

NETWORK MONITORING AND ALERTING APPLICATION USING PROMETHEUS, NODE EXPORTER, ALERTMANAGER AND GRAFANA

1. INTRODUCTION

This report provides an overview and documentation about our networking project focused on monitoring using Prometheus, Node Exporter, Alertmanager, and Grafana. The primary goal of this project is to establish a monitoring system that allows real-time tracking of system metrics, alerting on predefined thresholds, and visualizing data for analysis and decision-making.

1.1. OBJECTIVES OF THE PROJECT

The objectives of this project include:

- **Monitoring Infrastructure:** Implement Prometheus as the core monitoring system to collect and store time-series data related to various aspects of the network and system.
- **Node-Level Metrics:** Utilize Node Exporter to gather detailed metrics from individual nodes, including CPU usage, memory utilization, disk I/O, and network traffic.
- **Alerting Mechanism:** Configure Alertmanager to generate alerts based on defined rules and thresholds, ensuring timely notification of critical issues or anomalies.
- **Visualization and Analysis:** Leverage Grafana to create informative dashboards that provide insights into the performance and health of the network infrastructure.
- **Operational Efficiency:** Enhance operational efficiency by enabling proactive monitoring, rapid incident response, and data-driven decision-making processes.

2. TOOLS AND TECHNOLOGIES USED

1. Prometheus:

Prometheus is an open-source monitoring and alerting toolkit designed for reliability and scalability. Its role in this project is to continuously scrape and store time-series data from various targets, including servers, containers, and services. Prometheus uses a pull-based model to collect metrics, allowing real-time monitoring of system performance, resource utilization, and application health. It also supports the creation of custom metrics and alerting rules based on predefined conditions, enabling proactive incident detection and response.

2. Node Exporter:

Node Exporter is a Prometheus exporter responsible for collecting system-level metrics from target nodes. Its purpose in this project is to gather detailed information about node resources such as CPU usage, memory utilization, disk I/O operations, network traffic, and other relevant statistics. Node Exporter exposes these metrics in a format compatible with Prometheus, allowing them to be scraped and stored for further analysis, visualization, and alerting.

3. Alertmanager:

Alertmanager is a component of the Prometheus ecosystem that manages alerts and notifications generated by Prometheus server. In this project, Alertmanager plays a crucial role in handling alerting rules defined in Prometheus configuration (prometheus.yml). It receives alerts from Prometheus based on predefined conditions, applies routing and inhibition rules, and forwards notifications to appropriate receivers such as email, Slack, PagerDuty, or other communication channels. Alertmanager ensures timely and efficient alert handling, escalation, and resolution, improving incident response and system reliability.

4. Grafana:

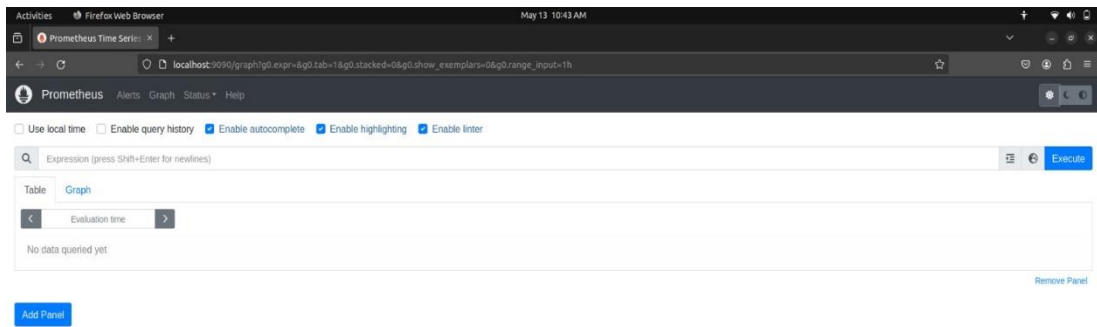
Grafana is a popular open-source platform for monitoring and observability that specializes in data visualization and dashboarding. Its role in this project is to provide a user-friendly interface for creating and managing dashboards that visualize metrics collected by Prometheus. Grafana integrates seamlessly with Prometheus as a data source, allowing users to build interactive, customizable dashboards with graphs, charts, tables, and other visualization components. Grafana's rich visualization capabilities enhance data analysis, trend identification, and performance monitoring, making it an essential tool for monitoring infrastructure and making informed decisions.

3. INSTALLATION GUIDE

3.1. PROMETHEUS

Prometheus has various ways of installation for example - pre-compiled binaries, from source, Docker. But to keep the installation simple we are going to use the pre-compiled binaries for installing Prometheus onto my Ubuntu Linux machine.

1. Goto Download Page of Prometheus and select the **prometheus-x.xx.x.linux-amd64.tar.gz** file for download for your operating system. (Note- Here x.xx.x is version number) (i.e. 2.52.0)
2. Extract the download binary file using the command **tar xvfz** for extracting tar file in linux os. (Note- Replace x.xx.x with the downloaded version of Prometheus) (i.e. **tar xvfz prometheus-x.xx.x.linux-amd64.tar.gz**)
3. Go into the extracted directory **cd prometheus-x.xx.x.linux-amd64.tar.gz**. Use **ls** command to see the files in the extracted directory.
4. Start the prometheus server with the command **'./prometheus'**
5. The Prometheus server should start on port 9090(Port 9090 is default port for prometheus)
6. You can access the Prometheus graph UI by visiting <http://localhost:9090/graph>

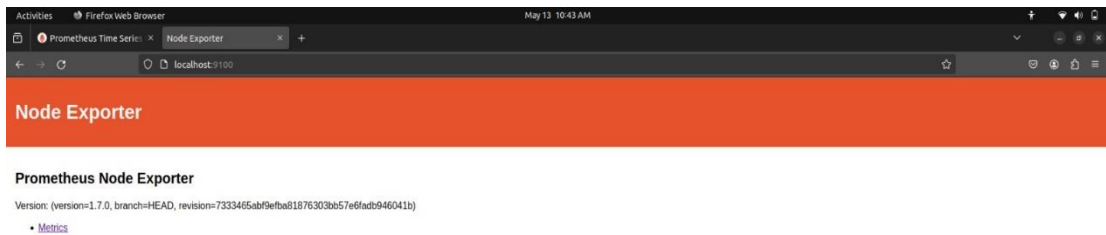


7. You can access the Prometheus metrics UI by visiting <http://localhost:9090/metrics>

3.2. NODE EXPORTER

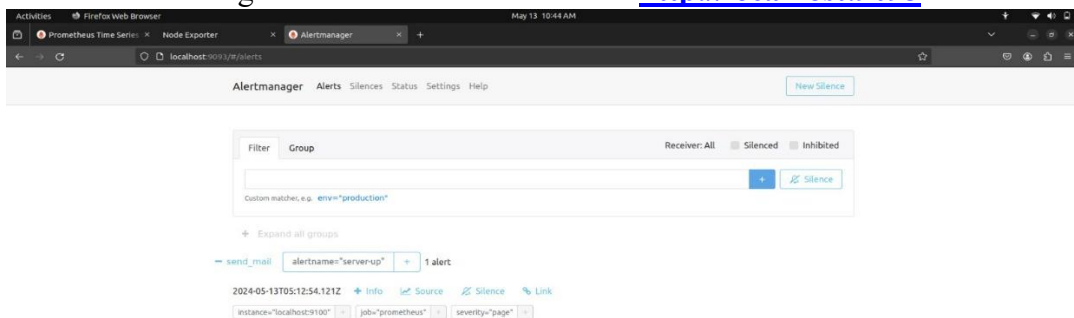
After installing the Prometheus in the previous step the next package we are going to install is Node Exporter. Node exporter is used for collecting various hardware and kernel-level metrics of your machine.

1. Download the binary of Node exporter based on the operating system.(In my case I am using Ubuntu linux machine)
2. Download the binary of Node exporter based on the operating system. You can download it from the Download Page of Prometheus and select the **node_exporter-x.xx.x.linux-amd64.tar.gz** file for download for your operating system. (Note- Here x.xx.x is version number) (i.e. 1.8.0)
3. Extract the download node exporter binary file. Using tar xvfz for extracting tar file in linux os. (Note- Replace x.xx.x with the downloaded version of Node exporter) (i.e. **tar xvfz node_exporter-x.xx.x.linux-amd64.tar.gz**)
4. Go into the extracted directory **cd node_exporter-x.xx.x.linux-amd64.tar.gz**. Use **ls** command to see the files in the extracted directory.
5. Start the node exporter with the command **'./node_exporter'**
6. Access the Node exporter metrics on the browser with URL - <http://localhost:9100>



3.3. ALERT MANAGER

1. Download the binary of Alertmanager based on the operating system.(In my case I am using Ubuntu linux machine)
2. Download the binary of Alertmanager based on the operating system. You can download it from the Download Page of Prometheus and select the **alertmanager-x.xx.x.linux-amd64.tar.gz** file for download for your operating system. (Note- Here x.xx.x is version number) (i.e. 0.27.0)
3. Extract the download alertmanager binary file. Using tar xvfz for extracting tar file in linux os. (Note- Replace x.xx.x with the downloaded version of Alertmanager) (i.e. **tar xvfz alertmanager-x.xx.x.linux-amd64.tar.gz**)
4. Go into the extracted directory cd **alertmanager-x.xx.x.linux-amd64.tar.gz**.Use **ls** command to see the files in the extracted directory.
5. Start the alertmanager with the command **'./alertmanager'**
6. Access Alertmanager on the browser with URL - <http://localhost:9093>



3.4. GRAFANA

After installing the Prometheus server and node exporter let's install the Grafana for setting up visualization using Grafana dashboard. Choose the correct Grafana binary based on your operating system.

Official Download link: <https://grafana.com/docs/grafana/latest/setup-grafana/installation/>

1. Install the prerequisite packages.
Command: **sudo apt-get install -y apt-transport-https software-properties-common wget**
2. Import the GPG key.
Command: **sudo mkdir -p /etc/apt/keyrings/**
 - i. **wget -q -O - https://apt.grafana.com/gpg.key | gpg --dearmor | sudo tee /etc/apt/keyrings/grafana.gpg > /dev/null**
3. To add a repository for stable releases.
Command: **echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] https://apt.grafana.com stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list**
4. To add a repository for beta releases.
Command: **echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] https://apt.grafana.com beta main" | sudo tee -a /etc/apt/sources.list.d/grafana.list**
5. Run the following command to update the list of available packages
Command: **sudo apt-get update**
6. To install Grafana OSS.
Command: **sudo apt-get install Grafana**
7. To install Grafana Enterprise.
Command: **sudo apt-get install grafana-enterprise**

Access the Grafana Dashboard using <http://localhost:3000/login>

The default login Username: **Password** for accessing the Grafana dashboard is **admin: admin**. Change the default admin password after the login

4. CONFIGURATION

4.1. PROMETHEUS CONFIGURATION:

1. **Targets Configuration:** Edit the prometheus.yml file to specify the targets Prometheus should scrape metrics from. Configure node exporters, application endpoints, and any other relevant targets.
Prometheus.yml

```

global:
  scrape_interval: 15s
  evaluation_interval: 15s
# Alertmanager
alerting:
  alertmanagers:
    - static_configs:
        - targets:
            - 'localhost:9093'
rule_files:
  - "rule1.yml"
scrape_configs:
  - job_name: "prometheus"
    static_configs:
      - targets: ['localhost:9100', '10.184.49.77:9100']

```

2. **Alerting Rules Setup:** Define alerting rules in the alert.rules file to trigger alerts based on specific conditions or thresholds.

Rules.yml

```

groups:
  - name: server-down
    rules:
      - alert: server-up
        expr: up == 1
        for: 1m
        labels:
          severity: page
        annotations:
          summary: Server is working
      - alert: HighCPULoad
        expr: node_load1 > 0.7
        for: 5m
        labels:
          severity: critical
        annotations:
          summary: "High CPU load detected"
          description: "The CPU load on {{ $labels.instance }} is above 0.7 for 5 minutes."
      - alert: HighRamUsage
        expr: (node_memory_Active_bytes / node_memory_MemTotal_bytes) > 0.3
        for: 2m
        labels:
          severity: warning
        annotations:
          summary: "High RAM usage detected"
          description: "The RAM usage on {{ $labels.instance }} is above 30% for 5 minutes."

```

4.2. ALERTMANAGER CONFIGURATION:

1. **Notification Integrations:** Configure notification integrations in the alertmanager.yml file to enable alert notifications via email, Slack, PagerDuty, etc.
 - Name and receiver should match
 - Alertmanager follows SMTP protocol for transferring of emails, so if we're using email as our way of medium for alerting users there are certain steps to be followed.
 - Enable two-step authentication in your mail ID and then generate a app password so that it should be used in '**auth_password**' as password to enable communication

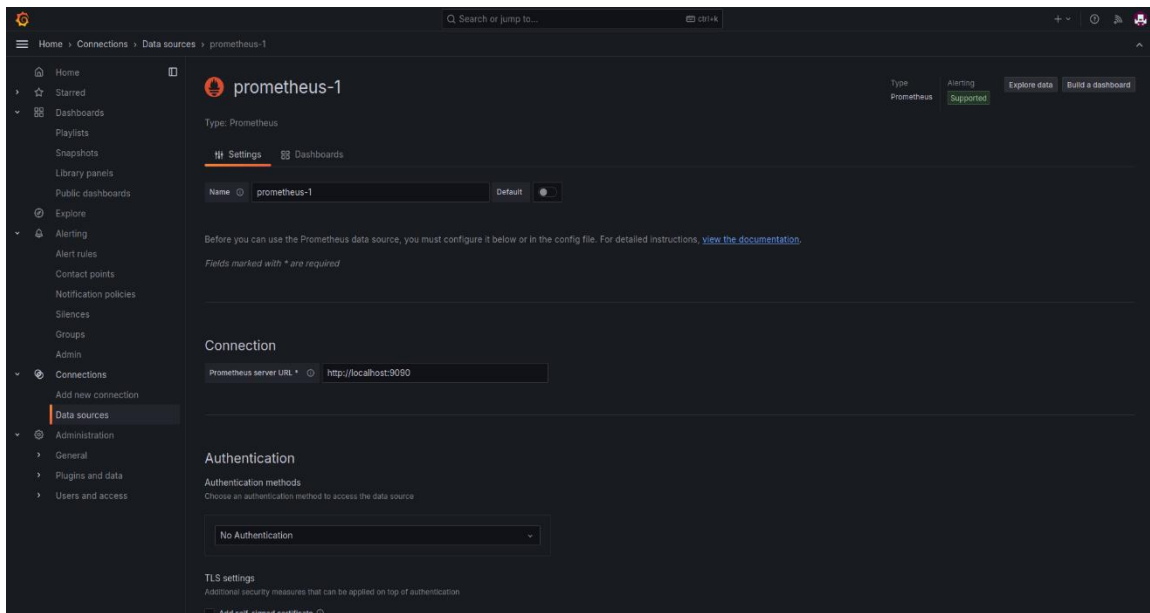
```
route:
  group_by: ['alertname']
  receiver: 'send_mail'
receivers:
- name: 'send_mail'
  email_configs:
  - to: boobashdayal@gmail.com
    from: boobashdayals64@gmail.com
    smarthost: smtp.gmail.com:587
    auth_username: boobashdayals64@gmail.com
    auth_password: buxm fidr aaqc gcso
    require_tls: true
```

2. **Routing and Inhibition Rules:** Define routing and inhibition rules in the alertmanager.yml file to control how alerts are routed and suppressed.

```
inhibit_rules:
- source_match:
  severity: 'critical'
  target_match:
  severity: 'warning'
  equal: ['alertname', 'dev', 'instance']
```

4.3. GRAFANA CONFIGURATION:

1. **Adding Prometheus as a Data Source:** In Grafana, navigate to Configuration > Data Sources and add Prometheus as a data source. Specify the URL of your Prometheus server.
Name: Prometheus
Type: Prometheus
URL: <http://localhost:9090>



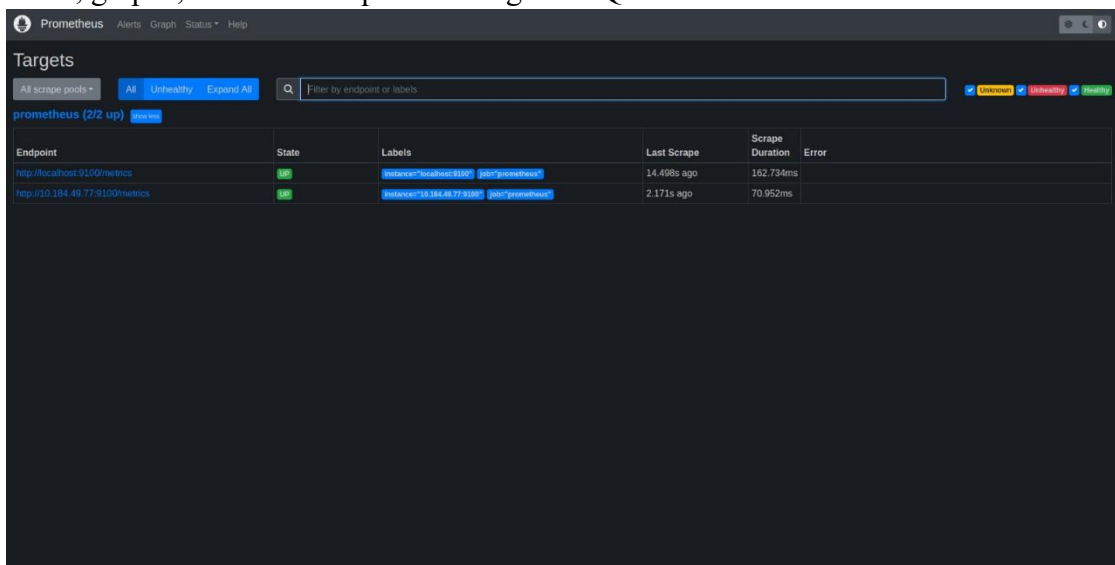
2. **Importing Pre-built Dashboards or Creating Custom Dashboards:** Import existing dashboards from the Grafana dashboard repository or create custom dashboards to visualize metrics from Prometheus.

- To import a dashboard, go to the Grafana dashboard section, click on "Import Dashboard," and provide the dashboard ID or JSON file.
- To create a custom dashboard, click on "Create > Dashboard" and add panels to visualize metrics fetched from Prometheus data source.

5. USAGE

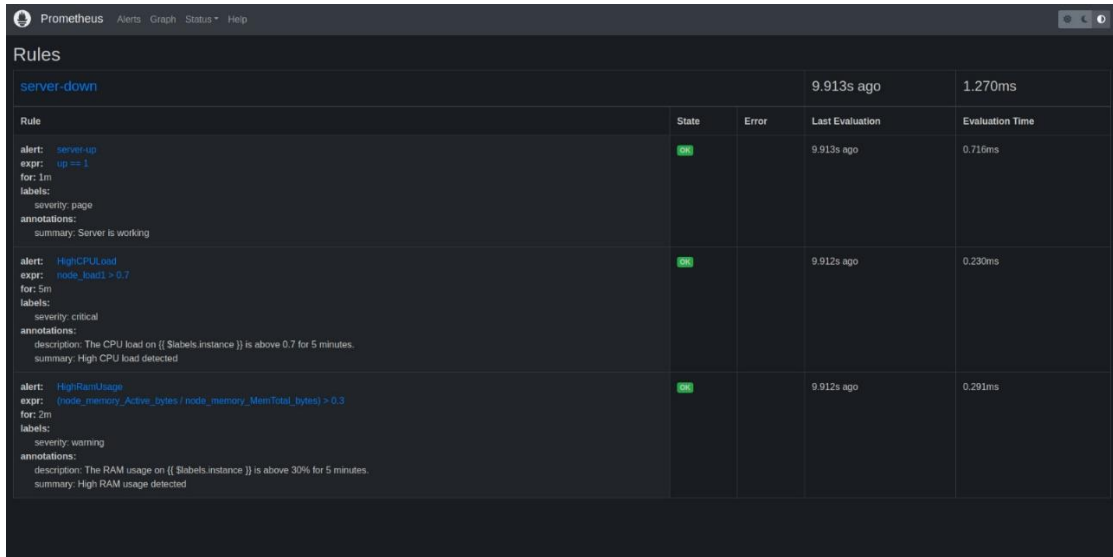
5.1. PROMETHEUS:

- Access Prometheus through a web browser using the URL: `http://localhost:9090` (replace localhost with your server's hostname or IP address if applicable).
- Prometheus provides a user-friendly web interface where you can view targets, metrics, alerts, graphs, and execute queries using PromQL.



5.1.1. MONITORING METRICS AND SYSTEM HEALTH:

- Use Prometheus's query interface to monitor metrics and system health.
- Explore predefined metrics and create custom queries using PromQL to analyze performance, resource utilization, network traffic, and other metrics.
- Set up alerting rules in Prometheus (alert.rules) to monitor critical metrics and trigger alerts when thresholds are breached.



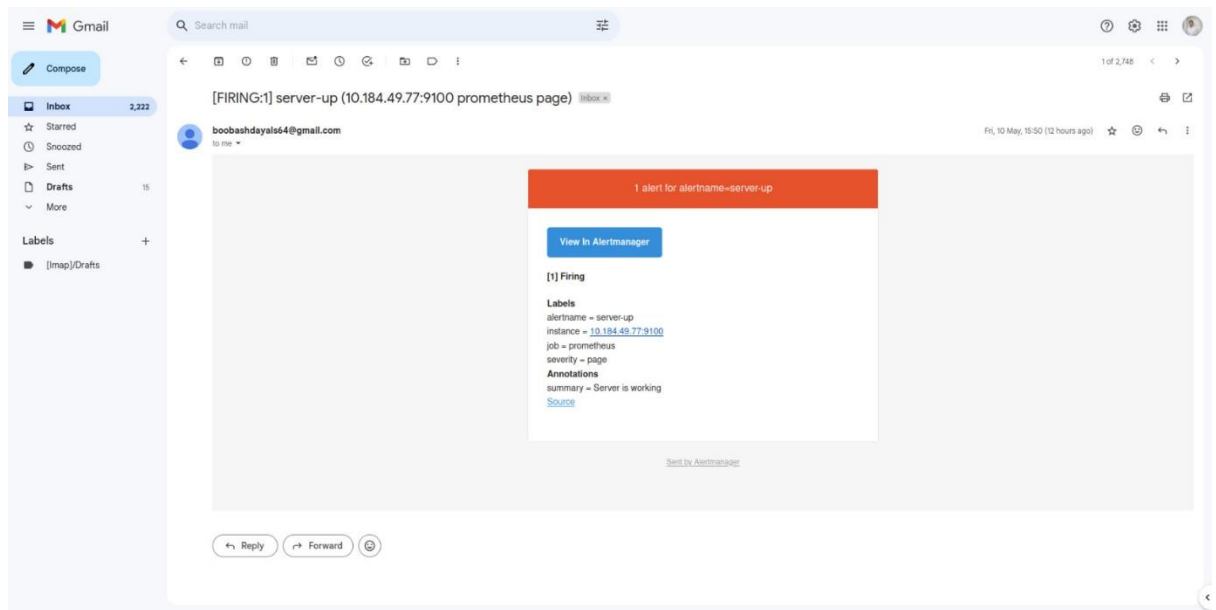
Rule	State	Error	Last Evaluation	Evaluation Time
server-down alert: server-down expr: up == 0 for: 1m labels: severity: page annotations: summary: Server is working	OK		9.913s ago	1.270ms
HighCPUload alert: HighCPUload expr: node_load1 > 0.7 for: 5m labels: severity: critical annotations: description: The CPU load on {{ \$labels.instance }} is above 0.7 for 5 minutes. summary: High CPU load detected	OK		9.912s ago	0.230ms
HighRamUsage alert: HighRamUsage expr: (node_memory_Active_bytes / node_memory_MemTotal_bytes) > 0.3 for: 2m labels: severity: warning annotations: description: The RAM usage on {{ \$labels.instance }} is above 30% for 5 minutes. summary: High RAM usage detected	OK		9.912s ago	0.291ms

5.2. ALERTMANAGER:

- Access Alertmanager through a web browser using the URL: <http://localhost:9093> (modify the URL as needed).
- Alertmanager's interface allows you to manage alert configurations, receivers, silence alerts, view alert status, and inspect notification history.

5.2.1. HANDLING ALERTS AND NOTIFICATIONS:

- Configure alerting rules in Prometheus (alert.rules) to define conditions for generating alerts.
- Use Alertmanager's interface to manage alert receivers, integrations (e.g., email, Slack), routing, inhibition rules, and notification templates.
- Monitor incoming alerts, acknowledge alerts, silence alerts, and resolve alerts based on severity and impact.

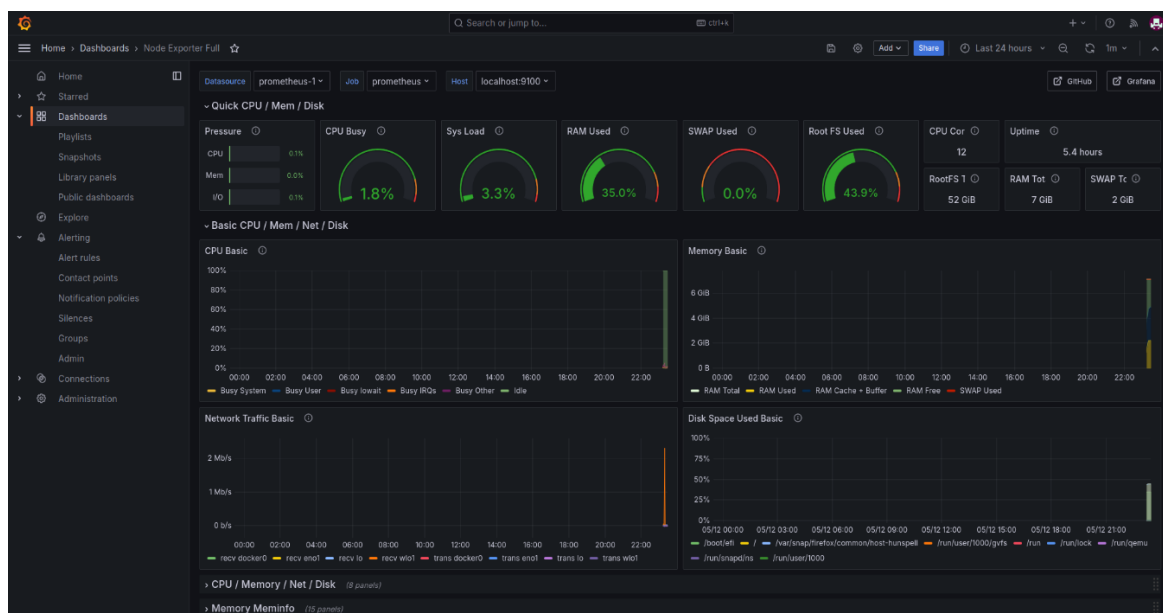


5.3. GRAFANA:

- Access Grafana through a web browser using the URL: <http://localhost:3000> (adjust the URL if Grafana is running on a different port or server).
- Grafana's interface provides a dashboarding platform where you can create, edit, and visualize dashboards using data from various sources, including Prometheus.

5.3.1. MONITORING METRICS AND SYSTEM HEALTH:

- Create dashboards in Grafana to visualize metrics fetched from Prometheus.
- Add panels such as graphs, tables, gauges, and logs to monitor specific metrics, system health, trends, and anomalies.
- Customize dashboard layouts, themes, and refresh intervals for real-time monitoring and analysis.



6. CONCLUSION

This project successfully implemented a monitoring system using Prometheus, Node Exporter, Alertmanager, and Grafana to monitor the network infrastructure and system health. Prometheus served as the core monitoring tool, collecting and storing time-series data, while Node Exporter gathered detailed node-level metrics. Alertmanager handled alerting and notifications, ensuring timely responses to critical issues. Grafana provided visualization and analysis capabilities, creating informative dashboards for monitoring performance and health metrics.