

# 컨테이너 자원 사용량(메모리) 미터링 커널 모듈 사용 예시

작성일:2019.11.18  
작성자:세종대학교 SYSCORE 연구실  
버전: 1.0.0-a1

# 목 차

1. 구동 환경
2. 프로메테우스 구동
3. Pushgateway 구동
4. Grafana 구동
5. 컨테이너 자원 사용량 모니터링을 위한 미터링용 커널 모듈
6. 커널 모듈 제어 스크립트 실행

## 1. 구동 환경

가. 컨테이너 모니터링 도구 구동 환경

(1) 본 모니터링 도구의 개발 및 테스트는 아래와 같은 환경에서 정상 구동을 확인하였다.

- 운영체제: Ubuntu 18.04
- Kernel version: 5.3.8-050308-generic
- docker version: 19.03.5
- Go version: go1.12.12

## 2. 프로메테우스 구동

가. 프로메테우스 설정 파일(prometheus.yml) 작성

(1) 프로메테우스 구동 전, 프로메테우스에 관한 설정 파일을 작성한다. 본 모니터링 도구가 구동하기 위한 최소한의 환경을 위한 설정은 아래 코드와 같다. 설정 파일(prometheus.yml)에서 Pushgateway IP와 Pushgateway Port는 구동 환경에 맞게 입력한다.

- 설정 파일 위치: /prometheus/prometheus.yml
- 설정 파일 내용

---

```
global:
  scrape_interval: 1s

scrape_configs:
  - job_name: 'Pushgateway'
    honor_labels: true
    static_configs:
      - targets: ['<Pushgateway IP>:<Pushgateway Port>']
```

---

<모니터링 도구 구동을 위한 프로메테우스 설정 파일(prometheus.yml)>

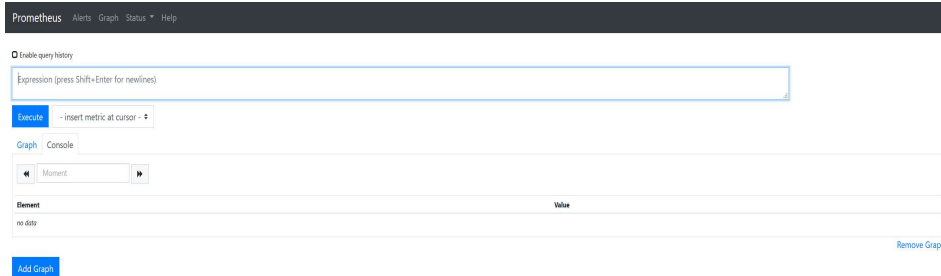
나. 프로메테우스 구동을 위한 명령어 수행

(1) 프로메테우스 구동을 위한 설정 파일이 작성된 상태에서 컨테이너 형태로 배포 중인 프로메테우스를 구동하기 위한 명령어는 다음과 같다. 외부 port 및 내부 port, prometheus.yml 파일의 위치는 구동 환경에 맞게 수정하여 사용하며 기본적으로 사용하는 port 번호는 9090이다.

- ❑ `docker run -td -p <외부 port>:<내부 port> --name prometheus_syscore -v <prometheus.yml 파일의 위치>:/etc/prometheus/prometheus.yml prom/prometheus`

- (2) 구동을 정상적으로 수행했을 때, 브라우저를 통해 정상 구동 여부를 확인할 수 있다. 아래 주소로 프로메테우스에 접근할 수 있으며, 정상 구동 시 [그림 1]와 같은 화면을 확인할 수 있다.

- Pushgateway 접속 주소: `http://<서버 IP>:<명령어 수행 시 설정한 port>`



[그림 1] 프로메테우스 구동 화면

### 3. Pushgateway 구동

가. Pushgateway 구동을 위한 명령어 수행

- (1) 컨테이너 형태로 배포 중인 Pushgateway를 구동하기 위한 명령어는 아래와 같다. 외부 port 및 내부 port의 내용은 구동 환경에 맞게 수정하여 사용한다. 기본적으로 사용하는 port 번호는 9091이다.

❑ `docker run -td -p <외부 port>:<내부 port> --name prometheus_syscore_gw kor easecurity/openfx:prometheus_gw`

- (2) 구동을 정상적으로 수행했을 때, 브라우저를 통해 정상 구동 여부를 확인할 수 있다. 아래 주소로 Pushgateway에 접근할 수 있으며, 정상 구동 시 [그림 2]와 같은 화면을 확인할 수 있다.

- Pushgateway 접속 주소: `http://<서버 IP>:<명령어 수행 시 설정한 port>`



[그림 2] Pushgateway 구동 화면

#### 4. Grafana 구동

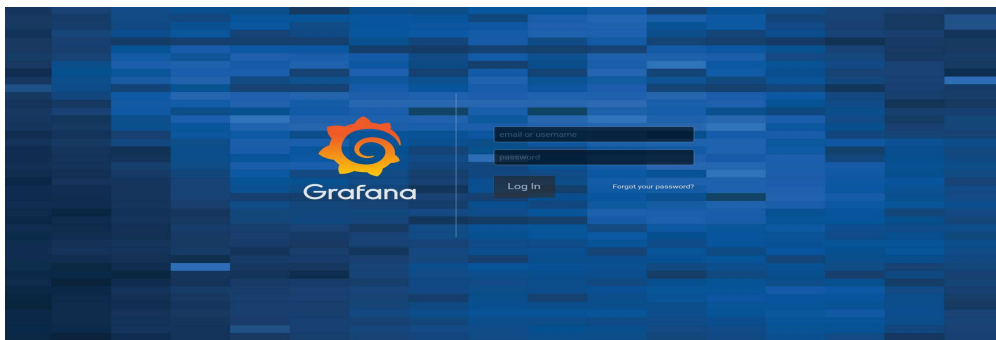
가. 수집 중인 Metric 데이터의 시각화를 위한 Grafana의 구동 명령어

- (1) 컨테이너 형태로 배포 중인 Grafana의 구동 과정은 다음과 같다. 외부 port 및 내부 port 번호는 구동 환경에 맞게 수정하여 사용하며 기본적으로 사용하는 port 번호는 3000이다.

❑ `docker run -td -p <외부 port>:<내부 port> --name grafana grafana/grafana`

- (2) 구동을 정상적으로 수행했을 때, 브라우저를 통해 정상 구동 여부를 확인할 수 있다. 아래 주소로 Grafana에 접근할 수 있으며, 정상 구동 시 [그림 3]와 같은 화면을 확인할 수 있다.

- Grafana 접속 주소: `http://<서버 IP>:<명령어 수행 시 설정한 port>`



[그림 3] Grafana 구동 화면

#### 5. 컨테이너 자원 사용량 모니터링을 위한 미터링용 커널 모듈

가. 미터링용 커널 모듈 컴파일 과정

- (1) 컨테이너 자원 사용량 모니터링을 위해 커널 모듈의 컴파일이 수행되어야 한다. 미터링용 커널 모듈의 위치는 *kernel\_module* 디렉터리 내부에 존재하며, 디렉터리 내부에서 *make* 명령어를 통해 *monitor.c* 파일에 대한 컴파일을 수행한다.

- (2) 컴파일 수행 후 *monitor.ko* 파일이 생성되며, 해당 모듈에 대한 적재 과정이 필요하다. 커널 모듈의 적재 과정은 아래와 같은 명령어를 통해 수행한다.

❑ `insmod monitor.ko pid=[] container_name=[] pid_count=len(pid)`

- (3) 모듈이 정상적으로 적재되면 *dmesg* 명령어를 통해 아래 그림과 같은 로그

메시지를 확인할 수 있다.

```
[ 9854.894203] [INFO] Metric Module ON
```

[그림 4] Kernel Module Load 확인

## 6. 커널 모듈 제어 스크립트 실행

가. 커널 모듈 제어를 위한 스크립트 실행

- (1) 커널 모듈 제어를 위한 스크립트는 모니터링 모듈을 동적으로 사용하기 위한 기능을 수행한다. 해당 스크립트는 컨테이너의 생성과 제거에 따라서 모듈 적재 과정을 자동으로 수행하기 때문에 *monitor.ko*의 적재 과정이 불필요하다. 해당 스크립트 파일명은 *kernel\_manager.py*이며 아래 명령어를 통해 실행 가능하다.

□ `python3 kernel_manager.py`

- (2) *kernel\_manager.py*가 정상적으로 실행되면 아래와 그림과 같은 메시지가 출력된다.

```
2019-11-18 18:09:46.636904 [init] Monitor
2019-11-18 18:09:46.808483 [info] Detection Container : 3
2019-11-18 18:09:46.808637 [info] Detection Container : 3
2019-11-18 18:09:46.972539 [info] Detection Container : 3
2019-11-18 18:09:46.972696 [update] Kernel Module
```

[그림 5] kernel-manager 구동 확인