System Monitoring with Prometheus and Node Exporter (Master + Worker Nodes)

This guide documents the full setup to monitor a multi-node Kubernetes (K3s) cluster using **Prometheus** and **Node Exporter**. It covers both **master node** and **worker node** configurations.

Table of Contents

Overview

We use **Prometheus** to scrape system metrics from:
- A **K3s master node** (also running Prometheus server)
- Multiple **K3s worker nodes** (e.g., Jetson AGX Orin)

Each node runs **Node Exporter**, a lightweight monitoring agent that exposes CPU, memory, disk, GPU (if added), and network stats via an HTTP endpoint.

Metrics We Collect

From each node via Node Exporter:
- CPU usage (per core)
- Memory usage (total, used)
- Disk I/O
- Network stats
- Boot time, thread count, etc.

For Jetson (NVIDIA) devices with additional GPU metrics:
- `jetson_gpu_usage_percent`
- `jetson_cpu_combined_percent`
- `jetson_memory_used_mb`

Worker Node Setup

> Repeat this on **every K3s worker node** you want to monitor.

1. Download Node Exporter

```bash
wget https://github.com/prometheus/node_exporter/releases/download/v1.9.1/node_exporter-1.9.1.linux-arm64.tar.gz
tar -xzf node_exporter-1.9.1.linux-arm64.tar.gz
cd node_exporter-1.9.1.linux-arm64
```

2. Add Optional GPU Metrics (Jetson/ARM)

Create a custom file that holds GPU/CPU/memory stats:

```bash
mkdir -p /var/lib/node_exporter
nano /var/lib/node_exporter/gpu_metrics.prom
```

Add metrics like:

```prometheus
jetson_gpu_usage_percent 35
jetson_cpu_combined_percent 12
jetson_memory_used_mb 4782
```

To update this file automatically, create a cron job or background script.

3. Start Node Exporter

```bash
./node_exporter   --collector.textfile.directory=/var/lib/node_exporter   --web.listen-address=":9100"
```

Ensure port `9100` is **open and reachable** from the master node.

4. (Optional) Run Node Exporter as a Systemd Service

Create a `node_exporter.service` unit file to run it persistently.

Master Node Setup (Prometheus Server)

1. Download Prometheus

```bash
wget https://github.com/prometheus/prometheus/releases/download/v2.52.0/prometheus-2.52.0.linux-amd64.tar.gz
tar -xzf prometheus-2.52.0.linux-amd64.tar.gz
cd prometheus-2.52.0.linux-amd64
```

2. Define Your Prometheus Configuration

Edit `prometheus.yml`. See below for format.

Prometheus Configuration (`prometheus.yml`)

Edit the `prometheus.yml` in your Prometheus directory to include the master and worker targets.

```yaml
global:
  scrape_interval: 5s

scrape_configs:
  - job_name: 'master-node'
    static_configs:
      - targets: ['localhost:9100']
  - job_name: 'worker-node-agx-desktop'
    static_configs:
      - targets: ['<WORKER_IP>:9100']
```

Replace `<WORKER_IP>` with your actual worker node IPs.

Verification Steps

1. Test Connectivity

On the master node:

```bash
ping <WORKER_IP>
curl http://<WORKER_IP>:9100/metrics
```

You should see raw metrics text output from the Node Exporter.

2. Start Prometheus

```bash
./prometheus --config.file=prometheus.yml
```

3. Open Prometheus Web UI

Visit:
```
http://<MASTER_IP>:9090
```

Navigate to **Status Targets** to confirm:
- All nodes are `UP`
- Scraping every 5 seconds

Notes for Jetson/ARM Devices

- Use `linux-arm64` release for Node Exporter
- Use `--collector.textfile.directory` to feed Jetson GPU stats
- Jetson metrics file must be formatted in **Prometheus `.prom` syntax**

Optional: Run Prometheus as a Systemd Service

Convert Prometheus into a background service using a `prometheus.service` unit file for auto-restarts.