Monitoring DevOps Menggunakan Prometheus dan Grafana

1. Maksud

Maksud dari praktikum ini adalah untuk memberikan pengalaman langsung dalam penerapan sistem monitoring pada lingkungan DevOps menggunakan dua alat populer, yaitu Prometheus sebagai engine pengumpulan metrik dan Grafana sebagai alat visualisasi dashboard. Modul ini dirancang agar mahasiswa/peserta mampu memahami dan mengimplementasikan monitoring infrastruktur secara real-time.

2. Tujuan

Setelah menyelesaikan praktikum ini, peserta diharapkan dapat:

a) Memahami konsep dasar monitoring DevOps.

b) Menginstal dan menjalankan Prometheus dan Grafana menggunakan Docker.

c) Mengintegrasikan Prometheus dengan node target (misalnya Node Exporter).

d) Membuat dashboard monitoring di Grafana.

e) Mengidentifikasi dan menyelesaikan permasalahan melalui analisis metrik yang terkumpul.

3. Dasar Teori

a. Prometheus

Prometheus adalah sistem monitoring open-source yang dikembangkan oleh SoundCloud dan saat ini dikelola oleh Cloud Native Computing Foundation (CNCF). Prometheus bekerja dengan cara melakukan scraping terhadap endpoint HTTP dari aplikasi atau layanan yang ingin dimonitor, lalu menyimpan data tersebut sebagai time-series database.

Fitur utama:

a) Query bahasa: PromQL

b) Scraping otomatis berdasarkan interval waktu

c) Alerting rule

d) Visualisasi sederhana via web UI

b. Grafana

Grafana adalah platform visualisasi data open-source yang mendukung banyak sumber data seperti Prometheus, MySQL, PostgreSQL, Loki, dll. Grafana digunakan untuk membuat dashboard interaktif dan profesional untuk memantau kinerja sistem secara real-time.

Fitur utama:

a) Dashboard multi-panel

b) Integrasi dengan Prometheus

c) Alerting

d) Plugin ekstensibilitas

c. Node Exporter

Node Exporter adalah exporter yang digunakan untuk mengambil metrik hardware dan sistem operasi dari server Linux/Windows. Metrik yang tersedia antara lain CPU usage, memory, disk I/O, jaringan, dll.

4. Prosedur Praktik

Langkah-Langkah Instalasi Prometheus & Grafana di Docker

Prasyarat

- Docker sudah terinstall di sistem kamu.

```
docker --version
```

1. Pull Image Prometheus dan Grafana

Jalankan perintah berikut untuk mendownload image Prometheus dan Grafana dari Docker Hub:

```
docker pull prom/prometheus
docker pull grafana/grafana
```

Tunggu sampai proses download selesai. Pastikan koneksi internet stabil.

2. Cek Image yang Telah Terdownload

Lihat apakah image Prometheus dan Grafana telah berhasil diunduh:

```
docker images
REPOSITORY          TAG       IMAGE ID       CREATED         SIZE
prom/prometheus     latest    abcdef123456   2 weeks ago     200MB
grafana/grafana     latest    ghijklm7890ab   3 weeks ago     300MB
```

3. Jalankan Container Prometheus

Gunakan perintah berikut untuk membuat dan menjalankan container Prometheus:

```
docker run -d \
  --name prometheus \
  -p 9090:9090 \
  prom/prometheus
```

Penjelasan:

- **-d**: Menjalankan container di background (detached mode)
- **--name**: Memberi nama container sebagai **prometheus**
- **-p**: Mapping port 9090 dari container ke host agar bisa diakses via browser

Catatan: Konfigurasi awal yang digunakan adalah bawaan dari image Prometheus. Untuk konfigurasi custom, lihat bagian tambahan di bawah.

4. Jalankan Container Grafana

Jalankan container Grafana dengan perintah berikut:

```
docker run -d \
  --name grafana \
  -p 3000:3000 \
  -e GF_SECURITY_ADMIN_PASSWORD=admin \
  grafana/grafana
```

Penjelasan:

- **-p 3000**: Mapping port Grafana agar bisa diakses melalui browser
- **-e GF_SECURITY_ADMIN_PASSWORD=admin**: Mengatur password default admin Grafana menjadi **admin**

5. Periksa Status Container

Pastikan kedua container berjalan tanpa error:

```
docker ps
```

Harus muncul output seperti ini:

```
CONTAINER ID   IMAGE                    COMMAND                  PORTS
NAMES
...            prom/prometheus          "/bin/prometheus ..."    0.0.0.0:9090-
>9090/tcp   prometheus
...            grafana/grafana          "/run.sh"                0.0.0.0:3000-
>3000/tcp   grafana
```

6. Akses Web UI Prometheus

Buka browser dan kunjungi:

```
http://localhost:9090
```

Kamu akan melihat antarmuka Prometheus. Di sini kamu bisa menulis query PromQL dan melihat target scraping.

7. Akses Web UI Grafana

Buka browser dan kunjungi:

```
http://localhost:3000
```

Login dengan:

- Username: **admin**
- Password: **admin**

8. Integrasi Grafana dengan Prometheus

Setelah login ke Grafana, lakukan langkah berikut:

1. Klik ikon Configuration (gear icon) → Data Sources → Add data source
2. Pilih Prometheus
3. Isi form:
    - Name: **Prometheus**
    - URL: **http://host.docker.internal:9090**
      (Alternatif: gunakan IP host atau pastikan konektivitas antar-container)
4. Klik tombol Save & Test

9. (Opsional) Buat File Konfigurasi Prometheus Custom

Jika ingin menggunakan konfigurasi scraping sendiri, buat file **prometheus.yml** di direktori lokal, contoh:

```
global:
  scrape_interval: 15s

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

Lalu jalankan ulang container Prometheus dengan mount file tersebut:

```
docker stop prometheus && docker rm prometheus

docker run -d \
  --name prometheus \
  -p 9090:9090 \
  -v $(pwd)/prometheus.yml:/etc/prometheus/prometheus.yml \
  prom/prometheus
```

5. Studi Kasus dan Penyelesaian

Studi Kasus: Monitoring Penggunaan CPU dan Memory Server

Deskripsi Masalah:

Sebuah server produksi mengalami penurunan performa. Perlu dilakukan analisis apakah ada lonjakan CPU atau memory usage.

Solusi:

1. Ambil metrik dari Node Exporter
    a) Metrik CPU: **node_cpu_seconds_total**
    b) Metrik Memory: **node_memory_MemAvailable_bytes**, **node_memory_MemTotal_bytes**
2. Query CPU Usage (PromQL)

```
instance:node_num_cpu:sum
```

atau

```
100 - (avg by (instance) (rate(node_cpu_seconds_total{mode="idle"}[5m])) * 100)
```

3. Query Memory Usage

```
(node_memory_MemTotal_bytes - node_memory_MemAvailable_bytes) / node_memory_MemTotal_bytes * 100
```

4. Buat Panel di Grafana
    a) Tambahkan panel baru
    b) Pilih datasource: Prometheus
    c) Masukkan query di atas
    d) Ganti tipe grafik menjadi “Time series” atau “Gauge”
5. Analisis Hasil
    a) Jika CPU > 80% atau Memory > 90%, maka perlu investigasi lebih lanjut (apakah ada proses yang overload? leak memory?)
    b) Rekomendasi: tambahkan alert jika threshold tertentu dilewati

6. Tugas

1. Tugas Mandiri
    a) Jalankan Prometheus, Grafana, dan Node Exporter menggunakan Docker Compose seperti langkah di atas.
    b) Pastikan semua target dapat di-scrape tanpa error.
    c) Buat satu dashboard di Grafana yang menampilkan:
        - Penggunaan CPU (%)
        - Penggunaan Memory (%)
        - Disk I/O
        - Jumlah user aktif

2. Tugas Analisis

    a) Simulasikan kondisi server overload (misalnya dengan menjalankan loop bash di terminal):

    ```
    while true; do echo "Simulating high load"; done
    ```

    b) Amati perubahan di dashboard Grafana selama simulasi.

    c) Catat dan analisis pola perubahan CPU dan memory usage.