Cloud Computing

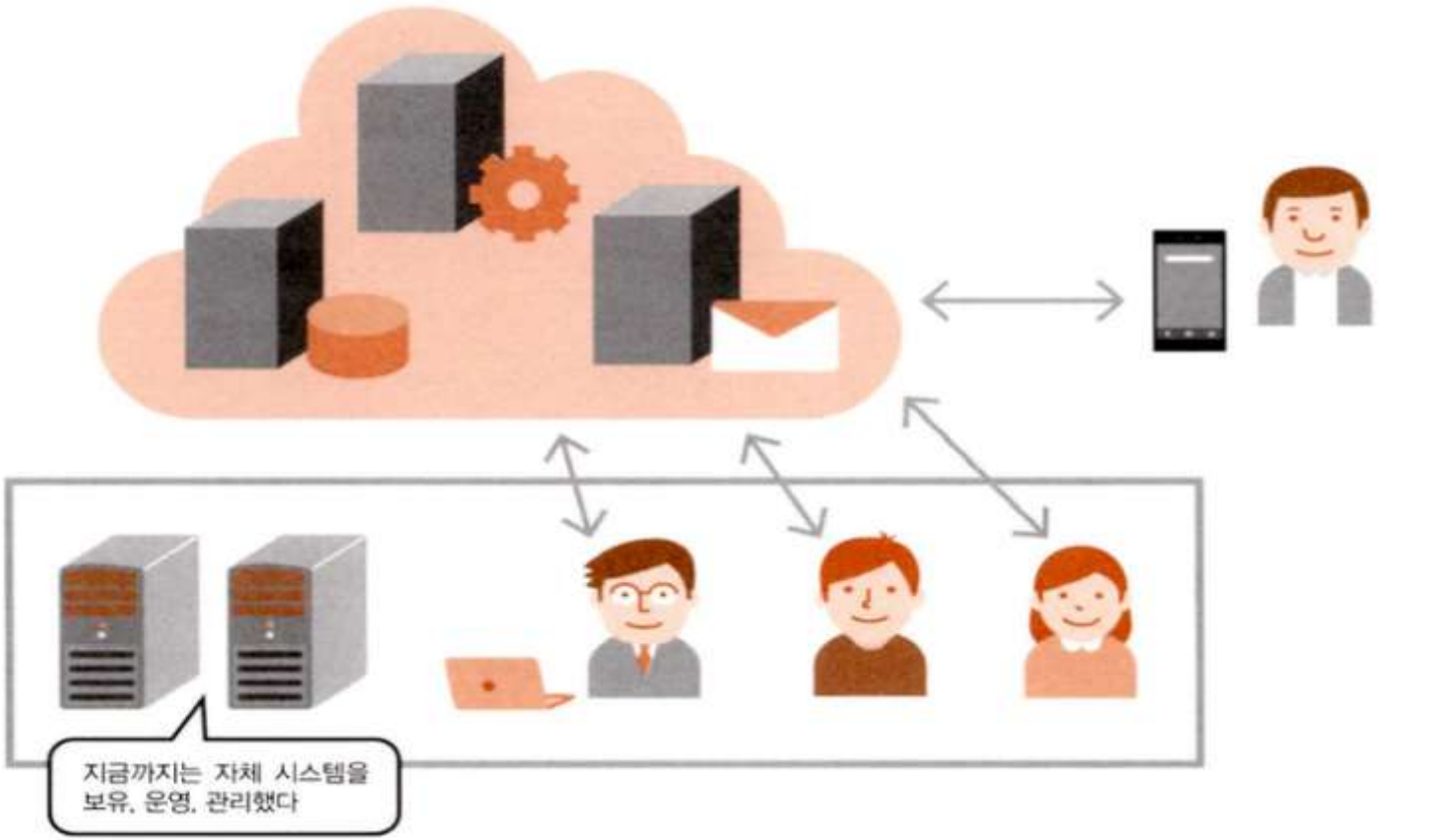
Cloud

- On-Premise
  - ✓ Cloud 개념이 도입되기 전에는 대부분의 기업이 자체 데이터 및 솔루션 등을 저장하기 위해 데이터 센터를 구축하여 IT 서비스를 수행하는 방식
  - ✓ 하드웨어를 포함한 모든 자원(CPU, 메모리, 디스크, OS, 네트워크, 라이선스 등)에 대한 초기 투자 비용과 탄력적이지 않은 제한된 용량으로 인해 지속적 관리 비용이 증가하는 단점이 있지만 기업에 내재화된 서비스를 통해 품질 및 보안에 대한 신뢰도는 높음
  - ✓ 최근 많은 기업이 On-Premise 방식에서 벗어나 Cloud 서비스 전환을 고민하고 있는데 그 이유는 높은 초기 도입 비용과 운용에 따른 추가 비용 때문
  - ✓ On-Premise 방식으로 설계 시 자원 사용량은 가급적 최대 사용량을 근거로 하고 네트워크 트래픽 또한 최대 순간 트래픽을 가정하기 때문에 고사양의 설계를 하게 되고 증설에 따른 시간적, 인적 비용도 무시할 수 없음
  - ✓ Cloud가 무조건 On-Premise 보다 비용이 낮은 것은 아닌데 동일 사양으로 1~5년간의 고정적 비용을 따져본다면 어느 시점부터는 Cloud 비용이 더 커질 수 있기 때문에 Cloud 도입은 비용 측면보다는 서비스의 가용성과 품질을 높여서 기업의 이익을 높일 수 있음
  - ✓ Cloud 접근 방식은 사용한 만큼 지불하는 정산 방식을 통해 필요에 따라 민첩하고 탄력적으로 사용할 수 있기 때문에 Cloud Computing 및 서비스에 대한 정확한 관리 및 조정 능력이 필요하고 현 상황에 적합한 Cloud 세팅 값을 찾는 것이 서비스의 안정화와 비용 절약의 시작

Cloud

❖ Cloud Computing

컴퓨터와 소프트웨어를 자신이 소유하는 것이 아니라, 네트워크를 통해 클라우드 사업자의 컴퓨터와 소프트웨어를 서비스로서 사용할 수 있습니다.

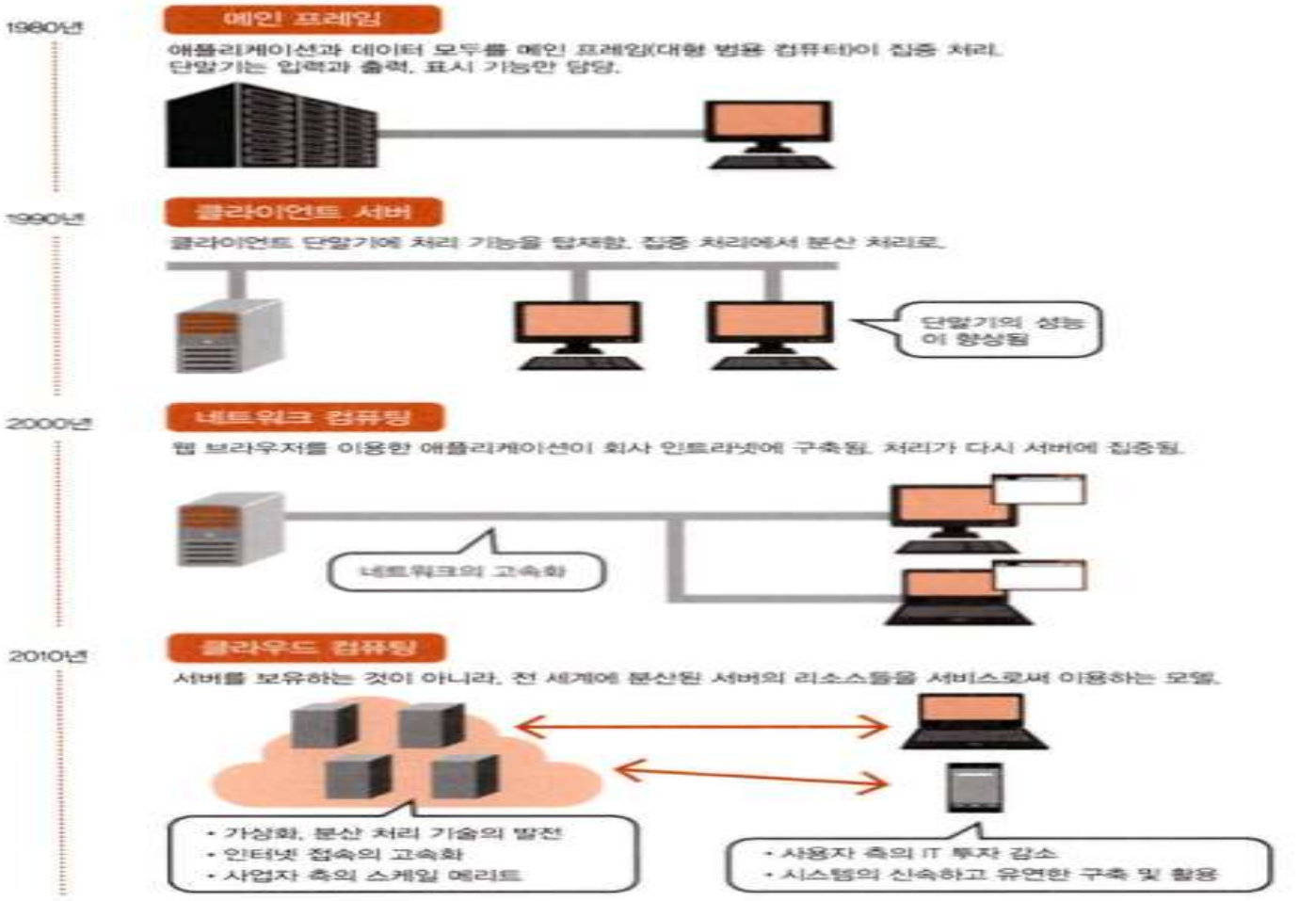


Cloud

- Cloud Computing
  - ✓ Cloud Computing이란 컴퓨터를 사용한 정보 처리를 자신이 보유한 PC가 아닌 인터넷 너머에 존재하는 Cloud 사업자의 컴퓨터에서 처리하는 것으로 사고 방식 혹은 개념을 나타내는 단어이며 어떤 특정 기술을 가리키는 것이 아님
  - ✓ Cloud는 구름이라는 뜻인데 이는 네트워크나 인터넷을 그림으로 표현할 때 구름 그림으로 표현했던 것에 유래
  - ✓ 네트워크 와 인터넷은 그 내용을 모르더라도 연결하기만 하면 서비스를 이용할 수 있는데 이와 마찬가지로 Cloud Computing 또한 인터넷 너머가 어떻게 되어 있는지 몰라도 서비스로써 사용할 수 있음
  - ✓ 기업이 Cloud를 이용할 경우 회사의 정보를 회사 안에 구축한 시스템에서 처리하는 것이 아니라 Cloud 사업자의 데이터 센터 안의 시스템에서 처리
  - ✓ IT 자산을 소유하는 것이 아니라 서비스로 이용하는 모델
  - ✓ Cloud 이용자는 인터넷에 접속한 후 웹 브라우저나 Cloud 서비스 전용 소프트웨어 등을 통해 서비스를 이용할 수 있음
  - ✓ 서비스 제공자의 최소한의 관리나 개입만으로도 신속하게 생성/제거/구성 할 수 있는 공유 공간의 컴퓨팅 자원들을 언제, 어디서 어떤 단말 인지와 관계없이 필요할 때 편리하게 네트워크에 접속할 수 있게 하는 모델

Cloud

- Cloud 등장 배경
  - ✓ Cloud가 등장하기까지의 흐름



Cloud

- Cloud 등장 배경
  - ✓ Cloud가 등장하기까지의 흐름
    - 1980년대는 메인 프레임이라고 하는 대형 범용 컴퓨터의 시대였는데 모든 데이터와 애플리케이션을 메인 프레임에 모아 처리하였고 단말기는 입력 과 출력 표시 기능만 담당
    - 1990년대에는 클라이언트 단말기에도 처리 기능을 부여한 분산형 클라이이언트 서버 모델이 주류
    - 2000년대에 들어서면서 사내 시스템이 네트워크 환경 위에 구축되게 되어 처리가 서버에 집중
    - 2010년경부터 전 세계에 분산 배치된 서버 리소스를 필요한 때 필요한 만큼 사용하는 Cloud Computing 모델로 발전

Cloud

- ❖ Cloud 등장 배경
  - ✓ Cloud가 보급된 배경
    - ● 다양한 기술의 발전
      - ◆ CPU의 처리 속도 고속화
      - ◆ 가상화 기술 과 분산 처리 기술 등의 발전
      - ◆ 모바일의 융성
      - ◆ 빨라지고 저렴해진 네트워크
      - ◆ 거대해진 데이터 센터에 의한 규모의 경제(스케일 메리트)
    - ● 기업 사용자와 Cloud 사업자 모두에게 Cloud를 받아들일 환경 조성
      - ◆ 기업 사용자에게는 IT 투자 비용의 절감, 유연한 서비스 설계 와 이용, 구축 및 운용 부담의 경감 등이 과제가 되고 있으며 Cloud를 통해 이를 해결하고자 하는 기대가 있음
      - ◆ Cloud 사업자에게는 기업 사용자에게 컴퓨팅 자원을 셀프 서비스의 형태로 제공하게 되므로 서비스 제공의 효율성이 높아지고 지속적인 매출을 올릴 수 있다는 점에서 안정적인 수익원이 된다는 장점이 있음

Cloud

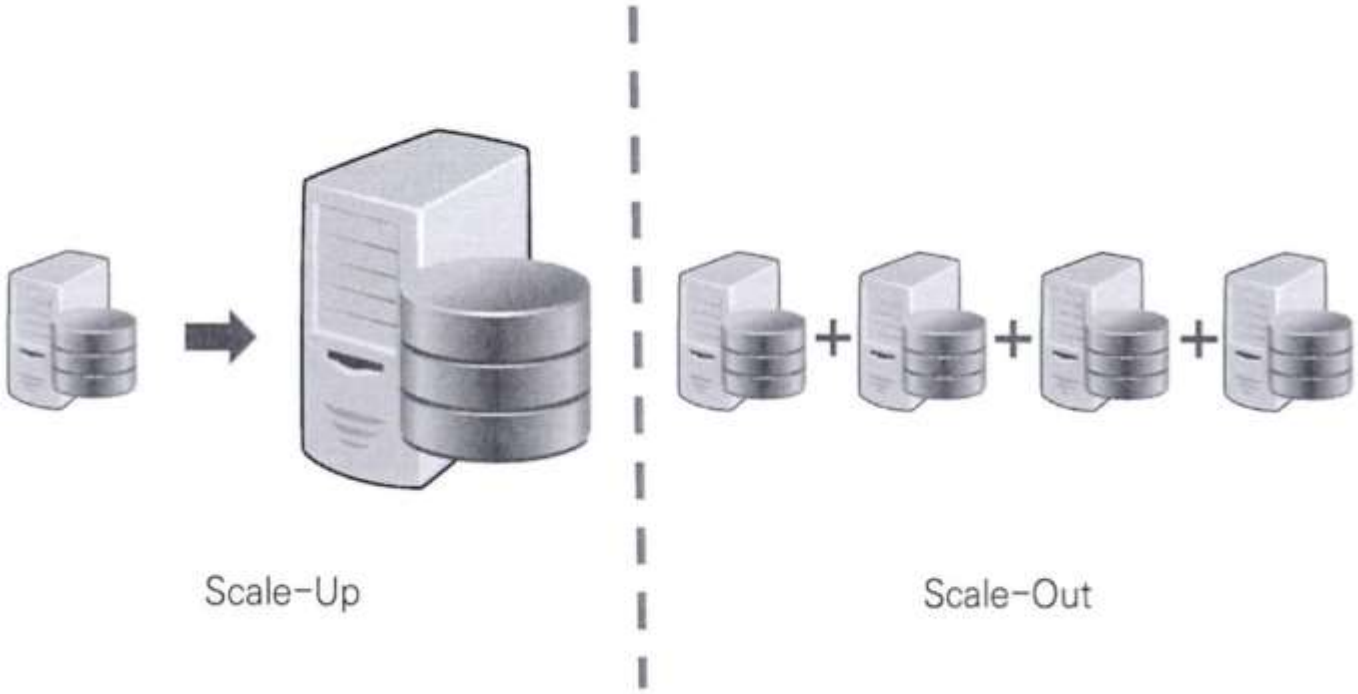
- Cloud 정의 와 특징
  - ✓ Cloud의 범주 안에서 이용자에게 제공되는 서비스의 종류와 이용 형태는 가지 각색
  - ✓ NIST(미국 국립 표준 기술연구소)가 정한 Cloud Computing의 정의
    - Cloud Computing이란 공유 구성이 가능한 컴퓨팅 리소스(네트워크, 서버, 스토리지, 애플리케이션 서비스)의 통합을 통해 어디서나 간편하게 요청에 따라 네트워크를 통해 접근하는 것을 가능하게 하는 모델이며 이는 최소한의 이용 절차 또는 서비스 공급자의 상호 작용을 통해 신속히 할당되어 제공되는 것
  - ✓ Cloud 특징
    - 주문형 셀프서비스(On-Demand Self-Service): 사용자가 별도의 기술 습득 없이 필요할 때 온라인으로 즉시 사용
    - 광대역망 액세스(Broad Network Access): 네트워크를 통한 컴퓨팅 자원 접근(Any time, Any place, Any device)
    - 자원 공동 관리(Resource Pooling): 다중 임대 모델을 통한 자원 할당(Multi-Tenancy)
    - 빠른 요구 탄력성(Rapid Elasticity): 비즈니스 상황에 따른 컴퓨팅 자원의 탄력적 사용(Flexibility, Scalability)
    - 측정 가능한 서비스(Measured Service): 사용량이 명확하게 기록되고 서비스를 사용한 만큼 비용 지불(Pay-Per-Use, Pay as you go)

Cloud

- Cloud 설계 시 고려할 사항
  - ✓ Fault Tolerance(내결함성)
    - 시스템에 어떤 문제가 발생하더라도 전체 서비스가 중단되지 않고 동작을 계속할 수 있는 능력
    - 하나의 서버나 네트워크 장비가 고장 나더라도 다른 자원이 그 역할을 대신해서 사용자는 아무 이상없이 서비스를 이용할 수 있도록 하는 구조
  - ✓ High Availability(고가용성)
    - 시스템이 가능한 한 오랫동안 끊김없이 작동하는 것
    - 고가용성을 확보하려면 단순히 서버를 하나 더 놓는 것만으로는 부족하고 구성 요소 전체를 이중화하고 네트워크 스토리지까지도 장애에 대비한 설계를 갖춰야 함

Cloud

- Cloud 설계 시 고려할 사항
  - ✓ Scalability(확장성, 유연성)

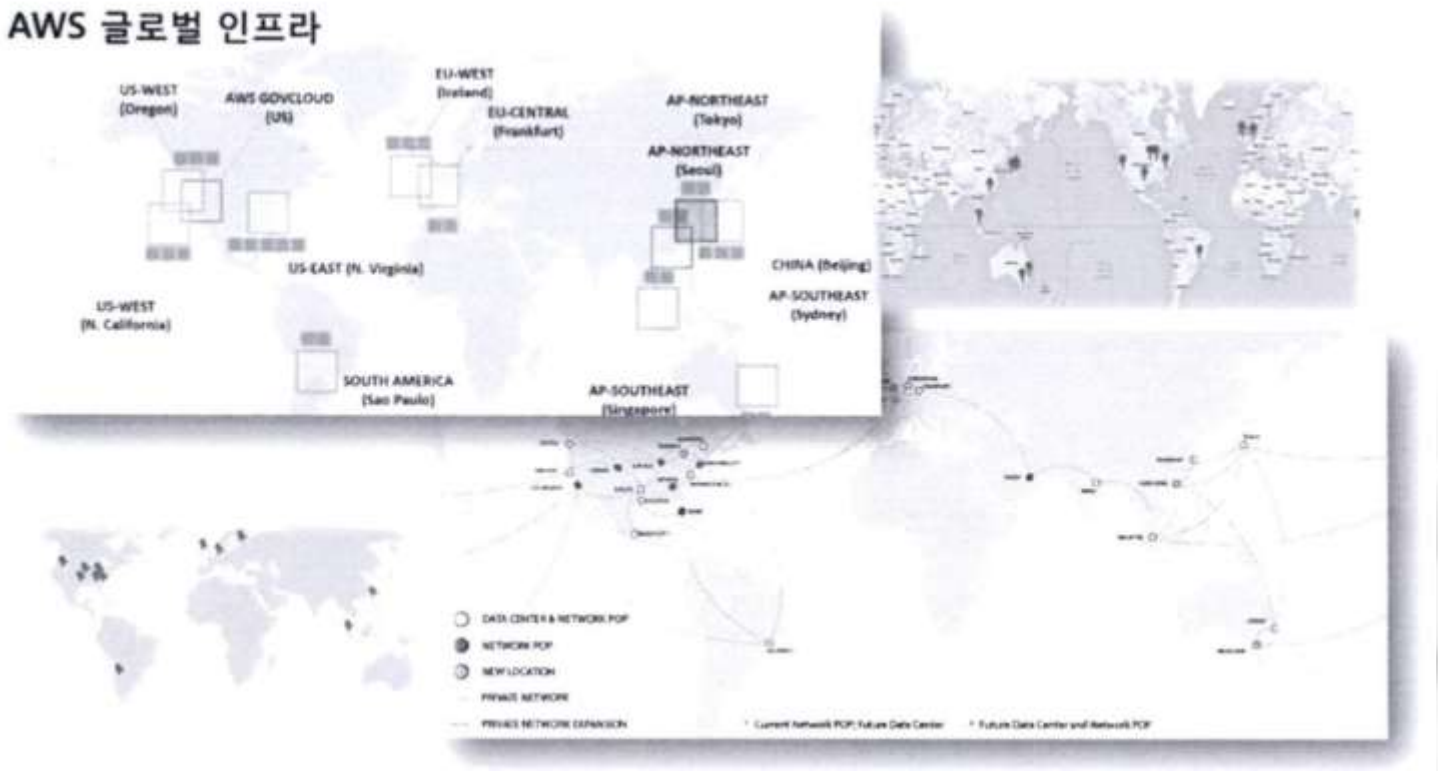


Cloud

- Cloud 설계 시 고려할 사항
  - ✓ Scalability(확장성, 유연성)
    - 자체 시스템을 구축할 경우 서버의 증축 및 시스템 확장에 고도의 기술과 많은 비용이 필요한 반면에 Cloud의 경우에는 컴퓨팅 리소스를 필요할 때 필요한 만큼 확장하고 많은 리소스가 필요하지 않을 때는 축소하는 등 유연한 활용이 용이
    - 시스템의 처리 능력을 수평 또는 수직으로 확장할 수 있는 능력
    - 수평 확장은 서버 수를 늘리는 것이고 수직 확장은 서버의 성능 자체를 높이는 것
    - 인디스쿨 사례
      - 약 10년 동안 인디스쿨은 국내 IDC 서버를 이용해 매달 고정된 금액을 지불하는 호스팅 서비스를 사용했는데 서버의 노후화로 인한 장애 및 속도 지연 이슈 뿐만 아니라 회원이 14만 명까지 늘어나 급증하는 트래픽에 유연하게 대응하지 못하는 한계가 있었음
      - 인디스쿨은 초등학교 선생님들이 회원이기 때문에 방학 중에는 트래픽이 거의 발생하지 않는 특징을 가지고 있는데 사용량이 거의 없는 방학 중에도 계속해서 비용이 그대로 지출되었고 서버 용량이 초과할 경우 서비스를 안정적으로 제공하기 위해 서버를 추가해야 했는데 서버 주문, 발주, 배송까지 준비하는 기간이 너무 김

Cloud

- Cloud 설계 시 고려할 사항
  - ✓ Redundancy or Resilience(중복성 또는 탄력성): Business Continuity Management(BCM)



Cloud

- Cloud 설계 시 고려할 사항
  - ✓ Redundancy or Resilience(중복성 또는 탄력성): Business Continuity Management(BCM)
    - 수요 변화에 따라 자원을 자동으로 늘리거나 줄일 수 있는 능력
    - 탄력성은 확장성과 비슷하지만 자동화라는 요소가 추가됨

Cloud

- Cloud 한계
  - ✓ Internet Access(No Internet = No Cloud)
  - ✓ Security
  - ✓ Privacy
  - ✓ Vendor Lock-in(Application migration may be impossible)

Cloud

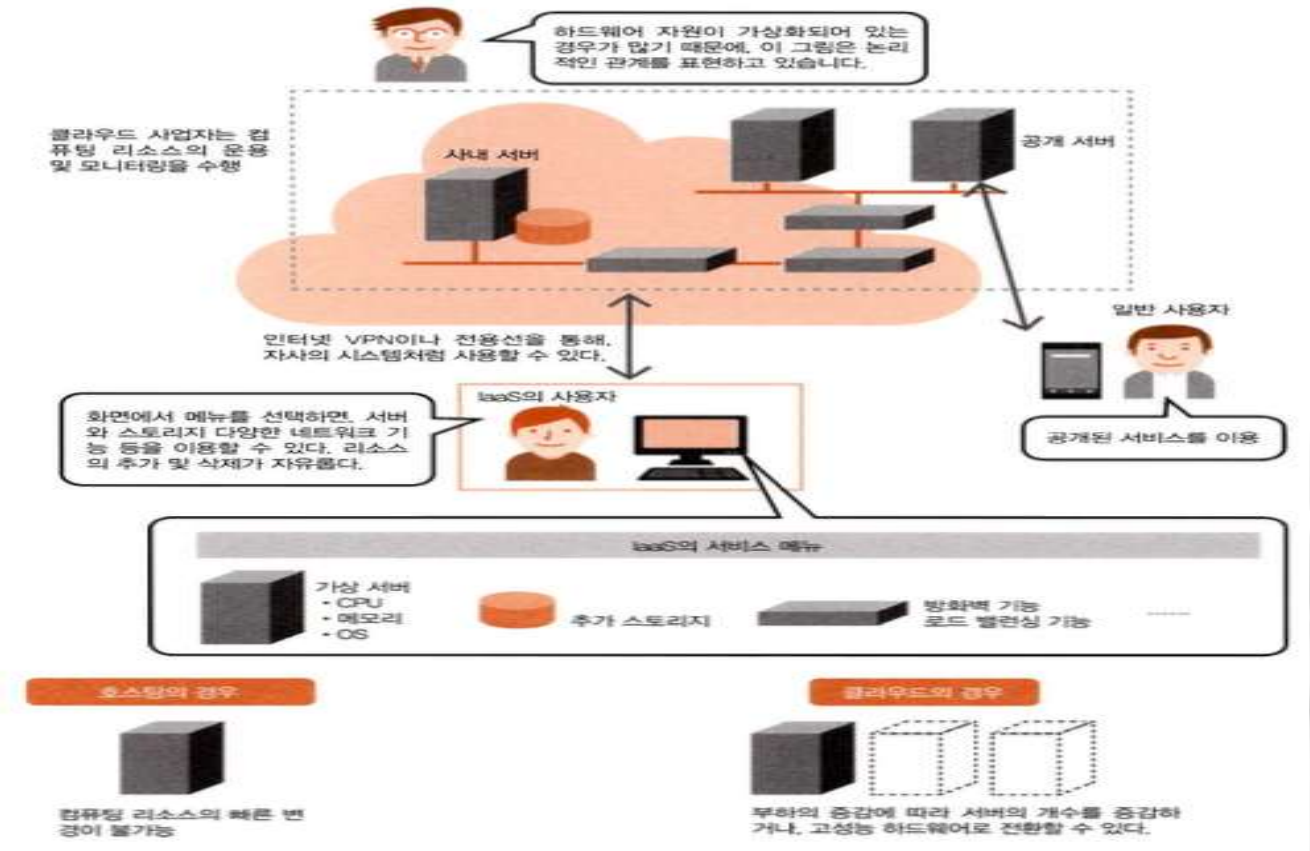
❖ Cloud 서비스 종류

Cloud

- Cloud 서비스 종류
  - ✓ Infrastructure as a Service - IaaS
    - 서버, 스토리지, 네트워크, 방화벽 및 로드밸런서 같은 인프라 하드웨어 자원을 가상화하여 사용자 요구에 따라 인프라 자원을 사용할 수 있게 제공하는 Cloud 서비스 방식
    - IaaS는 자동화되고 신속한 확장성을 갖고 있는 IT 인프라를 의미하며 비용은 사용량에 따라 지불하는 방식
    - 이 방식의 장점은 자유도가 높고 세밀한 설정이 가능하다는 것
    - 서비스로서의 인프라 스트럭처의 개념도 중요하지만 물리적으로 인프라에 사용되는 각각의 자원(CPU, 메모리, 디스크, 네트워크 등)의 특징에 대한 지식이 유지 관리 차원에서 반드시 필요
    - 대표적인 IaaS 서비스 중 하나가 Amazon Web Services가 제공하는 Amazon Elastic Compute Cloud(EC2)
    - All CSPs are providing IaaS to the public

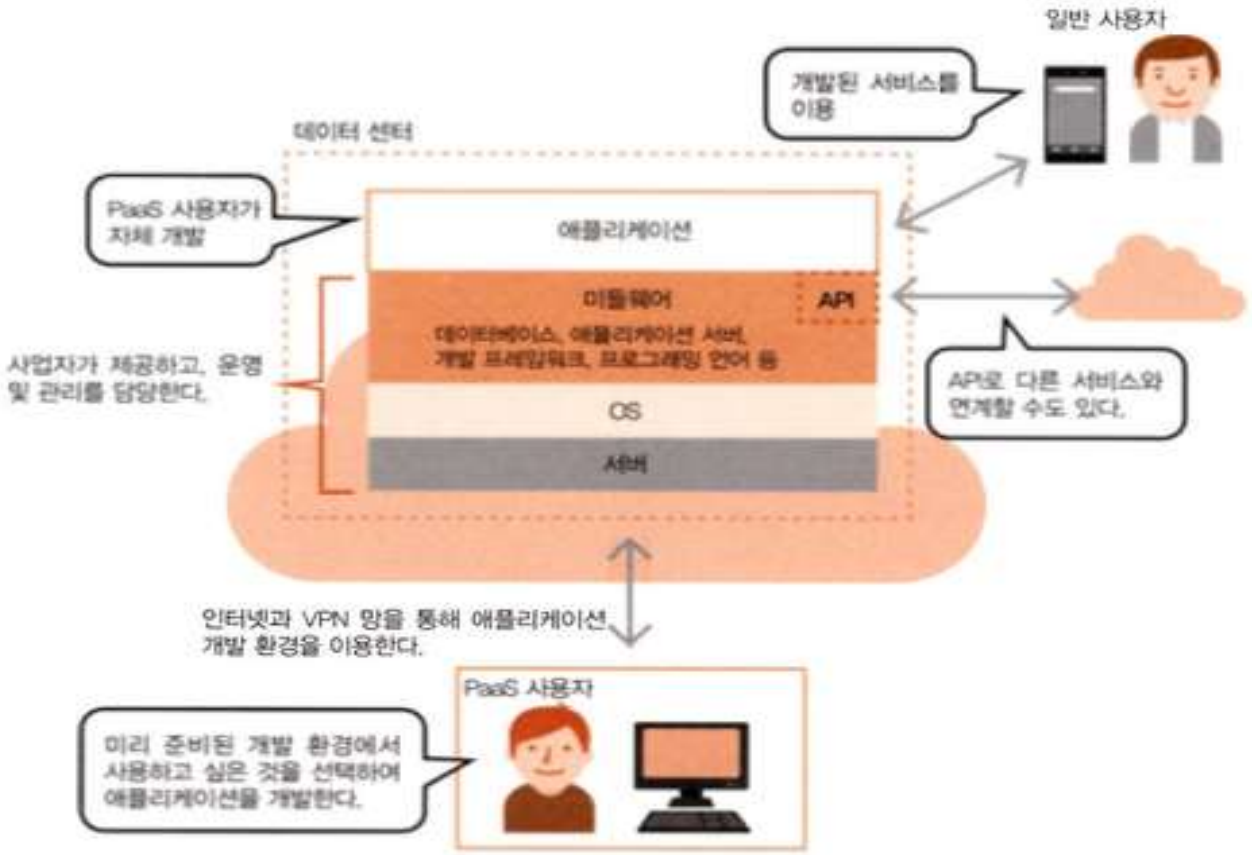
Cloud

- Cloud 서비스 종류
  - ✓ Infrastructure as a Service - IaaS



Cloud

- Cloud 서비스 종류
  - ✓ Platform as a Service - PaaS



Cloud

- Cloud 서비스 종류
  - ✓ Platform as a Service - PaaS
    - 서비스 개발자가 애플리케이션 개발, 실행, 관리 등을 할 수 있도록 안정적인 플랫폼(실행 환경) 또는 프레임워크를 제공하는 Cloud 서비스 방식
    - 기업 사용자가 자사에서 애플리케이션 개발 환경을 처음부터 구축하는 것은 많은 시간이 소요되므로 PaaS에는 Java, PHP, Ruby 등의 프로그래밍 언어를 지원하는 애플리케이션 실행 환경이나 데이터베이스 등이 미리 설치되어 있어서 인프라 구축을 하지 않아도 그 기반을 사용할 수 있으므로 단기간에 프로그램을 개발하여 서비스를 제공할 수 있음 – 서비스 개발에만 집중
    - laaS 와 차이점은 서버, 네트워크, 보안 부분을 Cloud 사업자에게 위임한다는 점으로 구축 및 운영이 쉽고 SaaS는 정해진 소프트웨어를 서비스로 제공하지만 PaaS는 자사에서 개발한 프로그램을 가동할 수 있기 때문에 애플리케이션 활용 자유도가 높다는 점이 장점이지만 반대로 말하면 서버 및 미들웨어의 상세한 설정을 할 수 없을 수 있으며 특정 PaaS 환경에 대한 의존도가 높아지게 되면 다른 환경으로의 마이그레이션이 어려워질 수도 있다는 단점이 있는데 Cloud 사업자가 마련한 환경이 자사에 어울리는지를 점검하는 것 또한 포인트
    - 대표적인 PaaS 서비스나 소프트웨어로는 세일즈 포스가 제공하는 force.com과 사이보우즈의 kintone, 오픈 소스 PaaS 기반 소프트웨어 인 Cloud Foundry, Google Cloud Platform, ServiceNow Platform, DBMS on Cloud (OCI) 등을 들 수 있음

Cloud

- Cloud 서비스 종류
  - ✓ Platform as a Service - PaaS
    - PaaS로 제공되는 대표적인 도구와 서비스

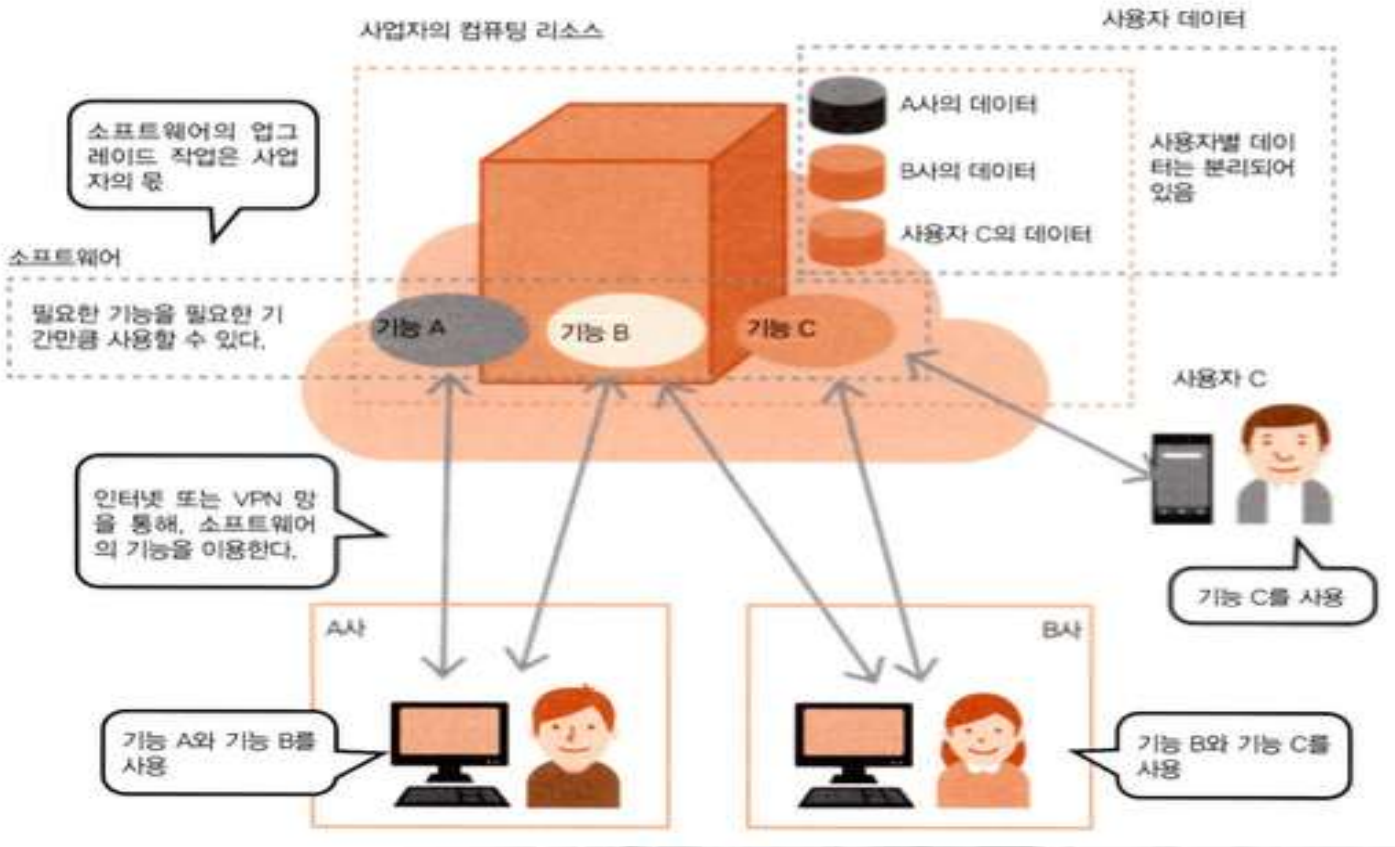
개발 도구, 부속 서비스	API 서비스	인증 · 과금, 알림, 분석 등의 부가 서비스
	SDK	모바일용 소프트웨어 개발킷 등
	개발 프레임워크	Ruby on Rails, Sinatra, Spring, Node.js, Eclipse 등
핵심 기능	프로그래밍 언어	Ruby, Java, Python, PHP 등
	애플리케이션 서버	Apache Tomcat, Jboss 등
	데이터베이스 서비스	MySQL, PostgreSQL, MongoDB, Amazon RDS, Oracle DB 및 Microsoft SQL Server 등
	메시징 미들웨어	RabbitMQ, Amazon SQS 등
	다른 서비스의 지원	추가 기능, API와의 연계 등
	기타	애플리케이션 통합, 비즈니스 프로세스 관리, 데이터 통합, 관리 파일 전송, 포털, 보안, 테스트 환경 등

Cloud

- Cloud 서비스 종류
  - ✓ Software as a Service - SaaS
    - 업무에서 사용하는 소프트웨어의 기능을 인터넷 등의 네트워크를 통해 필요한 만큼 서비스로 이용할 수 있도록 제공하는 형태
    - 하나의 서버를 여러 기업에서 공유하는 것을 전제한 멀티 테넌트 방식 서비스를 제공하지만 데이터는 기업 사용자 별로 분리되도록 설계하여 보안성을 확보
    - 소프트웨어의 업데이트 작업은 사용자가 아니라 Cloud 사업자가 수행하기 때문에 항상 최신 기능을 사용할 수 있으며 소프트웨어의 버그가 방치되지 않음
    - SaaS의 경우에는 서비스를 계약하고 사용자 계정이 준비되면 즉시 서비스 이용을 시작할 수 있기 때문에 패키지 소프트웨어를 구입하는 것에 비해 도입까지의 납기를 크게 줄일 수 있음
    - SaaS로 제공되는 대표적인 소프트웨어로는 전자 메일, 그룹웨어, CRM(Customer Relationship Management, 고객 관리 시스템) 등
    - SaaS는 인터넷을 통해 접속할 수 있기 때문에 회사의 PC는 물론 이동할 때에도 스마트폰, 태블릿과 같은 단말기로 접속이 가능
    - 전자 메일 및 일정, 영업 정보 등을 여러 사람과 교환할 수 있는 서비스를 채택하는 사례가 기업의 사업 규모와 관계없이 늘어나고 있음
    - SaaS를 도입한다면 처음에는 전자 메일과 그룹웨어 등을 검토해 보는 것이 좋을 것인데 재무 회계/인사 급여와 같이 기업의 핵심이 되는 미션 크리티컬 한 영역에도 도입 사례가 늘어나고 있음
    - SaaS 서비스로는 구글의 Google Apps, salesforce, MS Office 365, Naver Office 등

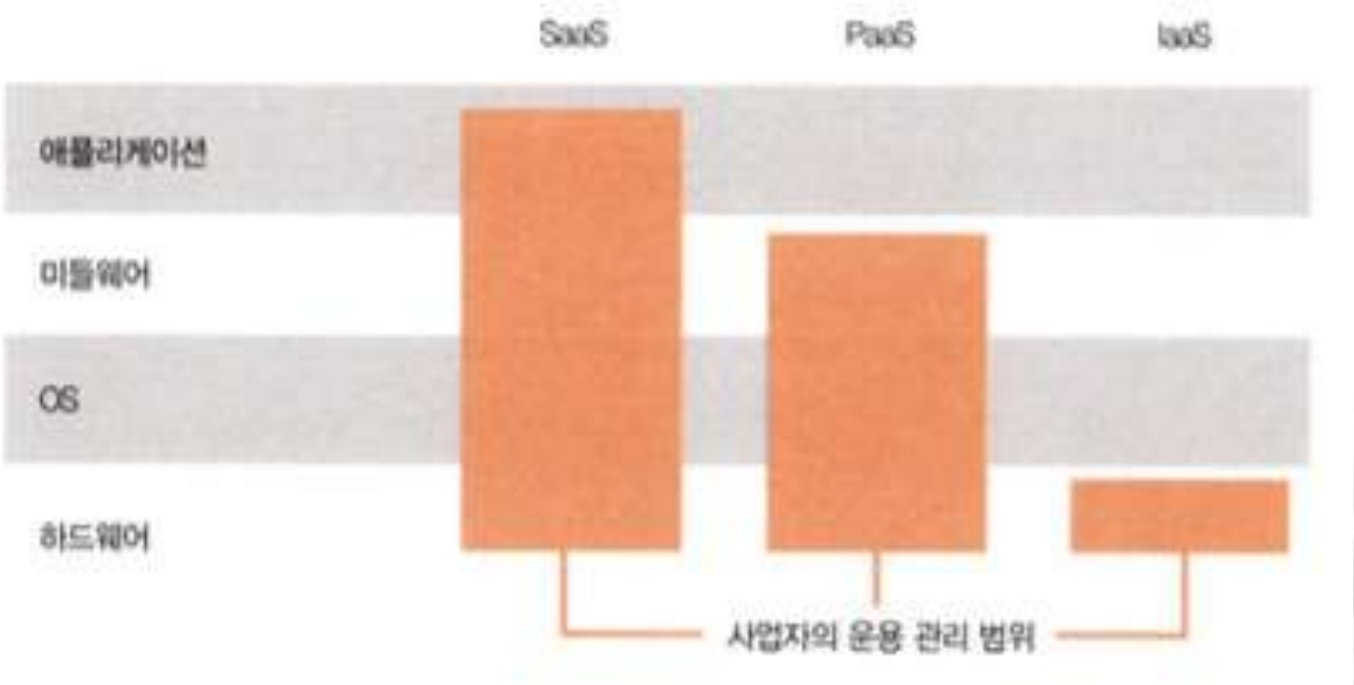
Cloud

- Cloud 서비스 종류
  - ✓ Software as a Service - SaaS



Cloud

- Cloud 서비스 종류
  - ✓ Software as a Service - SaaS



Cloud

- ❖ Cloud 서비스 종류
  - ✓ Desktop as a Service - DaaS
    - ● 개요
      - ◆ 서비스로서의 데스크톱을 의미하는데 인터넷만 연결되면 언제, 어디서나, 어떤 기기로도 기업 내부 망에 접속할 수 있는 Cloud 서비스의 일종
      - ◆ 모든 정보가 중앙 서버에 저장되기 때문에 직원의 개별 PC가 바이러스에 노출되거나 파손, 분실되어도 정보 유출의 위험이 적으며 외부에서 기업의 업무 망에 접근하는 가장 안전한 기술

Cloud

- ❖ Cloud 서비스 종류
  - ✓ Desktop as a Service - DaaS
    - ● 배경
      - ◆ 기존의 전통적인 근무 형태는 직원이 사무실에서 회사가 준 데스크톱으로 업무를 보는 것으로 해당 데스크톱에는 운영체제(OS), 보안 솔루션, 소프트웨어 등 업무 수행에 필요한 프로그램들이 설치되어 있지만, 출장, 재택 근무 등의 이유로 직원들이 사무실이 아닌 제3의 공간에서 일을 해야 할 때 회사 입장에서는 직원이 개인 업무 PC를 외부에 반출하도록 허용해야 하는지 등의 고민이 발생하는데 허용한다면 외부에서 사내 업무 망을 이용할 수 있도록 접근 권한을 줘야 할지 등의 문제가 뒤따라 오며 만약 직원이 업무 PC를 분실하거나 파손할 경우, 업무 자료가 소실될 수 있음
      - ◆ 업무 PC 반출을 불허한다면 직원 개인의 PC나 스마트폰으로 사내 업무 망을 이용할 수 있도록 접근 권한을 주어야 하는데 만약 직원 개인 단말기가 보안에 취약하다면 해당 기기를 통로로 기업 내부 망에 악성 코드가 침투할 수 있는데 이처럼 직원들이 원격지에서 근무할 때 기업은 여러 보안상의 문제들에 봉착하게 됨
      - ◆ 이러한 문제의 해결책으로 등장한 것이 DaaS

Cloud

- Cloud 서비스 종류
  - ✓ Desktop as a Service - DaaS
    - 특징
      - 연속성: 구성원들이 일하는 장소나 기기에 상관없이 사무실과 동일한 업무 경험을 유지할 수 있도록 지원
      - 비용 절감
        - 가상의 업무 공간을 구독의 형태로 제공받는 서비스
        - DaaS는 여타 Cloud 서비스와 마찬가지로 초기 구축 비용이 없음
        - 구축비의 유무는 가상화 데스크톱 기술인 VDI(virtual desktop infrastructure)와 DaaS를 차별화하는 요소
        - DaaS 와 VDI가 구현하는 기능은 데스크톱을 가상화하여 제공하는 것인데 다만 이런 가상화가 일어나는 공간이 사용자가 보유한 인프라라면 VDI고 전문 Cloud 공급사의 인프라라면 DaaS가 되는 것
        - DaaS는 VDI를 Cloud 형태로 제공하는 서비스라고 할 수 있는데 과거 기업이 VDI를 도입하려면 자체 인프라를 소유해야 했기에 많은 비용이 들었지만 DaaS는 약정 이용료만 되면 VDI를 구독의 형태로 이용할 수 있음

Cloud

- ❖ Cloud 서비스 종류
  - ✓ Desktop as a Service - DaaS
    - ● 특징
      - ◆ 보안성
        - ▪ 기업의 업무 망을 중앙의 가상화된 서버에 조성하여 직원 개인 PC에는 화면만 전송하는 기술이므로 직원 개개인의 접속 단말이 보안에 취약해도 회사의 중앙 업무 망은 영향을 받지 않음
        - ▪ DaaS 계정에 로그인하여 중앙 업무 망에 접속한 사용자는 개인 단말로 업무를 처리하지만 모든 작업은 중앙 업무 망에서만 이루어지므로 사용자는 중앙 업무 망에서 처리된 결괏값을 자신의 단말에서 볼 수만 있음
        - ▪ 중앙 업무 망의 화면이 사용자에게 그림 파일로 실시간 전송되기 때문에 DaaS를 사용하면 수많은 사람들이 업무 망을 자유롭게 이용하면서도 업무 망의 정보는 안전하게 보호
      - ◆ 관리 편의성
        - ▪ DaaS는 중앙 통제형 원격 서비스로 관리자는 중앙 서버에서 직원들에게 Cloud 데스크톱을 할당할 수 있으며 이후의 관리 및 회수도 일괄 처리할 수 있음
        - ▪ 각종 업무용 소프트웨어를 단시간 내 전체 직원에게 배포하는 것도 DaaS에서는 가능
        - ▪ 소프트웨어를 최신 버전으로 한꺼번에 업데이트하는 것은 물론 동시에 데스크톱 수량도 유연하게 늘리거나 줄일 수 있음

Cloud

- ❖ Cloud 이용 모델
  - ✓ Public Cloud
    - ● 개요
      - ◆ Cloud Service Provider(AWS, GCP, Azure, Oracle 등)가 시스템을 구축하고 인터넷 망 등의 네트워크를 통해 불특정 다수의 기업과 개인에게 서비스를 제공하는 형태

Cloud

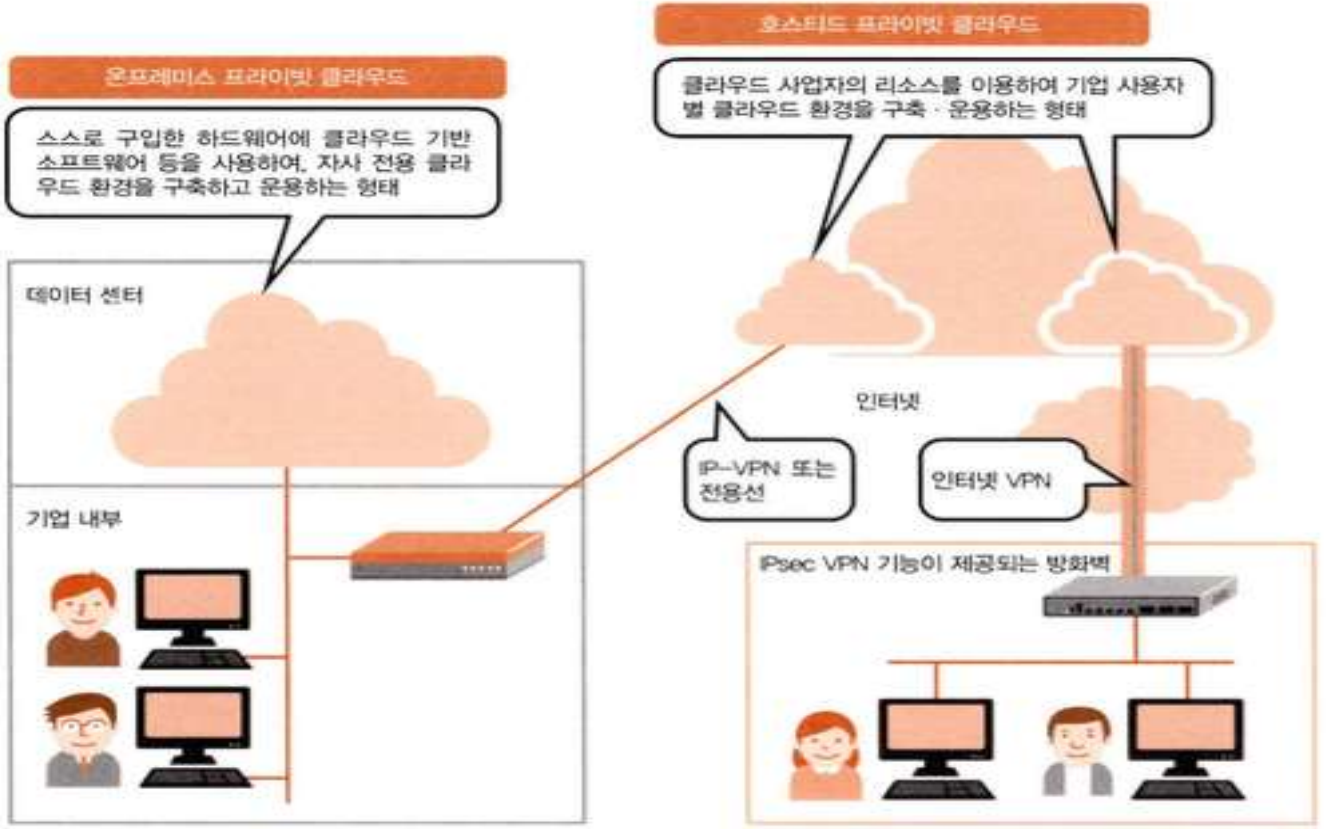
- Cloud 이용 모델
  - ✓ Public Cloud
    - 특징
      - ◆ 사용량에 따라 비용을 지불하는 요금 산정 방식을 사용하는데 사용자 및 그룹 단위로 권한 관리를 통해 서비스를 격리하기 때문에 사용자 간의 간섭이 발생하지 않음
      - ◆ 사용자는 IT 자산을 보유하지 않더라도 컴퓨팅 리소스를 서비스로 사용할 수 있으며 필요한 컴퓨팅 자원을 단기간에 저비용으로 마련할 수 있고 운용 관리 부담이 적다는 장점이 있음
      - ◆ Cloud 서비스 이용자를 제한하지 않는 방식
      - ◆ 인터넷 접속이 가능한 모든 사용자를 위한 Cloud 서비스
      - ◆ 서비스 내부에 저장된 데이터나 서버 자원은 사용자 별로 관리
      - ◆ 사용자 간에 데이터 간섭이 없도록 관리(Multi-Tenancy)

Cloud

- Cloud 이용 모델
  - ✓ Private Cloud
    - 폐쇄적인 구조
    - Cloud 서비스의 사용자 또는 사업자의 데이터 센터에 Cloud 관련 기술이 활용된 자사 전용 환경을 구축하여 컴퓨팅 리소스를 유연하게 이용할 수 있는 형태
    - 특정 기업의 특정 사용자 만을 대상으로 하는 Cloud 서비스
    - 가상화, 자동화와 같은 Cloud 관련 기술의 활용으로 인해 시스템의 성능과 비용이 최적화 되므로 유연한 사용자 정의가 가능하다는 점이 특징
    - 컴퓨팅 자원과 저장 데이터가 기업 내부에 저장
    - 자원과 데이터의 제어권을 기업 자체에서 가지고 있음
    - 물리적인 데이터 보안 측면이 Public Cloud보다 강함
    - 구현 방식
      - 자사가 보유하고 운용 - On-Premise Private Cloud
      - Cloud 사업자가 장비를 보유하고 Private Cloud 서비스를 제공하는 형태 - Hosted Private Cloud

Cloud

- ❖ Cloud 이용 모델
  - ✓ Private Cloud
    - ● 구현 방식



Cloud

- ❖ Cloud 이용 모델
  - ✓ Private Cloud
    - ● 구현 방식

	프라이빗 클라우드		퍼블릭 클라우드
	온프레미스	호스티드	
리소스	전용	전용	공유
인프라 운용	기업 사용자	사업자	사업자
구축 속도	△	○	◎
도입 비용	△	○	◎
운영 비용	△	○	◎
보안	◎	◎	○
유연성	○	◎	◎

Cloud

- Cloud 이용 모델
  - ✓ Community Cloud
    - 공통의 목적을 가진 특정 기업들이 Cloud 시스템을 형성하여 데이터 센터에서 공동 운영하는 형태
    - Public Cloud 와 Private Cloud의 중간적인 형태

Cloud

- Cloud 이용 모델
  - ✓ Hybrid Cloud
    - Hybrid Cloud 방식은 Public Cloud 와 Private Cloud를 네트워크를 통해 결합하여 두 가지 서비스의 장점을 활용할 수 있도록 만든 Cloud 서비스 방식
    - 서로 다른 Cloud 간에 데이터와 애플리케이션 공유 및 이동이 유연하게 처리될 수 있고 용도에 맞는 서비스 구현에 유리
    - 기업 내부의 민감하고 중요한 데이터 처리 작업은 통제력을 강화하기 위해 Private Cloud를 사용하고 일반 업무 데이터 처리 같은 보안 요구 사항이 낮은 작업이나 워크로드가 지속해서 증가하는 작업에는 리소스 자동 조정이 가능한 Public Cloud를 사용하기도 하는데 최근에는 단순히 가상 서버와 물리 서버의 결합으로 보기도 함

Cloud

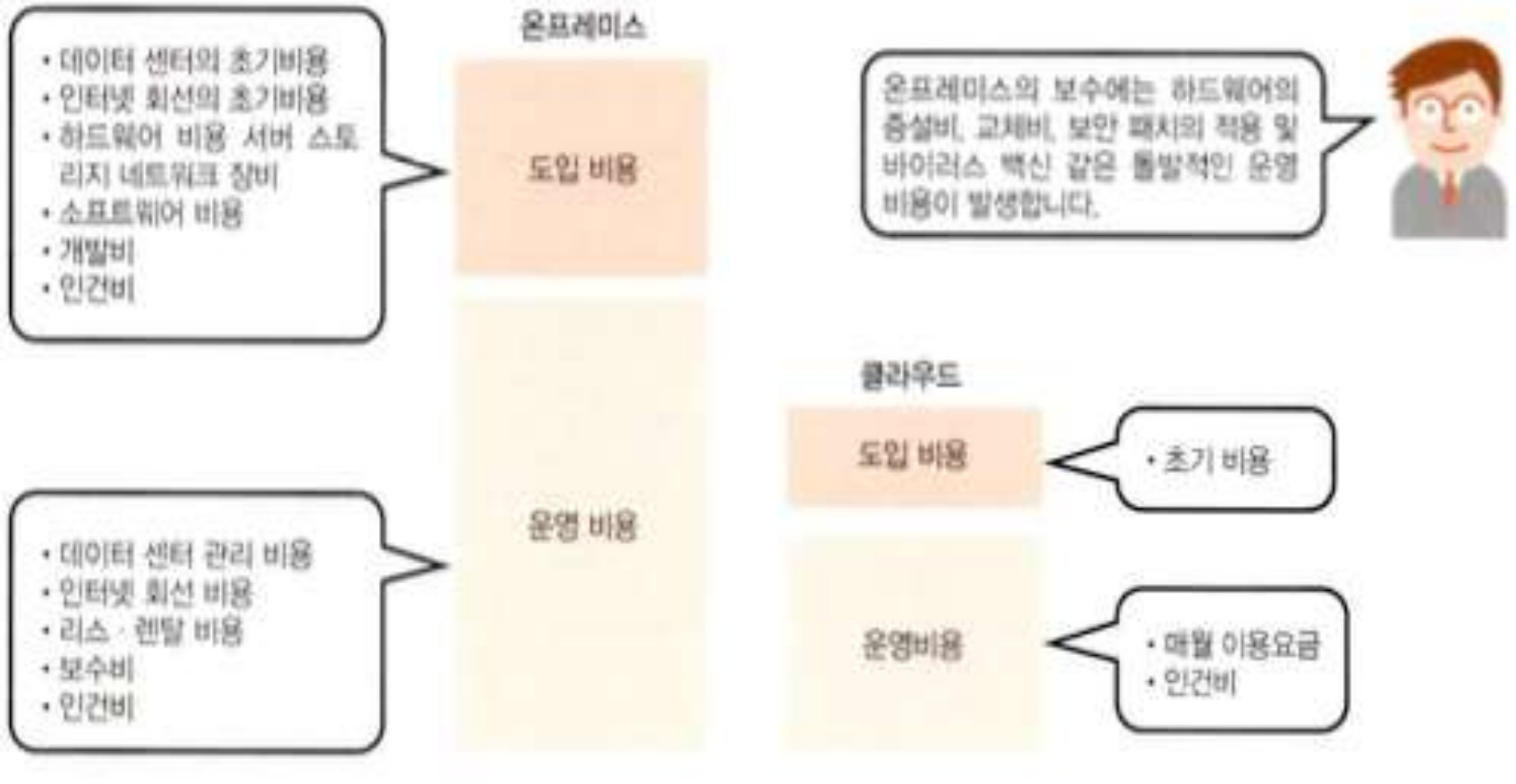
- Cloud 이용 모델
  - ✓ Multi Cloud
    - 멀티 클라우드는 조직이 2개 이상의 클라우드 제공업체의 클라우드 컴퓨팅 서비스를 사용하여 애플리케이션을 실행하는 것
    - 멀티 클라우드 환경은 단일 클라우드 스택을 사용하는 대신 일반적으로 2개 이상의 퍼블릭 클라우드, 2개 이상의 프라이빗 클라우드 또는 둘의 조합을 포함
    - 여러 공급업체를 활용하는 전략을 자유롭게 수립하여 특정 비즈니스 요구에 가장 적합한 기능을 선택하고 공급업체 종속을 최소화할 수 있음
    - 점점 더 많은 조직이 멀티 클라우드 전략과 멀티 클라우드 솔루션을 채택하여 복잡성 없이 필요한 곳에서 애플리케이션을 실행할 수 있도록 하고 있음
    - Kubernetes와 같은 오픈 소스 기술을 기반으로 구축된 멀티 클라우드 솔루션은 다양한 클라우드 및 컴퓨팅 환경에서 애플리케이션을 마이그레이션, 빌드, 최적화하는 유연성과 이동성을 제공
    - 멀티 클라우드 환경은 DevOps 개발 관행 및 컨테이너 와 마이크로서비스 아키텍처와 같이 이동성을 지원하는 기타 클라우드 기반 애플리케이션 기술과 함께 잘 작동

Cloud

- ❖ Cloud & On-Premise
  - ✓ 비용 비교
    - On-Premise 시스템의 경우 도입 시에 데이터 센터 등의 시설 비용과 서버, 스토리지, 네트워크 장비 등의 하드웨어 와 소프트웨어 조달 비용이 필요
    - 몇 년 후의 이용량을 예측해서 제품을 선택해야 하며 거액의 초기 도입 비용을 지불해야하고 도입 후 운영에는 시설 관리비와 하드웨어 리스 비용 및 임대 비용, 유지 보수 비용, 회선 비용, 운용 담당자의 인건비 같은 비용이 소요
    - Cloud의 경우 계약 기간을 걱정하지 않고 사용할 용도나 사용 빈도에 따라 필요할 때 필요한 만큼 가상 서버 같은 자원을 유연하게 사용할 수 있으므로 비용 최적화를 꾀할 수 있으며 운영 비용도 월 정액제로 사용할 수 있으므로 평준화 시킬 수 있고 시스템을 Cloud 사업자가 운용하므로 운용에 소요되는 인건비도 크게 줄일 수 있음
    - Cloud 사업자 간의 가격 경쟁이 거세게 진행되고 있으며 서비스의 가격 인하가 진행되고 있다는 점도 이용자에게는 이득
    - 사용자가 5년 이상의 장기간에 걸쳐 시스템을 계속 사용하는 경우 혹은 대규모 시스템을 구축하고 운용할 경우에는 Cloud보다 On-Premise가 저렴한 경우도 있고 Cloud 구축 시 기존 On-Premise 시스템의 수정이나 데이터 이행에 소요되는 비용이 많이 발생하는 경우도 있기 때문에 이미 안정적으로 장기간 실행되고 있는 시스템이라면 Cloud로 마이그레이션 할 때의 비용이 오히려 비쌀 수 있음
    - 시스템의 이용 상황과 갱신 시기를 바탕으로 Cloud로의 전환이 가져오는 비용 절감 여부를 검토할 필요가 있을 것

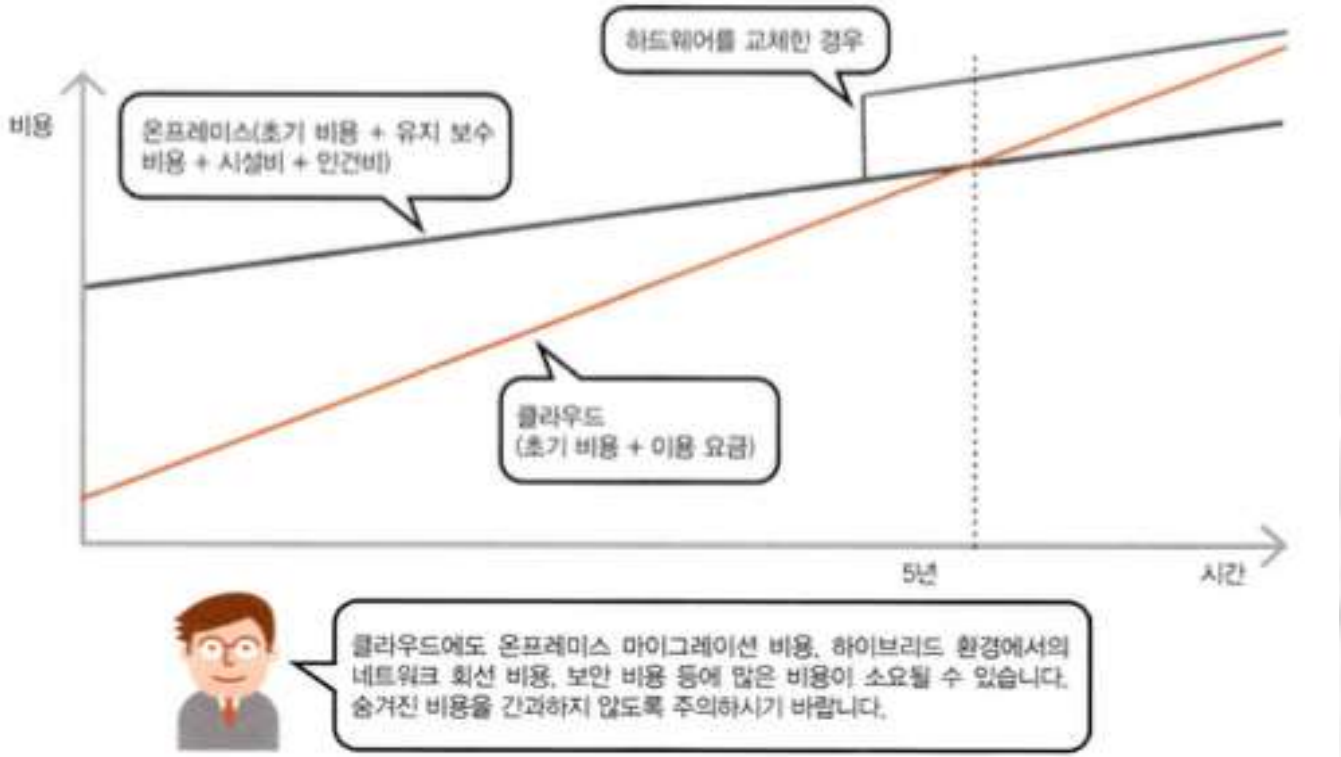
Cloud

- ❖ Cloud & On-Premise
  - ✓ 비용 비교
    - ● 비용 구조



Cloud

- ❖ Cloud & On-Premise
  - ✓ 비용 비교
    - ● 기간에 따른 비용 구조



Cloud

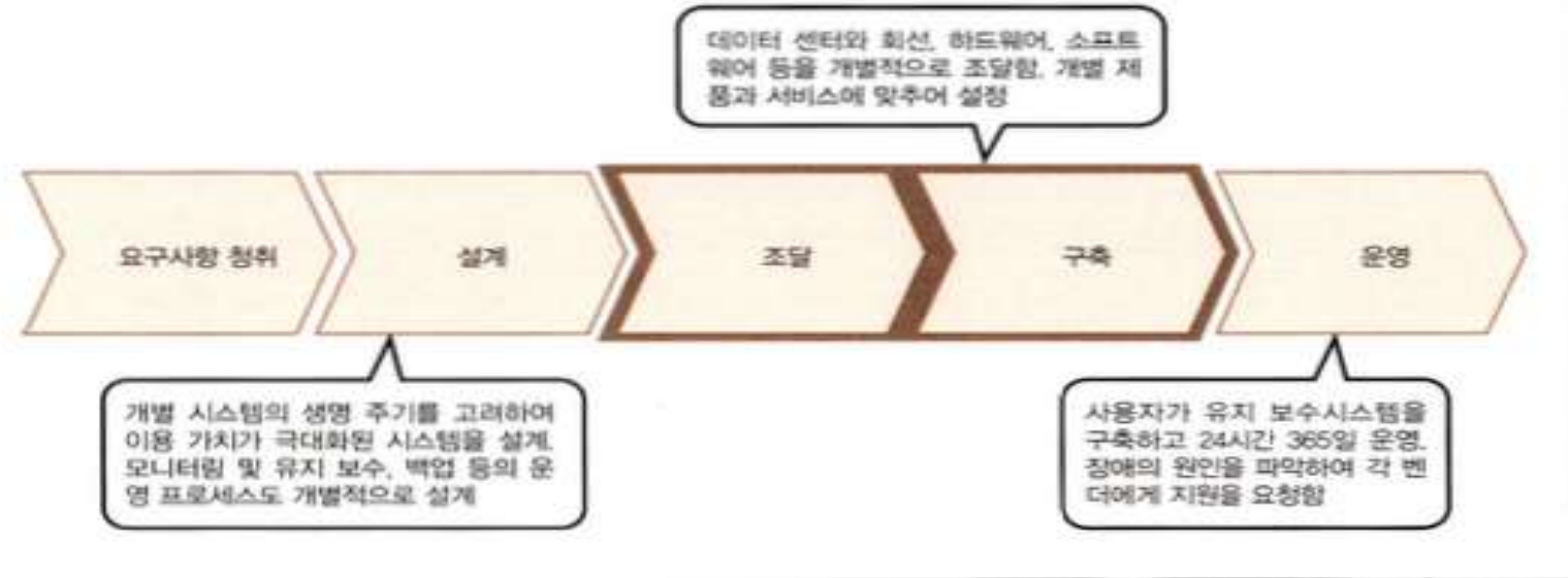
- ❖ Cloud & On-Premise
  - ✓ 비용 비교
    - ● CapEx에서 OpEx로
      - ◆ CapEx(Capital Expenditure)
        - ▪ IT 자원을 도입하려면 장비를 먼저 구입하고 설치하고 몇 년 동안 감가상각 하며 사용하는 방식
        - ▪ 한 번에 큰돈을 쓰고 오랜 기간 사용하며 가치를 뽑는 방식
        - ▪ 계획적이고 예측 가능하다는 장점이 있는데 일정 기간 동안 사용량을 예측하고 여유 자원을 확보해서 안정적인 운영을 추가
        - ▪ 유연성이 떨어지고 초기 비용이 크다는 단점이 존재
      - ◆ OpEx(Operational Expenditure)
        - ▪ 필요할 때 빌리고 사용하고 사용이 끝나면 반납할 수 있는 구조
        - ▪ 먼저 작게 시작하고 필요에 따라 늘려가자는 방식

Cloud

- ❖ Cloud & On-Premise
  - ✓ 확장성 비교
    - ● On-Premise 시스템의 도입과 확장성
      - ◆ 기업 사용자가 On-Premise 시스템을 도입하는 경우에는 자사 시스템의 개별 요구 사항에 따라 설계
      - ◆ 시스템의 조달 및 구축 시 데이터 센터는 A사에 회선은 B사에 네트워크 장비는 C사에 의뢰하는 등 각각의 벤더로부터 개별적으로 조달하고 구축할 수 있음
      - ◆ 시스템의 기획부터 개발, 운영 효과의 측정, 기능 확장, 재구축까지 상시 이용 가치의 극대화 및 시스템의 최적화를 꾀하는데 이 모두를 자사에서 실시하기 때문에 시스템의 도입 과 확장에는 고도의 기술력을 갖춘 인재와 큰 비용이 필요
      - ◆ 도입 후의 하드웨어 및 소프트웨어 자산의 관리, 모니터링 및 데이터 백업 시스템의 운영, 서버 실의 보안 관리 및 시스템의 보안 대응도 필요

Cloud

- ❖ Cloud & On-Premise
  - ✓ 확장성 비교
    - ● On-Premise 시스템의 도입과 확장성

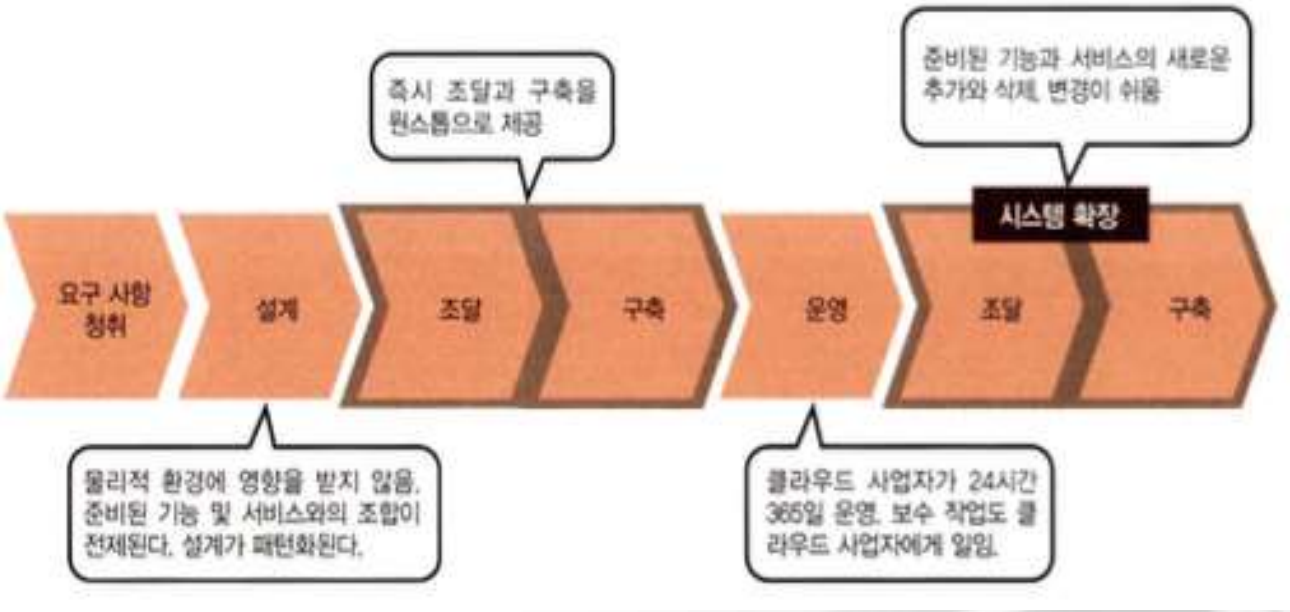


Cloud

- ❖ Cloud & On-Premise
  - ✓ 확장성 비교
    - ● Cloud의 도입과 확장성
      - ◆ 기업 사용자가 Cloud 서비스에 기반한 시스템을 도입할 경우 Cloud 사업자가 제공하는 서비스를 이용하므로 서비스에 관련된 조달에서 구축까지 Cloud 사업자가 원스톱으로 제공
      - ◆ 서비스의 유지 보수 또한 Cloud 사업자가 24시간 365일 대응
      - ◆ 기업 사용자는 사업의 확대와 이용 용도에 맞추어 가상 서버와 같은 리소스들을 확장하거나 축소할 수 있고 Cloud 사업자에게 구축 및 운영을 위탁하므로 시스템을 보유하는 리스크로부터 해방되며 지금까지 운영에 할당했던 인재들을 보다 비즈니스에 가까운 업무로 배치할 수 있음
      - ◆ 막대한 IT 투자가 어려운 중소기업의 경우에도 Cloud를 이용하여 대기업에 필적하는 규모의 서비스를 이용할 수 있음

Cloud

- ❖ Cloud & On-Premise
  - ✓ 확장성 비교
    - ● Cloud의 도입과 확장성



Cloud

- ❖ Cloud & On-Premise
  - ✓ 안정성 과 신뢰성
    - ● Cloud의 리스크
      - ◆ IaaS의 리스크로는 Cloud 사업자의 하드웨어 장애로 인한 데이터 손실이나 서비스 중단 등을 들 수 있고 네트워크 리스크로 통신 도청, 중간자 공격(통신 경로 사이에서 공격), 스푸핑 같은 통신 위협과 네트워크 관리 미비에 따른 시스템 다운 위협 등을 꼽을 수 있음
      - ◆ 사용자가 세울 수 있는 대책으로는 IaaS의 장애에 대비하여 가상 서버를 백업하거나 쉽게 가상 서버 환경을 구축하기 위한 템플릿을 마련하는 것

Cloud

- ❖ Cloud & On-Premise
  - ✓ 안정성 과 신뢰성
    - ● Cloud의 리스크

클라우드 서비스는 클라우드 사업자의 데이터 센터에 설치된 대량의 물리 서버, 물리 스토리지 물리 네트워크 위에 구축되어 있습니다. 그 장비들의 고장 리스크가 있습니다. 재해나 운영자의 조작 실수 같은 리스크도 있습니다.

클라우드와 사용자 거점을 연결하는 네트워크가 다운될 리스크, 악의적인 공격자가 통신을 도청하는 리스크, 중간자 공격, 스푸핑 같은 공격을 받을 리스크도 있습니다.

기업 사용자는 가상 서버 및 스토리지를 백업하거나 안전한 통신망을 이용하여 위험에 대비해야 합니다.

Cloud

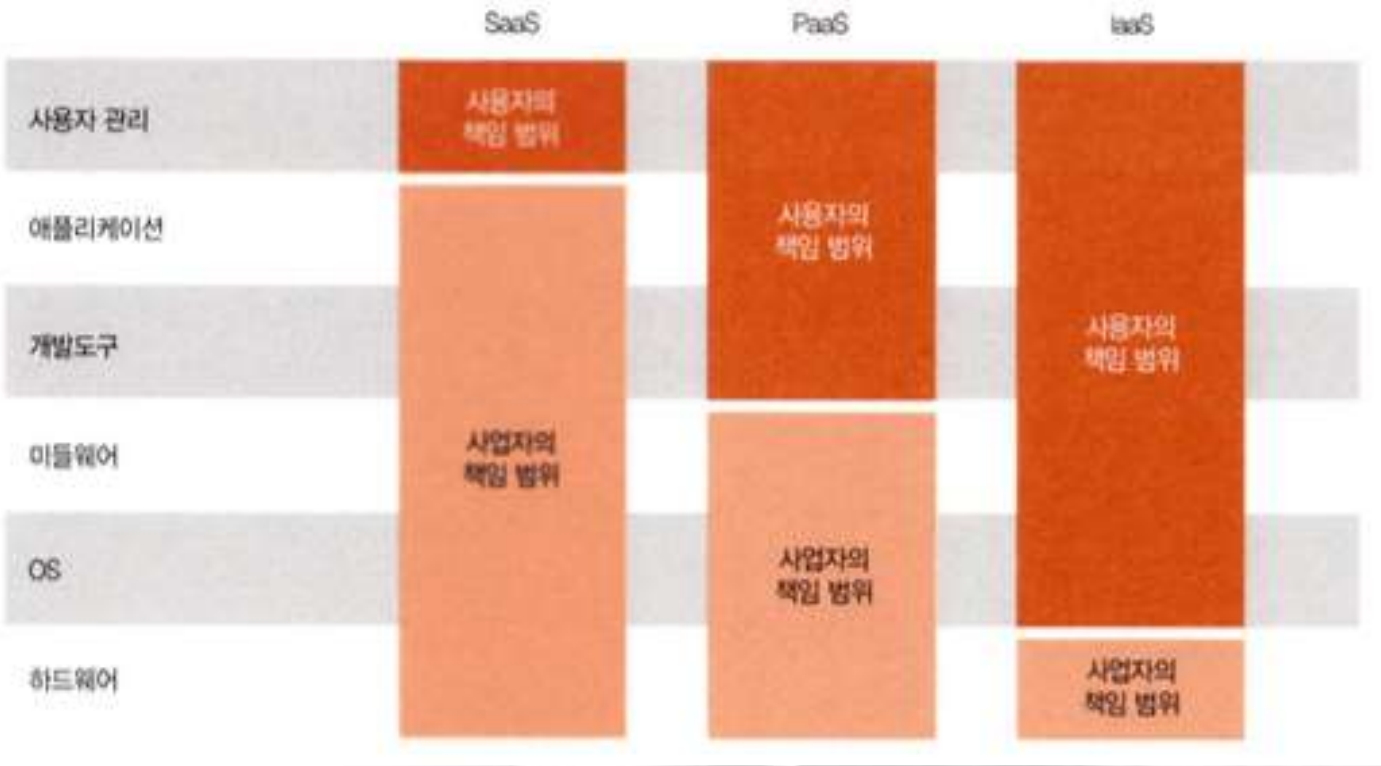
- ❖ Cloud & On-Premise
  - ✓ 안정성 과 신뢰성
    - ● Cloud 보안 거버넌스
      - ◆ Cloud를 이용함으로 인해 기업 사용자는 자사가 보유한 정보의 관리와 처리를 Cloud 사업자에게 맡겨 버리기 때문에 보안 등의 리스크를 모두 통제할 수 없다는 문제가 발생할 수 있음
      - ◆ Cloud 보안 거버넌스(Governance, 기업의 경영진이 Cloud를 이용할 때의 위험을 주체적이고 적절하게 관리하기 위한 구조를 구축하고 운용하는 것)의 관점에서 바라보면 Cloud 서비스의 Incident(사고로 이어질 수 있는 사건) 와 서비스 복구와 같은 사항들은 제어가 어렵고 Cloud 사업자의 갑작스런 파산이나 서비스 중단과 같은 Cloud 서비스의 연속성 리스크 또한 존재
      - ◆ 기업 사용자는 Cloud 사업자 및 이용자 측에 잠재된 보안 위험 요소를 확인해 두는 것이 중요하고 Cloud를 이용할 때 Cloud 사업자와 이용자 사이의 책임 분계선 등 이용자가 관리할 수 있는 범위를 파악한 후 보안 대책과 백업을 마련할 필요가 있을 것

Cloud

- ❖ Cloud & On-Premise
  - ✓ 책임 분계선
    - Cloud 서비스의 계약 및 이용에 앞서 Cloud 사업자와 이용자 간의 책임 분담을 명확히 확인해 두는 것이 중요한데 대다수의 Cloud 서비스 계약에 당사의 시설에 연결하는 인터넷 접속 서비스의 결함과 같은 이용자의 접속 환경 장애에 대해 당사는 책임을 지지 않는다고 규정되어 있는데 이처럼 Cloud 사업자는 사용자 기업이 제공한 인터넷 환경과 응용 프로그램이나 데이터베이스 같은 사용자 측의 환경에 의해 발생한 손해는 부담하지 않는다고 규정하고 있는데 Cloud 사업자가 어디까지 책임을 지고 사용자는 어디까지 책임을 져야 하는지는 사전에 서비스 사양을 사용자 스스로 충분히 확인해 두어야 하고 사용자의 책임 범위 안의 사항들은 사용자 자신이 판단하여 조치해야 함
    - Cloud 사업자가 IaaS까지 제공하는 경우라면 하드웨어와 CPU, 스토리지, Cloud 기반 소프트웨어 등의 컴퓨팅 자원까지가 Cloud 사업자의 책임이고 SaaS까지 제공하는 경우에는 미들웨어에서 애플리케이션까지의 부분도 Cloud 사업자의 책임 범위가 되는데 이처럼 Cloud 서비스 이용 모델에 따라 책임 분계선이 달라지며 PaaS와 IaaS를 함께 이용할 경우는 더 복잡
    - IaaS를 사용하는 경우 사용자 측에서 준비한 미들웨어 및 애플리케이션의 보안과 애플리케이션 가용성 확보, 데이터 보호 등은 사용자의 책임이 되고 사고가 발생한 경우에 Cloud 사업자가 어떻게 정보를 제공하는지 지원 체제는 어떻게 되어 있는지 등을 보안 사고에 대비하여 확인하고 책임 범위를 확인해서 체계화시켜야 하고 Cloud 사업자와의 원활한 커뮤니케이션 체계를 마련해야 함

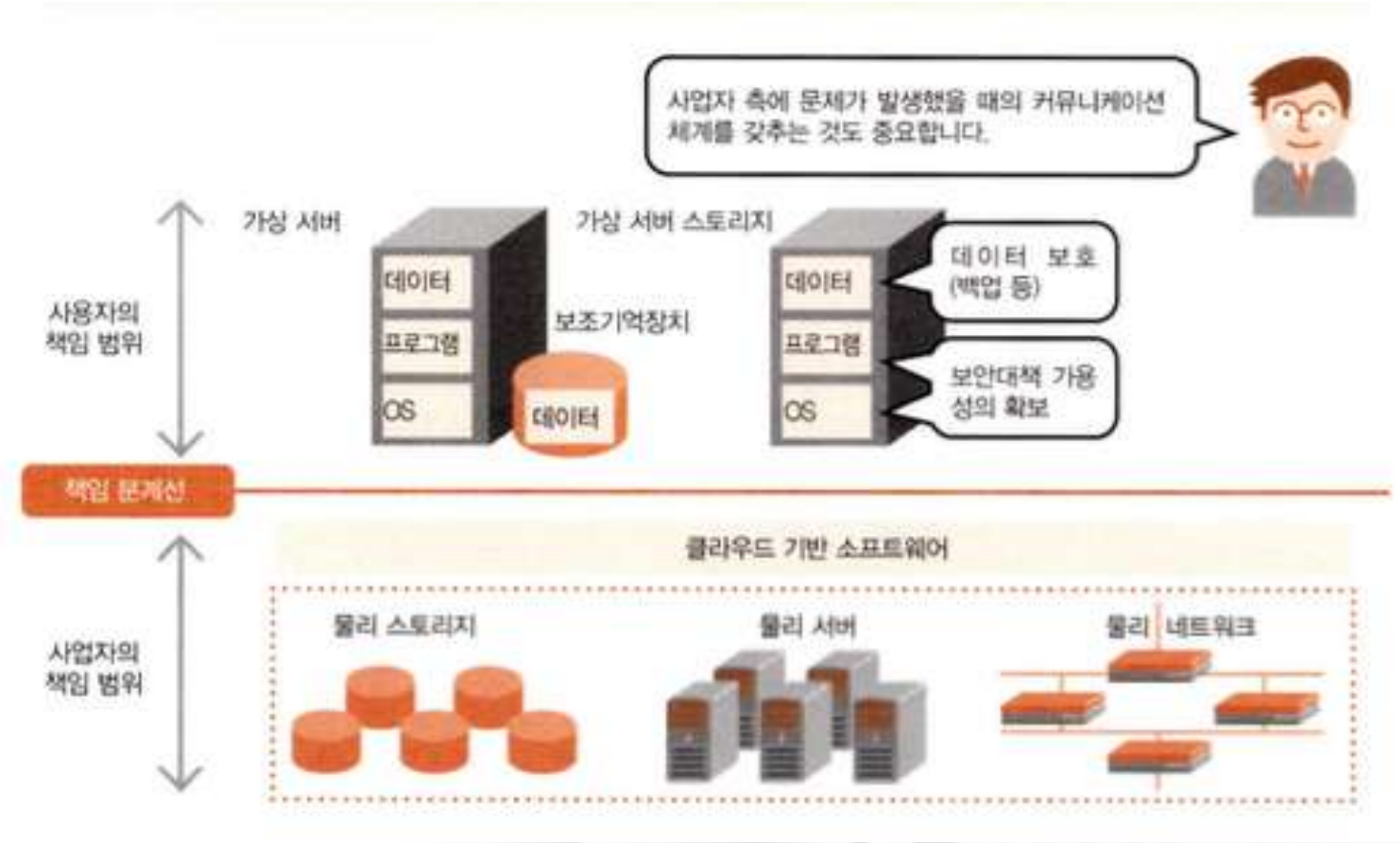
Cloud

- ❖ Cloud & On-Premise
  - ✓ 책임 분계선
    - ● 책임 범위



Cloud

- ❖ Cloud & On-Premise
  - ✓ 책임 분계선
    - ● 책임 범위



Cloud

- Cloud 도입
  - ✓ 장점
    - 경제성(Reduced Cost – Pay-per-use, Economics of Scale): 자체 시스템을 구축할 경우 피크 타임의 이용량을 예상하여 그 만큼의 하드웨어 와 소프트웨어를 구입해야만 하는 반면 Cloud의 경우에는 하드웨어와 소프트웨어를 소유하지 않고 사용하고자 하는 기능을 사용하고자 하는 기간만 서비스로써 사용할 수 있고 부서 나 사업소 등이 각자 관리하던 소프트웨어 와 데이터를 Cloud에서 통합 관리함으로써 소프트웨어 업데이트 작업 및 데이터 유지보수의 효율성을 높이고 비용을 절약할 수 있음
    - 자동화: 업데이트, 보안 패치, 백업 등
    - 이동성: 웹을 지원할 수 있는 모든 장치에서 사용 가능
    - 자원 공유
    - 빠른 구축 속도: 자체적으로 시스템을 구축할 경우 설계 후 하드웨어 와 소프트웨어를 조달하여 배치하기까지 시간이 걸리지만 Cloud의 경우 Cloud가 제공하는 하드웨어 와 소프트웨어를 이용하여 시스템을 신속하게 구축할 수 있음

Cloud

❖ Cloud 도입

✓ 장점

Cloud

❖ Cloud Computing 구조

클라우드에 필요한 기술

- ❖ 가상화
  - ✓ 개요
    - 가상화는 서버, 스토리지, 네트워크 및 기타 물리적 시스템에 대한 가상 표현을 생성하는 데 사용할 수 있는(추상화) 기술
    - 하나의 물리적인 컴퓨터를 마치 여러 대의 독립된 컴퓨터처럼 사용하는 기술
    - 가상화 소프트웨어는 물리적 하드웨어 기능을 모방하여 하나의 물리적인 컴퓨터에서 여러 가상 시스템을 동시에 실행
    - 기업은 가상화를 사용해 하드웨어 리소스를 효율적으로 사용하여 투자 대비 이익을 더 많이 얻을 수 있고 클라우드 컴퓨팅 서비스를 지원하여 조직의 인프라를 더욱 효율적으로 관리할 수 있음
    - 가상화에 대한 접근은 OS, 서버, 스토리지, 네트워크에서도 동일한 모습으로 실행되며 개별 하드웨어 관련 노하우가 소프트웨어로 점차 도입되고 있음
    - DevOps 지원 기술에서는 인프라의 전문성을 배제하는 개발 측면에서 인프라 구성에 접근할 수 있도록 가상화의 흐름을 맞이하고 있음

클라우드에 필요한 기술

- 가상화
  - ✓ 가상화 이점
    - 효율적인 리소스 사용
    - 자동화된 IT 관리
    - 신속한 재해 복구
  - ✓ 가상화 서비스
    - 운영체제 가상화
    - 서버 가상화
    - 스토리지 가상화
    - 네트워크 가상화
    - 데이터 가상화
    - 애플리케이션 가상화
    - 데스크톱 가상화

클라우드에 필요한 기술

- ❖ 가상화
  - ✓ 서비스
    - ● OS 가상화
      - ◆ OS 가상화는 하나의 OS 위에서 여러 사람이 서로 다른 어플리케이션을 만들고 싶다는 의도로 시작되었는데 1979년 파일 시스템(File System) 분리가 가능한 chroot가 등장한 것이 그 원형으로 파일 시스템 분리에 머물렀던 chroot에 반해서 2000년 FreeBSD 라는 OS에서 Jail이라는 방안이 생겨나고 일부 시스템의 변경 권한(root 권한)을 분리할 수 있게 됨
      - ◆ 2001년에는 Jail 방안을 기반으로 한 Linux-VServer 개발이 시작되어 하나의 서버에서 CPU 시간 과 메모리 및 네트워크 등 자원(Resource)을 분할하여 복수의 Linux 가상 서버를 작동시키는 방안이 생겨났으며 2005년에는 Solaris Containers가 등장하였고 2008년에는 Linux에도 LXC(The Linux Containers Project)가 생겨나 개별 프로세스 단위로 네임 스페이스(Name Space)에 의한 자원 분리를 실시함으로써 가상화를 실현하는 컨테이너라는 개념이 퍼짐
      - ◆ 2013년 dotCloud(현재는 Docker)에서 LXC를 이용하여 구현된 컨테이너인 Docker가 오픈 소스화 되어 폭발적으로 보급되었으며 Docker는 Go 언어로 구현되어 있는데 플랫폼 간 이식성(Portabillity)을 향상시키고 있음
      - ◆ OS 가상화는 계속 진화되고 있음
      - ◆ OS 가상화는 DevOps에 있어서 개발과 운용의 긴밀한 연계를 위해 인프라 구성을 소프트웨어로 끌어모아 관리를 일원화하거나 구성을 변경할 때 사용하는 Infrastructure as Code에서도 가장 많이 이용되는 기술 중 하나

클라우드에 필요한 기술

- ❖ 가상화
  - ✓ 서비스
    - ● OS 가상화
      - ◆ OS 가상화 방식

클라우드에 필요한 기술

- ❖ 가상화
  - ✓ 서비스
    - ● 서버 가상화
      - ◆ PC 위에 가상 PC를 만드는 컨셉으로 1999년에 VMware에서 VMware 1.0이 등장하면서 모든 물리적 하드웨어를 소프트웨어로 표현하고 OS 위에서 완전히 다른 OS를 구동시킬 수 있게 됨
      - ◆ 가상화된 하드웨어를 가상 머신, 가상 머신을 탑재한 원래의 OS를 가상화 호스트라고 부르는데 Hypervisor(Hypervisor)라고도 부름
      - ◆ Linux나 Windows와 같은 OS를 설치한 후에 Hypervisor 소프트웨어를 넣는 경우와 물리 하드웨어 위에 Hypervisor를 넣는 베어메탈 Hypervisor 방법이 있음
      - ◆ 2003년에는 Xen5이라는 리눅스 커널(Linux Kernel)을 Hypervisor화 하는 기술이 등장했고 2006년에는 KVM이라는 가상화 기술이 등장했으며 이듬해 2007년에는 리눅스 커널에 KVM이 통합되어 리눅스 디스트리뷰션(Linux Distribution)에서는 표준적인 서버 가상화를 도입할 수 있게 됨
      - ◆ 2009년에는 VMware vSphere가 등장해서 가벼운 가상화 전용 OS(베어메탈 Hypervisor)로 크게 주목을 받았고 실제 대기업에도 이용됨

클라우드에 필요한 기술

- 가상화
  - ✓ 서비스
    - 서버 가상화
      - OS 가상화와 서버 가상화 양쪽의 장점을 얻기 위해서 가상 머신 위에 컨테이너를 구성하는 하이브리드 구성이 주목을 받고 있는데 Google이 2015년에 Borg라는 컨테이너형 아키텍처를 발표했는데 하드웨어 가상화인 가상 머신 위에 컨테이너를 전개함으로써 하드웨어 자원에 대한 제약 사항에 신경 쓰지 않고 컨테이너에 의해서 하드웨어를 최대치까지 사용하는 것이 가능
      - 서버 가상화로 인해 보다 물리적 제약에서 벗어나서 서버 및 인프라 구성을 설계할 수 있게 됨
      - 오늘날에는 물리적 하드웨어에 대한 고려는 크게 줄었으며 인프라 담당자 밖에 모르는 지식 또한 점점 줄어들고 있음

클라우드에 필요한 기술

- ❖ 가상화
  - ✓ 서비스
    - ● 스토리지 가상화
      - ◆ 하나의 물리 스토리지에 논리 스토리지를 여러 개 생성하거나 정책에 기반하여 동작하는 내용을 배포 및 결정할 수 있는 스토리지 기기를 SDS(SoftWare Defined Storage)라고 부름
      - ◆ EMC와 NetApp 등 스토리지 기업에서 많은 스토리지용 OS가 소프트웨어로 제어할 수 있도록 API(Application Programming Interface)를 두고 있어 보다 소프트웨어와 가까워지고 있음
      - ◆ 스토리지는 전문 엔지니어가 각 서비스에 대한 용량을 계산하고 데이터 배치를 결정하거나 디스크 증설 계획을 세우는 등 개발 부문에서는 인프라 담당자가 무엇을 하는지 전혀 모르는 상태였지만 SDS로 가상화된 스토리지를 소프트웨어로 제어할 수 있게 되어 보다 개발과 운용이 긴밀하게 연계될 수 있는 환경이 됨

클라우드에 필요한 기술

- ❖ 가상화
  - ✓ 서비스
    - ● 네트워크 가상화
      - ◆ 네트워크 가상화의 수단으로써 이전부터 알려진 것이 VLAN(Virtual Local Area Network)인데 1994년에 구축된 것이 시초이며 이는 하나의 물리 스위치를 복수의 논리적인 스위치로 나누기 위한 기술
      - ◆ 대규모 네트워크의 유연한 변경에 대한 수요가 커졌으며 앞선 문제를 해결하기 위한 수단으로써 SDN(Software Defined Network)이 제창됨
      - ◆ SDN의 구현 방법은 몇 가지가 있지만 공통적인 것은 스위치를 Control Plane(설정 등의 관리 부문) 과 Data Plane(패킷 전송 부문)으로 분리하는 것으로 정책이나 설정 내용의 제어를 담당하는 Control Plane은 소프트웨어에 의해서 관리됨
      - ◆ 개발 부문은 낮은 학습 비용으로 소프트웨어 지식을 구사하면서 네트워크 설정을 변경할 수 있음
      - ◆ 방화벽(Firewall) 및 부하 분산(Load balancing)의 모습은 이전에는 전문 엔지니어만 만질 수 있었지만 소프트웨어에 의한 제어로 이동함에 따라 인프라 담당자뿐만이 아니라 개발 부문도 내용을 파악하거나 변경할 수 있게 되었음

클라우드에 필요한 기술

- ❖ 가상화
  - ✓ 서비스
    - ● 데스크톱 가상화
      - ◆ 데스크톱 가상화(VDI, Virtual Desktop Infrastructure)는 내 컴퓨터의 윈도우나 맥 화면을 내 모니터가 아니라 저 멀리 있는 기업의 거대한 서버(데이터 센터)에서 통째로 띄워서 사용하는 기술
      - ◆ 사용자는 눈앞에 있는 모니터, 키보드, 마우스를 이용하지만 실제로 모든 프로그램이 실행되고 데이터가 저장되는 곳은 중앙 서버
      - ◆ 서버 가상화는 서비스 제공이 목적이고 데스크톱 가상화는 사용자 작업 환경을 위한 것이 목적
      - ◆ 서버 가상화는 대부분 CLI 환경을 사용하지만 데스크톱 가상화는 GUI 환경을 많이 사용하기 때문에 그래픽 처리와 네트워크 지연에 민감해서 데스크톱 가상화 환경에서는 GPU 가상화 기술이나 저지연 네트워크 구성 같은 추가적인 고려가 필요

클라우드에 필요한 기술

- ❖ 가상화
  - ✓ 장단점

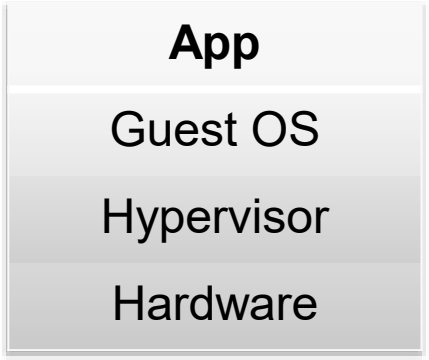
구분	장점	단점
자원 활용	하나의 서버에 여러 가상 머신을 실행하여 자원을 효율적으로 사용	여러 가상 머신이 자원을 공유하므로 과도한 사용 시 자원 간섭 발생 가능
비용	물리 장비 수 감소 -> 전력, 공간, 인건비 등 운영비 절감	일부 소프트웨어는 가상 환경에서 라이선스 비용 증가 가능성
배포 속도	가상 머신 생성 및 복제가 빠르고 스냅샷으로 쉽게 복구 가능	다수의 가상 머신을 운영하고 모니터링 해야 하므로 관리 복잡성 증가
보안 및 격리	각 가상 머신 간 격리로 장애 및 보안 문제 전파 차단	Hypervisor가 공격받을 경우 전체 시스템이 위협 받을 수 있음
운영 유연성	테스트 환경을 손쉽게 구성하고 폐기 가능하며 개발 및 운영 민첩성 향상	고성능이 필요한 작업에는 물리 서버 대비 성능 저하 가능

클라우드에 필요한 기술

- 가상화 방식
  - ✓ Hypervisor 가상화
    - Hypervisor
      - Hypervisor는 하나의 물리적 컴퓨터(호스트)에서 여러 개의 가상 머신(VM, Virtual Machine)을 동시에 실행하고 관리할 수 있게 해주는 소프트웨어
      - 한 대의 컴퓨터 안에 여러 대의 독립된 컴퓨터가 컴퓨터인 척 작동할 수 있도록 하드웨어 자원(CPU, 메모리, 저장공간 등)을 쪼개고 나누어 주는 가상화 교통정리원
      - Hypervisor는 물리적 하드웨어에 어떻게 설치되고 작동하느냐에 따라 크게 타입 1(Type 1 - 베어메탈)과 타입 2(Type 2)로 나눔

클라우드에 필요한 기술

- ❖ 가상화 방식
  - ✓ Hypervisor 가상화
    - ● Hypervisor 구현 방식
      - ◆ Type1: 네이티브 / 베어메탈 (Bare-Metal) Hypervisor
        - ▪ 하드웨어(컴퓨터) 위에 중간 운영체제(Windows나 Linux 등) 없이 Hypervisor가 직접 설치되는 방식
        - ▪ 하드웨어 바로 위에 설치된다고 하여 베어메탈(Bare Metal)이라고 부름
        - ▪ 특징
          - 중간에 거치는 운영체제가 없기 때문에 자원 낭비가 적고 성능이 매우 뛰어남
          - 하드웨어를 직접 제어하므로 안정성과 보안성이 높아 기업용 서버 환경에서 주로 사용
          - 자체적으로 머신에 대한 관리 기능이 없기 때문에 관리를 위한 컴퓨터나 콘솔(CLI)이 필요함

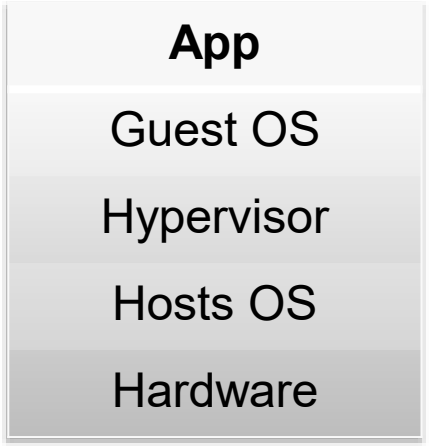


클라우드에 필요한 기술

- ❖ 가상화 방식
  - ✓ Hypervisor 가상화
    - ● Hypervisor 구현 방식
      - ◆ Type1: 네이티브 / 베어메탈 (Bare-Metal) Hypervisor
        - ▪ 대표적인 종류:
          - VMware ESXi: 전 세계 기업형 가상화 시장에서 가장 널리 쓰이는 표준적인 Hypervisor
          - Microsoft Hyper-V: 윈도우 서버 환경에 최적화된 마이크로소프트의 기술
          - KVM(Kernel-based Virtual Machine): 리눅스 커널을 Hypervisor로 변환하여 사용하는 오픈소스 기술로 AWS 등 대형 클라우드 서비스의 기반
          - Xen: 보안과 성능이 뛰어난 오픈소스 기반
          - Oracle VM Server: Xen 기반의 Oracle Hypervisor
          - Citrix Hypervisor(ZenServer): 오픈소스 기반의 강력한 엔터프라이즈 가상화 솔루션

클라우드에 필요한 기술

- ❖ 가상화 방식
  - ✓ Hypervisor 가상화
    - ● Hypervisor 구현 방식
      - ◆ Type2: 호스트형 (Hosted) Hypervisor
        - ▪ 물리적 하드웨어 위에 설치된 호스트 운영체제 위에 가상화 소프트웨어 와 가상 머신에 설치된 운영체제(게스트 운영체제)를 움직이는 방식
        - ▪ 특징
          - 일반 앱처럼 쉽게 설치하고 지울 수 있어서 사용법이 매우 직관적이고 편리
          - 하드웨어 자원을 쓰려면 [가상머신 -> Hypervisor -> 기존 운영체제 -> 하드웨어]의 단계를 거쳐야 하므로 타입 1에 비해 오버헤드(성능 저하)가 발생
          - 개발자 테스트용이나 개인 실습용으로 주로 사용
          - 가상의 하드웨어를 Emulating 하기 때문에 호스트 운영체제에 크게 제약 사항이 없음



클라우드에 필요한 기술

- ❖ 가상화 방식
  - ✓ Hypervisor 가상화
    - ● Hypervisor 구현 방식
      - ◆ Type2: 호스트형 (Hosted) Hypervisor
        - ▪ 가상화 소프트웨어
          - VMware Workstation(Pro/Player/Fusion): 강력한 성능을 보여주는 유료/무료 도구
          - Oracle VirtualBox: 무료이면서 Windows, Mac, Linux를 모두 지원해 개인 사용자와 개발자 실습용으로 가장 인기가 많음
          - Microsoft의 Virtual PC & Hyper-V Client: Windows에서 활성 가능한 가상화 소프트웨어
          - Parallels Desktop: 맥(macOS)에서 윈도우나 리눅스를 돌릴 때 사용하는 대표적인 프로그램
          - QEMU: 오픈소스로 가상화 및 에뮬레이션 기능 모두 지원

클라우드에 필요한 기술

- 가상화 방식
  - ✓ Hypervisor 가상화
    - ● 구성 요소
      - ◆ CPU
        - ▪ Hypervisor가 물리 CPU 자원을 여러 가상 CPU라고 부르는 vCPU(Virtual CPU)로 나눠서 각 가상 머신에 할당
        - ▪ Core 단위로 할당
      - ◆ Memory
        - ▪ Hypervisor가 실제 메모리를 일정 크기만큼 잘라서 할당
        - ▪ 실제로는 모두 같은 물리 메모리를 사용하지만 서로 간섭하지 않도록 격리되어 있음
      - ◆ Disk
        - ▪ 진짜 하드디스크가 아니고 하나의 파일로 구성
        - ▪ 하나의 파일에 운영체제를 설치하고 애플리케이션 과 데이터도 저장
        - ▪ Hypervisor가 이 파일을 하드디스크처럼 인식하도록 도와줌
      - ◆ 네트워크 어댑터: 가상 머신마다 독립된 네트워크 어댑터를 만들어주고 이를 통해 다른 가상 머신이나 외부 네트워크와 통신할 수 있도록 함

클라우드에 필요한 기술

- ❖ 가상화 방식
  - ✓ Hypervisor 가상화
    - ● Snapshop
      - ◆ 가상머신(VM)에서 스냅샷(Snapshot)은 컴퓨터 상태를 그대로 저장해 두는 파일
      - ◆ 스냅샷이 저장하는 것
        - ▪ 가상 디스크 상태: 그 순간 저장되어 있던 모든 파일과 데이터
        - ▪ 메모리(RAM) 상태: 현재 켜져 있는 프로그램, 작업 중이던 창, 인터넷 브라우저에 띄워둔 탭까지 그대로 기억
        - ▪ 하드웨어 설정: CPU 개수, 메모리 크기, 네트워크 설정 등 가상 머신의 스펙 정보
      - ◆ 동작 원리
        - ▪ 스냅샷은 가상머신의 전체 디스크를 매번 통째로 복사하는 것이 아니고 차이점만 기록하는 방식(Copy-on-Write)으로 작동
        - ▪ 기본 상태 (Base): 원래 가상머신의 가상 디스크 파일(A)
        - ▪ 스냅샷 생성: 스냅샷을 찍으면 기존 디스크 파일(A)은 읽기 전용으로 잠기고 그 위에 새롭게 변경되는 데이터만 저장하는 차동 디스크 파일(B)이 새로 만들어 짐
        - ▪ 이후 작업: 스냅샷 이후에 사용자가 파일을 지우거나 설치하는 모든 작업은 새 파일(B)에만 기록됨
        - ▪ 복구(Rollback): 문제가 생겨 되돌리고 싶다면 새 파일(B)을 통째로 날려버리고 읽기 전용으로 잠겨있던 깨끗한 원래 파일(A)로 돌아가는 방식

클라우드에 필요한 기술

- ❖ 가상화 방식
  - ✓ Hypervisor 가상화
    - ● Snapshop
      - ◆ 사용해야 하는 경우
        - ▪ 위험한 작업 전 필수: 윈도우 업데이트, 리눅스 커널 업데이트, 출처가 불분명한 프로그램 테스트, 중요한 시스템 설정 변경 직전
        - ▪ 개발 및 테스트: 악성코드 분석가들이 악성코드를 실행하기 전 스냅샷을 찍고, 분석이 끝나면 다시 깨끗한 상태로 되돌릴 때 필수적으로 사용
        - ▪ 여러 분기 테스트: 하나의 순수한 상태에서 스냅샷을 찍어두고, A 프로그램 설치 버전 / B 프로그램 설치 버전으로 나누어 테스트하고자 하는 경우
      - ◆ 스냅샷 사용 시 주의할 점
        - ▪ 많은 초보자가 하는 실수가 스냅샷을 백업 대용으로 쓰는 것
        - ▪ 스냅샷은 원본 디스크 파일에 의존하므로 만약 원본 디스크 파일 자체가 깨지거나 물리 하드웨어가 고장 나면 스냅샷도 모두 사용할 수 없음 (백업은 다른 저장장치에 파일을 통째로 복사하는 것)
        - ▪ 오래 방치하면 성능 저하: 스냅샷을 찍은 상태로 가상 머신을 계속 쓰면 변경 사항(차동 파일)이 점점 커지게 되서 스냅샷이 누적되면 디스크 읽기/쓰기 속도가 느려지고 컴퓨터 용량을 엄청나게 잡아먹음
        - ▪ 용무가 끝나면 병합(삭제): 테스트가 안전하게 끝났다면, 스냅샷을 삭제하여 변경된 내용을 원본 파일에 합쳐주는(Merge) 것이 좋음 (삭제는 데이터 파기가 아니라 원본에 반영하는 것을 의미)

클라우드에 필요한 기술

- ❖ Container의 개념과 활용
  - ✓ Container 개념
    - 애플리케이션과 그 애플리케이션을 제대로 수행하기 위해 필요한 의존성 요소(라이브러리, 설정 파일, 실행 환경 등)를 한데 묶은 독립적인 실행 단위
    - 가상 머신은 물리적인 하드웨어 위에 운영체제 전체를 가상화한 다음 그 위에 애플리케이션을 올리는 구조이지만 컨테이너는 호스트 운영체제와 직접 연결되며 게스트 운영체제 같은 추가 레이어가 없기 때문에 성능은 향상되고 리소스도 낭비되지 않음
    - Container의 경우 호스트 운영체제의 프로세스 수준에서 격리하는데 Container들이 의존성 요소를 공유하지는 않음
    - Docker 엔진에서는 Container용 Linux 네임스페이스와 컨트롤 그룹을 생성해 이를 처리하는데 Docker의 보안이 Linux 커널 프로세스 격리를 기반으로 하는 이유이기도 하며 이런 솔루션은 충분히 검증됐지만 가상 머신이 제공하는 전체 운영체제 기반 격리보다는 덜 안전하다고 여겨지기도 함
    - 장점
      - ◆ Hypervisor 와 게스트 운영체제가 없기 때문에 가벼움(수십 메가바이트)
      - ◆ 경량이기 때문에 만들어진 Image 복제, 이관, 배포가 쉬움
      - ◆ 게스트 운영체제를 부팅하지 않기 때문에 애플리케이션 시작 시간이 빠름
      - ◆ 가상 머신보다 경량이므로 더 많은 애플리케이션을 실행할 수 있음
    - 단점: 다양한 운영체제를 사용할 수 없고 보안적으로 완전히 격리되지 않음

클라우드에 필요한 기술

- ❖ Container의 개념과 활용
  - ✓ Container 개념

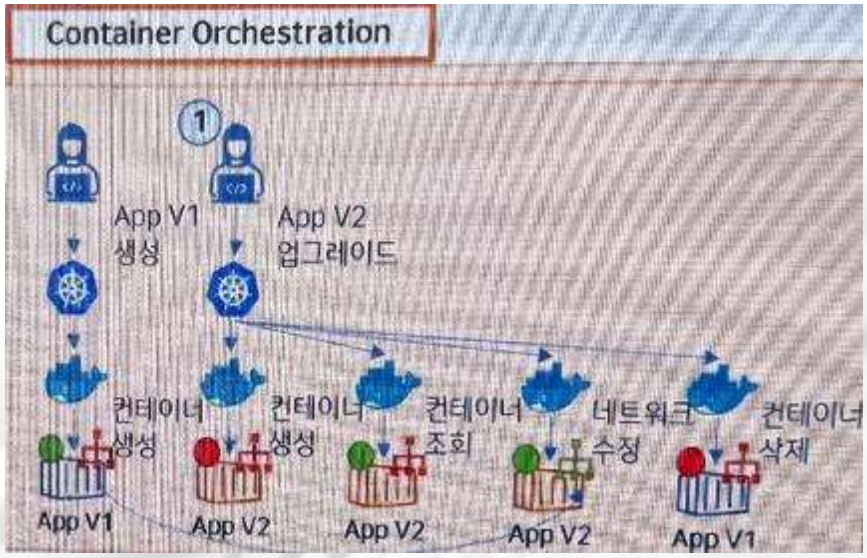
<table>
<tr><td>Virtual Environment</td><td>Virtual Environment</td></tr>
<tr><td>Application</td><td>Application</td></tr>
<tr><td>Middleware</td><td>Middleware</td></tr>
<tr><td colspan="2">Container Management Software</td></tr>
<tr><td colspan="2">OS</td></tr>
<tr><td colspan="2">Hardware</td></tr>
</table>

클라우드에 필요한 기술

- Container의 개념과 활용
  - ✓ Container 런타임: Container를 생성하고 실행할 수 있도록 도와주는 소프트웨어
    - Containerd
      - 컨테이너 실행을 관리하는 오픈 소스 런타임
      - Docker 및 Kubernetes와 같은 컨테이너 오케스트레이션 시스템에서 컨테이너 실행을 처리하는 데 사용
      - Container 이미지를 다운로드 받아 압축을 푸는 과정부터 Container를 실행하고 감독하는 과정까지 Container의 전체 수명 주기를 관리하는 기능을 제공
    - Docker
      - Containerd 위에 설치되는 데몬
      - Container를 생성하고 관리하는데 필요한 모든 기능을 제공
      - Docker는 컨테이너의 전체 라이프사이클(빌드, 배포, 실행)을 관리하는 반면 Containerd는 컨테이너 실행과 이미지 관리를 담당하는 핵심 컴포넌트로 Docker는 내부적으로 Containerd를 사용하며 Kubernetes도 Containerd를 기본 런타임으로 지원
    - CRI-O
      - Redhat, Intel, SUSE, Hyper, IBM 등의 관리자 및 커뮤니티를 중심으로 개발된 오픈 소스 런타임
      - CRI-O는 Docker를 대체하기 위한 툴로 개발

클라우드에 필요한 기술

- ❖ Container의 개념과 활용
  - ✓ Container Orchestration
    - ● 개요
      - ◆ 다수의 Container를 유기적으로 연결 및 실행할 뿐만 아니라 상태를 추적하고 보존하는 등 Container를 안정적으로 사용할 수 있게 만들어주는 솔루션
      - ◆ Container를 효과적으로 관리하도록 도와주는 것이 Container 오케스트레이션
      - ◆ Container 오케스트레이션은 여러 시스템에 Container를 분산해서 배치하거나 문제가 생긴 Container를 교체하는 등 여러 역할을 수행



클라우드에 필요한 기술

- ❖ Container의 개념과 활용
  - ✓ Container Orchestration
    - ● Container 오케스트레이션 솔루션

- ◆ Docker Swarm
  - ▪ 간단하게 설치할 수 있고 사용하기도 용이
  - ▪ 기능이 다양하지 않아 대규모 환경에 적용하려면 사용자 환경을 변경해야 할 수 있기 때문에 소규모 환경에서는 유용하지만 대규모 환경에서는 잘 사용하지 않는 편
- ◆ Mesos
  - ▪ Apache의 오픈 소스 프로젝트로 트위터, AirBnB, 애플, Uber 등 다양한 곳에서 이미 검증된 솔루션
  - ▪ 2016년 DC/OS(Data Center OS, 대규모 서버 환경에서 자원을 유연하게 공유하며 하나의 자원처럼 관리하는 도구)의 지원으로 매우 간결하지만 기능을 충분히 활용하려면 분산 관리 시스템과 연동해야 하기 때문에 여러가지 솔루션을 유기적으로 구성해야 하는 부담이 있음

클라우드에 필요한 기술

- ❖ Container의 개념과 활용
  - ✓ Container Orchestration
    - ● Container 오케스트레이션 솔루션
      - ◆ Nomad
        - ▪ Vagrant를 만든 HashiCorp 사의 Container 오케스트레이션으로 Vagrant처럼 간단한 구성으로 Container 오케스트레이션 환경을 제공
        - ▪ HashiCorp의 Consul(서비스 검색, 구성 및 분할 기능 제공) 과 Vault(암호화 저장소) 와 연동이 원할하므로 이런 도구에 대한 사용 성숙도가 높은 조직이라면 노매드 도입을 고려해볼 수 있음
      - ◆ Kubernetes
        - ▪ 컨테이너화된 애플리케이션을 자동으로 배포, 스케일링 및 운영하는 오픈소스 오케스트레이션 플랫폼
        - ▪ 구글이 내부적으로 사용하던 Borg 시스템에서 발전한 기술로 현재는 Cloud Native Computing Foundation(CNCF)에서 관리

클라우드에 필요한 기술

- ❖ Container의 개념과 활용
  - ✓ Container Orchestration
    - ● Container 오케스트레이션 솔루션

구분	도커 스웜	메소스	노매드	쿠버네티스
설치 난이도	쉬움	매우 어려움	쉬움	어려움
사용 편의성	매우 좋음	좋음	매우 좋음	좋음
세부 설정 지원	거의 없음	있음	거의 없음	다양하게 있음
안정성	매우 안정적임	안정적임	안정적임	매우 안정적임
확장성	어려움	매우 잘 됨	어려움	매우 잘 됨
정보량	많음	적음	적음	매우 많음
에코 파트너	없음	거의 없음	있음	매우 많음
학습 곡선	쉬움	매우 어려움	어려움	어려움

클라우드에 필요한 기술

- ❖ Serverless Computing 과 Event Driven Architecture
  - ✓ Serverless Computing
    - 개요
      - ◆ 서버리스 컴퓨팅(Serverless Computing)은 개발자가 서버를 직접 관리하거나 프로비저닝할 필요 없이 애플리케이션을 빌드하고 실행할 수 있는 클라우드 컴퓨팅 모델
      - ◆ 서버리스(Serverless)라고 해서 서버가 진짜로 없는 것은 아니며 개발자 관점에서 신경 써야 할 서버가 없다는 의미로 서버의 운영, 관리, 확장(Scaling) 등은 모두 클라우드 제공업체(AWS, Google Cloud, Azure 등)가 대신 처리해주는 형태
    - 핵심 특징 3가지
      - ◆ 서버 관리 불필요: 운영체제(OS) 패치, 하드웨어 설정, 네트워크 구성 등 복잡한 인프라 관리 업무를 클라우드 기업이 전담하므로 개발자는 오직 코드 작성에만 집중할 수 있음
      - ◆ 자동 확장(Auto-scaling): 사용자가 전혀 없을 때는 서버가 돌지 않다가 갑자기 수만 명의 요청이 몰리면 클라우드가 알아서 서버 자원을 자동으로 늘려줌(0에서 무한대로 유연한 확장)
      - ◆ 실제 사용량 기반 과금(Pay-as-you-go): 기존 서버처럼 24시간 켜놓은 시간에 따라 돈을 내는 것이 아니라 코드가 실제로 실행된 시간(밀리초 단위)과 요청 횟수에 대해서만 비용을 청구

클라우드에 필요한 기술

- ❖ Serverless Computing 과 Event Driven Architecture
  - ✓ Serverless Computing
    - FaaS (Function as a Service)
      - ◆ 서버리스를 구현하는 가장 대표적인 기술이 FaaS
      - ◆ 애플리케이션을 거대한 하나의 프로그램이 아니라 특정 이벤트(예: 사용자가 이미지를 업로드함)가 발생했을 때만 실행되는 작은 함수(Function) 단위로 쪼개어 등록하는 방식
    - 구현된 서비스
      - ◆ AWS Lambda (가장 대표적인 서비스)
      - ◆ Google Cloud Functions
      - ◆ Microsoft Azure Functions

클라우드에 필요한 기술

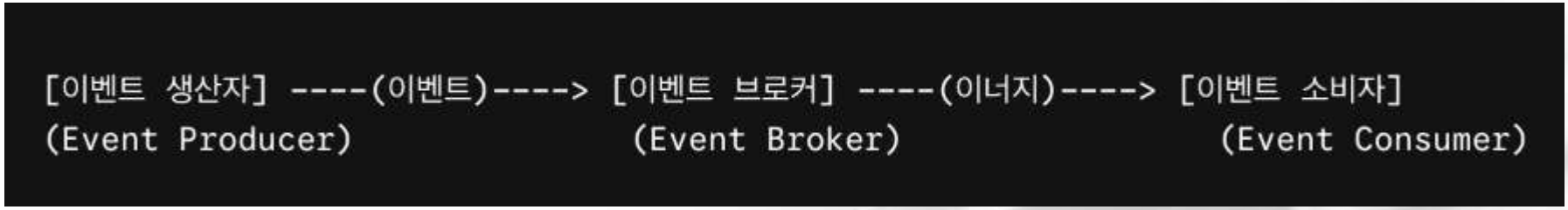
- ❖ Serverless Computing 과 Event Driven Architecture
  - ✓ Serverless Computing
    - 장점
      - ◆ 인프라 비용 절감: 사용한 만큼만 내므로 트래픽 변동이 심한 서비스에 유리
      - ◆ 빠른 출시: 서버 구축 단계가 생략되므로 아이디어를 빠르게 개발해 서비스할 수 있음
      - ◆ 비즈니스 집중: 개발팀이 인프라 장애 대응 대신 서비스 기능 개선에 전념 가능
    - 단점
      - ◆ 콜드 스타트(Cold Start): 오랫동안 실행되지 않다가 처음 호출될 때 서버를 켜는 준비 시간이 걸려 약간의 지연(Delay)이 발생할 수 있음
      - ◆ 플랫폼 종속성: AWS용으로 만든 서버리스 코드를 Google Cloud로 옮기기 까다로울 수 있음
      - ◆ 장기 실행 시 비용 증가: 24시간 내내 쉴 새 없이 돌아가는 대규모 서비스라면 오히려 일반 가상 서버(EC2 등)를 빌리는 게 더 저렴할 수 있음

클라우드에 필요한 기술

- ❖ Serverless Computing 과 Event Driven Architecture
  - ✓ Event Driven Architecture
    - 이벤트 기반 아키텍처(Event-Driven Architecture, EDA)는 소프트웨어 아키텍처 패턴 중 하나로 시스템 내·외부에서 발생한 의미 있는 상태의 변화인 이벤트(Event)를 생산(Publish)하고 감지(Detect)하여 소비(Consume)하는 방식으로 동작하는 시스템 구조
    - 핵심 구성 요소
      - ◆ 이벤트 생산자 (Producer)
        - 상태 변화(이벤트)를 감지하고 이를 표준화된 메시지 형태로 만들어 발행함
        - 예: 사용자가 쇼핑몰에서 '결제 완료' 버튼을 누름.
      - ◆ 이벤트 브로커 (Broker / 채널)
        - 생산자가 보낸 이벤트를 수집하고, 이를 필요로 하는 소비자들에게 안전하게 전달(라우팅)하는 중간 중재자 역할을 수행
        - 예: Apache Kafka, RabbitMQ, AWS EventBridge 등.
      - ◆ 이벤트 소비자 (Consumer)
        - 브로커를 모니터링하고 있다가 관심 있는 이벤트가 들어오면 이를 가져와서 자신의 비즈니스 로직을 수행
        - 예: 결제 완료 이벤트를 받아서 배송 지시를 내리는 서비스 또는 알림 톡을 발송하는 서비스

클라우드에 필요한 기술

- ❖ Serverless Computing 과 Event Driven Architecture
  - ✓ Event Driven Architecture
    - ● 가장 큰 특징: 느슨한 결합 (Loose Coupling)
      - ◆ 기존의 전통적인 아키텍처(API 기반 호출 방식)에서는 A 서비스가 B 서비스를 직접 호출(A -> B)해야 했는데 만약 B 서비스가 점검 중이거나 장애가 나면 A 서비스도 함께 멈추는 강한 결합(Tight Coupling) 문제
      - ◆ 이벤트 기반 아키텍처에서는 A 서비스는 그저 브로커에 나 결제 완료 이벤트 던졌어! 라고 말하고 자기 할 일을 끝내는데 B 서비스가 살았는지 죽었는지는 알 바가 아니며 B 서비스는 나중에 살아나서 브로커에 쌓여있던 이벤트를 가져가 처리하면 되기 때문에 각 서비스가 서로의 존재를 몰라도 시스템이 굴러감



클라우드에 필요한 기술

- ❖ Serverless Computing 과 Event Driven Architecture
  - ✓ Event Driven Architecture
    - 장점
      - ◆ 높은 유연성과 확장성: 새로운 기능(소비자)을 추가하고 싶을 때, 기존 코드를 건드릴 필요 없이 브로커에 빨대를 꽂듯 새 서비스만 연결하면 됨.
      - ◆ 장애 격리 (Fault Tolerance): 특정 서비스에 장애가 발생해도 브로커가 이벤트를 보관해 주므로, 시스템 전체가 마비되지 않고 장애가 복구된 후 이어서 처리 가능
      - ◆ 실시간성 응답: 대용량 데이터 처리나 트래픽 스파이크(급증)가 발생했을 때 브로커가 완충재 역할을 하여 시스템이 안정적으로 유지됨
    - 단점
      - ◆ 시스템 복잡도 증가: 메시지가 브로커를 거쳐 비동기로 흐르기 때문에 전체 시스템의 흐름을 한눈에 파악하고 디버깅(추적)하기가 까다로움
      - ◆ 데이터 일관성(Eventually Consistency) 문제: 실시간으로 모든 DB가 동시에 업데이트되는 것이 아니라 시간이 지나면서 점진적으로 일관성이 맞춰지므로 정밀한 트랜잭션 관리가 필요함
      - ◆ 중복 처리 고려 필요: 네트워크 문제 등으로 인해 동일한 이벤트가 두 번 전달될 수 있으므로, 소비자가 이를 중복 처리하지 않도록(Idempotency) 설계해야 함.

클라우드에 필요한 기술

- 스토리지와 데이터 구조
  - ✓ 종류
    - 객체 스토리지
    - 블록 스토리지
    - 파일 스토리지
  - ✓ 클라우드 스토리지를 잘 활용하려면 단순히 저장 공간이 필요하다는 생각을 넘어서 이 데이터는 어떤 방식으로 사용되고 얼마나 자주, 멀마나 빨리 접근해야 하는 가를 고려해야 함

클라우드에 필요한 기술

- 스토리지와 데이터 구조
  - ✓ 객체 스토리지
    - 클라우드에서 데이터를 저장하는 방식 중 가장 많이 사용되는 기술
    - 구조가 단순하고 대용량 데이터를 오래 보관할 수 있어서 많은 서비스에서 기본 스토리지로 활용
    - 객체
      - 데이터 본문(내용), 그에 대한 속성(메타 데이터) 그리고 고유한 이름 또는 주소를 가지고 있는 단위
      - 계층 구조가 아니라 평면 구조
      - 트리 구조처럼 보이게 만들 수는 있지만 실제로는 그냥 이름 규칙을 이용한 표현 방식일 뿐
    - 객체 스토리지는 Web 기반 API로 접근할 수 있게 설계되어 있어서 프로그램이나 서비스에서 직접 데이터를 주고받는데 적합
    - 장기 보관과 높은 내구성을 목표로 설계되었는데 내부적으로는 데이터를 여러 곳에 복제하고 무결성 검사를 주기적으로 수행

클라우드에 필요한 기술

- ❖ 스토리지와 데이터 구조
  - ✓ 블록 스토리지
    - 전통적인 하드디스크나 SSD와 비슷한 개념
    - 데이터를 아주 작은 블록 단위로 나누어 저장하고 운영체제가 이를 조합해서 하나의 디스크처럼 사용하는 방식
    - 가상 머신을 만들 때 자주 사용하는데 가상 머신은 운영체제를 설치하고 파일 시스템을 구성해야 하는데 운영체제가 올라갈 공간이 블록 스토리지
    - 읽기 와 쓰기가 빠르고 정교하게 제어된다는 점에서 강점이 있음
    - 데이터베이스처럼 끊임없이 데이터를 기록하고 다시 읽어야 하는 프로그램에서는 저장 속도와 반응 속도가 중용한데 이런 경우에는 객체 스토리지 보다 블록 스토리지가 훨씬 적합
    - 파일 전체를 변경하지 않고 일부분만 수정하거나 빠르게 덮어쓸 수 있다는 점도 장점
    - 블록 스토리지는 복잡한 설정이 추가되는데 운영체제가 포맷을 해야 하고 사용자가 직접 파일 시스템을 관리해야 하기 때문에 클라우드에서는 가상 머신에 딸려오는 디스크처럼 활용하고 백업이나 이미지 저장 같은 용도 보다는 시스템 운영과 관련된 스토리지로 자주 사용

클라우드에 필요한 기술

- ❖ 스토리지와 데이터 구조
  - ✓ 파일 스토리지
    - 네트워크 드라이브, 회사에서 공유 디렉토리 간은 것이 파일 스토리지
    - 여러 명이 동시에 접근해서 디렉토리 안의 파일을 읽고 쓰고, 수정하고 삭제할 수 있는 방식
    - 디렉토리 – 파일 구조를 그대로 적용
    - 협업에 적합한 방식
    - 웹 서버에서 이미지 파일이나 정적 리소스를 여러 인스턴스가 공유해야 할 때 유용
    - 객체 스토리지나 블록 스토리지보다 대용량 처리나 성능 측면에서는 제한이 있음
    - 많은 양의 데이터를 동시에 빠르게 읽고 써야 하는 환경에서는 병목이 생기기 쉬움
    - 편의성, 동시 접근성, 사용자 친화적 인터페이스라는 면에서 강점

클라우드에 필요한 기술

- ❖ 스토리지와 데이터 구조
  - ✓ 비교

항목	객체 스토리지	블록 스토리지	파일 스토리지
저장 단위	객체(데이터 + 메타데이터 + 식별자)	블록(고정 크기의 데이터 단위)	파일(디렉토리 + 파일 계층 구조)
접근 방식	HTTP API 기반	운영체제에서 디스크처럼 인식	네트워크를 통한 공유 디렉토리 접근
사용 예시	로그, 백업, 이미지, 동영상, 스트리밍 콘텐츠	데이터베이스 스토리지, 가상 머신 디스크, 고성능 트랜잭션 처리	협업 문서 저장, 웹 애플리케이션 공통 파일 리소스
장점	구조가 단순하고 무한에 가까운 확장	고속 읽기/쓰기, 파일 시스템 유연성	사용하기 쉬운 구조, 여러 사용자 간 공유 용이
단점	실시간 편집, 디렉토리 탐색에는 부적합	구조가 복잡하고 저장용도로는 과할 수 있음	대용량 처리나 고성능 요구에는 부적합
적합한 사용 상황	자주 접근하지 않는 데이터 장기 저장	고성능이 필요한 애플리케이션 스토리지	사용자 간 파일 공유, 소규모 협업 환경
대표 서비스	AWS S3, Azure Blob, Google Cloud Storage	AWS EBS, Azure Managed Disks, Google Persistent Disks	AWS EFS, Azure Files, Google Filestore

클라우드에 필요한 기술

- ❖ 스토리지와 데이터 구조
  - ✓ AWS S3 Glacier
    - 사용자가 업로드한 데이터를 자체적으로 자동 압축하여 저장하지 않음
    - 데이터 압축은 업로드 전 사용자나 애플리케이션 단계(예: .zip, .tar.gz 등)에서 진행해야 하며 Glacier는 수신된 원본(또는 압축된) 데이터를 아카이브(Archive) 단위의 객체 스토리지(Object Storage) 방식으로 저장
    - 아카이브 (Archive): Glacier에 저장되는 최소 기본 단위(파일, 문서, 압축파일 등)
    - 단일 아카이브의 크기는 최대 40TB까지 가능하며 업로드 시 시스템에서 고유한 아카이브 ID(Archive ID)를 부여받음
    - 볼트(Vault): 아카이브들을 담는 상위 컨테이너(폴더 개념)으로 계정당/지역당 최대 1,000개까지 생성 가능하며 접근 권한(Vault Access Policy) 및 수정 금지 규정(Vault Lock)을 설정
    - 기본 자동 암호화: 저장되는 모든 데이터는 서버 측 암호화(AES-256)가 기본 적용되어 안전하게 보관
    - 물리 매체 관리: 내부적으로 자기 테이프(Tape) 기술이나 고밀도 저전력 하드디스크 배열을 활용하여 상시 전력 소모와 유지 비용을 최소화

클라우드에 필요한 기술

- ❖ 스토리지와 데이터 구조
  - ✓ 요금 과 성능의 균형
    - 과금 기준
      - ◆ 저장 용량
      - ◆ 입출력 작업 수(I/O Request)
      - ◆ 데이터 전송량
    - 데이터의 특성과 사용 빈도를 기준으로 스토리지를 구분하고 적절한 요금제와 등급을 선택하는 것이 현명한 방법

클라우드에 필요한 기술

- ❖ 스토리지와 데이터 구조
  - ✓ 수명 주기 관리
    - 클라우드에서 데이터를 저장하는 경우 시간이 지남에 따라 데이터의 활용도나 중요도는 계속 변화 함
    - 사용자는 특정 조건을 기준으로 데이터를 자동으로 다른 스토리지 계층으로 옮기거나 일정 기간이 지나면 삭제되도록 규칙을 정할 수 있음
    - 로그 데이터는 일정 기간 동안은 자주 확인할 수 있지만 일정 시간이 지나면 단지 보관만 하면 되는 경우가 많으므로 90일 이후에는 장기 보관용 계층으로 이동하고 1년 후에는 삭제되도록 정책을 설정하면 편리
    - 스토리지 수명 주기 관리는 단순한 비용 절감 도구일 뿐 아니라 데이터 거버넌스 측면에서도 중요한 역할을 하는데 보관기간을 정해두고 오래된 데이터를 자동으로 정리함으로써 데이터 과일을 방지할 수 있는데 특히 개인 정보나 법적 보존 기간이 정해진 데이터를 다룰 때 수동 관리보다 훨씬 안정적이고 일관된 방식으로 데이터를 관리할 수 있음

클라우드에 필요한 기술

- 네트워크
  - ✓ VPN 과 클라우드 전용선
    - 클라우드에 회사 내부 시스템을 연결하거나 클라우드 리소스를 더 안전한 방식으로 접근하고자 하는 경우 사용 가능한 기술
    - 사용 가능한 기술
      - VPN
      - Dedicated Line(클라우드 전용선)

클라우드에 필요한 기술

- ❖ 네트워크
  - ✓ VPN 과 클라우드 전용선
    - ● VPN
      - ◆ 가상의 전용 네트워크를 만들어서 사설망처럼 안전하게 데이터를 주고 받을 수 있게 해주는 기술
      - ◆ 실제로는 인터넷을 이용하되 외부에서는 볼 수 없도록 데이터를 암호화 해서 전송하는 방식
      - ◆ 클라우드에서는 이 VPN을 구성할 수 있는 기능이 기본으로 제공됨
      - ◆ 사용자는 클라우드 상에 VPN 게이트웨이를 하나 만들어서 VPN 클라이언트와 연결하면 됨
      - ◆ VPN은 인터넷을 기반으로 하기 때문에 품질은 인터넷 연결 상태에 따라 달라짐
      - ◆ 화상회의나 대용량 데이터 전송처럼 지연에 민감하거나 안정성이 중요한 경우에는 VPN 만으로는 부적합

클라우드에 필요한 기술

- ❖ 네트워크
  - ✓ VPN 과 클라우드 전용선
    - ● 전용선
      - ◆ 회사 내부 네트워크와 클라우드 간을 직접 물리적으로 연결한 전용 통신선
      - ◆ 일반 인터넷을 거치지 않기 때문에 속도가 빠르고 안전성도 높고 보안도 뛰어남
      - ◆ 내부 시스템과 데이터베이스를 연결하거나 실시간 연동이 필요한 산업 시스템을 운영할 때는 전용선이 거의 필수
      - ◆ 구성이 복잡하고 비용이 비쌈

클라우드에 필요한 기술

- ❖ 네트워크
  - ✓ VPN 과 클라우드 전용선
    - ● 비교

항목	VPN	클라우드 전용선
연결 방식	공용 인터넷 망을 통해 가상의 암호화된 터널 생성	기업과 클라우드/데이터센터 간 물리적인 정용 회선 구축
구축 비용	저렴	고비용
설치 속도	빠름	느림
보안 수준	암호화로 보호되지만 인터넷 기반이라 위험 요소 존재	외부 노출없이 안전한 통신 가능
속도 및 품질	인터넷 품질에 따라 변동 가능	안정적이고 고속의 대역폭 제공
확장성	유연하게 확장 가능	확장 어려움
사용 예시	원격 근무, 관리용 접속, 일반 업무	대규모 서비스 운영, 실시간 데이터 전송, 금융 및 제조 환경 등

클라우드에 필요한 기술

- 네트워크
  - ✓ 로드 밸런서와 서비스 엔드포인트
    - 서비스 엔드포인트
      - 서비스 엔드포인트(Service Endpoint)는 네트워크 상에서 클라이언트(사용자, 앱, 다른 서버)가 특정 서비스나 애플리케이션 기능에 접근할 수 있도록 노드나 자원을 노출해 놓은 접점(진입점 통로)을 의미
    - 엔드포인트 사용의 장점
      - 캡슐화 및 추상화: 내부 시스템 구조(DB 위치, 백엔드 서버 개수 등)가 바뀌어도 외부 사용자는 동일한 엔드포인트 주소만 참조하면 됨
      - 보안 제어: 특정 엔드포인트 단위로 인증/인가, 방화벽 규칙, 트래픽 제한(Rate Limiting)을 적용할 수 있음
      - 로드 밸런싱: 하나의 엔드포인트 뒷단에 여러 대의 서버를 두어 트래픽을 분산 처리할 수 있음
    - 로드밸런서와 서비스 엔드포인트는 클라우드 기반 서비스에서 서비스의 안정성 과 확장성 그리고 보안성을 높여주기 위해서 사용

클라우드에 필요한 기술

- 네트워크
  - ✓ 클라우드 네이티브 네이밍 서비스
    - DNS는 도메인 이름을 통해 네트워크 통신을 쉽게 해줌
    - 클라우드 환경에서는 조금 더 복잡한 요구가 생김
    - 클라우드에서는 서버나 컨테이너 수가 수시로 만들어지고 없어지기 때문에 고정된 IP를 사용하는 것이 어렵고 비효율적
    - DNS와 유사한 역할을 하되 더 유연하고 자동화된 방식의 네이밍 서비스가 필요했는데 이러한 배경속에서 등장한 것이 클라우드 네이티브 네이밍 서비스
    - 클라우드 네이티브 환경에서는 수동으로 도메인을 등록하고 IP를 할당하는 방식 대신 시스템이 자동으로 이름을 붙이고 그에 해당하는 위치(IP 나 엔드포인트)를 실시간으로 연결해 줌
    - 이런 네이밍 방식은 특히 컨테이너 오케스트레이션 환경에서 많이 사용
    - 쿠버네티스 같은 시스템에서는 서비스가 생기면 자동으로 네임서버에 등록되고 다른 서비스가 그 이름으로 접근할 수 있게 해 줌
    - 장점
      - 유연성이 높음 – 컨테이너나 가상 머신의 위치가 변경되도 서비스 이름은 그대로 유지가 되므로 네트워크 구성을 수정할 필요가 없음
      - 자동화가 가능 – 직접 도메인 이름을 등록하거나 IP를 확인하지 않아도 시스템이 알아서 이름 과 위치를 연결해 줌
      - 서비스가 통신이 쉬움 – 클라우드 내부에서 마이크로 서비스들이 통신할 때 서비스 이름만 알면 통신이 가능

클라우드에 필요한 기술

- 네트워크
  - ✓ 클라우드 네이티브 네이밍 서비스
    - 일부 클라우드 환경에서는 프라이빗 DNS나 내부 전용 네이밍 서비스도 함께 제공해서 외부에 노출되지 않는 안전한 네트워크 내에서 내부 서비스들끼리 효율적으로 통신할 수 있음
    - IP 주소를 기억하고 관리하는 시대를 넘어서 이름 기반의 네트워크 설계가 가능

클라우드에 필요한 기술

- 네트워크
  - ✓ 클라우드 가상 네트워크 구조
    - 클라우드에서 서버를 만들고자 하는 경우에는 가상 네트워크라는 단위부터 생성을 해야 하는데 클라우드에서는 VPC(Virtual Private Cloud) 또는 Vnet(Virtual Network)라고 부름
    - 생성 과정
      - VPC 생성
      - IP 주소 범위를 설정하고 서브넷 생성
      - 서버 생성 – 필요에 따라 Private IP를 설정하고 외부로 연결되어야 하는 경우에는 Public IP 나 NAT Gateway를 설정
      - 라우팅 테이블로 흐름을 설정 – NAT Gateway를 거치게 하거나 온프레미스로 보내는 등
      - 보안 그룹 과 네트워크 ACL을 이용해 방화벽 기능을 구현

클라우드에 필요한 기술

- Infracstructure as Code
  - ✓ 등장 배경
    - 틀라우드 환경이 점점 복잡해지고 구성해야 할 리소스가 많아지면서 일관성을 유지하기가 어려워지졌는데 같은 작업을 반복해도 사람이 직접 클릭하는 방식은 실수를 유발할 수 있음
    - 수작업 방식은 협업을 어렵게 함
  - ✓ 개요
    - 시스템, 하드웨어 또는 인터페이스의 구성 정보를 파일(스크립트)을 통해 관리 및 프로비저닝
    - IT Infracstructure, 베어 메탈 서버 등의 물리 장비 및 가상 머신과 관련된 구성 리소스를 관리
    - 버전 관리를 통한 리소스 관리
    - 인프라를 코드로 관리하면 운영의 철학과 문화를 바꾸는 일

클라우드에 필요한 기술

- Infracstructure as Code
  - ✓ 도구 비교

항목	Terraform	Ansible
개발 회사	HashiCorp	Red Hat
주요 사용 용도	클라우드 인프라 관리	서버 구성 관리, 배포
도구 종류	IaC 도구	IT 자동화 도구
작업 정의 언어	HCL	YAML
에이전트 필요성	없음	없음 (에이전트리스)
모듈/플러그인	프로바이더	모듈
클라우드 지원	다양한 클라우드 프로바이더	일부 클라우드 프로바이더

클라우드에 필요한 기술

- Infracstructure as Code
  - ✓ 도구 비교
    - CloudFormation
      - AWS에서 제공하는 IaC 도구
      - AWS에 최적화된 인프라 구성 도구이고 다른 클라우드에서 사용 못함
      - JSON 이나 YAML로 인프라를 정의
      - 진입 장벽이 낮음
      - 템플릿 내에 조건문과 반복문 등을 사용할 수 있어서 복잡한 환경도 하나의 파일로 유연하게 표현할 수 있음

클라우드에 필요한 기술

- Infracstructure as Code
  - ✓ 장점
    - 불필요한 자원 생성을 방지
    - 자원의 자동 정리 가능
    - 환경 간 일관성 유지
    - 변경 이력 추적 가능
    - 검토 및 승인 프로세스 연동
    - 클라우드 환경의 통제력 향상

클라우드에 필요한 기술

- ❖ 보안 과 접근 제어
  - ✓ 공유 책임 모델
    - 클라우드의 보안은 클라우드 기업이 전적으로 책임져 주지 않음
    - 클라우드 보안은 클라우드 제공자가 전적으로 책임을 지는 것이 아니며 일정 부분 사용자도 함께 책임을 져야 함
    - 클라우드 기업은 물리적인 기반 인프라에 대해서는 전적으로 책임이 있지만 그 위에 올려 사용하는 소프트웨어에 대해서는 무엇을 올려서 어떻게 사용하는가에 따라 사용자에게도 책임이 있다는 뜻
    - 책임의 경계
      - ◆ IaaS: 운영체제부터 네트워크 설정을 포함한 거의 대부분을 사용자가 관리
      - ◆ PaaS: 플랫폼의 기본 구성과 일부 보안 설정은 클라우드가 책임지고 사용자는 애플리케이션 단의 보안을 담당
      - ◆ SaaS: 거의 모든 인프라 보안은 클라우드가 담당하지만 사용자 계정과 데이터 접근 권한 설정은 사용자 몫

클라우드에 필요한 기술

- ❖ 보안 과 접근 제어
  - ✓ IAM(Identity and Access Management)
    - 적절한 사용자(Identity)가 적절한 시점에 적절한 시스템 자원에 접근(Access)할 수 있도록 통제하고 관리하는 보안 체계이자 프레임워크
    - IAM의 핵심 구성 요소
      - ◆ 식별(Identification): 누가 접근하고 있는지를 명확히 구분하는 것으로 사용자는 사람일 수 도 있지만 서버나 프로그램 같은 시스템 일 수 도 있음
      - ◆ 인증(Authentication - AuthN): 사용자가 주장하는 신원이 맞는지 확인하는 과정(예: ID/비밀번호 입력, MFA/2단계 인증, 소셜 로그인)
      - ◆ 인가(Authorization - AuthZ): 인증된 사용자에게 특정 자원에 접근할 권한이 있는지 확인하는 과정 (예: 읽기 전용 권한, 관리자 권한 부여)
      - ◆ 계정 관리(Account / Identity Lifecycle): 신규 입사자의 계정 생성(Provisioning)부터 부서 이동 시 권한 변경, 퇴사 시 계정 삭제/비활성화(Deprovisioning)까지의 수명주기 관리
    - 각자에게 필요한 권한만 설정하는 것을 클라우드에서는 Policy 라고 함
    - 클라우드 보안의 기초이자 가장 강력한 통제 도구

클라우드에 필요한 기술

- ❖ 보안 과 접근 제어
  - ✓ IAM(Identity and Access Management)
    - ● 권한 관리 방식

구분	RBAC (역할 기반 접근 제어)	ABAC (속성 기반 접근 제어)
개념	사용자의 직책/역할에 따라 권한 부여	사용자, 환경, 자원의 속성/조건에 따라 권한 부여
예시	재무팀 직원' 역할 = 재무 시스템 접근 허용	한국 지사 직원'이 '업무 시간 중'에 접근할 때만 허용
특징	구조가 단순하고 관리가 쉬움	세밀한(Granular) 보안 정책 설정 가능

클라우드에 필요한 기술

- ❖ 보안 과 접근 제어
  - ✓ IAM(Identity and Access Management)
    - IAM이 필요한 이유 및 장점
      - ◆ 최소 권한의 원칙 (Principle of Least Privilege) 적용: 사용자에게 업무에 필요한 최소한의 권한만 부여하여 보안 사고 발생 시 피해 범위를 최소화
      - ◆ 중앙 집중식 관리: 클라우드(AWS, Azure 등) 및 기업 내 수많은 시스템의 계정과 권한을 한곳에서 통합 제어
      - ◆ 규정 준수(Compliance) 및 감사: 누가, 언제, 어떤 데이터에 접근하고 변경했는지에 대한 상세 기록(Audit Log)을 남겨 법적/보안 규제를 충족
    - 대표적인 IAM 솔루션 및 서비스
      - ◆ 클라우드 IAM: AWS IAM, Google Cloud IAM, Microsoft Entra ID(구 Azure AD)
      - ◆ 기업용 IAM/SSO 솔루션: Okta, Ping Identity, Keycloak

클라우드에 필요한 기술

- ❖ 보안 과 접근 제어
  - ✓ 최소 권한의 원칙
    - 최소 권한의 원칙(Principle of Least Privilege, PoLP)은 정보 보안의 가장 핵심적인 규칙 중 하나로 사용자, 프로세스, 프로그램에 주어진 업무나 임무를 수행하는 데 필요한 최소한의 접근 권한 만을 부여해야 한다는 원칙
    - 필요 이상의 권한을 미리 주지 않고 알 필요가 있는 정보(Need-to-Know)와 할 필요가 있는 행동(Need-to-Do)에 대해서만 딱 맞게 허용하는 방식
    - 최소 권한의 원칙이 필요한 이유
      - ◆ 보안 사고 피해 최소화 (Blast Radius 감소): 계정이 해킹당하거나 악성코드에 감염되더라도, 해당 계정이 가진 권한 범위 내에서만 피해가 발생하므로 전체 시스템으로의 피해 확산을 막을 수 있음
      - ◆ 내부자 위협 방지: 퇴사자나 불만을 품은 내부 직원이 인가되지 않은 중요 시스템이나 고객 정보에 접근하여 유출/파괴하는 행위를 사전 차단
      - ◆ 실수 및 오류로 인한 데이터 손실 방지: 단순 작업자에게 시스템 전체 삭제 권한이 없다면 잘못된 명령어 실행으로 인한 대형 장애를 예방할 수 있음
      - ◆ 규정 준수(Compliance): ISMS-P, GDPR, PCI-DSS 등 주요 보안 표준 및 법적 규제에서 필수 조건으로 요구

클라우드에 필요한 기술

- ❖ 보안 과 접근 제어
  - ✓ 최소 권한의 원칙
    - ● 일상 및 IT 환경에서의 적용 예시
      - ◆ 시스템 및 클라우드 환경
        - ▪ 루트(Root/Administrator) 계정 사용 자제: 평소에는 일반 계정으로 작업하고 필요한 순간에만 권한 상승 명령어(sudo 등)를 사용
        - ▪ AWS IAM 정책: 개발자에게 DB 접근 권한 전체(*)를 주는 대신 특정 테이블의 읽기 권한(s3:GetObject)만 부여
      - ◆ 개발 및 애플리케이션
        - ▪ 스마트폰 앱 권한 요청: 계산기 앱이 위치 정보나 연락처 권한을 요구하지 않도록 통제하는 것
        - ▪ DB 계정 분리: 웹 애플리케이션 연결용 DB 계정에는 SELECT, INSERT 권한만 주고 테이블 삭제 권한(DROP)은 부여하지 않음

클라우드에 필요한 기술

- ❖ 보안 과 접근 제어
  - ✓ 최소 권한의 원칙
    - ● 최소 권한 원칙 이행 방법 (Best Practices)
      - ◆ 기본 거부 (Default Deny): 모든 접근 요청을 기본적으로 차단(Deny All)해 두고 명시적으로 허용된 권한만 염
      - ◆ 역할 기반 접근 제어 (RBAC): 사용자 개개인이 아닌 부서/직책(Role)에 권한을 그룹화하여 부여하고 부서 이동 시 기존 권한을 바로 회수
      - ◆ 적시 권한 부여 (Just-In-Time Access): 고위 권한 작업이 필요한 경우에만 임시로 짧은 시간(예: 2시간) 동안만 권한을 승인받아 사용
      - ◆ 정기적인 권한 감사 (Access Review): 방치된 휴면 계정이나 과도하게 부여된 권한이 없는지 주기적으로 점검 및 회수

클라우드에 필요한 기술

- ❖ 보안 과 접근 제어
  - ✓ 비밀번호 나 키 관리 와 암호화
    - 많은 개발자들이 실수하는 부분 중 하나는 비밀번호나 API 키를 코드를 직접 적는 것인데 이 문제를 방지하려면 먼저 비밀 정보를 코드와 분리해서 관리해야 함
    - 환경 변수(environment variables) 나 외부 설정 파일(config file)을 사용해서 비밀번호나 키를 별도로 관리
    - 클라우드 제공자는 비밀 관리 서비스를 별도로 제공
      - ◆ 민감 정보를 암호화된 형태로 안전하게 저장
      - ◆ 암호화된 정보에 접근할 수 있는 주체와 범위, 방식까지도 정밀하게 통제
      - ◆ 암호화된 정보는 클라우드 내부의 암호화 엔진으로 관리되기 때문에 별도로 암호화 알고리즘이나 키 보관 방법을 직접 신경 쓸 필요없음
      - ◆ 키 회전(암호화 키(Key)를 정기적 또는 이벤트 발생 시 새 키로 교체/갱신하는 관리 프로세스) 자동화
    - 클라우드 기업이 제공하는 비밀 관리 서비스
      - ◆ AWS의 Secret Manager
      - ◆ Azure의 Key Vault
      - ◆ GCP의 Secret Manager

클라우드에 필요한 기술

- ❖ 보안 과 접근 제어
  - ✓ 인증 토큰 & MFA
    - ● 인증 토큰
      - ◆ 사용자가 시스템에 신원(아이디/비밀번호 등)을 한 번 증명한 뒤 이후 요청에서 매번 비밀번호를 입력하지 않고도 자격 증명을 대신하기 위해 서버가 발급해 주는 암호화된 증명서
      - ◆ 가장 대표적인 표준 규격은 JWT(JSON Web Token)
      - ◆ 토큰 유형
        - ▪ Access Token(엑세스 토큰)
          - • 리소스에 접근하기 위한 실질적인 인증 권한을 담은 토큰
          - • 유효 기간이 짧게 설정(예: 30분~2시간)되어 탈취 시 피해를 줄임
        - ▪ Refresh Token(리프레시 토큰)
          - • Access Token이 만료되었을 때 새 Access Token을 재발급받기 위한 토큰
          - • 유효 기간이 길며(예: 2주~1달), 서버 DB 또는 저장소에 저장하여 관리

클라우드에 필요한 기술

- ❖ 보안 과 접근 제어
  - ✓ 인증 토큰 & MFA
    - ● 인증 토큰
      - ◆ 토큰 인증의 장점 과 단점
        - ▪ 장점
          - Stateless(무상태성): 서버가 세션 상태를 저장하지 않아 확장성이 뛰어남
          - 모바일/다중 플랫폼 적합: CORS 문제 해결 및 REST API와 완벽 호환
        - ▪ 단점
          - 탈취 위험: 클라이언트에 저장되므로 XSS/CSRF 공격 대비 필요
          - 강제 폐기 불가: 발급된 토큰은 만료 전까지 서버에서 즉시 취소하기 어려움

클라우드에 필요한 기술

- ❖ 보안 과 접근 제어
  - ✓ 인증 토큰 & MFA
    - MFA
      - ◆ 사용자가 로그인하거나 시스템에 접근할 때 둘 이상의 서로 다른 인증 요소(Factor)를 조합하여 신원을 검증하는 보안 방식
      - ◆ 아이디와 비밀번호(1개 요소)만 사용하는 방식을 넘어 계정 유출로 인한 무단 접근을 차단하기 위해 필수적으로 적용
      - ◆ 2FA와의 차이점
        - 2FA (Two-Factor Authentication): 인증 요소를 정확히 2개 사용하는 방식 (MFA의 하위 개념)
        - MFA (Multi-Factor Authentication): 인증 요소를 2개 이상(2개, 3개 등) 사용하는 모든 방식
      - ◆ MFA를 사용하는 이유
        - 비밀번호 유출 대비: 비밀번호가 해킹, 피싱, 데이터 유출 등으로 노출되더라도, 소유한 기기나 생체 인증 없이는 로그인할 수 없음
        - 계정 침입 방지: 보안 연구에 따르면 MFA를 활성화하는 것만으로 자동화된 계정 공격의 99% 이상을 막을 수 있음
        - 규정 준수: 많은 기업 및 규제 기관(금융, Cloud 보안 등)에서 MFA 적용을 의무화하고 있음

클라우드에 필요한 기술

- ❖ 장애 대응
  - ✓ 장애
    - Human Error
    - 서비스간 의존성 문제
    - 물리적 리소스
    - 네트워크
  - ✓ 클라우드에서는 장애가 여러 계층에서 발생할 수 있기 때문에 어디서 문제가 발생했는지를 빠르게 파악하고 그에 맞는 대응 전략을 세우는 것이 중요
  - ✓ 장애는 언제든 발생할 수 있으니 이를 인정하고 장애에 대한 감지 ->대응 -> 복구의 전체 주기를 체계적으)로 준비하는 것이 중요
  - ✓ 장애에 대한 전략
    - 실시간 모니터링 과 알림 시스템
    - 자동 복구 및 자가 치유(Self Healing 기능
    - 다중 리전 이중화 구성
    - 정기적인 백업과 복원 시나리오 점검
    - 복구 시점 목표(RPO) 와 복구 시간 목표(RTO) 수립
    - 사후 분석(Postmortem) 과 재발 방지 체계

클라우드에 필요한 기술

- ❖ 장애 대응
  - ✓ 모니터링 과 로그 수집
    - 클라우드에서는 CPU 사용률, 가용한 메모리 용량, 서버가 응답하는 속도 같은 내용을 들여다 보면서 이상 징후가 생기면 알려줌
    - 클라우드에서는 로그를 한 곳으로 모아서 보는 시스템을 사용
    - 최근에는 서비스 흐름을 따라가는 기능도 제공하는데 이를 분산 추적(Distributed Tracing)
    - 장애를 빠르게 감지하고 원인을 찾아내는 방법
      - ◆ 실시간 감시를 통해 문제가 생기자 마자 알림을 받도록 하기
      - ◆ 로그를 통해 문제의 원인을 추적할 수 있도록 하기
      - ◆ 흩어진 데이터를 한 곳에 모아 보기 쉽게 정리하기

클라우드에 필요한 기술

- ❖ 장애 대응
  - ✓ 고가용성 구성
    - 시스템, 서버, 네트워크 등이 장애가 발생하더라도 중단 없이(또는 아주 짧은 시간 내에) 서비스를 지속적으로 제공할 수 있는 능력을 의미
    - 24시간 365일 멈추지 않고 계속 잘 작동하는 성질
    - 핵심 목표: 가용성 지표(Availability): 가용성은 주로 몇 퍼센트의 시간 동안 시스템이 정상 작동했는가로 측정하며 9(Nines)의 개수로 보안 및 가동 수준을 표현
      - ◆ 99.9% (Three Nines): 연간 누적 장애 시간 약 8.76시간
      - ◆ 99.99% (Four Nines): 연간 누적 장애 시간 약 52.6분(일반적인 엔터프라이즈 기준)
      - ◆ 99.999% (Five Nines): 연간 누적 장애 시간 약 5.26분(금융, 통신, 의료 등 핵심 인프라)

클라우드에 필요한 기술

- ❖ 장애 대응
  - ✓ 고가용성 구성
    - ● 고가용성을 구현하는 3가지 핵심 원칙
      - ◆ 단일 장애점 제거(No Single Point of Failure, SPOF): 시스템의 어느 한 부품이나 서버가 고장 나더라도 전체 서비스가 멈추지 않도록 설계
      - ◆ 이중화 및 이중 구성 (Redundancy)
        - ▪ 서버, DB, 네트워크 장비, 전원 공급 장치 등을 최소 2개 이상 배치
        - ▪ Active-Active: 여러 대가 동시에 트래픽을 처리하다가 하나가 죽으면 나머지가 부담
        - ▪ Active-Standby: 하나는 일하고 하나는 대기하다가, 메인 서버 장애 발생 시 대기 서버가 즉시 전환(Failover)
      - ◆ 장애 감지 및 자동 복구 (Failover & Monitoring): 시스템 상태를 실시간으로 감시(Health Check)하다가, 문제 발생 시 사람의 개입없이 자동으로 상시 대기 중인 시스템으로 전환

클라우드에 필요한 기술

- ❖ 장애 대응
  - ✓ 고가용성 구성
    - ● 고가용성을 지탱하는 관련 개념
      - ◆ 서버 이중화
        - ▪ Master-Slave 구조 또는 Read-Replica 구조
        - ▪ Multi-Region Deployment: 지리적 이중화
      - ◆ Load Balancer
        - ▪ Health Check
        - ▪ 부하 분산
      - ◆ Disaster Recovery(재해 복구)
        - ▪ 문제가 발생하면 서버를 자동으로 교체

클라우드에 필요한 기술

- ❖ 장애 대응
  - ✓ 고가용성 구성
    - ● 재해 복구 목표 수립
      - ◆ RPO 와 RTO는 복구 전략의 기준점으로 어떤 도구를 사용할지 또는 얼마나 자주 백업을 할지 어떤 인프라 구조를 선택할지 결정
      - ◆ RPO(복구 시점 목표)
        - ▪ 장애 발생 시 허용할 수 있는 최대 데이터 손실 기준을 의미
        - ▪ 질문 기준: 어느 시점의 데이터 백업본으로 돌아갈 것인가? 또는 데이터를 얼마나 잃어도 비즈니스에 지장이 없는가?
        - ▪ 예시
          - RPO = 24시간: 하루에 한 번 밤 12시에 백업을 받는 시스템. 오후 5시에 장애가 나면 오늘 새벽부터 오후 5시까지의 17시간 치 데이터는 손실
          - RPO = 0: 실시간 동기 복제를 수행하여 장애 순간에도 데이터 손실이 0인 상태

클라우드에 필요한 기술

- ❖ 장애 대응
  - ✓ 고가용성 구성
    - ● 재해 복구 목표 수립
      - ◆ RTO(복구 시간 목표)
        - ▪ 장애 발생 시점부터 시스템이 정상 가동될 때까지 허용되는 최대 복구 시간을 의미
        - ▪ 질문 기준: 서비스가 멈춘 후 다시 켜질 때까지 얼마나 걸리는가? 또는 다운타임을 얼마나 버틸 수 있는가?
        - ▪ 예시
          - RTO = 4시간: 장애 발생 후 복구 작업, 서버 재부팅, 네트워크 전환 등을 마쳐 4시간 이내에 서비스를 정상화해야 함을 의미
          - RTO = 0 (초 단위): 장애 즉시 대기 시스템으로 자동 전환(Failover)되어 사용자가 장애를 인지하지 못하는 수준

클라우드에 필요한 기술

- ❖ 장애 대응
  - ✓ 백업 정책
    - ● 백업 정책 설계 시 고려할 사항
      - ◆ 어떤 데이터를 백업
      - ◆ 얼마나 자주 백업
      - ◆ 백업 데이터를 어디에 저장
      - ◆ 백업된 데이터를 어떻게 암호화되고 보호
      - ◆ 백업 데이터를 얼마나 오래 보관
    - ● 중요한 백업은 다른 지역에 보관
    - ● 중요한 데이터를 다른 리전에 한 번 더 저장하고 복구 테스트를 해보는 것 부터 시작

클라우드에 필요한 기술

- ❖ 주요 클라우드 플랫폼 비교
  - ✓ AWS & Azure & GCP
    - ● 공통점
      - ◆ 서비스 모델 공유: 모두 IaaS(인프라), PaaS(플랫폼), SaaS(소프트웨어) 등 컴퓨팅, 저장소, 네트워크, 데이터베이스, AI/ML 전반에 걸친 종합 클라우드 리소스를 제공
      - ◆ 책임 공유 모델(Shared Responsibility Model): 인프라 및 물리적 보안은 클라우드 제공업체가 담당하고, 데이터, IAM(계정/권한), 애플리케이션 보안은 고객이 담당하는 보안 모델을 동일하게 적용
      - ◆ 유연한 과금 정책: 초기 구축 비용 없이 사용한 만큼 지불하는 종량제(Pay-as-you-go) 방식과 약정 할인(Reserved Instance / Committed Use Discount) 제도를 운영
      - ◆ 글로벌 인프라 및 고가용성: 전 세계 데이터센터(Region 및 Availability Zone)를 보유하여 멀티 리전 백업 및 고가용성(HA) 환경을 구성할 수 있음

클라우드에 필요한 기술

- ❖ 주요 클라우드 플랫폼 비교
  - ✓ AWS & Azure & GCP
    - ● 차이점

구분	AWS	Azure	GCP
시장 점유율	1위 (약 31%)	2위 (약 23~25%)	3위 (약 11~12%)
주요 강점	압도적인 서비스 다양성, 성숙도, 방대한 서드파티 생태계	MS 제품군(Office, Active Directory 등)과의 완벽한 연동, 강력한 하이브리드 클라우드	빅데이터 분석, 쿠버네티스(컨테이너) 최적화, 우수한 네트워크 성능
핵심 서비스	EC2, S3, RDS, Lambda, SageMaker, Bedrock	Azure VMs, Blob Storage, Azure SQL, Azure Open AI	Compute Engine, GCS, Cloud Spanner, GKE, BigQuery, Vertex AI
AI / ML 특화	Bedrock(다양한 LLM 선택권), SageMaker	Azure OpenAI Service(GPT-4o 등 exclusive 접근성)	Gemini, Vertex AI, 고성능 AI 칩셋 TPU
주요 타깃	스타트업부터 대기업까지 범용적인 환경	기존 Windows/MS 엔터프라이즈 인프라 사용 기업	데이터/AI 기반 기업, 테크 스타트업, 쿠버네티스 사용팀

클라우드에 필요한 기술

- ❖ 주요 클라우드 플랫폼 비교
  - ✓ AWS & Azure & GCP
    - 특징
      - ◆ AWS: 설정 가능한 항목이 많고 자유도가 높은 대신 진입 장벽이 높을 수 있음
      - ◆ Azure: Microsoft 환경에 익숙한 사용자에게 직관적인 구성과 연계를 제공
      - ◆ GCP: 간결하고 개발자 친화적인 UI, 문서 구조, 설정 흐름을 지니고 있음
    - 서비스 추천
      - ◆ AWS를 추천하는 경우: 가장 방대한 레퍼런스와 레고 블록처럼 다양한 서비스를 자유롭게 조합하여 완성형 표준 클라우드를 구축하고 싶을 때
      - ◆ Azure를 추천하는 경우: 기존에 Microsoft 계정(Active Directory), Windows Server, Office 365, .NET 기술 스택을 많이 사용하는 기업이나, 온프레미스와의 하이브리드 환경 구축이 필수일 때
      - ◆ GCP를 추천하는 경우: BigQuery를 활용한 대용량 실시간 데이터 분석, Kubernetes(GKE) 중심의 컨테이너 환경 구축, 혹은 Google Gemini/TPU 기반의 대규모 AI 연구개발을 진행할 때

클라우드에 필요한 기술

- ❖ 주요 클라우드 플랫폼 비교
  - ✓ 국내 클라우드
    - ● 특징

기업	주요 특화 분야	핵심 강점 및 특징
네이버클라우드	AI 생태계 소버린 클라우드 국내 CSP 점유율 1위	• 자체 초거대 AI '하이퍼클로바X' 연계 서비스 • 국내 최대 규모 IaaS/PaaS/SaaS 라인업(220개 이상) • 금융, 의료, 공공 분야 민감 데이터용 온프레미스형 뉴로클라우드 보유
KT 클라우드	공공/금융 인프라 인터넷 데이터센터(IDC)	• 전국 13개 이상 IDC 운영으로 인프라 규모 및 통신망 우수 • 국내 최초 CSAP 인증 기반 레퍼런스로 공공 클라우드 점유율 최상위 • 통신 연계 엣지 컴퓨팅 및 MS 협력 기반 보안 퍼블릭 클라우드 추진
NHN 클라우드	게임 인프라 공공 AI 데이터센터 SaaS 연계	• 20년 이상 게임 서비스 운영 노하우 기반 게임 특화 인프라/보안 • 광주 국가 AI 데이터센터 구축 및 공공 클라우드 네이티브 전환 주도 • 두레이(Dooray!), 메시징 등 다양한 업무용 PaaS/SaaS 라인업
카카오 클라우드	개발자 친화형 멀티 클라우드 AI/모바일 연계	• 쿠버네티스(Kubernetes) 중심의 현대적 클라우드 네이티브 환경 지원 • 카카오 플랫폼 기반의 대규모 트래픽 처리 역량 및 개발자 중심 콘솔 • 카카오 i(인공지능) 솔루션 및 모바일 생태계와의 높은 연계성

클라우드에 필요한 기술

- ❖ 주요 클라우드 플랫폼 비교
  - ✓ 국내 클라우드
    - ● 특징
      - ◆ 공공 및 보수적 금융권: 전통적인 강자인 KT 클라우드와 네이버클라우드 선호도가 높음
      - ◆ AI 도입 및 대규모 B2B 솔루션: 자사 거대언어모델(LLM)과 최신 AI 플랫폼을 갖춘 네이버클라우드와 카카오 클라우드가 강점을 보임
      - ◆ 게임 및 엔터테인먼트/SaaS: 게임 인프라 노하우와 데이터센터 고가용성이 검증된 NHN 클라우드가 우수한 선택지로 꼽힘

클라우드에 필요한 기술

- ❖ 주요 클라우드 플랫폼 비교
  - ✓ 클라우드 선택 기준
    - 기술적인 적합성
    - 비용
    - 지역성과 법적 요구사항
    - 각 기업의 전략 방향
    - 조직의 클라우드 성숙도
    - 내부 보안 정책
    - SLA 보장 범위
    - 마이그레이션 난이도

클라우드에 필요한 기술

- ❖ 주요 클라우드 플랫폼 비교
  - ✓ 하이브리드 와 멀티 클라우드
    - 하이브리드 클라우드 (Hybrid Cloud)
      - ◆ 온프레미스(사내 자체 데이터센터) 또는 프라이빗 클라우드와 퍼블릭 클라우드를 서로 연결(통합)하여 하나의 시스템처럼 운영하는 형태
      - ◆ 핵심 목적: 보안성과 유연성의 균형
      - ◆ 동작 방식: 민감한 고객 데이터나 핵심 시스템은 내부 프라이빗 서버에 보관하고 대용량 트래픽 처리나 웹 서비스는 확장성이 뛰어난 퍼블릭 클라우드를 활용
      - ◆ 예시: 금융권이나 의료기관에서 규제/법적 문제로 고객 정보 DB는 온프레미스에 두고 이벤트용 웹 페이지는 AWS를 활용하는 경우
    - 멀티 클라우드 (Multi-Cloud)
      - ◆ 서로 다른 CSP(클라우드 서비스 제공업체)의 퍼블릭 클라우드를 2개 이상 조합하여 사용하는 방식
      - ◆ 핵심 목적: 특정 벤더 종속(Lock-in) 방지 및 각 클라우드의 특화 기능 활용
      - ◆ 동작 방식: 각 클라우드의 장점만을 골라 애플리케이션별로 다르게 배치하거나 단일 클라우드 장애 발생 시 복구용(DR)으로 병행 활용
      - ◆ 예시: 메인 서버는 AWS를 사용하고 대용량 데이터 분석 및 AI 모델링은 GCP(BigQuery, Vertex AI)를 사용하며 MS 인프라 연동은 Azure를 사용하는 경우
        .