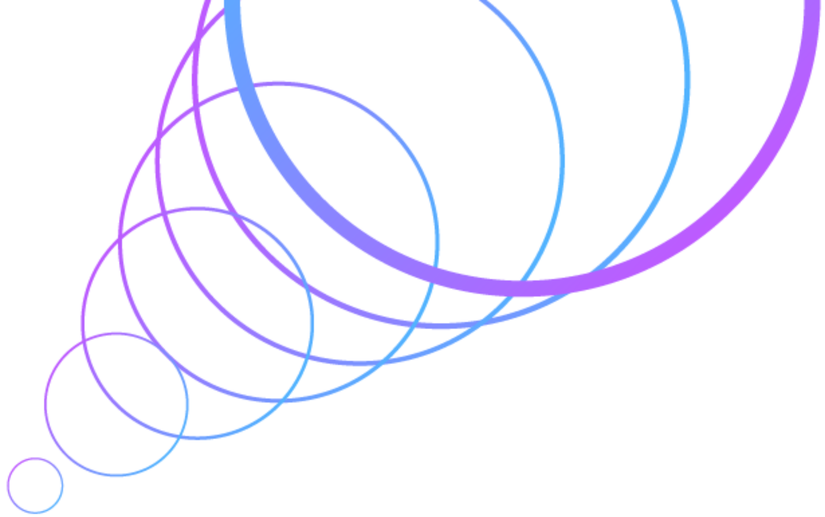


매니지드 쿠버네티스 딜리버리를 위한 기술

최규범 NAVER Cloud

CONTENTS

1. 네이버 클라우드 플랫폼 Kubernetes Service

1.1 네이버 클라우드 플랫폼 k8s 소개

DEVIEW 2021

- Cloud Controller Manager
- Managed Control Plane
- Container Registry
- Container Storage Interface
- NAVER CLOUD PLATFORM
- kubernetes
- Monitoring
- Cluster Autoscaler
- Metrics Server
- Node Pool
- Audit Log

1.2 CSP 관점에서의 k8s

DEVIEW 2021

Vendor Lock-in 약화

- Vendor Lock-in ?
특정 벤더의 독자적인 제품, 서비스, 시스템에 의존하고 있어, 다른 벤더가 제공하는 동종 기술로 이전하기 어렵게 되는 현상

CSP에서 제공하는 기술에서 Kubernetes로..

- 각각 CSP사에서 제공하는 API에 종속적인 코드 작성 필요 없이 k8s API만을 사용하여 인프라 제어

1.3 Managed Service로서의 쿠버네티스

Self-Managed Kubernetes

- 💸 비용 지불:
  - Provider-Managed k8s의 경우 Control Plane에 대한 시간당 관리 비용을 청구.
- 🚚 k8s version:
  - 최신 버전의 k8s 도입이 용의함. CSP사에서 제공하는 k8s의 경우, 실제 k8s 릴리즈보다 버전 업데이트가 대체적으로 늦음.
- 🛠️ Less control:
  - Provider-Managed k8s 사용시, CSP사에서 지원하는 인프라 옵션 및 통합으로 환경이 제한됨.

1.3 Managed Service로서의 쿠버네티스

Provider-Managed Kubernetes

- Control Plane에 대한 관리
  - k8s 운영에 대한 학습 곡선 감소 기대. CSP에서 보장하는 SLA로 운영 안정성 기대.

- 자동화 도구 제공
  - 클러스터 버전 업그레이드 호스트 서버(즉, 노드) 설정, 해당 노드를 클러스터로 구성, ‘
    실패 시 노드 교체와 같은 작업을 처리하는 자동화 도구를 제공.

- CSP사의 인프라와 통합
  - Cloud Controller Manager(CCM), Container Storage Interface(CSI),
    Cluster Autoscaler(CA)

2.Provider-Managed k8s 관리

2.1 Provider-Managed k8s 관리

VM Instance 사용량 모니터링

- Control Plane 리소스 모니터링

Agent 방식

- Control Plane의 VM Instance에서 agent 구동
- kube-system namespace에 필요한 관리 pod 구동

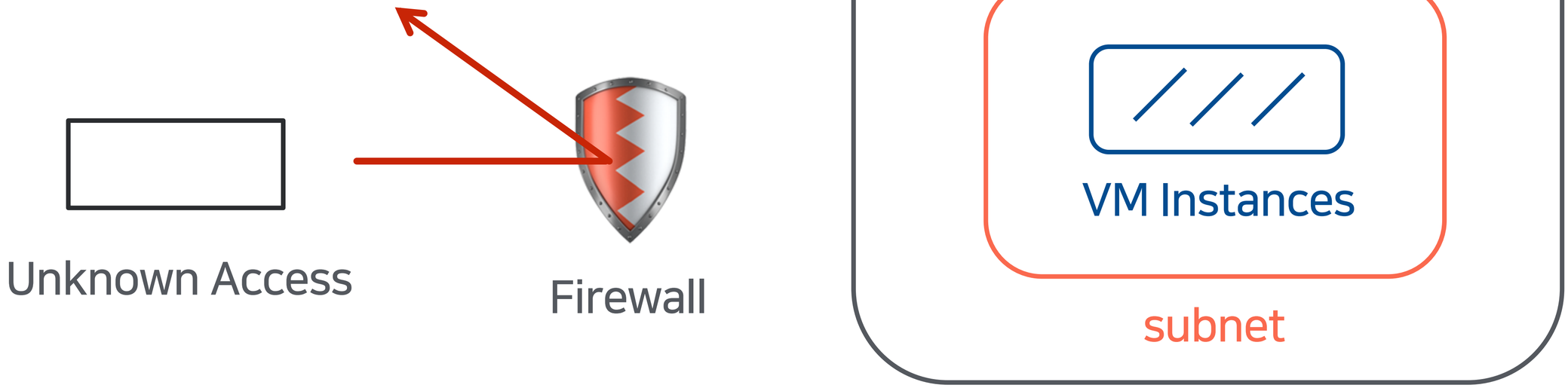
Agent-free 방식

- SSH를 이용한 VM Instance 직접 접근 방식

2.2 서버 접근 제어 관리

Problem: 서비스 관리영역에서의 ACL 고려

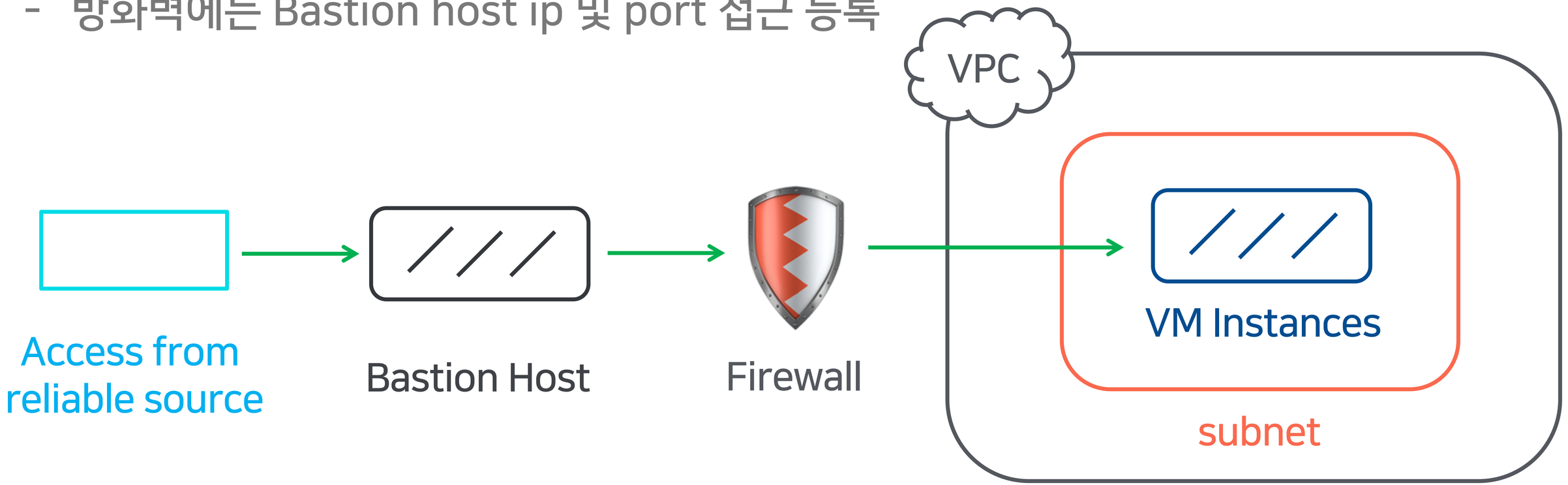
- 사용자의 VPC 내 VM Instance 들은 신뢰할 수 있는 호스트의 ssh접근 만을 허용해야 함.
- 접근 소스에 해당 하는 호스트를 방화벽에 등록
  - 😱 접근 소스가 늘어나거나 줄어들면
    그때마다 방화벽 변경 작업 수행?



2.2 서버 접근 제어 관리

Bastion Host

- 타겟 VM에 ssh로 직접 접근 하기보다, 게이트웨이 역할을 하는 Bastion host를 이용
- 방화벽에는 Bastion host ip 및 port 접근 등록



2.2 서버 접근 제어 관리

Bastion Host를 이용한 타겟 서버 접근


```
$ ssh -i ./target.key root@10.0.123.33 -o StrictHostKeyChecking=no -o \
>  "UserKnownHostsFile /dev/null" -o  \
>  "proxycommand ssh -o StrictHostKeyChecking=no -W %h:%p root@10.0.123.34 -i bastion.key"  \
>   ifconfig | grep 10.0.123.33
Warning: Permanently added '10.0.123.33' (ECDSA) to the list of known hosts.
        inet 10.0.123.33  netmask 255.255.255.0  broadcast 10.0.123.255
```

2.2 서버 접근 제어 관리

Bastion Host 도입 후..

- 방화벽 설계가 단순해 짐
- 상품 관리 서버의 증설 때 ACL 고려 문제에서 자유로움
- 네트워크 접근 이력 관련 작업을 한 곳에서 할 수 있음

아쉬운 점🤣

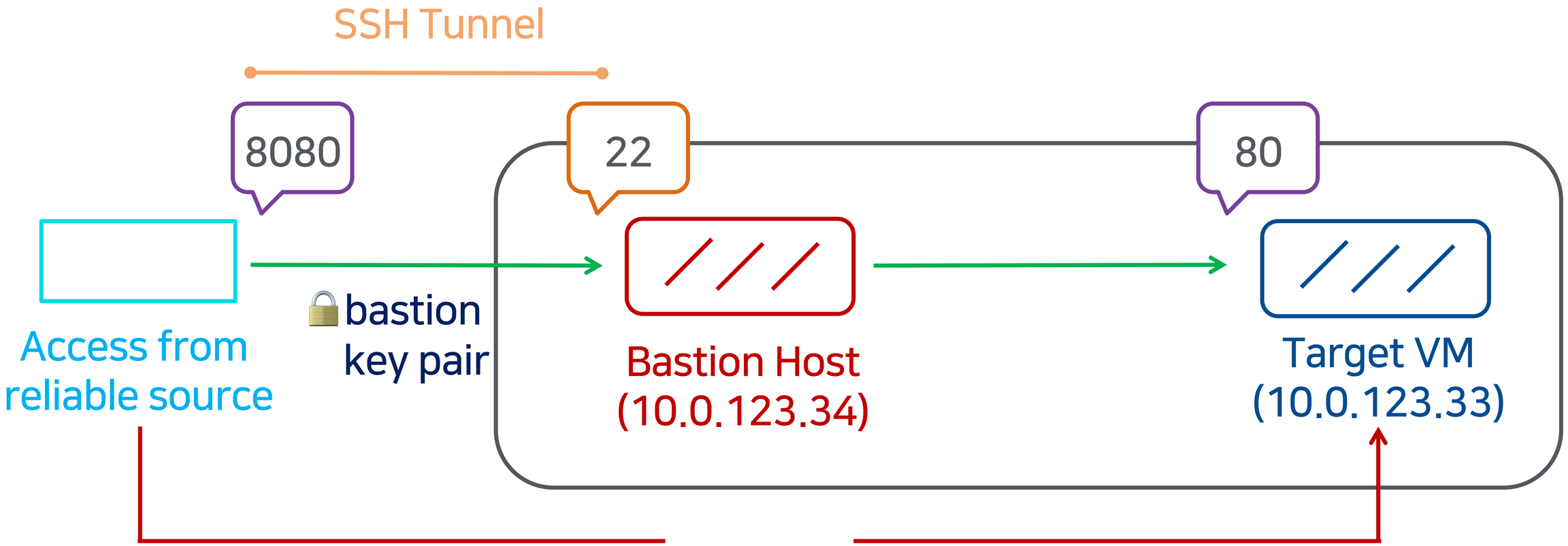
- SSH 프로토콜을 이용한 커맨드만 이용 가능..
- HTTP 요청 등을 보낼 수는 없을까?

2.3 SSH Tunneling을 이용한 요청 포워딩

DEVIEW 2021

SSH Tunneling

- Bastion Host가 프록시와 비슷한 역할을 수행하여, 요청을 포워딩

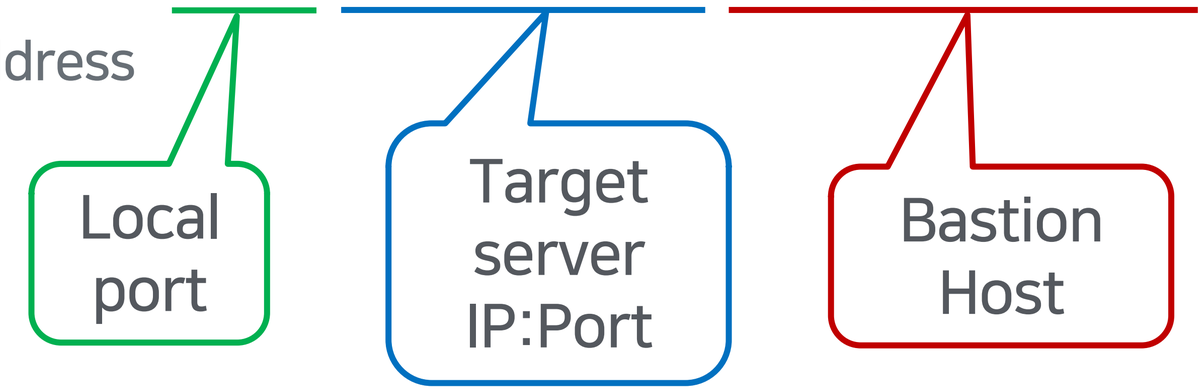


2.3 SSH Tunneling을 이용한 요청 포워딩

DEVIEW 2021

SSH Tunneling 커맨드 예:

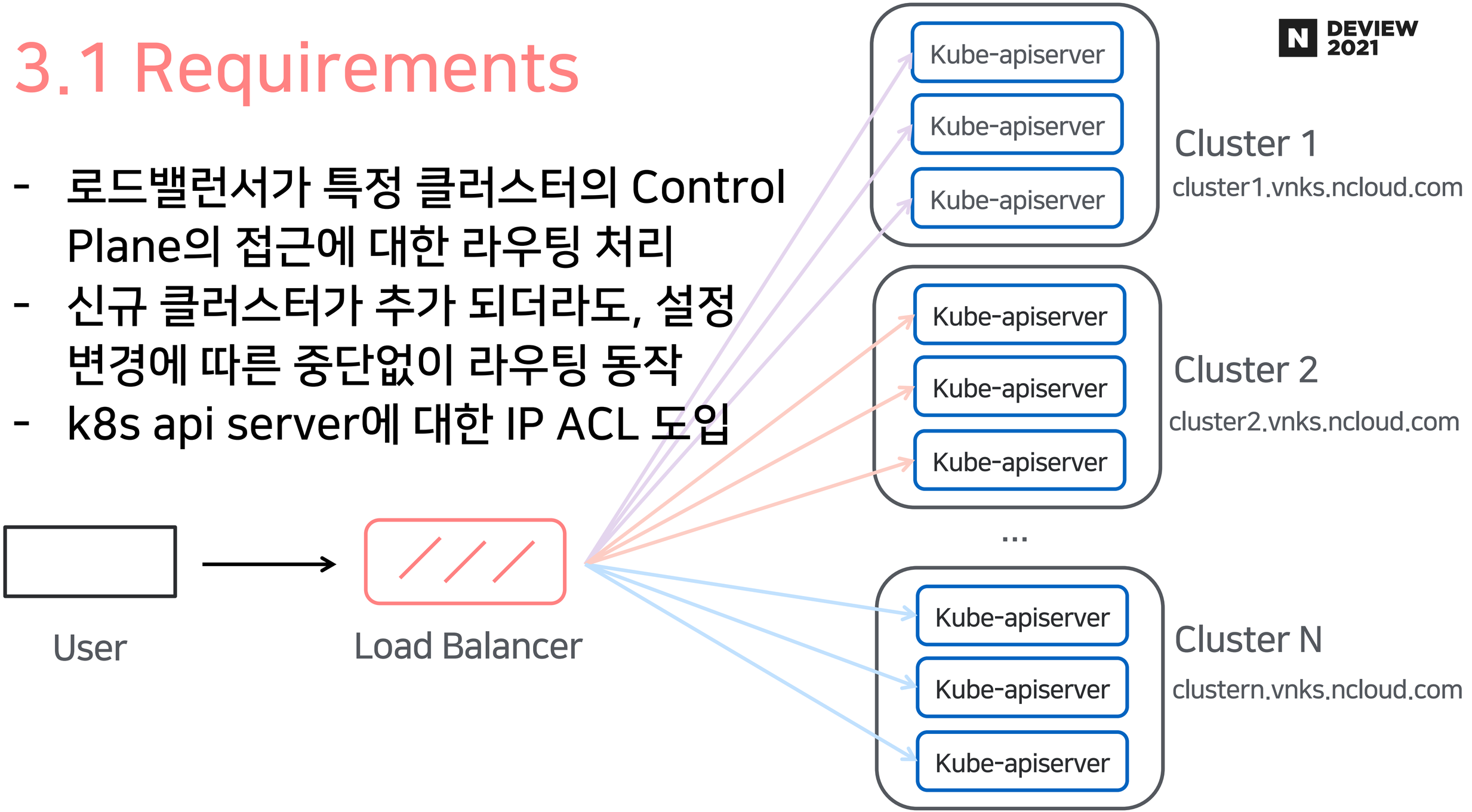
```
$ ## local:8080 → bastion(10.0.123.34):22 → target(10.0.123.33):80
$ curl localhost:8080
    curl: (7) Failed to connect to localhost port 8080: Connection refused
$ ssh  -o ExitOnForwardFailure=yes -f -N -L 8080:10.0.123.33:80  root@10.0.123.34
    bind: Cannot assign requested address
$ curl localhost:8080
    <!DOCTYPE html>
    <html>
    <head>
    <title>Welcome to nginx!</title>
    <body><h1>Welcome to nginx!</h1></body>
    </html>
```



3. OpenResty기반 SSL Passthrough 동적 로드밸런서

3.1 Requirements

- 로드밸런서가 특정 클러스터의 Control Plane의 접근에 대한 라우팅 처리
- 신규 클러스터가 추가 되더라도, 설정 변경에 따른 중단없이 라우팅 동작
- k8s api server에 대한 IP ACL 도입



3.2 OpenResty?

DEVIEW 2021

Nginx

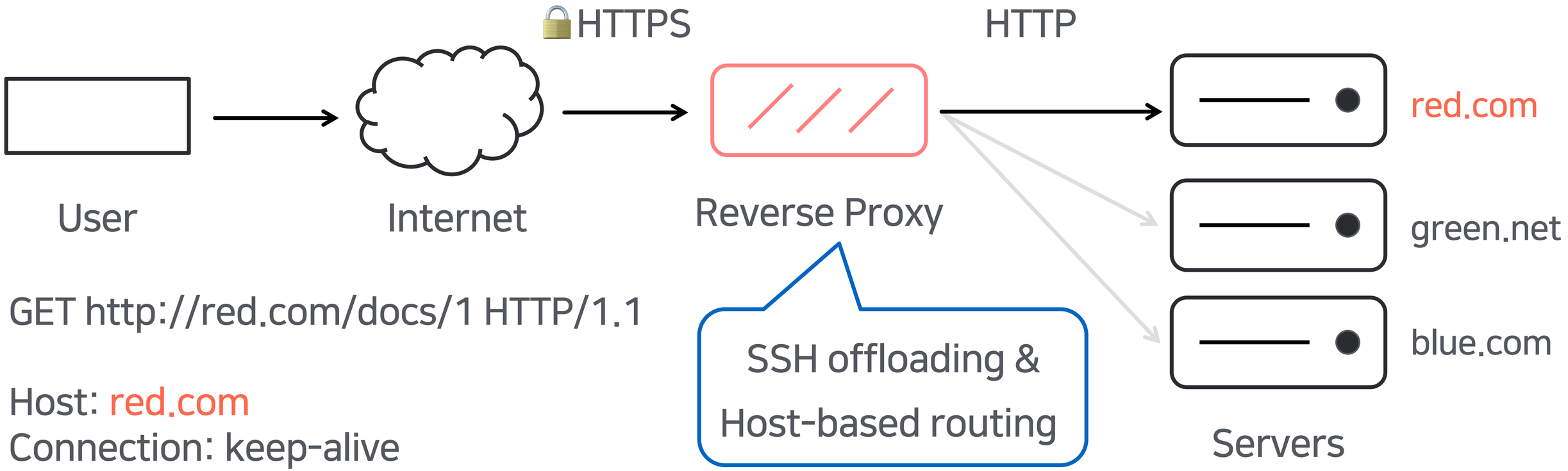
- High Performance Load Balancer
- Directives들로 구성된 Configuration file을 기반으로 동작을 설정

OpenResty: Nginx 기반 Lua 라이브러리를 통합한 확장성 있는 웹 플랫폼

- Nginx 서버를 웹 앱 서버로 효과적으로 전환하여 개발자가 Lua 프로그래밍 언어로 스크립팅 가능.
- 기존의 nginx 모듈과 Lua 모듈을 통합하고, 10K ~ 1000K+ 연결을 처리할 수 있는 초고성능 웹 애플리케이션 구성.

3.3 Reverse Proxy

클라이언트의 요청을 받고, 분석하여 적합한 서버로 전달하는 역할 수행



GET http://red.com/docs/1 HTTP/1.1

Host: red.com
Connection: keep-alive
Accept: text/html,application/xhtml+xml
Referer: http://red.com/

3.4 SSL Termination

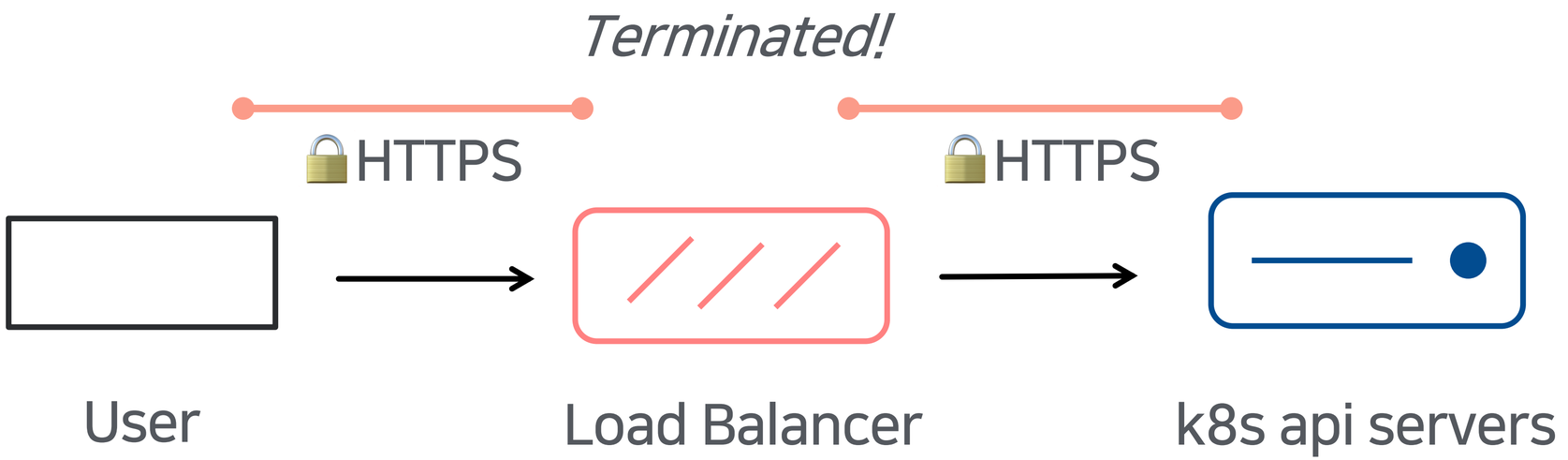
SSL Termination

- 암복호화 작업을 Proxy Server에게 위임하고, 뒷 단 웹 서버는 복호화된 plain HTTP 요청을 전달 받는 형태로, Reverse Proxy 구성 시 해당 형태로 많이 사용
- SSL Offloading 이라고도 함
- 로드밸런서가 tls 인증서를 가지고 클라이언트의 요청에 대한 암복호화를 수행
- ‘X-forwarded-*’ header 와 같이 추가 정보를 덧붙일 수 있음
- http 요청값을 확인하여 host / path 기반 L7 라우팅 가능

3.4 SSL Termination

🤔 Problem: SSL Termination을 k8s api 로드밸런싱에 적용한다면?

- Kube apiserver는 TLS 서버로 구성되어야하므로, 암복호화가 중복 수행됨.
- 로드밸런서에 신규 노드가 추가될 때마다 tls 인증서 등록이 필요.



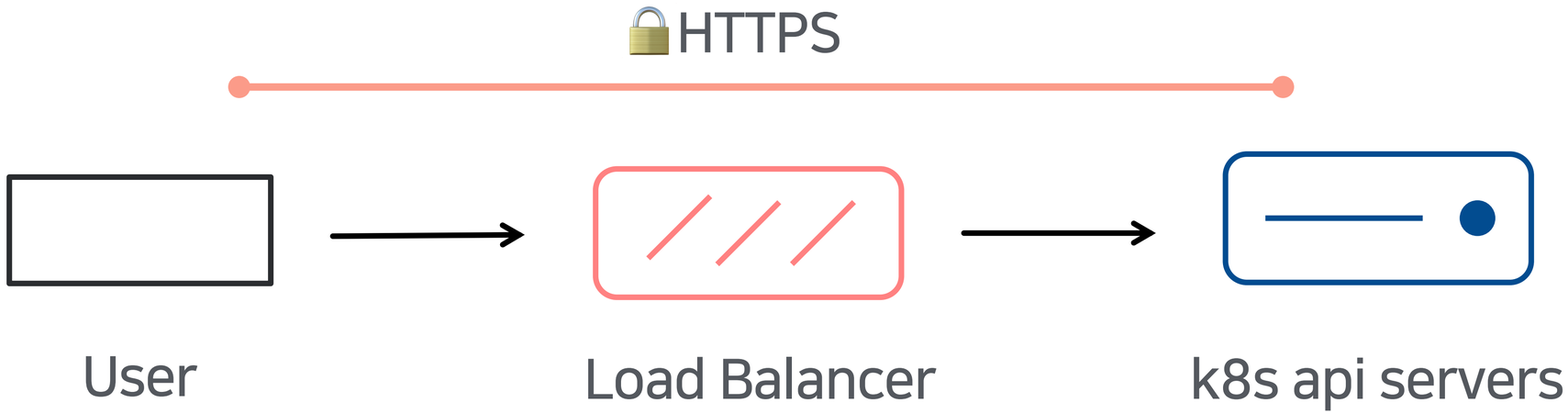
3.5 SSL Passthrough

SSL Passthrough

- tls 연결을 terminated하지 않고, backend 서버까지 그대로 통과시킴.
- backend 서버에서 tls 연결을 맺기때문에, tls 서버 구동을 위한 인증서 설정 작업이 필요.

😇 Advantage

- 요청을 Kube apiserver로 그대로 보내면 되므로, 구조가 단순해 짐.
- 로드밸런서에 신규 노드가 추가될 때 마다 tls 인증서 등록이 불필요



3.5 SSL Passthrough

😲 SSL Passthrough에서의 호스트 기반 라우팅이 가능한가?

- SSL Passthrough 방식에서는 host 기반 라우팅 가능 (path 기반 설정은 불가)
- SSL Termination에서는 HTTPS 요청을 복호화하여 확인하므로, HTTP 헤더의 host값 확인 가능
- SSL Passthrough 방식에서는 Payload 확인 불가..

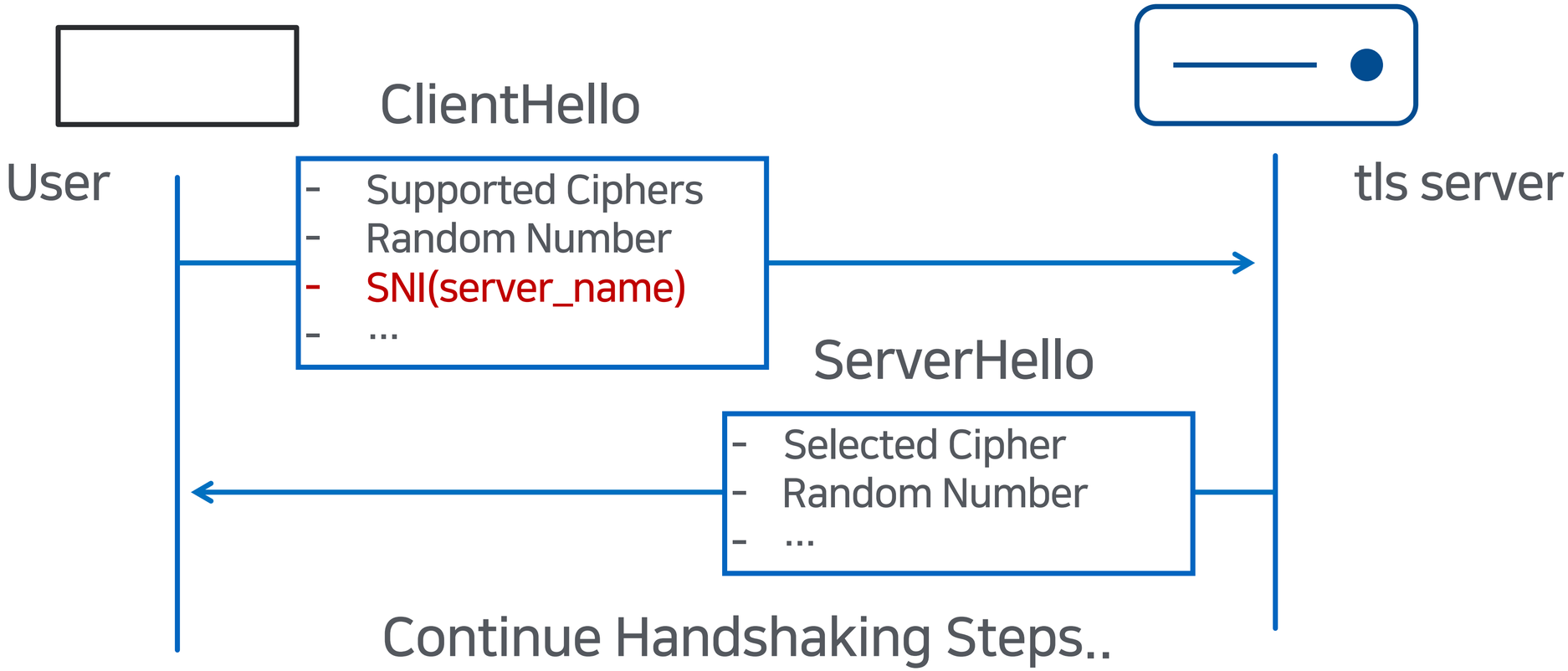
🤔 Problem

- HTTPS 요청을 Backend Server로 그대로 전달하는 Passthrough에선 어떻게 host값 획득?
  - Server Name Indication(SNI)를 이용하자!

3.6 Server Name Indication(SNI)

SNI의 역할

- tls extension에서 제공하는 기능으로, handshaking 초기 과정에서 클라이언트가 어느 호스트명에 접속 하려는지 서버에게 알리는 기능 수행

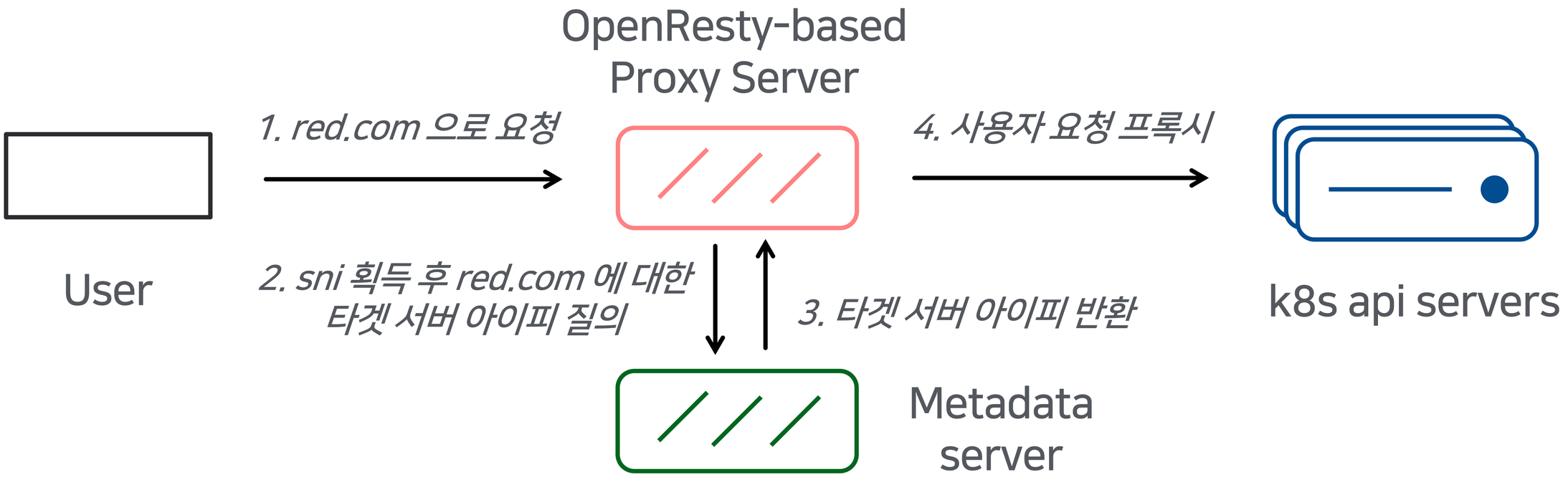


3.7 OpenResty기반 동적 업스트림 서버 재구성

DEVIEW 2021

시스템 구성 및 시나리오

1. 인입되는 요청의 TLS extension을 이용하여 서버 네임을 읽습니다.
2. 호스트 네임을 메타데이터 서버에 질의하여, 타겟이 되는 웹서버의 주소를 획득합니다.
3. 획득한 주소로 트래픽을 라우팅합니다.



3.8 OpenResty기반 동적 업스트림 서버 재구성

DEVIEW 2021

획득한 SNI로 타겟 서버 IP조회하는 Lua script 작성

- *ssl_preread on* 활성화
- lua block에서 server name 획득
- 획득한 server name으로 타겟 호스트 주소를 메타 데이터 서버에 질의
- 이후 과정은 backend Stream block에서 수행

```
server {
    listen 443;
    ssl_preread on;

    preread_by_lua_block {
      if ngx.var.ssl_preread_server_name == nil then
        return ngx.exit(ngx.ERROR)
      end

      ngx.ctx.servername = ngx.var.ssl_preread_server_name

      local http = require "resty.http"
      local httpc = http.new()

      local res, err = httpc:request_uri("http://192.168.0.4/targetServer", {
          method = "POST",
          headers= {["Content-Type"]="application/x-www-form-urlencoded"},
          body = "servername=" .. ngx.var.ssl_preread_server_name
      })
      ...
      local server_ip_list = res.body
      ...
      local ngx_re = require "ngx.re"
      local server_ip_table = ngx_re.split(server_ip_list, ",")

      ngx.ctx.server_ip_table = server_ip_table
    }

    proxy_pass backend;
}
```

3.8 OpenResty기반 동적 업스트림 서버 재구성

DEVIEW 2021

타겟 서버 IP로 트래픽 라우트하는 Lua Script 작성

😄 획득한 타겟 서버 IP로
set_current_peer 함수를 이용하여
현재 Client 요청을 프록시

set_more_tries에 의해 계속 재시도가 일어남
적절한 시도 횟수 이후에 실패 반환이 필요
😱 설정하지 않으면 CPU, MEM 사용량 급상승 초래

```
upstream backend {
    server 0.0.0.1:7777;
    balancer_by_lua_block {
      local server_ip_table = ngx.ctx.server_ip_table
      local balancer = require "ngx.balancer"
      assert(balancer.set_more_tries(1))
      local tries = (ngx.ctx.tries or 0) + 1
      ngx.ctx.tries = tries
      if tries >= 10 then
          ngx.log(ngx.ERR, "reach maximum trial count")
          return ngx.exit(ngx.ERROR)
      end
      local index = (ngx.ctx.index or 0) + 1
      ngx.ctx.index = index
      if index > #server_ip_table then
          ngx.ctx.index = 1
      end
      local hostip = server_ip_table[ngx.ctx.index]
      local port = "443"
      assert( balancer.set_current_peer(hostip, port) )
    }
}
```

4. Grafana Custom Data Source Plugin 기반 Control Plane 노드 모니터링 및 Alerting 설정

4.1 Motivation

Requirements 및 Grafana가 제공하는 기능

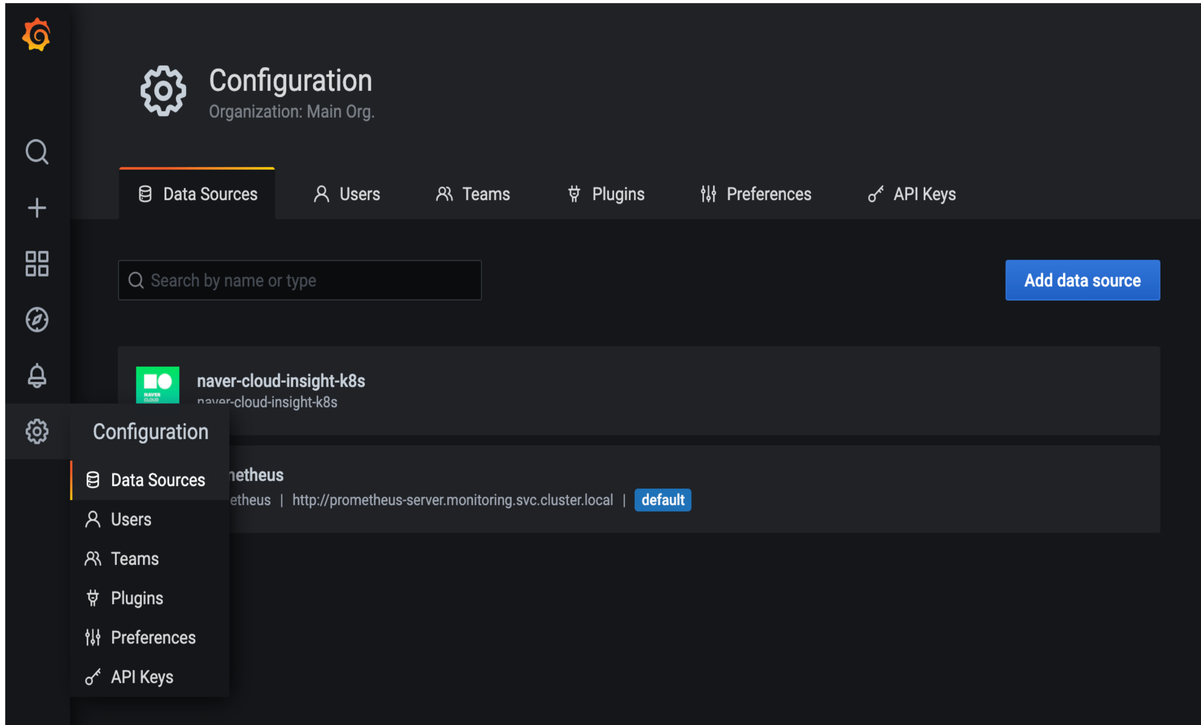
- 시계열 데이터 시각화를 위한 대시보드 제공
- 대시보드의 설정에 대한 import/export
- 운영 관점에서 중요한 Alerting 지원

4.2 Grafana Data Source

DEVIEW 2021

데이터 시각화를 위한 소스 데이터 설정

- Grafana에 대시 보드 생성을 위해 Data Source 선택 및 설정값 입력
- Data Source로 Dashboard 구성
- Query Editor로 데이터 조회 조건 작성
- 시계열 데이터의 시각화에 적합한 Graph 종류 선택
  - Line, Bar, Hitmap 등



4.2 Grafana Data source

DEVIEW 2021

Build Your Own Custom Grafana Data Source😊

- Grafana가 지원하는 Data Source가 아니라면 Custom Data Source를 작성하여 사용 하는 방법 고려

어떤 타입의 데이터 소스를 작성할 수 있을까?

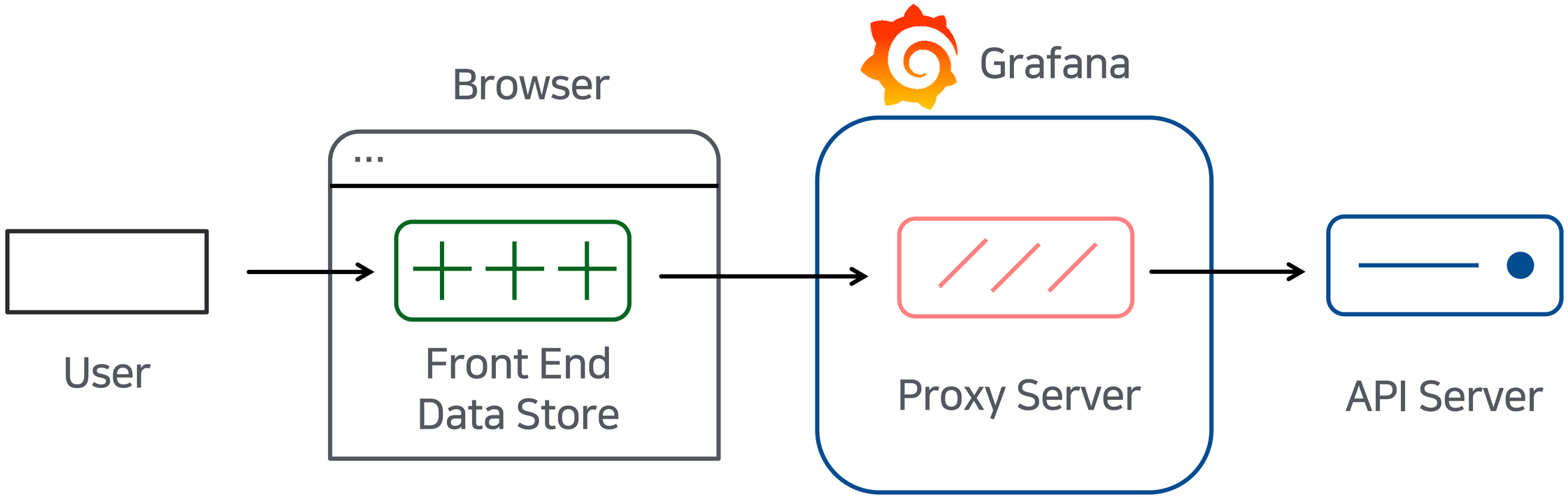
- Data Source Plugin
- Data Source Backend Plugin

4.2 Grafana Data source

DEVIEW 2021

Data Source Plugin: CORS 문제 회피할 수 있는 호출 방식

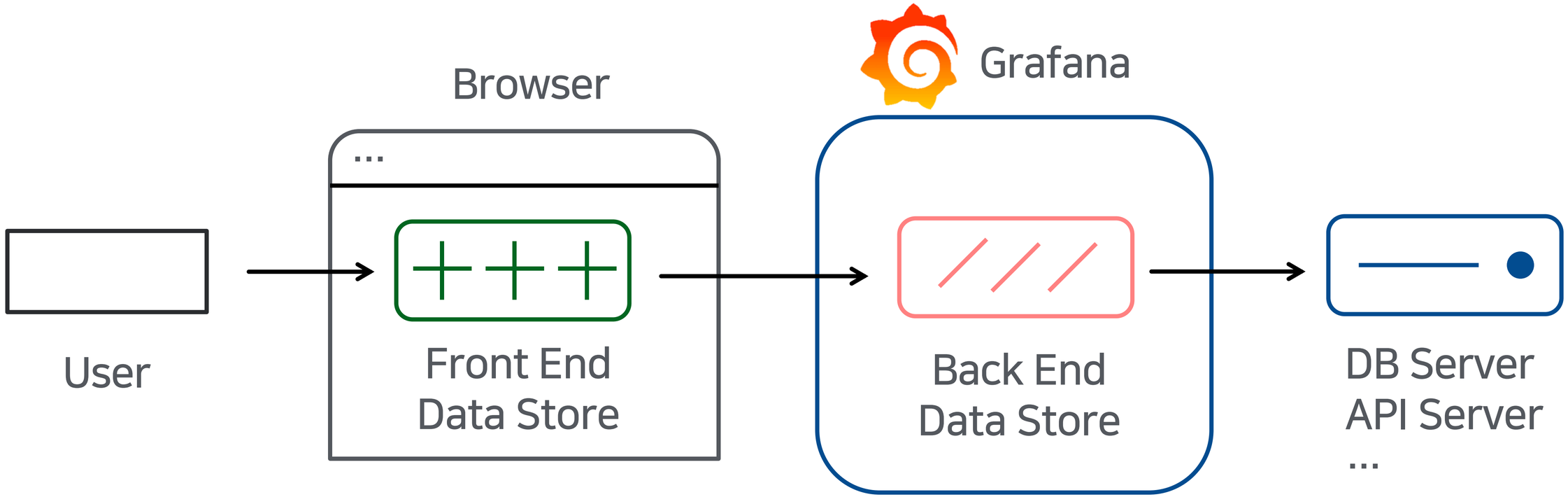
- Browser에서 다른 도메인으로 접근할 때 발생 할 수 있는 CORS 문제를 Grafana 내 프록시 서버 호출로 변경하여 회피



4.2 Grafana Data source

Data Source Backend Plugin: non-HTTP 방식 소스도 이용 가능한 호출 방식

- Front End에 있는 Data Store가 Restful 방식으로 Backend Data Store에 시계열 데이터 요청
- Back End Data Store는 HTTP, non-HTTP 다양한 방식으로 시계열 소스에 접근



4.2 Grafana Data source

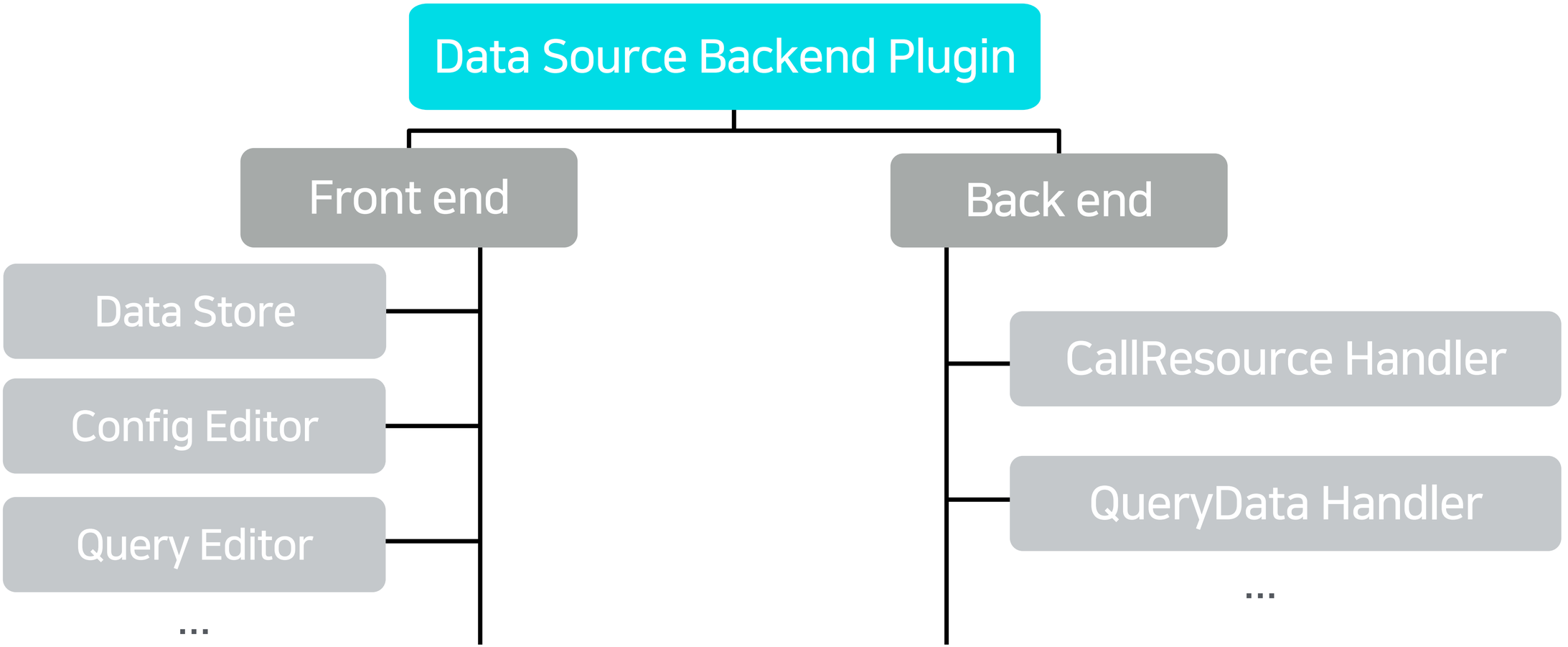
Data Source Plugin vs. Data Source Backend Plugin

	Data Source Plugin	Data Source Backend Plugin
필요 언어	Typescript	Typescript + Golang
CORS 이슈	Proxy Access를 통한 해결	Backend에서 DB를 호출하므로 문제가 없음
non-HTTP 지원	불가	Backend Plugin에서 DB접근 하므로 non-HTTP방식 가능
Alerting 지원	불가	지원

4.3 Custom Plugin 개발

DEVIEW 2021

Data Source Backend Plugin 구성



4.3 Custom Plugin 개발

Front End Side의 Editor 작성

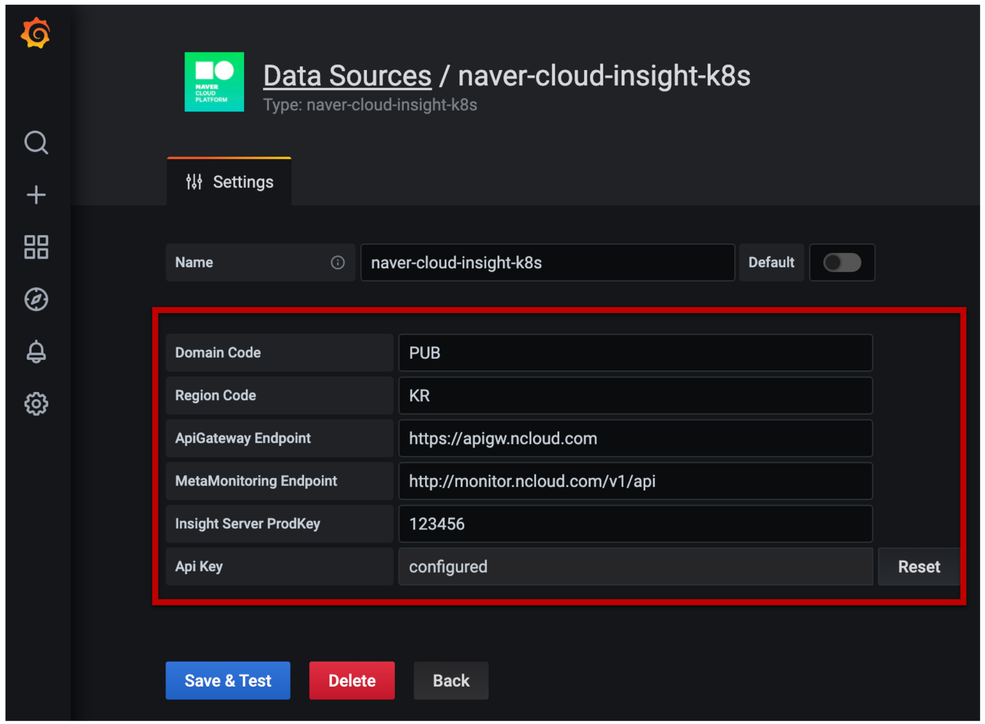
- Data Store 접근 위한 설정 Editor, Data 조회 등 위한 쿼리 작성 Editor 등 작성 필요
- Grafana가 제공하는 React 기반 Component 이용하여 Editor 구현 가능

```
export class ConfigEditor extends PureComponent<Props, State> {
  onApiKeyChange = (event: ChangeEvent<HTMLInputElement>) => {
    const { onOptionsChange, options } = this.props;
    onOptionsChange({
      ...options,
      secureJsonData: {
        ...options.secureJsonData,
        apiKey: event.target.value,
      },
    });
  };
  /** **/
  render() {
    const { options } = this.props;
    const { jsonData, secureJsonFields } = options;
    const secureJsonData = options.secureJsonData || {};

    return (
      <div className="gf-form-group">
        <div className="gf-form">
          <SecretFormField
            isConfigured={secureJsonFields && secureJsonFields.apiKey}
            value={secureJsonData.apiKey || ''}
            label="Api Key"
            onReset={this.onResetApiKey}
            onChange={this.onApiKeyChange}
          />
        </div>
      </div>
    );
  }
}
```

4.3 Custom Plugin 개발

Front End: Config Editor 작성

- Data Store 접근 위한 설정 값들을 위한 입력 폼

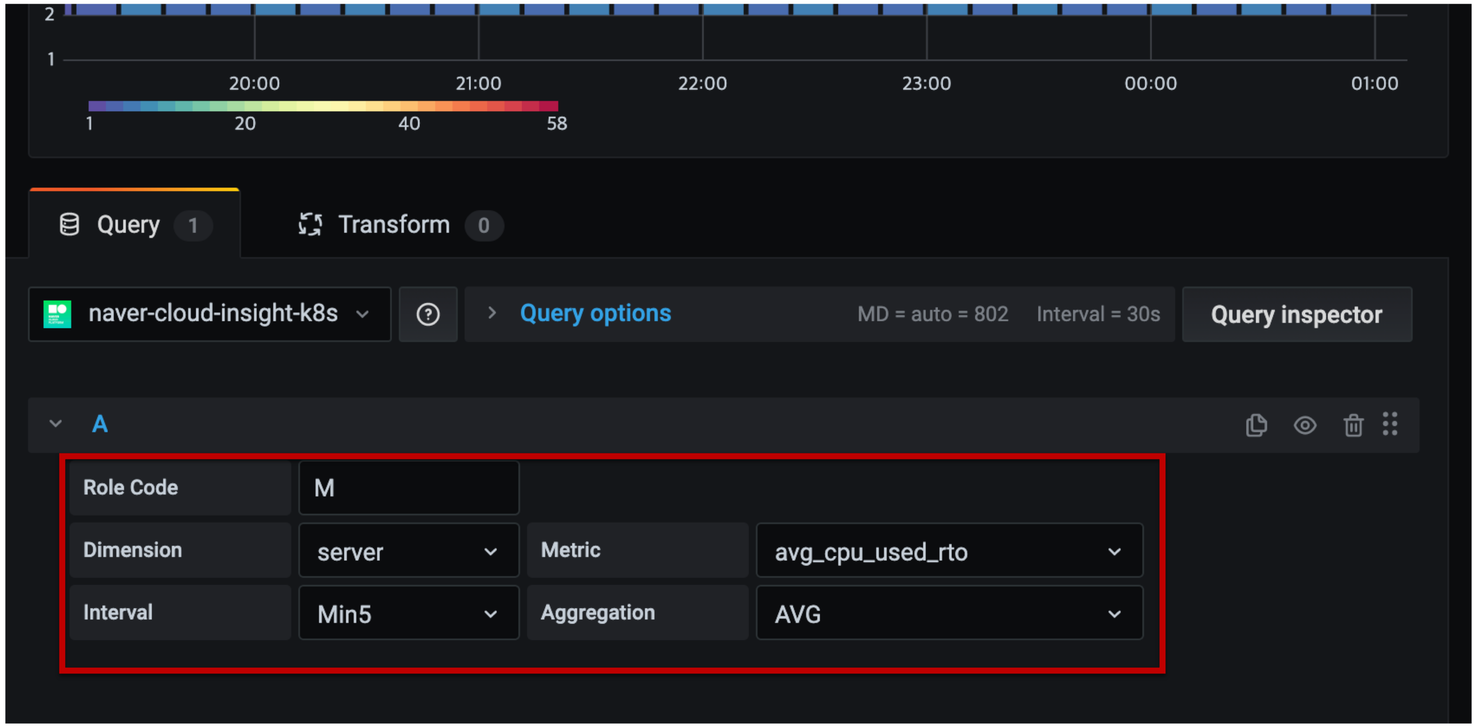



```
export class ConfigEditor extends PureComponent<Props, State> {
  onApiKeyChange = (event: ChangeEvent<HTMLInputElement>) => {
    const { onOptionsChange, options } = this.props;
    onOptionsChange({
      ...options,
      secureJsonData: {
        ...options.secureJsonData,
        apiKey: event.target.value,
      },
    });
  };
  /** **/
  render() {
    const { options } = this.props;
    const { jsonData, secureJsonFields } = options;
    const secureJsonData = options.secureJsonData || {};

    return (
      <div className="gf-form-group">
        <div className="gf-form">
          <SecretFormField
            isConfigured={secureJsonFields && secureJsonFields.apiKey}
            value={secureJsonData.apiKey || ''}
            label="Api Key"
            onReset={this.onResetApiKey}
            onChange={this.onApiKeyChange}
          />
        </div>
      </div>
    );
  }
}
```

4.3 Custom Plugin 개발

Front End: Query Editor 작성

- Data Store에 데이터 조회를 위한 조건을 입력하기 위한 폼
- Config Editor 작성과 유사한 방식으로 Query Editor 작성



4.3 Custom Plugin 개발

DEVIEW 2021

Backend: Data Store 정의

- Frontend에서 선언한 Data Store의 요청에 대한 Handler 선언
- Frontend의 Data Store 값 요청을 처리하기 위한 handler 등록

```go
// main.go
func main() {
  err := datasource.Serve(newDatasource())
  /***/
}
// datasource.go
func newDatasource() datasource.ServeOpts {
  im := datasource.NewInstanceManager(newDataSourceInstance)
  ds := &InsightDatasource{
    im: im,
  }

  return datasource.ServeOpts{
    CallResourceHandler: ds,
    QueryDataHandler:   ds,
    CheckHealthHandler: ds,
  }
}
```

4.3 Custom Plugin 개발

DEVIEW 2021

Backend: Query Data handler 작성

- req.PluginContext
  - jsonData: Config Editor에서 설정한 값
  - secureData: Config Editor에서 secretForm에 작성된 내용

- req.Queries: Query Editor에서 작성된 값

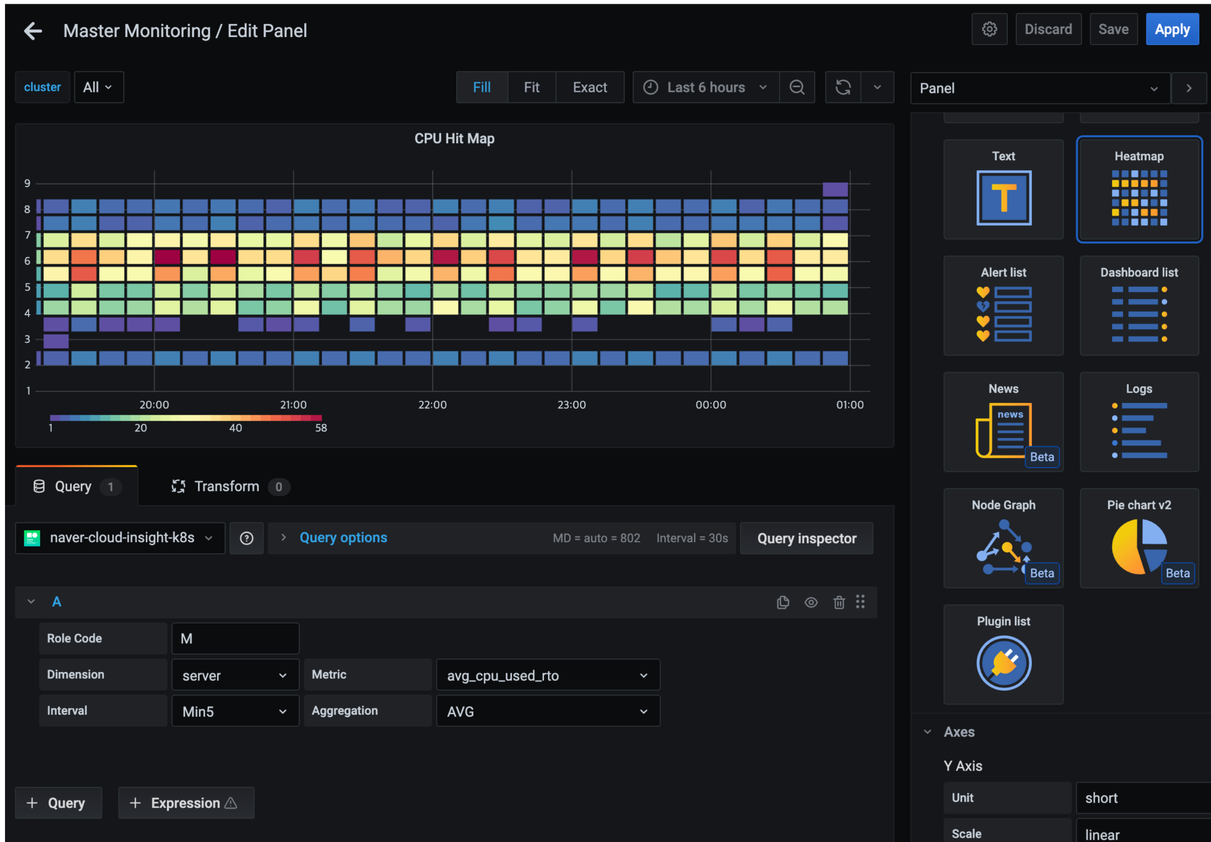
- PluginContext, Queries 값으로 실제 DB에 데이터 조회하여 결과 반환

```go
func (td *InsightDatasource) QueryData(
  ctx context.Context, req *backend.QueryDataRequest) (*backend.QueryDataResponse, error) {
  secureData := getSecureData(&req.PluginContext)
  jsonData, err := getJsonData(&req.PluginContext)
  if err != nil {
    return nil, err
  }

  // create response struct
  response := backend.NewQueryDataResponse()

  // loop over queries and execute them individually.
  for _, q := range req.Queries {
    res := td.query(ctx, secureData, jsonData, q)

    // save the response in a hashmap
    // based on with RefID as identifier
    response.Responses[q.RefID] = res
  }

  return response, nil
}
```

4.3 Custom Plugin 개발

작성한 Data Source로 Grafana Dashboard에서 새 Panel 생성

- 작성한 Query Editor에서 데이터 조회를 위한 값 입력
- Panel 설정에서 시각화를 위해, 데이터와 적합한 형태의 Graph 선택
  - Visualization: Hitmap, Bar, Line 등
  - Axis 단위 설정
  - Legend 설정
  - Tooltip 설정

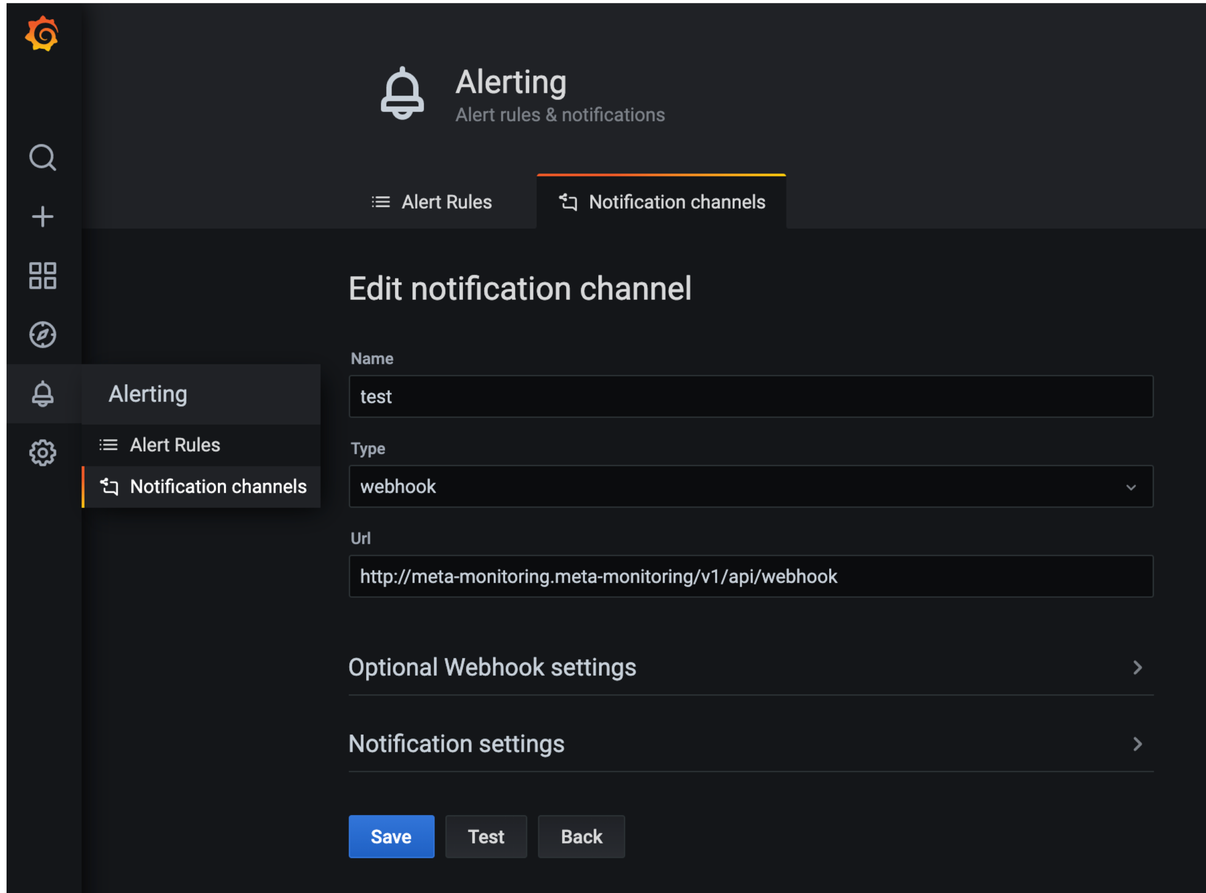


4.4 Custom Plugin 기반 Alerting

DEVIEW 2021

Notification Channel 생성

- Alerting Event 발생 시, Notification을 전달할 Channel 생성
- Channel Type 설정
  - Webhook
  - Telegram
  - Slack 등

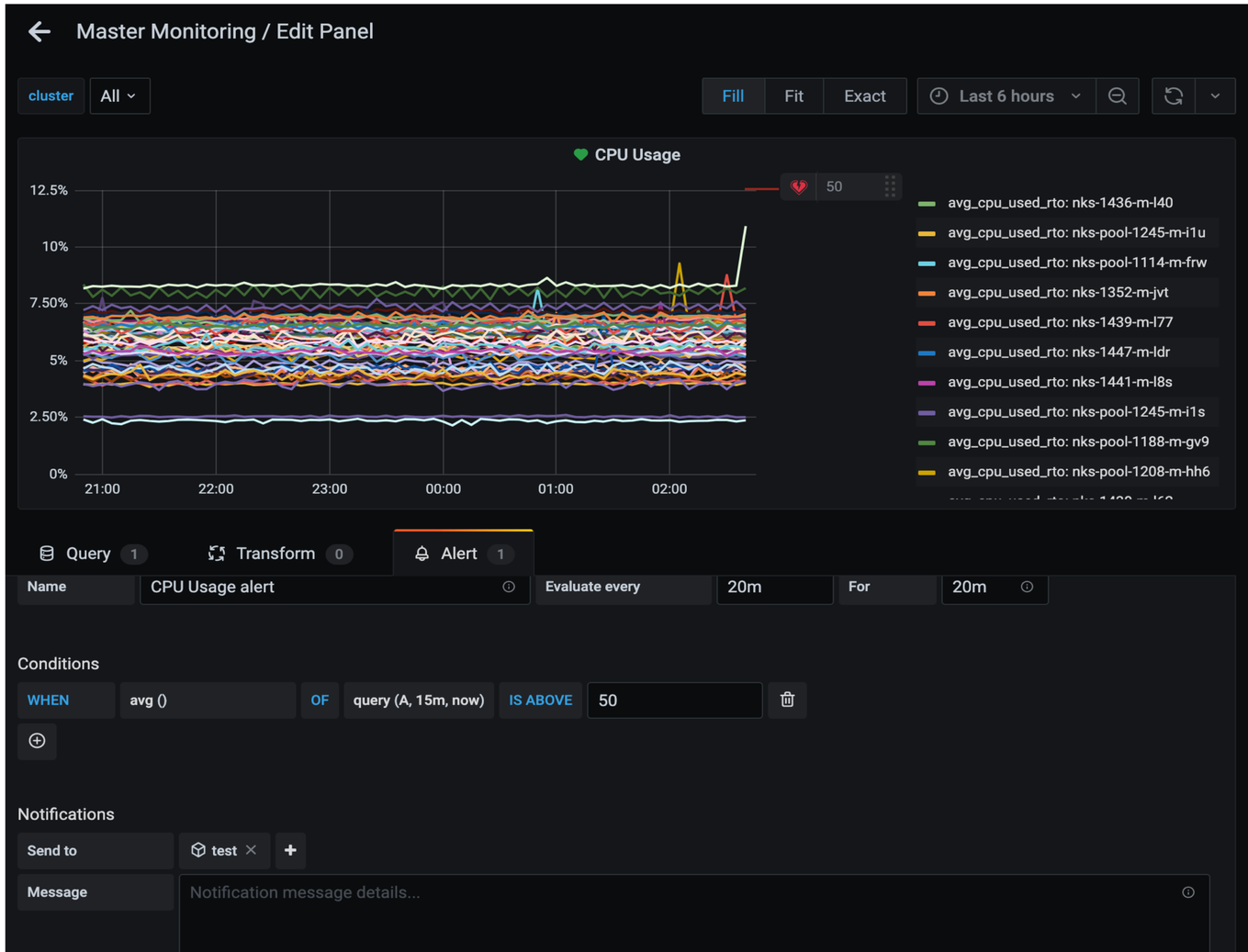


4.4 Custom Plugin 기반 Alerting

DEVIEW 2021

Alert 조건 설정

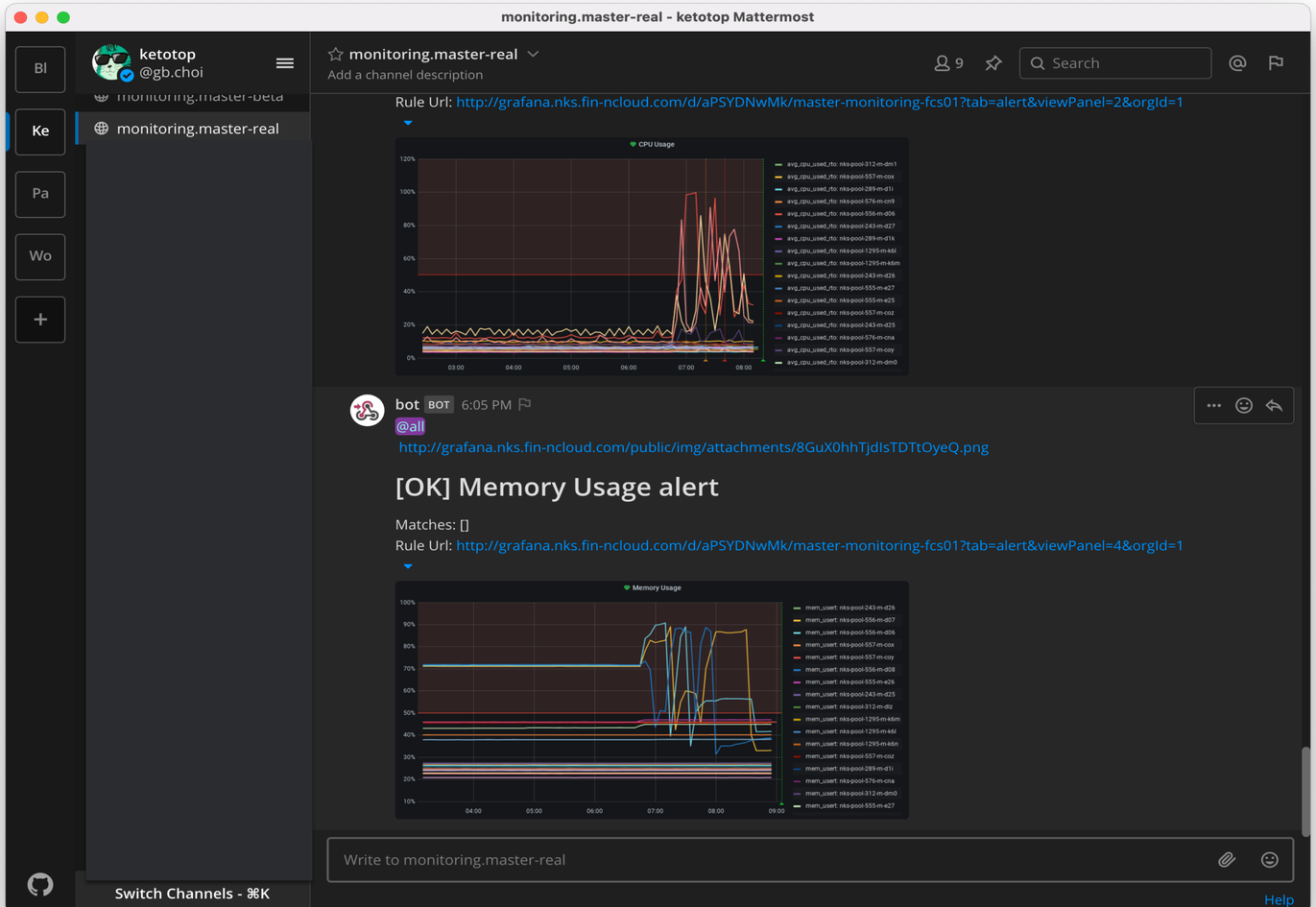
- Notification Channel로 어떤 조건이 몇 분 지속할 때 알람을 전송할 것인가?

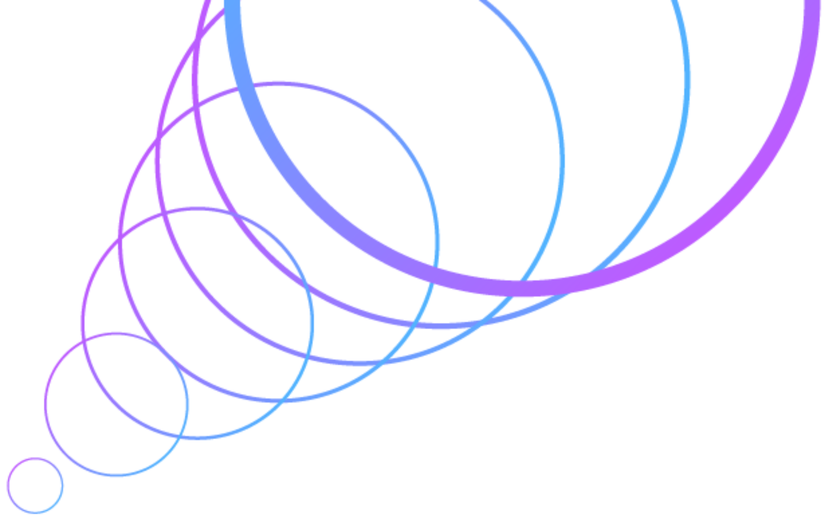


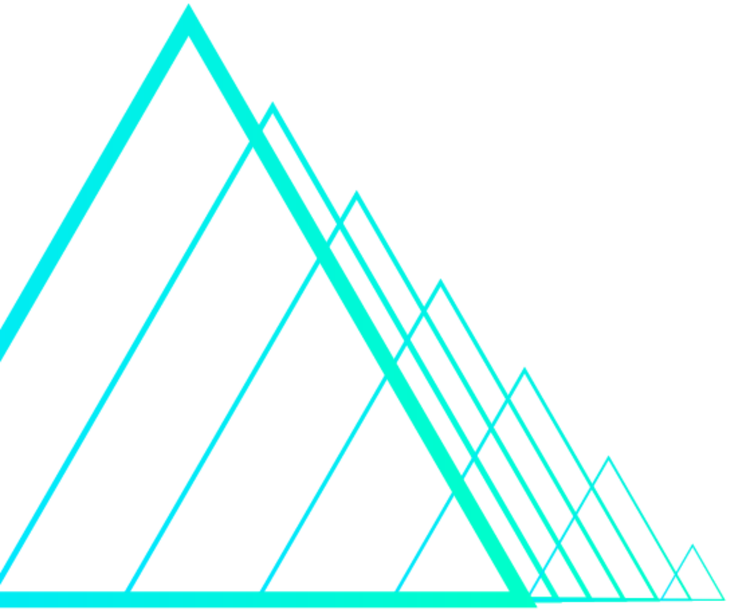
4.4 Custom Plugin 기반 Alerting

Alert 발생 예

- Alerting Event 발생 시, 등록한 Notification Channel로 이벤트 전송
- 🎨 Grafana Image Renderer와 연동하여 이미지 또한 이벤트 수신 시 확인 가능







Q & A

Thank You