PROJECT REPORT: INTEGRATING GRAFANA WITH A LINUX SERVER FOR CPU UTILIZATION MONITORING

**AIM**

* **The primary aim of this project is to monitor and visualize CPU utilization data from a Linux-based server environment using open-source tools like Prometheus, node exporter and grafana.**
* **The setup combines Prometheus for collection of a data automatically from a source and its storage.**
* **Node Exporter for system metrics collection which include CPU usage, memory usage, disk space, network traffic and system up-time.**
* **Grafana for intuitive, real-time dashboard visualization.**
* **System admins can see how resources (like CPU, memory) are being used.**
* **They can spot problems or slowdowns in the system (called bottlenecks)**
* **They can keep a clear, real-time view of how the system is working overall**

**TECHNICAL SKILLS USED**

* **Ubuntu Linux** – Server management and setup
* **WSL with systemd** – Linux on Windows with services
* **Command Line (CLI)** – Installing and configuring tools
* **Prometheus** – Collects system performance data
* **Node Exporter** – Sends system metrics to Prometheus
* **Grafana** – Visualizes data in dashboards
* **PromQL** – Queries data from Prometheus
* **systemd** – Manages background services
* **YAML** – Configures Prometheus jobs
* **Troubleshooting** – Fixes service and permission errors

**STEPS FOLLOWED**

1. Setup Node Exporter on the Target Linux Server

1. **Create a dedicated user (recommended for security):**

Bash sudo useradd -rs /bin/false node\_exporter

1. **Download Node Exporter: Go to the** [**Prometheus Node Exporter GitHub releases page**](https://github.com/prometheus/node_exporter/releases) **( ctrl+click, to follow the link )and find the latest stable version for your Linux architecture .**

**Bash**

**wget https://github.com/prometheus/node\_exporter/releases/download/v1.9.1/node\_exporter-1.9.1.darwin-amd64.tar**

1. **Extract and move the binary:**

**Bash**

**tar xvfz node\_exporter-1.9.1.linux-386.tar**

**sudo mv node\_exporter-1.9.1.linux-386.tar/node\_exporter /usr/local/bin/**

1. **Create a Systemd service file for Node Exporter:**

**Bash**

**sudo nano /etc/systemd/system/node\_exporter.service**

**Paste the following content:**

**[Unit]**

**Description=Node Exporter**

**After=network.target**

**[Service]**

**User=node\_exporter**

**Group=node\_exporter**

**Type=simple**

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

**[Install]**

**WantedBy=multi-user. target**

**Save and exit (Ctrl+O, Enter, Ctrl+X).**

1. **Reload Systemd, start, and enable Node Exporter:**

**Bash**

**sudo systemctl daemon-reload**

**sudo systemctl enable node\_exporter**

**sudo systemctl start node\_exporter**

1. **Verify Node Exporter is running:**

**Bash**

**sudo systemctl status node\_exporter**

**You should see "Active: running".**

1. **Check metrics (optional): Node Exporter typically listens on port 9100. You can access the metrics in your web browser or via curl:**

**Bash**

**curl http://localhost:9100/metrics**

**You should see a long list of metrics.**

1. **Open firewall port (if applicable): If you have a firewall (e.g., ufw), allow access to port 9100:**

**Bash**

**sudo ufw allow 9100/tcp**

**sudo ufw reload**

**Part 2: Set up Prometheus Server**

1. **Create a dedicated user:**

**Bash**

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

1. **Create directories for Prometheus:**

**Bash**

**sudo mkdir /etc/prometheus**

**sudo mkdir /var/lib/prometheus**

**sudo chown prometheus:prometheus /var/lib/prometheus**

1. **Download Prometheus Server: Go to the** [**Prometheus GitHub releases page**](https://github.com/prometheus/prometheus/releases) **and download the latest stable version for your Linux architecture.**

**Bash**

**wget https://github.com/prometheus/prometheus/releases/download/vX.Y.Z/prometheus-X.Y.Z.linux-amd64.tar.gz**

1. **Extract and move binaries/configuration:**

**Bash**

**tar xvfz prometheus-X.Y.Z.linux-amd64.tar.gz**

**sudo mv prometheus-X.Y.Z.linux-amd64/prometheus /usr/local/bin/**

**sudo mv prometheus-X.Y.Z.linux-amd64/promtool /usr/local/bin/**

**sudo cp -r prometheus-X.Y.Z.linux-amd64/consoles /etc/prometheus**

**sudo cp -r prometheus-X.Y.Z.linux-amd64/console\_libraries /etc/prometheus**

1. **Create Prometheus configuration file:**

**Bash**

**sudo nano /etc/prometheus/prometheus.yml**

**Paste the following. Crucially, replace your\_linux\_server\_ip with the actual IP address of the server where Node Exporter is running. If you're monitoring the same server, use localhost.**

**Save (ctrl + O), enter and exit (ctrl +X).**

**global:**

**scrape\_interval: 15s # How frequently to scrape targets**

**scrape\_configs:**

**- job\_name: 'prometheus'**

**static\_configs:**

**- targets: ['localhost:9090'] # Prometheus itself**

**- job\_name: 'linux\_servers'**

**static\_configs:**

**- targets: ['your\_linux\_server\_ip:9100']**

1. **Create a Systemd service file for Prometheus:**

**Bash**

**sudo nano /etc/systemd/system/prometheus.service**

**Paste the following:**

**CTRL + O (save)**

**Ini, TOML**

**[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**

**[Install]**

**WantedBy=multi-user.target**

**ENTER**

**CTRL+X (exit) 7. Reload Systemd, start, and enable Prometheus:**

**Bash**

**sudo systemctl daemon-reload**

**sudo systemctl start prometheus**

**sudo systemctl enable prometheus**

1. **Verify Prometheus is running:**

**Bash**

**sudo systemctl status prometheus**

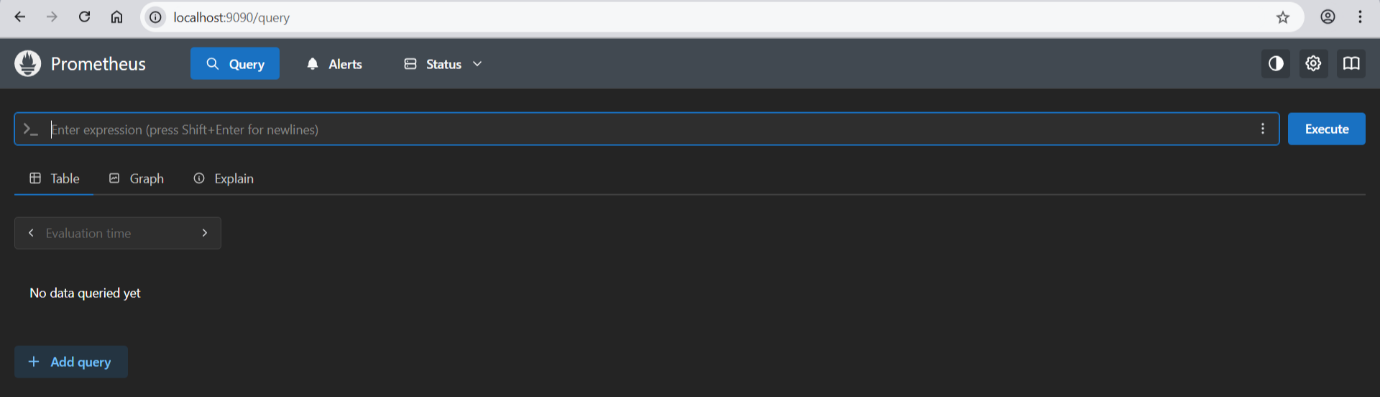
**You should see "Active: running".**

1. **Access Prometheus UI (optional): Open your web browser and go to http://your\_prometheus\_server\_ip:9090. Go to "Status" -> "Targets" to confirm that your linux\_servers target is "UP".**
2. **Open firewall port (if applicable): If you have a firewall, allow access to port 9090:**

**Bash**

**sudo ufw allow 9090/tcp**

**sudo ufw reload**

**Part 3: Install and Configure Grafana**

**This should also be on a separate server (or the same as Prometheus, or even on your local machine if you just need to access the dashboards).**

1. **Install Grafana (Debian/Ubuntu example):**

**Bash**

**sudo apt-get install -y grafana**

**wget** [**https://dl.grafana.com/enterprise/release/grafana-enterprise\_10.Y.Z\_amd64.deb**](https://dl.grafana.com/enterprise/release/grafana-enterprise_10.Y.Z_amd64.deb)

**sudo dpkg -i grafana-enterprise\_10.Y.Z\_amd64.deb**

1. **Start and enable Grafana:**

**Bash**

**sudo systemctl start grafana-server**

**sudo systemctl enable grafana-server**

1. **Verify Grafana is running:**

**Bash**

**sudo systemctl status grafana-server**

**You should see "Active: running".**

1. **Access Grafana**

Open your browser and navigate to http://localhost:3000. The default login credentials are:

Username: admin

Password: admin

You'll be prompted to change the password upon first login.

1. **Add Prometheus as a Data Source in Grafana:**
   * **Select Data sources( by searching on search bar).**
   * **Click Add data source.**
   * **Choose Prometheus.**
   * **In the HTTP section, set the URL to http://localhost:9090.**
   * **Scroll down and click Save & Test. You should see "Data source is working" or similar.**

**Part4: Create a CPU Utilization Dashboard in Grafana**

* 1. **Import a Pre-built Dashboard:**

**Grafana has a community dashboard for CPU metrics:**

* + **Navigate to Grafana Labs Dashboard 9617.**
  + **Note the dashboard ID: 9617.**
  1. **Import the Dashboard into Grafana:**
  + **In Grafana, go to Dashboards > Import.**
  + **Enter the dashboard ID 9617 and click Load.**
  + **Select your Prometheus data source.**
  + **Click Import.**

**This dashboard provides visualizations for CPU usage, including per-core metrics and overall utilization.**

1. **Navigate to “Add New Panel”**

**Once you’re in your dashboard:**

* **Click + in the left menu → choose Dashboard → click Add new panel.**
* **You’ll see a split view: Query editor on the bottom and preview graph on top. Look similar to the first image above.** [**github.com+12ruianding.com+12grafana.com+12**](https://www.ruianding.com/blog/monitoring-stack-part-3-visualizing-server-metrics-with-grafana-dashboards/?utm_source=chatgpt.com)

1. **Enter the CPU Query**

**In the query editor, select Prometheus as the data source, and in the expression field paste:**

**less**

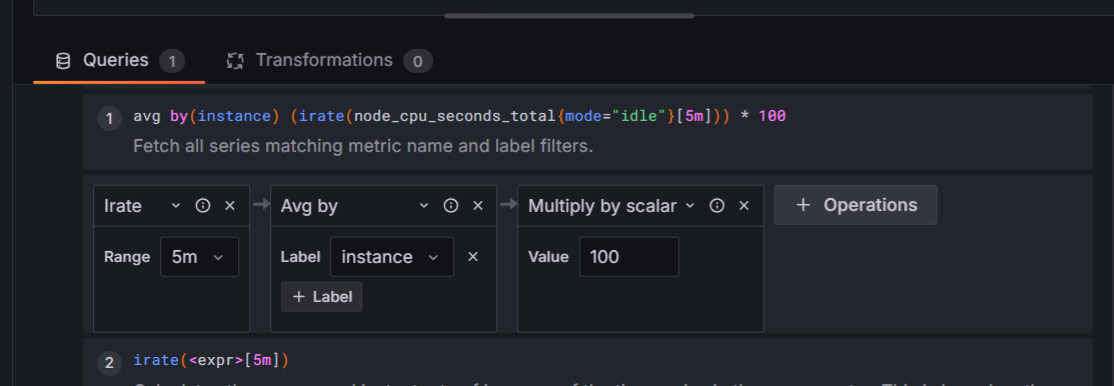
**Enter this in code: 100 - (avg by(instance)(irate(node\_cpu\_seconds\_total{mode="idle"}[5m])) \* 100)**

**Enter these in builder:**

**Range 🡪 irate(...[5m]) computes the recent rate of change (per second).**

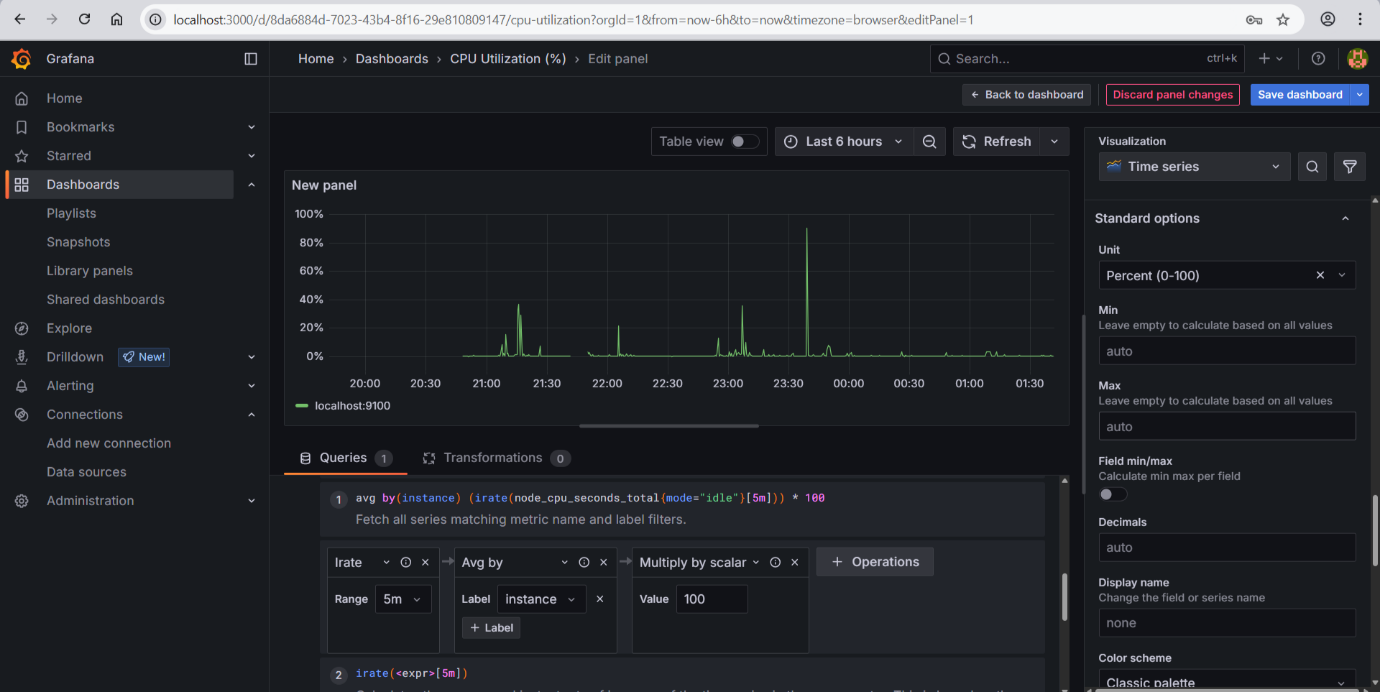
* **Avg by(instance)(...) averages across all cores per server.**
* **Multiply By 100 - (...) \*100 converts idle percentage to busy CPU utilization.**
* **node\_cpu\_seconds\_total{mode="idle"} gives idle CPU time.**

Metric name : node\_cpu\_seconds\_total

****

1. **See the Graph**

**Once you enter the query, the top preview updates in real-time. You’ll typically see a rising and falling line showing CPU usage over time—similar in style to the gauges and line charts in the second and fourth images.**

****

1. **Choose Visualization Type**

**Above the preview, switch from the default Time series to options like:**

* **Gauge – displays current utilization on a meter.**
* **Bar gauge – offers colored thresholds.**
* **Stat – shows a single current value.**

**In the carousel image, you can see gauges and line series used together.** [**mefmobile.org+11metricfire.com+11umatechnology.org+11**](https://www.metricfire.com/blog/grafana-dashboards-from-basic-to-advanced/?utm_source=chatgpt.com)[**github.com**](https://github.com/Sgate-droid/Monitoring-Linux-server-with-Grafana-and-prometheus?utm_source=chatgpt.com)

1. **Fine‑tune Panel Settings**

* **Click the panel’s “Panel options” tab:**
* **Add a title, e.g., “CPU Usage (%)”**
* **Set Y-axis range: 0 to 100**
* **Define thresholds (e.g. 70%, 90%) for color cues**
* **Adjust legend, display units, or add min/max**

**Resulting setups look like the gauges and graphs shown above.**

1. **Apply & Save**

**Click Apply (upper‑right) to add the panel to your dashboard.**

**Don’t forget to Save Dashboard (the floppy disk icon) to preserve it.**

**CHALLENGES DURING FOLLOWING STEPS**

1. **Access Denied to Files**
2. **Download Failed (File Not Found)**
3. **Node Exporter Not Working**
4. **System Services Not Starting in WSL**
5. **Grafana Query Errors**
6. **Graphs Show No Data**

**TROUBLESHOOTING STEPS**

**1. File Access Permissions in Home Directory**

**Precise Heading:** Resolving Permission Denied Errors in /home/username

**Problem Encountered:** When attempting to access, modify, or create files/directories within /home/username, operations failed with "Permission denied" errors. This often indicates incorrect user ownership or file permissions.

**Steps to Resolve:**

1. **Identify Current Ownership:** Use ls -ld /home/username to check the current owner and group of the directory. Similarly, ls -l /home/username to check individual files/subdirectories.
2. **Verify User ID (UID):** Ensure the user whichever user you're logged in as has the correct UID, and that this UID is consistent with file ownership. (Less common in direct WSL, more common in Docker or complex user setups).
3. **Change Ownership (if incorrect):** If the directory is owned by root or another user, change it to your user:

Bash

sudo chown -R sanchi:sanchi /home/sanchi

* + -R: Recursive, applies to all contents.
  + sanchi:sanchi: Sets both the user and group owner to sanchi.

Note: change sanchi:sanchi to your username.

1. **Adjust Permissions (if too restrictive):** Ensure your user has read, write, and execute permissions. For a home directory, 755 is typical for directories.

Bash chmod 755 /home/sanchi

* + 755: Owner has read/write/execute, group/others have read/execute.

1. **Verify Resolution:** Re-attempt the file operation that previously failed.

**2. wget Download Errors (404 Not Found)**

**Precise Heading:** Troubleshooting wget 404 Errors for Prometheus/Grafana Downloads

**Problem Encountered:** wget commands failed with "404 Not Found" errors, specifically when trying to download Prometheus or Grafana archives. This indicated that the requested URL did not point to an existing resource.

**Steps to Resolve:**

1. **Verify the Release URL:**
   * Go to the official GitHub releases page for the software (e.g., https://github.com/prometheus/prometheus/releases for Prometheus, https://grafana.com/grafana/download/ for Grafana).
   * Locate the specific version you intend to download (e.g., v3.4.1).
   * **Crucially, identify the exact filename** for your architecture (.linux-amd64.tar.gz for Prometheus, .deb for Grafana).
2. **Copy the Direct Download Link:** Right-click on the download link for the correct file and select "Copy link address" or "Copy URL."
3. **Construct wget Command with Exact URL:** Paste the copied URL directly into your wget command:

Bash

wget <Exact\_Copied\_URL\_Here>

* + **Avoid using placeholder version numbers (X.Y.Z) directly in URLs** unless you are absolutely certain of the versioning scheme and that the exact version exists at that URL. Always confirm against the official releases page.

1. **Check Internet Connectivity:** (Basic check) Ensure your WSL instance has active internet access using ping google.com.

**3. Node Exporter Service and Metrics Debugging**

**Precise Heading:** Diagnosing and Verifying Node Exporter Status and Metrics

**Problem Encountered:** Difficulty confirming if Node Exporter was running correctly and if it was exposing metrics as expected, leading to issues with Prometheus scraping.

**Steps to Resolve:**

1. **Check Service Status:**
   * **If using Systemd (after enabling it in WSL):**

Bash

sudo systemctl status node\_exporter

Look for Active: active (running).

* + **If running manually or as a background process:** Check ps aux | grep node\_exporter.

1. **Direct Metrics Endpoint Check:** Use curl to directly query Node Exporter's metrics endpoint. Node Exporter typically listens on port 9100.

Bash

curl http://localhost:9100/metrics

* + **Expected Output:** A long list of text-based metrics data.
  + **If curl fails:** Check if Node Exporter is listening on the correct port using sudo netstat -tulnp | grep 9100 (or sudo ss -tulnp | grep 9100).

1. **Review Node Exporter Logs:** Examine logs for any startup errors or issues.
   * **If using Systemd:** sudo journalctl -u node\_exporter -f (for real-time logs)
   * **If logging to a file:** Check the configured log file path.
2. **Verify Firewall (if applicable):** Ensure no firewall rules (e.g., ufw in Ubuntu, or Windows Firewall for WSL) are blocking traffic on port 9100. (Less common for localhost within WSL but could affect Windows access).
3. **Check Prometheus Configuration:** Confirm prometheus.yml has the correct job\_name and targets for Node Exporter (localhost:9100).

**4. Systemd Service Issues in WSL**

**Precise Heading:** Enabling Systemd for Service Management in WSL2

**Problem Encountered:** WSL2 does not natively run a full systemd init system by default, leading to service management commands (systemctl start/enable/status) failing or not working as expected.

**Steps to Resolve:**

1. **Identify WSL Version:** wsl -l -v from PowerShell/CMD. Systemd support is a relatively new feature in WSL2.
2. **Update WSL:** Ensure your WSL version is up-to-date, as recent versions (from late 2022 onwards) have better experimental systemd support.

PowerShell

wsl --update

wsl --shutdown

1. **Enable Systemd in WSL Configuration:**
   * Create or edit the /.wslconfig file in your Windows user profile directory (e.g., C:\Users\YourUsername\.wslconfig).
   * Add the following lines:

Ini, TOML

[boot]

systemd=true

* + Save the file.

1. **Restart WSL:**

PowerShell

wsl --shutdown

Then restart your WSL distribution (e.g., by opening a new Ubuntu terminal).

1. **Verify Systemd:** After restarting, try systemctl --version inside WSL. If it outputs version information, Systemd is enabled.
2. **Proceed with Service Commands:** Now, sudo systemctl start/enable/status/stop commands should work for services like Prometheus and Node Exporter.

**5. No Data in Grafana Graphs (Panel Configuration)**

**Precise Heading:** Troubleshooting Empty Grafana Panels and Data Visualization

**Problem Encountered:** Despite Prometheus and Node Exporter running and being accessible, Grafana panels showed no data, indicating a disconnect in data retrieval or visualization.

**Steps to Resolve:**

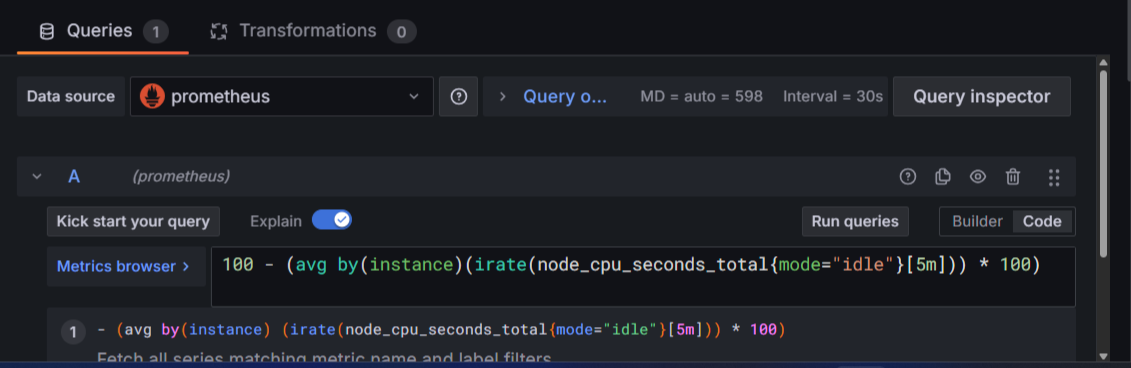
1. **Verify Data Source Connection:** (First line of defence) Go to Configuration -> Data Sources in Grafana, click on your Prometheus data source, and click Save & test. Ensure it says "Data source is working."
2. **Check Query in Panel:**
   * In the problematic Grafana panel, click Edit (usually the panel title -> Edit).
   * Go to the Query tab.
   * Ensure the Prometheus data source is selected.
   * Verify the PromQL query being used is correct and actually returns data in Prometheus's own UI (http://localhost:9090/graph).
   * Check the "Range" selected in the dashboard (e.g., "Last 5 minutes"). If data is recent, ensure the time range includes it.
3. **Inspect Metric Names:** Double-check that the metric names used in your PromQL queries exactly match the metrics exposed by Node Exporter (e.g., node\_cpu\_seconds\_total, not cpu\_total). Use curl http://localhost:9100/metrics to confirm exact names.
4. **Time Range and Refresh Rate:**
   * Ensure the time range selected in Grafana (top right corner) covers the period when data was being collected.
   * Check the refresh rate (next to the time range). Ensure it's not set to "Off" or a very long interval.
5. **Panel Type and Field Mapping:**
   * Ensure the Visualization type (e.g., "Graph," "Stat") is appropriate for the data.
   * In the Transform or Standard options tabs of the panel configuration, ensure the correct "Unit" (e.g., bytes, percent, s) is selected if specified, and that "Field" options are mapped correctly for the visualization type.
6. **Prometheus Targets "UP":** Re-verify in Prometheus UI (http://localhost:9090/targets) that the node\_exporter target is indeed UP. If it's DOWN, Prometheus isn't collecting data, so Grafana will have nothing to display.

**6. PromQL Syntax Errors in Grafana (Builder Mode)**

**Precise Heading:** Correcting PromQL Syntax and Utilizing Grafana's Query Modes

**Problem Encountered:** Grafana's query builder mode might generate incorrect PromQL expressions or limit complex queries, leading to "syntax errors" or unexpected results when trying to visualize Prometheus metrics.

**Steps to Resolve:**

1. **Switch to "Code" Mode (or "Text" Mode):** In Grafana's query editor panel, there's usually a toggle or button (often labelled "Code" or "Text") to switch from the visual builder to directly writing PromQL. This gives you full control over the query.
2. **Test Queries in Prometheus UI:** Before using complex queries in Grafana, test them directly in the Prometheus UI (http://localhost:9090/graph). The Prometheus UI provides immediate feedback on syntax and results.
3. **Understand Labels and Functions:** Ensure you're using correct metric names, labels, and PromQL functions (e.g., rate(), irate(), sum(), by()) for your specific use case. Pay attention to how labels are used (e.g., {instance="localhost:9100"}).

FINAL STATEMENT

This project effectively demonstrated the integration of Prometheus, Node Exporter, and Grafana for real-time CPU usage monitoring on a Linux system. From setting up each component individually to configuring services and resolving compatibility issues within WSL, the experience provided deep insights into the operational workflow of a modern monitoring stack. The use of PromQL to generate CPU metrics and the deployment of Grafana dashboards enabled a powerful visual representation of system activity. By overcoming challenges in service configuration, query debugging, and environment setup, this project lays a solid foundation for further enhancements such as monitoring memory, disk, and network performance. The skills acquired here are widely applicable in DevOps, system administration, and infrastructure monitoring roles. ###