# 클라우드 컴퓨팅 및 AWS 기초

## Cloud ML Ch01

# 학습 목표

## 이번 시간의 학습 목표

이번 시간을 통해 우리는 다음 개념들을 이해하고 설명하는 것을 목표로 합니다.

- 클라우드 컴퓨팅의 **핵심 개념**과 전통적 IT 인프라 대비 **주요 이점**을 설명할 수 있습니다.
- 클라우드 서비스 모델인 **IaaS, PaaS, SaaS**의 차이점을 비교하고 각각의 예시를 들 수 있습니다.
- AWS의 글로벌 인프라 구성 요소(**리전, 가용 영역** 등)의 관계를 이해하고 설명할 수 있습니다.
- 가상 서버(**EC2**)의 핵심 개념과 주요 구성 요소를 이해합니다.
- AWS의 접근 제어 서비스인 **IAM**의 역할과 기본 구성(사용자, 그룹, 역할, 정책)을 설명할 수 있습니다.
- 서버, 컨테이너, 서버리스 등 **컴퓨팅 모델의 진화** 과정을 이해합니다.

# 클라우드 컴퓨팅 개요

## 클라우드 컴퓨팅이란?

인터넷을 통해 서버, 스토리지, 데이터베이스, 네트워킹, 소프트웨어 등 컴퓨팅 리소스를 제공하는 서비스입니다.

## 전통적 IT 인프라 vs 클라우드 컴퓨팅

### 전통적 IT 인프라

- **구축**: 자체 서버실, 하드웨어 직접 구매
- **비용**: 높은 초기 투자 및 유지보수 비용
- **관리**: 모든 것을 직접 관리 (전력, 냉각, 보안)
- **확장성**: 물리적 증설에 오랜 시간 소요

### 클라우드 컴퓨팅

- **구축**: 클라우드 제공업체의 리소스 임대
- **비용**: 사용한 만큼만 지불 (Pay-as-you-go)
- **관리**: 제공업체가 인프라 관리 대행
- **확장성**: 필요에 따라 즉시 확장/축소 가능

# 클라우드 컴퓨팅의 주요 이점

## 비즈니스 성장을 가속화하는 6가지 핵심 이점

### 비용 절감

높은 초기 하드웨어 투자 비용을 사용한 만큼 지불하는 운영 비용으로 전환하여 총 소유 비용(TCO)을 절감합니다.

### 민첩성 및 속도

몇 번의 클릭만으로 새로운 서비스를 즉시 시작하고 배포하여 개발 및 혁신 주기를 단축시킵니다.

### 탄력적 확장성

비즈니스 요구에 따라 리소스를 몇 분 만에 전 세계적으로 확장하거나 축소하여 낭비를 줄이고 기회를 포착합니다.

### 유연성 및 생산성

인프라 관리 대신 비즈니스 핵심 가치에 집중할 수 있으며, 다양한 최신 기술을 쉽게 도입할 수 있습니다.

### 신뢰성 및 안정성

데이터 백업, 재해 복구, 다중 데이터센터 운영을 통해 서비스의 안정성과 고가용성을 쉽게 확보할 수 있습니다.

### 글로벌 규모의 성능

전 세계에 분산된 데이터 센터를 활용하여 사용자에게 더 가까운 곳에서 빠른 속도로 서비스를 제공합니다.

# 클라우드 서비스 모델: IaaS, PaaS, SaaS

## 누가, 무엇을 관리하는가에 따른 분류

서비스 모델은 클라우드 제공업체가 관리하는 영역과 사용자가 직접 관리해야 하는 영역의 범위를 정의합니다.

### IaaS (Infrastructure)

**인프라**를 빌려 쓰는 서비스. 가장 높은 수준의 제어 권한을 제공합니다.

- **사용자 관리**: OS, Middleware, App, Data
- **클라우드 관리**: 서버, 스토리지, 네트워크, 가상화
- **예시**: **AWS EC2, EBS, S3, VPC**, Microsoft Azure Virtual Machines, Google Cloud Compute Engine

### PaaS (Platform)

**플랫폼**을 빌려 쓰는 서비스. 개발자는 코드에만 집중할 수 있습니다.

- **사용자 관리**: App, Data
- **클라우드 관리**: OS, Middleware, 서버, ...
- **예시**: **Google App Engine, Cloud Run**, Azure App Service, **Heroku**

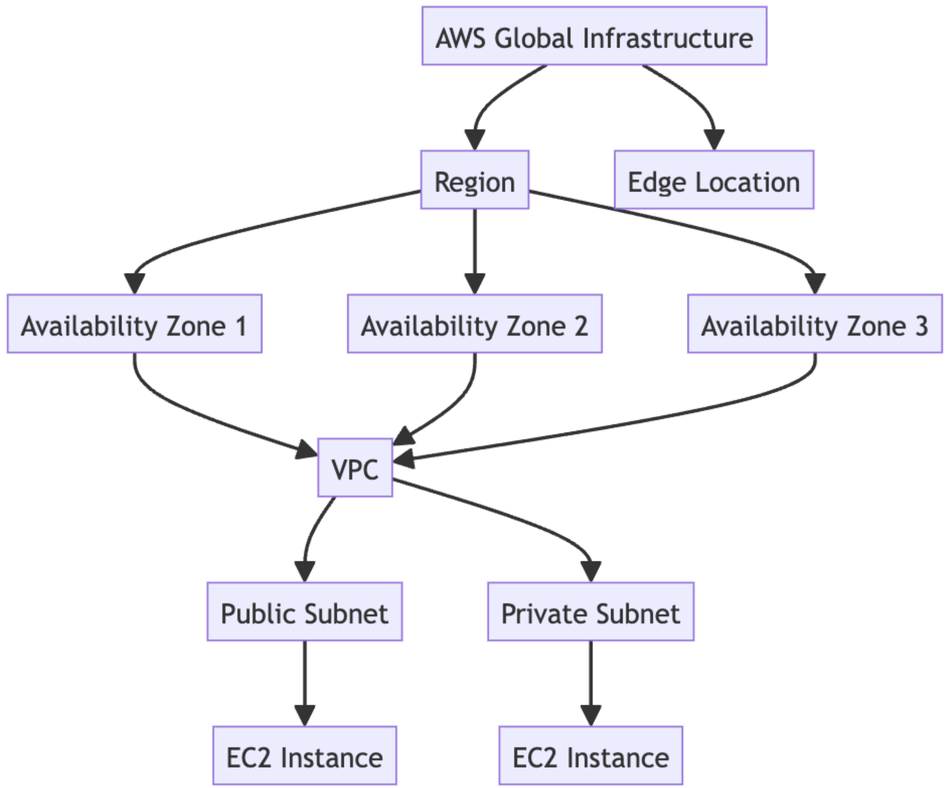
### SaaS (Software)

**소프트웨어**를 구독하는 서비스. 사용자는 즉시 앱을 사용합니다.

- **사용자 관리**: 없음
- **클라우드 관리**: 모든 것
- **예시**: **Salesforce**, Google Workspace, **Slack, Zoom, Dropbox**

# AWS 글로벌 인프라 구조

## 전 세계에 걸쳐 구축된 AWS의 물리적 인프라




- **리전 (Region)**: AWS 서비스가 제공되는 독립적인 지리적 영역 (예: 서울, 도쿄). 각 리전은 완전히 격리되어 있습니다.
- **가용 영역 (Availability Zone, AZ)**: 리전 내에 위치한 하나 이상의 데이터 센터. 각 AZ는 물리적으로 분리되어 고가용성을 보장합니다.
- **엣지 로케이션 (Edge Location)**: CloudFront CDN의 캐시 서버가 위치한 전 세계 분산 지점입니다.

# EC2 인스턴스와 핵심 개념

## EC2 (Elastic Compute Cloud): 클라우드의 가상 서버

AWS에서 "서버를 하나 띄웠다"는 것은 **EC2 인스턴스**를 생성했다는 의미입니다. 필요할 때 빠르게 생성하고, 필요 없으면 중지/종료하여 비용을 효율적으로 관리할 수 있습니다.

| 개념 | 설명 | 비유 |
| --- | --- | --- |
| **AMI (Amazon Machine Image)** | 인스턴스 시작에 필요한 정보(OS, 앱)를 담은 템플릿 | 컴퓨터의 디스크 이미지 (C 드라이브 복사본) |
| **EBS (Elastic Block Store)** | EC2에 연결하는 영구 블록 스토리지. 인스턴스가 종료되어도 데이터가 유지됨 | 컴퓨터의 외장 하드 디스크 |
| **보안 그룹 (Security Group)** | 인스턴스의 인바운드/아웃바운드 트래픽을 제어하는 가상 방화벽 | 건물의 출입 통제 시스템 |
| **키 페어 (Key Pair)** | 인스턴스에 안전하게 접속하기 위한 보안 자격 증명 (공개 키/개인 키) | 집 열쇠 |
| **Elastic IP** | 인스턴스에 할당할 수 있는 고정 공인 IP 주소 | 변하지 않는 집 주소 |

# IAM (Identity and Access Management) 기초

## AWS 리소스 접근을 제어하는 핵심 서비스

IAM은 AWS 리소스에 대한 **액세스를 안전하게 제어**하는 웹 서비스입니다. IAM을 통해 누가(**인증, Authentication**) 무엇을(**인가, Authorization**) 할 수 있는지를 중앙에서 관리합니다.

### IAM의 주요 구성 요소

- **사용자 (User)**: AWS와 상호 작용하는 사람 또는 애플리케이션.
- **그룹 (Group)**: 사용자들의 집합. 그룹에 권한을 부여하여 여러 사용자를 한 번에 관리합니다.
- **역할 (Role)**: 특정 개체(예: EC2 인스턴스)에 할당할 수 있는 임시 권한 세트.
- **정책 (Policy)**: 권한을 정의하는 JSON 문서. "누가, 무엇을, 어떻게 할 수 있는지"를 명시합니다.

### 제로 트러스트 (Zero Trust) 원칙

IAM은 기본적으로 모든 액세스를 거부하고, 정책을 통해 **명시적으로 허용된 작업만** 수행할 수 있도록 하는 보안 원칙을 따릅니다.

# IAM 구조 및 정책(Policy)

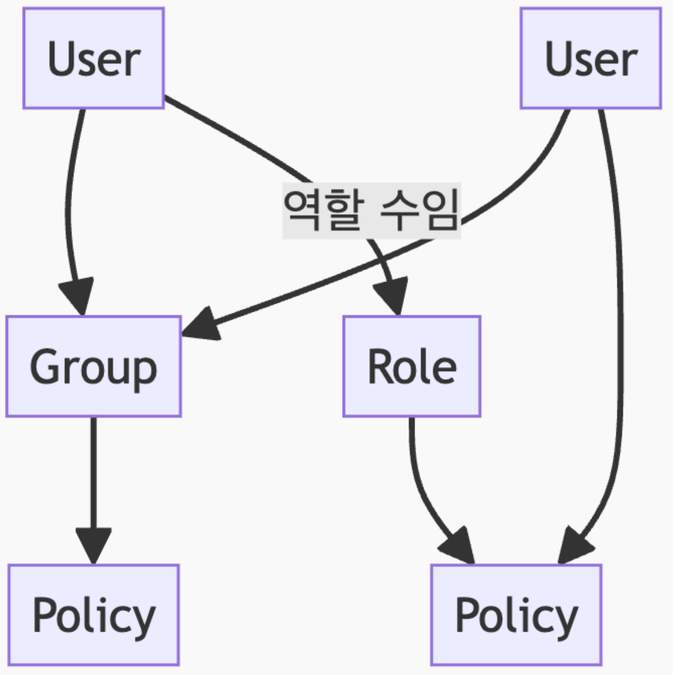
## JSON 문서로 권한을 정의하는 방법

### 정책(Policy) 기본 구조 예시

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::example-bucket/*"
    }
  ]
}
```

- **Effect**: `Allow` (허용) 또는 `Deny` (거부)
- **Action**: 허용/거부할 특정 API 작업 (예: S3 버킷에서 객체 가져오기)
- **Resource**: 작업이 적용되는 대상 리소스

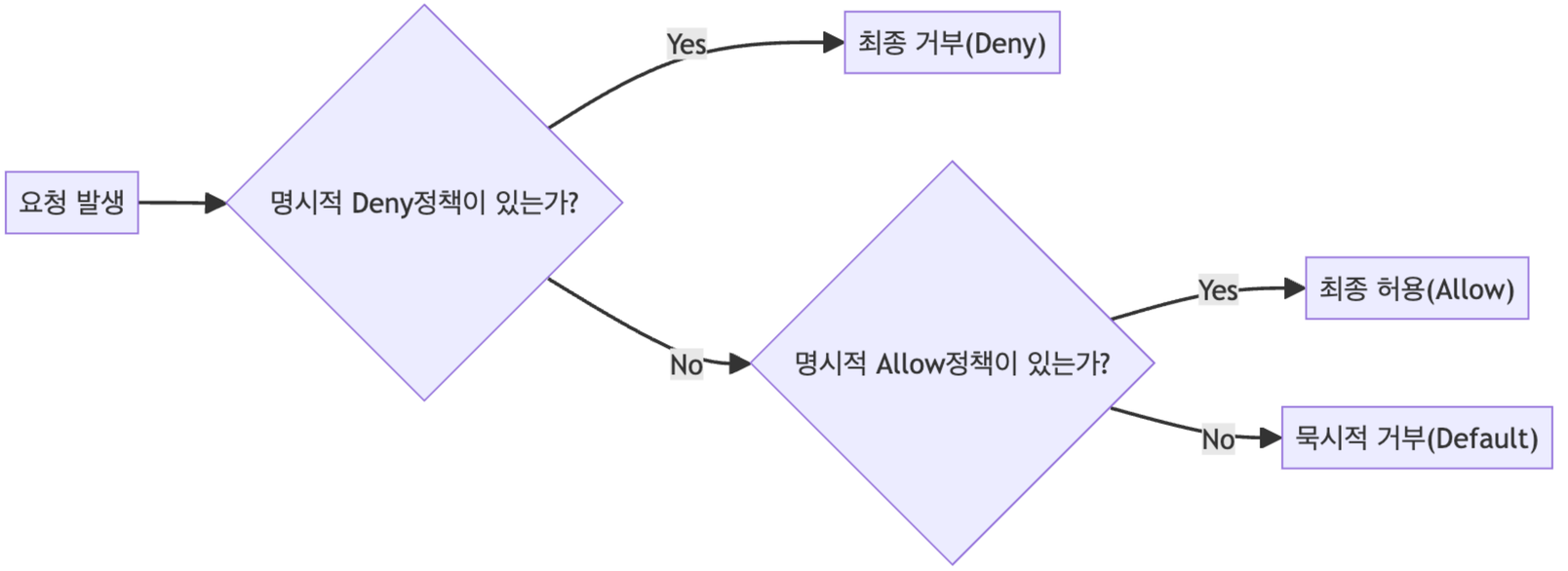
### IAM 구조



# IAM 정책 평가 로직

## 요청이 허용될지 거부될지 결정하는 규칙

AWS는 여러 정책이 적용될 때 다음 로직에 따라 최종적으로 접근 허용 여부를 결정합니다.




> **가장 중요한 규칙: Deny가 항상 우선합니다.**
>
> 단 하나의 Deny 정책이라도 존재하면, 다른 어떤 Allow 정책이 있더라도 최종적으로 요청은 거부됩니다.

# 접근 제어 모델: RBAC vs ABAC

## 역할 기반(RBAC)과 속성 기반(ABAC) 접근 제어 비교

조직의 규모와 복잡성에 따라 적합한 접근 제어 모델을 선택할 수 있습니다.

| 구분 | 역할 기반 액세스 제어 (RBAC) | 속성 기반 액세스 제어 (ABAC) |
| --- | --- | --- |
| **개념** | 사용자의 **역할(직무)**에 따라 권한 부여 | 사용자/리소스의 **속성(태그)**을 기반으로 권한 부여 |
| **장점** | • 직관적이고 이해하기 쉬움<br>• 대부분의 조직 구조와 잘 맞음 | • 권한이 자동으로 확장됨 (새 리소스에 태그만 붙이면 됨)<br>• 정책 수가 줄어들어 관리가 용이 |
| **단점** | • 리소스가 추가될 때마다 정책 업데이트 필요<br>• 세분화된 권한 관리가 복잡해질 수 있음 | • 초기 설정이 복잡할 수 있음<br>• 체계적인 태그 관리 전략이 필수 |

### ABAC 예시

사용자의 `CostCenter` 태그와 EC2 인스턴스의 `CostCenter` 태그가 **일치할 때만** 해당 인스턴스를 시작/중지할 수 있도록 허용하는 단일 정책을 만들 수 있습니다.

# 컴퓨팅 모델의 진화

## 서버 → 컨테이너 → 서버리스

인프라 관리의 부담을 줄이고 개발 생산성을 높이는 방향으로 발전해왔습니다.

### 서버 (Server)

전통적인 모델. 물리적 또는 가상 컴퓨터에서 애플리케이션을 직접 실행합니다. 인프라 관리에 대한 모든 책임이 사용자에게 있습니다.
**(예: EC2)**

### 컨테이너 (Container)

애플리케이션과 모든 종속성을 패키징한 경량 실행 환경. 서버 리소스를 더 효율적으로 활용할 수 있습니다.
**(예: Docker, Kubernetes)**

### 서버리스 (Serverless)

개발자가 서버 관리에 전혀 신경 쓰지 않고 코드 실행에만 집중하는 모델. 이벤트 기반으로 실행되며 사용한 만큼만 비용을 지불합니다.
**(예: AWS Lambda, Fargate)**

**핵심 변화**: "어디서 실행할까?" (서버 관리)에 대한 고민에서 **"무엇을 실행할까?" (비즈니스 로직)**에 대한 집중으로 전환

# 정리 및 Q&A

## 이번 시간 정리

- **클라우드 컴퓨팅**은 IT 리소스를 인터넷을 통해 빌려 쓰는 서비스로, 비용, 속도, 확장성 면에서 큰 이점을 가집니다.
- 서비스 모델은 관리 범위에 따라 **IaaS, PaaS, SaaS**로 나뉩니다.
- AWS는 전 세계 **리전**과 각 리전 내 여러 **가용 영역(AZ)**으로 구성되어 안정성과 고가용성을 제공합니다.
- **EC2**는 AWS의 가상 서버이며, AMI, EBS, 보안 그룹 등의 핵심 요소로 구성됩니다.
- **IAM**은 정책(Policy) 기반으로 사용자, 그룹, 역할의 권한을 제어하며, 'Deny 우선' 원칙을 따릅니다.
- 컴퓨팅 모델은 관리 부담을 줄이는 방향으로 **서버 → 컨테이너 → 서버리스**로 진화하고 있습니다.

## 질문 있으신가요?