

WatchDucks

설치가 필요 없는 트래픽 분석 서비스

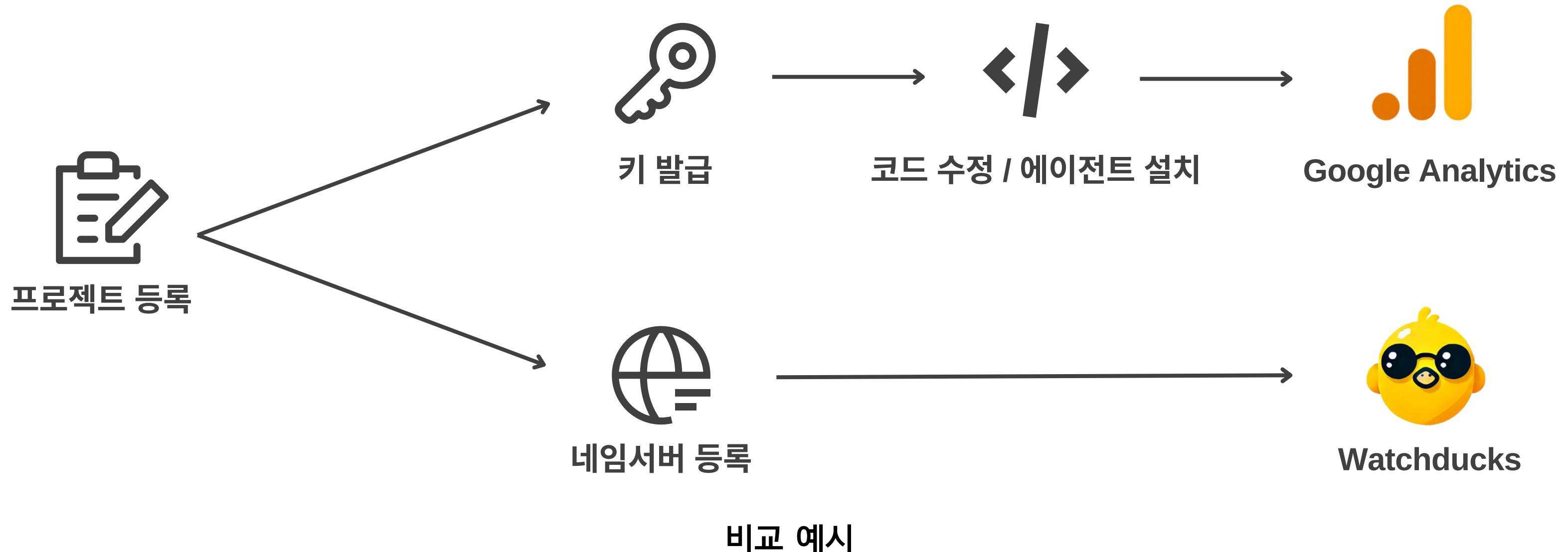
부스트캠프 9기 프로젝트



프로젝트 소개

코드 수정 없이 사용 가능한 트래픽 분석 서비스

- 트래픽 분석 서비스를 사용하기 위해서는 에이전트 설치 및 코드 추가가 불가피함
- Google Analytics의 경우 키를 발급받고 해당 서비스의 API를 코드 추가를 통해 연동해야함
- Watchducks는 별도의 코드 수정 없이 DNS 설정만으로 서비스를 사용 가능



프로젝트 소개

Register

그룹 프로젝트를 등록해 주세요

Watchdogs ✓
www.watchdogs.site ✓
192.0.0.1 ✓
1001lily@naver.com ✓
9기

등록하기

메인으로

구분 호스트명 구분 호스트명
1차 ns1 [REDACTED] 2차 ns2 [REDACTED]

소유자 인증
소유자 인증
작용 목록으로

※ 네임서버는 IP(숫자)를 제외한 호스트명만 입력합니다. (예. ns.gabia.co.kr)
※ 네임서버 값은 복사해서 입력하는 경우, 공란이 포함되지 않도록 주의하시기 바랍니다.
※ 각 도메인마다 네임서버 최소/최대 값이 다릅니다.
※ 따라서 여러 개의 네임 서버를 입력하여도 도메인의 허용된 개수에 따라 적용됩니다.
※ 가비아 네임서버는 DNSSEC을 지원하지 않습니다. DNSSEC이 설정된 도메인을 가비아 네임서버로 변경하시는 경우, 웹사이트 접속(리졸빙)이 제한됩니다.

등록 절차

- 프로젝트명, 도메인, 아이피 주소를 입력합니다.
- 등록 완료 후 표시된 네임 서버를 복사합니다.
- 상용 DNS (ex. Cloudflare, Gabia)의 가이드에 따라 네임 서버와 IP를 등록해 주세요.

Gabia 예시



간단한 등록 절차

- 도메인 네임, 사용할 서버의 IP 주소, 이메일 정보만으로 간단하게 등록 가능

DNS 호스팅 업체의 네임서버 변경

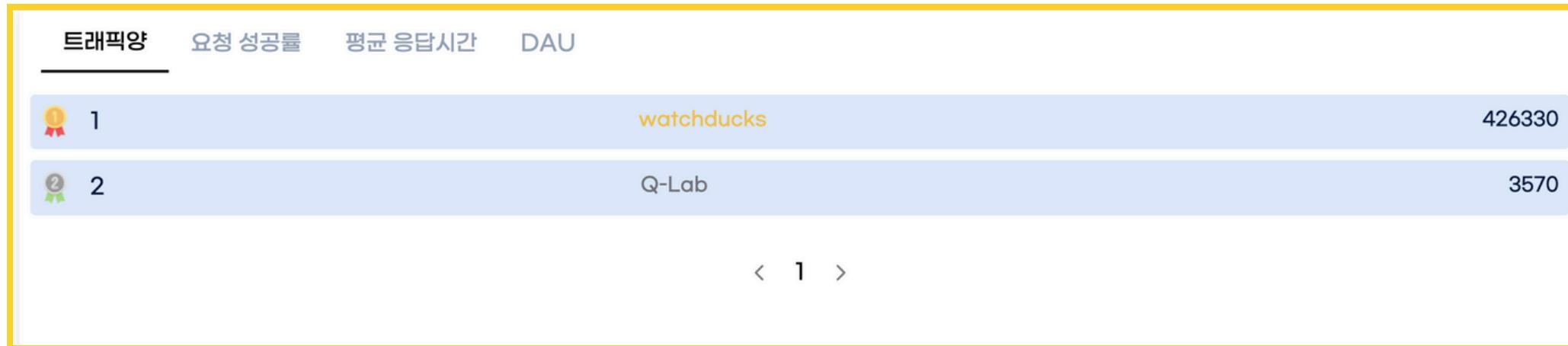
- DNS 호스팅 업체의 도메인 네임서버를 이메일로 전송받은 네임서버로 등록하기만 하면 트래픽 분석 서비스 사용 가능

자세한 사용 가이드 [보러가기](#)

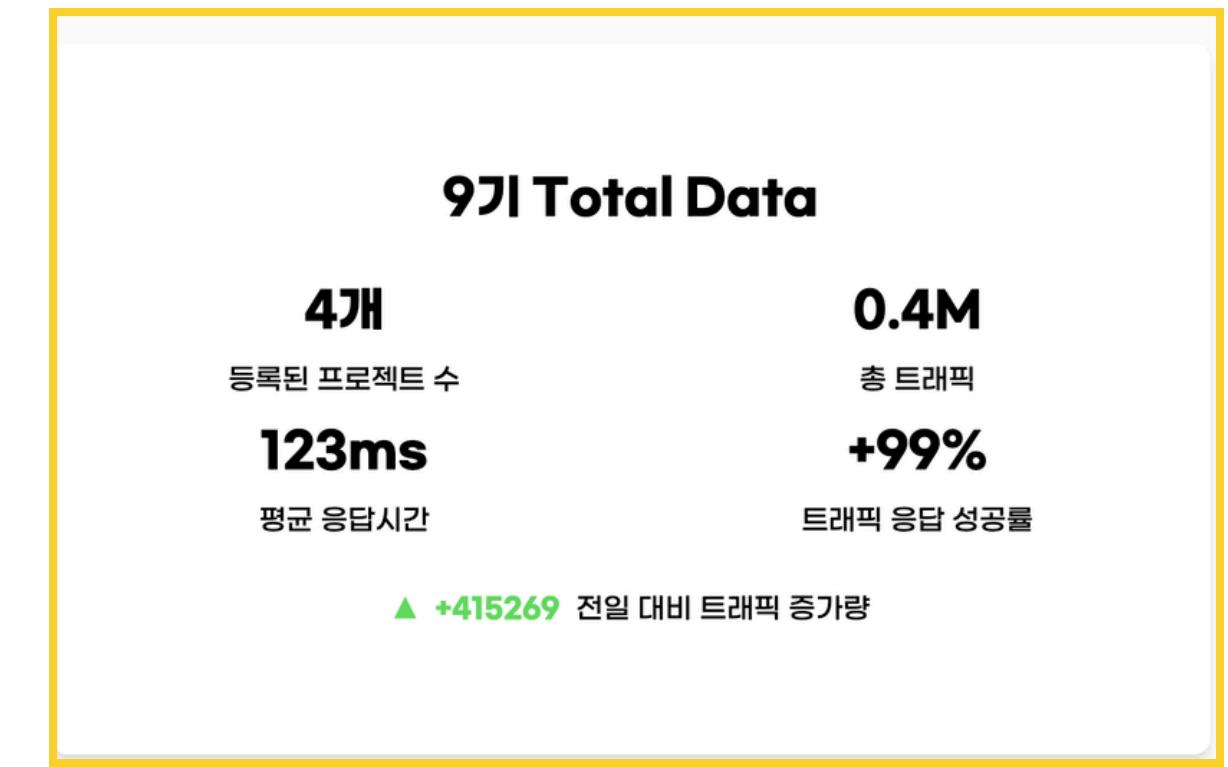
프로젝트 소개

기능 개요

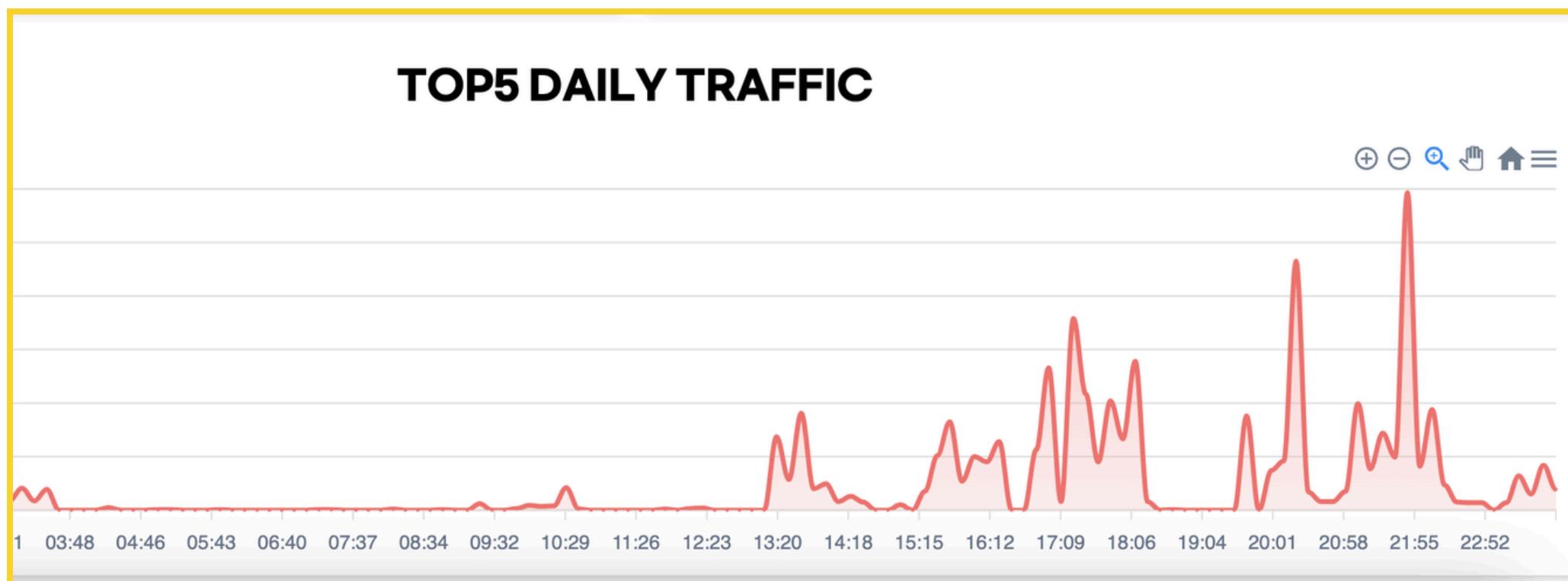
- Watchducks에 사용자의 서비스를 등록하게 되면 발생하는 HTTP 트래픽에 대해 다양한 지표들을 확인할 수 있으며 등록된 다른 서비스들과 함께 비교할 수 있음



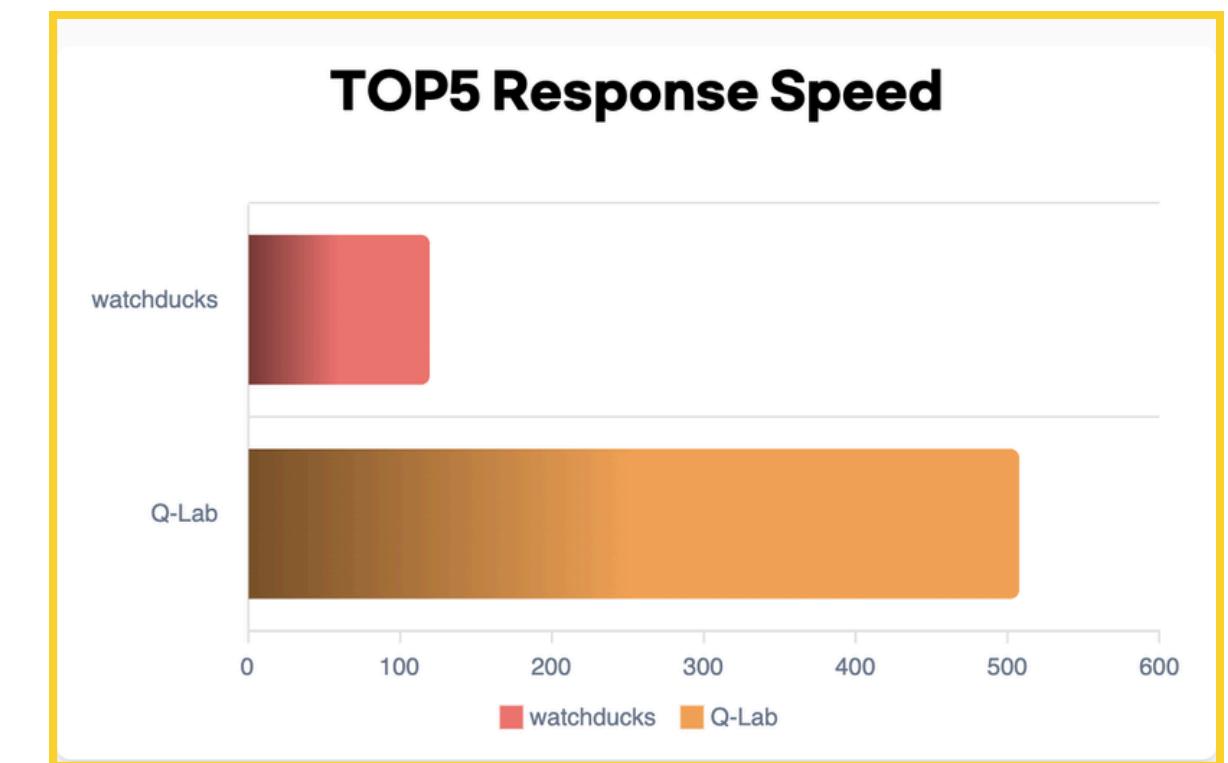
등록된 서비스 지표별 순위



전체 트래픽 지표

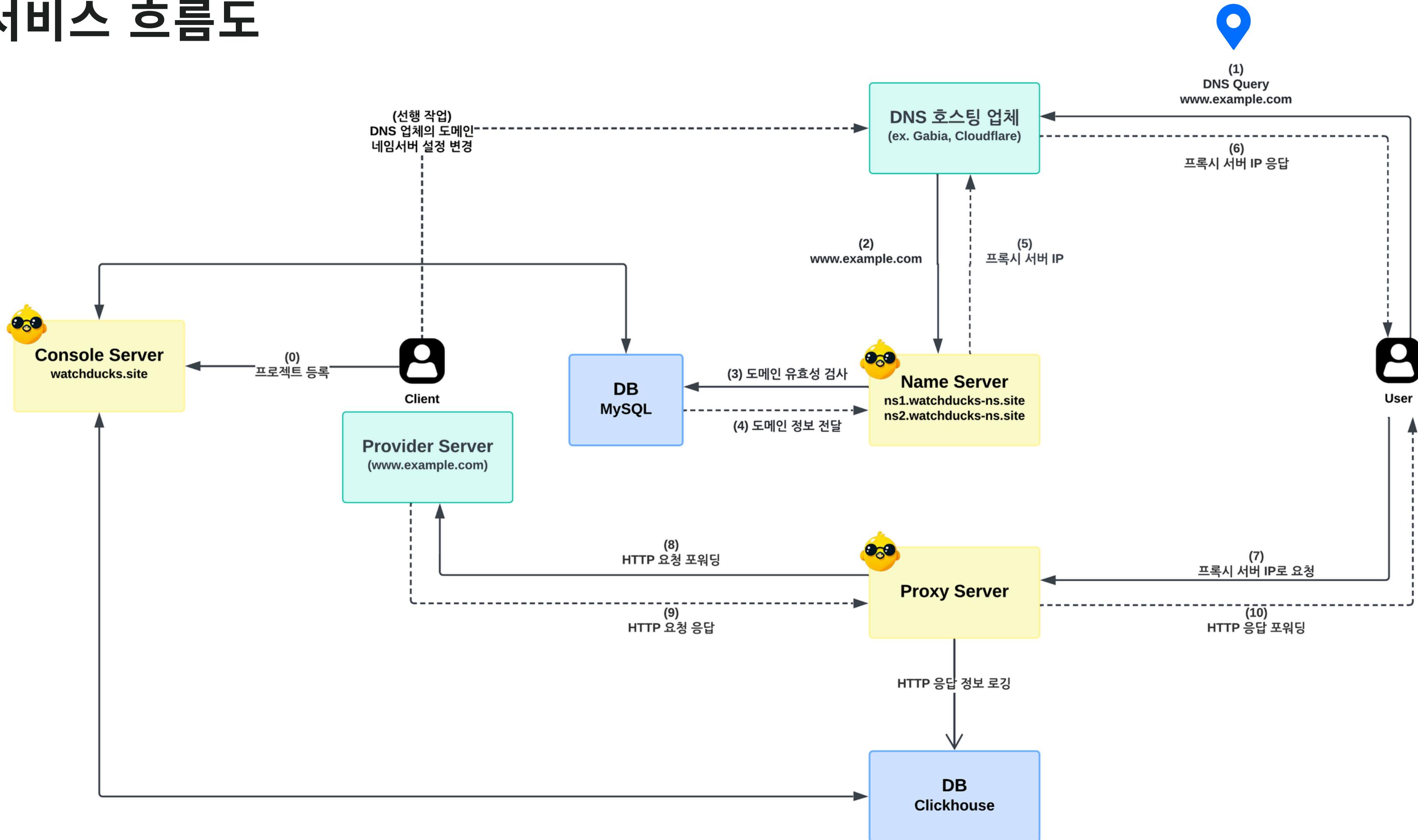


TOP 5 서비스 트래픽 시계열 그래프

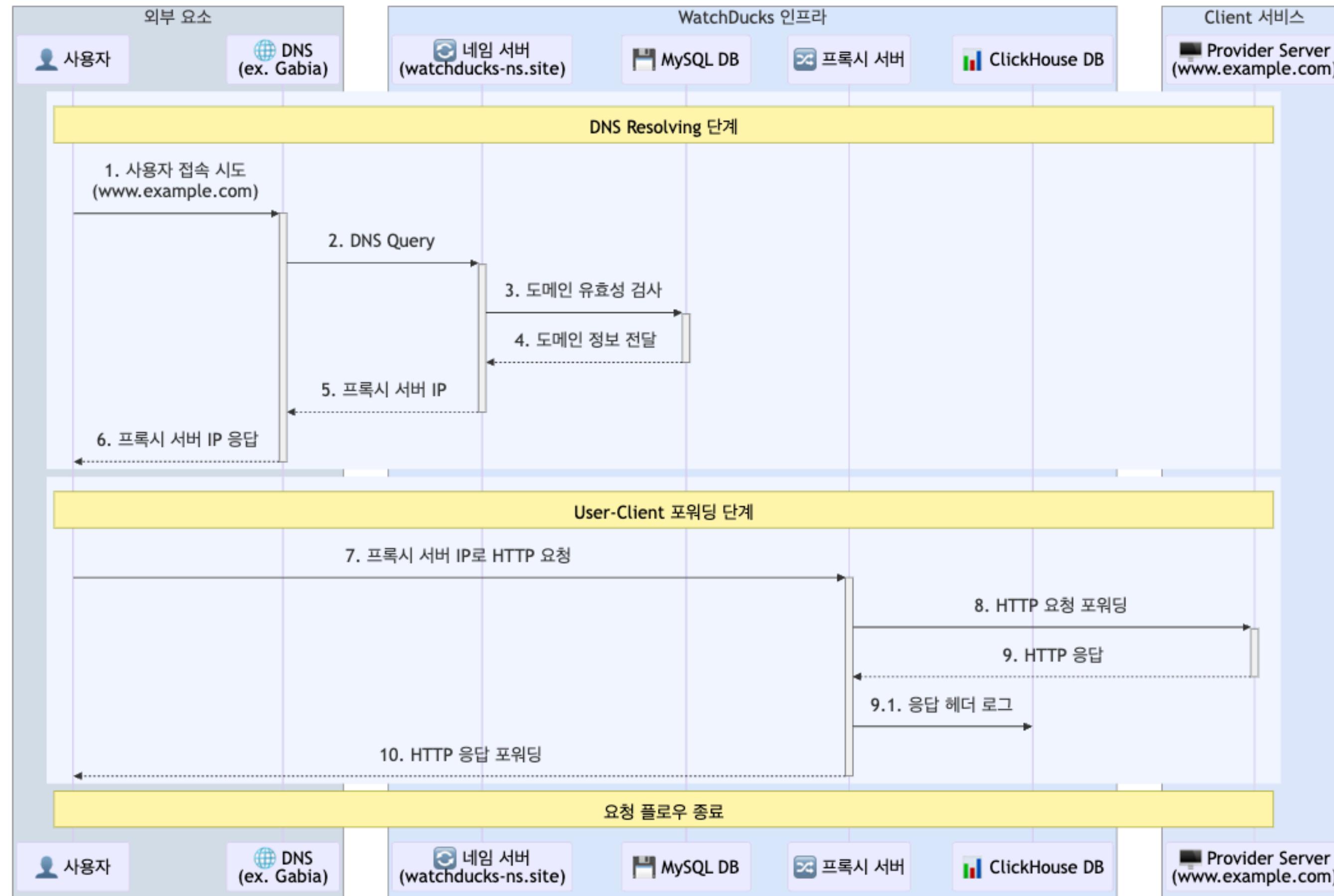


응답 속도 순위

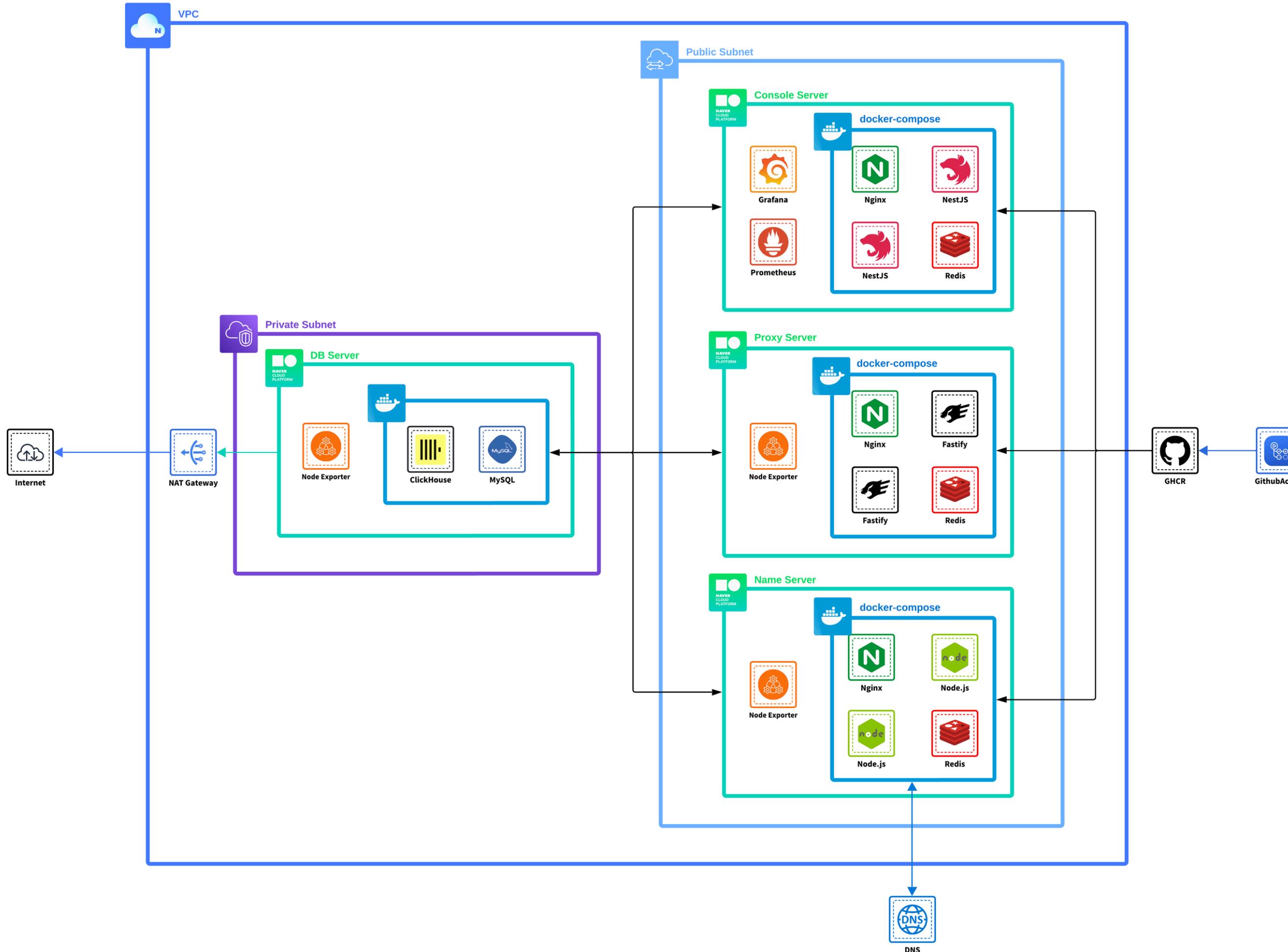
서비스 흐름도



시퀀스 다이어그램



구조



챌린지

01

무중단 배포전략

Nginx와 docker 컨테이너를 사용한
blue-green 전략 도입

02

프록시 서버와 TLS 인증 문제

프록시 서버에서 어떻게 HTTPS 연결을 수립할
수 있을까?

03

컬럼 기반 데이터베이스

Column-Oriented DB인 ClickHouse를 선택,
Insert, Aggregation 최적화

04

프록시 서버 장애대응

프록시 서버가 다운되면 클라이언트 서버도 다운
되는 문제는 어떻게 해결할 수 있을까?

05

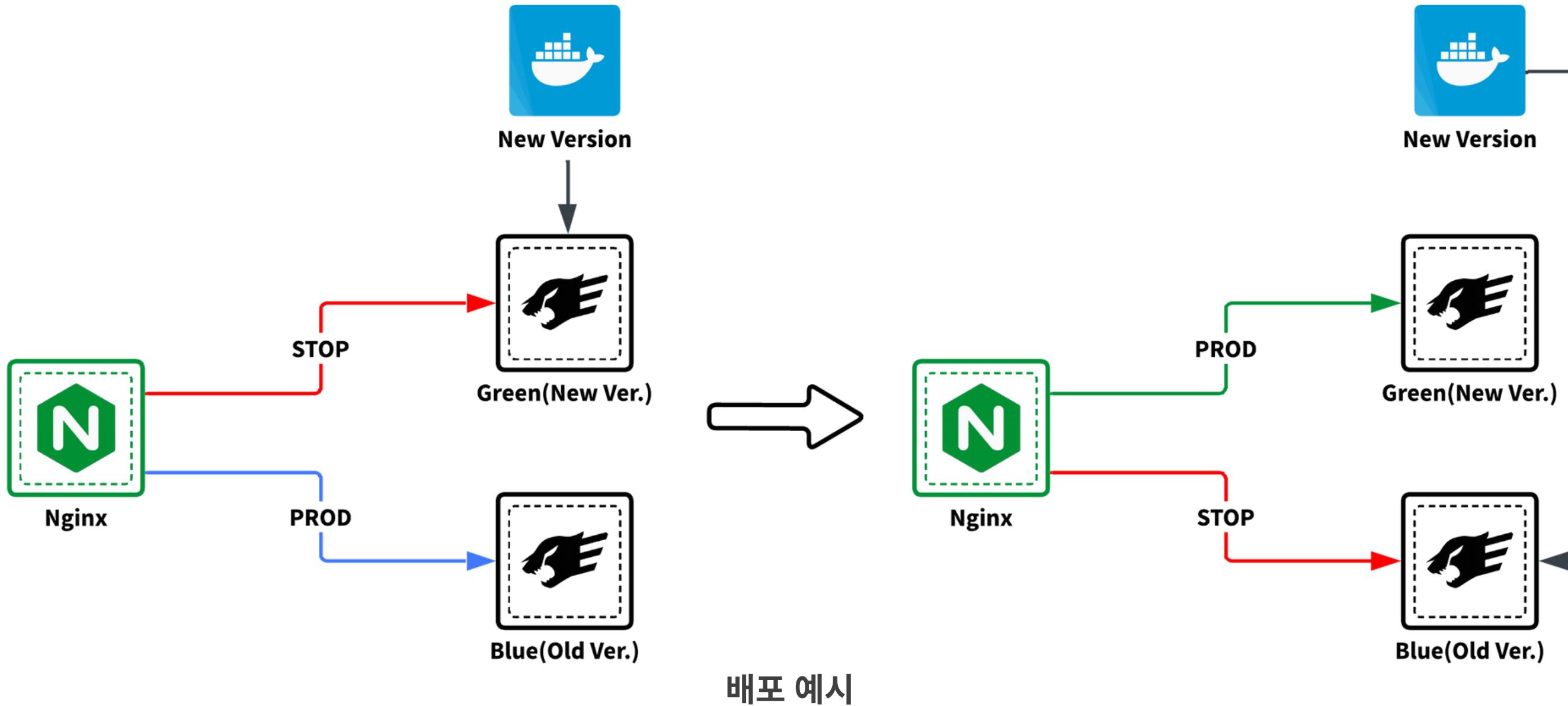
부하테스트와 성능 개선

서버의 안정성이 매우 중요한 Watchducks,
최적의 성능으로 어떻게 개선할 수 있을까?

챌린지

01 무중단 배포전략 - blue/green

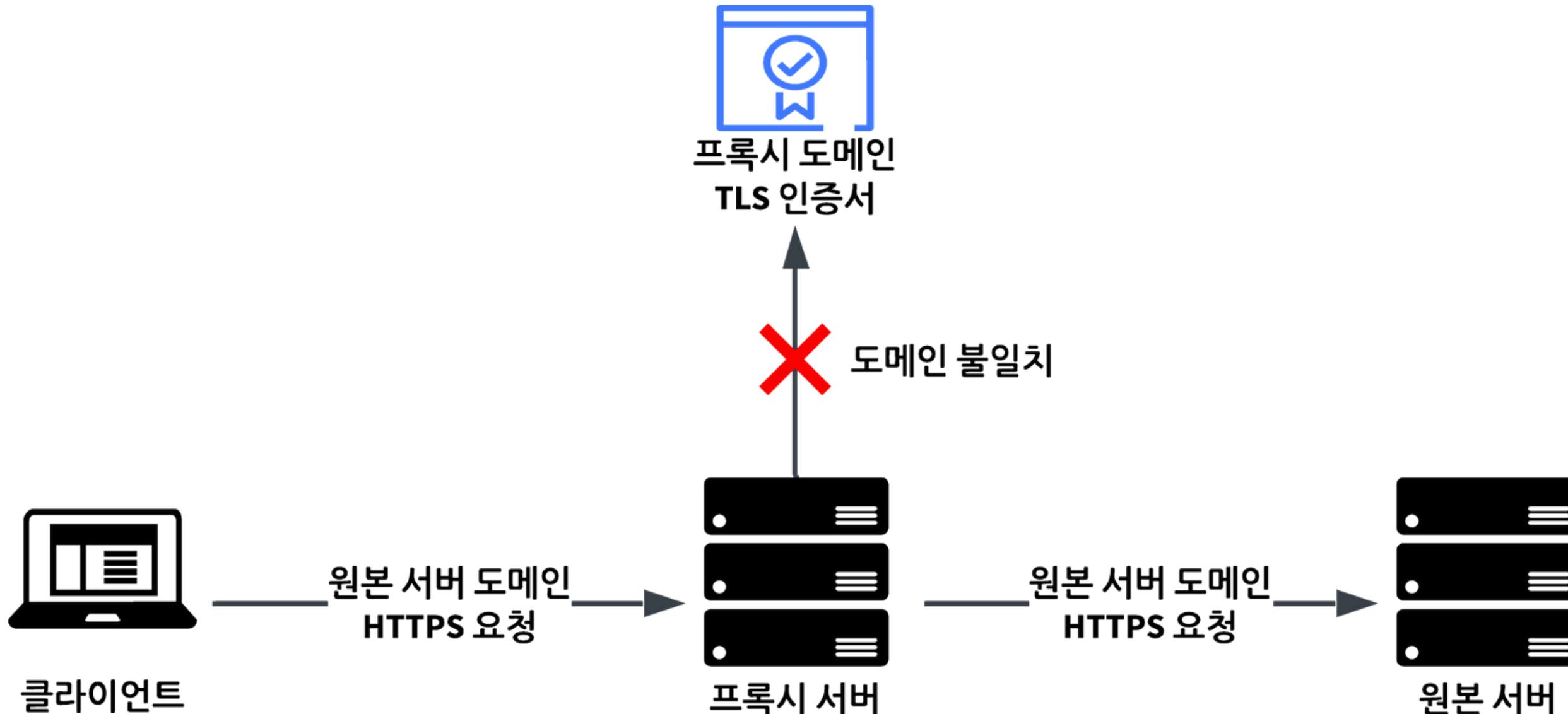
- Watchducks 서비스는 제 3자 서비스의 프록시 서버로 사용되기 때문에 서비스가 중단 될 경우 등록된 모든 3자 서비스들이 장애를 겪게 됨
- 때문에 배포 시 서비스가 중단되지 않고 지속적으로 서비스를 제공할 수 있는 무중단 배포가 필수적



챌린지

02 프록시 서버와 TLS 인증 문제

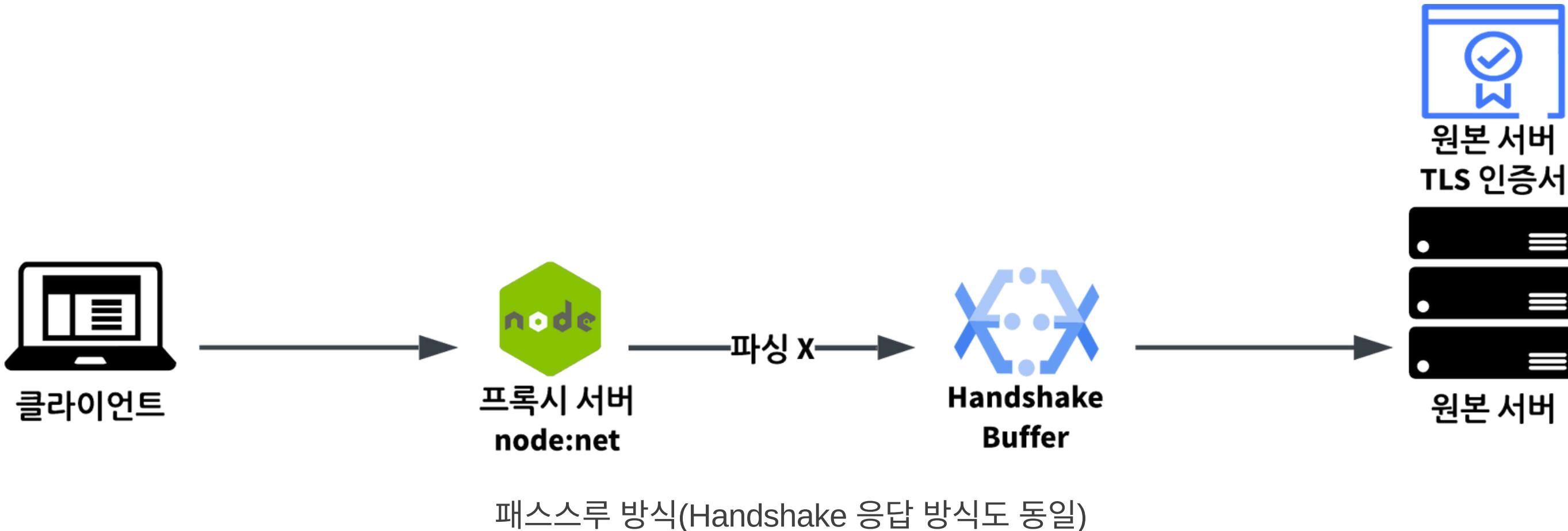
- 거의 모든 어플리케이션은 HTTPS를 사용하기 때문에 Watchdcks 프록시 서버 또한 HTTPS 연결을 지원해야한다
- 클라이언트 요청은 원본 서버 도메인으로 요청을 보내지만, 프록시 서버는 프록시 서버 도메인에 대한 TLS 인증서를 가지기 때문에 도메인 불일치 문제가 발생



챌린지

02 프록시 서버와 TLS 인증 문제 - 패스스루 방식

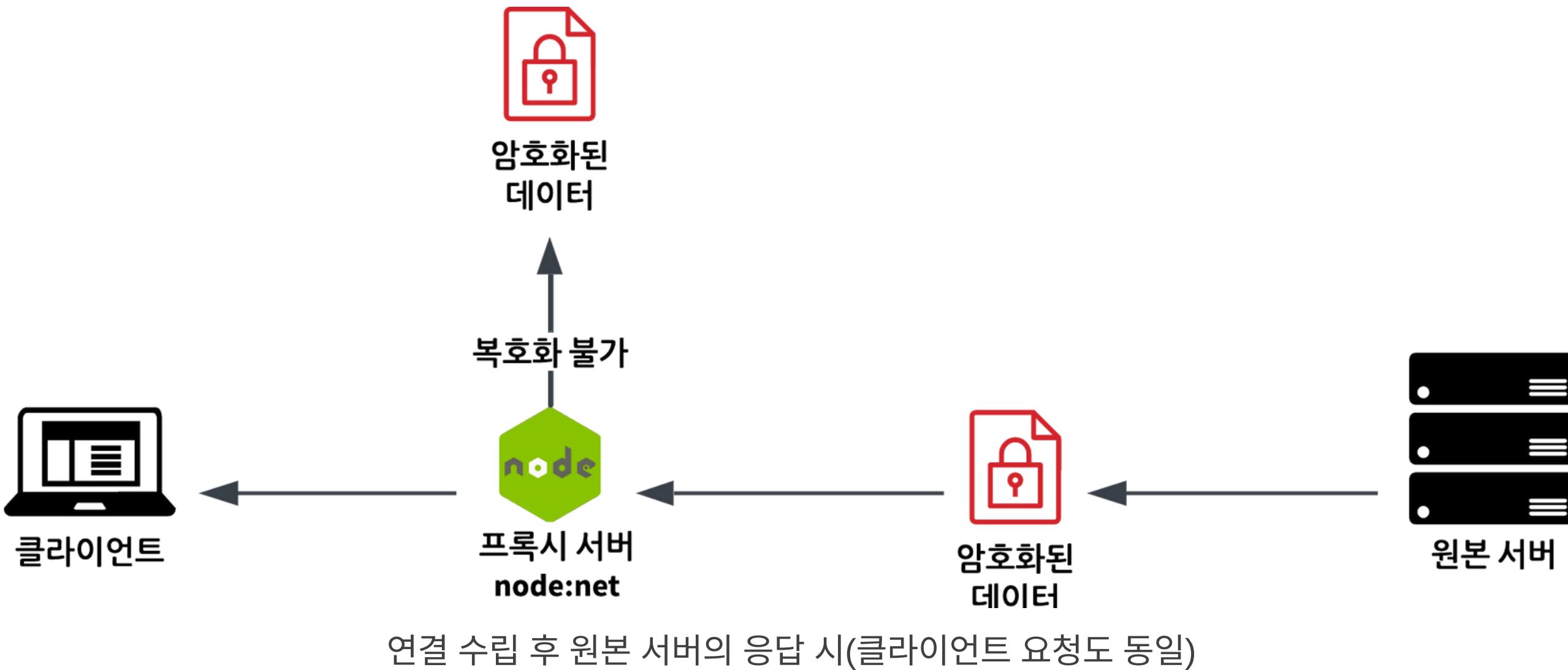
- 클라이언트 - 프록시 서버의 **HTTPS** 연결을 수립하지 않고 **클라이언트 - 원본 서버 연결로 전달하는 방식(패스스루)**으로 문제를 해결하고자 함
- 저수준 TCP 모듈인 `node:net` 모듈을 사용해 연결 수립을 위한 핸드셰이크 과정을 버퍼 그대로 원본 서버에 우회, 전달하는 방식으로 구현
- 핸드셰이크 과정을 그대로 전달하기 때문에 **원본 서버의 TLS 인증서를 사용하게 되며** 연결 수립 후 모든 요청과 응답은 프록시 서버를 거치게 됨



챌린지

02 프록시 서버와 TLS 인증 문제 - 패스스루 방식의 문제점

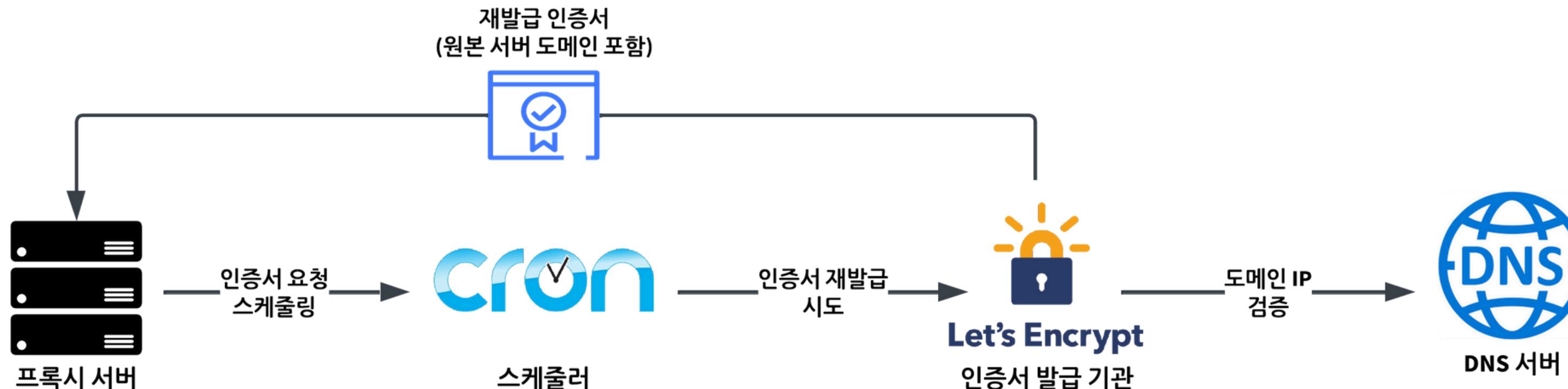
- HTTPS 방식은 데이터 교환 시 헤더를 포함한 모든 데이터를 암호화해서 전달하기 때문에 이를 복호화해야만 로그 데이터를 수집할 수 있음
- 프록시 서버는 핸드셰이크 과정에서 교환되는 복호화 키를 통해서 암호화된 데이터를 복호화 할 수 있을 거라고 생각했으나 **HTTPS는 디피-헬먼 키 교환 방식을 사용하기 때문에 교환되는 키 값만으로는 복호화 불가**



챌린지

02 프록시 서버와 TLS 인증 문제 - 해결 방법

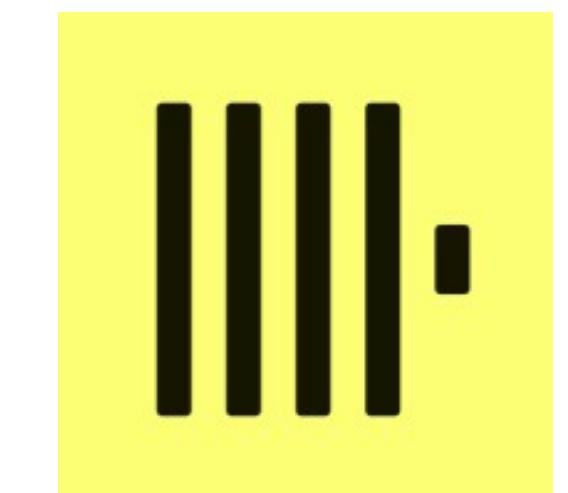
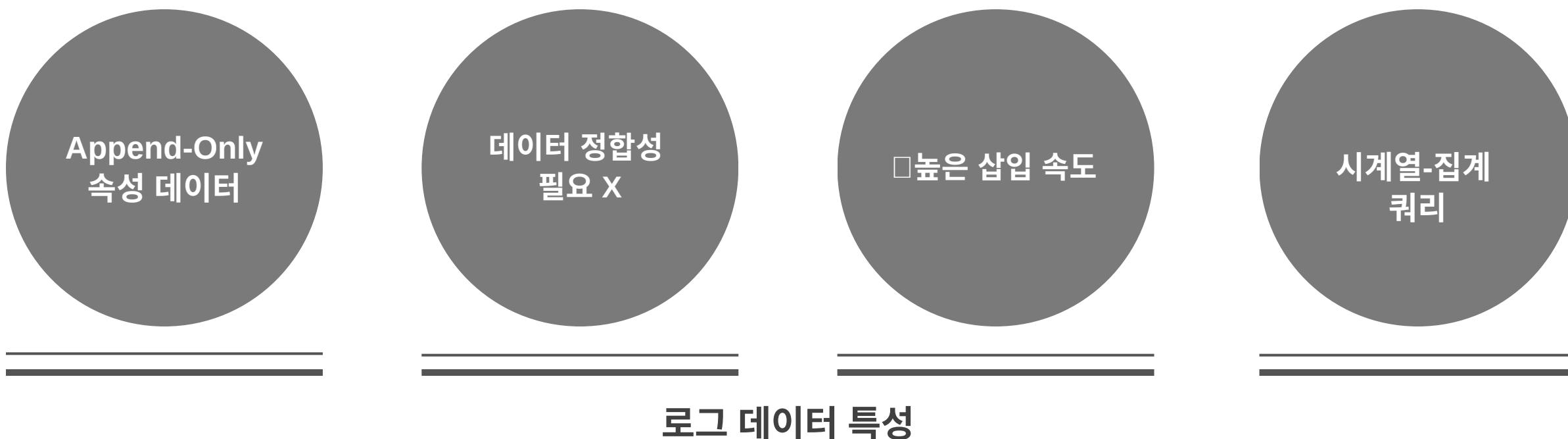
- 클라이언트의 요청 도메인과 프록시 서버의 도메인이 일치하지 않기 때문에 발생하는 문제이므로 SAN을 이용해 프록시 서버의 인증서에 등록된 모든 원본 서버의 도메인을 추가하는 방식을 사용하는 방식으로 문제 해결
- 다만, 인증서에 도메인을 추가하려면 인증서를 새로 발급받아야하는 문제 발생
- 인증서 발급 기관으로부터 새로운 인증서를 발급받으려면 DNS 서버에 등록된 도메인의 IP와 요청 IP가 동일해야함
- 따라서 새로 추가된 도메인이 포함된 인증서를 재발급 받기 위해서는 Watchducks 서비스 사용자가 DNS의 네임서버를 변경 한 후 전파되기 까지의 자연 시간동안 지속적으로 재발급 시도를 하는 방식으로 해결



챌린지

03 컬럼 기반 데이터베이스

- Watchducks에서 수집/저장하는 데이터는 HTTP 요청/응답에 대한 로그 데이터이기 때문에 데이터베이스 선택에 따라 천차만별의 성능을 보여줌
- 로그 데이터는 **삭제/수정이 일어나지 않으며(Append Only)**, 데이터 정합성이 필요하지 않고, 초당 수천 건의 로그 데이터를 삽입해야하기에 **높은 삽입속도**가 중요
- 또한 Watchducks는 수집된 데이터를 시계열 기준으로 집계하여 모니터링할 수 있어야하므로 **시계열-집계 쿼리에 강점을 가져야 함**



컬럼 기반 데이터베이스
Clickhouse

챌린지

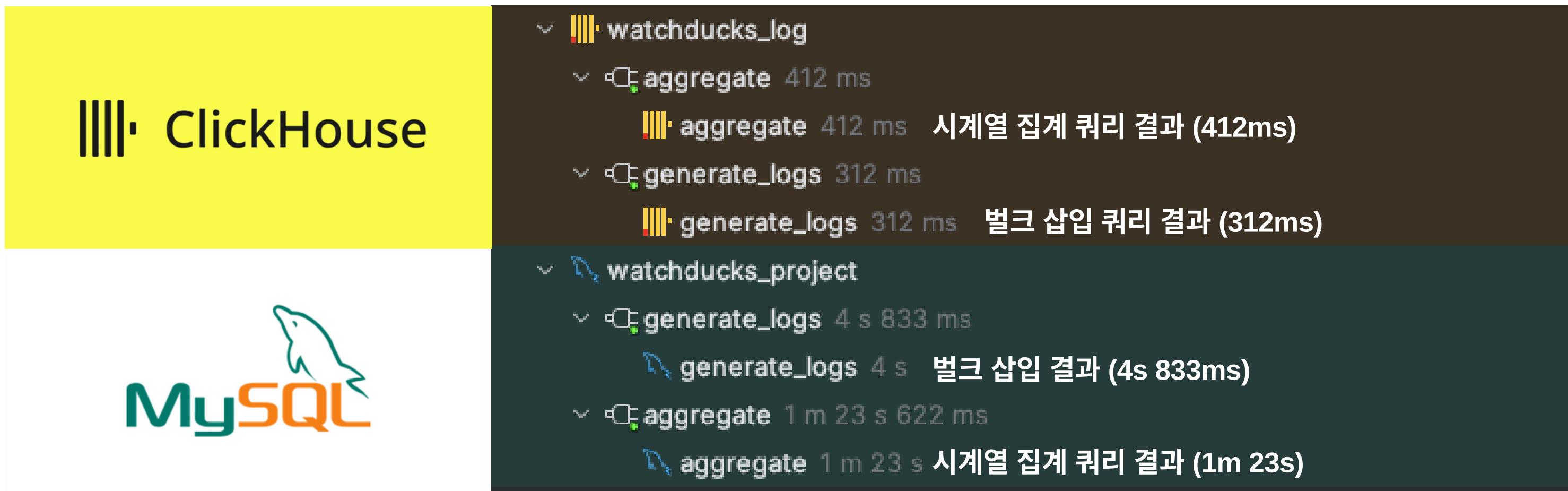
03 컬럼 기반 데이터베이스 - 성능 비교

- 천만 건의 HTTP 요청/응답 로그 데이터에 대해 다음 두 가지 조건으로 테스트
- 1. 시계열 집계 쿼리 - 특정 시계열 구간의 응답 속도에 대한 평균 값을 산출하는 쿼리 (aggregate)

결과 - Clickhouse(컬럼 기반): 412ms / MySQL(로우 기반): 1m 23s

- 2. 벌크 삽입 쿼리 - 수집된 로그 데이터 10만 건에 대한 삽입 쿼리 (generate_logs)

결과 - Clickhouse(컬럼 기반): 312ms / MySQL(로우 기반): 4s 833ms



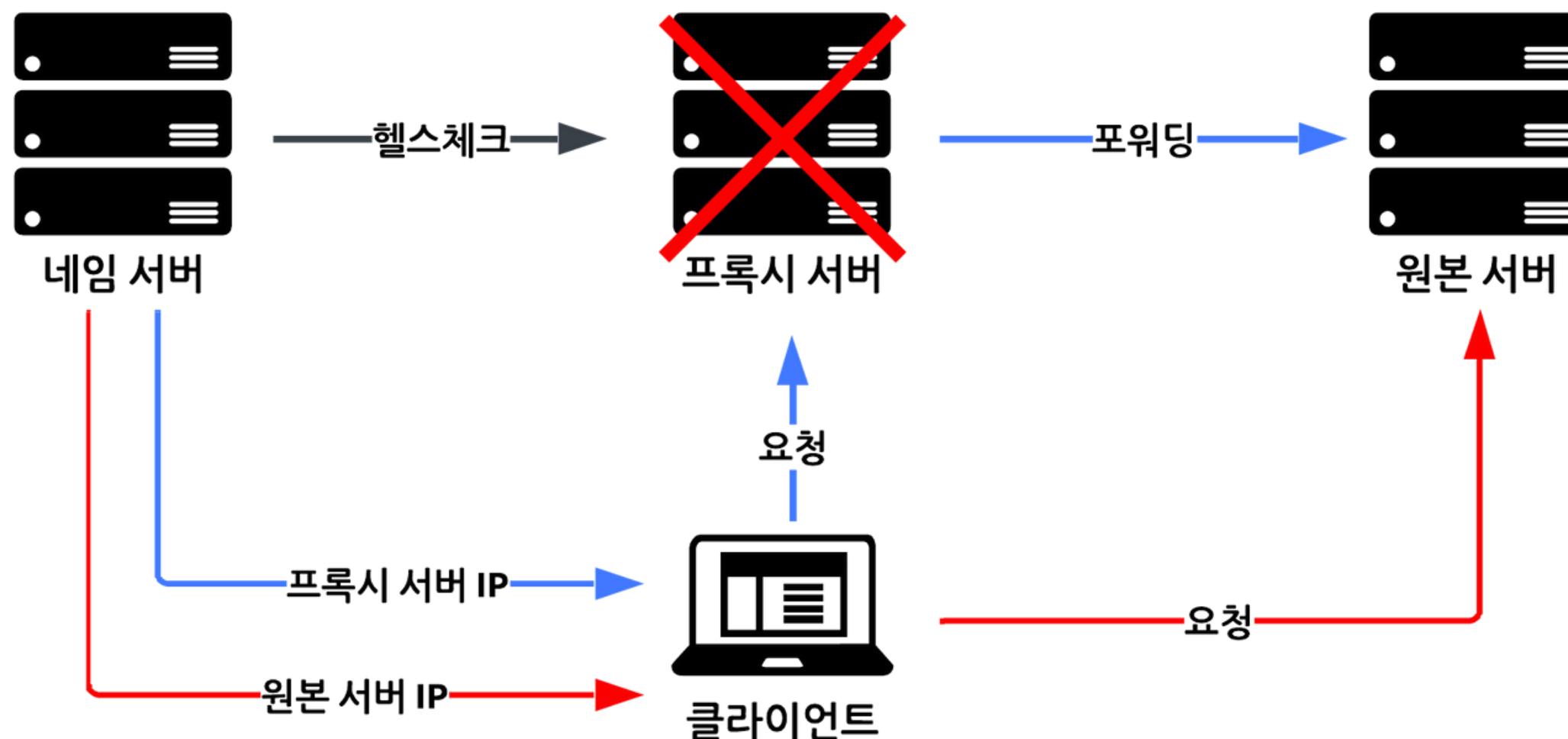
비교 테스트 결과

챌린지

04 프록시 서버 장애 대응

- 만약 프록시 서버에 장애가 발생해 클라이언트가 원본 서버의 서비스를 사용할 수 없는 상황이 발생해서는 안됨 (**프록시 서버의 장애가 원본 서버에 영향을 미쳐서는 안됨**)
- 프록시 서버 장애 발생 시 이에 대응하기 위해 네임서버는 짧은 주기로 프록시 서버에 헬스체크 메시지를 보냄으로써 프록시 서버의 정상 작동을 확인
- 만약 네임 서버가 프록시 서버의 장애를 감지하면 클라이언트에게 **프록시 서버의 IP가 아닌 원본 서버의 IP를 반환하는 장애 대응책 구현**

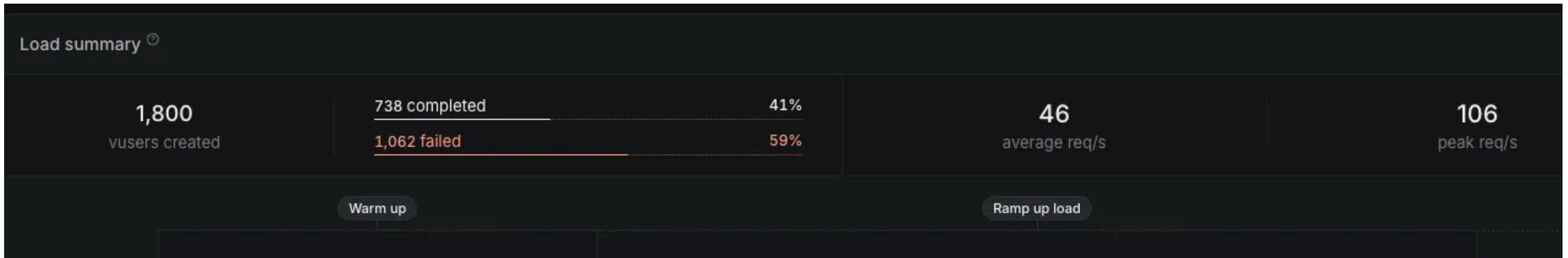
—— 헬스체크 실패 흐름
—— 헬스체크 성공 흐름



챌린지

05 부하테스트와 성능 개선 - 부하테스트 결과

- Watchducks 사용자 서비스의 모든 트래픽은 프록시 서버를 거치기 때문에 높은 부하를 견딜 수 있어야 함
- 부하테스트 도구인 Artillery를 사용해 초당 1000개 트래픽을 목표로 테스트를 진행했으며 낮은 트래픽 부터 점진적으로 증가시키며 최대 부하 지점을 찾고 원인을 분석/개선하는 과정을 거침
- 최초 부하테스트에서 피크 106req/s / 평균 46req/s 트래픽에도 59%에 달하는 요청 실패율을 보였으며 이는 기준치를 한참 미달한 채로 요청 실패를 보인 매우 충격적인 결과임



부하테스트 결과

챌린지

05 부하테스트와 성능 개선 - 원인 파악 및 개선

- Prometheus-Grafana를 이용한 리소스 모니터링 결과 데이터베이스 서버에서 높은 CPU 부하를 확인함
- 데이터베이스 서버에는 현재 사용자 정보를 저장하는 MySQL과 로그 데이터를 저장하는 Clickhouse 어플리케이션이 있으며 MySQL에서 특히 높은 CPU 점유율을 보임
- MySQL 쿼리 분석 결과, 포워딩 시 원본 서버의 IP를 불러오는 과정에서 매 트래픽마다 데이터베이스 쿼리가 발생하는 문제를 발견
- □ 해당 문제에 대한 방안으로 Redis를 이용한 캐시 전략 도입



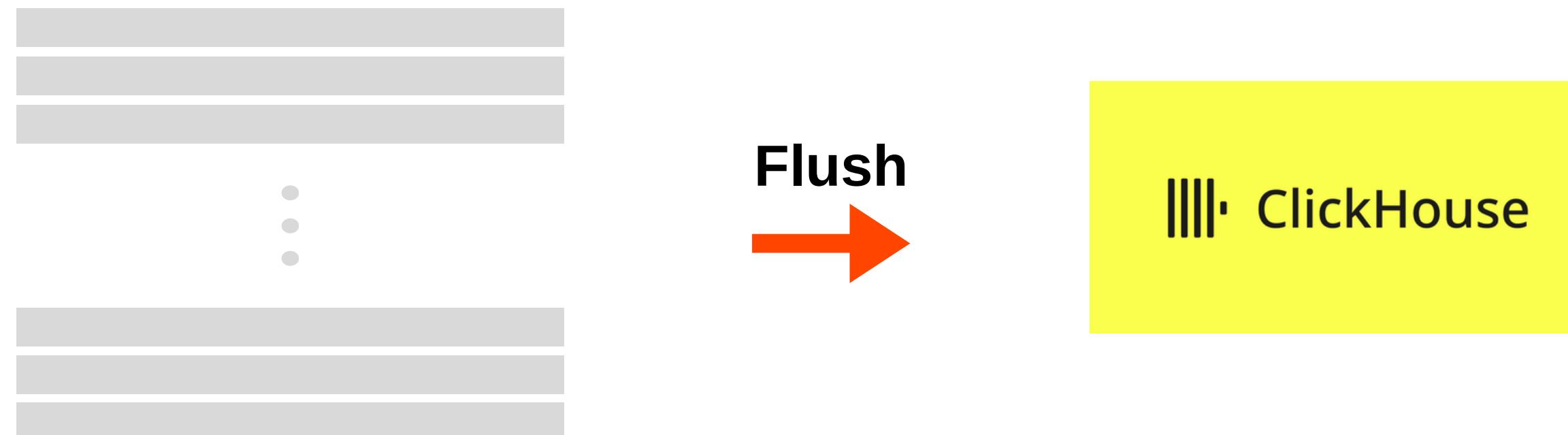
□데이터베이스 서버 리소스 모니터링 결과

챌린지

05 부하테스트와 성능 개선 - 원인 파악 및 개선 과정

- 이밖에도 매 트래픽 마다 Clickhouse에 로그 데이터를 저장하는 insert 쿼리가 발생하는 문제 발견
- 매 트래픽마다 insert 쿼리가 발생할 경우 어플리케이션-데이터베이스 서버간의 네트워크 비용과 I/O 발생 빈도가 높아지기 때문에 성능 저하가 심할 것으로 예상
- 어플리케이션 레벨에서 메모리 버퍼를 이용해 5초 주기 혹은 버퍼가 가득 찬을 때(1만 건의 row) Flush하는 방식으로 네트워크 및 I/O 작업의 빈도를 줄임으로써 성능 개선 (초당 1000건의 I/O 작업 -> 5초간 1-2건의 I/O 작업)

**Application Memory
Buffer**



챌린지

05 부하테스트와 성능 개선 - 원인 파악 및 개선 과정

- 이외에도 여러 방면에서 아래와 같은 개선 작업을 진행
 - Nginx의 로드 밸런싱과 멀티프로세스를 이용해 멀티 코어를 적극 활용한 성능 개선
 - 트래픽 데이터 API에 대해 Redis를 활용한 캐시전략 도입, 데이터베이스 부하 개선
 - Clickhouse 데이터베이스의 내부 버퍼를 활용한 쓰기 작업 최적화 옵션인 `async_insert` 활성화
 - undici 방식을 이용한 HTTP 처리 속도 및 동시 처리 개선

챌린지

05 부하테스트와 성능 개선 - 개선 결과

- 200번 이상의 부하테스트를 거쳐 최초 100req/s에서 조차 요청 실패가 발생했던 기존의 성능에서 1800req/s에서도 요청 성공률 100%를 지속적으로 달성하는 성능 개선을 이룸
- 해당 부하테스트 중 프록시 서버의 리소스 모니터링 결과 최대 70% / 평균 50%의 CPU 부하를 보임

