클라우드 구축 방안 계획

✅ 1. 프로젝트 개요

이번 프로젝트는 퍼블릭 클라우드(AWS 기반)와 프라이빗 클라우드(온프레미스 환경)를 대상으로, Kubernetes 기반의 마이크로서비스 운영 환경을 설계하고, 이를 DevOps 관점에서 평가하는 것을 목적으로 한다.

설계 대상은 총 10개 이상의 마이크로서비스로 구성되며, 각 서비스는 HTTP 요청을 처리하고 PostgreSQL 데이터베이스와 S3 API 호환 객체 스토리지를 필요로 한다. 모든 구성 요소에는 고가용성(High Availability, HA)이 적용되어야 하며, Dev / Staging / Production 환경 간의 통일성 있는 구성 또한 필수 조건이다.

또한, 각 환경은 오픈소스 기반의 CI/CD 파이프라인을 활용하여 자동화 배포가 가능해야 하며, 인프라 구성에는 Terraform, Helm, Ansible 등의 IaC 도구 사용이 허용된다. 본 문서는 두 환경을 설계하고, 다음의 4가지 기준에 따라 비교 분석한다:

- 관리 편의성
- 민첩성
- 비용 효율성
- 보안

✅ 2. 전체 아키텍처 구성

퍼블릭과 프라이빗 클라우드 모두 공통적으로 다음 요소를 포함하는 구조로 설계된다:

- 사용자는 Ingress Controller와 Load Balancer를 통해 Kubernetes 클러스터에 접근한다.
- 각 마이크로서비스는 컨테이너화되어 Kubernetes 클러스터 내에서 Pod로 실행된다.
- 데이터베이스는 PostgreSQL 기반이며, 각 서비스별로 분리된 DB를 사용한다.
- 객체 스토리지는 S3 API 호환 시스템으로 통합되어 있으며, 퍼블릭 클라우드에서는 Amazon S3, 프라이빗 클라우드에서는 MinIO를 사용한다.
- 애플리케이션 변경 시 GitOps 기반 배포 방식으로 자동 적용되도록 CI/CD 파이프라인을 구성한다.
- 모든 구성 요소는 고가용성과 확장성을 고려하여 배치된다.

> 💡 이 아키텍처는 클라우드 네이티브 환경의 핵심 요소를 기반으로 하며, 확장성, 이식성, 민첩성, 자동화를 모두 충족한다.

✅ 3. 클러스터 구성 방식 비교

퍼블릭 클라우드 (AWS)

클러스터는 Amazon EKS를 기반으로 `eksctl`을 통해 손쉽게 생성할 수 있다. AWS에서 Control Plane을 완전 관리하며, 사용자는 워커 노드와 어플리케이션에만 집중하면 된다. 워커 노드는 Auto Scaling Group을 통해 자동 확장이 가능하고, 관리형 노드 그룹(MNG)을 사용하면 유지보수도 간편하다.

프라이빗 클라우드 (온프레미스)

클러스터는 `kubeadm` 또는 `RKE`를 활용하여 직접 구축한다. 이 경우 Control Plane 구성, HAProxy + Keepalived를 통한 API 서버 이중화, etcd 클러스터 구축 등을 수동으로 설정해야 한다. 노드 확장도 수동 또는 자체 가상화 인프라 기반으로 관리되며, 유지보수에 더 많은 리소스가 소요된다.

요약

- 퍼블릭 클라우드는 클러스터 구성과 유지관리 부담이 적고, 운영에 집중할 수 있다.
- 프라이빗 클라우드는 완전한 제어권을 보장하지만, 운영과 유지관리 비용이 높다.

✅ 4. 서비스 구성 및 배포 구조

퍼블릭 클라우드 (AWS)

Ingress는 AWS ALB Ingress Controller를 사용하여 외부 요청을 수신하며, 자동으로 Application Load Balancer(ALB)가 생성되고 SSL 인증서, Path 기반 라우팅 등 다양한 기능을 기본 제공한다.

애플리케이션은 GitHub Actions에서 Docker 이미지 빌드 후 Amazon ECR에 푸시되고, Argo CD를 통해 GitOps 방식으로 EKS에 배포된다.

개발, 스테이징, 운영 환경은 별도 EKS 클러스터 또는 네임스페이스로 분리하여 관리한다.

프라이빗 클라우드 (온프레미스)

Ingress는 오픈소스 기반의 NGINX Ingress Controller를 사용하며, 외부 접근을 위한 Load Balancer는 MetalLB 또는 HAProxy로 구성한다.

CI/CD는 자체 구축한 Jenkins 또는 Argo CD를 활용하여 애플리케이션을 Kubernetes에 배포한다.

환경 분리는 Kubernetes 네임스페이스 또는 클러스터로 구분하며, 인증 및 스토리지 연동은 수동으로 설정해야 한다.

비교 요약

- 퍼블릭 클라우드는 자동화, 확장성, 안정성 측면에서 강점이 있다.
- 프라이빗 클라우드는 오픈소스를 기반으로 한 유연한 구성이 가능하지만 수동 작업이 많다.

✅ 5. 스토리지 및 DB 구성

퍼블릭 클라우드 (AWS)

데이터베이스는 Amazon RDS for PostgreSQL을 사용하며, 고가용성을 위한 Multi-AZ 구성과 자동 백업, 장애 복구 기능을 제공한다.

객체 스토리지는 Amazon S3를 사용하며, 높은 내구성과 확장성, S3 API 완전 호환을 제공한다.

이로 인해 운영자가 직접 복제, 백업, 확장 전략을 고민할 필요 없이 안정적인 데이터 운영이 가능하다.

프라이빗 클라우드 (온프레미스)

PostgreSQL은 StatefulSet 기반으로 구성하고, Streaming Replication 또는 Patroni 클러스터링을 활용하여 고가용성을 구현한다.

객체 스토리지는 MinIO 또는 Ceph RGW를 사용하여 S3 API 호환 인터페이스를 구성하고, 내부 저장소와 연동한다.

스토리지 및 DB 모두 이중화 구성과 백업 스크립트를 직접 작성해야 하며, 유지보수에 숙련된 인력이 필요하다.

비교 요약

- 퍼블릭 클라우드는 안정성과 자동화 측면에서 매우 유리하다.
- 프라이빗 클라우드는 커스터마이징 가능하지만, 관리 부담이 크다.

✅ 6. 오토스케일링 구성 비교

퍼블릭 클라우드 (AWS)

Pod 단위 확장은 Kubernetes Horizontal Pod Autoscaler(HPA)를 통해 CPU/메모리 사용률 기반으로 자동 조정된다.

노드 단위 확장은 Cluster Autoscaler가 Auto Scaling Group과 연동되어 자동으로 신규 노드를 추가하거나 제거한다.

또한 VPA(Vertical Pod Autoscaler)나 Karpenter 같은 최신 오토스케일링 도구를 쉽게 연동할 수 있다.

프라이빗 클라우드 (온프레미스)

Pod 단위의 HPA는 동일하게 적용 가능하지만, 노드 확장은 기본적으로 수동이며 자동화를 위해서는 Cluster Autoscaler를 커스터마이징하거나 가상 머신 자동 배포 환경이 추가되어야 한다.

리소스 기반의 자동화 구성에는 추가적인 오픈소스 도구와 자체 스크립트 관리가 필요하다.

비교 요약

- 퍼블릭 클라우드는 완성도 높은 오토스케일링 체계를 쉽게 도입 가능하다.
- 프라이빗 클라우드는 오토스케일링 구성은 가능하나, 자동화 구현에는 추가 작업이 필요하다.

✅ 7. 보안 및 접근 제어 구성

퍼블릭 클라우드 (AWS)

AWS는 IAM(Identity and Access Management)을 통해 사용자 및 서비스 단위로 세분화된 권한 설정이 가능하다.

네트워크 보안은 VPC, 서브넷, 보안 그룹(Security Group), NACL 등으로 계층적으로 구성되며, 기본적으로 클러스터가 퍼블릭 외부와 격리되도록 설정된다.

시크릿 관리는 AWS Secrets Manager 또는 KMS(Key Management Service)를 통해 암호화된 상태로 저장 및 주기적 회전이 가능하다.

또한, CloudTrail, GuardDuty 등을 통해 감사 로그, 이상 탐지 기능을 쉽게 사용할 수 있다.

프라이빗 클라우드 (온프레미스)

접근 제어는 Kubernetes RBAC와 자체 인증 시스템(e.g., Keycloak, OIDC)을 조합하여 구성한다.

시크릿은 기본적으로 Kubernetes Secret 또는 Sealed Secrets, HashiCorp Vault 등을 활용하여 저장되며, 자체 암호화 키를 운용해야 한다.

네트워크 통제는 NetworkPolicy 또는 방화벽 장비를 통한 IP/포트 기반 접근 제어 방식으로 구성된다.

감사 로그는 Fluentd + Elasticsearch + Kibana(ELK), Loki + Grafana 조합으로 수집 및 시각화할 수 있다.

비교 요약

- 퍼블릭 클라우드는 기본 보안 기능이 풍부하고 구성 간소화가 가능하다.
- 프라이빗 클라우드는 강력한 보안 통제가 가능하지만, 모든 구성과 유지보수가 운영자의 책임이다.

✅ 8. HA 구성 전략 비교

퍼블릭 클라우드 (AWS)

EKS는 Control Plane HA를 기본적으로 제공하며, AWS 측에서 장애 복구를 자동으로 수행한다.

RDS는 Multi-AZ 구성을 통해 DB 장애 시 자동으로 대기 노드로 전환된다.

S3는 99.999999999%(elven nine)의 내구성과 높은 가용성을 제공하며, 데이터 복제를 별도로 구성하지 않아도 된다.

Ingress 및 Load Balancer(ALB)는 다중 가용 영역에 분산되어 기본적으로 HA를 구성한다.

프라이빗 클라우드 (온프레미스)

Kubernetes Control Plane의 HA 구성은 Master 노드 이중화, etcd 클러스터, HAProxy + Keepalived 등을 수동으로 설정해야 한다.

PostgreSQL은 Streaming Replication, Patroni, Pgpool-II 등을 활용하여 장애 복구 체계를 구성한다.

MinIO는 distributed mode로 구성하거나 Ceph RGW를 활용하여 복제 및 Erasure Coding을 적용할 수 있다.

Ingress는 NGINX Controller를 이중화하고, HAProxy 또는 MetalLB로 외부 트래픽 이중화 처리한다.

비교 요약

- 퍼블릭 클라우드는 HA 구성이 대부분 기본 제공되며, 자동화되어 있다.
- 프라이빗 클라우드는 HA를 직접 구성해야 하며, 구성 복잡성과 유지 부담이 높다.

✅ 9. 종합 비교

평가 항목	퍼블릭 클라우드 (AWS)	프라이빗 클라우드 (온프레미스)
관리 편의성	🟢 관리형 서비스(EKS, RDS, S3) 제공으로 운영 부담 적음	🔴 모든 요소 직접 구성 및 유지 필요
민첩성	🟢 빠른 배포와 자동화 도구 연동 쉬움	🔴 구성 복잡하고 자동화에도 시간 소요
비용 효율성	🟡 단기 비용 유리하지만 장기 누적 비용 존재	🔴 초기 투자 크지만 장기 TCO에서 이점 가능
보안	🟢 내장 보안 도구 활용 가능, 규제 준수 쉬움	🟡 커스터마이징 자유도 높지만 운영자 책임 큼
고가용성 구성	🟢 구성 간소화 및 자동 복구 체계 지원	🔴 구성 복잡하지만 세밀한 제어 가능

✅ 10. 결론

본 프로젝트를 통해 퍼블릭 클라우드(AWS)와 프라이빗 클라우드(온프레미스 환경) 모두에서 Kubernetes 기반 마이크로서비스 환경을 설계하고, 주요 DevOps 구성 요소를 비교 분석하였다.

퍼블릭 클라우드는 자동화, 빠른 배포, 관리 효율성 측면에서 매우 뛰어나며, 특히 초기 인프라 셋업과 반복 배포가 중요한 개발/운영 환경에 적합하다. 스타트업, 빠른 MVP 론칭, 글로벌 서비스 확장에 최적화되어 있다.

프라이빗 클라우드는 보안과 통제력이 중요한 환경에서 강점을 보인다. 온프레미스 자원을 보유한 기관이나, 민감한 데이터를 직접 보호하고자 하는 조직에 적합하며, 장기적으로 비용 최적화 및 보안 정책 통제에 유리할 수 있다.

최종 선택은 조직의 기술 역량, 예산, 보안 요구사항, 서비스 운영 범위에 따라 전략적으로 결정되어야 한다.