Contents

[Slide 16: SLI/SLO/SLA 2](#_Toc130582288)

[Slide 24: Ứng dụng Prometheus 2](#_Toc130582289)

[Slide 49: Run Prometheus + Systemd 2](#_Toc130582290)

[Slide 74: Node\_Exporter install + systemd 3](#_Toc130582291)

[Slide 82: Prometheus Config 3](#_Toc130582292)

[Slide 101 TLS + Authen 3](#_Toc130582293)

[Slide 117: Sử dụng Promtool 4](#_Toc130582294)

[Slide 124: Docker prometheus 4](#_Toc130582295)

[Slide 126: Metric 4](#_Toc130582296)

[Phần 4: PromQL 5](#_Toc130582297)

[Slide 157: Data type trong PromQL 5](#_Toc130582298)

[Slide 164: Selector và Match 5](#_Toc130582299)

[Slide 174: Modifier offset (giá trị quá khứ) 5](#_Toc130582300)

[Labs Bài tập: PromQL Selectors, Matchers, and Modifiers 5](#_Toc130582301)

[Slide 184: Operator ( + - \* / và so sánh ==, !== , > , < , >=, <=) 6](#_Toc130582302)

[Slide 198: Vector matching 6](#_Toc130582303)

[Slide 213: Aggergation (nhóm) 6](#_Toc130582304)

[Labs Bài tập: Operators, Vector Matching, Aggregators 6](#_Toc130582305)

[Slide 222: Function 9](#_Toc130582306)

[Slide 237 Subquery 10](#_Toc130582307)

[Histogram 11](#_Toc130582308)

[Quantiles 11](#_Toc130582309)

[Lab bài tập – Functions, subqueries, Histogram, Summary 11](#_Toc130582310)

[Slide 260: Recording Rule 14](#_Toc130582311)

[Slide 272 Record rule naming 15](#_Toc130582312)

[Lab bài tập: Recording rule 15](#_Toc130582313)

[Slide 275 : HTTP API 15](#_Toc130582314)

[Phần 5: Dashboarding & Visualization 15](#_Toc130582315)

[Lab-Console-Template 15](#_Toc130582316)

[Phần 6: Application Instrumentation 16](#_Toc130582317)

[Slide 324: nameing metric cách đặt tên metric 16](#_Toc130582318)

[Phần 7: Service Discovery 16](#_Toc130582319)

[Slide 352: Re-labeling 16](#_Toc130582320)

[Lab-Re-labeling 17](#_Toc130582321)

[Phần 8 : Push Gateway 19](#_Toc130582322)

[Slide 375 Push-gateway 19](#_Toc130582323)

[Lab – PushGateway 20](#_Toc130582324)

[Phần 9: Alerting + AlertManager 21](#_Toc130582325)

[Slide 409: Alerting 21](#_Toc130582326)

[Slide 410: Alert with For 22](#_Toc130582327)

[Slide 416: Alert with Label 22](#_Toc130582328)

[Slide 418: Alert with annotation label 22](#_Toc130582329)

[Lab – Alertmanager Installation 22](#_Toc130582330)

[Slide 422: Alertmanager 23](#_Toc130582331)

[Slide 427: Cài đặt Alertmanager 23](#_Toc130582332)

[VD2: về config route trong prometheus 24](#_Toc130582333)

[Slide 443: Default route 24](#_Toc130582334)

[Slide 446: Sub-route 24](#_Toc130582335)

[Slide 449: Continue alertmanager (multi receiver) 24](#_Toc130582336)

[Slide 450 Group\_by trong Alertmanager 25](#_Toc130582337)

[Slide 452: Groupwait, group\_interval, repeat\_interval 25](#_Toc130582338)

[Slide 458: Receiver and notifiers 25](#_Toc130582339)

[Slide 461 Reveiver global config 26](#_Toc130582340)

[Slide 463: chỉnh Nội dung tin cảnh báo 26](#_Toc130582341)

[Slide 474: Silences 26](#_Toc130582342)

[Lab - AlertManager 2 26](#_Toc130582343)

[Phần 10: Monitoring Kubernetes 27](#_Toc130582344)

[Lab – Kubernetes & Prometheus 27](#_Toc130582345)

[Note querry 31](#_Toc130582346)

# Phần 2: Nguyên tắc cơ bản về giám sát

### Slide 16: SLI/SLO/SLA

I=indicator (Chỉ mức về trễ request, error rate, thông lượng....)

O=Object – target value or range for an SLI

A=Agreement – thỏa thuận đáp ứng SLO (99,99%)

### Slide 24: Ứng dụng Prometheus

* Vẽ đồ thị
* Cảnh báo
* Đưa ra phán đoán dựa theo đồ thị

Prometheus hoạt động tốt khi giám sát: CPU/ram/disk/service uptime/number exception / latency /pending request

# Phần 3: Cài đặt prometheus / Giải thích về metrics

### Slide 49: Run Prometheus + Systemd

Link tải toàn bộ các gói liên quan đến Prometheus <https://prometheus.io/download/>

wget <https://github.com/prometheus/prometheus/releases/download/v2.41.0/prometheus-2.41.0.linux-amd64.tar.gz>

tar -xvzf prometheus-2.41.0.linux-amd64.tar.gz

Trong thư mục giải nén có 3 file

* prometheus : File chạy
* prometheus.yml : File config
* promtool: file tools

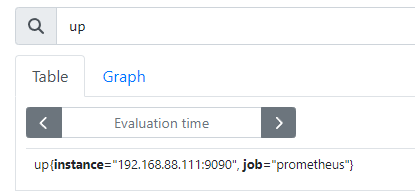
Ta sửa file config

static\_configs:

- targets: ["192.168.88.111:9090"]

Sau đó chạy

./prometheus



**Tạo Systemd tự động chạy**

**B1: Chuẩn bị user và thư mục**

useradd --no-create-home --shell /bin/false prometheus

mkdir /etc/prometheus

mkdir /var/lib/prometheus

chown prometheus:prometheus /etc/prometheus

chown prometheus:prometheus /var/lib/prometheus

**B2: Chuẩn bị file chạy**

cd prometheus-2.41.0.linux-amd64

cp prometheus /usr/local/bin/

cp promtool /usr/local/bin/

chown prometheus:prometheus /usr/local/bin/prometheus

chown prometheus:prometheus /usr/local/bin/promtool

**B3: Chuẩn bị thư mục console, thư viện**

cp -r consoles /etc/prometheus

cp -r console\_libraries /etc/prometheus

chown -R prometheus:prometheus /etc/prometheus/consoles

chown -R prometheus:prometheus /etc/prometheus/console\_libraries

B4: Chuẩn bị file config

cp prometheus.yml /etc/prometheus/prometheus.yml

vi /etc/systemd/system/prometheus.service

[Unit]

Description=Prometheus

Wants=network-online.target

After=network-online.target

[Service]

User=prometheus

Group=prometheus

Type=simple

ExecStart=/usr/local/bin/prometheus \

--config.file /etc/prometheus/prometheus.yml \

--storage.tsdb.path /var/lib/prometheus/ \

--web.console.templates=/etc/prometheus/consoles \

--web.console.libraries=/etc/prometheus/console\_libraries \

--web.enable-lifecycle

[Install]

WantedBy=multi-user.target

systemctl daemon-reload

systemctl start prometheus

Restart Prometheus:

Cách 1: systemctl restart prometheus

Cách 2: killall –HUP prometheus\_pid

Cách 3: gửi POST vào http://<prometheus\_ip>/-/reload . Nhưng yêu cầu systemctl thêm cờ --web.enable-lifecycle. Khi reload ta có thể check log message, sẽ thấy reload. curl -XPOST <http://192.168.88.111:9090/-/reload>

### Slide 74: Node\_Exporter install + systemd

wget <https://github.com/prometheus/node_exporter/releases/download/v1.5.0/node_exporter-1.5.0.linux-amd64.tar.gz>

tar -xvzf node\_exporter-1.5.0.linux-amd64.tar.gz

cd node\_exporter-1.5.0.linux-amd64

cp node\_exporter /usr/local/bin/

useradd --no-create-home --shell /bin/false node\_exporter

chown node\_exporter:node\_exporter /usr/local/bin/node\_exporter

vi /etc/systemd/system/node\_exporter.service

[Unit]

Description=Node Exporter

Wants=network-online.target

After=network-online.target

[Service]

User=node\_exporter

Group=node\_exporter

Type=simple

ExecStart=/usr/local/bin/node\_exporter

#ExecStart= /usr/local/bin/node\_exporter --web.config.file=/etc/node\_exporter/config.yml \

--web.listen-address=:9100 \

--web.telemetry-path="/metrics"

[Install]

WantedBy=multi-user.target

systemctl daemon-reload

systemctl start node\_exporter

<http://192.168.88.111:9100/metrics>

### Slide 82: Prometheus Config

scrape\_configs:

- job\_name: 'prometheus'

scrape\_interval: 5s

static\_configs:

- targets: ['localhost:9090']

- job\_name: 'node\_exporter\_1'

scrape\_interval: 30s

scrape\_timeout: 3s

scheme: http

metrics\_path: /metrics

static\_configs:

- targets: ['192.168.88.111:9100','192.168.88.112:9100']

### Slide 101 TLS + Authen

**Phần 1: Add SSL cho Prometheus và Node\_Exporter**

**Bước 1.1:** Cấu hình SSL cho node\_exporter

mkdir /etc/node\_exporter

cd /etc/node\_exporter

openssl req -new -newkey rsa:2048 -days 365 -nodes -x509 -keyout node\_exporter.key -out node\_exporter.crt -subj "/C=US/ST=VietNam/L=HaNoi/O=TuanDA/CN=localhost"

cat > /etc/node\_exporter/config.yml << EOF

tls\_server\_config:

cert\_file: node\_exporter.crt

key\_file: node\_exporter.key

EOF

chown -R node\_exporter:node\_exporter /etc/node\_exporter

**2. Sửa file systemd**

vi /etc/systemd/system/node\_exporter.service

[Unit]

Description=Node Exporter

Wants=network-online.target

After=network-online.target

[Service]

User=node\_exporter

Group=node\_exporter

Type=simple

ExecStart=/usr/local/bin/node\_exporter --web.config.file=/etc/node\_exporter/config.yml

[Install]

WantedBy=multi-user.target

3. Khởi động lại node\_exporter

systemctl daemon-reload

systemctl restart node\_exporter

Bước 1.2: Cấu hình SSL scrape cho Prometheus

Copy Public key từ node\_exporter sang cho Prometheus

scp /etc/node\_exporter/node\_exporter.crt 192.168.x.x:/etc/prometheus/node\_exporter.crt

scrape\_configs:

- job\_name: "prometheus"

static\_configs:

- targets: ["192.168.88.111:9090"]

- job\_name: "node1"

scheme: https

tls\_config:

ca\_file: /etc/prometheus/node\_exporter.crt

insecure\_skip\_verify: true

static\_configs:

- targets: ["192.168.88.111:9100"]

**Phần 2: Thêm authen basic**

Step 2.1 Tạo password và sửa node\_exporter

# htpasswd -nBC 12 "" | tr -d ':\n'

cat /etc/node\_exporter/config.yml

tls\_server\_config:

cert\_file: node\_exporter.crt

key\_file: node\_exporter.key

basic\_auth\_users:

prometheus: $2y$12$Cc1OeSGajFbIHCP3YECjpeZusdmUNFU4xNdOoMjbT/yatqUpIouPu

# systemctl restart node\_exporter

Chú ý: Account là prometheus, mật khẩu là 123. Sau đó ta vào mục Targer prometheus kiểm tra <http://192.168.88.111:9090/targets?search>=

Step 2.2 Sửa prometheus add authen

scrape\_configs:

- job\_name: "prometheus"

static\_configs:

- targets: ["192.168.88.111:9090"]

- job\_name: "node1"

scheme: https

basic\_auth:

username: prometheus

password: 123

tls\_config:

ca\_file: /etc/prometheus/node\_exporter.crt

insecure\_skip\_verify: true

static\_configs:

- targets: ["192.168.88.111:9100"]

# reload lại

### Slide 117: Sử dụng Promtool

# promtool check config /etc/prometheus/prometheus.yml

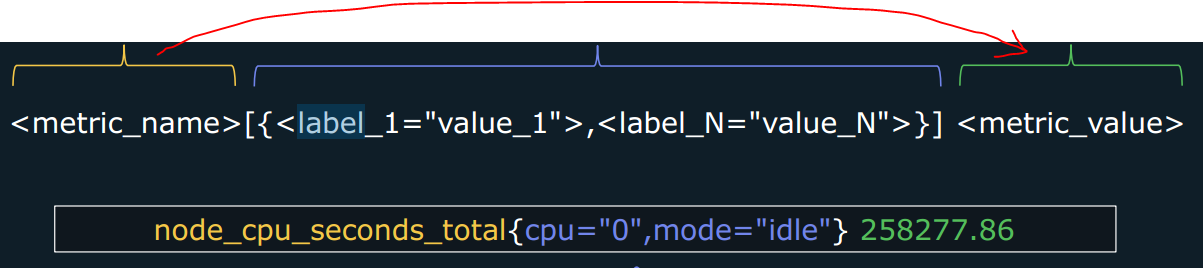
### Slide 124: Docker prometheus

docker container run -d -v /etc/prometheus/promethes.yml:/etc/prometheus/promethes.yml -p 9090:9090 prom/prometheus

<https://prometheus.io/docs/prometheus/latest/installation/>

### Slide 126: Metric

Cách đọc metric như sau:



Có 4 loại metric phổ biến:

* **Counter**: Số lần xuất hiện. VD: total request, total exception, total job execute...
* **Gauge**: Hiển thị giá trị current (có thể tăng/giảm). VD: CPU util, free memory, concurent request
* **Histogram**: đồ thị 2 chiều (ví dụ tổng request < 2s)
* **Summary**: giống với histogram

Mô tả các loại chi tiết <https://prometheus.io/docs/tutorials/understanding_metric_types/>

# Phần 4: PromQL

### Slide 157: Data type trong PromQL

* **String**: text bình thường.
* **Scala**: là float number (VD: 57.23)
* **Instant vector**: cùng 1 khoảng thời gian, hiển thị nhiều **loại** khác nhau (VD: node\_cpu\_seconds\_total)
* **Range vector**: cùng 1 mẫu (1 loại) hiểu thị nhiều thời gian khác nhau trên mẫu đó (VD: node\_cpu\_seconds\_total[3m])

### Slide 164: Selector và Match

**Selector:** Query bình thường với metric name gọi là selector (VD: node\_filesystem\_avail\_bytes)

**Match:** là thêm điều kiện = , != , =~, !~ (đây là regular exception) vào trong câu select query. Ví dụ:

* node\_filesystem\_avail\_bytes{instance="192.168.88.111:9100"}
* node\_filesystem\_avail\_bytes{device!="tmpfs"}
* node\_filesystem\_avail\_bytes{device=~"/dev/sda.\*"} Tìm với sda.\*
* node\_filesystem\_avail\_bytes{mountpoint!~"/boot.\*"} Tìm khác boot.\*
* node\_cpu\_seconds\_total{mode=~"user|system"}

**Multi-select** kết hợp:

* node\_filesystem\_avail\_bytes{instance="192.168.88.111:9100", device!="tmpfs"} tìm filesystem thừa, ngoại trừ thư mục tmpfs
* node\_network\_receive\_bytes\_total{instance="node02:9100", device!="lo"} tìm tổng dung lượng card mạng , ngoại trừ loopback

**RangeVector select:** node\_arp\_entries{instance="192.168.88.111:9100"}[2m]

### Slide 174: Modifier offset (giá trị quá khứ)

node\_arp\_entries{instance="192.168.88.111:9100"} offset 1h30m: (lấy trước đó 1h 30 phút)

node\_arp\_entries{instance="192.168.88.111:9100"} @1674699626 offset 5m (lấy 5 phút trước so với điểm epoch time)

node\_arp\_entries{instance="192.168.88.111:9100"}[5m] @1674699626 offset 10m (Lấy 5 phút range ở 10 phút trước tại điểm epoch)

## LAB1 Bài tập: PromQL Selectors, Matchers, and Modifiers

VD1: up

VD2: up{job="web"}

VD3: node\_memory\_MemAvailable\_bytes{job="loadbalancer"}

VD4: node\_arp\_entries{instance="node01:9100"}

VD5: Which of the following queries will return the number of bytes received on interface eth1 on instance node01:9100?

node\_network\_receive\_bytes\_total{instance="node01:9100",device="eth1"}

VD6: Each host includes a loopback interface i.e lo. Construct a query to return the number of bytes received on all interfaces except interface lo on instance node02:9100.

node\_network\_receive\_bytes\_total{instance="node02:9100", device!="lo"}

VD7: Using node\_memory\_MemAvailable\_bytes metric, construct a query to return the available memory bytes for past 5 minutes on node01:9100

node\_memory\_MemAvailable\_bytes{instance="node01:9100"}[5m]

VD8: Queries will return the 1h ago available memory bytes on node01:9100 host?

node\_memory\_MemAvailable\_bytes{instance="node01:9100"} offset 1h

VD9: Which of the following queries will return the value of the metric on timestamp 1654920221 i.e Jun 11, 2022 4:03:41 AM GMT for instance node02:9100?

node\_context\_switches\_total{instance="node02:9100"} @1654920221

VD10: Which of the following queries will return the value of the metric 30 minutes before Jun 11, 2022 4:03:41 AM GMT for instance node01:9100? The unix timestamp value for this time is 1654920221.

node\_context\_switches\_total{instance="node01:9100"} @1654920221 offset 30m

VD11: The node\_cpu\_seconds\_total metric has two labels. One is the cpu, which denotes for which cpu the time series is for and the other one is mode, which denotes the cpu operating mode.

Construct a query that will return a metric for all instances in the web job for cpu 0 only and the cpu mode can be either user or system. To match user or system, a regular expression like user|system will be used.

node\_cpu\_seconds\_total{job="web", cpu="0", mode=~"user|system"}

### Slide 184: Operator ( + - \* / và so sánh ==, !== , > , < , >=, <=)

node\_cpu\_seconds\_total{mode="system"} + 10 : Cộng thêm 10 giá trị

node\_cpu\_seconds\_total{mode="system"} / 10 : Chia 10 giá trị

node\_network\_flags > 100 : Tìm những giá trị lớn hơn 100

node\_filesystem\_avail\_bytes < bool 1000 : trả về 0 hoặc 1 boolean nếu nhỏ hơn 1000

Dùng AND: node\_filesystem\_avail\_bytes > 1000 and node\_filesystem\_avail\_bytes < 3000

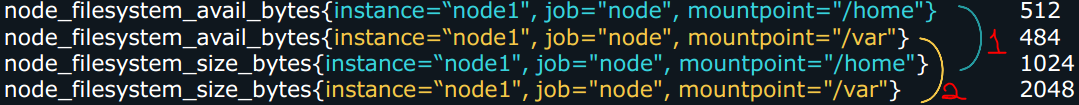
Dùng OR: node\_filesystem\_avail\_bytes < 500 or node\_filesystem\_avail\_bytes > 70000

Dùng Unless: node\_filesystem\_avail\_bytes > 1000 unless node\_filesystem\_avail\_bytes > 30000 (Ngoại trừ)

### Slide 198: Vector matching

1. Vector matching tự động trùng label

node\_filesystem\_avail\_bytes / node\_filesystem\_size\_bytes \* 100 : Lấy phần trăm dựa trên match chính xác từng label



2. Vector matching with IGNORE

3. Vector matching with keyword

### Slide 213: Aggergation (nhóm)

|  |  |
| --- | --- |
| Aggregator | Description |
| Sum | Tính tổng theo thứ nguyên |
| Min | Chọn tối thiểu trên kích thước |
| Max | Chọn tối đa trên kích thước |
| Avg | Trung bình trên kích thước |
| Group | Tất cả các giá trị trong vectơ kết quả là 1 |
| Stddev | Tính độ lệch chuẩn tổng thể theo thứ nguyên |
| Stdvar | Tính toán phương sai tiêu chuẩn tổng thể theo thứ nguyên |
| Count | Đếm số phần tử trong vector |
| Count\_values | Đếm số phần tử có cùng giá trị |
| Bottomk | Phần tử k nhỏ nhất theo giá trị mẫu |
| Topk | Phần tử k lớn nhất theo giá trị mẫu |
| Quantile | tính φ-phân vị (0 ≤ φ ≤ 1) theo thứ nguyên |

sum(http\_requests)

sum by(path) (http\_requests)

sum by(method) (http\_requests)

sum by(instance) (http\_requests)

sum by(instance, method) (http\_requests)

sum without(path) (http\_requests)

## LAB2 Bài tập: Operators, Vector Matching, Aggregators

VD1: Xây dựng một truy vấn để trả về tất cả các hệ thống tệp có sẵn trên 1000 byte trên tất cả các instance

node\_filesystem\_avail\_bytes{job="web"} > 1000

VD2:

node\_network\_receive\_bytes\_total{instance="loadbalancer:9100"} <= 10000

VD3: node\_filesystem\_files theo dõi tổng số nút tệp của hệ thống tệp. Tạo truy vấn chỉ trả về chuỗi thời gian lớn hơn 500000 và nhỏ hơn 10000000 trên tất cả các công việc:

node\_filesystem\_files > 500000 AND node\_filesystem\_files < 10000000

VD4: Chỉ số node\_filesystem\_avail\_bytes liệt kê các byte khả dụng cho tất cả các hệ thống tệp và chỉ số node\_filesystem\_size\_bytes liệt kê tổng kích thước của tất cả các hệ thống tệp. Chạy từng số liệu và xem kết quả đầu ra của chúng. Có ba thuộc tính/nhãn mà chúng sẽ trả về, thiết bị, fstype và điểm gắn kết.

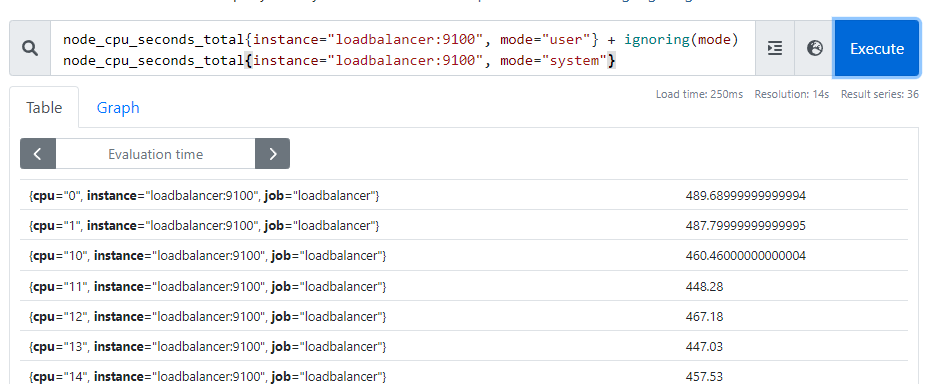
Truy vấn nào sau đây sẽ hiển thị phần trăm dung lượng đĩa trống cho tất cả các hệ thống tệp trên tất cả các mục tiêu trong công việc web có nhãn thiết bị không khớp với tmpfs?

node\_filesystem\_avail\_bytes{job="web", device!="tmpfs"}\*100 / node\_filesystem\_size\_bytes{job="web", device!="tmpfs"}

VD5: Có thể sử dụng truy vấn nào sau đây để theo dõi tổng số giây mà cpu đã sử dụng trong chế độ user + system đối với ví dụ loadbalancer:9100?

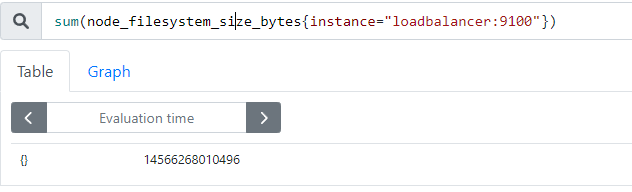
Giải: Yêu cầu tính tổng cpu\_sencond theo từng CPU với nhãn user và system

node\_cpu\_seconds\_total{instance="loadbalancer:9100", mode="user"} + ignoring(mode) node\_cpu\_seconds\_total{instance="loadbalancer:9100", mode="system"}



VD6: Trên phiên bản loadbalancer:9100, hãy tính tổng kích thước của tất cả các hệ thống tệp. Số liệu để lấy kích thước hệ thống tệp là node\_filesystem\_size\_bytes

sum(node\_filesystem\_size\_bytes{instance="loadbalancer:9100"}) : Tính tổng filesystem của server loadbanlancer:9100



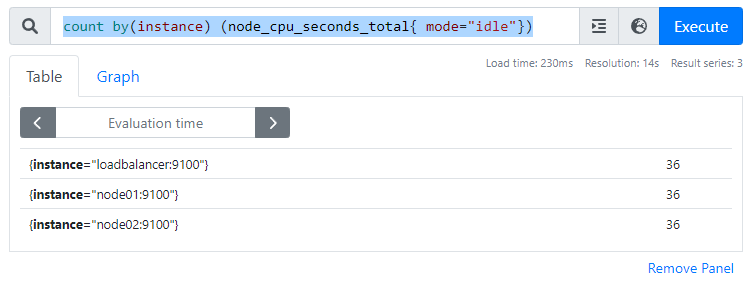
VD7: Construct a query to find how many CPUs instance loadbalancer:9100 have. You can use the node\_cpu\_seconds\_total metric to find out the same. Tìm tổng số CPU ở server loadbalancer:9100

Cách 1: count(node\_cpu\_seconds\_total{instance="loadbalancer:9100",mode="idle"}):

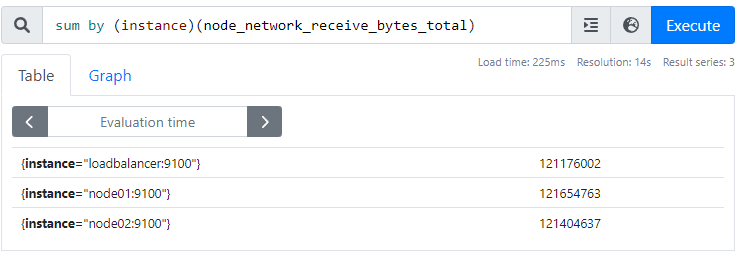
Cách 2: count(count by(cpu) (node\_cpu\_seconds\_total{instance="loadbalancer:9100"}))

VD8: Xây dựng một truy vấn sẽ hiển thị số lượng CPU trên mỗi phiên bản trên tất cả các công việc.

count by(instance) (node\_cpu\_seconds\_total{ mode="idle"}): Tính tổng số cpu. Hiển thị nhóm theo từng server.



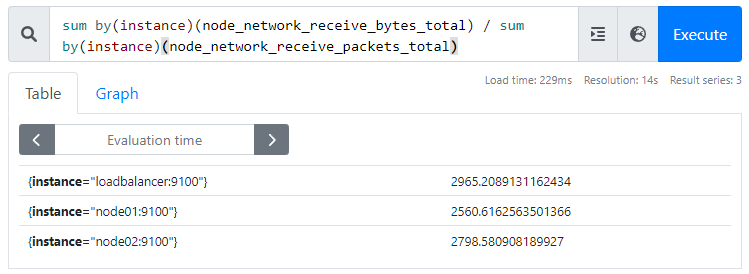
VD9: Sử dụng chỉ số node\_network\_receive\_bytes\_total để tính tổng của tổng số byte nhận được trên tất cả các giao diện trên cơ sở mỗi instance:

sum by(instance) (node\_network\_receive\_bytes\_total): Tìm tổng thông lượng nhận được, nhóm theo từng server. 

VD10: Truy vấn sẽ được sử dụng để tính toán kích thước gói trung bình cho từng instance?

sum by(instance)(node\_network\_receive\_bytes\_total) / sum by(instance)(node\_network\_receive\_packets\_total)

Calculate the average packet size for each instance?



VD11: Xây dựng một truy vấn sẽ tìm ra phần trăm thời gian mà mỗi cpu trên mỗi instance đã được sử dụng trong chế độ người dùng. Kết quả cuối cùng sẽ như bên dưới, với chuỗi thời gian cho từng instance và cpu với x là giá trị của chuỗi thời gian

{cpu="0", instance="node1"} x

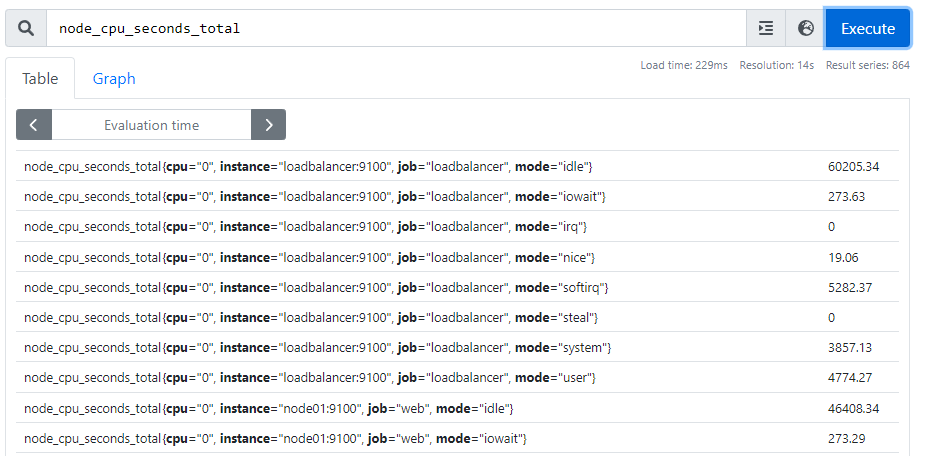
{cpu="1", instance="node1"} x

{cpu="0", instance="node2"} x

{cpu="1", instance="node2"} x

{cpu="0", instance="node3"} x

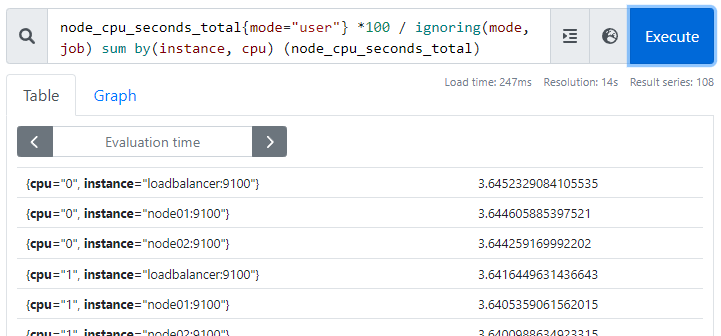
{cpu="1", instance="node3"} x



To calculate the percentage in mode user, get the total seconds spent in mode user and divide that by the sum of the time spent across all modes. Further, multiply that result by 100 to get a percentage.

node\_cpu\_seconds\_total{mode="user"}\*100 /ignoring(mode, job) sum by(instance, cpu) (node\_cpu\_seconds\_total)

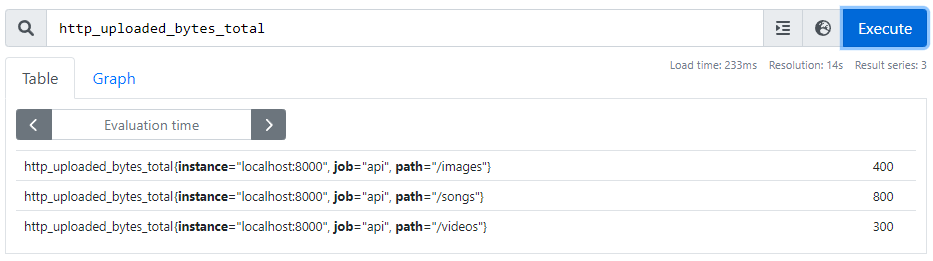
node\_spu\_second\_total có 4 trường là cpu0,1,2,3,4 và instance, job, mode. Để nhóm theo từng instance và cpu ta sẽ cần ignore mode và job.



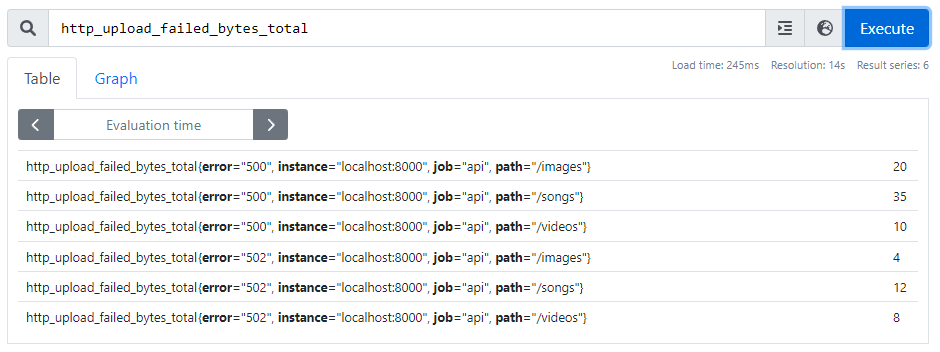
**VD12:**

Công việc api thu thập số liệu trên API được sử dụng để tải tệp lên. API có 3 endpoints /images /videos, and /songs, được sử dụng để tải lên các loại tệp tương ứng. API cung cấp 2 số liệu để theo dõi

http\_uploaded\_bytes\_total - tracks the number of uploaded bytes.



http\_upload\_failed\_bytes\_total - tracks the number of bytes failed to upload.



Xây dựng một truy vấn để tính phần trăm byte không thành công cho mỗi điểm cuối. Công thức cho cùng là

http\_upload\_failed\_bytes\_total\*100 / http\_uploaded\_bytes\_total.

Giải:

Kết quả cuối cùng: http\_upload\_failed\_bytes\_total\*100 / ignoring(error) group\_left http\_uploaded\_bytes\_total



### Slide 222: Function

#### Hàm phổ biến ceil, floor, abs, time

ceil(node\_cpu\_seconds\_total) : làm tròn trên

floor(node\_cpu\_seconds\_total): làm tròn dưới

abs(1-node\_cpu\_seconds\_total): trị tuyệt đối

time(): lấy thời gian hiện tại

time() - process\_start\_time\_seconds: tính khoảng thời gian chạy so với thời điểm time() hiện tại

#### Hàm date\_time

Minute()

Hour()

Day\_of\_week()

Day\_of\_month()

Days\_in\_month()

Month()

Year()

**- Change Type**

Chuyển vector sang scalar: scalar(process\_start\_time\_seconds)

**- Functino other:**

sort(node\_filesystem\_avail\_bytes) Mặc định theo ASC

sort\_desc(node\_filesystem\_avail\_bytes)

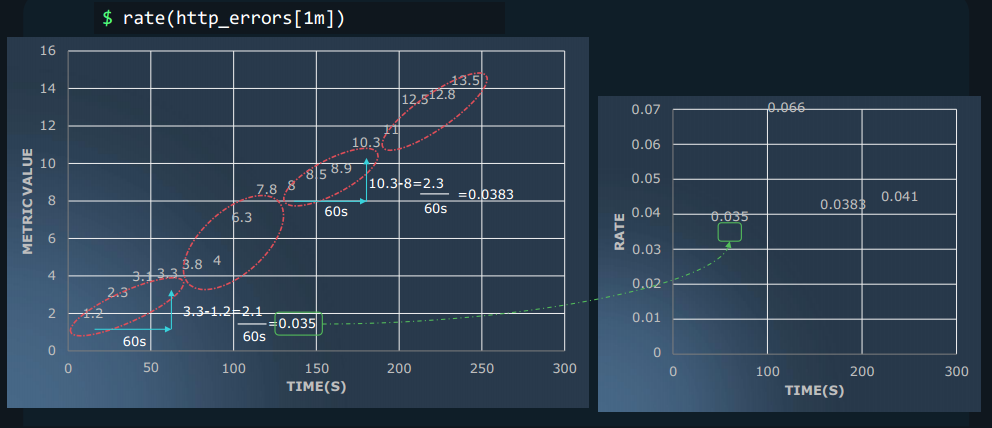
rate(http\_errors[1m]) : lấy trung bình theo từng giây thay đổi (giá trị đầu trừ giá trị cuối trong bước thời gian)

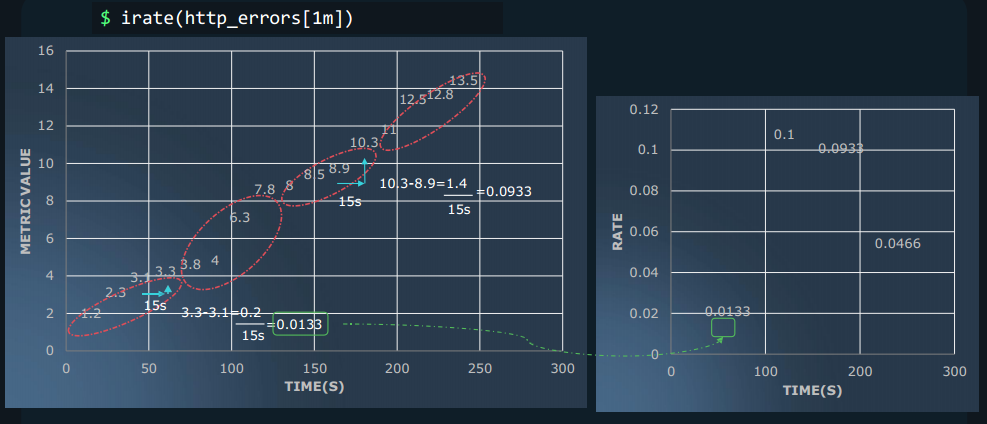
irate(http\_errors[1m]) : Lấy theo 2 mẫu gần nhất, rồi trừ đi cho nhau ra sự thay đổi

#### Rate và irate:

**Rate** lấy mẫu theo 1 phút, và lấy giá trị đầu trừ cuối. **Irate** lấy giá trị gần nhất trừ đi cho nhau trên 1 mẫu thời gian.

Rate thích hợp biểu đồ chậm tăng dần, còn irate tỉ lệ chênh lệch nhiều theo từng mẫu thời gian thì dùng irate.





### Slide 237 Subquery

max\_over\_time(node\_filesystem\_avail\_bytes[10m]) : lấy giá trị max trong toàn bộ thời gian

Format của subquery như sau: <instant\_query> [<range>:<resolution>] [offset <duration>]

VD1:

$ rate(http\_requests\_total[1m]) [5m:30s]

1m – sample range

5m – query range(get data from last 5 minutes)

30s – query step for subquery

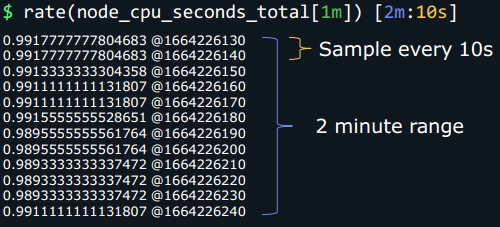
VD2:

$ max\_over\_time(rate(http\_requests\_total[1m]) [5m:30s])

Maximum rate of requests from the last 5 minutes with a 30s

query interval and a sample range of 1m

VD3: $ rate(node\_cpu\_seconds\_total[1m]) [2m:10s]



### Slide 242: Histogram ??????

### Sliede 249 Quantiles ????????

## LAB3 bài tập – Functions, subqueries, Histogram, Summary

Bài 1/2: There are three jobs we added in Prometheus, multimedia, auth, and api. Construct a query that will fetch the node\_cpu\_seconds\_total metric for all jobs and sort it in an ascending order

sort(node\_cpu\_seconds\_total)

sort\_desc(node\_memory\_Active\_bytes)

Bài 3: Calculate the percentage of free space for all filesystems on all instances under all jobs using below query:

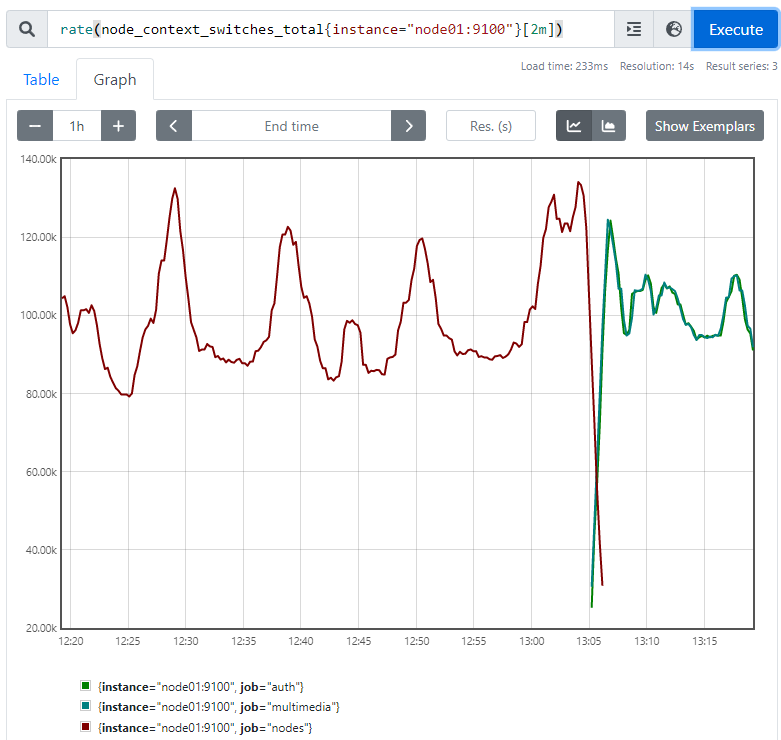
*node\_filesystem\_avail\_bytes\*100/node\_filesystem\_size\_bytes* . Note how the percentages have a lot of decimal values.

Use the round function to round the query's result to the nearest integer value.

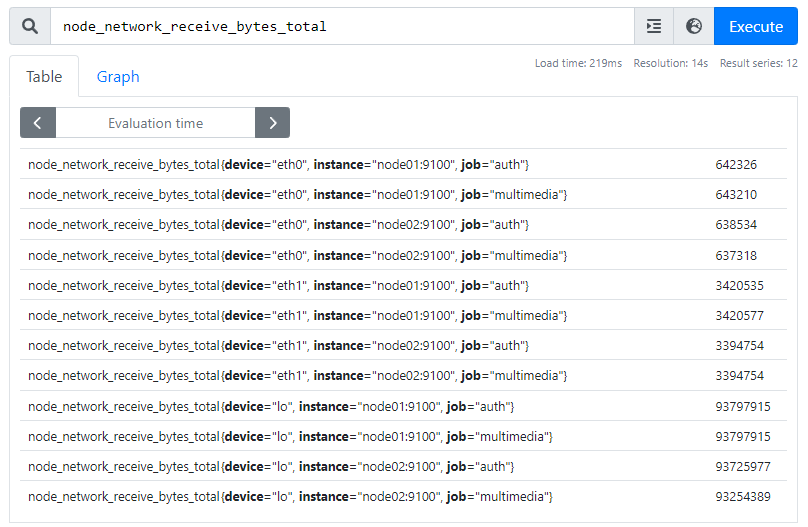
round(node\_filesystem\_avail\_bytes\*100/node\_filesystem\_size\_bytes): làm tròn phần trăm disk sử dụng

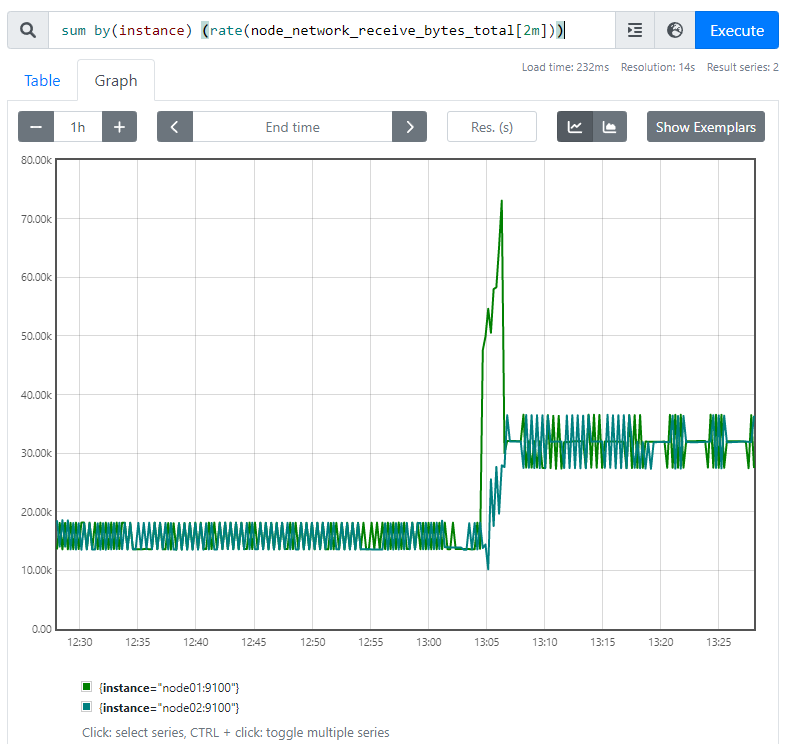
Bài 4: node\_context\_switches\_total{instance="node01:9100"} lệnh này hiển thị độ tăng tổng tuyến tính, nhưng để xem từng bước tăng theo khoảng thời gian. Ta dùng lệnh sau:

**rate**(node\_context\_switches\_total{instance="node01:9100"}[2m])

>>rate>>

Bài 5: Kiểm tra mức network nhận (bước rate mỗi 2 phút), sau lấy tổng theo cả instance, không phân biệt interface ở hình 2.

Ta dùng sum by để gộp các instance lại (2 hình dưới)

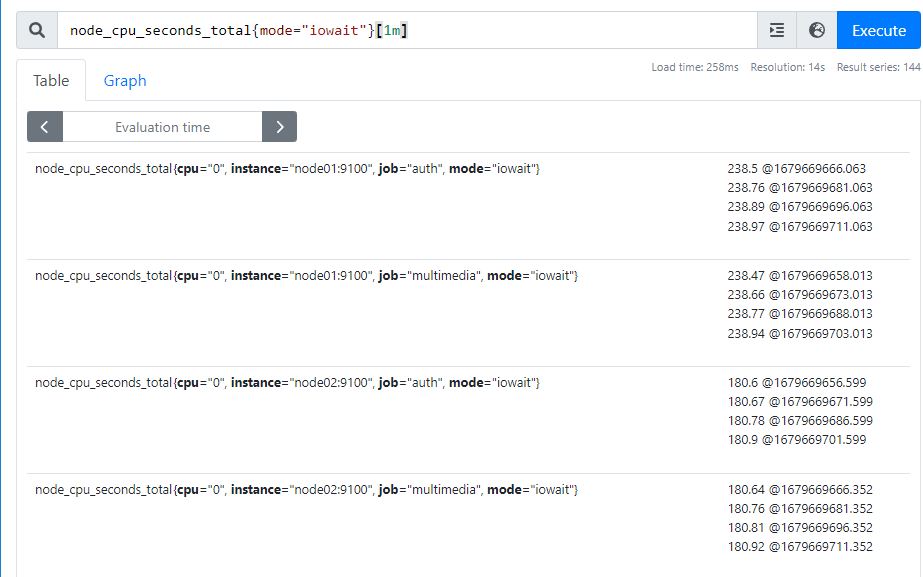


Bài 6: Đã có báo cáo về sự cố ngừng hoạt động nhỏ của một ứng dụng trong vài phút qua và một số cảnh báo chỉ ra khả năng iowait cao tiềm ẩn trên cpus. Chúng tôi cần tính toán thời điểm tỷ lệ iowait cao nhất trong 10 phút qua.

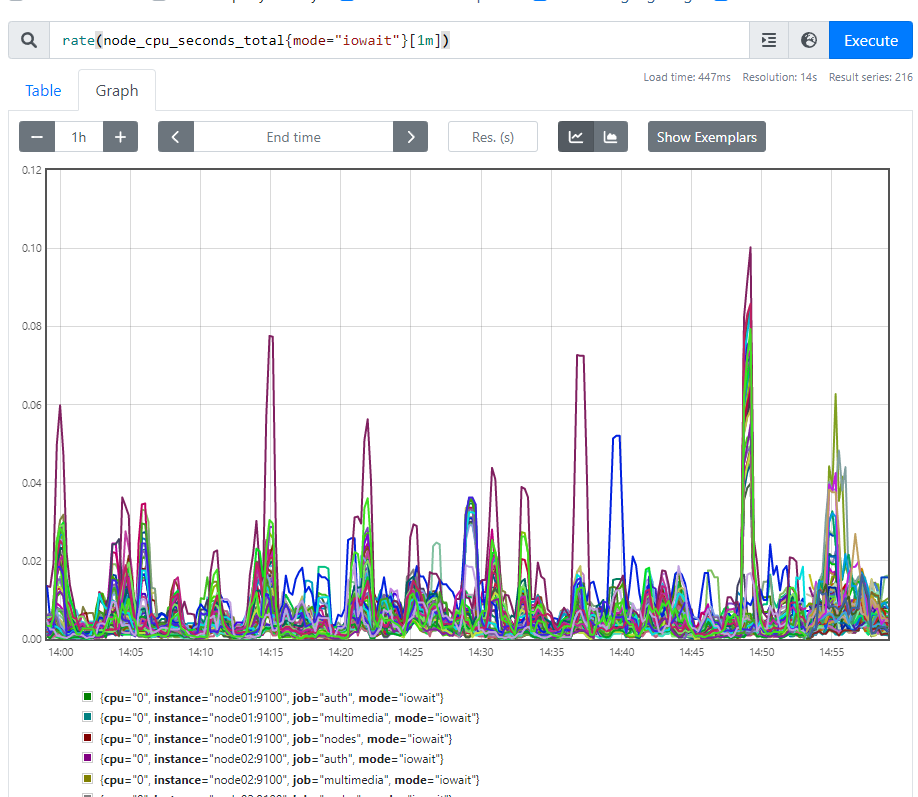
Xây dựng một truy vấn con sẽ tính toán tốc độ mà tất cả cpus đã sử dụng trong chế độ iowait bằng cách sử dụng cửa sổ thời gian 1 phút cho hàm phân giải tốc độ. Tìm giá trị tối đa của kết quả này trong 10 phút qua bằng cách sử dụng bước truy vấn 30 giây cho truy vấn con.

**Bài giải:**

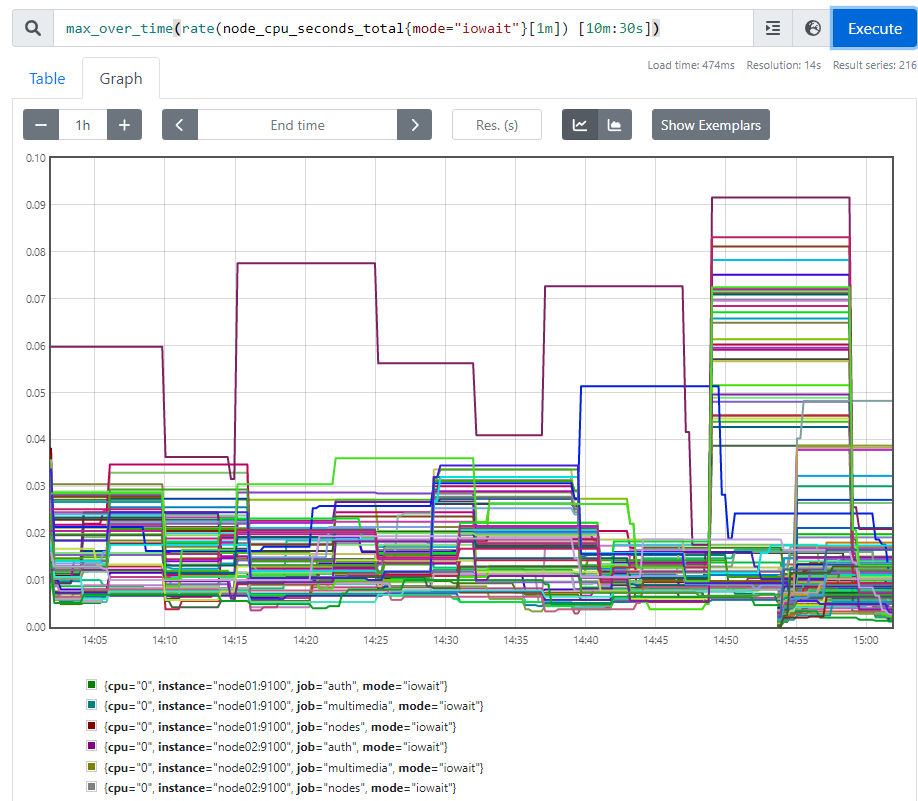
Bước 1: node\_cpu\_seconds\_total{mode="iowait"}[1m]



Bước 2: rate(node\_cpu\_seconds\_total{mode="iowait"}[1m])



max\_over\_time(rate(node\_cpu\_seconds\_total{mode="iowait"}[1m]) [10m:30s])



Để có được giá trị tối đa của một vectơ phạm vi, hàm max\_over\_time cần được sử dụng. Vì chúng ta cần tìm tốc độ tối đa mà cpus sử dụng trong chế độ iowait, nên hàm tốc độ trả về một vectơ tức thì và max\_over\_time mong đợi một vectơ phạm vi. Trong trường hợp này, một truy vấn phụ cần được sử dụng. Trong [] có hai giá trị truy vấn phụ cần được chuyển. Câu hỏi đầu tiên là chúng ta nên quay lại bao xa để có được tỷ lệ, câu hỏi đã hỏi trong 10 phút. Giá trị thứ hai là bước truy vấn. Bước truy vấn cho prometheus biết tần suất tính toán tỷ lệ. Vì vậy, bước truy vấn 30 giây có nghĩa là giá trị tốc độ sẽ được tính sau mỗi 30 giây và được chuyển vào hàm max\_over\_time. Truy vấn cuối cùng sẽ là: max\_over\_time(rate(node\_cpu\_seconds\_total{mode="iowait"}[1m]) [10m:30s])

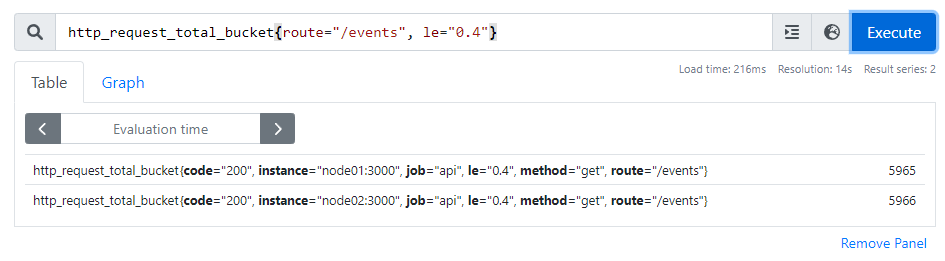
**Bài 7:** Tạo truy vấn để tính tỷ lệ trung bình theo thời gian (avg\_over\_time) của http\_requests\_total trong 20 phút qua bằng cách sử dụng bước truy vấn 1 phút. Đối với tỷ lệ, sử dụng cửa sổ thời gian 2m. (Không có dữ liệu mẫu để chụp ảnh)

avg\_over\_time(rate(http\_requests\_total[2m]) [20m:1m])

Bài 8: Đối với chỉ số http\_request\_total, tên truy vấn/chỉ số sẽ được sử dụng để nhận tổng số yêu cầu trên nút là gì (Không có dữ liệu mẫu để chụp ảnh)

http\_request\_total\_count{instance="node01:3000"}

Bài 9: Tạo truy vấn để trả về tổng số yêu cầu request/events với độ trễ dưới 0,4 giây

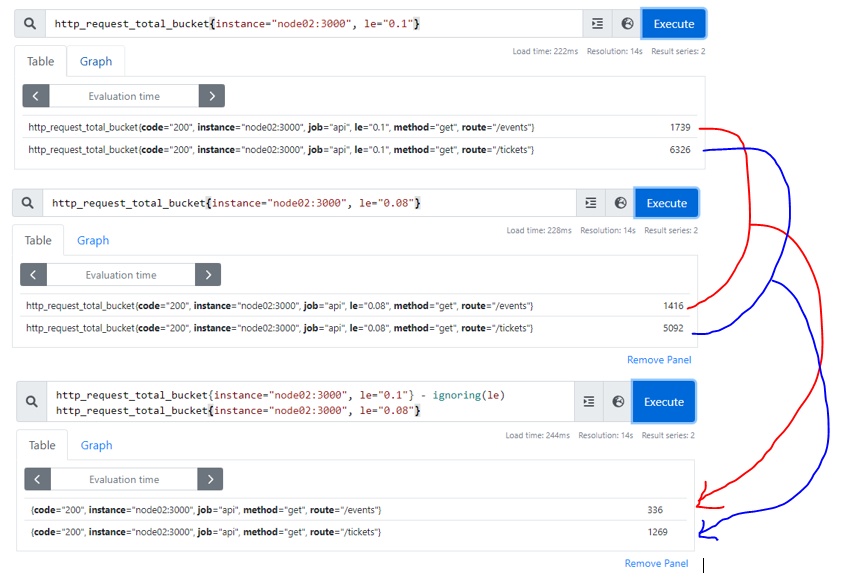


http\_request\_total\_bucket{route="/events", le="0.4"}

Bài 10: Xây dựng truy vấn để tìm hiểu xem có bao nhiêu yêu cầu được thực hiện trong khoảng từ 0,08 giây đến 0,1 giây trên nút node02:3000.

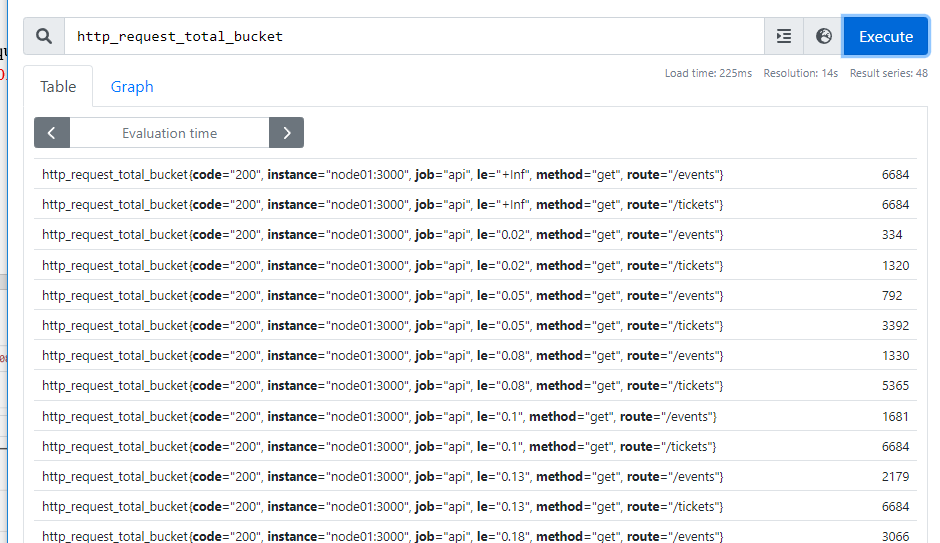
Giải: Để có được số lượng quan sát giữa hai nhóm, bạn sẽ cần trừ đi sự khác biệt giữa hai nhóm. Vì nhãn tập tin sẽ khác nhau giữa hai vectơ, nên chúng ta sẽ cần bỏ qua nhãn label:

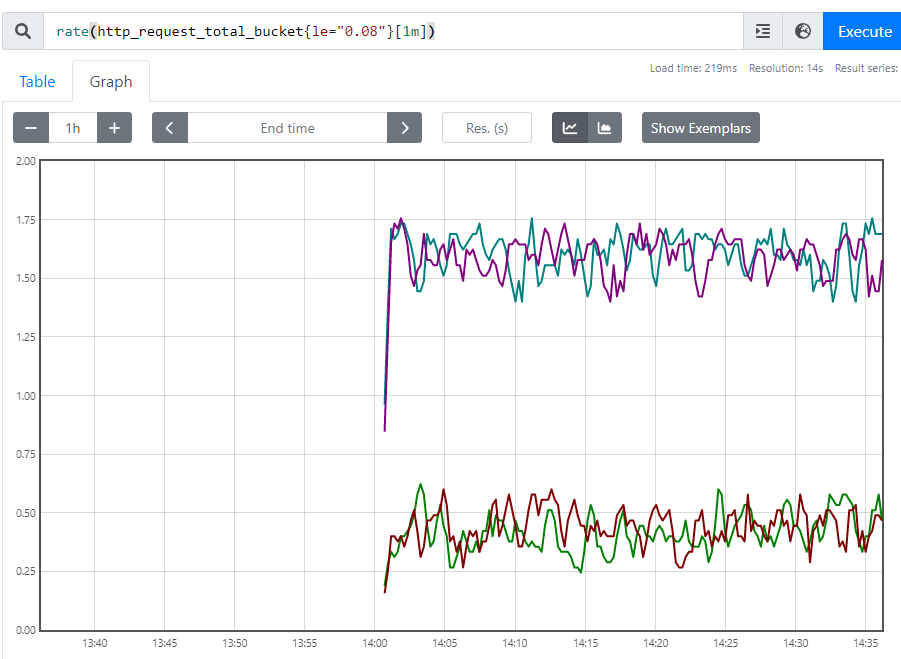
http\_request\_total\_bucket{instance="node02:3000", le="0.1"} - ignoring(le) http\_request\_total\_bucket{instance="node02:3000", le="0.08"}



Bài 11: Tạo truy vấn để tính tỷ lệ request http mất ít hơn 0,08 giây. Sử dụng cửa sổ thời gian 1m trên tất cả các nút

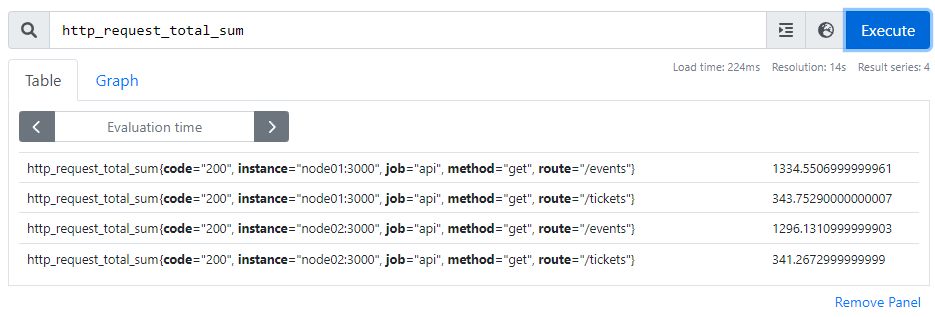
rate(http\_request\_total\_bucket{le="0.08"}[1m])

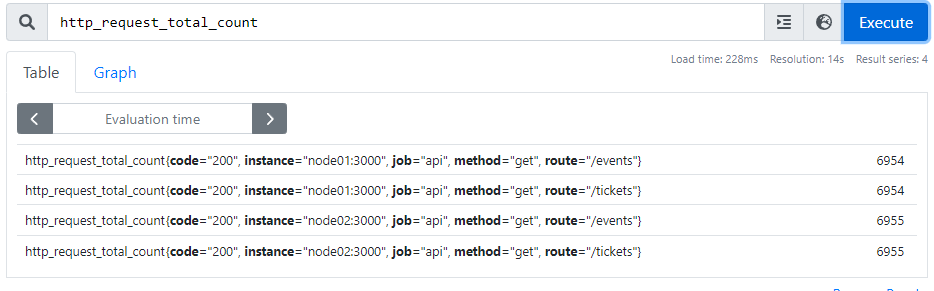


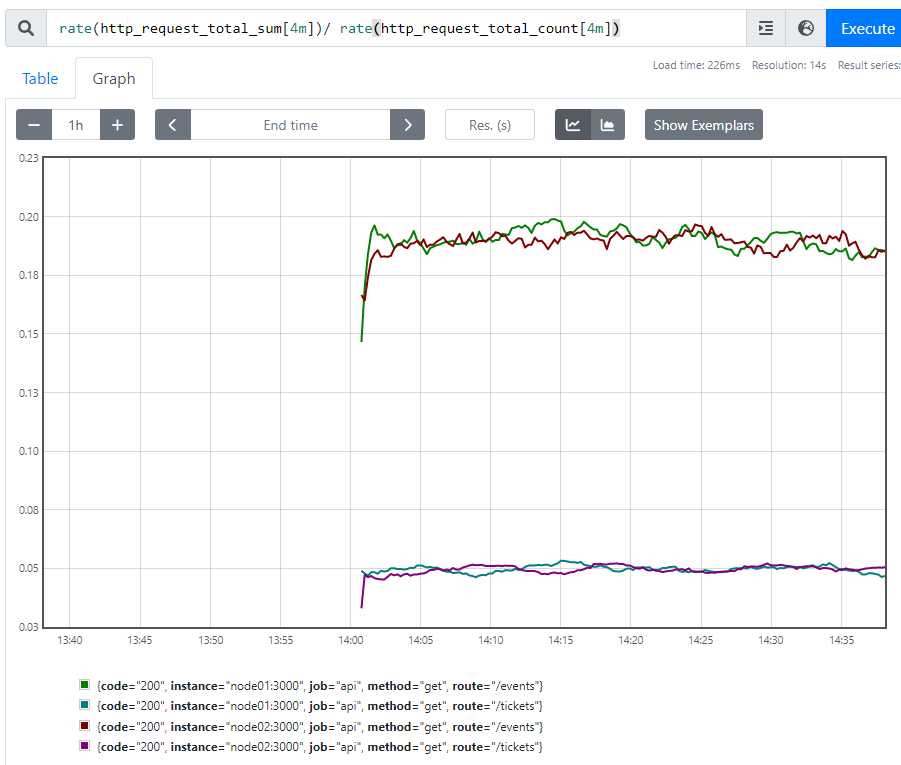


Bài 12: Tạo truy vấn để tính toán độ trễ trung bình của yêu cầu trong 4 phút qua. Sử dụng công thức bên dưới để tính độ trễ trung bình của yêu cầu. (rate of sum-of-all-requests / rate of count-of-all-requests)

rate(http\_request\_total\_sum[4m])/ rate(http\_request\_total\_count[4m])



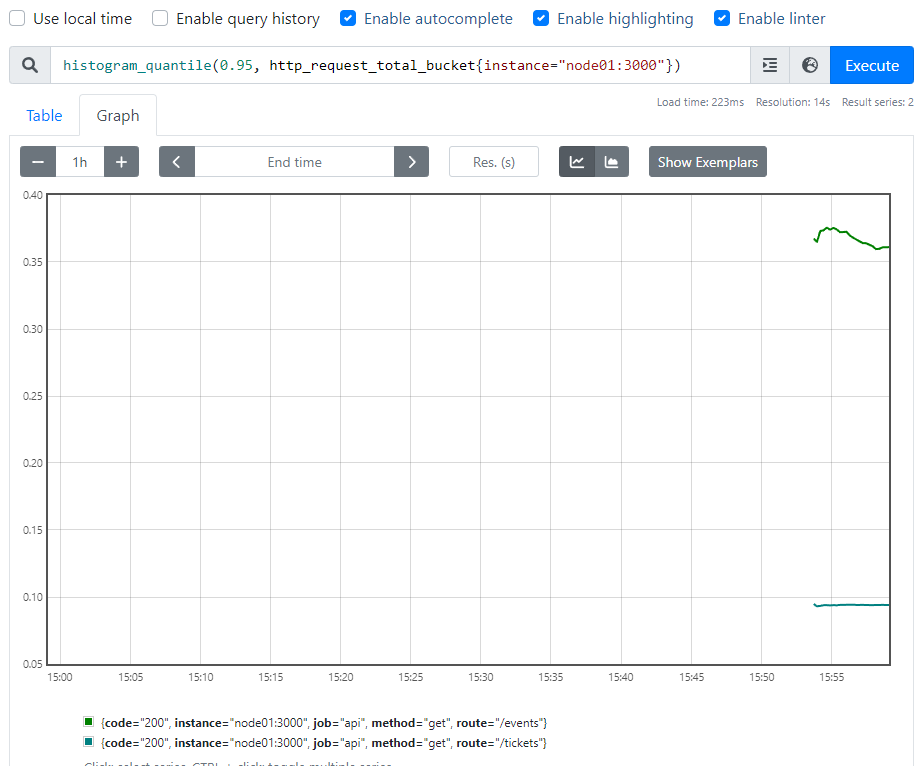




**Bài 13:** Management would like to know what is the 95th percentile for the latency of requests going to node node01:3000. Construct a query to calculate the 95th

Ban quản lý muốn biết phân vị thứ 95% cho độ trễ của các yêu cầu đến nút node01:3000 là bao nhiêu. Xây dựng một truy vấn để tính phân vị thứ 95.

histogram\_quantile(0.95, http\_request\_total\_bucket{instance="node01:3000"})

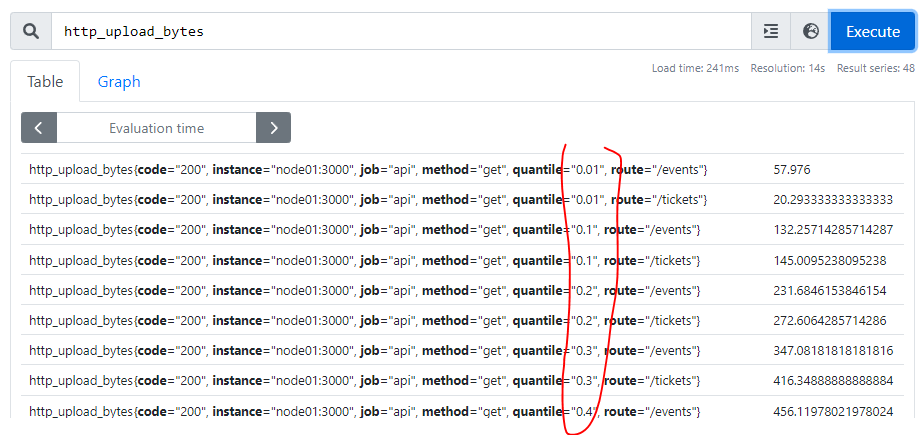


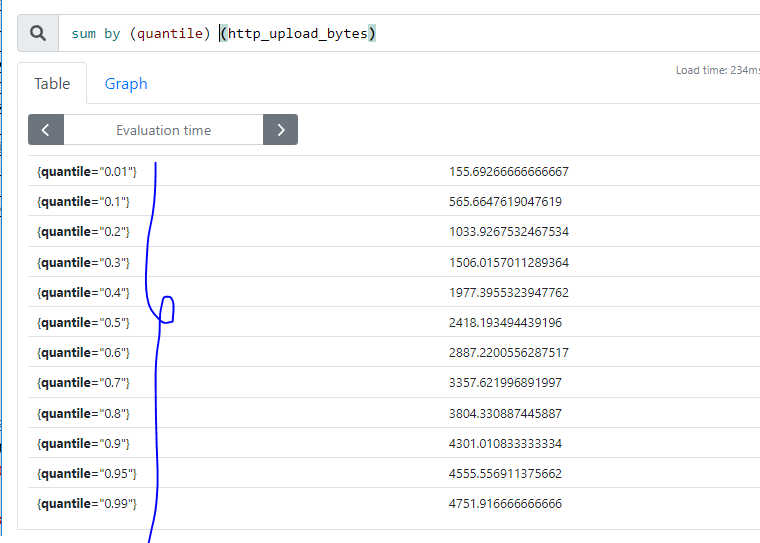
**Bài 14:** The company is now offering customers an SLO stating that, 95% of all requests will be under 0.15s. What bucket size will need to be added to guarantee that the histogram\_quantile function can accurately report whether or not that SLO has been met?

Công ty hiện đang cung cấp cho khách hàng một SLO nói rằng, 95% tất cả các yêu cầu sẽ dưới 0,15 giây. Kích thước bộ chứa nào sẽ cần được thêm vào để đảm bảo rằng hàm histogram\_quantile có thể báo cáo chính xác liệu SLO đó có được đáp ứng hay không?

Giải: Kích thước nhóm cần được thêm vào để đảm bảo rằng hàm histogram\_quantile có thể báo cáo chính xác liệu SLO đó có được đáp ứng hay không là 0,15.

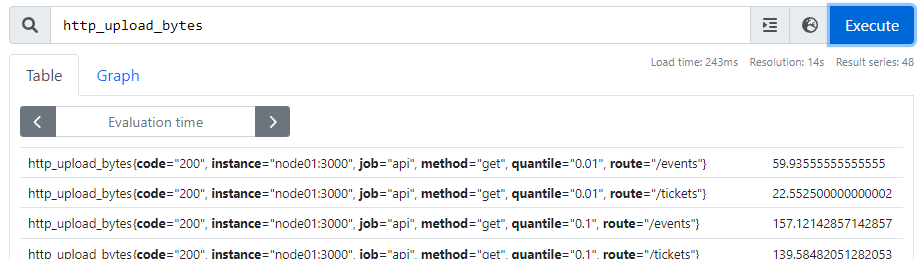
Bài 15: Số liệu tóm tắt http\_upload\_bytes đã được thêm vào để theo dõi lượng byte được tải lên trên mỗi yêu cầu. Phần trăm được báo cáo bởi số liệu này là gì?





Bài 16: Tạo một truy vấn để lấy phân vị thứ 90 cho các byte được tải lên trên nút node01:3000.

http\_upload\_bytes{instance="node01:3000", quantile="0.9"}



### Slide 260: Recording Rule

Recording rule sinh ra hỗ trợ alerting, mục đích là tái sử dụng nhiều lần trong các alert.

Bước 1, khai báo có chạy rule trong prometheus.yml

**# cat prometheus.yml**

global:

scrape\_interval: 15s

evaluation\_interval: 15s

alerting:

alertmanagers:

- static\_configs:

- targets:

- 192.168.88.111:9093

rule\_files:

- "/etc/prometheus/rules/first\_rules.yml"

# - "second\_rules.yml"

scrape\_configs:

Bước 2

# cat /etc/prometheus/rules/first\_rules.yml

groups:

- name: node

interval: 15s

rules:

#1 ví dụ về record rule

- record: node\_memory\_memFree\_percent

expr: 100 - (100 \* node\_memory\_MemFree\_bytes{job="node"} / node\_memory\_MemTotal\_bytes{job="node"})

#2 record rule ở trên có thể đc tận dụng làm biến ở record rule khác

- record: node\_filesystem\_free\_percent\_avg

expr: avg by(instance)(node\_filesystem\_free\_percent)

# Ví dụ về alert

- alert: LowMemory

expr: node\_memory\_memFree\_percent < 20

#Ví dụ về nhãn label

- alert: Node down

expr: up{job="node"} == 0

labels:

severity: warning

- alert: Multiple Nodes down

expr: avg without(instance)(up{job="node"}) <= 0.5

labels:

severity: critical

#Ví dụ về annotation giúp hiển thị chú thích

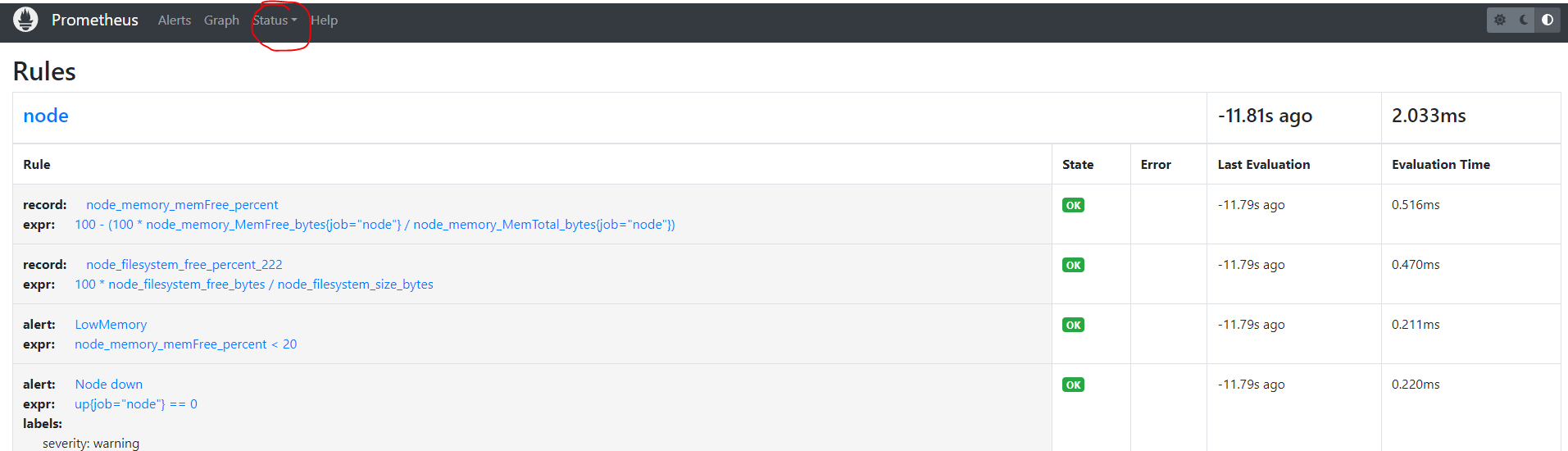
- alert: node\_filesystem\_free\_percent

expr: 100 \* node\_filesystem\_free\_bytes{job="node"} / node\_filesystem\_size\_bytes{job="node"} < 70

annotations:

description: "filesystem {{.Labels.device}} on {{.Labels.instance}} is low on space, current available space is {{.Value}}"

<http://192.168.88.111:9090/rules>



### Slide 272 Record rule naming

???

## LAB4 bài tập: Recording rule

VD1: Tạo quy tắc ghi để theo dõi tốc độ mà một nút đang nhận lưu lượng truy cập. Tìm bên dưới biết thêm chi tiết:

groups:

- name: node

interval: 15s

rules:

- record: node\_network\_receive\_bytes\_rate

expr: rate(node\_network\_receive\_bytes\_total{job="nodes"}[2m])

VD2: Cập nhật group node mà bạn đã tạo trước đó để thêm một quy tắc khác nhằm nhận được tỷ lệ trung bình của các gói đã nhận trên tất cả các interfaces trên một node, vì mỗi node có 2 interfaces.

groups:

- name: node

interval: 15s

rules:

- record: node\_network\_receive\_bytes\_rate

expr: rate(node\_network\_receive\_bytes\_total{job="nodes"}[2m])

- record: node\_network\_receive\_bytes\_rate\_avg

expr: avg by(instance) (node\_network\_receive\_bytes\_rate)

VD3: Cập nhật nhóm nút để thêm một quy tắc khác nhằm theo dõi phần trăm dung lượng trống của hệ thống tệp. Tìm bên dưới biết thêm chi tiết:

groups:

- name: node

interval: 15s

rules:

- record: node\_network\_receive\_bytes\_rate

expr: rate(node\_network\_receive\_bytes\_total{job="nodes"}[2m])

- record: node\_network\_receive\_bytes\_rate\_avg

expr: avg by(instance) (node\_network\_receive\_bytes\_rate)

- record: node\_filesystem\_free\_percent

expr: 100 \* node\_filesystem\_free\_bytes{job="nodes"} / node\_filesystem\_size\_bytes{job="nodes"}

VD4: Tạo 1 group mới, set interval là 15s tính trung bình độ trễ trong 2 phút

groups:

- name: api

interval: 15s

rules:

- record: avg\_latency\_2m

expr: rate(http\_request\_total\_sum{job="api"}[2m]) / rate(http\_request\_total\_count{job="api"}[2m])

VD5: Include all file

# vim prometheus.yml

rule\_files:

- "\*rules.yaml"

### Slide 275 : HTTP API

Thay vì query trên web, ta có thể dùng lệnh trên API:

curl <prometheus>:9090/api/v1/query --data 'query=node\_arp\_entries{instance="192.168.1.168:9100"}'

curl http://localhost:9090/api/v1/query --data 'query=node\_arp\_entries{instance="192.168.88.111:9100"}'

curl localhost:9090/api/v1/query --data 'query=node\_arp\_entries{instance="192.168.88.111:9100"}' --data 'time=1670380680.132'

curl localhost:9090/api/v1/query --data 'query=node\_arp\_entries{instance="192.168.1.168:9100"}[5m]' --data 'time=1670380680.132'

# Phần 5: Dashboarding & Visualization

Các loại visulization <https://grafana.com/docs/grafana/latest/panels-visualizations/visualizations/>

## Lab-Console-Template

**Bước 1. Chuẩn bị trang page basic**

vi /etc/prometheus/consoles/node-stats.html

{{template "head" .}}

{{template "prom\_content\_head" .}}

<h1>Node Stats</h1>

{{template "prom\_content\_tail" .}}

{{template "tail"}}

Ta truy cập vào đường dẫn này để xem trang web http://<prometheus:9090> /consoles/node-stats.html

**Bước 2: Thêm query prometheus vào page (thêm util memory)**

{{template "head" .}}

{{template "prom\_content\_head" .}}

<h1>Node Stats</h1>

<h1>Node Stats</h1>

<h3>Memory</h3>

<strong>Memory utilization:</strong> {{template "prom\_query\_drilldown" (args "100- (node\_memory\_MemAvailable\_bytes/node\_memory\_MemTotal\_bytes\*100)" "%") }}

{{template "prom\_content\_tail" .}}

{{template "tail"}}

Bước 3: Thêm memory size (màu đỏ), CPU màu xanh biển,

{{template "head" .}}

{{template "prom\_content\_head" .}}

<h1>Node Stats</h1>

<h3>Memory</h3>

<strong>Memory utilization:</strong> {{template "prom\_query\_drilldown" (args "100- (node\_memory\_MemAvailable\_bytes/node\_memory\_MemTotal\_bytes\*100)" "%") }}

<strong>**Memory Size:**</strong> {{template "prom\_query\_drilldown" (args "node\_memory\_MemTotal\_bytes/1000000" "Mb") }}

<h3>CPU</h3>

<strong>**CPU Count**:</strong> {{template "prom\_query\_drilldown" (args "count(node\_cpu\_seconds\_total{mode='idle'})") }}

<strong>CPU Utilization:</strong>

{{template "prom\_query\_drilldown" (args "sum(rate(node\_cpu\_seconds\_total{mode!='idle'}[2m]))\*100/8" "%") }}

<h3>Vẽ đồ thị CPU</h3>

<div id="cpu"></div>

<script>

new PromConsole.Graph({

node: document.querySelector("#cpu"),

expr: "sum(rate(node\_cpu\_seconds\_total{mode!='idle'}[2m]))\*100/2",

});

</script>

<h3>Vẽ đồ thị Network</h3>

<div id="network"></div>

<script>

new PromConsole.Graph({

node: document.querySelector("#network"),

expr: "rate(node\_network\_receive\_bytes\_total[2m])",

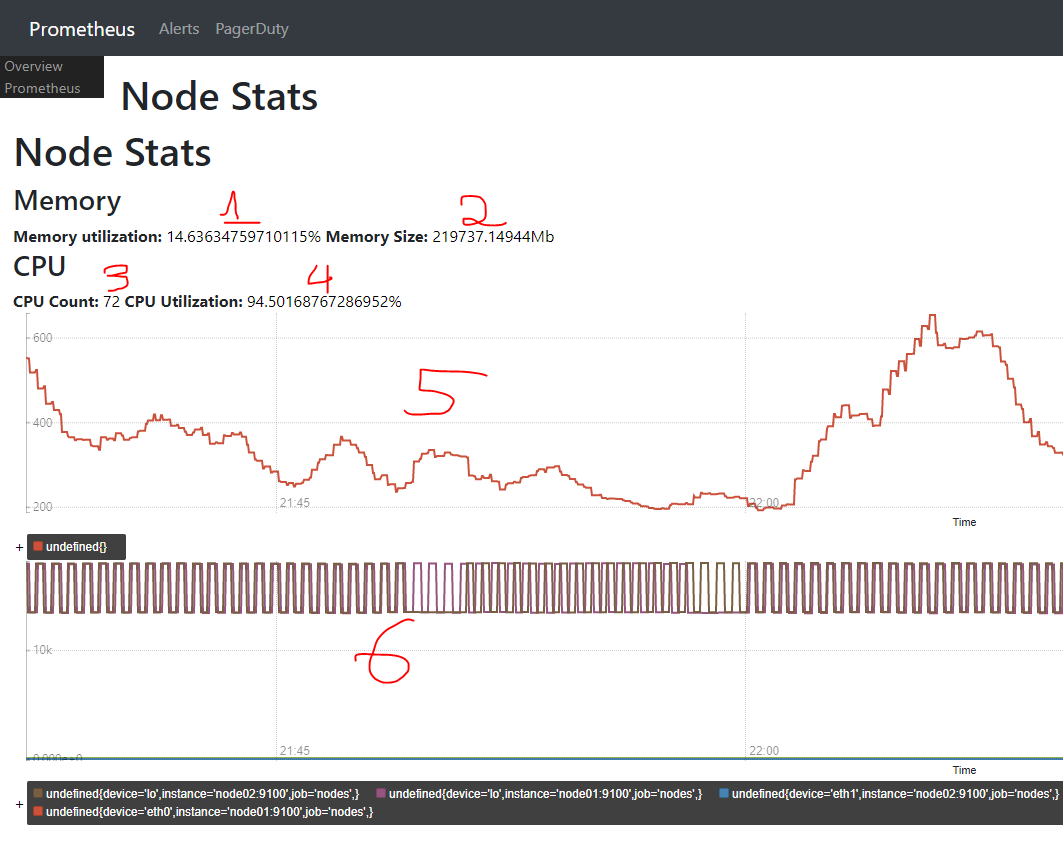
});

</script>

{{template "prom\_content\_tail" .}}

{{template "tail"}}

Kết quả



# Phần 6: Application Instrumentation

Toàn liên quan đến code python

### Slide 324: nameing metric cách đặt tên metric

Lab-Application-Instrumentaion

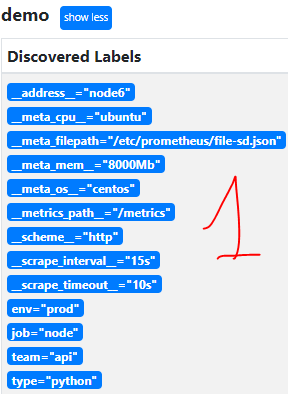
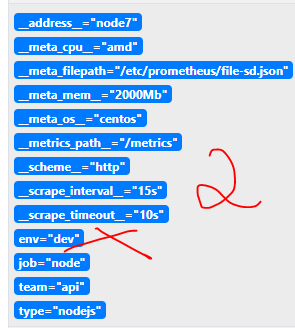
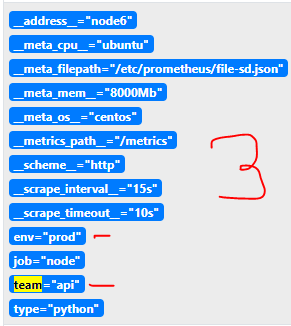
pip install prometheus-client

# Phần 7: Service Discovery

### Slide 352: Re-labeling

Phân biệt relabel\_configs và metric\_relabel\_configs

## Lab-Re-labeling

**File mẫu**: file-sd.json 

VD3: relabel\_configs , For the demo job, configure re-label configs to scrape only targets with env="prod" label and drop all other targets.

# vi /etc/prometheus/prometheus.yml

scrape\_configs:

- job\_name: "prometheus"

static\_configs:

- targets: ["localhost:9090"]

- job\_name: "nodes"

scheme: https

tls\_config:

ca\_file: /etc/prometheus/node\_exporter.crt

insecure\_skip\_verify: true

basic\_auth:

username: prometheus

password: secret-password

static\_configs:

- targets: ["node01:9100", "node02:9100"]

- job\_name: "demo"

file\_sd\_configs:

- files:

- /etc/prometheus/file-sd.json

relabel\_configs:

- source\_labels: [env]

regex: prod

action: keep

Chức năng loại bỏ những server có env khác prod. (env=dev bị xóa)

VD5: relabel\_configs ,We have decided to scrape the metrics from targets that have the following labels only:  
  team=api  
  env=prod  
Make the required changes for demo job.

Giải: sửa lại VD3 thành như sau

# vi /etc/prometheus/prometheus.yml

- job\_name: "demo"

file\_sd\_configs:

- files:

- /etc/prometheus/file-sd.json

relabel\_configs:

- source\_labels: [team, env]

regex: api;prod

action: keep

VD3: relabel\_configs Thực hiện relabel config từ team=api và team=database thành organization=org-api và organization=org-database

Ta thêm config vào trong job name demo

# vi /etc/prometheus/prometheus.yml

- job\_name: "demo"

file\_sd\_configs:

- files:

- /etc/prometheus/file-sd.json

relabel\_configs:

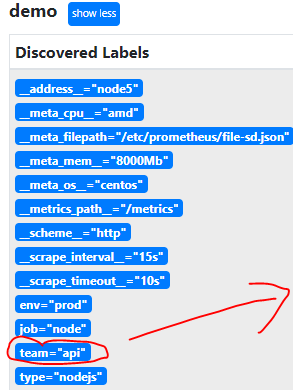
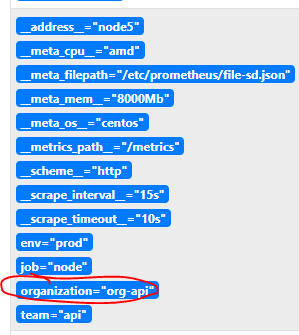
- source\_labels: [team]

regex: (.\*)

action: replace

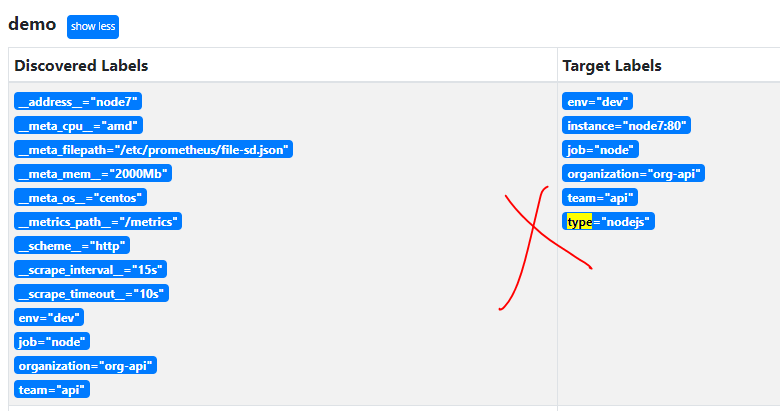
target\_label: organization

replacement: org-$1

 Kết Quả

**Vd4: relabel\_configs Xóa label**

The type label is no longer needed, set up a relabel policy for the demo job to drop this label.



Giải:

# vi /etc/prometheus/prometheus.yml

- job\_name: "demo"

file\_sd\_configs:

- files:

- /etc/prometheus/file-sd.json

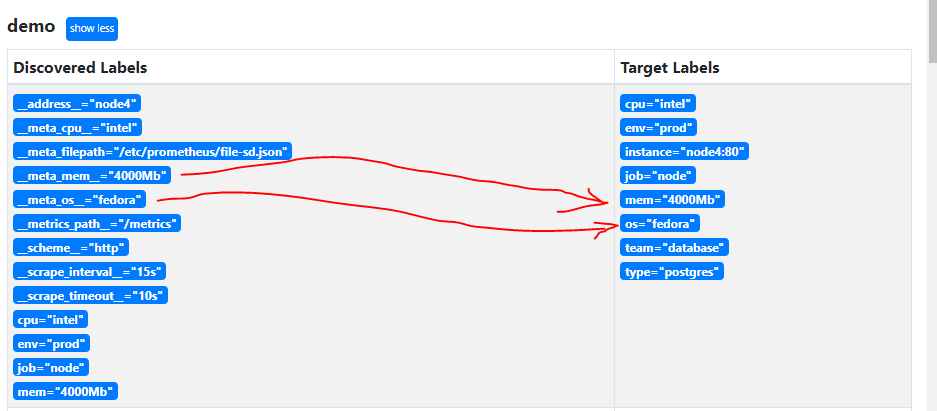
relabel\_configs:

- regex: type

action: labeldrop

VD9: relabel\_configs, sửa tên label

 \_\_meta\_os\_\_=centos ⇒ os=centos  
 \_\_meta\_mem\_\_=8000mb ⇒ mem=8000mb



Giải:

# vi /etc/prometheus/prometheus.yml

- job\_name: "demo"

file\_sd\_configs:

- files:

- /etc/prometheus/file-sd.json

relabel\_configs:

- regex: \_\_meta\_(.\*)\_\_

action: labelmap

replacement: $1

**VD10: metric\_relabel\_configs, Drop metric**

It has been determined that the metric node\_network\_transmit\_drop\_total is no longer needed, configure a metric\_relabel policy to drop this metric.

# vi /etc/prometheus/prometheus.yml

- job\_name: "demo"

metric\_relabel\_configs:

- source\_labels: [\_\_name\_\_]

regex: node\_network\_transmit\_drop\_total

action: drop

file\_sd\_configs:

- files:

- /etc/prometheus/file-sd.json

VD11: **metric\_relabel\_configs,** Rename metric

Update the Prometheus configuration to rename the metric node\_network\_receive\_bytes\_total to node\_network\_rx\_bytes\_total.

- job\_name: "demo"

metric\_relabel\_configs:

- source\_labels: [\_\_name\_\_]

regex: node\_network\_receive\_bytes\_total

action: replace

target\_label: \_\_name\_\_

replacement: node\_network\_rx\_bytes\_total

file\_sd\_configs:

- files:

- /etc/prometheus/file-sd.json

VD12: **metric\_relabel\_configs,** Drop label

The node\_filesystem\_files metric has a label called fstype. Drop this label.

- job\_name: "demo"

metric\_relabel\_configs:

- regex: fstype

action: labeldrop

file\_sd\_configs:

- files:

- /etc/prometheus/file-sd.json

VD13: **metric\_relabel\_configs,**

For network related metrics like node\_network\_receive\_bytes\_total, the device label provides the interface. Rename this label to interface instead of device, and then delete the old device label as well so that there isn’t an extra unnecessary label.

- job\_name: "demo"

metric\_relabel\_configs:

- source\_labels: [device]

regex: (.\*)

action: replace

target\_label: interface

replacement: $1

file\_sd\_configs:

- files:

- /etc/prometheus/file-sd.json

# Phần 8 : Push Gateway

### Slide 375 Push-gateway

**Chức năng:** nơi lưu trữ metrics trung gian, mục đích để prometheus gọi vào lấy dữ liệu

**B1: Cài đặt**

wget https://github.com/prometheus/pushgateway/releases/download/v1.5.1/pushgateway-1.5.1.linux-amd64.tar.gz

tar -xvzf pushgateway-1.5.1.linux-amd64.tar.gz

cd pushgateway-1.5.1.linux-amd64

useradd --no-create-home --shell /bin/false pushgateway

cp pushgateway /usr/local/bin/

chown pushgateway:pushgateway /usr/local/bin/pushgateway

**B2: Tạo systemd file**

vim /etc/systemd/system/pushgateway.service

[Unit]

Description=Prometheus Pushgateway

Wants=network-online.target

After=network-online.target

[Service]

User=pushgateway

Group=pushgateway

Type=simple

ExecStart=/usr/local/bin/pushgateway

[Install]

WantedBy=multi-user.target

**B3: Start pushgateway**

systemctl daemon-reload

systemctl restart pushgateway

systemctl enable pushgateway

**B4: Cấu hình prometheus.yml lấy data from pushgatway**

scrape\_configs:

- job\_name: pushgateway

honor\_labels: true

static\_configs:

- targets: ["192.168.1.168:9091"]

**B5: Push data into gateway**

**VD1: Basic**

echo "metric1\_name 123456" | curl --data-binary @- http://192.168.88.111:9091/metrics/job/<job\_name>/<label1>/<value1>/<label2>/<value2>

**Kết quả:**

echo "example\_metric 4421" | curl --data-binary @- <http://localhost:9091/metrics/job/db_backup>



**VD2: Basic**

cat <<EOF | curl --data-binary @- http://localhost:9091/metrics/job/job1/instance/instance1

# TYPE my\_job\_duration

my\_job\_duration{label="val1"} 42

# TYPE another\_metric counter

# HELP another\_metric Just an example.

another\_metric 12

EOF

**Kết quả:**





**Group data gửi sang pushgateway- dựa theo uri**

$ cat <<EOF | curl --data-binary @- http://localhost:9091/metrics/job/archive/db/mysql

# TYPE metric\_one counter

metric\_one{label="val1"} 11

# TYPE metric\_two gauge

# HELP metric\_two Just an example.

metric\_two 100

EOF

$ cat <<EOF | curl --data-binary @- http://localhost:9091/metrics/job/archive/app/web

# TYPE metric\_one counter

metric\_one{label="val1"} 22

# TYPE metric\_two gauge

# HELP metric\_two Just an example.

metric\_two 200

EOF

**Kết quả:**

curl localhost:9091/metrics | grep archive

metric\_one{db="mysql",instance="",job="archive",label="val1"} 11

metric\_two{db="mysql",instance="",job="archive"} 100

metric\_one{app="web",instance="",job="archive",label="val1"} 22

metric\_two{app="web",instance="",job="archive"} 200

**Cập nhập - sửa đổi dữ liệu đang có trên pushgateway**

POST: Sửa không xóa

cat <<EOF | curl -XPOST --data-binary @- <http://localhost:9091/metrics/job/archive/app/web>

# TYPE metric\_one counter

metric\_one{label="val1"} 33

EOF

kết quả metric\_one thay từ 22 lên 33

PUT: Sửa có xóa (ghi đè)

cat <<EOF | curl –X PUT --data-binary @- <http://localhost:9091/metrics/job/archive/app/web>

# TYPE metric\_one counter

metric\_one{label="val1"} 44

EOF

kết quả là metric\_one từ 22 thành 44, nhưng metric\_two đã bị xóa

DELETE: xóa toàn bộ hoặc 1 phần nhỏ

curl –X DELETE http://localhost:9091/metrics/job/archive/app/web

kết quả toàn bộ /app/web đều bị xóa hết metrics

## Lab – PushGateway

**VD1-2-3 là hướng dẫn cài đặt**

**VD4: Push metric**

echo "processing\_time\_seconds 120" | curl --data-binary @- <http://localhost:9091/metrics/job/video_processing>

**VD5: Đẩy multi dữ liệu lên PUSH gateway**

Push the following metrics to the PushGateway and group the metrics as /job/video\_processing/instance/mp4\_node1  
processing\_time\_seconds{quality="hd"} 120

processed\_videos\_total{quality="hd"} 10

processed\_bytes\_total{quality="hd"} 4400

Next, push the following metrics to the PushGateway and group the metrics as /job/video\_processing/instance/mov\_node1  
processing\_time\_seconds{quality="hd"} 400

processed\_videos\_total{quality="hd"} 250

processed\_bytes\_total{quality="hd"} 96000

Finally, be sure that these metrics are listed as gauge metrics.

Bài giải

cat <<EOF | curl --data-binary @- http://localhost:9091/metrics/job/video\_processing/instance/mp4\_node1

# TYPE processing\_time\_seconds gauge

processing\_time\_seconds{quality="hd"} 120

# TYPE processed\_videos\_total gauge

processed\_videos\_total{quality="hd"} 10

# TYPE processed\_bytes\_total gauge

processed\_bytes\_total{quality="hd"} 4400

EOF

cat <<EOF | curl --data-binary @- http://localhost:9091/metrics/job/video\_processing/instance/mov\_node1

# TYPE processing\_time\_seconds gauge

processing\_time\_seconds{quality="hd"} 400

# TYPE processed\_videos\_total gauge

processed\_videos\_total{quality="hd"} 250

# TYPE processed\_bytes\_total gauge

processed\_bytes\_total{quality="hd"} 96000

EOF

**VD8: Như ở bài 5 đang cso tham số là 120-10-4400. Hãy POST đổi từ 120 lên 999.**

cat <<EOF | curl --data-binary @- http://localhost:9091/metrics/job/video\_processing/instance/mp4\_node1

# TYPE processing\_time\_seconds gauge

processing\_time\_seconds{quality="hd"} 999

EOF

VD9: Hãy PUT đổi từ VD5 từ 10 lên 666

cat <<EOF | curl -X PUT --data-binary @- http://localhost:9091/metrics/job/video\_processing/instance/mp4\_node1

# TYPE processing\_time\_seconds gauge

**processing**\_time\_seconds{quality="hd"} 666

EOF

**VD10: Hãy xóa toàn bộ mp4\_node1 nhưng giữ lại mov\_node1 ở VD5**

curl -X DELETE http://localhost:9091/metrics/job/video\_processing/instance/mp4\_node1

# Phần 9: Alerting + AlertManager

### Slide 409: Alerting

Cấu hình Alert rule.yml như sau

first\_rules.yml

groups:

- name: node

interval: 15s

rules:

#1 ví dụ về recording rule, hỗ trợ khai báo expr alert ở #3

- record: node\_memory\_memFree\_percent

expr: 100 - (100 \* node\_memory\_MemFree\_bytes{job="worker-node1"} / node\_memory\_MemTotal\_bytes{job="worker-node1"})

#2 record rule ở trên có thể đc tận dụng làm biến ở record rule khác

- record: node\_filesystem\_free\_percent\_avg

expr: avg by(instance)(node\_filesystem\_free\_percent)

#3 Ví dụ về alert rule dựa trên recording rule.

- alert: LowMemory

expr: node\_memory\_memFree\_percent < 20

#4 Ví dụ về alert rule sử dụng trực tiếp. Có thêm nhãn severity

- alert: Node down

expr: up{job="worker-node1"} == 0

labels:

severity: warning

- alert: Multiple Nodes down

expr: avg without(instance)(up{job="worker-node1"}) <= 0.5

labels:

severity: critical

#5 Sử dụng For để delay cảnh báo

- alert: Node down

expr: up{job="worker-node1"} == 0

for: 5m

#6 Ví dụ về annotation giúp hiển thị chú thích cho alert rule

- alert: node\_filesystem\_free\_percent

expr: 100 \* node\_filesystem\_free\_bytes{job="worker-node1"} / node\_filesystem\_size\_bytes{job="worker-node1"} < 70

annotations:

description: "filesystem {{.Labels.device}} on {{.Labels.instance}} is low on space, current available space is {{.Value}}"

- alert: LowDiskSpace

expr: 100 \* node\_filesystem\_free\_bytes{job="worker-node1"} / node\_filesystem\_size\_bytes{job="worker-node1"} < 10

labels:

severity: warning

environment: prod

annotations:

message: "node {{.Labels.instance}} is down"

#7 Tạo 1 rule giả để test AlertManager

- alert: 07-Node down TEST

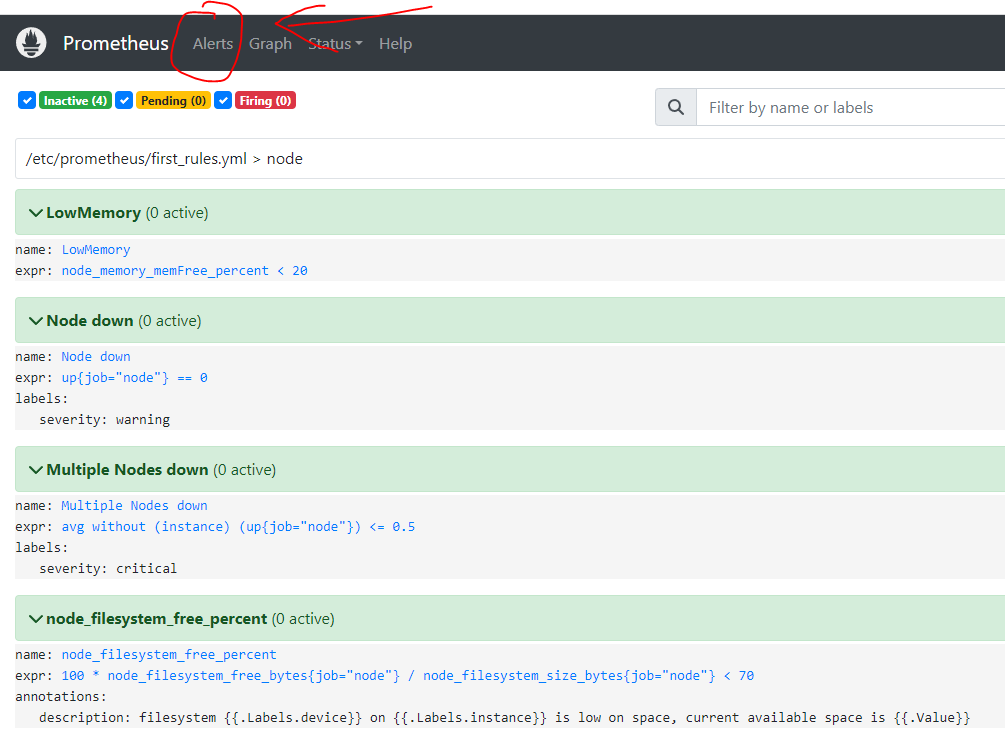
expr: up{job="worker-node1"} == 1

labels:

severity: test-only

Sau đó vào đây kiểm tra:

<http://192.168.88.111:9090/rules> và <http://192.168.88.111:9090/alerts>



### Slide 410: Alert with For

groups:

- name: node

rules:

- alert: node down

expr: up{job="node"} == 0

for: 5m

Lệnh for có chức năng node down trong 5 phút thì mới bắn firring cảnh báo

### Slide 416: Alert with Label

Label có thể cho vào expr để lọc kết quả chính xác hơn:

groups:

- name: node

rules:

- alert: Node down

expr: up{job="node"} == 0

labels:

severity: warning

- alert: Multiple Nodes down

expr: avg without(instance)(up{job="node"}) <= 0.5

labels:

severity: critical

### Slide 418: Alert with annotation label

#Ví dụ về annotation giúp hiển thị chú thích

- alert: node\_filesystem\_free\_percent

expr: 100 \* node\_filesystem\_free\_bytes{job="node"} / node\_filesystem\_siz e\_bytes{job="node"} < 70

annotations:

description: "filesystem {{.Labels.device}} on {{.Labels.instance}} is low on space, current available space is {{.Value}}"

## Lab – Alertmanager Installation

VD2: Tạo Alerting

# vi /etc/prometheus/rules.yaml

groups:

- name: node

rules:

- alert: LowDiskSpace

expr: 100 \* node\_filesystem\_free\_bytes{job="nodes"} / node\_filesystem\_size\_bytes{job="nodes"} < 10

labels:

severity: warning

environment: prod

annotations:

message: "node {{.Labels.instance}} is down"

Update the /etc/prometheus/prometheus.yml config file:

# vi /etc/prometheus/prometheus.yml

rule\_files:

- "/etc/prometheus/rules.yaml"

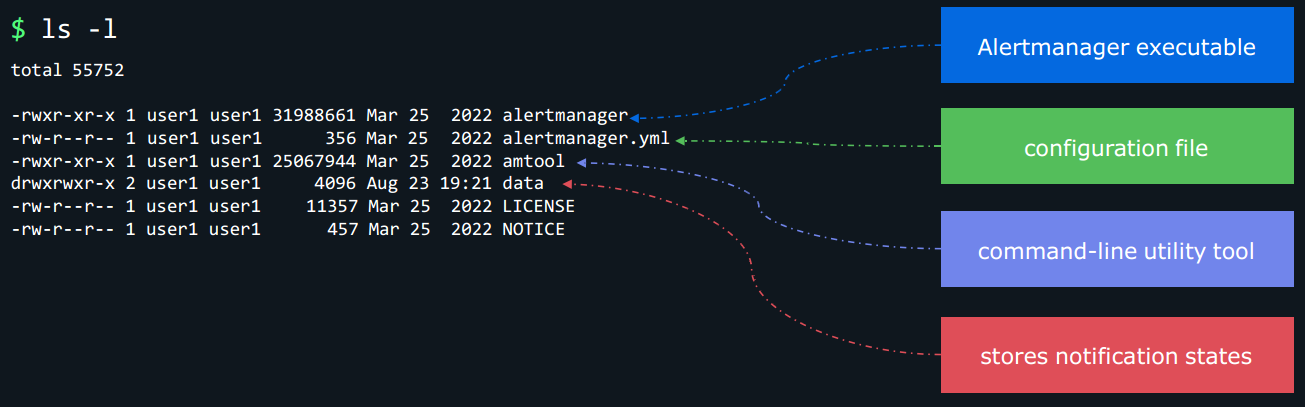
## Slide 422: Alertmanager

### Slide 427: Cài đặt Alertmanager

wget <https://github.com/prometheus/alertmanager/releases/download/v0.25.0/alertmanager-0.25.0.linux-amd64.tar.gz>

tar -xvzf alertmanager-0.25.0.linux-amd64.tar.gz

cd alertmanager-0.25.0.linux-amd64

 Chức năng từng file/folder

useradd --no-create-home --shell /bin/false alertmanager

mkdir /etc/alertmanager

/bin/cp -f alertmanager.yml /etc/alertmanager

chown -R alertmanager:alertmanager /etc/alertmanager

mkdir /var/lib/alertmanager ; chown -R alertmanager:alertmanager /var/lib/alertmanager

/bin/cp -f alertmanager /usr/local/bin

/bin/cp -f amtool /usr/local/bin

chown alertmanager:alertmanager /usr/local/bin/alertmanager

chown alertmanager:alertmanager /usr/local/bin/amtool

**Tạo systemd**

vi /etc/systemd/system/alertmanager.service

[Unit]

Description=Alert Manager

Wants=networkonline.target

After=networkonline.target

[Service]

Type=simple

User=alertmanager

Group=alertmanager

ExecStart=/usr/local/bin/alertmanager \

--config.file=/etc/alertmanager/alertmanager.yml \

--storage.path=/var/lib/alertmanager

Restart=always

[Install]

WantedBy=multi-user.target

systemctl daemon-reload

systemctl start alertmanager

**systemctl** enable alertmanager

**Cấu hình alertmanager trong prometheus.yml**

global:

scrape\_interval: 15s

evaluation\_interval: 15s

alerting:

alertmanagers:

- static\_configs:

- targets:

- 192.168.88.111:9093

**Cấu hình /etc/alertmanager/alertmanager.yml**

global:

smtp\_smarthost: 'mail.example.com:25'

smtp\_from: 'test@example.com'

route:

receiver: staff

group\_by: ['alertname', 'job']

routes:

- match\_re:

job: (node|windows)

receiver: infra-email

- matchers:

job: kubernetes

receiver: k8s-slack

receivers:

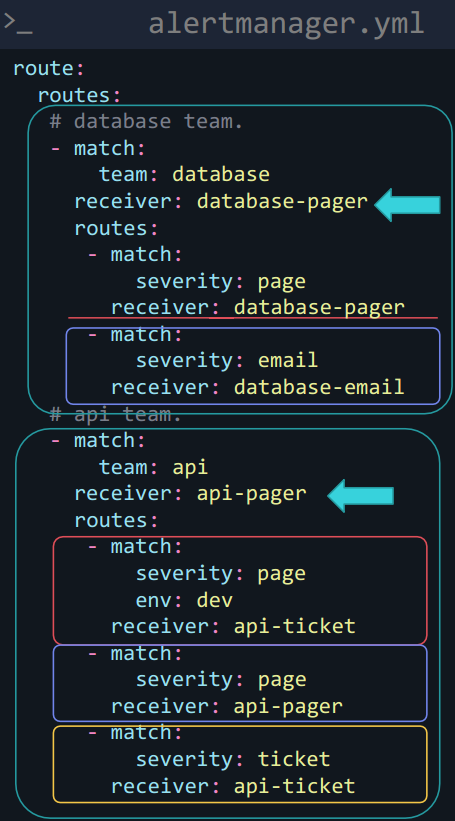
- name: 'k8s-slack'

slack\_configs:

- channel: '#alerts'

text: 'https://exampl.com/alerts/{{ .GroupLabels.app }}/

### VD2: về config route trong prometheus



### Slide 443: Default route

route:

receiver: staff

group\_by: ['alertname', 'job']

Trong cấu hình alertmanager.yml trên. Default route thể hiện ở chỗ này

Slide 445: route

routes:

- match\_re:

job: (node|windows)

receiver: infra-email

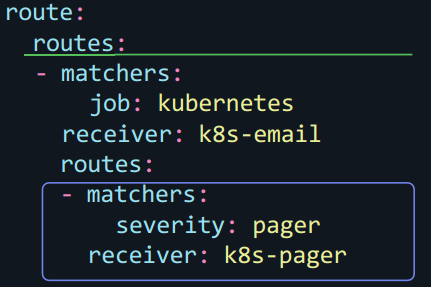
- matchers:

job: kubernetes

receiver: k8s-slack

Phân biệt match\_re và matchers là có thêm regular expresstion

### Slide 446: Sub-route



### Slide 449: Continue alertmanager (multi receiver)

Mặc định 1 cảnh báo prometheus sẽ nhận route đầu tiên bắn và không đến route tiếp theo nữa. Để bắn nhiều receiver ta sử dụng option Continue trong promethes

route:

routes:

- receiver: 'general-email'

continue: true

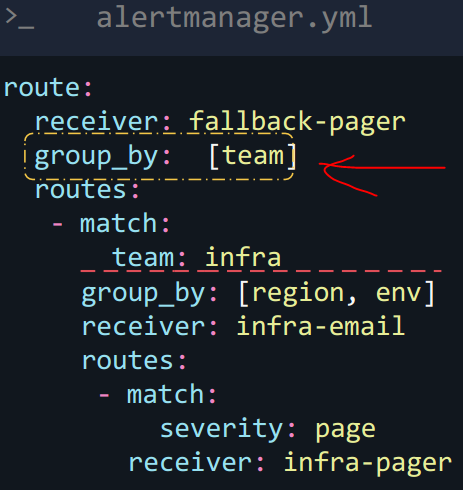
- matcher:

job: kubernetes

receiver: global-infra-email

### Slide 450 Group\_by trong Alertmanager

Giúp chúng ta nhóm nhãn lại khi cảnh báo (ví dụ group theo serverity, team infra-team customer...)



### Slide 452: Groupwait, group\_interval, repeat\_interval

route:

receiver: fallback-pager

group\_by: [team]

group\_wait: 10s

group\_interval: 5m

repeat\_interval: 4h

group\_wait: ví dụ có 3 cảnh báo trùng nhau trong 2 giây, thì tổng hợp cuối cùng 10 giây mới có 1 cảnh báo. Giúp tránh bắn nhiều lần cảnh báo (mặc định là 30s)

group\_interval: Tổng kết toàn bộ cảnh báo trong 5 phút: rồi bắn 1 lần (gộp lại)

repeat\_interval: Cứ trong vòng 4h, lại nhắc lại cảnh báo đều đặn

### Slide 458: Receiver and notifiers

#### Ví dụ về Webhook (đã test ok):

route:

group\_by: ['alertname']

group\_wait: 30s

group\_interval: 5m

repeat\_interval: 1h

receiver: 'web.hook'

receivers:

- name: 'web.hook'

webhook\_configs:

- url: 'http://127.0.0.1:5001/'

#### Ví dụ về telegram (đã test ok):

route:

group\_by: ['alertname']

group\_wait: 10s

group\_interval: 20s

repeat\_interval: 30s

receiver: 'telegram'

receivers:

- name: 'web.hook'

webhook\_configs:

- url: 'http://127.0.0.1:5001/'

- name: telegram

telegram\_configs:

- api\_url: https://api.telegram.org

bot\_token: 1152275874:xxxxxxxxxxxxxxxxxxx6aPmR0RgDSA

chat\_id: -82xxxxxxxx

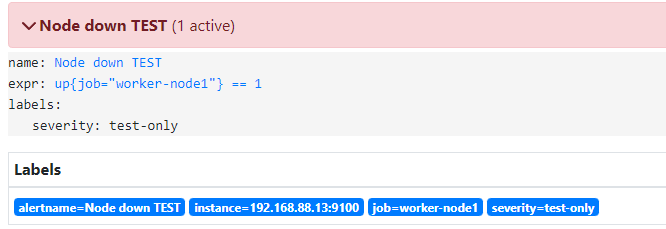
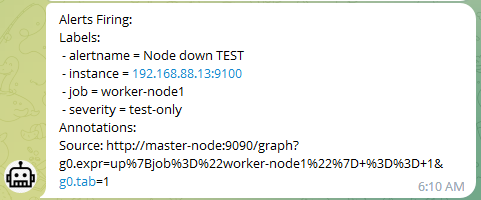
disable\_notifications: false

http\_config:

follow\_redirects: true

send\_resolved: true

parse\_mode: ""



### Slide 461 Reveiver global config

### Slide 463: chỉnh Nội dung tin cảnh báo

### Slide 474: Silences

## Lab - AlertManager 2

VD2+3: Tạo 1 rule thông báo hết ram khi 95%

# vi /etc/prometheus/rules.yaml

- name: node

rules:

- alert: NodeDown

expr: up{job="nodes"} == 0

for: 10s

labels:

severity: critical

team: global-infra

- alert: HostOutOfMemory

expr: node\_memory\_MemAvailable\_bytes{job="nodes"} / node\_memory\_MemTotal\_bytes{job="nodes"} \* 100 < 95

labels:

severity: warning

team: internal-infra

annotations:

message: "node {{.Labels.instance}} is seeing high memory usage, currently available memory: {{.Value}}%"

VD5: Cấu hình alertmanager.yml cảnh báo

# vi /etc/alertmanager/alertmanager.yml

global:

smtp\_smarthost: 'localhost:25'

smtp\_from: 'alertmanager@prometheus-server.com'

route:

group\_by: ['alertname']

group\_wait: 10s

group\_interval: 2m

repeat\_interval: 1h

receiver: 'general-email'

routes:

- match:

team: global-infra

receiver: global-infra-email

- match:

team: internal-infra

receiver: internal-infra-email

receivers:

- name: 'web.hook'

webhook\_configs:

- url: 'http://127.0.0.1:5001/'

- name: global-infra-email

email\_configs:

- to: "root@prometheus-server.com"

require\_tls: **false**

- name: internal-infra-email

email\_configs:

- to: "admin@prometheus-server.com"

require\_tls: **false**

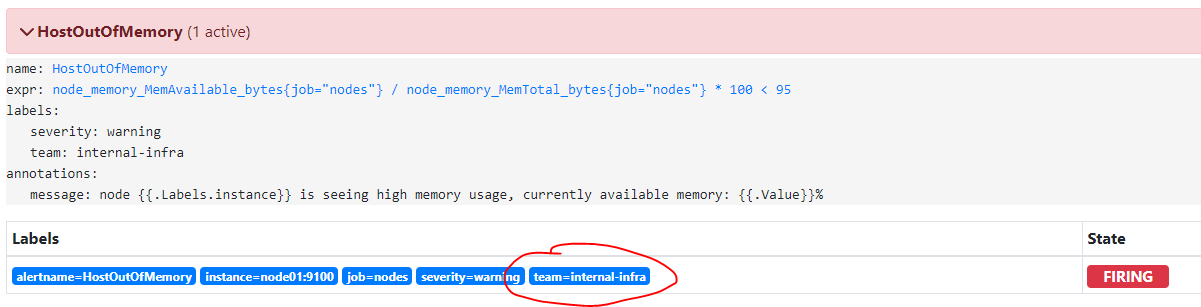
- name: general-email

email\_configs:

- to: "admin@prometheus-server.com"

require\_tls: **false**

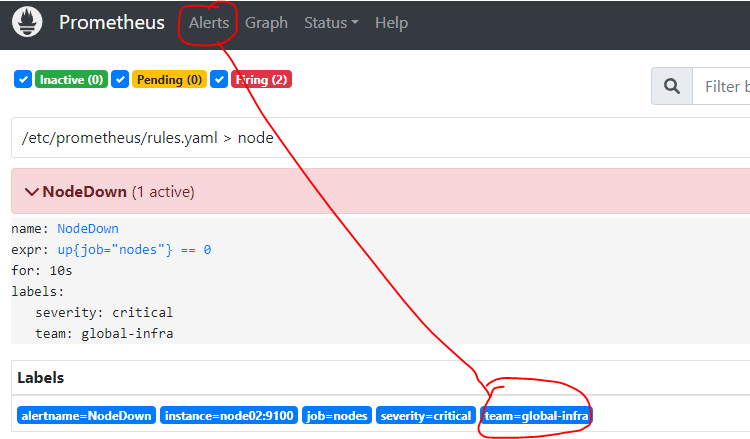
VD7:



Khi test để ram 95%, thì HostOutOfMemory sẽ cảnh báo. Nó có chứa nhãn team: internal-infra trong file rules.yaml.

* Đối chiếu alertmanager routes, ta sẽ thấy nó sẽ bắn vào reccevier internal-infra-email

VD8:



Nhãn **team=global-infra** trong rules.yaml. Đối chiếu alertmanager ta sẽ thấy bắn vào global-infra-mail root@prometheus-server.com

# Phần 10: Monitoring Kubernetes

## Lab – Kubernetes & Prometheus

**B1: Cài đặt helm**

wget https://get.helm.sh/helm-v3.10.3-linux-amd64.tar.gz

tar -zxvf helm-v3.10.3-linux-amd64.tar.gz

mv linux-amd64/helm /usr/local/bin/

rm -rf helm-v3.10.3-linux-amd64.tar.gz linux-amd64

**B2: Cài prometheus stack**

helm repo add prometheus-community https://prometheus-community.github.io/helm-charts

helm repo update

helm search repo prometheus-community/kube-prometheus-stack --versions

helm pull prometheus-community/kube-prometheus-stack --version 29.0.1

tar -xzf kube-prometheus-stack-29.0.1.tgz

cd kube-prometheus-stack

Ta sửa: values.yaml ta sửa admissionWebhooks: enable: false

helm -n monitoring install prometheus -f values.yaml .

kubectl -n monitoring get ds

kubectl -n monitoring patch ds prometheus-prometheus-node-exporter --type "json" -p '[{"op": "remove", "path" : "/spec/template/spec/containers/0/volumeMounts/2/mountPropagation"}]'

**B3: Kiểm tra + giải thích chức năng deployment và sts**

# k get deployments.apps

NAME READY UP-TO-DATE AVAILABLE AGE

prometheus-kube-prometheus-operator 1/1 1 1 103s

prometheus-kube-state-metrics 1/1 1 1 103s

prometheus-grafana 1/1 1 1 103s

Chức năng của 3 deployment:

prometheus-grafana - This is the grafana instance that gets deployed with the helm chart.

prometheus-kube-prometheus-operator - Responsible for deploying & managing the prometheus instance.

prometheus-kube-state-metrics - Collects cluster level metrics (pods, deployments, etc).

# kubectl get statefulset

NAME READY AGE

alertmanager-prometheus-kube-prometheus-alertmanager 1/1 2m48s

prometheus-prometheus-kube-prometheus-prometheus 1/1 2m46s

Chức năng của 2 statefullset

prometheus-prometheus-kube-prometheus-prometheus - This is the prometheus instance.

alertmanager-prometheus-kube-prometheus-alertmanager - This is the alertmanager instance.

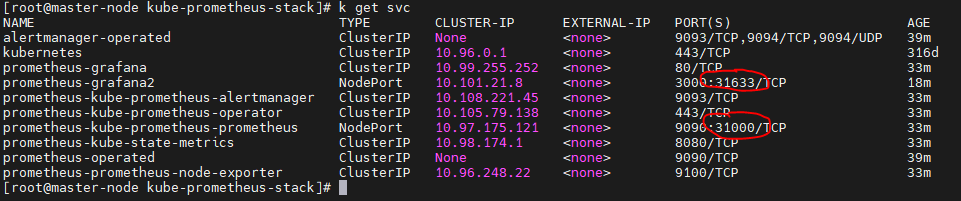
**Đổi lại svc về NodePort (nếu cần)**

# kubectl edit svc prometheus-kube-prometheus-prometheus

Đổi ClusterIP thành NodePort với nodePort: 31000

# k expose deployment prometheus-grafana --type=NodePort --port=3000 --target-port=3000 --name prometheus-grafana2

Truy cập vào grafana với mật khẩu như sau: admin / prom-operator



**B4.1: Cài đặt API-Application test**

Let’s now deploy an application to the Kubernetes cluster. There is an API built with NodeJS that listens on port 3000. It already has an instrumentation setup and the metrics can be accessed at /swagger-stats/metrics. It has two endpoints /recipes, and /ingredients.

# cat /root/api-deploy.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: api-deployment

labels:

app: api

spec:

replicas: 2

selector:

matchLabels:

app: api

template:

metadata:

labels:

app: api

spec:

containers:

- name: api

image: sanjeevkt720/prometheus-lab-demo

ports:

- containerPort: 3000

# k apply -f api-deploy.yaml

# k exec -it api-deployment-597dcbb5c9-dp7r9 -- curl localhost:3000

Hello World!

# k exec -it api-deployment-597dcbb5c9-dp7r9 -- curl localhost:3000/swagger-stats/metrics

nodejs\_process\_memory\_rss\_bytes 57020416

nodejs\_process\_memory\_heap\_total\_bytes 15388672

...

api\_response\_size\_bytes\_sum{method="GET",path="/",code="200"} 24

api\_response\_size\_bytes\_count{method="GET",path="/",code="200"} 2

# cat /root/api-service.yaml

---

apiVersion: v1

kind: Service

metadata:

name: api-service

labels:

job: node-api

app: api

spec:

type: NodePort

selector:

app: api

ports:

- name: web

protocol: TCP

port: 3000

targetPort: 3000

nodePort: 30112

# curl 192.168.88.12:30112/swagger-stats/metrics

**B4.2: Cấu hình prometheus scrape metrics của api application**

# cat /root/api-servicemonitor.yaml

apiVersion: monitoring.coreos.com/v1

kind: ServiceMonitor

metadata:

name: api-service-monitor

labels:

app: prometheus

release: prometheus

spec:

jobLabel: job

endpoints:

- interval: 30s

port: web

path: **/swagger-stats/metrics**

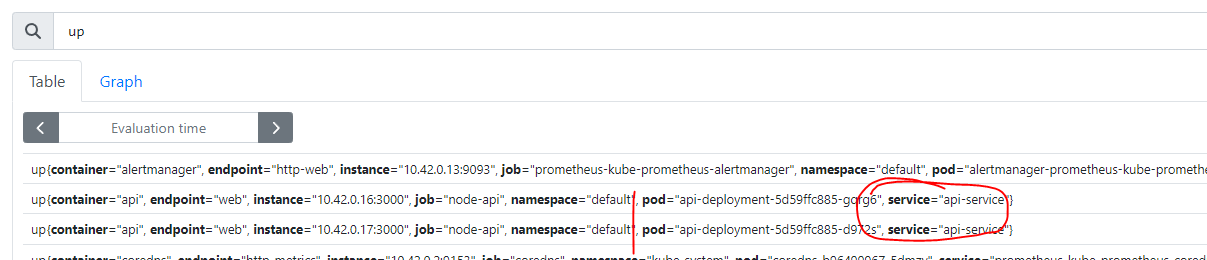
selector:

matchLabels:

app: **api**

# kubectl apply -f api-servicemonitor.yaml

Sau khi apply xong, 1 lúc sẽ ra như sau:



**B5: Cấu hình alert**

Custom resource define này sẽ quản lý yaml alert

# k get crd prometheusrules.monitoring.coreos.com -o yaml

Tạo rule:

Yêu cầu rule tên HostOutOfMemory, với expression là node\_memory\_MemAvailable\_bytes / node\_memory\_MemTotal\_bytes \* 100 < 10.

# cat rules.yaml

apiVersion: monitoring.coreos.com/v1

kind: PrometheusRule

metadata:

labels:

#PLACEHOLDER: Add label here (để lấy labels chính xác ta chạy: kubectl get prometheus -o yaml | grep -A 3 -i ruleSelector)

release: prometheus

name: example-rules

spec:

groups:

- name: node-example

rules:

- record: node\_filesystem\_free\_percent

expr: 100\* node\_filesystem\_free\_bytes / node\_filesystem\_size\_bytes

- alert: HostOutOfMemory

expr: node\_memory\_MemAvailable\_bytes / node\_memory\_MemTotal\_bytes \* 100 < 10

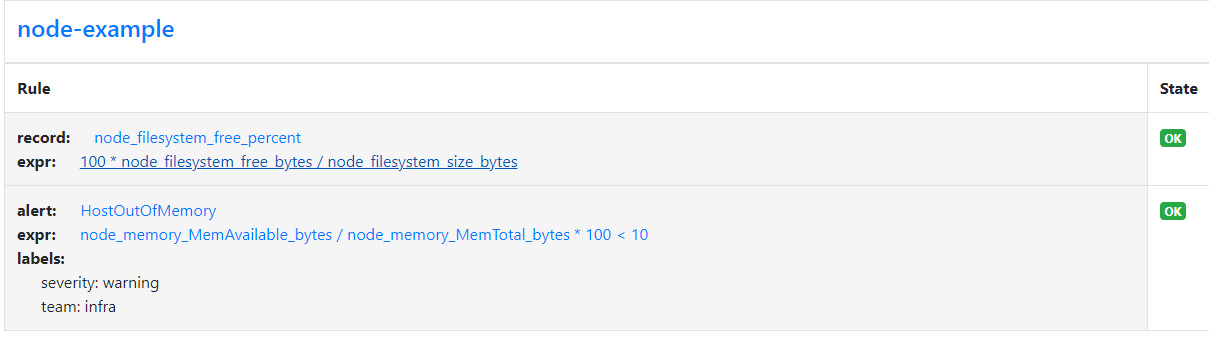
labels:

team: infra

severity: warning

Sau khi cấu hình placeholder chính xác, ta sẽ có ảnh sau

<http://192.168.88.12:31000/rules>



**B6: Tạo alert-manager**



# cat /root/alertmanager-rule.yaml

apiVersion: monitoring.coreos.com/v1alpha1

kind: AlertmanagerConfig

metadata:

name: alert-config

labels:

release: prometheus

spec:

route:

groupBy: ["alertname"]

groupWait: 30s

groupInterval: 5m

repeatInterval: 12h

receiver: "webhook"

routes:

- matchers:

- name: team

value: infra

receiver: "infra"

groupBy: ["severity"]

receivers:

- name: "webhook"

webhookConfigs:

- url: "http://example.com/"

- name: "infra"

webhookConfigs:

- url: "http://infra.com/"

**B7: Sửa lại ingress đưa api-service, prometheus, alert-manager ra ngoài**

# kubectl get ingress prom-ingress -o yaml

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: prom-ingress

annotations:

# kubernetes.io/ingress.class: nginx

namespace: default

spec:

rules:

- host: prometheus.kk-demo.com

http:

paths:

- backend:

service:

name: prometheus-kube-prometheus-prometheus

port:

number: 9090

path: /

pathType: Prefix

- host: api.kk-demo.com

http:

paths:

- backend:

service:

name: api-service

port:

number: 3000

path: /

pathType: Prefix

- host: alertmanager.kk-demo.com

http:

paths:

- backend:

service:

name: prometheus-kube-prometheus-alertmanager

port:

number: 9093

path: /

pathType: Prefix

# Note querry

up : hiển thị all job đang hoạt động

up{job="web"}

node\_cpu\_seconds\_total{job="web", cpu="0",mode=~"user|system"} : HOẶC yêu cầu kèm Exception =~ để enable HOẶC

max\_over\_time(node\_filesystem\_avail\_bytes[10m])

<https://prometheus.io/docs/prometheus/latest/configuration/configuration/>

nên đọc