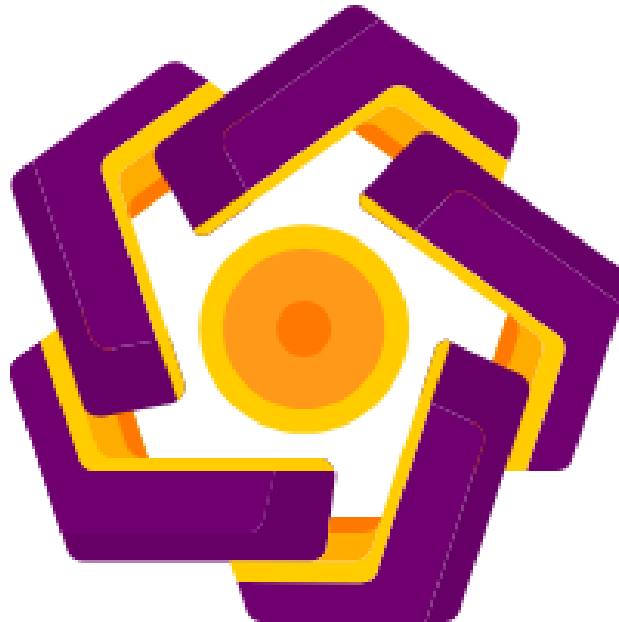


# **LAPORAN PRAKTIKUM MATA KULIAH DEVELOPMENT OPERATION**



**Nama** : Frumentios David Ivan Satria  
**NIM** : 23.01.5085  
**Kelas** : D3 Teknik Informatika 03  
**Praktikum ke-** : 10

**Pengampu :**  
Hastari Utama, S.Kom., M.Cs.

**FAKULTAS ILMU KOMPUTER  
UNIVERSITAS AMIKOM YOGYAKARTA  
SLEMAN 2025**

## LANGKAH-LANGKAH TUGAS :

Semua SourceCode saya ada di sini :

[https://github.com/c0l0sseum/devops\\_laprak10\\_5085.git](https://github.com/c0l0sseum/devops_laprak10_5085.git)

### TUGAS 1

- 1) Buat folder khusus saya menamainya *devops\_laprak10\_5085* kemudian didalamnya buat file dengan nama *docker-compose.yml*, *loki-config.yaml*, *promtail-config.yaml* .

```
david ~ master > ~ > kuliah > devops > devops_laprak10_5085 > pwd
/home/david/kuliah/devops/devops_laprak10_5085
david ~ master > ~ > kuliah > devops > devops_laprak10_5085 > ls
docker-compose.yml loki-config.yaml promtail-config.yaml
```

- 2) Isilah *docker-compose.yml* seperti dibawah ini :

```
home > david > kuliah > devops > devops_laprak10_5085 > docker-compose.yml
1 services:
2   loki:
3     image: grafana/loki:latest
4     ports:
5       - "3100:3100"
6     command: -config.file=/etc/loki/config.yaml
7     volumes:
8       - ./loki-config.yaml:/etc/loki/config.yaml
9     networks:
10      - loki-net
11
12   promtail:
13     image: grafana/promtail:latest
14     command: -config.file=/etc/promtail/config.yaml
15     volumes:
16       - ./promtail-config.yaml:/etc/promtail/config.yaml
17     depends_on:
18       - loki
19     networks:
20       - loki-net
21
22   grafana:
23     image: grafana/grafana:latest
24     ports:
25       - "3000:3000"
26     environment:
27       - GF_SECURITY_ADMIN_PASSWORD=admin
28     depends_on:
29       - loki
30     networks:
31       - loki-net
32
33 networks:
34   loki-net:
```

- 3) Isilah file *promtail-config.yaml* seperti dibawah ini :

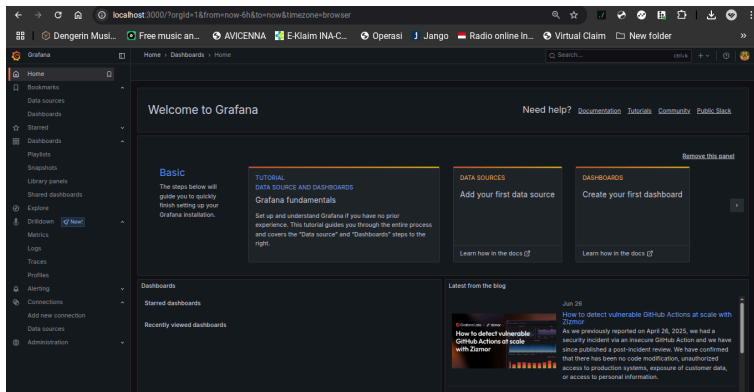
```
home > david > kuliah > devops > devops_laprak10_5085 > promtail-config.yaml
1 server:
2   http_listen_port: 9080
3   grpc_listen_port: 0
4
5 positions:
6   filename: /tmp/positions.yaml
7
8 clients:
9   - url: http://loki:3100/loki/api/v1/push
10
11 scrape_configs:
12   - job_name: system
13     static_configs:
14       - targets:
15         - localhost
16         labels:
17           job: varlogs
18           __path__: /var/log/*log
19
```

- 4) Isilah file *loki-config.yaml* dengan script berikut :

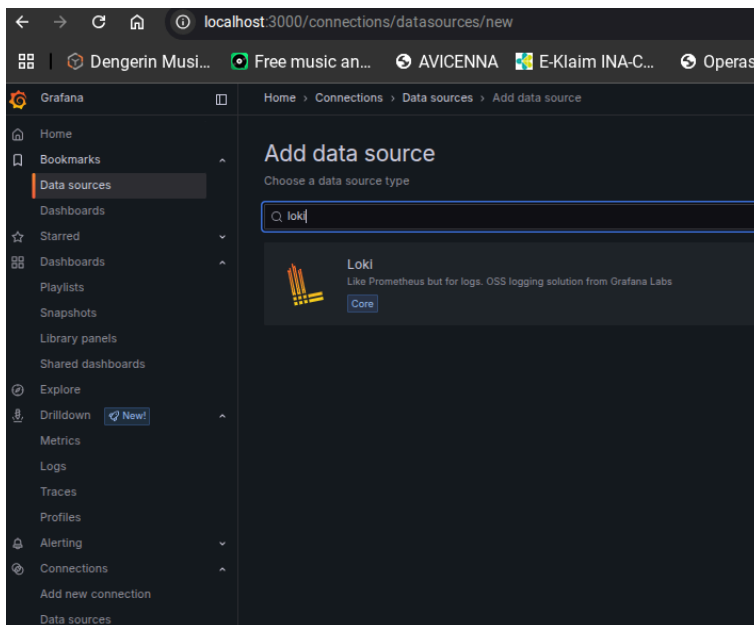
```
home > david > kuliah > devops > devops_laprak10_5085 > ! loki-config.yaml
1  auth_enabled: false
2
3  server:
4    http_listen_port: 3100
5
6  ingester:
7    lifecycler:
8      address: 127.0.0.1
9    ring:
10     kvstore:
11       store: inmemory
12     replication_factor: 1
13     final_sleep: 0s
14   chunk_idle_period: 1m
15   chunk_target_size: 1048576
16   max_chunk_age: 1m
17   wal:
18     dir: /tmp/loki/wal
19
20  schema_config:
21    configs:
22     - from: 2020-10-24
23       store: boltdb-shipper
24       object_store: filesystem
25       schema: v11
26       index:
27         prefix: index_
28         period: 24h
29
30  storage_config:
31    boltdb_shipper:
32      active_index_directory: /tmp/loki/boltdb-shipper-active
33      cache_location: /tmp/loki/boltdb-shipper-cache
34    filesystem:
35      directory: /tmp/loki/chunks
36
37  compactor:
38    working_directory: /tmp/loki/boltdb-shipper-compactor
39
40  limits_config:
41    allow_structured_metadata: false
42    reject_old_samples: true
43    reject_old_samples_max_age: 168h
44
45  table_manager:
46    retention_deletes_enabled: true
47    retention_period: 1d
48
```

- 5) Kemudian jalankan pada terminal menggunakan *docker-compose up -d* kemudian kita masuk ke Grafana <http://localhost:3000/>

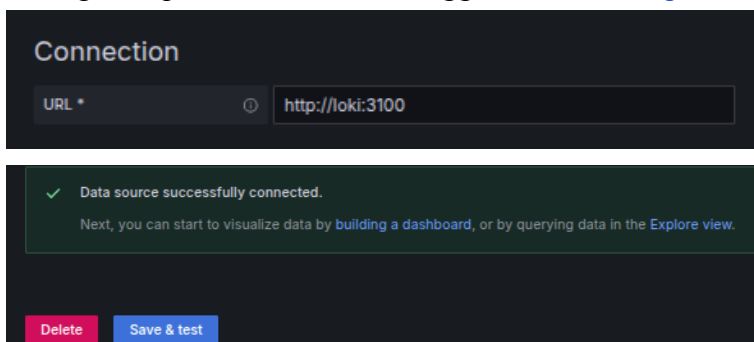
```
david@/ master ~$ docker-compose up -d
Creating network "devops_laprak10_5085_loki-net" with the default driver
Pulling loki (grafana/loki:latest)...
latest: Pulling from grafana/loki
0f8b424aa0b9: Pulling fs layer
80a8c047508a: Pulling fs layer
0f8b424aa0b9: Pull complete
80a8c047508a: Pull complete
3f4e2c586348: Pull complete
5391af8231d5: Pull complete
400fe5c6b87a: Pull complete
d557676654e5: Pull complete
f6e634b03e3b: Pull complete
f158ccafad5e: Pull complete
d02bc7a76a83: Pull complete
2e4cf50eeb92: Pull complete
1069fc2daed1: Pull complete
d558cbc252ad: Pull complete
b40161c0d83f: Pull complete
77c1282b16b6: Pull complete
3776MB/37.76MB|1 complete
ca78d010a0b3: Pull complete
Digest: sha256:a74594532ee4cc313401beedc4dd2708c43674c032084b1aeb87c14a5be1745
Status: Downloaded newer image for grafana/loki:latest
Creating devops_laprak10_5085_loki-1 ... done
Creating devops_laprak10_5085-promtail-1 ... done
Creating devops_laprak10_5085-grafana-1 ... done
```



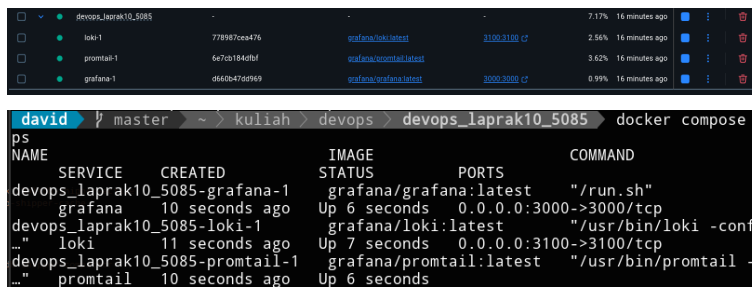
6) Masuk ke tab *Data Sources* > +add Data Source > pilih *Loki*



7) Konfigurasi pada *Connection* menggunakan link <http://loki:3100> > Save & Test



## 8) Berikut adalah hasil di Docker Desktop



The screenshot shows the Docker Desktop interface. At the top, a list of containers is displayed:

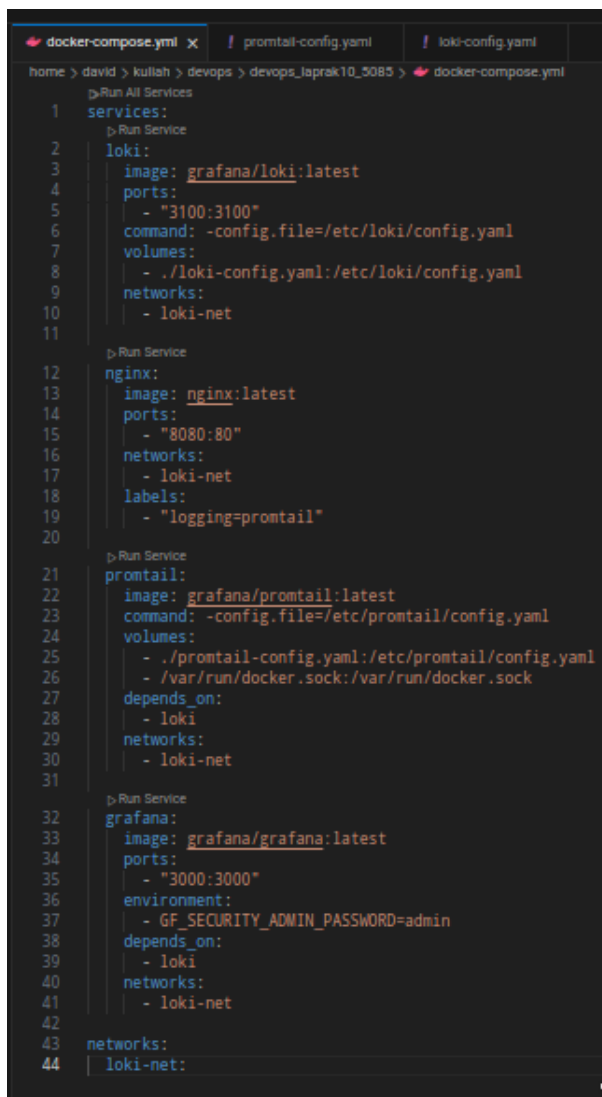
Container Name	ID	Image	Ports	Status	Uptime
devops_laprak10_5085	778987ce47f6	grafana/loki:latest	3100:3100	Up	16 minutes ago
loki-1	6e7cb184dbf1	grafana/promtail:latest	3000:3000	Up	16 minutes ago
grafana-1	d660b47d9f69	grafana/grafana:latest	3000:3000	Up	16 minutes ago

Below the container list, a terminal window shows the output of the `docker compose ps` command:

```
ps
NAME                SERVICE      CREATED      STATUS      PORTS      COMMAND
devops_laprak10_5085 grafana-1    10 seconds ago Up 6 seconds 0.0.0.0:3000->3000/tcp "/run.sh"
devops_laprak10_5085 loki-1       11 seconds ago Up 7 seconds 0.0.0.0:3100->3100/tcp "/usr/bin/loki -conf"
devops_laprak10_5085 promtail-1   10 seconds ago Up 6 seconds 0.0.0.0:3000->3000/tcp "/usr/bin/promtail -
```

## TUGAS 2

### 1) Perbaharui isi script `docker-compose.yml`

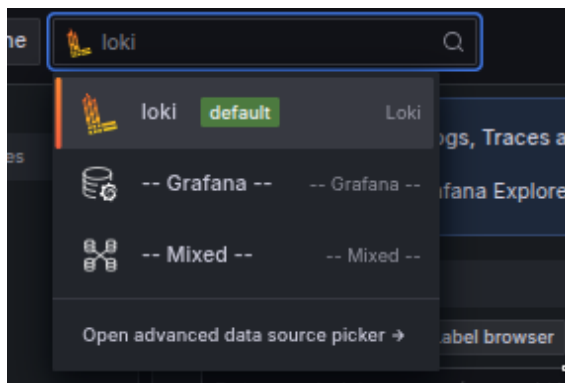


The screenshot shows a code editor with the `docker-compose.yml` file open. The file content is as follows:

```
1 services:
2   loki:
3     image: grafana/loki:latest
4     ports:
5       - "3100:3100"
6     command: -config.file=/etc/loki/config.yaml
7     volumes:
8       - ./loki-config.yaml:/etc/loki/config.yaml
9     networks:
10      - loki-net
11
12   nginx:
13     image: nginx:latest
14     ports:
15       - "8080:80"
16     networks:
17       - loki-net
18     labels:
19       - "logging=promtail"
20
21   promtail:
22     image: grafana/promtail:latest
23     command: -config.file=/etc/promtail/config.yaml
24     volumes:
25       - ./promtail-config.yaml:/etc/promtail/config.yaml
26       - /var/run/docker.sock:/var/run/docker.sock
27     depends_on:
28       - loki
29     networks:
30       - loki-net
31
32   grafana:
33     image: grafana/grafana:latest
34     ports:
35       - "3000:3000"
36     environment:
37       - GF_SECURITY_ADMIN_PASSWORD=admin
38     depends_on:
39       - loki
40     networks:
41       - loki-net
42
43 networks:
44   loki-net:
```

The screenshot shows the Grafana 7.5.10 web interface. The top navigation bar includes the Grafana logo, a search bar, and a user profile icon. The left sidebar contains a menu with options like Home, Bookmarks, Recent pages, Shared, Dashboards, Playlists, Snapshots, Library panels, Shared dashboards, and Explore (which is currently selected). The main content area displays a line graph titled 'CPU usage' with a green line representing the data. The x-axis is labeled 'Time' and ranges from 19:50 to 20:45. The y-axis is labeled 'CPU usage' and ranges from 0 to 100. The graph shows a fluctuating line with a peak around 20:00 and a dip around 20:20. The bottom status bar indicates 'Data source: Prometheus'.

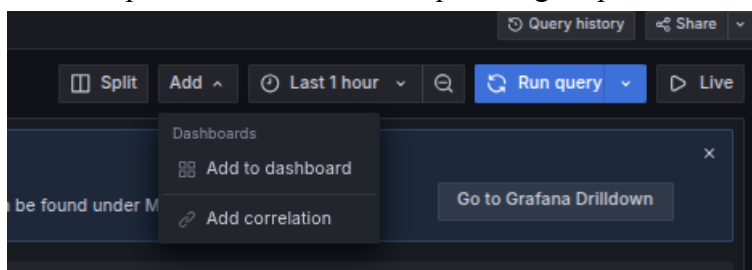
- 2) Pada bagian Outline kita pilih *loki*



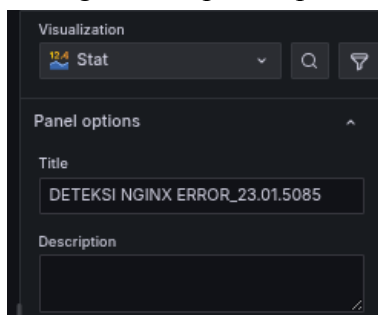
- 3) Konfigurasi log browser `count_over_time({container_name=~".*nginx.*"} |="error" [1h])` seperti dibawah ini



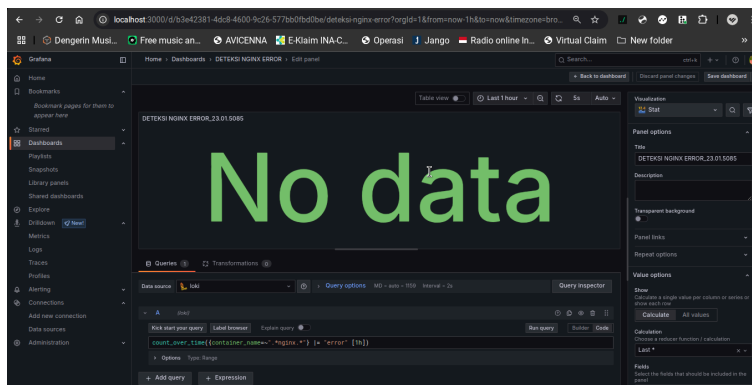
- 4) Lalu kita pilih *Add to Dashboard* pada bagian panel atas



- 5) Konfigurasilah panel option menjadi seperti dibawah, namun selera saja



6) Outputnya seperti dibawah, milik saya no data karena tidak ada error



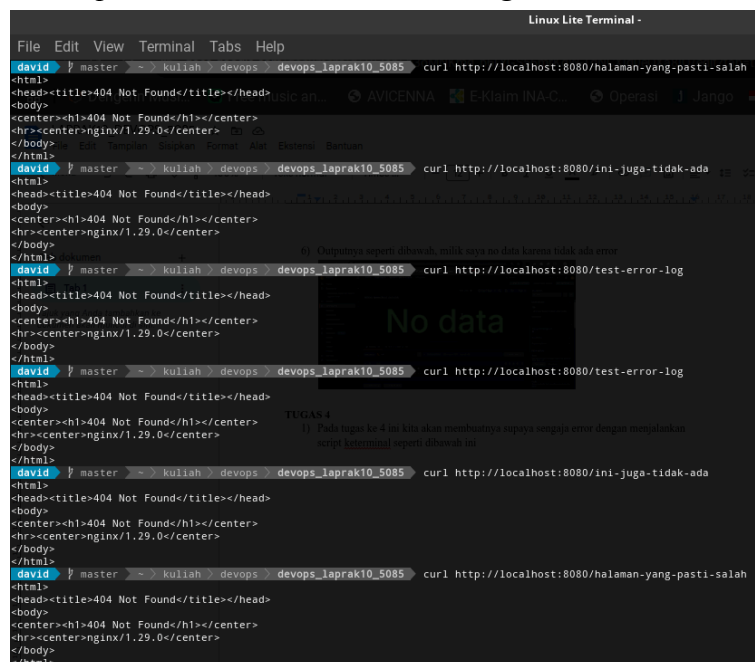
## TUGAS 4

1) Pada tugas ke 4 ini kita akan membuatnya supaya sengaja error dengan menjalankan script keterminal yang sudah masuk ke folder docker dijalankan tadi seperti dibawah ini :

*curl http://localhost:8080/halaman-yang-pasti-salah*

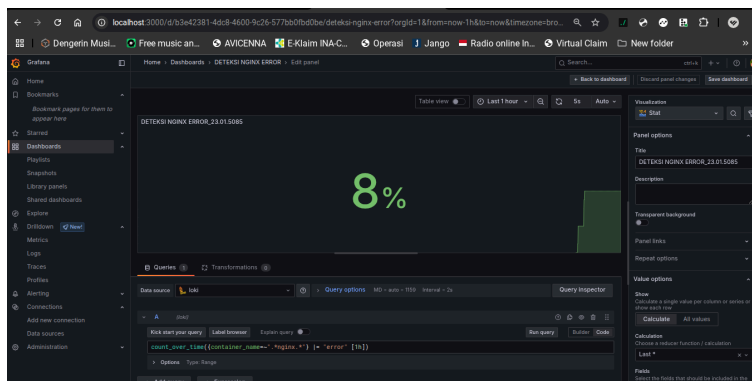
*curl http://localhost:8080/ini-juga-tidak-ada*

*curl http://localhost:8080/test-error-log*

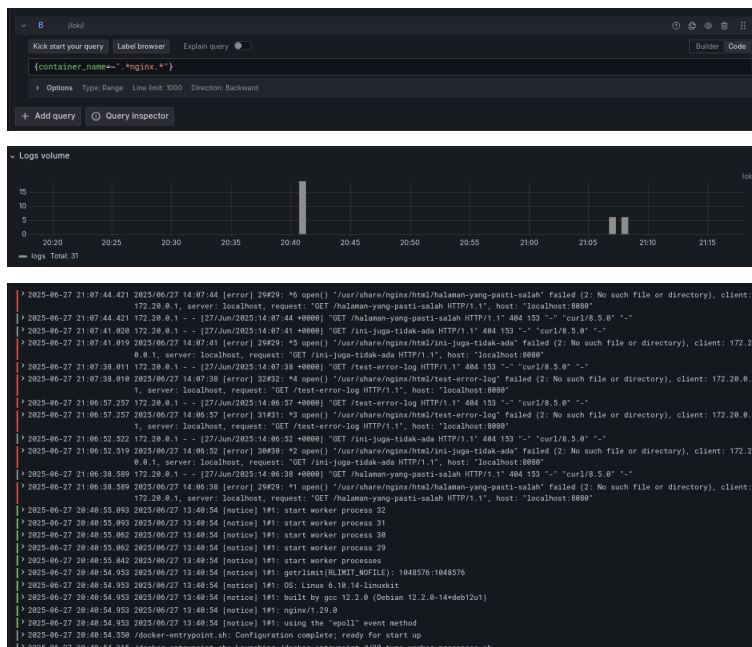




## 2) Output pada Grafana akan seperti dibawah ini



## 3) Untuk yang ini optional, kembali ke *Eksplor* kemudian bisa menambahkan query pada Log Browser dengan `{container_name=~".*nginx.*"}` untuk melihat log error 404 mentah yang baru saja saya hasilkan.



## TUGAS 5

### 1) Kerangka Observasi (Pengamatan)

- Proses Awal: Instalasi awal berdasarkan modul praktikum gagal karena konfigurasi `loki-config.yaml` tidak lagi kompatibel dengan versi terbaru dari image grafana/loki:latest.
- Proses Debugging: Ditemukan serangkaian error unmarshal (karena sintaks usang/sudah lawas seperti `shared_store`) dan permission denied (karena compactor dan ingester mencoba menulis ke direktori yang tidak diizinkan). Masalah ini diatasi dengan memperbarui file konfigurasi agar sesuai standar Loki modern, termasuk secara eksplisit mendefinisikan direktori kerja di `/tmp`.

- Konfigurasi Promtail: Konfigurasi Promtail diubah dari metode *static scrape* (/var/log) menjadi *Docker Service Discovery* yang lebih dinamis dan modern, yang memungkinkan Promtail menargetkan container secara spesifik menggunakan label Docker.
- Visualisasi & Simulasi: Query LogQL berhasil memfilter dan mengagregasi data log error dari Nginx. Simulasi error menggunakan curl berhasil ditampilkan secara *real-time* (dengan jeda beberapa detik) di dashboard Grafana.

## 2) Kerangka Kesimpulan (Pelajaran yang Didapat)

- Manfaat Loki Stack: Tumpukan Loki, Promtail, dan Grafana terbukti sangat efektif untuk sentralisasi logging dalam lingkungan berbasis container. Ini memudahkan pencarian, pemfilteran, dan visualisasi log dari berbagai layanan.
- Pentingnya Konfigurasi Modern: Pelajaran terbesar adalah betapa pentingnya menjaga file konfigurasi agar tetap sesuai dengan versi perangkat lunak yang digunakan. Konfigurasi yang usang adalah penyebab utama kegagalan sistem.
- Kekuatan Label: Penggunaan label (baik di docker-compose.yml maupun di dalam query LogQL) adalah inti dari kekuatan Loki. Ini memungkinkan pemfilteran yang sangat efisien tanpa perlu mem-parsing seluruh isi log.
- Peran Setiap Komponen:
  - Docker: Menyediakan platform untuk menjalankan semua layanan secara terisolasi dan konsisten.
  - Promtail: Bertindak sebagai agen pengumpul yang cerdas, yang dapat secara dinamis menemukan dan mengirimkan log.
  - Loki: Server yang efisien untuk menyimpan dan mengindeks log berbasis label.
  - Grafana: Antarmuka yang kuat untuk visualisasi, analisis, dan pembuatan dasbor.