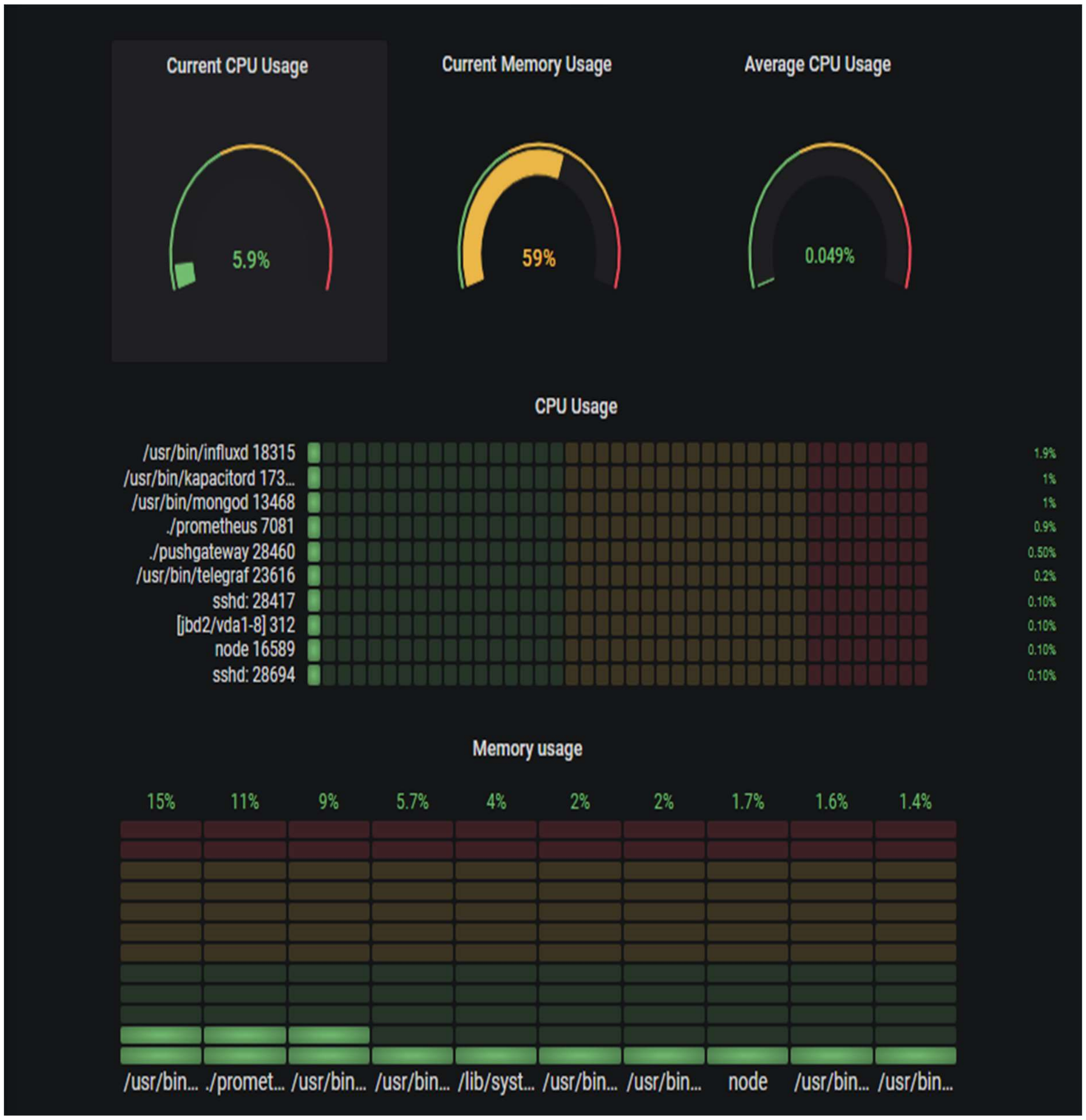
PROJECT-3

Integrate Grafana with Linux Server for highcpu utilization and create a graph in Grafana.



INDEX

I. INTRODUCTION

This project outlines how to integrate Grafana with a Linux server to monitor high CPU utilization and visualize the data with a Grafana dashboard. The integration involves setting up Prometheus to collect CPU metrics from the Linux server and then configuring Grafana to query and display these metrics in a user-friendly graph.

II. OBJECTIVE

Implement a robust monitoring solution for Linux server CPU utilization using Grafana, including data collection, visualization, and alert notifications.
The objective of this project, "Linux Server CPU Utilization Monitoring with Grafana," is to design, implement, and demonstrate a robust and userfriendly monitoring solution for Linux server CPU utilization.

III. WHY I BUILT THIS PROJECT?

These reasons often stem from real-world challenges faced in managing IT infrastructure. Here are the primary motivations:

- Proactive Problem Detection and Prevention
- Performance Optimization and Resource Planning
- Troubleshooting and Root Cause Analysis
- Operational Efficiency and Automation
- Cost-Effectiveness (especially with open-source tools)

IV.TECHNOLOGY USED

- Linux Operating System
- Node Exporter
- Node Exporter
- PromQL (Prometheus Query Language) Windows Operating System

V.SETUP CHECKLIST

a. <u>HARDWARE:</u>

- Window 10 or more
- 8GB RAM

b. <u>SOFTWARE:</u>

- Linux Server
- Node Exporter
- Prometheus
- Grafana Setup

VI.STEP-BY-STEP IMPLEMENTATION

Step-1: Linux Server - Node Exporter Installation
This part needs to be done on the Linux server you want to monitor.

a. SSH into your Linux Server:
Open your SSH client on Windows (e.g., PowerShell, Command Prompt, or PuTTY) and connect to your Linux server.

```
Bash

ssh your_username@your_linux_server_ip
(Replace your_username and your_linux_server_ip with your actual credentials and IP).
```

b. Download Node Exporter:
Go to the Prometheus Node Exporter GitHub releases page in your web browser to find the latest stable version. Look for the node_exporter-X.Y.Z.linux-amd64.tar.gz file.
On your Linux server, use wget to download it.

```
Bash

# Replace X.Y.Z with the latest version number you found (e.g., 1.8.1)
wgethttps://github.com/prometheus/node_exporter/releases/download/vX.
Y.Z/node_exporter-X.Y.Z.linux-amd64.tar.gz Example:
wgethttps://github.com/prometheus/node_exporter/releases/download/v1.
8.1/node_exporter-1.8.1.linux-amd64.tar.gz
```

Extract the Archive:

```
Bash

tar -xzf node_exporter-X.Y.Z.linux-amd64.tar.gz
This will create a directory like node_exporter-X.Y.Z.linux-amd64.
```

c. **Move the Binary to a System Path:**

```
Bash

sudo mv node_exporter-X.Y.Z.linux-amd64/node_exporter
/usr/local/bin/
```

d. Create a Dedicated System User for Node Exporter:

This is a security best practice.

```
Bash

sudouseradd -rs /bin/false node_exporter
```

e. **Create a Systemd Service File:**

This allows Node Exporter to run as a service, start automatically on boot, and be managed easily.

f. **Open a new file for editing using nano or vim:**

```
Bash

sudo nano /etc/systemd/system/node_exporter.service
```

Paste the following content into the file:

Ini, TOML

```
[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 the file: If using nano, press Ctrl+X, then Y to confirm save, then Enter.

g. **Reload Systemd, Enable, and Start Node Exporter Service:**

```
Bash

sudosystemctl daemon-reload sudosystemctl enable
node_exportersudosystemctl start node_exporter
```

h. **Verify Node Exporter Status:**

```
Bash

sudosystemctl status node_exporter
```

You should see "Active: active (running)". Press Q to exit the status view.

i. Verify Node Exporter Metrics Locally (Optional, on Linux server):

```
Bash

curl http://localhost:9100/metrics
```

This command should output a long list of metrics.

j. Open Firewall Port (if applicable on Linux server):

If your Linux server has a firewall enabled (like ufw or firewalld), you need to allow incoming connections on port 9100 from your Prometheus server (your Windows machine).

For UFW (Ubuntu/Debian):

```
Bash

sudoufw allow 9100/tcpsudoufw reload
```

For Firewalld (CentOS/RHEL):

```
Bash

sudo firewall-cmd --add-port=9100/tcp --permanent sudo firewall-cmd
–reload
```

k.Verify Node Exporter Access from Windows Browser:

On your Windows machine, open your web browser and go to: http://<your_linux_server_ip>:9100/metrics

You should see the same metrics page as you did with curl. If not, double-check the Linux server's IP, firewall, and that Node Exporter is running.

Step-2: Windows Machine - Prometheus Installation

This part needs to be done on your Windows machine.

a. **Download Prometheus:**

Go to the Prometheus Downloads page in your web browser.

Download the latest stable release for Windows (amd64) (it will be a .zip file).

b. **Extract Prometheus:**

Locate the downloaded .zip file (e.g., prometheus-X.Y.Z.windowsamd64.zip).
Right-click the file and select "Extract All...".
Choose a destination folder, for example, C:\ to extract it into C:\prometheus-X.Y.Z.windows-amd64.
For simplicity, rename the extracted folder to something like C:\Prometheus.

c. **Configure prometheus.yml:**

This is Prometheus's main configuration file, where you tell it what to scrape.

Navigate to your Prometheus installation directory (e.g., C:\Prometheus).
Open the prometheus.yml file using a text editor (e.g., Notepad++, VS Code).

Locate the scrape_configs: section. You'll likely see a default **job_name:** 'prometheus' for self-monitoring.

Add a new job_name entry below the existing prometheus one to scrape your Linux server's Node Exporter.
YAML

# my global config global: scrape_interval: 15s # Set the scrape interval to every 15 seconds.

Default is every 1 minute.   evaluation_interval: 15s # Evaluate rules every 15 seconds. Default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# A scrape configuration scraping as themselves.
scrape_configs:

```
-        job_name: "prometheus"     static_configs:
-        targets: ["localhost:9090"]

-        job_name: 'linux_server_cpu' # <--- ADD THIS BLOCK
static_configs:
-        targets: ['your_linux_server_ip:9100'] # <--- IMPORTANT:
REPLACE WITH YOUR LINUX SERVER'S ACTUAL IP
```

Save the prometheus.yml file.

d. **Run Prometheus:**

Open a Command Prompt or PowerShell window as Administrator.

e. **Navigate to your Prometheus directory:**

```
DOS

cd C:\Prometheus
```

Start Prometheus:

```
DOS

.\prometheus.exe --config.file=prometheus.yml
```

You will see a lot of log output in the command prompt. Look for messages indicating it's starting and scraping targets.

f. **Verify Prometheus UI and Targets:**

Open your web browser on Windows.

Go to http://localhost:9090
In the Prometheus UI, navigate to Status > Targets.
You should see two targets: prometheus and linux_server_cpu.
Ensure both have a State of UP. If linux_server_cpu is DOWN, recheck the IP address in prometheus.yml and the firewall on your Linux server.

Step-3: Windows Machine - Grafana Installation

This part needs to be done on your Windows machine.

a. Download Grafana:

Go to the Grafana Downloads page for Windows in your web browser.
Download the recommended installer (.msi file).

b. Install Grafana:

Locate the downloaded .msi file.
Double-click the installer.
Follow the on-screen prompts. Accept the license agreement, choose the default installation location, and click "Install."
Once the installation is complete, you can click "Finish." Grafana should typically start as a Windows service automatically.

c. Verify Grafana UI:

Open your web browser on Windows.
Go to http://localhost:3000
You will be presented with a login screen.
Default Username: admin
Default Password: admin
Upon first login, Grafana will prompt you to change the default password. Choose a strong password and keep it safe.

Step-4: Grafana Configuration & Dashboard Creation

Now we connect Grafana to Prometheus and build your CPU utilization graph.

a. **Add Prometheus as a Data Source in Grafana:** In Grafana (http://localhost:3000), click on Connections (the plug icon on the left sidebar).

Under "Connections," click on Data sources.

Click the "Add new data source" button.

Type "Prometheus" in the search bar or scroll down to find it, then click on the Prometheus option.

b. **Settings:**

Name: Prometheus-Local (or any descriptive name you prefer).

HTTP URL: http://localhost:9090 (This is the address where your Prometheus is running on Windows).

Click the "Save & Test" button at the bottom. You should see a green pop-up notification "Data source is working."

c. **Create a New Grafana Dashboard:**

Click the + icon on the left sidebar.

Select New Dashboard.

Click on "Add new panel".

d. **Create CPU Utilization Graph Panel:**

In the new panel editor, under the Query tab:

Ensure your Prometheus-Local data source is selected (it usually defaults to the first one).

In the "Code" editor area, paste the following PromQL query for overall CPU utilization (excluding idle time):

Code snippet

```
100 - (avg by (instance)
(rate(node_cpu_seconds_total{mode="idle",job="linux_server_cpu"
}[5m])) * 100
```

e. Explanation of the query:

node_cpu_seconds_total: Metric from Node Exporter. mode="idle": Filters for the time the CPU spent idle.
job="linux_server_cpu": Filters for metrics from your specific Linux server.
[5m]: Calculates the rate over the last 5 minutes. rate(...): Computes the per-second average rate of increase.
avg by (instance): Averages the rate across all CPU cores for each server instance.
100 - (...): Converts the idle percentage into CPU utilization percentage.
Observe the graph updating on the right side.

In the Visualization tab (or Panel options on the right side): Panel Title: Change it to "Linux Server CPU Utilization (%)" Under Standard options > Unit: Select percent (0-100).
(Optional) Under Legend: You can use {{instance}} as the Value. This will display the IP address of your Linux server in the legend.
Click "Apply" in the top right corner.

f. Save Your Dashboard:

Click the "Save dashboard" icon (looks like a floppy disk) in the top right corner of the dashboard.
Enter a descriptive name, e.g., "Linux Server Monitoring Dashboard".
Click "Save".

Step-5: Grafana Alerting Configuration

Now, let's set up an alert for high CPU utilization.

a. Configure a Contact Point:

This is where Grafana will send notifications. For testing, a simple email or webhook is good.

On the left sidebar in Grafana, click the bell icon (Alerting).
Go to Contact points.
Click "Add contact point".
Name: MyEmailNotifications (or MyWebhookNotifications) Type:
For Email: Select "Email". Enter your email address in "Addresses".
For Webhook (easy for testing): Select "Webhook". Go to https://webhook.site/ in a new browser tab, copy the unique URL it provides, and paste it into Grafana's "URL" field.
Click "Save & Test". If using email, check your inbox for a test email. If using webhook, check the webhook.site page for the incoming test request.

b. **Create an Alert Rule:**
In Grafana's Alerting section (bell icon), go to Alert rules.
Click "New alert rule".
Rule name: High CPU Utilization on Linux Server.
Folder: You can create a new folder (e.g., "Server Alerts") or use "General."
Data source: Select your Prometheus-Local data source.
Query: Use the same PromQL query as your graph panel:
Code snippet

```
100 - (avg by (instance)
(rate(node_cpu_seconds_total{mode="idle",job="linux_server_cpu"}[5
m])) * 100)
```

Condition:
WHEN: avg() of Query A
IS ABOVE: 80 (This means alert if CPU goes above 80%. You can adjust this threshold.)
FOR: 5m (This means the condition must be true for 5 continuous minutes before the alert fires, preventing flapping alerts from transient spikes.)
Evaluation Behavior:

Evaluate every: 1m (How often Grafana checks the rule).
Alert state if no data or all values are null: Select No Data or Alerting based on your preference (e.g., Alerting if you want to know if data stops flowing).
Alert state if execution error or timeout: Select Alerting.
Notifications:
Under "Send to," select your configured contact point (e.g., MyEmailNotifications).
Summary: High CPU on {{ $labels.instance }}: Current utilization is {{ $value | humanize }}%
Annotations (Optional, but useful): Add more details.
Key: description
Value: CPU utilization on server {{ $labels.instance }} has been above {{ $threshold }}% for 5 minutes.
Click "Save rule".

c. **Test the Alert (Important!):**

Go back to your Linux server (via SSH).
Install a stress testing tool (if you don't have one). stress-ng is powerful.
For Debian/Ubuntu:

```
Bash
sudo apt update
sudo apt install
stress-ng
```

For CentOS/RHEL:

```
Bash

sudo yum install epel-release sudo yum install stress-ng
```

Generate High CPU Load: Run stress-ng to utilize your CPU. Adjust --cpu to the number of cores on your server.

```
Bash

stress-ng --cpu $(nproc) –timeout 300s --metrics-brief
# nproc gets the number of CPU cores. --timeout 300s runs for 5 minutes.
```

Alternative simple CPU hog (for single core, run multiple times for multiple cores):

```
Bash

yes > /dev/null &
yes > /dev/null & # Repeat for each core
```

Monitor: Watch your Grafana dashboard. You should see the CPU utilization spike.

Wait: Wait for the FOR duration (e.g., 5 minutes) of your alert rule.

Verify Alert: Check your email or webhook.site to see if the alert notification was received.

Stop Stress Test:

For stress-ng, it will stop after the timeout.

For yes > /dev/null &, you'll need to kill the processes:

```
Bash
killall yes
```

Observe the alert state in Grafana returning to "Normal" and potentially a "Resolved" notification being sent.

This completes the step-by-step implementation. You now have a working monitoring solution with visualization and alerting!

Remember to capture screenshots at various stages as planned for your pictorial representation and presentation.

VII.OUTPUT & TESTING

Linux Server : Node Exporter running as a service, exposing CPU metrics on port 9100, accessible via http://<Linux_Server_IP>:9100/metrics.
Windows (Prometheus): Prometheus running, configured to scrape Node Exporter, and its web UI (http://localhost:9090/targets) shows the Linux server as an "UP" target, indicating successful data collection.
Windows (Grafana): Grafana running, successfully connected to Prometheus as a data source. A dashboard displays a dynamic CPU utilization graph, and an alert rule is configured for high CPU.
Notifications : Alert notifications (e.g., email, webhook) are received when the CPUthreshold is breached.

Testing :

Node Exporter : Verify service status (active (running)), local curl output, and remote browser access from Windows.
Prometheus : Check its Targets page in the UI to confirm Node Exporter is being scraped successfully (status UP).
Grafana Visualization : Observe the CPU graph on the dashboard updating in real-time. Perform a CPU load test on the Linux server and visually confirm the graph reflects the spike.
Grafana Alerting: Send a test notification from Grafana to verify the contact point. Then, induce high CPU load on the Linuxserver and confirm an alert notification is received within the configured timeframe.

Resolution : After stopping the load, observe the alert resolving and potentially a "resolved" notification.

VIII.CONCLUSION

This project successfully demonstrates the design, implementation, and operationalization of a robust and cost-effective monitoring solution for Linux server CPU utilization using a powerful open-source stack comprising Node Exporter, Prometheus, and Grafana.

Through a structured, sprint-based approach, we have achieved the following key objectives:

- Seamless Data Collection
- Centralized Data Storage and Retrieval
- Intuitive Visualization
- Proactive Alerting