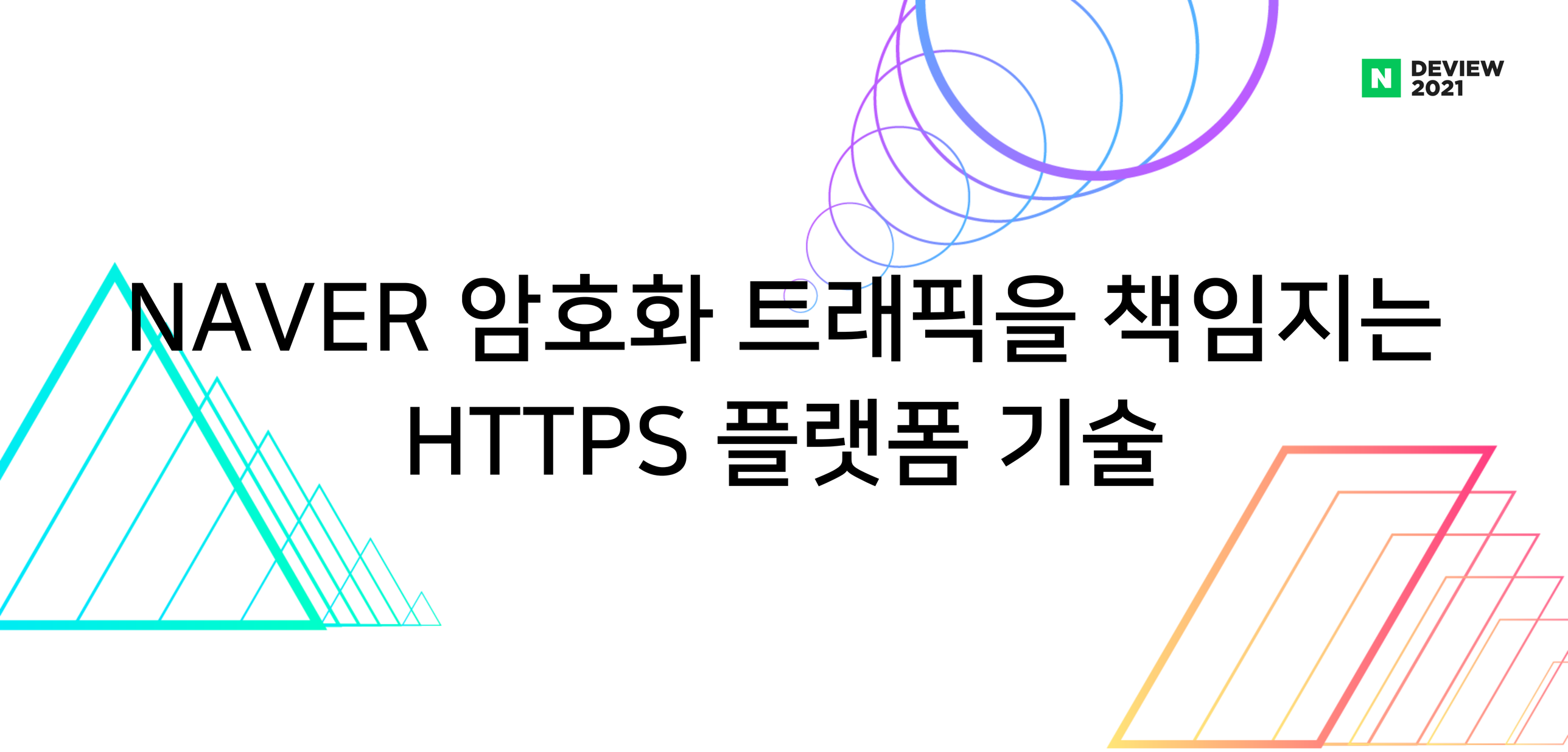


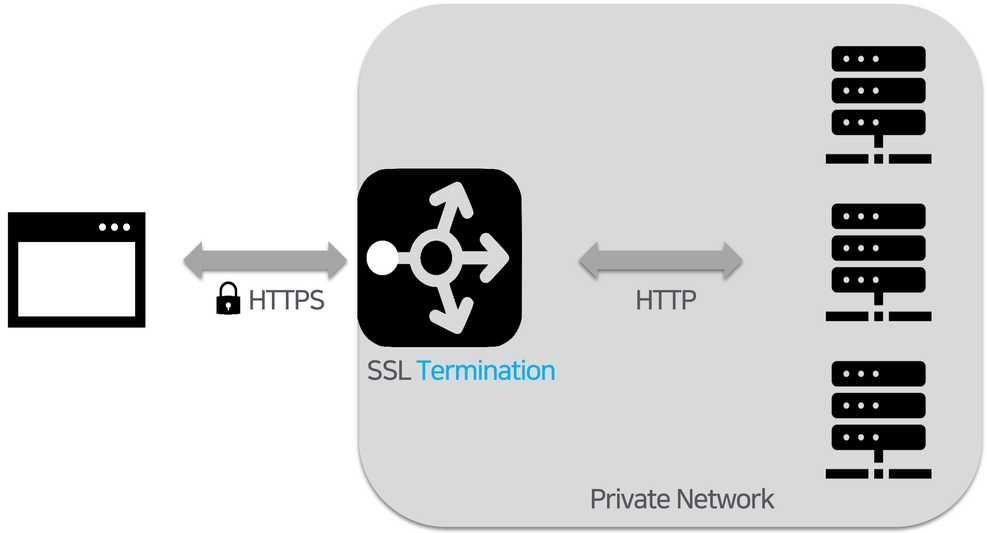
NAVER 암호화 트래픽을 책임지는 HTTPS 플랫폼 기술

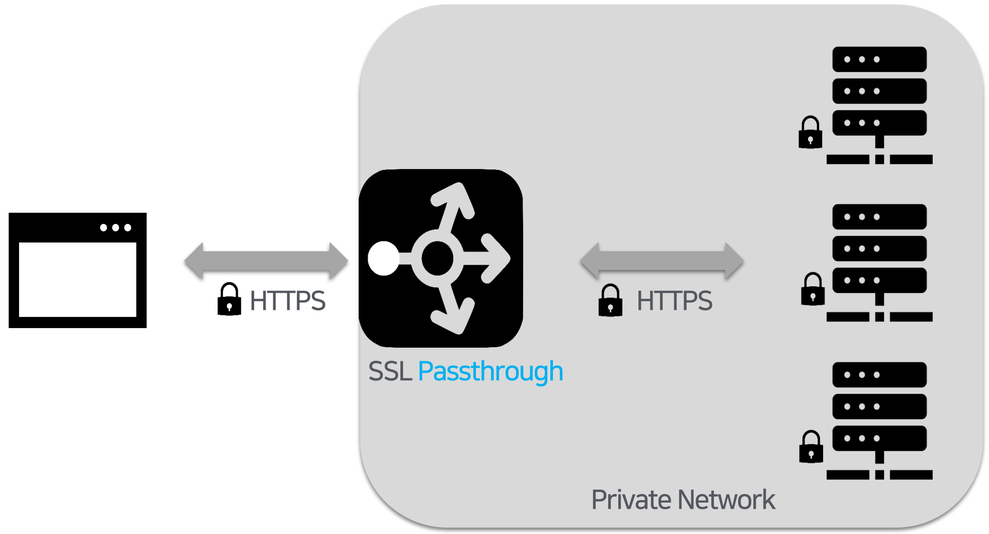
공진호, 김해랑 / NAVER CLOUD

CONTENTS

1.SSL Termination Platform (nFront)

1.1 SSL Termination?





암호화 트래픽에 대한 연산 작업을 대행 해주는 기능

- 복잡한 수학 연산으로 인한 CPU 리소스 확보
- 인증서 관리 용이성
- 보안 이슈 대응 용이성
- Backend 서버 성능 향상
- 서버를 외부에 노출 하지 않을 수 있음

트래픽을 Backend로 Bypass

- Backend 까지 암호화 트래픽 유지 가능
- Backend 서버 전체 인증서 관리 필요
- Backend 서버 추가 리소스 필요

1.2 For WHAT?

HTTPS 전환을 위한 리소스 효율화

암호화 트래픽은 필수!

서비스 별 HTTPS 전환에 추가 되는 리소스를 방지

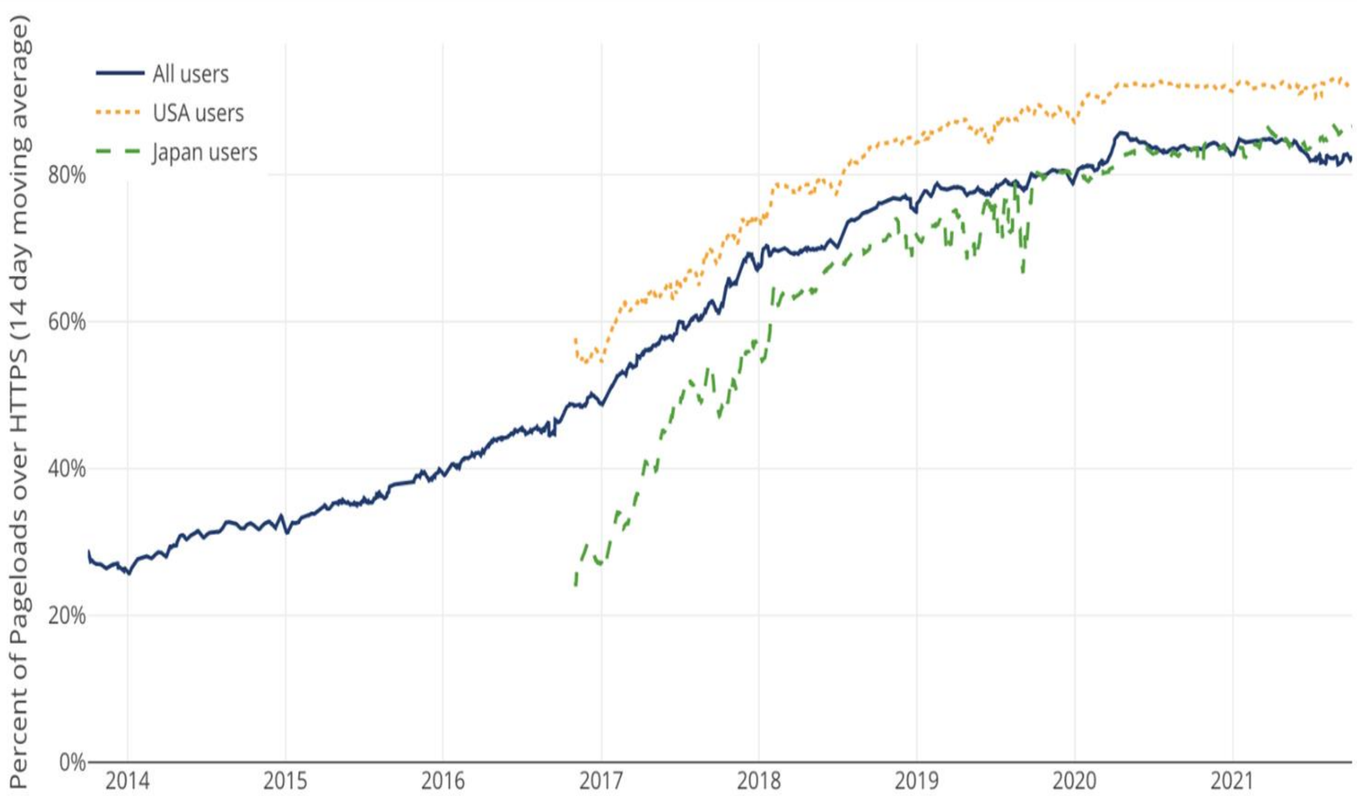
보안 대응 효율화

보안 취약점 대응 관리에 대한 어려움

(cipher, patch)

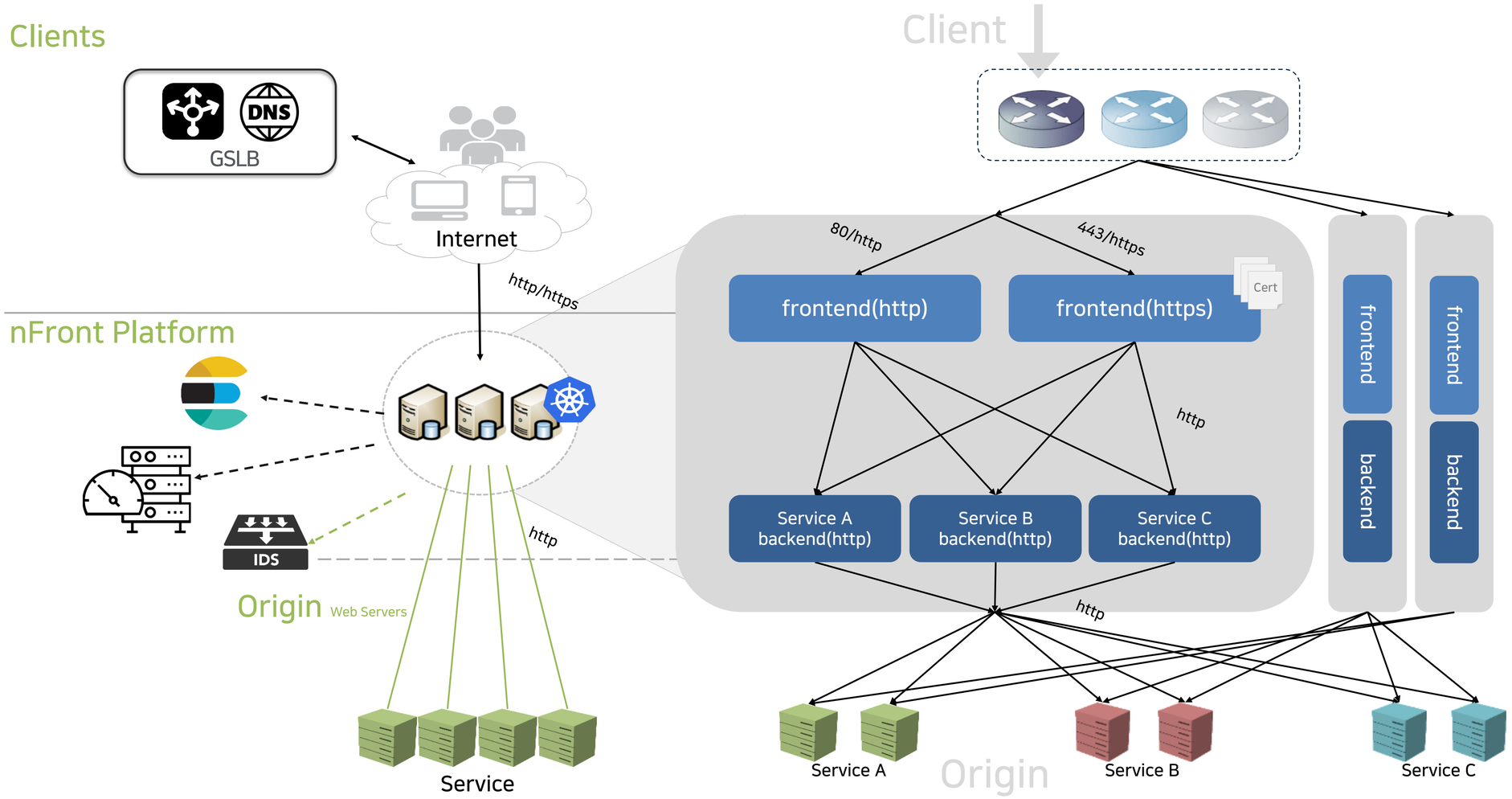
암호화 트래픽 가시성

암호화 트래픽 정책 탐지 및 차단의 어려움으로 복호화 트래픽 필요



https://docs.telemetry.mozilla.org/datasets/other/ssl/reference.html

1.3 How? => L7 Reverse Proxy



Load balancer

Anycast (RHI)

IP-hash-based Balance

Stateless

Seamless

HAproxy

frontend

to distinguish protocol

backend

to distribute by service

1.3 Why? HAproxy?

DEVIEW 2021

Latency

Proxy가 추가되어도 가장 적은 latency

Support SSL Termination

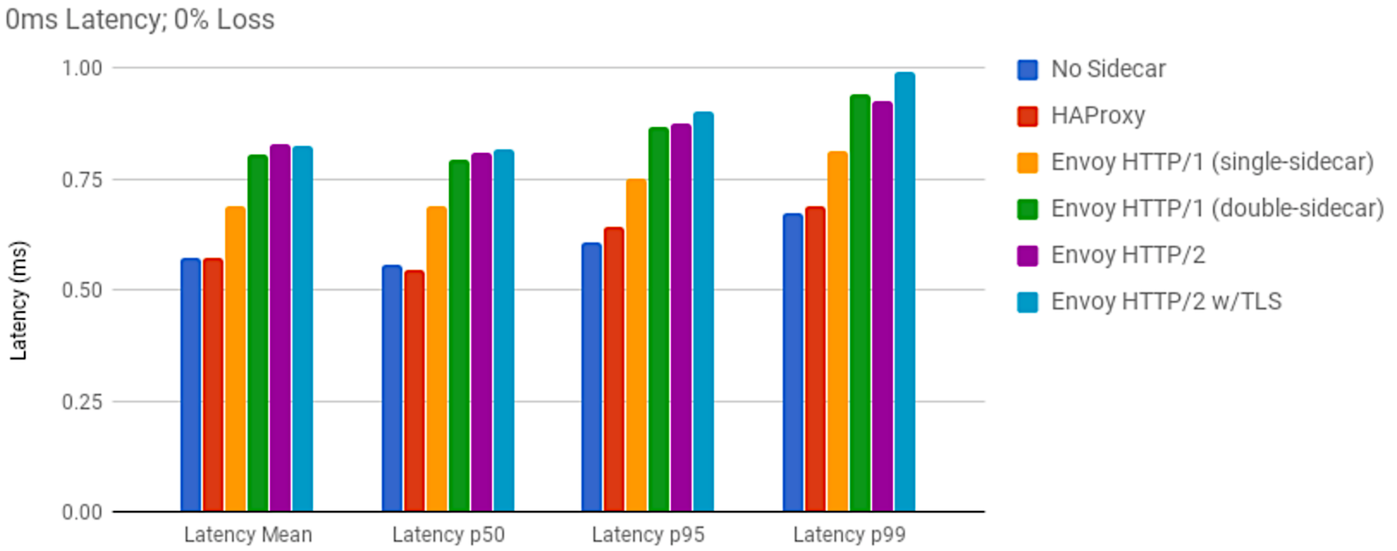
Multi bundle 사용에 최적화

Zero-Downtime

Seamless reload

Runtime api

Monitoring statistics



https://www.twilio.com/blog/2017/10/http2-issues.html

1.4 Benefits?

Performance

HTTP VS HTTPS 성능 차이
CPU 리소스 절감

Management

인증서 관리 용이

Security

취약점 대응 빠름
외부와 단절된 사용 가능

L7 Proxy

Application Layer 기능 사용

1.5 SSL Termination 전환 시 주의!

DEVIEW 2021

White list

Proxy IP에 대한 White list 추가 필요

SNI (Server Name Indication) 미 지원?

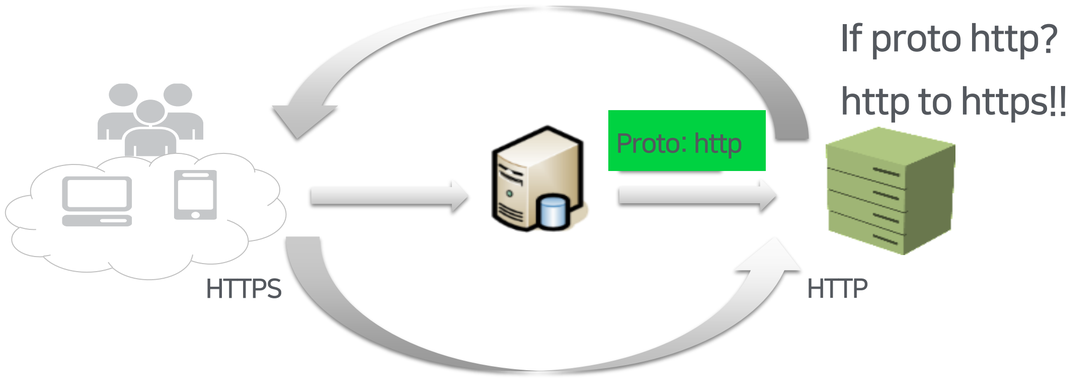
IP 추가/SAN(Subject Alternative Name) 필요

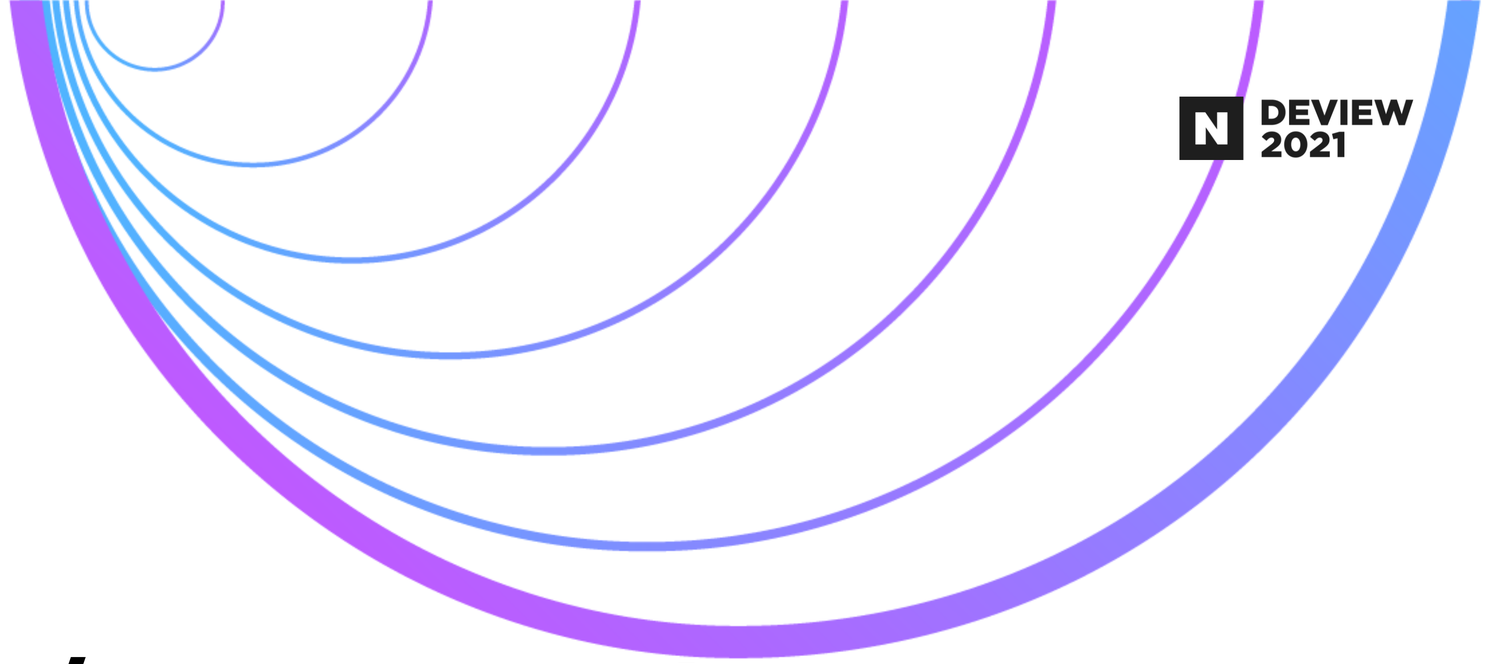
Client IP

특정 Header 값을 통한 Client IP 변환 필요

HTTPS redirect

HTTP -> HTTPS redirect 시 loop 발생 주의
특정 헤더에 proto 정보 삽입 전달 필요





2. SSL/TLS Tuning

2.1 SSL Performance Impacts

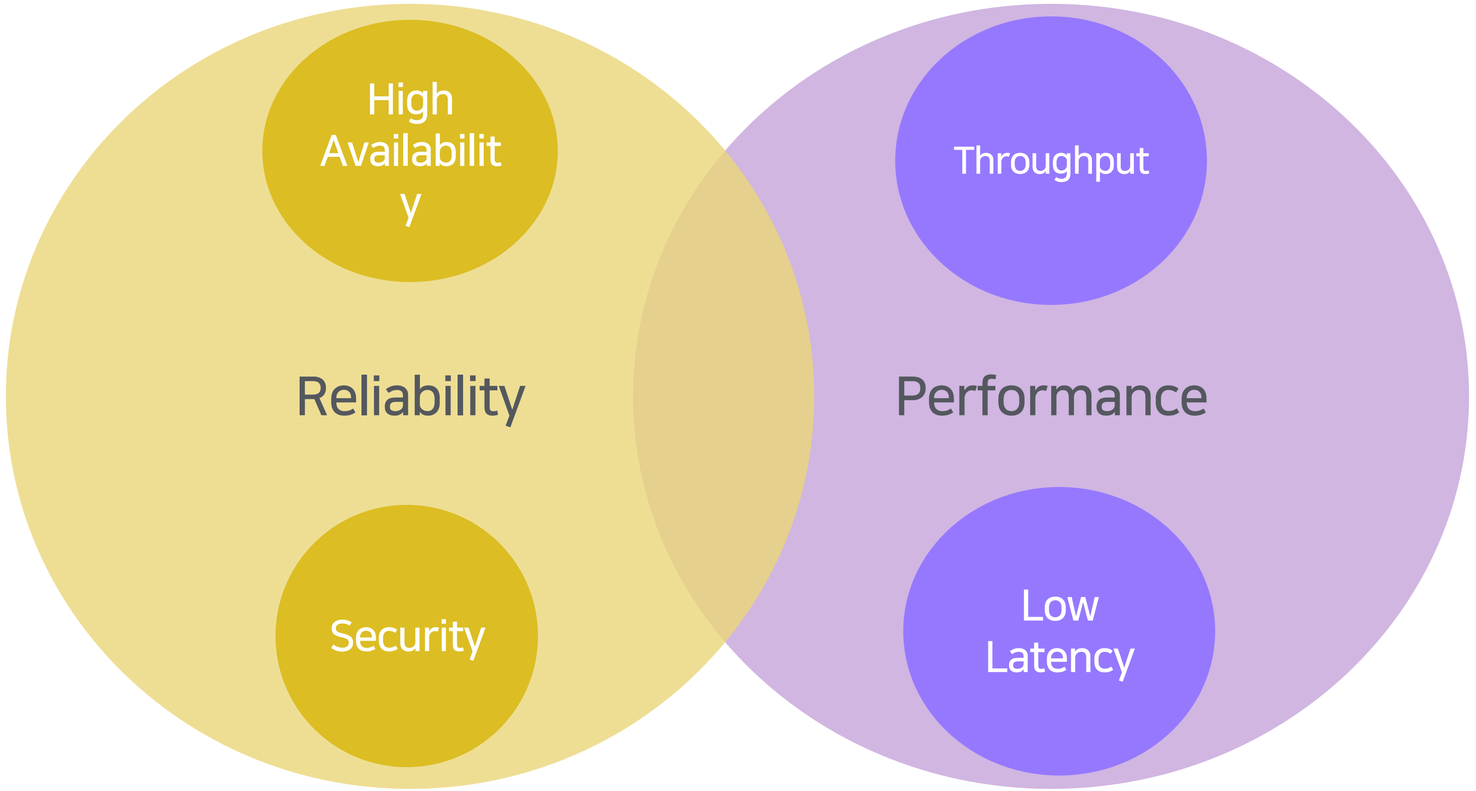
DEVIEW 2021

Hardware
- CPU
- NIC

Protocol version
- HTTP/1.1 < http/2 < HTTP/3 (QUIC)
- TLS 1.2 < TLS 1.3

Network status
- Latency and RTT
- Bandwidth
- QoS

TLS overhead
- Asymmetric key size
- symmetric algorithm
- Session resumption
- OCSP stapling



2.2 Hardware Tuning

Network / Facility

- GSLB/CDN
- Anycast L4 (Source Hash)
- Global Edge
- ISP line locate

CPU

- Core pinning (affinity)

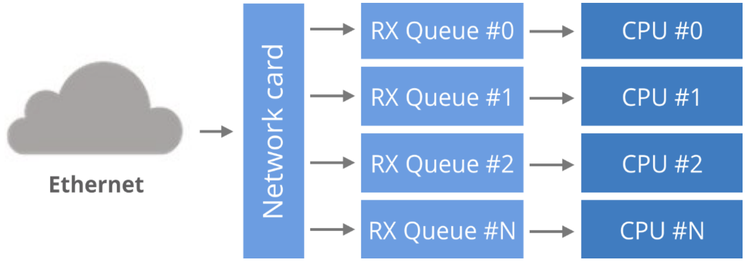
BIOS

- Disable C/P-state
- Disable HT for Core Affinity

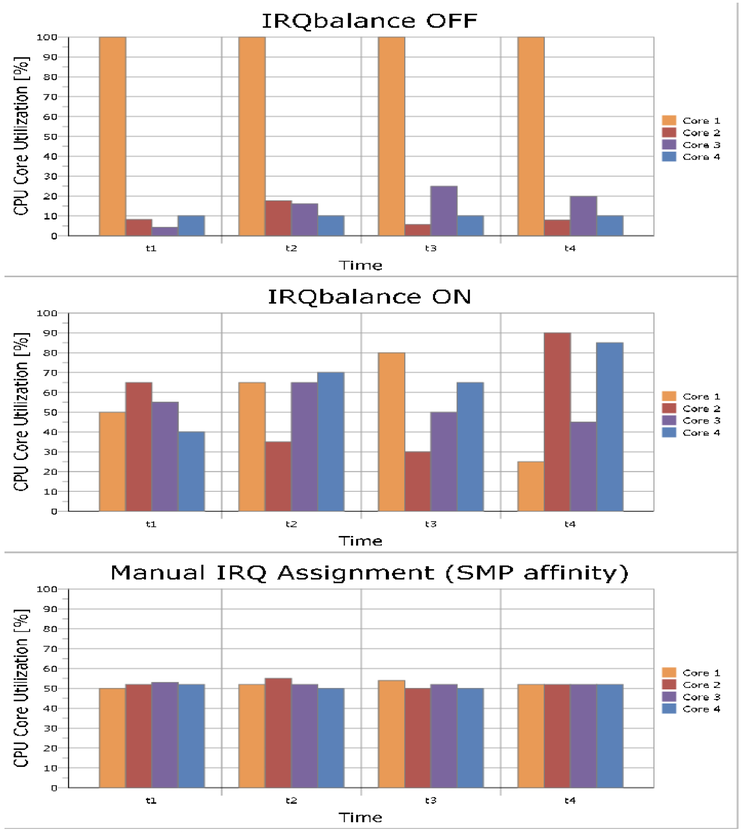
NIC

- Fiber NIC (20G or more)
- Ring Buffer (ethtool)
- Offloading (ethtool)
- Numa slot (irq)

2.3 Network IRQ Affinity



https://singhblogging.wordpress.com/2017/02/10/crash-course-to-multi-queue-nics/



https://www.researchgate.net/figure/Illustration-of-CPU-Core-Utilization-with-different-kinds-of-Interrupt-Handling_fig3_253241645

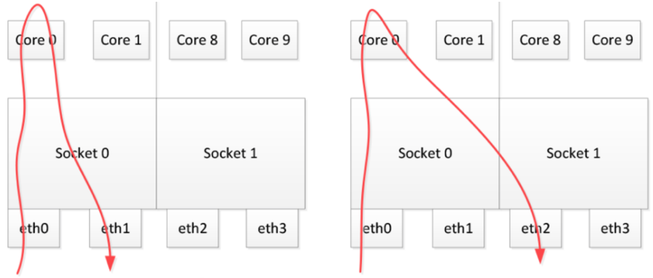


Figure 12 **Good**

Figure 13 **RX OK**

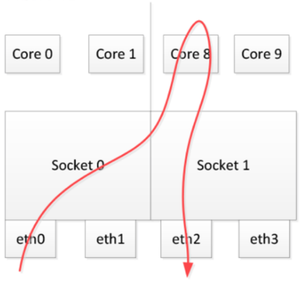


Figure 14 **TX OK**

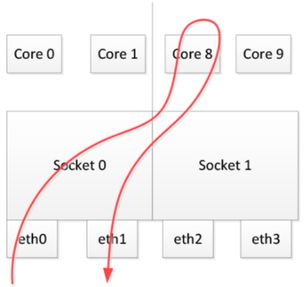


Figure 15 **Bad**

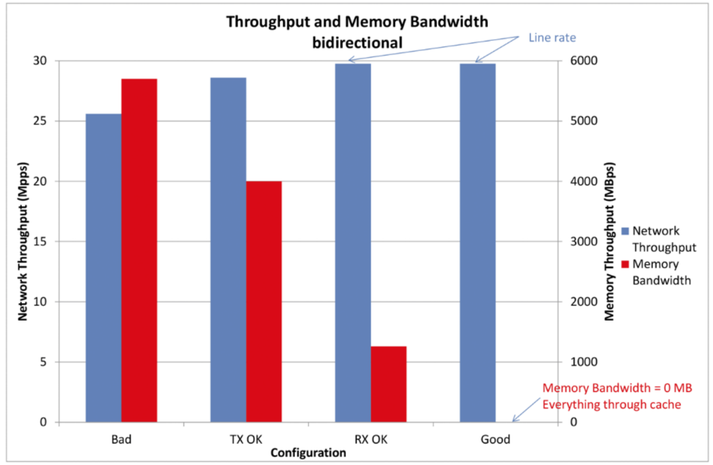


Figure 16

http://docplayer.net/5271505-Network-function-virtualization-virtualized-bras-with-linux-and-intel-

NIC queue에 core 할당

numa node에 맞는 core 할당 필요

Numa node 체크

```
# cat /sys/class/net/eth?/device/local_cpulist
# lstopo  ( hwloc 설치 필요 )
```

Irqbalance stop 및 affinity 설정

```
# systemctl  stop lrqbalance
# set_irq_affinity {core} eth?
Ex ) echo ‘core bit’ > /proc/irq/$IRQ/smp_affinity
```

Tunnel interface

Single Queue로 Core 분배 어려움

RPS 로 Multi Core 지정 필요

```
# echo ‘ffffff’ > /sys/class/net/tunl0/queues/rx-0/rps_cpus
```

2.4 Kernel Tuning

DEVIEW 2021

Increase
- fs.file-max
- net.ipv4.tcp_max_syn_backlog
- net.core.netdev_max_backlog
- net.core.somaxconn
- net.core.{r,w}mem_max
- net.ipv4.tcp_{r,w}mem
- net.ipv4.ip_local_port_range
- net.ipv4.tcp_max_tw_buckets
- net.netfilter.nf_conntrack_max (or disable)

Reduce (default 5)
- net.ipv4.tcp_syn_retries
- net.ipv4.tcp_synack_retries

Enable
- net.ipv4.tcp_syncookies
- net.ipv4.tcp_tw_reuse
- net.ipv4.tcp_timestamps
- net.ipv4.tcp_window_scaling

ETC
- net.ipv4.tcp_congestion_control = bbr

packet drop check

```
# ethtool –s eth?
# cat /proc/net/softnet_stat
```

accept queue full

```
# nstat –az|grep –i listen
TcpExtListenOverflows   xx
TcpExtListenDrops xx
```

listen backlog (**더 작은 값이 기준**)

net.core.somaxconn = 65534

Application: backlog 70000

```
# ss –nlp sport = :443
```

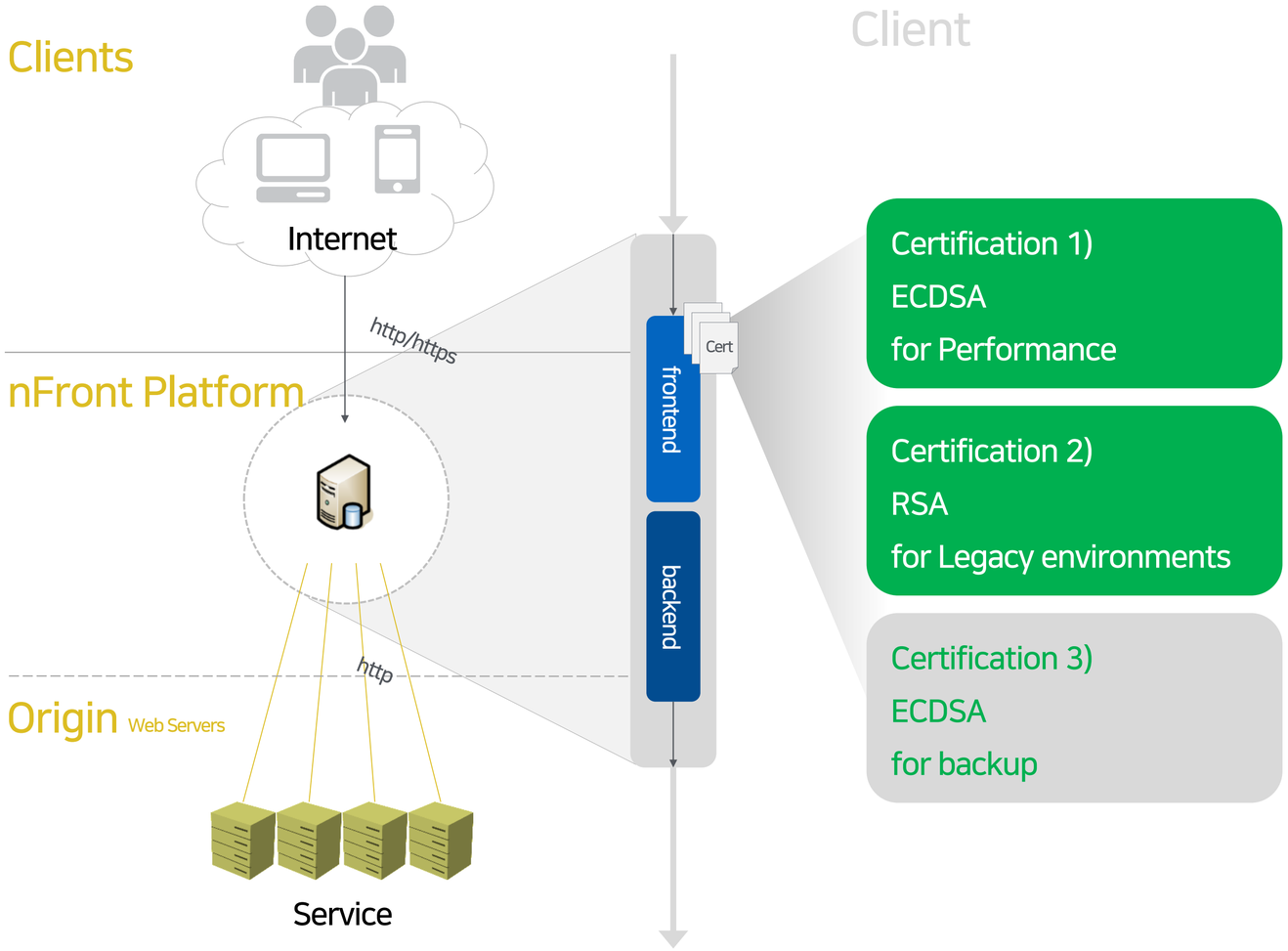
```
Netid  State      Recv-Q Send-Q                                    Local Address:Port
tcp    LISTEN     0      65534                                           *:443
 users:(("haproxy",pid=51630,fd=9))
```

tcp info

```
# ss –i  or # ip
```

```
tcp    ESTAB      0      0                                          :26560
         bbr wscale:7,11 rto:109 rtt:8.194/8.893 ato:40 mss:1428 rcvmss:536 advmss:1448 cwnd:4035 bytes_acked:300264431 bytes_receive
ata_segs_out:388114 data_segs_in:905873 bbr:(bw:439.4Mbps,mrtt:2.336,pacing_gain:2.88672,cwnd_gain:2.88672) send 5625.6Mbps lastsnd:1
2147.9Mbps delivery_rate 439.4Mbps app_limited busy:2218240ms retrans:0/69847 rcv_rtt:436931 rcv_space:43051 minrtt:0.124
```

2.5 Hybrid Certificate & Asymmetric Key



Certification 1)
ECDSA
for Performance

Certification 2)
RSA
for Legacy environments

Certification 3)
ECDSA
for backup

Type of Signature		Private Key (Bytes)	CSR Size (Bytes)	Certificate Size (Bytes)	Certificate Verification Computational Time (ms)	Certificate Transmission Time (ms)	Total Time for Authentication (ms)
Algorithm	Key Size/Curve						
RSA	1024	891	660	847	7	6.1	13.1
	2048	1675	1013	1200	9	8.4	17.4
	3072	2455	1358	1545	9	9.0	18
	7680	5977	2918	3105	10	13.4	23.4
	15,360	11,823	5518	5709	12	16.9	28.9
ECDSA	secp224r1	278	530	700	7	5.1	12.1
	secp521r1	436	737	904	9	6.4	15.4
	prime192v1	270	505	668	8	5.0	13
	prime256v1	302	542	696	7	5.0	13
	brainpoolP384r1	367	627	778	8	5.2	13.2
	brainpoolP512r1	436	725	875	9	6.2	15.2
	brainpoolP384t1	367	639	794	8	6.0	14
	brainpoolP512t1	436	729	887	9	6.2	15.2

https://itwiki.kr/w/%ED%83%80%EC%9B%90_%EA%B3%A1%EC%84%A0_%EC%95%94%ED%98%B8

성능을 생각 한다면 ECDSA

2.6 Symmetric Algorithm

안정성/성능 고려

-- TLS 1.2
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
(0xcca8)

-- TLS 1.3
TLS_AES_128_GCM_SHA256
TLS_AES_256_GCM_SHA384
TLS_CHACHA20_POLY1305_SHA256

암호 알고리즘(대칭키)

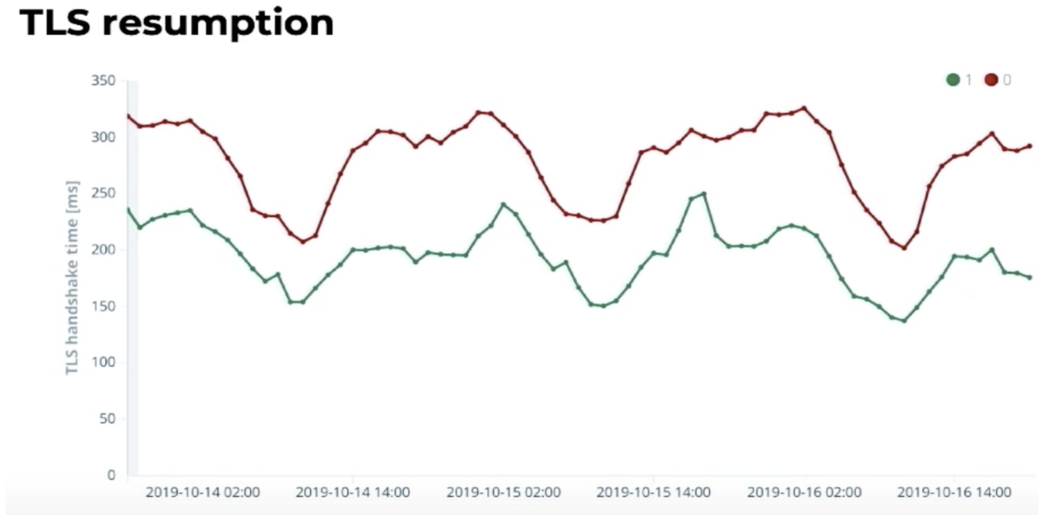
AES_GCM, CHACHA20

2.7 TLS Session Resumption

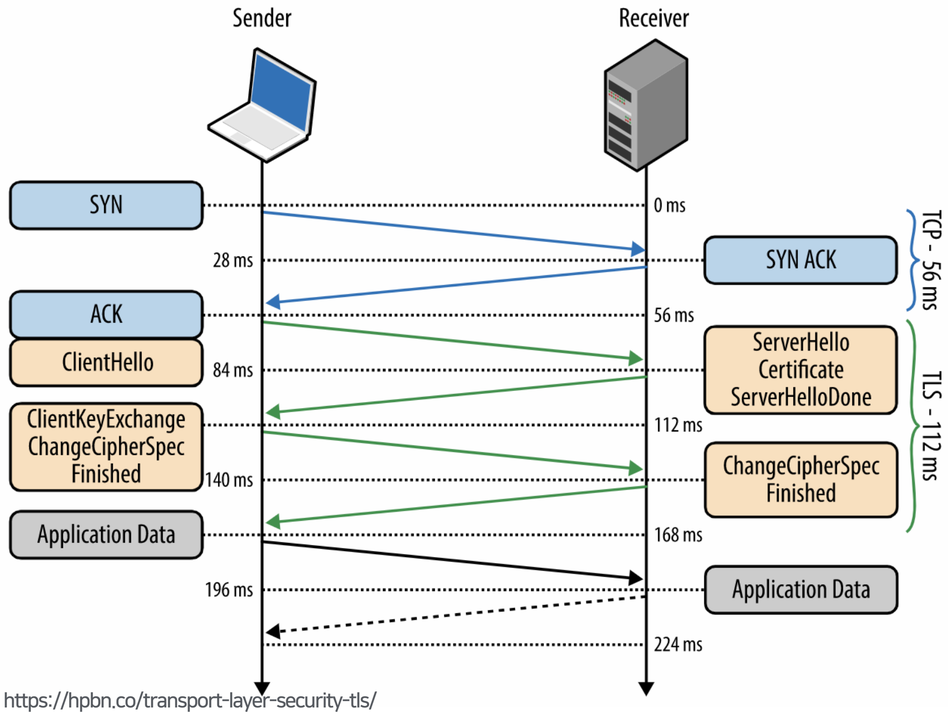
DEVIEW 2021

Abbreviated TLS Handshake (for Latency)

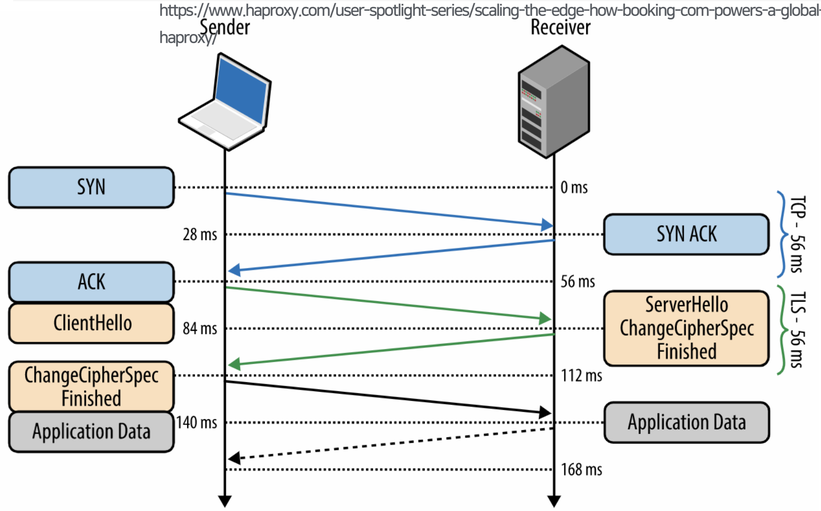
- Session ID
- Session Ticket



https://www.haproxy.com/user-spotlight-series/scaling-the-edge-how-booking-com-powers-a-global-application-delivery-network-with-haproxy/



https://hpbn.co/transport-layer-security-tls/



New Connection: 4 RTT

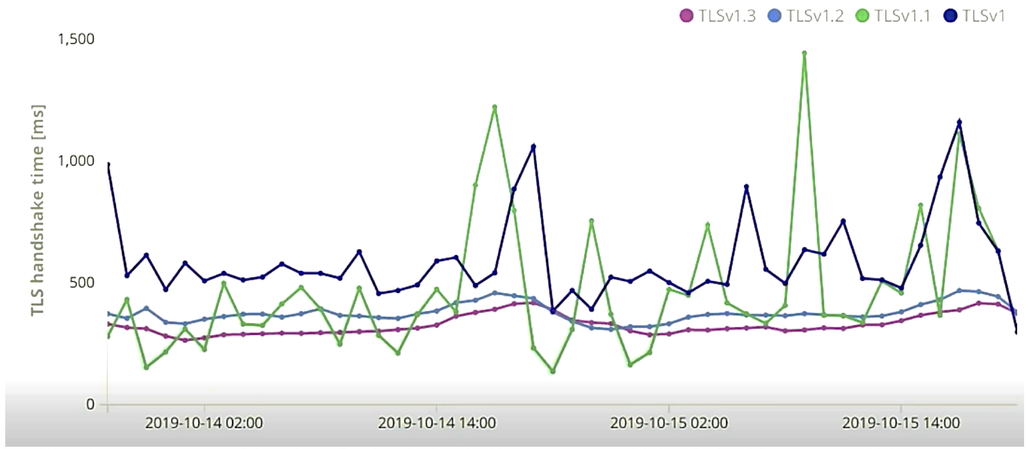
Resumed Connection: 3 RTT

2.8 Use TLS ver 1.3

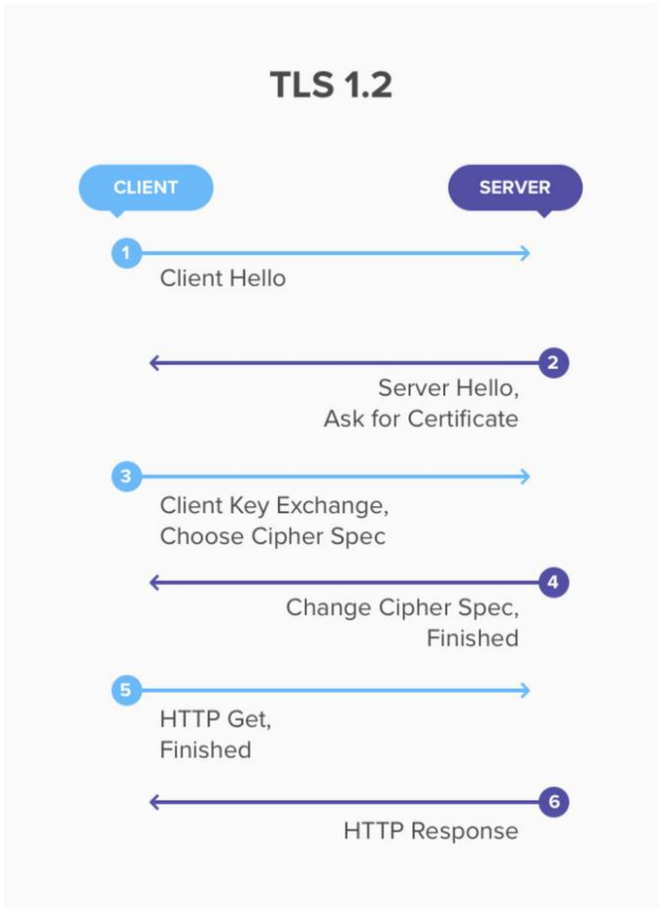
보안은 강화되고 속도는 올라감

- 취약 Cipher 제거 (암호화 모음 단순화)
- 비 대칭키 타원 곡선 고정
- TLS handshake (Key_share)

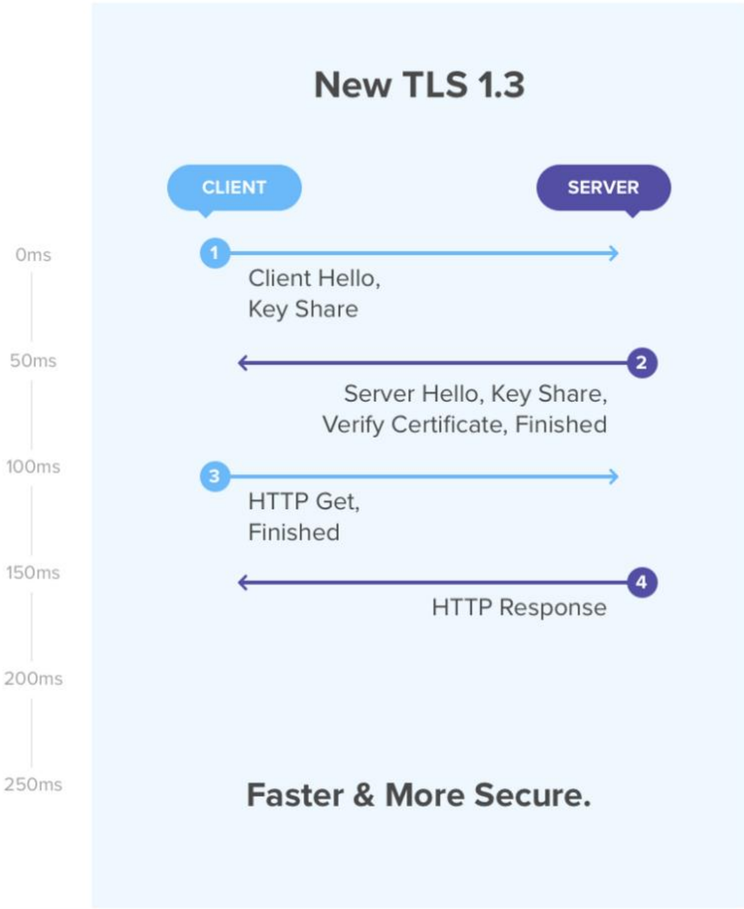
TLS handshake latency



https://www.haproxy.com/user-spotlight-series/scaling-the-edge-how-booking-com-powers-a-global-application-delivery-network-with-haproxy/



https://medium.com/@vanrijn/what-is-new-with-tls-1-3-e991df2caaac



0ms
50ms
100ms
150ms
200ms
250ms

New Connection: 3 RTT

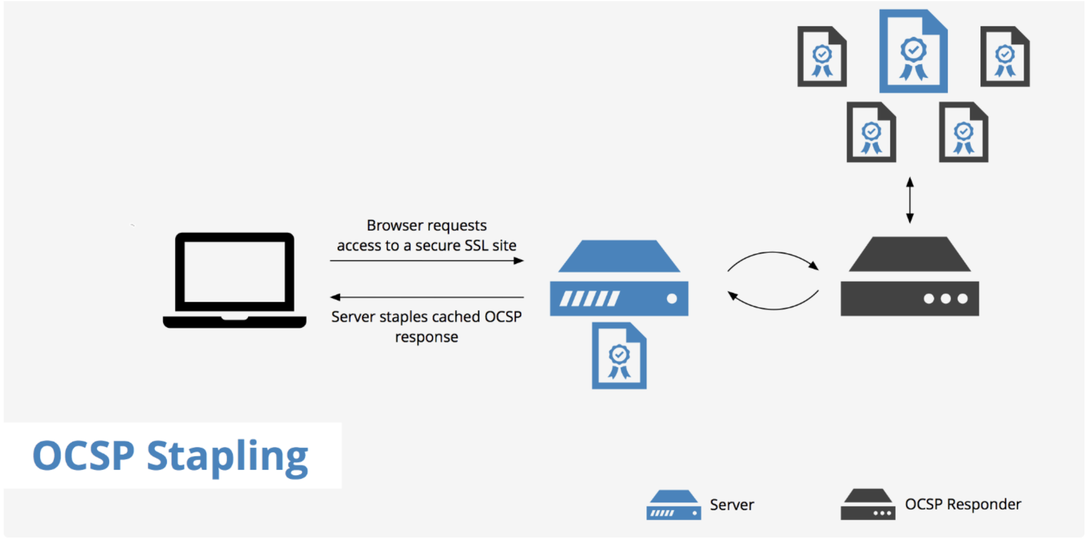
Resumed Connection: 3 RTT

0-RTT (early data) : 2 RTT (**보안 취약**)

2.9 TLS ETC.

DEVIEW 2021

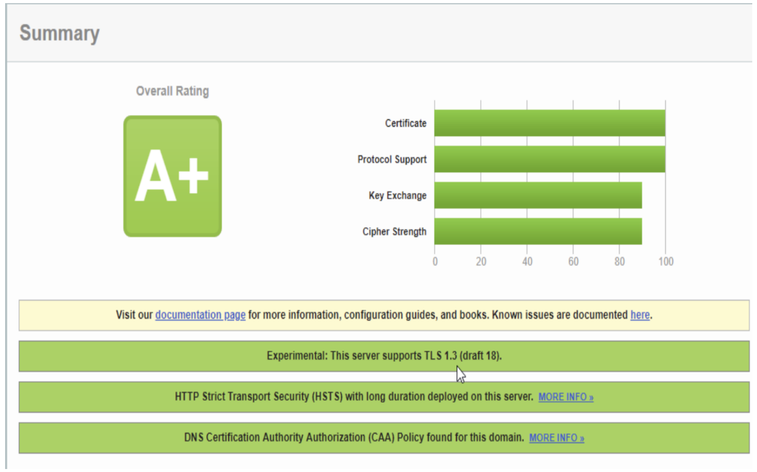
Use OCSP Stapling



https://rsec.kr/?p=386

```
CONNECTED(00000003)
OCSP response:
====================================
OCSP Response Data:
    OCSP Response Status: successful (0x0)
    Response Type: Basic OCSP Response
    Version: 1 (0x0)
    Responder Id: 0ABC0829178CA5396D7A0ECE33C72EB3EDFBC37A
    Produced At: Oct  5 00:54:49 2021 GMT
    Responses:
    Certificate ID:
      Hash Algorithm: sha1
      Issuer Name Hash: 2B1D1E98CCF37604D6C1C8BD15A224C804130038
      Issuer Key Hash: 0ABC0829178CA5396D7A0ECE33C72EB3EDFBC37A
      Serial Number: 0CDB20BC68647CE0902332460342B84A
    Cert Status: good
    This Update: Oct  5 00:39:03 2021 GMT
    Next Update: Oct 11 23:54:03 2021 GMT

    Signature Algorithm: ecdsa-with-SHA384
```

$ openssl s_client –connect xxx:443 –status



https://ssllabs.com check

Reduce Redirect (http -> https)

- Use HSTS(HTTP Strict Transport Security)
- Use permanent redirect (301, 307)

Optimize chain and certificate

- Remove Root CA key
- Reduce SAN (use wildcard)

Cert Life cycle

- Less than 1 year
- New key when renew

2.10 Application (HAproxy) 설정

CPU affinity (taskset)

Net irq 설정 하지 않은 Socket의 Core 할당 필요

```
### CPU Mapping ###
cpu-map auto:1/15-22 12 13 14 15 16 17 18 19
cpu-map auto:1/1-14 2 3 4 5 6 7 8 9 10 11 20 21 22 23
```

Connection reuse

Backend서버와 connection pooling 사용

Client Keep-Alive

CDN 사용외 Static 파일 Cache

Listen Backlog 설정

Max connection

frontend / backend

set-dumpable

Core dump 용도

Tcp Fast open

Timeout

Client/server/req/res

Session resumption

OCSP stapling

2.11 SSL Config Generator

DEVIEW 2021

moz://a SSL Configuration Generator

Server Software

- ○ Apache
- ○ AWS ALB
- ○ AWS ELB
- ○ Caddy
- ○ Dovecot
- ○ Exim
- ○ Go
- ○ HAProxy
- ○ Jetty
- ○ lighttpd
- ○ MySQL
- ◉ nginx
- ○ Oracle HTTP
- ○ Postfix
- ○ PostgreSQL
- ○ ProFTPD
- ○ Redis
- ○ Tomcat
- ○ Traefik

Mozilla Configuration

- ○ Modern
  Services with clients that support TLS 1.3 and don't need backward compatibility
- ◉ Intermediate
  General-purpose servers with a variety of clients, recommended for almost all systems
- ○ Old
  Compatible with a number of very old clients, and should be used only as a last resort

Environment

Server Version	1.17.7
OpenSSL Version	1.1.1l

Miscellaneous

- ☑ HTTP Strict Transport Security
  This also redirects to HTTPS, if possible
- ☑ OCSP Stapling

nginx 1.17.7, intermediate config, OpenSSL 1.1.1l

Supports Firefox 27, Android 4.4.2, Chrome 31, Edge, IE 11 on Windows 7, Java 8u31, OpenSSL 1.0.1, Opera 20, and Safari 9

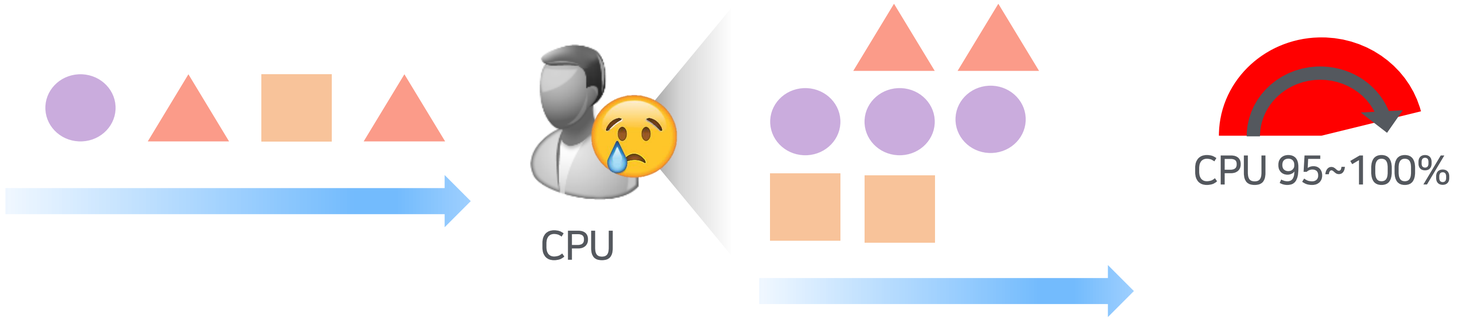
```
# generated 2021-09-30, Mozilla Guideline v5.6, nginx 1.17.7, OpenSSL 1.1.1l, intermediate configuration
# https://ssl-config.mozilla.org/#server=nginx&version=1.17.7&config=intermediate&openssl=1.1.1l&guideline=5.6
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    location / {
        return 301 https://$host$request_uri;
    }
}

server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;

    ssl_certificate /path/to/signed_cert_plus_intermediates;
    ssl_certificate_key /path/to/private_key;
    ssl_session_timeout 1d;
    ssl_session_cache shared:MozSSL:10m;  # about 40000 sessions
    ssl_session_tickets off;

    # curl https://ssl-config.mozilla.org/ffdhe2048.txt > /path/to/dhparam
    ssl_dhparam /path/to/dhparam;

    # intermediate configuration
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-
ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384;
    ssl_prefer_server_ciphers off;
```

Application & Version별 SSL 설정 지원

https://ssl-config.mozilla.org/

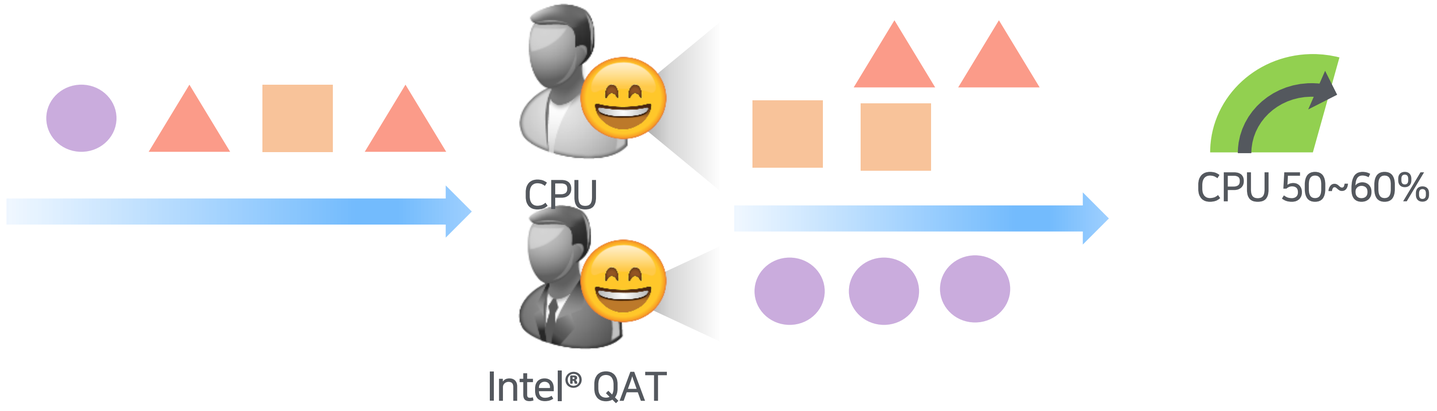
2.12 Accelerate Encrypt/Decrypt (H/W)

DEVIEW 2021



openssl + QAT engine build

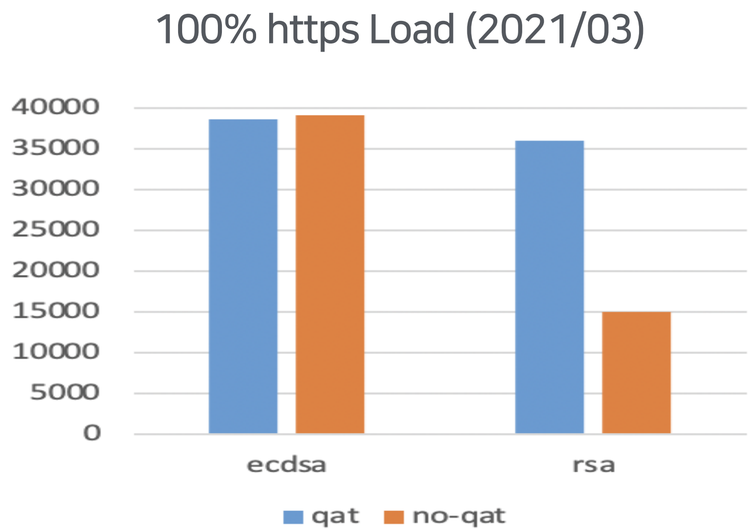
https://github.com/intel/QAT_Engine



Intel® QuickAssist Technology (QAT) = HW Chipset + SW Engine

- Accelerate Encrypt/Decrypt
- Reduce CPU Load
- Compatibility with Any Intel Architecture

2.12 Accelerate Encrypt/Decrypt (H/W)



With RSA certs, GOOD

With ECDSA certs, Not GOOD

on the Next Level?

Intel® Ice Lake

better performance on encryption/decryption

alternative to the other HW chipset

Upcoming SW Engine

enhanced when ECDSA certs

enhanced with HAproxy, Nginx

Ice Lake AES-NI instructions 성능 향상

AVX-512기반의 새로운 instruction

ECDSA 약 60%

RSA 약 300%

https://www.intel.com/content/www/us/en/developer/articles/guide/building-software-acceleration-features-in-the-intel-qat-engine-for-openssl.html

3. L7 Reverse Proxy

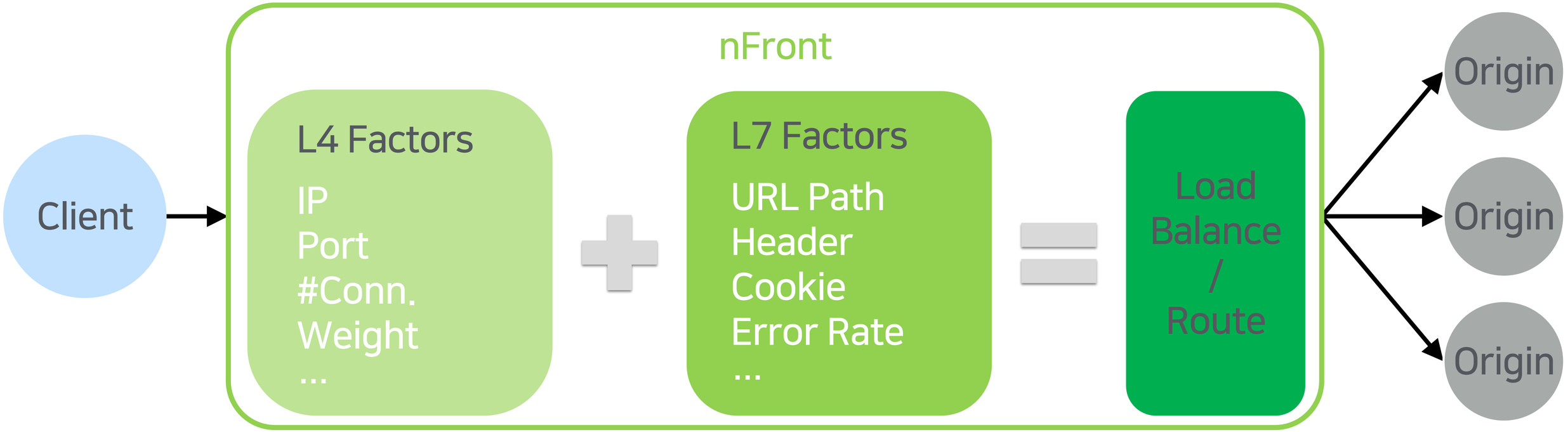
3.1 Load Balancer, LB

Layer 4 LB Methods를 기본 제공하며,

Layer 7 Data 기반으로, 더욱 고도화된 LB 제공

ex) Circuit Breaking, Content Serving 등

NAVER 서비스 단위 목적으로, 기본적으로 Host Header 기반 Route Architecture 구성



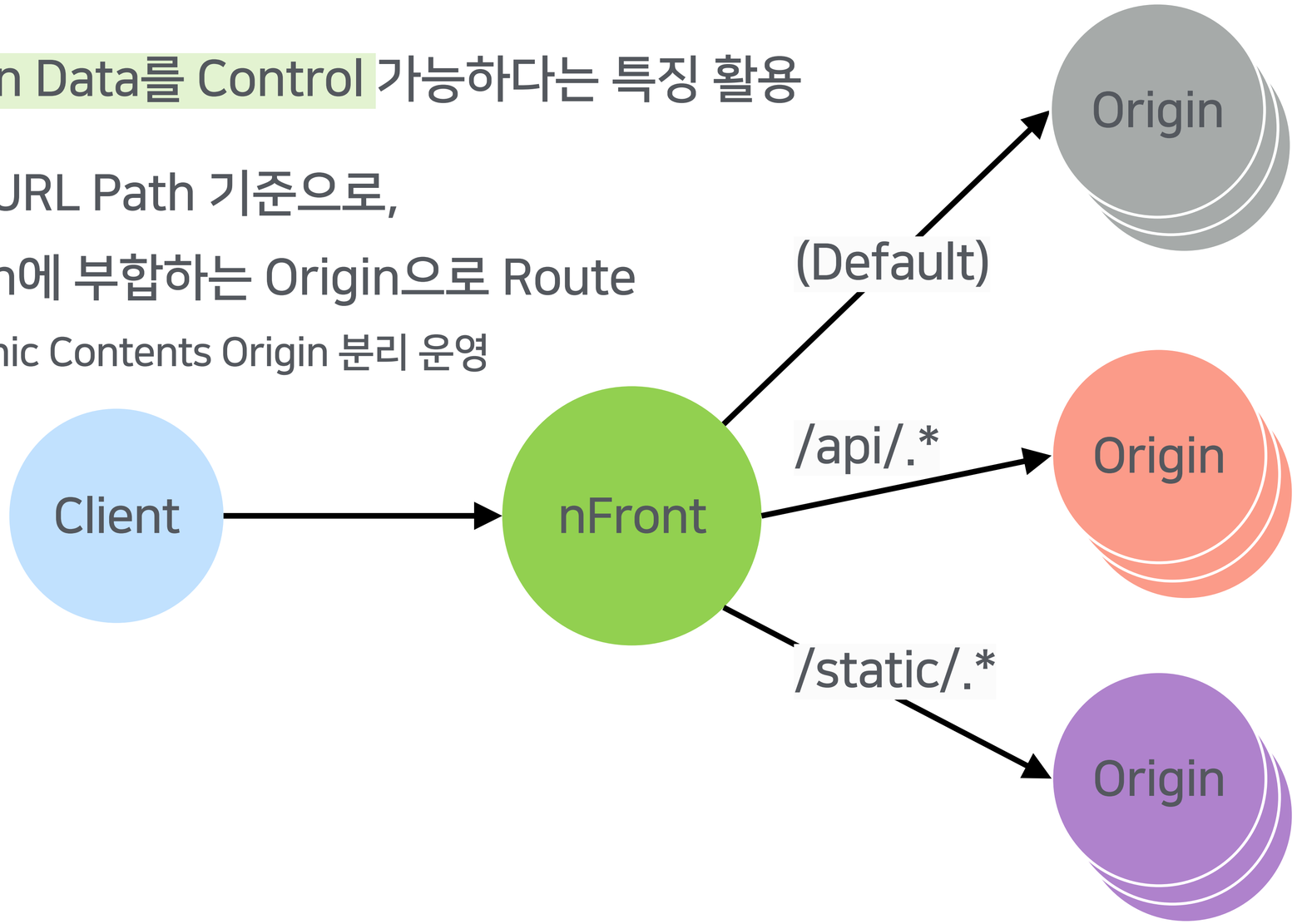
3.2 Path Based Route

L7 Application Data를 Control 가능하다는 특징 활용

Client 요청의 URL Path 기준으로,

정의된 Pattern에 부합하는 Origin으로 Route

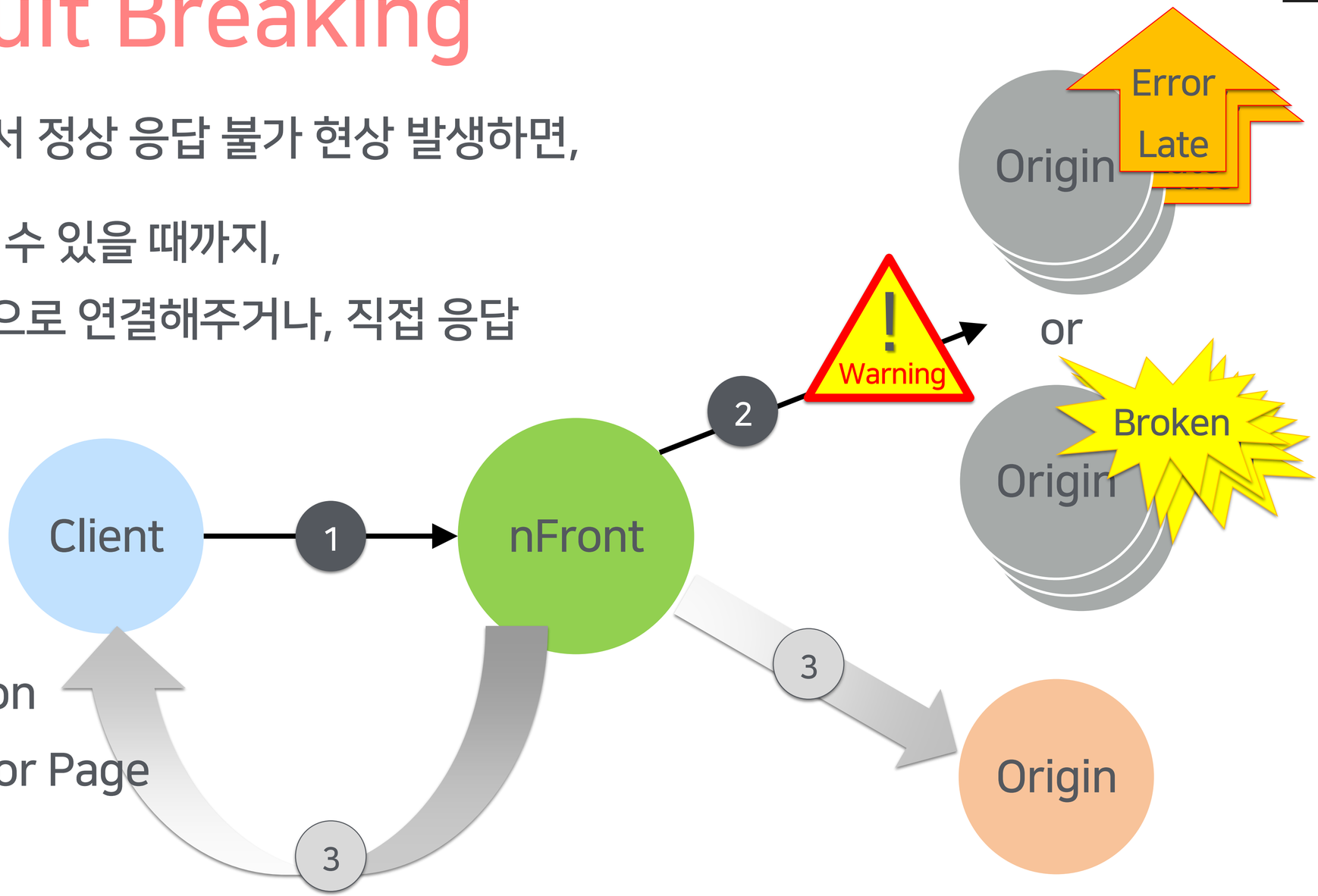
ex) Static/Dynamic Contents Origin 분리 운영



3.3 Circuit Breaking

모든 Origin에서 정상 응답 불가 현상 발생하면,

정상 서비스 할 수 있을 때까지,
다른 Origin군으로 연결해주거나, 직접 응답



- Redirection
- Static Error Page
- Parking

...

3.4 Web Application Firewall, WAF

HTTPS 공격 차단

암호화된 Data에 대해서도

복호화하여 L7 Data 기반으로

Origin 보호

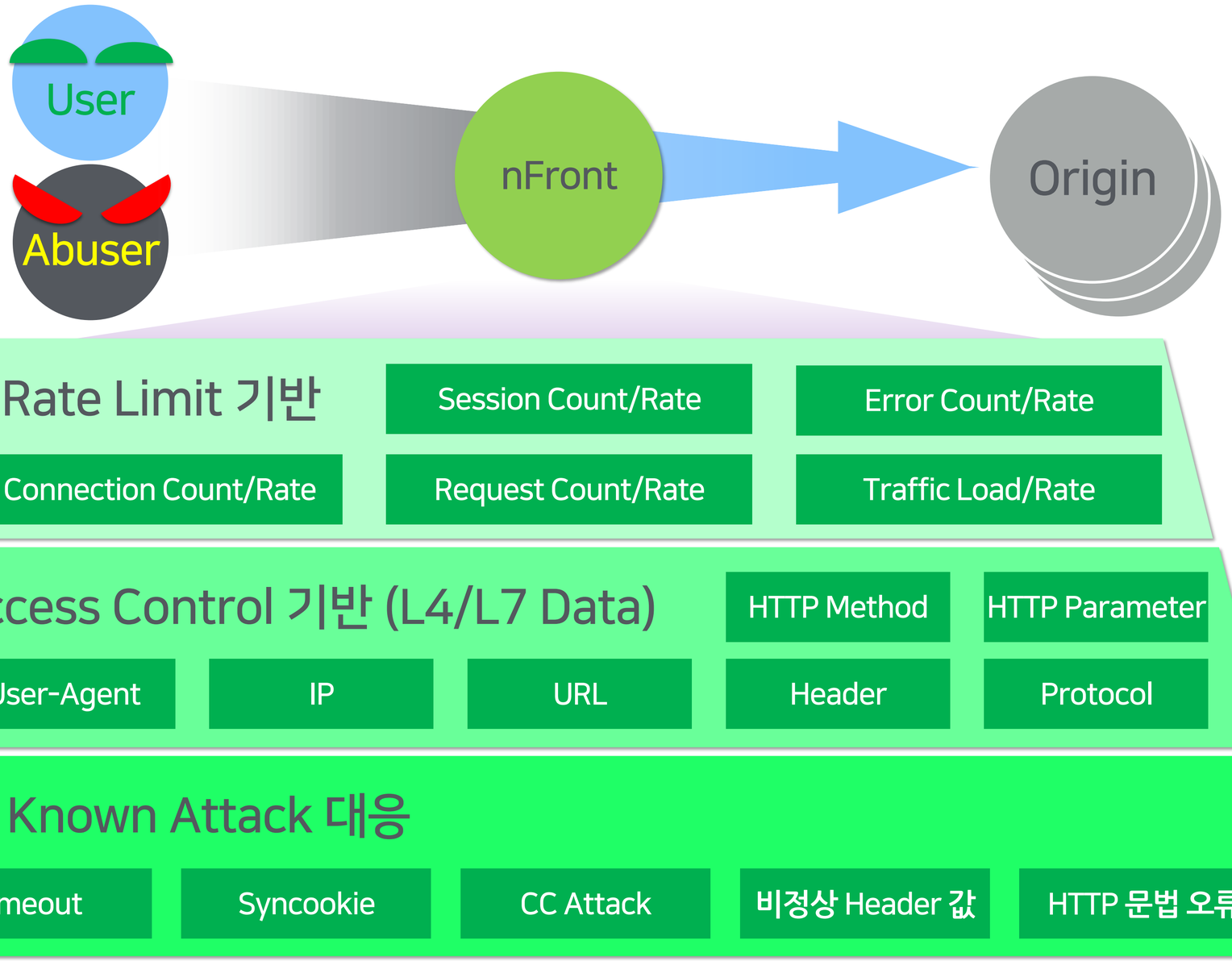
Origin Resource 미사용

Platform Resource만으로 제어

Seamless 적용

기존 서비스 영향 없이

제어 Pattern 변경 적용



4. Management/Monitoring

4.1 Certificates Management

Multiple Chains of Trust

Chain of Trust 이슈에 대비하여,

여분의 Chain of Trust 준비

OCSP Stapling

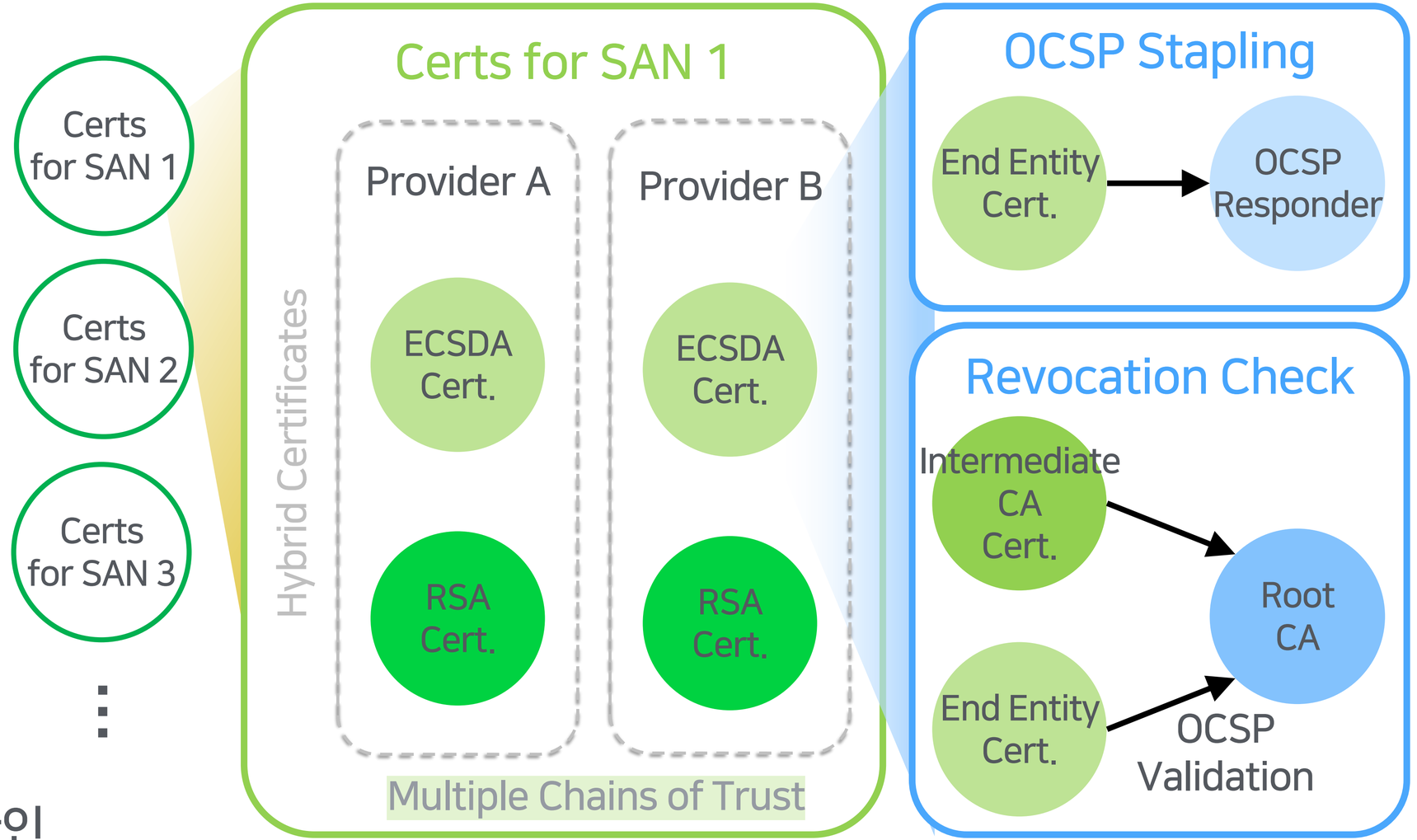
Client 접속 품질 개선을 위해,

OCSP를 주기적으로 Update

Revocation Check

Component의 유효성을 검증

Chain of Trust 전체의 Health 확인



4.2 Monitoring

DEVIEW 2021

Resource Data: RPS, Traffic, Error, CPU Usage, ···

Deployed Data: Config Files, Certificates, ···

Proxy Node

every 1 Min

HAPROXY

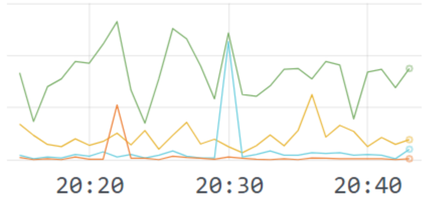
Resource Data

python

telegraf

OPENTSDB

Grafana



NAVER Developers

every 1 Day

010
110
001

Deployed Data

BASH
THE BOURNE-AGAIN SHELL

MySQL

python

RabbitMQ

HTML CSS JS

Web Program

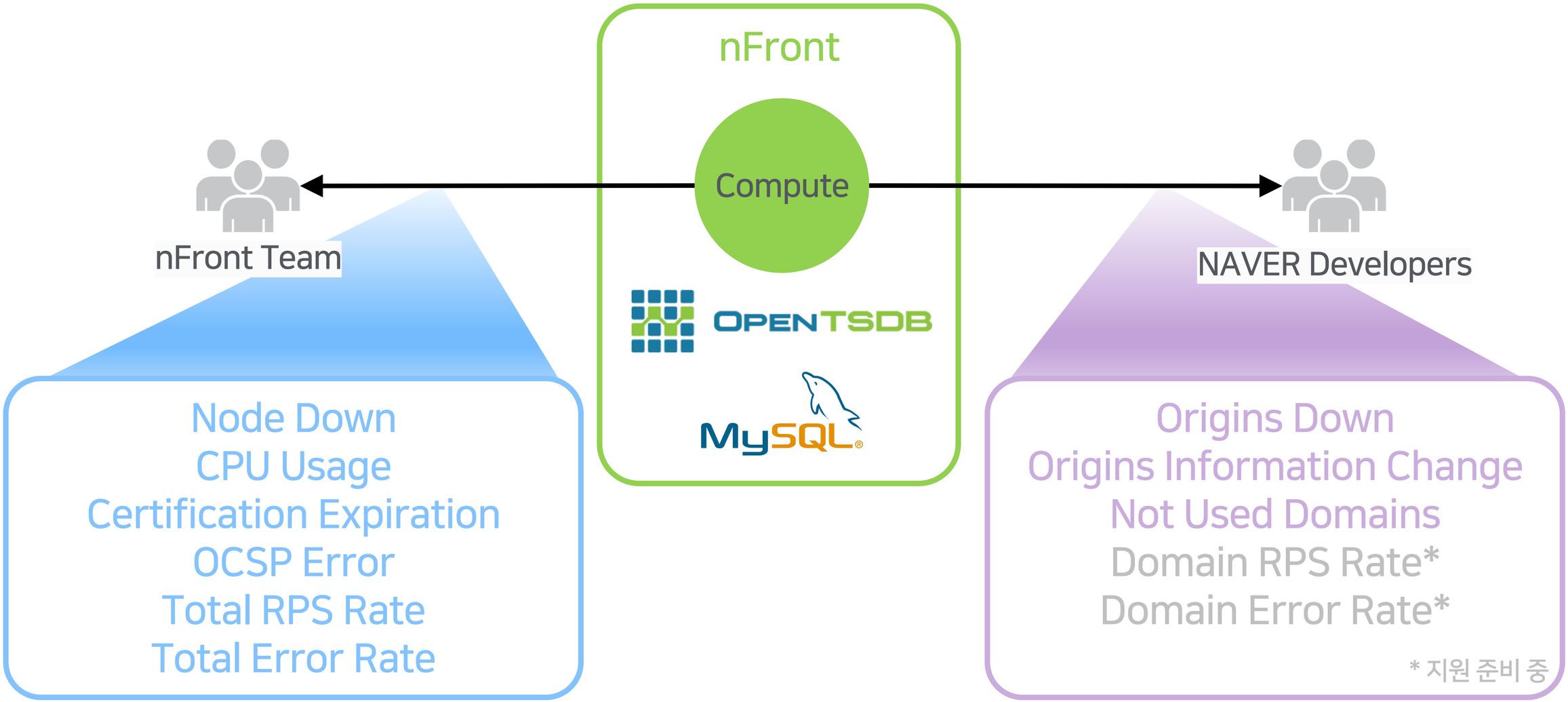
Cert	B	C
A	000	
X		111

Object | Agent | Server | User Interface | User

4.3 Alerts

Monitoring 상에서 수집된 데이터 기반으로,

Mail, Messages Bots 활용하여, 실시간 Alerts 제공



4.4 Log Management

Indexing 및 IO Performance 최적화를 위해,

일부 Data를 Structuralize하는 중간 과정 도입

ex) Host, Timestamp, Custom Column, …

nFront

HAPROXY

HAPROXY

HAPROXY

⋮

Unstructured Data
(Plain Syslog)

over 10 TB / day

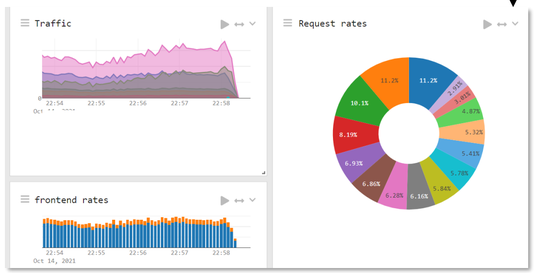
rsyslog

Semi Structured Data
(JSON Format)

elasticsearch

Archive Data
(Tar.gz File)

GlusterFS



단순 지표로 파악 어려운

실시간 Data 분석에 활용

ex) Abuser Pattern, Abnormal Request

5. User Friendly

5.1 Self Service for User

On-Demand

사용자가 원하는 시점에

Faster

95% 감소된 소요 시간으로 33분 -> 1분

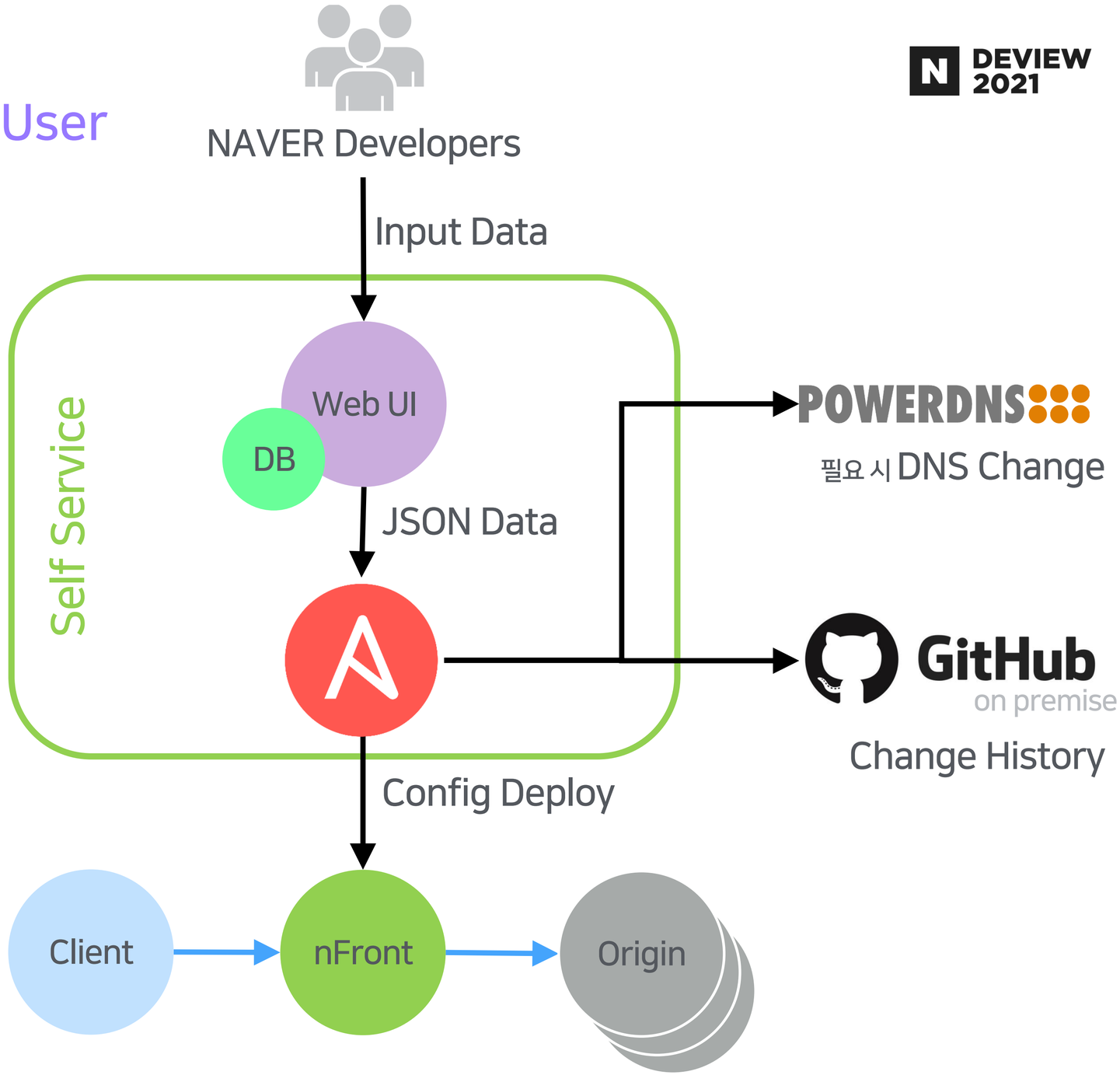
Error Minimization

최소한의 Human Error로

Humanless

운영 Resource는

미래지향적인 프로젝트로



5.2 Auto Scale for Platform

JudgeD

실시간 Resource 현황과 정의된 Policy 기준으로

Scale In/Out 판단

Resource Mgr.

Busy/Idle Node를 Listing하며,

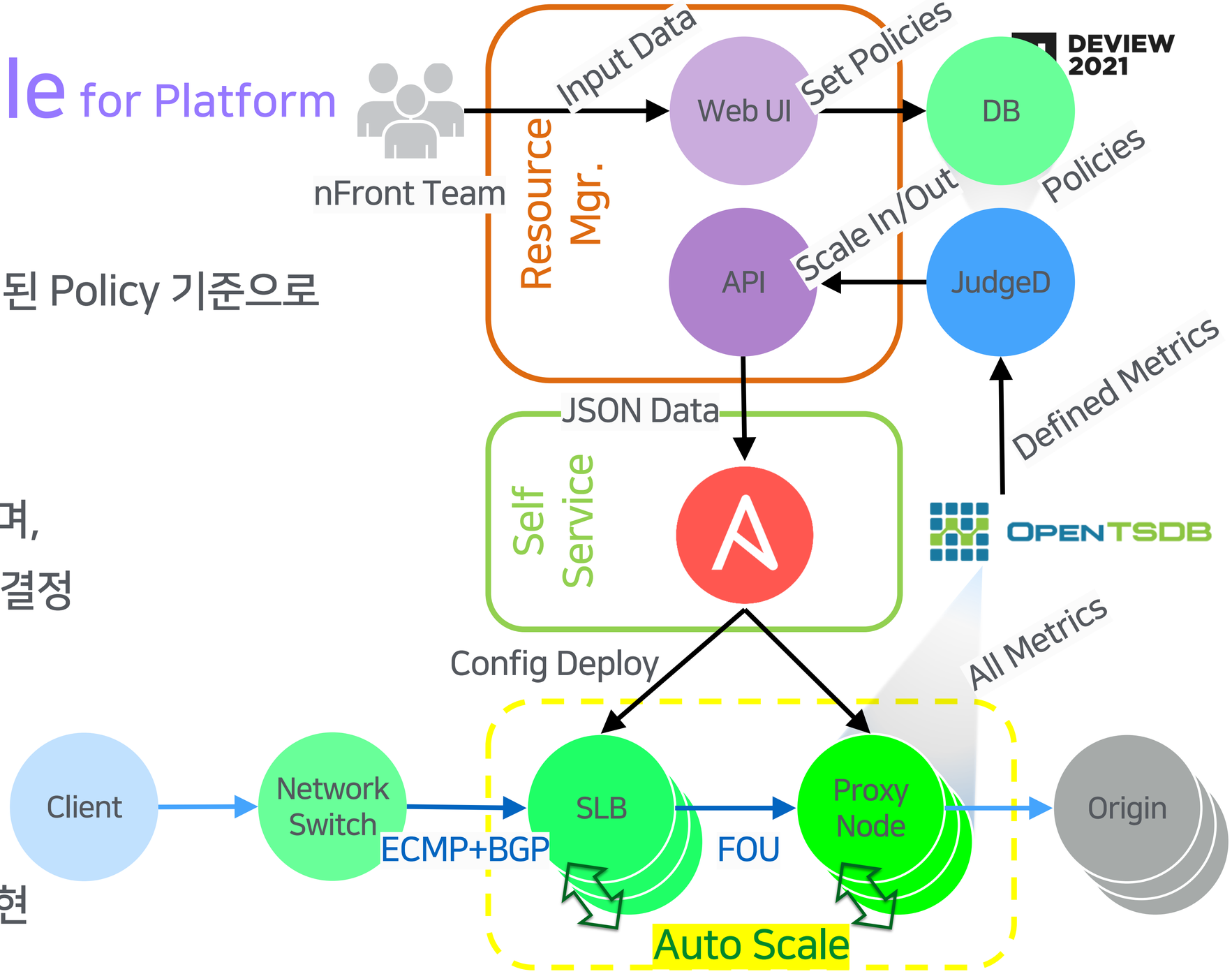
어떤 Node를 할당/회수 할 지 결정

Ansible

실제 Config Deploy 수행

SLB

ECMP + BGP + IPVS 기반 구현

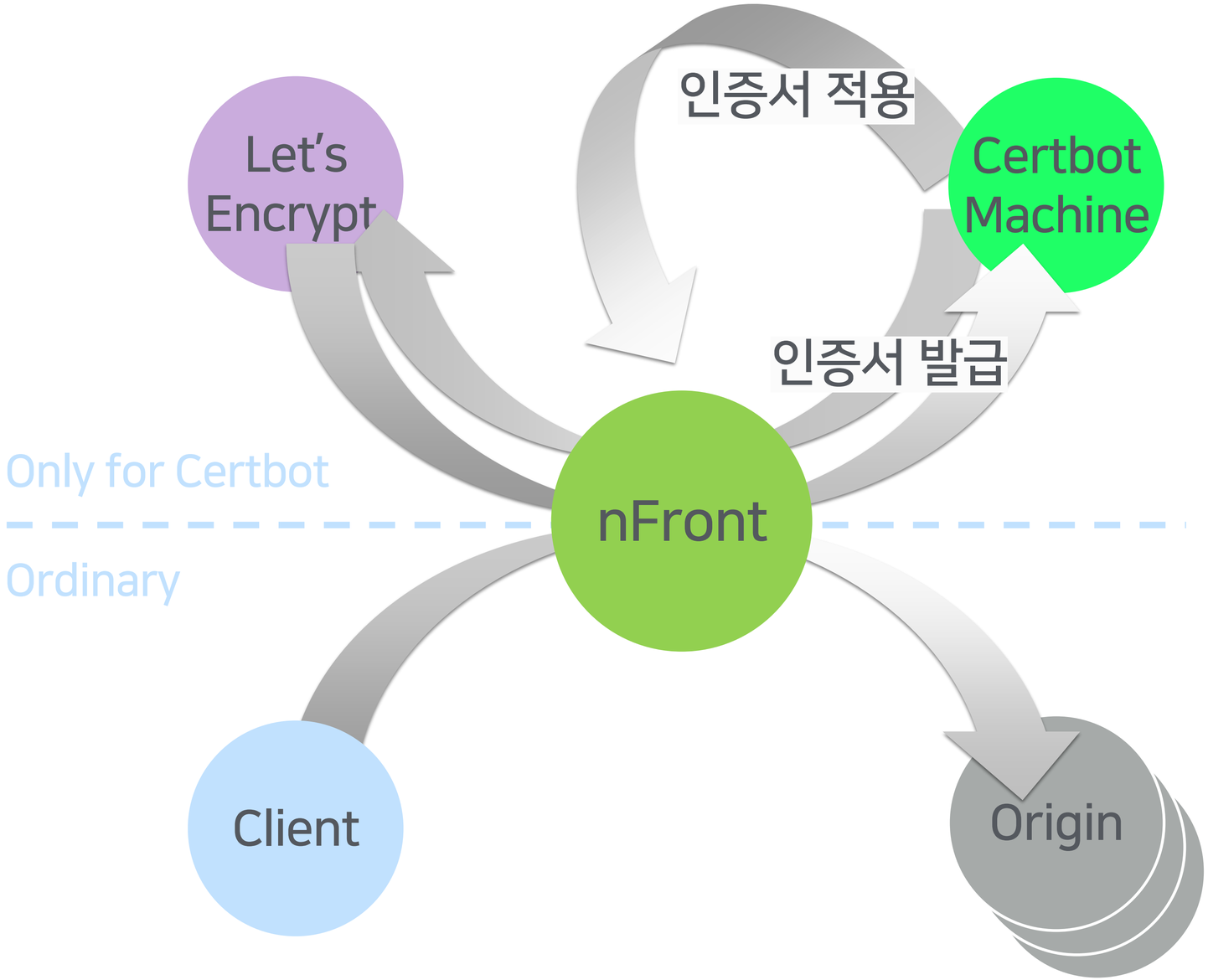


5.3 Let’s Encrypt for Certificates

임시/개발/테스트 Domain 대상으로,
Let’s Encrypt 서비스 활용하여,
즉각적인 https 통신 지원

일반 인증서 발급 과정 대비 시간 절감
1~3일 소요 → 3분 내외 소요

Let’s Encrypt Request는
Pattern 탐지하여,
Certbot Machine으로 전달



6. Troubleshoot

6.1 Apple ID Login Failed

DEVIEW 2021

현상

Sign in Token 검증 과정에서

TLS Handshake Failure 발생

원인

apple.com Client에서

ECDSA Cipersuite 미지원

해결

RSA Cert. 추가하여 긴급 지원

apple.com 측에 관련 내용 전달

현재는 ECDSA Cert. 지원 확인

https://favpng.com/png_view/computer-mouse-computer-mouse-pointer-cursor-window-png/rYt6YDLF

6.2 Intermediate CA Untrusted

인증서 갱신 주기를 짧게 하면, 예방 가능

6.3 HTTP/2 Connection Coalescing

Latency 향상 목적으로, 되도록 기존 Connection을 Reuse

RFC7540(HTTP/2) Connection Reuse

Case)

1. Use *.example.com Wildcard Cert
2. Still keep a connection to a.example.com and b.example.com
3. Newly make a connection to c.example.com

a.example.com: IP-x

b.example.com: IP-x

c.example.com: IP-y

Firefox

"c.example.com may be at IP-y."

Other Browsers

"I have to lookup c.example.com."

or

"I can not do anything on HTTP/2."

https://daniel.haxx.se/blog/2016/08/18/http2-connection-coalescing/

https://httpwg.org/specs/rfc7540.html#reuse

이 조건일 시,

421 MISDIRECTED REQUEST 응답하여, Re Lookup 유도하여 해결

7. Roadmap

7.1 서비스 이상 탐지 기술 강화 Machine Learning 기반

DEVIEW 2021

AS IS

Log를 통해 Pattern 도출 과정 필요

담당자 직접 제어

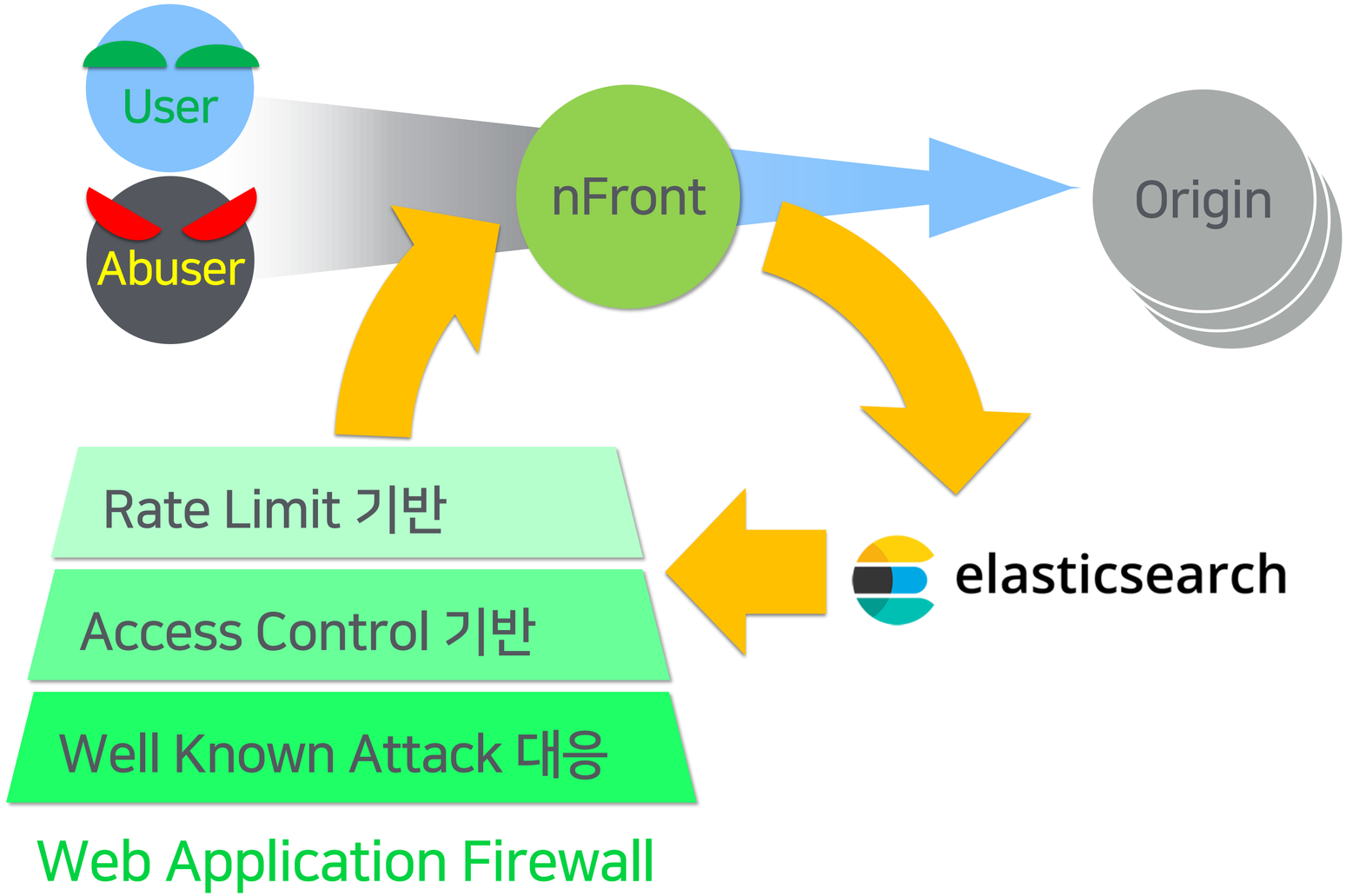
담당자 상황에 따라 방어 지연

TO BE

기계 학습된 Rule에 따라,

위험 상황 인지하여 자동 방어

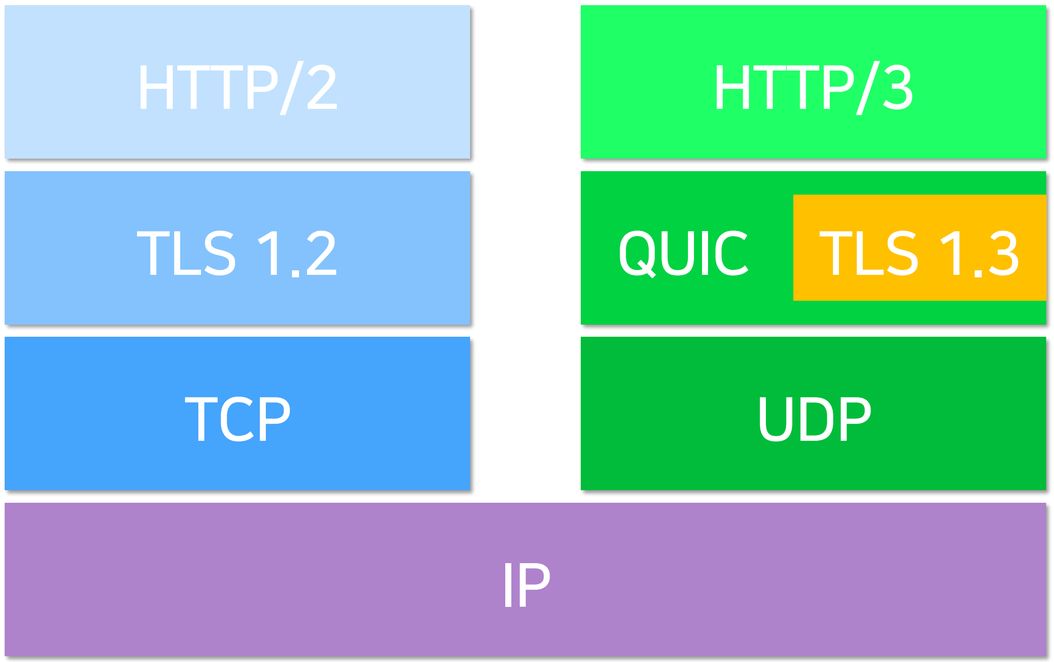
담당자 없이도 신속 방어

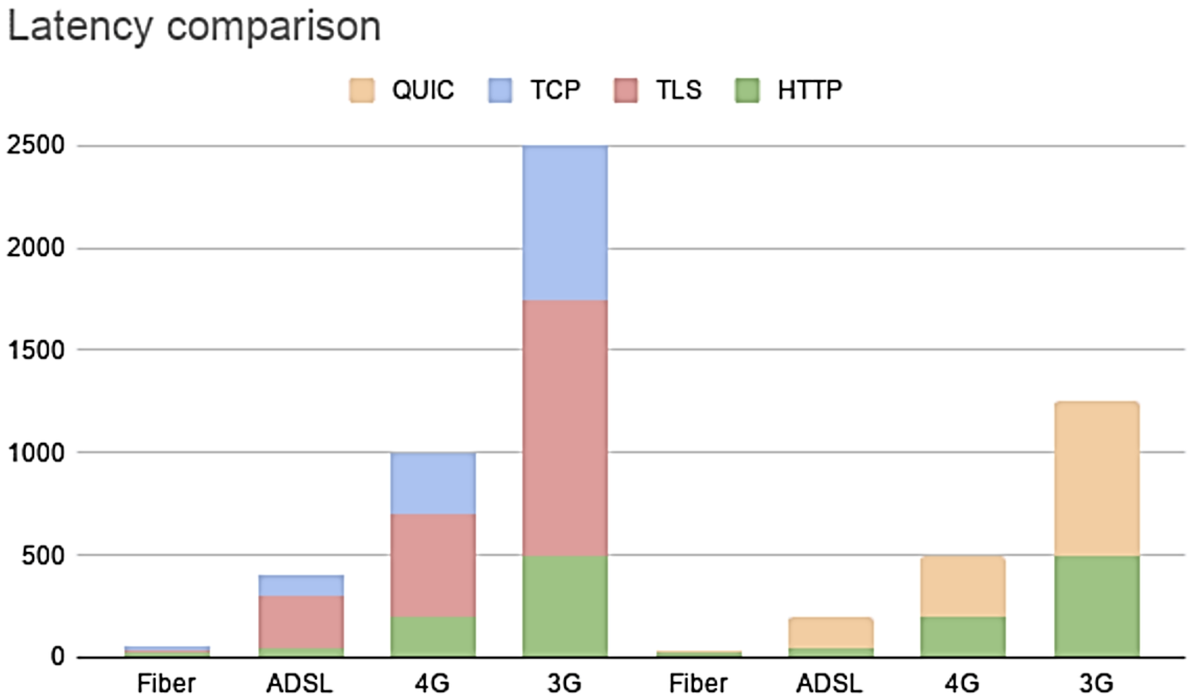


7.2 HTTP/3, QUIC 도입

HTTP/1.1 and HTTP/2 are over TCP

HTTP/3 is over QUIC(based on UDP, Quick UDP Internet Connections)



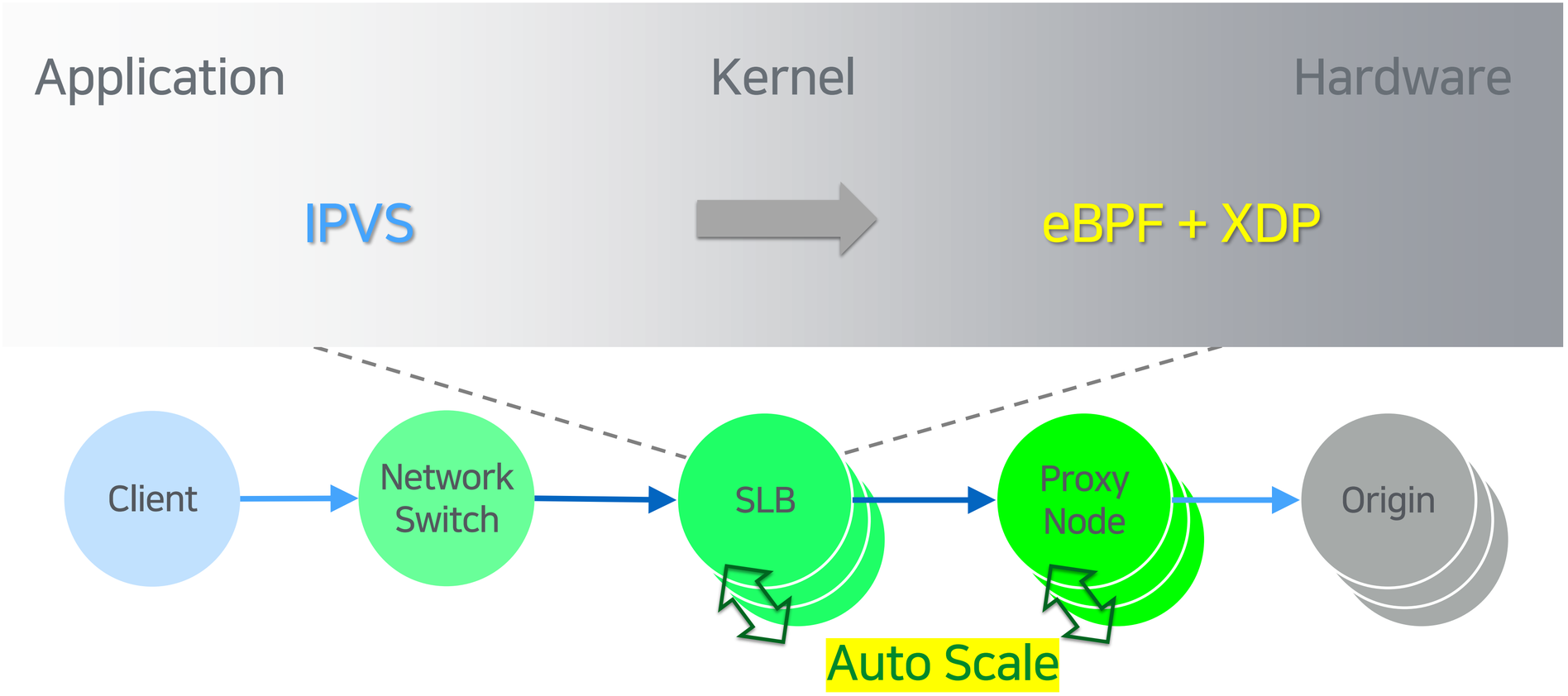


https://www.section.io/engineering-education/http3-vs-http2/

7.3 eBPF/XDP Based L4

eBPF (Extended Berkeley Packet Filter) and XDP (eXpress Data Path)

The Lower Level, The Faster Processing



7.4 Cloud 상품 개발

nFront

Self Service

NAVER Cloud Platform

Auto Scale

DNS Control

https://favpng.com/png_view/computer-mouse-computer-mouse-pointer-cursor-window-png/rYt6YDLF

Thank You