Monitoring with Prometheus and Grafana
Quick Guide

Version 25.07

Ahmed Abdelwahed
ahmed@abdelwahed.me
www.abdelwahed.me
LinkedIn

What is Prometheus?

Prometheus is a **tool for monitoring systems**. It collects data (metrics) from your servers, containers, and apps, and stores it over time.
You can **query the data**, **create alerts**, and **visualize it** using dashboards like Grafana.

Prometheus Components

Component	What it Does
Prometheus	**The main engine that pulls and stores data**
Exporters	**Tiny tools that give system/app data to Prometheus**
Alertmanager	**Sends alerts (e.g., to Slack, email)**
Grafana	**Nice dashboards to visualize metrics**

Pull vs Push (How Prometheus Gets Data)

- Prometheus **pulls** data from targets (servers, exporters)
- Most other tools use **push**, but pull gives more control and simplicity

What is a Metric?

A **metric** is one piece of monitoring data (like CPU usage or disk space).
Metrics have:

- A name: node_cpu_seconds_total
- Labels: instance="server1", cpu="0"
- A value: 1234.5
- A timestamp

Lab1: Deploying a Complete Monitoring Stack (Prometheus, Node Exporter, and Grafana) on Rocky Linux Using Docker

Step 1: Update System & Install Docker

```
sudo dnf update -y
sudo dnf install -y yum-utils
sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
sudo dnf install -y docker-ce docker-ce-cli containerd.io
```

Enable & start Docker

```
sudo systemctl enable docker
sudo systemctl start docker
sudo usermod -aG docker $USER
```

Verify:

```
docker --version
```

Step 2: Create Docker Network

Create a **custom bridge network** for Prometheus and Grafana to talk securely:

```
Sudo docker network create monitor-net
```

Step 3: Setup Prometheus Configuration

Create folder:

```
mkdir -p ~/prometheus
cd ~/prometheus
```

Create prometheus.yml:

```
nano prometheus.yml
```

Paste this content (replace YOUR_VM_PUBLIC_IP):

```
global:
  scrape_interval: 15s

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

  - job_name: 'node_exporters'
    static_configs:
      - targets:
          - 'localhost:9100'
          - '135.119.83.142:9100'
```

Use your **actual Azure VM public IP**, not localhost.

Step 4: Run Prometheus

```
sudo docker run -d \
  --name prometheus \
  --restart unless-stopped \
```

```
  --network monitor-net \
  -p 9090:9090 \
  -v ~/prometheus/prometheus.yml:/etc/prometheus/prometheus.yml \
  prom/prometheus \
  --config.file=/etc/prometheus/prometheus.yml
```

Step 5: Run Node Exporter

```
sudo docker run -d \
  --name node_exporter \
  --restart unless-stopped \
  --network host \
  -p 9100:9100 \
  prom/node-exporter
```

✅ --network host ensures Prometheus can reach it using public IP.

Step 6: Run Grafana

```
sudo docker run -d \
  --name grafana \
  --restart unless-stopped \
  --network monitor-net \
  -p 3000:3000 \
  -e GF_SECURITY_ADMIN_USER=admin \
  -e GF_SECURITY_ADMIN_PASSWORD=admin \
  grafana/grafana
```

Step 7: Verify Services

Prometheus

Visit: `http://YOUR_VM_PUBLIC_IP:9090`

Node Exporter Target

- Inside Prometheus UI → Status → Targets
- Should show YOUR_VM_PUBLIC_IP:9100 as UP

Grafana

Visit: `http://YOUR_VM_PUBLIC_IP:3000`
Login: admin / admin (or as set)

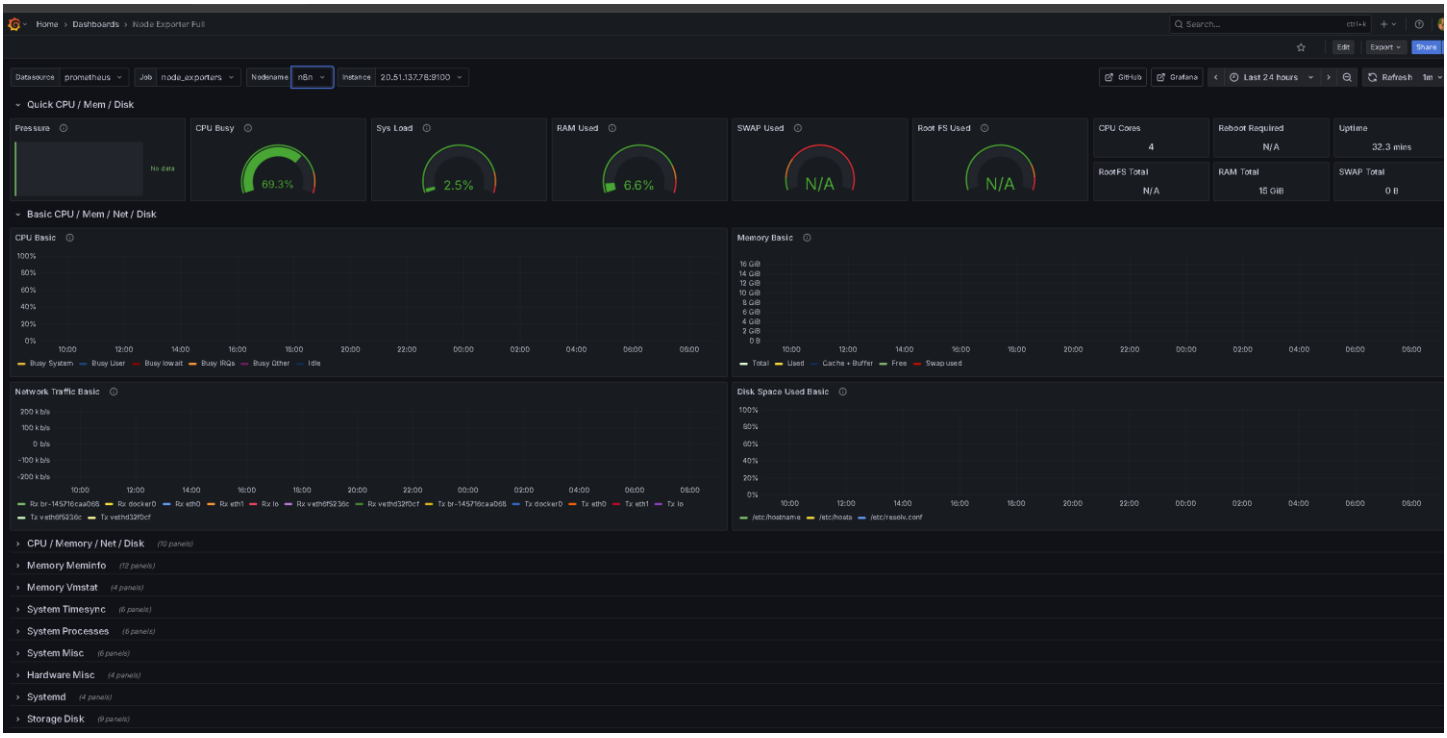
STEP 8: Configure Grafana

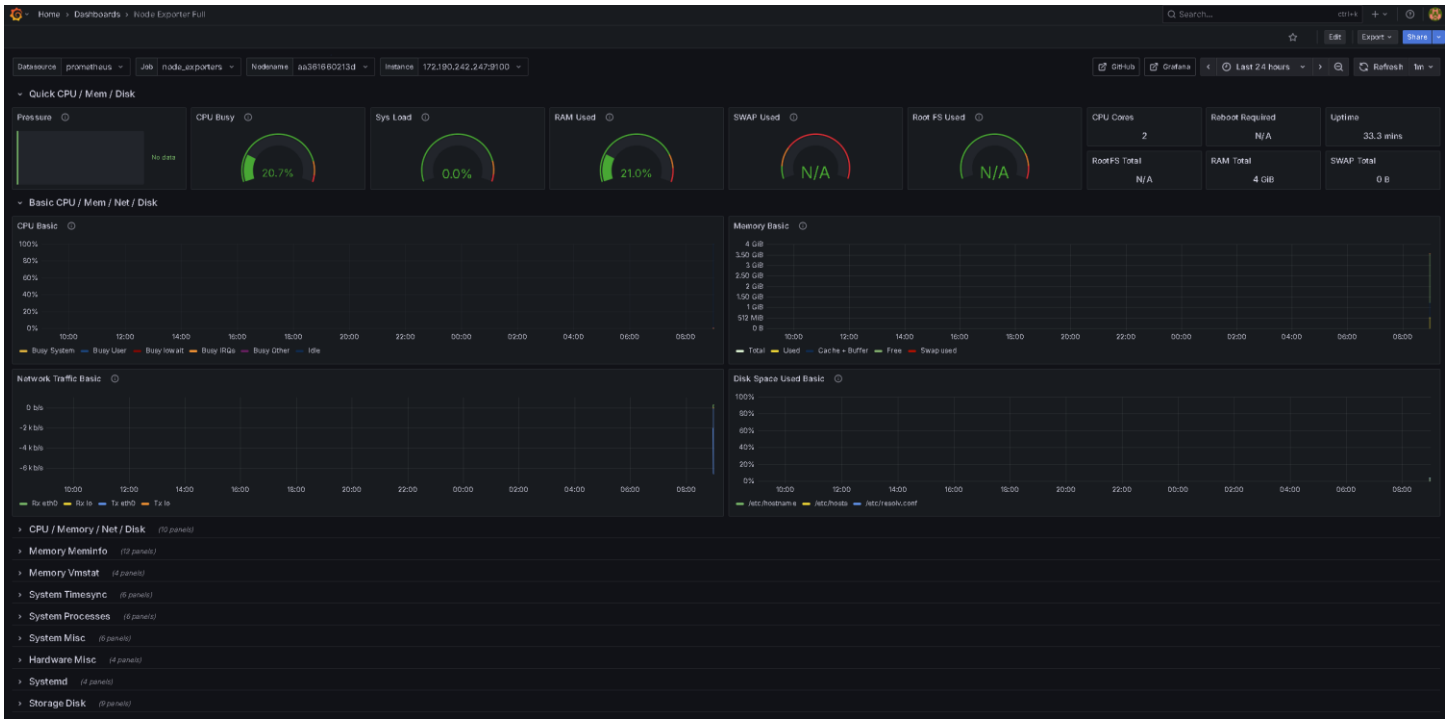
A. Add Data Source

1. Grafana → ⚙ connection → **Data Sources**
2. Choose **Prometheus**
3. Set URL: http://prometheus:9090
4. Click **Save & Test**

B. Import Dashboard

1. Grafana → Dashboards → Import
2. Use ID `1860` (Node Exporter Full)
3. Select Prometheus as data source
4. Click **Import**


Home › Dashboards › Node Exporter Full
Quick CPU / Mem / Disk
Pressure
CPU Busy
69.3%
Sys Load
2.5%
RAM Used
6.6%
SWAP Used
N/A
Root FS Used
N/A
CPU Cores
4
Reboot Required
N/A
Uptime
32.3 mins
RootFS Total
N/A
RAM Total
15 GiB
SWAP Total
0 B
Basic CPU / Mem / Net / Disk
CPU Basic
Memory Basic
Network Traffic Basic
Disk Space Used Basic
CPU / Memory / Net / Disk
Memory Meminfo
Memory Vmstat
System Timesync
System Processes
System Misc
Hardware Misc
Systemd
Storage Disk


Home › Dashboards › Node Exporter Full
Quick CPU / Mem / Disk
Pressure
CPU Busy
20.7%
Sys Load
0.0%
RAM Used
21.0%
SWAP Used
N/A
Root FS Used
N/A
CPU Cores
2
Reboot Required
N/A
Uptime
33.3 mins
RootFS Total
N/A
RAM Total
4 GiB
SWAP Total
0 B
Basic CPU / Mem / Net / Disk
CPU Basic
Memory Basic
Network Traffic Basic
Disk Space Used Basic
CPU / Memory / Net / Disk
Memory Meminfo
Memory Vmstat
System Timesync
System Processes
System Misc
Hardware Misc
Systemd
Storage Disk

Useful Commands

View logs

```
docker logs prometheus
docker logs grafana
docker logs node_exporter
```

Restart a container

```
docker restart grafana
```

Stop and remove

```
docker stop prometheus grafana node_exporter
docker rm prometheus grafana node_exporter
```

Clean-up (optional)

```
docker rm -f prometheus grafana node_exporter
docker network rm monitor-net
```

Lab 2: Container Performance Monitoring Lab (cAdvisor, Prometheus & Grafana)

Setup Overview

- **New Node (Rocky Linux)** already added to Prometheus
- We'll deploy containers **with systemd or Docker's restart policy**
- Metrics will show in **Grafana**

Step-by-Step Lab

1. On New Rocky Node: Prerequisites

```
sudo dnf install -y docker
sudo systemctl enable --now docker
```

Create a network (optional but cleaner):

```
docker network create app-net
```

2. Run Containers (e.g., NGINX, Redis, Alpine)

```
# Run NGINX
docker run -d \
  --name nginx \
  --restart unless-stopped \
  --network app-net \
  -p 80:80 \
  nginx
```

Run Redis

```
docker run -d \
  --name redis \
  --restart unless-stopped \
  --network app-net \
  redis
```

Run an Alpine container that prints uptime every 5s

```
docker run -d \
  --name logger \
  --restart unless-stopped \
  --network app-net \
  alpine sh -c "while true; do uptime; sleep 5; done"
```

All containers will:

- Restart on reboot
- Run in the background without SSH dependency

3. Ensure Node Exporter Captures Docker Metrics

By default, **Node Exporter doesn't monitor Docker containers** directly.

To expose Docker/container metrics:

Option A: 🧩 Install cAdvisor (by Google)

```
docker run -d \
  --name=cadvisor \
  --restart unless-stopped \
```

```
  --network app-net \
  -p 8080:8080 \
  -v /:/rootfs:ro \
  -v /var/run:/var/run:ro \
  -v /sys:/sys:ro \
  -v /var/lib/docker/:/var/lib/docker:ro \
  gcr.io/cadvisor/cadvisor
```

Option B: 💉 Use Prometheus Docker Daemon metrics (advanced — skip for now)

4. Update Prometheus Config (on controller)

Add to prometheus.yml:

```
- job_name: 'cadvisor'
  static_configs:
    - targets: ['<node-ip>:8080']
```

Replace <node-ip> with the IP of your new Rocky VM.

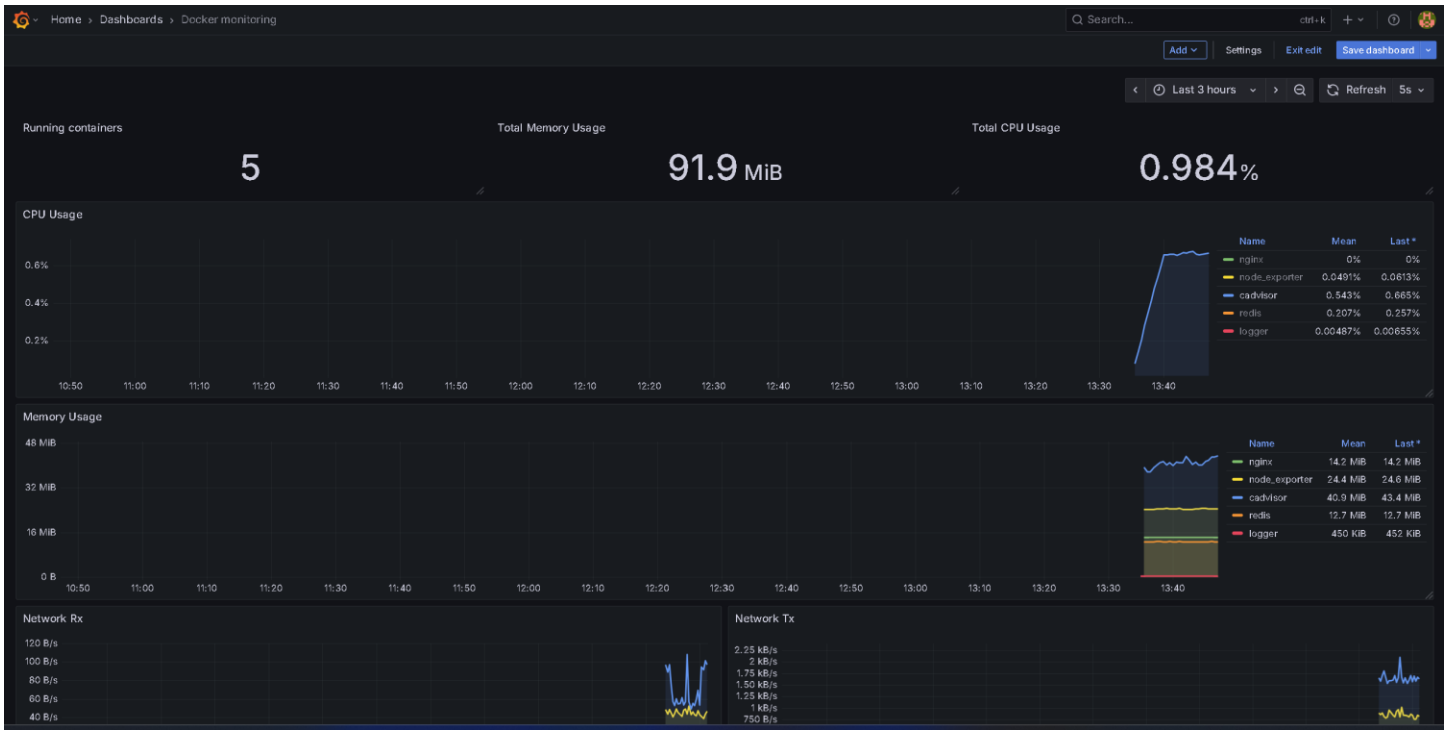
Then:

```
docker restart prometheus
```

5. Import Grafana Dashboard

Import cAdvisor dashboard:

1. Go to Grafana → Dashboards → Import
2. Use Dashboard ID: `193`
   or JSON from `https://grafana.com/grafana/dashboards/193`
3. Set Prometheus as the data source
4. View Docker container metrics in Grafana



Lab 3: Windows Server Monitoring with Prometheus and Grafana

Install and configure Windows Exporter with AD and DNS collectors, connect it to Prometheus, and visualize metrics in Grafana

On your Windows Server — Install & configure & Start Windows Exporter

```
cd $env:USERPROFILE\Downloads
msiexec /i windows_exporter-0.31.3-amd64.msi
ENABLED_COLLECTORS="[defaults],ad,dns" LISTEN_PORT=9182 /qn
Start-Service windows_exporter
Set-Service -Name windows_exporter -StartupType Automatic
```

Allow windows exporter port

```
New-NetFirewallRule -DisplayName "windows_exporter 9182" -Direction Inbound -Action
Allow -Protocol TCP -LocalPort 9182
```

On your Prometheus Server — Add Windows target

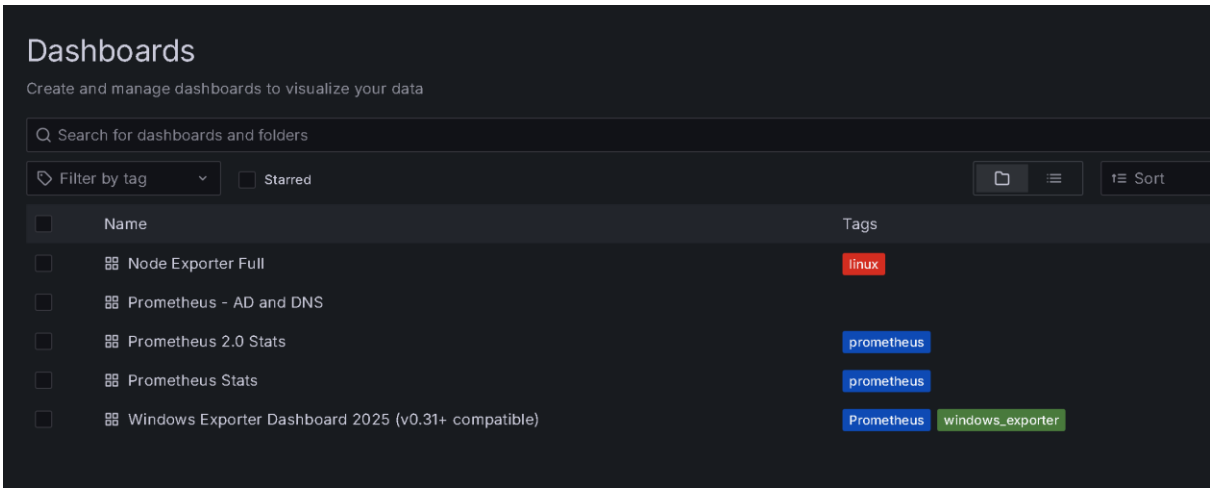
In this lab, I added two Windows servers: 10.0.0.4 and 10.0.0.5.

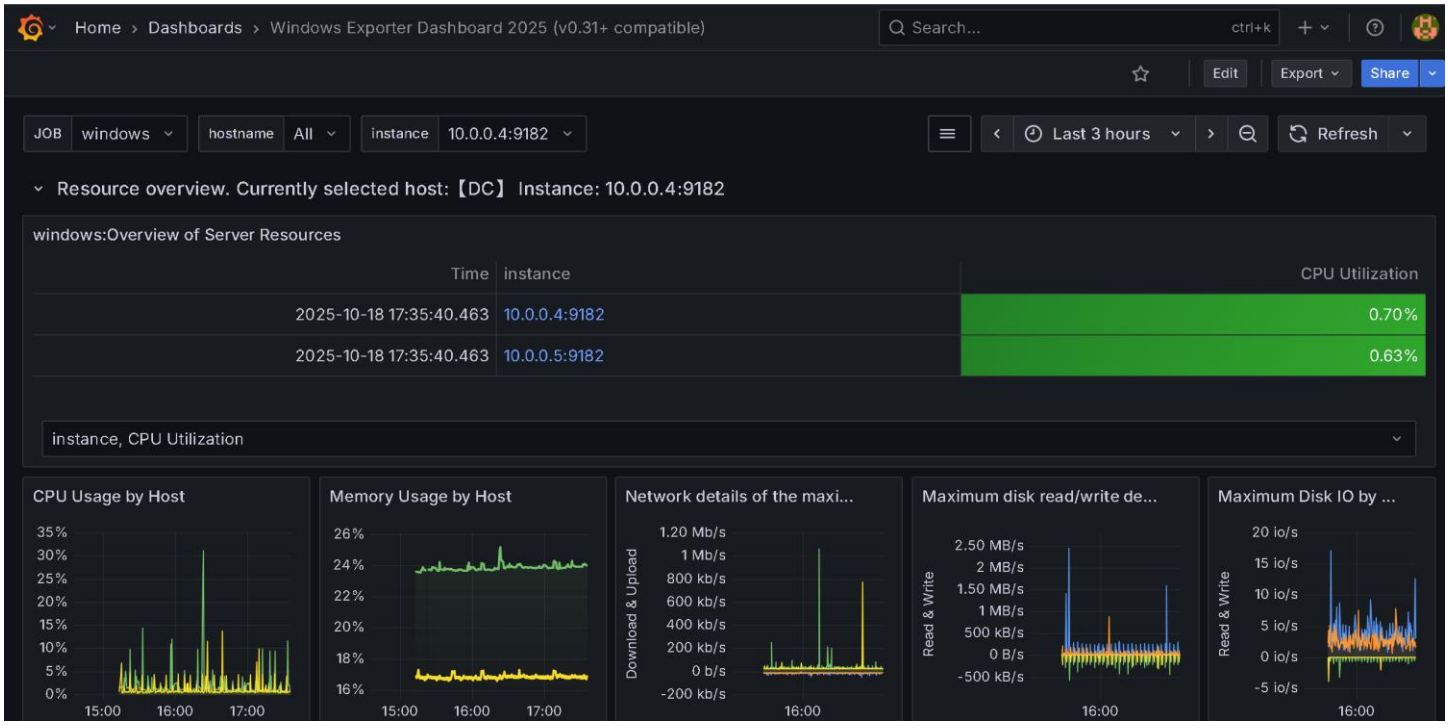
```
global:
  scrape_interval: 15s
scrape_configs:
  - job_name: 'windows'
    static_configs:
      - targets:
        - '10.0.0.4:9182'   # DC (AD/DNS collectors)
        - '10.0.0.5:9182'   # non-DC
# (optional) keep Prometheus' own metrics
  - job_name: 'prometheus'
    static_configs:
      - targets: ['10.0.0.6:9090']
```

Restart both grafana and promethus

```
docker restart prometheus Grafana
```

In Grafana — Import Windows dashboards in Grafana using IDs 21243 and 20920.





Lab 4: Linux Server Monitoring with Node Exporter, Prometheus, and Grafana

Install and configure Node Exporter on a Linux server (10.0.0.7), integrate it with Prometheus for metric collection, and visualize the data in Grafana

Download the latest Node Exporter:

```
cd /opt
curl -LO
https://github.com/prometheus/node_exporter/releases/download/v1.8.2/node_exporter-1.8.2.linux-amd64.tar.gz
tar xzf node_exporter-1.8.2.linux-amd64.tar.gz
mv node_exporter-1.8.2.linux-amd64 node_exporter
cd node_exporter
./node_exporter –version
```

Create a service account:

```
useradd --no-create-home --shell /bin/false nodeusr
```

Create a systemd service:

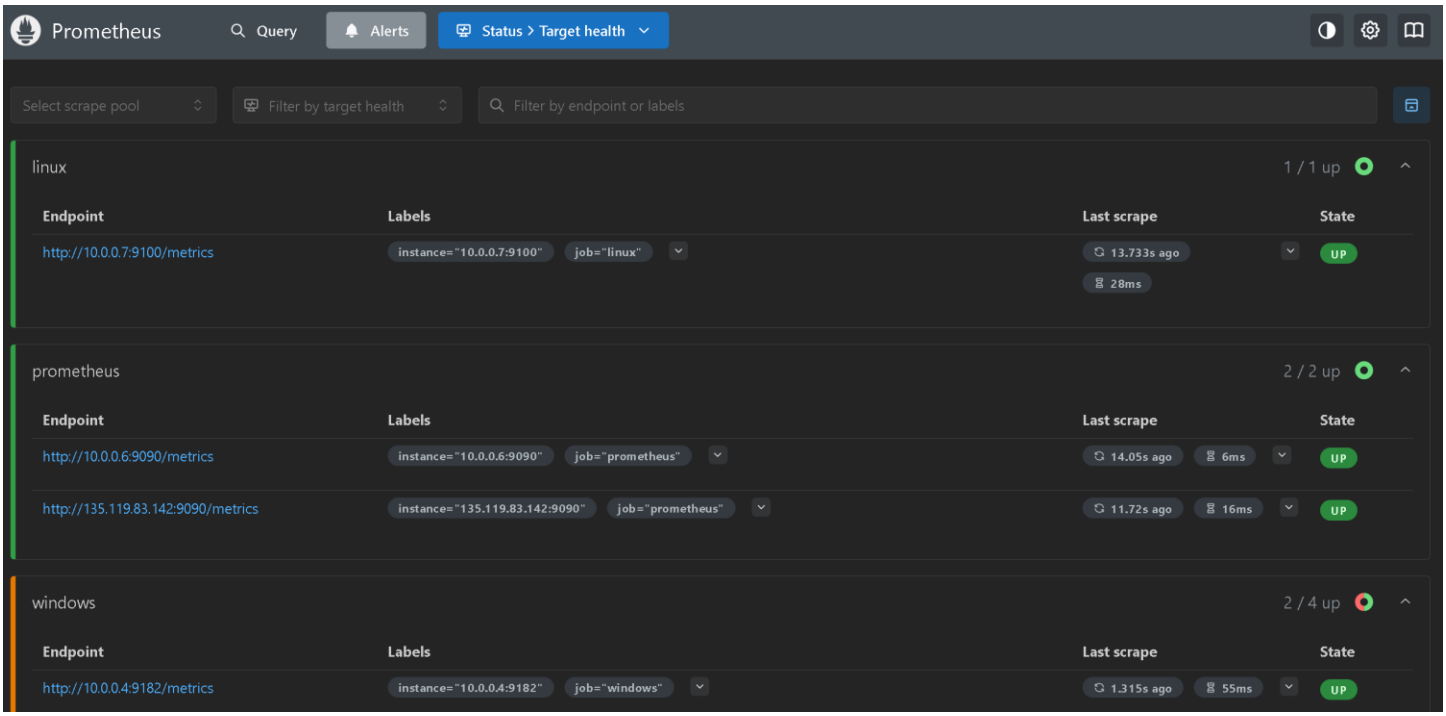
```
cat << 'EOF' | tee /etc/systemd/system/node_exporter.service
[Unit]
Description=Node Exporter
After=network.target

[Service]
User=nodeusr
ExecStart=/opt/node_exporter/node_exporter
Restart=always

[Install]
WantedBy=multi-user.target
EOF
```

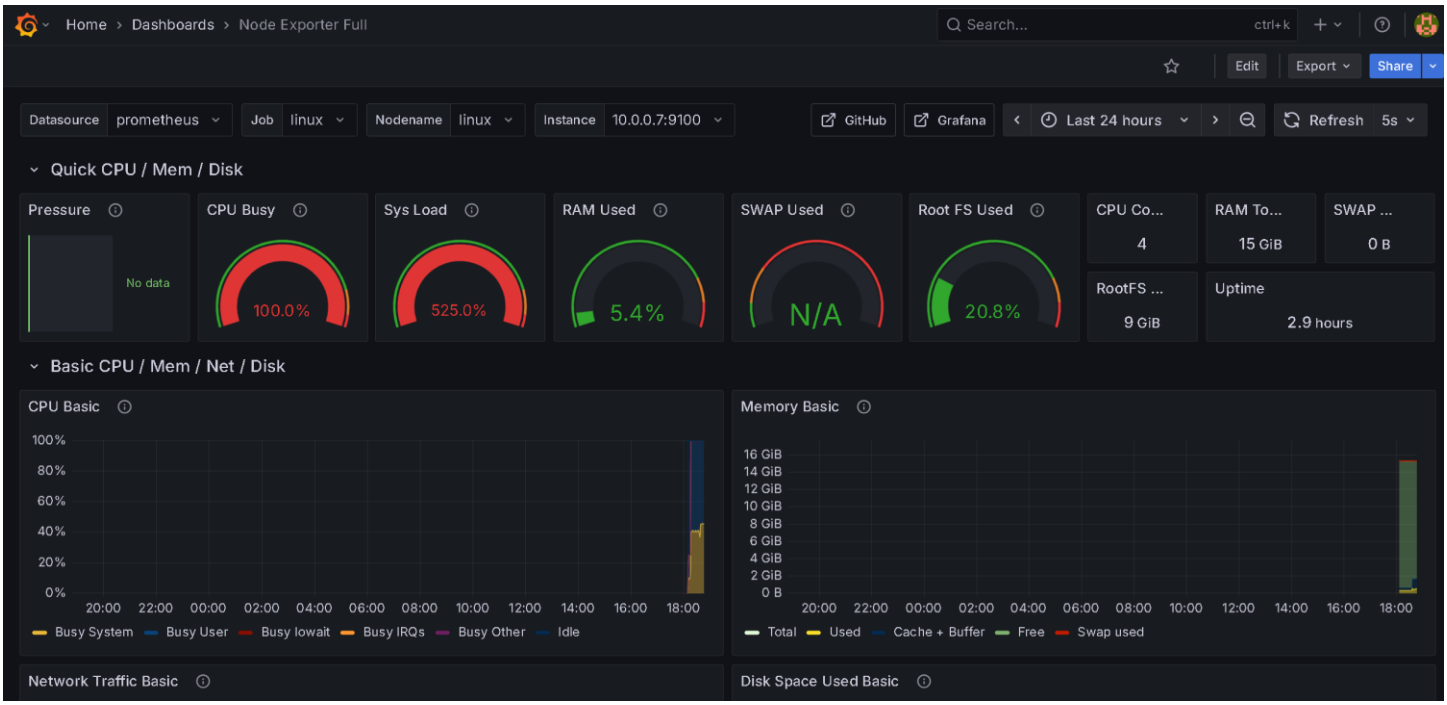
Enable and start:

```
systemctl daemon-reload
systemctl enable --now node_exporter
systemctl status node_exporter --no-pager
```

Add Linux Node to Prometheus Configuration

```
  - job_name: 'linux'
    static_configs:
      - targets: ['10.0.0.7:9100']
```


Prometheus
Query
Alerts
Status > Target health
Select scrape pool
Filter by target health
Filter by endpoint or labels
linux
1 / 1 up
Endpoint
Labels
Last scrape
State
http://10.0.0.7:9100/metrics
instance="10.0.0.7:9100"
job="linux"
13.733s ago
28ms
UP
prometheus
2 / 2 up
http://10.0.0.6:9090/metrics
instance="10.0.0.6:9090"
job="prometheus"
14.05s ago
6ms
http://135.119.83.142:9090/metrics
instance="135.119.83.142:9090"
11.72s ago
16ms
windows
2 / 4 up
http://10.0.0.4:9182/metrics
instance="10.0.0.4:9182"
job="windows"
1.315s ago
55ms


Home > Dashboards > Node Exporter Full
Search...
ctrl+k
Edit
Export
Share
Datasource prometheus
Job linux
Nodename linux
Instance 10.0.0.7:9100
GitHub
Grafana
Last 24 hours
Refresh 5s
Quick CPU / Mem / Disk
Pressure
No data
CPU Busy
100.0%
Sys Load
525.0%
RAM Used
5.4%
SWAP Used
N/A
Root FS Used
20.8%
CPU Co...
4
RAM To...
15 GiB
SWAP ...
0 B
RootFS ...
9 GiB
Uptime
2.9 hours
Basic CPU / Mem / Net / Disk
CPU Basic
100%
80%
60%
40%
20%
0%
20:00 22:00 00:00 02:00 04:00 06:00 08:00 10:00 12:00 14:00 16:00 18:00
Busy System
Busy User
Busy Iowait
Busy IRQs
Busy Other
Idle
Memory Basic
16 GiB
14 GiB
12 GiB
10 GiB
8 GiB
6 GiB
4 GiB
2 GiB
0 B
Total
Used
Cache + Buffer
Free
Swap used
Network Traffic Basic
Disk Space Used Basic

Lab 5: Configuring Alerts Usage with Prometheus and Alertmanager

Objective: Monitor CPU and Memory usage for both instances and containers using Prometheus and Alertmanager in Docker containers.

Prerequisites

- Docker installed
- Docker network monitor-net created:
  docker network create monitor-net

Step 1: Create Directories & Config Files

```
/home/azureuser/
├── prometheus/
│   ├── prometheus.yml
│   └── alert.rules.yml
└── alertmanager/
    └── config.yml
```

1.1 Prometheus Config: ~/prometheus/prometheus.yml

```
global:
  scrape_interval: 15s
alerting:
  alertmanagers:
    - static_configs:
        - targets:
#            - "host.docker.internal:9093"
           - "alertmanager:9093"

rule_files:
  - "alert.rules.yml"

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

  - job_name: 'node_exporters'
    static_configs:
      - targets: ['localhost:9100', '20.51.137.78:9100', '172.190.242.247:9100']

  - job_name: 'cadvisor'
    static_configs:
      - targets: ['172.190.242.247:8080']
```

1.2 Prometheus Alert Rules: ~/prometheus/alert.rules.yml

```
groups:
- name: linux-node-alerts
```

```
  rules:
    - alert: HighCPUUsage
      expr: 100 - (avg by (instance)
(rate(node_cpu_seconds_total{mode="idle"}[1m])) * 100) > 80
      for: 1m
      labels:
        severity: warning
      annotations:
        summary: "High CPU usage on {{ $labels.instance }}"
        description: "CPU usage has been above 80% for 1 minute."

    - alert: ContainerCPUUsage
      expr: rate(container_cpu_usage_seconds_total{container!=""}[1m]) > 0.8
      for: 1m
      labels:
        severity: warning
      annotations:
        summary: "High CPU usage in container"
        description: "{{ $labels.container }} CPU usage is above 80%."

    - alert: InstanceMemoryUsage
      expr: (node_memory_MemTotal_bytes - node_memory_MemAvailable_bytes) /
node_memory_MemTotal_bytes > 0.1
      for: 1m
      labels:
        severity: warning
      annotations:
        summary: "High memory usage on {{ $labels.instance }}"
        description: "Instance memory usage is above 90%."

    - alert: ContainerMemoryUsage
      expr: container_memory_usage_bytes{container!=""} /
container_spec_memory_limit_bytes{container!=""} > 0.1
      for: 1m
      labels:
        severity: warning
      annotations:
        summary: "High memory usage in container"
        description: "{{ $labels.container }} memory usage is above 90%."
```

Verify the rules: `docker exec -it prometheus promtool check rules /etc/prometheus/alert.rules.yml`

Step 2: Alertmanager Config File

Create ~/alertmanager/config.yml:

```
global:
```

```
  resolve_timeout: 5m

route:
  receiver: default

receivers:
  - name: default
```

Step 3: Run Alertmanager Container (Auto-Restart)

```
docker run -d \
  --name alertmanager \
  --restart unless-stopped \
  --network monitor-net \
  -p 9093:9093 \
  -v ~/alertmanager:/etc/alertmanager \
  prom/alertmanager \
  --config.file=/etc/alertmanager/config.yml
```

Reload Rules

If rules were updated, restart the container:

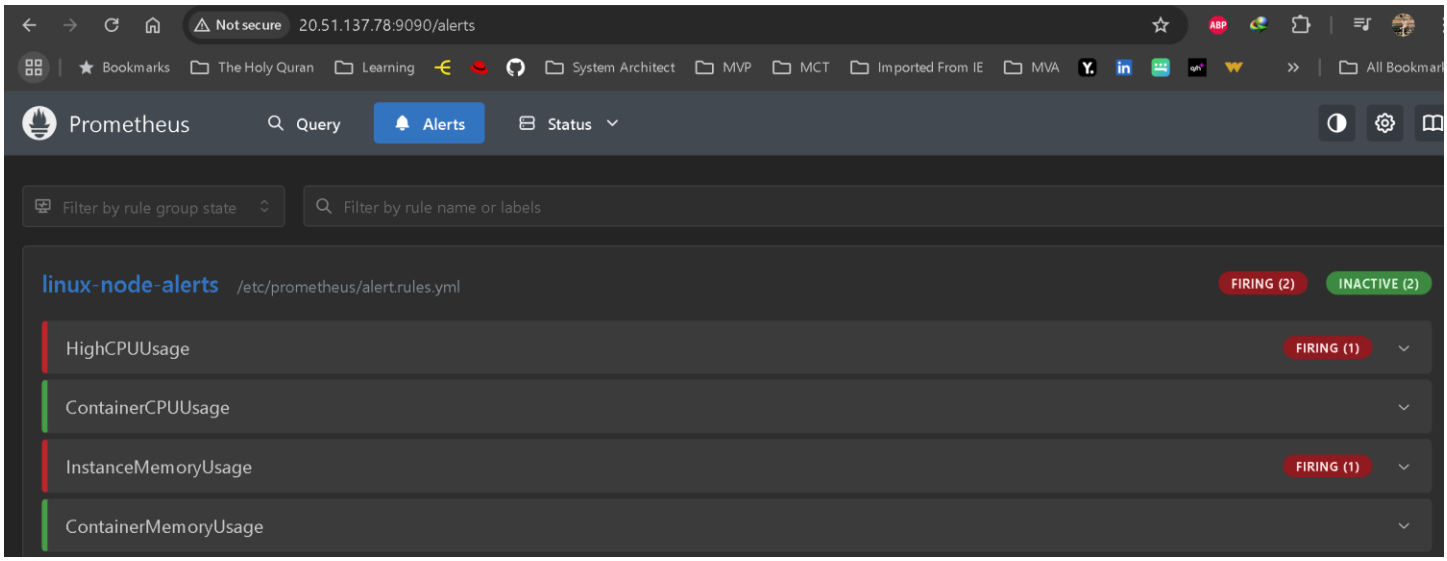
```
docker restart prometheus
```

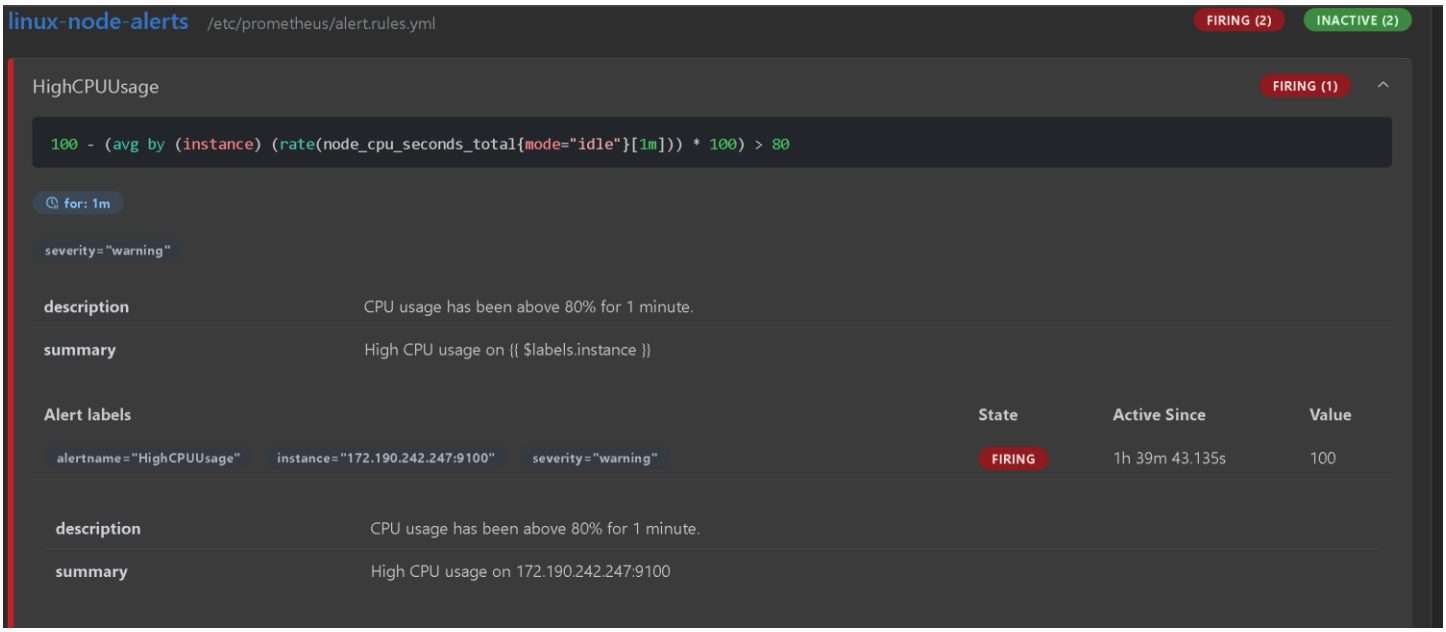
Or use signal:

```
docker exec -it prometheus kill -HUP 1
```

Test

```
Yes> /dev/null
```




linux-node-alerts
/etc/prometheus/alert.rules.yml
FIRING (2)
INACTIVE (2)
HighCPUUsage
FIRING (1)
100 - (avg by (instance) (rate(node_cpu_seconds_total{mode="idle"}[1m])) * 100) > 80
for: 1m
severity="warning"
description
CPU usage has been above 80% for 1 minute.
summary
High CPU usage on {{ $labels.instance }}
Alert labels
State
Active Since
Value
alertname="HighCPUUsage"
instance="172.190.242.247:9100"
severity="warning"
FIRING
1h 39m 43.135s
100
description
CPU usage has been above 80% for 1 minute.
summary
High CPU usage on 172.190.242.247:9100