

LAPORAN PROYEK
Monitoring Virtual Machine(VirtualBox) with Prometheus and Grafana
Mata Kuliah Workshop Administrasi Jaringan



Dosen:

Dr. Ferry Astika Saputra, S.T, M.Sc

OLEH :

MOCH. IRHAM KAFI BILLAH (3122600009)
SEPTIAN ACHMAD RAJABBI (3122060025)
NUFUS AKMALUL MAFAZA (3122600026)

Kelas 2 D4 Teknik Informatika A
Departemen Teknik Informatika dan Komputer
Politeknik Elektronika Negeri Surabaya

DAFTAR ISI

DAFTAR ISI.....	1
A. ABSTRAKSI.....	2
B. PENDAHULUAN.....	2
C. RUANG LINGKUP.....	3
D. DESAIN SISTEM.....	4
E. TIM & TUGAS.....	6
F. TAHAPAN PELAKSANAAN.....	6
G. IMPLEMENTASI.....	7
H. SISTEM TESTING.....	14

A. ABSTRAK

Perkembangan teknologi virtualisasi telah memungkinkan penggunaan mesin virtual (Virtual Machine/VM) secara luas untuk berbagai keperluan, mulai dari pengembangan hingga produksi. Namun, pemantauan kinerja VM menjadi tantangan tersendiri karena kompleksitas dan kebutuhan akan alat yang efisien dan akurat. Proyek ini mengusulkan implementasi sistem pemantauan mesin virtual berbasis VirtualBox menggunakan Prometheus dan Grafana. Prometheus digunakan sebagai sistem monitoring yang mengumpulkan dan menyimpan metrik dari VM, sedangkan Grafana digunakan untuk visualisasi data dan sistem alert. Visualisasi data tersebut dalam bentuk dashboard yang informatif dan mudah dipahami. Implementasi ini diharapkan dapat memberikan pemantauan real-time yang komprehensif terhadap berbagai metrik kinerja VM, seperti penggunaan CPU, Memory, Disk, dan Jaringan. Dengan demikian, administrator sistem dapat mengidentifikasi dan mengatasi masalah kinerja dengan lebih cepat dan tepat. Studi ini akan mencakup proses instalasi, konfigurasi, serta pengujian efektivitas dan efisiensi sistem pemantauan yang diusulkan. Hasil dari Proyek ini diharapkan dapat meningkatkan manajemen dan pemeliharaan infrastruktur virtualisasi secara keseluruhan.

B. PENDAHULUAN

Perkembangan teknologi informasi telah membawa perubahan signifikan dalam cara organisasi mengelola infrastruktur TI mereka. Salah satu inovasi terpenting dalam beberapa dekade terakhir adalah virtualisasi, yang memungkinkan satu perangkat keras fisik menjalankan beberapa mesin virtual (VM) secara bersamaan. VirtualBox, sebagai salah satu perangkat lunak virtualisasi open-source yang populer, telah digunakan secara luas untuk berbagai keperluan, mulai dari pengembangan perangkat lunak hingga pengujian dan produksi.

Meskipun virtualisasi menawarkan banyak keuntungan, seperti penghematan biaya, fleksibilitas, dan efisiensi sumber daya, pemantauan dan manajemen kinerja VM tetap menjadi tantangan yang signifikan. Kinerja VM yang optimal sangat penting untuk memastikan aplikasi dan layanan yang berjalan di atasnya tetap responsif dan andal. Oleh karena itu, diperlukan alat pemantauan yang mampu memberikan informasi real-time mengenai berbagai aspek kinerja VM, termasuk penggunaan CPU, memory, disk, dan jaringan.

Prometheus dan Grafana adalah dua alat yang dapat digunakan bersama untuk membangun sistem pemantauan yang kuat dan efektif. Prometheus, sebagai sistem monitoring dan alerting berbasis open-source, memungkinkan pengumpulan, penyimpanan, dan analisis matrik secara efisien. Di sisi lain, Grafana menyediakan

platform visualisasi yang kuat, memungkinkan data yang dikumpulkan oleh Prometheus ditampilkan dalam bentuk dashboard yang informatif dan mudah dipahami.

Proyek ini bertujuan untuk mengembangkan sistem pemantauan VM berbasis VirtualBox menggunakan Prometheus dan Grafana. Implementasi ini diharapkan dapat memberikan solusi yang komprehensif dan real-time untuk pemantauan kinerja VM, membantu administrator sistem dalam mengidentifikasi dan mengatasi masalah kinerja dengan lebih cepat dan efektif. Selain itu, studi ini juga akan mencakup proses instalasi, konfigurasi, serta evaluasi efektivitas dan efisiensi dari sistem pemantauan yang diusulkan.

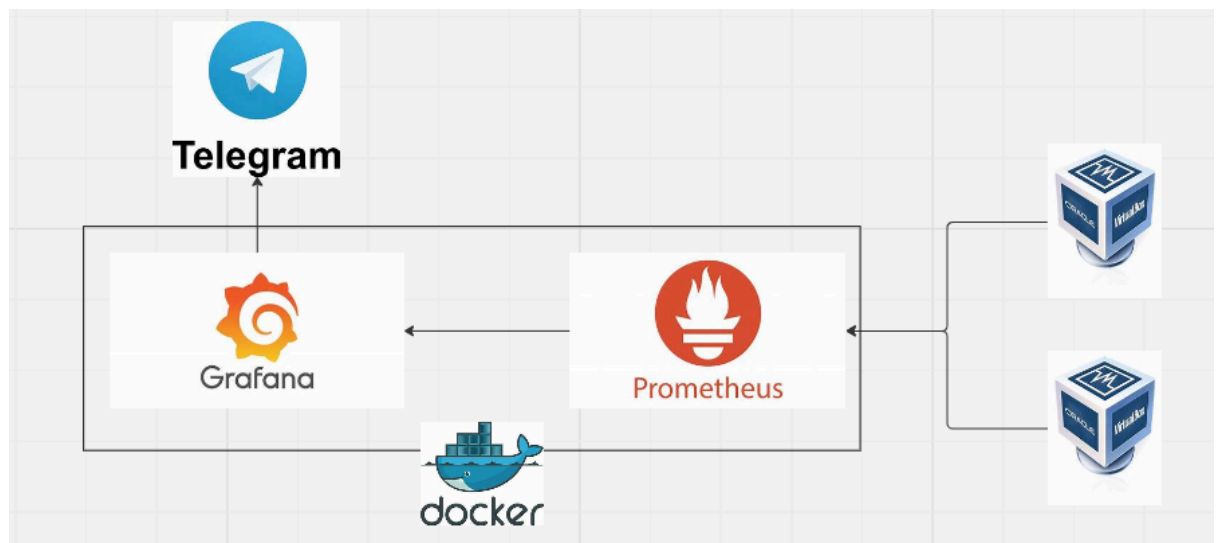
Dengan adanya sistem pemantauan yang diusulkan, diharapkan organisasi dapat meningkatkan manajemen infrastruktur virtualisasi mereka, memastikan kinerja yang optimal dan stabilitas operasional yang lebih baik

C. RUANG LINGKUP

Ruang lingkup laporan ini mengarah pada pembangunan sistem monitoring untuk CPU, Memory, Disk, Network di mesin virtual menggunakan Grafana dan Prometheus, yaitu sebagai berikut:

1. Penelitian ini dibatasi pada lingkungan mesin virtual yang digunakan dalam infrastruktur Teknologi Informasi.
2. Penelitian menggunakan metode implementasi dan integrasi alat monitoring Prometheus untuk pengumpulan data dan Grafana untuk visualisasi data.
3. Fokus utama adalah pada pemantauan kinerja Memori, Disk, Jaringan dan CPU secara real-time guna mendeteksi masalah sejak dini dan mengelola sumber daya secara efisien

D. DESAIN SISTEM



1. Debian (Virtual Machine)
 - Sistem operasi Debian berjalan pada mesin virtual. Mesin ini adalah target yang akan dimonitor.
2. Prometheus
 - Prometheus berfungsi sebagai alat pengumpul data (data collection). Prometheus diinstal pada Docker
 - Prometheus dikonfigurasi untuk mengumpulkan metrik dari mesin virtual melalui port 9090.
3. Node Exporter
 - Node Exporter mengumpulkan berbagai metrik dari sistem operasi dan perangkat keras, termasuk CPU, memory, disk, dan jaringan.
 - Metrik-metrik ini diperoleh dari `/proc` dan `/sys` filesystem di Linux
 - Node Exporter adalah agen yang berjalan pada mesin virtual Debian. Ia mengumpulkan data metrik seperti penggunaan CPU dan RAM dan mengirimkannya ke Prometheus melalui port 9100.
4. Grafana
 - Grafana adalah alat visualisasi data yang mengambil data dari Prometheus. Grafana diinstal pada docker
 - Dengan Grafana, administrator jaringan dapat membuat dashboard yang menampilkan metrik penggunaan RAM dan CPU secara real-time.
 - Dengan Grafana, administrator jaringan dapat membuat alert yang berkomunikasi pada Telegram.
5. Telegram
 - Telegram berfungsi untuk menerima notifikasi jika terjadi kondisi abnormal pada metrik yang dipantau.
6. Docker
 - Tempat diinstalnya Grafana dan Prometheus

DIAGRAM ALIR

1. Node Exporter berjalan di mesin virtual Debian dan mengumpulkan metrik.
|
v
2. Node Exporter mengirim metrik ke Prometheus melalui port 9100.
|
v
3. Prometheus (berjalan di Docker) mengumpulkan dan menyimpan data metrik dari Node Exporter.
|
v
4. Grafana (berjalan di Docker) mengambil data dari Prometheus melalui port 9090 untuk visualisasi.
|
v
5. Grafana menampilkan metrik dalam dashboard real-time.
|
v
6. Jika metrik mencapai kondisi abnormal, Grafana mengirim notifikasi ke Telegram.

E. TIM & TUGAS

Nufus Akmalul Mafaza
Moch. Irham Kafi Billah
Septian Achmad Rajabbi

Eksekutor	Task
Kafi, Septian, Nufus	Setup Node Exporter
Kafi, Septian, Nufus	Setup prometheus
Kafi, Septian, Nufus	Setup Grafana
Kafi	Integrasi Node Exporter ke Prometheus
Kafi	Menambahkan Dashboard Resource Monitoring di Grafana
Kafi, Septian, Nufus	Memilih Query Data yang akan diTampilkan di Dashboard
Nufus	Setup Alerting Contact Point dan bot Telegram

Kafi, Septian, Nufus	Memilih Query Data yang akan di set ke Alerting Rules
Nufus	Setup Alerting Alert Rules
Kafi, Septian, Nufus	Membuat Laporan Hasil dan PPT

F. TAHAPAN PELAKSANAAN

Tanggal	Keterangan
27 Mei 2024	Merancang Project Chart & Pembagian Management
28 Mei 2024	Instalasi & Setup Prometheus dan Layanannya
29 Mei 2024	Instalasi & Setup Grafana dan Layanannya
30 Mei 2024	Implementasi dari Integrasi Resource Monitor CPU dan RAM di VM Menggunakan Prometheus dan Grafana
1 Juni 2024	set up Alerting
2 Juni 2024	Pembuatan Laporan & Finalisasi Laporan

G. IMPLEMENTASI

Berikut adalah langkah-langkah untuk mengatur Node Exporter, Prometheus, dan Grafana dengan menggunakan Docker Compose dalam bahasa Indonesia:

Langkah-langkah Pengaturan :

1. Pengaturan Node Exporter

a. Download dan Eksekusi Node Exporter

1. Jalankan perintah berikut di terminal Debian untuk menginstall Node-Exporter:

```
wget
```

```
https://github.com/prometheus/node\_exporter/releases/download/v1.3.1/node\_exporter-1.3.1.linux-amd64.tar.gz
```

```
tar xvfz node_exporter-1.3.1.linux-amd64.tar.gz
```

2. Jalankan perintah berikut di terminal Debian untuk menjalankan Node-Exporter

```
cd node_exporter-1.3.1.linux-amd64
```

```
`./node_exporter`
```

b. Verifikasi Node Exporter:

Pastikan Node Exporter berjalan dengan benar dengan menjalankan perintah berikut dari dalam VM pada Browser:

```
curl http://localhost:9100/metrics
```

melihat output seperti berikut:

```
# HELP go_gc_duration_seconds A summary of the GC invocation
durations.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 3.8996e-05
go_gc_duration_seconds{quantile="0.25"} 4.5926e-05
go_gc_duration_seconds{quantile="0.5"} 5.846e-05
# dll.
```

2. Pengaturan Prometheus dan Grafana

a. Buat file `docker-compose.yml`:

Buat folder kerja baru dan di dalamnya buat file `docker-compose.yml` dengan isi sebagai berikut:

```
version: "3.3"
networks:
  mynetwork:
    external: true
services:
  prometheus:
    image: prom/prometheus:v2.34.0
    container_name: prometheus
    networks:
      - mynetwork
    ports:
      - "9090:9090"
    volumes:
      - ${PWD}/prometheus.yml:/etc/prometheus/prometheus.yml
  grafana:
    user: "472"
    image: grafana/grafana:8.4.4-ubuntu
    container_name: grafana
    volumes:
      - ${PWD}/docker_data/grafana_data:/var/lib/grafana
    networks:
      - mynetwork
    ports:
      - "3000:3000"
```



```
GNU nano 7.2                                docker-compose.yml *
```

```
version: "3.3"
networks:
  mynetwork:
    external: false
services:
  prometheus:
    image: prom/prometheus:v2.34.0
    container_name: prometheus
    networks:
      - mynetwork
    ports:
      - "9090:9090"
    volumes:
      - ${PWD}/prometheus.yml:/etc/prometheus/prometheus.yml
  grafana:
    user: "472"
    image: grafana/grafana:8.4.4-ubuntu
    container_name: grafana
    volumes:
      - ${PWD}/docker_data/grafana_data:/var/lib/grafana
    networks:
      - mynetwork
    ports:
      - "3000:3000"
```

**note perhatikan tabulasi dan juga spasi karena salah tab dapat menyebabkan tidak bekerjanya 'docker-composer.yml'

- b. Buat file 'prometheus.yml':

Di dalam folder yang sama dengan 'docker-compose.yml', buat file 'prometheus.yml' dengan isi sebagai berikut:

global:

```
scrape_interval: 10s
```

```
scrape_configs:
```

```
- job_name: myvm
```

```
  metrics_path: /metrics
```

```
  static_configs:
```

```
    - targets: ['<vm-host>:9100']
```

```
    - targets: ['<vm-host>:9100']
```

```
irhamkafi@sysadmin-3122600009: /usr...  x  irhamkafi@sysadmin-3122600009: /usr...  x  ▾
```

```
GNU nano 7.2                                prometheus.yml *
```

```
$global:
  scrape_interval: 10s
scrape_configs:
  - job_name: myvm
    metrics_path: /metrics
    static_configs:
      - targets: [ '192.168.212.110:9100' ]
      - targets: [ '192.168.212.203:9100' ]
```

3. Jalankan Prometheus dan Grafana dengan Docker Compose:**

- Dari dalam folder yang berisi `docker-compose.yml`, jalankan perintah berikut:

`docker-compose up -d`

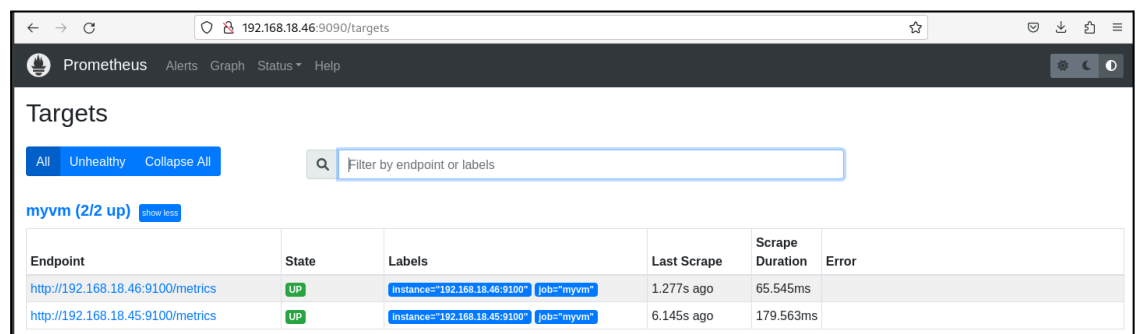
```
root@sysadmin-3122600009:/prometheus_grafana# docker-compose up -d
[+] Running 2/2
 ✓ Container grafana      Started
 ✓ Container prometheus   Started
```

4. **Verifikasi Prometheus:**

- Buka browser dan akses URL berikut untuk memastikan bahwa Prometheus dapat terhubung dengan Node Exporter:

<http://localhost:9090/targets>

- melihat halaman web dengan status "UP".



Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://192.168.18.46:9100/metrics	UP	instance="192.168.18.46:9100" job="myvm"	1.277s ago	65.545ms	
http://192.168.18.45:9100/metrics	UP	instance="192.168.18.45:9100" job="myvm"	6.145s ago	179.563ms	

5. Konfigurasi Data Source Prometheus di Grafana:

- Buka Grafana di browser pada URL berikut: <http://localhost:3000>, Masuk ke Grafana.

- Tambahkan Data Source:

- Menu kiri -> Configuration -> Data sources -> Add data source -> Prometheus

- Isi "Name" untuk data source.

- Isi URL HTTP untuk instance Prometheus, dalam hal ini: <http://prometheus:9090>

- Klik "Save & test".

- melihat pesan "Data source is working".

- Tambahkan Dashboard Grafana:

- Menu kiri -> "+" -> Import

- Di field "Import via grafana.com"

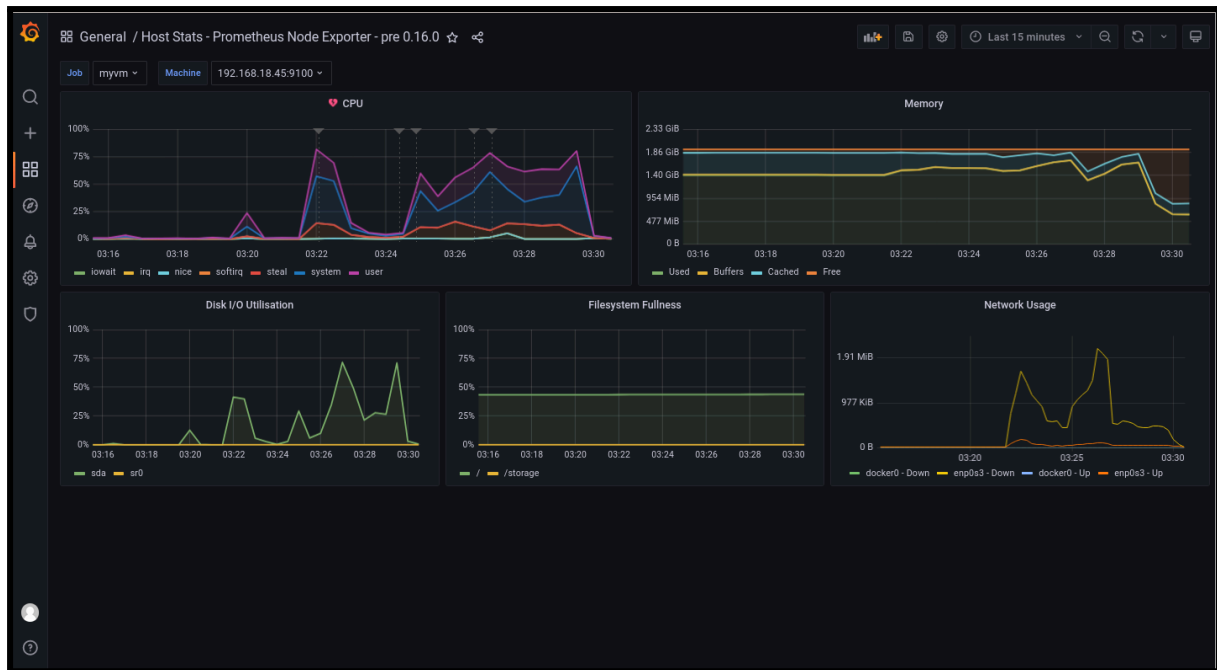
- masukkan URL berikut: <https://grafana.com/grafana/dashboards/718> dan tekan "Load".

- Pilih data source Prometheus yang telah dibuat sebelumnya.

- Klik "Import".

6. Verifikasi Dashboard:

Buka dashboard yang telah dibuat dan seharusnya melihat metrik penggunaan secara real-time.



Pada dashboard menampilkan CPU, Memory, Disk, File System dan Network:

1. CPU

- iowait : Waktu yang dihabiskan oleh CPU untuk menunggu operasi I/O (input/output) selesai. Ini mencakup waktu yang CPU habiskan menunggu disk, jaringan, atau perangkat lain untuk menyelesaikan tugas I/O.
- irq : Waktu yang dihabiskan oleh CPU untuk menangani interrupt hardware (hardware interrupts). Interrupt hardware adalah sinyal dari perangkat keras yang memberitahu CPU bahwa tindakan tertentu memerlukan perhatian segera.
- softirq : Waktu yang dihabiskan oleh CPU untuk menangani interrupt software (software interrupts). Ini adalah jenis interrupt yang dipicu oleh perangkat lunak atau sistem operasi untuk menangani tugas yang memerlukan perhatian segera, tetapi tidak cukup mendesak untuk ditangani oleh interrupt hardware.
- steal : Waktu yang dihabiskan oleh CPU menunggu virtual CPU lain dalam lingkungan virtualisasi. Ini terjadi ketika hypervisor (software yang mengelola mesin virtual) mengambil waktu CPU dari satu mesin virtual untuk diberikan ke mesin virtual lain.
- system : Waktu yang dihabiskan oleh CPU untuk menjalankan kode kernel sistem operasi. Ini termasuk waktu yang dihabiskan untuk tugas-tugas seperti pengelolaan memori, penjadwalan proses, dan manajemen perangkat keras.
- user : Waktu yang dihabiskan oleh CPU untuk menjalankan kode aplikasi (user space). Ini adalah waktu yang dihabiskan untuk menjalankan aplikasi dan proses pengguna.

2. Memory

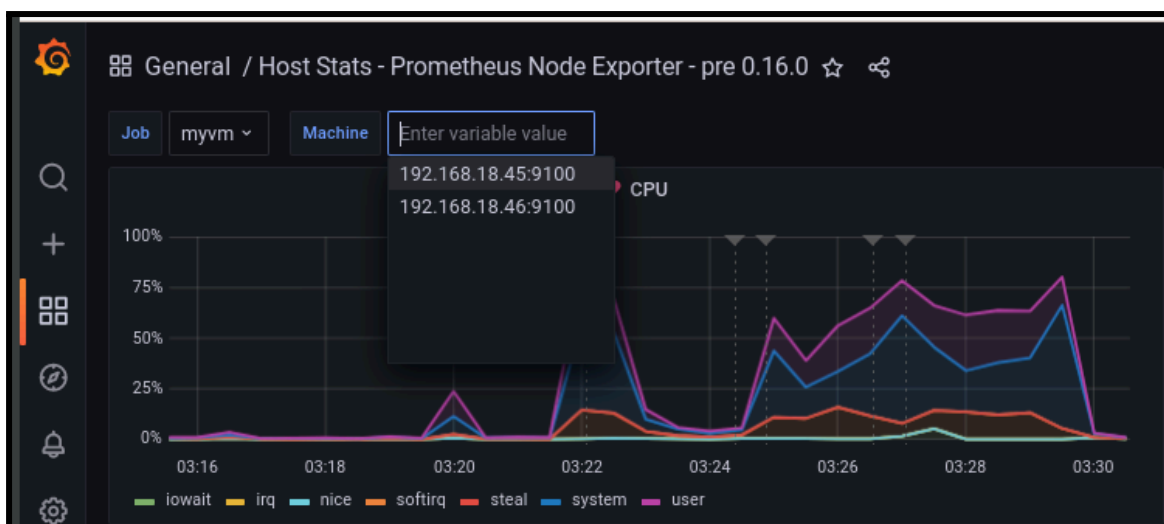
- Used : Memori yang saat ini digunakan oleh sistem, termasuk semua proses yang sedang berjalan dan kernel. Ini biasanya merupakan total memori dikurangi memori yang tersedia (free, buffers, cached)
- Buffers : Memori yang digunakan oleh kernel sebagai buffer untuk operasi I/O. Buffer digunakan untuk menyimpan data sementara saat sedang dipindahkan antara memori dan perangkat keras.
- Cached : Memori yang digunakan oleh sistem untuk menyimpan data yang sering diakses dari disk untuk mempercepat akses. Ini mencakup cache halaman file dan cache dari data disk lainnya.
- Free : Memori yang sepenuhnya tidak digunakan oleh sistem, yang tersedia untuk dialokasikan untuk proses baru atau penggunaan lain. Ini adalah memori yang benar-benar bebas.

3. Disk

- sda : nama default untuk disk SCSI pertama yang terdeteksi oleh sistem. Pada sistem Linux, **sda** biasanya merujuk ke hard disk pertama atau SSD (Solid State Drive).
- sr0 : nama default untuk drive CD/DVD pertama yang terdeteksi oleh sistem. Pada sistem Linux, **sr0** biasanya merujuk ke optical drive pertama.

4. Network

- enps03 : salah satu dari banyak antarmuka jaringan yang terdapat pada sistem. Biasanya, antarmuka jaringan pada sistem Linux disebut dengan nama yang dimulai dengan "en" diikuti dengan nomor yang menunjukkan urutan antarmuka



Terdapat 2 virtualbox yang bisa dipantau tinggal memilih pada machine.

ALERTING:

1. Buat bot telegram dengan botFather dan Get ID bot
2. Konfigurasi contact points

Home > Alerting > Contact points

Name *

Testing Grafana Alert

Integration

Telegram

Telegram Bot API Token

Configured Clear

Chat ID

Integer Telegram Chat Identifier

3. Konfigurasi Notification policies

- pada default policies tekan button “...” pada bagian kanan lalu pilih edit

Notification Policies Mute Timings

Search by matchers Search by contact point

Search Choose

Default policy All alert instances will be handled by the default policy if no other matching policies are found. + New child policy

0 Instances @ Delivered to Testing Grafana Alert @ Grouped by grafana_folder, alertname Wait 30s to group instances, 1m before sending updates

Auto-generated policies

0 Instances @ Delivered to Testing Grafana Alert Inherited 3 properties

Edit

Export

- Ubah default contact point menjadi Testing Grafana Alert

Edit notification policy

Default contact point

Testing Grafana Alert or [Create a contact point](#)

Group by

Group alerts when you receive a notification based on labels.

grafana_folder alertname

Timing options

Cancel Update default policy

4. Buat Alert

- masukan alert rule name
- masukan query yang mengarah pada data yang diinginkan
- atur juga expression
 - reduce, untuk menerima value dari A

Takes one or more time series returned from a query or an expression and turns each series into a single number.

Input A

Function Last Mode Strict

{instance="localhost:9100"} 3.79419

1 series

2. Threshold, untuk menerima value dari B lalu dibuat sebuah kondisi yang mana jika nilai lebih dari 70 maka terjadi firing (alart akan mengirimkan pemberitahuan)

The screenshot shows the 'Threshold' alert condition configuration. At the top, there's a tab labeled 'C' and a button 'Alert condition' with a trash icon. Below this, a description states: 'Takes one or more time series returned from a query or an expression and checks if any of the series match the threshold condition.' The 'Input' is set to 'B'. The condition is 'IS ABOVE' with a value of '70'. At the bottom, there's a 'Custom recovery threshold' toggle switch which is currently turned off.

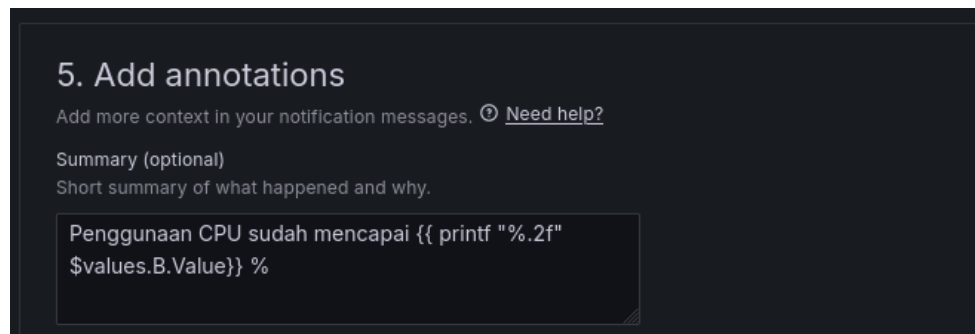
d. Konfigurasi evaluation behavior

The screenshot shows the '3. Set evaluation behavior' configuration page. It includes a 'Folder' dropdown set to 'Alert' and an 'Evaluation group' dropdown set to 'Default'. Below these, it states 'All rules in the selected group are evaluated every 1m.' The 'Pending period' is set to '1m'. There is a 'Pause evaluation' toggle switch which is turned off. At the bottom, there's a link to 'Configure no data and error handling'.

e. Konfigurasi labels and notification

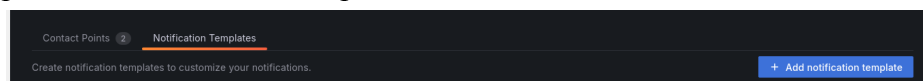
The screenshot shows the '4. Configure labels and notifications' configuration page. It has a 'Labels' section with a table for adding labels (key, value, and a trash icon). Below this is an 'Add label' button. The 'Notifications' section includes a 'Select contact point' button and a 'Use notification policy' button. It also shows the 'Alert manager' set to 'grafana' and a 'Contact point' dropdown set to 'Testing Grafana Alert'.

f. Konfigurasi annotation

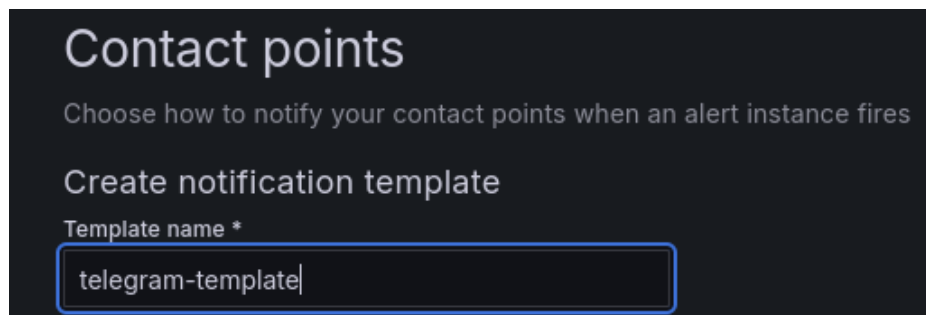


5. Konfigurasi template notifikasi

- Masuk ke contact points lalu pilih notification template
- pilih “add notification template”



- berikan nama pada bagian template name



- pada bagian content code berikut



- tekan save template

H. SISTEM TESTING

Memory Alert Rule Ketika Threshold Di Atas 10%

- a. **Tujuan:** Memastikan bahwa alert akan firing ketika penggunaan memori mencapai atau melebihi 10% dari threshold yang ditentukan.
- b. **Langkah-langkah:**
- Tentukan threshold penggunaan memori sebesar 10% pada konfigurasi Prometheus.
 - Buat aturan alert pada Prometheus yang menyatakan bahwa jika penggunaan memori melebihi 10%, maka alert harus firing.
 - Verifikasi aturan alert dengan memantau sistem hingga penggunaan memori melebihi 10%.
 - Periksa bahwa alert firing terjadi sesuai dengan kondisi yang telah ditentukan.
 - Pastikan notifikasi diterima melalui sistem yang telah diintegrasikan, seperti Grafana atau bot Telegram.

