AWS Technical Essentials

모듈 1. Amazon Web Service 소개

aws란?

- 짧게 “클라우드 컴퓨팅 서비스 제공” 으로 보면 된다.
- 따라서 클라우드 컴퓨팅의 이점은 다음과 같다.

1. 종량 과금제
2. 거대한 규모의 경제로 얻게 되는 이점
3. 용량 추정 불필요
4. 속도 및 대응력 향상
5. 비용 절감 실현
6. 몇분 만에 전 세계에 배포

중복되는 데이터 센터들의 클러스터 집합을 가용영역(Availability Zone)이라 한다.
즉 문제 발생 시 AZ 전체에 영향이 가지 않는한 안전.
하지만 AZ 전체에 영향이 갈 수 있으므로, AZ 또한 **중복**되게 클러스터로 묶는다.
그것을 Region, 지역, 리소스의 위치라고 한다.

이 데이터센터, AZ들을 중복되고 중첩되게 함으로써 고가용성과 복원력을 유지한다.

Region 선택 고려 사항

1. 규정 사항 – 데이터가 지역의 요인을 받는 것인지...
2. 대기 시간(Latency) – 데이터를 사용할 앱과 가까이에 하는 것이 좋겠죠?
3. 가격 – Region 마다 요금이 다름 (세금 때문에)
4. Serviec availability – 어떤 지역은 일부 서비스를 사용 못할 수 있음

AWS와 상호작용

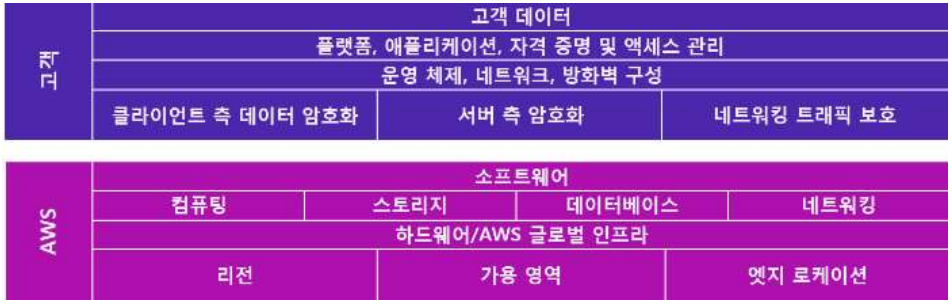
- 클라우드 특성상 서비스(인프라,플래폼 등)을 물리적이 아닌 가상적으로 제공하므로 서비스들을 관리하고 조작하는 것이 필요함.
- **AWS에서 수행하는 모든 작업은 인증 및 권한이 부여된 API 호출**이다.

즉 API를 호출하는 방법에는 3가지 방법이 있음

1. AWS Management Console
2. AWS Command Line Interface(AWS CLI)
3. AWS Software Development Kit (SDK)

보안 및 AWS 공동 책임 모델

- AWS 클라우드로 작업할 때 보안 및 규정 준수를 관리하는 것은 AWS와 고객의 공동 책임이다.
- 클라우드 자체의 보안(AWS) , 클라워드 내부의 보안(사용자) 가 각각 책임진다.



※컨테이너 서비스는 일부 사용자 보안 책임을 위한 것들이 컨테이너에 들어간 것이 있어 보다 사용자의 책임을 조금 덜 수도 있다.

AWS 루트 사용자 보호

- 루트 사용자는 말그대로 뭐든지 가능하기 때문에 보호하는 것이 필요하다.
- 루트 사용자는 두 가지 자격 증명 세트가 연결되어 있음.

1. 계정을 만드는데 사용되는 이메일 주소와 암호 –〉 AWS 관리 콘솔 접근
2. 액세스 키라고 하며 AWS CLI나 AWS API에서 프로그래밍 방식으로 요청

- 그래서 MFA 사용한다. : 루트 사용자 모바일에서 임시로 발급받는 비밀번호 등
- MFA는

1.사용자 이름 및 암호 또는 핀 번호와 같이 사용자가 알고 있는 것
2.하드웨어 디바이스 또는 모바일 앱의 일회용 암호와 같이 사용자가 가지고 있는 것
3. 지문 또는 얼굴 스캐닝 기술과 같은 사용자 자체

등의 조합을 권장한다.

- 또한 일반적인 작업에서는 IAM을 사용하여 권한을 줄이는 것이 좋다.
- 지원되는 MFA 디바이스 종류

1. 가상 MFA : 일회성 암호를 제공하는 SW 앱
2. 하드웨어 : 일회성 6자리 숫자 코드를 생성하는 하드웨어 디바이스 –〉 키 포브, 디스플레이 카드
3. U2F : USB로 연결하여 인증하는 하드웨어 디바이스

AWS Identity and Access Management

- Aws의 각 서비스는 리소스가 같은 계정에 있더라도 다른 서비스와 통신하려면 **자격 증명 및 액세스 관리**가 필요하다.
- 루트 사용자는 이미 모든 권한을 갖고 있어 그냥 요청만 하면 되지만, 이는 바람직 않기 때문에 IAM 계정을 만드는 것이 좋다.
- IAM은 권한 부여를 통해 API 호출을 허용/거부를 결정 짓는다.
- IAM 정책은 JSON 기반으로 구성되어 있다.
- Condition key를 통해 조건을 정해줄수도 있음

AWS에서의 역할 기반 액세스

- 일반적인 IAM 계정 뿐만 아니라 IAM Role을 통해 액세스를 허용할수도 있음
- 어떤 앱에 하드 코드 없이 액세스하려고 할 때 이미 권한 부여된 역할로서 인증받는다.
- IAM roles는 사용자 이름과 암호 같은 로그인 자격 증명이 없음
- 요청에 서명에 사용되는 자격증명은 임시 자격 증명이며 자동 교체됨
- IAM은 개개인별로 부여하는 계정이기 때문에 많은 사람들이 동일한 권한을 가진 역할을 수많은 IAM에 부여하는 것은 매우 번거로운 일이다. 따라서 IdP(Identity Provider)를 통해 직원 자격 증명 정보를 관리하는 것이 좋음.
- IdP는 AWS Single Sign-On과 같은 AWS 서비스이든 서드 파티 자격 증명 공급자든 조직의 모든 자격 증명에 대해 신뢰할 수 있는 단일 소스를 제공한다.

정리 - IAM 모범 고려사항

1. AWS 루트 사용자 잠그기
2. 최소 권한 원칙 따름
3. 적절하게 IAM 사용
4. 가능한 경우 IAM 역할 사용
5. 자격 증명 공급자 사용을 고려
6. AWS Single Sign-On을 고려

서버

서버는 HTTP 요청을 처리하고 응답을 내보내는 역할.

WINDOWS 서버 : 인터넷 정보 서비스(IIS)

리눅스 서버 : Apache HTTP 웹 서버, Ngnix, Apache Tomcat

Amazon EC2(Elastice Compute Cloud)

컴퓨팅을 제공하는 서비스

- 우리는 EC2를 사용하기 위해서는 먼저 Amazon Machine Image를 통해 스토리지 매핑, 아키텍처 유형, 추가 소프트웨어를 설정해줘야 그에 맞는 EC2를 생성할 수 있다.
- EC2를 한번 생성했다고 그에 대한 설정은 고정되는 것이 아니라, 계속 확장 가능하고 변경가능하다.
- AMI는 직접 설정하지 않아도 마켓 플레이스에 다른 사람이 이미 만들어둔 환경, 혹은 내가 전에 사용했던 환경들을 간단히 쉽게 불러와서 사용할 수 있다.
- EC2는 필요에 따라 쉽게 생성하고 쉽게 중지 및 삭제할 수 있어서 탄력적으로 사용할 수 있다.
- 다양한 옵션들이 있지만 **중지-최대 절전 모드**랑 종료만 잊지말자.
- 종료하면 완전한 인스턴스의 삭제를 의미한다. 반드시 이를 이용한 데이터는 백업이 반드시 필요하다. 삭제는 분명히 장단점이 있다.
- 실행 및 중지 상태에서 비용을 청구한다.
- 비용 절감을 위해서는 예약 인스터느나 스팟 인스턴스를 사용하자.
- 인스턴스는 **인스턴스 패밀리** 및 **인스턴스 크기**로 명시되어 있다.

컨테이너 서비스

2종류 서비스

1. Elastic Container Service
2. Elastic Kubernectics Service

- 컨테이너 오케스트르레이션 도구는 컨테이너 안의 많은 인스턴스들을 관리하기 위한 도구이다. 근데 이는 ECS에서 쓰고
- EKS에서는 비슷하지만 다른이름의 도구를 사용한다.
- 컨테이너는 가상서버보다 작동시간이 짧다.
- 컨테이너와 가상서버는 어떻게 보면 비슷한 개념이지만 관리해야할 자원은 더 적다. (물론 가상머신처럼 일부를 내가 직접 관리할 수 있다)

서버리스

- 솔루션을 호스팅하는 기본 인프라 또는 인스턴스를 보거나 액세스 할 수 없다.
- 대신 **프로비저닝**, **크기조정, 내결함성** 과 같은 모든 기본환경을 알아서 해주는거임
- 즉 사용자는 자신이 구동할 앱에만 집중하면 되고, 나머지 것들은 사용자에게서 분리된다.
- 어떻게 보면 책임 분리 원칙에서 말한 내용임.
- AWS Fargate는 ECS/EKS를 실행할 수 있는 플랫폼이다.

AWS 람다

- 하나의 서버리스 컴퓨팅 플랫폼
- 구축된 코드를 패키징하여 람다에 업로드하면 람다 함수를 생성할 수 있는데, 이를 트리거에 대해 응답하도록 한다.
- 트리거는 무수히 많을 수 있음. 또한 다른 서비스에서의 요청을 트리거할 수도 있음.
- 이 람다함수가 시작되면 환경은 알아서 관리해줌.
- 오직 런타임이 15분 미만에게만 실행되게 설계됨
- 이것도 오직 실행된 코드에서만 비용 청구

파이썬의 일반적인 구문

```
def handler_name(event, context):
...
return some_value
```

네트워킹

- 네트워킹에 대한 기본 지식은 알고 있으므로 스킵. 중요한 부분 정도만.
- CIDR 표기법(클래스 없는 도메인 간 라우팅)을 알아보자

이는 내가 알던 서브넷으로 자르는 것으로 IP주소를 늘릴 때 사용하던 것이다.

Amazon Virtual Private Cloud

- VPC는 데이터 센터를 둘러싼 하나의 벽이라 생각하면 된다.
- 즉 애플리케이션 및 리소스가 격리되는 공간을 만들어주는 역할을 한다.
- VPC를 생성할 때 두 개를 선택해야한다.

1. Region 2. IP range (CIDR에 따르는 서브넷 자르기)

- VPC를 만들었으면 각 분리한 서브넷에 따라 인스턴스를 배치한다.
- 이는 각 서브넷 마다 설정을 두어 다른 정책을 가질 수 있다. (Public/Private)
- 그보다 먼저 서브넷을 나눴지만 이를 지정하는 것도 필요한데, VPC, AZ, IP range를 통해 서브넷을 지정할 수 있다.
- VPC는 리소스를 격리하는 벽이라 했는데, 이를 외부와 연결하려면 인터넷 게이트웨이가 필요하다. 따라서 이를 생성하고 연결해야 작동할 수 있다.
- 근데 일반적인 게이트웨이는 개방되어 있으므로 Private 서비넷과의 연결은 Virtual Private Gateway를 생성해야한다. 이는 VPN 연결을 도와준다.
- 이 VPG는 양쪽에 생성해야되는데, 고객 측에 설치된 VPG는 물리적 디바이스 또는 소프트웨어 애플리케이션이 된다.
- 그리고 VPC는 한 리전에 생성되는 데 서브넷은 한 AZ에 생성된다. 따라서 다른 AZ에도 고가양성을 위해 복제된다.

Amazon VPC 라우팅

- 만약 VPC를 앞에 말한 생성도 완료를 했다고 가정하자.
- 한 사용자가 한 지역에 특정 VPC를 인터넷 게이트로 접근을 했다면, 그 다음부터는 어떤 AZ에 들어가야 할지 고려 해야 한다.
- 이는 고가용성을 위해 한 리전의 여러 AZ에 리소스를 복제해놧기 때문이다.
- 따라서 사용자는 어디로 갈지 모르기 땨문에 Route Table을 작성해야한다.
- 이 라우팅 테이블을 어떤 서브넷에 연결 가능하게 할건지 설정하여 세분화 할 수 있다.
- 기본적으로 서브넷 자체는 퍼블릭 접근을 제공하지 않는다.

Amazon VPC 보안

- 기본적으로 VPC를 생성(벽을 치더라도) 바깥은 개방된 인터넷과 연결되있는 것이다.
- 그래서 중요한 것은 ACL(액세스 제어 목록)과 보안그룹이다

○ ACL(액세스 제어 목록)

- ACL은 서브넷 수준의 방화벽이라고 보면 된다. 각 서브넷에 허용할 트래픽 주소 및 포트(프로토콜)를 지정하는 것이다.
- 이 ACL에서 접근하는 것(**인바운드**)를 허용해도 바같으로 나가는 (**아웃바운드**) 규칙을 설정해야 비로소 보안적인 통신을 할 수 있는 것이다.

○보안 그룹

- 보안 그룹은 EC2 수준에서의 방화벽이다.
- 그래서 필수사항은 아니지만 EC2를 생성하면 보안그룹 해주는 것이 권장된다.
- 기본적으로 보안그룹은 **인바운드는 모두 차단**하고, **아웃바운드는 모두 허용**한다.
- 보안 그룹은 상태 유지 리소스로 구성된다.???

※조금 중요한 차이라면 보안 그룹은 인바운드가 기본적으로 다 차단이고, ACL은 인바운드가 기본적으로 다 허용이다.

스토리지 유형

1. 블록 스토리지 : 고정 크기의 데이터 청크로 분할되어 파일을 저장 - 변경은 오직 일부 데이터 청크만 바꾸기 땨문에 오버헤드가 적음
2. 파일 스토리지 : 전체를 바꿔야됨 -〉 오버헤드가 큼
3. 객체 스토리지 : 이 또한 전체를 바꿔야 함. 또한 파일과 다르게 메타데이터도 포함

Elastic Block Store

인스턴스 스토어 EC2 내에 저장할 수 있는 공간이다.

- 인스턴스 스토어는 EC2 내에 있어 엄청 빠르지만, EC2 수명에 따라 이 수명도 결정됨
- 이 EC2 내의 데이터를 영구적으로 보관하고 싶으면 EBS 볼륨을 연결하여 그곳에 저장
- 외장하드처럼 생각하면 쉬운데 기본 1대1이지만, 다중 연결도 지원함\
- EBS 볼륨에 다양한 옵션들이 있음

-〉

EBS 프로비저닝된 IOPS SSD	EBS 범용 SSD
처리량 최적화 SSD	콜드 HDD

- 장점 : 고가용성, 데이터 지속성, 데이터 암호화, 유연성, 백업
- Amazon EBS 스냅샷을 생성하여 백업 옵션을 추가할 수도 있음

Elastic Simple Storage Service

- 보통 다중 연결, 많은 곳에서 액세스를 요구하는 경우, 즉 온라인 스토리지는 S3가 좋음
- S3는 플랫 구조를 사용하여 필요 리소스만 제공
- 최고의 가용성, 내구성
- 버킷 정책은 JSON 구조

http://doc.s3.amazonaws.com/2006-03-01/AmazonS3.html

버킷 — 객체/키

- S3에도 여러 가지 옵션들이 있으며, 객체 수명 주기 관리에 따라 티어가 변환되기도 함

AWS 기반 데이터베이스

- 사용자의 부담을 덜 수 있다.
- 즉 비곤리형 서비스로서 제공한다.

Amazon RDS

- 관계형 데이터베이스
- 다중 AZ를 통한 중복성이 제공된다.
- 이 중복성을 위해 각 DB는 DNS를 통해 통신한다.

DynamoDB

- 비관계형 데이터를 저장하는 데 좋은 데이터 베이스
- 완전관리형 NoSQL 데이터베이스 서비스
- 원활한 확장성과 함께 빠르고 예측 가능한 성능 제공
- 속성 추가 제거가 용이하며 변형이 쉽게 됨
- 쿼리의 유형도 더 단순한 경향이 있음
- 응답시간도 존나 빠름

AWS 데이터베이스 서비스 비교

데이터베이스 유형	사용 사례	AWS 서비스
관계형	기존 애플리케이션, ERP, CRM, 전자 상거래	Amazon RDS, Amazon Aurora, Amazon Redshift
키-값	높은 트래픽의 웹 앱, 전자 상거래 시스템, 게임 애플리케이션	Amazon DynamoDB
인 메모리	캐싱, 세션 관리, 게임 순위표, 지리 공간 애플리케이션	Amazon ElastiCache for Memcached, Amazon ElastiCache for Redis
문서	콘텐츠 관리, 카탈로그, 사용자 프로필	Amazon DocumentDB(MongoDB 호환 가능)
와이드 컬럼	장비 관리, 플릿 관리 및 라우팅 최적화에 사용하는 대규모 산업용 앱	Amazon Keyspaces(Apache Cassandra용)
그래프	사기 탐지, 소셜 네트워킹, 추천 엔진	Amazon Neptune
시계열	IoT 애플리케이션, DevOps, 산업용 텔레메트리	Amazon Timestream
원장	레코드 시스템, 공급망, 등록, 은행 거래	Amazon QLDB

모니터링

- 리소스의 운영 상태 및 사용량에 대한 데이터를 수집하고 분석하는 방법이 필요
- 지표를 시간으로 나누면 통계가 된다
- 데이터를 수집, 분석 및 사용하여 의사 결정을 내리거나 IT 리소스 및 시스템에 대한 질문에 답하는 행위를 모니터링이라 한다.
- 지표의 유형은 서비스 종류에 따라 다 달라진다.
- CLoudWatch에서 모니터링 가능
- 모니터링 이점

1. 최종 사용자가 운영 문제를 인식하기 전에 사전 대응할 수 있습니다.
2. 리소스의 성능 및 안정성을 개선합니다.
3. 보안 위협 및 이벤트를 인식합니다.
4. 비즈니스를 위해 데이터 중심의 의사 결정을 수립합니다.
5. 보다 비용 효율적인 솔루션을 구축합니다.

- CloudWatch에서는 가시성도 제공한다.

Amazon CloudWatch

- AWS 계정만 있으면 시작 가능
- 많은 AWS 서비스는 5분 간격으로 지표당 하나의 데이터 포인트의 속도로 CloudWatch에 지표를 무료로 자동 전송
- 1분 간격으로 세부적인 모니터링을 하고싶으면 비용 부과
- 기본적인 지표는 네임스페이스에 따르게 되고, 타임 스탬프가 있음.
- 사용자가 지정하여 지표를 기록할 수도 있음
- 대시보드를 텅해 데잍를 시각화하고 검토할 수도 있음
- 또한 로그를 분석할 수도 있음
- 로그에 대한 용어

1. 로그 이벤트: 로그 이벤트는 모니터링되는 애플리케이션 또는 리소스가 기록한 활동의 레코드이며 타임스탬프와 이벤트 메시지가 있습니다.
2. 로그 스트림: 로그 이벤트는 로그 스트림으로 그룹화됩니다. 로그 스트림은 모두 모니터링되는 동일한 리소스에 속하는 로그 이벤트의 시퀀스입니다. 예를 들어 EC2 인스턴스에 대한 로그는 인사이트를 필터링하거나 쿼리할 수 있는 로그 스트림으로 그룹화됩니다.
3. 로그 그룹: 그런 다음 로그 스트림이 로그 그룹으로 구성됩니다. 로그 그룹은 동일한 보존 및 권한 설정을 공유하는 로그 스트림으로 구성됩니다

- 또한 CloudWatch는 경보를 생성하여 지표의 변화에 따라 경보/작업을 트리거할 수 있음
- 경보에는 OK, ALARM, INSUFFICIENT_DATA 의 총 3가지 상태가 있음

솔루션 최적화

- 가용성을 높이려면 중복성이 필요. 가격에 따라 잘 정하자
- 고가용성 유형

1. **액티브-패시브** : 한 번에 두 인스턴스 중 하나만 사용가능한 형태. 이 방법의 한 가지 장점은 클라이언트의 세션에 대한 데이터가 서버에 저장되는 상태 유지 애플리케이션의 경우 고객이 항상 세션이 저장된 서버로 전송되기 때문에 문제가 발생하지 않는다는 것
2. **액티브 - 액티브 :** 액티브-패시브의 단점이자 액티브-액티브 시스템의 장점은 **확장성.** 두 서버를 모두 사용할 수 있게 하면 두 번째 서버가 애플리케이션의 로드를 일부 처리하므로 전체 시스템이 더 많은 로드를 수행할 수 있습니다. 그러나 애플리케이션이 상태 유지인 경우 두 서버 모두에서 고객의 세션을 사용할 수 없는 경우 문제가 발생할 수 있습니다. 무상태 애플리케이션은 액티브-액티브 시스템에서 더 잘 작동합니다.

Elastic Load Balancing

- 리소스 집합에 작업 및 트래픽을 분산하는 프로세스
- ELB는 직접적인 트래픽 경로에 존재함. - 연결로드 밸런서
- 로드 밸런스는 정확히 3가지로 분류될 수 있음

1. Applicataion Load Balancer - 7 계층, 응용 계층 수준의 트래픽을 분산
2. Network Load Balancer - 4계층, 네트워크 계층 수준. UDP, TCP 같은 트래픽 분산
3. Gateway Load Balancer - 3 + 4 계층 수준에서 작동하는 트래픽을 서드 파티 레이어에 분산하기 위한 밸런서

- ALB 의 구성요소에는 다음과 같은 것이 있다.

○Listener : 요청을 확인하는 역할 - 포트&프로토콜 지정
○Target Group(대상 그룹) : 백엔드 리소스를 그룹화해서 상태가 정상인지 확인하게 됨.
- IP주소, EC2 인스턴스 등 트래픽을 전송하려는 것이 백엔드 유형에 해당된다.
○Rules(규칙) : 요청이 대상으로 라우팅되는 방식을 정의

로드 밸런서도 어디서 분산하냐에 따라 크게 두 타입으로 나눔

1. 연결 로드 밸런서 : 외부와 내부 로드 밸런서라 생각하면됨
2. 내부 로드 밸런스 : 프라이빗 IP에서 프라이빗 IP의 트래픽을 분산

※리스너를 설정할 때 포트를 설정한다 했는데, 또한 VPC도 설정한다. 이 두 허용 규칙이 상충되게 하면 안될 것 같다.

ALB 특징

1. 요청 데이터를 기반으로 트래픽을 라우팅
2. 클라이언트에 직접 응답을 보냄
3. TLS 오프로딩을 사용
4. 사용자를 인증
5. 트래픽을 보호
6. 라운드 로빈 라우팅 알고리즘을 사용
7. 최소 미해결 요청 라우팅 알고리즘을 사용
8. 스티키 세션을 사용

NLB 특징

1. 네트워크 로드 밸런서는 TCP, UDP 및 TLS 프로토콜을 지원
2. 해시 라우팅 알고리즘을 사용
3. 스티키 세션이 있음
4. TLS 오프로딩을 지원
5. 초당 수백만 건의 요청을 처리
6. 고정 및 탄력적 IP 주소를 지원
7. 소스 IP 주소를 보존

특징	Application Load Balancer	네트워크 로드 밸런서
프로토콜	HTTP, HTTPS	TCP, UDP, TLS
Connection Draining(등록 취소 지연)	√	
IP 주소를 대상으로 사용	√	√
고정 IP 및 탄력적 IP 주소		√
소스 IP 주소 유지		√
소스 IP 주소, 경로, 호스트, HTTP 헤더, HTTP 메서드 및 쿼리 문자열을 기반으로 하는 라우팅	√	
리디렉션	√	
고정 응답	√	
사용자 인증	√	

Amazon EC2 Auto Scailing

- Amazon EC2 Auto Scaling 서비스는 Amazon CloudWatch의 지표를 기반으로 EC2 인스턴스를 자동으로 생성 및 제거하여 이 작업을 처리
- ELB 서비스는 EC2 Auto Scaling과 원활하게 통합됩니다
- 애플리케이션이 사용하는 용량과 정확히 일치하게 용량을 조정하면 애플리케이션에 필요한 만큼만 비용을 지불할 수 있음
- EC2 Auto Scaling의 3가지 주요 구성 요소

1. 시작 템플릿 또는 구성: 자동으로 크기를 조정할 리소스는 무엇입니까?
2. EC2 Auto Scaling 그룹: 리소스를 어디에 배포해야 합니까?
3. 크기 조정 정책: 언제 리소스를 추가 또는 제거해야 합니까?

- 기본적으로 Amazon EC2 Auto Scaling는 스케일 아웃/인을 집중적으로 지원
- 스케일 업은 약간의 제한이 있음