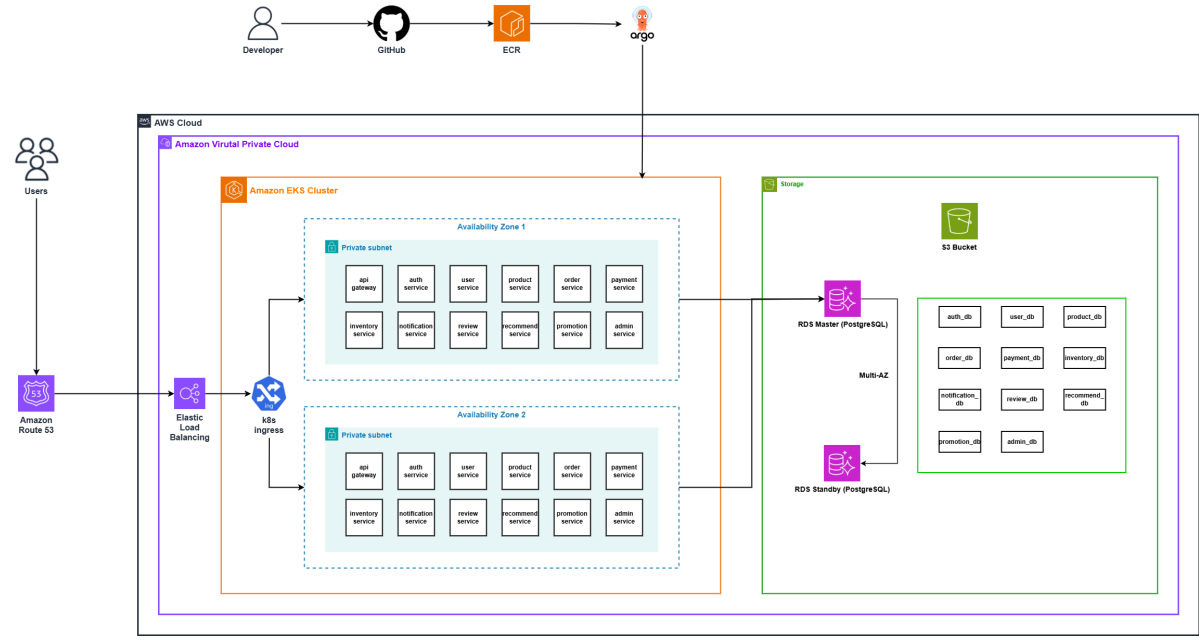
클라우드 구축 방안 비교

✅ 퍼블릭 클라우드



1. 주요 구성 요소

1) Amazon EKS (Kubernetes Cluster)

- 모든 마이크로서비스는 Kubernetes Pod로 EKS에 배포
- `Availability Zone` 2개에 Private Subnet 분산 배치
- 서비스 목록: `api-gateway` , `auth` , `user` , `order` , `product` , `inventory` , `review` , `recommend` , `promotion` , `admin` 등

2) ALB + Ingress Controller

- `Amazon ALB` 를 외부 트래픽 진입점으로 사용
- ALB → `Ingress Controller` (alb ingress or nginx ingress)
- 경로 기반 라우팅으로 각 마이크로서비스에 HTTP 요청 전달

3) Amazon RDS (PostgreSQL Multi-AZ)

- 고가용성 구성: Master-Standby 구조 (Multi-AZ)
- 하나의 인스턴스 안에서 논리적으로 서비스별 DB 분리
  - 예: `user_db` , `order_db` , `inventory_db` 등

4) Object Storage (S3)

- 정적 파일 저장, 로그 저장 등

- 기본은 Amazon S3, 필요 시 MinIO/Ceph S3 호환 구성도 가능

5) CI/CD 파이프라인

- **CI**: GitHub Actions
  - 애플리케이션 빌드, 테스트, Docker 이미지 빌드 및 ECR 푸시
- **CD**: Argo CD
  - GitOps 방식으로 Git 저장소 + ECR 변경 감지 후 자동 배포

2. 기술 스택 및 오토메이션 도구

목적	사용 도구
IaC	Terraform + Ansible + `eksctl`
컨테이너 오케스트레이션	Amazon EKS (Kubernetes)
CI/CD	GitHub Actions + Argo CD
이미지 저장소	Amazon ECR
데이터베이스	Amazon RDS (PostgreSQL)
오브젝트 스토리지	Amazon S3

3. 고가용성(HA) 구성 및 오토스케일링 전략

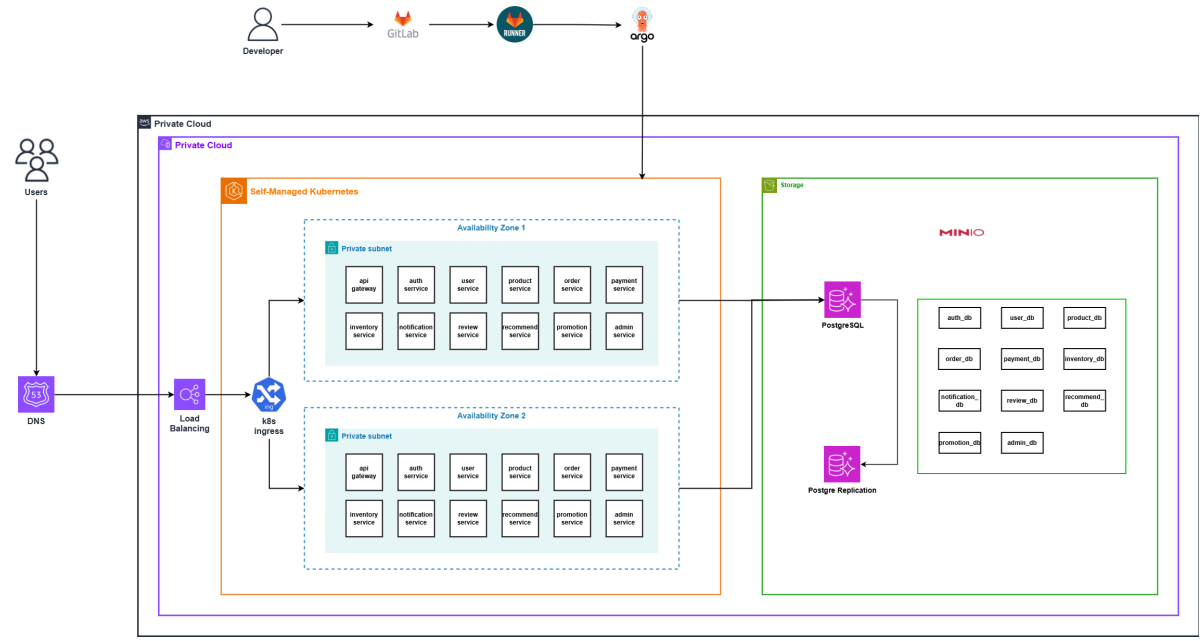
고가용성 구성

- EKS 노드 그룹 → Multi-AZ 구성
- RDS → Multi-AZ Master/Standby 구성
- ALB → Region-level HA

오토스케일링 구성

- **HPA** (Horizontal Pod Autoscaler): Pod 레벨 오토스케일링 (CPU/메모리 기반)
- **Cluster Autoscaler**: 필요 시 EC2 인스턴스 수 자동 증가
- **ALB 자체**: 고가용성 및 요청 트래픽 자동 분산

✅ Private Cloud



1. 주요 구성 요소 상세 설명

1) Self-Managed Kubernetes Cluster

- `kubeadm`, `RKE`, 또는 `Rancher` 등을 통해 직접 구축
- 노드들은 온프레미스 물리/가상 서버 기반
- AZ 개념을 서버 존 또는 랙 단위로 유사 구성 가능
- 마이크로서비스는 동일하게 K8s Pod로 배포
- 구성 서비스: `api-gateway`, `auth`, `user`, `product`, `order`, `inventory`, `notification`, `review`, `recommend`, `promotion`, `admin` 등

2) Ingress 구성

- `NGINX Ingress Controller` 또는 `HAProxy Ingress` 사용
- 외부 IP는 **MetalLB**를 통해 L2 방식으로 할당
- 내부 DNS 또는 사내 네트워크와 연동해 도메인 라우팅

3) PostgreSQL (Self-hosted)

- 고가용성 구성: `Streaming Replication`, `Patroni`, 또는 `pgpool-II`
- 1대의 Master + 1대 이상의 Standby 구성
- 논리적으로 서비스별 DB 분리 (예: `order_db`, `user_db` 등)

4) 오브젝트 스토리지

- Amazon S3 API와 호환되는 내부 오브젝트 스토리지
- MinIO: 경량, 단일 바이너리, 간편 구성
- 로그, 백업, 정적 파일, 모델 저장 등 용도

4) CI/CD 구성

- **CI**: GitLab + GitLab Runner
  - 빌드, 테스트, Docker 이미지 생성 후 Harbor에 푸시
- **CD**: Argo CD (Self-hosted, 클러스터 내부)
  - GitOps 방식으로 Git + Harbor 상태 감지 → 자동 배포

2. 기술 스택 및 오토메이션 도구

목적	사용 도구
IaC	Terraform + Ansible
K8s 설치	kubeadm, RKE, Rancher 등
CI/CD	GitLab + GitLab Runner + Argo CD
이미지 저장소	Harbor
데이터베이스	PostgreSQL (Streaming Replication or Patroni)
오브젝트 스토리지	MinIO or Ceph RGW
로드 밸런싱	MetalLB + NGINX Ingress

3. 고가용성(HA) 구성 및 오토스케일링 전략

고가용성 구성

- PostgreSQL → Replication 구조로 HA 구성
- Kubernetes → 여러 노드 분산 배치, Control Plane 이중화 가능
- MetalLB → 여러 IP pool / L2 구성
- MinIO → Distributed mode (4개 이상의 노드 구성)

오토스케일링 구성

- **HPA** (Horizontal Pod Autoscaler): CPU/메모리 기반 Pod 오토스케일링
- **Cluster Autoscaler**: 온프레미스 환경에서는 미지원, 대신
  - 노드 자동 확장은 수동 트리거 또는 클러스터 API 연동 필요 (ex. VMware, Proxmox 연동)