CNF

2026.03

멀티 클라우드 시대의 GSLB 전략 :
Active-Active vs Active-Passive DR

"멀티 클라우드로 전환했는데, 사이트 장애 시 DR 전환이 제대로 안 된다"
상용 GSLB 어플라이언스에 수억 원을 투자하고도 DNS TTL 캐싱 문제, Cold Standby 지연, 벤더 종속으로 인해 목표 RTO를 달성하지 못하는 조직이 여전히 많습니다.
이 백서에서 DR 구축 비용 구조 분석과, 금융·이커머스·통신 업종별 구축 사례, 그리고 PoC부터 프로덕션 Active-Active DR까지 단계별 구축 로드맵을 확인하실 수 있습니다.

hello@cncf.co.kr

02-469-5426

www.cncf.co.kr

Contents

1장. GSLB 개요

오늘날 기업의 IT 서비스는 전 세계 사용자를 대상으로 운영되며, 서비스 중단은 곧바로 매출 손실과 브랜드 신뢰도 하락으로 이어진다. 이러한 환경에서 지리적으로 분산된 인프라 간에 트래픽을 지능적으로 분배하는 기술인 GSLB(Global Server Load Balancing)는 엔터프라이즈 아키텍처의 핵심 구성 요소로 자리매김하고 있다. 본 장에서는 GSLB의 기본 개념과 필요성을 정의하고, DNS 기반 트래픽 라우팅의 동작 원리, 주요 라우팅 알고리즘, 그리고 헬스체크와 자동 페일오버 메커니즘을 상세히 살펴본다.

1.1 GSLB의 정의와 필요성

1.1.1 GSLB란 무엇인가

GSLB(Global Server Load Balancing)는 **지리적으로 분산된 복수의 데이터센터 간에 네트워크 트래픽을 지능적으로 분배하는 기술**이다. 일반적인 로드밸런서(SLB, Server Load Balancer)가 단일 데이터센터 내부에서 서버 간 트래픽을 분배하는 것과 달리, GSLB는 데이터센터 단위, 즉 사이트(Site) 간의 글로벌 트래픽 분배를 담당한다.

GSLB의 핵심 동작 원리는 **DNS(Domain Name System) 기반 라우팅**이다. 클라이언트가 특정 도메인에 접속하기 위해 DNS 쿼리를 발생시키면, GSLB는 각 데이터센터의 가용성, 부하 상태, 클라이언트의 지리적 위치, 관리자가 설정한 라우팅 정책 등을 종합적으로 평가하여 최적의 데이터센터 IP 주소를 DNS 응답으로 반환한다. 이후 클라이언트는 반환된 IP 주소로 직접 연결하며, GSLB 자체는 실제 데이터 전송 경로(Data Path)에 관여하지 않는다.

이 점이 GSLB와 일반 로드밸런서(SLB)의 가장 근본적인 차이점이다. SLB는 트래픽이 자신을 경유하며 실시간으로 분배 결정을 내리는 반면, GSLB는 DNS 레벨에서만 개입하여 클라이언트가 최초에 연결할 서버의 IP 주소를 결정한다.

구분	SLB (Server Load Balancer)	GSLB (Global Server Load Balancer)
분배 범위	사이트 내부 (로컬)	사이트 간 (글로벌)
동작 방식	트래픽이 LB를 경유 (인라인)	DNS 응답으로 IP 반환 (아웃오브밴드)

개입 시점	모든 요청에 대해 실시간 분배	DNS 쿼리 시점에만 개입
데이터 경로	Data Path 상에 위치	Data Path에서 벗어남
동작 계층	L4(TCP/UDP) / L7(HTTP)	DNS 계층
대표 기능	서버 간 부하 분산, 세션 유지, SSL 종료	사이트 선택, 재해복구 페일오버, 지리 기반 라우팅

이처럼 GSLB는 DNS 응답을 조작하여 클라이언트가 연결할 서버의 IP 주소를 결정하는 방식으로 동작하며, 이는 기존 DNS 인프라를 활용하므로 별도의 네트워크 장비 변경 없이 구축할 수 있다는 장점을 제공한다.

1.1.2 왜 GSLB가 필요한가

현대 엔터프라이즈 환경에서 GSLB가 필수적인 기술로 부상한 배경에는 다음과 같은 비즈니스 및 기술적 요인이 있다.

첫째, 글로벌 서비스 확대와 사용자 경험 요구이다. 기업의 서비스가 국내를 넘어 글로벌 시장으로 확대됨에 따라, 전 세계 사용자에게 낮은 지연 시간(Latency)으로 서비스를 제공해야 하는 요구가 증가하고 있다. 사용자가 물리적으로 가까운 데이터센터에 접속하도록 유도하는 GSLB의 지리 기반 라우팅(Geolocation Routing)은 이러한 요구를 충족하는 핵심 기술이다.

둘째, 무중단 운영(High Availability) 요구가 강화되고 있다. 서비스 중단에 대한 고객과 시장의 기대치는 매우 높아졌다. 단일 데이터센터에 장애가 발생하더라도 서비스가 중단되지 않도록 자동 페일오버(Automatic Failover)를 제공하는 GSLB는 99.99% 이상의 SLA(Service Level Agreement)를 달성하기 위한 필수 인프라이다.

셋째, 재해복구(DR, Disaster Recovery) 규제가 의무화되고 있다. 특히 금융권에서는 전자금융감독규정에 따라 RTO(Recovery Time Objective) 2시간 이내의 재해복구 체계를 의무적으로 갖추어야 한다. 이커머스 기업 역시 99.99% SLA를 보장하기 위해 멀티사이트 DR 체계를 구축하고 있으며, GSLB는 DR 구성에서 트래픽을 정상 사이트로 자동 전환하는 핵심 역할을 수행한다.

넷째, 멀티 클라우드 및 하이브리드 클라우드 환경이 확산되고 있다. 기업들이 AWS, Azure, GCP 등 복수의 클라우드 프로바이더를 동시에 활용하거나, 온프레미스와 퍼블릭 클라우드를 병행하는 하이브리드 아키텍처를 채택함에 따라, 이기종 인프라 간에 트래픽을 통합적으로 관리할 수 있는 GSLB의 중요성이 더욱 부각되고 있다.

다섯째, Kubernetes 멀티 클러스터 운영이 확산되고 있다. 컨테이너 오케스트레이션의 사실상 표준인 Kubernetes를 복수의 리전과 클라우드에 걸쳐 운영하는 기업이 증가하면서, Kubernetes 네이티브 방식으로 클러스터 간 트래픽을 분배하는 GSLB 솔루션에 대한 수요가 급격히 늘어나고 있다. K8GB와 같은 CNCF 프로젝트가 주목받는 이유가 바로 여기에 있다.

1.2 GSLB 핵심 동작 원리

1.2.1 DNS 기반 트래픽 라우팅 메커니즘

GSLB의 동작은 DNS 쿼리 처리 과정에 기반한다. 클라이언트가 특정 도메인에 접속하려 할 때, 다음과 같은 단계를 거쳐 최적의 데이터센터로 라우팅된다.

[GSLB DNS 기반 트래픽 라우팅 흐름]

1. 클라이언트가 example.com에 대한 DNS 쿼리를 발생시킨다

2. 재귀 DNS 리졸버가 GSLB가 설정된 권한 DNS 서버에 쿼리를 전달한다

3. GSLB 장비가 다음 요소들을 평가한다:
   - 각 데이터센터의 서버 헬스 상태 및 가용성
   - 현재 서버 부하 (연결 수, 패킷 레이트, 대역폭)
   - 클라이언트의 지리적 위치
   - 관리자가 설정한 라우팅 정책

4. 평가 결과에 따라 최적의 데이터센터 IP 주소를 DNS 응답으로 반환한다

5. 클라이언트는 반환된 IP로 직접 연결한다 (GSLB는 데이터 경로에서 벗어남)

이 과정에서 핵심적인 DNS 구성 요소들의 역할을 이해하는 것이 중요하다.

DNS 위임(NS 레코드): 기업의 권한 DNS 서버는 GSLB가 관리하는 서브도메인(예: `app.gslb.example.com`)에 대한 DNS 쿼리를 GSLB 장비의 ADNS(Authoritative DNS) 서비스로 위임한다. 이 위임은 NS(Name Server) 레코드를 통해 이루어지며, 상위 DNS 서버가 해당 서브도메인에 대한 쿼리를 받으면 GSLB의 ADNS로 쿼리를 전달한다.

권한 DNS(ADNS, Authoritative DNS): GSLB 장비 내에서 운영되는 ADNS 서비스는 위임받은 도메인에 대해 권한 있는 DNS 응답을 생성한다. 일반 DNS와 달리 GSLB의 ADNS는 정적인 레코드를 반환하는 것이 아니라, 실시간 헬스체크 결과와 라우팅 알고리즘을 적용하여 동적으로 최적의 IP를 선택하고 응답한다.

TTL(Time To Live) 캐싱의 영향: DNS 응답에 포함된 TTL 값은 해당 응답이 캐시에 유지되는 시간을 지정한다. GSLB 환경에서 TTL은 페일오버 속도에 직접적인 영향을 미친다. TTL이 짧으면(예: 30~60초) 장애 발생 시 빠르게 새로운 IP로 전환할 수 있지만, DNS 쿼리 부하가 증가한다. 반대로 TTL이 길면(예: 300초 이상) DNS 쿼리 부하는 감소하지만, 장애 발생 시 캐시된 구 IP로 계속 접속하는 클라이언트가 존재하여 페일오버가 지연된다.

670

[그림 1] 670

TTL 설정	페일오버 수렴 시간	DNS 쿼리 부하	사용 시나리오
5~30초	매우 빠름	매우 높음	미션 크리티컬 서비스, 즉각적 페일오버 필요 시
60초	빠름	높음	GSLB 일반 권장값, 페일오버 속도와 부하의 균형
300초 (5분)	보통	보통	CDN, 로드밸런싱 일반 용도
3,600초 (1시간)	느림	낮음	변경이 드문 정적 서비스
86,400초 (24시간)	매우 느림	매우 낮음	거의 변경되지 않는 레코드

주의할 점은 TTL 값을 아무리 낮게 설정하더라도 웹 브라우저의 자체 DNS 캐시(일반적으로 15분~6시간)로 인해 즉각적인 페일오버가 보장되지 않는다는 것이다. 이는 GSLB 기반 DR 설계 시 반드시 고려해야 하는 제약 사항이다.

1.2.2 주요 라우팅 알고리즘

GSLB는 다양한 라우팅 알고리즘을 제공하며, 관리자는 서비스의 특성과 요구사항에 따라 적절한 알고리즘을 선택할 수 있다. 다음은 주요 GSLB 라우팅 알고리즘의 비교표이다.

알고리즘	설명	적합 시나리오
Round Robin	각 사이트에 순차적으로 트래픽 배분	사이트 간 용량이 동일할 때
Weighted Round Robin	가중치 기반 비율 배분	사이트별 용량이 다를 때
Least Connection	현재 연결 수가 가장 적은 사이트로 배분	세션 길이가 불균일할 때
Geolocation/Proximity	클라이언트와 지리적으로 가장 가까운 사이트로 배분	글로벌 서비스, 지연 최소화
RTT (Round Trip Time)	네트워크 응답 시간이 가장 짧은 사이트로 배분	네트워크 품질 기반 최적화
Source IP Hash	Source IP 해시 기반 고정 배분	세션 지속성 필요 시

Round Robin은 가장 단순한 알고리즘으로, DNS 응답 시 각 사이트의 IP를 순차적으로 반환한다. 모든 사이트의 처리 용량이 동일하고 특별한 라우팅 요구사항이 없을 때 적합하다. 구현이 간단하고 트래픽이 균등하게 분산되는 장점이 있으나, 사이트 간 용량 차이나 클라이언트 위치를 고려하지 않는 한계가 있다.

Weighted Round Robin은 Round Robin에 가중치 개념을 추가한 것으로, 각 사이트에 처리 용량에 비례하는 가중치를 부여하여 트래픽을 비율 배분한다. 예를 들어 사이트 A의 가중치가 70이고 사이트 B의 가중치가 30이면, 전체 트래픽의 약 70%가 사이트 A로, 30%가 사이트 B로 배분된다. 카나리 배포(Canary Deployment)나 점진적 마이그레이션 시나리오에서도 유용하게 활용된다.

Least Connection은 MEP(Metrics Exchange Protocol) 등을 통해 수집한 각 사이트의 현재 활성 연결 수를 기반으로, 연결 수가 가장 적은 사이트로 트래픽을 배분한다. 세션의 길이가 불균일한 서비스(예: 파일 다운로드, 스트리밍)에서 효과적이다.

Geolocation/Proximity는 GeoIP 데이터베이스를 참조하여 클라이언트의 지리적 위치를 파악하고, 가장 가까운 사이트로 트래픽을 배분한다. 글로벌 서비스에서 사용자 경험(지연 시간)을 최적화하거나, 데이터 주권 규정(GDPR 등)에 따라 특정 지역의 트래픽을 해당 국가 내 데이터센터로 제한할 때 사용된다.

RTT(Round Trip Time) 기반 라우팅은 Geolocation보다 한 단계 발전된 방식으로, 실제 네트워크 왕복 시간을 측정하여 응답 시간이 가장 짧은 사이트로 트래픽을 배분한다. 네트워크 혼잡도를 반영하므로 더 정확한 성능 최적화가 가능하지만, 지속적인 RTT 모니터링에 따른 오버헤드가 존재한다.

Source IP Hash는 클라이언트의 소스 IP 주소에 해시 함수를 적용하여 항상 동일한 사이트로 배분하는 방식이다. 세션 퍼시스턴스(Session Persistence)가 필요한 서비스에서 활용되며, 동일 사용자가 항상 같은 사이트에 접속하도록 보장한다. 다만 NAT(Network Address Translation) 환경에서는 다수의 클라이언트가 동일한 공인 IP를 사용하므로 부하 편중이 발생할 수 있다.

1.2.3 헬스체크와 자동 페일오버

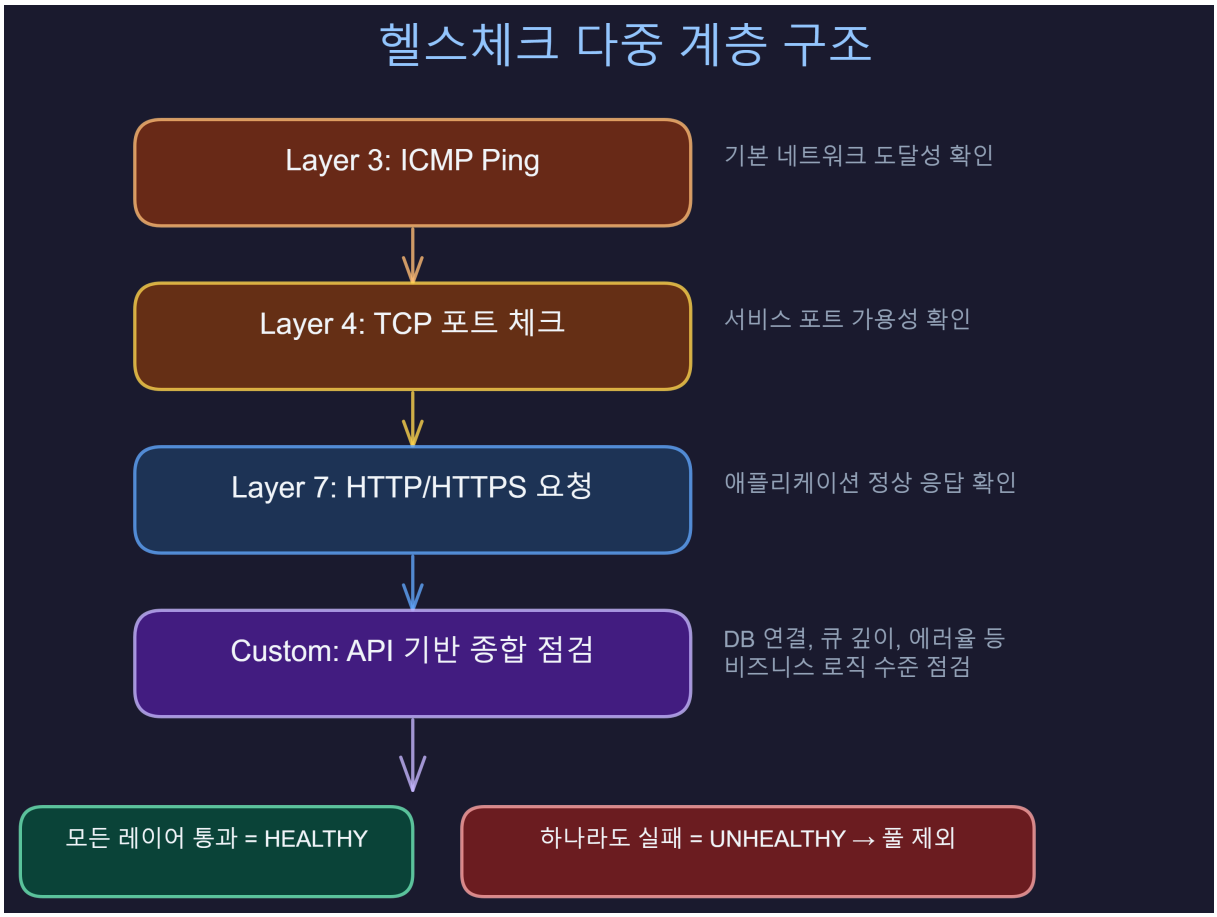
헬스체크(Health Check)는 GSLB가 관리하는 각 사이트 및 서비스의 가용 상태를 주기적으로 점검하는 메커니즘이다. 헬스체크 없이는 장애가 발생한 사이트로 트래픽을 계속 전송하게 되므로, 헬스체크는 GSLB의 지능적 라우팅을 가능하게 하는 핵심 구성 요소이다.

헬스체크 유형

GSLB에서 활용하는 헬스체크는 점검 수준에 따라 다음과 같이 분류된다.

유형	동작 방식	점검 수준	장점	단점
ICMP (Ping)	대상 서버에 ICMP Echo Request 전송, 응답 확인	네트워크 도달성	구현 간단, 오버헤드 최소	애플리케이션 장애 감지 불가
TCP	특정 포트에 TCP 3-Way Handshake 시도	포트 가용성	서비스 포트 장애 감지 가능	애플리케이션 로직 장애 감지 불가
HTTP/HTTPS	HTTP GET 요청 후 상태 코드 및 응답 본문 확인	애플리케이션 응답	웹 서비스 정상 동작 확인 가능	구현 복잡, 오버헤드 증가
Application-Level	API 호출, DB 연결 확인, 종합 점검 스크립트 수행	비즈니스 로직 수준	실질적 서비스 가용성 판단	커스텀 개발 필요, 복잡성 높음

실무에서는 이러한 헬스체크를 계층적으로 적용하는 것이 권장된다.



[그림 2] 672

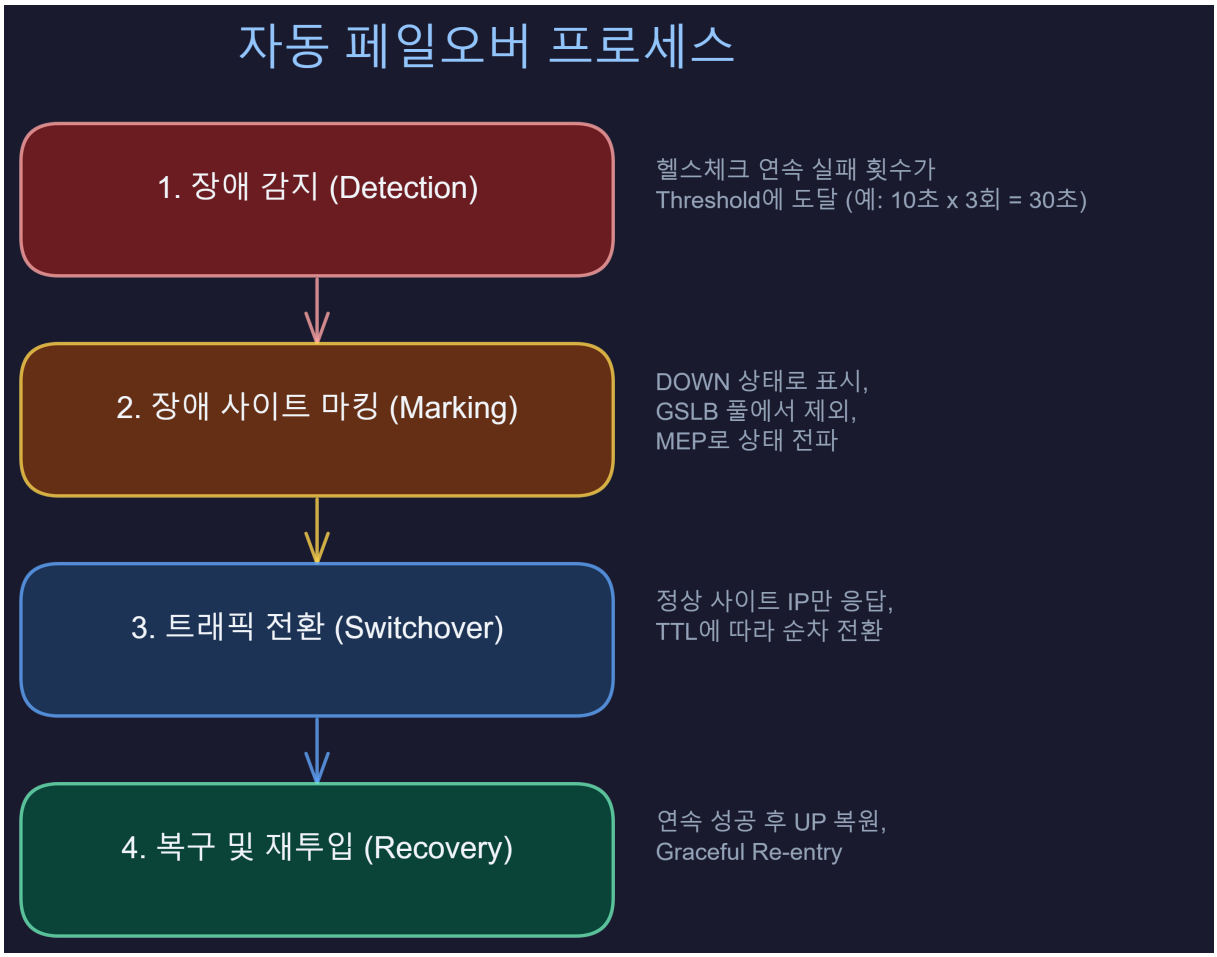
핵심 파라미터

헬스체크의 민감도와 안정성은 다음 파라미터들의 조합으로 결정된다.

파라미터	설명	일반적 설정값	설정 고려사항
Interval	헬스체크 수행 주기	10~30초	짧으면 빠른 감지, 길면 부하 감소
Timeout	단일 헬스체크의 응답 대기 시간	5~10초	네트워크 지연을 감안하여 설정
Threshold (Failure)	장애 판정을 위한 연속 실패 횟수	3회	높으면 오탐 감소, 낮으면 빠른 감지
Recovery Threshold	복구 판정을 위한 연속 성공 횟수	3~5회	플래핑(Flapping) 방지를 위해 Failure Threshold보다 높게 설정

자동 페일오버 프로세스

헬스체크 결과에 기반한 자동 페일오버는 다음 4단계로 이루어진다.



[그림 3] 641

이 프로세스에서 주의할 점은, 장애 감지부터 실제 모든 클라이언트의 트래픽 전환이 완료되기까지의 총 소요 시간이 단순히 헬스체크 파라미터만으로 결정되지 않는다는 것이다. DNS TTL 캐싱, 브라우저 자체 캐싱, 중간 재귀 리졸버의 캐싱 등 여러 계층의 캐시가 영향을 미치므로, GSLB 기반 DR 설계 시에는 이러한 캐싱 계층별 영향을 종합적으로 고려해야 한다.

[K8GB 맥락]

K8GB는 Kubernetes 네이티브 GSLB 솔루션으로서, 상용 GSLB 장비와는 차별화된 방식으로 헬스체크를 수행한다. 전통적인 GSLB가 별도의 헬스체크 모니터를 통해 서비스 상태를 점검하는 반면, K8GB는 **Kubernetes의 Liveness/Readiness Probe를 GSLB 헬스체크에 직접 활용**한다. K8GB Controller는 각 클러스터에서 실행 중인 Pod의 Readiness Probe 상태를 지속적으로 모니터링하며, Pod가 Ready 상태가 아닌 경우 해당 클러스터를 GSLB 라우팅 풀에서 자동으로 제외한다. 이 접근 방식은 별도의 헬스체크 인프라 구축 없이 Kubernetes가 이미 제공하는 헬스체크

메커니즘을 재활용함으로써 운영 복잡성을 크게 줄이는 장점이 있다. 또한 애플리케이션 개발자가 정의한 세밀한 Readiness 조건(DB 연결 상태, 캐시 워밍 완료 여부 등)이 그대로 GSLB 라우팅 결정에 반영되므로, 애플리케이션 수준의 정밀한 트래픽 관리가 가능하다.

1장에서는 GSLB의 기본 개념, DNS 기반 라우팅 원리, 라우팅 알고리즘, 헬스체크 메커니즘을 살펴보았다. GSLB가 DNS 레벨에서 동작하며 데이터 경로에 관여하지 않는다는 특성은 GSLB의 확장성과 유연성의 근간이 되지만, 동시에 DNS 캐싱으로 인한 페일오버 지연이라는 고유한 제약도 수반한다. 2장에서는 이러한 GSLB가 실제 운영 환경에서 어떤 기술들과 어떻게 결합하여 동작하는지, 그 기술 생태계를 상세히 살펴본다.

2장. GSLB 관련 기술 생태계

GSLB는 단독으로 동작하는 기술이 아니다. DNS, CDN, BGP Anycast, GeoIP, API Gateway, Reverse Proxy 등 다양한 기술과 유기적으로 결합하여 글로벌 트래픽 관리 아키텍처를 구성한다. 본 장에서는 GSLB의 기반 기술과 연동 기술, 그리고 관련 표준 및 프로토콜을 체계적으로 정리하여 GSLB를 둘러싼 기술 생태계의 전체 그림을 제시한다.

2.1 GSLB의 기반 기술

2.1.1 DNS — GSLB의 핵심 기반

DNS(Domain Name System)는 도메인 이름을 IP 주소로 변환하는 인터넷의 근간 시스템이며, GSLB는 이 DNS 해석(Resolution) 과정에 개입하여 트래픽을 분배한다. DNS가 GSLB의 핵심 기반인 이유는 다음 세 가지로 정리할 수 있다.

DNS 레벨 라우팅: GSLB는 DNS 응답을 조작하여 클라이언트가 연결할 서버의 IP 주소를 결정한다. 실제 애플리케이션 트래픽이 전송되기 전, DNS 단계에서 라우팅이 결정되므로 사용자는

처음부터 최적의 서버에 연결된다. 이러한 DNS 레벨 라우팅은 기존 DNS 인프라를 활용하므로 별도의 네트워크 장비 변경 없이 구축할 수 있다는 비용 효율성을 제공한다.

TTL(Time To Live) 관리: GSLB는 DNS 레코드의 TTL 값을 전략적으로 관리하여 장애 발생 시 빠른 페일오버를 가능하게 한다. 1장에서 살펴본 바와 같이, TTL이 짧을수록 변경 사항이 빠르게 전파되지만 DNS 쿼리 부하가 증가하는 트레이드오프가 존재한다. GSLB 운영자는 서비스의 가용성 요구수준에 따라 적절한 TTL 전략을 수립해야 한다.

GSLB의 DNS 프록시 역할: GSLB는 DNS 프록시로 동작하며, 일반 DNS와 달리 실시간 로드밸런싱 알고리즘을 적용하여 응답을 생성한다. 정적인 DNS 레코드가 아닌, 현재의 사이트 상태와 라우팅 정책에 기반한 동적인 DNS 응답을 반환하는 것이 GSLB DNS의 핵심 특성이다.

DNS 기반 GSLB의 한계점

DNS가 GSLB의 핵심 기반이지만, 다음과 같은 고유한 한계점도 존재한다.

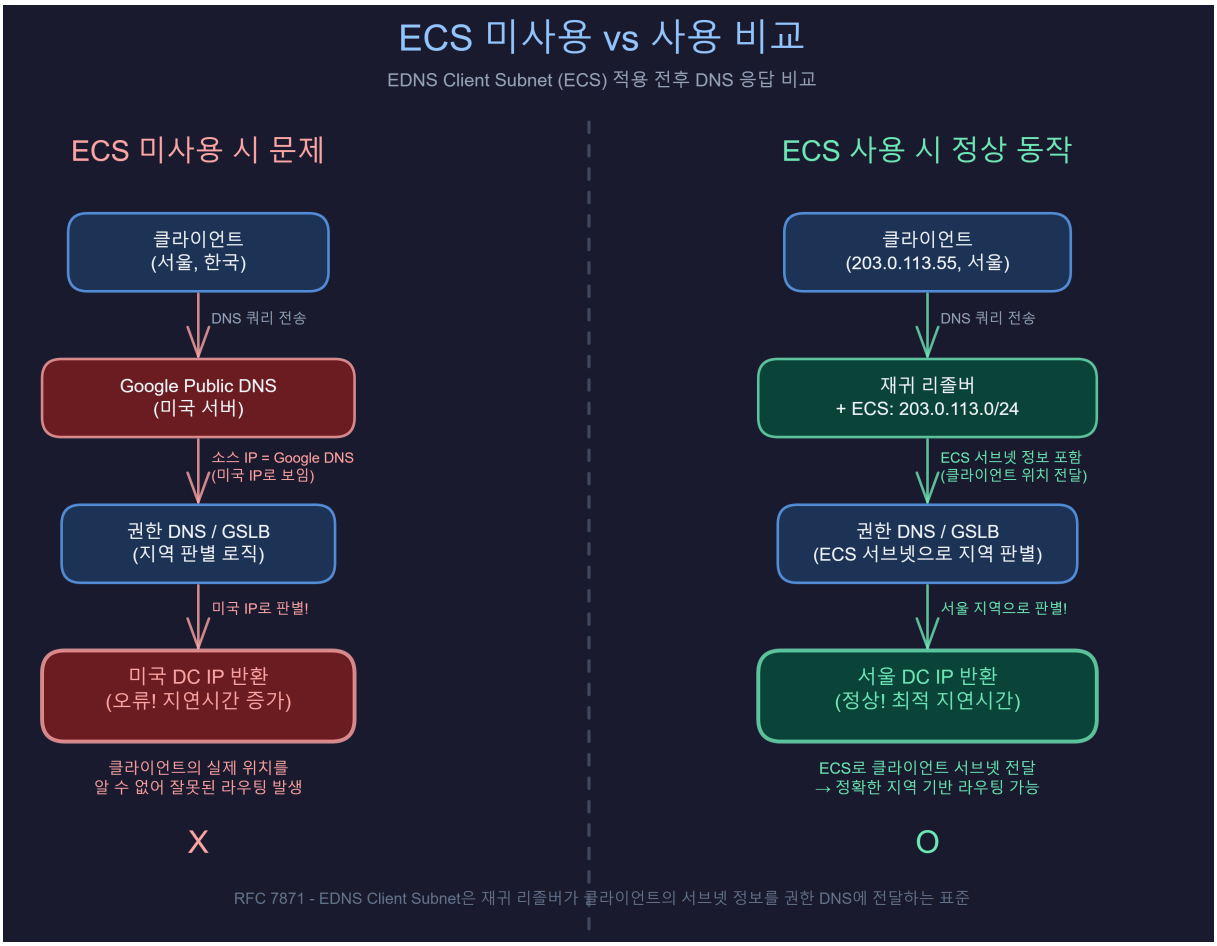
한계점	설명	영향
DNS 캐싱 전파 지연	DNS 캐싱으로 인해 페일오버 수렴 시간이 TTL에 종속됨	즉각적 페일오버 불가, 실질적 최소 20~60초
LDNS IP 기반 판단	클라이언트의 실제 IP가 아닌 LDNS(Local DNS) IP 기반으로 위치 판단	퍼블릭 DNS(8.8.8.8 등) 사용 시 위치 오판
EDNS 미지원 시 정확도 저하	EDNS Client Subnet 미지원 환경에서는 클라이언트 위치 파악 불가	지리 기반 라우팅 정확도 저하
Stateless 한계	DNS 쿼리 시점에만 개입, 연결 이후 세션 상태 관리 불가	세션 유지를 위한 추가 메커니즘 필요
브라우저 캐싱	대부분의 브라우저가 TTL 값을 무시하고 자체 캐싱 (15분~6시간)	TTL 설정과 무관한 페일오버 지연 발생

이러한 한계점을 보완하기 위해 EDNS Client Subnet(ECS), Anycast 병행 사용, 짧은 TTL 전략 등 다양한 기술적 보완 방안이 활용되며, 이에 대해서는 본 장의 후반부에서 상세히 다룬다.

EDNS Client Subnet (RFC 7871) 동작 방식

기존 DNS 기반 GSLB의 핵심 한계 중 하나인 “클라이언트 위치 부정확성” 문제를 해결하기 위해 RFC 7871에서 정의된 것이 EDNS Client Subnet(ECS)이다. ECS는 DNS 확장 메커니즘(EDNS0)의 옵션으로, 재귀 리졸버가 클라이언트의 서브넷 정보를 DNS 쿼리에 포함시켜 권한 DNS 서버에 전달하는 방법을 규정한다.

ECS가 없는 환경에서는 다음과 같은 문제가 발생한다.



서울에 있는 클라이언트가 Google Public DNS(8.8.8.8)를 사용하면, GSLB는 쿼리의 소스 IP인 Google DNS 서버(미국)의 위치를 기준으로 판단하여 미국 데이터센터의 IP를 반환할 수 있다.

ECS는 이 문제를 다음과 같이 해결한다.

재귀 리졸버가 클라이언트 IP의 일부를 마스킹(프라이버시 보호)하여 ECS 옵션에 포함시키면, 권한 DNS 서버(GSLB)는 리졸버 IP가 아닌 실제 클라이언트의 서브넷을 기준으로 최적 서버를 판단할 수 있다.

프라이버시 고려사항: ECS는 클라이언트의 전체 IP 주소가 아닌 서브넷 정보만 전달한다(일반적으로 IPv4는 /24, IPv6는 /56). 이를 통해 개별 사용자 식별 없이 대략적인 위치 정보만 전달하여 프라이버시를 보호하지만, 일부 프라이버시 옹호론자들은 서브넷 정보도 사용자 추적에 활용될 수 있다는 우려를 제기하고 있다.

2.1.2 BGP Anycast와 DNS 기반 GSLB 비교

BGP Anycast는 네트워크 계층(L3)에서 동작하는 라우팅 기술로, 지리적으로 분산된 여러 서버가 **동일한 IP 주소를 공유**하고 BGP 라우팅 프로토콜을 통해 해당 IP를 각자 광고(advertise)한다. 사용자가 Anycast IP로 요청을 보내면, BGP 라우팅 테이블에 따라 네트워크 토폴로지 기준으로 가장 "가까운" 서버로 요청이 전달된다.

GSLB(DNS 기반)와 BGP Anycast는 모두 글로벌 트래픽 분산을 목적으로 하지만, 동작 방식과 특성이 크게 다르다.

비교 항목	BGP Anycast	DNS 기반 GSLB
동작 계층	L3 네트워크 계층 (BGP 라우팅)	L7 DNS 계층
라우팅 기준	네트워크 토폴로지 (AS-path, IGP metric)	관리자 정책 (지리, 부하, RTT, 가용성)
IP 관리	모든 노드가 동일 IP 공유	각 노드가 고유 IP 보유, DNS가 선택
Failover 속도	BGP 수렴 시간에 의존 (수초~수분)	DNS TTL에 의존 (수십 초~수분)
세밀한 제어	제한적 (BGP 경로 기반만 가능)	다양한 정책 기반 세밀한 제어 가능
운영 복잡도	높음 (BGP 피어링, 트래픽 엔지니어링 필요)	상대적으로 낮음
적합 워크로드	DNS, CDN 등 Stateless 서비스	웹 애플리케이션, API 등 Stateful 서비스 포함

GSLB + Anycast 조합 아키텍처

실무에서는 GSLB와 Anycast를 대립적으로 선택하기보다, 두 기술을 조합하여 시너지를 극대화하는 것이 일반적이다. 대표적인 조합 패턴은 다음과 같다.

- **GSLB DNS 서버를 Anycast로 배포**: 많은 CDN 사업자는 GSLB DNS 서버 자체를 Anycast로 배포한다. 이렇게 하면 DNS 쿼리 자체가 네트워크상 가장 가까운 GSLB 노드에서 처리되어 DNS 응답 지연을 최소화하고, GSLB 인프라 자체의 고가용성과 DDoS 방어력을 확보할 수 있다.
- **계층적 구조**: Anycast(L3)로 가장 가까운 PoP(Point of Presence)에 도달한 후, 해당 PoP 내에서 GSLB(DNS 기반)가 최적의 백엔드 서버를 결정하는 계층적 구조가 널리 사용된다.
- **Anycast의 DDoS 방어 이점**: Anycast는 공격 트래픽을 여러 노드로 자동 분산시키므로 DDoS 완화에 효과적이다. GSLB 인프라를 Anycast로 구성하면 GSLB 자체의 가용성이

크게 향상된다.

2.2 GSLB 연동 기술

2.2.1 CDN과 GSLB의 상호 보완 관계

CDN(Content Delivery Network)은 전 세계에 분산 배치된 엣지(Edge) 서버 네트워크를 통해 사용자에게 콘텐츠를 가장 가까운 위치에서 전달하는 기술이다. GSLB와 CDN은 다음과 같은 상호 보완적 관계를 형성한다.

GSLB는 CDN의 핵심 라우팅 메커니즘이다. CDN은 GSLB 기술을 사용하여 사용자를 최적의 엣지 서버로 라우팅한다. 사용자의 DNS 요청이 CDN의 GSLB 시스템으로 전달되면, GSLB가 지리적 위치, 서버 부하, 네트워크 성능 등을 종합 평가하여 최적의 엣지 서버 IP를 반환한다. 다시 말해, GSLB 없는 CDN은 사용자를 올바른 엣지 서버로 유도할 수 없다.

CDN 사업자가 자체 GSLB를 운영하는 방식: Cloudflare, Akamai, AWS CloudFront 등 주요 CDN 사업자는 자체 GSLB 시스템을 운영하며, 이를 통해 전 세계 PoP(Point of Presence)에 트래픽을 분배한다. 엔터프라이즈 고객은 CDN 사업자의 GSLB를 활용하거나, 자체 GSLB 계층을 CDN 앞단에 배치하여 멀티 CDN 전략을 구현하기도 한다.

이 구조에서 GSLB는 사용자의 DNS 쿼리를 수신하고, 지리적 위치와 서버 부하 등을 평가하여 최적의 CDN 엣지 서버를 선택한다. 선택된 엣지 서버가 요청된 콘텐츠를 캐시에 보유하고 있으면 즉시 반환하고, 캐시에 없으면 오리진 서버에서 콘텐츠를 가져와 사용자에게 전달하고 캐시에 저장한다.

2.2.2 GeoIP, API Gateway, Reverse Proxy

GSLB 생태계에는 CDN 외에도 GeoIP 데이터베이스, API Gateway, Reverse Proxy 등 다양한 기술이 연동되어 계층적인 트래픽 관리 아키텍처를 구성한다.

GeoIP 데이터 소스와 한계

GeoIP는 IP 주소를 지리적 위치(국가, 지역, 도시, 위도/경도 등)에 매핑하는 데이터베이스 및 기술이다. GSLB는 GeoIP 데이터를 활용하여 사용자를 지리적으로 가장 가까운 서버로 라우팅 한다.

주요 GeoIP 데이터 소스로는 **MaxMind GeoIP2/GeoLite2**(가장 널리 사용되는 상용/무료 GeoIP 데이터베이스), **IP2Location**(대안적 GeoIP 데이터 제공자), **RIR(Regional Internet Registry) 데이터**(ARIN, RIPE, APNIC 등에서 제공하는 IP 할당 정보) 등이 있다.

그러나 GeoIP는 다음과 같은 한계를 가진다.

- VPN 또는 프록시 사용 시 IP 주소와 실제 사용자 위치가 불일치
- 모바일 사용자의 IP 위치 정확도 문제 (통신사 게이트웨이 위치로 매핑)
- GeoIP 데이터베이스의 주기적 업데이트 필요성
- LDNS의 IP가 사용자 위치와 다를 수 있음 (EDNS Client Subnet으로 일부 해결)

API Gateway와 GSLB 역할 분담

API Gateway는 API 요청의 단일 진입점(Single Entry Point)으로, 인증/인가, 속도 제한(Rate Limiting), 프로토콜 변환, 요청 라우팅 등의 기능을 제공하는 미들웨어이다. GSLB와 API Gateway는 각각 다른 계층에서 트래픽을 관리하며, 다음과 같이 역할을 분담한다.

기능	GSLB	API Gateway
트래픽 분배 범위	글로벌 (데이터센터 간)	로컬 (데이터센터 내 서비스 간)
라우팅 기준	지리적 위치, 서버 상태, 지연 시간	API 경로, 버전, 헤더 등
인증/인가	해당 없음	JWT 검증, OAuth, API Key 등
속도 제한	해당 없음	클라이언트별/API별 Rate Limiting
프로토콜 처리	DNS 레벨	HTTP, gRPC, WebSocket 등
장애 처리	사이트 단위 페일오버	서비스 단위 Circuit Breaker

글로벌 API 배포 환경에서는 GSLB가 사용자를 가장 가까운 API Gateway로 라우팅하고, API Gateway가 해당 지역의 마이크로서비스로 요청을 전달하는 계층적 구조를 형성한다.

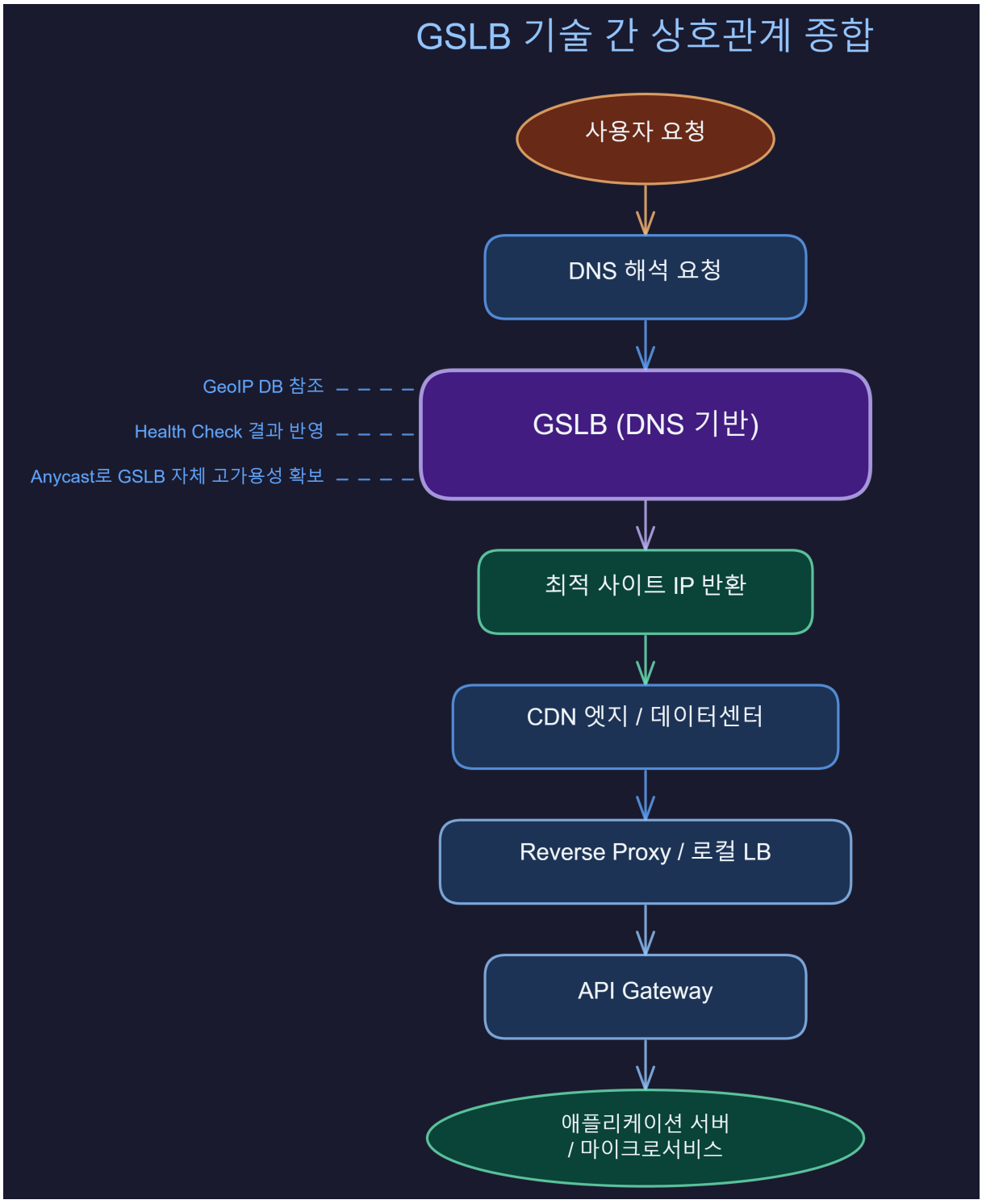
Reverse Proxy와 GSLB의 계층적 구조

Reverse Proxy(역방향 프록시)는 클라이언트와 백엔드 서버 사이에 위치하여 클라이언트의 요청을 대리 수신하고 적절한 백엔드 서버로 전달하는 서버이다. Nginx, HAProxy, Envoy 등이 대표적인 Reverse Proxy 솔루션이며, 로컬 로드밸런싱, SSL 종료, 캐싱, 압축 등의 기능을 제공한다.

GSLB와 Reverse Proxy는 “어느 사이트로 갈 것인가(GSLB)” 와 “사이트 내 어느 서버로 갈 것인가(Reverse Proxy)” 를 각각 결정하는 계층적 역할을 수행한다.

기술 간 상호관계 종합 다이어그램

다음 다이어그램은 GSLB를 중심으로 한 관련 기술들의 계층적 구조와 상호관계를 종합적으로 보여준다.



[그림 4] 475

이 계층 구조에서 각 기술의 역할을 요약하면 다음과 같다.

기술	GSLB에 대한 역할	관계 유형
DNS	GSLB의 동작 기반 프로토콜	기반 기술 (필수)

CDN	GSLB를 라우팅 엔진으로 활용하는 인프라	상위 소비자
Anycast	GSLB 인프라 자체의 고가용성 제공	보완 기술
Health Check	GSLB 라우팅 결정의 입력 데이터 제공	핵심 구성 요소
GeoIP	지리 기반 라우팅 판단의 데이터 소스	데이터 제공자
API Gateway	GSLB 하위에서 API 레벨 트래픽 관리 수행	하위 계층
Reverse Proxy	GSLB 하위에서 로컬 트래픽 분배 수행	하위 계층

2.3 GSLB 관련 표준 및 프로토콜

2.3.1 MEP (Metrics Exchange Protocol)

MEP(Metrics Exchange Protocol)는 **Citrix NetScaler(현 NetScaler ADC) 전용 독점 프로토콜**로, GSLB 구성에 참여하는 사이트 간에 메트릭 정보를 교환하는 데 사용된다. GSLB가 지능적인 트래픽 분산 결정을 내리기 위해서는 각 사이트의 실시간 상태 정보가 필요한데, MEP가 이 정보 교환을 담당한다.

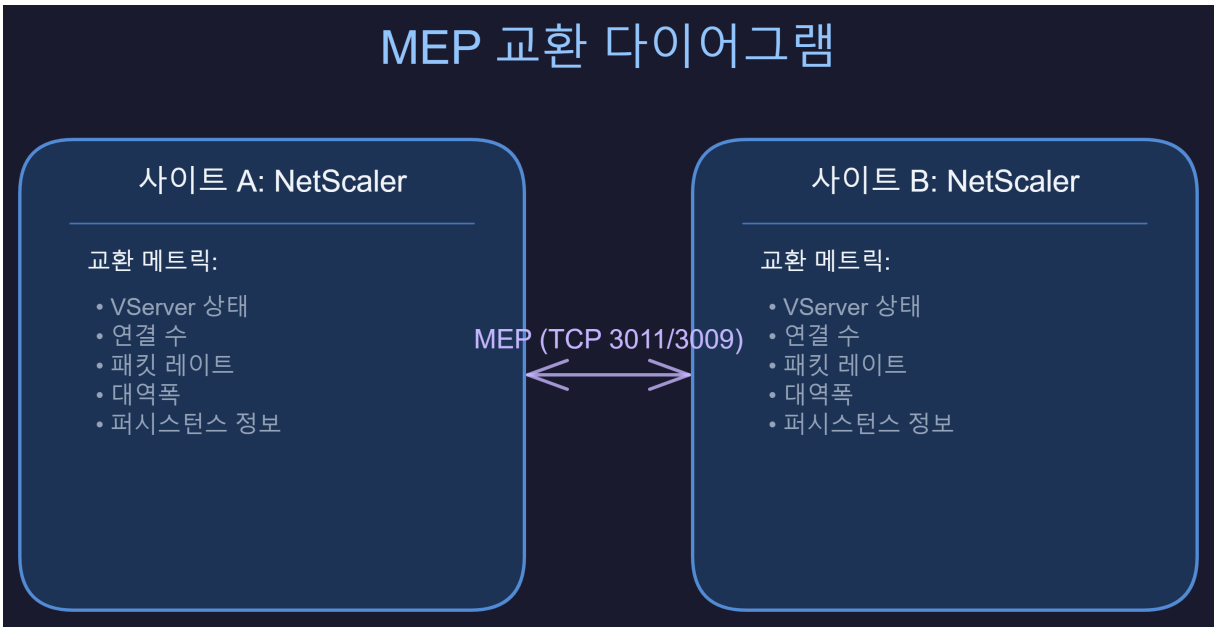
교환 메트릭 정보

MEP를 통해 교환되는 주요 정보는 세 가지 범주로 나뉜다.

범주	교환 정보
사이트 메트릭 (Site Metrics)	각 로드밸런싱/콘텐츠 스위칭 가상 서버의 상태, 현재 연결 수, 현재 패킷 레이트, 대역폭 사용량
네트워크 메트릭 (Network Metrics)	RTT(Round Trip Time), 네트워크 지연 시간 등 사이트 간 네트워크 성능 지표
퍼시스턴스 정보 (Persistence)	세션 지속성(persistence) 정보로 동일 클라이언트가 동일 사이트로 연결되도록 보장

통신 방식 및 토폴로지

MEP는 TCP 기반으로 동작하며, 포트 3011(비암호화 통신)과 3009(암호화 통신)를 사용한다. 기본적으로 메시(Mesh) 구조로 모든 사이트가 상호 통신하며, 대규모 배포 시에는 Parent-Child 계층형 구조를 사용한다.



토폴로지	최대 사이트 수	설명
Mesh (Active-Active)	최대 32개 사이트	모든 사이트가 상호 MEP 통신. 사이트 증가 시 MEP 트래픽이 기하급수적으로 증가
Parent-Child	부모 32개 + 자식 1,024개 (최대 1,056개)	대규모 배포를 위한 계층형 구조. 자식 사이트는 부모 사이트를 통해 메트릭 교환

멀티벤더 호환 불가 한계

MEP의 가장 큰 한계는 NetScaler 전용 독점 프로토콜이라는 점이다. F5 BIG-IP, A10 Networks 등 타 벤더 장비와의 상호운용이 불가능하므로, 멀티벤더 환경에서는 DNS 기반의 표준 방식이나 각 벤더별 API를 활용한 통합이 필요하다. 이러한 벤더 종속성은 기업이 GSLB 장비를 선택할 때 반드시 고려해야 하는 제약 사항이다.

2.3.2 EDNS Client Subnet (RFC 7871)

2.1.1절에서 EDNS Client Subnet(ECS)의 기본 개념과 동작 방식을 소개하였다. 본 절에서는 ECS의 기술적 세부 사항을 보다 심층적으로 다룬다.

배경: 재귀 DNS 리졸버 IP의 한계

재귀 DNS 리졸버 IP는 실제 클라이언트의 위치와 다를 수 있다. 특히 Google Public DNS(8.8.8.8), Cloudflare DNS(1.1.1.1) 등 글로벌 퍼블릭 DNS를 사용하는 클라이언트의 경우, 재귀 리졸버가 전 세계 여러 위치에 분산 배치되어 있으므로 클라이언트와 리졸버의 지리적

위치가 일치하지 않을 수 있다.

ECS 동작 흐름

[ECS 동작 흐름 상세]

[Step 1] 클라이언트(203.0.113.55)가 재귀 리졸버에 DNS 쿼리 전송

[Step 2] 재귀 리졸버가 클라이언트 IP의 일부를 마스킹하여 ECS 옵션에 포함
예: 203.0.113.0/24 (마지막 옥텟을 마스킹하여 프라이버시 보호)

[Step 3] 권한 DNS 서버(GSLB)가 ECS 정보를 수신

[Step 4] 리졸버 IP가 아닌 클라이언트 서브넷 203.0.113.0/24 기준으로
최적 서버를 판단

[Step 5] 응답에 SCOPE PREFIX-LENGTH를 포함하여 캐시 범위를 지정

ECS 옵션 필드 구조

필드	설명
FAMILY	주소 패밀리 (1 = IPv4, 2 = IPv6)
SOURCE PREFIX-LENGTH	클라이언트 IP에서 전달되는 비트 수 (프라이버시와 정확도의 균형). IPv4 일반적 /24, IPv6 일반적 /56
SCOPE PREFIX-LENGTH	응답이 유효한 범위 (권한 서버가 설정). 리졸버가 이 범위 내에서 응답을 캐시하고 재사용
ADDRESS	마스킹된 클라이언트 서브넷 주소

SOURCE PREFIX-LENGTH는 프라이버시와 위치 정확도 사이의 균형을 결정하는 핵심 파라미터이다. 값이 클수록 더 정확한 위치를 전달하지만 프라이버시 노출 위험이 증가하고, 값이 작을수록 프라이버시는 보호되지만 위치 정확도가 저하된다.

프라이버시 고려사항

- ECS는 클라이언트의 전체 IP 주소가 아닌 서브넷 정보만 전달한다
- 일반적으로 IPv4는 /24(256개 IP 범위), IPv6는 /56으로 마스킹한다
- 개별 사용자 식별 없이 대략적인 위치 정보만 전달하여 프라이버시를 보호한다
- 그러나 서브넷 정보와 다른 데이터를 결합하면 사용자 추적이 가능할 수 있다는 우려가 존재한다
- 프라이버시를 우선시하는 일부 리졸버(예: Cloudflare 1.1.1.1)는 ECS를 기본적으로 비활성화하고 있다

지원 현황

구분	지원 여부
퍼블릭 DNS	Google Public DNS, OpenDNS 등 주요 퍼블릭 DNS가 ECS 지원
GSLB 장비	NetScaler, F5 BIG-IP DNS 등 주요 벤더 지원
CDN	Cloudflare, Akamai, AWS CloudFront 등 주요 CDN 지원
ISP DNS	ISP마다 지원 여부가 다르며, 미지원 ISP도 다수 존재

[K8GB 맥락]

K8GB는 상용 GSLB 장비와는 근본적으로 다른 방식으로 DNS 프로토콜을 처리한다. 상용 장비가 자체적인 독점 DNS 엔진을 사용하는 것과 달리, K8GB는 **CoreDNS를 내장하여 DNS 프로토콜을 직접 처리**한다. CoreDNS는 CNCF Graduated 프로젝트이자 Kubernetes의 기본 클러스터 DNS로 채택된 오픈소스 DNS 서버로, 풍부한 플러그인 생태계와 검증된 안정성을 제공한다.

더욱 주목할 점은, K8GB가 상용 GSLB의 MEP(Metrics Exchange Protocol)와 같은 독점 프로토콜 대신 **DNS 레코드 기반으로 클러스터 간 상태를 공유**한다는 것이다. 각 클러스터의 K8GB Controller는 자신이 관리하는 서비스의 상태(Pod Readiness 기반)를 DNS TXT/A 레코드로 공개하며, 다른 클러스터의 K8GB는 이 DNS 레코드를 조회하여 원격 클러스터의 상태를 파악한다. 이 접근 방식은 MEP의 멀티벤더 호환 불가 한계를 근본적으로 해결하며, DNS라는 표준 프로토콜만으로 클러스터 간 통신이 가능하므로 방화벽 설정이 단순하고 네트워크 토폴로지에 대한 제약이 최소화

된다. 또한 ExternalDNS를 통해 Route 53, Cloud DNS, Infoblox 등 다양한 DNS 프로바이더와 연동할 수 있어, 특정 인프라에 종속되지 않는 유연한 구성이 가능하다.

2장에서는 GSLB의 기반 기술인 DNS와 BGP Anycast, 연동 기술인 CDN, GeoIP, API Gateway, Reverse Proxy, 그리고 관련 표준 및 프로토콜인 MEP와 EDNS Client Subnet을 살펴보았다. GSLB는 이러한 기술들과 유기적으로 결합하여 글로벌 트래픽 관리 아키텍처를 구성하며, 각 기술의 역할과 한계를 이해하는 것은 효과적인 GSLB 설계의 전제 조건이다. 3장에서는 이러한 기술적 이해를 바탕으로, 실제 엔터프라이즈 환경에서 활용되는 GSLB 아키텍처 패턴을 살펴본다.

3장. GSLB 아키텍처 패턴

앞서 1장에서 GSLB의 기본 동작 원리를, 2장에서 GSLB 관련 기술 생태계를 살펴보았다. 본 장에서는 이러한 기술적 이해를 기반으로, 실제 엔터프라이즈 환경에서 구현되는 대표적인 GSLB 아키텍처 패턴을 다룬다. 멀티사이트 레퍼런스 아키텍처부터 하이브리드 클라우드, 멀티 클라우드 환경에 이르기까지, 각 패턴의 구성 방식과 설계 고려사항을 아키텍처 다이어그램과 함께 상세히 제시한다.

3.1 멀티사이트 레퍼런스 아키텍처

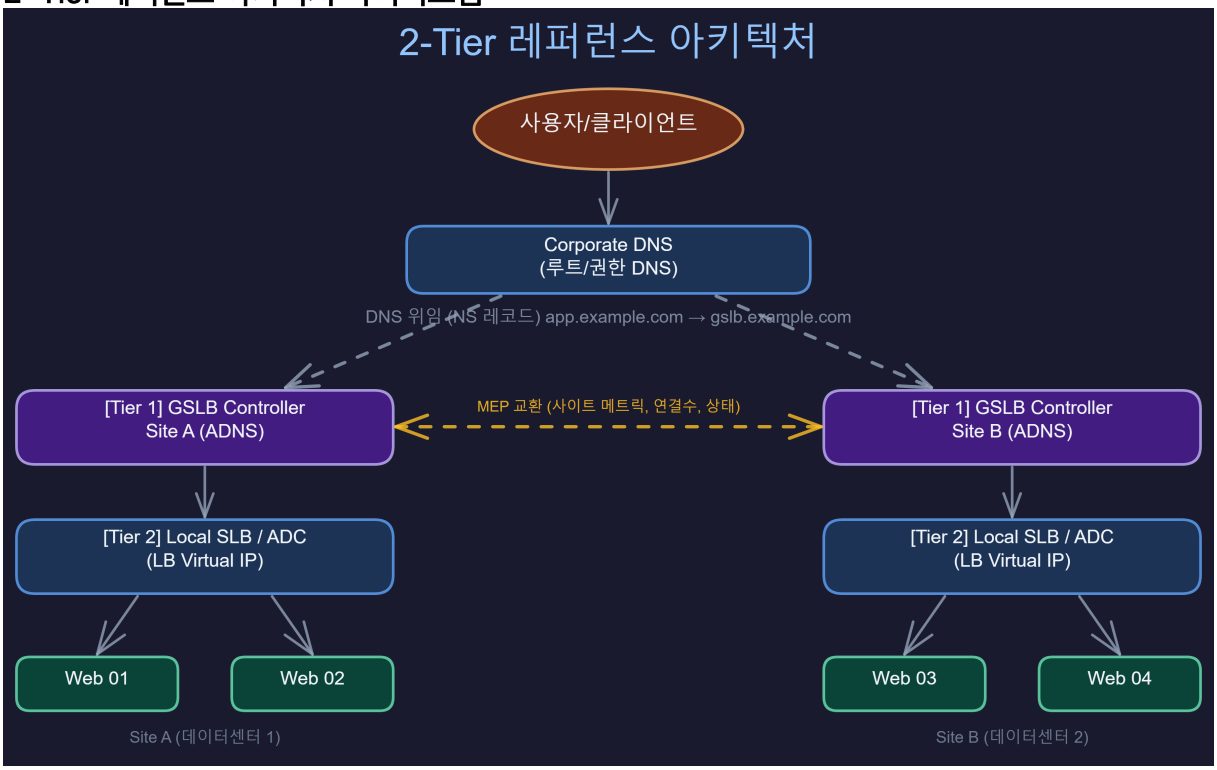
3.1.1 2-Tier 계층 구조 (Global DNS Tier + Local SLB Tier)

GSLB의 기본 레퍼런스 아키텍처는 **글로벌 DNS 티어(Tier 1)**와 **로컬 SLB 티어(Tier 2)**로 구성되는 2-Tier 계층 구조이다. 이 구조에서 각 계층의 역할과 구성 요소는 다음과 같다.

계층	역할	구성 요소
Tier 1: 글로벌 DNS 티어	사이트 간 트래픽 분배 결정	GSLB Controller, ADNS 서비스, MEP
Tier 2: 로컬 SLB 티어	사이트 내부 서버 간 로드밸런싱	Local LB VIP, 서버 팜, 헬스체크

Tier 1(글로벌 DNS 티어)에서는 GSLB Controller가 각 사이트의 상태를 MEP(또는 DNS 기반 상태 교환)를 통해 파악하고, ADNS(Authoritative DNS) 서비스가 클라이언트의 DNS 쿼리에 대해 최적의 사이트 VIP(Virtual IP)를 응답한다. Tier 2(로컬 SLB 티어)에서는 각 사이트의 로컬 로드밸런서가 Tier 1에서 선택된 사이트로 도착한 트래픽을 내부 서버 풀에 분배한다.

2-Tier 레퍼런스 아키텍처 다이어그램



트래픽 흐름 상세

DNS 요청부터 실제 애플리케이션 트래픽까지의 흐름은 다음과 같다.

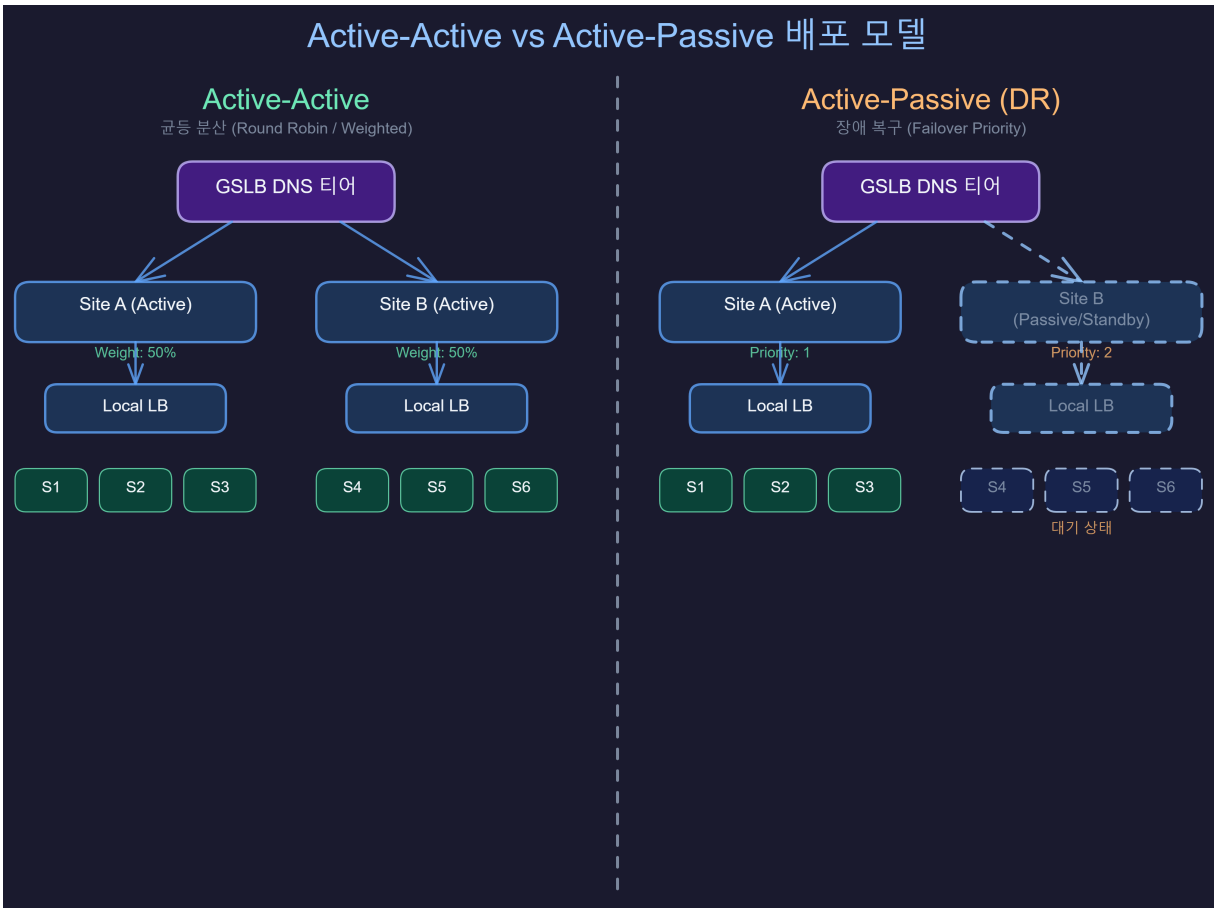
이 흐름에서 주목할 점은 GSLB가 (3)번 단계에서만 개입하며, 실제 애플리케이션 트래픽((5)~(6) 단계)은 GSLB를 경유하지 않고 직접 사이트의 로컬 LB를 통해 처리된다는 것이다. 이는 GSLB가 데이터 경로(Data Path)에 위치하지 않는 DNS 기반 방식의 핵심 특성이다.

3.1.2 Active-Active / Active-Passive 배포 모델

GSLB의 2-Tier 아키텍처는 Active-Active 또는 Active-Passive 배포 모델로 구현될 수 있다. 각 모델의 특성과 적합 시나리오는 서비스의 가용성 요구수준, 인프라 비용, 데이터 일관성 요구사항에 따라 달라진다.

Active-Active 배포

Active-Active 모델에서는 모든 사이트가 동시에 트래픽을 처리한다. 가중치(Weight) 기반으로 트래픽을 비율 배분하며, MEP를 통해 실시간으로 사이트 간 메트릭을 교환한다.



Active-Active 모델의 장점은 다음과 같다.

- 모든 인프라 자원을 상시 활용하므로 투자 효율성이 높음
- 사이트 간 부하를 분산하여 단일 사이트에 대한 과부하를 방지
- 한 사이트의 장애 시 나머지 사이트가 즉시 전체 트래픽을 흡수 (별도의 스탠바이 기동 시간 불필요)
- 가중치 조정으로 카나리 배포, 블루/그린 배포 등 배포 전략에 활용 가능

Active-Active 모델의 고려사항은 다음과 같다.

- 양방향 데이터 복제로 인한 Write Conflict, Split-Brain 문제 해결 필요
- 모든 사이트에서 동일한 서비스 수준을 유지해야 하므로 운영 복잡성 증가
- 데이터 일관성 보장을 위한 추가적인 아키텍처 설계 필요

Active-Passive (DR) 배포

Active-Passive 모델에서는 Primary 사이트가 모든 트래픽을 처리하고, Passive(Standby) 사이트는 Primary 사이트의 장애 시에만 활성화된다.

Active-Passive 모델의 장점은 다음과 같다.

- 데이터 일관성 관리가 상대적으로 단순 (단방향 복제)
- Write Conflict, Split-Brain 문제가 발생하지 않음
- DR 시나리오에 특화된 명확한 역할 분담

Active-Passive 모델의 고려사항은 다음과 같다.

- Passive 사이트가 평상시 유휴 상태이므로 인프라 투자 효율성이 낮음
- 페일오버 시 Passive 사이트의 워밍업(캐시, 연결 풀 등) 시간 필요
- DNS TTL 캐싱에 의한 페일오버 지연 발생 가능

DNS 위임 구성 방식

Active-Active 또는 Active-Passive 모델 모두에서 기업 DNS에서 GSLB로의 위임은 다음과 같은 NS 레코드와 CNAME 레코드를 통해 구성한다.

[외부 DNS 존 파일 구성 예시]

```
;; A 레코드 - 각 GSLB ADNS의 공인 IP
gslb-site-a    IN  A   203.0.113.10
gslb-site-b    IN  A   198.51.100.10

;; NS 레코드 - 서브도메인을 GSLB로 위임
gslb           IN  NS  gslb-site-a.example.com.
gslb           IN  NS  gslb-site-b.example.com.

;; CNAME - 실제 서비스 도메인을 GSLB 서브존으로 연결
app            IN  CNAME  app.gslb.example.com.
www            IN  CNAME  www.gslb.example.com.
```

이 구성에서 app.example.com에 대한 DNS 쿼리는 CNAME을 통해 app.gslb.example.com으로 리디렉션되고, NS 위임에 따라 GSLB ADNS(gslb-site-a 또는 gslb-site-b)가 쿼리

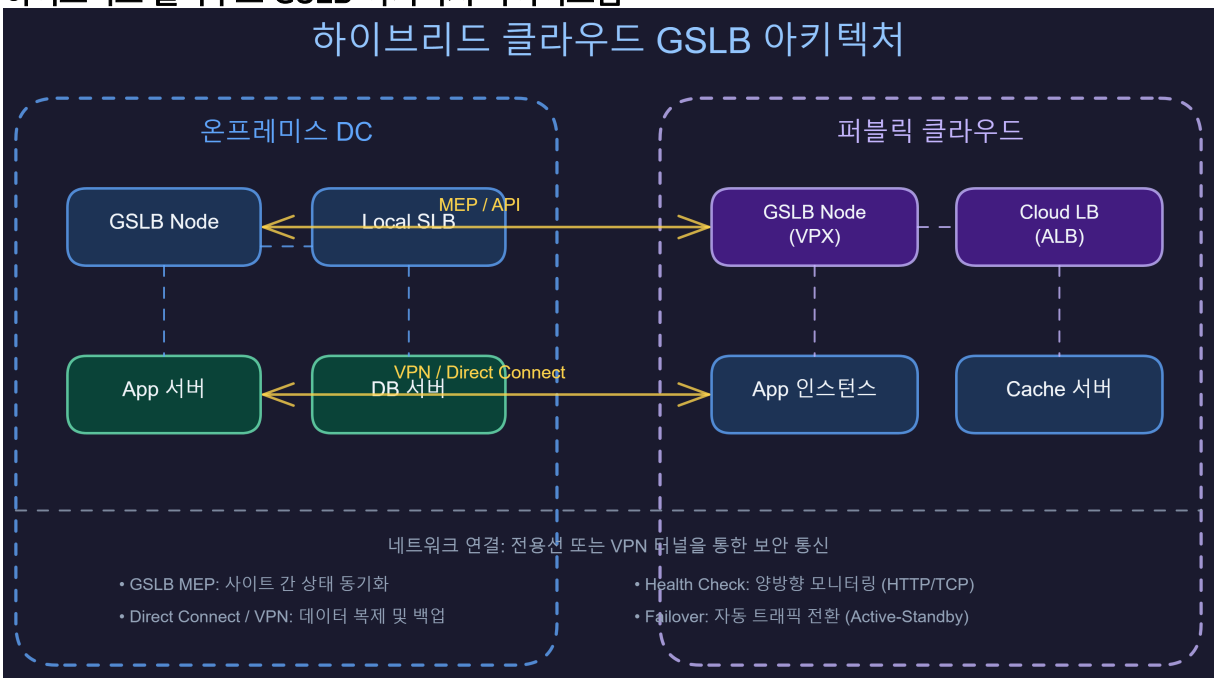
를 처리하여 최적의 사이트 VIP를 반환한다. NS 레코드가 두 개의 GSLB ADNS를 가리키므로, 하나의 GSLB 노드에 장애가 발생하더라도 다른 GSLB 노드가 DNS 쿼리를 처리할 수 있어 GSLB 자체의 고가용성이 확보된다.

3.2 하이브리드 클라우드 아키텍처

3.2.1 온프레미스 + 퍼블릭 클라우드 GSLB 구성

하이브리드 클라우드 GSLB는 온프레미스 데이터센터와 퍼블릭 클라우드 환경을 단일 GSLB 도메인으로 통합 관리하는 아키텍처이다. 온프레미스의 안정성과 보안성, 퍼블릭 클라우드의 확장성과 유연성을 결합하여 최적의 인프라 운영을 가능하게 한다.

하이브리드 클라우드 GSLB 아키텍처 다이어그램



GSLB 노드 배치 전략

하이브리드 환경에서 GSLB 노드의 배치는 각 환경의 특성에 맞게 설계해야 한다.

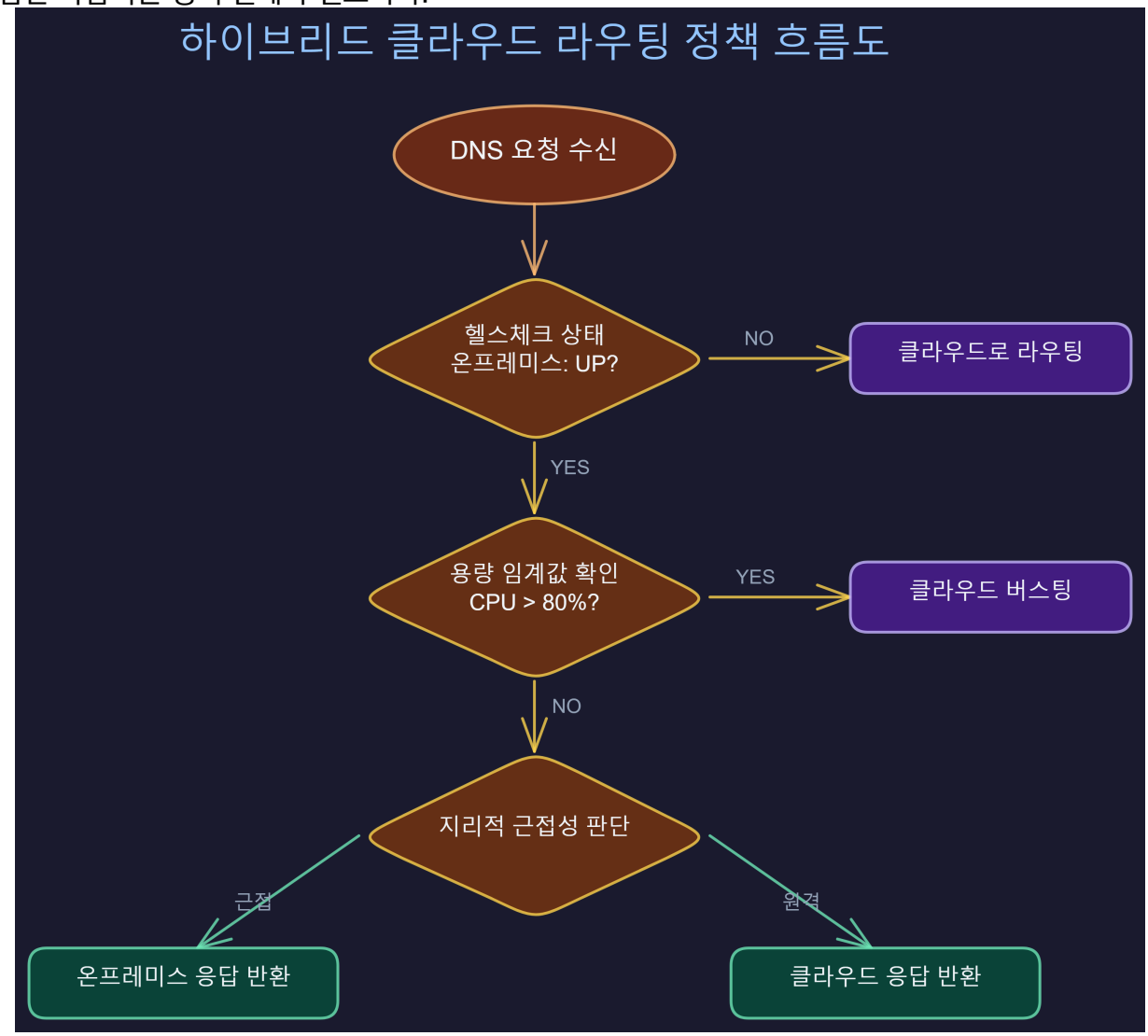
위치	GSLB 노드 형태	역할
온프레미스 DC	물리 어플라이언스 (F5 BIG-IP, NetScaler MPX)	주 GSLB 컨트롤러, ADNS, 온프레미스 로컬 LB

퍼블릭 클라우드	가상 어플라이언스 (VPX, VE) 또는 SaaS	보조 GSLB 노드, 클라우드 로컬 LB

온프레미스에서는 전용 하드웨어 어플라이언스의 높은 성능과 안정성을 활용하고, 퍼블릭 클라우드에서는 가상 어플라이언스를 통해 유연한 확장이 가능하도록 구성하는 것이 일반적이다.

트래픽 라우팅 정책 설계

하이브리드 환경에서의 트래픽 라우팅은 단순한 부하 분산을 넘어, 비용 최적화와 규정 준수를 포함한 복합적인 정책 설계가 필요하다.



이 흐름도에서 첫 번째 판단 기준은 헬스체크 상태이다. 온프레미스가 정상(UP)이면 용량 임계값을 확인하고, 임계값을 초과하면 클라우드 버스팅(Cloud Bursting)을 통해 퍼블릭 클라우드로 트래픽을 분산한다. 온프레미스가 비정상(DOWN)이면 즉시 클라우드로 페일오버한다.

주요 고려사항

하이브리드 클라우드 GSLB 설계 시 다음 네 가지 영역에 대한 면밀한 검토가 필요하다.

네트워크 연결 – 전용선 (Direct Connect / ExpressRoute): 안정적인 MEP 통신과 데이터 동기화에 필수적이다. 전용선은 예측 가능한 대역폭과 낮은 지연 시간을 제공하므로, 실시간 메트릭 교환과 데이터베이스 복제에 적합하다. – **VPN 백업**: 전용선 장애 시 대비한 IPSec VPN 이중화를 구성해야 한다. 전용선과 VPN을 Active-Standby로 운영하여 네트워크 연결의 가용성을 확보한다. – **대역폭 계획**: MEP 트래픽, 데이터 복제 트래픽, 헬스체크 트래픽을 포함한 총 대역폭을 산정하여 네트워크 용량을 계획해야 한다.

헬스체크 설계 – 온프레미스와 클라우드 간 비대칭 헬스체크 간격을 설정할 수 있다. 예를 들어 온프레미스는 10초 간격, 클라우드는 30초 간격으로 차별화할 수 있다. – 클라우드 환경에서는 CloudWatch(AWS), Azure Monitor 등 클라우드 네이티브 모니터링 서비스와 연동한 API 기반 헬스체크를 활용하는 것이 효율적이다. – 크로스 사이트 헬스체크(온프레미스에서 클라우드로, 또는 그 반대)의 경우, 사이트 간 네트워크 레이턴시를 감안하여 타임아웃 값을 넉넉하게 설정해야 한다. 네트워크 지연으로 인한 헬스체크 오탐(False Positive)을 방지하기 위함이다.

데이터 일관성 – DB 복제 전략: 온프레미스 Master에서 클라우드 Read Replica로의 비동기 복제가 일반적이다. Active-Active 구성 시에는 양방향 복제에 따른 Write Conflict 해소 전략이 필요하다. – 세션 상태 공유: Redis, Memcached 등 외부 세션 스토어를 활용하거나, JWT/Token 기반 Stateless 세션 관리를 채택하여 사이트 간 세션 일관성을 확보한다. – DNS TTL 관리: 페일오버 속도와 DNS 캐시 효율 간의 트레이드오프를 고려하여 적절한 TTL 값을 설정한다.

비용 최적화 – 클라우드 버스팅(Cloud Bursting): 평상시에는 온프레미스에서 트래픽을 처리하고, 피크 시에만 퍼블릭 클라우드로 자동 확장하는 전략이다. GSLB의 용량 기반 라우팅 정책과 결합하면, 온프레미스의 CPU/메모리 사용률이 임계값을 초과할 때 자동으로 클라우드로 트래픽을 분산할 수 있다. – **Egress 비용 고려**: 퍼블릭 클라우드에서 온프레미스 또는 인터넷으로 나가는 아웃바운드 트래픽에는 비용이 발생한다. GSLB 라우팅 정책 설계 시 이 Egress 비용을 가중치에 반영하여 비용을 최적화할 수 있다. – **예약 인스턴스 + 온디맨드 혼합**: 베이스라인 트래픽은 예약 인스턴스로, 버스팅 트래픽은 온디맨드 인스턴스로 처리하는 혼합 전략을 통해 비용 효율성을 극대화한다.

3.3 멀티 클라우드 GSLB 설계

3.3.1 클라우드 간 트래픽 분산 설계

멀티 클라우드 GSLB는 AWS, Azure, GCP 등 복수의 퍼블릭 클라우드에 분산 배포된 서비스를 단일 GSLB 도메인으로 통합 관리하는 아키텍처이다. 특정 클라우드 벤더에 대한 종속(Lock-in)을 방지하고, 각 클라우드의 강점을 결합하며, 단일 클라우드 장애 시 서비스 연속성을 보장하는 것이 핵심 목적이다.

클라우드 네이티브 GSLB 서비스 비교

각 주요 클라우드는 자체 GSLB 서비스를 제공하지만, 이들은 기본적으로 해당 클라우드 내부에 최적화되어 있으며 타 클라우드와의 통합에는 제약이 있다.

기능	AWS Route 53	Azure Traffic Manager	GCP Cloud DNS
라우팅 방식	지연시간, 지리적, 가중치, 페일오버	성능, 지리적, 가중치, 우선순위	지리적, WRR
헬스체크	HTTP/HTTPS/TCP	HTTP/HTTPS/TCP	- (외부 연동)
Anycast	지원	지원	지원
범위	AWS 내부 최적화	Azure 내부 최적화	GCP 내부 최적화
한계	타 클라우드 통합 제한적	타 클라우드 통합 제한적	타 클라우드 통합 제한적

각 클라우드의 네이티브 GSLB 서비스는 자사 클라우드 내부에서는 우수한 통합성을 제공하지만, 멀티 클라우드 환경에서는 “모든 클라우드를 동일한 수준으로 관리” 하기 어렵다는 구조적 한계가 존재한다. 이러한 한계를 극복하기 위해 벤더 중립적인 GSLB 계층의 도입이 필요하다.

멀티 클라우드 GSLB 아키텍처 다이어그램

이 아키텍처에서 최상위에 위치하는 **벤더 중립 GSLB 계층**이 핵심이다. 이 계층은 특정 클라우드에 종속되지 않는 독립적인 글로벌 트래픽 관리자로서, 모든 클라우드의 엔드포인트를 동일한 기준으로 평가하고 라우팅한다. 각 클라우드 내부에서는 해당 클라우드의 네이티브 GSLB/LB 서비스(Route 53, Traffic Manager, Cloud DNS 등)가 로컬 트래픽 관리를 담당한다.

벤더 중립적 GSLB 설계 원칙

벤더 중립 GSLB의 핵심은 특정 클라우드에 종속되지 않는 독립적인 글로벌 트래픽 관리 계층을 구축하는 것이다.

이 설계 원칙의 핵심은 Layer 3(글로벌 트래픽 관리)이 Layer 2(클라우드별 로컬 LB) 및 Layer 1(컴퓨팅)과 독립적으로 운영된다는 것이다. Layer 3에서 사용하는 GSLB 솔루션은 모든 클라우드의 엔드포인트를 동일한 인터페이스로 관리할 수 있어야 하며, Layer 4(관리/오케스트레이션)에서는 Terraform 등 IaC(Infrastructure as Code) 도구를 통해 모든 클라우드의 GSLB 설정을 코드로 관리한다.

벤더 중립 GSLB 솔루션 옵션

솔루션	유형	특징
F5 BIG-IP DNS	어플라이언스/가상	엔터프라이즈 GSLB, 온프레미스+클라우드 통합 지원
NetScaler (Citrix)	어플라이언스/가상	MEP 기반 메트릭 교환, ADC 통합, 하이브리드 환경 강점
NS1 (IBM)	SaaS	API 우선 설계, Filter Chain 기반 유연한 라우팅
Cloudflare LB	SaaS	글로벌 Anycast 네트워크, 간편한 설정, DDoS 방어 통합
A10 Thunder	어플라이언스/가상	하이브리드 클라우드 특화, 고성능
HashiCorp Consul	오픈소스	서비스 메시 통합, DNS 인터페이스 제공, 클라우드 네이티브

멀티 클라우드 GSLB 설계 시 핵심 고려사항

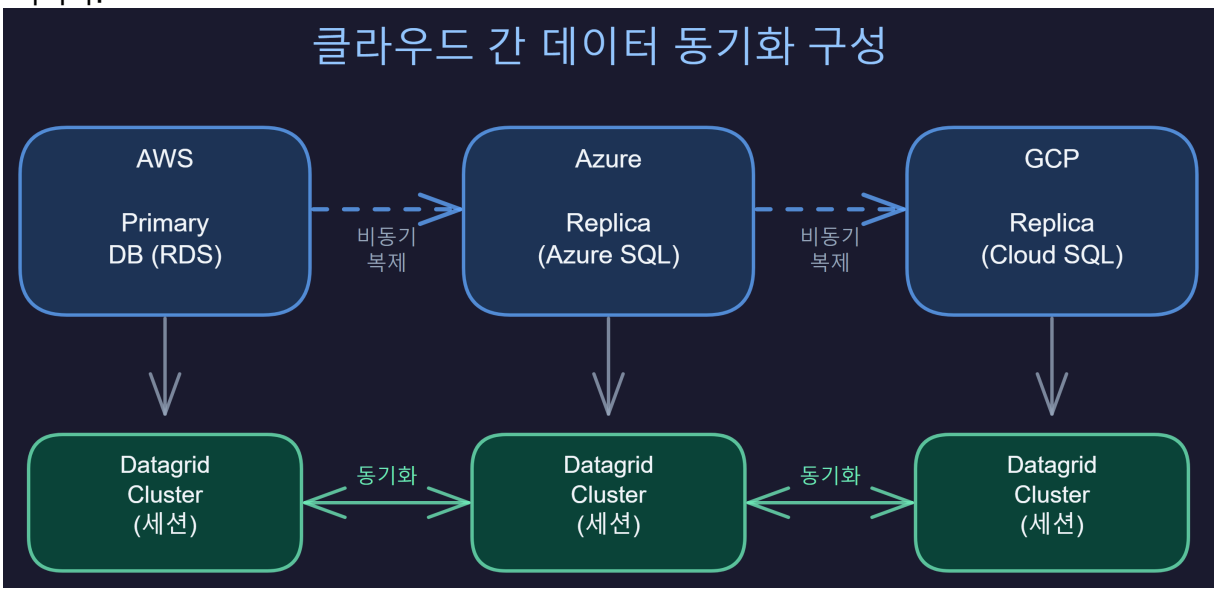
A. DNS TTL 전략

멀티 클라우드 환경에서의 DNS TTL 전략은 페일오버 속도와 DNS 쿼리 부하 간의 균형을 고려하여 설계해야 한다.

일반적으로 평상시에는 60~120초의 TTL을 유지하여 DNS 쿼리 부하를 관리하고, 장애 감지 시 동적으로 TTL을 30초로 단축하여 빠른 페일오버를 가능하게 하는 전략이 권장된다.

B. 클라우드 간 데이터 동기화

멀티 클라우드 환경에서 서비스 일관성을 유지하기 위해서는 클라우드 간 데이터 동기화가 필수적이다.

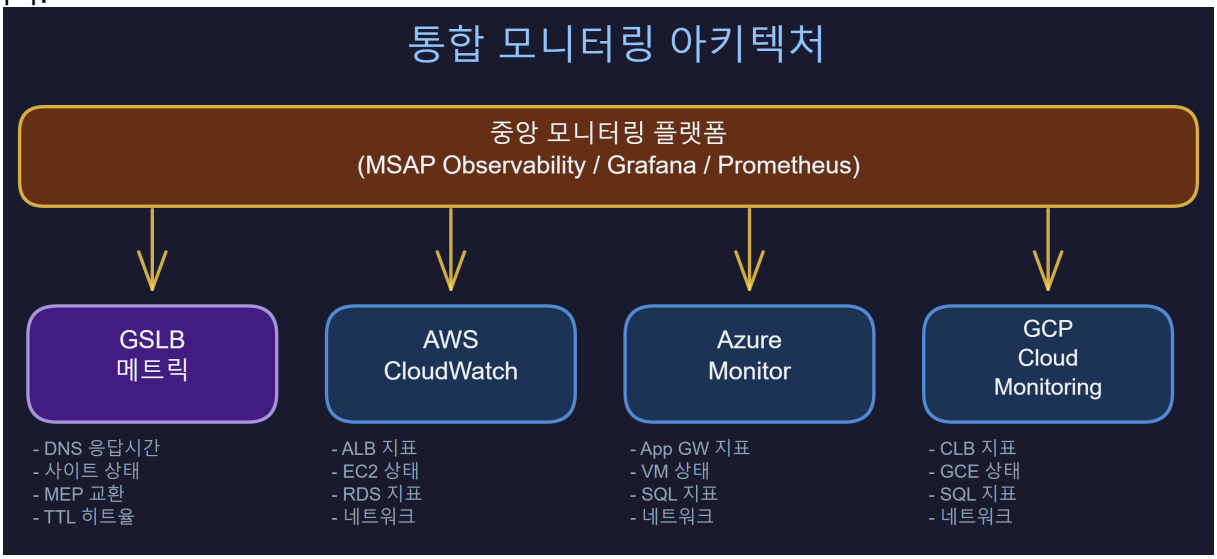


데이터베이스는 Primary-Replica 구조로 비동기 복제를 수행하고, 세션 데이터는 Redis 클러스터 간 실시간 동기화를 통해 사이트 간 세션 일관성을 확보한다. 이때 클라우드 간 네트워크

지연에 의한 복제 지연(Replication Lag)을 고려하여 Eventual Consistency 모델을 기반으로 설계하는 것이 현실적이다.

C. 통합 모니터링

멀티 클라우드 GSLB 환경에서는 각 클라우드의 개별 모니터링 서비스(CloudWatch, Azure Monitor, Cloud Monitoring)와 GSLB 자체의 메트릭을 통합하는 중앙 모니터링 플랫폼이 필수적이다.



통합 모니터링 플랫폼은 모든 클라우드와 GSLB의 메트릭을 단일 대시보드에서 조회할 수 있게 하며, 이상 징후 감지 시 통합 알림을 발생시켜 신속한 대응을 가능하게 한다. GSLB 레벨의 메트릭(DNS 응답 시간, 사이트 상태, 페일오버 이벤트 등)과 각 클라우드의 인프라 메트릭(서버 상태, DB 지표, 네트워크 지표 등)을 상관 분석(Correlation Analysis)함으로써 근본 원인 분석(Root Cause Analysis)의 효율성을 높일 수 있다.

[K8GB 맥락]

K8GB는 멀티 클라우드 GSLB 설계에서 **특정 클라우드 벤더에 종속되지 않는** 진정한 벤더 중립적 솔루션으로서의 가치를 제공한다. 상용 GSLB 장비(F5, NetScaler 등)가 자체 가상 어플라이언스를 각 클라우드에 배포하는 방식과 달리, K8GB는 **Kubernetes가 실행되는 모든 환경에서 동일하게 동작**하는 Kubernetes-native 방식을 채택한다.

K8GB의 멀티 클라우드 아키텍처에서 핵심적인 역할을 하는 것은 **ExternalDNS**이다. ExternalDNS는 K8GB Controller가 관리하는 DNS 레코드를 상위 Cloud DNS 서비스에 자동으로 동기화하는 컴포넌트로, 다음과 같은 DNS 프로바이더를 지원한다.

- AWS Route 53
- Google Cloud DNS
- Azure DNS
- Infoblox (온프레미스)
- CoreDNS (자체 호스팅)
- 기타 다수의 DNS 프로바이더

이러한 광범위한 DNS 프로바이더 지원을 통해, K8GB는 AWS EKS, Azure AKS, GCP GKE, 온프레미스 Kubernetes 클러스터 등 이기종 Kubernetes 환경 간에 **동일한 Gslb CRD와 동일한 운영 방식**으로 글로벌 트래픽을 관리할 수 있다. 각 클러스터에 독립적으로 배포되는 K8GB의 분산 아키텍처는 전용 관리 클러스터가 불필요하여 단일 장애점(SPoF)을 제거하며, 클러스터 간 상태 공유는 DNS 레코드 기반으로 이루어지므로 MEP와 같은 독점 프로토콜에 대한 의존성이 없다. 이는 멀티 클라우드 환경에서의 GSLB 운영 복잡성을 크게 줄이고, 진정한 의미의 클라우드 이식성(Cloud Portability)을 실현한다.

3장에서는 GSLB의 대표적인 아키텍처 패턴인 멀티사이트 2-Tier 구조, Active-Active/Active-Passive 배포 모델, 하이브리드 클라우드 아키텍처, 멀티 클라우드 GSLB 설계를 살펴보았다. 각 아키텍처 패턴은 서비스의 가용성 요구수준, 인프라 구성, 비용 구조, 규정 준수 요건에 따라 선택되며, 실제 환경에서는 이러한 패턴들이 조합되어 적용되는 경우가 많다. 4장에서는 GSLB가 재해복구(DR) 아키텍처에서 어떤 핵심 역할을 수행하는지, 그리고 DR 설계 시의 주요 고려사항을 살펴본다.

1~3장 요약

장	핵심 내용
1장. GSLB 개요	GSLB의 정의(DNS 기반 글로벌 트래픽 분배), SLB와의 차이, DNS 라우팅 메커니즘, 6가지 주요 알고리즘, 4단계 헬스체크/페일오버 프로세스
2장. GSLB 기술 생태계	DNS(GSLB의 핵심 기반), BGP Anycast 비교, CDN/GeoIP/API Gateway/Reverse Proxy 연동, MEP 독점 프로토콜, EDNS Client Subnet(RFC 7871)
3장. GSLB 아키텍처 패턴	2-Tier 레퍼런스 아키텍처, Active-Active/Active-Passive 배포, 하이브리드 클라우드 구성, 멀티 클라우드 벤더 중립 GSLB 설계

1장. GSLB 개요

오늘날 기업의 IT 서비스는 전 세계 사용자를 대상으로 운영되며, 서비스 중단은 곧바로 매출 손실과 브랜드 신뢰도 하락으로 이어진다. 이러한 환경에서 지리적으로 분산된 인프라 간에 트래픽을 지능적으로 분배하는 기술인 GSLB(Global Server Load Balancing)는 엔터프라이즈 아키텍처의 핵심 구성 요소로 자리매김하고 있다. 본 장에서는 GSLB의 기본 개념과 필요성을 정의하고, DNS 기반 트래픽 라우팅의 동작 원리, 주요 라우팅 알고리즘, 그리고 헬스체크와 자동 페일오버 메커니즘을 상세히 살펴본다.

1.1 GSLB의 정의와 필요성

1.1.1 GSLB란 무엇인가

GSLB(Global Server Load Balancing)는 **지리적으로 분산된 복수의 데이터센터 간에 네트워크 트래픽을 지능적으로 분배하는 기술**이다. 일반적인 로드밸런서(SLB, Server Load Balancer)가 단일 데이터센터 내부에서 서버 간 트래픽을 분배하는 것과 달리, GSLB는 데이터센터 단위, 즉 사이트(Site) 간의 글로벌 트래픽 분배를 담당한다.

GSLB의 핵심 동작 원리는 **DNS(Domain Name System) 기반 라우팅**이다. 클라이언트가 특정 도메인에 접속하기 위해 DNS 쿼리를 발생시키면, GSLB는 각 데이터센터의 가용성, 부하 상태, 클라이언트의 지리적 위치, 관리자가 설정한 라우팅 정책 등을 종합적으로 평가하여 최적의

데이터센터 IP 주소를 DNS 응답으로 반환한다. 이후 클라이언트는 반환된 IP 주소로 직접 연결하며, GSLB 자체는 실제 데이터 전송 경로(Data Path)에 관여하지 않는다.

이 점이 GSLB와 일반 로드밸런서(SLB)의 가장 근본적인 차이점이다. SLB는 트래픽이 자신을 경유하며 실시간으로 분배 결정을 내리는 반면, GSLB는 DNS 레벨에서만 개입하여 클라이언트가 최초에 연결할 서버의 IP 주소를 결정한다.

구분	SLB (Server Load Balancer)	GSLB (Global Server Load Balancer)
분배 범위	사이트 내부 (로컬)	사이트 간 (글로벌)
동작 방식	트래픽이 LB를 경유 (인라인)	DNS 응답으로 IP 반환 (아웃오브밴드)
개입 시점	모든 요청에 대해 실시간 분배	DNS 쿼리 시점에만 개입
데이터 경로	Data Path 상에 위치	Data Path에서 벗어남
동작 계층	L4(TCP/UDP) / L7(HTTP)	DNS 계층
대표 기능	서버 간 부하 분산, 세션 유지, SSL 종료	사이트 선택, 재해복구 페일오버, 지리 기반 라우팅

이처럼 GSLB는 DNS 응답을 조작하여 클라이언트가 연결할 서버의 IP 주소를 결정하는 방식으로 동작하며, 이는 기존 DNS 인프라를 활용하므로 별도의 네트워크 장비 변경 없이 구축할 수 있다는 장점을 제공한다.

1.1.2 왜 GSLB가 필요한가

현대 엔터프라이즈 환경에서 GSLB가 필수적인 기술로 부상한 배경에는 다음과 같은 비즈니스 및 기술적 요인이 있다.

첫째, 글로벌 서비스 확대와 사용자 경험 요구이다. 기업의 서비스가 국내를 넘어 글로벌 시장으로 확대됨에 따라, 전 세계 사용자에게 낮은 지연 시간(Latency)으로 서비스를 제공해야 하는 요구가 증가하고 있다. 사용자가 물리적으로 가까운 데이터센터에 접속하도록 유도하는 GSLB의 지리 기반 라우팅(Geolocation Routing)은 이러한 요구를 충족하는 핵심 기술이다.

둘째, 무중단 운영(High Availability) 요구가 강화되고 있다. 서비스 중단에 대한 고객과 시장의 기대치는 매우 높아졌다. 단일 데이터센터에 장애가 발생하더라도 서비스가 중단되지 않도록 자동 페일오버(Automatic Failover)를 제공하는 GSLB는 99.99% 이상의 SLA(Service Level Agreement)를 달성하기 위한 필수 인프라이다.

셋째, 재해복구(DR, Disaster Recovery) 규제가 의무화되고 있다. 특히 금융권에서는 전자금융감독규정에 따라 RTO(Recovery Time Objective) 2시간 이내의 재해복구 체계를 의무적으로 갖추어야 한다. 이커머스 기업 역시 99.99% SLA를 보장하기 위해 멀티사이트 DR 체계를 구축하고 있으며, GSLB는 DR 구성에서 트래픽을 정상 사이트로 자동 전환하는 핵심 역할을 수행한다.

넷째, 멀티 클라우드 및 하이브리드 클라우드 환경이 확산되고 있다. 기업들이 AWS, Azure, GCP 등 복수의 클라우드 프로바이더를 동시에 활용하거나, 온프레미스와 퍼블릭 클라우드를 병행하는 하이브리드 아키텍처를 채택함에 따라, 이기종 인프라 간에 트래픽을 통합적으로 관리할 수 있는 GSLB의 중요성이 더욱 부각되고 있다.

다섯째, Kubernetes 멀티 클러스터 운영이 확산되고 있다. 컨테이너 오케스트레이션의 사실상 표준인 Kubernetes를 복수의 리전과 클라우드에 걸쳐 운영하는 기업이 증가하면서, Kubernetes 네이티브 방식으로 클러스터 간 트래픽을 분배하는 GSLB 솔루션에 대한 수요가 급격히 늘어나고 있다. K8GB와 같은 CNCF 프로젝트가 주목받는 이유가 바로 여기에 있다.

1.2 GSLB 핵심 동작 원리

1.2.1 DNS 기반 트래픽 라우팅 메커니즘

GSLB의 동작은 DNS 쿼리 처리 과정에 기반한다. 클라이언트가 특정 도메인에 접속하려 할 때, 다음과 같은 단계를 거쳐 최적의 데이터센터로 라우팅된다.

[GSLB DNS 기반 트래픽 라우팅 흐름]

1. 클라이언트가 example.com에 대한 DNS 쿼리를 발생시킨다

2. 재귀 DNS 리졸버가 GSLB가 설정된 권한 DNS 서버에 쿼리를 전달한다

3. GSLB 장비가 다음 요소들을 평가한다:
   - 각 데이터센터의 서버 헬스 상태 및 가용성
   - 현재 서버 부하 (연결 수, 패킷 레이트, 대역폭)

- 클라이언트의 지리적 위치
- 관리자가 설정한 라우팅 정책

4. 평가 결과에 따라 최적의 데이터센터 IP 주소를 DNS 응답으로 반환한다

5. 클라이언트는 반환된 IP로 직접 연결한다 (GSLB는 데이터 경로에서 벗어남)

이 과정에서 핵심적인 DNS 구성 요소들의 역할을 이해하는 것이 중요하다.

DNS 위임(NS 레코드): 기업의 권한 DNS 서버는 GSLB가 관리하는 서브도메인(예: app.gslb.example.com)에 대한 DNS 쿼리를 GSLB 장비의 ADNS(Authoritative DNS) 서비스로 위임한다. 이 위임은 NS(Name Server) 레코드를 통해 이루어지며, 상위 DNS 서버가 해당 서브도메인에 대한 쿼리를 받으면 GSLB의 ADNS로 쿼리를 전달한다.

권한 DNS(ADNS, Authoritative DNS): GSLB 장비 내에서 운영되는 ADNS 서비스는 위임받은 도메인에 대해 권한 있는 DNS 응답을 생성한다. 일반 DNS와 달리 GSLB의 ADNS는 정적인 레코드를 반환하는 것이 아니라, 실시간 헬스체크 결과와 라우팅 알고리즘을 적용하여 동적으로 최적의 IP를 선택하고 응답한다.

TTL(Time To Live) 캐싱의 영향: DNS 응답에 포함된 TTL 값은 해당 응답이 캐시에 유지되는 시간을 지정한다. GSLB 환경에서 TTL은 페일오버 속도에 직접적인 영향을 미친다. TTL이 짧으면(예: 30~60초) 장애 발생 시 빠르게 새로운 IP로 전환할 수 있지만, DNS 쿼리 부하가 증가한다. 반대로 TTL이 길면(예: 300초 이상) DNS 쿼리 부하는 감소하지만, 장애 발생 시 캐시된 구 IP로 계속 접속하는 클라이언트가 존재하여 페일오버가 지연된다.

650

[그림 5] 650

TTL 설정	페일오버 수렴 시간	DNS 쿼리 부하	사용 시나리오
5~30초	매우 빠름	매우 높음	미션 크리티컬 서비스, 즉각적 페일오버 필요 시
60초	빠름	높음	GSLB 일반 권장값, 페일오버 속도와 부하의 균형
300초 (5분)	보통	보통	CDN, 로드밸런싱 일반 용도

3,600초 (1시간)	느림	낮음	변경이 드문 정적 서비스
86,400초 (24시간)	매우 느림	매우 낮음	거의 변경되지 않는 레코드

주의할 점은 TTL 값을 아무리 낮게 설정하더라도 웹 브라우저의 자체 DNS 캐시(일반적으로 15분~6시간)로 인해 즉각적인 페일오버가 보장되지 않는다는 것이다. 이는 GSLB 기반 DR 설계 시 반드시 고려해야 하는 제약 사항이다.

1.2.2 주요 라우팅 알고리즘

GSLB는 다양한 라우팅 알고리즘을 제공하며, 관리자는 서비스의 특성과 요구사항에 따라 적절한 알고리즘을 선택할 수 있다. 다음은 주요 GSLB 라우팅 알고리즘의 비교표이다.

알고리즘	설명	적합 시나리오
Round Robin	각 사이트에 순차적으로 트래픽 배분	사이트 간 용량이 동일할 때
Weighted Round Robin	가중치 기반 비율 배분	사이트별 용량이 다를 때
Least Connection	현재 연결 수가 가장 적은 사이트로 배분	세션 길이가 불균일할 때
Geolocation/Proximity	클라이언트와 지리적으로 가장 가까운 사이트로 배분	글로벌 서비스, 지연 최소화
RTT (Round Trip Time)	네트워크 응답 시간이 가장 짧은 사이트로 배분	네트워크 품질 기반 최적화
Source IP Hash	Source IP 해시 기반 고정 배분	세션 지속성 필요 시

Round Robin은 가장 단순한 알고리즘으로, DNS 응답 시 각 사이트의 IP를 순차적으로 반환한다. 모든 사이트의 처리 용량이 동일하고 특별한 라우팅 요구사항이 없을 때 적합하다. 구현이 간단하고 트래픽이 균등하게 분산되는 장점이 있으나, 사이트 간 용량 차이나 클라이언트 위치를 고려하지 않는 한계가 있다.

Weighted Round Robin은 Round Robin에 가중치 개념을 추가한 것으로, 각 사이트에 처리 용량에 비례하는 가중치를 부여하여 트래픽을 비율 배분한다. 예를 들어 사이트 A의 가중치가 70이고 사이트 B의 가중치가 30이면, 전체 트래픽의 약 70%가 사이트 A로, 30%가 사이트 B로 배분된다. 카나리 배포(Canary Deployment)나 점진적 마이그레이션 시나리오에서도 유용하게 활용된다.

Least Connection은 MEP(Metrics Exchange Protocol) 등을 통해 수집한 각 사이트의 현재 활성 연결 수를 기반으로, 연결 수가 가장 적은 사이트로 트래픽을 배분한다. 세션의 길이가 불균일한 서비스(예: 파일 다운로드, 스트리밍)에서 효과적이다.

Geolocation/Proximity는 GeoIP 데이터베이스를 참조하여 클라이언트의 지리적 위치를 파악하고, 가장 가까운 사이트로 트래픽을 배분한다. 글로벌 서비스에서 사용자 경험(지연 시간)을 최적화하거나, 데이터 주권 규정(GDPR 등)에 따라 특정 지역의 트래픽을 해당 국가 내 데이터센터로 제한할 때 사용된다.

RTT(Round Trip Time) 기반 라우팅은 Geolocation보다 한 단계 발전된 방식으로, 실제 네트워크 왕복 시간을 측정하여 응답 시간이 가장 짧은 사이트로 트래픽을 배분한다. 네트워크 혼잡도를 반영하므로 더 정확한 성능 최적화가 가능하지만, 지속적인 RTT 모니터링에 따른 오버헤드가 존재한다.

Source IP Hash는 클라이언트의 소스 IP 주소에 해시 함수를 적용하여 항상 동일한 사이트로 배분하는 방식이다. 세션 퍼시스턴스(Session Persistence)가 필요한 서비스에서 활용되며, 동일 사용자가 항상 같은 사이트에 접속하도록 보장한다. 다만 NAT(Network Address Translation) 환경에서는 다수의 클라이언트가 동일한 공인 IP를 사용하므로 부하 편중이 발생할 수 있다.

1.2.3 헬스체크와 자동 페일오버

헬스체크(Health Check)는 GSLB가 관리하는 각 사이트 및 서비스의 가용 상태를 주기적으로 점검하는 메커니즘이다. 헬스체크 없이는 장애가 발생한 사이트로 트래픽을 계속 전송하게 되므로, 헬스체크는 GSLB의 지능적 라우팅을 가능하게 하는 핵심 구성 요소이다.

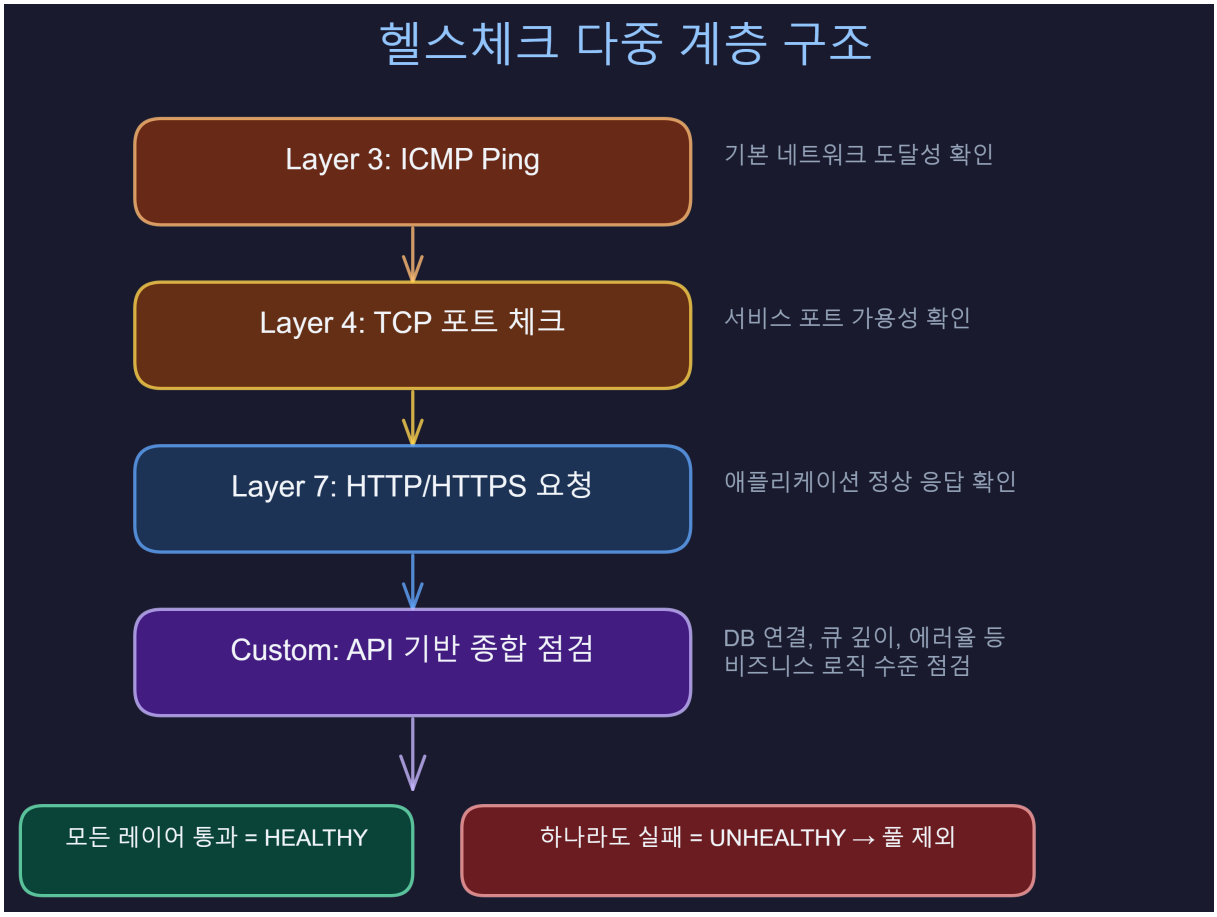
헬스체크 유형

GSLB에서 활용하는 헬스체크는 점검 수준에 따라 다음과 같이 분류된다.

유형	동작 방식	점검 수준	장점	단점
ICMP (Ping)	대상 서버에 ICMP Echo Request 전송, 응답 확인	네트워크 도달성	구현 간단, 오버헤드 최소	애플리케이션 장애 감지 불가
TCP	특정 포트에 TCP 3-Way Handshake 시도	포트 가용성	서비스 포트 장애 감지 가능	애플리케이션 로직 장애 감지 불가
HTTP/HTTPS	HTTP GET 요청 후 상태 코드 및 응답 본문 확인	애플리케이션 응답	웹 서비스 정상 동작 확인 가능	구현 복잡, 오버헤드 증가

Application-Level	API 호출, DB 연결 확인, 종합 점검 스크립트 수행	비즈니스 로직 수준	실질적 서비스 가용성 판단	커스텀 개발 필요, 복잡성 높음

실무에서는 이러한 헬스체크를 계층적으로 적용하는 것이 권장된다.



[그림 6] 672

핵심 파라미터

헬스체크의 민감도와 안정성은 다음 파라미터들의 조합으로 결정된다.

파라미터	설명	일반적 설정값	설정 고려사항
Interval	헬스체크 수행 주기	10~30초	짧으면 빠른 감지, 길면 부하 감소
Timeout	단일 헬스체크의 응답 대기 시간	5~10초	네트워크 지연을 감안하여 설정
Threshold (Failure)	장애 판정을 위한 연속 실패 횟수	3회	높으면 오탐 감소, 낮으면 빠른 감지
Recovery Threshold	복구 판정을 위한 연속 성공 횟수	3~5회	플래핑(Flapping) 방지를 위해 Failure Threshold보다 높게 설정

자동 페일오버 프로세스

헬스체크 결과에 기반한 자동 페일오버는 다음 4단계로 이루어진다.

[그림 7] 641

이 프로세스에서 주의할 점은, 장애 감지부터 실제 모든 클라이언트의 트래픽 전환이 완료되기까지의 총 소요 시간이 단순히 헬스체크 파라미터만으로 결정되지 않는다는 것이다. DNS TTL 캐싱, 브라우저 자체 캐싱, 중간 재귀 리졸버의 캐싱 등 여러 계층의 캐시가 영향을 미치므로, GSLB 기반 DR 설계 시에는 이러한 캐싱 계층별 영향을 종합적으로 고려해야 한다.

[K8GB 맥락]

K8GB는 Kubernetes 네이티브 GSLB 솔루션으로서, 상용 GSLB 장비와는 차별화된 방식으로 헬스체크를 수행한다. 전통적인 GSLB가 별도의 헬스체크 모니터를 통해 서비스 상태를 점검하는 반면, K8GB는 **Kubernetes의 Liveness/Readiness**

Probe를 GSLB 헬스체크에 직접 활용한다. K8GB Controller는 각 클러스터에서 실행 중인 Pod의 Readiness Probe 상태를 지속적으로 모니터링하며, Pod가 Ready 상태가 아닌 경우 해당 클러스터를 GSLB 라우팅 풀에서 자동으로 제외한다. 이 접근 방식은 별도의 헬스체크 인프라 구축 없이 Kubernetes가 이미 제공하는 헬스체크 메커니즘을 재활용함으로써 운영 복잡성을 크게 줄이는 장점이 있다. 또한 애플리케이션 개발자가 정의한 세밀한 Readiness 조건(DB 연결 상태, 캐시 워밍 완료 여부 등)이 그대로 GSLB 라우팅 결정에 반영되므로, 애플리케이션 수준의 정밀한 트래픽 관리가 가능하다.

1장에서는 GSLB의 기본 개념, DNS 기반 라우팅 원리, 라우팅 알고리즘, 헬스체크 메커니즘을 살펴보았다. GSLB가 DNS 레벨에서 동작하며 데이터 경로에 관여하지 않는다는 특성은 GSLB의 확장성과 유연성의 근간이 되지만, 동시에 DNS 캐싱으로 인한 페일오버 지연이라는 고유한 제약도 수반한다. 2장에서는 이러한 GSLB가 실제 운영 환경에서 어떤 기술들과 어떻게 결합하여 동작하는지, 그 기술 생태계를 상세히 살펴본다.

2장. GSLB 관련 기술 생태계

GSLB는 단독으로 동작하는 기술이 아니다. DNS, CDN, BGP Anycast, GeoIP, API Gateway, Reverse Proxy 등 다양한 기술과 유기적으로 결합하여 글로벌 트래픽 관리 아키텍처를 구성한다. 본 장에서는 GSLB의 기반 기술과 연동 기술, 그리고 관련 표준 및 프로토콜을 체계적으로 정리하여 GSLB를 둘러싼 기술 생태계의 전체 그림을 제시한다.

2.1 GSLB의 기반 기술

2.1.1 DNS — GSLB의 핵심 기반

DNS(Domain Name System)는 도메인 이름을 IP 주소로 변환하는 인터넷의 근간 시스템이며, GSLB는 이 DNS 해석(Resolution) 과정에 개입하여 트래픽을 분배한다. DNS가 GSLB의 핵심 기반인 이유는 다음 세 가지로 정리할 수 있다.

DNS 레벨 라우팅: GSLB는 DNS 응답을 조작하여 클라이언트가 연결할 서버의 IP 주소를 결정한다. 실제 애플리케이션 트래픽이 전송되기 전, DNS 단계에서 라우팅이 결정되므로 사용자는 처음부터 최적의 서버에 연결된다. 이러한 DNS 레벨 라우팅은 기존 DNS 인프라를 활용하므로 별도의 네트워크 장비 변경 없이 구축할 수 있다는 비용 효율성을 제공한다.

TTL(Time To Live) 관리: GSLB는 DNS 레코드의 TTL 값을 전략적으로 관리하여 장애 발생 시 빠른 페일오버를 가능하게 한다. 1장에서 살펴본 바와 같이, TTL이 짧을수록 변경 사항이 빠르게 전파되지만 DNS 쿼리 부하가 증가하는 트레이드오프가 존재한다. GSLB 운영자는 서비스의 가용성 요구수준에 따라 적절한 TTL 전략을 수립해야 한다.

GSLB의 DNS 프록시 역할: GSLB는 DNS 프록시로 동작하며, 일반 DNS와 달리 실시간 로드밸런싱 알고리즘을 적용하여 응답을 생성한다. 정적인 DNS 레코드가 아닌, 현재의 사이트 상태와 라우팅 정책에 기반한 동적인 DNS 응답을 반환하는 것이 GSLB DNS의 핵심 특성이다.

DNS 기반 GSLB의 한계점

DNS가 GSLB의 핵심 기반이지만, 다음과 같은 고유한 한계점도 존재한다.

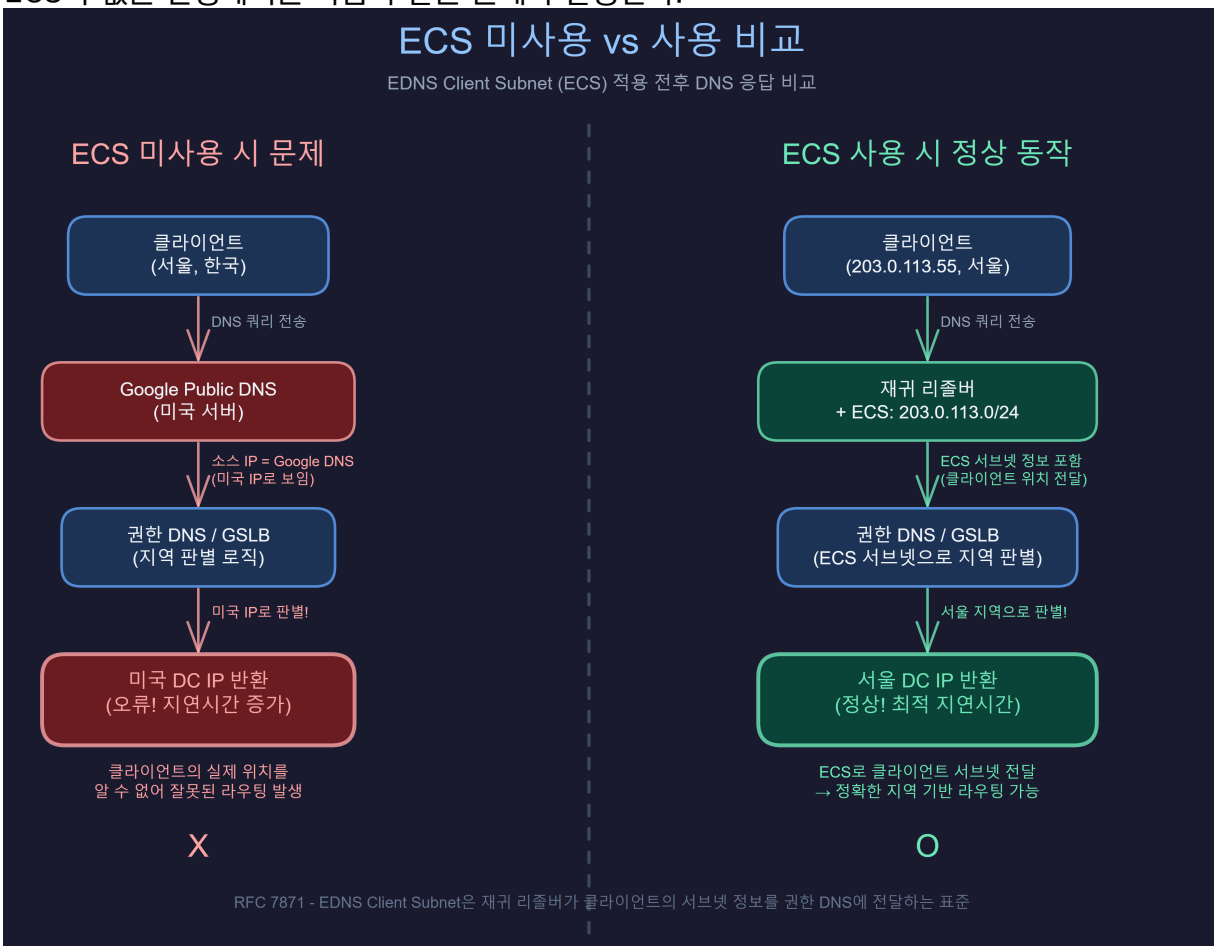
한계점	설명	영향
DNS 캐싱 전파 지연	DNS 캐싱으로 인해 페일오버 수렴 시간이 TTL에 종속됨	즉각적 페일오버 불가, 실질적 최소 20~60초
LDNS IP 기반 판단	클라이언트의 실제 IP가 아닌 LDNS(Local DNS) IP 기반으로 위치 판단	퍼블릭 DNS(8.8.8.8 등) 사용 시 위치 오판
EDNS 미지원 시 정확도 저하	EDNS Client Subnet 미지원 환경에서는 클라이언트 위치 파악 불가	지리 기반 라우팅 정확도 저하
Stateless 한계	DNS 쿼리 시점에만 개입, 연결 이후 세션 상태 관리 불가	세션 유지를 위한 추가 메커니즘 필요
브라우저 캐싱	대부분의 브라우저가 TTL 값을 무시하고 자체 캐싱 (15분~6시간)	TTL 설정과 무관한 페일오버 지연 발생

이러한 한계점을 보완하기 위해 EDNS Client Subnet(ECS), Anycast 병행 사용, 짧은 TTL 전략 등 다양한 기술적 보완 방안이 활용되며, 이에 대해서는 본 장의 후반부에서 상세히 다룬다.

EDNS Client Subnet (RFC 7871) 동작 방식

기존 DNS 기반 GSLB의 핵심 한계 중 하나인 “클라이언트 위치 부정확성” 문제를 해결하기 위해 RFC 7871에서 정의된 것이 EDNS Client Subnet(ECS)이다. ECS는 DNS 확장 메커니즘(EDNS0)의 옵션으로, 재귀 리졸버가 클라이언트의 서브넷 정보를 DNS 쿼리에 포함시켜 권한 DNS 서버에 전달하는 방법을 규정한다.

ECS가 없는 환경에서는 다음과 같은 문제가 발생한다.



서울에 있는 클라이언트가 Google Public DNS(8.8.8.8)를 사용하면, GSLB는 쿼리의 소스 IP인 Google DNS 서버(미국)의 위치를 기준으로 판단하여 미국 데이터센터의 IP를 반환할 수 있다.

ECS는 이 문제를 다음과 같이 해결한다.

재귀 리졸버가 클라이언트 IP의 일부를 마스킹(프라이버시 보호)하여 ECS 옵션에 포함시키면, 권한 DNS 서버(GSLB)는 리졸버 IP가 아닌 실제 클라이언트의 서브넷을 기준으로 최적 서버를 판단할 수 있다.

프라이버시 고려사항: ECS는 클라이언트의 전체 IP 주소가 아닌 서브넷 정보만 전달한다(일반적으로 IPv4는 /24, IPv6는 /56). 이를 통해 개별 사용자 식별 없이 대략적인 위치 정보만 전달하여 프라이버시를 보호하지만, 일부 프라이버시 옹호론자들은 서브넷 정보도 사용자 추적에 활용될 수 있다는 우려를 제기하고 있다.

2.1.2 BGP Anycast와 DNS 기반 GSLB 비교

BGP Anycast는 네트워크 계층(L3)에서 동작하는 라우팅 기술로, 지리적으로 분산된 여러 서버가 **동일한 IP 주소를 공유**하고 BGP 라우팅 프로토콜을 통해 해당 IP를 각자 광고(advertise)한다. 사용자가 Anycast IP로 요청을 보내면, BGP 라우팅 테이블에 따라 네트워크 토폴로지 기준으로 가장 "가까운" 서버로 요청이 전달된다.

GSLB(DNS 기반)와 BGP Anycast는 모두 글로벌 트래픽 분산을 목적으로 하지만, 동작 방식과 특성이 크게 다르다.

비교 항목	BGP Anycast	DNS 기반 GSLB
동작 계층	L3 네트워크 계층 (BGP 라우팅)	L7 DNS 계층
라우팅 기준	네트워크 토폴로지 (AS-path, IGP metric)	관리자 정책 (지리, 부하, RTT, 가용성)
IP 관리	모든 노드가 동일 IP 공유	각 노드가 고유 IP 보유, DNS가 선택
Failover 속도	BGP 수렴 시간에 의존 (수초~수분)	DNS TTL에 의존 (수십 초~수분)
세밀한 제어	제한적 (BGP 경로 기반만 가능)	다양한 정책 기반 세밀한 제어 가능
운영 복잡도	높음 (BGP 피어링, 트래픽 엔지니어링 필요)	상대적으로 낮음
적합 워크로드	DNS, CDN 등 Stateless 서비스	웹 애플리케이션, API 등 Stateful 서비스 포함

GSLB + Anycast 조합 아키텍처

실무에서는 GSLB와 Anycast를 대립적으로 선택하기보다, 두 기술을 조합하여 시너지를 극대화하는 것이 일반적이다. 대표적인 조합 패턴은 다음과 같다.

- **GSLB DNS 서버를 Anycast로 배포**: 많은 CDN 사업자는 GSLB DNS 서버 자체를 Any-

cast로 배포한다. 이렇게 하면 DNS 쿼리 자체가 네트워크상 가장 가까운 GSLB 노드에서 처리되어 DNS 응답 지연을 최소화하고, GSLB 인프라 자체의 고가용성과 DDoS 방어력을 확보할 수 있다.

- **계층적 구조**: Anycast(L3)로 가장 가까운 PoP(Point of Presence)에 도달한 후, 해당 PoP 내에서 GSLB(DNS 기반)가 최적의 백엔드 서버를 결정하는 계층적 구조가 널리 사용된다.
- **Anycast의 DDoS 방어 이점**: Anycast는 공격 트래픽을 여러 노드로 자동 분산시키므로 DDoS 완화에 효과적이다. GSLB 인프라를 Anycast로 구성하면 GSLB 자체의 가용성이 크게 향상된다.

2.2 GSLB 연동 기술

2.2.1 CDN과 GSLB의 상호 보완 관계

CDN(Content Delivery Network)은 전 세계에 분산 배치된 엣지(Edge) 서버 네트워크를 통해 사용자에게 콘텐츠를 가장 가까운 위치에서 전달하는 기술이다. GSLB와 CDN은 다음과 같은 상호 보완적 관계를 형성한다.

GSLB는 CDN의 핵심 라우팅 메커니즘이다. CDN은 GSLB 기술을 사용하여 사용자를 최적의 엣지 서버로 라우팅한다. 사용자의 DNS 요청이 CDN의 GSLB 시스템으로 전달되면, GSLB가 지리적 위치, 서버 부하, 네트워크 성능 등을 종합 평가하여 최적의 엣지 서버 IP를 반환한다. 다시 말해, GSLB 없는 CDN은 사용자를 올바른 엣지 서버로 유도할 수 없다.

CDN 사업자가 자체 GSLB를 운영하는 방식: Cloudflare, Akamai, AWS CloudFront 등 주요 CDN 사업자는 자체 GSLB 시스템을 운영하며, 이를 통해 전 세계 PoP(Point of Presence)에 트래픽을 분배한다. 엔터프라이즈 고객은 CDN 사업자의 GSLB를 활용하거나, 자체 GSLB 계층을 CDN 앞단에 배치하여 멀티 CDN 전략을 구현하기도 한다.

이 구조에서 GSLB는 사용자의 DNS 쿼리를 수신하고, 지리적 위치와 서버 부하 등을 평가하여 최적의 CDN 엣지 서버를 선택한다. 선택된 엣지 서버가 요청된 콘텐츠를 캐시에 보유하고 있으면 즉시 반환하고, 캐시에 없으면 오리진 서버에서 콘텐츠를 가져와 사용자에게 전달하고 캐시에 저장한다.

2.2.2 GeoIP, API Gateway, Reverse Proxy

GSLB 생태계에는 CDN 외에도 GeoIP 데이터베이스, API Gateway, Reverse Proxy 등 다양한 기술이 연동되어 계층적인 트래픽 관리 아키텍처를 구성한다.

GeoIP 데이터 소스와 한계

GeoIP는 IP 주소를 지리적 위치(국가, 지역, 도시, 위도/경도 등)에 매핑하는 데이터베이스 및 기술이다. GSLB는 GeoIP 데이터를 활용하여 사용자를 지리적으로 가장 가까운 서버로 라우팅 한다.

주요 GeoIP 데이터 소스로는 **MaxMind GeoIP2/GeoLite2**(가장 널리 사용되는 상용/무료 GeoIP 데이터베이스), **IP2Location**(대안적 GeoIP 데이터 제공자), **RIR(Regional Internet Registry) 데이터**(ARIN, RIPE, APNIC 등에서 제공하는 IP 할당 정보) 등이 있다.

그러나 GeoIP는 다음과 같은 한계를 가진다.

- VPN 또는 프록시 사용 시 IP 주소와 실제 사용자 위치가 불일치
- 모바일 사용자의 IP 위치 정확도 문제 (통신사 게이트웨이 위치로 매핑)
- GeoIP 데이터베이스의 주기적 업데이트 필요성
- LDNS의 IP가 사용자 위치와 다를 수 있음 (EDNS Client Subnet으로 일부 해결)

API Gateway와 GSLB 역할 분담

API Gateway는 API 요청의 단일 진입점(Single Entry Point)으로, 인증/인가, 속도 제한(Rate Limiting), 프로토콜 변환, 요청 라우팅 등의 기능을 제공하는 미들웨어이다. GSLB와 API Gateway는 각각 다른 계층에서 트래픽을 관리하며, 다음과 같이 역할을 분담한다.

기능	GSLB	API Gateway
트래픽 분배 범위	글로벌 (데이터센터 간)	로컬 (데이터센터 내 서비스 간)
라우팅 기준	지리적 위치, 서버 상태, 지연 시간	API 경로, 버전, 헤더 등
인증/인가	해당 없음	JWT 검증, OAuth, API Key 등
속도 제한	해당 없음	클라이언트별/API별 Rate Limiting
프로토콜 처리	DNS 레벨	HTTP, gRPC, WebSocket 등
장애 처리	사이트 단위 페일오버	서비스 단위 Circuit Breaker

글로벌 API 배포 환경에서는 GSLB가 사용자를 가장 가까운 API Gateway로 라우팅하고, API Gateway가 해당 지역의 마이크로서비스로 요청을 전달하는 계층적 구조를 형성한다.

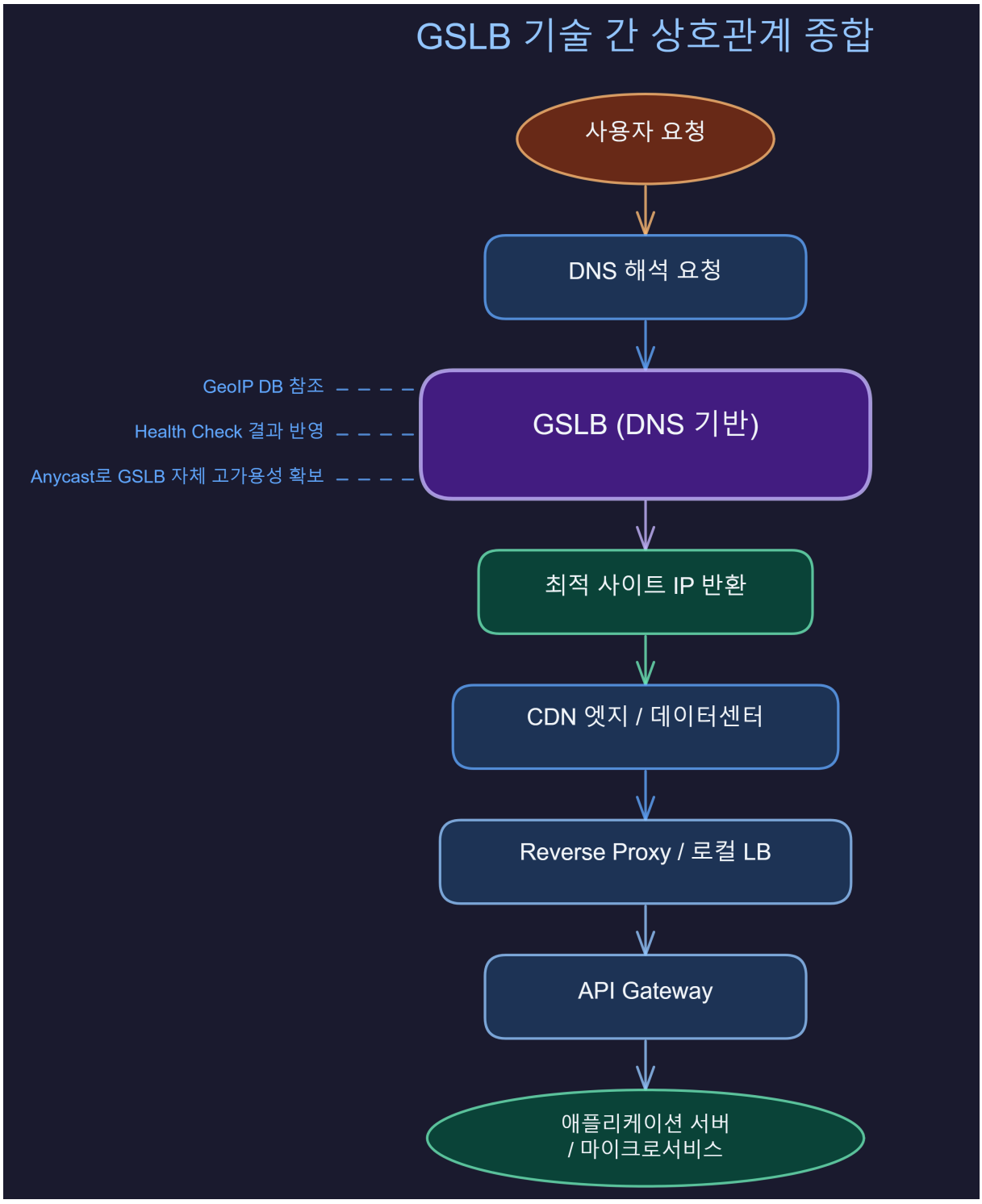
Reverse Proxy와 GSLB의 계층적 구조

Reverse Proxy(역방향 프록시)는 클라이언트와 백엔드 서버 사이에 위치하여 클라이언트의 요청을 대리 수신하고 적절한 백엔드 서버로 전달하는 서버이다. Nginx, HAProxy, Envoy 등이 대표적인 Reverse Proxy 솔루션이며, 로컬 로드밸런싱, SSL 종료, 캐싱, 압축 등의 기능을 제공한다.

GSLB와 Reverse Proxy는 “어느 사이트로 갈 것인가(GSLB)”와 “사이트 내 어느 서버로 갈 것인가(Reverse Proxy)”를 각각 결정하는 계층적 역할을 수행한다.

기술 간 상호관계 종합 다이어그램

다음 다이어그램은 GSLB를 중심으로 한 관련 기술들의 계층적 구조와 상호관계를 종합적으로 보여준다.



[그림 8] 475

이 계층 구조에서 각 기술의 역할을 요약하면 다음과 같다.

기술	GSLB에 대한 역할	관계 유형
DNS	GSLB의 동작 기반 프로토콜	기반 기술 (필수)

CDN	GSLB를 라우팅 엔진으로 활용하는 인프라	상위 소비자
Anycast	GSLB 인프라 자체의 고가용성 제공	보완 기술
Health Check	GSLB 라우팅 결정의 입력 데이터 제공	핵심 구성 요소
GeoIP	지리 기반 라우팅 판단의 데이터 소스	데이터 제공자
API Gateway	GSLB 하위에서 API 레벨 트래픽 관리 수행	하위 계층
Reverse Proxy	GSLB 하위에서 로컬 트래픽 분배 수행	하위 계층

2.3 GSLB 관련 표준 및 프로토콜

2.3.1 MEP (Metrics Exchange Protocol)

MEP(Metrics Exchange Protocol)는 **Citrix NetScaler(현 NetScaler ADC) 전용 독점 프로토콜**로, GSLB 구성에 참여하는 사이트 간에 메트릭 정보를 교환하는 데 사용된다. GSLB가 지능적인 트래픽 분산 결정을 내리기 위해서는 각 사이트의 실시간 상태 정보가 필요한데, MEP가 이 정보 교환을 담당한다.

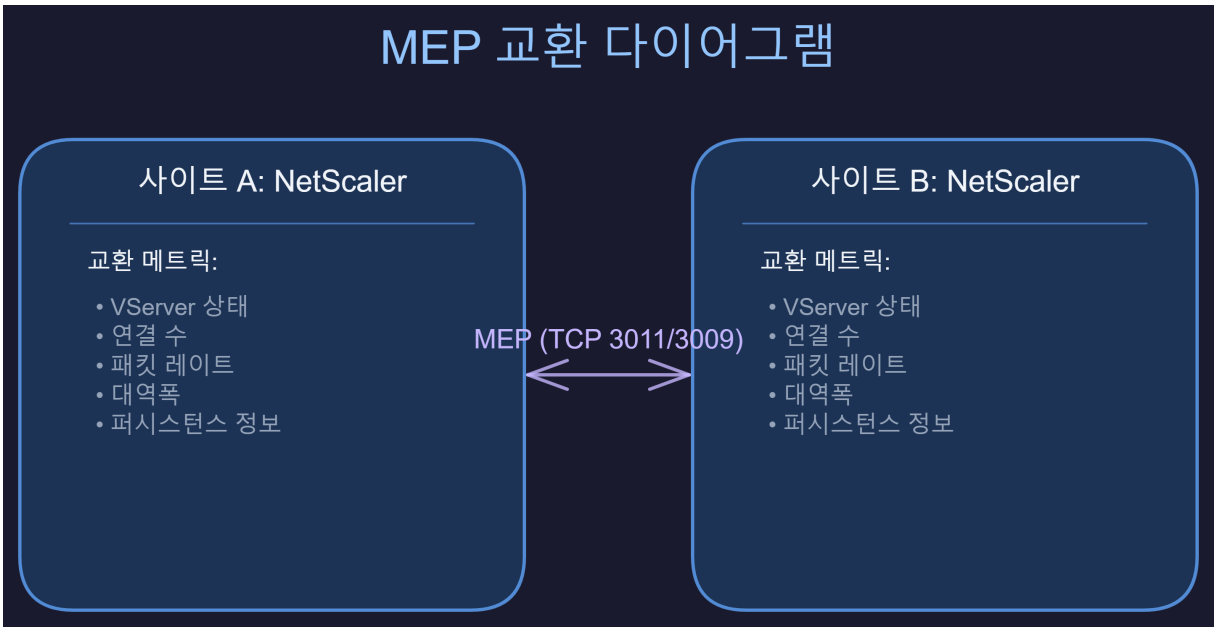
교환 메트릭 정보

MEP를 통해 교환되는 주요 정보는 세 가지 범주로 나뉜다.

범주	교환 정보
사이트 메트릭 (Site Metrics)	각 로드밸런싱/콘텐츠 스위칭 가상 서버의 상태, 현재 연결 수, 현재 패킷 레이트, 대역폭 사용량
네트워크 메트릭 (Network Metrics)	RTT(Round Trip Time), 네트워크 지연 시간 등 사이트 간 네트워크 성능 지표
퍼시스턴스 정보 (Persistence)	세션 지속성(persistence) 정보로 동일 클라이언트가 동일 사이트로 연결되도록 보장

통신 방식 및 토폴로지

MEP는 TCP 기반으로 동작하며, 포트 3011(비암호화 통신)과 3009(암호화 통신)를 사용한다. 기본적으로 메시(Mesh) 구조로 모든 사이트가 상호 통신하며, 대규모 배포 시에는 Parent-Child 계층형 구조를 사용한다.



토폴로지	최대 사이트 수	설명
Mesh (Active-Active)	최대 32개 사이트	모든 사이트가 상호 MEP 통신. 사이트 증가 시 MEP 트래픽이 기하급수적으로 증가
Parent-Child	부모 32개 + 자식 1,024개 (최대 1,056개)	대규모 배포를 위한 계층형 구조. 자식 사이트는 부모 사이트를 통해 메트릭 교환

멀티벤더 호환 불가 한계

MEP의 가장 큰 한계는 NetScaler 전용 독점 프로토콜이라는 점이다. F5 BIG-IP, A10 Networks 등 타 벤더 장비와의 상호운용이 불가능하므로, 멀티벤더 환경에서는 DNS 기반의 표준 방식이나 각 벤더별 API를 활용한 통합이 필요하다. 이러한 벤더 종속성은 기업이 GSLB 장비를 선택할 때 반드시 고려해야 하는 제약 사항이다.

2.3.2 EDNS Client Subnet (RFC 7871)

2.1.1절에서 EDNS Client Subnet(ECS)의 기본 개념과 동작 방식을 소개하였다. 본 절에서는 ECS의 기술적 세부 사항을 보다 심층적으로 다룬다.

배경: 재귀 DNS 리졸버 IP의 한계

재귀 DNS 리졸버 IP는 실제 클라이언트의 위치와 다를 수 있다. 특히 Google Public DNS(8.8.8.8), Cloudflare DNS(1.1.1.1) 등 글로벌 퍼블릭 DNS를 사용하는 클라이언트의 경우, 재귀 리졸버가 전 세계 여러 위치에 분산 배치되어 있으므로 클라이언트와 리졸버의 지리적

위치가 일치하지 않을 수 있다.

ECS 동작 흐름

[ECS 동작 흐름 상세]

[Step 1] 클라이언트(203.0.113.55)가 재귀 리졸버에 DNS 쿼리 전송

[Step 2] 재귀 리졸버가 클라이언트 IP의 일부를 마스킹하여 ECS 옵션에 포함
예: 203.0.113.0/24 (마지막 옥텟을 마스킹하여 프라이버시 보호)

[Step 3] 권한 DNS 서버(GSLB)가 ECS 정보를 수신

[Step 4] 리졸버 IP가 아닌 클라이언트 서브넷 203.0.113.0/24 기준으로
최적 서버를 판단

[Step 5] 응답에 SCOPE PREFIX-LENGTH를 포함하여 캐시 범위를 지정

ECS 옵션 필드 구조

필드	설명
FAMILY	주소 패밀리 (1 = IPv4, 2 = IPv6)
SOURCE PREFIX-LENGTH	클라이언트 IP에서 전달되는 비트 수 (프라이버시와 정확도의 균형). IPv4 일반적 /24, IPv6 일반적 /56
SCOPE PREFIX-LENGTH	응답이 유효한 범위 (권한 서버가 설정). 리졸버가 이 범위 내에서 응답을 캐시하고 재사용
ADDRESS	마스킹된 클라이언트 서브넷 주소

SOURCE PREFIX-LENGTH는 프라이버시와 위치 정확도 사이의 균형을 결정하는 핵심 파라미터이다. 값이 클수록 더 정확한 위치를 전달하지만 프라이버시 노출 위험이 증가하고, 값이 작을수록 프라이버시는 보호되지만 위치 정확도가 저하된다.

프라이버시 고려사항

- ECS는 클라이언트의 전체 IP 주소가 아닌 서브넷 정보만 전달한다
- 일반적으로 IPv4는 /24(256개 IP 범위), IPv6는 /56으로 마스킹한다
- 개별 사용자 식별 없이 대략적인 위치 정보만 전달하여 프라이버시를 보호한다
- 그러나 서브넷 정보와 다른 데이터를 결합하면 사용자 추적이 가능할 수 있다는 우려가 존재한다
- 프라이버시를 우선시하는 일부 리졸버(예: Cloudflare 1.1.1.1)는 ECS를 기본적으로 비활성화하고 있다

지원 현황

구분	지원 여부
퍼블릭 DNS	Google Public DNS, OpenDNS 등 주요 퍼블릭 DNS가 ECS 지원
GSLB 장비	NetScaler, F5 BIG-IP DNS 등 주요 벤더 지원
CDN	Cloudflare, Akamai, AWS CloudFront 등 주요 CDN 지원
ISP DNS	ISP마다 지원 여부가 다르며, 미지원 ISP도 다수 존재

[K8GB 맥락]

K8GB는 상용 GSLB 장비와는 근본적으로 다른 방식으로 DNS 프로토콜을 처리한다. 상용 장비가 자체적인 독점 DNS 엔진을 사용하는 것과 달리, K8GB는 **CoreDNS를 내장하여 DNS 프로토콜을 직접 처리**한다. CoreDNS는 CNCF Graduated 프로젝트이자 Kubernetes의 기본 클러스터 DNS로 채택된 오픈소스 DNS 서버로, 풍부한 플러그인 생태계와 검증된 안정성을 제공한다.

더욱 주목할 점은, K8GB가 상용 GSLB의 MEP(Metrics Exchange Protocol)와 같은 독점 프로토콜 대신 **DNS 레코드 기반으로 클러스터 간 상태를 공유**한다는 것이다. 각 클러스터의 K8GB Controller는 자신이 관리하는 서비스의 상태(Pod Readiness 기반)를 DNS TXT/A 레코드로 공개하며, 다른 클러스터의 K8GB는 이 DNS 레코드를 조회하여 원격 클러스터의 상태를 파악한다. 이 접근 방식은 MEP의 멀티벤더 호환 불가 한계를 근본적으로 해결하며, DNS라는 표준 프로토콜만으로 클러스터 간 통신이 가능하므로 방화벽 설정이 단순하고 네트워크 토폴로지에 대한 제약이 최소화

된다. 또한 ExternalDNS를 통해 Route 53, Cloud DNS, Infoblox 등 다양한 DNS 프로바이더와 연동할 수 있어, 특정 인프라에 종속되지 않는 유연한 구성이 가능하다.

2장에서는 GSLB의 기반 기술인 DNS와 BGP Anycast, 연동 기술인 CDN, GeoIP, API Gateway, Reverse Proxy, 그리고 관련 표준 및 프로토콜인 MEP와 EDNS Client Subnet을 살펴보았다. GSLB는 이러한 기술들과 유기적으로 결합하여 글로벌 트래픽 관리 아키텍처를 구성하며, 각 기술의 역할과 한계를 이해하는 것은 효과적인 GSLB 설계의 전제 조건이다. 3장에서는 이러한 기술적 이해를 바탕으로, 실제 엔터프라이즈 환경에서 활용되는 GSLB 아키텍처 패턴을 살펴본다.

3장. GSLB 아키텍처 패턴

앞서 1장에서 GSLB의 기본 동작 원리를, 2장에서 GSLB 관련 기술 생태계를 살펴보았다. 본 장에서는 이러한 기술적 이해를 기반으로, 실제 엔터프라이즈 환경에서 구현되는 대표적인 GSLB 아키텍처 패턴을 다룬다. 멀티사이트 레퍼런스 아키텍처부터 하이브리드 클라우드, 멀티 클라우드 환경에 이르기까지, 각 패턴의 구성 방식과 설계 고려사항을 아키텍처 다이어그램과 함께 상세히 제시한다.

3.1 멀티사이트 레퍼런스 아키텍처

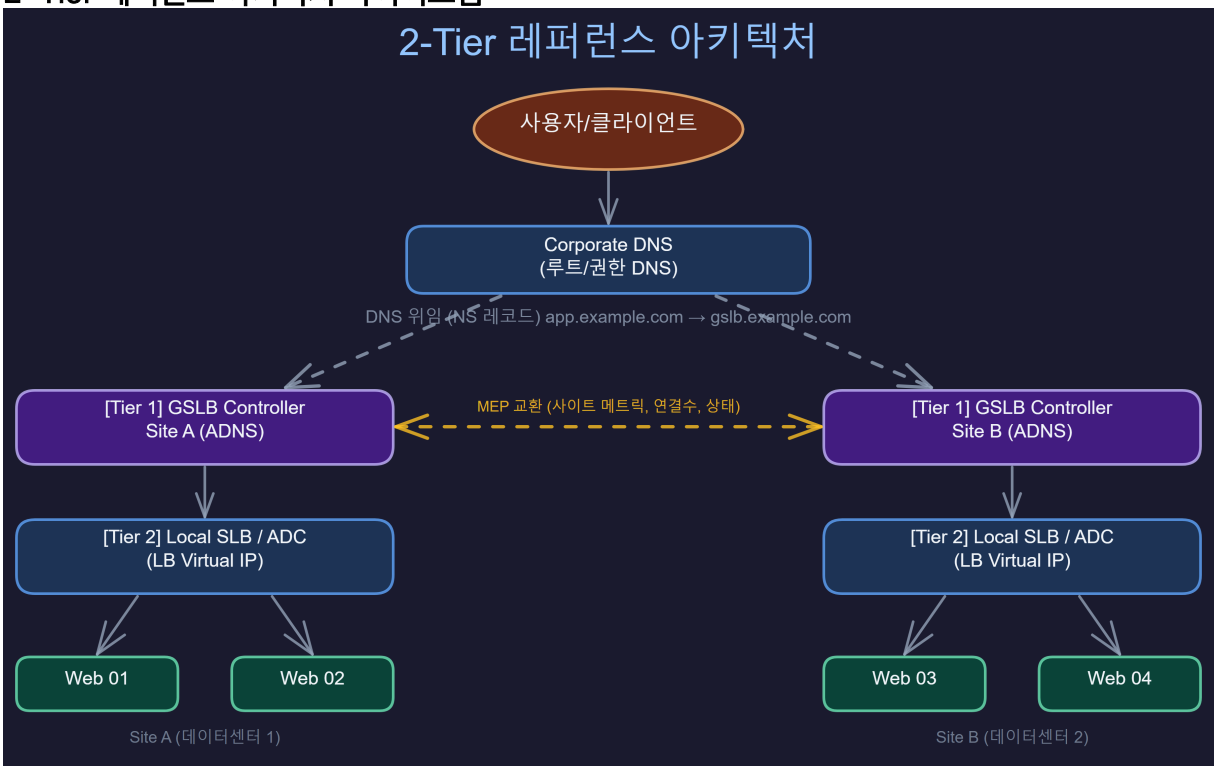
3.1.1 2-Tier 계층 구조 (Global DNS Tier + Local SLB Tier)

GSLB의 기본 레퍼런스 아키텍처는 **글로벌 DNS 티어(Tier 1)**와 **로컬 SLB 티어(Tier 2)**로 구성되는 2-Tier 계층 구조이다. 이 구조에서 각 계층의 역할과 구성 요소는 다음과 같다.

계층	역할	구성 요소
Tier 1: 글로벌 DNS 티어	사이트 간 트래픽 분배 결정	GSLB Controller, ADNS 서비스, MEP
Tier 2: 로컬 SLB 티어	사이트 내부 서버 간 로드밸런싱	Local LB VIP, 서버 팜, 헬스체크

Tier 1(글로벌 DNS 티어)에서는 GSLB Controller가 각 사이트의 상태를 MEP(또는 DNS 기반 상태 교환)를 통해 파악하고, ADNS(Authoritative DNS) 서비스가 클라이언트의 DNS 쿼리에 대해 최적의 사이트 VIP(Virtual IP)를 응답한다. Tier 2(로컬 SLB 티어)에서는 각 사이트의 로컬 로드밸런서가 Tier 1에서 선택된 사이트로 도착한 트래픽을 내부 서버 풀에 분배한다.

2-Tier 레퍼런스 아키텍처 다이어그램



트래픽 흐름 상세

DNS 요청부터 실제 애플리케이션 트래픽까지의 흐름은 다음과 같다.

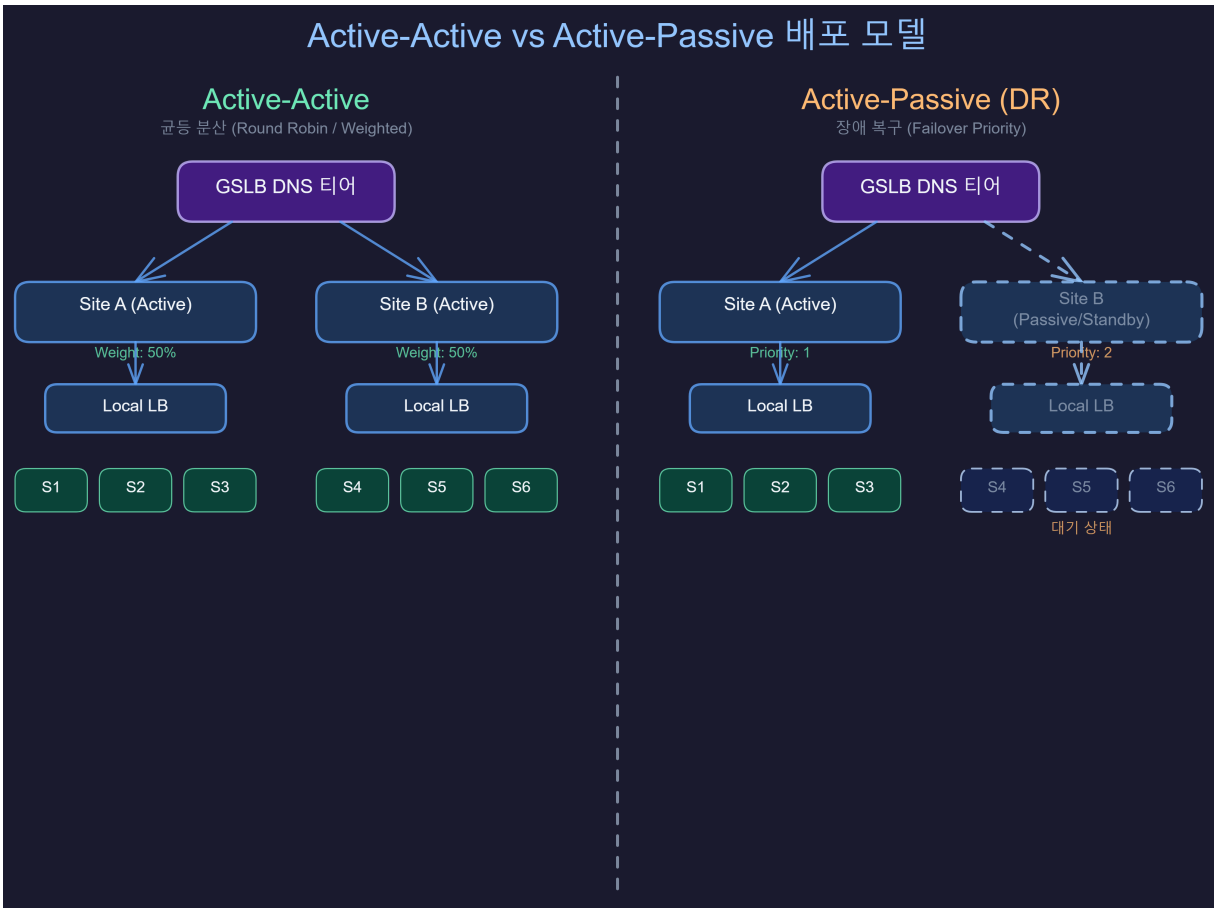
이 흐름에서 주목할 점은 GSLB가 (3)번 단계에서만 개입하며, 실제 애플리케이션 트래픽((5)~(6) 단계)은 GSLB를 경유하지 않고 직접 사이트의 로컬 LB를 통해 처리된다는 것이다. 이는 GSLB가 데이터 경로(Data Path)에 위치하지 않는 DNS 기반 방식의 핵심 특성이다.

3.1.2 Active-Active / Active-Passive 배포 모델

GSLB의 2-Tier 아키텍처는 Active-Active 또는 Active-Passive 배포 모델로 구현될 수 있다. 각 모델의 특성과 적합 시나리오는 서비스의 가용성 요구수준, 인프라 비용, 데이터 일관성 요구사항에 따라 달라진다.

Active-Active 배포

Active-Active 모델에서는 모든 사이트가 동시에 트래픽을 처리한다. 가중치(Weight) 기반으로 트래픽을 비율 배분하며, MEP를 통해 실시간으로 사이트 간 메트릭을 교환한다.



Active-Active 모델의 장점은 다음과 같다.

- 모든 인프라 자원을 상시 활용하므로 투자 효율성이 높음
- 사이트 간 부하를 분산하여 단일 사이트에 대한 과부하를 방지
- 한 사이트의 장애 시 나머지 사이트가 즉시 전체 트래픽을 흡수 (별도의 스탠바이 기동 시간 불필요)
- 가중치 조정으로 카나리 배포, 블루/그린 배포 등 배포 전략에 활용 가능

Active-Active 모델의 고려사항은 다음과 같다.

- 양방향 데이터 복제로 인한 Write Conflict, Split-Brain 문제 해결 필요
- 모든 사이트에서 동일한 서비스 수준을 유지해야 하므로 운영 복잡성 증가
- 데이터 일관성 보장을 위한 추가적인 아키텍처 설계 필요

Active-Passive (DR) 배포

Active-Passive 모델에서는 Primary 사이트가 모든 트래픽을 처리하고, Passive(Standby) 사이트는 Primary 사이트의 장애 시에만 활성화된다.

Active-Passive 모델의 장점은 다음과 같다.

- 데이터 일관성 관리가 상대적으로 단순 (단방향 복제)
- Write Conflict, Split-Brain 문제가 발생하지 않음
- DR 시나리오에 특화된 명확한 역할 분담

Active-Passive 모델의 고려사항은 다음과 같다.

- Passive 사이트가 평상시 유휴 상태이므로 인프라 투자 효율성이 낮음
- 페일오버 시 Passive 사이트의 워밍업(캐시, 연결 풀 등) 시간 필요
- DNS TTL 캐싱에 의한 페일오버 지연 발생 가능

DNS 위임 구성 방식

Active-Active 또는 Active-Passive 모델 모두에서 기업 DNS에서 GSLB로의 위임은 다음과 같은 NS 레코드와 CNAME 레코드를 통해 구성한다.

[외부 DNS 존 파일 구성 예시]

```
;; A 레코드 - 각 GSLB ADNS의 공인 IP
gslb-site-a    IN  A   203.0.113.10
gslb-site-b    IN  A   198.51.100.10

;; NS 레코드 - 서브도메인을 GSLB로 위임
gslb           IN  NS  gslb-site-a.example.com.
gslb           IN  NS  gslb-site-b.example.com.

;; CNAME - 실제 서비스 도메인을 GSLB 서브존으로 연결
app            IN  CNAME  app.gslb.example.com.
www            IN  CNAME  www.gslb.example.com.
```

이 구성에서 app.example.com에 대한 DNS 쿼리는 CNAME을 통해 app.gslb.example.com으로 리디렉션되고, NS 위임에 따라 GSLB ADNS(gslb-site-a 또는 gslb-site-b)가 쿼리

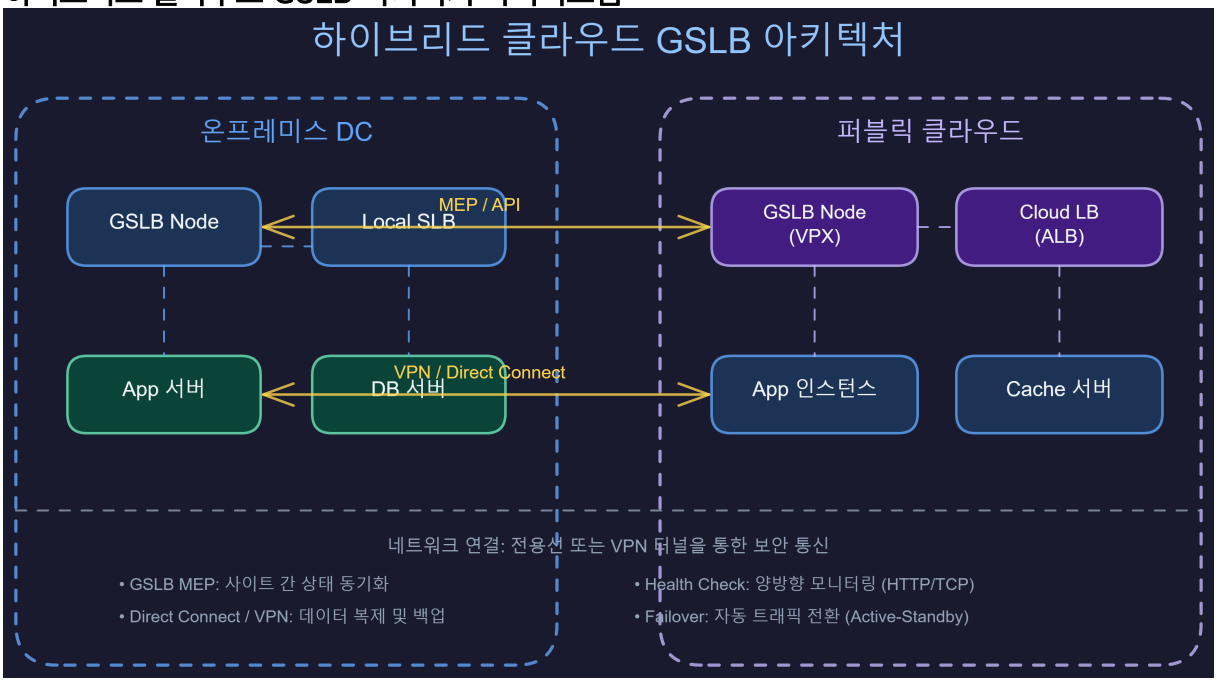
를 처리하여 최적의 사이트 VIP를 반환한다. NS 레코드가 두 개의 GSLB ADNS를 가리키므로, 하나의 GSLB 노드에 장애가 발생하더라도 다른 GSLB 노드가 DNS 쿼리를 처리할 수 있어 GSLB 자체의 고가용성이 확보된다.

3.2 하이브리드 클라우드 아키텍처

3.2.1 온프레미스 + 퍼블릭 클라우드 GSLB 구성

하이브리드 클라우드 GSLB는 온프레미스 데이터센터와 퍼블릭 클라우드 환경을 단일 GSLB 도메인으로 통합 관리하는 아키텍처이다. 온프레미스의 안정성과 보안성, 퍼블릭 클라우드의 확장성과 유연성을 결합하여 최적의 인프라 운영을 가능하게 한다.

하이브리드 클라우드 GSLB 아키텍처 다이어그램



GSLB 노드 배치 전략

하이브리드 환경에서 GSLB 노드의 배치는 각 환경의 특성에 맞게 설계해야 한다.

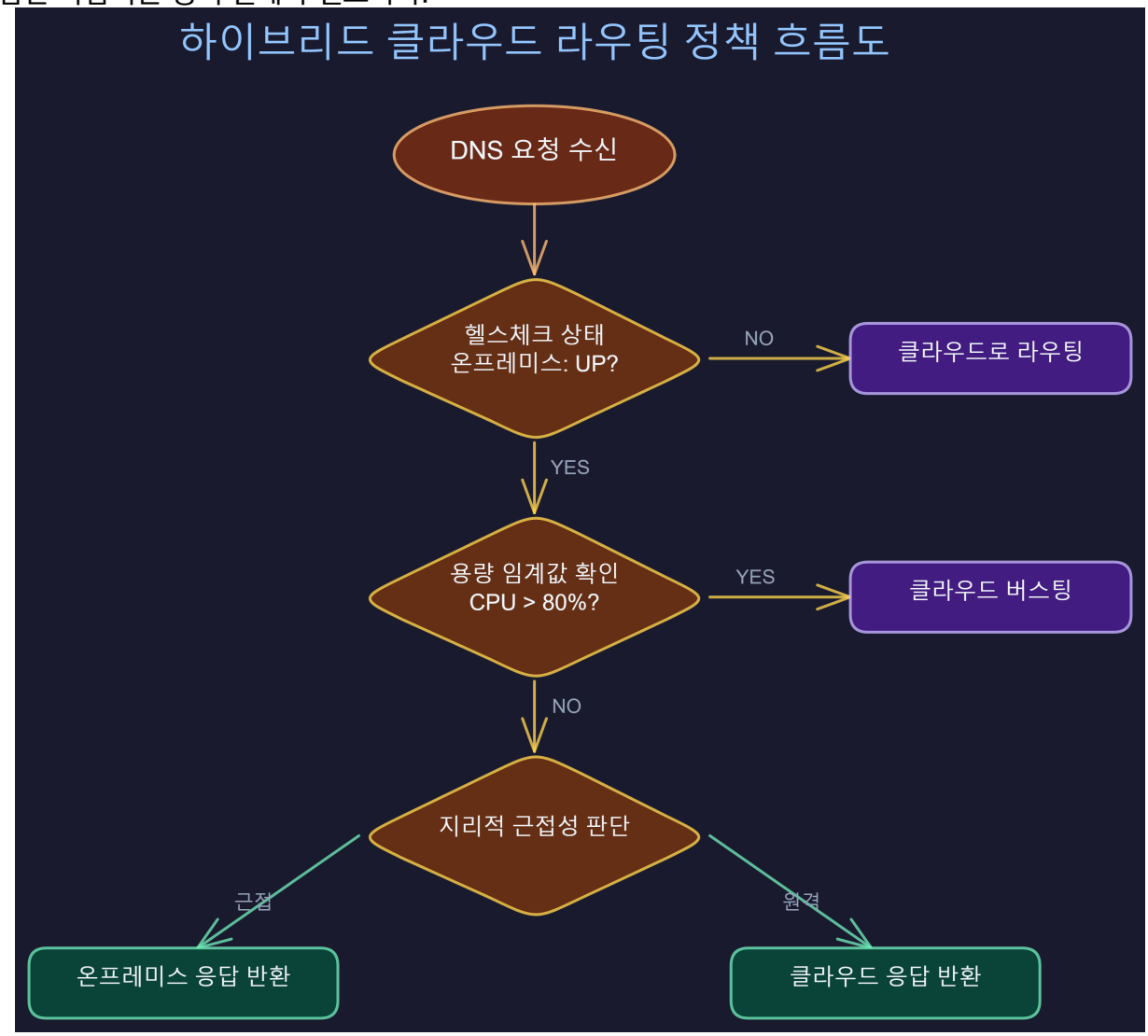
위치	GSLB 노드 형태	역할
온프레미스 DC	물리 어플라이언스 (F5 BIG-IP, NetScaler MPX)	주 GSLB 컨트롤러, ADNS, 온프레미스 로컬 LB

퍼블릭 클라우드	가상 어플라이언스 (VPX, VE) 또는 SaaS	보조 GSLB 노드, 클라우드 로컬 LB

온프레미스에서는 전용 하드웨어 어플라이언스의 높은 성능과 안정성을 활용하고, 퍼블릭 클라우드에서는 가상 어플라이언스를 통해 유연한 확장이 가능하도록 구성하는 것이 일반적이다.

트래픽 라우팅 정책 설계

하이브리드 환경에서의 트래픽 라우팅은 단순한 부하 분산을 넘어, 비용 최적화와 규정 준수를 포함한 복합적인 정책 설계가 필요하다.



이 흐름도에서 첫 번째 판단 기준은 헬스체크 상태이다. 온프레미스가 정상(UP)이면 용량 임계값을 확인하고, 임계값을 초과하면 클라우드 버스팅(Cloud Bursting)을 통해 퍼블릭 클라우드로 트래픽을 분산한다. 온프레미스가 비정상(DOWN)이면 즉시 클라우드로 페일오버한다.

주요 고려사항

하이브리드 클라우드 GSLB 설계 시 다음 네 가지 영역에 대한 면밀한 검토가 필요하다.

네트워크 연결 – 전용선 (Direct Connect / ExpressRoute): 안정적인 MEP 통신과 데이터 동기화에 필수적이다. 전용선은 예측 가능한 대역폭과 낮은 지연 시간을 제공하므로, 실시간 메트릭 교환과 데이터베이스 복제에 적합하다. – **VPN 백업**: 전용선 장애 시 대비한 IPSec VPN 이중화를 구성해야 한다. 전용선과 VPN을 Active-Standby로 운영하여 네트워크 연결의 가용성을 확보한다. – **대역폭 계획**: MEP 트래픽, 데이터 복제 트래픽, 헬스체크 트래픽을 포함한 총 대역폭을 산정하여 네트워크 용량을 계획해야 한다.

헬스체크 설계 – 온프레미스와 클라우드 간 비대칭 헬스체크 간격을 설정할 수 있다. 예를 들어 온프레미스는 10초 간격, 클라우드는 30초 간격으로 차별화할 수 있다. – 클라우드 환경에서는 CloudWatch(AWS), Azure Monitor 등 클라우드 네이티브 모니터링 서비스와 연동한 API 기반 헬스체크를 활용하는 것이 효율적이다. – 크로스 사이트 헬스체크(온프레미스에서 클라우드로, 또는 그 반대)의 경우, 사이트 간 네트워크 레이턴시를 감안하여 타임아웃 값을 넉넉하게 설정해야 한다. 네트워크 지연으로 인한 헬스체크 오탐(False Positive)을 방지하기 위함이다.

데이터 일관성 – DB 복제 전략: 온프레미스 Master에서 클라우드 Read Replica로의 비동기 복제가 일반적이다. Active-Active 구성 시에는 양방향 복제에 따른 Write Conflict 해소 전략이 필요하다. – 세션 상태 공유: Redis, Memcached 등 외부 세션 스토어를 활용하거나, JWT/Token 기반 Stateless 세션 관리를 채택하여 사이트 간 세션 일관성을 확보한다. – DNS TTL 관리: 페일오버 속도와 DNS 캐시 효율 간의 트레이드오프를 고려하여 적절한 TTL 값을 설정한다.

비용 최적화 – 클라우드 버스팅(Cloud Bursting): 평상시에는 온프레미스에서 트래픽을 처리하고, 피크 시에만 퍼블릭 클라우드로 자동 확장하는 전략이다. GSLB의 용량 기반 라우팅 정책과 결합하면, 온프레미스의 CPU/메모리 사용률이 임계값을 초과할 때 자동으로 클라우드로 트래픽을 분산할 수 있다. – **Egress 비용 고려**: 퍼블릭 클라우드에서 온프레미스 또는 인터넷으로 나가는 아웃바운드 트래픽에는 비용이 발생한다. GSLB 라우팅 정책 설계 시 이 Egress 비용을 가중치에 반영하여 비용을 최적화할 수 있다. – **예약 인스턴스 + 온디맨드 혼합**: 베이스라인 트래픽은 예약 인스턴스로, 버스팅 트래픽은 온디맨드 인스턴스로 처리하는 혼합 전략을 통해 비용 효율성을 극대화한다.

3.3 멀티 클라우드 GSLB 설계

3.3.1 클라우드 간 트래픽 분산 설계

멀티 클라우드 GSLB는 AWS, Azure, GCP 등 복수의 퍼블릭 클라우드에 분산 배포된 서비스를 단일 GSLB 도메인으로 통합 관리하는 아키텍처이다. 특정 클라우드 벤더에 대한 종속(Lock-in)을 방지하고, 각 클라우드의 강점을 결합하며, 단일 클라우드 장애 시 서비스 연속성을 보장하는 것이 핵심 목적이다.

클라우드 네이티브 GSLB 서비스 비교

각 주요 클라우드는 자체 GSLB 서비스를 제공하지만, 이들은 기본적으로 해당 클라우드 내부에 최적화되어 있으며 타 클라우드와의 통합에는 제약이 있다.

기능	AWS Route 53	Azure Traffic Manager	GCP Cloud DNS
라우팅 방식	지연시간, 지리적, 가중치, 페일오버	성능, 지리적, 가중치, 우선순위	지리적, WRR
헬스체크	HTTP/HTTPS/TCP	HTTP/HTTPS/TCP	- (외부 연동)
Anycast	지원	지원	지원
범위	AWS 내부 최적화	Azure 내부 최적화	GCP 내부 최적화
한계	타 클라우드 통합 제한적	타 클라우드 통합 제한적	타 클라우드 통합 제한적

각 클라우드의 네이티브 GSLB 서비스는 자사 클라우드 내부에서는 우수한 통합성을 제공하지만, 멀티 클라우드 환경에서는 “모든 클라우드를 동일한 수준으로 관리” 하기 어렵다는 구조적 한계가 존재한다. 이러한 한계를 극복하기 위해 벤더 중립적인 GSLB 계층의 도입이 필요하다.

멀티 클라우드 GSLB 아키텍처 다이어그램

이 아키텍처에서 최상위에 위치하는 **벤더 중립 GSLB 계층**이 핵심이다. 이 계층은 특정 클라우드에 종속되지 않는 독립적인 글로벌 트래픽 관리자로서, 모든 클라우드의 엔드포인트를 동일한 기준으로 평가하고 라우팅한다. 각 클라우드 내부에서는 해당 클라우드의 네이티브 GSLB/LB 서비스(Route 53, Traffic Manager, Cloud DNS 등)가 로컬 트래픽 관리를 담당한다.

벤더 중립적 GSLB 설계 원칙

벤더 중립 GSLB의 핵심은 특정 클라우드에 종속되지 않는 독립적인 글로벌 트래픽 관리 계층을 구축하는 것이다.

이 설계 원칙의 핵심은 Layer 3(글로벌 트래픽 관리)이 Layer 2(클라우드별 로컬 LB) 및 Layer 1(컴퓨팅)과 독립적으로 운영된다는 것이다. Layer 3에서 사용하는 GSLB 솔루션은 모든 클라우드의 엔드포인트를 동일한 인터페이스로 관리할 수 있어야 하며, Layer 4(관리/오케스트레이션)에서는 Terraform 등 IaC(Infrastructure as Code) 도구를 통해 모든 클라우드의 GSLB 설정을 코드로 관리한다.

벤더 중립 GSLB 솔루션 옵션

솔루션	유형	특징
F5 BIG-IP DNS	어플라이언스/가상	엔터프라이즈 GSLB, 온프레미스+클라우드 통합 지원
NetScaler (Citrix)	어플라이언스/가상	MEP 기반 메트릭 교환, ADC 통합, 하이브리드 환경 강점
NS1 (IBM)	SaaS	API 우선 설계, Filter Chain 기반 유연한 라우팅
Cloudflare LB	SaaS	글로벌 Anycast 네트워크, 간편한 설정, DDoS 방어 통합
A10 Thunder	어플라이언스/가상	하이브리드 클라우드 특화, 고성능
HashiCorp Consul	오픈소스	서비스 메시 통합, DNS 인터페이스 제공, 클라우드 네이티브

멀티 클라우드 GSLB 설계 시 핵심 고려사항

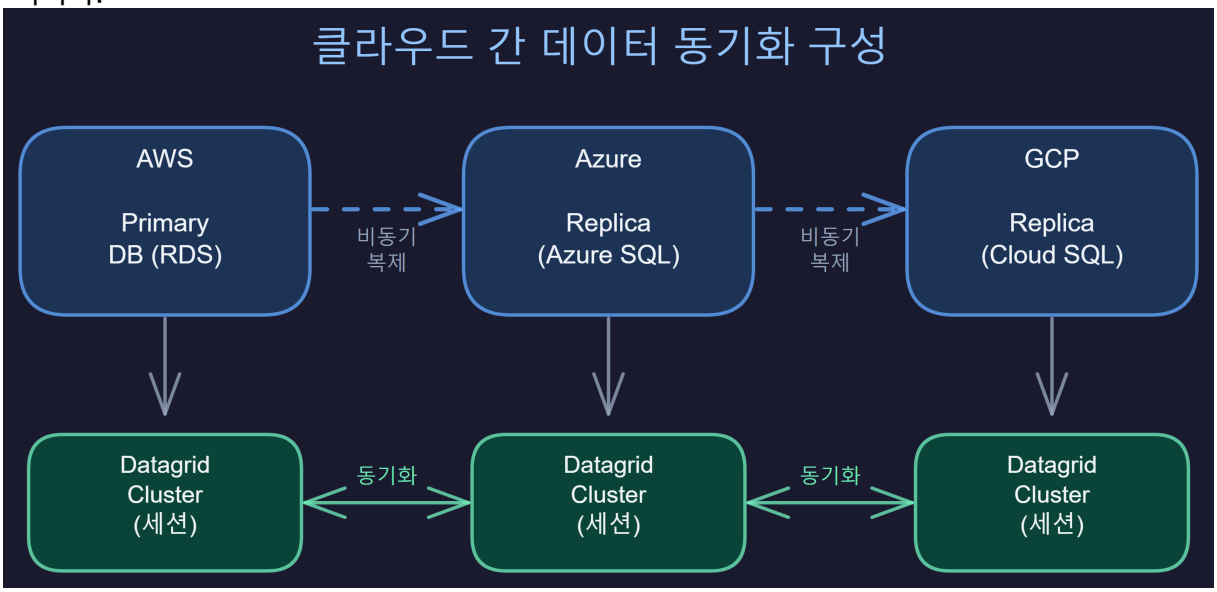
A. DNS TTL 전략

멀티 클라우드 환경에서의 DNS TTL 전략은 페일오버 속도와 DNS 쿼리 부하 간의 균형을 고려하여 설계해야 한다.

일반적으로 평상시에는 60~120초의 TTL을 유지하여 DNS 쿼리 부하를 관리하고, 장애 감지 시 동적으로 TTL을 30초로 단축하여 빠른 페일오버를 가능하게 하는 전략이 권장된다.

B. 클라우드 간 데이터 동기화

멀티 클라우드 환경에서 서비스 일관성을 유지하기 위해서는 클라우드 간 데이터 동기화가 필수적이다.

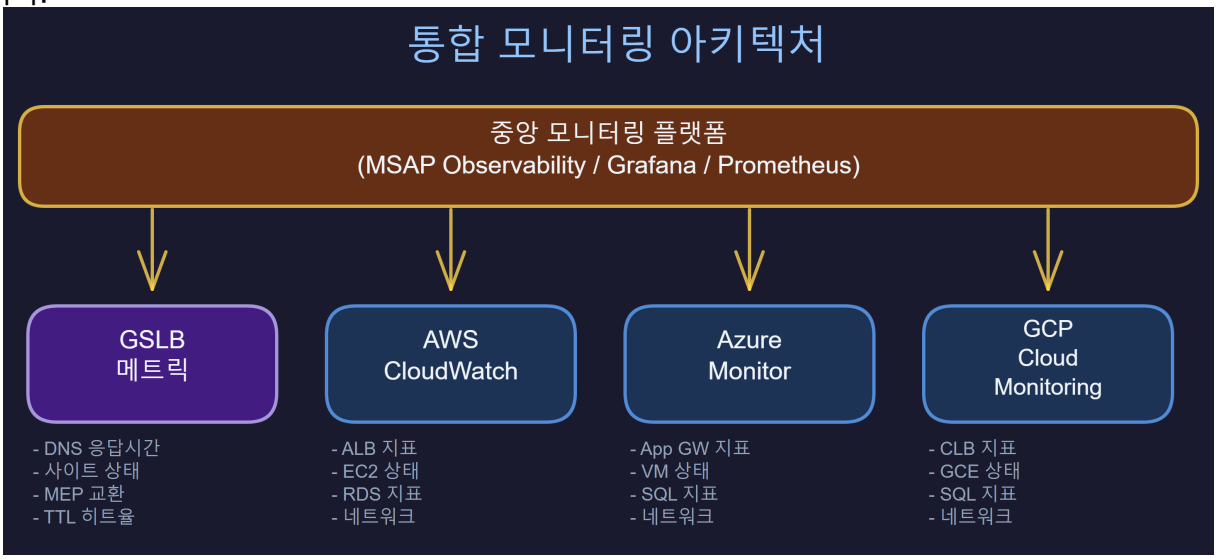


데이터베이스는 Primary-Replica 구조로 비동기 복제를 수행하고, 세션 데이터는 Redis 클러스터 간 실시간 동기화를 통해 사이트 간 세션 일관성을 확보한다. 이때 클라우드 간 네트워크

지연에 의한 복제 지연(Replication Lag)을 고려하여 Eventual Consistency 모델을 기반으로 설계하는 것이 현실적이다.

C. 통합 모니터링

멀티 클라우드 GSLB 환경에서는 각 클라우드의 개별 모니터링 서비스(CloudWatch, Azure Monitor, Cloud Monitoring)와 GSLB 자체의 메트릭을 통합하는 중앙 모니터링 플랫폼이 필수적이다.



통합 모니터링 플랫폼은 모든 클라우드와 GSLB의 메트릭을 단일 대시보드에서 조회할 수 있게 하며, 이상 징후 감지 시 통합 알림을 발생시켜 신속한 대응을 가능하게 한다. GSLB 레벨의 메트릭(DNS 응답 시간, 사이트 상태, 페일오버 이벤트 등)과 각 클라우드의 인프라 메트릭(서버 상태, DB 지표, 네트워크 지표 등)을 상관 분석(Correlation Analysis)함으로써 근본 원인 분석(Root Cause Analysis)의 효율성을 높일 수 있다.

[K8GB 맥락]

K8GB는 멀티 클라우드 GSLB 설계에서 **특정 클라우드 벤더에 종속되지 않는** 진정한 벤더 중립적 솔루션으로서의 가치를 제공한다. 상용 GSLB 장비(F5, NetScaler 등)가 자체 가상 어플라이언스를 각 클라우드에 배포하는 방식과 달리, K8GB는 **Kubernetes가 실행되는 모든 환경에서 동일하게 동작**하는 Kubernetes-native 방식을 채택한다.

K8GB의 멀티 클라우드 아키텍처에서 핵심적인 역할을 하는 것은 **ExternalDNS**이다. ExternalDNS는 K8GB Controller가 관리하는 DNS 레코드를 상위 Cloud DNS 서비스에 자동으로 동기화하는 컴포넌트로, 다음과 같은 DNS 프로바이더를 지원한다.

- AWS Route 53
- Google Cloud DNS
- Azure DNS
- Infoblox (온프레미스)
- CoreDNS (자체 호스팅)
- 기타 다수의 DNS 프로바이더

이러한 광범위한 DNS 프로바이더 지원을 통해, K8GB는 AWS EKS, Azure AKS, GCP GKE, 온프레미스 Kubernetes 클러스터 등 이기종 Kubernetes 환경 간에 **동일한 Gslb CRD와 동일한 운영 방식**으로 글로벌 트래픽을 관리할 수 있다. 각 클러스터에 독립적으로 배포되는 K8GB의 분산 아키텍처는 전용 관리 클러스터가 불필요하여 단일 장애점(SPoF)을 제거하며, 클러스터 간 상태 공유는 DNS 레코드 기반으로 이루어지므로 MEP와 같은 독점 프로토콜에 대한 의존성이 없다. 이는 멀티 클라우드 환경에서의 GSLB 운영 복잡성을 크게 줄이고, 진정한 의미의 클라우드 이식성(Cloud Portability)을 실현한다.

3장에서는 GSLB의 대표적인 아키텍처 패턴인 멀티사이트 2-Tier 구조, Active-Active/Active-Passive 배포 모델, 하이브리드 클라우드 아키텍처, 멀티 클라우드 GSLB 설계를 살펴보았다. 각 아키텍처 패턴은 서비스의 가용성 요구수준, 인프라 구성, 비용 구조, 규정 준수 요건에 따라 선택되며, 실제 환경에서는 이러한 패턴들이 조합되어 적용되는 경우가 많다. 4장에서는 GSLB가 재해복구(DR) 아키텍처에서 어떤 핵심 역할을 수행하는지, 그리고 DR 설계 시의 주요 고려사항을 살펴본다.

1~3장 요약

장	핵심 내용
1장. GSLB 개요	GSLB의 정의(DNS 기반 글로벌 트래픽 분배), SLB와의 차이, DNS 라우팅 메커니즘, 6가지 주요 알고리즘, 4단계 헬스체크/페일오버 프로세스
2장. GSLB 기술 생태계	DNS(GSLB의 핵심 기반), BGP Anycast 비교, CDN/GeoIP/API Gateway/Reverse Proxy 연동, MEP 독점 프로토콜, EDNS Client Subnet(RFC 7871)
3장. GSLB 아키텍처 패턴	2-Tier 레퍼런스 아키텍처, Active-Active/Active-Passive 배포, 하이브리드 클라우드 구성, 멀티 클라우드 벤더 중립 GSLB 설계

GSLB 이해하기 백서 — 4장~5장

4장: GSLB와 재해복구(DR)

4.1 DR 구성에서 GSLB의 역할

재해복구(Disaster Recovery, DR)는 자연재해, 하드웨어 장애, 네트워크 단절 등 예기치 못한 사고 발생 시 서비스 연속성을 보장하기 위한 전략이다. GSLB는 DR 아키텍처에서 **트래픽 라우팅 제어의 핵심 계층**으로 기능하며, 장애 감지와 자동 페일오버를 통해 사용자 요청을 정상 운영 중인 사이트로 즉시 전환하는 역할을 수행한다.

DR 구성은 크게 Active-Active와 Active-Passive 두 가지 방식으로 분류되며, 각 방식에서 GSLB가 수행하는 역할과 동작 특성이 상이하다. 본 절에서는 양 방식의 구조적 차이를 비교하고, GSLB가 각 시나리오에서 어떻게 활용되는지 심층적으로 분석한다.

4.1.1 Active-Active DR과 GSLB

Active-Active DR의 정의

Active-Active DR은 **두 개 이상의 데이터센터(또는 클러스터)가 동시에 실제 트래픽을 처리**하는 구성이다. 모든 사이트가 상시 운영 상태를 유지하므로, 특정 사이트에 장애가 발생하더라도 나머지 사이트가 즉시 전체 트래픽을 흡수할 수 있다. 이 방식은 높은 가용성과 낮은 RTO(Recovery Time Objective)를 제공하지만, 데이터 동기화와 일관성 관리에 대한 복잡도가 증가한다.

Active-Active vs Active-Passive 전면 비교

다음 표는 두 DR 구성 방식의 핵심 특성을 종합적으로 비교한 것이다.

항목	Active-Active	Active-Passive
트래픽 처리	모든 사이트가 동시에 실시간 트래픽을 처리	Primary 사이트만 처리, Standby 사이트는 대기 상태 유지
리소스 활용률	높음 (모든 인프라가 상시 활용)	낮음 (Standby 리소스가 유휴 상태)
RTO (복구 시간 목표)	수초~수십초	수분~수십분
RPO (복구 시점 목표)	0에 가까움 (실시간 동기화)	비동기 복제 간격만큼 데이터 손실 가능
데이터 동기화	실시간 동기/양방향 복제 필수	단방향 비동기 복제
비용	높음 (이중 인프라 상시 운영)	상대적으로 낮음
Failover 방식	자동 (GSLB가 장애 사이트를 즉시 제외)	수동 또는 반자동 (관리자 개입 필요)
데이터 일관성	CAP 정리 상 Trade-off 발생	단일 쓰기 지점으로 일관성 유지 용이
장애 복구 신뢰성	상시 검증 가능 (실제 트래픽 처리 중이므로)	Standby 사이트의 정기적 검증 절차 필요
아키텍처 복잡도	높음 (충돌 해소, 동기화 로직 필요)	상대적으로 낮음
확장성	수평 확장 용이	확장 시 Standby 추가 비용 발생

Active-Active 방식의 장점

1. **최소 다운타임**: 장애 발생 시 GSLB가 자동으로 비정상 사이트를 DNS 응답에서 제외하므로, 사용자는 거의 중단 없이 서비스를 이용할 수 있다.

2. **높은 리소스 효율성**: 모든 사이트가 상시 트래픽을 처리하므로 인프라 투자 대비 활용률이 극대화된다.
3. **지리적 분산에 의한 지연시간 감소**: 사용자와 가장 가까운 사이트에서 트래픽을 처리하여 응답 지연을 최소화할 수 있다.
4. **상시 검증**: 모든 사이트가 실제 트래픽을 처리하므로, 장애 복구 능력이 별도의 DR 훈련 없이도 지속적으로 검증된다.

Active-Active 방식의 단점

1. **데이터 일관성 관리의 복잡성**: 양방향 데이터 복제 시 Write Conflict가 발생할 수 있으며, 이를 해소하기 위한 별도의 전략(LWW, CRDT 등)이 필요하다.
2. **높은 초기 투자 비용**: 모든 사이트에 동일 수준의 인프라를 구축해야 하므로 초기 비용이 상당하다.
3. **네트워크 대역폭 요구사항 증가**: 실시간 양방향 복제를 위해 사이트 간 고대역폭, 저지연 네트워크 연결이 필수적이다.
4. **Split-Brain 위험**: 사이트 간 네트워크가 단절되면 양 사이트가 독립적으로 동작하여 데이터 불일치가 발생할 수 있다.

Active-Passive 방식의 장점

1. **단순한 아키텍처**: 단일 쓰기 지점을 유지하므로 데이터 일관성 관리가 용이하다.
2. **낮은 비용**: Standby 사이트는 최소 사양으로 운영할 수 있어 인프라 비용을 절감할 수 있다.
3. **검증된 안정성**: 전통적인 DR 패턴으로 운영 노하우가 풍부하다.

Active-Passive 방식의 단점

1. **높은 RTO**: Standby 사이트 기동, DNS 전파, 데이터 정합성 확인 등에 시간이 소요된다.
2. **리소스 낭비**: Standby 사이트의 인프라가 평상시 유휴 상태로 유지된다.
3. **DR 검증의 어려움**: 실제 트래픽을 처리하지 않으므로, 정기적인 DR 훈련 없이는 복구 능력을 보장하기 어렵다.

Active-Active 구성에서의 GSLB 역할

Active-Active DR에서 GSLB는 **트래픽 분산**과 **장애 감지/자동 페일오버**라는 이중 역할을 수행한다. GSLB가 지원하는 주요 트래픽 분산 알고리즘은 다음과 같다.

알고리즘	동작 방식	적합한 시나리오
Round Robin	DNS 쿼리마다 각 사이트의 IP를 순차적으로 응답	모든 사이트의 용량이 동일한 경우
Weighted Round Robin	사이트별 가중치에 따라 응답 비율 조절	사이트 간 용량이 다른 경우
GeoIP 기반	클라이언트 IP의 지리적 위치에 따라 가장 가까운 사이트로 라우팅	글로벌 서비스, 지연시간 최소화
Latency 기반	실측 지연시간이 가장 낮은 사이트로 라우팅	실시간성이 중요한 서비스
Health-aware	헬스체크 결과에 따라 비정상 사이트를 자동 제외	모든 DR 시나리오의 기본 요소

Active-Active GSLB 동작 흐름

Active-Active 구성에서 GSLB의 동작 흐름은 다음과 같다.

[정상 상태]

1. 클라이언트 → DNS 쿼리 (app.example.com)
2. GSLB → 헬스체크 결과 확인 (Site-A: 정상, Site-B: 정상)
3. GSLB → 라우팅 알고리즘 적용 (예: GeoIP → 클라이언트와 가까운 Site-A 선택)
4. GSLB → DNS 응답 (Site-A의 IP 반환, TTL: 30초)
5. 클라이언트 → Site-A에 직접 연결

[장애 발생 시]

1. GSLB 헬스체크 → Site-A 비정상 감지 (HTTP 503, TCP 연결 실패 등)
2. GSLB → Site-A를 DNS 응답 풀에서 제외
3. 클라이언트 → DNS 쿼리 (TTL 만료 후)
4. GSLB → Site-B의 IP만 응답
5. 클라이언트 → Site-B에 자동 연결 (사용자 개입 불필요)

4.1.2 Active-Passive DR과 GSLB

Passive 사이트 기동과 페일오버 지연

Active-Passive DR에서 GSLB의 역할은 주로 **장애 감지**와 **DNS 기반 트래픽 전환**에 집중된다. 그러나 Active-Active와 달리, Passive 사이트가 Cold 또는 Warm Standby 상태에 있을 수 있으므로 페일오버에 추가적인 시간이 소요된다.

Passive 사이트의 기동 상태에 따른 복구 시간 차이는 다음과 같다.

Standby 유형	설명	기동 시간	총 페일오버 시간
Hot Standby	애플리케이션이 실행 중이며 데이터가 실		
_____	_____	_____	_____

DNS TTL 캐싱에 의한 페일오버

Active-Passive DR 에서 가장 큰 제약은 **DNS TTL(Time-To- GSLB 일반 권장값, 페일오버 속도와 부하의 균

형

B가 장애를 감지하고 DNS 레코드를 업데이트하더라도, 중간 캐시 계층에 남아 있는 이전 레코드로 인해 클라이언트가 여 을 시도할 수 있다.

Failover 시간 구성 요소

전체 페일오버 시간은 다음 요소들의 합산으로 결 설명	소요 시간	
장애 감지	GSLB 헬 상 상태를 확인	10초~60초 (체크 주기 및 실패 임계값에 따라)
GSLB 내부 전파	장애 감지 결과를 GSLB 내부 라우팅 테이블에 반영	1초~5초
DNS 레코드 업데이트	권한 DNS 서버의 레코드 변경 및 전파	5초~30초
DNS 캐시 만료 대기	ISP/퍼블릭 DNS 리졸버의 캐시가 TTL 만료될 때까지 대기	0초~TTL 값 (일반적으로 30초~300초)

클라이언트 재연결	클라이언트 애플리케이션이 새로운 IP로 연결을 수립	1초~10초
Passive 사이트 기동 (해당 시)	Warm/Cold Standby의 경우 애플리케이션 기동	0초~수십분

> **참고**: Active-Passive DR의 총 페일오버 시간은 일반적으로 **수분에서 수십분** 사이이며, 이 중 DNS 캐시 만료 대기 시간이 가장 큰 변수로 작용한다. 따라서 낮은 TTL 설정과 헬스체크 주기 최적화가 핵심적인 튜닝 포인트가 된다.

4.2 DR 설계 시 고려사항

DR 아키텍처를 설계할 때는 트래픽 전환 메커니즘뿐만 아니라, 데이터 일관성, DNS 캐싱 영향, 세션 관리 등 다양한 측면을 종합적으로 고려해야 한다. 본 절에서는 DR 설계의 핵심 고려사항을 체계적으로 정리한다.

4.2.1 데이터 동기화와 일관성

DB 복제 방식 비교

DR 구성에서 가장 근본적인 과제는 **사이트 간 데이터 일관성 유지**이다. 데이터베이스 복제 방식에 따라 데이터 손실 가능성, 성능 영향, 일관성 수준이 달라진다.

복제 방식	동작 원리	RPO	성능 영향	일관성	적용 사례
동기 복제 (Synchronous)	트랜잭션 커밋 전에 원격 사이트 기록 완료를 확인	0 (데이터 무손실)	높음 (지연시간 증가)	강한 일관성	금융 거래, 결제 시스템
비동기 복제 (Asynchronous)	트랜잭션 커밋 후 원격 사이트에 비동기적으로 전파	복제 지연만큼 손실 가능	낮음	최종 일관성	로그 데이터, 분석 시스템
반동기 복제 (Semi-synchronous)	원격 사이트가 로그 수신을 확인하면 커밋 진행	0에 가까움	중간	준강한 일관성	범용 OLTP

Write Conflict 해소 전략

Active-Active 구성에서 양방향 복제 시 동일 데이터에 대한 동시 쓰기가 발생하면 Write Conflict가 발생한다. 이를 해소하기 위한 주요 전략은 다음과 같다.

1. LWW (Last Writer Wins)

가장 단순한 충돌 해소 방식으로, 타임스탬프가 더 늦은 쓰기가 최종 값으로 채택된다. 구현이 간단하나, 이전 쓰기가 무조건 폐기되므로 데이터 손실 가능성이 존재한다. 모든 사이트 간 시계 동기화(NTP)가 전제 조건이다.

2. Application-Level Resolution

애플리케이션 로직에서 충돌을 감지하고 비즈니스 규칙에 따라 해소한다. 예를 들어, 재고 관리 시스템에서 두 사이트가 동시에 재고를 차감한 경우, 두 차감을 모두 반영(Additive Merge)하는 방식이다. 비즈니스 의미를 반영할 수 있으나 구현 복잡도가 높다.

3. CRDT (Conflict-free Replicated Data Types)

수학적으로 충돌이 발생하지 않도록 설계된 데이터 구조를 사용한다. G-Counter, PN-Counter, OR-Set 등이 대표적이다. 최종 일관성을 보장하며 네트워크 파티션 상황에서도 안전하게 동작하나, 지원 가능한 데이터 구조와 연산이 제한적이다.

4. Consensus Protocol (Paxos/Raft)

분산 합의 알고리즘을 통해 쓰기 순서를 결정한다. 강한 일관성을 보장하나, 사이트 간 네트워크 지연으로 인해 쓰기 성능이 저하될 수 있다. CockroachDB, TiDB 등의 분산 데이터베이스가 이 방식을 채택한다.

Split-Brain 방지 전략

사이트 간 네트워크가 단절되면 각 사이트가 독립적으로 동작하는 Split-Brain 상태가 발생할 수 있다. 이는 데이터 불일치, 이중 처리 등 심각한 문제를 야기하므로 반드시 방지 메커니즘을 구현해야 한다.

1. Quorum 기반 의사결정

전체 노드의 과반수(Quorum)가 합의해야만 쓰기를 허용한다. 네트워크 파티션 시 과반수를 확보하지 못한 측은 자동으로 읽기 전용 모드로 전환된다. 최소 3개 이상의 사이트(또는 Witness

노드)가 필요하다.

2. Witness/Arbiter 노드

데이터를 저장하지 않지만 투표에 참여하는 경량 노드를 제3의 위치에 배치한다. 2개 사이트 + 1개 Witness 구성으로 Quorum을 확보할 수 있어 비용 효율적이다.

3. STONITH/Fencing

Split-Brain이 감지되면 한쪽 사이트의 스토리지 접근을 강제로 차단(Fence)하여 데이터 손상을 방지한다. “Shoot The Other Node In The Head”의 약자로, 클러스터 펜싱(Cluster Fencing)이라고도 한다.

세션 동기화 전략

사용자 세션 데이터의 사이트 간 동기화는 DR 페일오버 시 사용자 경험에 직접적인 영향을 미친다. 주요 세션 관리 방식과 특성은 다음과 같다.

방식	설명	장점	단점	DR 호환성
Sticky Session	로드밸런서가 사용자를 최초 접속 서버에 고정	구현 단순, 추가 인프라 불필요	페일오버 시 세션 유실, 부하 불균형	낮음
External Session Store	Redis, Memcached 등 외부 저장소에 세션 집중 보관	서버 무관하게 세션 유지, 수평 확장 용이	외부 저장소가 SPOF 가능, 네트워크 지연	높음 (저장소 복제 시)
Session Replication	클러스터 내 모든 서버에 세션을 실시간 복제	어떤 서버로 접속해도 세션 유지	메모리 사용량 증가, 복제 오버헤드	중간
JWT/Token 기반	세션 상태를 서명된 토큰에 인코딩하여 클라이언트에 저장	서버 무상태(Stateless), 사이트 무관하게 유효	토큰 크기 제한, 실시간 무효화 어려움	매우 높음

DR 관점 권장 사항: JWT/Token 기반 방식은 서버 측 세션 저장이 불필요하므로 DR 페일오버 시 가장 원활한 사용자 경험을 제공한다. 민감한 세션 데이터가 필요한 경우, External Session Store를 Active-Active 복제 구성으로 배포하는 방식이 적합하다.

4.2.2 DNS TTL 캐싱과 페일오버 지연

DNS 캐시 체인

DNS 기반 GSLB에서 페일오버 지연의 가장 큰 원인은 **다단계 DNS 캐시 체인**이다. DNS 쿼리 결과는 여러 계층에서 캐싱되며, 각 계층의 캐시가 만료될 때까지 이전 레코드가 유효한 것으로 간주된다.

[그림 9] 687

캐싱 계층별 영향 분석

캐싱 계층	캐시 지속 시간	TTL 준수 여부	관리 가능 여부	영향도
권한 DNS (GSLB)	TTL에 따라 결정	해당 없음 (직접 제어)	완전 제어 가능	없음 (원본 변경 즉시 반영)
ISP DNS 리졸버	TTL 또는 자체 최소 TTL	부분적 (일부 ISP는 최소 TTL 강제)	불가능	높음
퍼블릭 DNS 리졸버	TTL 준수	대체로 준수	불가능	중간
OS DNS 캐시	TTL 또는 OS 설정	대체로 준수	제한적 (OS 설정 변경)	중간
브라우저 캐시	자체 정책 (수십초~수분)	브라우저마다 상이	불가능	중간~높음
애플리케이션 캐시	구현에 따라 상이	구현에 따라 상이	애플리케이션 레벨에서 제어	높음 (장기 캐싱 시)

TTL 전략별 트레이드오프

TTL 값	페일오버 속도	DNS 쿼리 부하	네트워크 영향	적합한 시나리오
5~15초	매우 빠름	매우 높음	DNS 인프라 부하 증가	미션 크리티컬 서비스 (금융, 의료)
30~60초	빠름	높음	관리 가능한 수준	일반적인 DR 구성 (권장)
120~300초	보통	보통	적음	변경 빈도가 낮은 서비스
600초 이상	느림	낮음	최소	DR 요구사항이 낮은 정적 콘텐츠

해결 방안

1. 낮은 TTL 설정

DR 대상 서비스의 DNS TTL을 30~60초로 설정하여 캐시 만료를 가속화한다. 단, TTL을 과도하게 낮추면 DNS 쿼리 부하가 증가하므로, DNS 인프라의 처리 용량을 사전에 확인해야 한다.

2. Anycast DNS 병행

Anycast를 활용하면 동일한 DNS 서버 IP를 여러 지리적 위치에서 광고할 수 있다. 이를 통해 DNS 쿼리 지연을 최소화하고, DNS 서버 자체의 가용성도 향상시킬 수 있다.

3. 클라이언트 재시도 로직

애플리케이션 레벨에서 연결 실패 시 DNS를 재질의하고, 다른 IP로 재연결을 시도하는 로직을

구현한다. 이를 통해 DNS 캐시 만료를 기다리지 않고도 빠른 페일오버가 가능하다.

4. Happy Eyeballs 알고리즘 (RFC 8305)

Happy Eyeballs는 클라이언트가 DNS 응답으로 받은 복수의 IP 주소에 대해 **동시에 연결을 시도**하여, 가장 먼저 응답하는 주소를 사용하는 알고리즘이다. 원래 IPv4/IPv6 듀얼 스택 환경을 위해 설계되었으나, DR 페일오버 시에도 유사한 효과를 제공한다.

동작 방식은 다음과 같다: 1. DNS 쿼리로 복수의 IP 주소(A/AAAA 레코드)를 수신한다. 2. 첫 번째 IP로 TCP 연결을 시도한다. 3. 250ms 이내에 응답이 없으면, 두 번째 IP로도 동시에 연결을 시도한다. 4. 가장 먼저 성공한 연결을 채택하고 나머지를 취소한다.

이 알고리즘이 클라이언트 측에 구현되어 있으면, DNS 캐시에 장애 사이트의 IP가 남아 있더라도 정상 사이트로의 연결이 빠르게 이루어질 수 있다. 현대의 주요 브라우저(Chrome, Firefox, Safari)와 운영체제(Windows 10+, macOS, iOS, Android)는 이미 Happy Eyeballs를 지원한다.

Failover 시나리오 시뮬레이션

TTL을 60초로 설정한 Active-Passive DR 환경에서의 페일오버 진행 상황을 시간별로 시뮬레이션한다.

T+0분 (장애 발생) - Primary 사이트(Site-A)에 장애가 발생한다. - GSLB 헬스체크가 아직 다음 주기에 도달하지 않았으므로, DNS 응답은 여전히 Site-A의 IP를 반환한다. - 신규 접속 사용자와 기존 사용자 모두 Site-A로 연결을 시도하여 오류가 발생한다.

T+5분 (장애 감지 및 DNS 전환 완료) - GSLB 헬스체크가 Site-A의 장애를 감지하고 (약 30초~1분), DNS 레코드를 Site-B의 IP로 업데이트한다. - 권한 DNS 서버는 이미 Site-B의 IP를 응답하지만, 중간 캐시 계층에는 아직 이전 레코드가 남아 있다. - TTL이 만료된 클라이언트는 새로운 DNS 쿼리를 통해 Site-B로 연결된다. - TTL이 아직 유효한 클라이언트는 여전히 Site-A로 연결을 시도하여 오류가 지속된다. - Happy Eyeballs가 지원되는 클라이언트는 Site-B로 자동 전환될 수 있다.

T+15분 (캐시 완전 만료) - 대부분의 DNS 캐시에서 이전 레코드가 만료되어, 거의 모든 클라이언트가 Site-B로 연결된다. - 일부 ISP의 비표준 캐싱 정책으로 인해 소수의 사용자가 여전히 Site-A를 참조할 수 있으나, 이는 시간이 지남에 따라 자연적으로 해소된다. - Site-B의 트래픽이

100%에 근접하며, 정상 서비스가 완전히 복구된다.

종합 권장 구성

구성 항목	권장 값/방식	근거
DNS TTL	30~60초	페일오버 속도와 DNS 부하 간 균형점
헬스체크 주기	10~15초	장애 감지 지연 최소화
헬스체크 실패 임계값	2~3회 연속 실패	일시적 오류에 의한 오탐 방지
클라이언트 재시도	지수 백오프 + DNS 재질의	캐시 만료 전 빠른 복구
Happy Eyeballs	활성화 권장	캐시 문제 완화
모니터링	실시간 헬스체크 + DNS 쿼리 분석	장애 감지 및 복구 상태 확인
DNS 인프라	Anycast DNS + 다중 리졸버	DNS 자체의 가용성 확보
세션 관리	JWT 또는 External Store (Active-Active 복제)	페일오버 시 세션 유지

5장: K8GB — Kubernetes 네이티브 GSLB

5.1 K8GB 프로젝트 소개

5.1.1 K8GB 개요와 탄생 배경

K8GB(Kubernetes Global Balancer)는 Kubernetes 환경에 특화된 **오픈소스 글로벌 서버 로드밸런싱(GSLB) 솔루션**이다. CNCF(Cloud Native Computing Foundation) Sandbox 프로젝트로 등록되어 있으며, Apache 2.0 라이선스 하에 완전한 오픈소스로 제공된다.

탄생 배경

전통적인 GSLB 솔루션(F5 BIG-IP DNS, Citrix GSLB, Infoblox 등)은 다음과 같은 한계를 가지고 있었다.

1. **높은 라이선스 비용**: 상용 GSLB 어플라이언스는 수천만 원에서 수억 원에 이르는 초기 비용과 연간 유지보수 비용이 발생한다.
2. **Kubernetes 비친화적**: 전통 GSLB는 물리/가상 서버의 IP 주소를 기반으로 동작하며, Kubernetes의 Pod 레벨 헬스체크나 동적 서비스 디스커버리를 직접 활용하지 못한다.
3. **중앙 집중형 관리**: 별도의 GSLB 관리 클러스터나 전용 어플라이언스가 필요하여 단일 장애 지점(Single Point of Failure, SPoF)이 될 수 있다.
4. **Infrastructure as Code 비호환**: 선언적 구성 관리, GitOps 워크플로우와의 통합이 제한적이다.

K8GB는 이러한 문제를 해결하기 위해 탄생하였으며, Kubernetes의 확장 메커니즘(CRD, Controller)을 활용하여 GSLB 기능을 클러스터 내부에 내장하는 접근 방식을 채택하였다.

핵심 특징

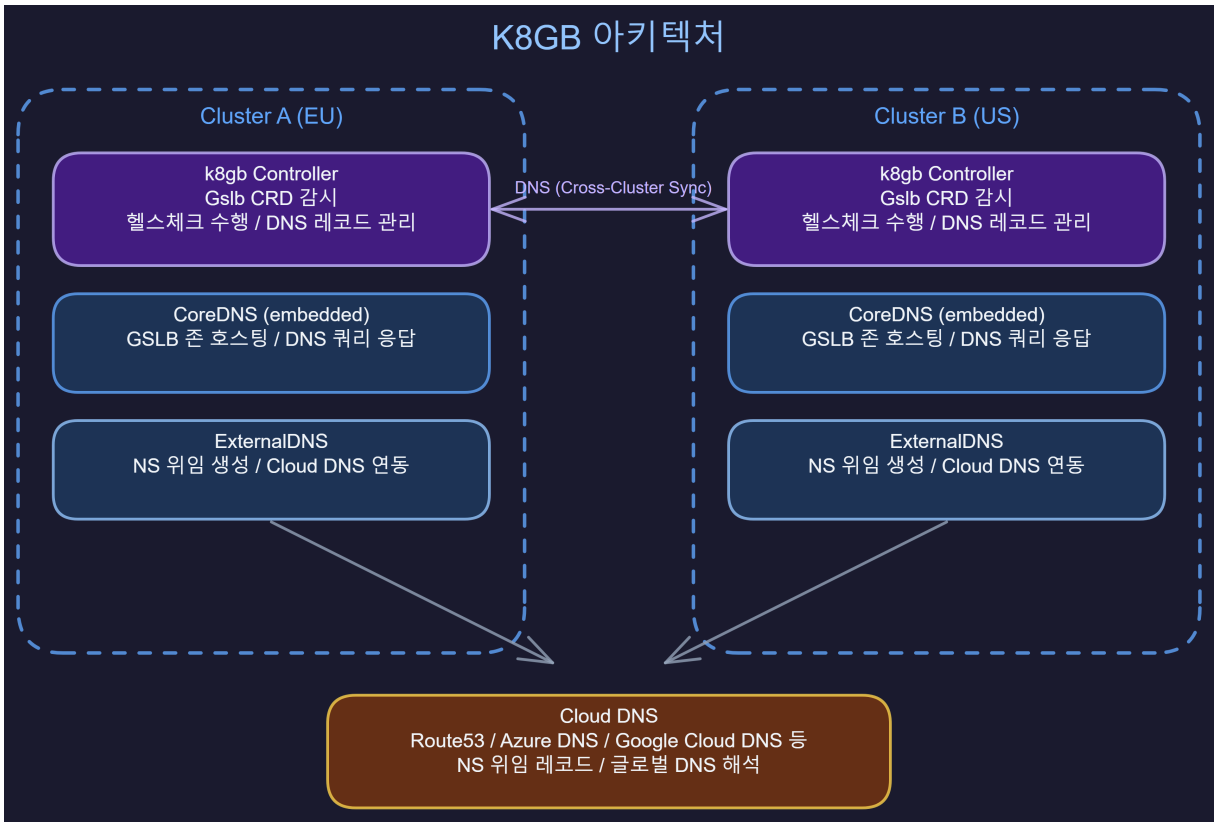
K8GB의 주요 설계 원칙과 특징은 다음과 같다.

특징	설명
DNS 프로토콜 기반	표준 DNS 프로토콜을 사용하여 글로벌 트래픽을 라우팅하므로, 특수한 클라이언트 에이전트나 프로토콜이 불필요하다
전용 관리 클러스터 불필요	각 클러스터에서 독립적으로 동작하며 클러스터 간 DNS 프로토콜로만 통신하므로, 중앙 관리 지점이 없어 SPoF가 제거된다
Kubernetes 네이티브 헬스체크	Pod의 Readiness Probe 상태를 직접 참조하여 서비스 건강 상태를 판단하므로, 별도의 헬스체크 엔드포인트 구성이 불필요하다
다양한 로드밸런싱 전략	failover, roundRobin, weightRoundRobin, geoip 등 주요 GSLB 전략을 모두 지원한다
선언적 구성	Kubernetes CRD를 통해 GSLB 정책을 선언적으로 정의하고, GitOps 워크플로우와 자연스럽게 통합된다
경량 아키텍처	CoreDNS와 ExternalDNS를 재사용하여 추가 인프라 요구사항을 최소화한다
CNCF 생태계 통합	Prometheus 메트릭, OpenTelemetry 트레이싱, Helm/Kustomize 배포를 기본 지원한다

5.1.2 K8GB 아키텍처

K8GB는 **k8gb Controller**, **CoreDNS(내장)**, **ExternalDNS**의 세 가지 핵심 컴포넌트로 구성된다. 각 Kubernetes 클러스터에 이 세 컴포넌트가 독립적으로 배포되며, 클러스터 간 통신은 DNS 프로토콜을 통해 이루어진다.

K8GB 아키텍처 다이어그램



컴포넌트 역할 상세

컴포넌트	역할	상세 설명
k8gb Controller	Gslb CRD를 감시하고, Readiness Probe 상태에 따라 CoreDNS 레코드를 업데이트	Kubernetes Operator 패턴으로 구현된 핵심 컨트롤러이다. Gslb Custom Resource의 생성/수정/삭제 이벤트를 감시하며, 대상 Ingress 리소스에 연결된 서비스의 Readiness Probe 상태를 주기적으로 확인한다. 건강한 엔드포인트의 IP를 수집하여 CoreDNS의 DNS 레코드를 갱신하고, 원격 클러스터의 상태 정보를 DNS TXT 레코드를 통해 교환한다.

CoreDNS (내장)	실제 GSLB 로드밸런싱 존을 호스팅하고 DNS 쿼리에 응답	K8GB에 내장된 CoreDNS 인스턴스로, GSLB 전용 DNS 존을 관리한다. k8gb Controller가 업데이트한 DNS 레코드를 기반으로 클라이언트의 DNS 쿼리에 응답하며, 로드밸런싱 전략(failover, roundRobin 등)에 따라 적절한 IP 주소를 반환한다. 또한 원격 클러스터의 CoreDNS와 DNS 기반 상태 교환을 수행한다.
ExternalDNS	상위 Cloud DNS에 NS 위임 레코드를 자동 생성	Route53, Azure DNS, Google Cloud DNS 등의 클라우드 DNS 서비스에 NS 위임(delegation) 레코드를 자동으로 생성하고 관리한다. 이를 통해 외부 클라이언트의 DNS 쿼리가 K8GB의 CoreDNS로 올바르게 라우팅된다. ExternalDNS는 CNCF 인큐베이팅 프로젝트로, K8GB와 독립적으로 발전하는 범용 DNS 레코드 관리 도구이다.

5.2 K8GB 동작 방식과 설정

5.2.1 트래픽 흐름 상세

클라이언트의 DNS 쿼리가 K8GB를 통해 최종적으로 애플리케이션에 도달하는 전체 과정을 5단계로 상세히 설명한다.

Step 1: DNS 위임 설정

ExternalDNS가 Cloud DNS(예: Route53)에 NS 위임 레코드를 생성한다. 이 레코드는 GSLB 대상 도메인(예: `app.cloud.example.com`)에 대한 DNS 쿼리를 각 클러스터의 CoreDNS로 위임하도록 지시한다.

```
;; Cloud DNS (Route53)에 생성되는 NS 레코드
cloud.example.com.  NS  gslb-ns-eu-cloud.example.com.
cloud.example.com.  NS  gslb-ns-us-cloud.example.com.

;; Glue Record (NS의 IP 주소)
gslb-ns-eu-cloud.example.com.  A  203.0.113.10  ; EU CoreDNS
gslb-ns-us-cloud.example.com.  A  198.51.100.20 ; US CoreDNS
```

Step 2: 헬스체크 및 상태 동기화

k8gb Controller가 로컬 클러스터의 서비스 Readiness 상태를 확인하고, 건강한 엔드포인트의 IP 목록을 CoreDNS에 등록한다. 동시에 원격 클러스터와 DNS TXT 레코드를 통해 상태 정보를 교환한다.

```
;; EU 클러스터의 CoreDNS에 등록되는 레코드
app.cloud.example.com.  60  A     10.0.1.100  ; EU Ingress IP
app.cloud.example.com.  60  A     10.0.1.101  ; EU Ingress IP (추가)
app.cloud.example.com.  60  TXT   "eu:10.0.1.100,10.0.1.101"

;; 원격 클러스터(US)의 상태 정보 (DNS를 통해 수신)
app.cloud.example.com.  60  TXT   "us:10.0.2.200,10.0.2.201"
```

Step 3: 클라이언트 DNS 쿼리

사용자가 app.cloud.example.com에 접속하면, 클라이언트의 DNS 리졸버가 재귀적 쿼리를 수행한다. Cloud DNS의 NS 위임에 따라 최종적으로 K8GB의 CoreDNS에 쿼리가 도달한다.

```
클라이언트 → 로컬 DNS → Root DNS → .com TLD → example.com 권한 DNS
→ Route53 (NS 위임) → K8GB CoreDNS (EU 또는 US)
```

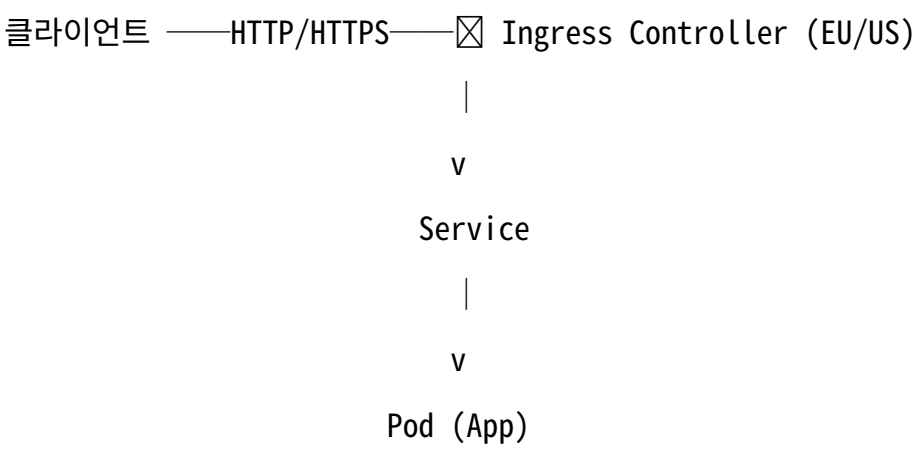
Step 4: 로드밸런싱 결정

CoreDNS는 설정된 전략(failover, roundRobin, geoip 등)에 따라 응답할 IP 주소를 결정한다.

- **failover 전략**: Primary GeoTag(예: EU)의 엔드포인트가 건강하면 EU의 IP만 반환. EU가 비정상이면 US의 IP를 반환.
- **roundRobin 전략**: 모든 건강한 클러스터의 IP를 순차적으로 또는 무작위로 반환.
- **geoip 전략**: 클라이언트의 소스 IP 지역을 기반으로 가장 가까운 클러스터의 IP를 반환.

Step 5: 트래픽 전달

클라이언트가 DNS 응답으로 받은 IP 주소(Ingress Controller의 외부 IP)로 직접 HTTP/HTTPS 연결을 수립한다. 이후의 트래픽은 K8GB를 경유하지 않고 **클라이언트와 대상 클러스터 간 직접 통신**으로 처리된다. K8GB는 DNS 레벨에서만 개입하며, 실제 데이터 플레인에는 관여하지 않는다.



> **핵심 포인트**: K8GB는 **컨트롤 플레인(DNS)**에서만 동작하며, **데이터 플레인(실제 트래픽)**에는 관여하지 않는다. 이 아키텍처는 K8GB 자체의 장애가 기존 트래픽에 영향을 주지 않음을 의미한다(DNS 캐시가 유효한 동안).

5.2.2 Gslb CRD와 로드밸런싱 전략

Gslb CRD 구조

K8GB의 핵심 인터페이스는 `Gslb` Custom Resource Definition(CRD)이다. 이 CRD를 통해 GSLB 정책을 선언적으로 정의한다.

```yaml
apiVersion: k8gb.absa.oss/v1beta1
kind: Gslb
metadata:
  name: app-gslb
  namespace: production
spec:
```

```
  ingress:
    ingressRef:
      name: app-ingress           # 대상 Ingress 리소스 이름
  strategy:
    type: failover                # 로드밸런싱 전략
    dnsTtlSeconds: 30             # DNS TTL (초)
    primaryGeoTag: "eu"           # failover 시 Primary 리전
    splitBrainThresholdSeconds: 300  # Split-Brain 감지 임계값
status:
  serviceHealth:                  # 자동 업데이트되는 상태 정보
    app.cloud.example.com:
      "eu": "Healthy"
      "us": "Healthy"
  healthyRecords:
    app.cloud.example.com:
      - ip: 10.0.1.100
        geoTag: "eu"
      - ip: 10.0.2.200
        geoTag: "us"
  geoTag: "eu"
```

로드밸런싱 전략 비교

K8GB가 지원하는 로드밸런싱 전략의 특성과 사용 사례를 비교한다.

전략	설명	동작 방식	사용 사례	구성 예시
failover	Primary 리전 우선, 장애 시 Secondary로 전환	primaryGeoTag로 지정된 클러스터의 IP를 우선 반환. 해당 클러스터의 엔드포인트가 모두 비정상이면 다른 클러스터로 전환	DR 시나리오, Active-Passive 구성	type: failover + primaryGeoTag: "eu"
roundRobin	모든 건강한 클러스터에 균등 분배	건강한 모든 클러스터의 IP를 DNS 응답에 포함하여 균등하게 분산	글로벌 트래픽 분산, Active-Active 구성	type: roundRobin
weightRoundRobin	가중치 기반 비율 분배	각 클러스터에 지정된 가중치에 따라 DNS 응답 비율을 조절	카나리 배포, 점진적 마이그레이션	type: weightRoundRobin + weights 설정
geoip	클라이언트 위치 기반 가장 가까운 클러스터 라우팅	클라이언트 DNS 리졸버의 IP 위치를 기반으로 지리적으로 가장 가까운 클러스터의 IP를 반환	지연시간 최소화, 데이터 주권 준수	type: geoip

전략별 CRD 설정 예시

Failover 전략

```yaml
apiVersion: k8gb.absa.oss/v1beta1
kind: Gslb
metadata:
  name: app-failover
  namespace: production
spec:
  ingress:
    ingressRef:
      name: app-ingress
  strategy:
    type: failover
    dnsTtlSeconds: 60
    primaryGeoTag: "eu"
    splitBrainThresholdSeconds: 300
```

Round Robin 전략

```yaml
apiVersion: k8gb.absa.oss/v1beta1
kind: Gslb
metadata:
  name: app-roundrobin
  namespace: production
spec:
  ingress:
    ingressRef:
      name: app-ingress
  strategy:
    type: roundRobin
    dnsTtlSeconds: 30
```

Weighted Round Robin 전략

```yaml
apiVersion: k8gb.absa.oss/v1beta1
kind: Gslb
metadata:
  name: app-weighted
  namespace: production
spec:
  ingress:
    ingressRef:
      name: app-ingress
  strategy:
```

```
    type: weightRoundRobin
    dnsTtlSeconds: 30
    weights:
      eu: 80
      us: 20
```

GeoIP 전략

```
apiVersion: k8gb.absa.oss/v1beta1
kind: Gslb
metadata:
  name: app-geoip
  namespace: production
spec:
  ingress:
    ingressRef:
      name: app-ingress
  strategy:
    type: geoip
    dnsTtlSeconds: 30
```

5.2.3 설치 및 멀티 클러스터 구성

사전 요구사항

K8GB를 배포하기 위해서는 다음 조건이 충족되어야 한다.

요구사항	상세
Kubernetes 클러스터	2개 이상의 Kubernetes 클러스터 (v1.19 이상 권장)
Ingress Controller	Nginx Ingress Controller, Traefik, 또는 기타 호환 Ingress Controller
Cloud DNS 접근 권한	Route53, Azure DNS, Google Cloud DNS 등에 대한 API 접근 권한 (ExternalDNS용)
네트워크 연결	클러스터 간 DNS 트래픽(UDP/TCP 53) 통신 가능
Helm v3	K8GB Helm Chart 배포를 위한 Helm v3
도메인	GSLB에 사용할 도메인 및 DNS 존 관리 권한

Helm을 통한 설치

Step 1: Helm Repository 추가

```
helm repo add k8gb https://www.k8gb.io
helm repo update
```

Step 2: EU 클러스터에 K8GB 설치

```
# EU 클러스터 컨텍스트로 전환
kubectl config use-context eu-cluster

helm install k8gb k8gb/k8gb \
  --namespace k8gb \
  --create-namespace \
  --set k8gb.clusterGeoTag="eu" \
  --set k8gb.extGslbClustersGeoTags="us" \
  --set k8gb.edgeDNSZone="example.com" \
  --set k8gb.edgeDNSServers[0]="1.2.3.4" \
  --set k8gb.dnsZone="cloud.example.com" \
  --set k8gb.reconcileRequeueSeconds=30 \
  --set externaldns.provider=aws \
  --set externaldns.aws.credentials.secretName=route53-credentials
```

Step 3: US 클러스터에 K8GB 설치

```
# US 클러스터 컨텍스트로 전환
kubectl config use-context us-cluster

helm install k8gb k8gb/k8gb \
  --namespace k8gb \
  --create-namespace \
  --set k8gb.clusterGeoTag="us" \
  --set k8gb.extGslbClustersGeoTags="eu" \
  --set k8gb.edgeDNSZone="example.com" \
  --set k8gb.edgeDNSServers[0]="1.2.3.4" \
  --set k8gb.dnsZone="cloud.example.com" \
  --set k8gb.reconcileRequeueSeconds=30 \
  --set externaldns.provider=aws \
  --set externaldns.aws.credentials.secretName=route53-credentials
```

Multi-Zone 구성 (v0.14+)

K8GB v0.14부터는 단일 클러스터에서 여러 DNS 존을 동시에 관리할 수 있는 Multi-Zone 기능이 도입되었다. 이를 통해 하나의 K8GB 인스턴스로 복수의 도메인에 대한 GSLB를 운영할 수 있다.

```
# values.yaml - Multi-Zone 구성
k8gb:
  clusterGeoTag: "eu"
  extGslbClustersGeoTags: "us"

  # Primary Zone
  dnsZone: "cloud.example.com"
  edgeDNSZone: "example.com"
  edgeDNSServers:
    - "1.2.3.4"

  # Additional Zones (v0.14+)
  dnsZones:
    - zone: "cloud.internal.example.com"
      edgeDNSZone: "internal.example.com"
      edgeDNSServers:
        - "5.6.7.8"
    - zone: "cloud.staging.example.com"
      edgeDNSZone: "staging.example.com"
      edgeDNSServers:
        - "9.10.11.12"

  reconcileRequeueSeconds: 30

externaldns:
  provider: aws
  aws:
    credentials:
      secretName: route53-credentials
```

설치 검증

설치가 완료되면 다음 명령으로 K8GB 컴포넌트의 상태를 확인한다.

```
# K8GB Pod 상태 확인
kubectl get pods -n k8gb

# 예상 출력:
# NAME                          READY   STATUS    RESTARTS   AGE
```

```
# k8gb-7b9d8f6c5-abcde          1/1     Running   0          2m
# k8gb-coredns-abcdef1234-fg    1/1     Running   0          2m
# k8gb-externaldns-xyz789-hi    1/1     Running   0          2m

# Gslb CRD 등록 확인
kubectl get crd | grep gslb
# gslbs.k8gb.absa.oss                    2024-01-15T10:30:00Z

# DNS 레코드 확인 (dig 사용)
dig app.cloud.example.com @<CoreDNS-Service-IP>
```

5.3 K8GB와 클라우드 네이티브 생태계

K8GB는 독립적으로도 강력한 GSLB 기능을 제공하지만, 클라우드 네이티브 생태계의 다른 도구들과 결합하면 더욱 강력한 인프라를 구축할 수 있다. 본 절에서는 Service Mesh, GitOps, IaC 도구 등과의 통합 방안을 다룬다.

5.3.1 Service Mesh(Istio)와의 연동

GSLB vs Service Mesh 비교

K8GB(GSLB)와 Service Mesh(Istio, Linkerd 등)는 모두 트래픽 관리를 수행하지만, 동작 계층과 범위가 근본적으로 다르다. 두 기술은 대체 관계가 아닌 **상호 보완 관계**로 이해해야 한다.

비교 항목	GSLB (K8GB)	Service Mesh (Istio)
동작 계층	L4/L7 DNS 레벨	L4/L7 데이터 플레인 (Sidecar Proxy)
트래픽 방향	클러스터 간 (North-South, 인바운드)	클러스터 내/간 (East-West, 서비스 간)
라우팅 기준	DNS 레코드, 지리적 위치, 헬스체크	HTTP 헤더, URL 경로, 가중치, mTLS ID
헬스체크	DNS 기반, K8s Readiness Probe 연동	Sidecar Proxy의 L7 헬스체크
보안	DNS 레벨 보안 (DNSSEC)	mTLS, 인증/인가 정책
가시성	DNS 쿼리 메트릭, 헬스 상태	분산 트레이싱, 서비스 메트릭, 트래픽 그래프
페일오버 단위	클러스터/사이트 단위	개별 서비스/Pod 단위
오버헤드	최소 (DNS 쿼리만 처리)	Sidecar Proxy에 의한 지연시간/리소스 추가

Istio 멀티 클러스터 토폴로지

Istio를 K8GB와 함께 사용할 때 고려할 수 있는 멀티 클러스터 토폴로지는 두 가지이다.

1. Primary-Remote 토폴로지

하나의 클러스터(Primary)에 Istio 컨트롤 플레인을 배포하고, 다른 클러스터(Remote)는 해당 컨트롤 플레인에 연결하여 메시를 확장한다.

- **장점**: 단일 컨트롤 플레인으로 관리가 간편하다.
- **단점**: Primary 클러스터 장애 시 Remote 클러스터의 메시 기능이 제한될 수 있다.
- **K8GB 연동**: K8GB가 클러스터 레벨의 인바운드 트래픽 분산을 담당하고, Istio가 클러스터 내부의 서비스 간 트래픽을 관리한다.

2. Multi-Primary 토폴로지

각 클러스터에 독립적인 Istio 컨트롤 플레인을 배포하고, 상호 메시 피어링을 설정한다.

- **장점**: 각 클러스터가 독립적으로 동작하므로 단일 장애 지점이 없다.
- **단점**: 설정 동기화와 인증서 관리가 복잡하다.
- **K8GB 연동**: K8GB가 글로벌 인바운드 라우팅을, Istio Multi-Primary가 크로스 클러스터 서비스 디스커버리와 트래픽 관리를 담당한다.

Istio VirtualService와 K8GB 연동

K8GB로 인바운드 트래픽을 적절한 클러스터로 라우팅한 후, Istio VirtualService를 사용하여 클러스터 내부에서 세밀한 트래픽 관리를 수행할 수 있다.

```yaml
# Istio VirtualService 예시 - 클러스터 내부 카나리 배포
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: app-vs
  namespace: production
spec:
  hosts:
    - app.cloud.example.com
  gateways:
    - app-gateway
```

```
  http:
    - match:
        - headers:
            x-canary:
              exact: "true"
      route:
        - destination:
            host: app-service
            subset: canary
          weight: 100
    - route:
        - destination:
            host: app-service
            subset: stable
          weight: 90
        - destination:
            host: app-service
            subset: canary
          weight: 10
---
apiVersion: networking.istio.io/v1beta1
kind: DestinationRule
metadata:
  name: app-dr
  namespace: production
spec:
  host: app-service
  trafficPolicy:
    connectionPool:
      tcp:
        maxConnections: 100
      http:
        h2UpgradePolicy: DEFAULT
        http1MaxPendingRequests: 100
    outlierDetection:
      consecutive5xxErrors: 3
      interval: 30s
      baseEjectionTime: 30s
  subsets:
    - name: stable
      labels:
        version: v1
    - name: canary
      labels:
        version: v2
```

위 구성에서의 트래픽 흐름은 다음과 같다:

1. **K8GB (글로벌 레벨)**: 클라이언트 요청을 적절한 클러스터(EU/US)로 DNS 라우팅
2. **Istio Gateway (클러스터 레벨)**: 클러스터 인바운드 트래픽 수신

3. **Istio VirtualService (서비스 레벨)**: 카나리/Stable 버전으로 가중치 기반 분배
4. **Istio DestinationRule**: 연결 풀 관리, Outlier Detection으로 비정상 Pod 자동 제외

Linkerd 멀티 클러스터 연동

Linkerd는 Istio에 비해 경량화된 Service Mesh로, K8GB와의 연동 시 리소스 오버헤드를 최소화할 수 있다.

Linkerd의 멀티 클러스터 기능은 **Service Mirror** 패턴을 사용한다. 원격 클러스터의 서비스를 로컬 클러스터에 미러링하여, 마치 로컬 서비스처럼 접근할 수 있게 한다.

K8GB + Linkerd 조합의 구성은 다음과 같다:

1. **K8GB**: 글로벌 DNS 라우팅 (클러스터 선택)
2. **Linkerd Multi-Cluster**: 선택된 클러스터 내/간 서비스 통신에 mTLS 적용
3. **Linkerd Service Mirror**: 크로스 클러스터 서비스 디스커버리

이 조합은 Istio보다 낮은 리소스 사용량과 단순한 운영 모델을 제공하면서도, mTLS 기반 보안과 L7 메트릭을 확보할 수 있다.

5.3.2 GitOps/Helm/Kustomize 워크플로우 통합

K8GB의 선언적 CRD 기반 아키텍처는 GitOps 워크플로우와 자연스럽게 통합된다. Git 저장소를 단일 진실 공급원(Single Source of Truth)으로 사용하여 GSLB 정책을 관리할 수 있다.

Helm Values 기반 환경별 구성

동일한 Helm Chart를 사용하면서 환경별로 다른 values 파일을 적용하여 GSLB 설정을 관리한다.

production-eu/values.yaml

```yaml
# production-eu/values.yaml
k8gb:
  clusterGeoTag: "eu"
  extGslbClustersGeoTags: "us"
```

```
  dnsZone: "cloud.example.com"
  edgeDNSZone: "example.com"
  edgeDNSServers:
    - "ns1.example.com"
  reconcileRequeueSeconds: 30

externaldns:
  provider: aws
  aws:
    region: eu-west-1
    credentials:
      secretName: route53-credentials

gslb:
  strategy:
    type: failover
    dnsTtlSeconds: 30
    primaryGeoTag: "eu"
    splitBrainThresholdSeconds: 300

resources:
  controller:
    requests:
      cpu: 100m
      memory: 128Mi
    limits:
      cpu: 500m
      memory: 512Mi
  coredns:
    requests:
      cpu: 50m
      memory: 64Mi
    limits:
      cpu: 200m
      memory: 256Mi
```

staging-eu/values.yaml

```
# staging-eu/values.yaml
k8gb:
  clusterGeoTag: "eu-staging"
  extGslbClustersGeoTags: "us-staging"
  dnsZone: "cloud.staging.example.com"
  edgeDNSZone: "staging.example.com"
  edgeDNSServers:
    - "ns1.staging.example.com"
  reconcileRequeueSeconds: 60

externaldns:
  provider: aws
```

```
  aws:
    region: eu-west-1
    credentials:
      secretName: route53-credentials-staging

gslb:
  strategy:
    type: roundRobin
    dnsTtlSeconds: 60

resources:
  controller:
    requests:
      cpu: 50m
      memory: 64Mi
    limits:
      cpu: 200m
      memory: 256Mi
```

Kustomize 기반 구성

Kustomize를 사용하면 base 구성을 공유하면서 환경별 차이만 overlay로 관리할 수 있다.

디렉터리 구조

```
k8gb-config/
├── base/
│   ├── kustomization.yaml
│   ├── namespace.yaml
│   ├── gslb.yaml
│   └── ingress.yaml
├── overlays/
│   ├── production-eu/
│   │   ├── kustomization.yaml
│   │   └── gslb-patch.yaml
│   ├── production-us/
│   │   ├── kustomization.yaml
│   │   └── gslb-patch.yaml
```

```
│   └── staging/
│       ├── kustomization.yaml
│       └── gslb-patch.yaml
```

base/kustomization.yaml

```yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

namespace: production

resources:
  - namespace.yaml
  - gslb.yaml
  - ingress.yaml

commonLabels:
  app.kubernetes.io/managed-by: kustomize
  app.kubernetes.io/part-of: app-gslb
```

base/gslb.yaml

```yaml
apiVersion: k8gb.absa.oss/v1beta1
kind: Gslb
metadata:
  name: app-gslb
spec:
  ingress:
    ingressRef:
      name: app-ingress
  strategy:
    type: roundRobin
    dnsTtlSeconds: 30
```

overlays/production-eu/kustomization.yaml

```yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

bases:
  - ../../base

patchesStrategicMerge:
  - gslb-patch.yaml
```

```
commonAnnotations:
  environment: production
  region: eu-west-1
```

overlays/production-eu/gslb-patch.yaml

```
apiVersion: k8gb.absa.oss/v1beta1
kind: Gslb
metadata:
  name: app-gslb
spec:
  strategy:
    type: failover
    dnsTtlSeconds: 30
    primaryGeoTag: "eu"
    splitBrainThresholdSeconds: 300
```

ArgoCD Application 구성

ArgoCD를 사용하여 K8GB 설정을 GitOps 방식으로 자동 배포한다.

```
# argocd/application-k8gb-eu.yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: k8gb-production-eu
  namespace: argocd
  finalizers:
    - resources-finalizer.argocd.argoproj.io
spec:
  project: infrastructure
  source:
    repoURL: https://github.com/org/k8gb-config.git
    targetRevision: main
    path: overlays/production-eu
  destination:
    server: https://eu-cluster.example.com
    namespace: production
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
      allowEmpty: false
    syncOptions:
      - CreateNamespace=true
```

```
      - PrunePropagationPolicy=foreground
      - PruneLast=true
    retry:
      limit: 5
      backoff:
        duration: 5s
        factor: 2
        maxDuration: 3m0s
  ignoreDifferences:
    - group: k8gb.absa.oss
      kind: Gslb
      jsonPointers:
        - /status
```

Flux Kustomization 구성

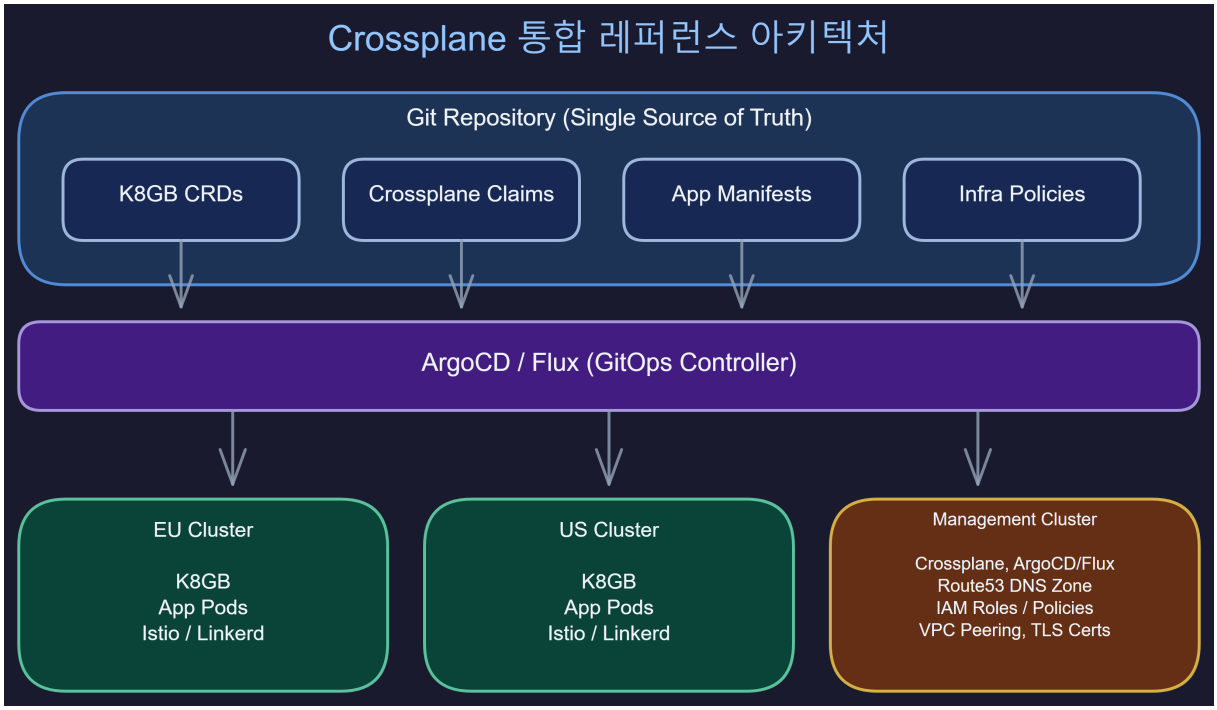
Flux를 사용하는 경우의 GitOps 구성 예시이다.

```
# flux/k8gb-production-eu.yaml
apiVersion: kustomize.toolkit.fluxcd.io/v1
kind: Kustomization
metadata:
  name: k8gb-production-eu
  namespace: flux-system
spec:
  interval: 5m0s
  path: ./overlays/production-eu
  prune: true
  sourceRef:
    kind: GitRepository
    name: k8gb-config
  targetNamespace: production
  healthChecks:
    - apiVersion: k8gb.absa.oss/v1beta1
      kind: Gslb
      name: app-gslb
      namespace: production
  timeout: 3m0s
  retryInterval: 1m0s
---
apiVersion: source.toolkit.fluxcd.io/v1
kind: GitRepository
metadata:
  name: k8gb-config
  namespace: flux-system
spec:
  interval: 1m0s
  url: https://github.com/org/k8gb-config.git
```

```
  ref:
    branch: main
  secretRef:
    name: git-credentials
```

Crossplane 통합 레퍼런스 아키텍처

Crossplane을 활용하면 K8GB 배포와 함께 Cloud DNS, 네트워크, IAM 등의 인프라를 통합적으로 관리할 수 있다.



[그림 10] 649

IaC(Infrastructure as Code) 모범 사례

K8GB를 포함한 GSLB 인프라를 IaC로 관리할 때의 핵심 모범 사례를 정리한다.

모범 사례	설명	도구/기법
선언적 관리	GSLB 정책을 명령형이 아닌 선언적 YAML로 정의하여 원하는 상태를 기술	K8GB Gslb CRD, Helm, Kustomize
버전 관리	모든 GSLB 구성 변경을 Git에 커밋하여 변경 이력을 추적하고 롤백 가능	Git, GitHub/GitLab

환경 분리	개발/스테이징/프로덕션 환경의 GSLB 설정을 분리하여 관리	Kustomize overlays, Helm values
자동 동기화	Git 변경 시 자동으로 클러스터에 적용하여 구성 드리프트 방지	ArgoCD, Flux
드리프트 감지	클러스터의 실제 상태와 Git의 선언적 상태 간 차이를 자동 감지	ArgoCD Sync Status, Flux Reconciliation
테스트	GSLB 정책 변경 전 자동화된 테스트로 정합성 검증	kubeconform, OPA/Gatekeeper, CI 파이프라인
재사용성	공통 GSLB 패턴을 모듈화하여 여러 서비스에서 재사용	Helm Library Chart, Kustomize components

---

5.3.3 CoreDNS-GSLB 플러그인과 헬스체크 연동

CoreDNS 플러그인 구성

K8GB에 내장된 CoreDNS는 GSLB 전용 플러그인을 통해 로드밸런싱 존을 관리한다. 다음은 CoreDNS의 Corefile 설정 예시이다.

```
# Corefile - K8GB CoreDNS 설정
cloud.example.com:5353 {
    k8gb {
        filter  cloud.example.com
        negttl  300
        loadbalance  roundrobin
    }

    forward . /etc/resolv.conf {
        max_concurrent 1000
    }

    errors
```

```
    log
    health {
        lameduck 5s
    }

    ready

    prometheus 0.0.0.0:9153

    cache 30
    loop
    reload
}
```

각 지시어의 역할은 다음과 같다:

- **k8gb**: K8GB GSLB 플러그인. `filter`로 관리할 존을 지정하고, `negttl`로 NXDOMAIN 응답의 캐시 시간을 설정한다.
- **forward**: GSLB 존에 해당하지 않는 쿼리를 업스트림 DNS로 전달한다.
- **health**: CoreDNS 자체의 헬스체크 엔드포인트를 제공한다.
- **prometheus**: Prometheus 형식의 메트릭을 노출한다.
- **cache**: DNS 응답 캐시를 설정한다.

백엔드 DNS 레코드 구성

k8gb Controller가 CoreDNS에 등록하는 DNS 레코드의 구조는 다음과 같다.

```yaml
# k8gb가 관리하는 DNS 레코드 (내부적으로 자동 생성)
# DNSEndpoint CRD 형태로 관리됨
apiVersion: externaldns.k8s.io/v1alpha1
kind: DNSEndpoint
metadata:
  name: app-gslb
```

```
  namespace: production
  labels:
    k8gb.absa.oss/dnstype: local
spec:
  endpoints:
    # A 레코드: 로컬 클러스터의 건강한 Ingress IP
    - dnsName: app.cloud.example.com
      recordTTL: 30
      recordType: A
      targets:
        - 10.0.1.100
        - 10.0.1.101

    # TXT 레코드: 클러스터 상태 정보 (크로스 클러스터 동기화용)
    - dnsName: app.cloud.example.com
      recordTTL: 30
      recordType: TXT
      targets:
        - "Heritage=k8gb,k8gb-strategy=failover"

    # NS 위임 레코드: 상위 DNS로의 위임 경로
    - dnsName: cloud.example.com
      recordTTL: 30
      recordType: NS
      targets:
        - gslb-ns-eu-cloud.example.com
```

Pod Health (Liveness/Readiness Probe) 연동

K8GB의 가장 강력한 특징 중 하나는 Kubernetes의 **Readiness Probe와 직접 연동**되는 헬스체크 메커니즘이다. 별도의 GSLB 헬스체크 엔드포인트를 구성하지 않아도, Pod의 Readiness 상태가 GSLB 라우팅 결정에 자동으로 반영된다.

헬스체크 연동 흐름

[그림 11] 667

Probe 설정 예시

K8GB와 연동되는 애플리케이션의 Probe 설정 예시이다. Readiness Probe가 GSLB 라우팅 결정의 기반이 되므로, 서비스의 실제 가용성을 정확하게 반영하도록 설정해야 한다.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app
  namespace: production
spec:
  replicas: 3
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
        - name: app
          image: myapp:v1.0.0
          ports:
            - containerPort: 8080

          # Liveness Probe: Pod 재시작 여부 판단
          # 애플리케이션 프로세스의 생존 여부를 확인
          livenessProbe:
            httpGet:
              path: /healthz
              port: 8080
            initialDelaySeconds: 15
            periodSeconds: 10
            timeoutSeconds: 5
            failureThreshold: 3
            successThreshold: 1

          # Readiness Probe: 트래픽 수신 가능 여부 판단
          # ★ K8GB가 이 상태를 기반으로 GSLB 라우팅을 결정
          readinessProbe:
            httpGet:
              path: /ready
              port: 8080
              httpHeaders:
                - name: X-Health-Check
                  value: "k8gb"
            initialDelaySeconds: 5
            periodSeconds: 5
```

```
    timeoutSeconds: 3
    failureThreshold: 2
    successThreshold: 1

  # Startup Probe: 초기 기동 완료 여부 판단
  # 기동이 느린 애플리케이션의 오탐 방지
  startupProbe:
    httpGet:
      path: /healthz
      port: 8080
    failureThreshold: 30
    periodSeconds: 10

  resources:
    requests:
      cpu: 100m
      memory: 128Mi
    limits:
      cpu: 500m
      memory: 512Mi
```

각 Probe의 역할과 K8GB와의 관계를 정리하면 다음과 같다.

Probe 유형	목적	K8GB 영향	실패 시 동작
Liveness Probe	Pod 생존 여부 확인	간접 영향 (Pod 재시작 → Readiness 변경)	kubelet이 컨테이너 재시작
Readiness Probe	트래픽 수신 가능 여부 확인	**직접 영향** (GSLB DNS 레코드에 반영)	Endpoints에서 제외 → K8GB가 감지 → DNS 업데이트
Startup Probe	초기 기동 완료 여부 확인	간접 영향 (기동 완료 전 Readiness 체크 억제)	기동 중 Liveness/Readiness 체크를 지연

설계 권장 사항: Readiness Probe는 단순한 프로세스 생존 확인이 아니라, **서비스가 실제로 요청을 처리할 수 있는 상태**인지를 확인해야 한다. 데이터베이스 연결 상태, 필수 외부 서비스 가용성, 캐시 워밍업 완료 여부 등을 종합적으로 검사하는 `/ready` 엔드포인트를 구현하는 것이 바람직하다. 이를 통해 K8GB가 실질적으로 요청 처리가 가능한 클러스터에만 트래픽을 라우팅하게 된다.

5장 요약

K8GB는 Kubernetes 환경에서 GSLB를 구현하기 위한 가장 적합한 오픈소스 솔루션이다. 핵심 내용을 요약하면 다음과 같다.

1. **아키텍처**: k8gb Controller + CoreDNS + ExternalDNS의 3개 컴포넌트로 구성되며, 각 클러스터에 독립적으로 배포되어 SPoF 없이 동작한다.
2. **트래픽 흐름**: DNS 컨트롤 플레인에서만 개입하며, 실제 데이터 트래픽은 클라이언트와 대상 클러스터 간 직접 통신으로 처리된다.
3. **로드밸런싱 전략**: failover(DR), roundRobin(균등 분산), weightRoundRobin(가중치 분산), geoip(위치 기반) 등 주요 전략을 모두 지원한다.
4. **Kubernetes 네이티브**: CRD를 통한 선언적 구성, Readiness Probe 기반 자동 헬스체크, Helm/Kustomize 배포를 통해 Kubernetes 생태계와 완전히 통합된다.
5. **생태계 연동**: Istio/Linkerd(Service Mesh), ArgoCD/Flux(GitOps), Crossplane(IaC) 등과 결합하여 완전한 클라우드 네이티브 멀티 클러스터 인프라를 구축할 수 있다.

> **다음 장 안내**: 6장에서는 GSLB 운영 및 모니터링에 관한 실무적 가이드를 다룬다. Prometheus/Grafana를 활용한 GSLB 메트릭 모니터링, 장애 시나리오별 트러블슈팅, 그리고 GSLB 운영 자동화 방안을 상세히 설명할 예정이다. # GSLB 이해하기 백서 — 6장, 7장 및 부록

6장. GSLB 제품 비교 및 K8GB 포지셔닝

엔터프라이즈 환경에서 GSLB 솔루션을 선택하는 일은 조직의 인프라 전략, 운영 역량, 비용 구조에 직접적인 영향을 미치는 중대한 의사결정이다. 본 장에서는 클라우드 GSLB 서비스, 온프레미스

GSLB 제품, 그리고 Kubernetes 네이티브 오픈소스 솔루션인 K8GB를 체계적으로 비교·분석하여 조직에 최적화된 GSLB 솔루션 선택을 위한 판단 기준을 제공한다.

6.1 클라우드 GSLB 서비스

클라우드 GSLB 서비스는 인프라 관리 부담을 최소화하면서 글로벌 트래픽 분산을 구현할 수 있는 관리형(Managed) 서비스이다. 각 클라우드 벤더는 자사 생태계에 최적화된 GSLB 기능을 제공하며, 독립 벤더는 멀티 클라우드 환경에 특화된 솔루션을 제공한다.

6.1.1 주요 클라우드 서비스 비교

1) AWS Route 53

AWS Route 53은 Amazon Web Services에서 제공하는 고가용성, 고확장성 DNS 웹 서비스로, GSLB 기능을 포함한 포괄적인 DNS 관리 플랫폼이다.

주요 특징

- **100% 가용성 SLA**: 업계 유일의 100% 가용성 서비스 수준 계약(SLA)을 보장하며, 이는 Route 53의 글로벌 Anycast 네트워크 아키텍처에 기반한다.
- **7가지 라우팅 정책**: Simple, Weighted, Latency-based, Failover, Geolocation, Geoproximity, Multivalue Answer의 7가지 라우팅 정책을 지원하여 다양한 트래픽 분산 시나리오에 대응한다.
- **Traffic Flow**: 시각적 편집기를 통해 복잡한 라우팅 로직을 설계할 수 있는 정책 관리 기능을 제공한다. 여러 라우팅 정책을 트리 형태로 조합하여 계층적 의사결정 로직을 구현할 수 있다.
- **AWS 생태계 완전 통합**: CloudFront, ALB/NLB, S3, Global Accelerator 등 AWS 서비스와 네이티브 통합을 지원한다. Alias 레코드를 통해 AWS 리소스에 대한 무료 DNS 쿼리를 제공한다.

- **Health Check**: HTTP, HTTPS, TCP 기반 헬스체크를 지원하며, CloudWatch 경보와 연동하여 애플리케이션 계층까지 감시 범위를 확장할 수 있다. 글로벌 8개 리전에서 헬스체크를 수행한다.

장점 – AWS 인프라와의 원활한 통합으로 최소 구성으로 GSLB 구현 가능 – Alias 레코드를 통한 AWS 리소스 대상 무료 DNS 쿼리 – Traffic Flow를 활용한 복잡한 라우팅 로직의 시각적 설계 – 매우 높은 안정성과 100% SLA 보장

단점 – AWS 외부 환경과의 통합 시 추가 구성 필요 – Traffic Flow 정책 사용 시 추가 비용 발생 (정책당 월 $50) – Geoproximity 라우팅의 편향(Bias) 설정이 직관적이지 않음 – 실시간 트래픽 기반 지능형 라우팅 부재 (RUM 미지원)

적합 환경: AWS를 주력 클라우드로 사용하는 조직, AWS 서비스 간 트래픽 라우팅이 핵심인 아키텍처, 100% SLA가 계약상 요구되는 환경

---

2) Azure Traffic Manager

Azure Traffic Manager는 Microsoft Azure에서 제공하는 DNS 기반 글로벌 트래픽 분산 서비스로, Azure 서비스뿐 아니라 외부 엔드포인트까지 포괄적으로 관리할 수 있다.

주요 특징

- **6가지 라우팅 방식**: Priority, Weighted, Performance, Geographic, Multivalue, Subnet의 6가지 라우팅 방식을 지원한다.
- **중첩 프로필(Nested Profile)**: Traffic Manager 프로필을 중첩하여 계층적 라우팅 구조를 설계할 수 있다. 이를 통해 리전 간 Failover와 리전 내 성능 기반 라우팅을 결합하는 복합 정책을 구현할 수 있다.
- **외부 엔드포인트 지원**: Azure 리소스 외에 온프레미스 서버, 타사 클라우드 서비스를 엔드포인트로 등록하여 하이브리드/멀티 클라우드 라우팅이 가능하다.
- **Real User Measurements(RUM)**: 실제 사용자의 네트워크 지연 데이터를 수집하여 Performance 라우팅의 정확도를 향상시킨다.

- **Traffic View**: 사용자 트래픽 패턴을 시각화하여 지리적 분포, 지연 시간 등을 분석할 수 있는 대시보드를 제공한다.

장점 – 외부 엔드포인트 지원으로 하이브리드/멀티 클라우드 시나리오에 유연하게 대응 – 중첩 프로필을 통한 정교한 계층적 라우팅 설계 가능 – Azure DevOps, ARM 템플릿과의 긴밀한 통합 – Real User Measurements 기반 성능 최적화

단점 – DNS 기반 라우팅만 지원하여 IP 수준 Anycast 라우팅 불가 – 헬스체크 주기가 최소 10초로 상대적으로 느림 – Azure Front Door와의 기능 중복으로 아키텍처 선택이 복잡 – 트래픽이 프록시를 경유하지 않으므로 DDoS 방어 기능 미제공

적합 환경: Azure 기반 인프라와 온프레미스 자원을 함께 운영하는 하이브리드 환경, Microsoft 기술 스택 중심 조직, 중첩 프로필을 활용한 복잡한 라우팅 설계가 필요한 환경

---

3) GCP Cloud DNS + Cloud Load Balancing

Google Cloud Platform은 Cloud DNS와 Cloud Load Balancing을 결합하여 DNS 기반 라우팅과 네트워크 계층 로드 밸런싱을 통합적으로 제공한다.

주요 특징

- **단일 Anycast IP**: 전 세계적으로 단일 Anycast IP 주소를 통해 트래픽을 수신하며, Google의 글로벌 네트워크를 활용하여 최적의 백엔드로 라우팅한다. 클라이언트는 별도의 DNS 라우팅 없이도 가장 가까운 PoP에 자동으로 연결된다.
- **202개 이상의 PoP**: Google의 글로벌 에지 네트워크에 202개 이상의 PoP(Point of Presence)를 보유하여 전 세계적으로 일관된 저지연 서비스를 제공한다.
- **네트워크 레벨 로드 밸런싱**: L4(TCP/UDP) 및 L7(HTTP/HTTPS) 로드 밸런싱을 글로벌 규모로 제공하며, 트래픽이 Google 백본 네트워크를 통해 최적 경로로 전달된다.
- **Cloud Armor 통합**: DDoS 방어, WAF, IP 화이트리스트/블랙리스트 기능을 Cloud Load Balancing과 네이티브 통합하여 보안 계층을 제공한다.
- **Traffic Director**: 서비스 메시 기반의 글로벌 트래픽 관리를 지원하며, Envoy 프록시와 통합하여 마이크로서비스 환경에서의 정밀한 트래픽 제어가 가능하다.

장점 – 단일 Anycast IP를 통한 아키텍처 단순화 및 DNS 전파 지연 문제 해소 – Google 글로벌 네트워크 활용으로 우수한 성능 – Cloud Armor를 통한 통합 보안 계층 제공 – GKE(Google Kubernetes Engine)와의 긴밀한 통합

단점 – GCP 생태계 밖에서의 활용이 제한적 – DNS 기반 GSLB 기능이 다른 클라우드 대비 상대적으로 제한적 – Cloud Load Balancing의 세밀한 가중치 제어가 복잡 – 타사 DNS와의 통합 시 추가 구성 필요

적합 환경: GCP를 주력 클라우드로 사용하는 조직, Anycast IP 기반 단순한 글로벌 라우팅이 필요한 서비스, GKE 기반 컨테이너 워크로드, 고성능 네트워크 라우팅이 핵심인 환경

---

4) Cloudflare Load Balancing

Cloudflare Load Balancing은 CDN, DDoS 방어, WAF와 통합된 벤더 중립적 글로벌 로드 밸런싱 서비스이다.

주요 특징

- **330개 이상의 PoP**: 전 세계 330개 이상의 데이터 센터에서 Anycast 기반으로 트래픽을 처리하며, 100개 이상 국가에 분포되어 있다.
- **CDN/DDoS/WAF 통합**: 로드 밸런싱, CDN 캐싱, DDoS 방어(Spectrum, Magic Transit), WAF(OWASP 규칙셋)가 단일 플랫폼에서 통합 운영된다.
- **벤더 중립**: 특정 클라우드에 종속되지 않으며, AWS, Azure, GCP, 온프레미스 등 모든 오리진 서버를 백엔드로 등록할 수 있다.
- **Steering Policies**: Random, Hash, Geo, Dynamic(Latency), Proximity, Least Outstanding Requests, Least Connections 등 다양한 스티어링 정책을 지원한다.
- **Workers 통합**: Cloudflare Workers(서버리스 컴퓨팅)를 활용하여 엣지에서 커스텀 라우팅 로직을 실행할 수 있다.

장점 – 보안(DDoS, WAF)과 성능(CDN)이 통합된 올인원 플랫폼 – 벤더 중립적 멀티 클라우드 로드 밸런싱 – Workers를 통한 엣지 컴퓨팅 기반 커스텀 라우팅 로직 – 직관적인 대시보드와 빠른 설정

단점 – 프록시 모드 필수로 오리진 IP가 Cloudflare를 경유 – Enterprise 플랜 필요 기능이 다수 (고급 DDoS, 전용 인증서 등) – 중국 본토 PoP는 Enterprise 플랜에서만 이용 가능 – DNS 전용 모드에서는 GSLB 기능이 제한적

적합 환경: 멀티 클라우드/하이브리드 환경에서 벤더 중립적 GSLB가 필요한 조직, 보안(DDoS/WAF)과 성능(CDN)을 통합 관리하려는 조직, 엣지 컴퓨팅 기반 커스텀 로직이 필요한 서비스

---

5) IBM NS1 Connect

IBM NS1 Connect(구 NS1)는 Filter Chain 아키텍처와 RUM(Real User Monitoring) 기반 지능형 라우팅을 핵심으로 하는 고성능 관리형 DNS 및 트래픽 스티어링 서비스이다.

주요 특징

- **Filter Chain**: 다수의 필터를 직렬로 연결하여 트래픽 라우팅 의사결정 파이프라인을 구성하는 고유한 아키텍처이다. Geo-IP 필터, Sticky 필터, Weighted Shuffle 필터, Up 필터 등을 자유롭게 조합하여 복잡한 라우팅 로직을 선언적으로 정의할 수 있다.
- **RUM 기반 지능형 라우팅**: 실제 최종 사용자의 성능 데이터(지연 시간, 가용성)를 실시간으로 수집·분석하여 최적의 엔드포인트로 라우팅한다. JavaScript 비콘을 통해 클라이언트 측 성능 데이터를 수집한다.
- **API 우선(API-First) 설계**: 모든 기능이 RESTful API를 통해 제공되며, Terraform, Ansible 등 IaC 도구와의 통합이 원활하다.
- **Pulsar**: 실시간 트래픽 텔레메트리 플랫폼으로, 글로벌 트래픽 흐름을 분석하고 라우팅 의사결정에 반영한다.
- **고성능 DNS**: 지능형 DNS 응답을 밀리초 이내에 처리하며, 대규모 쿼리 볼륨을 안정적으로 처리한다.

장점 – Filter Chain을 통한 업계 최고 수준의 라우팅 유연성 – RUM 기반 실시간 성능 최적화 라우팅 – API-First 설계로 자동화/IaC 통합에 최적 – 벤더 중립적 멀티 클라우드 트래픽 관리

단점 – IBM 인수 후 제품 로드맵 불확실성 – CDN, WAF 등 보안 기능 자체 미제공 (별도 솔루션 필요) – 소규모 조직에는 과도한 기능과 비용 – Filter Chain 설계 시 학습 곡선 존재

적합 환경: 복잡한 트래픽 라우팅 로직이 필요한 대규모 서비스, API 기반 자동화가 핵심인 DevOps/SRE 조직, RUM 기반 실시간 성능 최적화가 필요한 글로벌 서비스, 멀티 CDN 스티어링이 필요한 미디어/스트리밍 서비스

6.1.2 클라우드 GSLB 서비스 종합 비교

비교 항목	AWS Route 53	Azure Traffic Manager	GCP Cloud DNS + LB	Cloudflare	IBM NS1 Connect
라우팅 방식	7가지 (Simple, Weighted, Latency, Failover, Geolocation, Geoproximity, Multivalue)	6가지 (Priority, Weighted, Performance, Geographic, Multivalue, Subnet)	Anycast + Backend Service 가중치	7가지 이상 (Random, Hash, Geo, Dynamic, Proximity, LOR, LC)	Filter Chain 기반 무제한 조합
헬스체크	HTTP/HTTPS/TCP, 글로벌 8개 리전, CloudWatch 연동	HTTP/HTTPS/TCP, 최소 10초 주기	HTTP/HTTPS/TCP/SSL, 글로벌 분산	HTTP/HTTPS/TCP, 5초~300초 주기, 다중 리전	HTTP/HTTPS/TCP, RUM 기반 보강
Anycast	DNS 응답에 Anycast 적용	미지원 (DNS 기반만)	단일 Anycast IP (데이터 플레인)	DNS + 데이터 플레인 모두 Anycast	DNS 응답에 Anycast 적용
DDoS 방어	AWS Shield 연동 (별도 비용)	Azure DDoS Protection 연동 (별도 비용)	Cloud Armor 네이티브 통합	기본 제공 (모든 플랜)	미제공 (별도 솔루션 필요)
DNSSEC	지원	미지원	지원	지원	지원
멀티클라우드	제한적 (외부 IP 등록 가능)	지원 (외부 엔드포인트)	제한적 (하이브리드 NEG)	완전 지원 (벤더 중립)	완전 지원 (벤더 중립)
API/자동화	AWS SDK, CLI, CloudFormation, Terraform	Azure SDK, CLI, ARM, Terraform	gcloud CLI, Terraform	API, Terraform, Workers	API-First, Terraform, Ansible
SLA	100%	99.99%	99.99% (Cloud LB)	100% (Enterprise)	99.99%

글로벌 PoP	100개 이상 엣지 로케이션	Azure 리전 기반	202개 이상	330개 이상	30개 이상 PoP
WAF/보안 통합	AWS WAF 연동 (별도)	Azure WAF 연동 (별도)	Cloud Armor 통합	WAF 기본 내장	미제공
비용 구조	쿼리 수 + 호스팅 존 + 헬스체크	쿼리 수 + 헬스체크 + Traffic View	전달 규칙 + 트래픽	요청 수 + 오리진 수 + 헬스체크	쿼리 수 + 레코드 수 + RUM
최적 환경	AWS 단일 클라우드	Azure + 온프레미스 하이브리드	GCP 단일 클라우드	멀티클라우드 + 보안 통합	복잡한 라우팅 + 자동화

6.2 온프레미스 GSLB 제품

온프레미스 GSLB 제품은 조직의 데이터센터 내에 물리적 또는 가상 어플라이언스 형태로 배치되어 DNS 기반 글로벌 트래픽 관리를 수행한다. 규제 산업, 데이터 주권, 전용 하드웨어 성능이 요구되는 환경에서 여전히 핵심적인 역할을 담당한다.

6.2.1 주요 온프레미스 제품 비교

1) F5 BIG-IP DNS (구 BIG-IP GTM)

F5 BIG-IP DNS는 업계에서 가장 오랜 역사와 가장 넓은 시장 점유율을 보유한 엔터프라이즈 GSLB 솔루션이다.

주요 특징

- **최대 1억 QPS(Queries Per Second)**: 하드웨어 어플라이언스 기준 업계 최고 수준의 DNS 처리 성능을 제공한다.
- **iRules**: TCL 기반 프로그래머블 스크립팅 엔진으로, DNS 요청 및 응답에 대해 사실상 무제한의 커스텀 로직을 적용할 수 있다. HTTP 헤더, 클라이언트 속성, 외부 데이터소스를 참조하는 동적 라우팅 의사결정이 가능하다.

- **광범위한 라우팅 방식**: Round Robin, Ratio, Topology, Static Persist, Global Availability, QoS, Least Connections, Completion Rate, Kilobytes/Second, Return to DNS, LDNS Probe 등 12가지 이상의 로드 밸런싱 방식을 지원한다.
- **Wide-IP / Pool 구조**: Wide-IP(글로벌 가상 서버)와 Pool(엔드포인트 그룹)의 계층적 구조를 통해 정교한 트래픽 분산 정책을 설계할 수 있다.
- **BIG-IP LTM 통합**: 동일 플랫폼 내에서 SLB(Local Traffic Manager)와 GSLB(DNS)가 긴밀하게 통합되어 로컬-글로벌 헬스체크 정보를 실시간으로 공유한다.

장점 - 업계 최고의 성능과 안정성으로 미션 크리티컬 환경에 검증됨 - iRules를 통한 무한한 커스터마이징 가능성 - F5 LTM과의 통합을 통한 엔드투엔드 트래픽 관리 - 풍부한 레퍼런스와 기술 생태계

단점 - 업계 최고 수준의 라이선스 비용 (수천만 원~수억 원대) - iRules 기반 커스터마이징 시 높은 학습 곡선 - 하드웨어 어플라이언스 기반 구성 시 초기 투자 비용 과다 - Kubernetes 네이티브 통합이 상대적으로 미흡 (CIS 컨트롤러 별도 필요)

---

2) Citrix NetScaler ADC (구 Citrix ADC)

Citrix NetScaler ADC는 애플리케이션 전송 컨트롤러(ADC)에 GSLB 기능을 통합한 플랫폼으로, 특히 Citrix VDI(Virtual Desktop Infrastructure) 환경과의 시너지가 뛰어나다.

주요 특징

- **MEP(Metric Exchange Protocol)**: NetScaler 어플라이언스 간 사이트 메트릭(부하, 지연, 대역폭, 헬스 상태)을 실시간으로 교환하는 독자 프로토콜이다. 이를 통해 사이트 간 동적 부하 분산이 가능하다.
- **StyleBook**: 선언적 JSON 기반 구성 템플릿으로, 복잡한 GSLB 토폴로지를 코드로 정의하고 반복 배포할 수 있다. IaC(Infrastructure as Code) 패러다임에 부합한다.
- **통합 ADC**: SLB, GSLB, SSL 오프로드, 콘텐츠 스위칭, 압축, 캐싱, WAF(AppFirewall) 기능이 단일 플랫폼에 통합되어 있다.

- **VDI 환경 최적화**: Citrix Virtual Apps and Desktops(CVAD)와 네이티브 통합되어 가상 데스크톱 트래픽의 글로벌 분산에 최적화되어 있다.
- **ADNS(Authoritative DNS)**: 자체 권한 DNS 서버를 내장하여 외부 DNS 서버 없이 GSLB 서비스를 제공할 수 있다.

장점 – Citrix VDI/CVAD 환경과의 완벽한 통합 – MEP를 통한 사이트 간 실시간 메트릭 교환 – StyleBook 기반 선언적 구성 관리 – SLB/GSLB/보안 기능의 통합 플랫폼

단점 – Citrix 생태계 외부 환경에서의 활용 시 상대적 비효율 – MEP 프로토콜의 방화벽 통과 이슈 (포트 3011/3009) – 라이선스 모델의 복잡성 (Platinum, Enterprise, Standard 등급별 기능 차이) – 2024년 Cloud Software Group 인수 후 제품 방향성 불확실

---

3) A10 Networks Thunder ADC

A10 Networks Thunder ADC는 SLB와 GSLB 기능을 통합한 올인원 ADC 플랫폼으로, F5 대비 가성비를 핵심 차별점으로 내세운다.

주요 특징

- **간편 구성**: aFleX(TCL 기반 스크립팅)를 제공하지만, 대부분의 GSLB 시나리오를 GUI/CLI의 표준 구성만으로 구현할 수 있어 운영 복잡도가 낮다.
- **F5 대비 가성비**: 동등한 성능 대비 30~50% 낮은 총 소유 비용(TCO)을 제공한다는 것이 A10의 공식 포지셔닝이다.
- **SLB/GSLB 통합**: 단일 어플라이언스에서 로컬 로드 밸런싱과 글로벌 로드 밸런싱을 통합 관리할 수 있다.
- **Thunder TPS**: 별도의 DDoS 방어 전용 제품과 연동하여 대규모 공격에 대한 방어 체계를 구축할 수 있다.
- **Harmony Controller**: 멀티 클라우드 환경에서 Thunder ADC 인스턴스를 중앙 관리하는 컨트롤러를 제공한다.

장점 – F5 대비 합리적인 비용으로 유사한 기능 제공 – 직관적인 구성 인터페이스로 빠른 배포 가능 – SLB/GSLB 통합으로 관리 포인트 단일화 – aFleX 스크립팅을 통한 커스터마이징 지원

단점 – F5 대비 제한적인 커스터마이징 범위 – 대형 엔터프라이즈 레퍼런스가 상대적으로 부족 – 에코시스템(커뮤니티, 문서, 써드파티 통합)이 F5 대비 작음 – 일부 고급 GSLB 기능(Filter Chain 등)의 부재

---

4) Kemp LoadMaster GEO

Kemp LoadMaster GEO는 GSLB 시장에서 가장 낮은 진입 비용을 제공하는 솔루션으로, 중소규모 환경에 적합하다.

주요 특징

- **최저 비용**: 소프트웨어 라이선스 기준 연간 수백만 원대의 비용으로 GSLB를 구현할 수 있어, 중소기업이나 예산이 제한된 프로젝트에 적합하다.
- **빠른 배포**: 설치부터 GSLB 구성까지 수 시간 내에 완료할 수 있는 간결한 배포 프로세스를 제공한다.
- **Microsoft 환경 통합**: Active Directory, Exchange, SharePoint, Teams 등 Microsoft 인프라와의 통합에 최적화되어 있다.
- **LoadMaster 통합**: Kemp LoadMaster SLB 제품과 동일 관리 인터페이스에서 로컬/글로벌 로드 밸런싱을 통합 관리할 수 있다.
- **RESTful API**: 자동화를 위한 RESTful API를 제공하며, PowerShell 모듈을 통해 Microsoft 환경에서의 스크립팅이 용이하다.

장점 – 업계 최저 수준의 라이선스 비용 – 신속한 배포와 간단한 운영 – Microsoft 기술 스택과의 원활한 통합 – 직관적인 웹 기반 관리 인터페이스

단점 – 고급 GSLB 기능(RUM, Filter Chain, 복잡한 토폴로지) 미지원 – 대규모 글로벌 배포 시 확장성 제한 – 커스터마이징 범위가 매우 제한적 – 커뮤니티 및 써드파티 생태계가 작음

6.2.2 온프레미스 GSLB 제품 종합 비교

비교 항목	F5 BIG-IP DNS	Citrix NetScaler	A10 Thunder ADC	Kemp LoadMaster GEO
DNS 처리 성능	최대 1억 QPS	최대 수백만 QPS	최대 수천만 QPS	최대 수십만 QPS
GSLB 라우팅	12가지 이상	8가지 이상	6가지 이상	4가지 (RR, Weighted, Proximity, Failover)
통합 ADC 기능	LTM/ASM/APM 완전 통합	SLB/GSLB/WAF/SSL 통합	SLB/GSLB/DDoS 통합	SLB/GSLB 기본 통합
클라우드 지원	BIG-IP VE (AWS/Azure/GCP)	VPX/CPX (멀티 클라우드)	vThunder (멀티 클라우드)	VLM (AWS/Azure)
구성 난이도	높음 (iRules 학습 필요)	중상 (MEP/StyleBook 학습)	중 (표준 구성 중심)	낮음 (GUI 중심)
커스터마이징	최상 (iRules/iApps)	상 (정책 기반/StyleBook)	중 (aFleX 스크립팅)	하 (제한적 설정)
자동화 API	iControl REST, AS3	NITRO API, ADM	aXAPI (REST), Terraform	RESTful API, PowerShell
가격 수준	최상 (수천만~수억 원)	상 (수천만 원대)	중상 (F5 대비 30~50% 저렴)	최하 (수백만 원대)
안정성	최상 (업계 검증)	상 (금융권 다수 도입)	상 (안정적 운영 실적)	중 (중소 환경 검증)
기술지원	글로벌 24x7, 풍부한 문서	글로벌 24x7, Citrix 커뮤니티	글로벌 24x7	기본 지원, 제한적 문서
최적 환경	대형 금융/통신, 미션 크리티컬	Citrix VDI, 금융/공공	F5 대안, 비용 민감 엔터프라이즈	중소기업, Microsoft 환경

6.3 K8GB의 포지셔닝

6.3.1 상용 제품 대비 K8GB의 차별점

K8GB는 기존 상용 GSLB 솔루션과 근본적으로 다른 설계 철학을 기반으로 한다. 다음 표는 상용 GSLB 제품과 K8GB의 핵심 차별점을 비교한다.

비교 기준	상용 GSLB (클라우드/온프레미스)	K8GB
라이선스 비용	연간 수천만~수억 원 (클라우드: 사용량 과금)	Apache 2.0 오픈소스, 라이선스 비용 제로
운영 모델	전용 어플라이언스 또는 관리형 서비스	Kubernetes Operator로 클러스터 내 동작

구성 방식	GUI/CLI 기반 명령적(Imperative) 구성	CRD 기반 선언적(Declarative) 구성
벤더 종속	특정 벤더/클라우드에 종속	벤더 중립 (모든 클라우드/DNS 프로바이더 지원)
관리 아키텍처	전용 관리 클러스터 또는 중앙 집중 서버 필요	전용 관리 클러스터 불필요, SPoF(단일 장애점) 없음
GitOps/IaC 통합	제한적 (API 기반 커스텀 통합 필요)	Kubernetes 네이티브 → GitOps/IaC 워크플로우 자연 통합
확장 범위	DNS 전용 또는 ADC 통합	Kubernetes Ingress/Service와 직접 통합
장애 격리	중앙 관리 서버 장애 시 전체 영향	각 클러스터 독립 동작, 분산 아키텍처
업데이트/패치	벤더 릴리스 주기에 의존, 다운타임 가능	Helm 기반 롤링 업데이트, 무중단
커뮤니티	폐쇄적 벤더 생태계	CNCF 오픈소스 커뮤니티

CAPEX/OPEX 비교 관점에서의 K8GB 도입 비용 효과

K8GB 도입 시 비용 구조의 변화를 CAPEX(자본 지출)와 OPEX(운영 지출) 관점에서 분석하면 다음과 같다.

비용 항목	상용 GSLB	K8GB
CAPEX: 라이선스	수천만~수억 원 (초기 도입)	0원 (오픈소스)
CAPEX: 하드웨어	전용 어플라이언스 수천만 원 (온프레미스)	기존 K8s 클러스터 활용 (추가 인프라 불필요)
CAPEX: 구축 컨설팅	전문 엔지니어 구축 비용 (수천만 원)	내부 K8s 엔지니어가 직접 구축 가능
OPEX: 연간 유지보수	라이선스의 15~25% (수백만~수천만 원)	0원 (커뮤니티 지원)
OPEX: 클라우드 사용료	쿼리 수 + 헬스체크 + 기능별 과금	DNS 프로바이더 비용만 발생 (Route 53 호스팅 존 등)
OPEX: 운영 인력	전용 네트워크/ADC 운영 인력 필요	기존 K8s 운영팀이 겸임 가능
OPEX: 교육 비용	벤더별 자격증 및 교육 프로그램 (수백만 원/인)	Kubernetes 기술 역량 활용 (추가 교육 최소화)
3년 TCO 추정	1억~5억 원 이상	1천만~3천만 원 (DNS 호스팅 + 운영 인력)

참고: 상기 비용 추정은 중규모 엔터프라이즈(2~4개 사이트, 수십 만 QPS) 기준이며, 실제 비용은 조직 규모와 요구사항에 따라 크게 달라질 수 있다.

6.3.2 K8GB 적합 환경과 도입 판단 기준

K8GB가 최적인 환경

조건	상세 설명
Kubernetes 기반 인프라	워크로드가 Kubernetes 클러스터에서 실행되며, Ingress/Service 리소스를 활용하는 환경
멀티 클라우드 전략	2개 이상의 클라우드 프로바이더(AWS, Azure, GCP 등) 또는 클라우드 + 온프레미스를 동시에 운영하는 환경
DevOps 성숙 조직	GitOps, IaC, CI/CD 파이프라인이 확립되어 있으며, 선언적 인프라 관리 문화가 정착된 조직
비용 민감 환경	GSLB 라이선스 비용이 프로젝트 예산에서 큰 비중을 차지하여 오픈소스 대안이 필요한 환경
빠른 반복 배포	GSLB 구성을 코드로 관리하고, PR/MR 기반 변경 관리와 자동 배포가 필요한 환경
클라우드 네이티브 아키텍처	마이크로서비스, 서비스 메시, 컨테이너 오케스트레이션이 핵심인 아키텍처

상용 제품이 여전히 유리한 환경

조건	상세 설명
레거시 인프라	물리 서버, VM 기반 전통적 인프라로 Kubernetes 도입 계획이 없는 환경
규제 산업	금융, 통신, 의료 등 벤더 SLA 및 기술 지원 계약이 규제 준수의 필수 요건인 산업
벤더 지원 SLA 필수	24x7 전문 기술 지원, 긴급 패치 배포, 장애 시 에스컬레이션 체계가 반드시 필요한 환경
비-K8s 환경	워크로드가 Kubernetes 밖에서 실행되며, DNS 기반 GSLB만으로 충분한 환경
초고성능 요구	수천만~1억 QPS 이상의 DNS 처리 성능이 필요한 환경
복잡한 레거시 라우팅	iRules, Filter Chain 등 고도로 커스터마이징된 라우팅 로직이 이미 구축된 환경

클라우드 GSLB 선택 기준

상황	추천 제품	이유
AWS 단일 클라우드, 최고 안정성 필요	**AWS Route 53**	100% SLA, AWS 생태계 네이티브 통합, Alias 레코드 무료 쿼리
Azure + 온프레미스 하이브리드	**Azure Traffic Manager**	외부 엔드포인트 지원, 중첩 프로필, MS 생태계 통합
GCP 중심, Anycast 기반 단순 아키텍처	**GCP Cloud DNS + LB**	단일 Anycast IP, Google 네트워크 성능, GKE 통합
멀티 클라우드 + 보안 통합 필요	**Cloudflare**	벤더 중립, CDN/DDoS/WAF 통합, 330+ PoP
복잡한 라우팅 + API 자동화 중심	**IBM NS1 Connect**	Filter Chain, RUM 기반 라우팅, API-First 설계
K8s 기반 멀티 클라우드, 비용 민감	**K8GB**	오픈소스 무료, CRD 선언적 관리, GitOps 자연 통합

온프레미스 GSLB 선택 기준

상황	추천 제품	이유
대형 금융/통신, 미션 크리티컬	**F5 BIG-IP DNS**	최고 성능(1억 QPS), iRules 커스터마이징, 검증된 안정성
Citrix VDI 환경, 통합 ADC 필요	**Citrix NetScaler**	MEP 기반 사이트 간 동기화, CVAD 네이티브 통합
F5 대안, 비용 효율 엔터프라이즈	**A10 Thunder ADC**	F5 대비 30~50% TCO 절감, 간편 구성, SLB/GSLB 통합
중소기업, Microsoft 환경	**Kemp LoadMaster GEO**	최저 비용, 빠른 배포, MS 생태계 통합
K8s 네이티브 DR, 오픈소스 선호	**K8GB**	라이선스 비용 제로, CRD 기반, 벤더 중립

7장. GSLB 구축 사례와 실전 가이드

GSLB의 아키텍처 이론과 제품 비교를 넘어, 실제 구축 현장에서 겪는 도전과 이를 극복한 사례를 통해 실전적인 구축 역량을 배양하는 것이 본 장의 목적이다. 업종별 구축 패턴, 클라우드 레퍼런스 아키텍처, 그리고 실패 사례에서 도출한 교훈을 체계적으로 정리한다.

7.1 업종별 구축 사례

7.1.1 금융 (은행, 증권)

금융 산업은 GSLB 구축에서 가장 엄격한 요구사항을 제시하는 업종이다. 규제 준수, 데이터 주권, 극한의 가용성 요구가 결합되어 독특한 구축 패턴을 형성한다.

핵심 요구사항

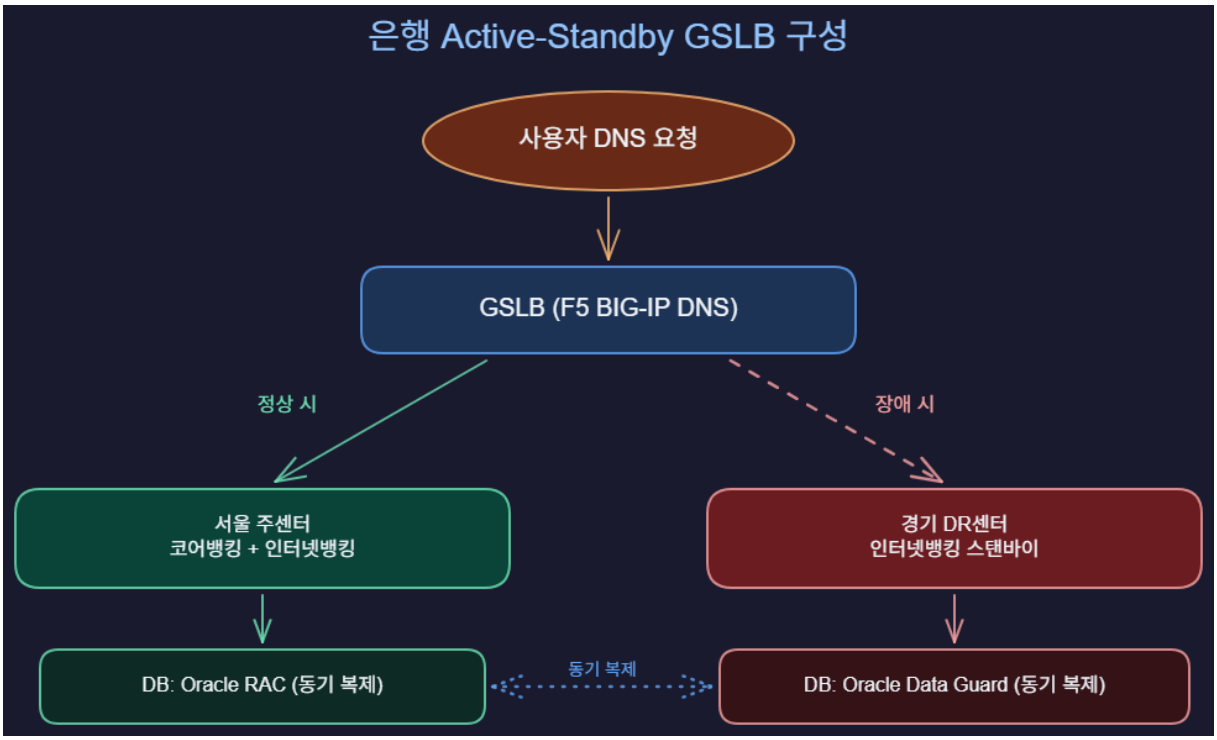
요구사항	상세 내용
규제 준수	금융감독원 전자금융감독규정, 전자금융거래법, 정보통신망법 준수. 재해복구 체계 의무화
데이터 주권	고객 금융 데이터의 국내 저장 의무, 국외 이전 시 규제 기관 사전 승인 필요
RTO/RPO	RTO(Recovery Time Objective) 2시간 이내, RPO(Recovery Point Objective) 0~수분 이내
감사 추적성	모든 트래픽 라우팅 변경 이력의 기록 및 보존, 규제 기관 감사 대응

전형적 구축 패턴

항목	구축 패턴
토폴로지	Active-Standby (주센터-DR센터), 최근 Active-Active 전환 추세
라우팅	Priority(Failover) 기반, 주센터 우선 라우팅
데이터 동기화	동기 복제(주센터~DR 50km 이내), 비동기 복제(원격지)
규제 고려	국내 IDC 한정, 금감원 DR 구축 기준 준수, 연 1회 이상 DR 훈련
헬스체크	L7 애플리케이션 계층, 코어뱅킹 시스템 트랜잭션 확인까지 포함

사례 1: 국내 시중은행 인터넷뱅킹

국내 A 시중은행은 인터넷뱅킹 서비스의 무중단 운영을 위해 서울(주센터)과 경기(DR센터) 간 Active-Standby GSLB 구성을 운영하고 있다.



- **GSLB 구성**: F5 BIG-IP DNS를 주센터와 DR센터에 이중화 배치하여 BIG-IP 간 동기화를 통해 구성 일관성을 보장한다.
- **헬스체크**: HTTP 요청으로 인터넷뱅킹 로그인 페이지 응답 확인에 더해, 코어뱅킹 시스템의 잔액 조회 테스트 트랜잭션까지 수행하여 실질적 서비스 가용성을 확인한다.
- **Failover 절차**: 주센터 장애 감지 시, GSLB는 DNS 응답을 DR센터 IP로 전환한다. TTL 60초 설정으로 최대 60초 이내 전환을 목표로 하며, 실제 전환 시간은 클라이언트 캐싱을 고려하여 2~5분으로 산정한다.
- **DR 훈련**: 연 2회 정기 DR 전환 훈련을 실시하여 Failover 절차의 실효성을 검증한다. 훈련 결과는 금감원에 보고한다.

사례 2: 증권사 HTS/MTS

B 증권사는 HTS(Home Trading System)와 MTS(Mobile Trading System)의 초저지연 트래픽 처리를 위해 GSLB를 구축하였다.

- **특이점**: 주식 거래 시스템은 TCP 기반 전용 프로토콜을 사용하므로, DNS 기반 GSLB와 함께 클라이언트 측 Failover 로직을 병행한다. HTS/MTS 애플리케이션에 주센터 IP와 DR센터 IP를 모두 내장하여, DNS 레벨 Failover와 독립적으로 클라이언트 자체 Failover를 수행한다.

- **장 중 무중단 요구**: 증시 거래 시간(09:00~15:30) 중에는 Failover가 발생하더라도 체결 주문의 정합성이 보장되어야 한다. 이를 위해 주문 큐잉 시스템과 GSLB를 연동하여, 장애 전환 시 미체결 주문의 재처리를 자동화한다.

7.1.2 이커머스

이커머스 산업은 글로벌 저지연, 트래픽 폭증 대응, 극한의 가용성, 비용 최적화라는 네 가지 요구사항이 동시에 작용하는 환경이다.

핵심 요구사항

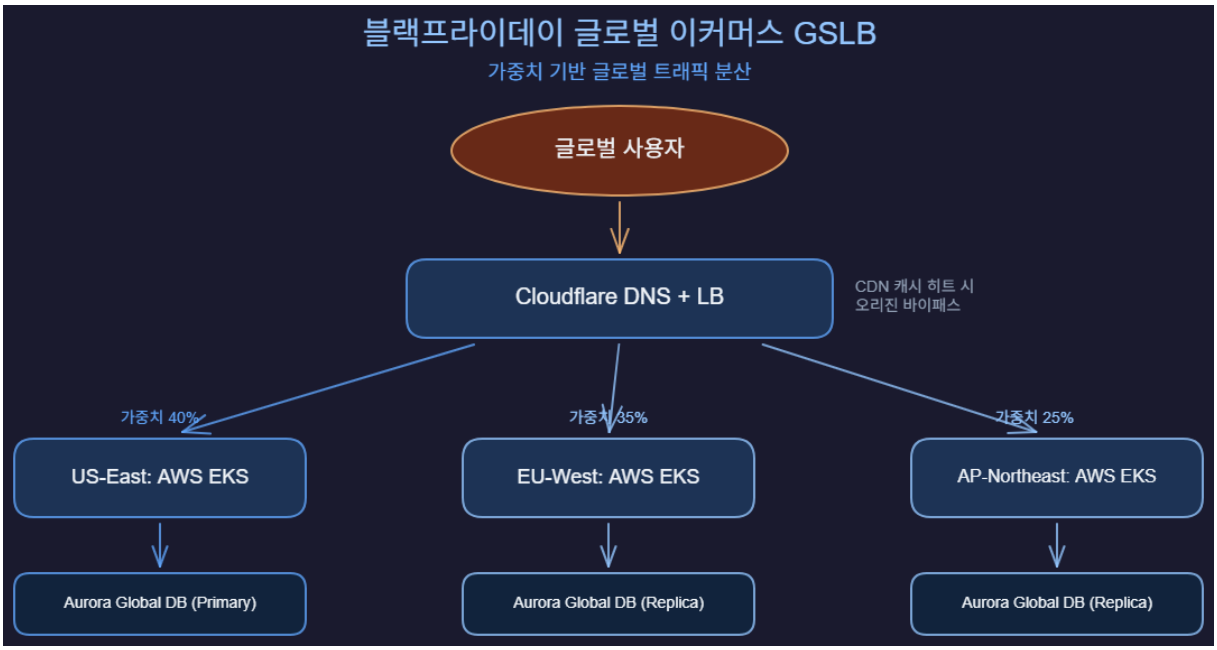
요구사항	상세 내용
글로벌 저지연	전 세계 사용자에게 200ms 이내의 페이지 로딩 시간 보장
트래픽 폭증	Black Friday, 광군제 등 이벤트 시 평시 대비 500~1,000% 트래픽 급증 대응
99.99% 가용성	연간 다운타임 52.6분 이내, 1분 다운타임 = 수억 원 매출 손실
비용 최적화	글로벌 인프라 운영 비용을 매출 대비 적정 비율로 유지

구축 패턴

항목	구축 패턴
토폴로지	Active-Active 멀티 리전 (최소 3개 리전: 북미, 유럽, 아시아)
라우팅	Latency-based + Geolocation 결합, CDN과 연동
데이터 동기화	비동기 복제 (최종 일관성), 상품/가격 데이터는 글로벌 캐시
헬스체크	L7 애플리케이션 (장바구니, 결제 API 확인), 합성 트랜잭션
확장 전략	오토스케일링 + GSLB 가중치 동적 조절

사례: 글로벌 패션 이커머스 — Black Friday 트래픽 500% 급증 대응

글로벌 패션 이커머스 기업 C사는 Black Friday 기간 중 평시 대비 500% 트래픽 급증에 대응하기 위해 다음과 같은 GSLB 전략을 수립하였다.



- **사전 준비 (D-30)**: 지난 3년간 Black Friday 트래픽 데이터를 분석하여 리전별 예상 트래픽을 산정하고, GSLB 가중치를 사전 조정한다. 오토스케일링 상한을 평시의 5배로 확장한다.
- **동적 가중치 조절 (D-Day)**: 실시간 트래픽 모니터링에 기반하여 GSLB 가중치를 동적으로 조절한다. 특정 리전의 응답 지연이 임계치를 초과하면 해당 리전의 가중치를 자동으로 감소시키고 여유 리전으로 트래픽을 전환한다.
- **CDN 연동**: 정적 자산(이미지, CSS, JS)은 CDN 캐시에서 직접 제공하여 오리진 서버 부하를 70% 이상 경감한다. GSLB는 동적 콘텐츠(API, 결제)에 대해서만 라우팅을 수행한다.

설계 핵심 고려사항

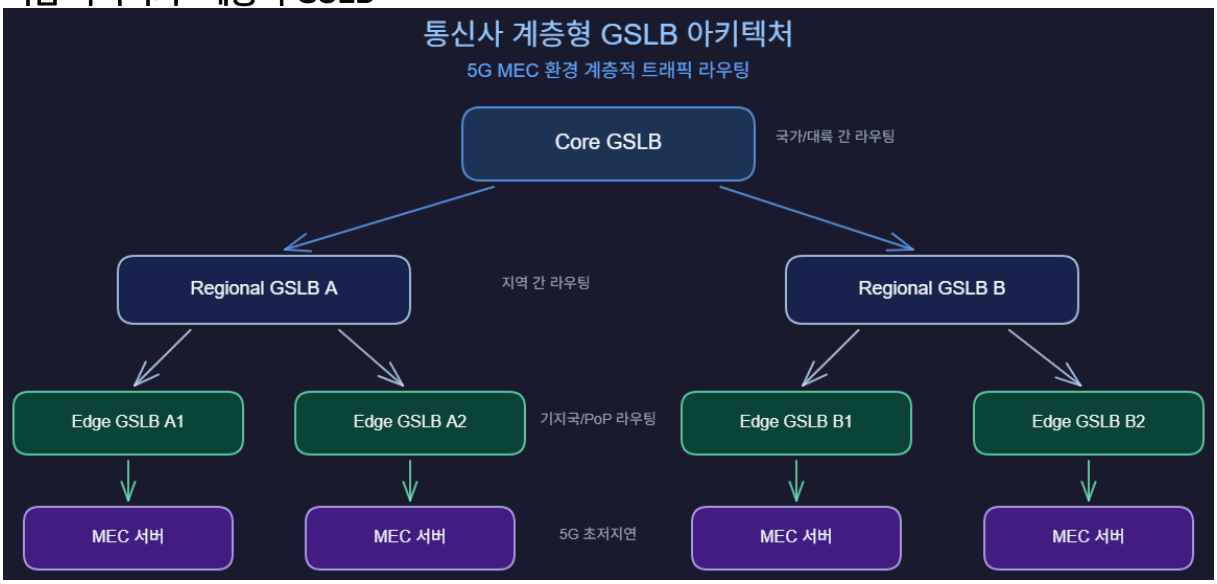
고려사항	설계 전략
세션 지속성	Redis 기반 글로벌 세션 스토어로 리전 간 세션 공유. Sticky Session은 GSLB 수준이 아닌 SLB 수준에서 처리
데이터 정합성	상품/재고는 최종 일관성(Eventual Consistency) 허용. 결제/주문은 강한 일관성(Strong Consistency) 보장
결제 PG 연동	결제 요청은 반드시 PG사 지정 리전에서 처리. GSLB의 Geolocation 라우팅으로 결제 트래픽을 PG 연동 리전으로 강제
GDPR Geo-Fencing	EU 사용자 데이터는 EU 리전에서만 처리. GSLB의 Geolocation 정책으로 EU 트래픽을 EU 리전으로 한정

7.1.3 통신 및 게임

통신

통신 산업은 가장 높은 가용성 요구(99.999%, Five Nines)와 대규모 인프라 운영이 특징이다.

핵심 아키텍처: 계층적 GSLB



- **계층적 GSLB (Core-Regional-Edge)**: Core GSLB는 국가/대륙 간 글로벌 라우팅을 담당하고, Regional GSLB는 국내 지역 간 라우팅을, Edge GSLB는 기지국/PoP 수준의 로컬 라우팅을 담당한다.
- **5G/MEC 연동**: 5G 네트워크의 MEC(Multi-access Edge Computing) 인프라와 GSLB를 연동하여, 사용자에게 가장 가까운 엣지 서버로 초저지연 라우팅을 수행한다.
- **99.999% SLA**: 연간 다운타임 5.26분 이내를 보장하기 위해 Active-Active 이중화, 계층적 Failover, 자동 복구 체계를 구축한다.

게임

게임 산업은 초저지연(20ms 이하)이 핵심 요구사항이며, 사용자 경험에 직접적인 영향을 미친다.

핵심 설계 원칙

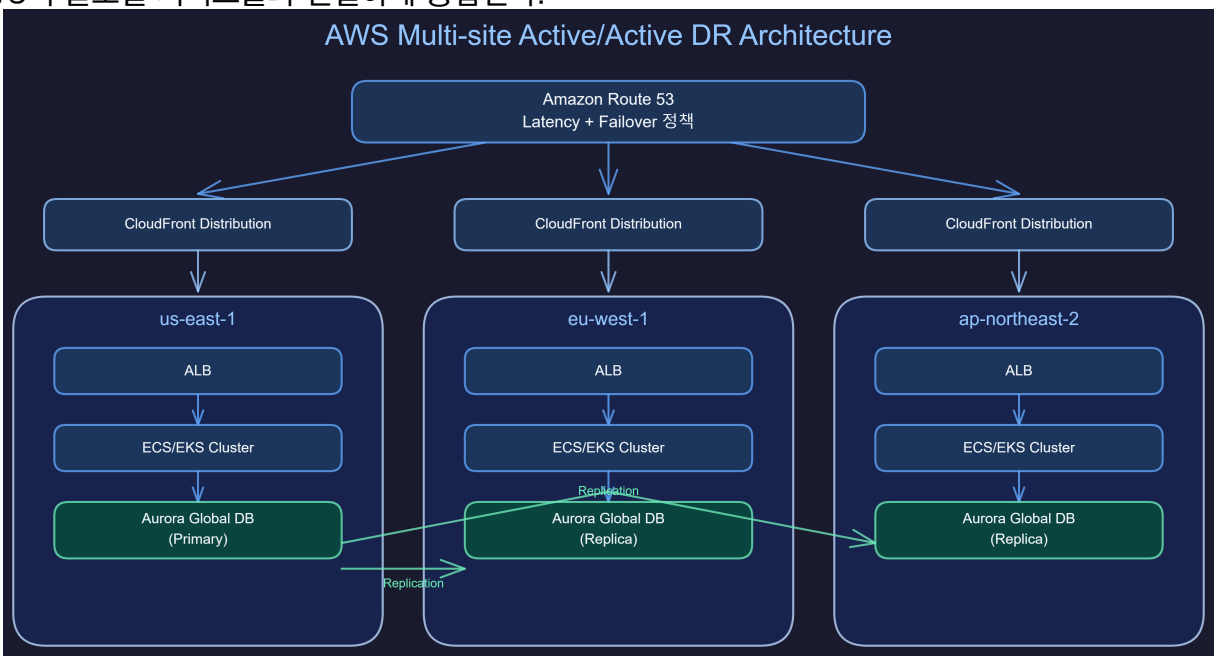
항목	설계 전략
초저지연 목표	서버-클라이언트 간 RTT 20ms 이하 (FPS, MOBA 장르)

라우팅 방식	Latency-based 라우팅이 기본, Geolocation 라우팅 보조
클라이언트 사이드 GSLB	게임 클라이언트가 직접 여러 서버의 지연 시간을 측정(Ping Test)하여 최적 서버를 선택하는 방식. DNS GSLB와 병행하여 이중 안전장치 구성
DDoS 방어	게임 서버는 DDoS 공격의 주요 대상. GSLB 앞단에 DDoS 스크러빙 서비스를 배치하고, 피공격 서버 감지 시 GSLB가 트래픽을 클린 서버로 전환
매치메이킹 연동	매치메이킹 시스템이 GSLB와 연동하여, 같은 리전 사용자끼리 매칭하여 지연 시간을 최소화

7.2 클라우드 레퍼런스 아키텍처

7.2.1 AWS Multi-site Active/Active DR 아키텍처

AWS 환경에서의 글로벌 Active/Active DR 아키텍처는 Route 53을 GSLB 핵심으로 활용하며, AWS의 글로벌 서비스들과 긴밀하게 통합된다.



핵심 구성 요소

구성 요소	역할	주요 기능
Route 53	GSLB (DNS 기반 글로벌 라우팅)	Latency + Failover 결합 정책, Health Check, Traffic Flow

Global Accelerator	네트워크 계층 글로벌 가속	Anycast IP, AWS 백본 네트워크 활용, 즉각적 Failover
CloudFront	CDN + 엣지 캐싱	정적 콘텐츠 캐싱, Lambda@Edge, 오리진 장애 시 캐시 제공
Aurora Global DB	글로벌 관계형 데이터베이스	1초 이내 크로스 리전 복제, 1분 이내 리전 간 Failover
DynamoDB Global Tables	글로벌 NoSQL 데이터베이스	멀티 리전 Active-Active 복제, 밀리초 단위 지연

Route 53 라우팅 정책 비교

정책	동작 방식	최적 사용 시나리오
Simple	단일 값 반환	단일 리소스 라우팅
Weighted	가중치 비율로 분산	A/B 테스트, 단계적 마이그레이션
Latency	최저 지연 리전으로 라우팅	글로벌 Active-Active 서비스
Failover	Primary 장애 시 Secondary로 전환	Active-Standby DR
Geolocation	사용자 위치 기반 라우팅	규제 준수, 콘텐츠 지역화
Geoproximity	지리적 근접성 + Bias 기반	리전 간 트래픽 비율 조절
Multivalue Answer	다중 건강한 IP 반환	간이 로드 밸런싱

Route 53 vs Global Accelerator 비교

비교 항목	Route 53	Global Accelerator
동작 계층	DNS (L7)	네트워크 (L3/L4)
IP 유형	도메인 기반 (DNS 응답)	고정 Anycast IP 2개
Failover 속도	DNS TTL 의존 (수십 초~수 분)	즉각적 (수 초 이내)
트래픽 경로	인터넷 경유	AWS 글로벌 네트워크 경유
적합 시나리오	DNS 기반 글로벌 라우팅	즉각적 Failover, 고정 IP 필요, 비-HTTP 프로토콜
비용	쿼리 수 기반 (저비용)	고정비 + 트래픽 기반 (상대적 고비용)
결합 사용	Route 53 → Global Accelerator → ALB 구조로 DNS 라우팅 + 네트워크 가속을 결합	

데이터 계층 전략

데이터 서비스	복제 방식	RPO	Failover 시간	적합 데이터 유형
Aurora Global DB	비동기 (1초 이내 지연)	~1초	~1분 (리전 승격)	관계형 데이터 (주문, 사용자)
DynamoDB Global Tables	Active-Active 멀티 리전	~0초 (최종 일관성)	즉각적 (로컬 리전 사용)	세션, 장바구니, 설정
ElastiCache Global Datastore	비동기 크로스 리전 복제	~수초	~수십 초	캐시, 실시간 데이터
S3 Cross-Region Replication	비동기 객체 복제	~수분	즉각적 (리전별 버킷)	정적 자산, 로그, 백업

7.3 구축 시 실패 사례와 교훈

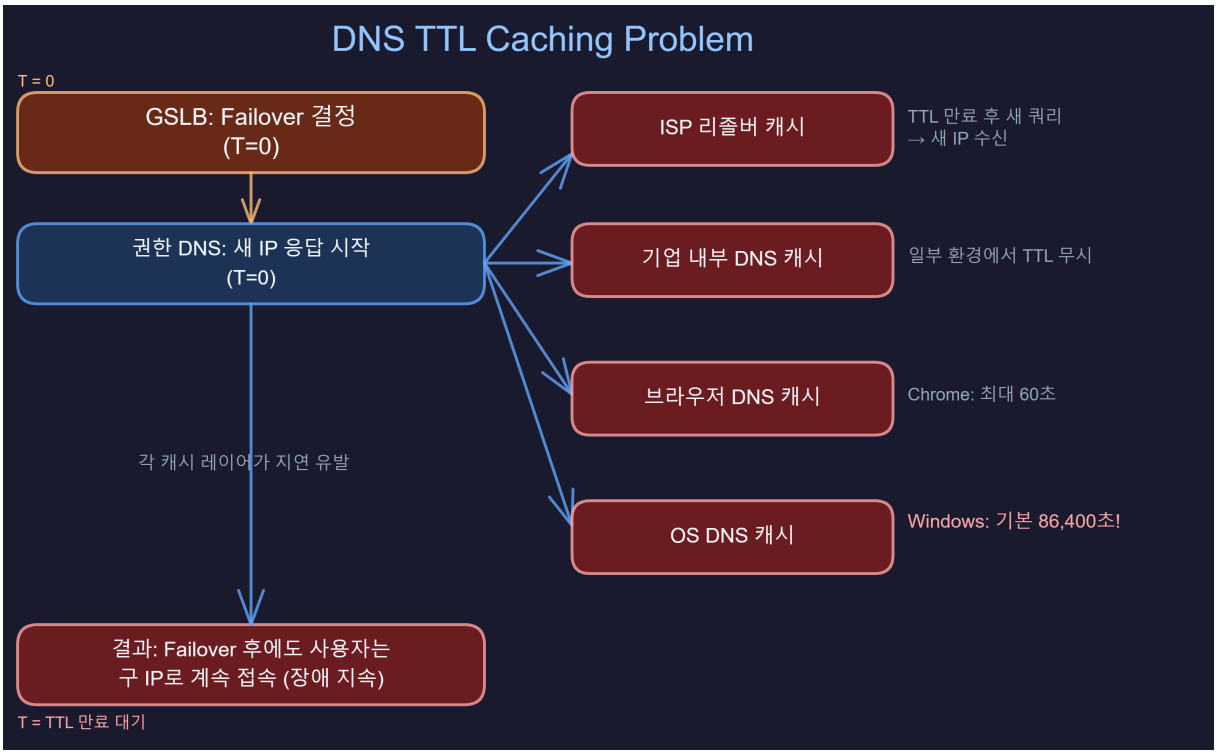
GSLB 구축 과정에서 발생하는 실패는 대부분 예측 가능한 패턴을 따른다. 본 절에서는 현장에서 빈번하게 발생하는 실패 패턴과 이를 방지하기 위한 구체적인 교훈을 정리한다.

7.3.1 흔한 실패 패턴과 회피 방안

실패 패턴 1: DNS TTL 캐싱 문제

DNS TTL 캐싱은 GSLB Failover의 가장 큰 장애물이다. GSLB가 DNS 응답을 변경하더라도, 클라이언트와 중간 리졸버의 캐시가 만료되기 전까지는 실제 트래픽 전환이 이루어지지 않는다.

근본 원인 다이어그램



실패 시나리오

주센터 장애 발생 후 GSLB가 즉시 Failover를 결정했음에도, TTL 300초로 설정된 DNS 레코드로 인해 최대 5분간 사용자 트래픽이 장애 서버로 계속 전달되었다. 더 심각한 경우, 일부 ISP 리졸버가 TTL을 무시하고 자체적으로 더 긴 캐시 기간을 적용하여 최대 30분까지 전환이 지연되었다.

교훈 및 대응

교훈	대응 방안
TTL은 짧게, 그러나 과도하지 않게	운영 환경: TTL 30~60초 권장. 너무 짧으면 DNS 쿼리 폭증
ISP 리졸버의 TTL 무시 대비	다수의 권한 네임서버 배포, Anycast DNS 활용
클라이언트 캐시 고려	브라우저/OS 캐시를 고려한 Failover 시간 산정
TTL 사전 단축 (Pre-staging)	계획된 유지보수 전 TTL을 30초로 사전 단축 후 기존 TTL 시간만큼 대기
Global Accelerator 병행	DNS 기반 Failover의 한계를 보완하기 위해 네트워크 계층 Failover 병행

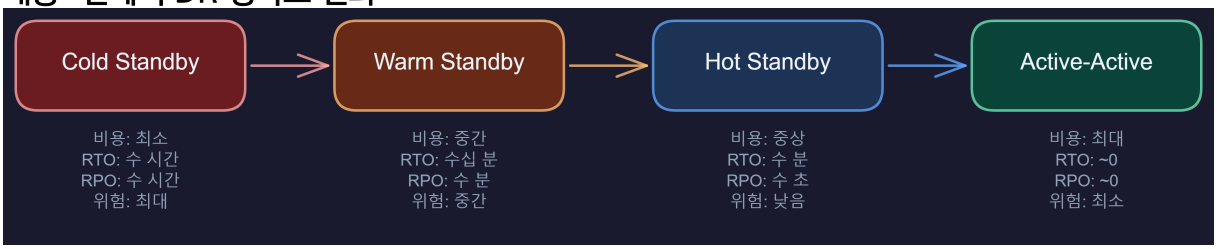
실패 패턴 2: Cold Standby 실패

Cold Standby(또는 Pilot Light) DR 구성에서 실제 Failover 시 DR 사이트가 기대한 서비스 수준을 제공하지 못하는 경우가 빈번하다.

실패 요인

실패 요인	상세 설명
용량 부족	DR 사이트가 주센터 트래픽의 100%를 처리할 수 있는 용량으로 프로비저닝되지 않음. 비용 절감을 위해 최소 사양으로 운영하다 실제 장애 시 과부하 발생
환경 드리프트(Drift)	주센터의 구성 변경(패치, 설정 변경, 새 컴포넌트 추가)이 DR에 반영되지 않아 환경 불일치 발생
데이터 갭	Cold Standby의 데이터가 최신 상태가 아니거나, 복제 중단/지연으로 데이터 손실 발생
네트워크 경로 미검증	DR 사이트로의 실제 트래픽 경로(방화벽 규칙, VPN 터널, CDN 설정)가 사전에 검증되지 않음

대응: 단계적 DR 성숙도 진화



- **Cold → Warm**: 최소한의 인프라를 상시 실행 상태로 유지하고, 정기적 데이터 동기화를 수행한다.
- **Warm → Hot**: 모든 인프라를 실행 상태로 유지하되 트래픽은 수신하지 않으며, 실시간 데이터 복제를 적용한다.
- **Hot → Active-Active**: 양 사이트가 모두 실 트래픽을 처리하며, 상시 검증된 상태를 유지한다.

실패 패턴 3: MEP 트래픽 폭증

MEP(Metric Exchange Protocol)는 GSLB 사이트 간 상태 정보를 교환하는 핵심 프로토콜이지만, 잘못된 구성 시 네트워크 장애를 유발할 수 있다.

실패 요인과 대응

실패 요인	상세 설명	대응 방안
연결 끊김 오탐	WAN 구간의 일시적 지연/패킷 손실을 MEP 연결 끊김으로 오탐하여 불필요한 Failover 트리거	MEP 타임아웃 값을 네트워크 환경에 맞게 조정 (기본값보다 여유 있게 설정)
Source IP 불일치	NAT, 프록시 환경에서 MEP 패킷의 Source IP가 변경되어 인증 실패	MEP 트래픽을 NAT 바이패스 경로로 구성, Source IP 허용 목록 확장
SSL 인증서 문제	MEP 암호화 통신 시 인증서 만료/불일치로 MEP 연결 중단	인증서 자동 갱신 체계 구축, 만료 30일 전 알림 설정
Full-Mesh 폭증	N개 사이트의 Full-Mesh MEP 구성 시 N*(N-1)/2개의 연결이 발생하여 대규모 환경에서 관리 부담 증가	Hub-Spoke 토폴로지로 MEP 연결 수 최소화, 또는 BigIP 그룹 당 대표 노드만 MEP 참여

실패 패턴 4: 헬스체크 오탐/미탐

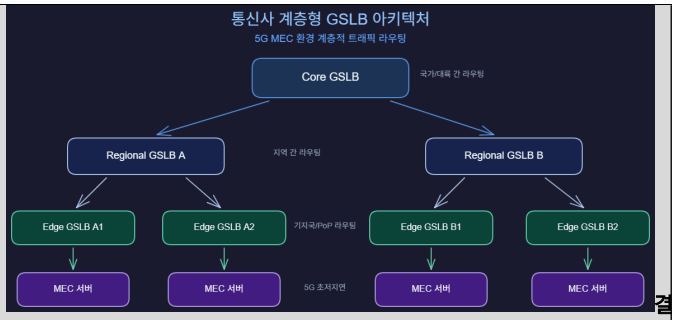
GSLB 헬스체크의 정확성은 전체 시스템의 신뢰성을 좌우하는 핵심 요소이다. 오탐(False Positive)과 미탐(False Negative) 모두 심각한 서비스 장애를 유발한다.

오탐(False Positive) — 정상을 비정상으로 판단

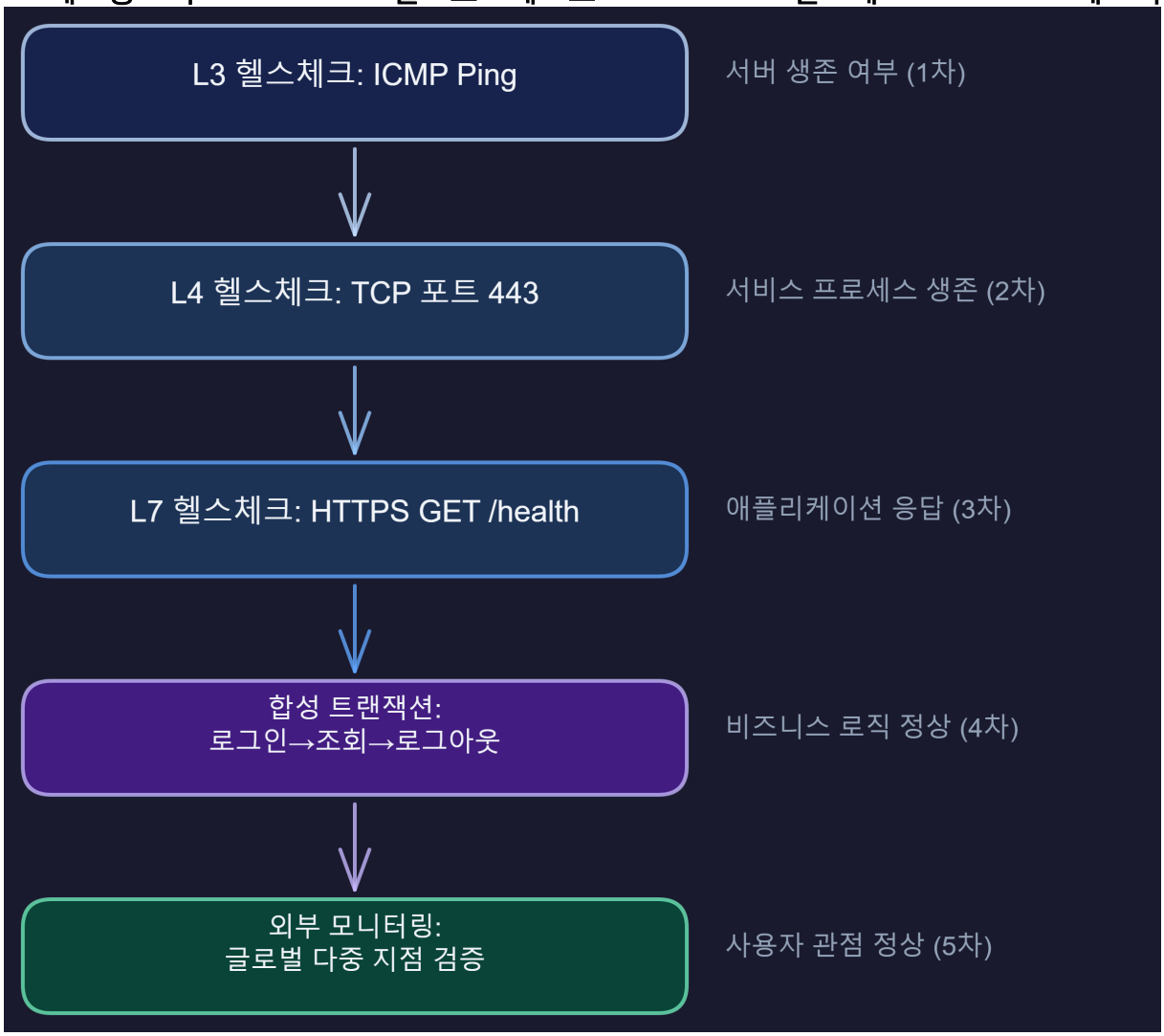
원인	결과
헬스체크 엔드포인트의 일시적 응답 지연	정상 서버가 다운으로 판정되어 불필요한 Failover 발생
헬스체크 경로의 네트워크 이슈	서비스는 정상이나 헬스체크 패킷이 도달하지 못해 장애로 판정
헬스체크 임계치의 과도한 민감성	1~2회 실패만으로 즉시 다운 판정, 일시적 부하에도 과잉 반응

미탐(False Negative) — 비정상을 정상으로 판단

원인	과
헬스체크 엔드포인트만 정상 (부분 장애)	웹 서버는 응답하나 DB 연결 실패 상태를 감지하지 못함
헬스체크 주기가 너무 길음	장애 발생 후 감지까지 수 분의 공백 발생
L3/L4 헬스체크만 수행	TCP 포트 오픈 여부만 확인하여 애플리케이션 계층 장애를 놓침



계 층 적 헬 스 체 크 설 계 예 시



실패 패턴 5: 데이터 불일치

Active-Active 구성에서 데이터 불일치는 가장 해결하기 어려운 문제 중 하나이다.

불일치 유형

불일치 유형	상세 설명
Split-Brain	네트워크 단절로 양 사이트가 각각 자신이 Primary라고 판단하여 독립적으로 쓰기 연산을 수행. 네트워크 복구 후 데이터 충돌 발생
복제 지연	비동기 복제 환경에서 Primary 사이트의 최신 데이터가 DR 사이트에 도달하기 전에 Failover 발생. 최근 데이터 손실
세션 불일치	사용자 세션이 특정 사이트에 저장된 상태에서 GSLB가 다른 사이트로 라우팅. 로그인 세션 끊김 또는 장바구니 초기화
캐시 불일치	사이트 간 캐시 데이터의 버전 차이로 사용자에게 불일치한 데이터 제공

Active-Active 데이터 정합성 결정 매트릭스

데이터 유형	일관성 모델	충돌 해결 전략	복제 방식
사용자 인증	강한 일관성	Primary 사이트 기준	동기 복제
금융 트랜잭션	강한 일관성	분산 트랜잭션 (2PC/Saga)	동기 복제
상품 카탈로그	최종 일관성	LWW(Last Writer Wins)	비동기 복제
사용자 세션	최종 일관성	타임스탬프 기반 병합	비동기 복제 (Redis Global)
장바구니	최종 일관성	CRDT 기반 자동 병합	비동기 복제
로그/분석	최종 일관성	Append-Only 병합	비동기 복제

7.3.2 GSLB 구축 전 체크리스트

GSLB 구축 프로젝트를 시작하기 전 반드시 확인해야 할 항목을 체크리스트 형태로 정리한다.

아키텍처 설계

☐ GSLB 토폴로지 결정 (Active-Standby / Active-Active / Multi-site)

☐ 라우팅 정책 결정 (Failover / Weighted / Latency / Geolocation / 복합)

☐ DNS TTL 전략 수립 (운영 TTL, 비상 시 TTL, Pre-staging 절차)

☐ 데이터 동기화 전략 수립 (동기/비동기 복제, 일관성 모델)

☐ SPoF(단일 장애점) 분석 및 제거 계획

헬스체크 설계

☐ 헬스체크 계층 설계 (L3/L4/L7/합성 트랜잭션)

☐ 헬스체크 엔드포인트 구현 (의존 서비스 상태 포함)

☐ 헬스체크 주기/타임아웃/재시도 횟수 설정

☐ 오탐/미탐 방지를 위한 임계치 튜닝

☐ 헬스체크 모니터링 및 알림 체계 구축

네트워크 및 보안

☐ DNS 위임(Delegation) 구성 확인

☐ 방화벽 규칙 사전 검증 (헬스체크 경로, MEP 포트, 데이터 복제 경로)

☐ DNSSEC 적용 여부 결정

☐ DDoS 방어 계층 설계

☐ SSL/TLS 인증서 관리 전략 (멀티 사이트 인증서, 자동 갱신)

데이터 계층

☐ 데이터 복제 방식 결정 (동기/비동기/CRDT)

☐ RPO 목표에 따른 복제 지연 검증

☐ Split-Brain 방지 전략 (쿼럼, Fencing)

☐ 세션 관리 전략 (글로벌 세션 스토어, Sticky Session)

☐ 캐시 일관성 전략

운영 및 테스트

☐ Failover/Failback 절차서 작성

☐ DR 전환 훈련 계획 (분기별/반기별)

☐ 모니터링/관측성 체계 구축 (메트릭, 로그, 트레이스)

- ☐ 롤백 절차 수립
- ☐ 용량 계획 및 성능 테스트

규제 및 컴플라이언스

- ☐ 데이터 주권 요구사항 확인 (데이터 저장 위치 제한)
- ☐ 규제 기관 보고 요건 확인 (DR 훈련 결과, 장애 보고)
- ☐ 감사 로깅 요구사항 확인 (라우팅 변경 이력 보존)
- ☐ SLA 요구사항 정의 (RTO, RPO, 가용성 목표)

7.3.3 K8GB를 활용한 Kubernetes 기반 Active-Active DR 구축 실전 가이드

K8GB는 Kubernetes 환경에서 GSLB를 구현하기 위한 오픈소스 솔루션으로, CRD(Custom Resource Definition) 기반의 선언적 구성을 통해 멀티 클러스터 트래픽 관리를 제공한다.

Gslb CRD 기본 설정 예시

```yaml
apiVersion: k8gb.absa.oss/v1beta1
kind: Gslb
metadata:
  name: app-failover
  namespace: production
spec:
  ingress:
    ingressClassName: nginx
    rules:
      - host: app.example.com
        http:
          paths:
            - path: /
              pathType: Prefix
              backend:
                service:
                  name: app-service
                  port:
                    number: 80
  strategy:
    type: failover
    primaryGeoTag: us-east-1
    dnsTtlSeconds: 30
    splitBrainThresholdSeconds: 300
```

헬스체크 튜닝 가이드

K8GB는 Kubernetes의 Readiness Probe를 기반으로 헬스체크를 수행한다. 다음은 헬스체크 최적화를 위한 권장 설정이다.

```yaml
# 애플리케이션 Pod의 Readiness Probe 설정
readinessProbe:
  httpGet:
    path: /health
    port: 8080
  initialDelaySeconds: 5
  periodSeconds: 5          # 5초 간격으로 확인
  failureThreshold: 3       # 3회 연속 실패 시 Not Ready
  successThreshold: 1       # 1회 성공 시 Ready
  timeoutSeconds: 3         # 응답 타임아웃 3초
```

튜닝 항목	권장 값	설명
periodSeconds	5~10초	감지 속도와 부하 사이의 균형. 미션 크리티컬 환경은 5초
failureThreshold	3~5회	오탐 방지. 일시적 지연을 고려하여 3회 이상
timeoutSeconds	2~5초	네트워크 지연을 고려하되, 너무 길면 감지 지연
splitBrainThresholdSeconds	300~600초	양 클러스터 간 통신 단절 시 Split-Brain 판정까지의 대기 시간

DNS TTL 최적화 팁

시나리오	권장 TTL	이유
일상 운영	30초	Failover 시 빠른 전환과 DNS 쿼리 부하의 균형
계획된 유지보수 전	10~15초	유지보수 시작 2시간 전 TTL 단축하여 캐시 갱신 유도
장애 복구 후	300초	안정화 확인 후 TTL을 원래 수준으로 복원하여 DNS 부하 경감
테스트/개발 환경	5~10초	빠른 변경 반영을 위해 짧은 TTL 설정

결론

본 백서의 6장과 7장을 통해 GSLB 솔루션의 비교 분석 및 실전 구축 가이드를 포괄적으로 살펴보았다. 클라우드 GSLB 서비스(AWS Route 53, Azure Traffic Manager, GCP Cloud DNS, Cloudflare, IBM NS1 Connect), 온프레미스 GSLB 제품(F5 BIG-IP DNS, Citrix NetScaler, A10 Thunder ADC, Kemp LoadMaster GEO), 그리고 Kubernetes 네이티브 오픈소스 솔루션인 K8GB는 각각 고유한 강점과 적합 환경을 보유하고 있다. GSLB 솔루션의 선택은 단순한 기능 비교가 아닌, 조직의 인프라 전략, 운영 역량, 비용 구조, 규제 환경을 종합적으로 고려한 의사결정이어야 한다. 특히 Kubernetes 기반 클라우드 네이티브 환경으로의 전환이 가속화되는 현재, K8GB와 같은 오픈소스 GSLB 솔루션은 비용 효율성, 벤더 중립성, GitOps 통합이라는 차별적 가치를 제공하며, 이는 향후 GSLB 시장의 중요한 변화 축이 될 것이다.

부록

부록 A. 용어 정의

용어	정의
GSLB	Global Server Load Balancing. 지리적으로 분산된 서버 간 글로벌 트래픽을 분산하는 기술
SLB	Server Load Balancing. 단일 사이트 내에서 서버 간 트래픽을 분산하는 로컬 로드 밸런싱
ADC	Application Delivery Controller. SLB, SSL 오프로드, 캐싱, 압축, WAF 등 애플리케이션 전송 최적화 기능을 통합한 장비
DNS	Domain Name System. 도메인 이름을 IP 주소로 변환하는 인터넷 인프라
ADNS	Authoritative DNS Server. 특정 도메인에 대한 최종 권한을 보유한 DNS 서버
TTL	Time To Live. DNS 레코드의 캐시 유효 시간 (초 단위)
Anycast	동일한 IP 주소를 여러 네트워크 위치에서 광고하여 가장 가까운 노드로 트래픽을 전달하는 라우팅 기술
BGP	Border Gateway Protocol. 자율 시스템(AS) 간 경로 정보를 교환하는 인터넷 라우팅 프로토콜

MEP	Metric Exchange Protocol. GSLB 사이트 간 메트릭 정보(부하, 지연, 상태)를 교환하는 프로토콜
EDNS	Extension Mechanisms for DNS. DNS 프로토콜의 확장 메커니즘 (RFC 6891)
ECS	EDNS Client Subnet. DNS 쿼리에 클라이언트 서브넷 정보를 포함하는 확장 (RFC 7871)
CDN	Content Delivery Network. 콘텐츠를 지리적으로 분산된 엣지 서버에 캐싱하여 전송 성능을 향상시키는 네트워크
GeoIP	클라이언트 IP 주소로부터 지리적 위치(국가, 도시, 좌표)를 추정하는 기술
PoP	Point of Presence. 네트워크 서비스가 제공되는 물리적 접속 지점
RTO	Recovery Time Objective. 장애 발생 후 서비스 복구까지의 목표 시간
RPO	Recovery Point Objective. 장애 발생 시 허용 가능한 데이터 손실 범위 (시간 단위)
DR	Disaster Recovery. 재해 발생 시 IT 서비스를 복구하기 위한 계획, 절차, 인프라
HA	High Availability. 서비스의 지속적인 운영을 보장하는 설계 원칙 및 아키텍처
SPoF	Single Point of Failure. 장애 발생 시 전체 시스템의 동작을 중단시키는 단일 컴포넌트
CRD	Custom Resource Definition. Kubernetes API를 확장하여 사용자 정의 리소스를 정의하는 메커니즘
K8GB	Kubernetes Global Balancer. CNCF 오픈소스 프로젝트로, Kubernetes 네이티브 GSLB 솔루션
CoreDNS	Kubernetes 기본 DNS 서버. 플러그인 체인 아키텍처 기반의 유연한 DNS 서버
ExternalDNS	Kubernetes 리소스를 기반으로 외부 DNS 프로바이더의 레코드를 자동 관리하는 도구
Ingress	Kubernetes 클러스터 외부로부터의 HTTP/HTTPS 트래픽을 클러스터 내부 서비스로 라우팅하는 리소스
Service Mesh	마이크로서비스 간 통신을 관리하는 인프라 계층 (Istio, Linkerd 등)
GitOps	Git 저장소를 Single Source of Truth로 사용하여 인프라와 애플리케이션을 선언적으로 관리하는 운영 모델
IaC	Infrastructure as Code. 인프라를 코드로 정의하고 버전 관리하는 관행 (Terraform, Pulumi 등)
CRDT	Conflict-free Replicated Data Type. 분산 시스템에서 충돌 없이 병합 가능한 데이터 구조
mTLS	Mutual TLS. 클라이언트와 서버가 상호 인증서를 검증하는 양방향 TLS 인증

부록 B. K8GB Gslb CRD 레퍼런스

B.1 Gslb CRD 필드 명세

필드 경로	타입	필수	설명
apiVersion	string	예	API 버전. k8gb.absa.oss/v1beta1
kind	string	예	리소스 종류. Gslb
metadata.name	string	예	Gslb 리소스 이름
metadata.namespace	string	예	Gslb 리소스가 생성될 네임스페이스
spec.ingress	object	예	Ingress 사양. 표준 Kubernetes Ingress 스펙을 따름
spec.ingress.ingressClassName	string	아니오	Ingress Controller 클래스 이름 (예: nginx)
spec.ingress.rules	array	예	Ingress 라우팅 규칙 목록
spec.ingress.rules[].host	string	예	GSLB 대상 호스트명 (FQDN)
spec.ingress.rules[].http.paths	array	예	HTTP 경로별 라우팅 규칙
spec.ingress.rules[].http.paths[].path	string	예	URL 경로
spec.ingress.rules[].http.paths[].pathType	string	예	경로 매칭 방식 (Prefix, Exact, ImplementationSpecific)
spec.ingress.rules[].http.paths[].backend.service.name	string	예	백엔드 서비스 이름
spec.ingress.rules[].http.paths[].backend.service.port.number	integer	예	백엔드 서비스 포트
spec.strategy	object	예	GSLB 전략 설정
spec.strategy.type	string	예	전략 유형: failover, roundRobin, weightRoundRobin, geoip
spec.strategy.primaryGeoTag	string	조건부	Primary 클러스터의 Geo 태그 (failover 전략 시 필수)
spec.strategy.dnsTtlSeconds	integer	아니오	DNS TTL (초). 기본값: 30
spec.strategy.splitBrainThresholdSeconds	integer	아니오	Split-Brain 판정 대기 시간 (초). 기본값: 300

spec.strategy.weight	map	조건부	클러스터별 가중치 (weightRoundRobin 전략 시 필수). 예: {us-east-1: 80, eu-west-1: 20}
status.geoTag	string	–	현재 클러스터의 Geo 태그 (자동 설정)
status.serviceHealth	map	–	서비스 헬스 상태 (자동 갱신)
status.healthyRecords	map	–	건강한 DNS 레코드 목록 (자동 갱신)

B.2 전략별 YAML 예시

Failover 전략

Primary 클러스터에 장애가 발생하면 Secondary 클러스터로 자동 전환하는 전략이다.

```yaml
apiVersion: k8gb.absa.oss/v1beta1
kind: Gslb
metadata:
  name: app-failover
  namespace: production
spec:
  ingress:
    ingressClassName: nginx
    rules:
      - host: app.example.com
        http:
          paths:
            - path: /
              pathType: Prefix
              backend:
                service:
                  name: frontend-service
                  port:
                    number: 80
  strategy:
    type: failover
    primaryGeoTag: us-east-1        # Primary 클러스터 Geo 태그
    dnsTtlSeconds: 30               # DNS TTL 30초
    splitBrainThresholdSeconds: 300  # Split-Brain 판정 5분
```

Round Robin 전략

모든 건강한 클러스터에 동일한 비율로 트래픽을 분산하는 전략이다.

```yaml
apiVersion: k8gb.absa.oss/v1beta1
kind: Gslb
metadata:
  name: app-roundrobin
  namespace: production
spec:
  ingress:
    ingressClassName: nginx
    rules:
      - host: app.example.com
        http:
          paths:
            - path: /
              pathType: Prefix
              backend:
                service:
                  name: api-service
                  port:
                    number: 8080
  strategy:
    type: roundRobin
    dnsTtlSeconds: 30
    splitBrainThresholdSeconds: 300
```

Weighted Round Robin 전략

클러스터별 가중치에 따라 트래픽을 비율적으로 분산하는 전략이다.

```yaml
apiVersion: k8gb.absa.oss/v1beta1
kind: Gslb
metadata:
  name: app-weighted
  namespace: production
spec:
  ingress:
    ingressClassName: nginx
    rules:
      - host: app.example.com
        http:
          paths:
            - path: /
              pathType: Prefix
```

```
            backend:
              service:
                name: web-service
                port:
                  number: 80
  strategy:
    type: weightRoundRobin
    dnsTtlSeconds: 30
    splitBrainThresholdSeconds: 300
    weight:
      us-east-1: 70    # US 클러스터에 70% 트래픽
      eu-west-1: 30     # EU 클러스터에 30% 트래픽
```

GeoIP 전략

클라이언트의 지리적 위치(IP 기반)에 따라 가장 가까운 클러스터로 라우팅하는 전략이다.

```
apiVersion: k8gb.absa.oss/v1beta1
kind: Gslb
metadata:
  name: app-geoip
  namespace: production
spec:
  ingress:
    ingressClassName: nginx
    rules:
      - host: app.example.com
        http:
          paths:
            - path: /
              pathType: Prefix
              backend:
                service:
                  name: regional-service
                  port:
                    number: 80
  strategy:
    type: geoip
    dnsTtlSeconds: 30
    splitBrainThresholdSeconds: 300
```

참고: GeoIP 전략 사용 시 각 클러스터의 Geo 태그가 올바르게 설정되어 있어야 하며, ExternalDNS와 연동되는 DNS 프로바이더가 GeoIP 기반 레코드를 지원해야 한다.

부록 C. GSLB 제품 비교 종합표

아래 표는 본 백서에서 다룬 모든 GSLB 솔루션을 하나의 매트릭스로 종합 비교한 것이다.

비교 항목	AWS Route 53	Azure Traffic Manager	GCP Cloud DNS + LB	Cloudflare	IBM NS1 Connect	F5 BIG-IP DNS	Citrix NetScaler	A10 Thunder ADC	Kemp LoadMaster GEO	K8GB
유형	클라우드 관리형	클라우드 관리형	클라우드 관리형	클라우드 관리형	클라우드 관리형	온프레미스/가상	온프레미스/가상	온프레미스/가상	온프레미스/가상	오픈소스 (K8s Operator)
라우팅 방식 수	7가지	6가지	Anycast + 가중치	7가지 이상	Filter Chain (무제한)	12가지 이상	8가지 이상	6가지 이상	4가지	4가지 (failover, roundRobin, weightRoundRobin, geoip)
헬스체크	HTTP/HTTPS/TCP + CloudWatch	HTTP/HTTPS/TCP	HTTP/HTTPS/TCP	HTTP/HTTPS/TCP/SSL 다중 리전	HTTP/HTTPS/TCP + RUM	L3/L4/L7 + iRules 커스텀	L3/L4/L7 + MEP 연동	L3/L4/L7	L3/L4/L7 기본	K8s Readiness Probe 연동
Anycast	DNS 레벨	미지원	DNS + 데이터 플레인	DNS + 데이터 플레인	DNS 레벨	미지원 (별도 구성)	미지원	미지원	미지원	미지원 (DNS 프로바이더 의존)
DDoS 방어	Shield 연동 (별도)	DDoS Protection 연동 (별도)	Cloud Armor 통합	기본 내장	미제공	미제공 (별도)	미제공 (별도)	Thunder TPS 연동	미제공	미제공 (별도)
멀티클라우드	제한적	지원 (외부 엔드포인트)	제한적	완전 지원	완전 지원	지원 (VE)	지원 (VPX/CPX)	지원 (vThunder)	제한적 (VLM)	완전 지원 (벤더 중립)
Kubernetes 지원	EKS 연동	AKS 연동	GKE 네이티브	비-네이티브	비-네이티브	CIS 컨트롤러 (별도)	Ingress Controller (별도)	제한적	미지원	Kubernetes 네이티브 (CRD)

오픈소스	아니오	아니오	아니오	아니오	아니오	아니오	아니오	아니오	아니오	예 (Apache 2.0)
비용	사용량 과금	사용량 과금	사용량 과금	사용량 과금	사용량 과금	매우 높음	높음	중상	낮음	무료 (DNS 호스팅 비용만)
최적 환경	AWS 단일 클라우드	Azure + 하이브리드	GCP 단일 클라우드	멀티클라우드 + 보안	복잡한 라우팅 + 자동화	대형 금융/통신	Citrix VDI 환경	F5 대안 엔터프라이즈	중소기업/MS 환경	K8s 멀티클라우드

본 백서는 GSLB 기술의 이해와 실전 적용을 위한 기술 참조 문서로 작성되었습니다.

Contact Us

hello@cncf.co.kr

02-469-5426

www.cncf.co.kr

CNF Blog

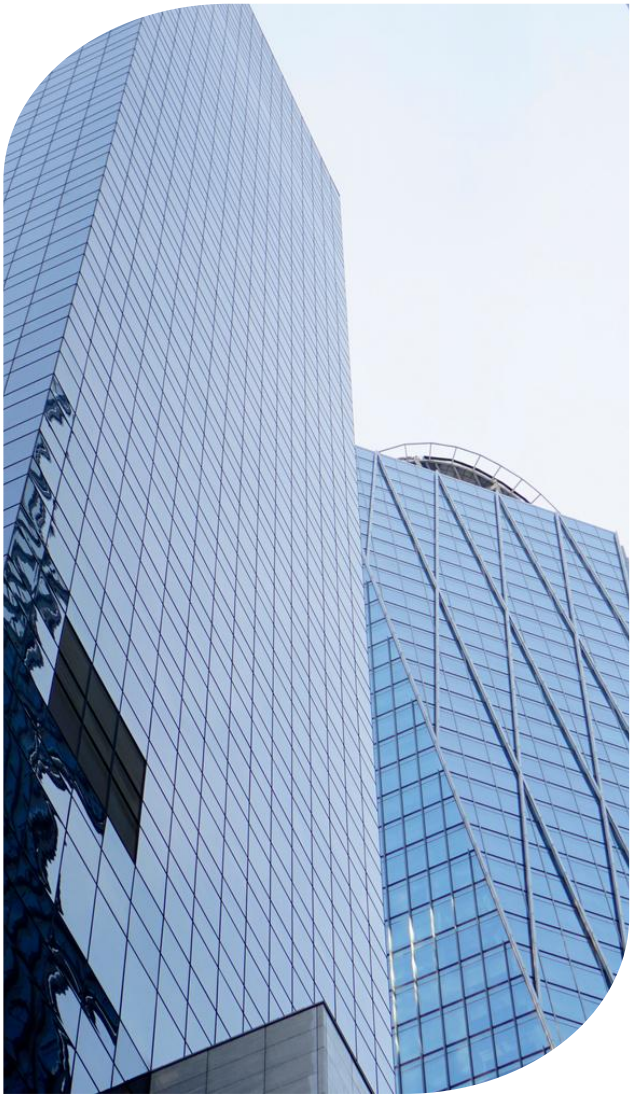
다양한 콘텐츠와 전문 지식을 통해
더 나은 경험을 제공합니다.

CNF eBook

이제 나도 클라우드 네이티브 전문가
쿠버네티스 구축부터 운영 완전 정복

CNF Resource

Community Solution의 최신 정보와
유용한 자료를 만나보세요.



씨엔에프 | CNF

전화 : (02) 469 - 5426
팩스 : (02) 469 - 7247
메일 : hello@cncf.co.kr