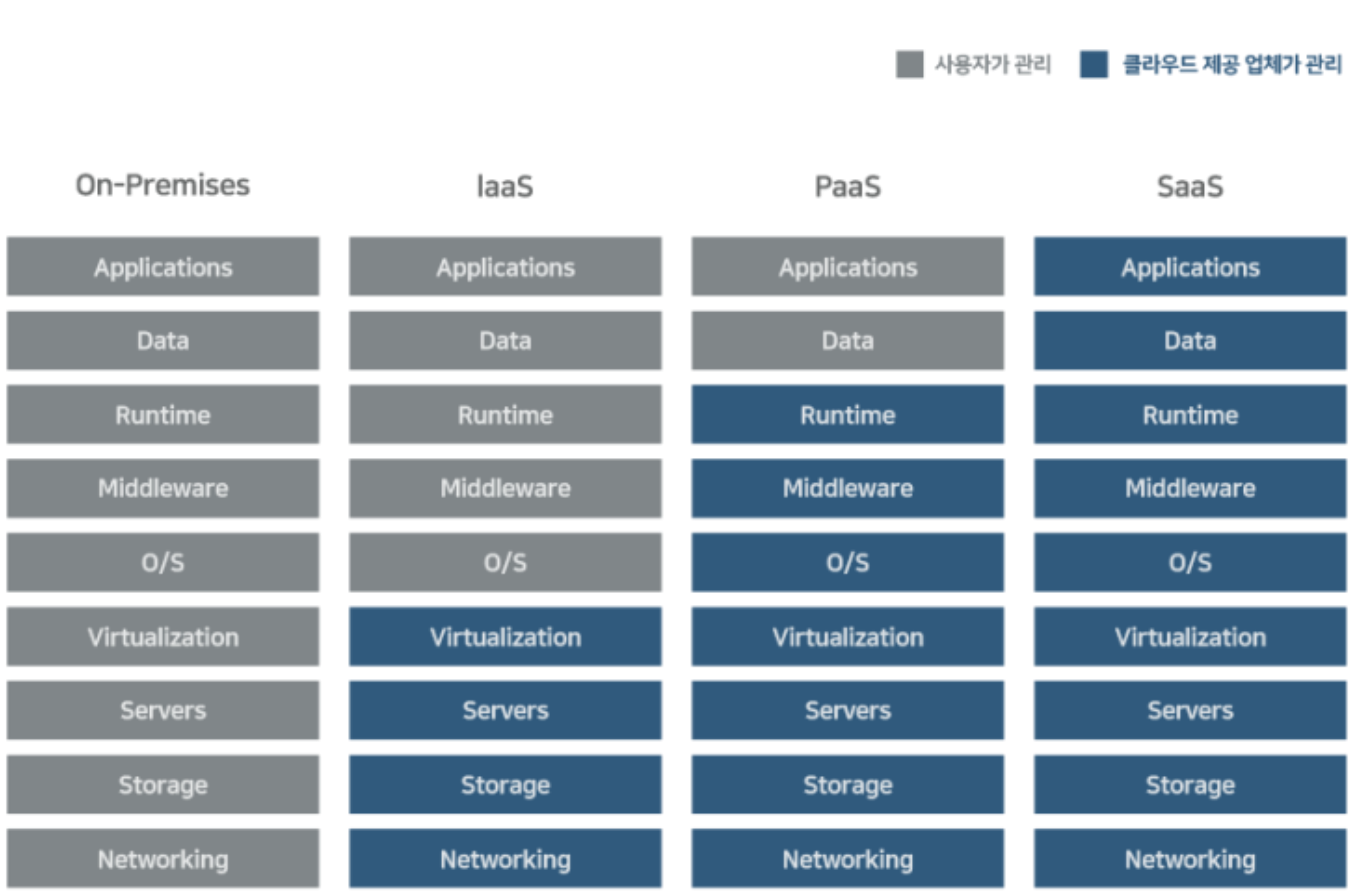
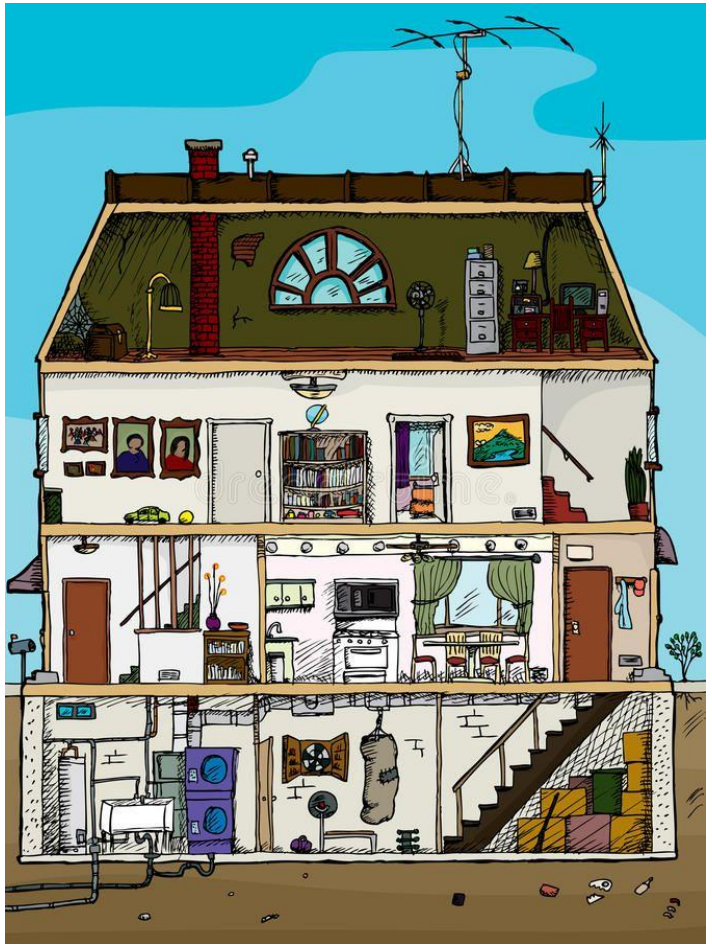
클라우드 서비스 모델

IaaS / PaaS / SaaS

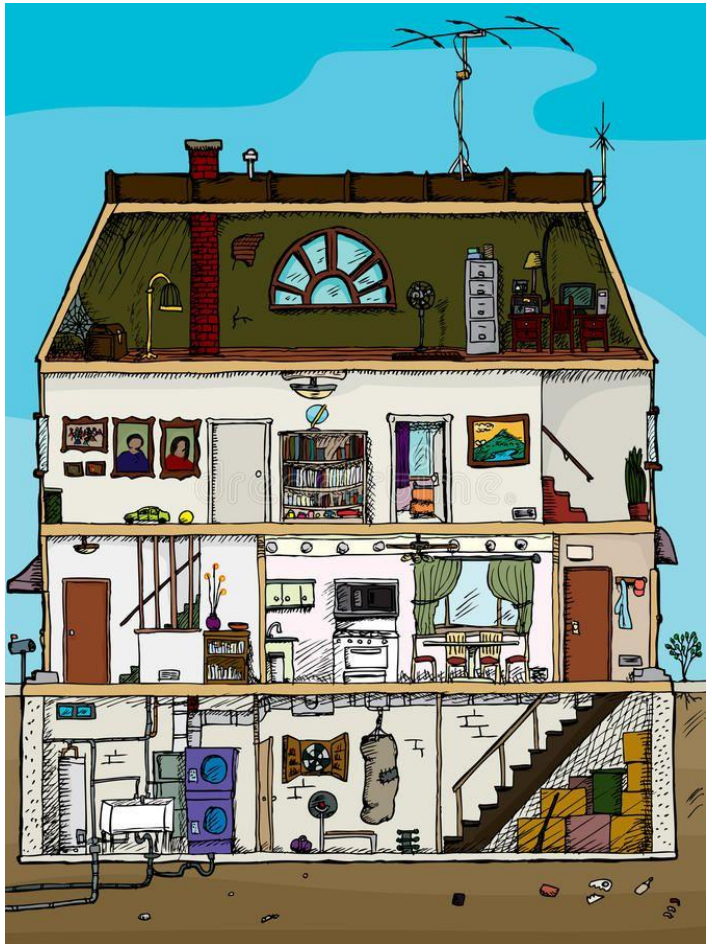
이지호(Jiho Lee)

On-premise vs **IaaS** vs **PaaS** vs **SaaS**





On-premise vs IaaS vs PaaS vs SaaS



🟫 **땅 (Infrastructure 기반)**
Networking / Storage / Servers / Virtualization

- **Networking(네트워킹, 통신망)**
→ 도로, 상하수, 전기, 통신망
✓ 집이 혼자 있어도, 연결되지 않으면 아무 의미 없음
- **Storage(스토리지, 데이터 저장공간)**
→ 땅속 창고, 지하 저장고
✓ 물건을 오래 보관하는 공간
- **Servers(서버, 프로그램 실행 및 요청 처리)**
→ 실제 건물이 올라가는 대지
✓ 집이 올라갈 물리적 공간
- **Virtualization(가상화, 물리 서버를 가상화&나눠쓰기)**
→ 토지 분할, 필지 나누기
✓ 하나의 땅을 여러 집이 공유 가능하게 만드는 기술

🏠 **집 (운영 환경)**
OS / Middleware / Runtime

- **OS(운영체제, 서버 동작 운영체제)**
→ 집의 구조, 건축 방식
✓ 벽, 문, 바닥 같은 기본 규칙
- **Middleware(미들웨어, OS-앱 사이 공통 기능 제공)**
→ 배관, 가스관, 엘리베이터
✓ 집 안 기능들을 연결해주는 설비
- **Runtime(런타임, 앱 실행 환경)**
→ 전기·수도 공급 상태
✓ 가전이 '실제로 작동'할 수 있게 해줌

📺 **가전·가구 (사용자 가치)**
Applications / Data

- **Applications(애플리케이션, 직접 사용 서비스/프로그램)**
→ TV, 냉장고, 세탁기
✓ 사용자가 '클라우드를 쓴다'고 느끼는 지점
- **Data(데이터, 앱에서 생성·저장·처리하는 정보)**
→ 냉장고 속 음식, TV 속 콘텐츠
✓ 가전의 실질적인 가치

On-premise 환경

- 앞에서 본 집 그림으로 치면?

☞ 땅을 사서 직접 지하수도 파고 전력도 끌어오고 지반을 다져서 타설도 하고 지하실부터 1, 2층에 다락방까지 직접 짓고 가전 및 가구도 일일이 사서 배치

- 장점 : 집 구조 ~ 콘센트 위치까지 내가 정함, 예측 가능한 비용 구조, 외부 환경변화의 영향이 적음

- 단점 : 초기 비용이 많이 들며, 필요할 때 확장/축소가 어렵고 구축, 운영, 유지보수의 모든 책임을 직접 부담해야 함

On-premise 환경

주요 특징	내용	세부내용
완전한 통제권	• 모든 계층을 직접 통제 가능	• 하드웨어, OS, 네트워크 설정까지 전부 내 결정 • 보안 정책·접근 통제·로그 관리도 조직 기준 그대로 적용 가능
예측 가능한 비용 구조	• 초기 비용은 크지만, 장기 운영 비용은 예측 가능	• 트래픽 증가로 요금 폭탄 맞을 일 없음 • 내부 회계·예산 관리에 유리
규제·보안·정책 대응에 유리	• 데이터 위치·접근 경로를 명확히 통제 가능	• 금융, 공공, 의료 등 규제 산업에 적합 • 데이터가 어디에 있는지 설명할 수 있음
커스터마이징 자유 극대화	• 특정 업무에 맞춘 극단적 최적화 가능	• 특수 하드웨어, 레거시 시스템, 맞춤형 미들웨어 구성 • 클라우드에서 허용되지 않는 설정도 가능
외부 환경 변화 영향 최소화	• 클라우드 사업자 정책 변경·서비스 종료 영향 없음	• 가격 인상, 리전 변경, 서비스 디프리케이션과 무관 • 장기 시스템 운영에 안정적

IaaS(Infrastructure as a Service)

- 앞에서 본 집 그림으로 치면?

☞ 땅은 이미 정비된 곳을 임대하여 전력·통신·기반 공사는 제공받은 상태에서, 집 구조는 직접 설계·시공하며 가전 및 가구도 필요한 만큼 직접 구매·배치

- 장점 : 초기 인프라 구축 부담 감소, 필요 시 집을 빠르게 증축·철거 가능, 기반 인프라는 담당 사업자가 관리

- 단점 : 건물과 내부는 여전히 직접 관리해야 함, 집을 넓게 쓰거나 가전을 많이 들일수록 월 유지비의 변동성이 존재함

IaaS(Infrastructure as a Service)

주요 특징	내용	세부내용
부분적 관리 책임	• 사용자와 클라우드 제공자 간 인프라 관리 책임 분리	• 서버·스토리지·네트워크는 제공자가 관리 • OS 이상 계층은 사용자가 직접 관리
초기 투자 부담 감소	• 하드웨어 구매 없이 인프라 사용 가능	• 서버·스토리지 장비 구매 불필요 • 초기 CAPEX* 없이 즉시 사용 가능
확장성 및 탄력성	• 수요 변화에 따라 자원을 즉시 확장·축소 가능	• 트래픽 증가 시 빠른 자원 증설 • 사용량 감소 시 자원 회수 가능
운영 책임 지속성	• 시스템 운영·보안 책임은 여전히 사용자에게 존재	• OS 패치, 보안 설정, 미들웨어 관리 필요 • 장애 대응 및 운영 자동화 직접 수행
사용량 기반의 비용구조	• 사용한 만큼 비용을 지불하는 구조	• 고정비가 아닌 OPEX* 중심 • 자원 사용량 증가 시 비용 증가

CAPEX(Capital Expenditure, 자본적 지출) : 한 번에 크게 투자해서 오래 쓰는 비용 || **OPEX(Operating Expenditure, 운영비)** : 쓰는 만큼 계속 나가는 비용

PaaS(Platform as a Service)

- 앞에서 본 집 그림으로 치면?

☞ 땅과 기반 시설은 물론, 집의 골조·전기·수도·보일러까지 이미 완성된 집에 입주해, 가구를 들이고 공간을 쓰기만 하면 되는 상태

- 장점 : 초기 인프라 구축 부담이 적고 입주가 간단, 필요에 따라 방을 더 쓰거나 덜 쓰는 등 선택지 존재, 대지·전기·수도·건물·보일러 고장 등의 문제는 집주인이 처리, 거주자는 집 관리보다 생활 자체에 집중 가능

- 단점 : 벽을 허물거나 구조를 바꾸는 등 집의 근본적 형태 변경은 어려움, 정해진 구조에 맞춰 살아야 하므로 생활 방식의 제약 발생, 개인 취향이 강한 경우 답답할 수 있음

PaaS(Platform as a Service)

주요 특징	내용	세부내용
애플리케이션 중심 운영	• 인프라·플랫폼 계층을 신경 쓰지 않고 애플리케이션 개발·실행에 집중 가능	• 서버, OS, 미들웨어, 런타임은 클라우드 제공자가 관리 • 사용자는 애플리케이션 코드와 데이터만 관리
개발 생산성 극대화	• 개발·배포·운영에 필요한 환경이 사전 구성되어 즉시 사용 가능	• 개발 환경 구성 시간 최소화 • 개발부터 배포, 상태 확인까지 필요한 기본 도구들이 갖춰져 있음
운영 부담 대폭 감소	• 시스템 운영·유지보수 책임이 상당 부분 클라우드 제공자에게 이전	• 시스템 보안과 버전 관리를 자동으로 처리해 줌 • 장애 대응·확장 작업이 플랫폼 단위로 수행
자동 확장성 내재	• 트래픽 변화에 따라 애플리케이션 단위로 자동 확장·축소	• 별도 장비 추가 없이 서비스 규모를 자동으로 늘릴 수 있음 • 접속자가 급증하는 상황에도 비교적 안정적으로 대응 가능
플랫폼 종속성 존재	• 특정 클라우드 제공자의 플랫폼 구조에 의존	• 다른 환경으로 이전 시 재구현 필요 가능성 • 사용 가능한 언어·프레임워크에 제약 존재

SaaS(Software as a Service)

- 앞에서 본 집 그림으로 치면?

☞ 땅과 기반 시설은 물론, 집의 골조·전기·수도·보일러까지 이미 완성된 집에 가구와 가전, 기본 생활도구까지 모두 갖춰진 집에 바로 들어가 생활하는 상태

- 장점 : 집을 짓거나 꾸밀 필요 없이 바로 입주 가능, 관리사업자가 전반적으로 생활 공간을 다 관리, 고장, 점검, 관리 등의 문제 신경 쓸 필요 없이 생활에만 집중 가능

- 단점 : 집 구조나 가구 배치를 바꾸는 것은 거의 불가능, 제공된 방식대로 살아야 하므로 생활 방식의 선택 폭이 좁음, 집이 마음에 들지 않으면 다른 집으로 옮기는 것 외에 대안이 없음*(이건 장점이 될 수도 있음)*

SaaS(Software as a Service)

주요 특징	내용	세부내용
즉시 사용 개시 가능	• 설치나 환경 구성 없이 인프라~애플리케이션까지 사용 가능	• 웹이나 앱에 접속만 하면 바로 서비스 이용 가능 • 별도의 서버 구축, 프로그램 설치 과정 불필요
운영·관리 부담 최소화	• 시스템 운영과 관리는 제공자가 담당하므로 사용자는 관리 작업을 신경 쓸 필요가 없음	• 서버, 보안, 업데이트, 장애 대응 등을 서비스 제공자가 처리 • 사용자는 관리 작업을 신경 쓸 필요 없음
자동 업데이트 환경	• 항상 최신 버전의 기능과 보안 패치 등이 유지됨	• 기능 개선, 보안 패치가 자동 반영됨 • 버전 업그레이드를 직접 수행할 필요 없음
사용량 기반 이용 요금 부담	• 필요한 만큼만 사용 가능	• 사용자 수나 요금제에 따라 비용 조절 가능 • 업무 규모 변화에 유연하게 대응 가능
커스터마이징 제한	• 제공된 기능 범위 내에서만 사용할 수 있음	• 기능 추가나 구조 변경에는 제약이 있음 • 서비스 제공자가 정해준 방식에 맞춰 사용해야 함