

LAPORAN
MONITORING DOCKER CONTAINER DENGAN
PROMETHEUS DAN GRAFANA



Disusun Oleh:

2 D4 Teknik Informatika A (Kelompok 7)

Muhammad Qois Haidar (3122600001)

Raihan Eka Pramudya (3122600011)

Muhammad Arief S. W. (3122600015)

PROGRAM STUDI D4 TEKNIK INFORMATIKA
DEPARTEMEN TEKNIK INFORMATIKA DAN KOMPUTER
POLITEKNIK ELEKTRONIKA NEGERI SURABAYA
2024

I. Abstraksi

Proyek ini bertujuan untuk membangun sistem monitoring container Docker menggunakan Prometheus, cAdvisor, dan Grafana. Sistem ini akan memantau performa dan kesehatan container Docker, serta mengirimkan notifikasi melalui bot Telegram jika terjadi kondisi abnormal. Implementasi sistem ini diharapkan dapat meningkatkan efisiensi dalam pengelolaan dan pemeliharaan container Docker.

II. Pendahuluan

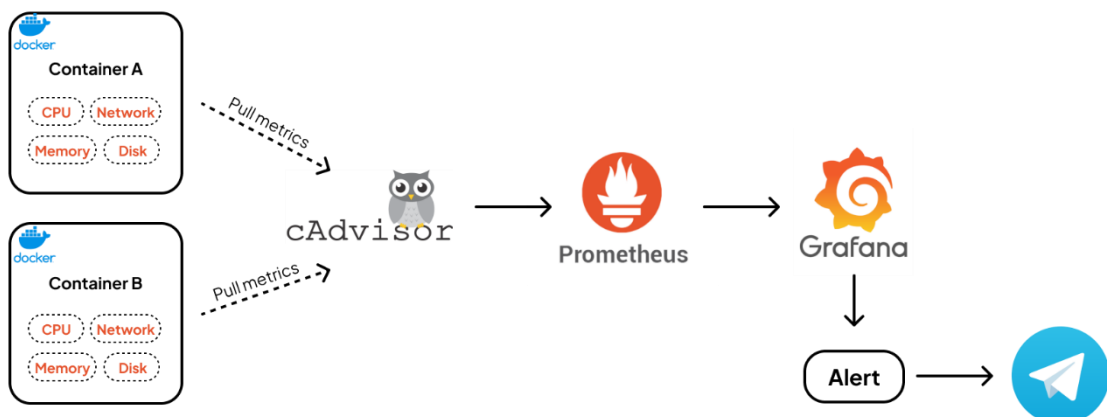
Container Docker adalah alat yang populer untuk mengemas aplikasi dengan dependensi mereka, memastikan bahwa aplikasi berjalan dengan konsisten di berbagai lingkungan. Untuk memastikan kinerja dan kesehatan container tetap optimal, diperlukan sistem monitoring yang baik. Proyek ini menggunakan cAdvisor untuk pengumpulan metrik dari container, Prometheus untuk menyimpan metrik, dan Grafana untuk visualisasi data. Selain itu, sistem ini juga diintegrasikan dengan bot Telegram untuk mengirimkan notifikasi alert.

III. Ruang Lingkup

Proyek ini mencakup:

- Instalasi dan konfigurasi Prometheus, cAdvisor, dan Grafana.
- Pengaturan bot Telegram untuk menerima notifikasi alert.
- Monitoring metrik performa utama seperti CPU, memori, dan jaringan untuk container Docker.
- Visualisasi metrik tersebut menggunakan Grafana.
- Pengaturan alert di Grafana yang dikirimkan melalui bot Telegram jika terjadi kondisi kritis pada container.

IV. Desain Sistem



- cAdvisor: Mengumpulkan metrik dari container Docker.
- Prometheus: Bertugas untuk mengumpulkan dan menyimpan metrik dari cAdvisor.
- Grafana: Menyediakan antarmuka visual untuk memantau metrik yang dikumpulkan oleh Prometheus.
- Telegram Bot: Mengirimkan notifikasi alert dari Grafana kepada tim/user melalui Telegram

V. Tim dan Tugas

- **Muhammad Qois Haidar**
 - Melakukan pengujian sistem untuk memastikan semua komponen berfungsi dengan baik.
- **Raihan Eka Pramudya**
 - Bertanggung jawab atas instalasi dan konfigurasi Prometheus, cAdvisor, dan Grafana.
 - Mengawasi keseluruhan proyek dan memastikan tujuan tercapai.
 - Menangani integrasi dengan bot Telegram.
- **Muhammad Arief S. W.**
 - Memastikan infrastruktur server berjalan dengan aman dan optimal.

VI. Tahapan Pelaksanaan

1. **Perencanaan:** Mendefinisikan tujuan dan ruang lingkup proyek, serta pembagian tugas.
2. **Instalasi dan Konfigurasi:**
 - Instalasi Docker, Prometheus, cAdvisor, dan Grafana.
 - Konfigurasi Prometheus untuk mengumpulkan data dari cAdvisor.
 - Konfigurasi Grafana untuk visualisasi data.
3. **Integrasi Bot Telegram:**
 - Membuat bot Telegram dan mendapatkan token API.
 - Konfigurasi Alertmanager untuk mengirim alert ke Telegram.
4. **Pengaturan Alert:** Membuat aturan alert di Grafana.
5. **Pengujian Sistem:** Melakukan pengujian untuk memastikan semua komponen berfungsi dengan baik dan alert dikirim dengan benar.
6. **Implementasi dan Peluncuran:** Mendemokan sistem monitoring yang telah berjalan.

VII. Implementasi

1. Instalasi Docker:

```
raihan@debian:~$ sudo docker --version
Docker version 26.1.3, build b72abbb
raihan@debian:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; preset: enabled)
   Active: active (running) since Sat 2024-06-01 16:27:06 WIB; 38s ago
 TriggeredBy: ● docker.socket
     Docs: https://docs.docker.com
    Main PID: 5863 (dockerd)
       Tasks: 10
      Memory: 35.2M
         CPU: 581ms
    CGroup: /system.slice/docker.service
            └─5863 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.s
```

2. Membuat direktori baru bernama Monitoring:

```
root@debian:/home/raihan# mkdir Monitoring
root@debian:/home/raihan# ls
Desktop  Documents  Downloads  Monitoring
```

3. Membuat file docker-compose.yml dan prometheus.yml:

```
root@debian:/home/raihan/Monitoring# vi docker-compose.yml
root@debian:/home/raihan/Monitoring# vi prometheus.yml
```

4. Instalasi kontainer Prometheus, Grafana, cAdvisor pada file docker-compose.yml:

```
version: '3.2'

services:
  prometheus:
    image: prom/prometheus:latest
    container_name: prometheus
    ports:
      - 9090:9090
    command:
      - --config.file=/home/raihan/Monitoring/prometheus.yml
    volumes:
      - ./prometheus.yml:/home/raihan/Monitoring/prometheus.yml:ro
    depends_on:
      - cAdvisor

  grafana:
    image: grafana/grafana:latest
    container_name: grafana
    ports:
      - "3000:3000"
    volumes:
      - grafana-storage:/var/lib/grafana

  cAdvisor:
    image: gcr.io/cadvisor/cadvisor:latest
    container_name: cAdvisor
    ports:
      - 8080:8080
    volumes:
      - /:/rootfs:ro
      - /var/run:/var/run:rw
      - /sys:/sys:ro
      - /var/lib/docker:/var/lib/docker:ro
    depends_on:
      - redis

  redis:
    image: redis:latest
    container_name: redis
    ports:
      - 6379:6379

volumes:
  grafana-storage:
```

5. Konfigurasi prometheus pada file prometheus.yml untuk mengumpulkan metrik:

```
global:
  scrape_interval: 15s

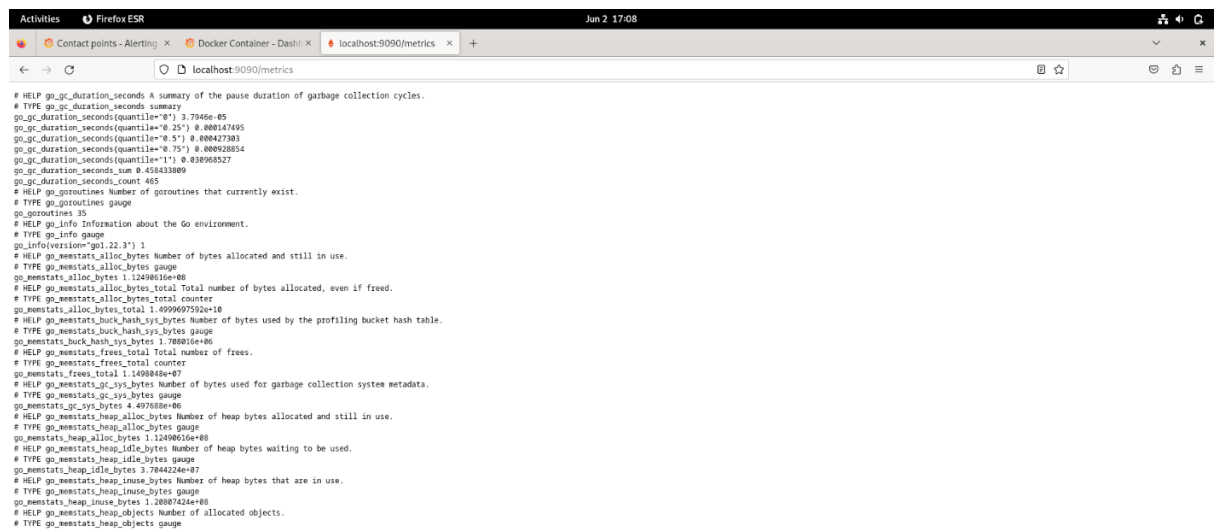
scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']

  - job_name: 'cadvisor'
    static_configs:
      - targets: ['cadvisor:8080']
```

6. Mengecek kontainer:

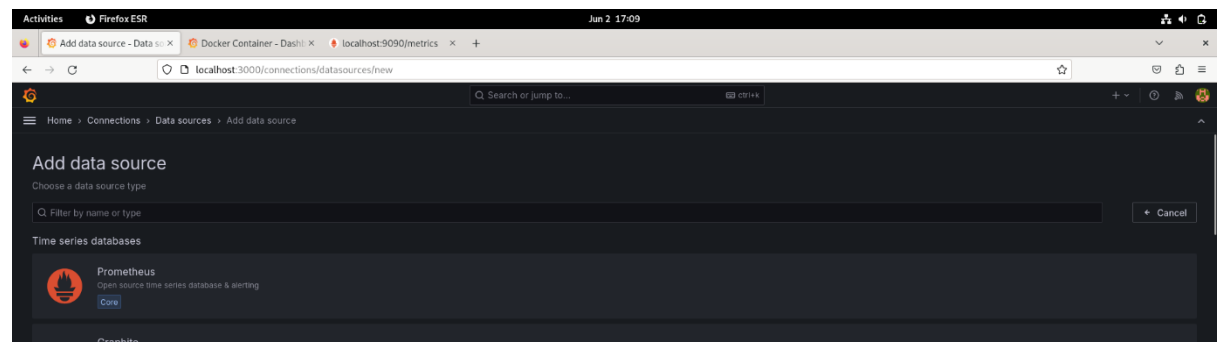
```
root@debian:/home/raihan/Monitoring# docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
64f37411a638   prom/prometheus:latest             "/bin/prometheus --c..." 10 hours ago  Up 10 hours   0.0.0.0:9090->9090/tcp,
::9090->9090/tcp
c4f22a7ff36a   gcr.io/cadvisor/cadvisor:latest    "/usr/bin/cadvisor -..." 10 hours ago  Up 10 hours (healthy)  0.0.0.0:8080->8080/tcp,
::8080->8080/tcp
f751eac3cd91   grafana/grafana:latest             "/run.sh"                10 hours ago  Up 10 hours      0.0.0.0:3000->3000/tcp,
::3000->3000/tcp
ee27b2ba9b82   redis:latest                        "docker-entrypoint.s..." 10 hours ago  Up 7 minutes       0.0.0.0:6379->6379/tcp,
::6379->6379/tcp
```

7. Mengecek metrics pada prometheus:

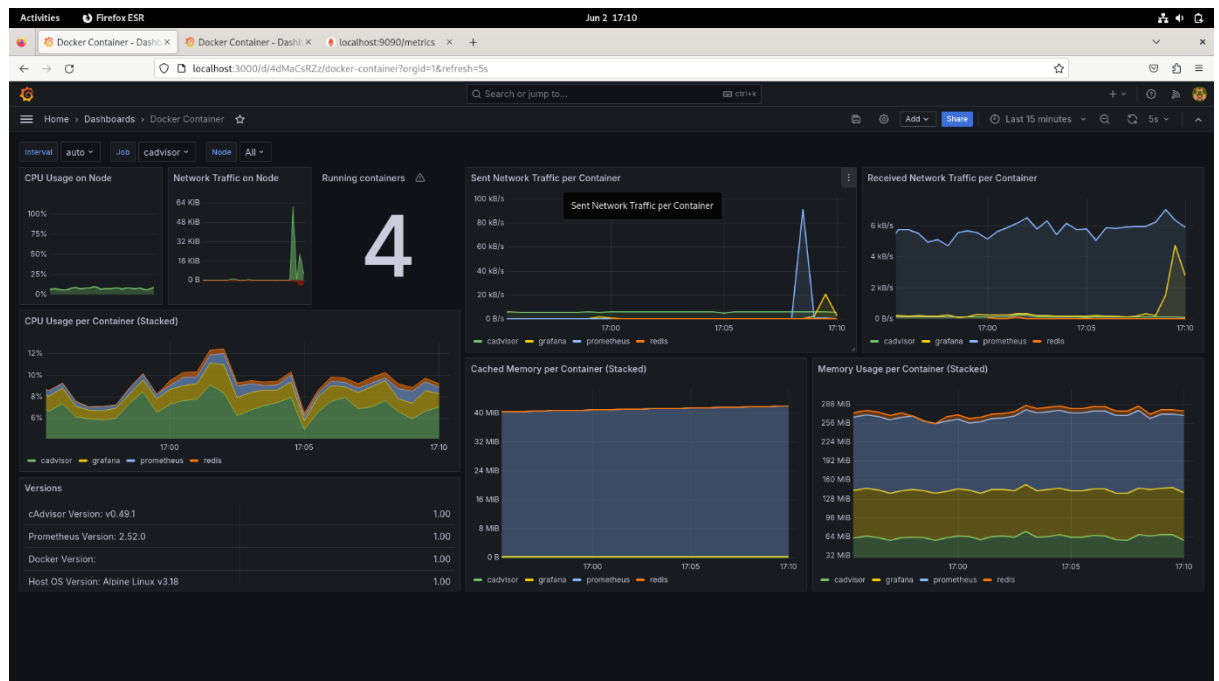


```
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 3.7940e+03
go_gc_duration_seconds{quantile="0.25"} 0.000147495
go_gc_duration_seconds{quantile="0.5"} 0.000427383
go_gc_duration_seconds{quantile="0.75"} 0.000920834
go_gc_duration_seconds{quantile="1"} 0.030909327
go_gc_duration_seconds_sum 0.458433809
go_gc_duration_seconds_count 465
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 35
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.22.3"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 1.1249010e+08
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 1.4999697592e+10
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.708810e+00
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 1.1498840e+07
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 4.407680e+00
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and still in use.
# TYPE go_memstats_heap_alloc_bytes gauge
go_memstats_heap_alloc_bytes 1.1249010e+08
# HELP go_memstats_heap_idle_bytes Number of heap bytes waiting to be used.
# TYPE go_memstats_heap_idle_bytes gauge
go_memstats_heap_idle_bytes 3.7844224e+07
# HELP go_memstats_heap_inuse_bytes Number of heap bytes that are in use.
# TYPE go_memstats_heap_inuse_bytes gauge
go_memstats_heap_inuse_bytes 1.2080742e+00
# HELP go_memstats_heap_objects Number of allocated objects.
# TYPE go_memstats_heap_objects gauge
```

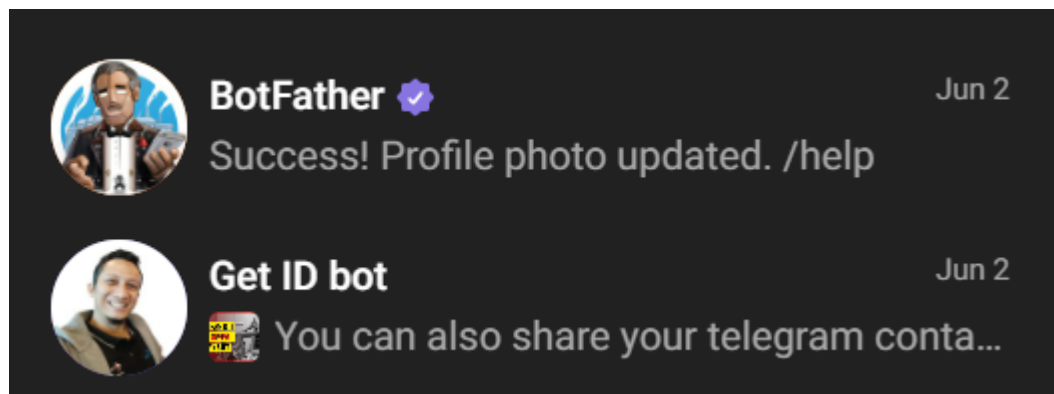
8. Menambahkan data source baru dari prometheus:



9. Tampilan kondisi kontainer pada dashboard grafana:



10. Membuat bot pada telegram dengan BotFather serta Get ID bot:



11. Konfigurasi Contact Points untuk telegram:

The screenshot shows the 'Contact points' configuration page in the Grafana Alertmanager interface. The page is titled 'Contact points' and includes a subtitle 'Choose how to notify your contact points when an alert instance fires'. The 'Integration' dropdown is set to 'Telegram'. The 'BOT API Token' field is filled with 'Telegram BOT API Token'. The 'Chat ID' field is filled with 'Integer Telegram Chat Identifier'. There are buttons for 'Test', 'Duplicate', and 'Delete'. At the bottom, there are buttons for 'Save contact point' and 'Cancel'.

12. Konfigurasi Notification Policy untuk alert:

The screenshot shows the 'Edit notification policy' dialog. It has a 'Default contact point' dropdown set to 'grafana-telegram' with a link to 'Create a contact point'. Below is a 'Group by' section with a dropdown showing 'grafana_folder X' and 'alertname X'. The 'Timing options' section includes three fields: 'Group wait' set to '30s', 'Group interval' set to '1m', and 'Repeat interval' set to '1h'. At the bottom are 'Cancel' and 'Update default policy' buttons.

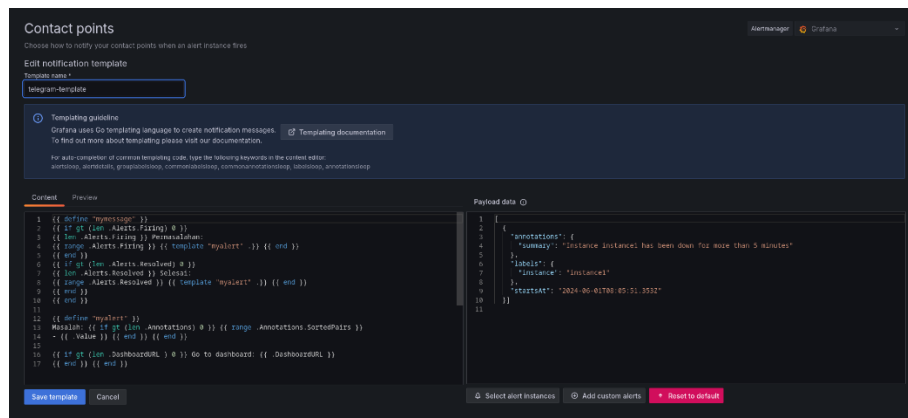
13. Konfigurasi alert rules untuk container yang mati:

The screenshot shows the 'Edit rule' page. Step 1 'Enter alert rule name' has 'Container Alert' entered. Step 2 'Define query and alert condition' shows a Prometheus query: `count(rate(container_last_seen{name=~".+"}[30s]))`. The 'Rule type' is 'Grafana-managed'. Under 'Expressions', a 'Threshold' condition is set with 'Input' A.

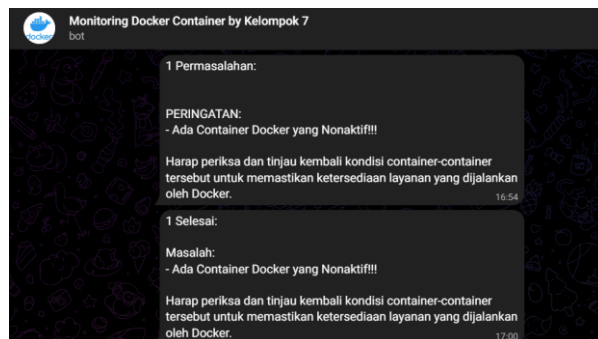
14. Konfigurasi alert rules untuk memory pada tiap container:

The screenshot shows the 'Edit rule' page. Step 1 'Enter alert rule name' has 'Memory Alert' entered. Step 2 'Define query and alert condition' shows a Prometheus query: `sum(container_memory_usage{name=~".+"}) by (name) / 1024000`. The 'Rule type' is 'Grafana-managed'. Under 'Expressions', a 'Threshold' condition is set with 'Input' B.

15. Konfigurasi template message untuk susunan teks telegram:



16. Percobaan ketika ada container yang mati:



VIII. Sistem Testing

1. Uji Fungsional:

- Verifikasi bahwa Prometheus mengumpulkan metrik dari cAdvisor.
- Pastikan Grafana menampilkan metrik yang dikumpulkan dengan benar.
- Uji pengiriman alert melalui bot Telegram dengan memicu kondisi (misalnya, mematikan salah satu container docker).

2. Uji Integrasi:

- Pastikan integrasi antara cAdvisor, Prometheus, Grafana, dan bot Telegram berjalan dengan baik.
- Verifikasi pengiriman alert ke Telegram dan pastikan pesan alert sesuai dengan aturan yang ditentukan.

3. Uji Performa:

- Uji kinerja sistem monitoring dengan beberapa container untuk memastikan tidak ada bottleneck.
- Monitor penggunaan sumber daya (CPU, memori) oleh Prometheus, cAdvisor, dan Grafana.