**[SNMP capture를 통한 NMS]  
프로젝트 통합 보고서**

|  |  |
| --- | --- |
| **작 성 일** | 2024.07.22 |
| **작성부서** | codej |
| **작 성 자** | 배훈규, 양현성 |



**목차**

[1 사업개요 4](#_Toc175058790)

[1.1 추진배경 4](#_Toc175058791)

[1.2 추진목적 4](#_Toc175058794)

[1.3 추진방침 4](#_Toc175058796)

[2 과업 내역 5](#_Toc175058797)

[2.1 과업 내역 총괄 5](#_Toc175058798)

[2.2 과업 세부 내역 5](#_Toc175058799)

[2.3 용역 수행 절차 및 범위 6](#_Toc175058800)

[2.3.1 수행 단계별 산출물 6](#_Toc175058801)

[2.3.2 표준 개발 환경 6](#_Toc175058802)

[3 추진 일정 7](#_Toc175058803)

[3.1 주요 업무별 마일스톤 7](#_Toc175058804)

[3.2 실제 업무 진행 이력 7](#_Toc175058805)

[4 용역 추진 관리 8](#_Toc175058806)

[4.1 주요 보고 업무 8](#_Toc175058807)

[5 설계 9](#_Toc175058808)

[5.1 아키텍처 설계 9](#_Toc175058809)

[5.2 아키텍처 상세 내용 9](#_Toc175058810)

[5.3 ERD 설계 11](#_Toc175058811)

[5.4 시스템 구성 11](#_Toc175058812)

[6 구현 12](#_Toc175058813)

[6.1 개발 내역 12](#_Toc175058814)

[6.2 개발 내용 12](#_Toc175058815)

[6.2.1 AP 데이터 수집 12](#_Toc175058816)

[6.2.2 Log 데이터 수집 16](#_Toc175058817)

[6.2.3 Queue data 처리 18](#_Toc175058818)

[6.2.4 사용자 인터페이스 구성 25](#_Toc175058819)

# 사업개요

## 추진배경

* SNMP의 MIB를 통한 AP의 데이터 수집
* Syslog의 수집과 AP의 데이터와 연동

## 추진목적

* AP의 데이터 수집을 통한 해당 LAN의 상태 확인 및 장애 확인
* AP를 통한 정보와 각각의 Syslog를 통한 정보로 자세한 상황 확인

## 추진방침

* 분할 업무를 통한 개발속도 증가
* 주간 회의를 통한 feedback 활용
* 업무 전환 시 품질 확보 및 장애 예방을 위한 품질 활동 강화

# 과업 내역

## 과업 내역 총괄

|  |  |  |
| --- | --- | --- |
| **구분** | **주요 기능** | **비고** |
| **데이터 수집** | ㆍAP의 데이터 수집  ㆍsyslog의 데이터 수집 |  |
| **DB 관리** | ㆍ수집한 AP의 데이터에서 네트워크 관리나 syslog 연결에 필요한 요소 저장  ㆍ수집할 syslog의 구조에 맞게 저장 및 syslog의 구조 수정 |  |
| **UI구성** | ㆍinterface별 traffic 및 활성화 정보 표시  ㆍAP에 연결되어 있는 IP의 정보 표시  ㆍ특정 IP들의 syslog 확인 |  |

## 과업 세부 내역

|  |  |
| --- | --- |
| **구분** | **기능 설명** |
| **AP의 데이터 수집** | ㆍtelegraf에서 SNMP를 이용하여 AP의 데이터 수집  ㆍinterface별로 발생하는 traffic이나 해당 interface로 연결되어 있는 IP 주소 등의 데이터 수집 |
| **Syslog의 데이터 수집** | ㆍIP별로 발생하는 log를 수집  ㆍlog의 형태를 지정하여 심각도 및 전송한 log의 IP를 함께 전송 |
| **수집한 데이터 처리** | ㆍ각각의 데이터는 RabbitMQ에 저장  - telegraf를 사용한 AP의 데이터는 그대로 Json의 형태로 저장  - syslog의 경우 XML의 형식으로 spring boot를 사용하여 저장  ㆍ수집한 데이터 처리  - 두 개의 데이터 모두 spring boot를 이용하여RabbitMQ에서  PostgreSQL의 DB로 저장  - AP의 데이터의 경우 Json의 형식으로 syslog의 경우 XML의 형식으로 parsing의 방식으로 사용하여 데이터를 처리 |
| **UI구성** | ㆍinterface별 발생하는 traffic표시  ㆍAP에 연결되어 있는 IP 확인  ㆍ연결되어 있는 IP의 syslog 표시  ㆍ연결되어 있는 IP 별로 traffic 표시 |

## 용역 수행 절차 및 범위

### 수행 단계별 산출물

* KTSSP(KT SW개발표준) 산출물 작성 및 제출

| **프로세스** | **단계** | **필수 산출물** | **형식** |
| --- | --- | --- | --- |
| 프로젝트계획 | 프로젝트계획 | WBS | 문서 |
| 프로젝트실행 | 아키텍처정의 | 시스템아키텍처정의서 | 문서 |
| 설계 | 물리ERD | 문서 |
| 테이블정의서 | 문서 |
| 구현 | 소스코드 | 소스 |
| 프로젝트종료 | 완료 | 프로젝트완료보고서 | 문서 |

- 산출물: 6종 (문서 5종 + 소스코드 1종)

### 표준 개발 환경

| **구분** | **표준개발도구** | **설명** | **필수** |
| --- | --- | --- | --- |
| 형상관리 | Git | 소스코드 버전관리 도구 | 필수 |
| 통합 빌드 | spring boot | 지속적인 통합 빌드/배포 도구 | 필수 |

# 추진 일정

## 주요 업무별 마일스톤

텍스트, 스크린샷, 디스플레이, 번호이(가) 표시된 사진

자동 생성된 설명

## 실제 업무 진행 이력

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

# 용역 추진 관리

|  |  |  |  |
| --- | --- | --- | --- |
| **주요 업무** | **주기** | **협력업체** | **감독 부서** |
| **주간 업무 보고** | 주간 | 금주 실적/차주 계획 작업 | 주간 보고 주관 및 확인 |
| **장애 및 이슈 사항 보고** | 수시 | 장애 및 이슈 사항 보고 | 장애 및 이슈 사항 해결/조정 |

## 주요 보고 업무

# 설계

## 아키텍처 설계

* AP 및 syslog 데이터 수집
* Grafana를 이용한 traffic 감시

텍스트, 스크린샷, 도표, 직사각형이(가) 표시된 사진

자동 생성된 설명

## 아키텍처 상세 내용

* Telegraf를 사용하여 SNMP를 통해 AP의 데이터 수집
* Rsyslog를 통해 log의 데이터 전송 및 spring boot를 사용해 log를 수집
* 수집한 데이터를 RabbitMQ를 통해 처리

텍스트, 폰트, 라인, 도표이(가) 표시된 사진

자동 생성된 설명

* Spring boot를 사용하여 RabbitMQ에 처리한 데이터를 데이터 베이스에 저장

텍스트, 스크린샷, 도표, 라인이(가) 표시된 사진

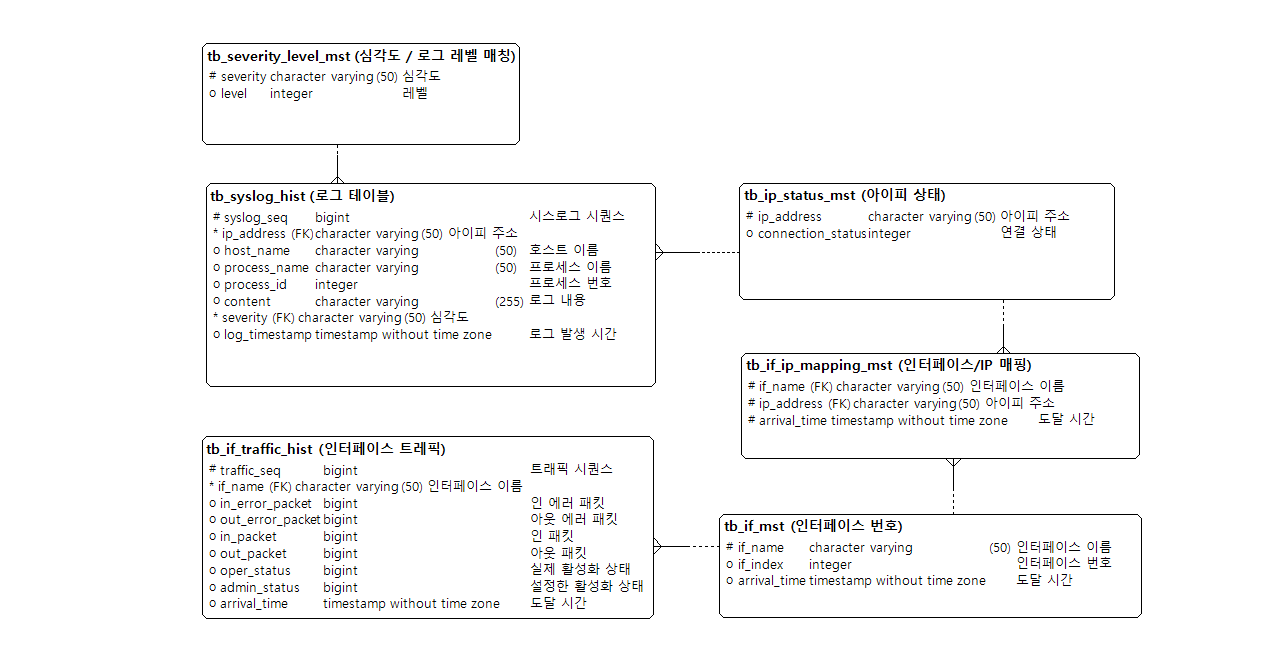
자동 생성된 설명

* Grafana를 사용하여 데이터 베이스에 저장한 트래픽 감시

텍스트, 스크린샷, 도표, 라인이(가) 표시된 사진

자동 생성된 설명

## ERD 설계



## 시스템 구성

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

# 구현

## 개발 내역

|  |  |
| --- | --- |
| **구 분** | **과 제 내 용** |
| 데이터 수집 | * Telegraf를 활용한 AP 데이터 수집 * Spring boot를 사용한 server 및 local의 syslog 수집 * 수집한 두 데이터를 모두 RabbitMQ에 저장 |
| RabbitMQ와 DB 관리 | * RabbitMQ에 저장된 데이터를 spring boot를 사용하여 PostgreSQL에 저장 * 멀티쓰레드 큐를 사용 |
| UI구성 | * DB에 저장한 데이터를 Grafana를 사용하여 표시 * Grafana 대시보드 구성 및 코딩 * Alerting 기능 추가 |

## 개발 내용

### AP 데이터 수집

* Telegraf를 활용한 AP의 연결 정보 수집: 다중 큐를 활용해 병렬적으로 구성
  1. ifTrafficQueue: 데이터 송수신 량 및 인터페이스 정보 등 수집
  2. ipIfIndexQueue: 현재 연결된 Ip주소들과 인터페이스 정보 수집
* Telegraf.conf 파일 구성

1. agent 구성

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

* interval: Telegraf가 데이터를 수집하는 주기
* flush\_interval: Telegraf가 수집하 데이터를 출력 플러그인으로 전송하는 주기

1. 출력 플러그인 구성

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

* outputs.amqp: 데이터를 amqp브로커(RabbitMQ)로 전송하기 위한 출력 플러그인 설정
* url: RabbitMQ 서버 연결 정보 설정
* exchange: 메시지 전송에 사용할 exchange 설정
* routing\_key: 메시지 전송 시 사용할 routing key 설정
* data\_format: 출력 데이터 형식 지정
* namepass: input에서 특정 플러그인의 데이터만 전송하도록 필터링하여 다중 큐 구현

1. 입력 플러그인 구성: Traffic Queue

[[inputs.snmp]]

interval = "10s"

agents = ["192.168.0.1:161"]

version = 2

community = "codej"

[[inputs.snmp.table]]

name = "interface"

inherit\_tags = ["hostname"]

[[inputs.snmp.table.field]]

  name = "ifDescr"

oid = "1.3.6.1.2.1.2.2.1.2"

is\_tag = true

[[inputs.snmp.table.field]]

        name = "ifIndex"

        oid = "1.3.6.1.2.1.2.2.1.1"

[[inputs.snmp.table.field]]

        name = "ifInOctets"

        oid = "1.3.6.1.2.1.2.2.1.10"

[[inputs.snmp.table.field]]

        name = "ifOutOctets"

        oid = "1.3.6.1.2.1.2.2.1.16"

[[inputs.snmp.table.field]]

        name = "ifInErrors"

        oid = "1.3.6.1.2.1.2.2.1.14"

[inputs.snmp.table.field]]

        name = "ifOutErrors"

        oid = "1.3.6.1.2.1.2.2.1.20"

[[inputs.snmp.table.field]]

        name = "ifInDiscards"

        oid = "1.3.6.1.2.1.2.2.1.13"

[[inputs.snmp.table.field]]

name = "ifOutDiscards"

        oid = "1.3.6.1.2.1.2.2.1.19"

[[inputs.snmp.table.field]]

        name = "ifAdminStatus"

        oid = "1.3.6.1.2.1.2.2.1.7"

[[inputs.snmp.table.field]]

        name = "ifOperStatus"

        oid = "1.3.6.1.2.1.2.2.1.8"

[[inputs.snmp.tagpass]]

snmp\_table = ["interface"]

* inputs.snmp: 입력 플러그인 정의 - SNMP프로토콜을 이용해 데이터 수집
* interval: 데이터 수집 주기
* agent: SNMP 에이전트의 주소와 포트
* version: SNMP버전
* community: 보안 및 인증을 위한 커뮤니티 문자열
* inputs.snmp.table: 테이블 형식의 데이터를 수집하기 위한 설정 블록
* inputs.snmp.table.field: 수집된 데이터 항목
* inputs.snmp.tagpass: 수집된 메트릭에 대해 특정 태그가 일치하는 메트릭만 전송하도록 제한함. 위 코드에서는 “interface” 태그를 가지는 output에 수집된 매트릭을 전송

1. 입력 플러그인 구성: ipIndex Queue

[[inputs.snmp]]

interval = "10s"

agents = ["192.168.0.1:161"]

version = 2

community = "codej"

[[inputs.snmp.table]]

name = "active\_ip"

inherit\_tags = ["hostname"]

[[inputs.snmp.table.field]]

name = "ipAdEntIfIndex"

oid = "1.3.6.1.2.1.4.22.1.3"

[[inputs.snmp.table.field]]

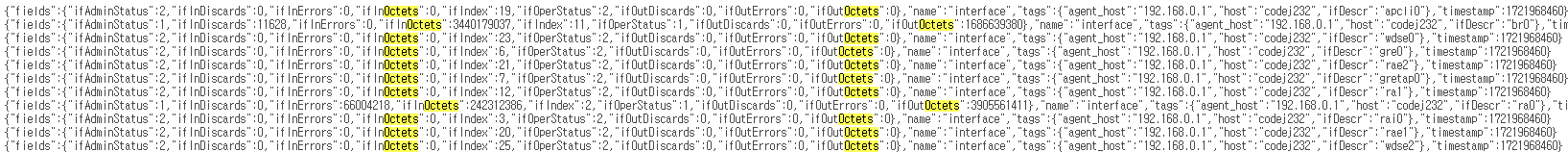
name = "ipNetToMediaIfIndex"

oid = "1.3.6.1.2.1.4.22.1.1"

[[inputs.snmp.tagpass]]

snmp\_table = ["active\_ip"]

* inputs.snmp: 입력 플러그인 정의 - SNMP프로토콜을 이용해 데이터 수집
* interval: 데이터 수집 주기
* agent: SNMP 에이전트의 주소와 포트
* version: SNMP버전
* community: 보안 및 인증을 위한 커뮤니티 문자열
* inputs.snmp.table: 테이블 형식의 데이터를 수집하기 위한 설정 블록
* inputs.snmp.table.field: 수집된 데이터 항목
* inputs.snmp.tagpass: 수집된 메트릭에 대해 특정 태그가 일치하는 메트릭만 전송하도록 제한함. 위 코드에서는 “active\_ip” 태그를 가지는 output에 수집된 매트릭을 전송
* ifTrafficQueue

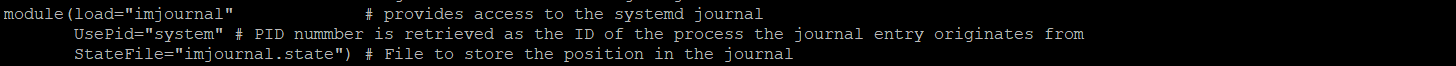


* ipIfIndexQueue



### Log 데이터 수집

* Log forwarding 지정 및 XML의 tag 형태로 지정



* Log의 timestamp 형식 설정

: module (load=”builtin:omfile” Template=”RSYSLOG TraditionalFileFormat”)

* Log forwarding 지정(UDP) 및 XML의 tag 형태로 지정

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

<logCaptureTimeStamp> => 로그 시간

<logCaptureHostName> => 호스트 이름

<logCaptureSyslogTag> => 로그가 발생한 프로세스 이름 및 아이디

<logCaptureMsg> => 로그 내용

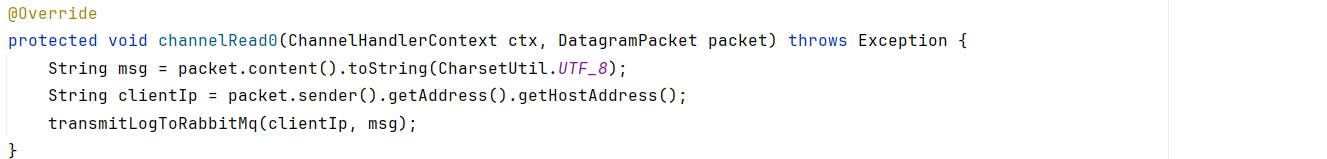
<logCaptureSeverity> => 로그의 심각도

* 위의 로그를 spring boot로 UDP의 방식으로 수신 및 RabbitMQ로 전송

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

* udpServerHandler에서 로그를 전송한 device의 IP 주소 확인 후 message에 추가 후 RabbitMQ로 전송



### Queue data 처리

* ifTrafficQueue의 data 처리 흐름도
  1. 메시지 처리

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

* 1. DB 연동

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

* ifTrafficQueue의 메시지 처리 Code
  1. 메시지 각 줄에 대해 서비스 클래스의 메서드 호출

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

* RabbitMQ에 줄로 구분하여 저장되어 있으므로 “\n”로 구분하여 메시지 처리
* 각 데이터가 json의 형태이므로 역직렬화를 통해 Map 자료구조로 변환한 후 필요한 데이터 처리
  1. DB 저장

1. TrafficRepository: arrival\_time 기준으로 가장 최근 항목 하나 가져오기

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

1. fields 데이터의 트래픽 정보와 tags 데이터의 ifDescr을 묶어서 DB에 저장

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

1. fields 데이터의 트래픽 정보와 tags 데이터의 ifDescr을 묶어서 DB에 저장

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* RabbitMQ에서 parsing한 데이터들을 DB에 저장
* 저장하는 데이터의 종류에 따라 오류가 발생하는 것을 고려
* JpaRepository: Spring Data JPA에서 제공하는 인터페이스, DAO 수행
* ifIndexQueue의 data 흐름도
  1. 메시지 처리

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

* 1. DB 연동

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

* ifIndexQueue의 code
  1. 각 줄의 데이터 마다 클래스를 호출하여 메시지 처리

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

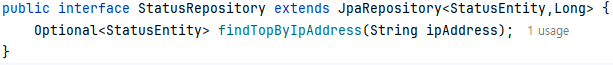
* + - 각 줄마다 하나의 데이터가 존재하므로 줄로 구분하여 메시지 처리
    - 각 데이터가 json의 형태이므로 역직렬화를 통해 Map자료구조로 변환한 후 필요한 데이터 처리
  1. DB 저장



텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

* if\_index를 바탕으로 IP와 ifDescr 매핑해 DB에 저장
* MappingEntity로 추출한 데이터와 시간을 연결하여 저장
* JpaRepository: Spring Data JPA에서 제공하는 인터페이스, DAO수행
  1. 최신 테이블 업데이트



텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

* 현재 메시지에서 실시간으로 수집한 데이터(current\_up\_status)를 통해 ip\_status 갱신
* 현재 메시지에서 실시간으로 수집한 데이터(current\_ip\_status)를 바탕으로 현재 메시지를 처리하기 전 가장 최근에 수집됐던 데이터(ip\_status)를 갱신
* JpaRepository: Spring Data JPA에서 제공하는 인터페이스, DAO수행
* findTopByIpAddress: 중복 데이터 방지를 위해 Ip주소로 검색된 결과 중 한 개를 가져오며 StatusEntity 객체를 안전하게 Optional로 감싸 반환 값이 존재하지 않더라도 안전하게 처리하는 쿼리 메서드
* 메시지 데이터의 Key 값에 해당하는 항목들 enum으로 저장

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

* LogQueue의 data 흐름도

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

* LogQueue의 메시지 처리 code
  1. Log 메시지를 바탕으로 LogEntity를 반환하는 서비스 클래스 호출



* 1. Log 메시지를 Parsing하하여 LogEntity를 반환하는 서비스 클래스 code

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

* 1. LogEntity를 DB에 저장하는 code



텍스트, 폰트, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* JpaRepository: Spring Data JPA에서 제공하는 인터페이스, DAO수행

### 사용자 인터페이스 구성

* Traffic 감시

스크린샷, 텍스트, 멀티미디어 소프트웨어, 그래픽 소프트웨어이(가) 표시된 사진

자동 생성된 설명

* Interface 마다 10초 동안 발생한 in/out/total traffic
* 실제 DB에 저장되는 것은 누적 값으로 이전의 값과 현재 값의 차이로 10초 동안의 traffic 양을 결정 (누적의 최댓값은 2^32 byte, 만약 이전의 누적 값이 현재 값보다 크게 되면 2^32을 더해서 계산)
* Interface 마다 10초 동안 발생한 in/out/total error traffic
* Interface 별 발생 traffic과 동일한 방식으로 계산
* 특정 시간 별로 자세한 수치 확인 가능
* Interface 정보 확인

텍스트, 스크린샷, 폰트, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

* Interface 전체에서 10초 동안 발생하는 전체 트래픽 및 에러 트래픽의 양 표시
* Interface 별로 연결된 IP의 종류 확인
* 해당 interface의 활성화 상태 확인

(Admin: 관리자가 설정한 Interface의 상태, Oper: 실제 작동하는 Interface의 상태)

* Admin과 Oper의 상태로 interface의 오류 확인 후 Alert 발생
* Syslog data

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

* AP에 연결되어 있는 모든 IP 주소 확인 가능
* 연결되어 있는 IP 중에서 특정 IP의 log를 확인 가능

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명



* 날짜 및 로그 레벨로 log를 filtering 가능
* Traffic 패널과 날짜가 연결되어 있어 특정 시간대의 로그를 확인하면 자동으로 traffic 패널에서 해당 로그 시간 범위의 traffic을 확인 가능
* Log number를 이용하여 확인하는 로그의 개수 제한
* Alerting
  1. Condition

스크린샷, 텍스트, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명 텍스트, 소프트웨어, 멀티미디어 소프트웨어, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* + - 관리자가 설정한 interface의 활성화 상태(admin status)와 실제 interface의 활성화 상태(oper status)가 서로 다른 특정 경우에 오류의 상황으로 구분
    - B의 condition을 sum으로 계산하도록 하여 A의 condition에서 에러 상황이 발생하면 Alert를 발생시키도록 설계
    - 에러 상황
  + Admin\_status = 1: interface 활성화
  + Oper\_status = 2: interface 비활성화 => interface에 문제 발생
  + Oper\_status = 4: interface 확인 불가
  + Oper\_status = 6: interface의 구성요소 누락
  + Oper\_status = 7: 하위 interface의 오류 발생
  1. 주기

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

* Evaluation interval: Alerting 위반 상황을 확인하는 주기.
* Pending period: 장애가 해당 시간동안 유지되는 경우에 Alerting을 발생
  1. 메시지 내용

텍스트, 스크린샷, 멀티미디어 소프트웨어, 디자인이(가) 표시된 사진

자동 생성된 설명

* Summary: 간단한 요약 내용
* Description: Go Language를 이용하여 오류가 발생한 interface의 이름과 발생한 오류의 방식을 알림
  1. Error case & 테스트 화면

텍스트, 스크린샷, 번호, 라인이(가) 표시된 사진

자동 생성된 설명

* + - 모든 에러의 경우로 총 4가지 경우의 상황을 모두 테스트

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

* + - 오류가 발생한 상황에서 Grafana의 화면입니다.

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* + - 오류가 발생한 경우 telegram으로 전송되는 메시지입니다.