

Extensions

release date. 2026. 09. 10.

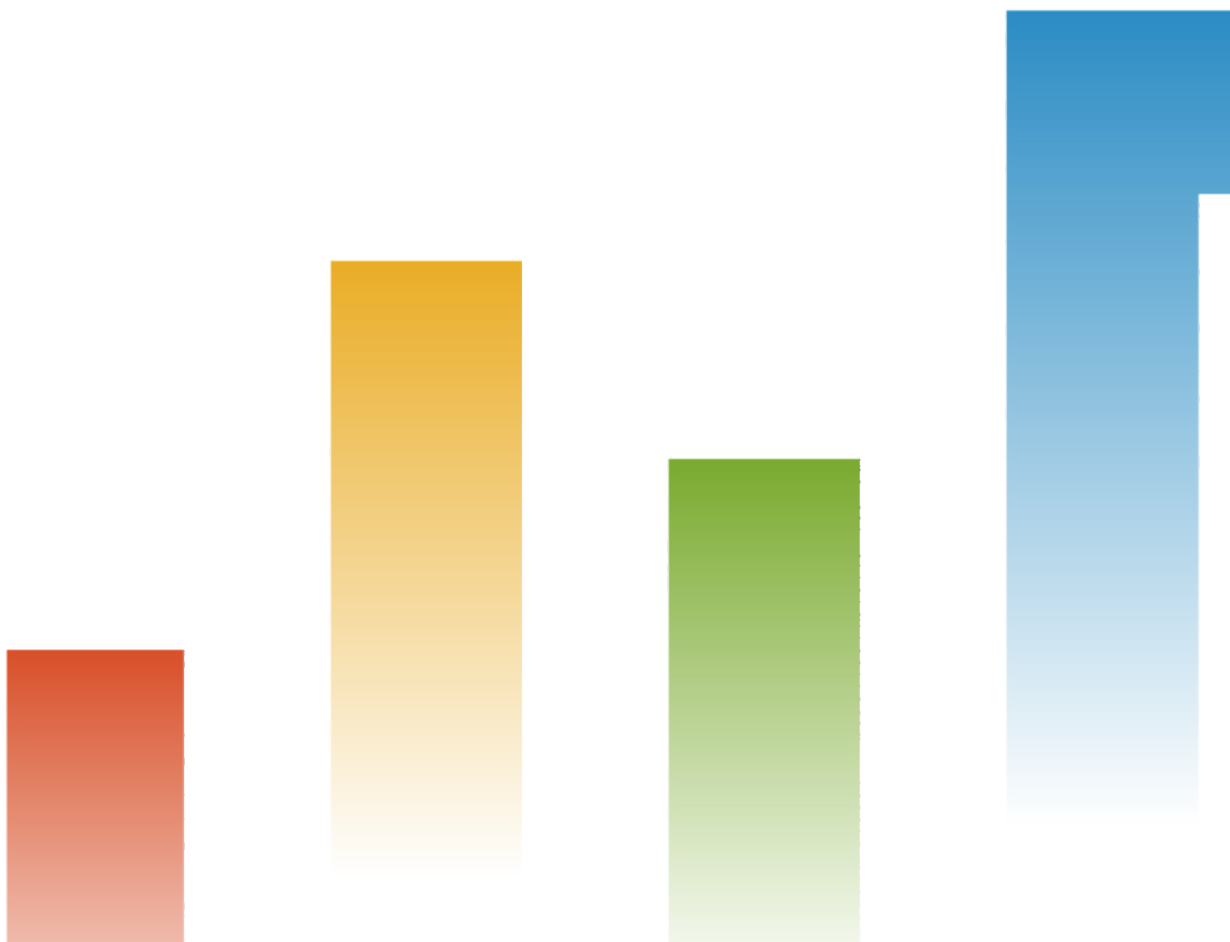


Table of Contents

• 확장 도구(Extensions)	3
◦ Focus	4
▪ 지원 환경	5
▪ 설치하기	6
▪ 설정하기	8
▪ 사용 예시	15
◦ Telegraf	18
▪ 플러그인 설치와 활용	
▪ 플러그인 설치	21
▪ 설정하기	28
▪ 에이전트 설치와 활용	
▪ 지원 환경	30
▪ 에이전트 설치	31
▪ 설치 문제 해결	36
▪ 설정하기	37
▪ 관리하기	40
▪ 사용 예시	43
◦ OpenTelemetry	47
▪ 에이전트 설치	49
▪ 에이전트 설정	59
▪ 릴리즈 노트	0
▪ 최신 에이전트 다운로드	68

확장 도구(Extensions)

와탭 서비스 수집 데이터를 확장할 수 있는 도구를 소개합니다. 다음 자료 외에 추가 내용 요청 등의 피드백은 docs@whatap.io로 보내주세요.

Focus

4 항목

Telegraf

3 항목

OpenTelemetry

4 항목

Focus란?

Focus는 와탭 모니터링에서 제공하지 않는 임의의 데이터를 시계열로 업로드하는 프로그램입니다.

운영팀에서 자체 개발한 체크 스크립트나 데이터베이스 쿼리를 주기적 혹은 일회성으로 작동하여 수집된 값을 업로드 할 수 있습니다. 업로드된 값은 해당 프로젝트의 분석/메트릭스 차트 메뉴에서 확인할 수 있습니다. 또 Flex 보드에 추가할 수 있습니다.

Focus를 사용하려면 와탭 프로젝트 중 Server, Applicatoon(Java, PHP, NodeJS, Python), Database, Kubernetes 프로젝트 중 하나를 선택 후에 해당 프로젝트의 프로젝트 액세스 키 문자열과 수집 서버 IP를 실행 옵션으로 지정하여 사용하실 수 있습니다.

지원 환경

X86 기반에서 동작하는 Windows, Linux, FreeBSD, OSX 및 ARMv5,6,7 및 ARM64 Linux OS에 적용할 수 있습니다.

제품	환경	지원 환경
Focus	Windows	Windows 2008R2 이상
	Linux	Debian 7.0 이상
		Ubuntu 12.04 이상
		CentOS, Red Hat 6.0 이상
		Amazon Linux 1.0 이상
		SUSE 12.1 이상
	FreeBSD	FreeBSD 10 이상
	OSX	Lion 10.7 이상
	Raspberry Pi OS	8 이상

ⓘ 지원 리스트에 포함되지 않는 환경인 경우 support@whatap.io 로 별도 문의하세요.

설치하기

와탭 포커스는 단일 실행 파일로 구성되어 별도의 설치 과정이나 등록 과정이 불필요합니다. 실행 파일을 다운로드해 임의의 위치에서 실행할 수 있습니다.

실행 파일 다운로드

와탭 Focus는 단일 실행 파일로 구성되어 버전별로 다운로드해 바로 사용할 수 있습니다.

64 bit Debian / Ubuntu / Red Hat / CentOS / Amazon Linux / SUSE

```
wget https://repo.whatap.io/focus/linux_amd64/focus
chmod +x ./focus
```

64 bit OSX

```
wget https://repo.whatap.io/focus/darwin_amd64/focus
chmod +x ./focus
```

64 bit FreeBSD

```
wget https://repo.whatap.io/focus/freebsd_amd64/focus
chmod +x ./focus
```

64 bit OSX

```
wget https://repo.whatap.io/focus/darwin_amd64/focus
chmod +x ./focus
```

64 bit Windows

```
Invoke-WebRequest -Uri https://repo.whatap.io/focus/windows_amd64/focus.exe -OutFile focus.exe
```

ArmV5 Linux

```
wget https://repo.whatap.io/focus/linux_arm_5/focus
chmod +x ./focus
```

ArmV6 Linux

```
wget https://repo.whatap.io/focus/linux_arm_6/focus
chmod +x ./focus
```

ArmV7 Linux

```
wget https://repo.whatap.io/focus/linux_arm_7/focus
chmod +x ./focus
```

Arm64 Linux

```
wget https://repo.whatap.io/focus/linux_amd64/focus
chmod +x ./focus
```

Focus 파일 구성

- Linux / FreeBSD / OSX / Raspberry Pi OS
- **focus**: 데이터 수집 및 전송 프로그램입니다.
- Windows
- **focus.exe**: 데이터 수집 및 전송 프로그램입니다.

설정하기

와탭 프로젝트에서 임의의 수집 데이터를 업로드할 수 있습니다. Focus는 해당 프로젝트의 기본 에이전트가 설치되지 않아도 사용할 수 있습니다.

프로젝트 액세스 키 및 수집 서버 IP 확인

선택한 프로젝트의 관리 > 에이전트 설치 메뉴에서 프로젝트 코드, 수집 서버 IP와 프로젝트 액세스 키를 확인합니다.

- [Linux Shell](#)
Windows Powershell

SH

```
export WHATAP_LICENSE=xxxx-xxxxxx-xxxxxx
export WHATAP_HOST=xxxx.xxxx.xxxx
export WHATAP_PCODE=xxx
```

SH

```
$WHATAP_LICENSE=xxxx-xxxxxx-xxxxxx
$WHATAP_HOST=xxxx.xxxx.xxxx
$WHATAP_PCODE=xxx
```

필터를 위한 특정 태그 추가

사용자가 필터를 위한 특정 문자열을 태그로 설정할 수 있습니다. 설정을 통해 와탭 모니터링 서비스 화면에서 수집된 데이터를 해당 태그의 이름 및 값으로 필터링할 수 있습니다.

- [Linux Shell](#)
Windows Powershell

SH

```
1 # -tag.[사용자 태그 키 이름][공백][사용자태그 값]
2
3 ./focus -license $WHATAP_LICENSE \
4   -pcode $WHATAP_PCODE -server.host $WHATAP_HOST \
```

```
5 -tag.CustomTagName CustomTagValue \
6 -tag.MyServer1 server_01 \
7 ...
```

Powersehll

```
1 # -tag.[사용자 태그 키 이름][공백][사용자태그 값]
2
3 .\focus.exe -license $WHATAP_LICENSE `
4 -pcode $WHATAP_PCODE -server.host $WHATAP_HOST `
5 -tag.CustomTagName CustomTagValue `
6 -tag.MyServer1 server_01 `
7 ...
```

알림 보내기

사용자 정의 알림을 즉시 보낼 수 있습니다. 알림을 보낸 후 Focus는 종료합니다.

- [Linux Shell](#) Windows Powershell

SH

```
#알림의 심각도를 선택합니다.
level={info|warn|fatal}
#알림의 제목
title=원하는 알림의 제목
#알림의 본문
message=원하는 알림의 본문
./focus -license $WHATAP_LICENSE \
-pcode $WHATAP_PCODE -server.host $WHATAP_HOST \
-alert -level $level -title $title -message $message
```

Powersehll

```
#알림의 심각도를 선택합니다.
$level="{info|warn|fatal}"
#알림의 제목
$title="원하는 알림의 제목"
#알림의 본문
```

```
$message="원하는 알림의 본문"
.\focus.exe -license $WHATAP_LICENSE `
  -pcode $WHATAP_PCODE -server.host $WHATAP_HOST `
  -alert -level $level -title $title -message $message
```

SQL Query 결과 수집

사용자 정의 SQL Query의 실행 결과를 시계열로 무기한 수집합니다.

- [Linux Shell](#) Windows Powershell

SH

```
#매트릭 카테고리
category=my_category
#데이터 베이스 드라이버
driver={mysql|postgres}
#데이터베이스 접속 정보
dburl="아이디:패스워드@tcp(아이피:포트)/데이터베이스"
#Sql Query
sqlquery="select some, columns from sometable"
./focus -license $WHATAP_LICENSE \
  -category $category \
  -pcode $WHATAP_PCODE -server.host $WHATAP_HOST \
  -rdb $driver -rdb.connect $dburl \
  -rdb.sql $sqlquery
```

Powersehll

```
#매트릭 카테고리
$CATEGORY="my_category"
#데이터 베이스 드라이버
$driver="{mysql|postgres}"
#데이터베이스 접속 정보
$dburl="아이디:패스워드@tcp(아이피:포트)/데이터베이스"
#Sql Query
$sqlquery="select some, columns from sometable"

.\focus.exe -license $WHATAP_LICENSE `
```

```
-pcode $WHATAP_PCODE -server.host $WHATAP_HOST `
-category $CATEGORY `
-rdb $driver -rdb.connect $dburl `
-rdb.sql $sqlquery
```

로그 파일 키워드 매칭 로그 수집

지정한 텍스트 로그 파일에서 로그 발생 시 키워드를 포함하면 해당 로그 라인을 수집합니다.

- **Linux Shell** Windows Powershell

SH

```
#매트릭 카테고리
category=my_category
#로그 파일
LOG_FILE=로그 파일 경로
#2개 이상의 로그 키워드를 입력할때 키워드 사이에 사용할 구분자
LOG_KEYWORDS_SEPERATOR=,
#로그 키워드를 ,로 연결하여 입력
LOG_KEYWORDS=keyword1,keyword2
./focus -license $WHATAP_LICENSE \
-category $category \
-pcode $WHATAP_PCODE -server.host $WHATAP_HOST \
-tail $LOG_FILE \
-tail.keys $LOG_KEYWORDS \
-tail.seperator $LOG_KEYWORDS_SEPERATOR
```

Powersehll

```
#매트릭 카테고리
$CATEGORY="my_category"
#로그 파일
LOG_FILE=로그 파일 경로
#2개 이상의 로그 키워드를 입력할때 키워드 사이에 사용할 구분자
LOG_KEYWORDS_SEPERATOR=,
#로그 키워드를 구분자로 연결하여 입력
LOG_KEYWORDS=keyword1,keyword2
```

```
.\focus.exe -license $WHATAP_LICENSE `
  -pcode $WHATAP_PCODE -server.host $WHATAP_HOST `
  -category $CATEGORY `
  -tail $LOG_FILE `
  -tail.keys $LOG_KEYWORDS `
  -tail.seperator $LOG_KEYWORDS_SEPERATOR
```

와탭 로그 분석 서비스

지정한 텍스트 로그 파일에서 로그 발생 시 와탭 로그 분석 서비스로 실시간 업로드합니다.

- [Linux Shell](#) Windows Powershell

SH

```
#카테고리
category=my_category
#로그 파일 wildcard(*), 날짜패턴(http://strftime.org) 포함 가능
LOG_FILE=로그 파일 경로
.\focus -license $WHATAP_LICENSE \
  -category $category \
  -pcode $WHATAP_PCODE -server.host $WHATAP_HOST \
  -logsink $LOG_FILE
```

Powersehll

```
#카테고리
$CATEGORY="my_category"
#로그 파일 wildcard(*), 날짜패턴(http://strftime.org) 포함 가능
$LOG_FILE=로그 파일 경로
.\focus.exe -license $WHATAP_LICENSE `
  -pcode $WHATAP_PCODE -server.host $WHATAP_HOST `
  -category $CATEGORY `
  -logsink $LOG_FILE
```

OS 자원 사용량 수집

Focus가 작동 중인 서버의 자원 사용량을 수집합니다.

Linux Shell

```
#매트릭 카테고리
category=my_category
#디스크 모니터링 활성화 여부
diskenabled=true|false
#모니터링할 디스크 마운트
diskmount=/mypartition
#NIC 모니터링 활성화 여부
nicenabled=true|false
#모니터링할 NIC
nic=eth0

./focus -license $WHATAP_LICENSE \
  -category $category \
  -pcode $WHATAP_PCODE -server.host $WHATAP_HOST \
  -sys \
  -sys.disk.enabled $diskenabled \
  -sys.disk $diskmount \
  -sys.net.enabled $nicenabled \
  -sys.net $nic
```

임의 프로그램 및 스크립트 실행 결과 수집

임의의 프로그램 및 스크립트를 실행하고 stdout으로 출력되는 결과를 지속적으로 수집합니다. 와탭 Focus는 json dictionary 형식으로 stdin으로 입력되면 해당 dictionary의 key, value를 수집합니다. 입력 수단으로 파이프를 사용하게 되므로 버퍼링을 사용하지 않도록 설정하는 것이 필요합니다.

- ❗ • 지속적으로 수집하는 것이 아니라 일회성으로 수집하기 원하면 다음 옵션을 적용할 수 있습니다. `-onetime`
- 수집 시간을 지정하고 싶으면 다음 옵션으로 지정할 수 있습니다. `-now {unix epoch time(second)}`

Linux Shell

```
# 매트릭 카테고리
CATEGORY="my_category"

프로그램 혹은 스크립트 | \
json dictionary 형태로 재가공 | \
./focus -license $WHATAP_LICENSE \
  -category $category \
  -pcode $WHATAP_PCODE -server.host $WHATAP_HOST
```

다음 예시는 top 명령을 실행하여 특정 프로세스의 CPU, memory 사용량을 계속하여 수집합니다.

Linux Shell

```
# 매트릭 카테고리
CATEGORY="my_category"

export PID=수집하기 원하는 프로세스의 PID
top -b -p $PID | awk '/$PID/{ printf "{\"pid\": %s, \"cpuPercent\": %s, \"memoryPercent\": %s, \"cmd\": \"%s\"}\n", $1, $9, $10, $12}; system("")' | \
./focus -license $WHATAP_LICENSE \
  -pcode $WHATAP_PCODE -server.host $WHATAP_HOST \
  -category $CATEGORY
```

사용 예시

Focus를 활용해 자주 사용하는 명령 및 기타 활용 사례를 안내합니다. 이를 통해 개발 및 운영 시 확인할 수 있는 데이터를 시계열로 확인할 수 있도록 하여 개발 운영에 도움이 되었으면 합니다.

프로젝트 액세스 키 및 수집 서버 IP 확인

선택한 프로젝트의 관리 > 에이전트 설치 메뉴에서 프로젝트 코드, 수집 서버 IP와 프로젝트 액세스 키를 확인합니다.

- Linux Shell Windows Powershell

SH

```
export WHATAP_LICENSE=xxxx-xxxxxx-xxxxxx
export WHATAP_HOST=xxxx.xxx.xxx
export WHATAP_PCODE=xxx
```

SH

```
$WHATAP_LICENSE=xxxx-xxxxxx-xxxxxx
$WHATAP_HOST=xxxx.xxx.xxx
$WHATAP_PCODE=xxx
```

필터를 위한 특정 태그 추가

다음 예시를 참조해 와탭 모니터링 서비스 화면에서 수집된 데이터를 해당 태그의 이름 및 값으로 필터링할 수 있습니다.

SH

```
1 # -tag.[사용자 태그 키 이름][공백][사용자태그 값]
2
3 ./focus -license $WHATAP_LICENSE \
4 -pcode $WHATAP_PCODE -server.host $WHATAP_HOST \
5 -tag.CustomTagName CustomTagValue \
6 -tag.MyServer1 server_01 \
```

TOP 명령어 pid 별 CPU, Memory 수집

다음 예시는 top 명령을 실행하여 특정 프로세스의 CPU, memory 사용량을 계속하여 수집합니다.

Linux Shell

```
# 매트릭 카테고리
CATEGORY="my_category"

export PID=수집하기 원하는 프로세스의 PID
top -b -p $PID | awk '/$PID/{ printf "\tpid\: %s, \cpuPercent\: %s, \memoryPercent\: %s, \cmd\: \"%s\"\n", $1, $9, $10, $12}; system("")' | \
./focus -license $WHATAP_LICENSE \
-pcode $WHATAP_PCODE -server.host $WHATAP_HOST \
-category $CATEGORY
```

NETSTAT 활용

다음 예시는 netstat 명령을 실행하여 하여 TCP Connection 상태 별 개수를 수집합니다.

Linux Shell

```
# 매트릭 카테고리
CATEGORY="my_category"

netstat -nat | tail -n+3 | awk '{print $6}' | sort | uniq -c | awk 'BEGIN { printf "{" } {if (NR!=1) {printf ", "}}{printf "\n%s\:%s", $2, $1} END { print "}" }' | \
./focus -license $WHATAP_LICENSE \
-pcode $WHATAP_PCODE -server.host $WHATAP_HOST \
-category $CATEGORY -onetime
```

VMSTAT 활용

다음 예시는 vmstat 명령을 실행하여 항목별 지수를 수집합니다.

Linux Shell

```
#매트릭 카테고리
CATEGORY="my_category"

vmstat -n 5 | awk 'NR>2 {printf "{ \"r\":%s,\"b\":%s,\"swpd\":%s,\"free\":%s,\"buff\":%s,\"cache\":%s,\"si\":%s,\"so\":%s,\"bi\":%s,\"bo\":%s,\"in\":%s,\"cs\":%s,\"us\":%s,\"sy\":%s,\"id\":%s,\"wa\":%s,\"st\":%s }\n", $1,$2, $3,$4, $5,$6, $7,$8, $9,$10, $11,$12, $13,$14, $15,$16,$17}\n' | \
./focus -license $WHATAP_LICENSE \
-pcode $WHATAP_PCODE -server.host $WHATAP_HOST \
-category $CATEGORY
```

DU 활용

다음 예시는 du 명령을 실행하여 하위 임의의 디렉터리 용량을 수집합니다.

Linux Shell

```
#매트릭 카테고리
CATEGORY="my_category"
TARGET=용량수집을 원하는 디렉터리

du -sb $TARGET --max-depth=0 | awk 'BEGIN { printf "{" } {if (NR!=1) {printf ", "}}{printf "\"%s\":%s", $2,$1} END { print "}" }' | \
./focus -license $WHATAP_LICENSE \
-pcode $WHATAP_PCODE -server.host $WHATAP_HOST \
-category $CATEGORY -onetime
```

Telegraf란?

Telegraf는 데이터베이스, 시스템 및 IoT 센서에서 메트릭 및 이벤트를 수집하고 전송하기 위한 플러그인 중심 서버 에이전트입니다. Telegraf는 Go로 작성되었으며 외부 종속성 없이 단일 바이너리로 컴파일되며 최소한의 메모리 공간만 필요합니다.

- 모든 종류의 데이터 수집 및 전송

- 데이터베이스

- MongoDB, MySQL, Redis 등의 데이터 소스에 연결하여 메트릭을 수집하고 보냅니다.

- 시스템

- 현대적인 클라우드 플랫폼, 컨테이너 및 오케스트레이터 스택에서 메트릭을 수집합니다.

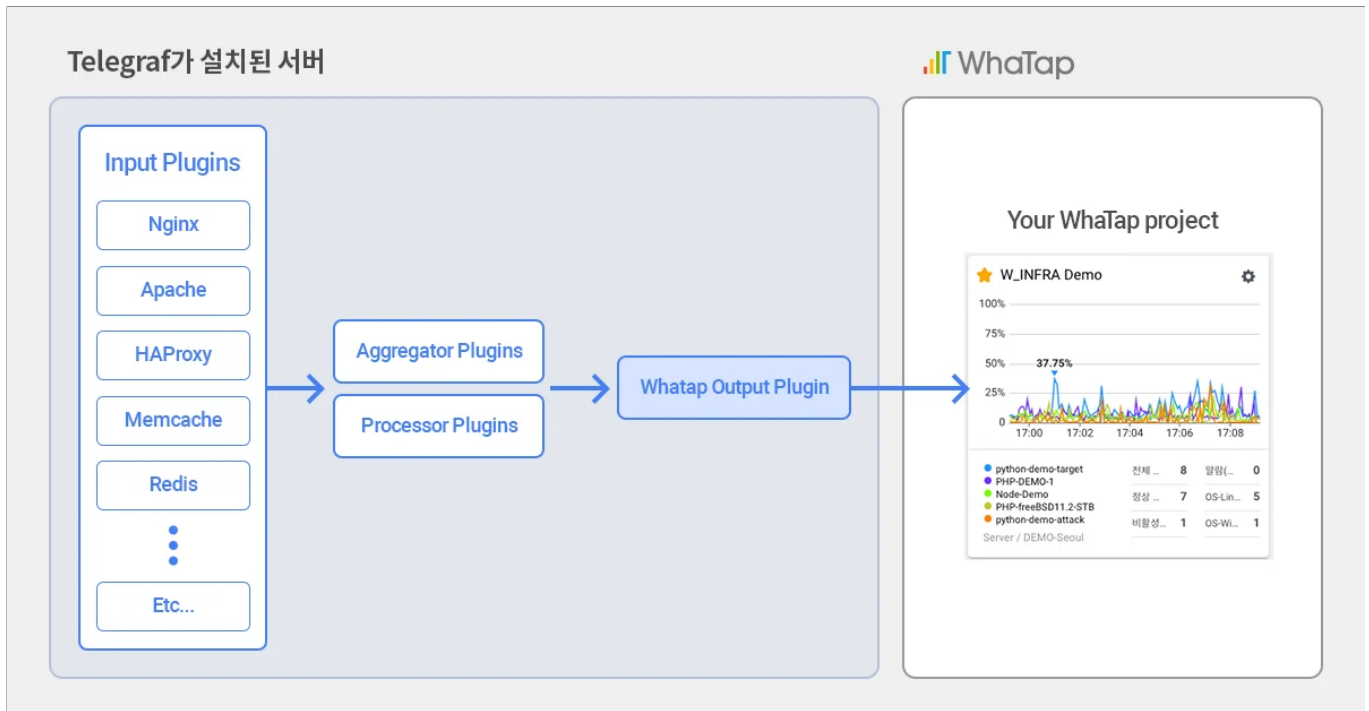
- IoT 센서

- IoT 센서 및 장치에서 중요한 상태 저장 데이터(압력 수준, 온도 수준 등)를 수집합니다.

- Telegraf 플러그인

Telegraf는 메트릭을 수집, 처리, 집계 및 작성하는 플러그인 중심 에이전트입니다. 입력, 출력, 애그리게이터 및 프로세서를 포함한 4가지 범주의 플러그인을 지원합니다.

Telegraf의 와탭 output plugin을 활용하고 와탭 Telegraf 에이전트를 설치하여 Telegraf에서 수집하는 메트릭스를 와탭 수집 서버로 전달할 수 있습니다.



ⓘ Telegraf 플러그인 또는 에이전트는 사용자가 해당 서비스에 직접 설치하고 설정해야 합니다.

플러그인 또는 에이전트 설치 방법 및 사용 예시 안내를 다음과 같이 제공합니다.

📄 플러그인 설치와 활용

2 항목

📄 에이전트 설치와 활용

5 항목

 사용 예시

사용 예시를 안내합니다.

플러그인 설치

whatap output plugin

Telegraf의 whatap output plugin(tcp)을 활용해 Telegraf에서 수집하는 메트릭스를 와탭 수집 서버로 전달할 수 있습니다. 수집된 데이터는 whatap의 프로젝트 내부에서 확인 가능합니다.

! Telegraf 플러그인은 사용자가 해당 서비스에 직접 설치하고 설정해야 합니다.

지원 환경

다음 목록에 있는 운영체제는 다운로드 설치가 가능합니다. 다른 운영체제는 [컴파일 설치](#)로 진행해 주세요.

- macOS
- Red Hat & CentOS
- Ubuntu & Debian
- FreeBSD
- windows

다운로드 설치

Telegraf release 버전을 기준으로 whatap plugin을 추가하여 미리 생성한 설치 파일입니다. Telegraf가 이미 설치되었으면 실행 파일만 교체해 whatap plugin을 활성화할 수 있습니다.

Telegraf release 버전

- [1.21.1](#) 1.16.0 1.15.4 1.14.5 1.13.2 1.12

- Red Hat & CentOS

- <https://repo.whatap.io/telegraf/telegraf-release-1.21.1/linux/amd64/telegraf> (실행파일)
- <https://repo.whatap.io/telegraf/telegraf-release-1.21.1/linux/amd64/telegraf-1.21.1-1.amd64.rpm>
- https://repo.whatap.io/telegraf/telegraf-release-1.21.1/linux/amd64/telegraf-1.21.1_linux_amd64.tar.gz
- **Ubuntu & Debian**
 - <https://repo.whatap.io/telegraf/telegraf-release-1.21.1/linux/amd64/telegraf> (실행파일)
 - https://repo.whatap.io/telegraf/telegraf-release-1.21.1/linux/amd64/telegraf_1.21.1-1_amd64.deb
 - https://repo.whatap.io/telegraf/telegraf-release-1.16.0/linux/amd64/telegraf-1.21.1_linux_amd64.tar.gz
- **macOS**
 - <https://repo.whatap.io/telegraf/telegraf-release-1.16.0/darwin/amd64/telegraf> (실행파일)
 - https://repo.whatap.io/telegraf/telegraf-release-1.16.0/darwin/amd64/telegraf-1.16.0_darwin_amd64.tar.gz
- **Red Hat & CentOS**
 - <https://repo.whatap.io/telegraf/telegraf-release-1.16.0/linux/amd64/telegraf> (실행파일)
 - https://repo.whatap.io/telegraf/telegraf-release-1.16.0/linux/amd64/telegraf-1.16.0-1.x86_64.rpm
 - https://repo.whatap.io/telegraf/telegraf-release-1.16.0/linux/amd64/telegraf-1.16.0_linux_amd64.tar.gz
- **Ubuntu & Debian**
 - <https://repo.whatap.io/telegraf/telegraf-release-1.16.0/linux/amd64/telegraf> (실행파일)
 - https://repo.whatap.io/telegraf/telegraf-release-1.16.0/linux/amd64/telegraf_1.16.0-1_amd64.deb
 - https://repo.whatap.io/telegraf/telegraf-release-1.16.0/linux/amd64/telegraf-1.16.0_linux_amd64.tar.gz
- **FreeBSD**
 - <https://repo.whatap.io/telegraf/telegraf-release-1.16.0/freebsd/amd64/telegraf> (실행파일)
 - https://repo.whatap.io/telegraf/telegraf-release-1.16.0/freebsd/amd64/telegraf-1.16.0_freebsd_amd64.tar.gz
- **windows**
 - <https://repo.whatap.io/telegraf/telegraf-release-1.16.0/windows/amd64/telegraf.exe> (실행파일)
 - https://repo.whatap.io/telegraf/telegraf-release-1.16.0/windows/amd64/telegraf-1.16.0_windows_amd64.zip

- macOS

- <https://repo.whatap.io/telegraf/telegraf-release-1.15.4/darwin/amd64/telegraf> (실행파일)
- https://repo.whatap.io/telegraf/telegraf-release-1.15.4/darwin/amd64/telegraf-1.15.4_darwin_amd64.tar.gz

- Red Hat & CentOS

- <https://repo.whatap.io/telegraf/telegraf-release-1.15.4/linux/amd64/telegraf> (실행파일)
- https://repo.whatap.io/telegraf/telegraf-release-1.15.4/linux/amd64/telegraf-1.15.4-1.x86_64.rpm
- https://repo.whatap.io/telegraf/telegraf-release-1.15.4/linux/amd64/telegraf-1.15.4_linux_amd64.tar.gz

- Ubuntu & Debian

- <https://repo.whatap.io/telegraf/telegraf-release-1.15.4/linux/amd64/telegraf> (실행파일)
- https://repo.whatap.io/telegraf/telegraf-release-1.15.4/linux/amd64/telegraf_1.15.4-1_amd64.deb
- https://repo.whatap.io/telegraf/telegraf-release-1.15.4/linux/amd64/telegraf-1.15.4_linux_amd64.tar.gz

- FreeBSD

- <https://repo.whatap.io/telegraf/telegraf-release-1.15.4/freebsd/amd64/telegraf> (실행파일)
- https://repo.whatap.io/telegraf/telegraf-release-1.15.4/freebsd/amd64/telegraf-1.15.4_freebsd_amd64.tar.gz

- windows

- <https://repo.whatap.io/telegraf/telegraf-release-1.15.4/windows/amd64/telegraf.exe> (실행파일)
- https://repo.whatap.io/telegraf/telegraf-release-1.15.4/windows/amd64/telegraf-1.15.4_windows_amd64.zip

- macOS

- <https://repo.whatap.io/telegraf/telegraf-release-1.14.5/darwin/amd64/telegraf> (실행파일)
- https://repo.whatap.io/telegraf/telegraf-release-1.14.5/darwin/amd64/telegraf-1.14.5_darwin_amd64.tar.gz

- Red Hat & CentOS

- <https://repo.whatap.io/telegraf/telegraf-release-1.14.5/linux/amd64/telegraf> (실행파일)
- https://repo.whatap.io/telegraf/telegraf-release-1.14.5/linux/amd64/telegraf-1.14.5-1.x86_64.rpm

- https://repo.whatap.io/telegraf/telegraf-release-1.14.5/linux/amd64/telegraf-1.14.5_linux_amd64.tar.gz
- **Ubuntu & Debian**
 - <https://repo.whatap.io/telegraf/telegraf-release-1.14.5/linux/amd64/telegraf> (실행파일)
 - https://repo.whatap.io/telegraf/telegraf-release-1.14.5/linux/amd64/telegraf_1.14.5-1_amd64.deb
 - https://repo.whatap.io/telegraf/telegraf-release-1.14.5/linux/amd64/telegraf-1.14.5_linux_amd64.tar.gz
- **FreeBSD**
 - <https://repo.whatap.io/telegraf/telegraf-release-1.14.5/freebsd/amd64/telegraf> (실행파일)
 - https://repo.whatap.io/telegraf/telegraf-release-1.14.5/freebsd/amd64/telegraf-1.14.5_freebsd_amd64.tar.gz
- **windows**
 - <https://repo.whatap.io/telegraf/telegraf-release-1.14.5/windows/amd64/telegraf.exe> (실행파일)
 - https://repo.whatap.io/telegraf/telegraf-release-1.14.5/windows/amd64/telegraf-1.14.5_windows_amd64.zip
- **macOS**
 - <https://repo.whatap.io/telegraf/telegraf-release-1.13.2/darwin/amd64/telegraf> (실행파일)
 - https://repo.whatap.io/telegraf/telegraf-release-1.13.2/darwin/amd64/telegraf-1.13.0~842282d_darwin_amd64.tar.gz
- **Red Hat & CentOS**
 - <https://repo.whatap.io/telegraf/telegraf-release-1.13.2/linux/amd64/telegraf> (실행파일)
 - https://repo.whatap.io/telegraf/telegraf-release-1.13.2/linux/amd64/telegraf-1.13.0~842282d-0.x86_64.rpm
 - https://repo.whatap.io/telegraf/telegraf-release-1.13.2/linux/amd64/telegraf-1.13.0~842282d_linux_amd64.tar.gz
- **Ubuntu & Debian**
 - <https://repo.whatap.io/telegraf/telegraf-release-1.13.2/linux/amd64/telegraf> (실행파일)
 - https://repo.whatap.io/telegraf/telegraf-release-1.13.2/linux/amd64/telegraf_1.13.0~842282d-0_amd64.deb

- https://repo.whatap.io/telegraf/telegraf-release-1.13.2/linux/amd64/telegraf-1.13.0~842282d_linux_amd64.tar.gz
- **FreeBSD**
 - <https://repo.whatap.io/telegraf/telegraf-release-1.13.2/freebsd/amd64/telegraf> (실행파일)
 - https://repo.whatap.io/telegraf/telegraf-release-1.13.2/freebsd/amd64/telegraf-1.13.0~842282d_freebsd_amd64.tar.gz
- **windows**
 - <https://repo.whatap.io/telegraf/telegraf-release-1.13.2/windows/amd64/telegraf.exe> (실행파일)
 - https://repo.whatap.io/telegraf/telegraf-release-1.13.2/windows/amd64/telegraf-1.13.0~842282d_windows_amd64.zip
- **macOS**
 - <https://repo.whatap.io/telegraf/telegraf-release-1.12/darwin/amd64/telegraf> (실행파일)
 - https://repo.whatap.io/telegraf/telegraf-release-1.12/darwin/amd64/telegraf-1.12.0~842282d_darwin_amd64.tar.gz
- **Red Hat & CentOS**
 - <https://repo.whatap.io/telegraf/telegraf-release-1.12/linux/amd64/telegraf> (실행파일)
 - https://repo.whatap.io/telegraf/telegraf-release-1.12/linux/amd64/telegraf-1.12.0~842282d-0.x86_64.rpm
 - https://repo.whatap.io/telegraf/telegraf-release-1.12/linux/amd64/telegraf-1.12.0~842282d_linux_amd64.tar.gz
- **Ubuntu & Debian**
 - <https://repo.whatap.io/telegraf/telegraf-release-1.12/linux/amd64/telegraf> (실행파일)
 - https://repo.whatap.io/telegraf/telegraf-release-1.12/linux/amd64/telegraf_1.12.0~842282d-0_amd64.deb
 - https://repo.whatap.io/telegraf/telegraf-release-1.12/linux/amd64/telegraf-1.12.0~842282d_linux_amd64.tar.gz
- **FreeBSD**
 - <https://repo.whatap.io/telegraf/telegraf-release-1.12/freebsd/amd64/telegraf> (실행파일)
 - <https://repo.whatap.io/telegraf/telegraf-release-1.12/freebsd/amd64/>

[telegraf-1.12.0~842282d_freebsd_amd64.tar.gz](https://repo.whatap.io/telegraf/telegraf-release-1.12.0~842282d_freebsd_amd64.tar.gz)

- windows
 - <https://repo.whatap.io/telegraf/telegraf-release-1.12/windows/amd64/telegraf.exe> (실행파일)
 - https://repo.whatap.io/telegraf/telegraf-release-1.12/windows/amd64/telegraf-1.12.0~842282d_windows_amd64.zip

컴파일 설치

사전 준비 사항

- Go SDK 구성
- go dep 설치
- 소스 파일

`${GOPATH}` [/src/github.com/influxdata/telegraf](https://src.github.com/influxdata/telegraf) 디렉터리 밑에 압축을 해제하세요.

① Telegraf release 버전에 whatap plugin을 포함한 소스 파일입니다.

- https://repo.whatap.io/telegraf/telegraf-release-1.12_whatap.tar.gz
- https://repo.whatap.io/telegraf/telegraf-release-1.13.2_whatap.tar.gz
- https://repo.whatap.io/telegraf/telegraf-release-1.14.5_whatap.tar.gz
- https://repo.whatap.io/telegraf/telegraf-release-1.15.4_whatap.tar.gz
- https://repo.whatap.io/telegraf/telegraf-release-1.16.0_whatap.tar.gz

설치 과정

① 상세한 빌드 안내는 [Telegraf github](#) 페이지를 참조하세요.

`${GOPATH}` [/src/github.com/influxdata/telegraf](https://src.github.com/influxdata/telegraf) 이하 Makefile로 빌드를 진행하세요.

1. 의존성 설치를 진행하세요. 의존성 설치는 go dep을 사용하기 때문에 go dep이 먼저 설치되어야 합니다.

```
# make deps
```

2. 실행 파일을 빌드하세요. `${GOPATH}/bin` 디렉터리에 실행 파일(telegraf)이 생성됩니다.

```
# make go-install
```

3. 설치 패키지가 필요하면 `make package` 명령어로 생성할 수 있습니다. 크로스 컴파일을 진행하여 운영 체제별로 패키지 파일을 생성하세요.

```
# make package
```

설정하기

Telegraf의 config 파일([telegraf.conf](#))에 다음 [outputs.whatap](#) 설정을 활성화하세요.

! Telegraf 플러그인은 사용자가 해당 서비스에 직접 설정해야 합니다.

whatap plugin이 포함된 config 파일은 다음과 같이 생성할 수 있습니다.

```
# telegraf --sample-config > telegraf.conf
```

whatap 프로젝트 정보

홈 화면에서 프로젝트를 선택하고 프로젝트의 정보를 설정하세요. 연결할 프로젝트의 에이전트 설치 안내에서 확인 가능합니다.

- **프로젝트 액세스 키** : 프로젝트를 식별하는 프로젝트 액세스 키입니다.
- **프로젝트 코드** : 프로젝트를 식별하는 숫자형 코드입니다.
- **수집 서버 정보(IP, 포트)** : 수집 정보를 전달할 와탭 수집 서버의 아이피와 포트입니다.

config 설정(toml)

/etc/telegraf/telegraf.conf

```
# # Configuration for WhaTap
[[outputs.whatap]]
# ## You can create a project on the WhaTap site(https://www.whatap.io)
# ## to get license, project code and server IP information.
#
# ## WhaTap license. Required
license = "x2tggnopk2t9-xxxxxxxxx-aaaaaaaaaa"
#
# ## WhaTap project code. Required
pcode = 118
#
# ## WhaTap server IP. Required
```

```
# # Put multiple IPs with / as delimiters. e.g. "1.1.1.1/2.2.2.2"

servers = ["tcp://1.2.3.4:6600", "tcp://5.6.7.8:6600"]

# ## Connection timeout.
# # timeout = "60s"
```

- `[[outputs.whatap]]` 주석(#)을 삭제하세요.
- `license` 의 주석(#)을 삭제하세요. 프로젝트의 프로젝트 액세스 키 정보를 입력하세요. 문자형입니다.
- `pcode` 의 주석(#)을 삭제하세요. 프로젝트 코드를 입력하세요. 숫자형입니다.
- `servers` 의 주석(#)을 삭제하세요. 수집 서버 정보(IP, 포트)를 설정하세요. 프로토콜은 tcp만 지원합니다. 문자 형식으로 "tcp://아이피:포트"를 등록하세요. 배열 형식으로 등록하세요. 콤마(,)로 구분하여 배열 형식으로 등록하세요.

 Telegraf를 다시 시작해야 변경된 설정이 적용됩니다.

지원 환경

와탭 Telegraf 에이전트를 설치하기 전에 지원 환경을 확인하세요.

- 운영체제
 - Linux Red Hat 6.x x64 이상
 - Ubuntu 12.x x64 이상
 - FreeBSD 10.x x64 이상
- Telegraf
 - Telegraf 1.3 이상

① socket_writer output plugin을 지원하는 버전만 가능합니다.

에이전트 설치

와탭 Telegraf 에이전트 이용을 위한 기본 설치 방법을 안내합니다.

Telegraf에서 수집하는 메트릭스를 와탭 수집서버로 전달할 수 있습니다. 수집된 데이터는 whatap의 프로젝트 내부에서 확인 가능합니다. 이 페이지에서는 whatap-telegraf 에이전트를 설치하여 Telegraf에서 전달받은 데이터를 수집 서버로 중계하는 방법을 설명합니다.

기존에 설치된 Telegraf의 `socket_writer output plugin` 을 활성화하여 whatap-telegraf(Local TCP) 에이전트로 데이터를 전달합니다.

와탭 서비스를 사용하기 위해서는 [회원 가입](#) 후 프로젝트를 생성하고 대상 서버에 에이전트를 설치해야 합니다. 회원 가입에 관한 자세한 내용은 [다음 문서](#)를 참조하세요.

❗ Telegraf 에이전트는 사용자가 해당 서비스에 직접 설치하고 설정해야 합니다.

프로젝트 생성하기

에이전트를 설치하기 전에 먼저 프로젝트를 생성하세요.

1. [와탭 모니터링 서비스](#)에 로그인합니다.
2. 왼쪽 사이드 메뉴에서 **전체 프로젝트** > **+ 프로젝트** 버튼을 클릭합니다.
3. **상품 선택** 화면에서 설치할 제품을 선택합니다.
4. 아래 항목을 입력하거나 선택합니다.
 - **프로젝트 이름**: 프로젝트의 이름을 입력합니다.
 - **데이터 서버 지역**: 데이터 서버가 위치한 리전을 선택합니다. 리전은 클라우드 서비스를 제공하는 데이터 센터의 묶음입니다. 선택한 리전에 사용자의 데이터가 저장됩니다.
 - **타임 존**: 알림 및 보고서 생성 시 기준이 되는 시간을 설정합니다.
 - **알림 언어 설정**: 경고 알림 메시지의 언어를 설정합니다. (한글, 영어 지원)
 - **프로젝트 그룹**: 여러 프로젝트를 그룹으로 묶어 관리할 수 있습니다. 소속될 그룹이 있으면 선택하세요.
 - **프로젝트 설명**: 프로젝트에 대한 추가 설명이나 세부 정보를 입력합니다.

5. 모든 설정을 완료하면 **프로젝트 생성하기** 버튼을 클릭합니다.

- ❗ 조직을 선택한 상태에서 프로젝트를 추가할 경우 **조직 하위 그룹**을 필수로 설정해야 합니다.
그룹에 대한 자세한 설명은 [다음 문서](#)를 참고하세요.

프로젝트 액세스 키 확인

프로젝트 액세스 키는 와탭 서비스 활성화를 위한 고유 ID입니다.

설치 안내 섹션에서 **프로젝트 액세스 키 발급받기** 버튼을 클릭하세요. **프로젝트 액세스 키**를 자동으로 발급 받은 후 다음 단계를 진행합니다.

- ✅ 프로젝트를 생성한 다음에는 자동으로 **에이전트 설치** 페이지로 이동합니다. **에이전트 설치** 페이지로 이동하지 않는다면 왼쪽 메뉴에서 **전체 프로젝트**를 선택한 다음 새로 생성한 프로젝트를 선택하세요.

설치 순서와 파일 구성

1. 와탭 리포지토리를 설치하세요.
2. whatap-telegraf 리눅스 패키지(yum, apt-get)를 설치하세요.
3. 프로젝트 액세스 키 및 와탭 서버 정보를 설정하세요.
4. Telegraf의 `socket_writer output plugin` 을 설정하여 데이터를 연동하세요.

Telegraf 에이전트 파일 구성

- ❗
- **whatap_telegraf**
서비스 실행 파일로 Telegraf에서 전달된 정보를 수집 서버로 전송하는 프로그램입니다.
 - **/etc/init.d/whatap-telegraf** (FreeBSD /usr/local/etc/whatap_telegraf)
서비스 스크립트입니다.



- **whatap.conf**

설정 파일입니다. 수집 서버의 주소와 서버의 프로젝트 액세스 키가 입력되는 파일입니다.

- **whatap-telegraf-#.log**

에이전트 로그 파일입니다. (</usr/whatap/telegraf/logs>)

패키지 설치

- [Red Hat/CentOS](#) [Debian/Ubuntu](#) [Amazon Linux](#) [FreeBSD](#)

1. 와탭 리포지토리를 설치하세요.

```
$ sudo rpm -Uvh https://repo.whatap.io/centos/5/noarch/whatap-repo-1.0-1.noarch.rpm
```

2. 다음 명령어를 통해 패키지를 설치하세요.

```
$ sudo yum install whatap-telegraf
```

3. 설정 스크립트를 실행하여 서비스를 시작합니다.

```
echo "license=[발급받은 프로젝트 액세스 키]" | sudo tee /usr/whatap/telegraf/whatap.conf
echo "whatap.server.host=[와탭 서버 주소]" | sudo tee -a /usr/whatap/telegraf/whatap.conf
sudo service whatap-telegraf restart
```

1. 와탭 리포지토리를 설치하세요.

```
$ wget https://repo.whatap.io/debian/release.gpg -O - | sudo apt-key add -
$ wget https://repo.whatap.io/debian/whatap-repo_1.0_all.deb
$ sudo dpkg -i whatap-repo_1.0_all.deb
$ sudo apt-get update
```

2. 다음 명령어를 통해 패키지를 설치하세요.

```
$ sudo apt-get install whatap-telegraf
```

3. 설정 스크립트를 실행하여 서비스를 시작합니다.

```
echo "license=[발급받은 프로젝트 액세스 키]" | sudo tee /usr/whatap/telegraf/whatap.conf
echo "whatap.server.host=[와탭 서버 주소]" | sudo tee -a /usr/whatap/telegraf/whatap.conf
sudo service whatap-telegraf restart
```

1. 와탭 리포지토리를 설치하세요.

```
$ sudo rpm --import https://repo.whatap.io/centos/release.gpg
$ echo "[whatap]" | sudo tee /etc/yum.repos.d/whatap.repo > /dev/null
$ echo "name=whatap packages for enterprise linux" | sudo tee -a /etc/yum.repos.d/whatap.repo > /dev/null
$ echo "baseurl=https://repo.whatap.io/centos/latest/\$basearch" | sudo tee -a /etc/yum.repos.d/whatap.repo > /dev/null
$ echo "enabled=1" | sudo tee -a /etc/yum.repos.d/whatap.repo > /dev/null
$ echo "gpgcheck=0" | sudo tee -a /etc/yum.repos.d/whatap.repo > /dev/null
```

2. 다음 명령어를 통해 패키지를 설치하세요.

```
$ sudo yum install whatap-telegraf
```

3. 설정 스크립트를 실행하여 서비스를 시작합니다.

```
echo "license=[발급받은 프로젝트 액세스 키]" | sudo tee /usr/whatap/telegraf/whatap.conf
echo "whatap.server.host=[와탭 서버 주소]" | sudo tee -a /usr/whatap/telegraf/whatap.conf
sudo service whatap-telegraf restart
```

1. 와탭 리포지토리를 설치하세요.

```
$ wget https://s3.ap-northeast-2.amazonaws.com/repo.whatap.io/freebsd/10/whatap-telegraf-0.0.4.txz
$ pkg install whatap-telegraf-0.0.4.txz
```

2. 설정 스크립트를 실행하여 서비스를 시작합니다.

```
echo "license=[발급받은 프로젝트 액세스 키]" | tee /usr/whatap/telegraf/whatap.conf
echo "whatap.server.host=[와탭 서버 주소]" | tee -a /usr/whatap/telegraf/whatap.conf
sudo service whatap_telegraf restart
```

Telegraf 연동

Telegraf.conf에 `socket_writer output plugin` 설정을 하여 whatap-telegraf로 수집된 정보를 전달합니다.

TOML

```
# # Generic socket writer capable of handling multiple socket types.
[[outputs.socket_writer]]

# ## URL to connect to
address = "tcp://127.0.0.1:6600"

# # data_format = "influx"
data_format = "json"
```

- `[[outputs.socket_writer]]` 주석을 해제합니다.
- address 항목에 tcp 연결을 설정합니다.
- 데이터 유형은 json 형식으로 설정합니다.

Telegraf 에이전트 설치를 완료했습니다. [다음 문서](#)에서 설치 문제 해결 방법을 확인하세요.

설치 문제 해결

whatap-telegraf 서비스(Service)가 실행 중이지 않거나 오류가 발생한 경우 다시 시작합니다.

⚠️ Telegraf 플러그인 및 에이전트는 사용자가 해당 서비스에 직접 설치하고 설정해야 합니다.

- Red Hat/CentOS

```
$ service whatap-telegraf restart
```

- Debian/Ubuntu

```
$ sudo service whatap-telegraf restart
```

- FreeBSD

```
$ service whatap_telegraf restart
```

설정하기

와탭 에이전트는 에이전트 별 필요한 설정을 [whatap.conf](#) 파일에 작성합니다. 에이전트는 환경변수를 통해 설정 파일의 위치를 파악하고 로딩합니다. [whatap.conf](#) 파일을 통해 설정할 수 있는 옵션은 다음과 같습니다.

ⓘ Telegraf 에이전트는 사용자가 직접 설정해야 합니다.

에이전트 이름 식별

와탭은 모니터링 정보 수집 대상인 인프라 서버 식별을 위해 기본적으로 서버로부터 수집한 정보를 활용합니다.

- ⓘ • 에이전트 이름은 프로젝트 단위로 고유해야 합니다.
- 에이전트 아이디/이름을 변경하면 이전 데이터와 연결되지 않습니다.

- **object_name** String

기본값 {type} - {hostname} - {ip2} - {ip3} - {docker}

애플리케이션을 식별하기 위한 에이전트 이름(ONAME) 구성 방식입니다. ONAME을 토대로 OID가 생성됩니다.

명칭	설명
{type}	app_name에 설정된 값을 사용합니다. 기본값은 Telegraf입니다.
{ip#}	IP를 나누었을 때 #번째 자리를 사용합니다.
{hostname}	서버 호스트명을 사용합니다.
{docker}	도커 컨테이너 아이디를 사용합니다.

- **app_name** String

애플리케이션을 식별하기 위한 에이전트 이름(ONAME)의 구성 요소입니다. object_name 의 {type} 에 해당하는

값입니다.

① whatap-telegraf 서비스를 다시 시작 후 적용됩니다.

에이전트 통신 설정

- **license** String

에이전트를 설치할 때 서버로부터 부여받은 프로젝트 액세스 키를 설정합니다. 프로젝트 액세스 키에는 에이전트가 속한 프로젝트와 보안 통신을 위한 암호 키를 포함하고 있습니다.

- **whatap.server.host** ip_address

기본값 127.0.0.1,127.0.0.1

에이전트가 수집한 데이터를 전송할 서버를 지정합니다. 수집 서버 이중화로 2개 이상의 IP를 가진 경우 콤마(,)로 분리하여 지정할 수 있습니다. 지정된 IP에는 수집 서버 proxy 데몬이 리스닝 상태로 서비스되어야 합니다.

- **whatap.server.port** tcp_port

기본값 6600

수집 서버 PORT를 지정합니다. 포트는 하나만 지정할 수 있으므로 whatap_server_host 에 지정된 수집 서버들은 동일 PORT를 사용해야 합니다.

- **tcp_so_timeout** Millisecond

기본값 60000

수집 서버와 통신하는 TCP 세션의 Socket Timeout 값을 지정합니다.

- **tcp_connection_timeout** Millisecond

기본값 5000

수집 서버와 통신하는 TCP 세션의 Connection Timeout 값을 지정합니다.

- **net_send_max_bytes** Byte

기본값 5242880

수집 서버로 데이터를 전송할 때 한번에 전송되는 최대 크기를 지정합니다.

- **net_send_buffer_size** Byte

기본값 1024

데이터 전송을 하기 위해 가지고 있는 최대 바이트 크기입니다.

Telegraf

- **telegraf_tcp_port** tcp_port

기본값 6600

Telegraf socker_writer output plugin이 연결할 TCP Port입니다.

① whatap-telegraf 서비스를 다시 시작 후 적용됩니다.

- **telegraf_delta_fields** String

기본값 1024

누적값을 가지는 필드의 증가 값을 계산하여 별도의 필드를 추가합니다. name.fields 형식으로 지정하고, 여러 개는 콤마(,)로 구분합니다.

예, nginx.accepts, nginx.requests nginx의 accepts, requests 값의 증가 값을 acceptsDelta, requestsDelta 필드 이름으로 추가 전달합니다.

로그

- **log_keep_days** Day

기본값 7

로그 파일 보관 기간을 설정 합니다.

관리하기

Telegraf 에이전트 관리를 위해 whatap.conf 파일을 통해 설정할 수 있는 추가 옵션에 대해 알아봅니다.

업데이트

- [Red Hat/CentOS](#) [Debian/Ubuntu](#) [FreeBSD](#)

1. 패키지 정보 갱신을 위해 캐시 정보를 삭제하세요.

```
$ yum clean all
```

2. whatap-telegraf 패키지를 업데이트 하세요.

```
$ yum update whatap-telegraf
```

3. 특정 버전을 설치하기 위해서는 설치 시 버전 정보를 명시해 줍니다.

```
$ sudo yum install whatap-telegraf-{version}
```

1. 패키지 정보 갱신을 위해 캐시 정보를 삭제하세요.

```
$ sudo apt-get update
```

2. whatap-telegraf 패키지를 업데이트 하세요.

```
$ sudo apt-get install --only-upgrade whatap-telegraf
```

3. 특정 버전을 설치하기 위해서는 설치 시 버전 정보를 명시해 줍니다.

```
$ sudo apt-get install whatap-telegraf={version}
```

1. 패키지를 다시 설치하세요.

SH

```
# wget https://s3.ap-northeast-2.amazonaws.com/repo.whatap.io/freebsd/10/whatap-telegraf-0.0.4.txz
# pkg install whatap-telegraf-0.0.4.txz
```

2. 특정 버전을 설치하기 위해서는 `{version}` 에 버전 정보를 입력해 설치하세요.

SH

```
# wget https://s3.ap-northeast-2.amazonaws.com/repo.whatap.io/freebsd/10/whatap-telegraf-{version}.txz
# pkg install whatap-telegraf-{version}.txz
```

일시 중지

- [Red Hat/CentOS](#) [Debian/Ubuntu](#) [FreeBSD](#)

SH

```
$ sudo service whatap-telegraf stop
```

SH

```
$ sudo service whatap-telegraf stop
```

SH

```
$ service whatap_telegraf stop
```

삭제

- 패키지(yum, apt-get)를 삭제하세요.
- [Red Hat/CentOS](#) [Debian/Ubuntu](#) [FreeBSD](#)

SH

```
$ sudo yum remove whatap-telegraf
```

SH

```
$ sudo apt-get purge whatap-telegraf
```

SH

```
$ pkg delete whatap-telegraf
```

- 필요한 경우 `/usr/whatap/telegraf` 하위의 로그 파일 및 기타 파일을 삭제하세요.

사용 예시

Telegraf에 관한 자세한 내용은 influxdata의 [Telegraf documentation](#)을 참조하세요.

❗ Telegraf 플러그인 및 에이전트는 사용자가 해당 서비스에 직접 설치하고 설정해야 합니다.

SNMP

SNMP input plugin을 사용하여 SNMP 성능 데이터를 실시간으로 수집 및 차트 표시 가능합니다. 수집된 데이터는 `telegraf_snmp`, `telegraf_interface` 카테고리로 검색하실 수 있습니다.

Linux Shell

```
cat >/etc/telegraf/telegraf.d/snmp_device.conf <<EOL
[[inputs.snmp]]
  ## Agent addresses to retrieve values from.
  ## format: agents = ["<scheme://><hostname>:<port>"]
  ## scheme: optional, either udp, udp4, udp6, tcp, tcp4, tcp6.
  ##      default is udp
  ## port: optional
  ## example: agents = ["udp://127.0.0.1:161"]
  ##      agents = ["tcp://127.0.0.1:161"]
  ##      agents = ["udp4://v4only-snmp-agent"]

  agents = ["udp://1.1.1.1:161"]

  ## Timeout for each request.
  # timeout = "5s"

  ## SNMP version; can be 1, 2, or 3.
  version = 2

  ## SNMP community string.
  community = "xxx@xxxx"
```

```

## Agent host tag
# agent_host_tag = "agent_host"

## Number of retries to attempt.
# retries = 3

## The GETBULK max-repetitions parameter.
# max_repetitions = 10

## SNMPv3 authentication and encryption options.
##
## Security Name.
# sec_name = "myuser"
## Authentication protocol; one of "MD5", "SHA", "SHA224", "SHA256", "SHA384", "SHA512" or "".
# auth_protocol = "MD5"
## Authentication password.
# auth_password = "pass"
## Security Level; one of "noAuthNoPriv", "authNoPriv", or "authPriv".
# sec_level = "authNoPriv"
## Context Name.
# context_name = ""
## Privacy protocol used for encrypted messages; one of "DES", "AES", "AES192", "AES192C", "AES256", "AES256C", or "".
### Protocols "AES192", "AES192C", "AES256", and "AES256C" require the underlying net-snmp tools
### to be compiled with --enable-blumenthal-aes (http://www.net-snmp.org/docs/INSTALL.html)
# priv_protocol = ""
## Privacy password used for encrypted messages.
# priv_password = ""

## Add fields and tables defining the variables you wish to collect. This
## example collects the system uptime and interface variables. Reference the
## full plugin documentation for configuration details.
[[inputs.snmp.field]]
    oid = "RFC1213-MIB::sysUpTime.0"
    name = "uptime"

[[inputs.snmp.field]]
    oid = "RFC1213-MIB::sysName.0"
    name = "source"
    is_tag = true

[[inputs.snmp.table]]

```

```

oid = "IF-MIB::ifXTable"
name = "interface"
inherit_tags = ["source"]

[[inputs.snmp.table.field]]
oid = "IF-MIB::ifDescr"
name = "ifDescr"
is_tag = true

[[aggregators.derivative]]
period = "60s"
max_roll_over = 1

fieldpass = ["*Octets", "*Pkts"]
drop_original = false

[aggregators.derivative.tags]
aggr = "derivative"

[[processors.starlark]]
source = ""
def apply(metric):
    for (k, v) in metric.fields.items():
        if k.endswith('Octets_rate'):
            metric.fields[k] *= 8

    return metric

'''

EOL
service telegraf restart

```

SNMP Trap

SNMP Trap input plugin을 사용하여 SNMP Trap 이벤트를 실시간으로 수집 및 이메일, 문자 메시지 및 메신저로 전파 가능합니다. 수집된 데이터는 `telegraf_snmp_trap` 카테고리로 검색하실 수 있습니다.

Linux Shell

```
cat >/etc/telegraf/telegraf.d/snmp_trap.conf <<EOL
[[inputs.snmp_trap]]
    service_address = "udp://:162"
EOL
service telegraf restart
```

OpenTelemetry 소개

⚠ OpenTelemetry 모니터링은 베타 버전으로 예기치 않은 오류가 발생할 수 있습니다. 중요한 데이터나 운영 환경에서는 사용하는 것을 권장하지 않습니다. 피드백이나 문제점이 발생하면 지원 팀(support@whatap.io)으로 문의하세요.

이 문서는 **OpenTelemetry Collector**로 수집한 데이터를 **와탭 모니터링 서비스**에 연동해 통합 옵저버빌리티(Observability) 환경을 구성하려는 사용자를 위한 가이드입니다.

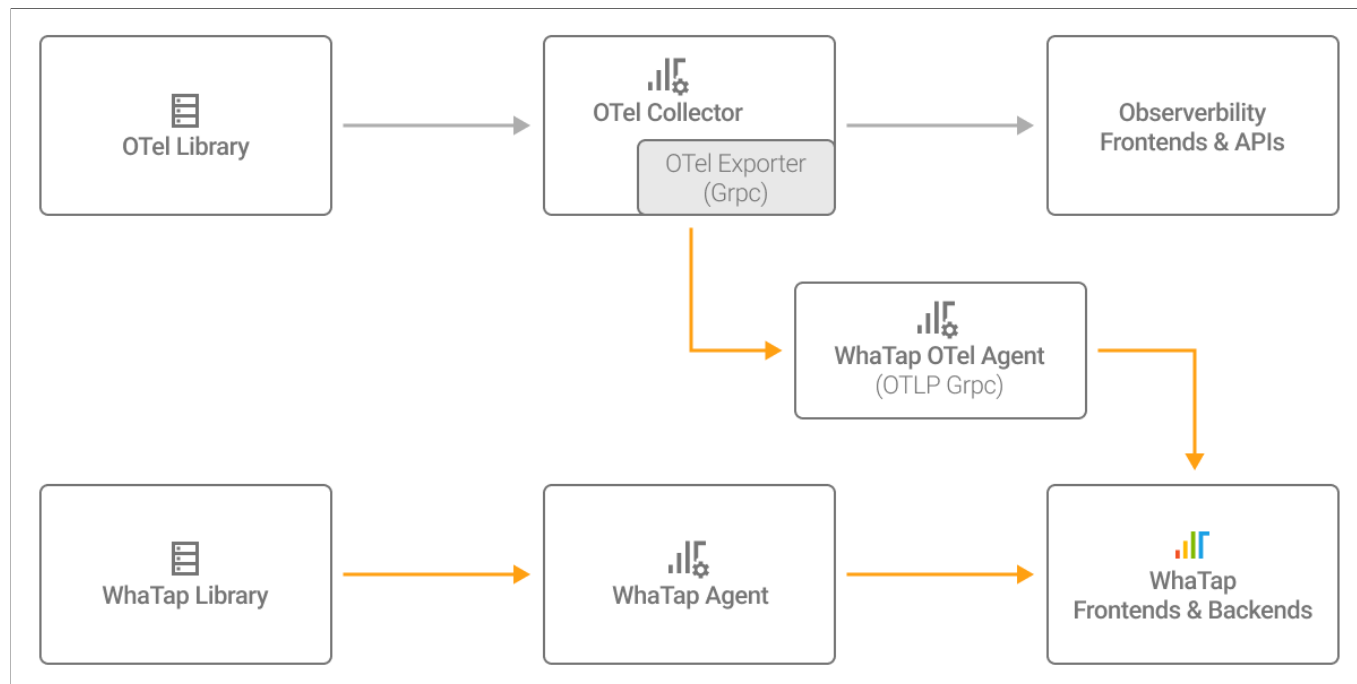
OpenTelemetry 통해 와탭 모니터링 사용하기

OpenTelemetry는 원격 측정 데이터(메트릭, 로그 및 추적)를 계측 및 생성, 수집, 내보내기 위한 오픈 소스 옵저버빌리티(Observability) 프레임워크입니다. OpenTelemetry를 통해 수집한 측정 데이터를 와탭 모니터링 플랫폼으로 전송해 애플리케이션이나 서비스의 성능과 동작을 분석하고, 문제를 해결할 수 있습니다.

만약 운영 중인 애플리케이션 또는 서비스가 OpenTelemetry 라이브러리로 수집되는 경우라면 추적 및 지표, 로그 데이터를 와탭 모니터링 플랫폼으로 가져오는 방법을 선택할 수 있습니다. 와탭 모니터링 플랫폼의 대시보드 및 분석 메뉴를 활용해 가시성 높은 옵저버빌리티 모니터링을 제공합니다.

OpenTelemetry 에이전트

OpenTelemetry 에이전트는 OpenTelemetry Collector의 OTLP Exporter를 통해 수집된 데이터를 와탭 모니터링 플랫폼으로 가공 및 전달하는 에이전트입니다.



에이전트 설치 및 실행을 위한 과정은 다음을 참조하세요.

1. OpenTelemetry Collector가 설치된 인스턴스에 [와탭 에이전트를 설치하고 실행하세요.](#)
2. 실행된 whatap-otel 서비스는 OTLP Exporter를 위한 Grpc 서버를 실행합니다.
 - 리스닝 포트는 기본 6600 포트입니다
 - OpenTelemetry Collector의 기본 설정과 동일하게 gzip 압축 설정은 `True` 입니다.

지원 환경

OpenTelemetry 에이전트를 설치할 수 있는 운영체제는 다음과 같습니다.

- CentOS/Red Hat 6.x 버전(64bit) 이상
- Debian/Ubuntu 12.04 버전(64bit) 이상
- Amazon linux (64bit)
- Alpine linux (64bit)

에이전트 설치

⚠ OpenTelemetry 모니터링은 베타 버전으로 예기치 않은 오류가 발생할 수 있습니다. 중요한 데이터나 운영 환경에서는 사용하는 것을 권장하지 않습니다. 피드백이나 문제점이 발생하면 지원 팀(support@whatap.io)으로 문의하세요.

OpenTelemetry 에이전트 설치 가이드는 OpenTelemetry Collector와 연동하여 애플리케이션, 컨테이너, 클라우드 환경의 모니터링 데이터를 수집하고 전송하는 방법을 설명합니다.

이 문서에서는 **OpenTelemetry 에이전트 최신 버전 다운로드**, **리눅스 환경 설치**, **Docker 및 Kubernetes 배포 설정**, 그리고 **와탭 모니터링 서비스 연동 절차**를 단계별로 안내합니다.

에이전트 다운로드

다음 명령어를 실행해 OpenTelemetry Collector가 설치된 인스턴스에 에이전트를 다운로드하세요.

- [Red Hat/Centos](#) [Debian/Ubuntu](#) [Amazon Linux](#) [Alpine Linux](#) [Alpine Linux Docker](#)

```
# Register package repository
```

```
$ sudo rpm -Uvh https://repo.whatap.io/centos/5/noarch/whatap-repo-1.0-1.noarch.rpm
```

```
# Install package
```

```
$ sudo yum install whatap-otel
```

```
# Register package repository
```

```
$ wget https://repo.whatap.io/debian/release.gpg -O - | sudo apt-key add -
```

```
$ wget https://repo.whatap.io/debian/whatap-repo_1.0_all.deb
```

```
$ sudo dpkg -i whatap-repo_1.0_all.deb
```

```
$ sudo apt-get update
```

```
# Install package
```

```
$ sudo apt-get install whatap-otel
```

```
# Register package repository
```

```
$ sudo rpm --import https://repo.whatap.io/centos/release.gpg
```

```
$ echo "[whatap]" | sudo tee /etc/yum.repos.d/whatap.repo > /dev/null
```

```
$ echo "name=whatap packages for enterprise linux" | sudo tee -a /etc/yum.repos.d/whatap.repo > /dev/null
$ echo "baseurl=https://repo.whatap.io/centos/latest/\$basearch" | sudo tee -a /etc/yum.repos.d/whatap.repo > /dev/null
$ echo "enabled=1" | sudo tee -a /etc/yum.repos.d/whatap.repo > /dev/null
$ echo "gpgcheck=0" | sudo tee -a /etc/yum.repos.d/whatap.repo > /dev/null

# Install package
$ sudo yum install whatap-otel
```

```
$ wget https://s3.ap-northeast-2.amazonaws.com/repo.whatap.io/alpine/x86_64/whatap-otel.tar.gz
$ tar -xvzf whatap-otel.tar.gz -C /
```

FROM alpine

FROM alpine

RUN apk update && apk upgrade

Install WhatAp Go monitoring agent

RUN wget https://s3.ap-northeast-2.amazonaws.com/repo.whatap.io/alpine/x86_64/whatap-otel.tar.gz

RUN tar -xvzf whatap-otel.tar.gz -C /

에이전트 기본 설정

에이전트 설정을 위해서는 와탭 프로젝트의 액세스 키와 수집 서버 IP 주소가 필요합니다. [와탭 모니터링 서비스](#)에서 애플리케이션(Application) 제품의 프로젝트로 이동하세요. **관리 > 에이전트 설치** 메뉴에서 액세스 키와 수집 서버 IP 주소를 확인하고 준비하세요.

whatap.conf 파일 설정하기

에이전트 설정([whatap.conf](#)) 파일에 액세스 키와 수집 서버 IP 주소를 설정하세요.

```
echo "accesskey={ACCESS_KEY}" >> /usr/whatap/otel/whatap.conf
echo "whatap.server.host={COLLECTOR_SERVER_IP_ADDRESS}" >> /usr/whatap/otel/whatap.conf
```

환경 변수로 설정하기

환경 변수(Environment)로 액세스 키와 수집 서버 IP 주소를 설정할 수 있습니다.

```
WHATAP_ACCESSKEY={ACCESS_KEY}
WHATAP_SERVER_HOST={COLLECTOR_SERVER_IP_ADDRESS}
WHATAP_OTEL_GRPC_SERVER_PORT={gRPC_SERVER_LISTENING_PORT}
```

❗ 환경 변수와 [whatap.conf](#) 파일을 모두 설정한 경우, [whatap.conf](#) 파일의 설정값이 우선합니다.

Docker 설치

Opentelemetry Collector가 설치되어 있다면, Docker container로 whatap-otel을 실행하여 Collector의 데이터를 GRPC 6600 포트로 전달받을 수 있습니다.

```
docker pull whatap/whatap-otel
```

docker-compose.yaml

```
version: '3.7'

services:
  whatap_otel:
    image: whatap/whatap-otel:latest

    environment:
      WHATAP_ACCESSKEY: x60511111111-z11111111-x1111111111m
      WHATAP_SERVER_HOST: 1.1.1.1/2.2.2.2
      WHATAP_OTEL_GRPC_SERVER_PORT: 6600
      otel_metric_server_enabled: true
      otel_log_server_enabled: true
      log_stdout_enabled: true
      #debug_otel_grpc_server_enabled: true

    ports:
      - 6600:6600
```

Kubernetes 설치

Opentelemetry Collector와 whatap-otel을 하나의 POD 안에 같이 생성하는 deployment 예제입니다.

otelcol-whatap-deploy.yaml

```

metadata:
  name: otel-collector-deploy
spec:
  replicas: 1
  selector:
    matchLabels:
      app: otel-collector
  template:
    metadata:
      name: otel-collector-pod
      labels:
        app: otel-collector
    spec:
      containers:
        - name: otel-collector
          image: otel/opentelemetry-collector-contrib:0.106.0
          ports:
            - containerPort: 4317 # OTLP gRPC
            - containerPort: 4318 # OTLP HTTP
          volumeMounts:
            - name: config-volume-otelcol
              mountPath: /etc/otel/config.yaml
              subPath: otel-collector-config.yaml
          args: ["--config", "/etc/otel/config.yaml"]
        - name: whatap-otel-agent
          image: whatap/whatap-otel:latest
          ports:
            - containerPort: 6600
          env:
            - name: "WHATAP_ACCESSKEY"
              value: "x60511111111-z11111111-x1111111111m"
            - name: "WHATAP_SERVER_HOST"
              value: "1.1.1.1/2.2.2.2"

```

```

- name: "otel_metric_server_enabled"
  value: "true"
- name: "otel_log_server_enabled"
  value: "true"
- name: "log_stdout_enabled"
  value: "true"
- name: NODE_IP
  valueFrom: {fieldRef: {fieldPath: status.hostIP}}
- name: NODE_NAME
  valueFrom: {fieldRef: {fieldPath: spec.nodeName}}
- name: POD_NAME
  valueFrom: {fieldRef: {fieldPath: metadata.name}}
- name: whatap.micro.enabled
  value: "true"
volumes:
- name: config-volume-otelcol
  configMap:
    name: otelcol-config

```

otelcol-whatap-deploy.yaml

```

apiVersion: v1
kind: ConfigMap
metadata:
  name: otelcol-config
data:
  otel-collector-config.yaml: |
    receivers:
      otel:
        protocols:
          grpc:
            endpoint: 0.0.0.0:4317
          http:
            endpoint: 0.0.0.0:4318
    exporters:
      otel:
        endpoint: "127.0.0.1:6600"
      tls:
        insecure: true
    logging:

```

```

    loglevel: debug
  processors:
    batch:
  service:
  pipelines:
    traces:
      receivers: [otlp]
      processors: [batch]
      exporters: [otlp, logging]
    metrics:
      receivers: [otlp]
      processors: [batch]
      exporters: [otlp, logging]
    logs:
      receivers: [otlp]
      processors: [batch]
      exporters: [otlp, logging]

```

Opentelemetry Collector 설정

whatap-otel로 데이터를 전달하도록 OTLP Exporter를 설정하세요.

otel-collector-config.yaml

```

1  receivers:
2    otlp:
3      protocols:
4        grpc:
5          endpoint: 0.0.0.0:4317
6  processors:
7    batch:
8
9  exporters:
10   logging:
11   # otlp exporter에 whatap-otel을 대상으로 설정합니다.
12   otlp:
13     endpoint: 127.0.0.1:6600
14     tls:
15       insecure: true

```

```

16
17 service:
18 pipelines:
19 traces:
20 receivers:
21 - otlp
22 processors:
23 - batch
24 exporters:
25 - logging
26 - otlp
27 telemetry:
28 logs:
29 level: debug #activate debug mode

```

- 엔드포인트(endpoint) 설정: 127.0.0.1:6600
로컬에 설치한 에이전트가 사용하는 기본 포트는 6600입니다.
- tls.secure 설정: insecure: true
- gzip 압축 설정을 기본 사용합니다. (gzip 사용)

❗ OpenTelemetry Collector 설치와 관련한 자세한 내용은 다음 링크를 참조하세요.

- [Install the Collector](#)
- [Building a custom collector](#)

기타 플랫폼

Tyk Gateway

[Tyk Gateway](#)의 OpenTelemetry 기능을 활성화한 다음 OpenTelemetry Collector를 통해서 트랜잭션 정보를 확인할 수 있습니다.

1. OpenTelemetry Collector를 Tyk Gateway에 연결합니다.

tyk.conf

```
{
  "log_level": "debug",
  "listen_port": 8081,

  ...

  "opentelemetry": {
    "enabled": true,
    "endpoint": "111.222.333.444:4317",
  }
}
```

1. 연결된 OpenTelemetry Collector에 whatap-otel 에이전트가 연결되면 트레이스 정보를 수집합니다.

Kong Gateway

Kong Gateway의 내장 OpenTelemetry 플러그인을 설정하면 OpenTelemetry Collector를 통해 트랜잭션 정보를 확인할 수 있습니다.

OpenTelemetry 플러그인에서 OpenTelemetry Collector의 endpoint를 지정합니다. Collector에 Whatap-otel 에이전트가 연결되면 trace 정보가 수집됩니다.

1. kong.conf

trace 설정을 활성화 합니다.

```
tracing_instrumentations = all
tracing_sampling_rate = 1
```

2. OpenTelemetry 플러그인 설정

OpenTelemetry Collector의 endpoint URL을 등록합니다.

Collector의 OTLP receiver 기본 설정은 아래와 같습니다.

- Protocol : http
- Port : 4318

- URL :
 - Trace : /v1/traces
 - <http://x.x.x.x:4318/v1/traces>
 - Log : /v1/logs
 - <http://x.x.x.x:4318/v1/logs>

- 기본 설정 예시

```
traces_endpoint: http://otelcollector:4318/v1/traces
logs_endpoint: http://otelcollector:4318/v1/logs
```

- kong.yaml 설정 예시

```
plugins:
- name: opentelemetry
  config:
    traces_endpoint: http://otelcollector:4318/v1/traces
    logs_endpoint: http://otelcollector:4318/v1/logs
    resource_attributes:
      service.name: kong-otel
```

- Admin API로 플러그인 설정 예시

```
curl -X POST http://localhost:8001/services/plugins \
--header "accept: application/json" \
--header "Content-Type: application/json" \
--data '
{
  "name": "opentelemetry",
  "config": {
    "traces_endpoint": "http://otelcollector:4318/v1/traces",
    "logs_endpoint": "http://otelcollector:4318/v1/logs",
    "resource_attributes": {
      "service.name": "kong-dev"
    }
  }
}
```

```
}
}
'
```

- 지원 버전: Kong Gateway 3.3.x 이상

ⓘ 설정에 대한 자세한 내용은 다음 링크를 참조하세요.

- [Set up Kong Gateway](#)
- [OpenTelemetry Configuration](#)

모니터링 시작하기

- 모든 설정을 완료한 다음 애플리케이션 서버를 다시 시작하면 에이전트가 정보를 수집하기 시작합니다. **대시보드 > 애플리케이션 대시보드** 메뉴로 이동해 수집된 데이터를 실시간으로 모니터링하세요.
- 에이전트 설정에 대한 자세한 내용은 [다음 문서](#)를 확인하세요.

에이전트 설정

❗ OpenTelemetry 모니터링은 베타 버전으로 예기치 않은 오류가 발생할 수 있습니다. 중요한 데이터나 운영 환경에서는 사용하는 것을 권장하지 않습니다. 피드백이나 문제점이 발생하면 지원 팀(support@whatap.io)으로 문의하세요.

이 문서는 **OpenTelemetry 에이전트 설정 방법**을 안내합니다.

에이전트를 구성하면 **OpenTelemetry Collector**로부터 전달된 **Trace, Metric, Log** 데이터를 와탭 모니터링 플랫폼으로 수집하고 분석할 수 있습니다.

에이전트 설정 파일([whatap.conf](https://whatap.io/whatap.conf)) 또는 환경 변수를 통해

Grpc 서버 포트, 데이터 처리 옵션, 속성(attribute) 매핑 설정을 세부적으로 제어할 수 있습니다.

이 가이드는 OpenTelemetry 기반의 통합 오피저빌리티 환경을 구축하려는 사용자를 위한 구성 문서입니다.

에이전트

- **otel_grpc_server_enabled** Boolean

기본값 `true`

OpenTelemetry protocol(OTLP)를 통해 전달되는 데이터를 수신하기 위한 Grpc 서버를 활성화합니다.

- **otel_grpc_server_port** String

기본값 `6600`

Grpc 서버의 Listen 포트입니다.

❗ 포트 번호를 변경 후에는 whatap-otel 서비스를 재시작해야 합니다.

- **otel_grpc_server_gzip_enabled** Boolean

기본값 `true`

Grpc 서버에서 읽는 데이터의 암호화로 `gzip` 을 활성화합니다.

- **debug_otel_grpc_server_enabled** Boolean

기본값 `false`

OpenTelemetry Trace 데이터 처리에 관련된 디버그 로그의 출력을 활성화합니다.

에이전트 로그

- `log_keep_days` Day `Int`

기본값 `7`

로그 파일 보관 기간을 설정합니다.

- `log_stdout_enabled` `Boolean`

기본값 `false`

파일 로그와 동일하게 Stdout으로 출력하는 옵션을 추가합니다.

Trace

OpenTelemetry Trace 데이터 처리와 관련한 에이전트 설정 옵션입니다. 와탭은 OpenTelemetry Trace 구성 요소인 **Span**의 `attributes` 객체로부터 값을 가져와 모니터링 정보로 활용합니다.

Span의 `attributes` 객체가 와탭의 에이전트 설정과 어떻게 연관되는지 다음 코드를 참조하세요.

OpenTelemetry - Span example

```
1 {
2   "name": "/v1/sys/health",
3   "context": {
4     "trace_id": "7bba9f33312b3dbb8b2c2c62bb7abe2d",
5     "span_id": "086e83747d0e381e"
6   },
7   "parent_id": "",
8   "start_time": "2021-10-22 16:04:01.209458162 +0000 UTC",
9   "end_time": "2021-10-22 16:04:01.209514132 +0000 UTC",
10  "status_code": "STATUS_CODE_OK",
11  "status_message": "",
12  "attributes": {
13    "net.transport": "IP.TCP",
14    "net.peer.ip": "172.17.0.1",
```

```

15     "net.peer.port": "51820",
16     "net.host.ip": "10.177.2.152",
17     "net.host.port": "26040",
18     "http.method": "GET",
19     "http.target": "/v1/sys/health",
20     "http.server_name": "mortar-gateway",
21     "http.route": "/v1/sys/health",
22     "http.user_agent": "Consul Health Check",
23     "http.scheme": "http",
24     "http.host": "10.177.2.152:26040",
25     "http.flavor": "1.1"
26 },
27 "events": [
28   {
29     "name": "",
30     "message": "OK",
31     "timestamp": "2021-10-22 16:04:01.209512872 +0000 UTC"
32   }
33 ]
34 }

```

whatap.conf

```

otel_attr_tx_host_keys=http.host
otel_attr_tx_method_keys=http.method
otel_attr_tx_useragent_keys=http.user_agent

```

- ❗ • 와탭의 트레이스 분석과 관련한 자세한 내용은 [다음 문서](#)를 참조하세요.
- OpenTelemetry의 Trace 데이터 처리에 대한 자세한 내용은 [다음 링크](#)를 참조하세요.
- OpenTelemetry에서 제공하는 속성(`attributes`)에 대한 자세한 내용은 다음 링크를 참조하세요.
 - [Database](#)
 - [Error](#)
 - [Host](#)
 - [HTTP](#)
 - [Network](#)



- URL
- User agent

- **otel_trace_server_enabled** Boolean

기본값 `true`

OpenTelemetry Trace 데이터 처리를 활성화합니다.

- **otel_trace_queue_size** Int

기본값 `5000`

OpenTelemetry Trace 데이터 처리를 위한 큐 크기를 설정합니다.

- **otel_span_collector_max** Int

기본값 `100000`

OpenTelemetry Trace 데이터의 Span 데이터 버퍼의 최대 크기(개수)를 지정합니다.

- **otel_attr_dbcon_keys** String

기본값 `sql.system,db.connection_string,db.system`

DB Connection 정보로 활용할 속성 이름을 입력하세요. 해당 정보를 활용해 와탭에서는 **DB Connection** 스텝으로 표시합니다.

- **otel_attr_sql_keys** String

기본값 `db.statement,sql.query`

SQL 쿼리 정보로 활용할 속성 이름을 입력하세요. 해당 정보를 활용해 와탭에서는 **SQL** 스텝으로 표시합니다.

- **otel_attr_httpc_keys** String

기본값 `url.full,http.url,http.target`

HTTP call 정보로 활용할 속성 이름을 입력하세요. 해당 정보를 활용해 와탭에서는 **HTTP Call** 스텝으로 표시합니다.

- **otel_attr_tx_url_keys** String

기본값 `http.url,http.route,http.target,tyk.api.path,http.url`

트랜잭션의 서비스 URL 정보로 활용할 속성 이름을 입력하세요.

- **otel_attr_tx_host_keys** String

기본값 `http.host`

트랜잭션의 호스트 정보로 활용할 속성 이름을 입력하세요.

- **otel_attr_tx_method_keys** *String*

기본값 `http.method,http.request.method`

HTTP Method 정보로 활용할 속성 이름을 입력하세요.

- **otel_attr_tx_status_keys** *String*

기본값 `http.status_code`

트랜잭션의 HTTP 상태 코드 정보로 활용할 속성 이름을 입력하세요.

- **otel_attr_tx_useragent_keys** *String*

기본값 `http.user_agent`

트랜잭션의 User-Agent 정보로 활용할 속성 이름을 입력하세요.

- **otel_attr_tx_clientip_keys** *String*

기본값 `net.peer.ip,net.socket.peer.addr`

트랜잭션의 클라이언트 IP 정보로 활용할 속성 이름을 입력하세요.

- **otel_attr_tx_referer_keys** *String*

트랜잭션의 HTTP Referer 정보로 활용할 속성 이름을 입력하세요.

- **otel_attr_error_keys** *String*

기본값 `error.type`

트랜잭션의 에러 정보로 활용할 속성 이름을 입력하세요.

- **otel_events_exception_keys** *String*

기본값 `exception.type`

Event 데이터의 속성(Attribute) 이름을 지정하여 오류 타입 정보로 활용합니다.

- **otel_events_error_message_keys** *String*

기본값 `exception.message`

Event 데이터의 속성(Attribute) 이름을 지정하여 오류 메시지 정보로 활용합니다.

- **otel_events_error_stack_keys** *String*

기본값 `exception.stacktrace`

Event 데이터의 속성(Attribute) 이름을 지정하여 오류 스택 정보로 활용합니다.

- **otel_tx_path_setby_name_enabled** `Boolean`

기본값 `false`

트랜잭션 이름을 Span의 이름으로 지정합니다.

- **otel_mtrace_stepid_setby_trace_spanid_enabled** `Boolean`

기본값 `false`

Trace Span의 SpanID를 외부 연결의 ParentSpanID로 인식하게 설정합니다. HTTP call span 정보가 없을 때 강제로 설정합니다.

- **otel_virtual_httpc_enabled** `Boolean`

기본값 `false`

분산 추적을 위해서 HTTP call span 정보가 없을 때 강제로 설정합니다. HTTP call 주소는 자신의 트랜잭션 이름(URL)에 `otel_virtual_httpc_name` 을 붙여서 설정합니다.

- **otel_virtual_httpc_name** `String`

기본값 `external http call`

HTTP call span 정보가 없을 때 강제로 설정하는 이름을 지정합니다.

- **transaction_status_error_enable** `Boolean`

기본값 `false`

status code가 400 이상일 경우 오류로 표기하도록 설정합니다.

- **tx_status_meter_enabled** `Boolean`

기본값 `true`

전체 URL의 Status Code별로 건수를 10초 간격으로 수집합니다.

- **stat_txstatus_enabled** `Boolean`

기본값 `true`

URL별 Status Code 건수를 5분 간격으로 수집합니다.

Gateway

OpenTelemetry Gateway 데이터 처리와 관련한 에이전트 설정 옵션입니다.

- **otel_gateway_enabled** Boolean

기본값 `false`

게이트웨이 정보 수집 여부를 설정합니다.

- 전체 URL의 트랜잭션, 게이트웨이, API 호출 각각의 경과 시간 10초 간격으로 수집합니다.
- URL, status별로 트랜잭션, 게이트웨이, API 호출 각각의 경과 시간을 5분 간격으로 수집합니다.

- **otel_attr_tx_gw_time_keys** String

기본값 `duration.kong`

게이트웨이 경과 시간을 가진 속성(attribute) 이름(key)을 지정합니다.

- **otel_attr_tx_api_time_key** String

기본값 `duration.proxy`

지정된 속성(attribute)의 값으로 API 호출 경과 시간을 설정합니다. API 호출 경과 시간을 가진 속성(attribute) 이름(key)을 지정합니다.

- **otel_replace_tx_resp_time_enabled** Boolean

기본값 `duration.proxy`

트랜잭션은 시작시간과 종료시간으로 응답시간을 계산합니다. 지정한 속성의 키값으로 트랜잭션의 응답시간을 변경하는 기능을 활성화합니다.

- 응답시간을 특정 속성(attribute)의 키 값은 `otel_attr_replace_tx_resp_time_keys` 로 설정합니다.

- **otel_attr_replace_tx_resp_time_keys** String

기본값 `duration.kong`

응답시간으로 설정할 속성의 키 이름을 지정합니다.

Metric

OpenTelemetry Metric 데이터 처리와 관련한 에이전트 설정 옵션입니다.

❗ OpenTelemetry의 Metric 데이터 처리에 대한 자세한 내용은 [다음 링크](#)를 참조하세요.

- **otel_metric_server_enabled** Boolean

기본값 `false`

OpenTelemetry Metric 데이터 처리를 활성화합니다.

- **otel_metric_queue_size** Int

기본값 `5000`

Opentelemetry Metric 데이터 처리를 위한 큐 크기를 설정합니다.

Log

OpenTelemetry Log 데이터 처리와 관련한 에이전트 설정 옵션입니다.

❗ OpenTelemetry의 Log 데이터 처리에 대한 자세한 내용은 [다음 링크](#)를 참조하세요.

- **otel_log_server_enabled** Boolean

기본값 `false`

Opentelemetry Log 데이터 처리를 활성화합니다.

- **otel_log_queue_size** Int

기본값 `5000`

Opentelemetry Log 데이터 처리를 위한 큐 크기를 설정합니다.

릴리즈 노트

최신 버전: v0.2.7

데이터를 불러오고 있습니다. 잠시만 기다려 주세요.

최신 에이전트 다운로드

최신 버전 0.2.7

OS	파일 형식	아키텍처	다운로드
Red Hat / CentOS / Amazon linux	RPM	x86 x64	다운로드
		aarch64	다운로드
Ubuntu 12 and above / Debian	DEB	amd64	다운로드
		arm64	다운로드
Alpine linux	tar gzip	x86 x64	다운로드
		aarch64	다운로드

❗ **OpenTelemetry**로 수집한 데이터를 와탭 모니터링 서비스에서 확인하는 방법에 대한 자세한 내용은 [다음 문서](#)를 참조하세요.