

AWS 개념(7/23 ~)

<클라우드 컴퓨팅 서비스>

- IT의 하드웨어적/소프트웨어적 리소스를 필요시(on demand) 인터넷을 통해 언제나 사용하는 서비스
- 사용자의 필요만큼만 사용하는 장점(용량 추정 불필요)
- 자본비용을 가변비용으로 대체
- 가격을 낮추는 전략(박리다매) > 규모의 경제로 얻게 되는 이점
- 콘솔을 통한 원하는 지역/용량으로 빠르게 배포 > 속도 및 민첩성 개선

<AWS Well-Architected 프레임워크> WAF

- 하나의 아키텍처(인프라)를 설정하는데 고려해야하는 대상



- AAA (Authentication/Authorization/Accounting) 보안 프레임워크 > IAM
- 장애복구 같은 안정성 > 스냅샷(백업), Auto Scaling ...
- 모든 리소스들의 비용 최적화
- 장애, 업데이트, 배포와 같은 기능들을 모니터링하며 효율적인 운영
- 복잡하지 않은 콘솔창을 이용해 인프라의 구성이 신속, 여러 서비스로 효율적인 성능

<클라우드 서비스 인프라 환경>

- 서비스의 배포범위(내구성)
- 리전은 AZ들의 모임, AZ는 하나의 데이터센터와 같은 의미
- 교차리전복제/ 중복 가용영역 서비스
- 물리적인 의미로 데이터센터
ex. S3(스토리지 서비스)에서의 리전영역
EC2는 AZ영역

1. 가용영역(AZ)

- 하나 이상의 데이터 센터로 구성
- 물리적인 데이터센터의 독립(격리)되어 있어 내결함성을 갖도록 설계
- 프라이빗 링크를 통해 다른 가용영역과 상호 연결

2. 리전

- 다른 나라(위치)들간 동일 서버를 사용하게 되면 재해 또는 서버장애 발생으로 인한 문제 때문에 생겨난 분리된 각 물리적 위치
- 두개이상의 가용영역으로 이루어짐
- 고객에게 가장 가까운 리전을 선택하여 지연시간을 최소화

3. 엣지 로케이션

- CDN 서비스인 CloudFront / Route 53 / WAF 위한 캐시서버
- 각 콘텐츠를 신속하게 다운로드 받기 위해 멀리 떨어진 서버를 캐시 서버로 가까운 위치에 구축해놓은 개념
- AWS 인프라(백본)와 사용자들의 경계점

*** CDN

> 데이터를 처리, 저장 및 공유하기 위해 함께 작용하는 많은 네트워크 컴퓨터의 모임

> 콘텐츠를 서버와 물리적으로 사용자들이 빠르게 받을 수 있도록 전세계 곳곳에 위치한 캐시 서버에 복제해주는 서비스

*** 캐시서버

> 인터넷 서비스 속도를 높이기 위해 사용자와 가까운 곳에 데이터를 임시 저장하여 빠르게 제공해주는 저장소 서버

Amazon Web Services의 범위



리전



- ✓ Amazon S3 버킷
- ✓ Amazon 머신 이미지(AMI)
- ✓ Amazon CloudWatch 지표
- ✓ Amazon EBS 스냅샷
- ✓ Amazon ElastiCache 클러스터
- ✓ Virtual Private Cloud(VPC)

전 세계



- ✓ AWS IAM 사용자, 그룹, 역할
- ✓ Amazon Route 53 호스팅 영역 및 레코드 세트
- ✓ Amazon CloudFront 배포

가용 영역



- ✓ Amazon EC2 인스턴스
- ✓ Amazon EBS 볼륨
- ✓ Amazon RDS 데이터베이스 인스턴스
- ✓ 서브넷

© 2019 Amazon Web Services, Inc. or its affiliates. All rights reserved.

<전반적인 AWS 구성>

리소스





웹 서버

사용자로부터 들어오는 작업 요청을 수락



데이터베이스

사용자 정보 및 작업 요청을 저장



Auto Scaling

필요에 따라 증가/감소



로드 밸런싱

가용 웹 서버로 트래픽을 분산

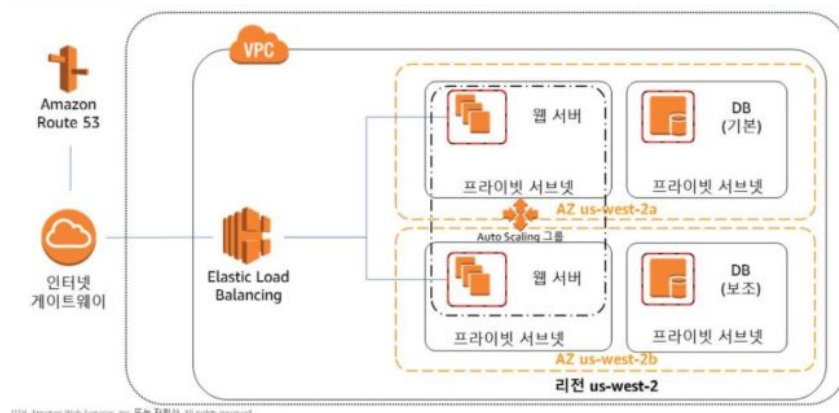


모니터링 및 로그

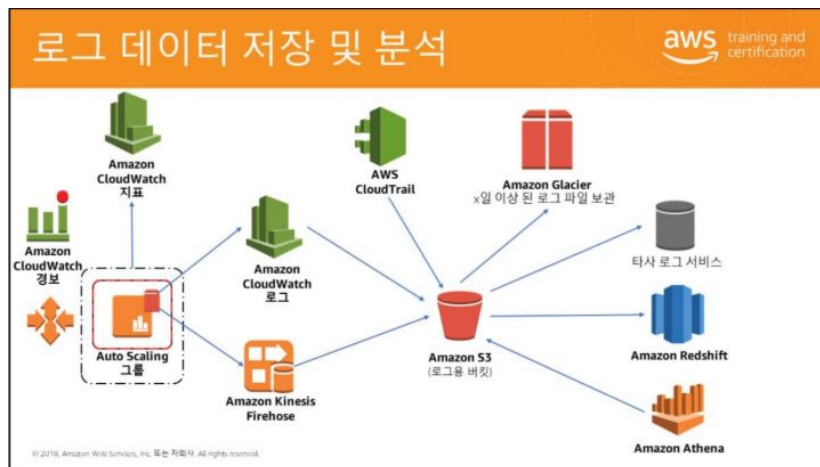
웹 서버 액세스 로그, 인스턴스 모니터링

- 사용자로부터 들어오는 작업요청을 수락하는 웹서버환경
- 사용자 정보 및 작업 요청이 되는 DB
- 시간대별로 인스턴스 확장/축소를 자동으로 하여 과금 발생을 줄이는 Auto scaling
- 여러 인스턴스로 트래픽을 분산하여 부하를 줄이는 로드 밸런싱
- 관리자가 웹서버 액세스 로그를 모니터링하는 시스템

기본적인 컴퓨팅 구성



저장소 구성



<S3> Simple Storage Service

- Region 단위의 리소스
- 높은 내구성을 실현한 스토리지
- 데이터를 저장 및 검색하도록 구축된 객체(object) 스토리지
- Web을 통해서 접근 가능하기 때문에 어디서든지 액세스가 가능
- '버킷'이라는 컨테이너를 만들고 '객체'라는 파일을 저장
- 버킷이 생성되면 하나의 DNS가 객체를 만들면 URL이 생김 > 이름이 고유해야함
- 객체수의 제한이 없음, 객체당 최대 5TB의 제한
- 스토리지 사용량만큼 과금
- 객체의 기본적인 권한은 버킷을 만든사람으로 제한
- File get 시에도 과금이 발생
- 버킷 갯수는 계정당 100개, 추가시 AWS에 요청
- 보안을 강화하기 위한 버킷에 MFA 설정

160GB를 초과하는 파일을 업로드하려면 AWS CLI, AWS SDK 또는 Amazon S3 REST API를 사용합니다.



[https://\[bucket name\].s3.amazonaws.com](https://[bucket name].s3.amazonaws.com)



[https://\[bucket name\].s3.amazonaws.com/Video.mp4](https://[bucket name].s3.amazonaws.com/Video.mp4)



*** S3 권한

- > IAM이 없을때 ACL로 권한 제한
- > IAM이 생기고 버킷정책이라는 모듈로 권한 제한
- > 버킷에 외부에서 접근하려면 퍼블릭액세스 차단 설정 no check
- > 객체에 외부에서 접근하려면 '작업 - 퍼블릭으로 설정'



*** S3의 퍼블릭 액세스(정적 웹페이지 호스팅)

- > 버킷 자체를 정적 웹사이트로 사용하는 기능
- > 버킷 속성에서 설정 가능 (기본적인 사용안함의 설정)

엔드포인트: <http://dongbaris3.s3-website-ap-northeast-2.amazonaws.com>

- ☐ 이 버킷을 사용하여 웹 사이트를 호스팅합니다. [세부 정보](#)
- ☐ 요청 리디렉션 [세부 정보](#)
- ☒ 웹 사이트 호스팅 사용 안 함

전체 정적 웹 사이트 호스팅



*** 객체 잠금 기능

- > 데이터 보존 또는 보호를 위해 S3의 객체를 잠금
- > WORM(Write Once Read Many) 기능을 사용하여 스토리지 내에서 실수로 덮어쓰거나 삭제하는 일을 방지
- > 법적보존 / 보존기간설정

*** 이벤트 알림 기능

- > 특정 버킷으로 객체가 업로드되거나 삭제되는 등 특정 이벤트가 발생할 때 자동 알림을 보내도록 설정
- > 알림은 사용자에게 전달되거나, Lambda 스크립트와 같은 다른 프로세스를 트리거하는데 사용될 수 도 있음

*** 버전관리

- > 기존의 객체와 같은 이름의 파일이 업로드되면 새로운 파일만 객체로 저장됨
- > 기존의 객체도 삭제되지 않고 그대로 남아있는 기능이 버전관리
- > 속성 - 버전관리 활성화
- > 기존의 화면에서 버전 '숨기기/표시' 기능이 생김
- > 버전 표시 상태로 해놓으면 이전의 객체와 새로운 객체가 표시됨
- > 객체를 삭제해도 버전관리 기능에 의해 복구가 가능함
- > 삭제마커를 삭제하면 기존의 객체가 복구됨

버전 관리

동일 버킷 내에 한 객체의 여러 버전을 보관합니다.

세부 정보

aws s3api

create-multipart-upload
put-object-acl
put-bucket-policy
list-object-versions

활성

버전

숨기기

표시

test.txt

<input type="checkbox"/>		2020. 8. 5. 오후 8:20:2...	g5cTPVdfS56pKk0E...
<input type="checkbox"/>		2020. 8. 5. 오후 8:17:56	null

*** CORS S3 액세스 제어

> 한 도메인에서 로드되어 있는 클라이언트 웹 API가 다른 도메인에 있는 리소스와 상호 작용하는 방법을 정의

*** 객체 스토리지 클래스

> S3 Standard

: 멀티 가용영역 복제, 액세스가 자주 일어나는 파일의 클래스

> S3 standard IA

: 멀티 가용영역 복제, Standard보다 비용이 저렴하지만 액세스가 적은 파일의 클래스

> S3 One_Zone IA

: 다른 가용영역으로의 복제를 하지 않는 파일의 클래스



*** 멀티파트 업로드

> 대용량의 객체를 관리 가능한 파트로 나눠 일관된 업로드하는 기능

> 업로드가 끝나면 개별 조각으로부터 전체 객체를 다시 생성함



*** 교차리전복제

> 다른 리전의 S3에 자동으로 복제 할 수 있는 기능

> 높은 수준의 내구성을 구성할때 사용

*** S3 Transfer Acceleration

- > 특정 리전에 있는 사용자가 다른 리전으로 업로드를 한다면 CloudFront의 엣지 로케이션을 통해서만 S3버킷으로 빠르고 간편한 데이터 전송을 할 수 있는 기능
- > 사용자가 대용량의 파일을 직접 인터넷을 통한 업로드를 하는 것보다 엣지 로케이션을 사용하여 보안적으로 안전하고 신속한 기능

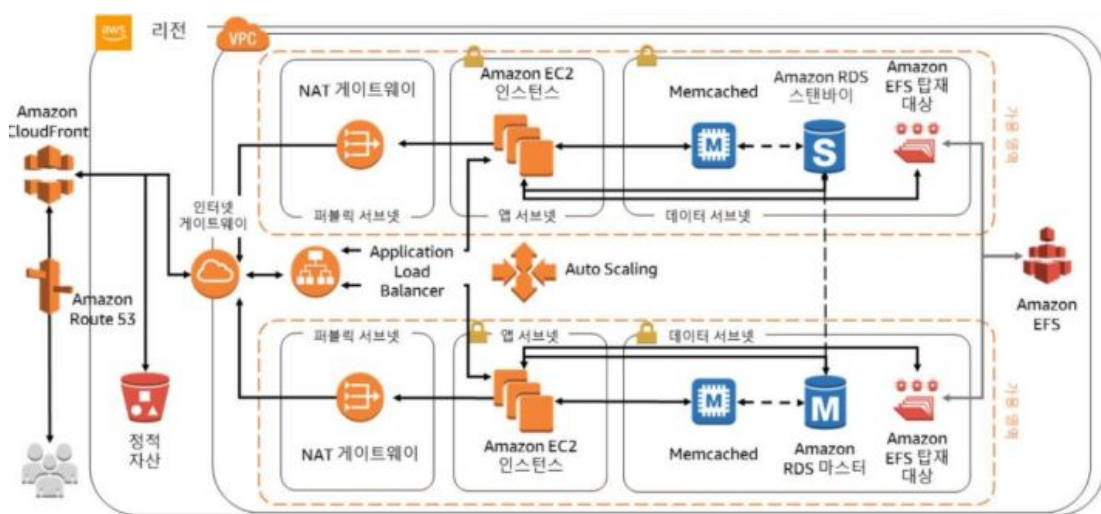


<Glacier>

- 액세스 빈도가 낮고, 장기 백업을 위한 안전한 스토리지 서비스
- 스토리지 비용이 S3보다 저렴
- 아카이브(객체) / 볼트(아카이브의 모음)
- 요청에 의해서만 복원가능(복원시간이 다소 걸림)
- 액세스가 없는 아카이브를 저장소 잠금을 통해 누구도 접근하지 못하게 만듦
- S3 클래스 선택에서 Gracier 선택 가능
- S3에 객체로 저장되지만 다운로드가 되지 않음 (복원 기능 활성화)
- 신속검색 / 표준검색 / 대량검색의 복원 옵션이 있어 비용차이가 발생

*** 수명주기관리 (Life Cycle)

- > 저장된 데이터의 수명주기를 자동화하여 액세스가 감소하면 객체 클래스를 변경하여 비용을 절감하는 기능
- > 수명주기규칙추가를 하여 사용자가 설정 가능



모듈3. 컴퓨팅 계층 추가 (애플리케이션 구동)



<EC2>

- 웹호스팅, DB서버, 인증서비스, 서버의 모든 역할
- 온프레미스 서버보다 자유로운 활동
- 가상 컴퓨팅 환경(가상화 서버 시스템 박스)
- AMI 선택 (Linux, Windows) > OS + Source program
- AWS 관리형 AMI, 사용자 생성 AMI
- 인스턴스를 생성하는 웹서비스
- 로그인을 위한 키페어(퍼블릭 키/프라이빗 키) 생성
 - > 퍼블릭 키 암호화 기법이 퍼블릭 키를 사용하여 암호등의 데이터를 암호화
 - > 사용자는 프라이빗 키를 사용하여 해당 데이터를 해독
- 시작 템플릿 설정
 - > 버전관리 가능

네트워크 ⓘ ↕ 🔄 새 VPC 생성

서브넷 ⓘ ↕ 새 서브넷 생성

퍼블릭 IP 자동 할당 ⓘ ↕

*** AMI

- > 루트볼륨안에 소프트웨어의 기능들을 추가한 일종의 템플릿
- > 반복성, 재사용성, 복구성, 비용만 지불하면 다양한 벤더사의 os 사용(Market Place) 백업
- > 클라우드의 가상서버인 인스턴스를 시작하는데 필요한 정보를 제공
- > 인스턴스 루트 볼륨 템플릿(OS, 애플리케이션 서버, 애플리케이션)

빠른 시작

- 나의 AMI
- AWS Marketplace
- 커뮤니티 AMI



사전 구축



AWS Marketplace



자체 생성

*** 사용자데이터

- > 리눅스는 bashshell, 윈도우는 powershell
- > 인스턴스가 시작되는 최초 1회 자동으로 실행하려는 사용자 정의 데이터
- > 명령어를 통해 메타데이터를 OS 안에서도 확인이 가능하게끔 할 수 있음
- > os의 업데이트, 필수 어플리케이션의 다운 등을 스크립트 언어로 미리 만들어 놓는 시스템

```
#!/bin/bash
yum update -y
hostname = $(curl -s
http://169.254.169.254/latest/meta-data/public-
hostname)
```

사용자 데이터 ⓘ ☒ 텍스트로 ☐ 파일로 ☐ 입력이 이미 base64로 인코딩됨

(선택 사항)

*** 메타데이터

> 인스턴스의 정보들을 알려주는 데이터

인스턴스 ID	i-0d2c940e7ca106d7f	퍼블릭 DNS(IPv4)	ec2-13-125-68-82.ap-northeast-2.compute.amazonaws.com
인스턴스 상태	running	IPv4 퍼블릭 IP	13.125.68.82
인스턴스 유형	t2.micro	IPv6 IP	-
결과	권장 사항을 위해 AWS Compute Optimizer에 옵트 인합니다. 자세히 알아보기	탄력적 IP	
프라이빗 DNS	ip-172-31-13-79.ap-northeast-2.compute.internal	가용 영역	ap-northeast-2a
프라이빗 IP	172.31.13.79	보안 그룹	launch-wizard-1, 인바운드 규칙 보기, 아웃바운드 규칙 보기
보조 프라이빗 IP		예약된 이벤트	예약된 이벤트 없음
VPC ID	vpc-0e60c265	AMI ID	amzn2-ami-hvm-

*** 인스턴스 유형

ex) t 2 micro
패밀리 이름 세대 크기



*** 인스턴스 프로파일

- > EC2 인스턴스가 시작될 때 역할(role) 정보를 EC2 인스턴스에 전달하는 데 사용하는 IAM 역할용 컨테이너
- > 액세스 키를 인스턴스에 자동 전파

<EBS>

- 인스턴스와 네트워크로 연결되어 있는 스토리지(SAN)
- 블록 단위의 스토리지 (S3와의 차이점)
- 네트워크로 연결되어 있어 인스턴스가 중지되어도 데이터는 남아있음
- 인스턴스에 종속되어 있는게 아니라서 다른 인스턴스와의 연결도 가능
- 인스턴스 생성시 저장소 목록 존재, 종료시삭제 옵션
- EBS 볼륨을 따로 저장하려면 스냅샷으로 S3로 저장하는게 저렴
- EBS는 2개이상의 인스턴스에 연결하지 못함

- SSD / HDD / Magnetic 유형 제공
- 루트볼륨은 HDD 유형을 사용할 수 없음

볼륨 유형 ⓘ	디바이스 ⓘ	스냅샷 ⓘ	크기(GiB) ⓘ	볼륨 유형 ⓘ	IOPS ⓘ	처리량(MB/초) ⓘ	종료 시 삭제 ⓘ	암호화 ⓘ
루트	/dev/xvda	snap-008d2b3aaaf1dffa	8	범용 SSD(gp2)	100/3000	해당 사항 없음	<input checked="" type="checkbox"/>	암호화5 ▼

*** EBS 최적화 인스턴스

- > 네트워크로 연결된 환경 때문에 트래픽의 속도를 보장받지 못함
- > 인스턴스 생성시, 인스턴스 유형 선택에서 확인 가능 ('예')

** 인스턴스 스토어(스토리지)

- > 인스턴스 안에서 구동되는 스토리지(저장소)
- > 인스턴스가 중지되면 데이터가 삭제되는 휘발성

*** 공유파일 시스템

- > 여러 인스턴스가 하나의 저장소에 연결하지 못하는점에 대한 EBS의 한계점
- > 이런 한계점을 공유파일시스템인 EFS/FSx를 사용
- > 초반 EFS라는 리눅스에서만 사용가능한 시스템이 활발했는데, FSx라는 윈도우 전용 파일시스템이 생겨남



Amazon EBS는 하나의 인스턴스에만 연결됩니다.



Amazon S3가 하나의 옵션이지만 이상적인 것은 아닙니다.



이런 작업에는 Amazon EFS 및 Amazon FSx가 적합합니다.

*** AWS License Manager

- > 타사의 라이선스를 등록하는 서비스
- > 사용자는 인스턴스 가동시 라이선스 규칙을 사용하여 범위 이상의 라이선스를 사용해 위반하지 않도록 제어하거나 다른 서버에 단기적으로 라이선스를 재할당 할 수 있음

*** EC2 전용 인스턴스 / 전용 호스트

- > 전용인스턴스는 다른 AWS 계정과 물리적으로 격리하는 기능
- > 전용호스트는 사용자가 선택한 특정 하드웨어에서 실행되어 비용이 절감, 온디맨드(시간)로 구입 가능

*** EC2 테넌시

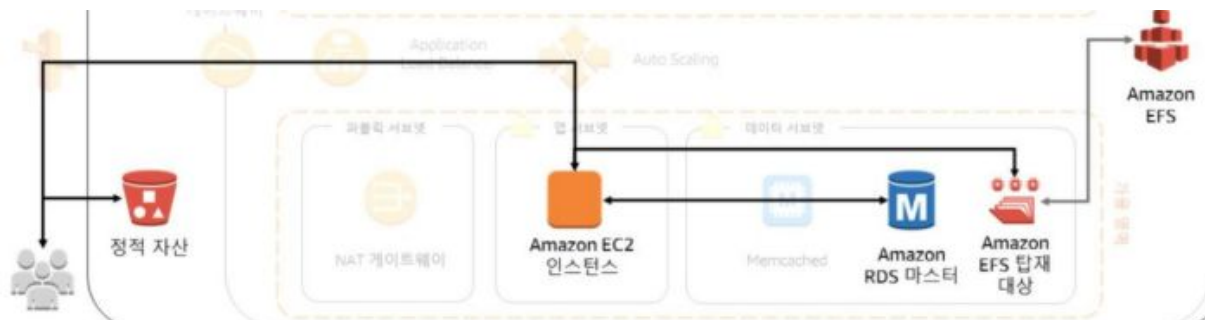
	하드웨어를 자신의 계정만 사용합니까?	설명
기본값	아니요	인스턴스가 공유된 하드웨어에서 실행됩니다.
전용 인스턴스	예	비 특정 하드웨어에서 실행됩니다.
전용 호스트	예	고객이 선택한 특정 하드웨어에서 실행되어 고객이 보다 세밀하게 제어할 수 있습니다.

테넌시 ⓘ 공유됨 - 공유된 하드웨어 인스턴스 실행
ference ⓘ 공유됨 - 공유된 하드웨어 인스턴스 실행
전용 - 전용 인스턴스 실행
전용 호스트 - 전용 호스트에서 이 인스턴스 실행
 추가 옵션이 있습니다

*** 배치그룹 전략 (아키텍처 고려사항)

배치 그룹 전략 ⓘ spread
cluster
spread
partition

모듈4. 데이터베이스 계층추가

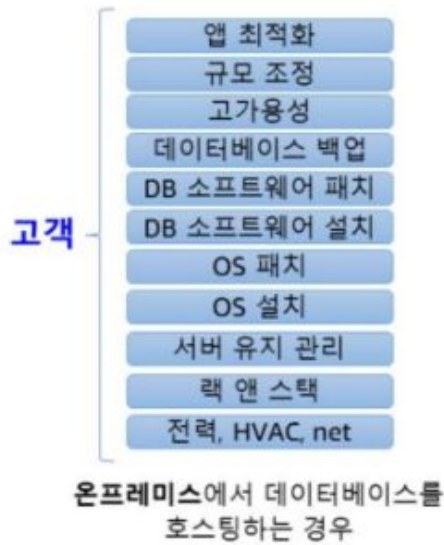


> 데이터베이스는 인스턴스에 직접 엔진 구성(비관리)과 관리형 데이터베이스를 사용하는 방법등 다양한 방법이 존재

> 고가용성(복제) 과 확장성을 고려해야함

*** 비관리형 DB

> 온프레미스에서 DB환경을 구축, EC2 인스턴스내에 DB환경 구축



*** 관리형 DB

> RDS 완전관리형 DB환경 구축



관계형 데이터베이스



Amazon RDS



Amazon Redshift



Amazon Aurora

비관계형 데이터베이스



Amazon DynamoDB



Amazon ElastiCache



Amazon Neptune

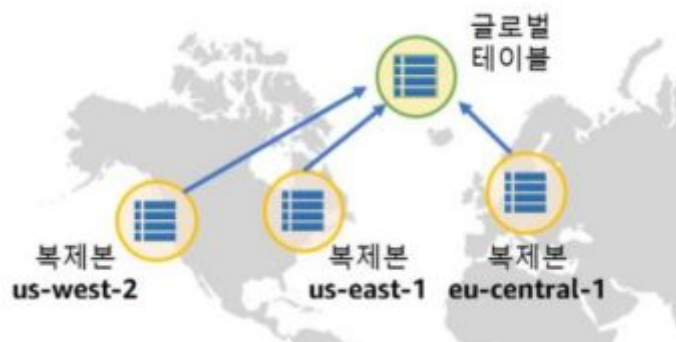
	관계형/SQL	NoSQL
데이터 스토리지	행 및 열	키 값, 문서 및 그래프
스키마	고정	동적
쿼리	SQL 기반 쿼리	문서 수집에 집중
확장성	수직적	수평적

<Amazon RDS>

- 완전관리형 관계형 DB
- 온프레미스 환경보다 신속한 프로비저닝
- 간단한 수직 확장 조정
- 대부분의 DB 리소스들을 지원
- 자동 백업 보존 기능으로 실수에 의한 손실을 복구
- 백업된 데이터는 수명주기정책을 통해 기간 정의

<Dynamo DB>

- 완전관리형 비관계형 DB
- 간단하고 비용 효율적인 방법으로 데이터를 저장
- 짧은 지연시간으로 신속한 처리가 필요한 애플리케이션에 적합
- SSD 스토리지 자동 3방향 복제기능으로 성능, 안정성 및 보안이 기본 제공됨
- 각 리전마다 복제본을 가지는 글로벌 복제본 테이블을 갖고 있어 다중 리전에 다중 마스터 DB를 만들 수 있는 종합 관리형 기능을 제공



*** RDS 보안제어

- > 마스터 사용자 계정을 통한 자격증명 (ID,PW)
- > 마스터 사용자가 계정을 만들어 DB 인스턴스에 접근 할 수 있는 권한을 제한
- > 데이터베이스 보안그룹의 기본값은 “모두 거부” 액세스 모드
- > 사용자가 데이터베이스 서버 포트에 대한 액세스만을 허용하도록 명시적 정의
- > 저장시 데이터를 암호화(기본 스토리지, 자동백업, 읽기전용복제본, 스냅샷)
- > 전송시 SSL을 사용한 암호화
- > 이벤트 알림

*** Dynamo DB 보안제어

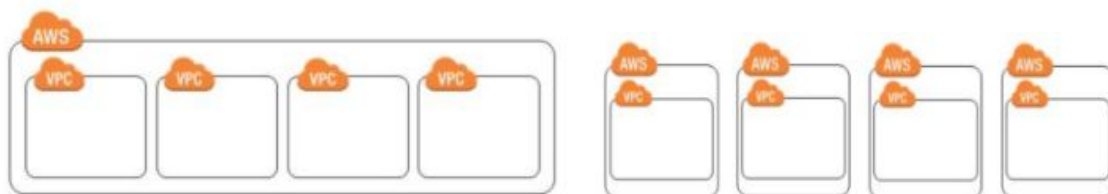
- > 데이터베이스의 항목,속성까지 모든것에 대한 액세스를 부여
- > 완전관리형 저장 암호화 기능을 제공
- > HTTPS 프로토콜을 사용한 통신

모듈5. AWS 기반 네트워킹



<VPC>

- 리전 안의 가용영역 모두 활용 가능
- 독립된 프라이빗 네트워크 공간
- 각 조직(워크로드)에 대한 논리적 격리를 제공
- VPC 작업에서 DNS 호스트네임 활성화를 시켜야 포함되는 인스턴스에 DNS가 부여
- Subnet에서 Public IP 할당
- 다중 VPC를 쓰는 것이 일반적인 구성
 - > 하나의 계정에 여러 VPC를 쓰는 경우
 - 관리자 계정이 모든 VPC를 관리하기 때문에 단일팀 또는 소규모의 회사에서 사용
 - > 각 부서마다 계정을 따로 격리하여 VPC를 쓰는 경우
 - 대규모의 회사나 큰 프로젝트의 경우에 사용
 - AWS Organization으로 통합해서 관리
- 단일 VPC를 쓰는 경우는 테스트, 학습, 소규모의 애플리케이션을 쓰는 경우
- 기본 계정당 리전마다 5개의 VPC를 생성 할 수 있음 (추가 시 요청)
- 생성시 , default subnet/RT/SG/NACL 자동으로 생성



*** 테넌시

> VPC를 논리적으로 격리하는 설정, 비용이 많이 듦

*** Subnet

- > 특정 가용영역 안에서 구성, 가용영역 2개 이상 걸쳐서 구성 할 수 없음
- > Private 서브넷은 Public Subnet의 인스턴스를 통해서 관리
- > Public 인스턴스안에 접속, 접속키를 복사해서 똑같은 이름의 파일을 생성 (비공개 권한 설정 chmod 400)



데이터 스토어 인스턴스



프라이빗 서브넷



배치 처리 인스턴스



프라이빗 서브넷



백엔드 인스턴스



프라이빗 서브넷



웹 애플리케이션 인스턴스



퍼블릭 또는 프라이빗 서브넷

*** 라우팅테이블

- > 서브넷에 하나의 라우터가 속해있는 개념으로 생각
- > VPC를 생성하면 하나의 기본 라우팅테이블이 자동으로 생성됨
- > VPC 기반의 서브넷은 기본적으로 해당 라우팅 테이블에 속해 있음(명시적 지정 필요)
- > 일반적으로 하나의 서브넷에는 하나의 라우팅테이블을 지정
- > 지정될수 있는 경로가 100개로 제한

*** 인터넷 게이트웨이

- > 외부에서 내부로 들어오는 통로(인바운드) 내부에서 외부로 나가는 통로(아웃바운드)
- > 인스턴스의 사설IP와 공인IP의 연결을 하는 라우팅 정보를 가지고 있음
- > 완전관리형 서비스로 복수의 게이트웨이가 자동으로 생성(사용자에게는 보이지 X)
- > 가용성,중복성,수평적으로 자동 확장

*** NAT 게이트웨이

- > Private Instance의 요청이 외부로 나갈 수 있게 해주는 IP변환 기능
- > Public Subnet에 생성
- > 완전관리형 서비스 , 과금의 이유로 NAT 인스턴스 사용
- > 가용성을 위해 각 AZ에 하나의 NAT 적용을 권장

NAT 게이트웨이 생성:

1. 다음을 지정:

- ✓ 게이트웨이가 속할 퍼블릭 서브넷
- ✓ 탄력적 IP 주소

2. 다음을 업데이트:

- ✓ 프라이빗 서브넷의 라우팅 테이블

*** NAT 인스턴스

- > NAT 기능을 구현한 인스턴스
- > 커뮤니티 AMI에서 NAT를 검색해서 기능을 구현한 AMI 선택
- > NAT 인스턴스의 작업에서 '소스/대상 확인' 비활성화
- > NAT 인스턴스는 EC2 AMI에 NAT 기능을 추가한 인스턴스 (비용절감)

*** ENI (가상 네트워크 인터페이스 / 탄력적 네트워크 인터페이스)

- > 하나의 인스턴스에 하나이상의 ENI에 연결할 수 있음
- > 동일한 가용영역 안에서 EC2 인스턴스간에 이동할 수 있음

네트워크 인터페이스 eth0

인터페이스 ID [eni-0401d26b31d8987bd](#)
VPC ID [vpc-00caa3a717d575508](#)

- 관리 네트워크 생성
- VPC에서 네트워크 및 보안 어플라이언스 사용
- 별도의 서브넷에 있는 워크로드/역할로 이중 홈 인스턴스 생성

*** EIP (탄력적 IP)

- > 인스턴스에 할당되는 기본적인 공인IP는 인스턴스 시작시 할당(유동적)
- > 공인 IP를 고정IP로 할당받는 기능
- > 인스턴스가 재시작되도 탄력적IP라는 고정된 IP가 항상 할당됨

Firewall (L3 - IP / L4 - Port) + L7 - DPI(Deep Packet Inspection)

1세대 방화벽 (Stateless)

: Packet Filter , 응답을 별도로 허용하지 않으면 단방향 통신

> Reflexible ACL , 인바운드의 룰을 뒤집어 아웃바운드에 적용

> Port를 두개 이상 쓰는 서비스(동적포트서비스)에서는 적용이 되지 않음 ex. FTP

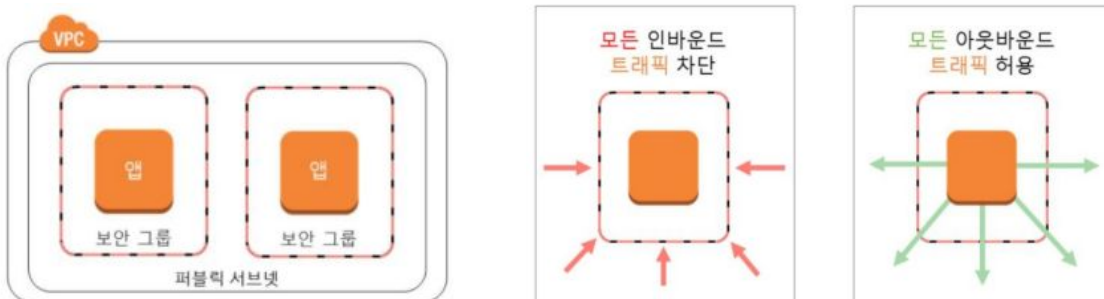
2세대 방화벽(Proxy 방화벽)

: 서버기반의 방화벽

3세대 방화벽(Stateful)

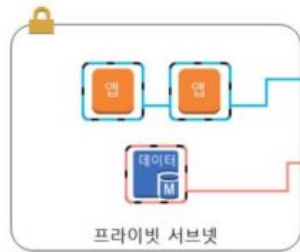
: 1세대 방화벽을 보완, 방화벽을 통과한 세션 정보를 저장하여 나갈때의 룰을 따로 설정하지 않아도 됨

*** 보안그룹 (Security Group)



- > AWS 리소스에 대한 인바운드/아웃바운드 트래픽을 제어하는 가상방화벽
- > TCP 프로토콜, IP주소로 허용
- > 상태저장 SPI(Stateful Packet Inspection) 규칙
- > 인바운드만 설정하면 아웃바운드는 상태저장 룰에 의해 설정하지 않아도 됨
- > 인바운드는 기본적으로 모두 차단되어 있고, 아웃바운드는 모두 허용되어 있음
- > 규칙은 트래픽의 허용 설정만 할 수 있음 (Deny x)

대부분 조직은 각 기능 티어에 대한 인바운드 규칙으로 보안 그룹을 생성합니다.



앱 티어
보안 그룹
웹 티어
보안 그룹

웹 티어
보안 그룹

애플리케이션
보안 그룹

데이터베이스
보안 그룹



인바운드 규칙
HTTPS 포트 443 허용
소스: 0.0.0.0/0(모두)

인바운드 규칙
HTTP 포트 80 허용
소스: 웹 티어

인바운드 규칙
TCP 포트 3306 허용
소스: 앱 티어

- > 각 보안그룹에는 여러개의 인스턴스를 포함 시킬 수 있다
- > 각 인스턴스의 IP는 유동적이고, 다른 보안그룹에 트래픽 허용을 정의 할 때 IP정보를 고정적으로 정의하지 못하여, 특정 보안그룹의 트래픽 허용을 지정한다(체인 다이어그램)
- > 보안그룹에서 보안그룹으로의 트래픽 규칙을 정의하는 단위를 '티어'라고 함

*** Network ACL

- > 서브넷 경계의 방화벽
- > 기본적으로 모든 인바운드 및 아웃바운드 트래픽을 허용함
- > 순차적인 적용에 의해 '모든 트래픽 Deny'는 무시됨
- > 상태 비저장(Stateless) 이므로 인바운드와 아웃바운드 모두 명시적인 규칙이 필요
- > 보안그룹과 달리 특정 프로토콜/IP 의 트래픽을 차단(Deny) 할 수 있음

규칙 #	유형	프로토콜	포트 범위	소스	허용/거부
100	모두 트래픽	모두	모두	0.0.0.0/0	ALLOW
*	모두 트래픽	모두	모두	0.0.0.0/0	DENY

Nacl-11223344

인바운드:
Rules # 100: SSH 172.31.1.2/32 ALLOW
Rules # *: ALL traffic 0.0.0.0/0 DENY

아웃바운드:
Rules # 100: Custom TCP 172.31.1.2/31 ALLOW
Rules # *: All traffic 0.0.0.0/0 DENY

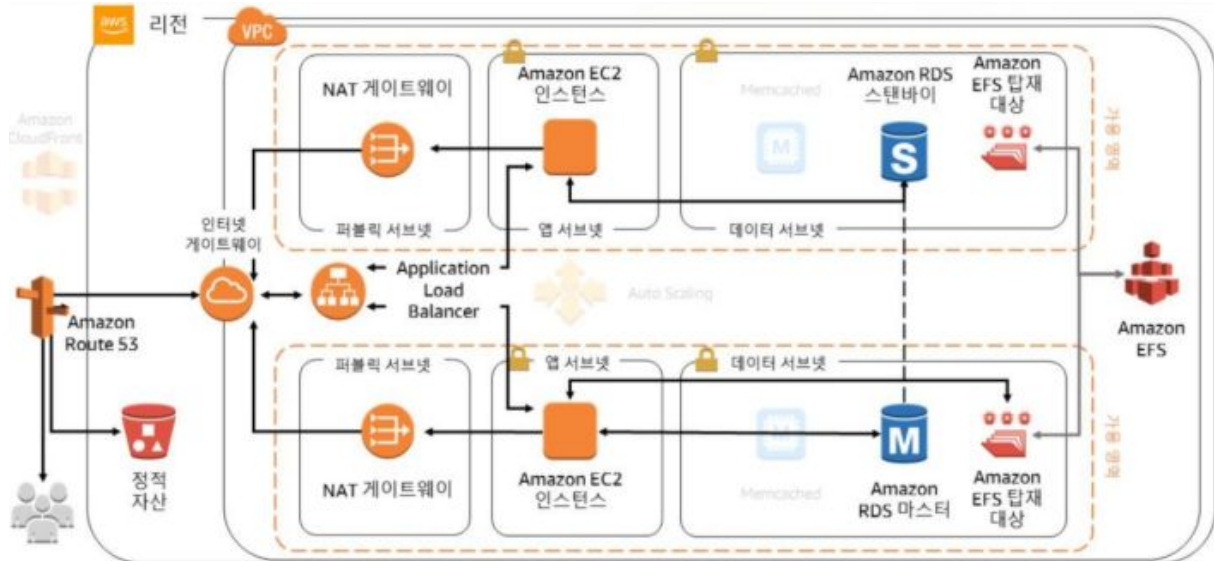
*** 보안성이 강화된 인프라 구조



*** 인터넷 액세스를 위한 구조



모듈6. AWS 기반 네트워킹2



*** VPN / Direct Connect / CloudHub / Transit GW

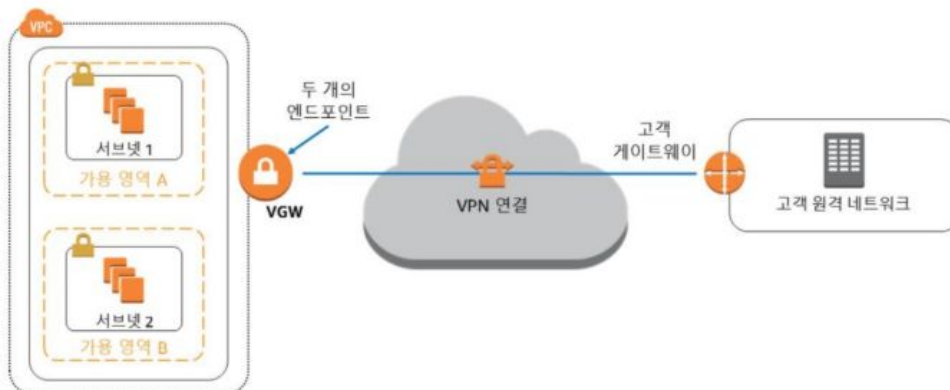
- > on-premise 환경에서 AWS환경으로 연결을 가능하게 하는 기술
- > 인터넷 GW를 통하지 않는 보안성을 강화한 연결 방법

VPN

- 1) site - to - site (IPsec)
: 고정적인 여러개의 프라이빗 네트워크를 연결해주는 구성(본사-지사)
- 2) Remote Access (SSL VPN)
: 원격환경에서 프라이빗 네트워크에 접속하는 구성(본사-출장지)

*** 가상 프라이빗 게이트웨이(VGW)

- > VPC와 다른 네트워크 사이에 프라이빗 연결(VPN)을 해주는 게이트웨이
- > 완전관리형 서비스, 장애조치 구현
- > 실제 인터넷 망을 사용하지만, 암호화를 통해 외부로의 노출을 피함
- > 보안은 강화되지만, 인터넷의 부하가 늘어나 안정성이 보장되지 않음

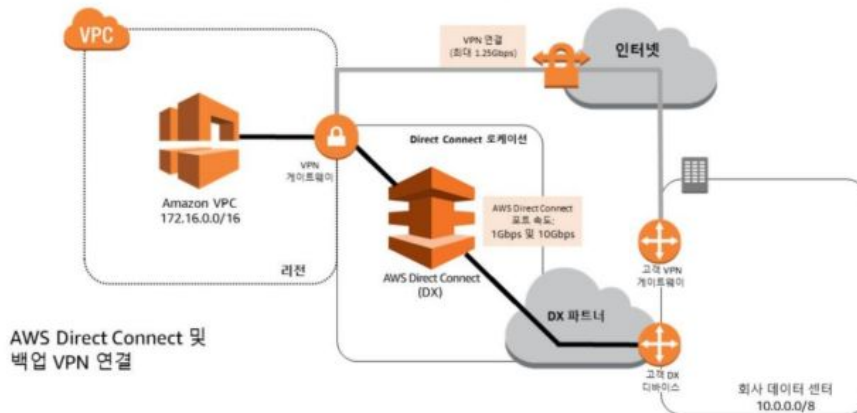


*** AWS Direct Connect

- > DR 환경을 클라우드 환경에서 구성할 때 사용
- > 인터넷망을 쓰는 VGW의 안정성(데이터 손실X)을 보장하는 기능
- > 전용선을 사용하여 직접적으로 두 네트워크를 연결(인터넷망 X)
- > 트래픽이 많거나, 인터넷망을 쓸 수 없는 통신에 사용
- > 전용선에 문제가 생기면 전체적인 통신 오류가 발생하는 단점

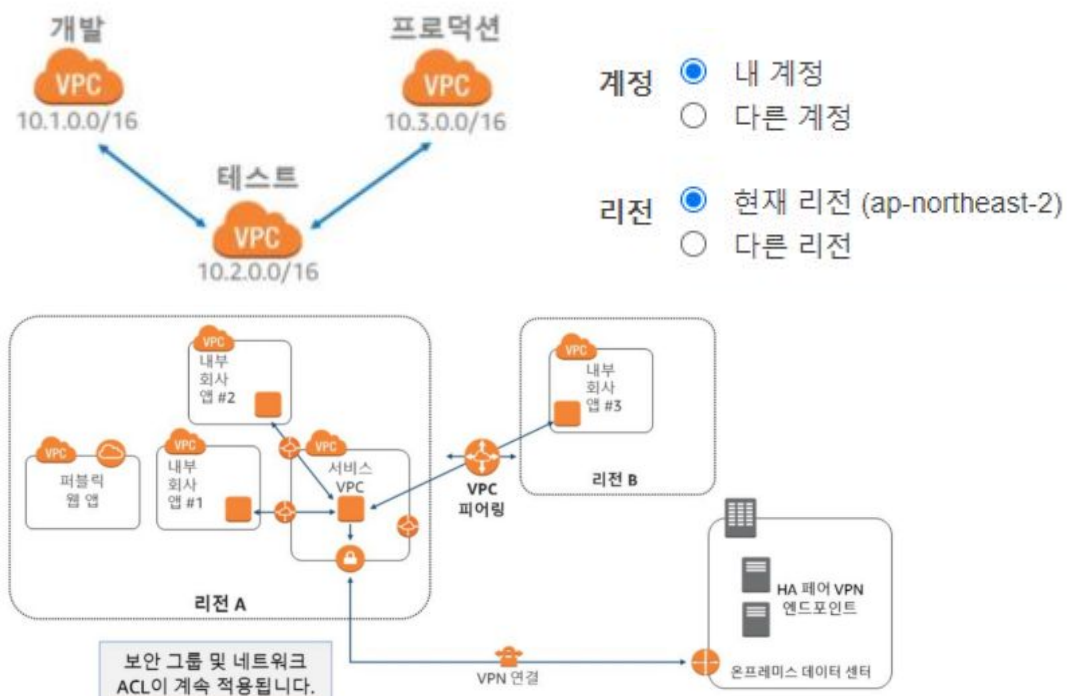
*** 크리티컬 워크로드

- > Direct connect와 VPN을 동시에 사용하는 방법
- > 두 연결방법의 단점을 서로 보완하는 방법



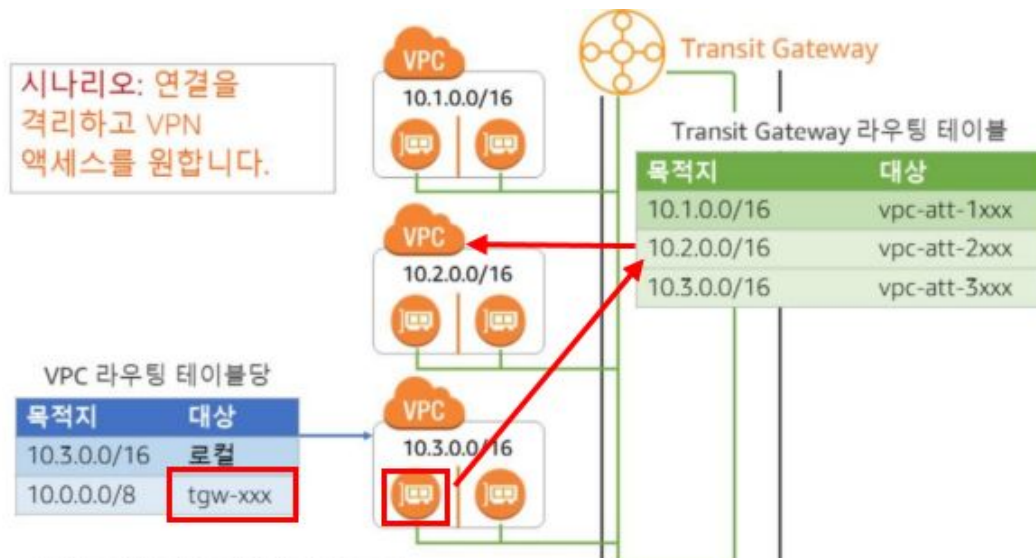
*** VPC 피어링

- > 기본적으로 VPC를 서로 격리되어 있는 인프라를 구성함
- > VPC간 연결은 인터넷을 통해 가능하지만, 보안성을 강화하기 위해 고안
- > 이때, VPC간의 통신을 위한 기능을 의미
- > VPC 사설 IP간의 통신이 가능하지만, VPC간 IP가 중복되면 안됨
- > 다른 Region간의 VPC, 다른계정간의 VPC 연결이 모두 가능함
- > 전이적 피어링(피어링의 전달) 관계는 불가능, 직접적인 피어링 연결만 가능
- > 완전관리형 서비스, 장애복구

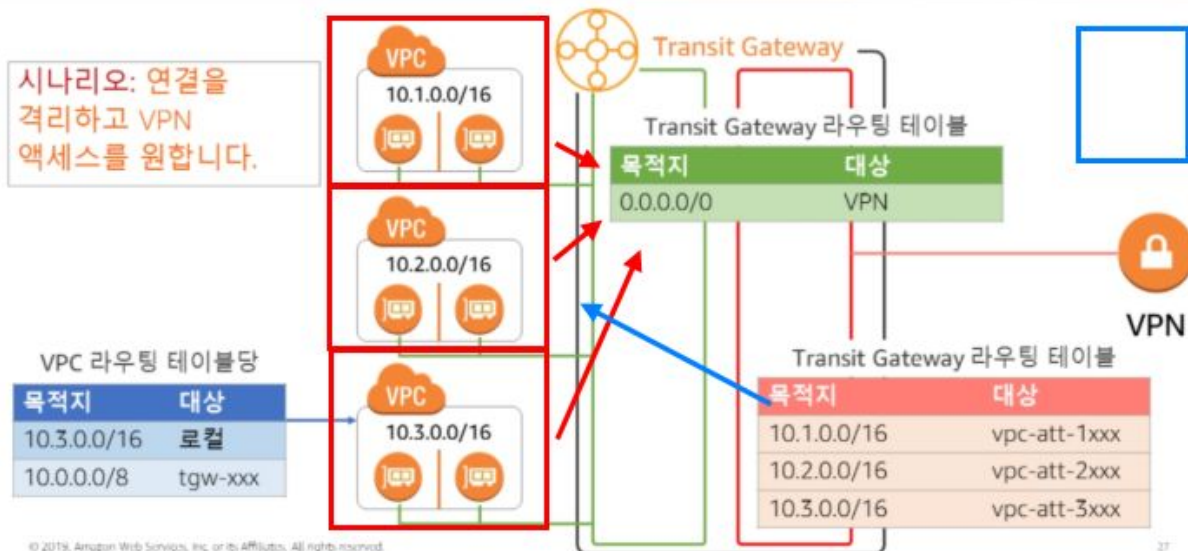


*** Transit Gateway

- > 최대 5000개의 VPC와 온프레미스 환경을 연결, 가운데에서 허브같은 역할
- > 완전관리형,고가용성, 유연한 라우팅 서비스



- > Gateway환경안에 여러개의 라우팅 테이블을 정의 할 수 있음
- > 이 기능으로 VPN 전용 라우터를 따로 만들어 양방향 통신이 가능하게 하고, VPC간에는 격리를 시킬 수 있음



*** VPC 엔드포인트

- > DynamoDB, S3 등과 같은 인터넷이 필요한 서비스들을 인터넷망이 존재하는 인프라에서 사용하게 만들 수 있는 기능
- > PrivateLink를 구동하여 인터넷 GW를 통하지 않고 접근 가능

- 1) 인터페이스 엔드포인트
- 2) 게이트웨이 엔드포인트

*** VPC DNS (Route 53)

- > Route 53에서 퍼블릭, 프라이빗 호스팅을 설정 할 수 있음
- > Route 53 resolver에서 인바운드, 아웃바운드 설정

인터페이스 엔드포인트

- Amazon CloudWatch Logs
- AWS CodeBuild
- Amazon EC2 API
- Elastic Load Balancing API
- AWS Key Management Service (AWS KMS)
- Amazon Kinesis Data Streams
- AWS Service Catalog
- Amazon Simple Notification Service (Amazon SNS)
- AWS Systems Manager
- 다른 AWS 계정에서 호스팅하는 엔드포인트 서비스
- 그 외 다수

게이트웨이 엔드포인트

- Amazon Simple Storage Service(Amazon S3)
- Amazon DynamoDB

** VPC 보안



AWS Shield Standard는 가장 일반적이고 빈번하게 발생하며 웹 사이트 또는 애플리케이션을 목표로 하는 네트워크 및 전송 계층 DDoS 공격으로부터 보호합니다.

<ELB> Elastic Load Balancing



- 동일한 목적의 서버의 세션을 균등하게 분산해주는 서비스
- 헬스체크 기능을 갖고 있어 장애가 있는 인스턴스에는 트래픽을 분산하지 않음
- 완전관리형 서비스, 세부적인 설정은 불가하지만 장애복구 기능
- 각 로드밸런서에는 DNS이름이 부여된다(엔드포인트)
- 고가용성, 상태확인(헬스체크), 어느정도의 Ddos 보호(보안)
- LB가 직접 SSL 복호화를 지원(중앙 집중식 관리)하여 각 인스턴스들과는 평문통신을 해서 유연성이 생김(TLS 종료)
- 외부(external), 내부(internal)에 위치 할 수 있음
- 대상그룹에 해당되는 인스턴스가 여러 AZ에 걸쳐 가용중이라면, 인스턴수의 수를 같게 해야함(각각 AZ에 1대1로 분산하는 성격의 이유)

ELB: 옵션

Application Load Balancer	Network Load Balancer	Classic Load Balancer
<div>HTTP HTTPS</div> <ul style="list-style-type: none"> • 유연한 애플리케이션 관리 • HTTP 및 HTTPS 트래픽의 고급 로드 밸런싱 • 요청수준(계층 7)에서 운영됨 • (계층 7) 	<div>TCP, TLS, UDP</div> <ul style="list-style-type: none"> • 애플리케이션에 대한 탁월한 성능 및 정적 IP • TCP, TLS, UDP 트래픽의 로드 밸런싱 • 연결 수준(계층 4)에서 운영됨 	<div>이전 세대 (HTTP, HTTPS 및 TCP용)</div> <ul style="list-style-type: none"> • EC2 Classic 네트워크 내에 구축된 기존 애플리케이션 • 요청 수준 및 연결 수준에서 운영됨

- Application LB (L7스위치)
: 어플리케이션 계층의 프로토콜(HTTP,HTTPS)에 대하여 LB를 해주는 옵션
 : 뒷단에 웹서버를 갖고 있으면 대부분 해당 LB를 사용
- Network LB
: 클라이언트로부터 수신되는 트래픽을 수락하고, 해당 트래픽을 동일한 가용영역 내 대상 전체로 분산 , TCP/UDP 프로토콜에만 의존하는 애플리케이션에 사용하는 옵션

*** Connection Draining

> 트래픽이 감소하여 Scale in을 할 때, 바로 인스턴스가 삭제되는 것이 아니라 고객이 요청한 응답을 모두 처리하고 삭제되는 기능



상태 검사

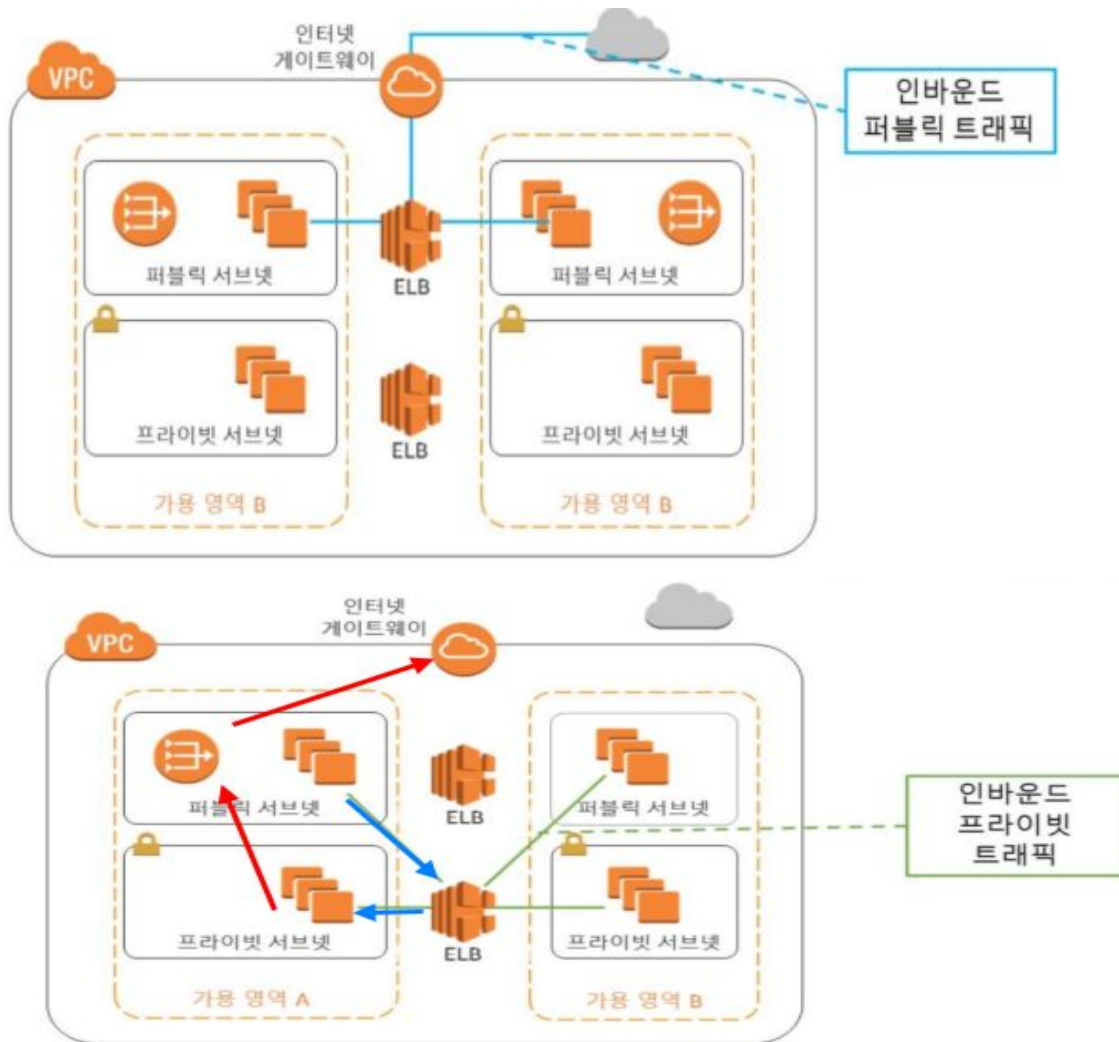
프로토콜 ⓘ

HTTP

경로 ⓘ

/

> URL Path(프로토콜,경로) 정보에 따라 각 기능을 하는 서버에게만 트래픽을 분산하는 기능



<Route 53>

- 도메인 이름을 IP주소로 변환하는 DNS server의 역할
- 도메인을 구입하여 등록하고 이에대한 설정을 자동으로 구성
- Region,AZ 단위의 보조서버 등록 가능
- S3 의 정적웹사이트 , 다른 웹인스턴스를 secondary 레코드(보조서버)로 설정
- 상태검사를 통한 SNS 알림 서비스를 제공하여 빠른 장애복구 조치 지원

Route 53

대시보드

호스팅 영역

상태 검사

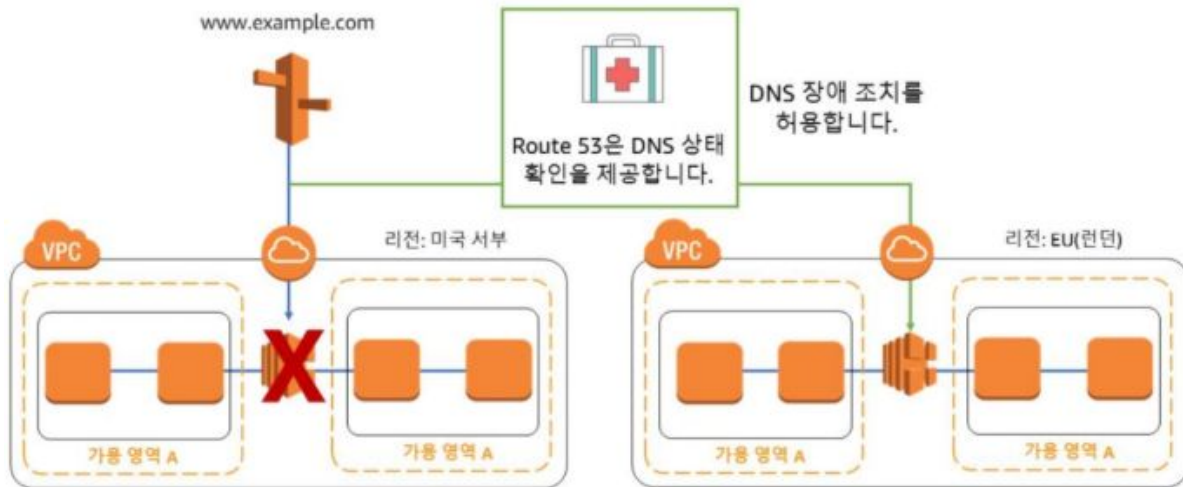
▼ 도메인

등록된 도메인

대기 중인 요청

*** Failover test

> Route53은 해당 A 레코드를 항상 감시하여 장애가 발생하면 보조 서버,보조 ELB의 IP를 사용자에게 알려주는 기능



*** Route 53 라우팅 옵션

간단한 라운드 로빈

가중치 기반 라운드 로빈

지연 시간 기반 라우팅

상태 확인 및 DNS 장애 조치

지리 위치 라우팅

트래픽 바이어스를 통한 지리 근접 라우팅

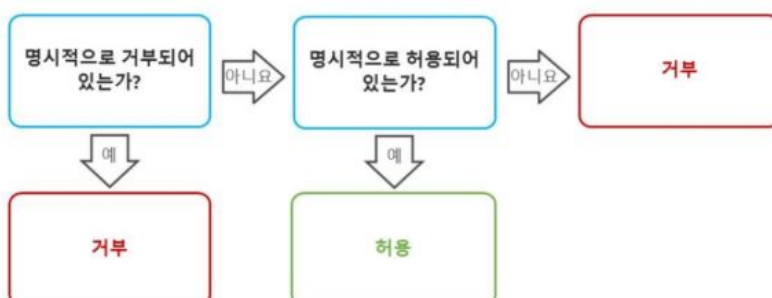
다중 값 응답



모듈7. IAM

<IAM 사용자>

- AWS계정(루트 사용자)은 계정의 모든 서비스 및 리소스에 대한 전체 액세스 권한(결제정보,개인데이터,전체 아키텍처 및 해당 구성요소)을 가짐
- AWS 계정은 IAM 사용자를 생성하고, 각 담당하는 역할에 따라 적절한 권한을 부여
- AWS 계정으로의 자격증명(로그인)을 사용하지 않는 것이 좋음
- 사용자들에게 최소한의 권한만 부여하는것이 보안에 좋음
- 사용자들에게 여러 정책을 부여할 경우, 정책에 맞물려 있는 것들 중 명시적인 거부가 1순위로 적용되고, 명시적인 허용이 2순위, 묵시적인 허용/차단 권한이 그 다음의 우선순위로 부여된다
- Billing(결제정보) 서비스에대한 권한을 갖고 있더라도 루트사용자가 다른 설정을 해줘야 접근 가능



루트사용자 - 내계정 - 하단페이지

▼ 결제 정보에 대한 IAM 사용자 및 역할 액세스

IAM 사용자 및 연합된 사용자에게 계정 정보에 액세스하는 역할 권한을 줄 수 있습니다. 여기에는 계정 설정, 결제 방법 및 보고서 페이지에 대한 액세스가 포함됩니다. IAM 정책을 생성함으로써 어떤 사용자 및 역할이 결제 정보를 볼 수 있는지 제어합니다. 자세한 내용은 [Controlling Access to Your Billing Information](#)을 참조하십시오.

☒ IAM 액세스 활성화

업데이트

취소

IAM 정책 생성 옵션

▼ 작업 EC2에서 허용되는 작업 지정 ?

[권한 거부로 전환](#) ⓘ

닫기

Q 작업 필터링

수동 작업 (작업 추가)

☐ 모든 EC2 작업(ec2:*)

• 리소스 기반 – 연결된 AWS 리소스

• 자격 증명 기반 – 연결된 IAM 보안 주체

<IAM 정책>

1. 자격증명정책

- 각 리소스에 대한 권한을 묶어 정책을 만들고 해당 정책을 IAM 사용자, 그룹, 역할에 부여
- 필요한 권한들을 모아 미리 정의해둔 AWS 관리형 정책, 사용자가 필요로하는 권한들을 모아 정의한 사용자 지정 정책이 존재
- 하나의 계정에 다양한 정책을 부여 할 수 있음
- 정책 유형은 고객 관리형, AWS 관리형, 인라인 정책
- 인라인 정책은 1회성의 성격으로 한 사용자에게 정책을 부여하면 해당 설정이 남아있지 않음

2. 리소스 기반 정책

- AWS 리소스에 부여하는 정책 (ex.S3버킷정책)
- 지정된 보안 주체가 해당 리소스에 대해 수행할 수 있는 작업 및 관련 조건을 제어
- 리소스 기반 정책은 인라인 정책, 관리형 정책은 존재하지 않음

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": "*",
    "Resource": "*"
  }
]
```

> 정책에 관한 JSON 코드 Effect(허용,차단) Action(모든활동) Resource(모든대상)

```

{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": ["dynamodb:*", "s3:*"],
    "Resource": ["arn:aws:dynamodb:region:account-number-without-hyphens:table/table-name",
    "arn:aws:s3::bucket-name",
    "arn:aws:s3::bucket-name/*"]
  },
  {
    "Effect": "Deny",
    "Action": ["dynamodb:*", "s3:*"],
    "NotResource": ["arn:aws:dynamodb:region:account-number-without-hyphens:table/table-name",
    "arn:aws:s3::bucket-name",
    "arn:aws:s3::bucket-name/*"]
  }
]
}

```

사용자에게 액세스 권한 부여(특정 DynamoDB 테이블과...

...특정 Amazon S3 버킷 및 해당 콘텐츠

명시적 거부 문은 보안 주체가 지정된 테이블 및 버킷이 아닌 AWS 작업 또는 리소스를 사용할 수 없도록 합니다.

명시적 거부 문은 허용문보다 우선 적용됩니다.

> 명시적 허용과 명시적 거부를 같이 사용하는게 보안상 안전함

<IAM Group>

- 사용자 개별에게 정책을 부여하는 것에 대한 한계
- 해당그룹에 필요한 정책을 부여하고 새로운 사용자를 그룹에 추가
- 사용자 개별보다는 그룹으로 묶어서 사용하는 것을 권장

<IAM Role>

- 임시적인 자격증명을 부여하여 단기적인 작업을 할 때 사용(테스트)
- 사용자가 역할을 부여받으면 원래 가지고 있던 권한을 버리고, 역할의 권한을 부여
- IAM 사용자 또는 그룹에 따로 연결되지 않으며, 사용자 또는 서비스에 위임
- AWS 리소스에 AWS 서비스에 대한 액세스를 제공
- 외부 인증 사용자(다른계정)에게 액세스를 제공
- 세션 지속시간동안만 해당 role의 액세스를 부여
> 세션시간이 만료되면
- role을 정의하고, 정책을 생성하여 role 전환을 할 수 있는 권한을 사용자에게 부여
- 해당 사용자로 로그인후, 역할전환



최대 세션 지속 시간 1 시간

*** STS(Security Token Service) - 쓰기 권한(AssumeRole)

> 해당 권한을 가진 사용자가 Role을 가지고 올 수 있는 권한, Resource에서 role의 ADN을 부여하여 정책을 부여받은 사용자가 사용할 role에 대한 제어 가능



user01 @ dongbari

IAM 사용자:
user01
계정:
dongbari
내 계정
내 조직
내 서비스 할당량
내 결제 대시보드
주문 및 인보이스
내 보안 자격 증명
역할 전환

```
****
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::802163073439:role/ec2_role"
    }
  ]
}
```

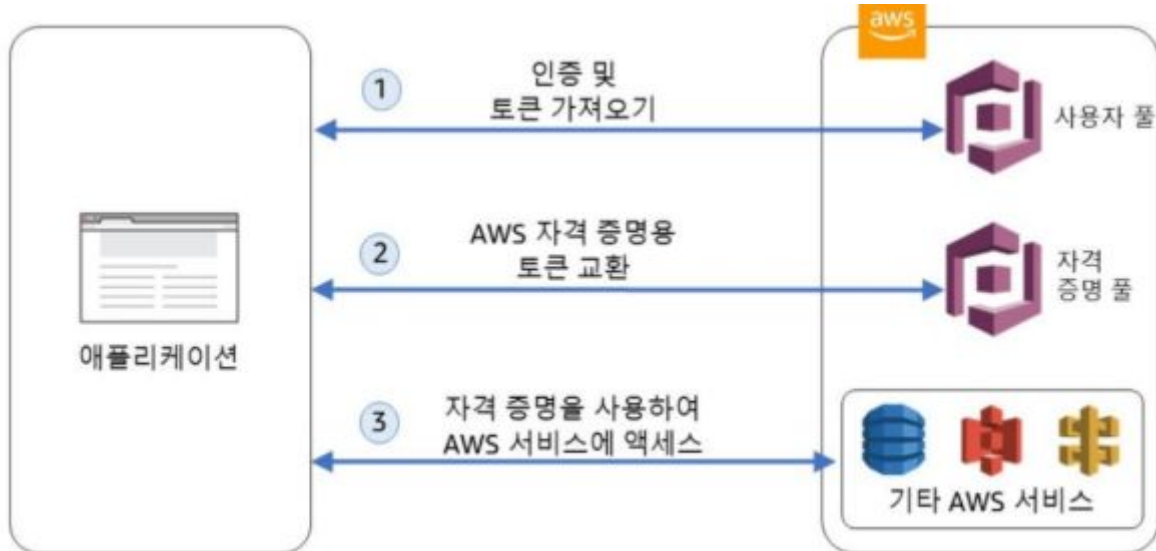
리소스에 특정
서비스 권한 부여
1. 리소스의
소스코드에 사용자
액세스 키 부여

- > 리소스에 액세스키를 부여하면 마치 해당 사용자처럼 부여된 권한을 위임받는다
- > 액세스키가 공개적인 환경에서도 보일 수 있는 위험이 있기 때문에 보안상 권장하지 않음

2. 리소스에 IAM role 부여

<Amazon Cognito>

- 웹, 모바일 어플리케이션 등 AWS와 독립된 프로그램에 자격증명을 해주는 관리형서비스
- 1. 사용자풀
: 애플리케이션 사용자의 인증정보를 제공하고 토큰을 부여하는 디렉터리
- 2. 자격증명풀
: 사용자에게 토큰을 받으면 해당 role을 임시부여하는 디렉터리



<AWS Organization>

- 다중 VPC와 다중계정이 AWS 권장사항(Role을 사용한 교차계정 액세스)
- 각 서비스(VPC)의 분리를 통해 보안을 강화
- 이런 환경에서 여러 계정들을 한번에 관리하기 위한 중앙집중형관리 시스템
- 그룹(API 단위)기반으로 계정을 관리하고, 통합적인 결제
- 보안상의 이유로 여러계정을 각 조직내에서 사용
- > 다양한 사업부의 개별 계정
- Root를 시작으로 트리 형식으로 ou,계정이 뻗어있는 구성이다.



admin_role @ 802163073439

현재 다음으로 활성:

admin_role

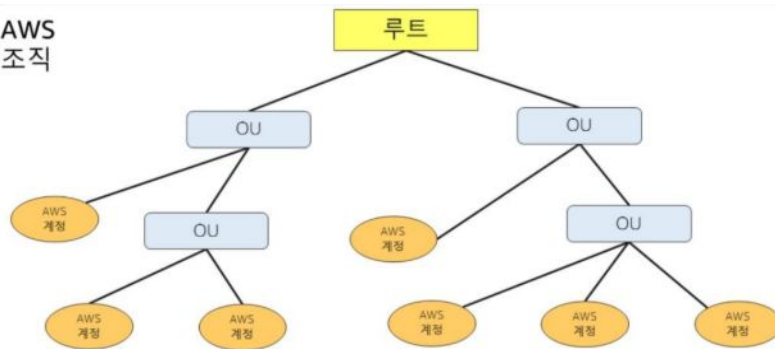
계정:

dongbari

내 계정

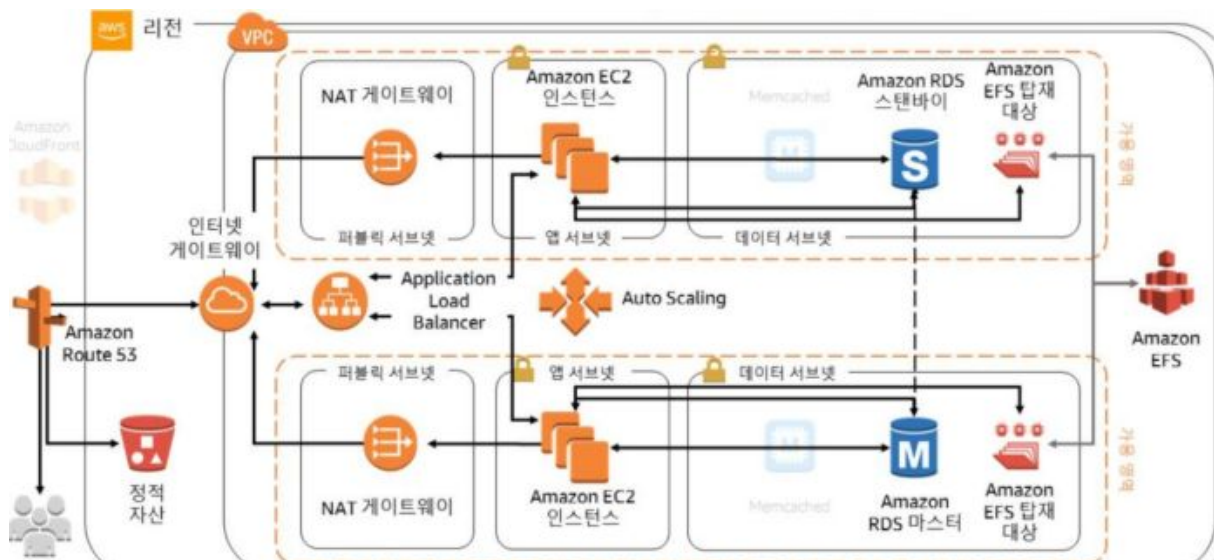
내 조직

AWS 조직



- 상위 OU에 정책을 적용하면, 하위 OU에도 정책이 적용된다

모듈8. 탄력성,고가용성 및 모니터링 > 온프레미스와 비교하여 클라우드의 가장 큰 장점



<Auto Scaling> 탄력성

- 조건을 설정하여(CPU) 인스턴스를 시작 또는 종료하는 서비스
- 새인스턴스를 자동으로 추가, 삭제하여 탄력성을 보장
- 여러 가용영역에 걸쳐 Auto Scaling Group을 설정 할 수 있음
- 최소용량과, 최대용량을 설정하여 인스턴스의 탄력성을 관리

<CloudWatch> 감시

- 리소스에 관한 지표를 정하고, 그 지표를 수집하고 추적함
- 경보를 생성하고 알림을 전송(SNS,Twilio,SQS,Auto scaling 정책)
- 다양한 리소스들과의 연동으로 모든 인프라의 상태를 확인하고 경보를 받을 수 있음
- 설정한 규칙에 따라 리소스의 용량 변화를 트리거 할 수 있음
- 최대 보존기간은 2주, 최소 분석할 수 있는 시간은 1분(default 5분)
- 보존기간이 짧고, 더 세부적인 분석을 원한다면 다른 오픈소스의 감시 프로그램 사용

*** CloudWatch 에이전트

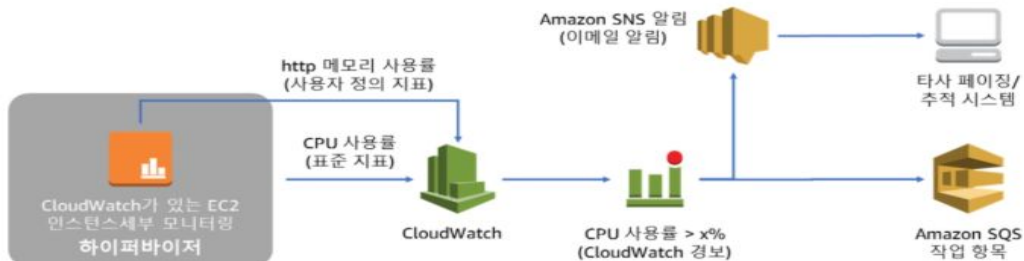


시스템 수준 지표를 수집하는 CloudWatch 에이전트

- ✓ Amazon EC2 인스턴스
- ✓ 온프레미스 서버

> EC2에서는 기본으로 추가되어있고, On Promise 환경에서는 추가를 따로 해야함

*** CloudWatch 모니터링 과정



*** CloudWatch 지표

표준 지표

- 서비스 이름을 기준으로 그룹화
- 직전 15개월 이내에 서비스를 사용한 경우에만 제공
- 선택한 지표를 비교할 수 있도록 그래픽 방식으로 표시
- AWS CLI 또는 API를 통해 프로그래밍 방식으로 연결 가능

사용자 지정 지표

- 사용자 정의 네임스페이스를 기준으로 그룹화
- AWS CLI, API 또는 CloudWatch 에이전트를 사용하여 CloudWatch로 게시

*** 경보상태



*** CloudWatch 이벤트

> AWS 환경의 변화를 나타냄

> AWS 리소스는 상태가 변경되면 이벤트를 생성 할 수 있음

ex) EC2 동작상태 변경 후 이벤트 생성

> 대상

: 이벤트를 처리하는 역할, 대상은 JSON 형식으로 이벤트를 수신

> 규칙

: 들어오는 이벤트에서 일치하는 것을 찾아서 대상으로 라우팅하여 처리

단일 규칙으로 여러개의 대상으로 라우팅을 할 수 있으며, 이들은 모두 병렬 처리

이벤트

규칙

이벤트 버스

*** CloudWatch 로그

> EC2 인스턴스에서 AWS에 어떤 로그든 자동으로 업로드 할 수 있는 기능 제공



<CloudTrail>

- API 호출을 기록하고 로그파일을 사용자에게 전달하는 웹 서비스
- 리전단위로 활성화
- 호출자의 자격증명, 호출시간, IP 주소, 요청파라미터 반환 응답 같은 정보가 기록

CloudTrail은 **계정에서 이루어지는 모든 API 호출을 기록하고**, 지정된 Amazon S3 버킷에 **로그를 저장합니다**.



- 오랫동안 실행되는 인스턴스가 종료된 이유는 무엇이며, 누가 종료했습니까? (조직적 추적 가능성 및 책임)
- 누가 보안 그룹 구성을 변경했습니다? (책임 및 보안 감사)
- 알 수 없는 IP 주소 범위에서 나온 작업이 있습니까? (퍼블릭 네트워크를 향한 잠재적 외부 공격)
- 권한 부족으로 인해 거부된 작업은 무엇입니까? (네트워크를 향한 잠재적 내외부 공격)

**** 이벤트 헤더

```
{
  "eventVersion" : "1.01",
  "userIdentity" : {
    "type" : "IAMUser",
    "principalId" : "AIDAYYYYYYYYYYYYYYYY",
    "arn" : "arn:aws:iam:xxxxxxxxxxxx:user/tests3user",
    "accountId" : "xxxxxxxxxxxx",
    "userName" : "tests3user"
  },
  "eventTime" : "2018-09-23T22:41:38Z",
  "eventSource" : "signin.amazonaws.com",
  "eventName" : "ConsoleLogin",
  "awsRegion" : "us-east-1",
  "sourceIPAddress" : "54.240.217.10",
  "userAgent" : "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0)"
}
```

<VPC Flow Logs>

- VPC의 네트워크 인터페이스에서 송수신되는 IP트래픽에 대한 정보를 캡처 할 수 있는 기능
- VPC 흐름 로그 데이터는 CloudWatch logs를 통해 저장되고, 데이터를 확인하고 가져올 수 있음

모듈9 자동화

*** 수동 프로세스의 위험

> 버전관리 불가, 감사내역부족, 일관성 없는 데이터 관리

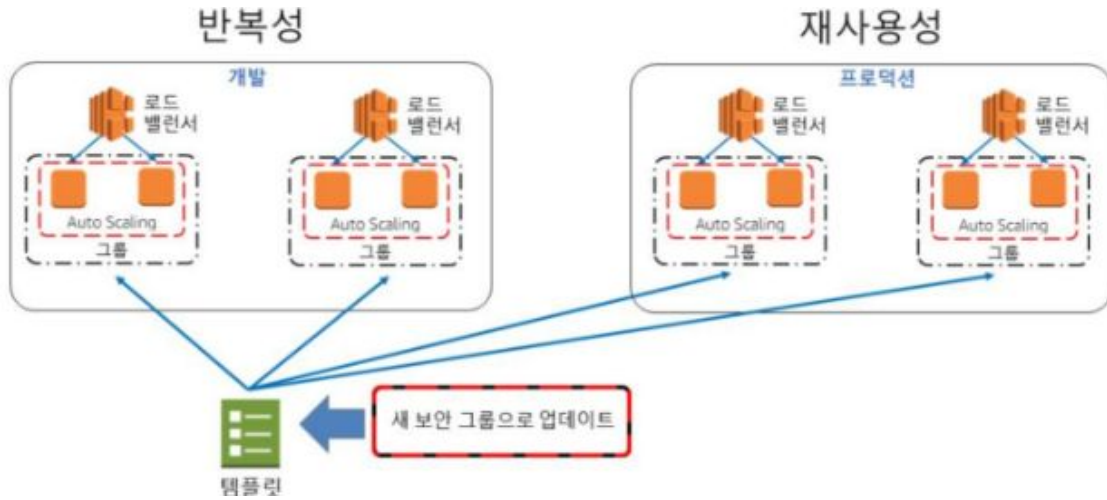
<CloudFormation>

- JSON/YAML 형식의 코드(템플릿)를 추가하여 스택을 생성

- 업데이트를 통한 인프라를 확장/변경

*** 코드형 인프라(IaC)

- > 인프라를 코드형으로 구축하는 개념(반복성,재사용성)
- > 조건에 따른 다른 환경을 생성 할 수 있음 (



*** 계층화된 아키텍처



- > 템플릿을 각 인프라/기능에 따라 분류해서 여러개를 생성한다
- > 1개의 기능을 템플릿에 구현시키는것이 일반적이다

*** Quick Start / 템플릿 조각

AWS 서비스별 템플릿 및 조각

AWS 서비스별 [샘플 템플릿](#)을 찾아보십시오.

AWS 서비스별 [템플릿 조각](#)을 찾아보십시오.

더 많은 사례와 [참조](#)를 보려면 [개발자 설명서](#)를 참조하십시오.

AWS 리전별 [샘플 템플릿](#)을 찾아보십시오.

참조 구현

[AWS Quick Start](#)는 Windows Server 및 SAP HANA 같은 널리 사용되는 IT 워크로드에 대한 AWS CloudFormation 템플릿과 자세한 배포 안내서를 제공합니다.

샘플 솔루션

▼ 사용 사례별로 필터링

- ☐ 분석
- ☐ 블록체인
- ☐ 비즈니스 생산성
- ☐ 소통
- ☐ 문의 센터
- ☐ 컨테이너 & 마이크
로 서비스
- ☐ 데이터 레이크
- ☐ 데이터베이스
- ☐ DevOps
- ☐ 의료 서비스
- ☐ 인프라
- ☐ 통합 플랫폼
- ☐ IoT
- ☐ 생명 과학
- ☐ 기계 학습 및 AI
- ☐ 미디어 서비스
- ☐ 마이그레이션
- ☐ 네트워킹 및 원격 액
세스

> AWS에서 제공하는 샘플 템플릿

> 각 기능들에 대한 Yaml/JSON 기본코드들을 제공하는
'템플릿 조각'

> 인프라 사례별로 나누는 'Quick Start'

> 모든 부트스트래핑 및 배포를 사용자 대신에 처리

> 모든 구성요소가 어떻게 생성되었는지 보였누느 배포안내서 제공

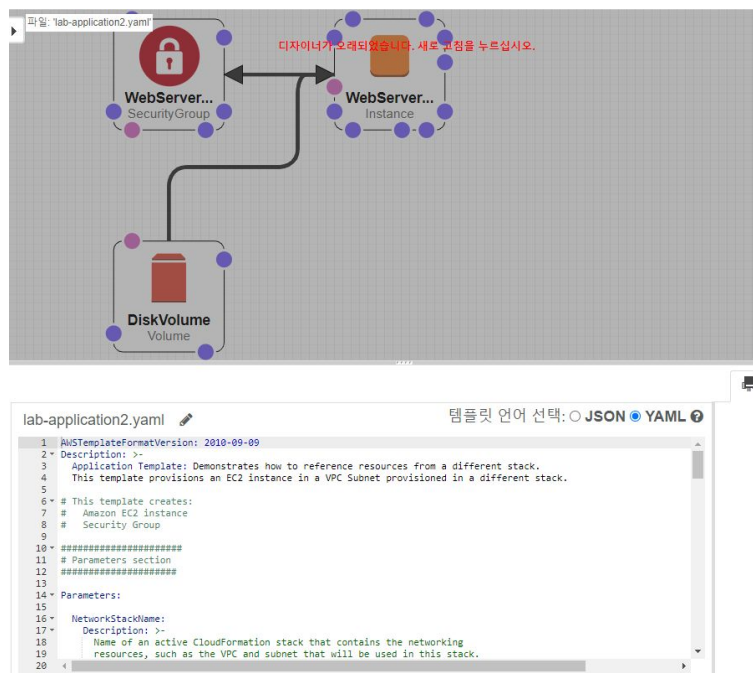
> 생성 및 실행하는데 요금이 발생

*** StackSet ??

*** AWS CloudFormation Designer

> 템플릿을 작성,확인,수정하기 위한 그래픽 도구

> 드래그 앤 드롭 인터페이스를 사용하여 템플릿 리소스를 다이어그램으로 표시 후, 편집기를 사용해 해당 세부정보를 편집 할 수 있음



<Systems Manager>

- 소프트웨어 인벤토리, os 패치 적용, 시스템 이미지 생성, Windows 및 Linux 운영체제 구성을 수행 할 수 있는 관리형 서비스
- on-promise(별도의 agent 프로그램 설치) 또는 AWS에서 실행되는 시스템의 구성과 관리를 자동화하는데 중점을 둠
- 관리할 인스턴스를 선택하고 수행할 관리작업을 정의

- SSH를 수행할 필요 없이 인스턴스의 소프트웨어 구성을 검토하고 유효성 검사
- cloudformation으로 ec2를 자동 생성 하고, 내부적인 script 구성들을 자동화
- SystemManager의 권한을 가지고 있어야 함



명령 실행



유지 관리 기간



패치 관리



상태 관리자



세션 관리자



인벤토리

> Run Command

: SSH, 원격 Powershell로 로그인 하지 않고, 안전하게 콘솔 환경에서 인스턴스의 레지스트리 편집, 사용자관리, 소프트웨어 및 패치 설치와 같은 일반적인 관리작업을 그룹전체에서 자동화하는 간단한 방법, IAM과의 통합을 통해 세분화된 권한을 적용하여 사용자가 인스턴스에서 수행할 수 있는 작업제어

Run command 환경에서 어플리케이션 설치

명령 실행	이름	소유자	플랫폼 유형
	qls-14846-3cdc22f35c55329f-InstallDashboardApp-8VLHAQAWI73T	190528540052	Linux

CLI 환경에서의 명령어

플랫폼

Linux/Unix/OS X

CLI 명령

```
aws ssm send-command --document-name "qls-14846-3cdc22f35c55329f-InstallDashboardApp-8VLHAQAWI73T" --document-version "1" --targets '[{"Key": "InstanceIds", "Values": ["i-01ba3355b3fc9a85"]}]' --parameters '{}' --timeout-seconds 600 --max-concurrency "50" --max-errors "0" --region ap-northeast-1
```

> 세션관리자

: 대화형 원클릭 브라우저 기반 셸 또는 AWS CLI를 통해 인스턴스를 관리 할 수 있는 완전관리형 기능. 인바운드 포트를 열거나 접속 호스트를 유지하거나 SSH 키를 관리할 필요 없이 안전하고 감사 가능한 인스턴스 관리가 가능

세션관리자를 통한 접속

<input type="radio"/>	Managed Instance	i-01ba3355b3fc9a85	2.3.1319.0	🟢 실행 중
<div> <div>취소</div> <div>세션 시작</div> </div>				

세션 ID: awsstudent-08b8cd8aae0bf281b

인스턴스 ID: i-01ba33555b3fc9a85

sh-4.2\$

> 패치관리자

: 승인하거나 거부하는 자동 승인 규칙이 포함된 패치 기준을 생성해 패치 적용을 자동화 하는 기능

> 파라미터 스토어

: 콘솔환경에서 암호,키,라이선스코드,DB 문자열 같은 구성정보를 파라미터로 설정하여 스크립트와 명령에서 참조하는 기능.

파라미터 지정

이름

Q 파라미터 이름 또는 경로를 입력하십시오.

설명 — Optional

유형

☒ 문자열
모든 문자열 값입니다.

☐ 문자열 목록
심표를 사용하여 문자열을 구분합니다.

☐ 보안 문자열
계정 또는 다른 계정의 KMS 키를 사용하여 중요한 데이터를 암호화합니다.

데이터 형식

text ▼

값

암호화되지 않은 파라미터 값입니다.

최대 길이는 4,096자입니다.

> 인벤토리

: 인스턴스와 인스턴스에 설치된 소프트웨어에 대한 정보를 수집하여 시스템 구성과 설치된 어플리케이션을 이해 할 수 있음.

인벤토리를 통한 인스턴스에 대한 정보 자동 수집

인스턴스 ID: i-01ba3355b3fc9a85 작업 ▼

설명 **인벤토리** 태그 관리 연결 패치 구성 규정 준수

인벤토리 유형
AWS:Application ▼

Q < 1 2 3 4 5 ... >

이름	버전	게시자	애플리케이션 유형	설치 시간 (UTC)
pth	2.0.7	Amazon Linux	System Environment/Libraries	Mon, 22 Jun 2020 19:50:28 GMT
kbd-misc	1.15.5	Amazon Linux	System Environment/Base	Mon, 22 Jun 2020 19:50:04 GMT
libtasn1	4.10	Amazon Linux	System Environment/Libraries	Mon, 22 Jun 2020 19:50:29 GMT
basesystem	10.0	Amazon Linux	System Environment/Base	Mon, 22 Jun 2020 19:50:05 GMT

<AWS OpsWorks>



CHEF



puppet

- CHEF(개발)/Puppet(개발)을 사용하여 모든 형태와 규모의 애플리케이션을 구성하고 운영하도록 지원하는 구성 관리 서비스
- 애플리케이션의 아키텍처 및 각 구성 요소의 사양을 정의
- 구성요소에는 패키지 설치, 소프트웨어 구성 및 리소스(ex.스토리지)가 포함
- 수명주기 이벤트가 존재

*** CloudFormation X OpsWorks

AWS CloudFormation을 사용하여 인프라(VPC, IAM 역할)를 구축하고, AWS OpsWorks Stacks를 사용하여 애플리케이션 계층을 배포합니다.



> 템플릿으로 환경의 인프라를 생성하고(VPC,IAM..) 별도의 템플릿을 사용하여 해당 인프라 내에서 배포되는 OpsWorks Stacks 스택을 생성한다

<Elastic Beanstalk>

- AWS 인프라를 모르는 개발자가 개발코드만 작업하면 자동으로 해당 환경에 맞는 인프라를 구성

- 사용자는 애플리케이션을 디플로이 하기만 하면 서비스를 자동 시작 할 수 있음
- war(자바) , zip(이외의 언어) 파일을 업로드해서 디플로이
- Web server(http/https)와 Worker라는 2가지 환경 종류가 있음



AWS Elastic Beanstalk

인프라를 프로비저닝 및 운영하고 **사용자를 위해 애플리케이션 스택을 관리**

완전한 투명성 - 생성되는 모든 것을 확인할 수 있습니다

적절한 규모를 유지합니다. 애플리케이션의 크기를 **자동으로 늘리거나 줄입니다**

고객



코드

HTTP 서버

애플리케이션 서버

언어 인터프리터

운영 체제

호스트



상위 수준 서비스

DIY



AWS Elastic Beanstalk



AWS OpsWorks



AWS CloudFormation

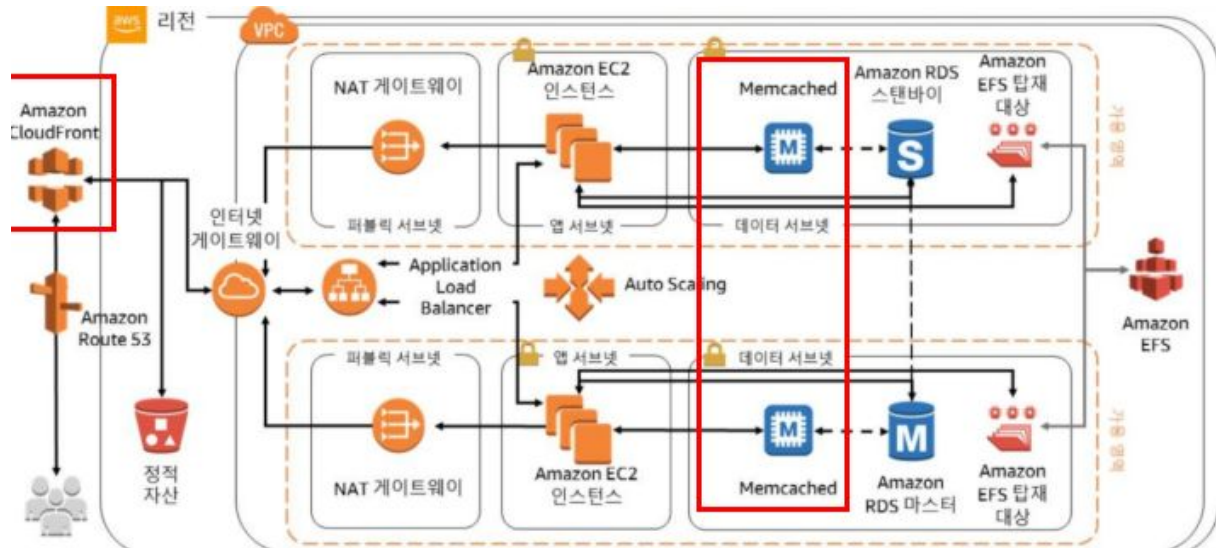


Amazon EC2

간편성

제어

모듈10. 캐싱



*** 캐시

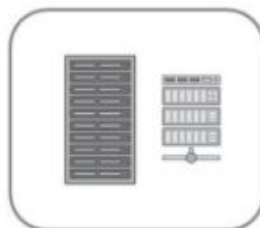
- > 데이터를 매번 서버에서 액세스하는게 아니라 데이터 앞단에 캐싱저장소를 만들어 좀더 신속한 데이터 액세스가 가능
- > 정적인 데이터를 자주 액세스하는 데이터, 수집이 느리고 비싼 쿼리가 필요한 데이터, 주식 가격 처럼, 일정기간 동안 변화가 없을 수 있는 정보에 캐시 데이터를 사용
- > 애플리케이션의 속도 향상, DB쿼리의 부담완화, 응답지연시간 감소

클라이언트 측



클라이언트 측 웹 브라우저

서버 측



웹 서버



역방향 프록시 캐시

*** AWS 캐시

1. CDN (Contents Delivery Network)

- > 콘텐츠를 효율적으로 전달하기 위해 여러 노드를 가진 네트워크에 데이터를 저장하여 제공하는 시스템
- > ISP의 CDN 서버에 콘텐츠를 분산시키고 유저의 네트워크 경로 상 가장 가까운 곳의 서버로부터 콘텐츠를 전송받도록 하여 트래픽이 특정 서버에 집중되지 않고 각 지역 서버로 분산되도록 하는 기술

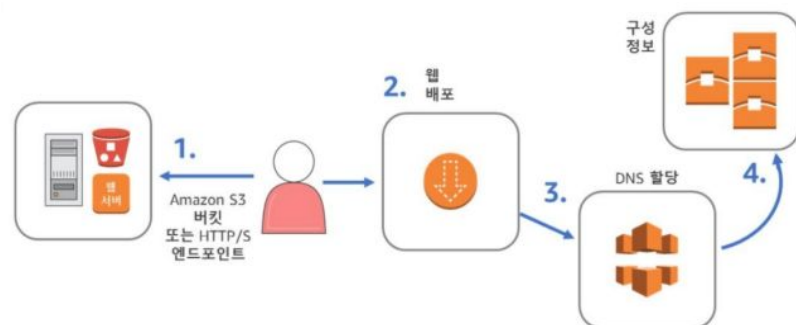


<CloudFront>

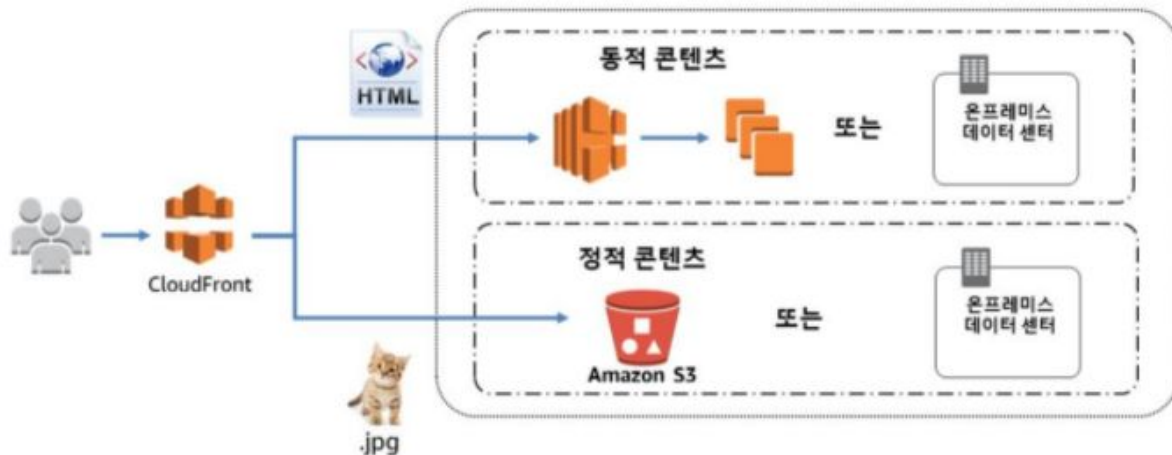
- 엣지로케이션 단위로 생성
- AWS CMS 서비스를 사용하지 못하는 인스턴스에 자격(SSL)을 대신 부여
- DDos와 같은 외부 공격을 완화하는 AWS shield와 연동을 통해 보안강화
- 그외 S3,EC2,ELB,Lambda@Edge와 같은 서비스와 연동
- 사용자가 캐싱 데이터의 엔드포인트로 접속하면 최초 1회 엣지로케이션이 origin 서버에 세션을 요청한다
- 그 후, 사용자는 캐시에 저장된 데이터를 통해 신속하게 접속 할 수 있음
- TTL 값을 정의하여 캐시 콘텐츠 만료 시간을 정할 수 있음



- > 보안 : SSL(https)을 통해 콘텐츠를 안전하게 제공
- > 동적 : 사용자 정의 콘텐츠 및 캐싱할 수 없는 콘텐츠
- > 사용자 입력 : http의 작업을 지원(Put/Post)
- > 정적 : TTL이 높은 이미지, js, html
- > 동영상 : rtmp 및 http 스트리밍 지원

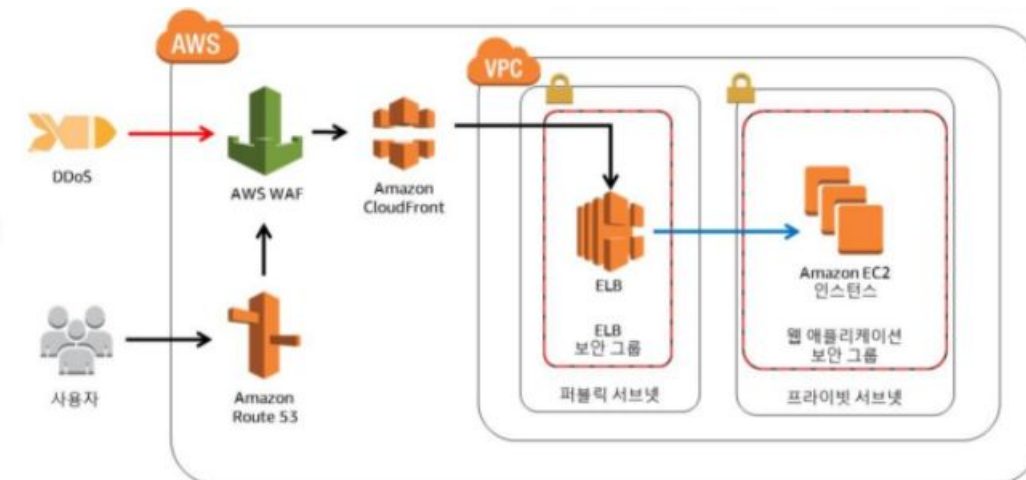


*** 일반적인 CloudFront를 구성방법



- > 사용자는 주로 정적 콘텐츠를 캐시한다
- > 정적 자산에 대해 상대 URL 참조 대신, 절대 URL 참조를 생성
- > 정적 자산을 S3에 저장하고, WORM을 최적화
- > 동적 콘텐츠의 캐싱은 아키텍처의 성능 향상에 영향을 끼친다
- > 동적 콘텐츠도 S3에 저장 후 캐싱하면 성능향상에 도움이 된다

*** DDoS 공격에 대한 보호



- > Cloudfront는 DDoS 공격을 방지/완화하는데 도움이 될 수 있는 복원력이 뛰어난 아키텍처
- > CloudWAF 서비스와의 연동으로 1차적인 방화벽을 구축, Route53 및 API Gateway와 같이 엣지 로케이션에서 제공하는 서비스를 이용하여 더 큰 내결함성을 제공하고 더 많은 양의 트래픽을 관리하기 위한 확장성을 증진
- > 또한 ELB 및 EC2 와 같이 리전에서 제공하는 서비스를 사용하여 DDoS 복원력을 구축하며, 확장을 통해 특정 리전내에서 예상치 못한 트래픽 양을 처리가능

*** Cloudfront 설정 예

Web

Create a web distribution if you want to:

- Speed up distribution of static and dynamic content, for example, .html, .css, .php, and graphics files.
- Distribute media files using HTTP or HTTPS.
- Add, update, or delete objects, and submit data from web forms.
- Use live streaming to stream an event in real time.

You store your files in an origin - either an Amazon S3 bucket or a web server. After you create the distribution, you can add more origins to the distribution.

[Get Started](#)

RTMP



CloudFront is discontinuing support for RTMP distributions on December 31, 2020. For more information, please [read the announcement](#).

Create an RTMP distribution to speed up distribution of your streaming media files using Adobe Flash Media Server's RTMP protocol. An RTMP distribution allows an end user to begin playing a media file before the file has finished downloading from a CloudFront edge location. Note the following:


- To create an RTMP distribution, you must store the media files in an Amazon S3 bucket.
- To use CloudFront live streaming, create a web distribution.

[Get Started](#)

> CloudFront 옵션

Price Class	Use All Edge Locations (Best Performance) ^	
Default Cache Behavior Web ACL	Use Only U.S., Canada and Europe Use U.S., Canada, Europe, Asia, Middle East and Africa Use All Edge Locations (Best Performance)	
Alternate Domain Names (CNAMEs)		

> 엣지로케이션 생성 지역 옵션

Alternate Domain Names (CNAMEs)	<input type="text"/>	
SSL Certificate	<input checked="" type="radio"/> Default CloudFront Certificate (*.cloudfront.net)	
	Choose this option if you want your users to use HTTPS or HTTP to access your content (e.g., https://d1111111abcdef8.cloudfront.net/logo.jpg). Important: If you choose this option, CloudFront requires that browser	

> SSL 자격증명 옵션

Logging	<input type="radio"/> On <input checked="" type="radio"/> Off
Bucket for Logs	<input type="text"/>
Log Prefix	<input type="text"/>

> 로그 파일 S3저장 옵션

http,host==dvmynol1bbflq.cloudfront.net						
	Time	Source	Destination	Protocol	Length	Info
5493	25.290977	192.168.59.32	54.192.70.195	HTTP	335	GET / HTTP/1.1
5582	25.600543	192.168.59.32	54.192.70.195	HTTP	356	GET /style.css HTTP/1.1
5599	25.625041	192.168.59.32	54.192.70.195	HTTP	376	GET /script.js HTTP/1.1
6618	26.902044	192.168.59.32	54.192.70.195	HTTP	291	GET /favicon.ico HTTP/1.1



> cloudfront의 엔드포인트로 접속

Edit Geo-Restrictions

Geo-Restriction Settings

Enable Geo-Restriction ☒ Yes ☐ No ?

Restriction Type ☒ Whitelist ☐ Blacklist ?

Countries ?

AF -- AFGHANISTAN
AX -- ALAND ISLANDS
AL -- ALBANIA
DZ -- ALGERIA
AS -- AMERICAN SAMOA
AD -- ANDORRA

Add >>
<< Remove

403 ERROR

The request could not be satisfied.

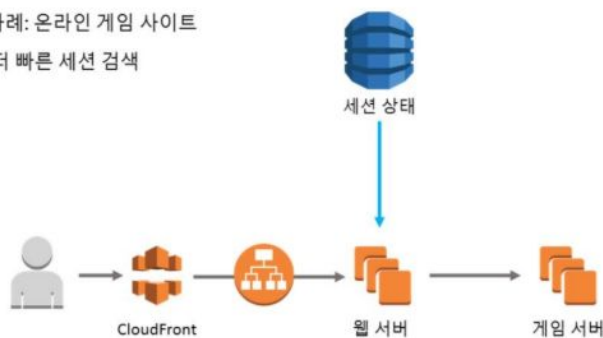
> Cloudfront - Restrictions 에서 해당 국가의 접근을 허용/차단 할 수 있음 (서울리전차단)

<웹 단위 캐싱데이터 (세션정보 캐싱)>

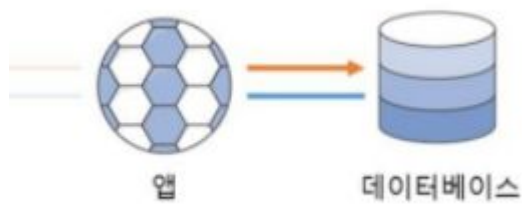
- 유동적인 서버의 성격으로 사용자 세션을 관리하는 특정 서버로 요청을 라우팅 할 수 있어야함
- 클라우드내에서는 ELB를 통한 분산이 되기 때문에 '고정세션' 기능을 통하여 쿠키를 저장한 서버에만 접속 할 수 있다 (권장 X)
- 클라우드에서는 세션에 대한 정보들을 캐싱 데이터를 따로 보관

사용 사례: 온라인 게임 사이트

문제: 더 빠른 세션 검색



<데이터베이스 단위 캐싱데이터>



- 고객에 대한 DB 접근시간을 줄일 때, 부하가 큰 대용량 요청으로 DB가 넘치는 경우, DB에 대한 비용을 줄이고 싶을 때 사용

*** Dynamo DB Accelerator(DAX)

- > Dynamo DB용 완전관리형 인 메모리 캐시
- > Dynamo DB 응답 시간을 밀리초에서 마이크로초로 단축시켜 탁월한 성능을 보장
- > 3노드의 DAX 클러스터로 시작하여 최대 10노드의 클러스터에 이르기까지 필요에 따라 용량을 확장 할 수 있음(확장성)
- > Dynamo DB와 마찬가지로 완전관리형 서비스로, 장애탐지/복구, 소프트웨어 패치와 같은 일반적인 관리 작업들을 자동으로 실행
- > Dynamo DB API와 호환이 되기 때문에 따로 코드를 변경할 필요가 없음
- > IAM 서비스로 각 사용자에게 고유한 보안 자격증명을 할당하고 CloudWatch, CloudTrail 등으로 모니터링/감사를 통해 보안을 강화

*** Amazon ElastiCache

- > 클라우드에서 인 메모리 캐시를 배포, 운영, 조정하는데 사용되는 웹 서비스
- > 웹 시스템의 속도/신뢰성을 위해 데이터베이스 쿼리 결과를 캐싱
- > Memcached 또는 Redis와 같은 인 메모리 캐시 엔진을 사용해 인 메모리 캐시 환경을 제공하는 서비스 (RDS의 캐시 버전)
- > 확장, 패치자동적용, 자동백업을 해주는 완전관리형 서비스
- > 가장 작은 블록인 '노드'의 단위로 구성
- > 노드는 다른 노드와 분리되어 존재, 혹은 다른 노드와의 일부관계(클러스터)에서도 존재가능
- > 각 노드에는 고유한 DNS 이름 및 포트가 존재

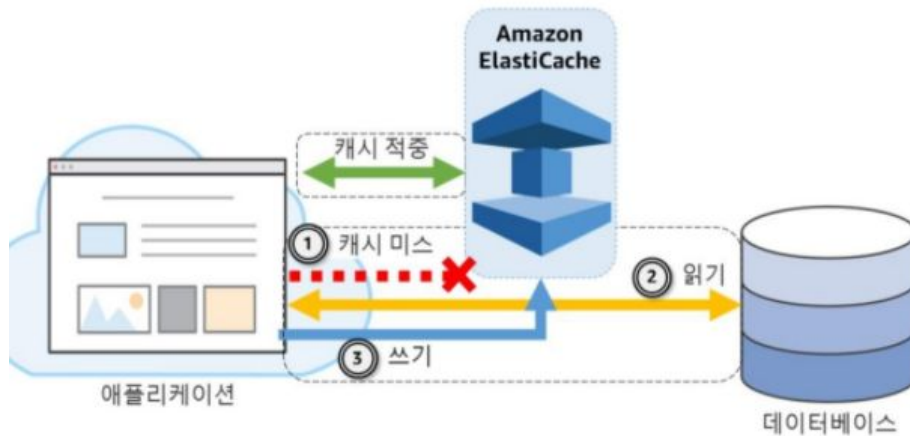
1. Memcached 엔진
 - 클러스터당 최대 20개의 노드까지 확장 가능
 - 데이터가 클러스터의 모든 노드에 분할되므로 수요가 증가하면 확장으로 고가용성 지원
2. Redis 엔진
 - 데이터 액세스 향상을 위해 최대 90개의 노드까지 확장 가능
 - 자동 장애조치가 적용된 다중AZ를 통해 고가용성 지원

	Memcached	Redis
DB 부하를 오프로드하는 단순 캐시	예	예
쓰기/스토리지를 위해 수평적으로 확장할 수 있는 기능	예	예
다중 스레드 성능	예	아니요
고급 데이터 유형	아니요	예
데이터 세트 정렬/순위 지정	아니요	예
Pub/Sub 기능	아니요	예
자동 장애 조치가 있는 다중 가용 영역	아니요	예
지속성	아니요	예

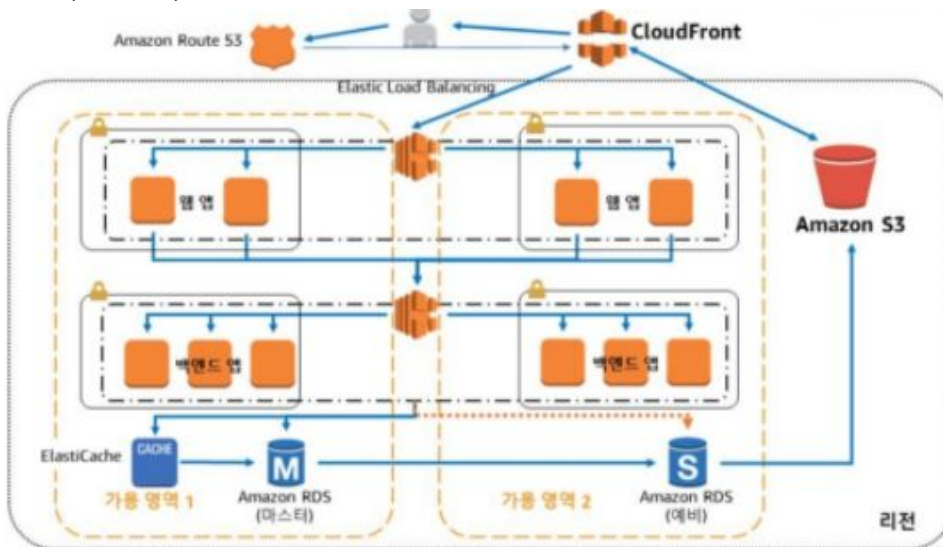
*** 레이지 로딩

> 필요할 때만 데이터를 캐시로 로드하는 캐싱 전략

> ElastiCache는 이 전략에서 애플리케이션과 데이터베이스 사이에 위치



*** 3 tier (웹-앱-DB)에서의 호스팅 과정



모듈12 마이크로 서비스 및 서버리스 아키텍처

- 마이크로 서비스 구축
- 컨테이너 서비스
- 서버리스 환경 구현

*** 마이크로 서비스 < > 모놀리식 애플리케이션

> 애플리케이션이 각 애플리케이션 프로세스를 서비스로 실행하는 독립적인 구성요소로 구축

> 아키텍처의 각 구성 요소 서비스는 다른 서비스의 기능에 영향을 미치지 않고 개발, 배포, 운용 및 조정이 가능(자율성)

> 각 서비스는 일련의 기능을 제공하도록 설계되어 집중적으로 특정 문제를 해결하고 구동한다(전문성)

모놀리식 포럼 애플리케이션



마이크로서비스 포럼 애플리케이션



*** 컨테이너 서비스

- > 컨테이너는 리소스 격리 프로세스에서 애플리케이션과 종속 항목을 실행하게 해주는 운영 시스템 가상화 방법
- > 애플리케이션의 코드, 구성 및 종속 항목을 사용이 간편한 빌딩 블록으로 손쉽게 패키징하여 환경일관성, 운영효율성, 개발자 생산성, 버전제어를 제공\
- > Kernel 기능이 지원되고 도커 데몬이 있는 어떤 Linux 시스템에서나 실행 가능(휴대성)

*** 컨테이너 이미지

- > 컨테이너가 사용 할 수 있는 파일 시스템의 스냅샷
- > 컨테이너 이미지는 공간 측면에서 가상 머신보다 훨씬 작게 작동
- > 컨테이너를 사용하면 고속이고 휴대 가능하며 인프라에 구애받지 않음

*** 가상머신 vs 컨테이너

- > 가상머신과 비교하여 컨테이너는 하이퍼바이저가 필요 없음
- > 컨테이너는 하이퍼바이저가 필요 없어 오버헤드가 거의 발생하지 않음

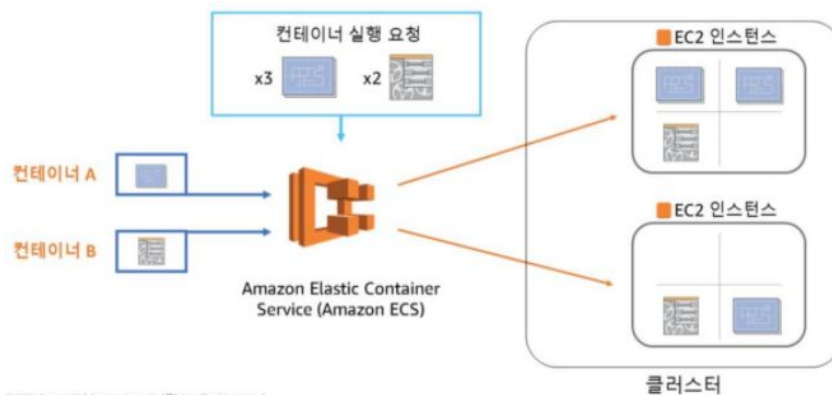


*** Amazon EC2 컨테이너 구조



<ECS> Elastic Container Service

- 도커 컨테이너를 지원하는 확장성과 성능이 뛰어난 컨테이너 관리 서비스
- Amazon EC2 인스턴스의 관리형 클러스터에서 애플리케이션을 손쉽게 실행 가능
- 최대 수천개의 인스턴스까지 확장 가능하고, 컨테이너 배포를 모니터링
- Fargate 시작 유형과 EC2 시작 유형의 두가지 모드가 존재
- Fargate 유형은 , 애플리케이션을 컨테이너로 패키징하고, CPU 및 메모리 요구 사항을 지정하고, IAM 정책을 정의만 하여 시작
- EC2 시작유형은 서버 수준의 좀 더 세분화된 제어를 할 수 있음



© 2019 Amazon Web Services, Inc. or its affiliates. All rights reserved.



*** AWS Fargate

- > 서버 또는 클러스터를 관리할 필요 없이 컨테이너를 실행 할 수 있게 해주는 Amazon ECS용 기능
- > 해당 기능을 사용하면 더 이상의 컨테이너를 실행하기 위해 가상 머신을 프로비저닝, 구성 및 조정할 필요가 없음

- > 서버유형을 선택하거나, 클러스터를 조정할 시점을 결정하거나, 클러스터 패킹을 최적화 할 필요가 없음
- > Fargate를 사용하면 애플리케이션을 실행하는 인프라의 관리 대신 애플리케이션 설계 및 구축에 집중 할 수 있음

**** 서버리스 컴퓨팅

- > 서버를 사용하지 않고 실행하는 애플리케이션과 서비스
- > 사용자가 서버를 프로비저닝, 확장, 관리할 필요가 없는 편리함
- > 개발자가 관리 및 운영에 신경을 쓰는 대신 핵심 제품에 집중 할 수 있다는 장점

<AWS Lambda>

- 서버를 프로비저닝하거나 관리할 필요 없이 코드를 실행할 수 있는 완전관리형 컴퓨팅 서비스
- 사용자가 지원하는 언어(Node.js,Java,C#,Python,Ruby)중 하나로 코드를 제공하기만 하면 고가용성 컴퓨팅 인프라에서 코드를 실행하고 서버 및 운영 체제 유지 관리, 용량 프로비저닝 자동 조정,코드 모니터링 및 로깅 등 모든 컴퓨팅 리소스 관리를 수행
- 핵심 구성 요소로 이벤트 소스와 Lambda함수로 이루어짐
- 계층을 사용하면 함수 개발자가 패키지, 바이너리, 런타임 및 Lambda 함수에 필요한 그 밖의 파일을 함수 코드와 별개의 구성 요소로 유지 할 수 있음
(ex. 파이썬으로 작성된 서버리스 애플리케이션 > PyMySQL 같은 패키지 사용)



*** Lambda 함수

- > 코드,구성,종속성
- > 구성

: 이벤트를 수신하는 핸들러, 사용자 대신 Lambda 함수를 실행하기 위한 IAM 역할, 할당할 컴퓨팅 리소스, 실행 제한 시간 등의 정보

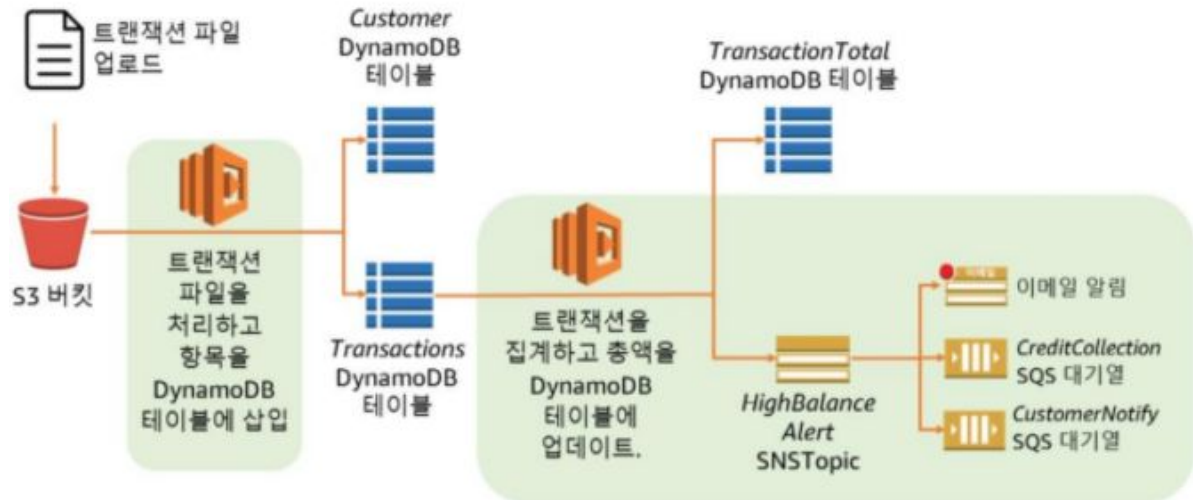
*** 이벤트 소스

- > ALB를 사용하여 HTTP/HTTPS를 통해 Lambda 함수에 트래픽을 전송할 수 있음
- > ALB가 콘텐츠 기반 라우팅이므로 ALB로 들어오는 요청의 호스트 또는 호스트 및 URL 경로를 기반으로 다른 Lambda함수에 트래픽을 전송 할 수도 있음

*** Lambda@Edge

- > CloudFront CDN에서 생성된 이벤트에 대한 응답으로 Lambda 함수를 실행하고 고가용성을 유지한 채로 최종 사용자에게 가장 가까운 엣지 로케이션으로 코드를 확장

*** Amazon S3와 Lambda 함수



<Amazon API Gateway>

- 애플리케이션의 '현관' 역할을 하는 API를 생성 할 수 있음
- EC2, Lambda, 웹 어플리케이션 에서 실행되는 워크로드를 처리 할 수 있음
- 엔드포인트 노출방지와 DDoS 및 명령어 주입 공격으로부터 보호

모듈13. RTO/RPO 및 백업 복구 설정

> 멀티 AZ, 볼륨백업, DB백업, 재해복구 계획 사전 성립

<RPO / RTO>

RPO (복구시점목표)

> 수용가능한 데이터 손실량을 시간으로 측정한 값

> 복구가 가능한 시간이 언제인가?

RTO (목표복구시점)

> 서비스가 중단후 복구하기까지 걸리는 시간을 의미

> 서비스의 중단 시간은 어떻게 되는가?



*** 재해복구 계획 인프라 구성

> 저장소 복제, 중복, 백업(스냅샷)

S3 복제

개요속성권한관리엑세스 지점

수명 주기복제분석지표

인벤토리

+ 규칙 추가우선 순위 편집편집삭제작업

이 버킷에 대해 생성한 복제 규칙이 없습니다.

교차 리전 복제

교차 리전 복제를 사용하면 서로 다른 AWS 리전에 있는 버킷 간에 객체를 자동 및 비동기식으로 복사할 수 있습니다. 객체 복제를 위해 구성된 버킷은 동일한 AWS 계정 또는 다른 계정에서 소유할 수 있습니다.

세부 정보

동일 리전 복제

동일 리전 복제를 사용하면 동일한 AWS 리전에 있는 버킷 간에 객체를 자동 및 비동기식으로 복사할 수 있습니다. 객체 복제를 위해 구성된 버킷은 동일한 AWS 계정 또는 다른 계정에서 소유할 수 있습니다.

세부 정보

볼륨 스냅샷 멀티리전 복제

스냅샷 생성작업

내 소유

삭제

볼륨 생성

Manage Fast Snapshot Restore

이미지 생성

복사

권한 수정

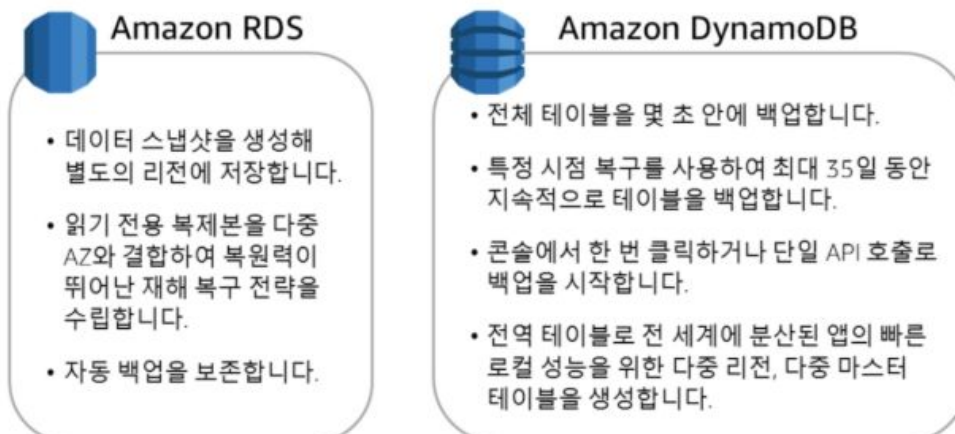
태그 추가/편집



- > 컴퓨팅 백업 조정은 쉬워야함
- > 네트워크 재해복구



- > 데이터베이스의 백업 및 중복
(RDS, Dynamo DB는 관리형 서비스이기 때문에 자동백업)
(Read Replica DB는 다른 리전 멀티 AZ 복제 가능, 마스터 DB 장애시 마스터 DB로 승격)



- > 자동화를 사용한 신속한 복구



AWS CloudFormation

템플릿을 사용하여 필요에 따라 리소스 모음을 신속하게 배포합니다.



AWS Elastic Beanstalk

단 몇 번의 클릭으로 전체 스택을 빠르게 재배포합니다.



AWS OpsWorks

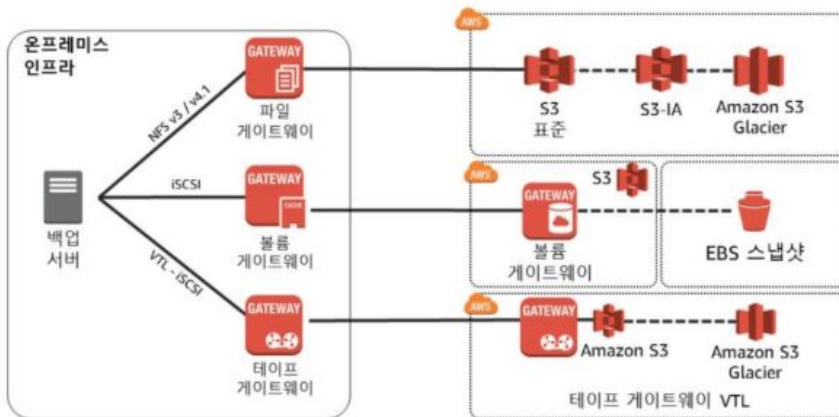
- 자동 호스트 교체
- 복구 단계에서 AWS CloudFormation과 결합합니다.
- 정의된 RTO를 지원하는 새로운 스택을 제공합니다.

<복구전략>

- 백업을 생활화, 복구는 최대한 신속

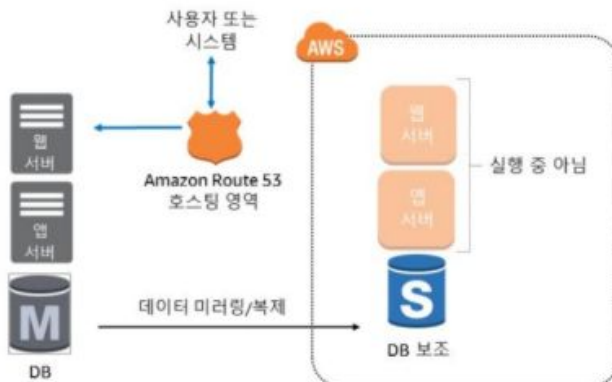
*** Storage GW (파일/블록/테이프 GW)

> 온프레미스 환경에서 파일,블록,테이프 형식의 데이터를 클라우드 환경의 저장소로 백업 할 때 사용하는 통로



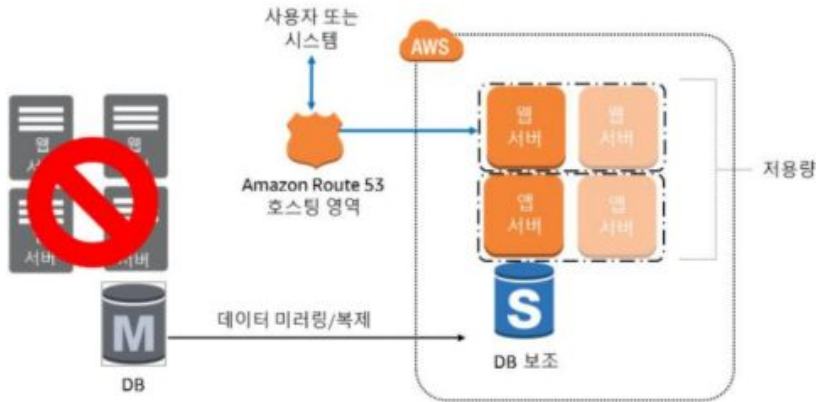
*** 파일럿 라이트 (Cold standby)

- > 인스턴스, 네트워크 등 똑같은 환경을 구축하여 중지 상태로 유지하는 개념
- > DB 인스턴스는 Slave DB에 동기화 해야되기 때문에 전원을 켜 상태로 유지
- > 백업 방법보다 비용 효율적이며, 복구가 신속하다



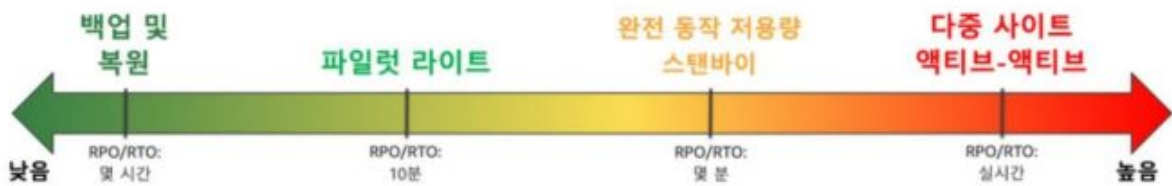
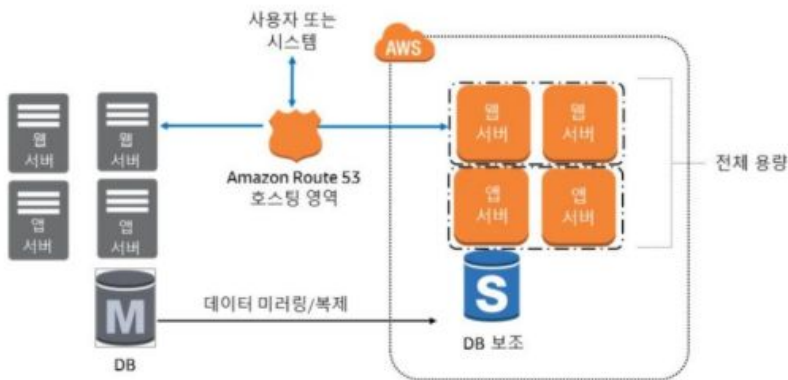
*** 완전동작 저용량(WARM) 스탠바이

- > 파일럿 라이트와 비슷한 개념으로, 중지 상태가 아닌 규모가 좀 작은 경우로 구동시켜 유지하는 방법



*** 다중사이트 액티브 (Hot Standby)

- > Main 사이트와 똑같은 구성을 항상 가동시키는 방법 (비용이 비쌈)
- > Auto Scaling 시간이 없어 즉각적인 장애조치



<ul style="list-style-type: none"> 우선순위가 낮은 사용 사례 솔루션: Amazon S3, Storage Gateway 	<ul style="list-style-type: none"> 더 낮은 RTO와 RPO 요구 사항 충족 핵심 서비스 DR 이벤트에 대한 응답으로 AWS 리소스 확장 	<ul style="list-style-type: none"> RTO 및 RPO를 몇 분 만에 필요로 하는 솔루션 비즈니스 크리티컬 서비스 	<ul style="list-style-type: none"> AWS의 환경을 실행 중인 복제본으로 자동 장애 조치
비용: \$	비용: \$\$	비용: \$\$\$	비용: \$\$\$\$