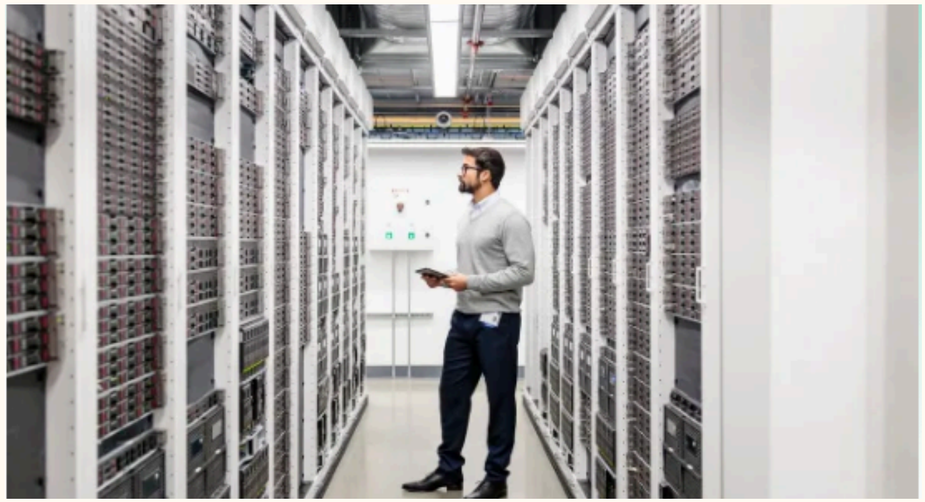
클라우딩 컴퓨팅

설현원

전통적인 컴퓨팅
(온프레미스, On-Premise)

기업이 서버·스토리지·네트워크 장비를 직접 구매하고,
자체 데이터센터에 설치하여 운영하는 방식



문제	설명
💸 비용 문제	서버·장비 구매비 + 유지보수 인건비 부담
🕐 확장성 한계	트래픽 급증 시 즉시 서버 증설 불가
🧑‍🔧 관리 복잡성	OS·보안패치·백업 등 반복 작업
🌍 글로벌 서비스 필요	해외 유저에게는 지연(latency) 발생

“서버를 사서 직접 돌리던 시대”
— 빠른 확장과 유연성이 부족했다.

클라우딩 컴퓨팅

“필요한 만큼 빌려 쓰는 컴퓨팅”

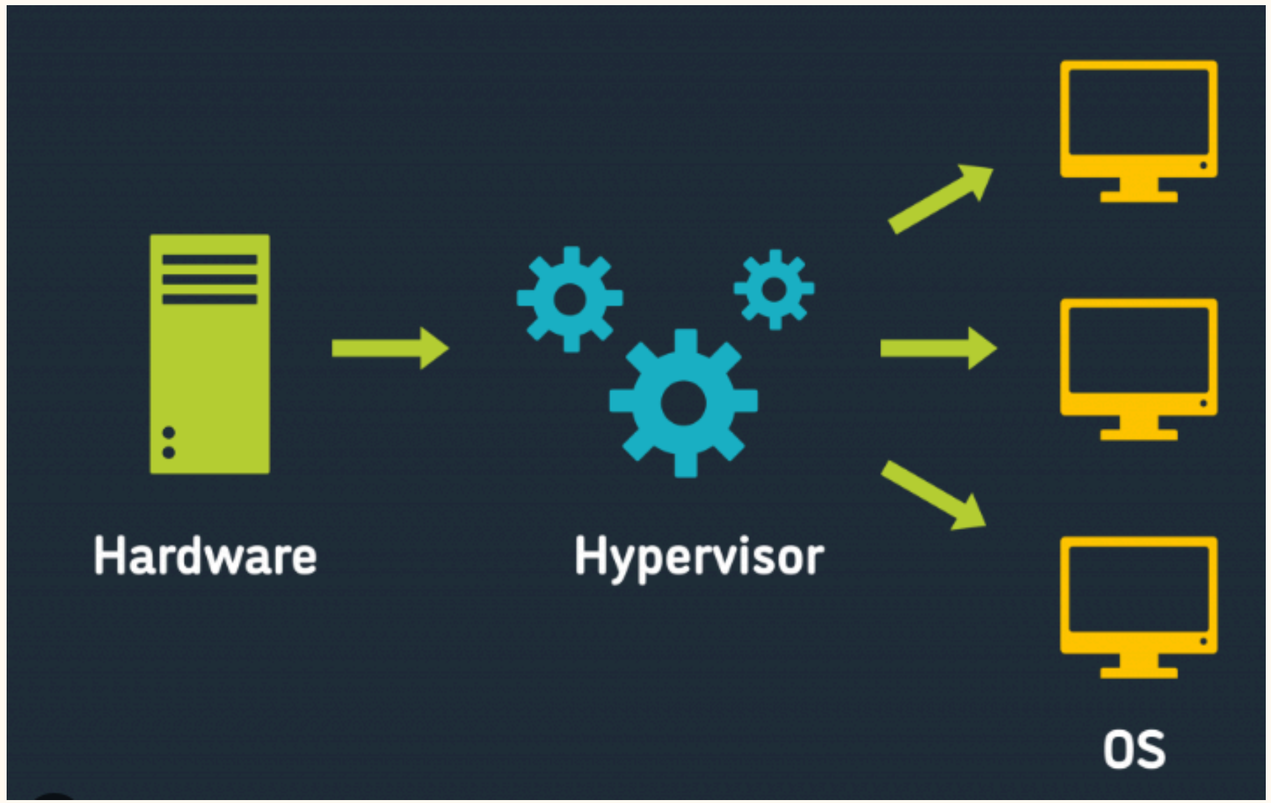
구분	장점	설명
💰 비용 효율성	초기 투자 없음	사용량만큼 지불 (운영비 중심, OpEx)
⚡ 확장성 & 유연성	자원 자동 증감	트래픽 폭주 대응 용이
🛠️ 운영 효율	하드웨어 관리 불필요	인프라 유지보수 부담 감소
🌐 접근성	인터넷만 있으면 사용	장소/기기 제한 없음
🔒 안정성 & 보안	고가용성, 백업 자동화	클라우드 제공업체가 관리
🧩 글로벌 확장	전 세계 리전(Region) 지원	해외 유저에게 빠른 서비스

과거엔 서버를 “소유”했다면,
지금은 컴퓨팅 자원을 “서비스로 이용”하는 시대다.

가 상 화

항목	설명
비용 절감	하드웨어 구매·유지비용 절약
확장성	VM을 빠르게 복제·확장 가능
복원력	장애 발생 시 VM 스냅샷으로 신속 복구
보안성	VM 간 격리로 한 시스템의 장애가 다른 시스템에 영향 X
테스트 용이성	다양한 OS·환경을 손쉽게 실험 가능

하이퍼바이저(Hypervisor)



리 소 스　풀 링

공유(Shared Resource)

모든 자원(서버·저장소·네트워크 등)은 중앙의 풀(pool)에서 관리됨

사용자가 요청할 때마다 풀에서 필요한 만큼 자원을 “할당”받음

사용이 끝나면 자원을 다시 풀로 “반납”

멀티 테넌시(Multi-tenancy)

여러 사용자가 같은 인프라를 공유하지만,

데이터와 환경은 서로 완전히 논리적으로 분리(격리) 되어 있음

이를 통해 보안과 안정성을 동시에 확보함

[사용자 A 요청] → [자원 풀에서 자원 할당]
[사용자 B 요청] → [남은 자원 중 일부 할당]
[사용자 A 작업 종료] → [자원 반환 → 풀로 복귀]

자동화 및 오케스트레이션 (Automation & Orchestration)

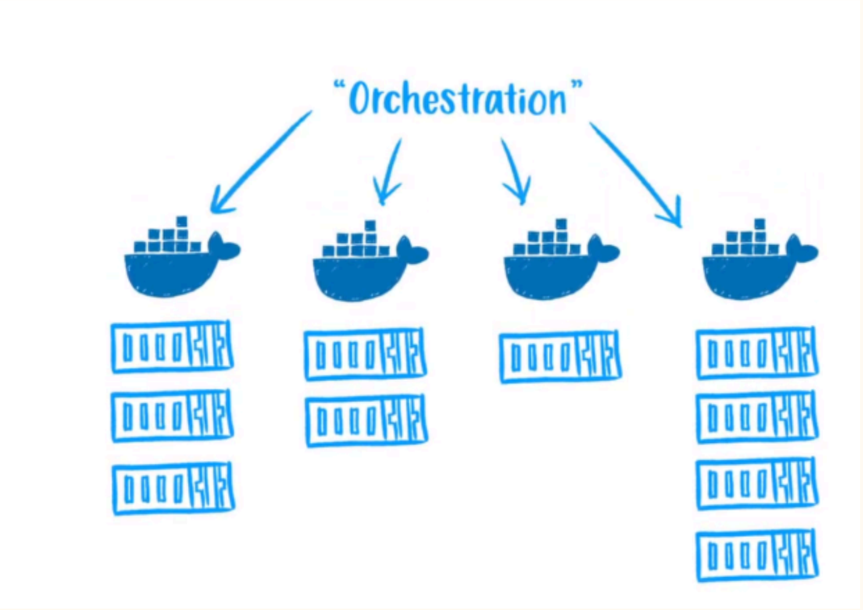
자동화 (Automation)

반복적인 IT 작업(배포, 백업, 설정 등)을 자동으로 처리하는 것
서버 자동 생성, 스토리지 자동 연결

오케스트레이션 (Orchestration)

여러 자동화된 작업들을 논리적 순서에 따라 연결·관리하는 것
앱 배포 시 VM 생성 → 네트워크 설정 → DB 연결 순으로 수행

사용자 요청 → 자동화 엔진(Automation)
→ 오케스트레이터(Orchestrator)
→ 리소스 풀(Resource Pool) 내 자원 배분

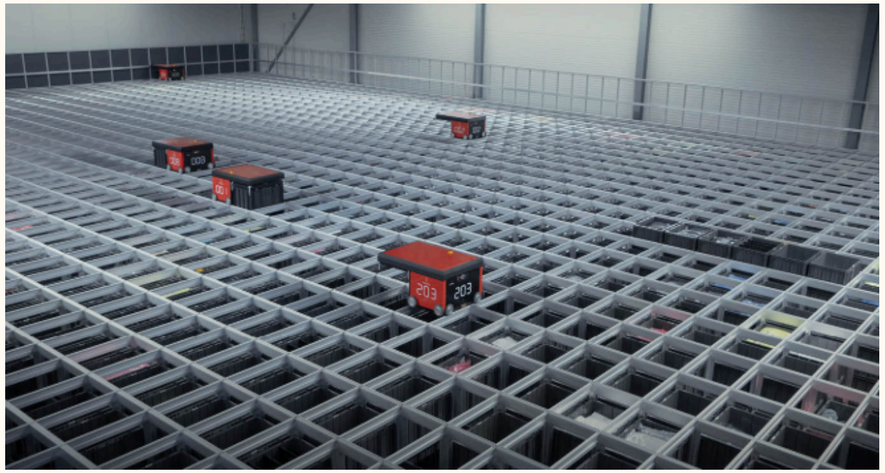


유연한 확장성 (Elasticity)

클라우드 자원을 사용량에 따라 자동으로 확장(Scale-out)
하거나 축소(Scale-in) 하는 기능

구분	설명	예시
수직 확장 (Vertical Scaling)	한 서버의 성능(CPU, RAM 등)을 높이는 방식	EC2 인스턴스 타입을 t2.micro → t2.large로 변경
수평 확장 (Horizontal Scaling)	서버 개수를 늘리는 방식	웹 서버를 여러 대 추가해 트래픽 분산 처리

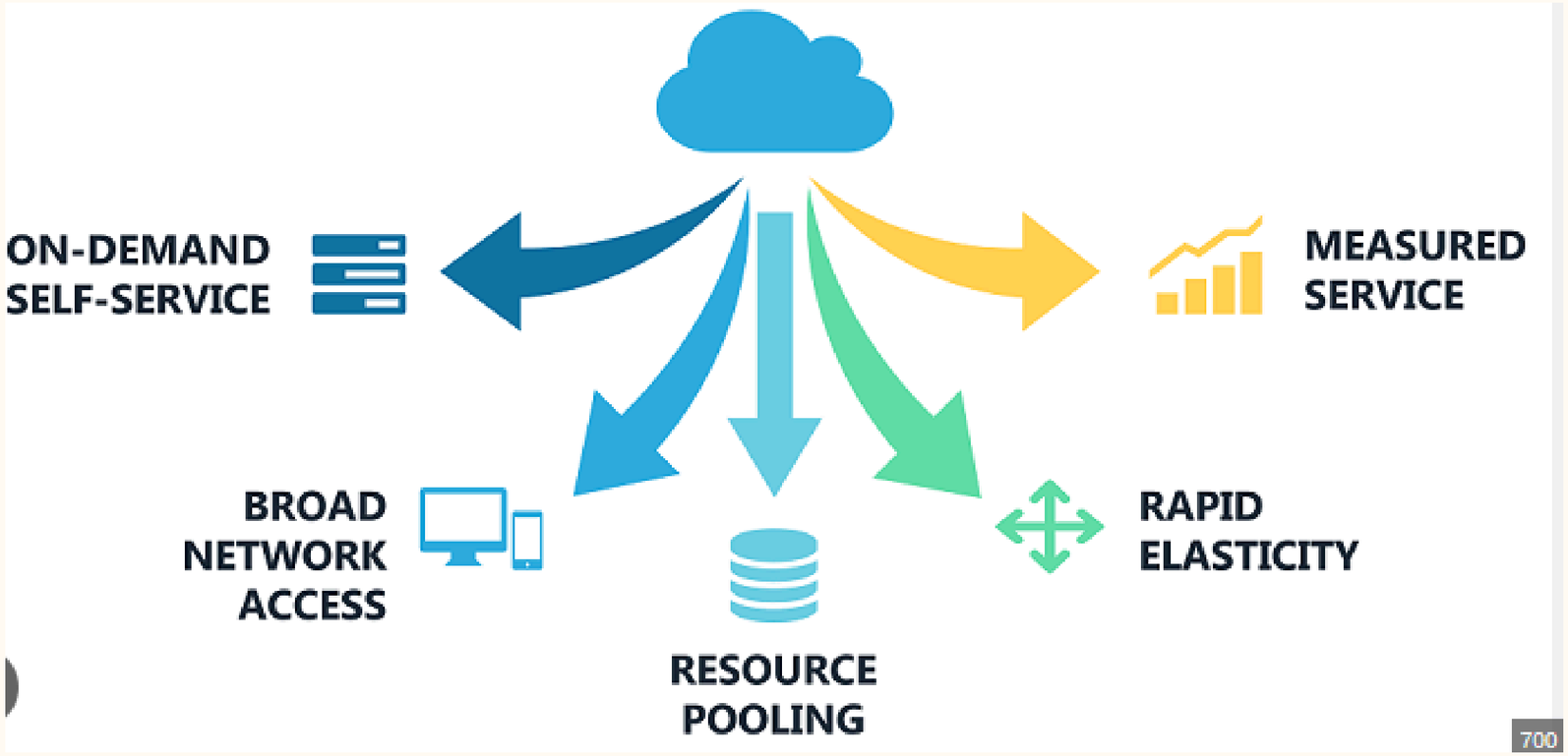
"트래픽이 몰리면 서버를 늘리고, 줄어들면
자동으로 줄이는 시스템"



측정 가능한 서비스(Measured Service)

클라우드 제공자가 각 사용자의 자원 사용량을 자동으로 감시·기록·계량하여
이에 따라 요금을 부과(pay-as-you-go) 하는 기능

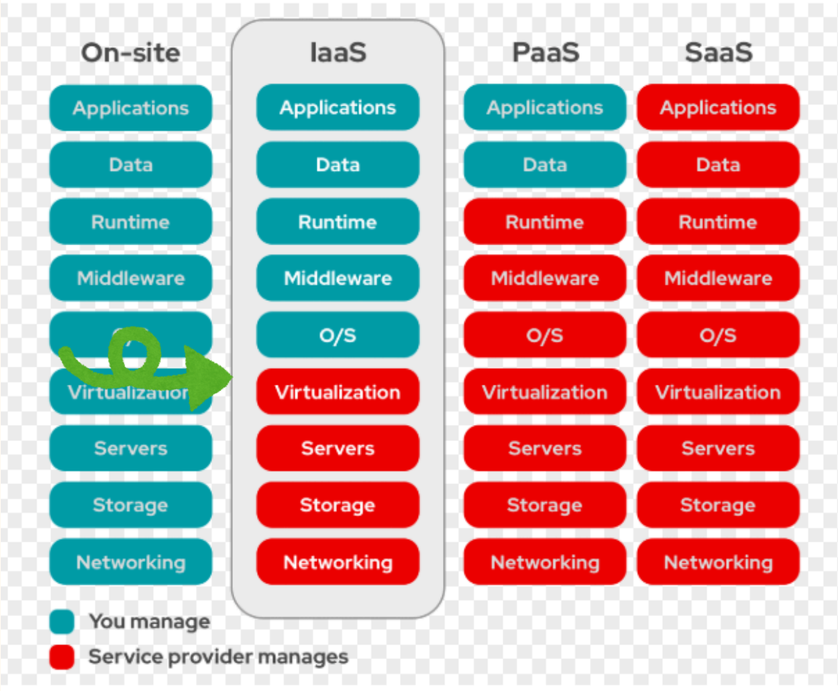
단계	설명	예시
모니터링(Monitoring)	자원 사용 상태를 실시간으로 감시	CPU, RAM, 네트워크 트래픽, API 호출 수 등
계량(Metering)	감시된 데이터를 수량으로 계산	초당 트래픽, 저장 용량(GB), 시간당 실행 횟수 등
과금(Billing)	요금 정책에 따라 금액을 산정	“1시간당 0.01달러” 식의 계산 후 자동 청구


ON-DEMAND
SELF-SERVICE
MEASURED
SERVICE
BROAD
NETWORK
ACCESS
RAPID
ELASTICITY
RESOURCE
POOLING
700

클라우드 서비스 모델

IaaS (Infrastructure as a Service)

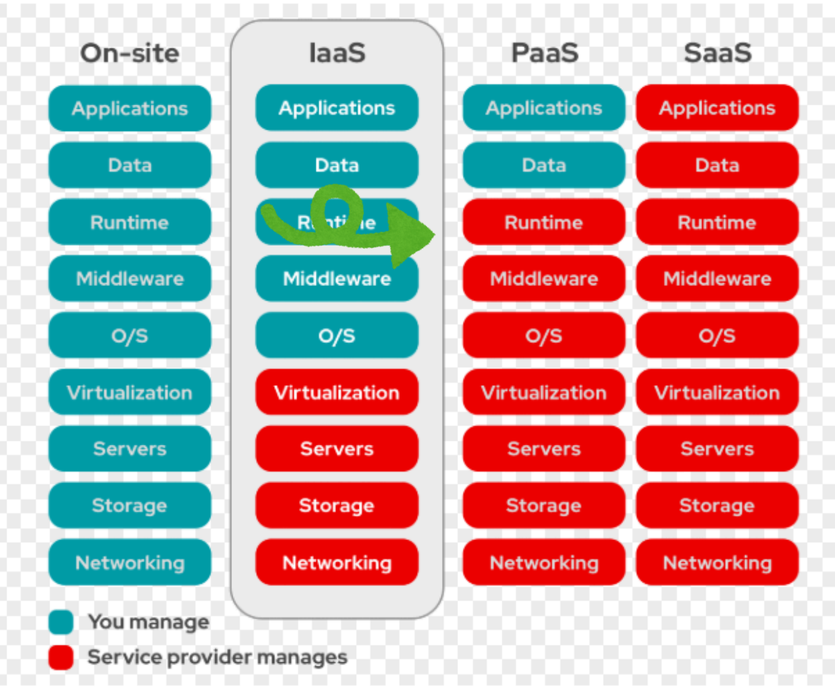
서버, 네트워크, 스토리지 같은 **하드웨어 인프라**를 가상화하여 제공



- AWS EC2AWS EC2 (가상 서버)
- Microsoft Azure VM
- Google Compute Engine
- OpenStack

PaaS (Platform as a Service)

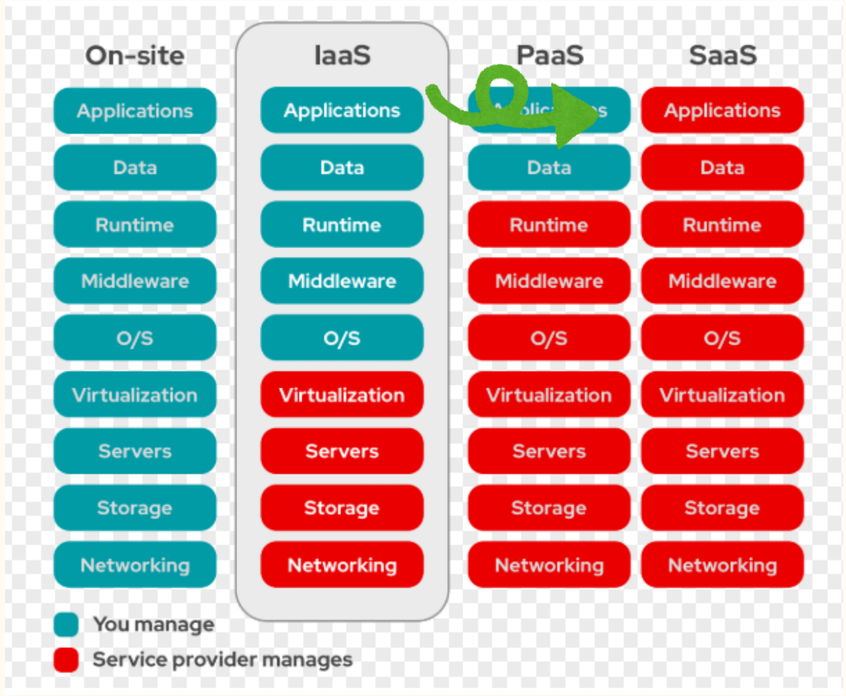
개발자용 플랫폼 서비스로,
애플리케이션을 개발·배포할 수 있는 운영 환경 자체를 제공



- Google App Engine
- AWS Elastic Beanstalk
- Microsoft Azure App Service
- Heroku

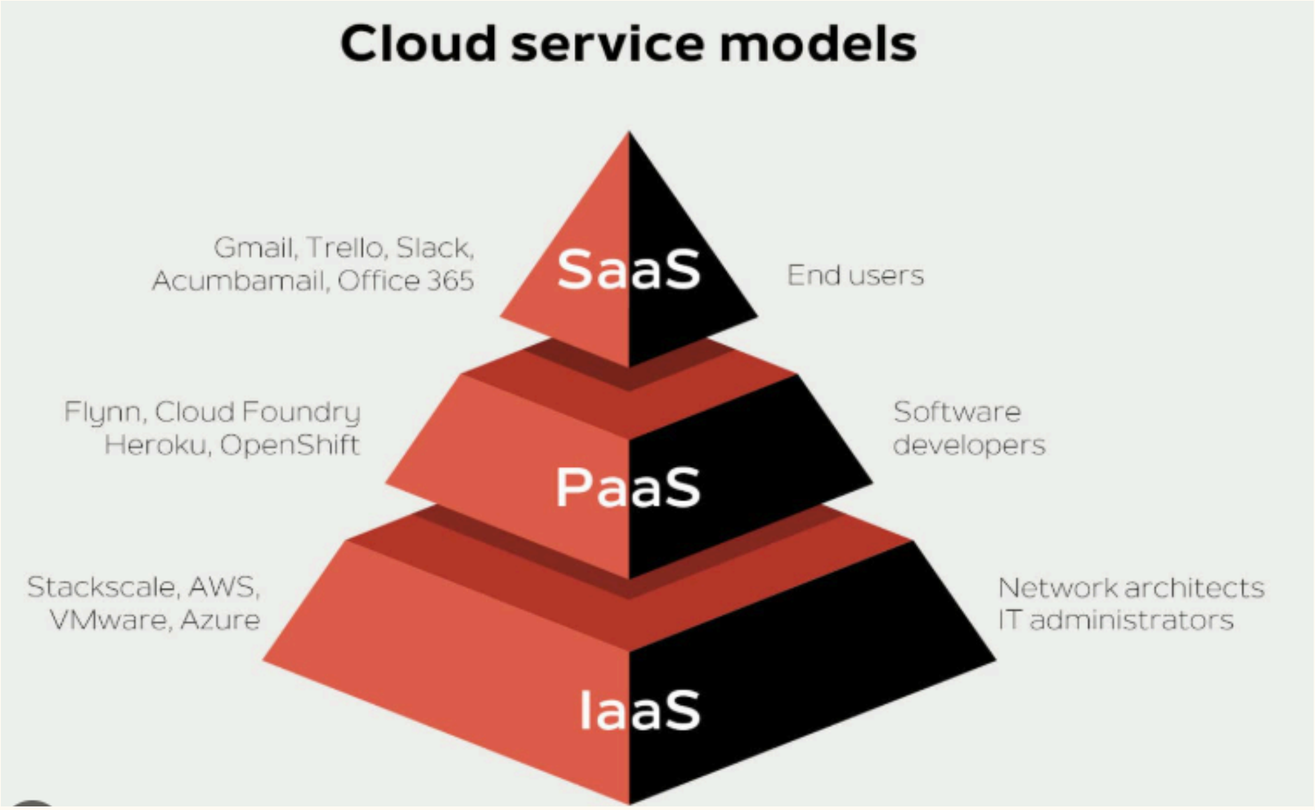
SaaS (Software as a Service)

가장 완성도 높은 형태의 클라우드 서비스로,
사용자가 직접 설치하지 않고 인터넷을 통해 소프트웨어를 바로 사용하는 형태



- Google Workspace (Docs, Gmail, Drive 등)
- Microsoft 365

상위 계층으로 갈 수록 편의성은 높지만 제어권은 줄어듦

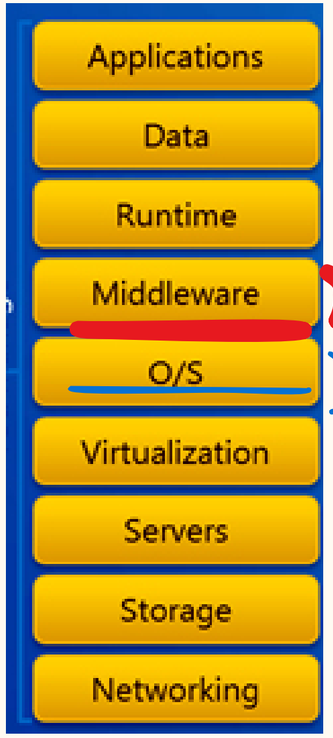


클라우드 계층

② 가상화 계층 (Virtualization Layer
물리 인프라를 가상화(Virtualization) 기술로 추상화하여
하나의 서버를 여러 개의 “가상 서버(Virtual Machine, VM)”로 나누는 층.

① 물리 인프라 계층 (Physical Infrastructure Layer)

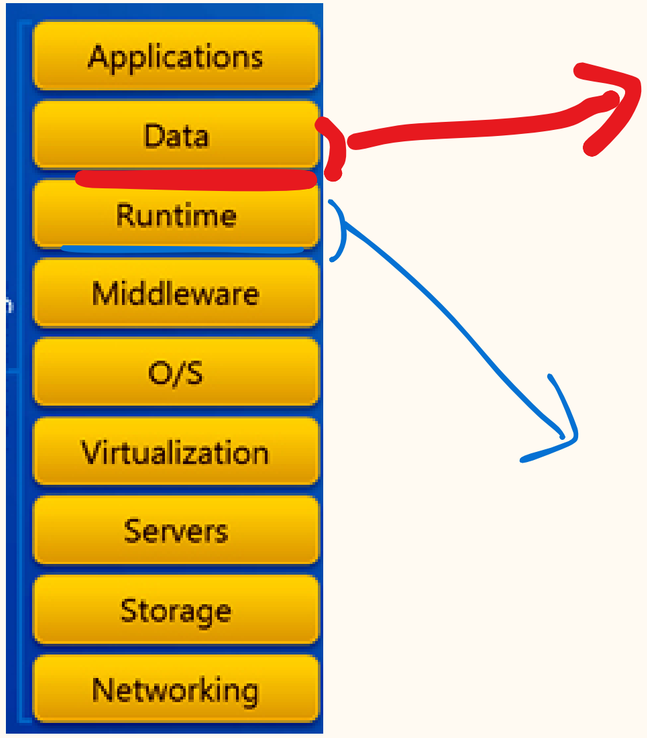
클라우드의 가장 하단, 실제 ‘하드웨어’ 영역.
데이터센터에 위치한 서버, 스토리지 장치, 네트워크, 전력 설비 등이 포함



④ 미들웨어 계층 (Middleware Layer)

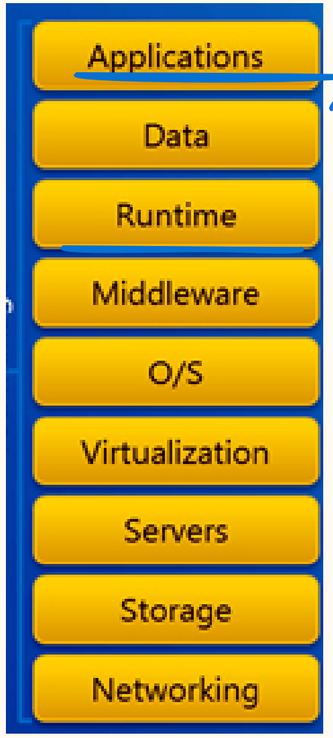
운영체제와 애플리케이션 사이에서 데이터 전달·연결을 담당하는 중간 소프트웨어 계층.
즉, 개발자가 OS의 세부 코드를 몰라도 서비스를 구현할 수 있게 돕는 “중간 다리”.

③ 운영체제 계층 (Operating System Layer
가상화된 서버(VM) 위에 올라가는 운영체제(OS) 계층
OS는 하드웨어 자원을 제어하고, 애플리케이션이 동작할 수 있도록 환경을 제공



⑥ 데이터 계층 (Data Layer)
서비스나 앱이 사용하는 모든 데이터의 저장·처리·백업을 담당하는 층입니다.

⑤ 런타임 계층 (Runtime Layer)
애플리케이션이 실제로 실행되는 환경(Execution Environment) .
즉, 코드가 “돌아가는” 곳이에요.



⑦ 애플리케이션 계층 (Application Layer)

사용자에게 직접 제공되는 서비스 그 자체
웹사이트, 모바일 앱, 클라우드 기반 프로그램 등이 여기에 해당.