

\$ PromQL & Metrics

4 Golden Signals & Alert tuning thực chiến.



1 Latency (Độ trễ)

```
histogram_quantile(0.95,  
  sum by(le)(rate(<bkt>[5m])))
```

P95: 95% request nhanh hơn mức này.

```
rate(<sum>[5m]) / rate(<cnt>[5m])
```

Độ trễ trung bình toàn bộ request.

```
rate(<bkt>{le="0.3"}[5m])
```

Tần suất request hoàn tất < 300ms.

2 Traffic (Lưu lượng)

```
sum by(job)(rate(<req>[5m]))
```

RPS (req/s) tổng hợp theo service.

```
topk(5, rate(<req>[5m]))
```

Top 5 endpoint nhận nhiều tải nhất.

```
sum(rate(<net>_bytes[5m]))
```

Băng thông mạng in/out (bytes/giây).

3 Errors (Tỉ lệ lỗi)

```
sum(rate(<req>{status=~"5.."}[5m]))  
/ sum(rate(<req>[5m])) * 100
```

% lỗi 5xx trên tổng số request.

```
sum by(code)(rate(<err>[5m]))
```

Tần suất từng mã HTTP status.

```
increase(<panic>[1h]) > 0
```

Báo động app crash/panic trong 1h.

4 Saturation (Bão hoà)

```
100 - (avg(rate(<idle>[5m]))*100)
```

% CPU bận rộn trên toàn cụm.

```
(1 - <avail> / <total>) * 100
```

% RAM đã dùng theo MemAvailable.

```
(<fs_free> / <fs_size>) * 100 < 15
```

Ổ cứng còn trống dưới 15%.

5 Alert Tuning (Chống flap)

```
for: 5m
```

Chờ vi phạm 5 phút mới bắn alert.

```
predict_linear(<fs>[4h], 86400) < 0
```

Dự báo cạn ổ đĩa trong 24h tiếp theo.

```
rate(<req>[5m]) == 0
```

Blackhole: service rớt sạch traffic.

6 Debug Target & TSDB

```
up == 0
```

Lọc nhanh target/exporter bị rớt.

```
prometheus_tsdb_head_series
```

Theo dõi cardinality series có nổ RAM?

```
curl -X POST :9090/-/reload
```

Reload config không cần restart.

4 Golden Signals & Chuẩn PromQL

Latency dùng histogram, né avg **Traffic** đo rate thay vì increase **Errors** chia % tổng để alert

Saturation giám sát queue & ram **Cửa sổ** [5m] tối thiểu 4x scrape_int **Cardinality** tránh user_id vào label

<bkt> = metric_bucket · <req> = requests · rate() cho Counter **Vibe DevOps** · Build · Run · Fix