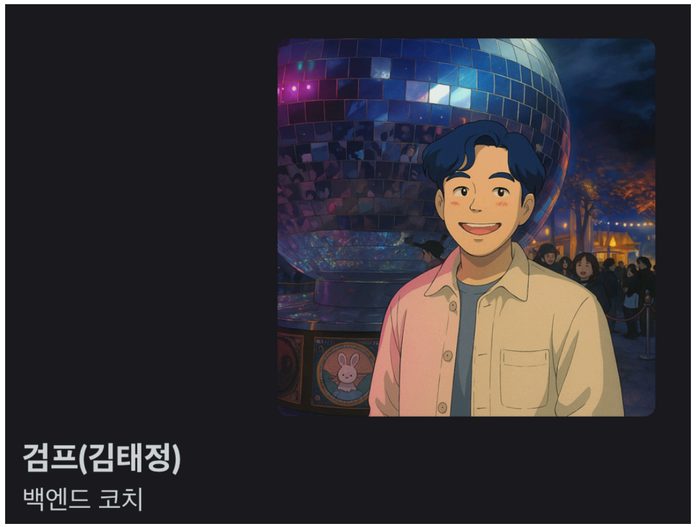
런칭데이 대비
처리율 제한기 및 로드밸런서 적용기

계기

어차피, 서비스 지금 많아봐야 100명밖에 안쓰는데

지금 트래픽 고려하는거 오버엔지니어링 아닌가요?

계기



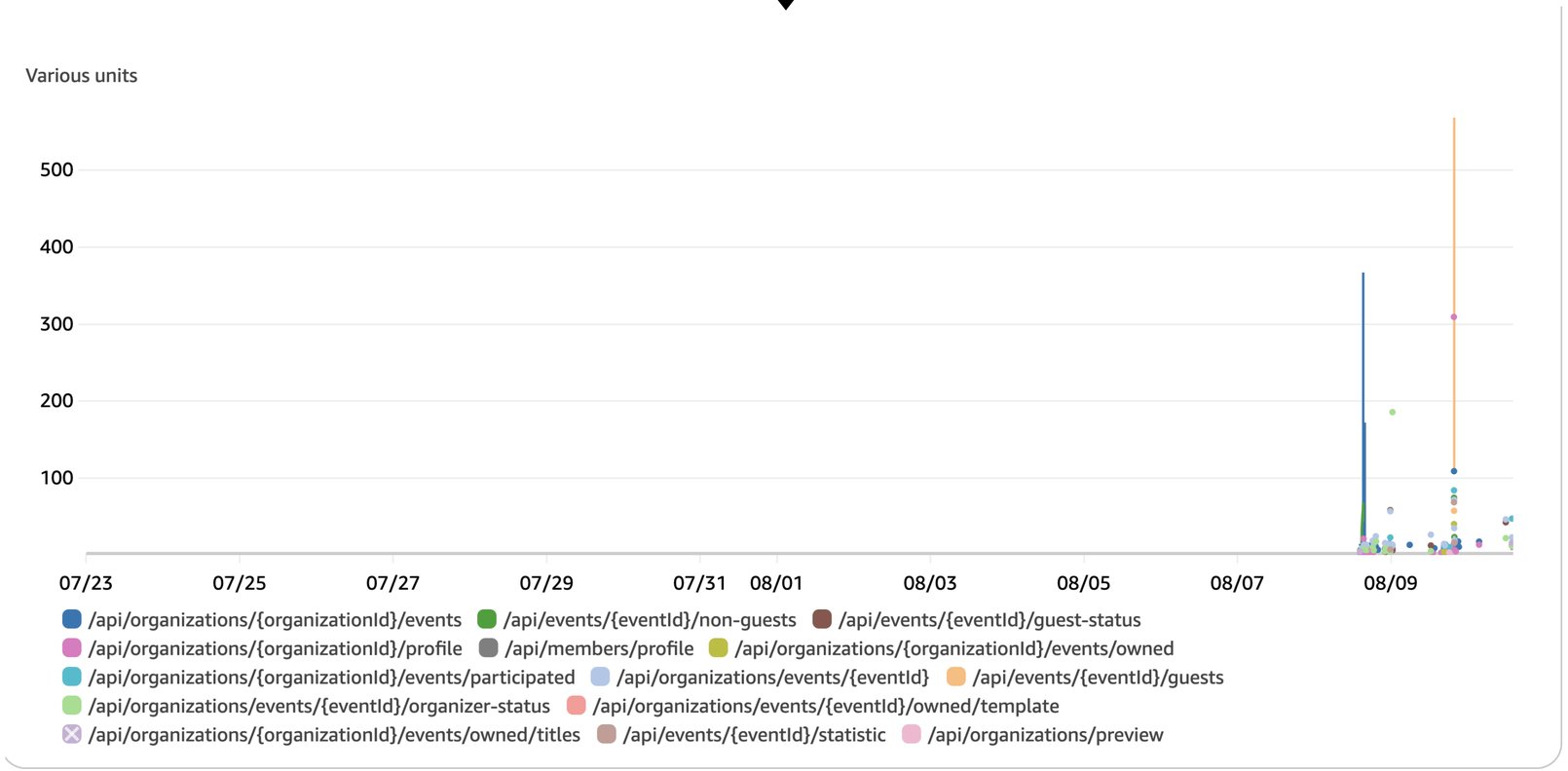
1. public에서 서버에 접근할 수 있는 한
   악의적인 사용자는 언제나 존재할 수 있다.
2. 아무리 좋은 기능을 만들어도 서버에서 안정성있게 서빙하지 못하면 의미가 없다
3. 서버 장애는 실질적으로 회사에, 비즈니스에 타격을 준다

이전 데모데이에서는..

서비스가 갑자기 느려졌는데, 확인 부탁드립니다.

이전 데모데이에서는..

서비스가 갑자기 느려졌는데, 확인 부탁드립니다.



약 11만개의 요청이 옴

11만개의 리퀘스트 처리 - 아이디어 1

11만개의 리퀘스트를 어떻게 막으란거지?

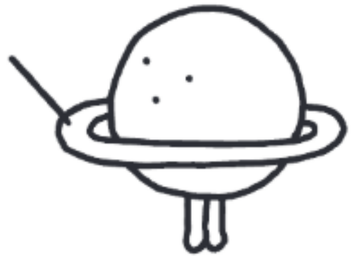
"**스케일아웃**"밖에 답이 없는거 아닌가?

11만개의 리퀘스트 처리 - 아이디어 1

아이디어 1.

한 ip인데 1분내에 과도하게 많은 요청이 오면

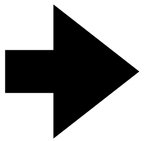
"악의적인 사용자"로 간주한다.



11만개의 리퀘스트 처리 - 아이디어 1

해결 방법

nginx단 Rate Limit 설정 적용



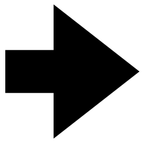
```
http {

    # IP 주소당 1초에 100개의 요청을 처리하는 규칙 정의
    # zone=perip:10m -> 'perip'라는 이름의 10MB 메모리 공간 사용
    # rate=100r/s -> 초당 100개 요청 허용
    limit_req_zone $remote_addr zone=perip:10m rate=10r/s;
}
```

nginx 설정

11만개의 리퀘스트 처리 - 아이디어 1

해결 방법

nginx단 Rate Limit 설정 적용



3	39.125.9.126 - - [20/Aug/2025:14:59:33 +0900] "GET / HTTP/2.0" 404 102 "-" "curl/8.7.1"
4	39.125.9.126 - - [20/Aug/2025:14:59:33 +0900] "GET / HTTP/2.0" 404 102 "-" "curl/8.7.1"
5	39.125.9.126 - - [20/Aug/2025:14:59:33 +0900] "GET / HTTP/2.0" 404 102 "-" "curl/8.7.1"
6	39.125.9.126 - - [20/Aug/2025:14:59:33 +0900] "GET / HTTP/2.0" 404 102 "-" "curl/8.7.1"
7	39.125.9.126 - - [20/Aug/2025:14:59:33 +0900] "GET / HTTP/2.0" 404 102 "-" "curl/8.7.1"
8	39.125.9.126 - - [20/Aug/2025:14:59:33 +0900] "GET / HTTP/2.0" 404 102 "-" "curl/8.7.1"
9	39.125.9.126 - - [20/Aug/2025:14:59:33 +0900] "GET / HTTP/2.0" 404 102 "-" "curl/8.7.1"
10	39.125.9.126 - - [20/Aug/2025:14:59:33 +0900] "GET / HTTP/2.0" 404 102 "-" "curl/8.7.1"
11	39.125.9.126 - - [20/Aug/2025:14:59:33 +0900] "GET / HTTP/2.0" 404 102 "-" "curl/8.7.1"
12	39.125.9.126 - - [20/Aug/2025:14:59:33 +0900] "GET / HTTP/2.0" 404 102 "-" "curl/8.7.1"
13	39.125.9.126 - - [20/Aug/2025:14:59:33 +0900] "GET / HTTP/2.0" 404 102 "-" "curl/8.7.1"
14	39.125.9.126 - - [20/Aug/2025:14:59:33 +0900] "GET / HTTP/2.0" 404 102 "-" "curl/8.7.1"
15	39.125.9.126 - - [20/Aug/2025:14:59:33 +0900] "GET / HTTP/2.0" 404 102 "-" "curl/8.7.1"
16	39.125.9.126 - - [20/Aug/2025:14:59:33 +0900] "GET / HTTP/2.0" 503 206 "-" "curl/8.7.1"
17	39.125.9.126 - - [20/Aug/2025:14:59:33 +0900] "GET / HTTP/2.0" 503 206 "-" "curl/8.7.1"
18	39.125.9.126 - - [20/Aug/2025:14:59:33 +0900] "GET / HTTP/2.0" 503 206 "-" "curl/8.7.1"
19	39.125.9.126 - - [20/Aug/2025:14:59:33 +0900] "GET / HTTP/2.0" 503 206 "-" "curl/8.7.1"
20	39.125.9.126 - - [20/Aug/2025:14:59:33 +0900] "GET / HTTP/2.0" 503 206 "-" "curl/8.7.1"
21	39.125.9.126 - - [20/Aug/2025:14:59:33 +0900] "GET / HTTP/2.0" 503 206 "-" "curl/8.7.1"

악의적인 요청으로 판단시

503과 함께 거부

11만개의 리퀘스트 처리 - 아이디어 2

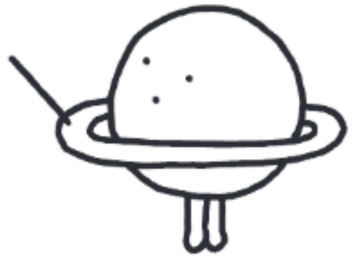
ip는 쉽게 위조 가능한데, 너무 쉽게 뚫리는 것 아닌가?

11만개의 리퀘스트 처리 - 아이디어 2

아이디어 2.

서버에서 처리 비용이 큰 api들은 유저 권한이 필요함

- **유저 단위로 처리율 제한을 걸자**

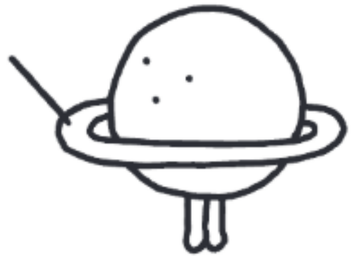


11만개의 리퀘스트 처리 - 아이디어 2

아이디어 2.

서버에서 처리 비용이 큰 api들은 유저 권한이 필요함

- **유저 단위로 처리율 제한을 걸자**



11만개의 리퀘스트 처리 - 아이디어 2

```
@Override
protected void doFilterInternal(
        final HttpServletRequest request,
        final HttpServletResponse response,
        final FilterChain filterChain
) throws ServletException, IOException {
    String authorizationHeader = request.getHeader(HttpHeaders.AUTHORIZATION);
    Long memberId = extractMemberIdSafely(authorizationHeader);

    if (memberId == null) {
        filterChain.doFilter(request, response);
        return;
    }

    long now = Instant.now()
            .toEpochMilli();
    Deque<Long> timestamps = requestLogs.computeIfAbsent(memberId, id -> new ConcurrentLinkedDeque<>());

    if (isRateLimited(timestamps, now)) {
        respondTooManyRequests(request, response, timestamps, now);
        return;
    }

    filterChain.doFilter(request, response);
}
```

멤버별 윈도우 관리

처리율 확인

11만개의 리퀘스트 처리 - 아이디어 2

```
private boolean isRateLimited(final Deque<Long> timestamps, final long now) {
    synchronized (timestamps) {
        while (!timestamps.isEmpty() && timestamps.peekFirst() < now - WINDOW_MILLIS) {
            timestamps.pollFirst();
        }

        if (timestamps.size() >= MAX_REQUESTS) {
            return true;
        }

        timestamps.addLast(now);
        return false;
    }
}
```

이미 1분이 지난것은

큐에서 제외

처리율 초과시 반환

11만개의 리퀘스트 처리 - 아이디어 2

처리율 제한 적용 이후 부하 테스트 설계

- active user 300명을 가정, 각 유저가 500개의 요청을 보낸다고 가정
- active user는 300명이지만 애플리케이션내에서 식별되는 유저는 한명
- 처리율 제한으로 인해 많은 요청이 429로 쳐내질 것이므로 응답속도가 향상될것으로 예상

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
get all events	150000	121	8	14317	519.51	99.91%	1201.4/sec	426.71	470.48	
get organization	150000	121	7	14315	525.98	99.96%	1202.0/sec	426.74	403.81	
TOTAL	300000	121	7	14317	522.76	99.93%	2402.7/sec	853.17	874.01	

메인화면 기준 15만 리퀘스트에 대해서 평균 0.3초 달성

11만개의 리퀘스트 처리 - 아이디어 2

처리율 제한 적용 이후 부하 테스트 설계

- active user 300명을 가정, 각 유저가 500개의 요청을 보낸다고 가정
- active user는 300명이지만 애플리케이션내에서 식별되는 유저는 한명
- 처리율 제한으로 인해 많은 요청이 429로 쳐내질 것이므로 응답속도가 향상될것으로 예상

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
get all events	150000	121	8	14317	519.51	99.91%	1201.4/sec	426.71	470.48	
get organization	150000	121	7	14315	525.98	99.96%	1202.0/sec	426.74	403.81	
TOTAL	300000	121	7	14317	522.76	99.93%	2402.7/sec	853.17	874.01	

메인화면 기준 15만 리퀘스트에 대해서 평균 0.3초 달성

11만개의 리퀘스트 처리 - 아이디어 2

11만개의 리퀘스트 처리 - 아이디어 2

에러가 나면 안좋은거 아닌가요?

악의적인 유저에 대해서 빠르게 필터단에서 요청 거부.

=> 데이터베이스 등 자원의 병목을 줄이고

정상적인 유저들이 서비스를 더 오래 사용할 수 있게 됨

11만개의 리퀘스트 처리 - 아이디어 2

해결 방법

멤버별로 최근 요청을 담는 슬라이딩 윈도우 방식

- 최근 요청이 1분내 100개가 넘으면 거부