2024.11.26 졸업논문발표

cAdvisor를 이용한 도커 컨테이너 모니터링 시스템 구현

학과 : 컴퓨터공학과
학번 : 20191194
이름 : 이우진

Contents

01

개발 배경 및 목적

1

컨테이너 기술의
사용 증가

2

컨테이너 환경
실시간 확인

3

성능 문제, 보안 사고
사전에 식별 및 해결

02

논문의 이론적 배경

Docker는 컨테이너 기반 가상화 플랫폼으로, 응용 프로그램을 격리된 환경인 컨테이너로 패키징하여 실행하는 기술

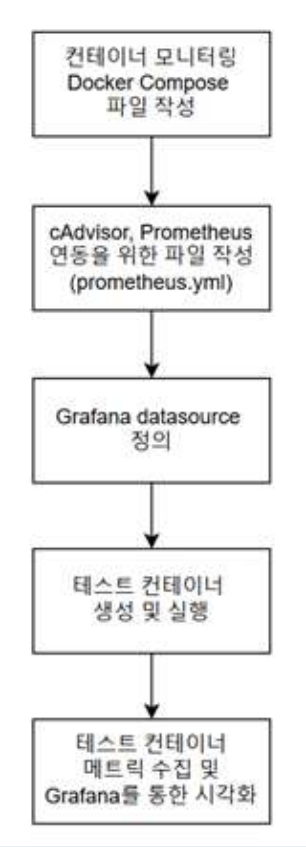
cAdvisor는 컨테이너에 대한 정보를 수집, 처리 및 내보내는 데몬

Prometheus는 다양한 대상에서 시간 경과에 따른 지표를 수집하여 시계열 데이터베이스 형태로 저장하는 오픈 소스 모니터링 도구

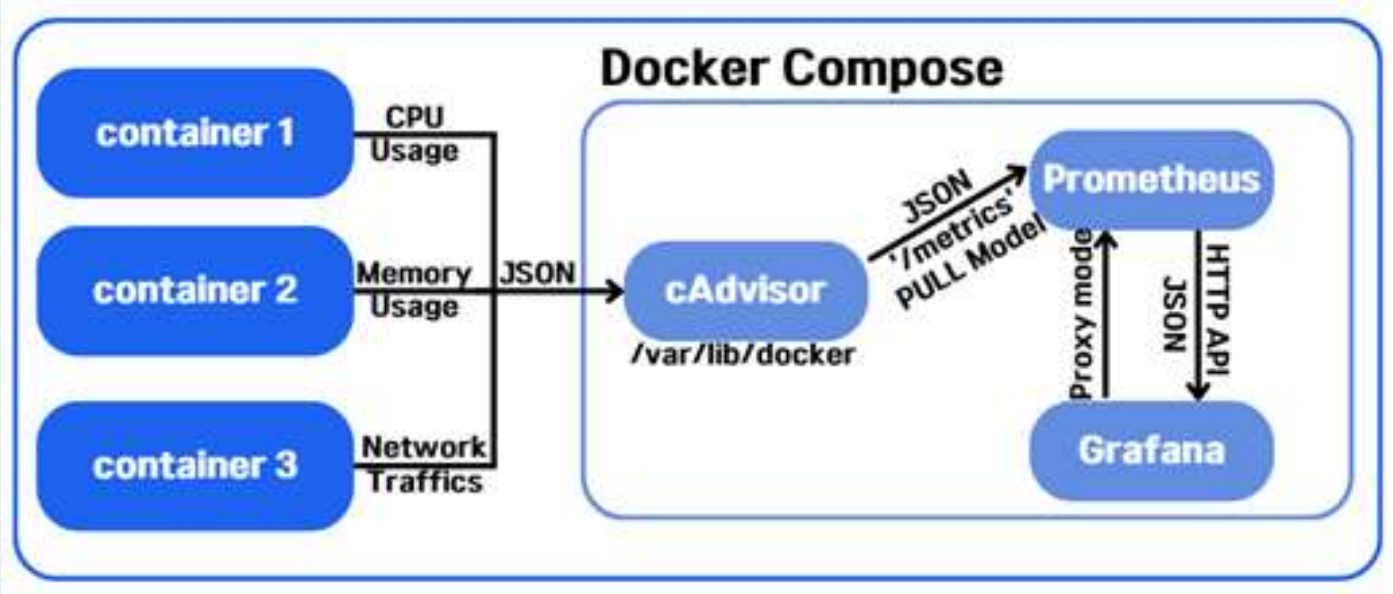
Grafana는 시계열 메트릭 데이터를 시각화 하는데 대시보드를 제공해주는 오픈소스

03

시스템 작동 설계



순서도



작동 요약

04

실험 내용

- cAdvisor 컨테이너 생성하며 volumes 섹션의 /var/lib/docker/:/var/lib/docker 설정을 통해 cAdvisor가 Docker의 메타데이터 및 컨테이너 파일들이 저장된 디렉터리를 마운트하여 Docker 관련 정보를 수집할 수 있게 한다.

- privileged 섹션에서 루트 권한을 부여하여 cAdvisor가 더 많은 시스템 리소스와 하드웨어에 접근할 수 있게 한다.

- devices 섹션에서 호스트 커널 로그 파일을 컨테이너에 노출시켜 cAdvisor가 시스템 로그에 접근할 수 있도록 한다.

```
cadvisor:
  container_name: cadvisor
  image: gcr.io/cadvisor/cadvisor:latest
  network_mode: "host"
  ports:
    - "8080:8080"
  volumes:
    - "/:/rootfs"
    - "/var/run:/var/run"
    - "/sys:/sys"
    - "/var/lib/docker/:/var/lib/docker"
    - "/dev/disk/:/dev/disk"
  privileged: true
  devices:
    - "/dev/kmsg"
```

cAdvisor 컨테이너 생성

05

실험 내용

- prometheus 컨테이너 생성하며 volumes 섹션에 호스트의 prometheus.yml 파일을 마운트하여 데이터 수집 간격과 수집 대상을 설정한다.

- prometheus.yml에 메트릭 수집 간격과 수집된 데이터에 기반하여 규칙을 평가하는 작업을 수행하는 간격을 15초로 설정한다.

- 메트릭을 수집하는 대상을 promethus와 cadvisor로 설정한다.

```
prometheus:
  container_name: prometheus
  image: prom/prometheus:latest
  network_mode: "host"
  ports:
    - "9090:9090"
  volumes:
    - "./prometheus.yml:/etc/prometheus/prometheus.yml"
  privileged: true
  depends_on:
    - cadvisor
```

prometheus 컨테이너 생성

```
global:
  scrape_interval: 15s
  evaluation_interval: 15s

scrape_configs:
  - job_name: "prometheus"
    static_configs:
      - targets: ["localhost:9090"]

  - job_name: "cadvisor"
    static_configs:
      - targets: ["localhost:8080"]
```

prometheus.yml

06

실험 내용

- DS_PROMETHEUS=promethus를 통해 prometheus에서 데이터를 가져오도록 한다.
- datasources.yml 파일을 Grafana의 프로비저닝 디렉터리에 마운트한다.
- datasources.yml 파일에 Grafana가 사용할 데이터 소스로 Prometheus를 지정하고 있으며, 데이터를 가져올 위치와 proxy 모드를 통해 Grafana 서버가 간접적으로 Prometheus에 데이터를 요청할 수 있게 한다.

```
grafana:
  container_name: grafana
  image: grafana/grafana:latest
  network_mode: "host"
  ports:
    - "3000:3000"
  environment:
    - GF_PATHS_PROVISIONING=/etc/grafana/provisioning
    - DS_PROMETHEUS=prometheus
  volumes:
    - "grafana-data:/var/lib/grafana"
    - "./datasources.yml:/etc/grafana/provisioning/datasources/datasources.yml"
  privileged: true
  depends_on:
    - prometheus
```

Grafana 컨테이너 생성

```
datasources:
  - name: prometheus
    type: prometheus
    url: http://localhost:9090
    access: proxy
    isDefault: true
```

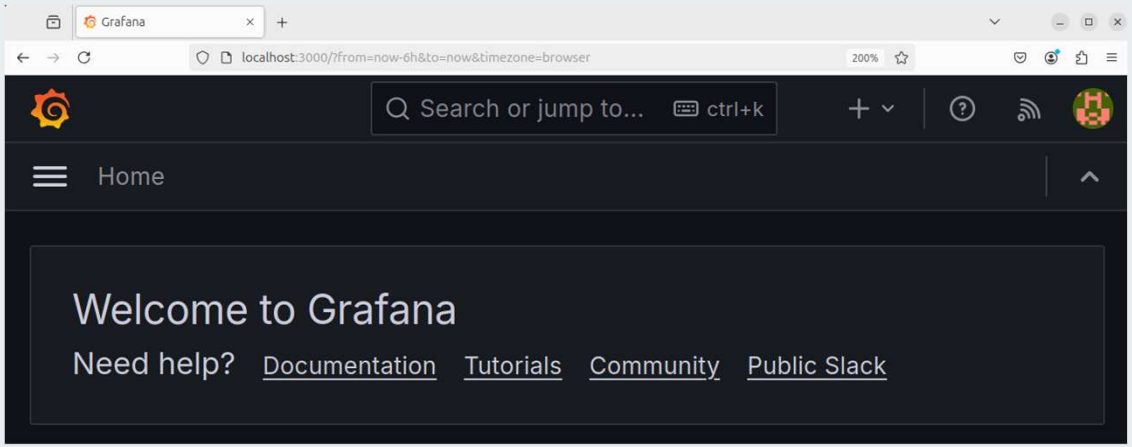
datasources.yml

07

실험 내용

- 할당한 Grafana 주소로 접속하여 대시보드를 구성한다.

- CPU 사용량, memory 사용량, network traffic을 일시적으로 높이는 파이썬 코드를 담은 컨테이너를 실행시켜 모니터링 시스템을 테스트한다.



실행된 Grafana 웹 UI

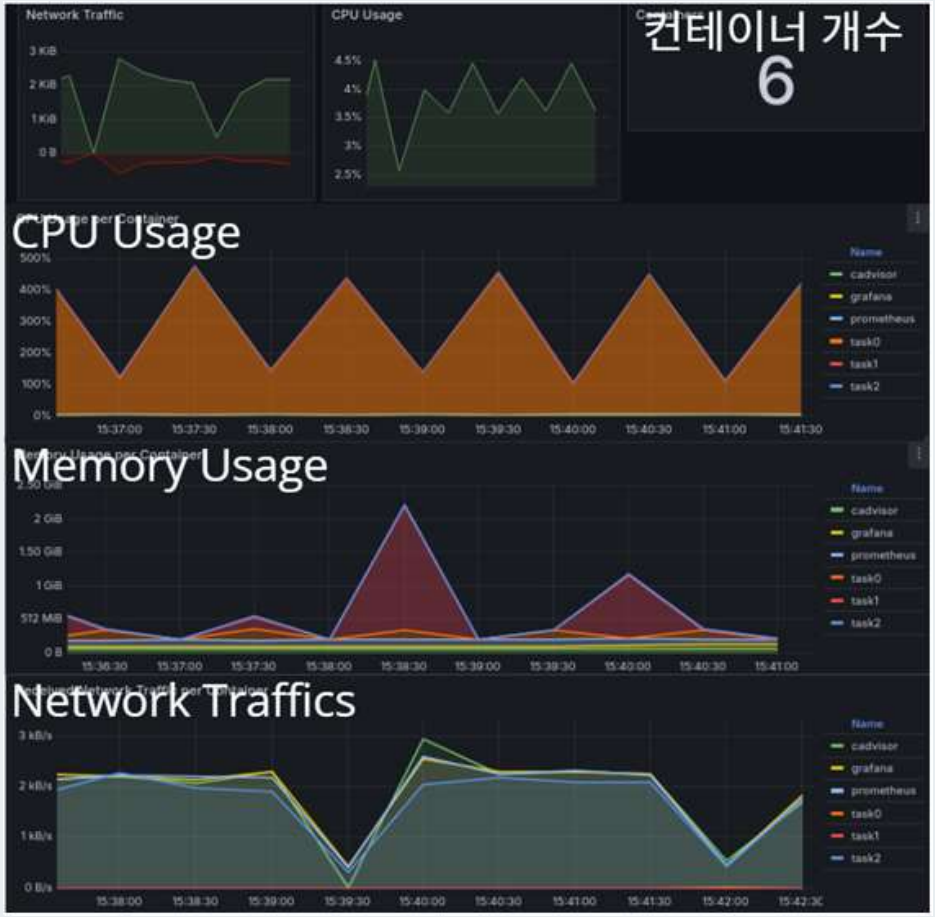
```
woojin@ububtu-docker:~$ docker ps -a
CONTAINER ID   IMAGE                              COMMAND
6b2c01761f33   memory-task                        "python memory_task.…"
ed276400721a   cpu-task                           "python cpu_task.py"
5e8765b76544   network-task                       "python network_task…"
012aa5707123   grafana/grafana:latest             "/run.sh"
0981323c0b25   prom/prometheus:latest             "/bin/prometheus --c…"
3929ff292148   gcr.io/cadvisor/cadvisor:latest    "/usr/bin/cadvisor -…"
```

실행중인 컨테이너 목록

실험 결과

- http://localhost:3000으로 Grafana에 접속하여 대쉬보드를 구성한 모습이다.

- 현재 실행 중인 컨테이너의 개수를 확인할 수 있으며, 각 컨테이너의 CPU 사용량, Memory 사용량, Network Traffics의 변화량을 실시간으로 확인할 수 있다.



09

결론

- cAdvisor를 통해 실시간으로 Docker 컨테이너의 메트릭 데이터를 수집할 수 있었다.
- Prometheus는 자체 시계열 데이터베이스에 수집된 데이터를 효율적으로 저장하고 관리함으로써, 장기적인 데이터를 바탕으로 컨테이너 성능 변화를 분석하고 예측할 수 있게 하였다.
- Grafana는 수집된 데이터를 시각화 시킴으로써, 메트릭 정보를 한눈에 파악하고 문제의 근본적인 원인을 쉽게 찾을 수 있는 환경을 제공하였다.
- 이와 같이 구현된 시스템을 통해 Docker 컨테이너 환경에서 발생할 수 있는 성능 문제를 사전에 식별하고 해결할 수 있으며, 시스템 자원 소모를 관리함으로써 지속적인 애플리케이션의 안전성을 확보할 수 있을 것이다.

감사합니다!