

Kullanıcınız ya da trafiğiniz artmadı
ancak backend request sayısı
neden 9 katına çıktı?



Bu haftanın kontrolü

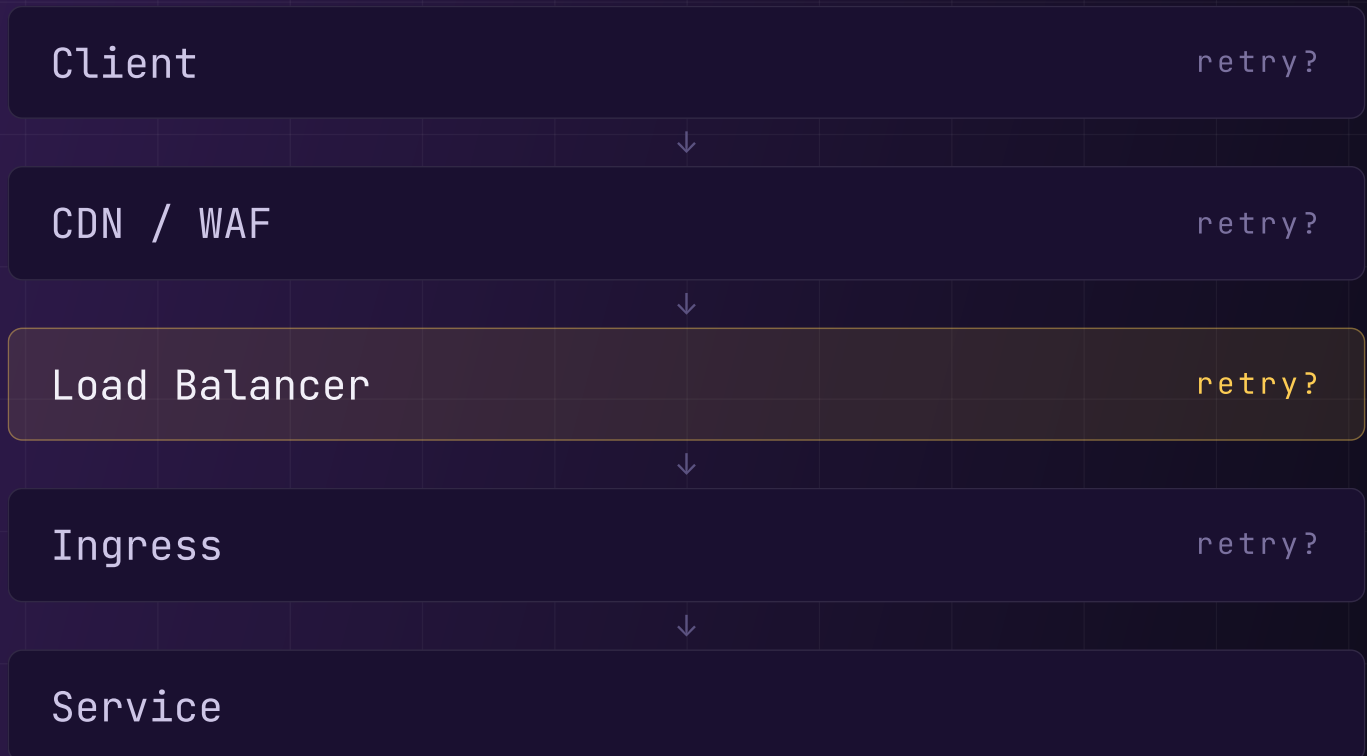
Load Balancer Retries

Don't Deploy on Friday. Verify on Friday.

Tek request?

Kullanıcı yalnızca bir request gönderdiğini düşünür.

Ancak request şu katmanlardan geçebilir:



Bu katmanların her biri kendi retry politikasına sahip olabilir.

Bir request, backend'de **bir request olarak kalmayabilir.**

Retry çarpımı

CLIENT

1 initial attempt + 2 retry = 3 attempt



LOAD BALANCER

Her client attempt'i için

1 initial attempt + 2 retry = 3 attempt



Sonuç: $3 \times 3 = 9$

Bir kullanıcı isteği, backend'de 9 attempt'e dönüşebilir.

Arıza döngüsü

Backend yavaşlar.



Timeout'lar başlar.



Client ve load balancer retry eder.



Backend daha fazla request alır.



Latency ve timeout daha da yükselir.

↻ döngü başa döner

Retry, kısmi arızayı **tam kesintiye** çevirebilir.

İkinci risk

Retry edilen işlem:

```
POST    /orders
POST    /payments
PATCH  /inventory
```

ise ikinci attempt yalnızca trafik yaratmayabilir. Aynı işlemi yeniden gerçekleştirebilir.

×2 Duplicate order

×2 Double payment

≠ Yanlış inventory

×2 Tekrarlanan notification

Non-idempotent request'lerde retry bir availability ayarı değil, **data consistency riskidir.**

Friday check

Sisteminizde retry yapan katmanları bulun:

Client / SDK

CDN / WAF

Load Balancer

Ingress / Service Mesh

Application

● ● ● NGINX

```
$ nginx -T 2>&1 | grep -E \  
'proxy_next_upstream|proxy_next_upstream_tries|proxy_next_upst
```

● ● ● HAProxy

```
$ grep -E \  
'retries|retry-on|redispatch' \  
/etc/haproxy/haproxy.cfg
```

Kaç katmanın retry yaptığını biliyor musunuz?

Gizli trafiği ölçün

İzlenmesi gereken temel oran:

Upstream Attempts

hedef

Incoming Requests

≈ 1

Bu oran 1'den uzaklaşıyorsa, sistemin görünmeyen trafiği büyüyor demektir.

Birlikte izleyin:

Retry rate ve retry reason

Exhausted retries

Backend saturation

Timeout ve p99 latency

Duplicate transaction

Retry sonrası başarı oranı

Başarısız olan retry, sadece incident'a **yeni trafik ekler.**

Reliability ilkesi

Retry is not free capacity.

Retry politikalarını:

- 01 Sadece geçici hatalarla sınırlayın
- 02 Toplam retry budget belirleyin
- 03 Exponential backoff ve jitter kullanın
- 04 Katmanlar arasında koordine edin
- 05 Non-idempotent işlemleri koruyun

Bir retry request'i kurtarabilir.

Kontrolsüz retry **outage'ı büyütebilir.**

Don't Deploy on Friday. Verify on Friday.

Every Friday. One Production Check.