 수 없음

Amazon Elastic Block Store (EBS)

EBS 암호화

- EBS 를 암호화함으로써 부팅 및 데이터 볼륨을 모두 암호화 할 수 있음
- 다음 유형의 데이터가 암호화 됨
  - 볼륨 내부 데이터
  - 볼륨과 인스턴스 사이에서 이동하는 데이터
  - 스냅샷에서 생성된 모든 볼륨(스냅샷이 암호화되면 해당 스냅샷에서 생성된 볼륨은 자동 암호화)
- AES-256 알고리즘 사용

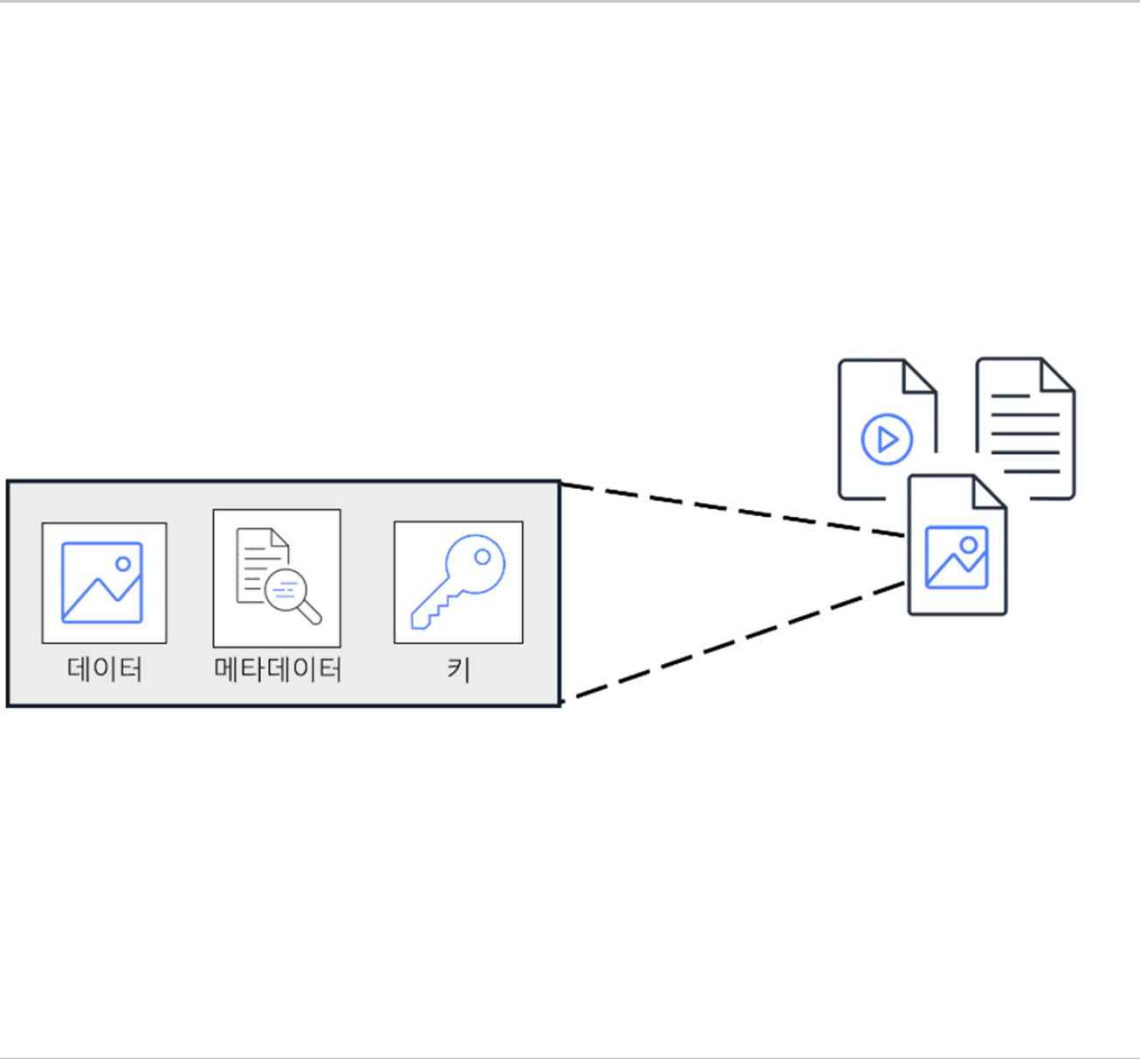


https://docs.aws.amazon.com/ko_kr/AWSEC2/latest/UserGuide/EBSSnapshots.html

5. Simple Storage Service(S3)

Amazon Simple Storage Service (S3)

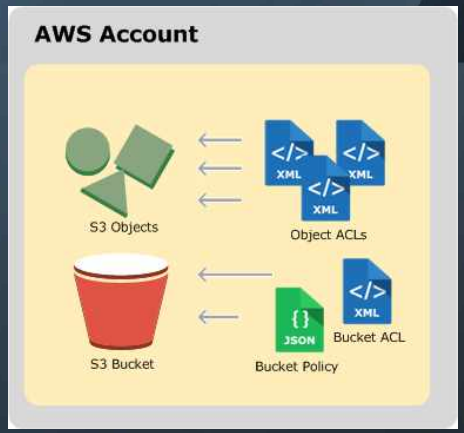
- **객체 스토리지란?**
  - 객체 스토리지에서 각 객체는 데이터, 메타데이터, 키로 구성
  - 데이터는 이미지, 동영상, 텍스트 문서 또는 기타 유형의 파일
  - 메타데이터에는 데이터의 내용, 사용 방법, 객체 크기 등에 대한 정보가 포함
  - 객체의 키는 고유한 식별자



Amazon Simple Storage Service (S3)

- **Amazon Simple Storage Service (S3) 란?**
  - 웹서비스 인터페이스(HTTP)를 이용하여 웹에서 언제 어디서나 원하는 양의 데이터를 저장하고 검색할 수 있는 스토리지
  - 버킷(Bucket)과 객체(Object)로 나뉘며, 저장하고자 하는 모든 요소는 하나의 객체로 저장되고, 오브젝트를 담는 곳이 바로 버킷
  - S3 자체는 글로벌 서비스이지만 버킷을 생성할 때에는 리전을 선택해야 함
  - 객체는 객체 데이터와 메타 데이터로 나뉘며, 각자의 고유한 URL을 가지며 해당 URL로 접속 가능

Amazon Simple Storage Service (S3)



버킷(Bucket)의 정의와 특징

- 객체를 담고 있는 구성요소
- 크기는 무제한이며, 리전을 지정하여 버킷을 생성해야 함
- 버킷의 이름은 반드시 고유해야 하며 중복될 수 없음
- 한 번 설정된 버킷의 이름은 다른 계정에서 사용할 수 없음

객체(Object)의 정의와 특징

- S3에 업로드 되는 1개의 데이터를 객체라 함
- 키, 버전ID, 값, 메타데이터 등으로 구성됨
- 객체 하나의 최소 크기는 1byte ~ 5TB
- 스토리지 클래스, 암호화, 태그, 메타데이터, 객체 잠금 설정 가능
- 객체의 크기가 매우 클 경우 멀티파트 업로드를 통해 신속하게 업로드 가능

Amazon Simple Storage Service (S3)

Amazon S3 스토리지 클래스_객체의 접근빈도 및 저장기간에 따라 결정되는 객체의 특성

스토리지 클래스	설 명
S3 Standard	• 99.999999999% 내구성 • 데이터가 3개 이상의 시설에 저장 →두 곳에 동시에 문제 발생해도 데이터 유지 • 정적 웹사이트 호스팅 • 접근 빈도가 높은 데이터 저장 • 다른 스토리지 클래스보다 비용이 비싸다.$$
S3 Standard-Infrequent Access (S3 Standard-IA)	• S3 최소 3개의 가용 영역에 데이터를 저장 • S3 Standard보다 스토리지 가격은 더 저렴하고 검색 가격은 더 비쌈 → 자주 액세스하지 않는 데이터에 이상적
S3 One Zone-Infrequent Access (S3 One Zone-IA)	• 단일 가용 영역에 데이터를 저장 • S3 Standard-IA보다 낮은 스토리지 가격 • 스토리지 비용을 절감하려는 경우 • 가용 영역 장애가 발생할 경우 데이터를 손쉽게 재현할 수 있는 경우
S3 Intelligent-Tiering	• 액세스 패턴을 알 수 없거나 자주 변화하는 데이터에 이상적 • 객체에 30일 연속 객체에 액세스하지 않으면 Amazon S3→ S3 Standard-IA 이동 • 객체에 액세스하면 Amazon S3→ S3 Standard 이동 • 객체당 소량의 월별 모니터링 및 자동화 요금을 부과
S3 Glacier	• 데이터 보관용으로 설계된 저비용 스토리지 • 객체를 몇 분에서 몇 시간 이내에 검색
S3 Glacier Deep Archive	• 보관에 이상적인 가장 저렴한 객체 스토리지 클래스 • 객체를 12시간 이내에 검색

Amazon Simple Storage Service (S3)

Amazon S3 스토리지 클래스 전반에 걸친 성능

	S3 Standard	S3 Intelligent-Tiering*	S3 스탠다드-IA	S3 One Zone-IA†	S3 Glacier	S3 Glacier Deep Archive
내구성을 위한 설계	99.999999999% (11개의 9)	99.999999999% (11개의 9)	99.999999999% (11개의 9)	99.999999999% (11개의 9)	99.999999999% (11개의 9)	99.999999999% (11개의 9)
가용성을 위한 설계	99.99%	99.9%	99.9%	99.5%	99.99%	99.99%
가용성 SLA	99.9%	99%	99%	99%	99.9%	99.9%
가용 영역	≥3	≥3	≥3	1	≥3	≥3
객체당 최소 용량 요금	해당 사항 없음	해당 사항 없음	128KB	128KB	40KB	40KB
최소 스토리지 기간 요금	해당 사항 없음	30일	30일	30일	90일	180일
검색 요금	해당 사항 없음	해당 사항 없음	검색한 GB당	검색한 GB당	검색한 GB당	검색한 GB당
첫 번째 바이트 지연 시간	밀리초	밀리초	밀리초	밀리초	분 또는 시간 선택	시간 선택
스토리지 유형	객체	객체	객체	객체	객체	객체
수명 주기 전환	예	예	예	예	예	예

Amazon Simple Storage Service (S3)

멀티 파트 업로드

- 오브젝트의 크기가 클 경우, 이를 조각 내어 병렬적으로 처리하여 처리량을 개선하는 방법
- 보통 객체의 크기가 100MB이상일 경우 사용하는 것을 권고하며 최대 가능 크기는 5TB
- 조각의 개수는 최대 1만개까지 가능하며, 조각의 크기는 대개 5MB ~ 5GB 정도임
- GUI가 아닌 CLI를 통한 멀티 파트 업로드 API로 실행 가능

버전관리

- 동일한 객체에 대해 여러 버전을 가질 수 있도록 하는 기능
- 이미 S3에 존재하는 객체에 내용을 변경하여 업데이트할 경우 기존 버전이 사라지지 않고 이전 버전으로 존재할 수 있음
- 최신 버전과 이전 버전 모두 확인 가능
- 기존 객체와 동일한 이름으로 파일을 업로드 하면 덮어씀
  (항상 최신버전을 유지하지만 기존 버전이 삭제 됨)
- 객체를 삭제하더라도 바로 삭제되는 것이 아닌 삭제 마커를 붙여 다시 복구할 수 있는 기능 제공

Amazon Simple Storage Service (S3)

수명주기 관리

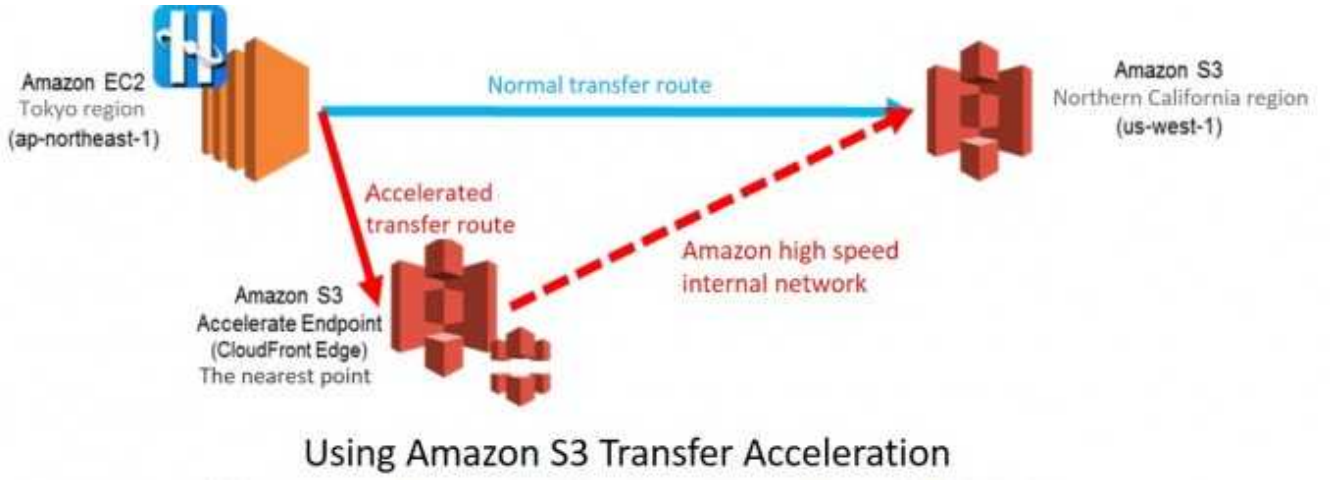
- S3에 있는 오브젝트를 일정 시간 후에 다른 타입으로 변경하는 것을 의미
- 예를 들어, 자주 사용하던 Standard Type의 오브젝트를 60일 이후 더 이상 사용하지 않을 경우 Glacier Type으로 변환하여 비용을 줄일 수 있음
- 현재 버전과 이전 버전에 대해 설정 가능
- 또한 완료되지 않은 멀티파트 업로드와 버전관리가 적용된 오브젝트의 삭제 마커를 정리할 수 있음
- 더 이상 사용하지 않은 오브젝트에 대해 만료기간을 설정하여 정한 기간 후 삭제하도록 설정 가능(만료 기간 설정)
- Standard-IA와 One Zone IA는 보관 후 30일 이후 이전 가능

정적 웹 사이트 호스팅

- S3로 하여금 정적 페이지를 제공할 수 있는 호스팅 기능을 제공하게 하는 것
- 활성화 후 특정 페이지를 지정하면 S3 접속 시 해당 페이지를 띄움
- 굳이 서버를 이용하지 않고 S3를 이용하여 웹 호스팅을 할 수 있음
- 호스팅 뿐만 아니라 요청을 다른 버킷 혹은 도메인으로 리디렉션 가능

Amazon Simple Storage Service (S3)

- **전송 속도 향상(Transfer Acceleration)**
  - CloudFront 의 Edge Location을 이용하여 파일 업로드를 보다 빠르게 하는 기능
  - S3로 직접 업로드 하는 것이 아닌 가장 가까운 Edge Location으로 전송하고 아마존 백본 네트워크를 통해 S3로 도달



Using Amazon S3 Transfer Acceleration

http://s3-accelerate-speedtest.s3-accelerate.amazonaws.com/en/accelerate-speed-comparsion.html

Amazon Simple Storage Service (S3)

- **Amazon S3는 다음과 같은 경우 사용**
  - 한 번 쓰고 여러 번 읽어야 하는 경우
  - 컨텐츠가 다양하고, 데이터 양이 지속적으로 증가하는 경우
  - 사용자가 많고, 데이터 접근이 일시적으로 급증하는 경우

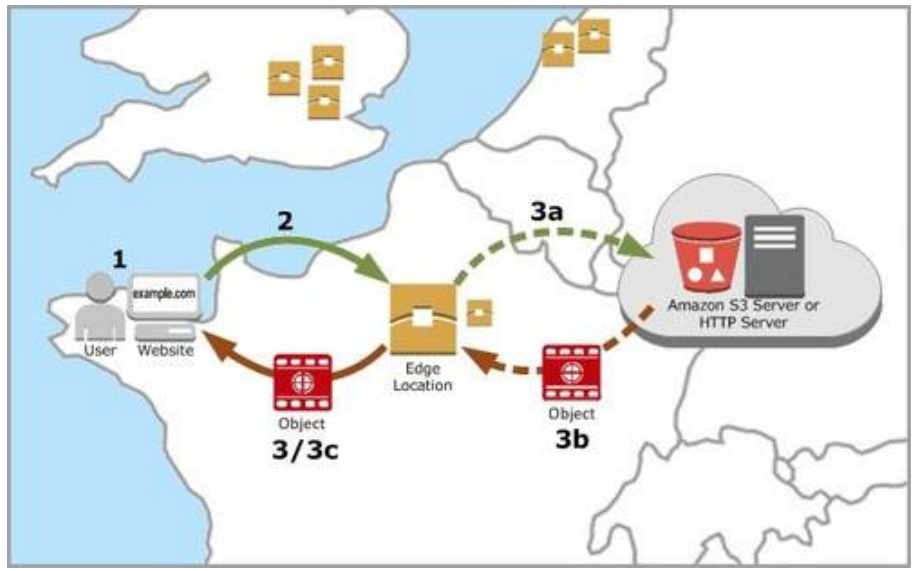
- **Amazon S3가 적합하지 않은 경우**
  - 여러 번 쓰기 작업을 해야 하는 데이터
  - 블록 스토리지가 필요한 경우

Amazon CloudFront

- **Amazon CloudFront 란?**
  - CDN(Content Delivery Network)
  - 간단히 HTTP/HTTPS를 이용하여 S3 및 ELB, EC2, 외부서버 등을 캐시하고 빠른 속도로 컨텐츠를 전달하는 캐시서버
  - 전 세계 각지에 퍼져 있는 Edge Location의 주변 Origin Server(EC2 등)의 컨텐츠를 Edge Location에 캐싱 하고 각 Edge Location간 공유를 통해 컨텐츠를 전달
  - Distribution은 Edge Location의 집합을 의미
  - 각 Edge Location간에는 아마존의 백본 네트워크를 통하기 때문에 매우 빠른 속도로 전달 가능
  - S3, ELB, EC2 등의 AWS 서비스 뿐만 아니라, 외부의 서버도 캐싱 가능(Custom Origin)
  - TTL을 조정해서 캐시 주기를 통제할 수 있음

Amazon CloudFront

- **컨텐츠 제공 방법**
  - 사용자가 웹 사이트 혹은 앱에 액세스하고 이미지 혹은 HTML 파일을 요청(정적 데이터)
  - DNS가 요청을 최적으로 서비스 할 수 있는 Cloudfront Edge Location으로 요청을 라우팅
  - Edge Location에서 해당 캐시에 요청된 파일이 있는지 확인하고 없으면 오리진 서버에 요청하여 확보 후 전달, 그리고 캐시 적재



https://quizlet.com/330487926/aws-cloudfront-flash-cards/

Amazon CloudFront

OAI (Origin Access Identity)

- S3를 오리진 서버로 사용 시, Cloudfront를 제외하고 다른 경로로 S3 접근하는 것을 막는 방법
- OAI를 설정하게 되면 각각의 Distribution이 별도의 Identity를 갖게 되며, S3의 버킷 정책을 수동 혹은 자동으로 수정할 수 있음

Presigned URL

- 인증된 사용자만이 해당 Distribution을 사용할 수 있도록 제어하는 기능
- 만료 날짜 및 시간까지 설정 가능
- CloudFront 설정 시 Presigned URL 사용과 CloudFront Key Pair 를 계정의 보안자격증명에서 생성해야 함
- 이를 조합하여 URL 서명을 생성하고 해당 URL을 통해 CloudFront에 접근할 수 있음

Amazon Elastic File System(EFS)

Amazon Elastic File System(EFS)이란?

- 네트워크 파일 시스템(NFS v4)를 사용하는 파일 스토리지 서비스
- VPC 내에서 생성되며, 파일 시스템 인터페이스를 통해 EC2에 액세스
- 수천 개의 EC2에서 동시에 액세스 가능하며, 탄력적으로 파일을 추가하고 삭제함에 따라 자동으로 Auto Scaling 가능, 즉 미리 크기를 프로비저닝 할 필요가 없음
- 페타 바이트단위 데이터 까지로 확장 가능
- 최대 1천개의 파일 시스템 생성 가능
- Amazon EFS는 리전 수준의 서비스로 여러 가용 영역에 걸쳐 데이터를 저장 가능

스토리지 클래스

- Standard Class : 자주 액세스하는 파일을 저장하는데 사용하는 클래스
- Infrequent Access(IA) Class : 저장기간이 길지만 자주 액세스하지 않는 파일을 저장하기 위한 클래스

Amazon Elastic File System(EFS)

가용성

- 여러 가용영역에서 액세스 가능
- 여러 가용영역에 중복 저장되기 때문에 하나의 가용영역이 파괴되더라도 다른 AZ에서 서비스 제공 가능
- IPSEC VPN 또는 Direct Connect를 통해 On-premise에서 접속 가능

성능 모드 / 처리량 모드

- 성능 모드에 있어서 대부분의 파일시스템에 Bursting Mode를 권장하지만 처리량이 많을 경우, Provisioned Mode를 권장
- 처리량 모드에 있어서 액세스하는 EC2가 매우 많을 경우, MAX I/O mode를 사용하는 것이 바람직하며(지연시간이 약간 길어 짐) 그 밖의 경우 General Mode를 사용하는 것을 권장

수명 주기 관리

- 위에서 언급한 스토리지 클래스와 관련된 서비스로 사용자가 지정한 일정기간동안 액세스하지 않은 파일을 Infrequent Access Class로 옮길 수 있도록 하는 기능

파일시스템 정책

- EFS를 사용하는 모든 NFS 클라이언트들에게 적용되는 IAM 리소스 정책
- 전송 중 암호화, 루트 액세스 비활성화, 읽기 전용 액세스 등 설정 가능

Amazon Elastic File System(EFS)

https://aws.amazon.com/ko/efs/when-to-choose-efs/

Comparing Amazon Cloud Storage

		File EFS	Object S3	Block EBS
퍼포먼스	작업 별 대기 시간	낮음, 일관됨	낮음, 여러 요청 타입 처리, CloudFront와 통합	가장 낮음, 일관됨
	처리량 크기	초당 GBs	초당 GBs	초당 GB
특성	가용성 / 내구성 (availability, durability)	여러 AZ에 중복 저장	여러 AZ에 중복 저장	단일 AZ에 중복 저장
	엑세스	1개~수천개의 EC2인스턴스 / on-premise 서버 / 여러 AZ에서 동시 접근	웹을 통한 수백만 개의 연결	하나의 AZ에 속한 하나의 EC2 인 스턴스
	사용 케이스	웹 서비스 및 컨텐츠 관리 엔터프라이즈 어플리케이션 홈 디렉토리 데이터베이스 백업 개발자 도구 컨테이너 스토리지 빅데이터 분석	웹 서비스 및 컨텐츠 관리 미디어 및 엔터테인먼트 백업 빅데이터분석 데이터 레이크	부팅 볼륨 트랜잭션 및 NoSQL 데이터베이스 데이터 웨어하우징 및 ETL

6. Relational Database Service(RDS)

관계형 데이터베이스와 비관계형 데이터베이스

[관계형 데이터베이스와 비관계형 데이터베이스(SQL vs. NoSQL)]

SQL의 장점

- 명확하게 정의 된 스키마, 데이터 무결성 보장
- 관계를 통해 각 데이터를 중복없이 한 번만 저장

SQL의 단점

- 상대적으로 덜 유연하며, 데이터 스키마는 미리 알고 계획해야 함 (나중에 수정하는 것이 어렵거나 불가능)
- JOIN문이 많은 매우 복잡한 쿼리가 만들어 질 수 있음
- 수평 확장이 어렵고, 보통 수직 확장만 가능
  즉 어느 시점에서 처리량/처리 능력과 관련하여 약간의 성장 한계에 직면하게 될 수 있음

관계형 데이터베이스와 비관계형 데이터베이스

[관계형 데이터베이스와 비관계형 데이터베이스(SQL vs. NoSQL)]

NoSQL의 장점

- 스키마가 없기때문에, 유연성이 높음
  즉, 저장된 데이터를 언제든지 조정하고 새로운 "필드" 추가 가능
- 데이터는 애플리케이션에 필요한 형식으로 저장
  이렇게 하면 데이터를 가져오는 속도가 빨라 짐
- 수직 및 수평 확장이 가능하므로 데이터베이스가 애플리케이션에서 발생시키는 모든 읽기 / 쓰기 요청 처리 가능

NoSQL의 단점

- 유연성 때문에, 데이터 구조 결정을 늦어질 수 있음(바로 계획, 결정해야하는 것이 아니기 때문)
- 복사된 데이터가 변경되면 여러 콜렉션 과 문서를 수정해야 함

Amazon Relational Database Service (RDS)

Relational Database Service 란?

- 관계형 데이터베이스를 AWS상에서 사용할 수 있도록 지원하는 서비스. 즉 Database 서비스
- 생성 후 서비스를 이용하기만 하면 되므로 SaaS에 해당
- DB 인스턴스에 대한 Shell 지원 불가 및 OS 제어 불가(AWS 관리)
- 백업, 소프트웨어 패치, 장애감지 및 복구를 AWS가 관리
- Storage 용량에 대하여 Auto Scaling 지원

- **지원 가능 엔진**
  - **PostgreSQL**
  - **MySQL**
  - **MariaDB**
  - **Oracle Database**
  - **Microsoft SQL Server**

Amazon Relational Database Service (RDS)

. 지원 가능 엔진

- Amazon Aurora

- 엔터프라이즈 관계형 데이터베이스
- MySQL 및 PostgreSQL 지원하며, 가격은 상용 데이터베이스 비용의 10분의 1
- MySQL 데이터베이스보다 최대 5배 빠르며 표준 PostgreSQL 데이터베이스보다 최대 3배
- 데이터베이스 리소스의 안정성 및 가용성을 유지하면서도 불필요한 입/출력(I/O) 작업을 줄여 데이터베이스 비용을 절감
- 6개의 데이터 복사본을 3개의 가용 영역에 복제하고 지속적으로 Amazon S3에 데이터를 백업, 읽기 전용 복제본을 15개까지 배포 가능

Amazon Relational Database Service (RDS)

DB Instance

- Instance: RDS 기본 구성 요소로서 클라우드에서 실행하는 격리된 데이터베이스 환경을 의미함
  이 인스턴스 내에서는 여러 사용자가 만든 데이터베이스가 포함되며 액세스할 여러 도구와 앱 사용 가능
- DB 인스턴스도 EC2처럼 다양한 클래스를 가지고 있음(db.m5, db.r5 등)
- RDS도 클라우드에서 실행되므로 하나의 AZ에서 격리되어 인스턴스로서 실행

DB Instance Storage

- Instance Storage: 데이터베이스 유지를 위해 EBS(Elastic Block Storage)를 사용하며 필요한 스토리지 용량에 맞춰 자동으로 데이터를 여러 EBS 볼륨에 나누어 저장
- 스토리지 유형
  - 범용 SSD: 대부분의 워크로드에서 사용하는 무난한 스토리지
  - 프로비저닝 IOPS: 빠르고 일관적인 I/O 성능이 필요하고 일관적으로 낮은 지연시간이 요구될 경우 사용하는
- 스토리지
  - 마크네틱: 접속 빈도가 적은 워크로드에 적합한 스토리지

Amazon Relational Database Service (RDS)

Multi-AZ

- RDS는 Multi-AZ라는 기능을 통해 고가용성을 지원
- 다수의 AZ에 DB인스턴스를 둠으로써 하나 혹은 그 이상의 AZ가 파괴되어 서비스가 불가능할 때를 대비
- 기본 인스턴스가 수행해야 할 작업(백업, 스냅샷 생성) 등을 대신하여 수행함으로써 기본 인스턴스의 부담을 줄임
- 기본 인스턴스에서 스냅샷을 캡쳐 한 후 다른 AZ에 복원하여 '동기식' 예비 복제본을 생성함
- 즉, Active(AZ-A)-Standby(AZ-B,C) 구조를 형성한 후 지속적으로 동기화
- '예비' 복제본 이기 때문에 읽기, 쓰기 작업을 수행할 수 없음
- Multi-AZ를 사용하는 경우 단일 AZ 배포에 비해 쓰기 및 저장 지연 시간이 길어질 수 있음(Standby에 데이터 동기화)
- Multi-AZ를 활성화한 상태에서 DB 인스턴스에 문제가 발생하면 자동으로 다른 AZ의 예비 복제본(Standby)로 전환하여 서비스를 이어나 감
- 전환에 사용되는 시간은 약 60-120초
- 전환되는 상황
  - 가용영역(AZ) 중단
  - 기본 DB 인스턴스 오류
  - DB 인스턴스 서버 유형 변경
  - 기본 DB 인스턴스 OS에서 소프트웨어 패치 실시
  - 장애 조치 재부팅(Failover) 실시

Amazon Relational Database Service (RDS)

Read Replica

- 읽기 전용 복제 본
- 기본 DB 인스턴스가 읽기와 쓰기를 담당한다면 Read Replica는 읽기 작업만을 담당하여 마스터 DB인스턴스의 부하를 줄임
- 우선 DB 인스턴스의 스냅샷을 캡쳐 한 후, 이를 기반으로 Read Replica를 생성하며, 데이터를 '비동기' 복제 방식을 통해 업데이트
- MySQL, MariaDB, PostgresSQL, Oracle 에서 사용 가능
- 다른 리전에도 Read Replica를 두는 것이 가능
- 리전당 최대 5개까지 두는 것이 가능
- Read Replica 또한 독립된 인스턴스로 승격 가능

Automated Backup

- RDS의 자동백업으로 개별 데이터베이스를 백업하는 것이 아닌 DB 인스턴스 전체를 백업하는 것
- 매일 백업이 이루어지면, 기본 보존기간은 CLI로 생성 시 1일, 콘솔로 생성시 7일이며, 최저 1일부터 35일까지 가능
- 특정시점을 지정하여 복원 가능하며 복원 기간내로부터 최근 5분까지 특정시점을 지정하여 복원 가능
- 사용자가 지정한 백업시간에 자동적으로 백업되며, 백업 중에는 스토리지 I/O 가 일시적으로 중단될 수 있음 (Multi-AZ 사용 시 Standby 에서 백업 실시)

Amazon Relational Database Service (RDS)

Snapshot

- DB 인스턴스의 특정시점을 스냅샷으로 생성하는 것
- 자동백업과 마찬가지로 스냅샷 역시 자동으로 생성 가능하며 수동으로 생성 가능
- 자동백업과는 달리 스냅샷 생성시점으로만 복원 가능
- 스냅샷으로 복원 시 DB 인스턴스를 복원하는 것이 아닌 개별 DB 인스턴스가 생성됨(DB 스냅샷에서 기존 DB 인스턴스로 복원할 수 없으며, 복원하면 새 DB 인스턴스가 생성됨)
- 스냅샷 복사, 공유, 마이그레이션이 가능함
- 스냅샷 생성 중에는 스토리지 I/O가 일시적으로 중단될 수 있음(Multi-AZ 사용시 Standby에서 백업 실시)

Enhanced Monitoring

- RDS의 지표를 실시간으로 모니터링하는 '강화된' 모니터링
- 모니터링 지표는 CloudWatchs Logs에 30일간 저장됨
- 일반 모니터링과의 차이점은 Enhanced Monitoring은 인스턴스 내 에이전트를 통해 지표를 수집하는 반면, 일반 모니터링은 하이퍼바이저에서 수집한다는 점
- 최대 1초단위까지 수집 가능

Amazon Relational Database Service (RDS)

RDS vs DB in EC2

- RDS
- 백업 및 소프트웨어 패치 등 서비스 유지에 필요한 기능을 AWS가 관리
- SaaS이므로 서비스 최적화에 집중할 수 있음
- SSH를 통한 접속이 불가하며, EC2를 통한 접속 혹은 Workbench와 같은 접속 프로그램을 통해서만 가능
- 설정을 변경할 수 있는 파라미터 그룹이 존재하지만 변경 불가능한 설정도 존재
- DB in EC2
- EC2 위에 데이터베이스를 직접 올리는만큼 설정을 마음대로 변경할 수 있고 커스터마이징 또한 가능
- RDS와는 반대로 백업과 패치 등 관리를 직접해야 함
- EC2에 설치하는 것이기에 SSH 접속 가능

Amazon Aurora

- '클라우드에서 데이터베이스를 처음부터 설계하면 어떨까'라는 생각에서 출발한 DB 서비스
- MySQL과 PostgreSQL과 호환 가능함
- 각 AZ마다 2개의 데이터 복사본을 자동으로 유지하며, 에러를 스스로 찾아내고 복구함
- Read Replica는 다른 DB 서비스와 달리 최대 15개까지 가능하며, 백업과 스냅샷이 퍼포먼스에 영향을 주지 않음

Amazon DynamoDB

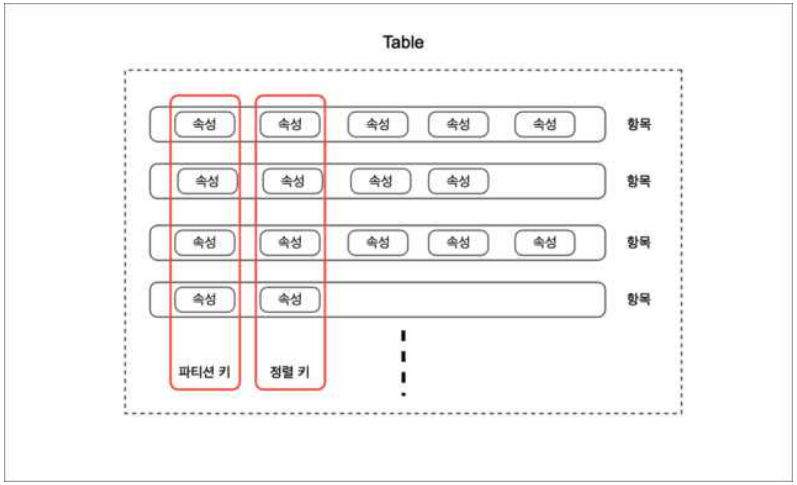
Amazon DynamoDB 란?

- 빠르고 유연한 NoSQL DB 서비스
- 완전관리형 서비스
- SSD 스토리지에 저장되며, 지리적으로 나누어진 3곳의 데이터 센터(AZ)에 분산저장됨
- 테이블, 항목, 속성 등의 구성요소로 나뉘며 테이블의 각 항목을 나타내는 고유 식별자인 '기본 키'가 있음
- Dynamo DB의 일관성 모델 : Eventual Consistent Reads, Strongly Consistent Reads
- Eventual Consistent Reads
  - 모든 데이터의 복사본은 수 초 내에 도달 가능하나 업데이트되지 않은 데이터가 전달될 수 있음, 읽기를 반복하면 최신 데이터를 반환함
- Strongly Constitent Reads
  - 결과값을 리턴 할 때 읽기 전 모든 변경점을 반영한 상태로 리턴 함
- Auto Scaling 지원

Amazon DynamoDB

Amazon DynamoDB 주요 개념

- 항목(Item)
  - 테이블에 Insert, Update, Delete 하게 될 속성들의 집합을 의미, RDBMS에서의 행과 유사한 개념
- 속성(Attribute):
  - 항목을 구성하는 각각의 데이터들을 의미
- 파티션 키 (Partition Key):
  - 테이블을 생성할 때 필수적으로 설정을 해야 하는 기본 키
  - 파티션 키는 물리적 공간인 파티션을 특정 하는 키로, 이 파티션 키에 따라서 저장하는 항목들의 분배가 결정
  - 쉽게 말해 항목들의 주소로서 역할
  - 이 파티션 키를 사용함으로써 특정 항목을 조회할 때, 테이블 전체를 조회하는 것이 아닌 특정 파티션 키를 가진 항목만 조회할 수 있어 성능을 대폭 향상
  - 이러한 특징 때문에 파티션 키는 고유 값이 많은 속성으로 설정을 하는 것을 권장.(예를 들어, 고객 ID, 디바이스 ID 등)

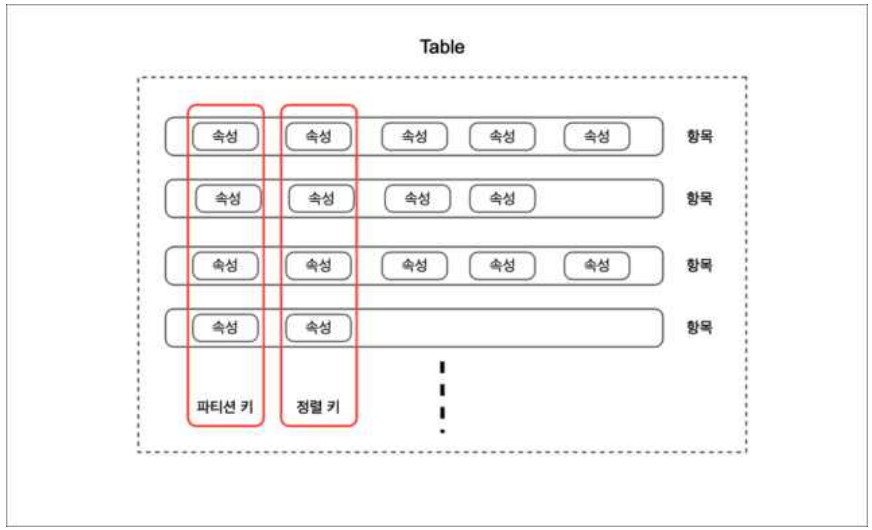


https://dev.classmethod.jp/articles/introduce_amazon_dynamodb/

Amazon DynamoDB

Amazon DynamoDB 주요 개념

- 정렬 키(Sort Key)
  - 테이블을 생성할 때 선택적으로 설정할 수 있는 기본 키
  - 정렬 키는 동일한 파티션 키를 가진 데이터를 정렬할 때 사용
  - 정렬 키와 파티션 키를 함께 이용 함으로서 <, >, between, contains 등의 연산자를 이용한 범위 검색이 가능
  - 예를 들어, DynamoDB를 사용하고 있는 쇼핑몰 사이트가 있다고 가정
  - 사용하고 있는 DynamoDB 테이블의 파티션 키를 고객 ID, 정렬 키를 고객이 물건을 주문한 시간이라고 설정한다면 between 연산자를 이용해 특정 기간 동안 고객이 주문한 물건에 대한 항목들을 검색하는 것이 가능
  - 여기서 파티션 키와 정렬 키는 기본 키임으로 생략할 수 없고, 또한 변경을 할 수 없기 때문에 주의



https://dev.classmethod.jp/articles/introduce_amazon_dynamodb/

Amazon ElastiCache

- **Cache란 무엇이며 왜 써야 하는가?**
  - Cache는 CPU 칩 안에 들어가 있는 작은 메모리 (물리적 실체).
  - 프로세서가 필요한 데이터가 있을 때마다 메인 메모리에 일일이 접근하여 속도가 지연되는 것을 막기 위해
  - 자주 사용하는 데이터를 담아두는 곳
  - 즉 처리 속도 향상을 위해 존재하는 작은 칩이자 메모리
  - L1,L2,L3로 나뉘며 숫자가 적을 수록 도달하는 속도가 빠름
  - Cache는 CPU와 메모리 사이 뿐만 아니라, 메모리와 디스크 사이에서도 발생함
  - In Memory Cache는 메모리와 디스크 사이의 Caching을 의미

Amazon ElastiCache

- **In Memory Cache(In Memory DataBase)**
  - 데이터 처리 속도를 향상시키기 위한 메모리 기반의 DBMS
  - 메모리 위에 모든 데이터를 올려 두고 사용하는 데이터베이스의 일종(ElastiCache가 AWS 카테고리에서 DB 부분에 있는 이유)
  - 디스크에 최적화된 Database(RDS 등)에서 저장된 쿼리 결과나 자주 사용하는 데이터를 메모리에 적재하여 사용하는 것은 비효율적
  - 즉 모든 데이터를 메모리 위에 올려 두어 굳이 디스크 기반의 데이터베이스에까지 이동하여 데이터를 가져와 속도가 저하되는 것을 막음
  - 또한 데이터베이스의 데이터 뿐만 아니라, 디스크, 세션, 기타 동적으로 생성된 데이터를 저장할 수 있음
  - 다만 메모리 기반의 데이터베이스이기 때문에, 휘발성 메모리라는 단점이 존재하며 전원 공급 차단 시 모든 데이터가 유실되고 할당된 메모리에 한해 저장 가능

Amazon ElastiCache

ElastiCache

- AWS의 In Memory Cache Service
- Memcached와 Redis로 나뉨
- Memcached, Redis 모두 비관계데이터베이스형(NoSQL) 서비스이며, Key-value 기반임
- Memcached, Redis 모두 이미 존재하는 서비스이며 AWS에서 사용 가능하도록 구현한 것
- ElastiCache는 Node로 구성되어 서비스를 제공하며, Node는 EC2처럼 다양한 Type을 가지고 유형에 따라 다양한 메모리 크기를 가짐
- 다양한 Type을 갖는 이유는 적은 양의 메모리가 필요할 경우, 작은 Type의 Node를 사용하여 비용을 적게 들게 하기 위함
- 유형이 결정된 Node들은 '고정된' 메모리 크기를 가지며, 각자의 DNS로 이루어진 엔드 포인트를 보유함

Amazon ElastiCache

Memcached의 특징

- Cluster로 구성되어 있으며, Cluster 내에는 Node들이 존재하여 인 메모리 캐시 로서의 역할을 담당함
- 각 Node는 Type별로 메모리를 보유하며 서비스를 제공하며, 필요시 Node를 늘려 서비스 용량을 향상시킬 수 있음
- 각 Node별로 AZ를 따로 둘 수 있지만, 장애 조치(Failover)가 불가능하고 복제본을 둘 수 없음

Redis의 특징

- 기본적으로 Cluster로 구성되지는 않지만, Cluster로 구성이 가능하며 Shard와 Node를 가지고 있음
- Shard는 여러 Node로 구성되며, 하나의 Node가 읽기/쓰기를 담당하고 나머지 Node는 복제본 역할을 함
- Cluster로 구성되지 않은 Redis는 하나의 Shard만을 가지지만, Cluster로 구성될 경우 다수의 Shard를 갖게 됨
- 복제본을 가지므로, 장애조치(복제본을 기본 Node로 승격)가 가능하며 Multi-AZ 기능을 지원함

Amazon Redshift

- **필요 개념**
  - Data Warehouse(DW) : 하나의 통합된 데이터 저장공간으로서, 다양한 운영 환경의 시스템 들로부터 데이터를 추출, 변환, 통합해서 요약한 데이터베이스
    - 데이터베이스가 관련 있는 업무 데이터는 잘 저장하나, 저장된 데이터들을 제대로 활용하지 못 하는 것에서 착안
    - 기본적으로 관계형 데이터베이스가 있는 상태를 가정하여 DW를 구성하며, 동영상이나 음악처럼 DB에 저장할 수 없는 파일도 필요한 부분을 추출하여 보여주어야 함
  - ETL(Extract, Transform, Load) : 데이터를 추출하고, 변형하여, (Data Warehouse에) 적재하는 과정을 일컫는 말
  - BI(Business Intelligence) : 데이터 추출/통합/리포팅을 위한 기본도구 집합, DW에서 분석된 데이터를 통해 숨겨진 패턴을 찾아냄
    => ETL을 통해 뽑아낸 데이터를 DW에 적재하고, BI를 이용하여 분석하는 기본 과정을 거침

Amazon Redshift

- **Redshift란?**
  - PostgreSQL를 기반으로 하는 AWS의 Data Warehouse Service
  - 모든 데이터를 표준 SQL 혹은 BI 도구를 사용하여 효율적으로 분석할 수 있도록 지원
  - 대량 병렬처리(MPP)를 통해 복잡한 쿼리라도 빠른 속도로 실행하여 대용량 처리 가능
  - 열(Column) 단위 데이터 저장방식
  - COPY 명령어를 통해 Amazon EMR, Amazon DynamoDB, S3로부터 데이터를 병렬 로드 가능
  - Enhanced VPC Routing을 통해 클러스터와 VPC 외부의 COPY, UNLOAD 트래픽을 모니터링할 수 있음
  - WLM(Workload Management)를 통해 사용자가 작업 부하 내 우선 순위를 유연하게 관리하도록 지원
  - 보존기간이 1일인 자동 백업을 지원하며, 최대 35일까지 설정 가능
  - 단일 AZ 배포만을 지원함

EMR: Elastic MapReduce

Amazon Redshift

Redshift의 구성

- 클러스터 : Redshift의 핵심 요소로, 하나의 리더 노드와 다수의 컴퓨팅 노드를 가지고 있는 구성 요소
- 리더 노드 : 클라이언트 프로그램과 일어나는 통신을 비롯해 컴퓨팅 노드간의 모든 통신/작업 관리
- 컴퓨팅 노드 : 실제 작업을 수행하는 노드로, 각 노드마다 전용 CPU와 메모리 내장 디스크 스토리지를 따로 보유함

Redshift vs RDS

- Redshift는 보고 및 분석에 사용되지만, RDS는 OLTP(온라인 트랜잭션) 워크로드에 사용
- Redshfit는 대용량 데이터 세트를 대상을 복합적인 분석 쿼리를 빠르게 실행하는 것에 목표를, RDS는 단일행 트랜잭션에 목표를 둠

7. Virtual Private Cloud(VPC)

Virtual Private Cloud(VPC)

- **Virtual Private Cloud(VPC)란?**
  - AWS의 계정 전용 가상 네트워크 서비스
  - VPC 내에서 각종 리소스(EC2, RDS, ELB 등)을 시작할 수 있으며 다른 가상 네트워크와 논리적으로 분리되어 있음
  - S3, Cloudfront 등은 非VPC 서비스로 VPC 내에서 생성되지 않음
  - 각 Region별로 VPC가 다수 존재할 수 있음
  - VPC는 하나의 사설 IP 대역을 보유하고, 서브넷을 생성하며 사설 IP 대역 일부를 나누어 줄 수 있음
  - 허용된 IP 블록 크기는 /16(IP 65536개) ~ /28(IP 16개)
  - 권고하는 VPC CIDR 블록(사설 IP 대역과 동일함)
    - 10.0.0.0 ~ 10.255.255.255(10.0.0.0/8)
    - 172.16.0.0 ~ 172.31.255.255(172.16.0.0/12)
    - 192.168.0.0 ~ 192.168.255.255.(192.168.0.0/16)

Virtual Private Cloud(VPC)

- **Subnet**
  - VPC 내 생성된 분리된 네트워크로 하나의 서브넷은 하나의 AZ(Availability Zone)에 연결됨
  - VPC가 가지고 있는 사설 IP 범위 내에서 '서브넷'을 쪼개어 사용가능
  - 실질적으로 리소스들은 이 서브넷에서 생성이 되며 사설 IP를 기본적으로 할당 받고 필요에 따라 공인 IP를 할당 받음
  - 하나의 서브넷은 하나의 라우팅 테이블과 하나의 NACL(Network ACL)을 가짐
  - 서브넷에서 생성되는 리소스에 공인 IP 자동할당 여부를 설정할 수 있음
    - 이 기능을 통해 Public Subnet과 Private Subnet을 만들어 커스터마이징 가능
    - 서브넷 트래픽이 후술할 인터넷 게이트웨이로 라우팅 되는 경우 해당 서브넷을 Public Subnet, 그렇지 않은 서브넷의 경우 Private Subnet이라 함
  - 각 서브넷의 CIDR 블록에서 4개의 IP 주소와 마지막 IP 주소는 예약 주소로 사용자가 사용할 수 없음, 예를 들어 서브넷 주소가 172.16.1.0/24일 경우
    - 172.16.1.0 : 네트워크 주소(Network ID)
    - 172.16.1.1 : VPC Router용 예약 주소(Gateway)
    - 172.16.1.2 : DNS 서버의 IP 주소
    - 172.16.1.3 : 향후 사용할 예약 주소
    - 172.16.1.255 : 네트워크 브로드 캐스트 주소

Virtual Private Cloud(VPC)

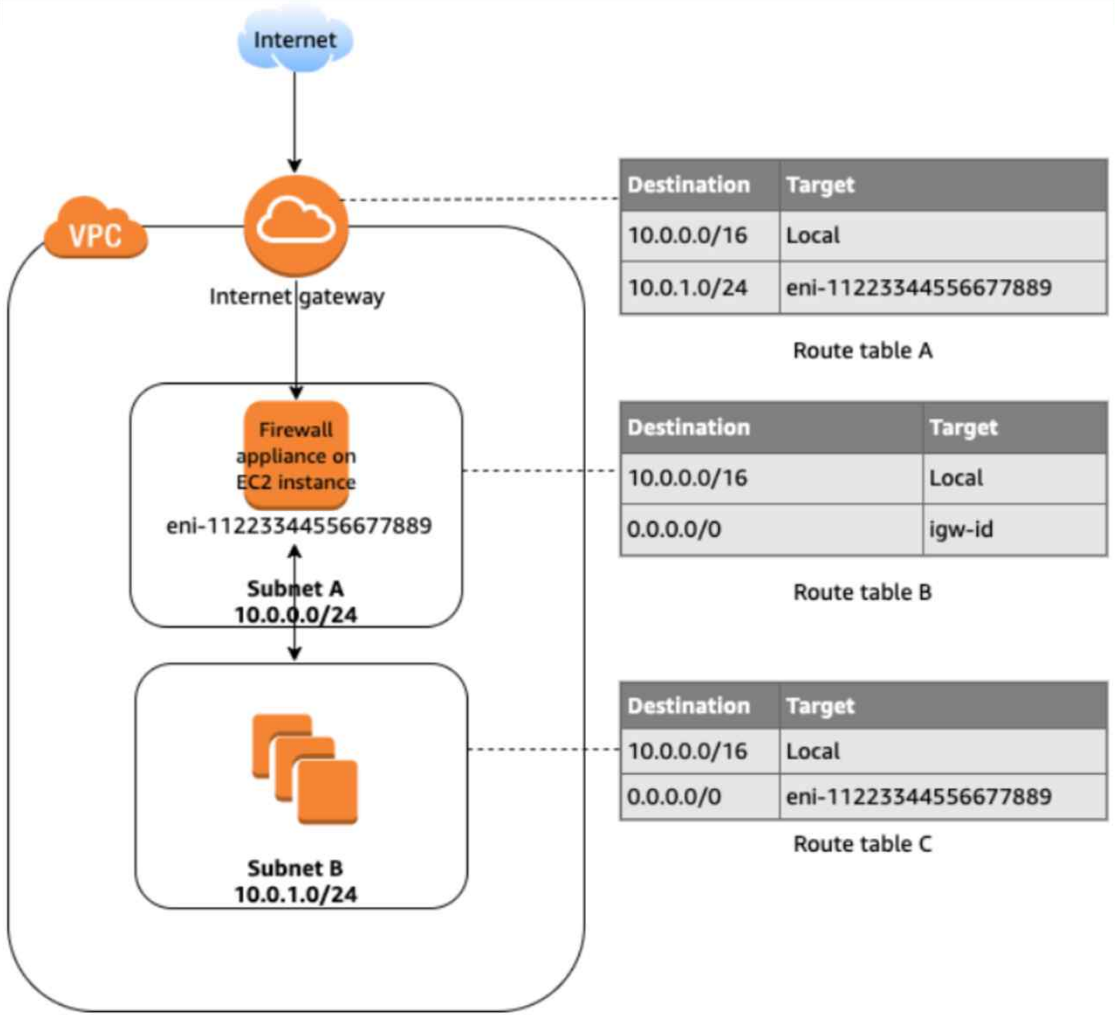
ENI(Elastic Network Interface)

- 가상 네트워크 인터페이스
- VPC 내 리소스들은 ENI와 사설 IP를 기본적으로 할당 받음
  - 선택적으로 공인 IP도 할당 가능
  - Public Subnet의 경우, 리소스 생성시 자동으로 공인 IP를 할당함
- 사설 IP 주소는 추가로 할당 가능하며 공인 IP 역시 나중에 할당 가능

Routing Table

- 서브넷 내의 트래픽이 전송되는 위치를 결정하는 라우팅의 규칙 집합
- 라우팅 테이블은 기본적으로 VPC의 범위에 해당하는 범위를 기본 라우팅으로 가지며(ex. 172.16.1.0/24) 이를'Local'로 표시함
- 또한 Internet Gateway, NAT Gateway, VPC Endpoint, Peering 등을 설정하고 그 서비스로 트래픽을 보내도록 라우팅 설정할 수 있음
- '0.0.0.0/0'은 default routing을 뜻하며 트래픽이 가고자 하는 목적지가 라우팅 테이블에 없는 경우(ex.172.16.1.0/24 네트워크에서 외부 인터넷으로 나아가고자 할 때) 사용하는 라우팅이며 보통 Internet Gateway나 NAT Gateway로 외부 인터넷을 지정할 때 사용
- 하나의 서브넷은 하나의 라우팅 테이블만 가지지만, 하나의 라우팅 테이블은 다수의 서브넷을 가질 수 있음
- Public Subnet의 라우팅 테이블에는 인터넷 게이트웨이로의 라우팅이 있으며, Private Subnet에서 외부 인터넷 통신이 필요할 경우 NAT Gateway로의 라우팅이 잡혀 있음

Virtual Private Cloud(VPC)



Destination	Target
10.0.0.0/16	Local
10.0.1.0/24	eni-11223344556677889

Route table A

Destination	Target
10.0.0.0/16	Local
0.0.0.0/0	igw-id

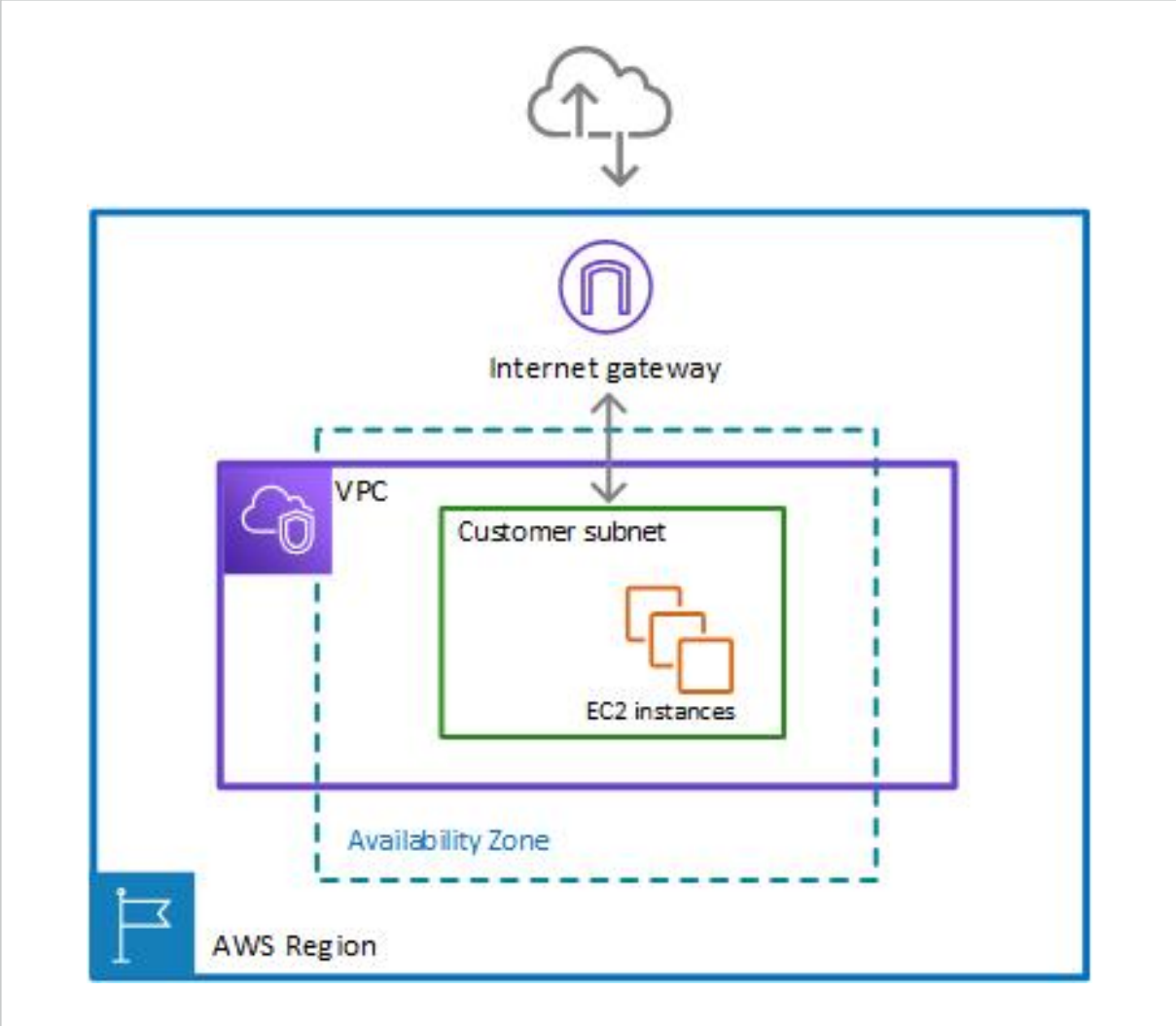
Route table B

Destination	Target
10.0.0.0/16	Local
0.0.0.0/0	eni-11223344556677889

Route table C

Virtual Private Cloud(VPC)

- **Internet Gateway (IGW)란?**
  - VPC 내 리소스가 외부 인터넷을 사용하고자 할 때 사용하는 게이트웨이
  - 인터넷 게이트웨이가 없으면 외부 인터넷을 사용할 수 없음
  - 인터넷 게이트웨이를 생성한 후, '0.0.0.0/0'에 대하여 라우팅 테이블을 인터넷 게이트웨이로 잡아주면 사용 가능
  - 다만 인터넷 게이트웨이가 있다 하더라도, VPC 내 리소스가 공인 IP를 가지고 있지 않다면 인터넷 사용 불가능
  - 또한 위의 설정을 모두 했음에도 인터넷이 제대로 되지 않는다면, 보안그룹과 Network ACL을 확인해야 함



https://docs.aws.amazon.com/ko_kr/network-firewall/latest/developerguide/arch-single-zone-igw.html

Virtual Private Cloud(VPC)

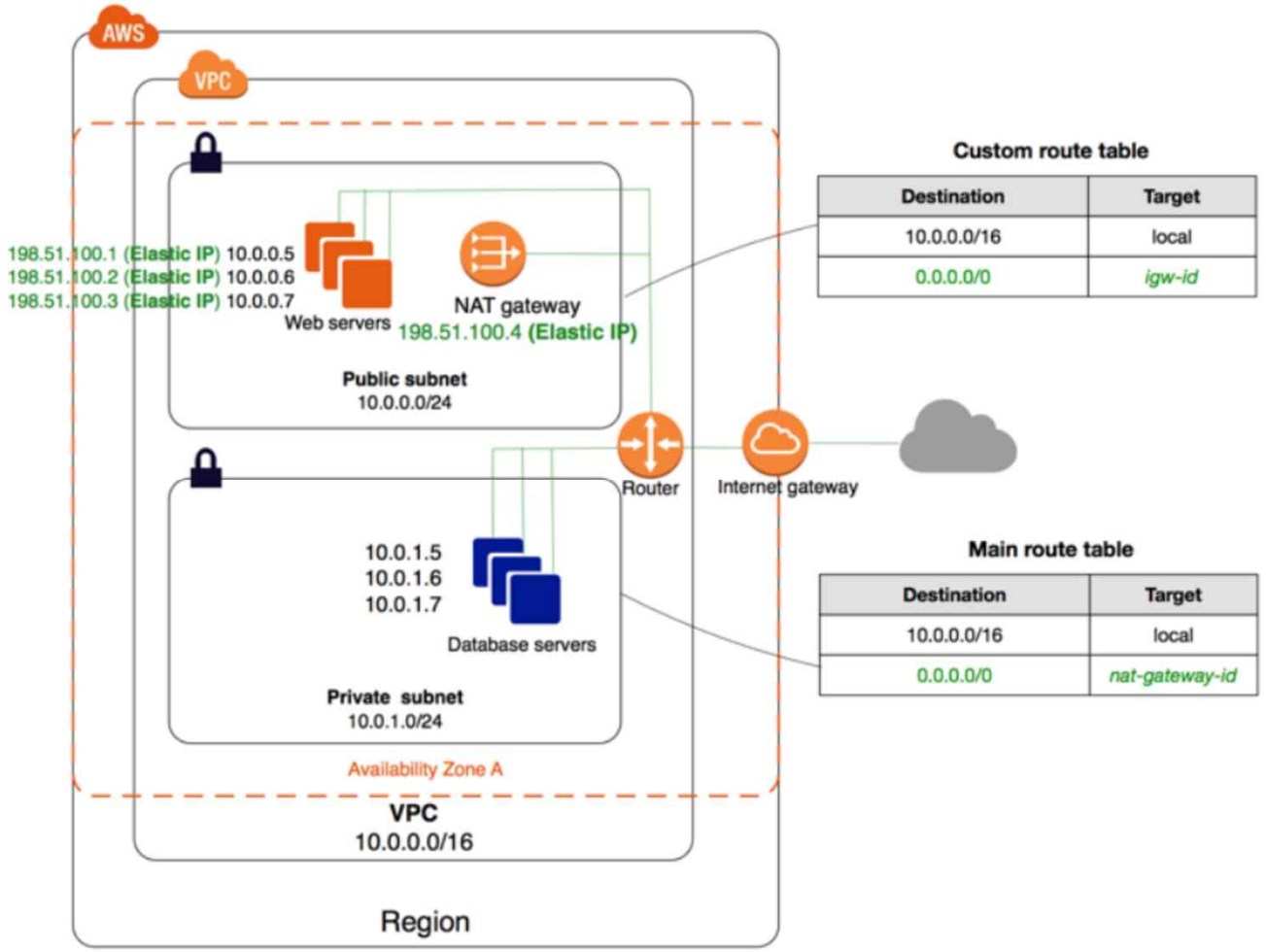
- **NAT Gateway**
  - 외부에서의 접촉이 원천적으로 차단되어 있는 Private Subnet에서 인터넷 접속을 통해야 할 경우 사용하는 게이트웨이
  - VPC 내부 리소스가 NAT Gateway를 통해 인터넷을 접속할 수 있지만, 외부에서 NAT Gateway를 통해 VPC 내부로 들어올 수 없음
  - 인터넷이 연결된 Public sunbet에 NAT Gateway(EIP를 갖고 있음)를 생성한 후, Private Subnet의 라우팅 테이블에 '0.0.0.0/0'에 대하여 라우팅을 NAT Gateway로 잡아주면 사용 가능
  - CloudWatch를 이용하여 모니터링 가능

- **NAT Instance**
  - Public Subnet에 생성된 NAT Gateway 대신 EC2 인스턴스를 사용하는 방법
  - Public Subnet에 공인 IP를 가진 특수한 인스턴스를 게이트웨이로 삼고, Private Subnet의 라우팅 테이블에'0.0.0.0/0'에 대하여 NAT Instance를 Gateway로 설정한 후, 'SrcDestCheck' 속성을 비활성화해야 함
  - 커뮤니티 AMI에 있는 'ami-vpc-nat'로 시작하는 인스턴스로 사용 가능
  - 인스턴스이기 때문에 보안그룹의 설정을 적용 받으므로 트래픽을 제어할 수 있음

Virtual Private Cloud(VPC)

- **NAT Instance vs NAT Gateway**
  - NAT Gateway는 AWS에서 관리하기 때문에 유지보수 할 필요가 없으나 NAT Instance는 사용자가 직접 관리해야 함
  - NAT Instance는 인스턴스이기 때문에 장애가 발생할 가능성이 있어 스크립트로 인스턴스간 Failover를 신경 써야 함
  - NAT Gateway는 대역폭을 최대 45Gbps까지 확장할 수 있지만, NAT Instance는 인스턴스 유형에 따라 다름
  - NAT Gateway는 보안그룹을 사용할 수 없지만, NAT Instance는 보안그룹을 사용할 수 있음
  - NAT Gateway와 NAT Instance 모두 NACL을 통해 트래픽 제어 가능

Virtual Private Cloud(VPC)



Virtual Private Cloud(VPC)

- **보안 그룹 (Security groups)이란?**
  - VPC 내 인스턴스에 대한 인바운드 및 아웃바운드 트래픽을 제어하는 가상의 방화벽
  - 서브넷 수준이 아닌 인스턴스 수준에서 작동하기 때문에 각 인스턴스를 서로 다른 보안그룹으로 지정할 수 있음
  - 기본적으로 모든 인바운드 트래픽을 거부하고 모든 아웃바운드 트래픽을 허용
  - 인스턴스당 최대 5개의 보안그룹을 설정할 수 있음
  - 보안그룹의 가장 큰 특성은 이른바 'Stateful'로 인바운드 트래픽과 아웃바운드 트래픽에 별도의 규칙을 지정할 수 있는 것
  - 즉 인바운드에는 HTTP(80) 포트가 허용되어 있으나, 아웃바운드에 없는 경우 HTTP 트래픽이 '외부에서 들어왔다가 나갈 때' 전혀 지장이 없음
  - 허용 규칙만 존재하며 거부 규칙이 존재하지 않음
  - Source IP, Protocol, Port 등을 설정할 수 있음
  - 설정 변경 시 즉시 적용됨

Virtual Private Cloud(VPC)

Inbound			
Source	**Protocol**	**Port Range**	**Description**
0.0.0.0/0	TCP	80	모든 IPv4 주소에서 이루어지는 인바운드 HTTP 액세스를 허용
::/0	TCP	80	모든 IPv6 주소에서 이루어지는 인바운드 HTTP 액세스를 허용
0.0.0.0/0	TCP	443	모든 IPv4 주소에서 이루어지는 인바운드 HTTPS 액세스 허용
::/0	TCP	443	모든 IPv6 주소에서 이루어지는 인바운드 HTTPS 액세스 허용
네트워크의 퍼블릭 IPv4 주소 범위	TCP	22	네트워크의 IPv4 IP 주소로부터 Linux 인스턴스로 접근하는 인바운드 SSH 액세스 허용(인터넷 게이트웨이 경유)
네트워크의 퍼블릭 IPv4 주소 범위	TCP	3389	네트워크의 IPv4 IP 주소로부터 Windows 인스턴스로 접근하는 인바운드 RDP 액세스 허용(인터넷 게이트웨이 경유)
Outbound			
Destination	**Protocol**	**Port Range**	**Description**
Microsoft SQL Server 데이터베이스 서버용 보안 그룹의 ID	TCP	1433	지정된 보안 그룹의 인스턴스로 아웃바운드 Microsoft SQL Server 액세스 허용
MySQL 데이터베이스 서버용 보안 그룹의 ID	TCP	3306	지정된 보안 그룹의 인스턴스로 아웃바운드 MySQL 액세스 허용

Virtual Private Cloud(VPC)

- **Network ACL**
  - 서브넷 내부와 외부의 트래픽을 제어하기 위한 가상 방화벽으로 네트워크 스위치의 ACL과 역할이 같음
  - 서브넷의 가상 방화벽이기 때문에 서브넷에 속한 모든 인스턴스가 영향을 받음
  - 기본적으로 모든 인바운드와 아웃바운드 트래픽을 허용함
  - 하나의 ACL은 다수의 서브넷과 연결될 수 있으며, 하나의 서브넷은 하나의 ACL만 연결됨
  - Network ACL의 가장 큰 특징은 'Stateless'로서 인바운드 규칙과 아웃바운드 규칙이 서로 영향을 줌
  - 즉 인바운드에는 HTTP(80) 포트가 허용되어 있으나, 아웃바운드에 없는 경우 HTTP 트래픽이 '외부에서 들어왔다가 나갈 때' 아웃바운드 통신이 되지 않음
  - Security Group과 달리 우선순위 값이 존재하며 가장 작은 값이 가장 높은 우선순위를 가지고 우선순위부터 순서대로 적용됨
  - 허용과 거부 모두 가능
  - 설정 변경시 즉시 적용됨

Virtual Private Cloud(VPC)

인바운드						
규칙 #	유형	프로토콜	포트 범위	소스	허용/거부	설명
100	HTTP	TCP	80	0.0.0.0/0	허용	어떤 IPv4 주소에서 이루어지는 인바운드 HTTP 트래픽도 모두 허용
110	HTTPS	TCP	443	0.0.0.0/0	허용	어떤 IPv4 주소에서 이루어지는 인바운드 HTTPS 트래픽도 모두 허용
120	SSH	TCP	22	192.0.2.0/24	허용	홈 네트워크의 퍼블릭 IPv4 주소 범위로부터의 인바운드 SSH 트래픽 허용(인터넷 게이트웨이를 통해)
130	RDP	TCP	3389	192.0.2.0/24	허용	홈 네트워크의 퍼블릭 IPv4 주소 범위로부터 웹 서버로의 인바운드 RDP 트래픽 허용(인터넷 게이트웨이를 통해)
140	사용자 지정 TCP	TCP	32768-65535	0.0.0.0/0	허용	인터넷으로부터의 인바운드 리턴 IPv4 트래픽 허용(즉, 서브넷에서 시작되는 요청에 대해). 이 범위는 예시일 뿐입니다. 적절한 휘발성 포트 범위를 선택하는 방법에 대한 자세한 내용은 휘발성 포트 단원을 참조하십시오.
*	모든 트래픽	모두	모두	0.0.0.0/0	DENY	이전 규칙에서 아직 처리하지 않은 모든 인바운드 IPv4 트래픽 거부(수정 불가)
아웃바운드						
규칙 #	유형	프로토콜	포트 범위	대상 주소	허용/거부	설명
100	HTTP	TCP	80	0.0.0.0/0	허용	서브넷에서 인터넷으로의 아웃바운드 IPv4 HTTP 트래픽 허용
110	HTTPS	TCP	443	0.0.0.0/0	허용	서브넷에서 인터넷으로의 아웃바운드 IPv4 HTTPS 트래픽 허용
120	사용자 지정 TCP	TCP	32768-65535	0.0.0.0/0	허용	인터넷에서 클라이언트에 대한 아웃바운드 IPv4 응답 허용(예: 서브넷에 있는 웹 서버를 방문하는 사람들에게 웹 페이지 제공). 이 범위는 예시일 뿐입니다. 적절한 휘발성 포트 범위를 선택하는 방법에 대한 자세한 내용은 휘발성 포트 단원을 참조하십시오.
*	모든 트래픽	모두	모두	0.0.0.0/0	DENY	이전 규칙에서 아직 처리하지 않은 모든 아웃바운드 IPv4 트래픽 거부(수정 불가)

Virtual Private Cloud(VPC)

- **Security Group vs Network ACL**
  - Security Group은 'Stateful', Network ACL은 'Stateless'
    - Stateful : Inbound Rule과 Outbound Rule 중 하나가 적용되면 다른 하나는 적용되지 않음
    - Stateless : Inbound Rule과 Outbound Rule 모두 적용됨
  - Security Group은 허용만 가능, Network ACL은 허용 및 거부 가능
  - Security Group은 규칙 리스트에 있는 것 중 적용, Network ACL은 우선순위에 따라 우선 규칙 적용
  - Security Group은 인스턴스의 가상 방화벽, Network ACL은 서브넷의 가상 방화벽

Virtual Private Cloud(VPC)

보안 그룹	네트워크 ACL
인스턴스 레벨에서 운영	서브넷 레벨에서 운영
허용 규칙만 지원	허용 및 거부 규칙 지원
상태 저장(stateful) → 규칙에 관계없이 반환 트래픽이 자동으로 허용	상태 비저장(stateless) → 반환트래픽이 규칙에 의해 명시적 허용 되어야함
트래픽 허용 여부를 결정하기 전에 모든 규칙을 평가	트래픽 허용 여부를 결정할 때 번호가 가장 낮은 규칙부터 순서대로 규칙을 처리
인스턴스 시작시 누군가 보안 그룹을 지정하건, 나중에 보안 그룹을 인스턴스와 연결하는 경우에만 인스턴스에 적용	연결된 서브넷의 모든 인스턴에서 자동 적용 됨 (보안 그룹 규칙이 지나치게 허용적일 경우 추가 보안 계층 제공)

https://docs.aws.amazon.com/ko_kr/vpc/latest/userguide/VPC_Security.html

Virtual Private Cloud(VPC)

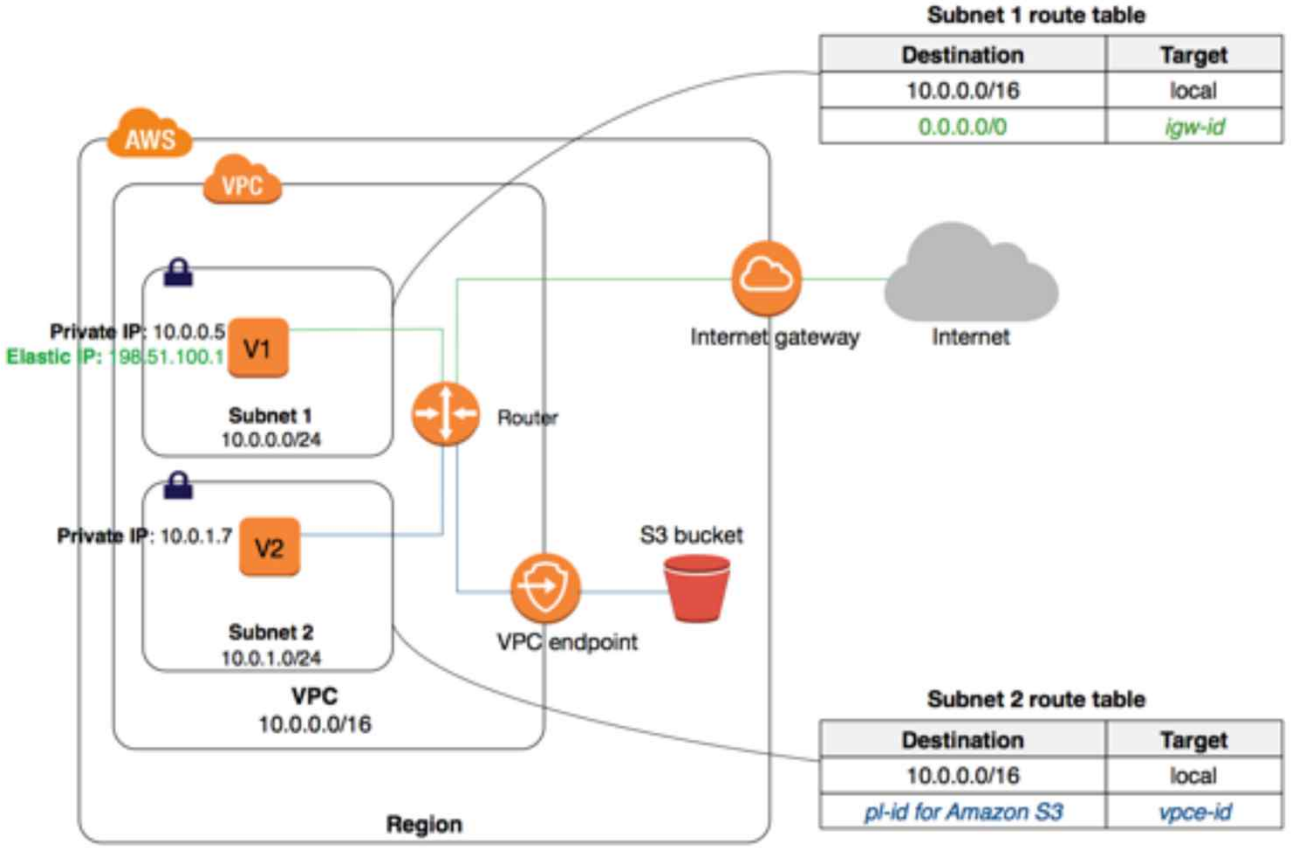
VPC Peering

- 두 VPC 간의 트래픽을 전송하기 위한 기능
- Source VPC와 같은 / 다른 리전의 VPC를 Destination으로 선택하여 Peering 요청을 보낸 후, 수락시 Peering 가능
- 요청과 수락이 필요한 이유는 다른 계정의 VPC도 연결 가능하기 때문
- Peering 생성 후 라우팅 테이블에 해당 peering을 집어넣으면 통신 시작
- VPC Peering은 Transit Routing 불가(2개의 VPC가 하나의 VPC를 통해 통신하는 것)

VPC Endpoint

- VPC 내 요소들과 비VPC 서비스(S3, Cloudwatch, Athena 등)을 외부인터넷을 거치지 않고 아마존 내부 백본 네트워크를 통해 연결하는 방법
- 그러므로 후술할 Direct Connect와 같은 전용선 서비스나 VPN, 인터넷 게이트웨이와 같은 외부 연결이 되어 있지 않는 서브넷에서 아마존의 여러 서비스를 연결할 수 있음
- 간단히 말하여 아마존 서비스 전용선
- VPC 엔드포인트에는 Interface Endpoint, Gateway Endpoint 두 종류가 존재
- **Gateway Endpoint는 S3와 Dynamo DB만 가능**

Virtual Private Cloud(VPC)



Virtual Private Cloud(VPC)

VPN(Virtual Private Network)

- 이 VPN을 통해 AWS와 On-premise의 VPN을 연결하는 것이 가능
- 고객 측 공인 IP를 뜻하는 'Customer Gateway'와 AWS 측 게이트웨이인 'Virtual Private Gateway' 생성 후 터널을 생성하면 사용 가능
- 반드시 VPC에서 VPN 터널 쪽으로 라우팅을 생성해야 함

Direct Connect

- AWS의 전용선 서비스
- 표준 이더넷 광섬유 케이블을 이용하여 케이블 한쪽을 사용자 내부 네트워크의 라우터에 연결하고 다른 한 쪽을 DirectConnect 라우터에 연결하여 내부 네트워크와 AWS VPC를 연결함
- 보통 On-premise의 네트워크와 VPC를 연결할 때 사용
- VPN보다 더 안전하고 빠른 속도를 보장받고 싶을 때 사용

8. Route 53

Amazon Route 53

Route 53이란

- AWS의 DNS 서비스
  - 도메인 등록
  - DNS 라우팅
  - 상태 체크(Health check)
- 도메인 등록시 약 12,000원 정도 지불해야 하며, 최대 3일 정도 걸림
- 해당 도메인을 AWS 내 서비스(EC2, ELB, S3 등)와 연결할 수 있으며 AWS 외 요소들과도 연결 가능함
- 도메인 생성 후 레코드 세트를 생성하여 하위 도메인을 등록할 수 있음
- 레코드 세트 등록시에는 IP 주소, 도메인, 'Alias(별칭, CNAME와 동일한 개념)' 등을 지정하여 쿼리를 라우팅할 수 있음

Route 53의 라우팅 정책

- Simple : 동일 레코드 내에 다수의 IP를 지정하여 라우팅 가능, 값을 다수 지정한 경우 무작위로 반환함
- Weighted : Region별 부하 분산 가능, 각 가중치를 가진 동일한 이름의 A 레코드를 만들어 IP를 다르게 줌
- Latency-based : 지연시간이 가장 적은, 즉 응답시간이 가장 빠른 리전으로 쿼리를 요청함
- Failover : A/S 설정에서 사용됨, Main과 DR로 나누어 Main 장애시 DR로 쿼리
- Geolocation : 각 지역을 기반으로 가장 가까운 리전으로 쿼리 수행, 레코드 생성시 지역을 지정할 수 있음
- Geo-proximity : Traffic flow를 이용한 사용자 정의 DNS 쿼리 생성 가능
- Multi-value answer : 다수의 IP를 지정한다는 것은 simple과 비슷하지만, health check가 가능함(실패 시 자동 Failover)

Amazon Route 53

- **Alias(별칭)**
  - AWS만의 기능으로 Route-53 DNS 기능에 고유한 확장명을 제공
  - AWS의 리소스는 도메인으로 이루어져 있는데 이 도메인을 쿼리 대상으로 지정할 수 있도록 하는 기능
  - Route 53에서 별칭 레코드에 대한 DNS 쿼리를 받으면 다음 리소스로 응답(즉 다음 리소스로 Alias 지정 가능)
    - Cloudfront Distribution
    - ELB
    - Web Site Hosting이 가능한 S3 Bucket
    - Elastic Beanstalk
    - VPC Interface Endpoint
    - 동일한 Hosting 영역의 다른 Route 53 레코드

9. Elastic Load Balancer(ELB)

Amazon Elastic Load Balancing (Amazon ELB)

Amazon Elastic Load Balancing (ELB) 란?

- 단일 또는 여러 가용 영역에서 다양한 애플리케이션의 부하를 처리할 수 있는 네 가지의 로드 밸런서는 모두 애플리케이션의 내결함성에 필요한 고가용성, 자동 조정, 강력한 보안을 갖추고 있음

AWS Elastic Load Balancer (ELB) 종류

- Application Load Balancer (ALB)
- Network Load Balancer (NLB)
- Gateway Load Balancer (GLB)
- Classic Load Balancer (CLB)

Amazon Elastic Load Balancing (Amazon ELB)

보안

- VPC를 사용할 경우 ELB와 관련된 보안 그룹을 생성 및 관리하여 ALB 및 CLB를 위한 추가 네트워킹 및 보안 옵션을 제공
- 원하는 Load Balancer를 인터넷과 연결되도록 구성하거나 퍼블릭 IP 주소 없이 로드 밸런서를 생성하여 인터넷에 연결되지 않은 내부 로드 밸런서로 사용 가능

고가용성 및 탄력성

- 수신되는 트래픽을 단일 가용 영역 또는 여러 가용 영역의 Amazon EC2 인스턴스 전체에 걸쳐 배포
- 수신되는 애플리케이션 트래픽에 대응하여 요청 처리 용량을 자동으로 조정
- 대상이 사용 가능하고 정상 상태인지 확인하기 위해 구성 가능한 케이던스로 대상에 대해 상태 확인을 실행
- 기본 애플리케이션 서버의 사용률에 따라 자동으로 추가되거나 제거

높은 처리량

- 트래픽 증가를 처리할 수 있도록 설계되었으며 초당 수백만 개의 요청을 로드 밸런싱
- 갑작스럽고 변동이 심한 트래픽 패턴을 처리

Amazon Elastic Load Balancing (Amazon ELB)

상태 확인

- EC2 인스턴스, 컨테이너, IP 주소, 마이크로서비스, Lambda 함수, 어플라이언스 등 정상 상태인 대상으로만 트래픽을 라우팅
- 두 가지 방식으로 애플리케이션 상태에 대한 통찰력을 개선
  - 상태 확인 개선을 통해 상세한 오류 코드를 구성할 수 있습니다.
  - 새로운 지표로 EC2 인스턴스에서 실행되는 각 서비스의 트래픽을 파악할 수 있습니다.

고정 세션

- 요청을 동일한 클라이언트에서 동일한 대상으로 라우팅하는 메커니즘을 바탕으로 고정성은 대상 그룹 수준에서 정의

운영 모니터링 및 로깅

- Amazon CloudWatch가 ALB와 CLB에 대해 요청 횟수, 오류 횟수, 오류 유형, 요청 지연 시간 등의 지표를 보고
- NLB와 GLB에 대해 활성 플로우 수, 새 플로우 수, 처리된 바이트 등의 지표를 추적

삭제 방지

- Elastic Load Balancer에서 삭제 방지를 활성화하여 우발적 삭제를 방지

Amazon Elastic Load Balancing (Amazon ELB)

AWS로 마이그레이션

- 자동 조정 기능이 포함되어 있어 용량을 예측할 필요가 없고, 기존 애플리케이션과 클라우드 네이티브 애플리케이션 모두의 로드 밸런싱에 적합
- Terraform, Ansible과 같이 이미 익숙한 일반적인 관리 도구와도 잘 통합

하이브리드 클라우드 구축

- AWS와 온프레미스 리소스 전체에서 로드 밸런싱 할 수 있는 기능을 제공
- 모든 리소스를 동일한 대상 그룹에 등록하고 이 대상 그룹을 로드 밸런서와 연결하면 이를 달성 가능
- AWS와 온프레미스 리소스에 DNS 기반 가중치 적용 방식의 로드 밸런싱을 사용 가능

서버리스 운영 및 컨테이너로 애플리케이션 현대화

- Lambda를 사용하는 경우 복잡한 구성을 거치거나 API 게이트웨이를 사용하지 않고도 ALB를 활용하여 네이티브 HTTP 기반 엔드포인트를 제공
- Kubernetes를 사용한 컨테이너 및 컨테이너 오케스트레이션도 지원하여 클라이언트와 애플리케이션 간의 로드 밸런싱과 서비스 간 통신을 제공

Amazon Elastic Load Balancing (Amazon ELB)

타사 가상 어플라이언스 확장

- GLB를 사용하면 클라우드에서 실행할 때의 확장성과 유연성을 활용하면서 원하는 공급업체의 어플라이언스를 배포 가능

통합 및 글로벌 접근성

- EC2, ECS/EKS, Global Accelerator 및 운영 도구 등의 다른 AWS 서비스와 긴밀하게 통합

Amazon Elastic Load Balancing (Amazon ELB)

- **[알고리즘]**

- **라운드 로빈 방식**
  - 라운드 로빈은 클라이언트로부터 받은 요청을 로드 밸런 싱 대상 서버에 순서대로 할당 받는 방식
  - 첫 번째 요청은 첫 번째 서버, 두 번째 요청은 두 번째 서버, 세 번째 요청은 세 번째 서버에 할당
  - 서버의 성능이 동일하고 처리 시간이 짧은 애플리케이션의 경우, 균등하게 분산이 이루어짐

- **가중 라운드 로빈 방식**
  - 가중 라운드 로빈 방식은 실제 서버에 서로 다른 처리 용량을 지정할 수 있음
  - 각 서버에 가중치를 부여할 수 있으며, 여기서 지정한 정수 값을 통해 처리 용량을 정함

- **최소 연결 방식**
  - 최소 연결 방식은 연결 수가 가장 적은 서버에 네트워크 연결방향을 정함
  - 동적인 분산 알고리즘으로 각 서버에 대한 현재 연결 수를 동적으로 카운트할 수 있고, 동적으로 변하는 요청에 대한 부하를 분산시킬 수 있음

Amazon Elastic Load Balancing (Amazon ELB)

- **ALB(Application Load Balancer)**
  - HTTP 및 HTTPS프로토콜과 1~65535의 TCP 포트, Lambda 호출, ECS를 사용하는 애플리케이션의 로드 밸런싱을 지원
  - 리디렉션, 고정 응답, 인증 및 전달을 위해 호스트 헤더, 경로, HTTP 헤더, 방법, 쿼리 매개 변수 및 소스 IP CIDR와 같은 규칙을 정할 수 있으며 규칙을 사용하여 특정 요청을 라우팅할 방법을 결정할 수 있음
  - ACM과 통합되면서 아주 간단하게 인증서를 로드 밸런서에 연결할 수 있어 SSL/TLS 인증서를 구매, 업로드 및 갱신하는 작업이 매우 간편
  - ALB가 실행된 시간 또는 부분 시간 그리고 시간당 사용된 로드 밸런서 용량 단위(LCU)에 대해 요금이 부과

- **NLB(Network Load Balancer)**
  - IP 프로토콜 데이터를 기반으로 Amazon VPC 내의 대상(Amazon EC2 인스턴스, 마이크로서비스 및 컨테이너)으로 연결을 라우팅
  - TCP 및 UDP 트래픽을 이상적인 매우 짧은 대기 시간을 유지
  - 가용 영역당 단일 고정 IP 주소를 사용하여 갑작스럽고 변동성이 큰 트래픽 패턴을 처리하도록 최적화
  - Auto Scaling, ECS, Amazon CloudFormation 및 ACM와 같은 다른 인기 있는 AWS 서비스와 통합 가능
  - WebSocket 유형의 애플리케이션에 매우 유용한 장기간 연결도 지원
  - 단일 (공용/인터넷 IP, 프라이빗 IP)만 지원
  - NLB가 실행된 시간 또는 부분 시간 그리고 시간당 NLB에서 사용된 NLB용량 단위(NLCU)에 대해 요금이 부과

Amazon Elastic Load Balancing (Amazon ELB)

- **CLB(Classic Load Balancer)**
  - 여러 EC2 인스턴스에서 기본 로드 밸런싱을 제공하고 요청 수준과 연결 수준 모두에서 작동
  - EC2 네트워크 내에 구축된 애플리케이션을 HTTP 포트 80과 HTTPS 포트 443을 단일로 매핑 가능
  - TTP, HTTPS(보안 HTTP), SSL(보안 TCP) 및 TCP 프로토콜을 사용하여 애플리케이션의 로드 밸런싱을 지원
  - ACM과 통합되면서 아주 간단하게 인증서를 로드 밸런서에 연결할 수 있어 SSL/TLS 인증서를 구매, 업로드 및 갱신하는 작업이 매우 간편
  - CLB가 실행된 각 시간 또는 한 시간 미만, 그리고 로드 밸런서를 통해 전송된 각 GB 단위 데이터에 대해 비용이 청구

Amazon Elastic Load Balancing (Amazon ELB)

- **GLB(Classic Load Balancer)**
  - 타사 가상 어플라이언스를 쉽게 배포, 확장 및 관리 가능
  - 흐름을 소스 IP, 대상 IP, 프로토콜, 소스 포트 및 대상 포트로 구성된 5-튜플의 조합으로 여러 가상 어플라이언스에 트래픽을 분산하는 동시에 수요에 따라 확장하거나 축소할 수 있는 하나의 게이트웨이를 제공하여 네트워크의 잠재적인 실패 지점이 제거되고 가용성이 높아짐
  - 배포 프로세스를 간소화하여 현재와 동일한 공급업체와 협력하든 새로운 시도를 하든 관계없이 가상 어플라이언스의 가치를 더 빨리 확인
  - 서드 파티 가상 어플라이언스를 통한 모든 계층 3 트래픽을 투명하게 전달
  - GLB가 실행된 시간 또는 부분 시간 그리고 시간당 GLB에서 사용된 GLB 용량 단위(GLCU)에 대해 요금이 부과
  - GLB는 AWS PrivateLink 기술을 기반으로 한 새로운 유형의 VPC 종단점인 GLB 엔드포인트(GWLBE)를 사용하며, 이 엔드포인트는 애플리케이션이 VPC 경계에서 GWLB와 트래픽을 안전하게 교환하는 방법을 간소화
  - GWLBE에는 별도의 요금이 책정 및 청구

Amazon Elastic Load Balancing (Amazon ELB)

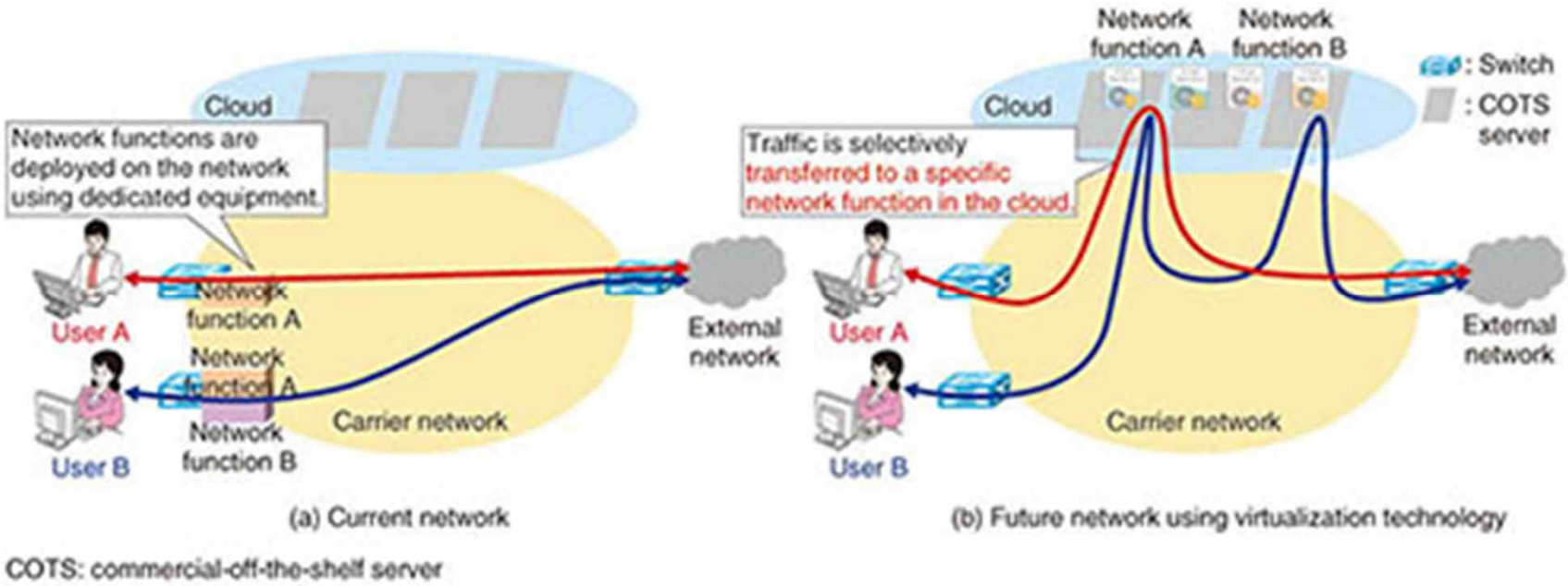
SDN 기능 SFC(Service Function Chaining)

- 2010년 초반 부터 네트워크 업계에는 SDN(Software-Defined Networking) 기술이 학술적으로 부터 조명
- 하지만 기존 네트워크 장비와 호환성이 없다 보니 SDN 환경으로 전환이 어렵고 고객 사례가 많지 않음
- 2014년 즈음 SDN의 기능 중 **SFC(Service Function Chaining)** 이라는 개념 등장
- 말 그대로 특정한 서비스(방화벽, NAT, 패킷 분석, DPI, L4/L7 등)를 연결하는 체인 구조로 접속을 해주는 것

SFC 장점

- 기존 트래픽 전달 환경에 영향도를 최소화 하면서 해당 기능(라우팅)을 제공
- → 기존 환경에서 유사한 기능 제공 시 인라인(In-line)으로 제공해야 하여 장애 시 문제 발생 하거나 혹은 급격한 성능 대응의 어려움이 존재
- 유연하게(동적) SFC 기능(라우팅)을 제공 가능
- → 외부에서 공격 트래픽이 들어오고 있다고 판단 시 동적으로 SFC 기능을 통해 '방화벽, 패킷 분석' 서비스를 경유하게 대응 가능
- 가상화/자동화를 통하여 쉽게 SFC서비스 환경 제공이 가능하며 부하에 따라서 자동으로 오토 스케일링을 통한 성능 대응 가능

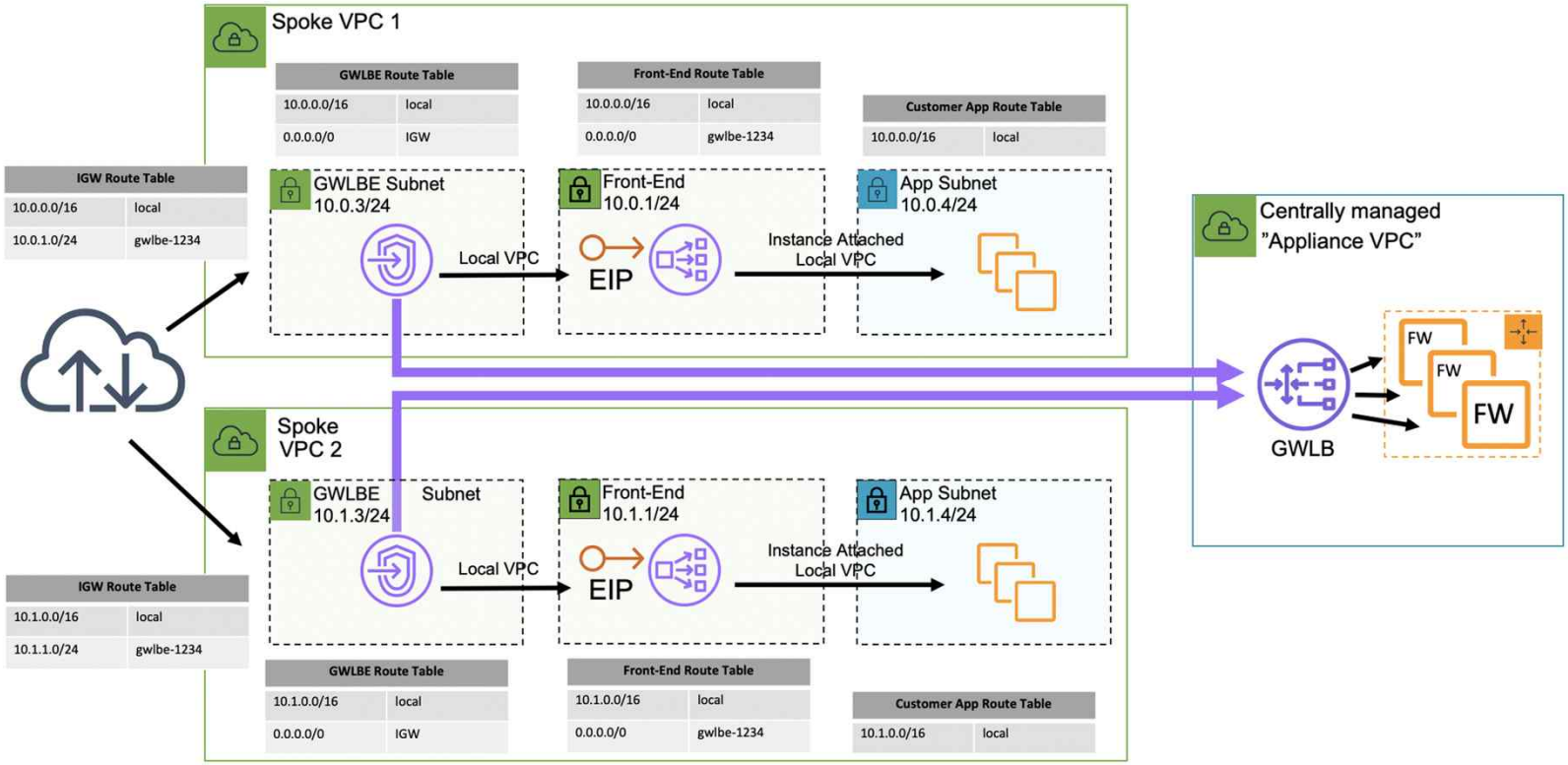
Amazon Elastic Load Balancing (Amazon ELB)



https://www.notion.so/AWS-Gateway-Load-Balancer-GWLB-bda54412433043a1974b83af478f8a9b

Amazon Elastic Load Balancing (Amazon ELB)

- **AWS Gateway Load Balancer (GWLB)**
  - AWS 에 VPC Ingress Routing 기능을 사용하면 IGW(인터넷 게이트웨이)나 VGW(가상 프라이빗 게이트웨이)를 통해 IN/OUT 되는 트래픽을 '방화벽(마켓플레이스 파트너 솔루션)' 을 경유하게 하여 보안 기능을 제공



https://www.notion.so/AWS-Gateway-Load-Balancer-GWLB-bda54412433043a1974b83af478f8a9b

Amazon Elastic Load Balancing (Amazon ELB)

- **Sticky Session**
  - 세션 고정 기능
  - 하나의 요청이 특정 EC2로 들어와 세션이 생성되었다가 종료되고, 들어왔던 요청이 다시 들어올 경우 이미
  - 연결되었던 특정 EC2로 전달하는 기능
  - ALB와 CLB는 쿠키(Cookie)를 활용하며, NLB는 Source IP를 기억하는 방법을 사용

- **교차영역 로드밸런싱**
  - 기본적으로 ELB는 대상그룹에 등록된 EC2별로 적절히 부하분산을 하는 것이 아닌 가용영역별로 비율을 나눔
    - 교차영역이 2개라면 각 가용영역에 50%씩 전달
    - 이렇게 될 경우, A 가용영역에 EC2가 2개이고, B 가용영역에 EC2가 8개일 경우 A/B 가용영역에 50%씩 전달되므로 A 가용영역의 EC2의 부하가 심해짐
  - 이를 방지하기 위해 교차영역 로드 밸런싱을 활성화하면 가용영역이 아닌 EC2 갯수를 기반으로 계산하여 전달하므로 부하가 고르게 나누어짐
  - A 가용영역에 EC2가 8개이고, B 가용영역에 EC2가 2개라 하더라도 가용영역별 부하분산이 아닌 EC2 10개에 대해 고르게 분산함

Amazon Elastic Load Balancing (Amazon ELB)

X-Forwared-for

- Client의 IP를 담고 있는 HTTP Header
- HTTP를 활용하는 ALB, CLB 사용 가능
- EC2 내 서비스가 Client의 IP를 확인할 필요가 있는 경우 ELB는 HTTP Header에 X-Forwared-for를 장착하여 대상그룹에 전달

Connection draining(커넥션 드레이닝)

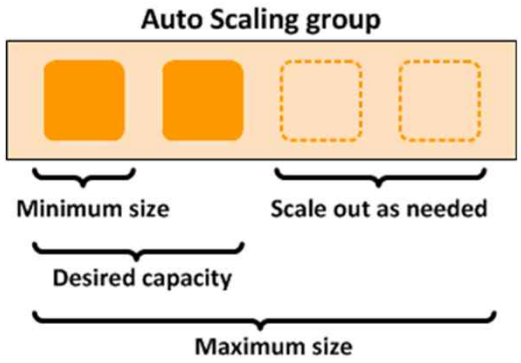
- Autoscaling과 ELB를 함께 사용시 필요에 따라 EC2가 없어질 경우, 해당 EC2에 요청이 들어와 있다면 사용중인 커넥션(세션)이 피해를 볼 수 있음
- 삭제될 예정인 EC2를 사용중인 커넥션(세션)이 작업을 마칠 때까지 기다리는 기능
- ALB, NLB, CLB 사용 가능

10. Auto Scaling

Amazon Auto Scaling

- **Amazon Auto Scaling 이란?**
  - EC2 인스턴스를 자동으로 시작하거나 종료하여 애플리케이션 로드를 처리하기에 적절한 수의 EC2를 유지할 수 있도록 하는 서비스
  - 사용자가 정의하는 조건에 따라 EC2 개수를 자동으로 확장 또는 축소가 가능함
  - 모니터링을 통해 비정상 인스턴스를 탐지하고 교체할 수 있음
  - 수요가 급증할 경우 EC2 수를 자동으로 늘려 성능을 유지하고 수요가 적을 경우 수를 줄여 비용을 절감
  - ELB의 대상그룹을 Auto Scaling Group(ASG)에 포함시켜 자동생성된 EC2로 하여금 트래픽 부하분산을 하도록 설정 가능
  - 수요 변화가 예측 가능한 경우 '예약된 일정'을 통해 정해진 시간에 늘리거나 줄이도록 설정 가능
  - ASG 내 손상된 인스턴스가 발견될 경우, Auto Scaling은 이를 자동으로 종료하고 새로운 인스턴스로 교체함
    - ELB를 사용하는 경우, ELB가 손상된 인스턴스를 트래픽 요청 대상에서 분리시킨 후, Auto Scaling이 이를 새로운 인스턴스로 교체함
  - 비정상 서버 탐지 후 Auto Scaling이 새로운 인스턴스를 In Service 상태로 만들기까지 5분 이내 소요

Amazon Auto Scaling



- **Amazon Auto Scaling 이란?**

Auto Scaling 구성 요소	설 명
그룹	• EC2 인스턴스는 조정 및 관리 목적의 논리 단위로 취급될 수 있도록 그룹으로 구성 • 그룹을 생성할 때 EC2 인스턴스의 최소 및 최대 인스턴스 수와 원하는 인스턴스 수를 지정
구성 템플릿	• 그룹은 EC2 인스턴스에 대한 구성 템플릿으로 시작 템플렛 사용 • 인스턴스의 AMI ID, 인스턴스 유형, 키 페어, 보안 그룹, 블록 디바이스 매핑 등의 정보를 지정
조정 옵션	• Amazon EC2 Auto Scaling은 Auto Scaling 그룹을 조정하는 다양한 방법 정의 • 예를 들어, 지정한 조건의 발생(동적 확장) 또는 일정에 따라 조정하도록 그룹을 구성

Amazon Auto Scaling

시작 구성(Launch Configuration)

- Auto Scaling에서 새로운 인스턴스를 시작할 때 기반이 되는 구성
- AMI(Amazon Machine Image), 인스턴스 유형, 보안그룹, 스토리지 등 EC2를 생성할 때와 마찬가지로 옵션을 구성할 수 있음
- 시작 구성은 한 번 생성시 수정이 불가하며, 복사와 삭제만 가능
- 시작 구성을 변경하기 위해서는 기존의 시작구성을 새로운 시작 구성을 만드는 재료로 사용해야 함
- 하나의 ASG는 하나의 시작구성을 반드시 가짐

조정 정책(Scaling)

- 최소, 목표, 최대 크기를 설정할 수 있고, 별다른 정책이 없을 경우 목표 크기를 유지하려 함
- 또한 3가지 동적 정책 사용 가능(3가지 정책 모두 모든 인스턴스의 평균 값을 지표로 사용함)
- 대상 추적 정책 : ASG의 지표 평균값을 목표로 인스턴스 수를 조절
  - 평균 CPU 사용률, 네트워크 입/출력, 로드 밸런서 요청 수 3가지 활용 가능
- 단순 조정 정책 : 지표의 임계치에 도달할 경우, 사용자가 정한 인스턴스 수를 늘리거나 감소시키는 정책
- 단계 조정 정책 : 단순 조정 정책과 기본은 같지만, 지표 값에 따라 증감 수를 다르게 줄 수 있음(즉 트리거가 여러 개)

Amazon Auto Scaling

조정 휴지(Scaling Cooldown)

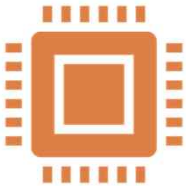
- 시작 구성을 이용한 ASG 생성 시점을 포함하여 인스턴스 생성 혹은 제거 후, 지표의 임계값을 넘더라도 인스턴스를 생성하지 않고 기다리는 시간
- 기본 300초

수명주기 후크(Life Cycle Hook)

- Scale In / Out : 'In'은 감소를, 'Out'은 증가를 뜻함
- 지표의 임계값에 의해, Scale In / Out 이 되고 난 후 'In Service' 상태에 돌입하기 전에 사용자가 정의한 작업을 수행하는 시간을 설정하는 기능
- 기본 3600초
- 즉 인스턴스 시작/종료시 사용자가 작업을 수행할 수 있음
- 이 시간동안 인스턴스에 설치 / 설정 등이 가능
- 가령, Cloudwatch를 사용하면 수명주기 작업이 발생시, Lambda 함수를 호출시키도록 설정 가능

11. CloudFormation

AWS CloudFormation



IaC(Infrastructure as Code)란

AWS의 환경이나 OS 등의 인프라를 구성할 때 코드 기반으로 작성하여 관리하거나 갱신하는 방식

IaC 장점

코드로 관리하기 때문에, 한눈에 모든 설정을 확인

Github등의 버전 관리 서비스를 이용하면, 다른 사람과 협업하거나, 커밋로그로 변경을 관리

템플릿으로 관리함으로써, 어떤 환경에서도 같은 구성으로 테스트

템플릿으로 배포된 리소스들은 추적되며, 템플릿을 삭제하는 것만으로 생성된 모든 리소스를 완벽하게 삭제

IaC 단점

구성에 따라 다양한 툴들을 사용할 수 있어야 하여 러닝코스트가 높음(사람 의존적)

GUI 환경에서 클릭해서 설정하는 것이 더 효율적일 수 있음

IaC로 생성된 리소스는 기본적으로 IaC로 관리해야 하며, GUI에서 수정 어려움 (스택 업데이트시 코드와 환경이 다르면 롤백이 발생)

AWS CloudFormation

- **CloudFormation 이란?**
  - AWS CloudFormation을 활용하면 Infrastructure as Code를 통해 손쉬운 방법으로 관련된 AWS 및 서드 파티 리소스 모음을 모델링하고, 일관된 방식으로 간단히 프로비저닝하고, 수명 주기 전반에 걸쳐 관리
  - CloudFormation 템플릿에는 원하는 리소스와 종속성이 설명되어 있으므로 이를 모두 하나의 스택으로 구성하고 시작
  - 리소스를 개별적으로 관리하는 대신 템플릿을 통해 전체 스택을 단일 단위로 처리하여 필요한 만큼 자주 생성 및 업데이트하고 삭제
  - 스택은 여러 AWS 계정 및 AWS 리전에 걸쳐 관리 및 프로비저닝

AWS CloudFormation

- **Stack**
  - 하나의 단위로 관리할 수 있는 AWS 리소스들의 모음
  - 스택을 생성, 업데이트 또는 삭제하여 리소스 모음을 생성, 업데이트, 삭제를 할 수 있음
  - 스택에서 실행중인 리소스를 변경해야 하는 경우 스택을 업데이트할 수 있는데 이 업데이트된 세트를 '변경세트'라 함
  - 스택을 삭제하는 경우 삭제할 스택을 지정하면 해당 스택과 스택 내 모든 리소스를 삭제함
  - AWS에서 리소스를 삭제할 수 없는 경우 스택이 삭제되지 않음
  - 스택의 리소스 중 하나라도 성공적으로 생성되지 않은 경우 성공적으로 생성한 모든 리소스를 모두 삭제함(이를 Automatic rollback on error라 함)

AWS CloudFormation

- **Template**
  - 스택을 구성하는 AWS 리소스를 JSON 혹은 YAML 형식으로 선언한 텍스트 파일
  - 템플릿은 로컬 혹은 S3에 저장되며, 템플릿을 불러올 때 S3 bucket을 지정할 수 있음
  - 템플릿을 'Designer'를 통해 생성할 수도 있으며 S3 bucket에 저장된 것을 불러와 생성할 수 있음
  - 템플릿의 여러 가지 요소
    - Parameters : 선택 섹션, 스택 생성 및 업데이트시 템플릿에 전달하는 값, 사용자가 선택하는 여러 요소들(EC2유형 – t2.micro 등)
    - Conditions : 선택 섹션, 조건문, 리소스가 생성되는 조건을 만들어 조건 충족시에만 리소스를 만들 수 있도록 하는 요소
    - Resources : 필수 섹션, Cloudformation에 포함될 리소스
    - Metadata : 선택 섹션, 템플릿에 대한 세부 정보를 제공하는 임의의 JSON, YAML 객체
    - Mappings : 선택 섹션, 프로그래밍 언어로 따지면 'Switch' 조건문에 해당하며 '키'에 해당하는 값 세트를 생성하고 해당하는 키가 있으면 값 세트에 맞춰 리소스를 생성함

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Description" : "A simple EC2 instance",
  "Resources" : {
    "MyEC2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "ImageId" : "ami-0ff8a91507f77f867",
        "InstanceType" : "t1.micro"
      }
    }
  }
}
```

- **Cloudformation의 과금 특징**
  - Cloudformation 서비스 자체는 무료이지만, Cloudformation을 통해 생성되는 모든 리소스는 유료임
  - 즉, 생성된 리소스는 독립적인 요소로서 과금이 부여됨

12. IAM

Amazon Web Service 보안

- **Amazon Web Service 보안**
  - **데이터를 안전하게 유지**
    - 전 세계의 리전을 구성하고, 다양한 서비스들의 데이터 복제 기능 등 빠른 복원력을 갖춘 인프라 제공
    - 강력한 암호화 기술 및 접근 제어 및 권한 설정 등을 효율적으로 지원
  - **지속적인 개선**
    - 빠르게 새로운 보안 취약점을 분석하고 안전하게 보장하기 위해서 끊임없이 진화하는 보안 서비스 제공
  - **필요한 만큼 지불**
    - 실시간으로 발생하는 위험 요소를 모니터링 하고 더 낮은 운영 비용으로 요구 사항을 충족할 수 있도록 함

Amazon Web Service 보안

네트워크 보안

- 내장 방화벽
- 전송 중 암호화
- 프라이빗 네트워크와 연결
- DDoS 완화

데이터 암호화

- 암호화 기능
- 키관리 옵션 (AWS Key Management Service)
- 하드웨어 기반 암호화 키 스토리지 옵션 (AWS CloudHSM)

Amazon Web Service 보안

액세스 제어 및 관리

- Identity and Access Management (IAM)
- Multi-factor Authentication (MFA)
- 기업 디렉토리와 통합 및 연동
- Amazon Cognito
- AWS SSO

모니터링 및 로깅

- API 호출에 대한 심층적인 가시성
- 로그 집계 및 옵션
- 경고 알림

Amazon Web Service 보안

AWS 공동 책임 모델

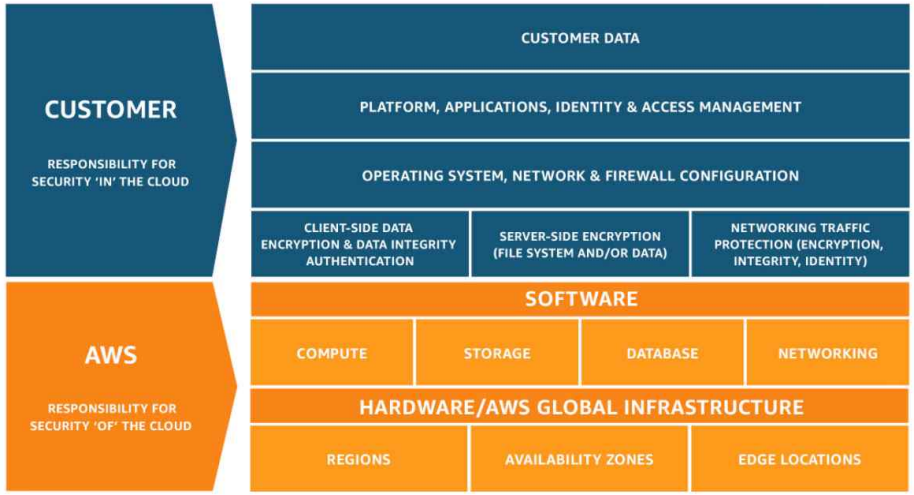
- 보안과 규정 준수는 AWS와 고객의 공동 책임

고객 (클라우드 내부의 보안, responsible for security IN the cloud)

- AWS 클라우드 내에서 생성하고 배치하는 모든 것의 보안을 책임
- 콘텐츠, 사용하는 AWS 서비스, 해당 콘텐츠에 액세스할 수 있는 사용자를 포함하여 콘텐츠에 대한 보안 요구 사항을 관리할 책임은 고객

AWS (클라우드 자체의 보안, responsible for security OF the cloud)

- AWS는 인프라의 모든 계층에서 구성 요소를 운영, 관리 및 제어
- AWS 클라우드에서 제공되는 모든 서비스를 실행하는 인프라 보호 책임
- 하드웨어, 소프트웨어, 스토리지, 네트워킹 및 시설, 글로벌 인프라 등



AWS Identity and Access Management (IAM)

AWS Identity and Access Management(IAM) 이란?

- AWS 리소스에 대한 액세스를 안전하게 제어할 수 있는 웹 서비스
- IAM을 사용하여 리소스를 사용하도록 인증(로그인) 및 권한 부여(권한 있음)된 대상을 제어

AWS IAM 루트 계정 (root account)

- 전체 AWS 서비스 및 계정 리소스에 대해 완전한 액세스 권한
- 이메일 주소 및 암호로 로그인

AWS Identity and Access Management (IAM)

AWS IAM 사용자 (User)

- AWS에서 생성하는 엔터티로서 AWS와 상호 작용하기 위한 사람 또는 애플리케이션
- **최소 권한의 원칙 적용** : 사용자에게 필요한 것만 액세스 부여

→ 기본적으로 AWS에서 새 IAM 사용자를 생성하면 해당 사용자와 연결된 권한이 없는 상태로 생성

→ AWS에서 EC2인스턴스 시작, S3 버킷 생성등 특정 작업을 수행할 수 있도록 허용하려면 IAM 사용자에게 필요한 권한을 부여 해야함.

- 동일한 수준의 액세스가 필요한 직원이 여러 명이더라도 각 직원 별 개별 IAM 사용자 생성 권장

AWS IAM 정책 (Policy)

- AWS 서비스 및 리소스에 대한 권한을 허용하거나 거부하는 문서
- IAM 정책을 사용하여 사용자가 리소스에 액세스

```
{
        "Version": "2012-10-17",
        "Statement": {
                "Effect": "Allow",
                "Action": "s3:ListObject",
                "Resource": "arn:aws:s3:::book_stock"
        }
}
```

Effect : Allow or Deny
Action : AWS API 호출
Resource : API 호출을 받을 리소스

AWS IAM 그룹 (Group)

- IAM 사용자의 모음
- 그룹에 IAM 정책을 할당하면 해당 그룹의 모든 사용자에게 정책에 지정된 권한 부여

AWS Identity and Access Management (IAM)

AWS IAM 사용자 (User)

- AWS에서 생성하는 엔터티로서 AWS와 상호 작용하기 위한 사람 또는 애플리케이션
- **최소 권한의 원칙 적용** : 사용자에게 필요한 것만 액세스 부여

→ 기본적으로 AWS에서 새 IAM 사용자를 생성하면 해당 사용자와 연결된 권한이 없는 상태로 생성

→ AWS에서 EC2인스턴스 시작, S3 버킷 생성등 특정 작업을 수행할 수 있도록 허용하려면 IAM 사용자에게 필요한 권한을 부여 해야함.

- 동일한 수준의 액세스가 필요한 직원이 여러 명이더라도 각 직원 별 개별 IAM 사용자 생성 권장

AWS IAM 정책 (Policy)

- AWS 서비스 및 리소스에 대한 권한을 허용하거나 거부하는 문서
- IAM 정책을 사용하여 사용자가 리소스에 액세스

AWS IAM 그룹 (Group)

- IAM 사용자의 모음
- 그룹에 IAM 정책을 할당하면 해당 그룹의 모든 사용자에게 정책에 지정된 권한 부여

AWS Identity and Access Management (IAM)

AWS IAM 역할 (Role)

- IAM 역할은 특정 작업을 허용하거나 거부하는 권한이 연결되어 있으며, 임시로 권한에 액세스하기 위한 자격 증명
- IAM 사용자, 애플리케이션 또는 서비스가 IAM 역할을 가지려면 먼저 해당 역할로 전환할 수 있는 권한 필요
- IAM 역할을 수행한다는 것은 이전 역할의 모든 권한을 포기하고 새 역할에 지정된 권한 수행

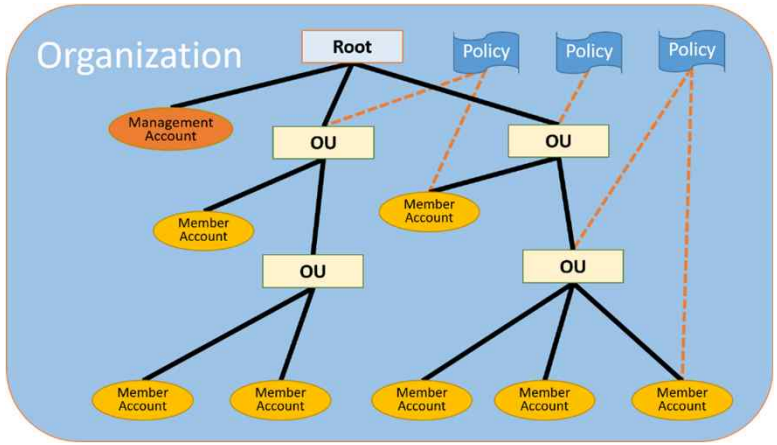
Multi-Factor Authentication

- 신원을 확인 하기 위해서 여러 가지 정보를 제공하도록 요구 하는 인증 방법
- 암호 입력 후 email, SMS, OTP 등의 두 번째 인증 요구

AWS Identity and Access Management (IAM)

- **AWS Organizations**
  - 여러 AWS 계정을 사용자가 생성하고 중앙에서 관리하는 단일 조직으로 통합할 수 있는 계정 관리 서비스
  - AWS Organizations가 조직의 모든 계정에 대한 상위 컨테이너 루트를 자동 생성
  - 서비스 제어 정책(Service Control Policy, SCP)을 사용하여 조직의 계정에 대한 권한을 중앙에서 제어
  - SCP를 사용하면 각 계정의 사용자 및 역할이 액세스할 수 있는 AWS 서비스, 리소스 및 개별 API 작업을 제한 가능

- **조직 단위 (Organization Unit)**
  - AWS Organizations에서는 계정을 조직 단위(OU)로 그룹화하여 비슷한 비즈니스 또는 보안 요구 사항이 있는 계정을 효율적으로 관리
  - OU에 정책을 적용하면 OU의 모든 계정이 정책에 지정된 권한을 자동으로 상속
  - 개별 계정을 OU로 구성하면 특정 보안 요구 사항이 있는 워크로드 또는 애플리케이션을 보다 간편하게 격리



Amazon CloudWatch

Cloudwatch란?

- AWS 클라우드 리소스와 AWS에서 실행되는 애플리케이션을 위한 모니터링 서비스
- 리소스 및 애플리케이션에 대해 측정할 수 있는 변수인 '지표'를 수집하고 추적 가능
- 사용중인 모든 AWS 서비스에 대한 지표가 자동으로 표시되며, 사용자 지정 대시보드를 통해 사용자 지정 애플리케이션에 대한 지표를 표시하고 지정 집합 표시 가능
- '지표'는 Cloudwatch에 게시된 시간 순서별 데이터 요소 세트이며, 모니터링할 변수(가령 EC2의 CPU 사용량은 EC2가 제공하는 하나의 지표)
- 기본 모니터링과 세부 모니터링으로 나뉘며, 각각 5분과 1분 주기로 수집함
- 기본 모니터링은 자동활성화이지만, 세부 모니터링은 선택사항임
- 기본적으로 CPU, Network, Disk, Status Check 등을 수집함(Memory 항목이 없음)
- 지표 데이터의 보존기간은 다음과 같음
  - 기간 60초 미만의 경우, 3시간
  - 기간 60초의 경우, 15일
  - 기간 300초의 경우, 63일
  - 기간 3600초의 경우, 455일
- AWS CLI 혹은 API를 이용하여, Cloudwatch에 사용자 정의 지표 게시 가능
- '경보' 기능을 사용하여 어떤 지표가 일정기간동안 일정값에 도달할 경우 각 서비스가 취해야할 행동을 정의할 수 있음
  - 모니터링하기로 선택한 측정치가 정의한 임계값을 초과할 경우 하나 이상의 자동화 작업을 수행하도록 구성
  - EC2의 경우, 경보에 따라 인스턴스 중지, 복구, 종료, 재부팅 가능

Amazon CloudWatch

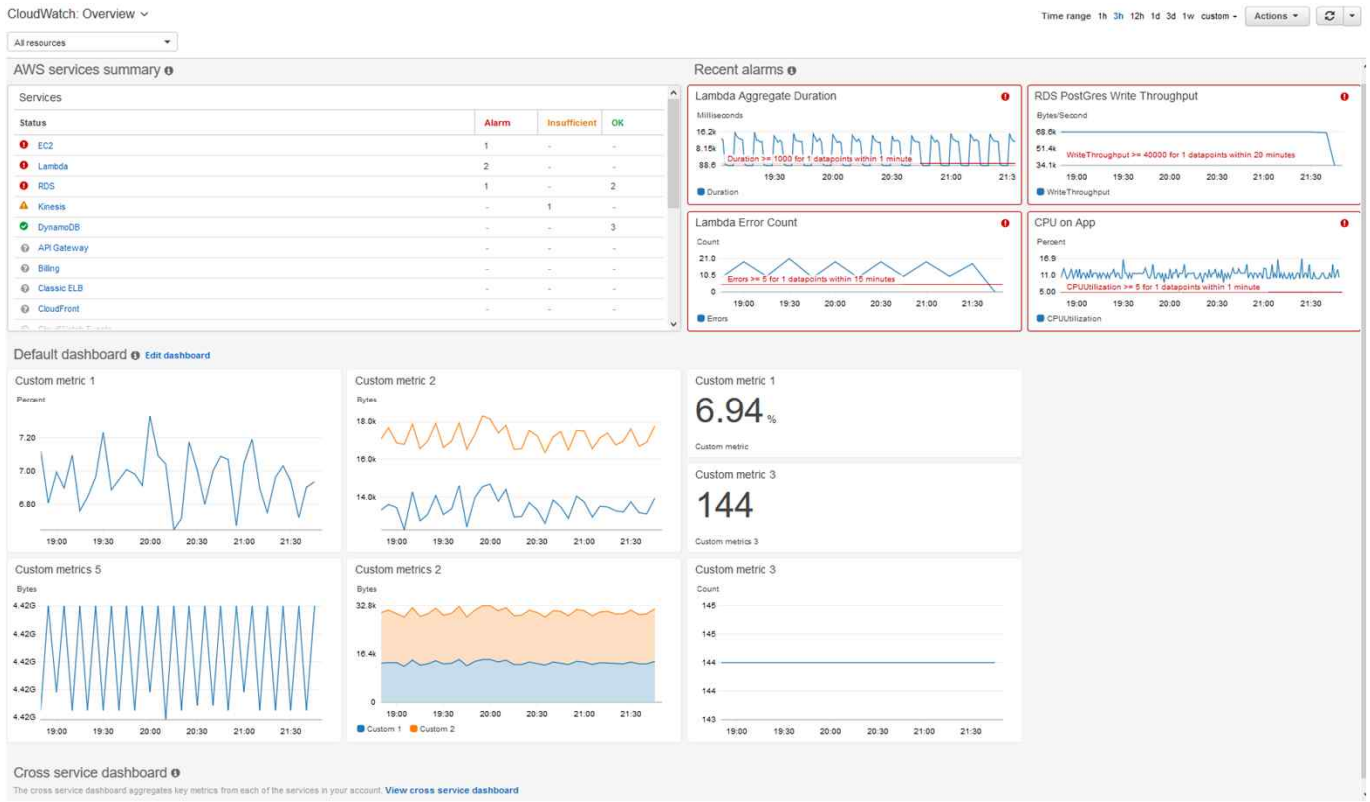
Cloudwatch Agent

- EC2에 Agent를 설치하게 되면 더 많은 시스템 수준 지표를 수집할 수 있음
- 온프레미스 서버 또한 Cloudwatch Agent 사용 가능
- 여기에 Memory 항목이 포함
  - 메모리 항목은 사용자 지정 지표로 수집가능하며, 이는 Agent를 통해 수집함
- Cloudwatch Agent는 로그를 수집할 수 있으며, Cloudwatch Logs 기능 사용 가능

Cloudwatch Logs

- EC2(Agent에서 수집된), CloudTrail, Route 53, VPC Flow log 등 기타 소스에서 발생한 로그 파일을 모니터링, 저장 및 액세스하는 기능
- Cloudwatch Agent를 사용하여 로그를 수집함
- Cloudwatch Log Insights를 사용하여 CloudWatch Logs에서 로그 데이터를 대화식으로 검색해 분석할 수 있음
- Agent는 기본적으로 5초마다 로그 데이터를 전송함

Amazon CloudWatch



* kinesis: 실시간 데이터 분석 처리 시스템

https://docs.aws.amazon.com/ko_kr/AmazonCloudWatch/latest/monitoring/GettingStarted.html

Amazon CloudWatch

Cloudwatch Events

- AWS 각 서비스의 이벤트가 사용자가 지정한 이벤트 패턴과 일치하거나 일정이 트리거 될 경우,
  사용자가 원하는 기능을 발동 시키도록 하는 기능

- 이벤트 소스와 대상으로 나뉨
  - 이벤트 소스 : AWS 환경에서 발생하는 이벤트이며, 가령 S3의 경우 오브젝트 등록, 삭제 등을 들 수 있음
  - 대상 : 이벤트 발생시 해야 할 행동을 정의하는 것이며, SNS 전송 혹은 람다, SQS 게시 등을 설정할 수 있음

- 이벤트 소스에 해당하는 규칙이 트리거 될 경우 대상에 해당하는 서비스를 실행시킴
  - 이벤트가 시스템에 생성해 둔 규칙과 일치하는 경우, AWS Lambda 함수를 자동으로 호출하고,
    해당 이벤트를 Amazon Kinesis 스트림에 전달하고, Amazon SNS 주제를 알림

AWS Artifact

AWS Artifact

- AWS Artifact는 ISO 인증, 신용카드 업계(PCI) 및 SOC(Service Organization Control) 등과 같은 AWS 보안 및 규정 준수 보고서 및 일부 온라인 계약에 대한 온디맨드 액세스를 제공하는 서비스

AWS Artifact Agreements

- AWS 서비스 전체에서 특정 유형의 정보를 사용하기 위해 AWS와 계약을 체결 → AWS Artifact Agreements
- 개별 계정 및 AWS Organizations 내 모든 계정에 대한 계약을 검토, 수락 및 관리

AWS Artifact Reports

- 외부 감사 기관이 작성한 규정 준수 보고서를 제공하고 이러한 감사 기관에서 AWS가 다양한 글로벌, 지역별, 산업별 보안 표준 및 규정을 준수했음을 검증
- 릴리스 된 최신 보고서가 반영되어 항상 최신 상태로 유지하고 감사 또는 규제 기관에 AWS 보안 제어 항목의 증거로서 AWS 감사 아티팩트를 제공

고객 규정 준수 센터

- **AWS 보안 규정 준수 프로그램**
  - AWS 규정 준수에 대해 자세히 알아볼 수 있는 리소스 제공
  - 고객 규정 준수 사례를 읽고 규제 대상 업종의 기업들이 다양한 규정 준수, 거버넌스 및 감사 과제를 어떻게 해결했는지 확인
  - 다음과 같은 주제에 관한 규정 준수 백서 및 설명서에 액세스
    - 주요 규정 준수 질문에 대한 AWS 답변
    - AWS 위험 및 규정 준수 개요
    - 보안 감사 체크리스트
  - 감사자 학습 경로 (Auditor Learning Path) 제공
    - AWS 클라우드에서 보안을 구축하는 방법과 AWS 도구를 이용해 AWS 환경에서 감사를 실시하는 데 필요한 정보를 수집하는 방법에 대한 기본 및 고급 교육을 제공

https://aws.amazon.com/ko/compliance/customer-center/

AWS Shield

서비스 거부 공격 (Denial of Service Attack, DoS)

- 시스템을 악의적으로 공격해 해당 시스템의 리소스를 부족하게 하여 원래 의도된 용도로 사용하지 못하게 하는 공격
- 웹 사이트 또는 애플리케이션이 과부하가 걸려 더 이상 응답할 수 없을 때까지 과도한 네트워크 트래픽으로 플러드(flood)

분산 서비스 거부 공격 (Distributed Denial of Service Attack, DDoS)

- 여러 대의 공격자를 분산적으로 배치해 동시에 서비스 거부 공격을 하는 방법

AWS Shield

- DDoS 공격으로부터 애플리케이션을 보호하는 서비스
- **AWS Shield Standard**
  - 모든 AWS 고객을 자동으로 보호하는 무료 서비스
  - AWS 리소스를 가장 자주 발생하는 일반적인 DDoS 공격으로부터 네트워크 트래픽이 애플리케이션으로 들어오면 AWS Shield Standard는 다양한 분석 기법을 사용하여 실시간으로 악성 트래픽을 탐지하고 자동으로 완화
- **AWS Shield Advanced**
  - 상세한 공격 진단 및 정교한 DDoS 공격을 탐지하고 완화할 수 있는 기능을 제공하는 유료 서비스
  - **Amazon CloudFront, Amazon Route 53, Elastic Load Balancing**과 같은 다른 서비스와도 통합
  - 복잡한 DDoS 공격을 완화하기 위한 사용자 지정 규칙을 작성하여 AWS Shield를 **AWS WAF**와 통합 가능

AWS Key Management (AWS KMS)

- **AWS Key Management Service(AWS KMS) 이란?**
  - Key Management System는 말그대로 'key'를 관리하는 시스템으로 데이터를 암호화하고 복호화하는 기능을 담당함
  - EBS, S3, RDS, EFS, SNS 등의 서비스에서 데이터 암/복호화 기능을 제공
  - 다른 서비스와 통합되어 사용됨
  - KMS를 사용하여 암호화를 관리할 경우 키가 외부로 유출되는 위험이 적으며, AWS가 직접 키의 안전을 책임짐
  - 다음 3가지 유형의 키로 나뉨
    - AWS 관리형 키
    - 고객 관리형 키
    - 사용자 키 스토어

AWS Key Management (AWS KMS)

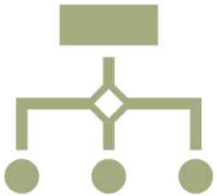
고객 마스터 키(Customer Master Key, CMK)

데이터를 암호화하는데 사용하는 데이터 키의 생성에 관여하는 키

AWS 서비스가 암호화를 시작할 때 CMK의 생성을 요청한 후 데이터 암호화를 시작

즉, CMK를 생성한 후에 데이터 키를 생성하고 데이터 암호화를 시작함

위의 3가지 유형 키는 CMK를 누가 관리 하느냐 에 따라 달라짐



AWS 관리형 키(CMK)

AWS가 직접 CMK를 생성, 관리하는 서비스

사용자가 CMK에 대한 제어권한이 없음

AWS가 주기적으로 CMK를 변경하여 사용함

고객 관리형 키를 쓰고 있지 않는데, 암호화를 사용 중이라면 AWS 관리형 키를 이용해 암호화하고 있는 것

AWS Key Management (AWS KMS)

고객 관리형 키(CMK)

- 고객이 직접 CMK를 생성하고 관리하는 서비스
- 키의 활성화, 비활성화, 삭제 등 제어 권한을 가짐
- IAM을 이용하여 CMK에 접근할 주체를 정할 수 있음

사용자 지정 키 스토어

- CMK를 KMS가 아닌 CloudHSM에 저장하여 사용하는 방식
- CloudHSM 클러스터가 생성되어 있어야 사용 가능

* CloudHSM(Hardware Security Module)

AWS Web Application Firewall (AWS WAF)

- **AWS Web Application Firewall (AWS WAF)**
  - 가용성에 영향을 주거나, 보안을 위협하거나, 리소스를 과도하게 사용하는 일반적인 웹 공격으로부터 웹 애플리케이션이나 API를 보호하는 데 도움이 되는 웹 애플리케이션 방화벽
  - SQL injection 또는 크로스 사이트 간 스크립팅과 같은 일반적인 공격 패턴을 차단하는 보안 규칙 및 사용자가 지정하는 웹 보안 규칙을 정의함으로써 어떤 트래픽에 웹 애플리케이션에 대한 액세스를 허용하거나 차단할지 제어 가능
  - 몇 분 이내에 새로운 규칙이 배포되므로 변화하는 트래픽 변화에 신속하게 대응
  - 사전 약정 없이 요금은 배포한 규칙 수와 애플리케이션이 수신한 웹 요청 수를 기준으로 부과
  - Amazon CloudFront, EC2에서 실행되는 웹 서버 또는 오리진 서버의 전방에 배치된 Application Load Balancer, REST API용 Amazon API Gateway, 또는 GraphQL API용 AWS AppSync등에 WAF 배포 가능

AWS Inspector & GuardDuty

AWS Inspector

- 자동화된 보안 평가를 실행하여 애플리케이션의 보안 및 규정 준수를 개선할 수 있는 서비스
- Amazon EC2 인스턴스에 대한 오픈 액세스, 취약한 소프트웨어 버전 설치와 같은 보안 모범 사례 위반 및 보안 취약성 검사 수행한 후에 보안 탐지 결과 목록을 제공

→ 심각도 수준에 따라 우선 순위가 결정되고 각 보안 문제에 대한 자세한 설명 및 권장 해결 방법 포함

→ AWS는 제공된 권장 사항으로 모든 잠재적 보안 문제가 해결됨을 보장하지 않음

→ 공동 책임 모델에 따라 고객은 AWS 서비스에서 실행되는 애플리케이션, 프로세스 및 도구의 보안에 대한 책임

Amazon GuardDuty

- AWS 환경 내의 네트워크 활동 및 계정 동작을 지속적으로 모니터링하여 위협을 식별하는 지능형 위협 탐지 서비스
- GuardDuty는 VPC Flow Logs 및 DNS 로그를 비롯한 여러 AWS 소스의 데이터를 지속적으로 분석
- 위협을 탐지한 경우 AWS Management Console에서 위협에 대한 자세한 탐지 결과를 검토
- GuardDuty 보안 탐지 결과에 대한 응답으로 자동으로 문제 해결 단계를 수행하도록 AWS Lambda 함수를 구성 가능

AWS CloudTrail

감사(Audit)

- IT에서 트랜잭션을 감사하는 기능은 대부분의 규정 준수 구조에서 필수적인 요소
- AWS는 모든 작업이 API 호출을 통해서 이루어 지기 때문에 감사 정보 획득이 용이

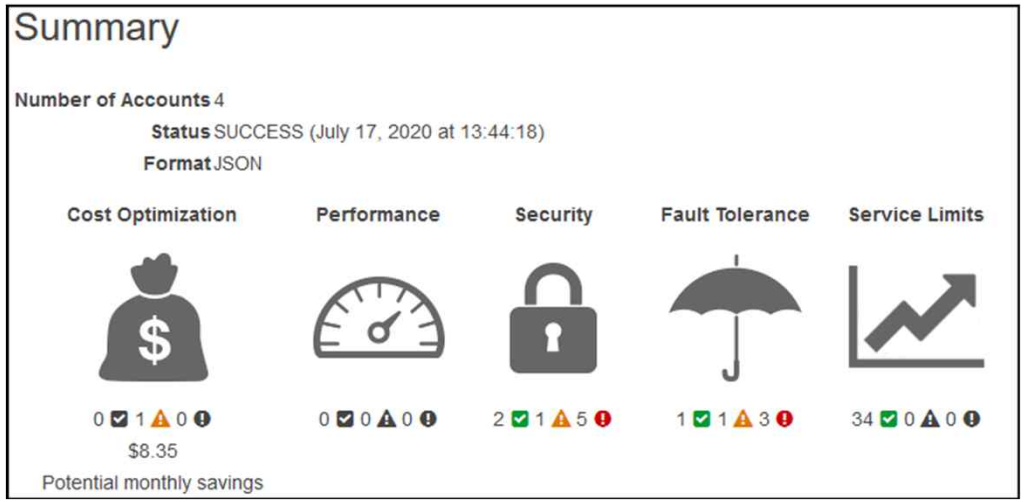
AWS CloudTrail

- 포괄적인 API 감사 도구
- CloudTrail을 사용하여 애플리케이션 및 리소스에 대한 사용자 활동 및 API 호출의 전체 내역 확인 가능
- 이벤트는 일반적으로 API 호출 후 15분 이내에 CloudTrail에서 업데이트
- API 호출이 발생한 날짜 및 시간, 작업을 요청한 사용자, API 호출에 포함된 리소스 유형 등을 지정하여 이벤트를 필터링
- CloudTrail은 이러한 로그를 안전한 S3 버킷에 무제한으로 저장
- **CloudTrail Insights (optional)** : AWS 계정에서 비정상적인 API 활동을 자동으로 감지

AWS Trusted Advisor

- **AWS Trusted Advisor**
  - AWS 환경을 검사하고 AWS 모범 사례에 따라 실시간 권장 사항을 제시하는 웹 서비스

  - 5가지 범주를 AWS 모범 사례와 비교
    - **비용 최적화 (Cost Optimization)**
    - **성능 (Performance)**
    - **보안 (Security)**
    - **내결함성 (Fault Tolerance)**
    - **서비스 한도 (Service Limits)**



AWS Cognito

AWS Cognito

- 웹 그리고 모바일 앱에 대한 사용자 인증, 액세스 권한 부여, 사용자 관리를 제공하는 서비스
  - 앱 로그인, 유저 권한 제공 등을 의미

- SAML 또는 OpenID Connect를 지원하는 외부 자격 증명 공급자, 소셜 자격 증명 공급자(Facebook, Twitter, Amazon) 등과 연동 가능

- 개발자가 손쉽게 앱에 사용자 관리와 사용자의 디바이스 간 데이터 동기화 기능을 제공할 수 있음

- User Pool과 Identity Pool로 나뉨

- Cognito를 사용하고 있지 않다면, AWS STS의 AssumeRoleWithWebIdentity를 호출해야 함
  - AWS 리소스에 대한 액세스를 제어할 수 있는 임시 보안 자격 증명을 생성하여 신뢰할 수 있는 사용자에 제공

AWS Cognito

User Pool

- Cognito의 사용자 디렉터리
- User Pool이 있으면 사용자는 Cognito를 통해 웹 혹은 앱에 로그인 가능
- Cognito의 User Pool로 사용자를 생성하고 로그인을 허용할 수 있으며, 소셜 자격 증명(Google, Facebook, Amazon) 혹은 SAML 자격 증명 공급자를 통해서도 로그인 가능
- 사용자 인증 후, Cognito는 JSON 웹 토큰(JWT)를 발행하며 이를 사용하여 API에 대한 액세스, 자격 증명을 수행하거나 AWS 자격 증명으로 교환 가능

Identity Pool

- AWS 서비스에 액세스할 수 있는 임시 자격 증명 제공
- 사용자에게 S3 Bucket 또는 Dynamo DB 테이블과 같은 AWS 리소스에 대한 액세스 권한을 부여함
- User Pool의 사용자를 비롯하여, 외부 자격 증명 공급자, 소셜 자격 증명 공급자(Facebook, Twitter, Amazon) 등과 연동하여 임시 자격 증명 제공 가능
- AWS IAM를 통해 사용자의 권한을 제어할 수 있음
- 인증되지 않은 사용자(Guest)에게도 권한 부여가 가능하며 IAM을 통해 권한 범위를 정할 수 있음

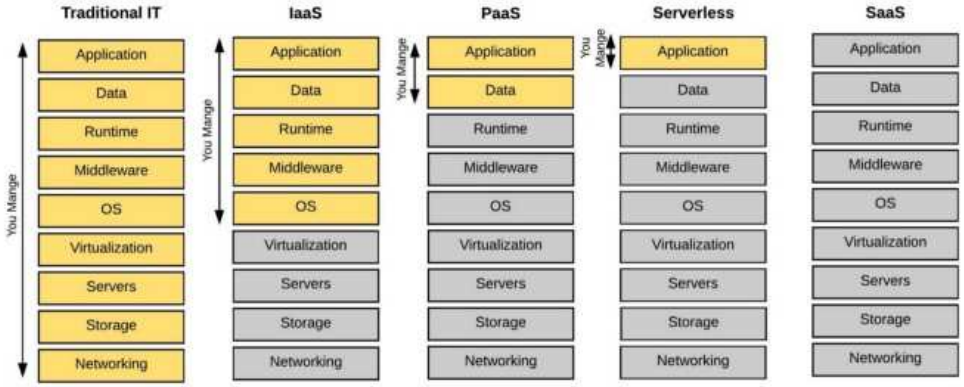
User Pool vs Identity Pool

- User Pool은 인증(자격 증명 확인)을 위한 서비스이지만, Identity Pool은 권한 부여(액세스 제어)를 위한 서비스

13. Lambda

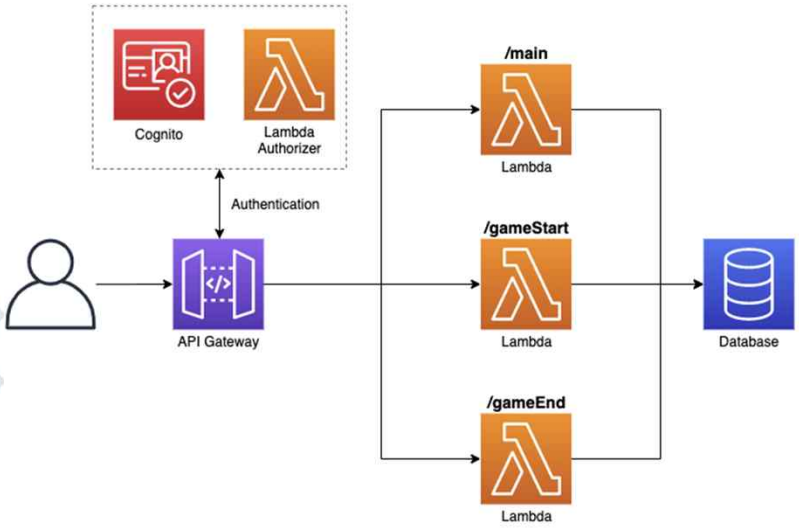
AWS Lambda

- **서버 리스 컴퓨팅 (Serverless Computing)란?**
  - EC2를 사용할 때는 사용자가 새 소프트웨어 패키지가 출시되면 인스턴스 패치, 규모 조정, 소프트웨어의 가용성 등 많은 부분의 관리 요구
  - 서버리스 아키텍처의 핵심으로 서버에 대한 걱정 없이 코드를 실행할 수 있도록 제공하는 서비스
  - 애플리케이션을 호스팅하는 기본 인프라나 인스턴스를 보거나 이용할 수 없다는 의미
  - 프로비저닝, 규모 조정, 고가용성 및 유지 관리와 관련된 모든 기본 환경 관리를 대신 처리
  - 쉽게 말하자면 OS가 필요없는 컴퓨팅



AWS Lambda

- **Lambda란?**
  - AWS Lambda는 서버를 프로비저닝 하거나 관리할 필요 없이 코드를 실행할 수 있는 서비스
  - 사용한 컴퓨팅 시간에 대해서만 비용을 지불 → 코드를 실행하는 동안에만 요금이 부과
  - 수신 트리거가 한 개 든 1,000개가 있든 Lambda는 수요에 맞게 함수의 규모를 조정
  - 코드를 15분 미만으로 실행하도록 설계되어 있기 때문에 딥 러닝 같은 실행 시간이 긴 프로세스는 적합 하지 않음



https://aws.amazon.com/ko/lambda/?nc2=h_ql_prod_fs_lbd

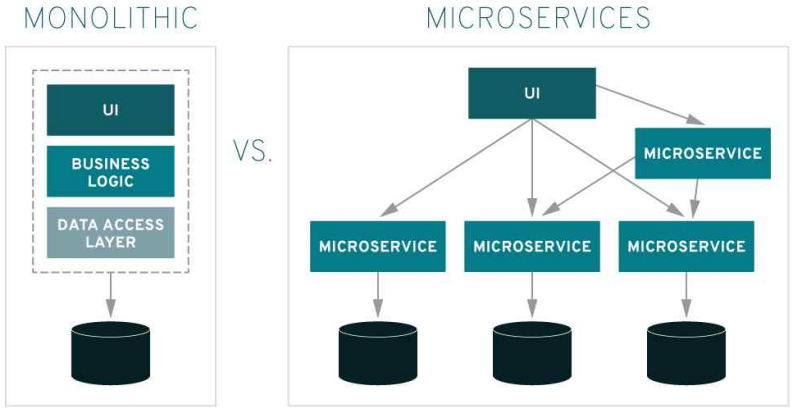
AWS Lambda

- **Lambda 의 특징**
  - AWS에서는 2021년 2월 현재 C#, PowerShell, Go, Java, JavaScript, Python, Ruby을 네이티브 하게 지원
  - 람다 함수는 런타임에 따라서 Amazon Linux 혹은 Amazon Linux 2에서 실행
  - 람다는 기본적으로 이벤트 드라이븐 방식으로 동작
  - API Gateway와 Elastic Load Balancer의 HTTP 요청을 처리할 수 있으며, S3 객체, DynamoDB, Kinesis 등에서 발생하는 이벤트를 트리거로 실행하는 것도 가능
  - 람다의 컴퓨팅 자원은 128MB와 3,008MB 사이에서 64MB 단위로 결정
  - 메모리를 크게 지정할 수록 요금이 올라감
  - 메모리 할당량은 성능에도 영향을 주며, AWS 람다 공식 FAQ 문서에 따르면 CPU 용량과 기타 리소스가 메모리 크기에 비례해서 증가
  - 예를 들어 람다 함수에 256MB를 할당한 경우 128MB 할당한 경우보다 CPU 용량은 2배

14. AWS Products

모놀리식 아키텍처와 마이크로서비스 아키텍처 비교

- **모놀리식 아키텍처**
  - 모든 프로세스가 긴밀하게 결합되고 단일 서비스로 실행 따라서
  - 애플리케이션의 한 프로세스에 대한 수요가 급증하면 해당 아키텍처 전체를 확장
  - 복잡성으로 인해 실험에 제한을 받고 새로운 아이디어를 구현하기 어려움
  - 종속 관계를 이루며 긴밀하게 결합된 많은 프로세스로 인해 단일 프로세스의 실패로 인한 영향이 증가 → 애플리케이션 가용성 위험
- **마이크로서비스 아키텍처**
  - 애플리케이션이 독립적인 구성 요소로 구축되어 각 애플리케이션 프로세스가 서비스로 실행
  - 경량 API를 사용하여 잘 정의된 인터페이스를 통해 통신
  - 서비스는 비즈니스 기능을 위해 구축되며 서비스마다 한 가지 기능을 독립적으로 수행하고, 비즈니스 수요에 맞춰 업데이트, 배포 및 확장



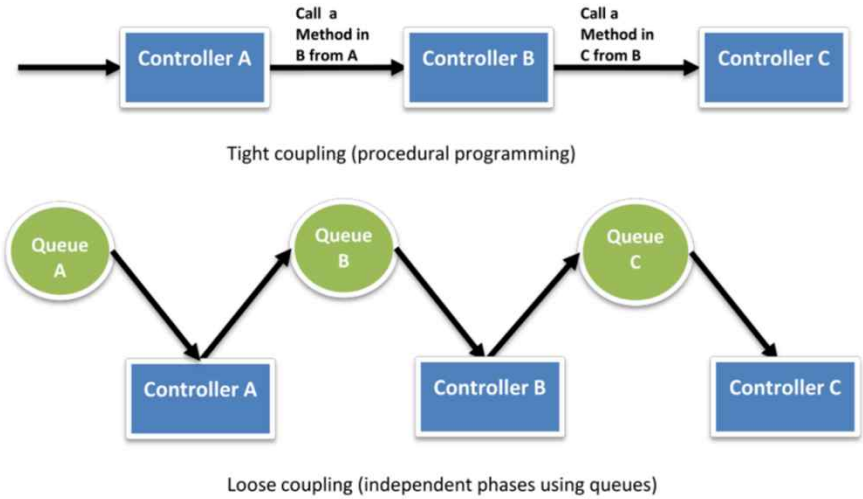
https://www.redhat.com/ko/topics/microservices/what-are-microservices

Messaging and Queueing

- **tighty coupled architecture**
  - 밀접 결합된 아키텍처의 대표적인 특징은 구성 요소 하나가 고장 나거나 변경되면 다른 구성 요소나 심지어 전체 시스템에 문제가 발생한다는 점
  - 예를 들어 애플리케이션 A가 애플리케이션 B에 메시지를 직접 보낼 때 애플리케이션 B에 장애가 발생하여 메시지를 받을 수 없게 되면 애플리케이션 A에도 오류가 표시
- **loosely coupled architecture**
  - 특정 구성 요소에 장애가 발생하면 구성 요소가 격리되기 때문에 전체 시스템 장애로 확장되지 않는다.
  - 느슨한 결합이 된 아키텍처를 사용하도록 애플리케이션은 애플리케이션 A가 메시지를 대기 열로 전달한 후 애플리케이션 B가 처리

→ 애플리케이션 B에 장애가 발생해도 애플리케이션 A에는 중단이 발생하지 않음

→ 전송 중인 메시지는 여전히 대기 열로 전송될 수 있으며 처리될 때까지 대기 열에 존재



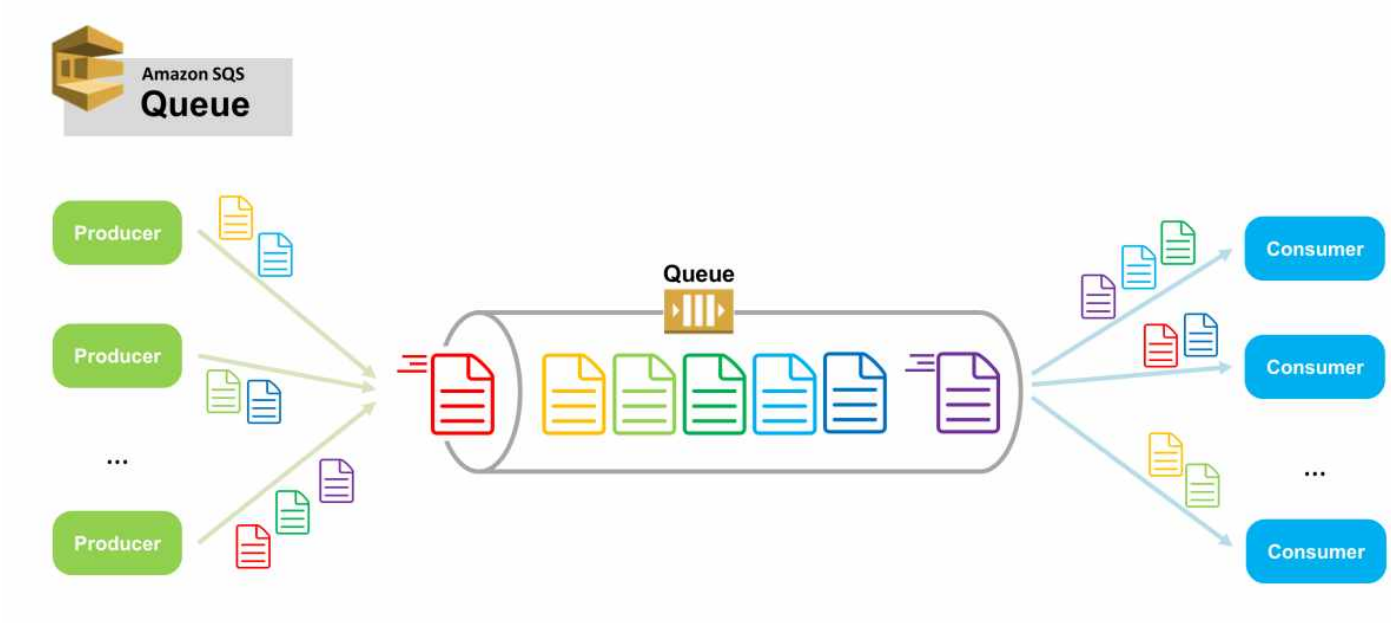
https://adriangrigoras.com/blog/aws-cloud-architecture-design-principles/

Amazon Simple Queue Service (Amazon SQS)

- **Amazon Simple Queue Service(Amazon SQS) 란?**
  - AWS 최초의 서비스
  - 2004년 런칭 시작
  - 서버와 서버 혹은 서버와 다른 엔드포인트 통신 중 장애시 모든 요청을 잃어버릴 수 있는 사태를 대비할 수 있음
  - 서버1이 서버2로 요청을 보내는 경우, 서버2에 직접 보내지 않고 큐(SQS)에 담아두면, 서버2가 우체통 열듯 큐(SQS)를 열어 메시지를 확인하고 수행함
  - 수행한 메시지는 서버2가 확인 후 삭제
  - 각 메시지는 최대 256KB의 텍스트로 구성될 수 있음
  - 최대 14일까지 저장 가능하나, 기본값은 4일
  - 처리되지 않은 서비스는 다시 큐에 보존되며 다른 요청자가 열람할 수 있음
  - 즉, 서비스 요청을 저장하고 대기열을 만들어 처리할 수 있도록 하는 서비스
  - Standard 대기열과 FIFO 대기열로 나뉨
    - Standard 대기열 : 표준 서비스로 초당 무제한에 가까운 요청을 처리할 수 있으며 최소 한 번의 요청을 처리하나 순서가 보장되지 않음
    - FIFO 대기열 : 선입선출을 지키는 대기열, 초당 300개까지 처리 가능

Amazon Simple Queue Service (Amazon SQS)

- **Amazon Simple Queue Service(Amazon SQS) 란?**



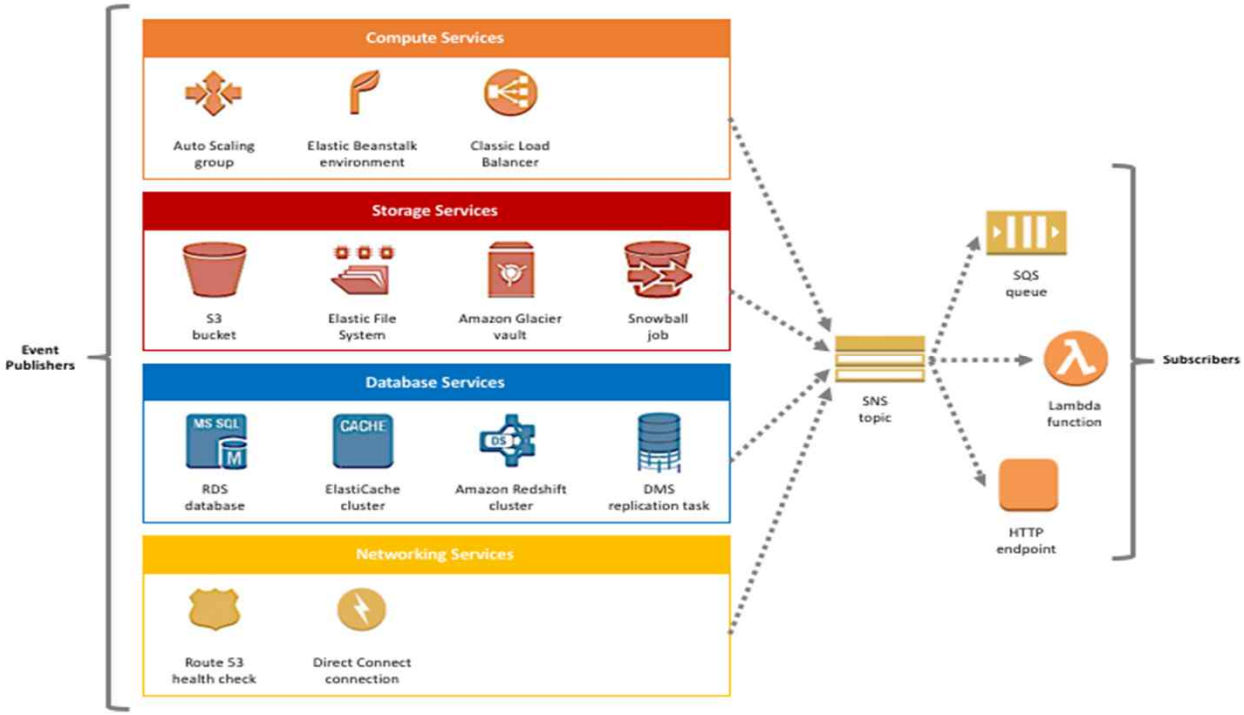
https://ctoasaservice.org/2019/08/30/the-aws-messaging-stack-sqs-sns-kinesis/

Amazon Simple Notification Service(Amazon SNS)

- **Amazon Simple Notification Service(Amazon SNS)란?**
  - '구독'중인 엔드포인트 혹은 사용자에게 메시지를 보내는 서비스
  - 주제(Topic)와 구독(Subscription)으로 나뉨
  - 하나의 주제에 다수의 구독자로 이루어질 수 있음
  - 주제는 다음과 같은 요소를 설정함
    - 메시지를 게시할 수 있는 대상과 받을 수 있는 대상
    - 암호화 여부
    - 메시지 전송 정책
  - 구독은 메시지를 받을 대상을 설정함
    - HTTP(웹)
    - SQS
    - 이메일
    - Lambda(메시지를 전송하여 Lambda 호출 가능)
    - 모바일 애플리케이션
  - '푸시' 기반 서비스

Amazon Simple Notification Service(Amazon SNS)

- **Amazon Simple Notification Service(Amazon SNS)란?**



https://aws.amazon.com/ko/blogs/korea/event-driven-computing-with-amazon-sns-compute-storage-database-and-networking-services/

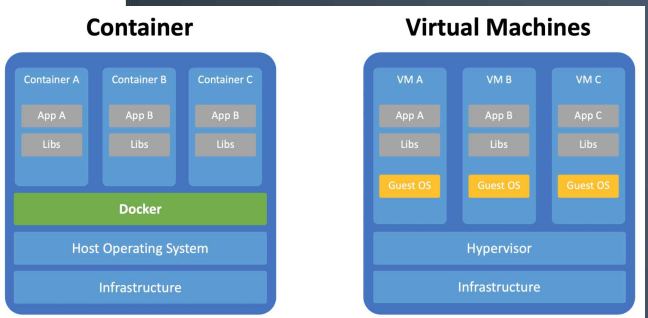
SQS vs. SNS

SNS는 Push 기반 서비스이나
SQS는 Pull 기반 서비스임

즉 SNS는 와서 전달하고,
SQS는 가서 찾아와야 함

Amazon Elastic Container Service(Amazon ECS)

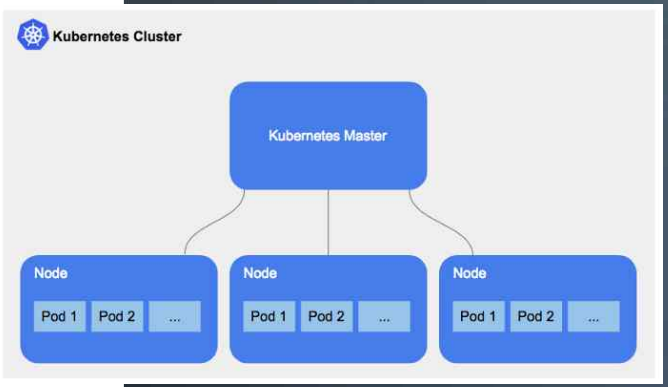
- **Amazon Elastic Container Service(Amazon ECS) 란?**
  - AWS에서 컨테이너식 애플리케이션을 실행하고 확장할 수 있는 확장성이 뛰어난 고성능 컨테이너 관리 시스템
  - Amazon ECS는 Docker 컨테이너를 지원
  - Docker는 많은 곳에서 사용하는 플랫폼으로 운영 체제 수준 가상화를 사용하여 컨테이너에 소프트웨어를 제공
  - 애플리케이션과 관련 종속성 및 애플리케이션에서 실행해야 하는 모든 구성을 모아 놓은 코드 패키지



https://yceffort.kr/2019/08/06/experience-docker-aws-ecs

Amazon Elastic Kubernetes Service(Amazon EKS)

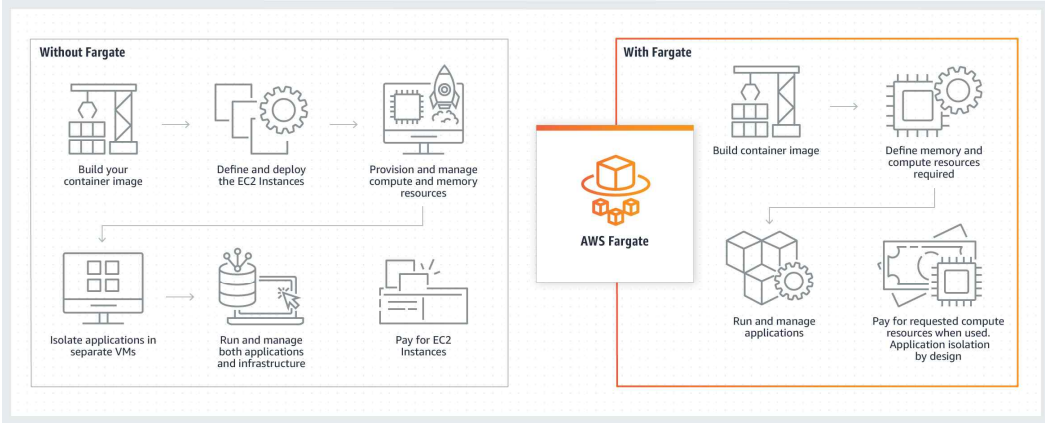
- **Amazon Elastic Kubernetes Service(Amazon EKS)란?**
  - Amazon EKS는 AWS에서 Kubernetes를 실행하는 데 사용할 수 있는 완전 관리형 서비스
  - kubernetes는 컨테이너형식의 애플리케이션을 대규모로 배포하고 관리하는데 사용할 수 있는 오픈 소스 오케스트레이션 소프트웨어
  - Amazon EC2를 사용해 Kubernetes 인프라를 직접 관리하거나 Amazon EKS를 통해 자동으로 프로비저닝되고 관리되는 Kubernetes 컨트롤 플레인을 사용 가능



https://medium.com/@tomerf/so-you-want-to-configure-the-perfect-db-cluster-inside-a-kubernetes-cluster-a4d2c26aca7a

AWS Fargate

- **AWS Fargate란?**
  - AWS Fargate는 컨테이너용 서버리스 컴퓨팅 엔진으로, Amazon ECS와 Amazon EKS에서 작동
  - AWS Fargate를 사용하는 경우 서버를 프로비저닝하거나 관리할 필요 없음.
  - AWS Fargate는 자동으로 서버 인프라를 관리하며 컨테이너를 실행하는 데 필요한 리소스에 대해서만 비용 지불



https://aws.amazon.com/ko/fargate

AWS 인공지능(AI) & 기계 학습 (ML) 서비스

인공 지능(AI) 기반 서비스 제공

- **Amazon Transcribe** : 음성을 텍스트로 변환
- **Amazon Comprehend** : 텍스트에서 패턴을 검색
- **Amazon Fraud Detector** : 잠재적인 온라인 사기 행위를 식별
- **Amazon Lex** : 음성 및 텍스트 챗 봇 빌드

기계 학습

- 기존의 기계 학습(ML) 개발은 복잡하고, 비용이 많이 들고, 시간이 오래 걸리고, 오류 발생이 빈번
- AWS는 이 프로세스에서 어려운 작업을 제거하여 ML 모델을 신속하게 빌드, 훈련, 배포
- → **Amazon SageMaker** 를 제공

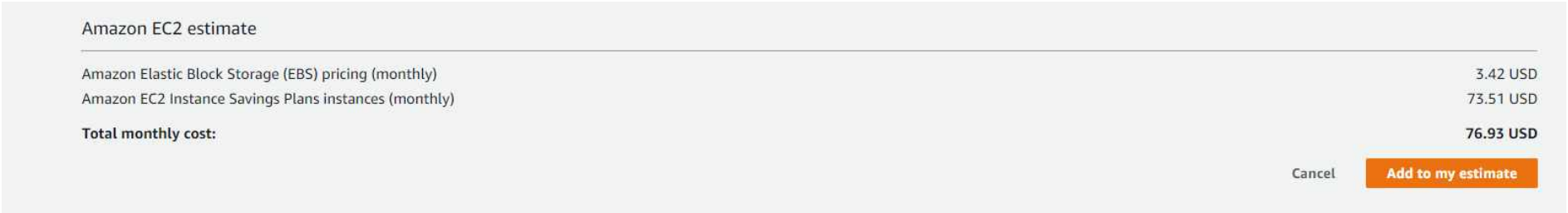
15. Billing and Support

요금 (Billing)

- **AWS 요금**
  - **사용량에 따라 지불**
    - 장기 계약 또는 복잡한 라이선스 없이 각 서비스에서 실제로 사용한 리소스만큼 지불
  - **예약하는 경우 지불 비용 감소**
    - 일부 서비스는 온디맨드 인스턴스 요금에 비해 상당한 할인을 제공하는 예약 옵션을 제공
    - Amazon Ec2 Instance Savings Plans
  - **더 많이 사용하여 더 적은 비용 지불**
    - 일부 서비스는 계층화된 요금을 제공하므로 사용량이 증가함에 따라 점차 단위 비용이 낮아짐
    - Amazon S3 스토리지 공간을 많이 사용할수록 GB당 비용이 낮아짐
  - **AWS 규모가 커짐에 따라 더 적은 비용 지불**
  - **참고 사이트 : https://aws.amazon.com/ko/pricing/**

프리티어

- **AWS 프리티어 (Free tier)**
  - 계정 생성 후 지정된 기간 동안 무료로 특정 서비스를 사용
  - 3가지 유형 제공
  - **상시 무료** : 해당 유형의 제품은 만료되지 않으며 모든 AWS 고객에게 제공
    - AWS Lambda에서는 매월 무료 요청 1백만 건과 최대 320만 초의 컴퓨팅 시간
    - Amazon DynamoDB에서는 매월 25GB의 무료 스토리지를 사용
  - **12개월 무료** : AWS에 처음 가입한 날로부터 12개월 동안 무료로 제공
    - Amazon S3 Standard 스토리지
    - 월별 Amazon EC2 컴퓨팅 시간 한도
    - Amazon CloudFront 데이터 전송량
  - **평가판** : 단기 무료 평가판 제품은 특정 서비스를 활성화한 날짜부터 시작하며 각 평가판의 기간은 일수 또는
- 서비스 사용량을 기준으로 다를 수 있다.
  - Amazon Inspector는 90일 무료 평가판 제공
  - Amazon Lightsail은 30일 동안 750 시간



AWS 요금 계산기 (Pricing Calculator)

- **AWS 요금 계산기**
  - 솔루션을 구축하기 전에 솔루션을 모델링하고 추정을 바탕으로 사전에 비용을 미리 계산
  - 가격대 및 계산을 탐색하고 필요에 맞는 사용 가능한 인스턴스 유형 및 계약 조건 확인 가능
  - https://calculator.aws

AWS 결제 및 비용 대시 보드 (Billing Dashboard)

AWS Billing Dashboard

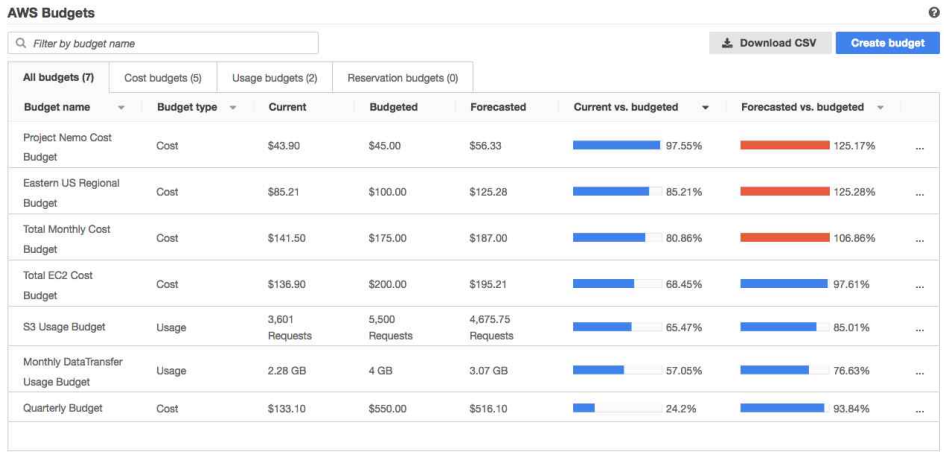
- AWS 청구서를 결제하고, 사용량을 모니터링하고, 비용을 분석 및 제어

통합 결제 (Consolidated billing)

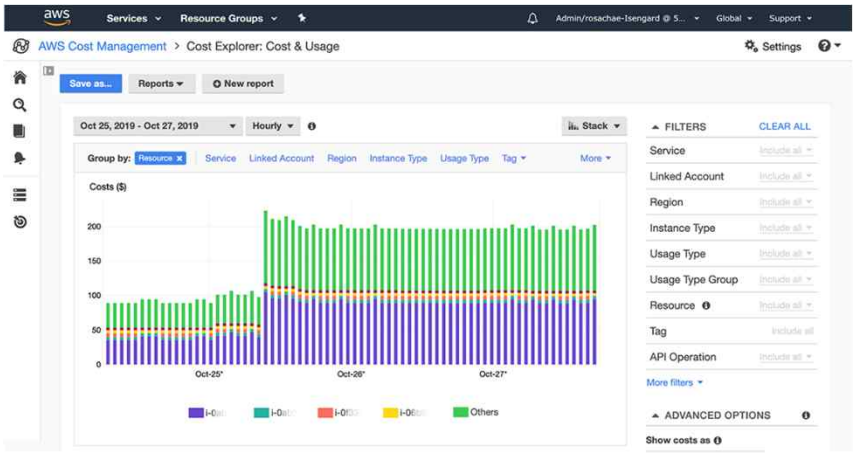
- 한 회사에서 여러 AWS 계정을 소유하는 경우 AWS Organizations를 사용하여 관리
- AWS Organizations 계정들의 통합 결제 가능
- 결제 시스템의 일원화가 가능해지고
- AWS 서비스를 대량 사용하는 경우 계정들의 사용량을 통합하여 할인 혜택 가능

예산(budget)

- 비용 또는 사용량이 예산 금액을 초과하거나
초과할 것으로 예상되면 알림 → 임계치 사전 설정
- AWS 예산에서는 정보가 하루에 세 번 업데이트



AWS Cost Explorer



- **AWS Cost Explorer**
  - 시간에 따른 AWS 비용과 사용량을 시각화, 이해 및 관리할 수 있는 손쉬운 인터페이스를 제공
  - 비용 및 사용량 데이터를 분석하는 사용자 지정 보고서를 작성

요금 (Billing) _ Lambda

Amazon Lambda

- AWS Lambda를 사용하면 사용한 만큼만 비용을 지불
- 함수 **요청** 수와 **기간**, 코드를 실행하는 데 걸리는 시간에 따라 요금이 청구

	요금
요청	요청 1백만 건당 0.20 USD
시간	GB-초당 0.0000166667 USD

기간에 대한 요금은 함수에 할당한 메모리 양에 따라 결정됩니다. 128MB와 10,240MB 사이에서 1MB 증분 단위로 함수에 필요한 양의 메모리를 할당할 수 있습니다. 아래 표에는 다양한 메모리 크기와 관련된 1밀리초당 요금에 대한 몇 가지 예가 포함되어 있습니다.

메모리(MB)(대형 메모리 포함)	1밀리초당 요금
128	0.0000000021 USD
512	0.0000000083 USD
1,024	0.0000000167 USD

함수에 512MB 메모리를 할당하고 한 달에 3백만 회 실행하며
매번 1초간 실행되었다면 요금은 다음과 같이 계산됩니다.

월별 컴퓨팅 요금
월별 컴퓨팅 요금은 GB-초당 0.00001667 USD이고
프리 티어에서 400,000GB-초 제공
총 컴퓨팅(초) = 3M * (1초) = 3,000,000초
총 컴퓨팅(GB-초) = 3,000,000 * 512MB/1024 = 1,500,000GB-초
총 컴퓨팅 – 프리 티어 컴퓨팅 = 월별 청구 대상 컴퓨팅 GB-초
1,500,000GB-초 – 400,000 프리 티어 GB-초 = 1,100,000GB-초

월별 컴퓨팅 요금 = 1,100,000 * 0.00001667 USD = 18.34 USD

월별 요청 요금
월별 요청 요금은 1백만 회 요청당 0.20 USD이고
프리 티어에서 월별 1백만 회 요청을 제공합니다.

총 요청 수 – 프리 티어 요청 수 = 월별 청구 대상 요청 수
3백만 회 요청 – 1백만 회 프리 티어 요청 = 2백만 회 월별 청구 대상 요청
월별 요청 요금 = 2백만 회* 0.2 USD/월 = 0.40 USD

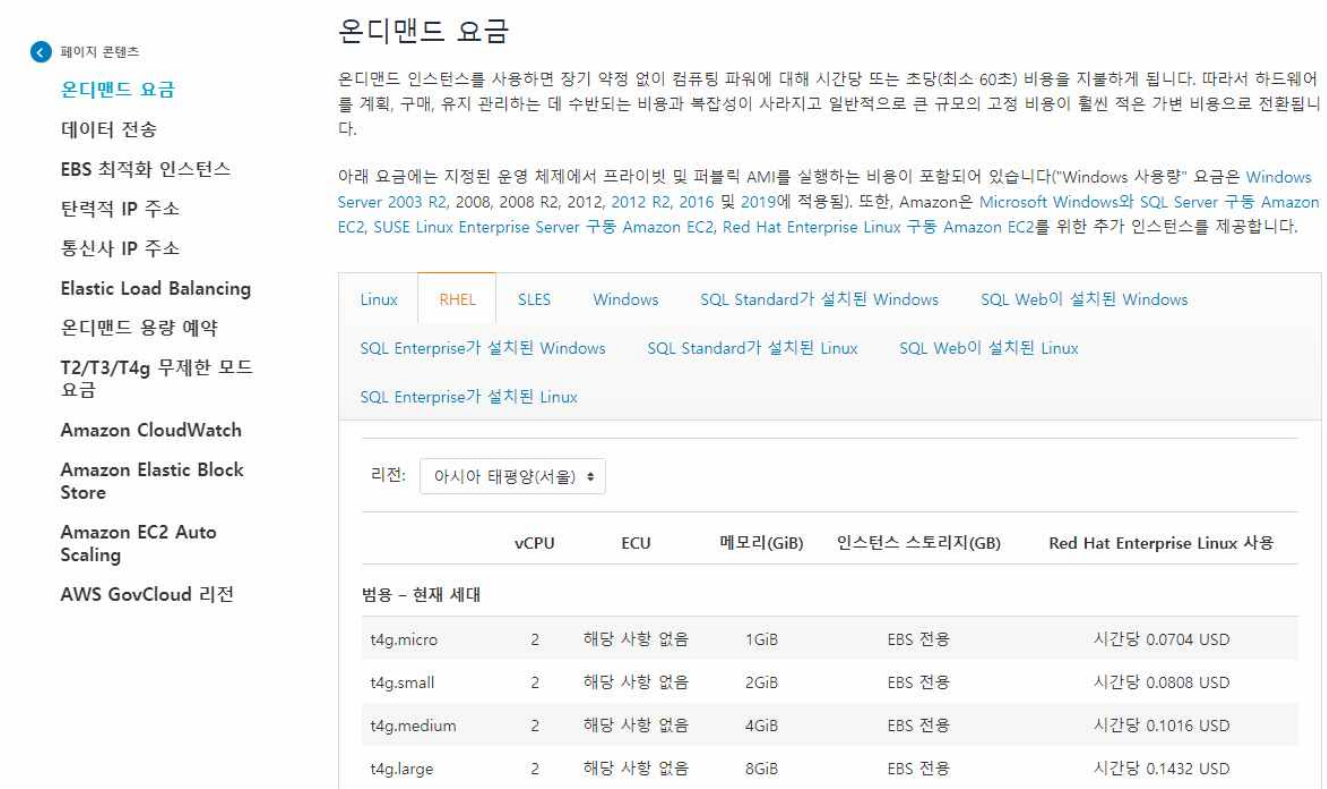
총 월별 요금

총 요금 = 컴퓨팅 요금 + 요청 요금 = 18.34 USD + 0.40 USD = 월별 18.74 USD

요금 (Billing) _ EC2

Amazon EC2

- 인스턴스 타입과 유형에 따라 비용이 결정
- https://aws.amazon.com/ko/ec2/pricing/on-demand/



AWS Support 플랜

AWS Support 플랜

- 소규모 스타트업에서 대기업까지 민간 부문 또는 공공 부문 관계없이 모든 비즈니스가 특정 요구에 맞게 설계된 지원 옵션
- Support plan 비교 : https://aws.amazon.com/ko/premiumsupport/plans

Support 플랜	설 명

Basic Support	• 모든 AWS 고객에게 무료 제공
	• 고객 서비스 및 커뮤니티
	  • 고객 서비스, 설명서, 백서 및 지원 포럼에 대한 연중무휴 24시간 상시 액세스를 제공AWS에 결제 관련 질문 및 서비스 한도 증가에 대해 문의
	• 제한된 AWS Trusted Advisor 검사 접근
	  • 7개의 핵심 Trusted Advisor 점검 사항 및 지침에 액세스하여 모범 사례에 따라 리소스를 프로비저닝
	• AWS Personal Health Dashboard
	  • AWS 서비스 상태에 대한 맞춤형 보기 및 리소스가 영향을 받을 때 알림

Developer Support	• AWS에서 테스트 또는 조기 개발 중인 경우 • 업무 시간 동안 기술 지원(email)을 받을 수 있는 기능과 구축 및 테스트 시 일반적인 아키텍처 지침 제공
Business Support	• AWS에서 프로덕션 워크로드를 실행 중 인 경우 • 엔지니어 기술 지원 상시 액세스, Health API 액세스 및 사용 사례에 대한 상황별 아키텍처 지침 제공
Enterprise Support	• Basic, Developer 및 Business Support 플랜에 포함된 모든 기능 • 우수한 엔지니어의 상시 기술 지원, 환경 상태를 자동으로 관리할 수 있는 도구 및 기술, 애플리케이션 및 컨설팅 형식의 아키텍처 지침 및 사전 예방적 프로그램에 액세스하여 조율할 수 있는 지정 TAM(기술 지원 관리자)와 AWS SME(주제 전문가) 지원 • 15분 내 응답, 전화, 채팅 또는 이메일로 연중무휴 24시간

기술 지원 관리자 (Technical Account Manager , TAM)

AWS 기술 지원 관리자 (TAM)

- Enterprise Support 플랜이 있는 경우 TAM을 통해서 AWS 지원 제공
- 애플리케이션을 계획, 배포, 최적화할 때 TAM이 지속적으로 커뮤니케이션하면서 권장 사항, 아키텍처 검토를 제공
- 모든 AWS 서비스에 대한 전문 지식을 제공하고 통합 접근 방식을 통해 여러 서비스를 함께 효율적으로 사용하는 솔루션을 설계하는 과정 지원

16. 마이그레이션 (Migration)

AWS 클라우드 적용 프레임 워크(AWS Cloud Adoption Framework, CAF)

AWS Cloud Adoption Framework(CAF)란?

- 마이그레이션에 관여해야 하는 다양한 역할에 초점을 맞춰 가이드로 6영역으로 구성

영 역	설 명
비즈니스 관점	• IT가 비즈니스 요구 사항을 반영하고 IT 투자가 주요 비즈니스 결과와 연계되도록 보장 • 비즈니스 전략 및 목표가 IT 전략 및 목표에 부합하는지 확인 • Common Roles : 비즈니스 관리자, 재무 관리자, 예산 소유자, 전략 이해당사자
인력 관점	• 클라우드 채택을 성공하기 위한 조직 전반의 변화 관리 전략 개발을 지원 • 조직 구조 및 역할, 새로운 기술 및 프로세스 요구 사항을 평가하고 격차를 파악 • 교육, 인력 배치 및 조직 변화의 우선 순위를 지정 • Common Roles : 인사 관리, 인력 배치, 인력 관리자
거버넌스 관점	• IT 전략이 비즈니스 전략에 부합하도록 조정하는 기술 및 프로세스에 중점 • 비즈니스 가치를 극대화하고 위험을 최소화 • 필요한 직원 기술 및 프로세스를 업데이트하는 방법을 이해 • Common Roles : 최고 정보 책임자(CIO), 프로그램 관리자, 엔터프라이즈 아키텍트, 비즈니스 분석가, 포트폴리오 관리자
플랫폼 관점	• 클라우드를 기반으로 새로운 솔루션을 구현하고 온프레미스 워크로드를 클라우드로 마이그레이션하기 위한 원칙과 패턴 • 다양한 아키텍처 모델을 사용하여 IT 시스템의 구조와 그 관계를 이해하고 전달 • Common Roles : 최고 기술 책임자(CTO), IT 관리자, 솔루션스 아키텍트
보안 관점	• 조직이 가시성, 감사 가능성, 제어 및 민첩성에 대한 보안 목표를 충족하도록 보장 • 조직의 요구 사항에 맞춰 보안 제어의 선택 및 구현을 구성 • Common Roles : 최고 정보 보안 책임자(CISO), IT 보안 관리자, IT 보안 분석가
운영 관점	• 비즈니스 이해당사자와 합의된 수준까지 IT 워크로드를 구현, 실행, 사용, 운영 및 복구 • 비즈니스를 구현 방법을 의하고, 비즈니스 운영을 반영하고 지원 • Common Roles : IT 운영 관리자, IT 지원 관리자

AWS 응용프로그램 마이그레이션 전략(Migration Strategy)

AWS 응용프로그램 마이그레이션 전략 (Application Migration Strategy)

- https://aws.amazon.com/ko/blogs/enterprise-strategy/6-strategies-for-migrating-applications-to-the-cloud/

전 략	설 명
리호스팅 (Rehosting)	• '리프트 앤 시프트(lift-and-shift)'라고도 하는 리호스팅에서는 애플리케이션을 변경 없이 이전 비즈니스 전략 및 목표가 IT 전략 및 목표에 부합하는지 확인 • 대규모 레거시 마이그레이션의 시나리오에서는 대부분의 애플리케이션이 리호스팅
리플랫포밍 (Replatforming)	• 리프트 앤 시프트 및 수정(lift, tinker, and shift)이라고도 하는 리플랫포밍에서는 실질적인 이점을 실현하기 위해 몇 가지 클라우드 최적화를 수행 • 최적화는 애플리케이션의 핵심 아키텍처를 변경하지 않고 달성
리팩터링 (Refactoring)	• 클라우드 네이티브 기능을 사용하여 애플리케이션을 설계하고 개발하는 방식을 재구성 • 비즈니스 요구 사항으로 인해, 다른 방법으로는 기존 환경의 애플리케이션에서 실현하기가 까다로운 기능 추가, 확장 또는 성능 개선의 필요성이 클 때 활용
재구매 (Repurchasing)	• 기존 라이선스를 Software-as-a-Service 모델로 전환합니다. • 예를 들어 기업은 고객 관계 관리(CRM) 시스템에서 Salesforce.com으로 마이그레이션하여 재구매 전략을 구현
유지 (Retaining)	• 비즈니스에 중요한 애플리케이션을 소스 환경에 유지
폐기 (Retiring)	• 폐기는 더 이상 필요하지 않은 애플리케이션을 제거하는 프로세스

감사합니다!