Name: Venkata Lahari Goruputi

Role: DevOps Intern

Topic: Node Exporter and Alert Manager in Prometheus and Grafana

Install Node Exporter and Alert Manager

✓ Node Exporter is a tool used with Prometheus to collect hardware and OS-level metrics from Linux machines. Alert Manager is a component that handles alerts sent by Prometheus, managing routing, silencing, and notifications (like email).

What is Node Exporter:

- Node Exporter is a monitoring tool used to collect system metrics from Linux and Unix-based systems.
- Node Exporter is a powerful tool for collecting system metrics, which can be scraped by Prometheus and visualized in Grafana
- It acts as an exporter that exposes these metrics in a format that Prometheus can scrape and store.
- Grafana is often used in conjunction with Prometheus to visualize the metrics collected by Node Exporter.
- Here, I wanted to monitor an EC2 instance so for that we need to install node exporter to expose the metrics of the EC2 instance to prometheus and grafana.

  Node Exporter collects a variety of system metrics, such as:
  - CPU usage
  - Memory usage
  - Disk I/O
  - Network statistics
  - Filesystem statistics
  - System load
- These metrics are made available via an HTTP endpoint which Prometheus scrapes.

What is Alert Manager:

- Alert Manager is a component of the Prometheus ecosystem that handles alerts generated by Prometheus servers. Its primary functions include:

- **Alert Deduplication:**
  - Combining duplicate alerts into a single notification to reduce noise.
- **Alert Routing:**
  - Directing alerts to the appropriate recipients based on preconfigured rules.
- **Silencing:**
  - Temporarily muting alerts during maintenance or known downtime.
- **Notification Management:**
  - Sending notifications via various channels like email, Slack, PagerDuty, etc., based on alert severity and other criteria.

- Alert Manager helps manage and streamline alerting workflows, ensuring that the right people are notified about issues in a timely and efficient manner.

✓ Installation and Configuration:

✓ Installing Node Exporter:

❖ Download Node Exporter:

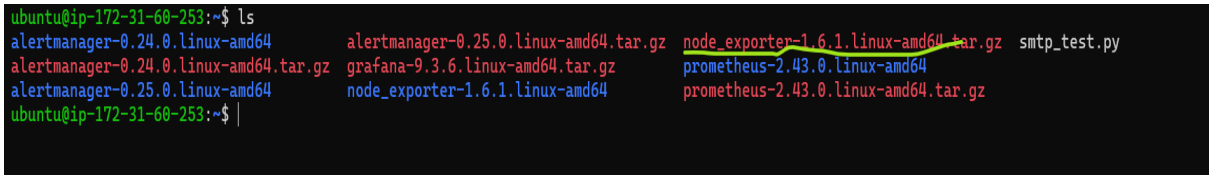
- Download the tar file of node exporter from the official docs.

Download Node Exporter

Command:

[wget https://github.com/prometheus/node_exporter/releases/download/v1.6.1/node_exporter-1.6.1.linux-amd64.tar.gz]

- The above command will download the tar file of Node Exporter. Here I have used [node_exporter-1.6.1.linux-amd64.tar.gz] version.



❖ Extracting the downloaded Node Exporter tar file:

- After Downloading the tar file, we need to extract the tar file to get all the binaries required to install the Node Exporter.

Command:

[sudo tar xvfz node_exporter-1.6.1.linux-amd64.tar.gz]

- This command will extract the tar file in the home path, after the extraction is done we will get a folder named node_exporter-1.6.1.linux-amd64 which contains all the binaries and dependencies required to install the Node Exporter.

❖ **Move the Extracted Files:**

- Move the extracted Node Exporter directory to **/usr/local/bin.**

Command:

[sudo mv node_exporter-1.3.1.linux-amd64/node_exporter /usr/local/bin/]

- The above commands will move the files to bin folder/PATH variable to let the PATH variable know the default location of the Node Exporter.
- Because Executables are accessible system-wide by placing them in **/usr/local/bin/.**

```
ubuntu@ip-172-31-60-253:~$ cd /usr/local/bin/
ubuntu@ip-172-31-60-253:/usr/local/bin$ ls
alertmanager  amtool  grafana-cli  grafana-server  node_exporter  prometheus  promtool
ubuntu@ip-172-31-60-253:/usr/local/bin$
```

❖ **Create a User for Node Exporter:**

- Create a new system user for running Node Exporter. We need to create a dedicated user for Node Exporter so that service follows best practices for Unix/Linux system administration.
- So, we need create a user and change the ownership and permissions for the user accordingly.

Command:

[sudo useradd -rs /bin/false node_exporter]

```
ubuntu@ip-172-31-60-253:~$ cat /etc/passwd | grep node
node_exporter:x:1002:1002::/home/node_exporter:/bin/false
ubuntu@ip-172-31-60-253:~$
```

❖ **Create Service for Node Exporter:**

- Now we need to create the service for Node Exporter. Creating a new systemd service file for Node Exporter is essential for managing Node Exporter as a background service on a Linux system.

[sudo nano /etc/systemd/system/node_exporter.service]

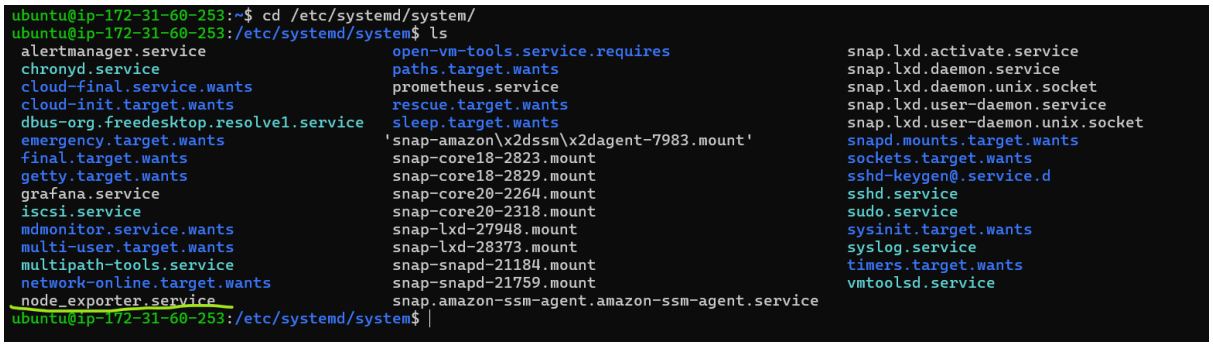
(or)

[sudo vi /etc/systemd/system/node_exporter.service]

- This above command will create a service file for Node Exporter in **/etc/systemd/system** path with the service name as **node_exporter.**

```
[Unit]
Description=Node Exporter
Wants=network-online.target
After=network-online.target

[Service]
User=node_exporter
Group=node_exporter
ExecStart=/usr/local/bin/node_exporter

[Install]
WantedBy=default.target
```

- In the service file we need to add the below content which is essential to tun Node Exporter as a service. This service file contains the location of Node Exporter and other details to tun Node Exporter as a service.
- After adding the above content in **node_exporter.service** file we need to save the file.
- Save the file using **:wq!** in vi editor and with **ctrl+X and Y** in nano editor.



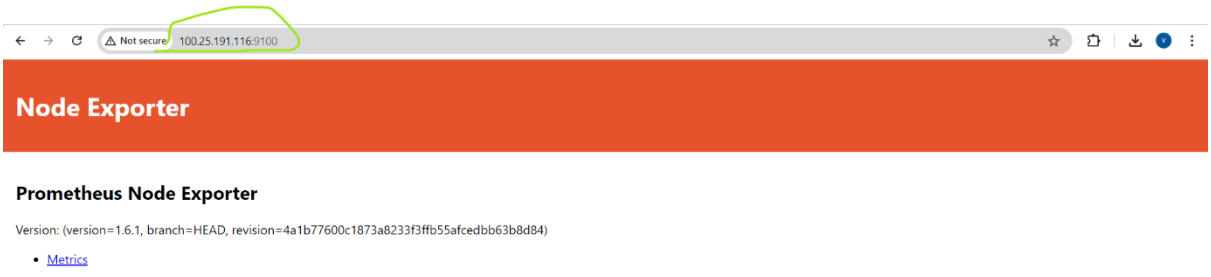
❖ Run and Check Node Exporter Status:

- Then we need to run commands which will restart the daemon and start and enables the node_exporter service and also let us know the Node Exporter status.

```
sudo systemctl daemon-reload

sudo systemctl start node_exporter

sudo systemctl enable node_exporter

sudo systemctl status node_exporter
```

```
ubuntu@ip-172-31-60-253:~$ sudo systemctl status node_exporter.service
● node_exporter.service - Node Exporter
     Loaded: loaded (/etc/systemd/system/node_exporter.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2024-06-30 06:04:40 UTC; 52min ago
       Docs: https://github.com/prometheus/node_exporter
   Main PID: 388 (node_exporter)
      Tasks: 4 (limit: 9497)
     Memory: 21.6M
        CPU: 2.311s
     CGroup: /system.slice/node_exporter.service
             └─388 /usr/local/bin/node_exporter

Jun 30 06:04:41 ip-172-31-60-253 node_exporter[388]: ts=2024-06-30T06:04:41.173Z caller=node_exporter.go:117 level=info collector=thermal_zone
Jun 30 06:04:41 ip-172-31-60-253 node_exporter[388]: ts=2024-06-30T06:04:41.173Z caller=node_exporter.go:117 level=info collector=time
Jun 30 06:04:41 ip-172-31-60-253 node_exporter[388]: ts=2024-06-30T06:04:41.173Z caller=node_exporter.go:117 level=info collector=timex
Jun 30 06:04:41 ip-172-31-60-253 node_exporter[388]: ts=2024-06-30T06:04:41.173Z caller=node_exporter.go:117 level=info collector=udp_queues
Jun 30 06:04:41 ip-172-31-60-253 node_exporter[388]: ts=2024-06-30T06:04:41.173Z caller=node_exporter.go:117 level=info collector=uname
Jun 30 06:04:41 ip-172-31-60-253 node_exporter[388]: ts=2024-06-30T06:04:41.173Z caller=node_exporter.go:117 level=info collector=vmstat
Jun 30 06:04:41 ip-172-31-60-253 node_exporter[388]: ts=2024-06-30T06:04:41.173Z caller=node_exporter.go:117 level=info collector=xfs
Jun 30 06:04:41 ip-172-31-60-253 node_exporter[388]: ts=2024-06-30T06:04:41.173Z caller=node_exporter.go:117 level=info collector=zfs
Jun 30 06:04:41 ip-172-31-60-253 node_exporter[388]: ts=2024-06-30T06:04:41.179Z caller=tls_config.go:274 level=info msg="Listening on" address=[::]:9100
Jun 30 06:04:41 ip-172-31-60-253 node_exporter[388]: ts=2024-06-30T06:04:41.179Z caller=tls_config.go:277 level=info msg="TLS is disabled." http2=false addr
lines 1-21/21 (END)
```

- If our service is up and running, we can now able to access the node_exporter UI in the browser using the public IP of the server on port 9100. **http://<public-ip>:9100.**



❖ Configure Prometheus to Scrape Node Exporter:

- After the successful installation we need to configure Node Exporter with prometheus to scrape metrics to prometheus by editing the prometheus.yml file which will be present in /etc/prometheus path.
- Edit the Prometheus Configuration File (prometheus.yml) as below and add the content to scrape metrics to prometheus.

```
scrape_configs:
  - job_name: 'node_exporter'
    static_configs:
      - targets: ['localhost:9100']
```

```
ubuntu@ip-172-31-60-253:~$ sudo vi /etc/prometheus/prometheus.yml
ubuntu@ip-172-31-60-253:~$
```

```
# 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. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets:
          # - alertmanager:9093
          - '54.210.122.30:9093'


# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"
  - "alert.rules.yml"
  - "alert_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["54.210.122.30:9090"]

  - job_name: "node_exporter"
    static_configs:
      - targets: ["172.31.60.253:9100"]
~
~
~
"/etc/prometheus/prometheus.yml" 37L, 1109B                                   33,32
```

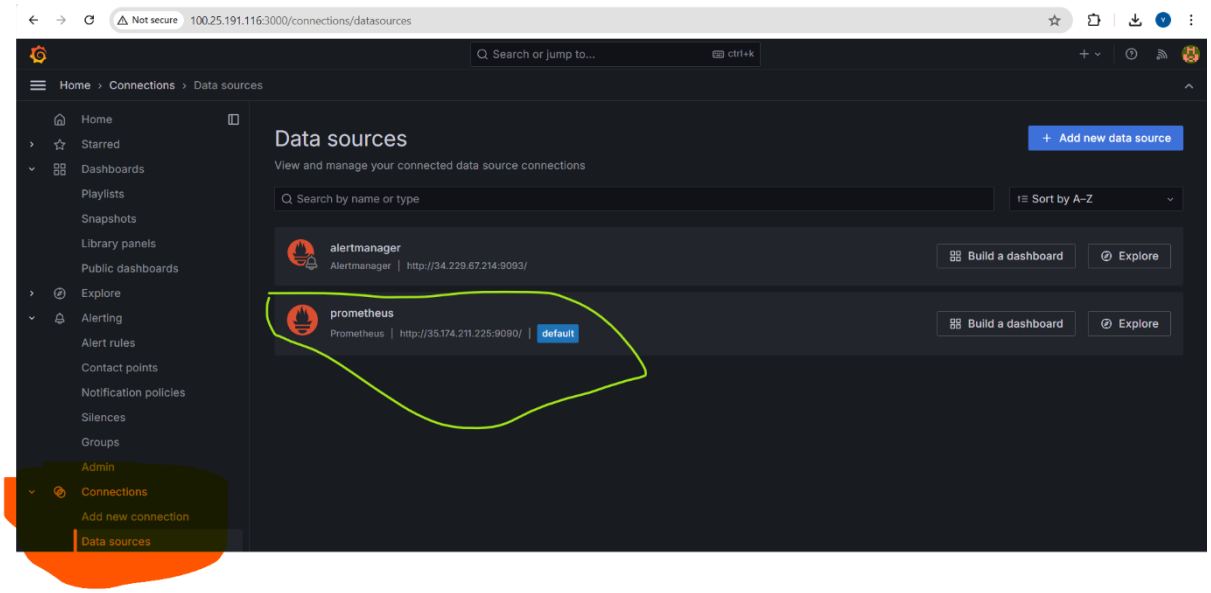
Note:

- Replace 'localhost:9100' with the appropriate address if Node Exporter is running on a different machine.
- As we have changed prometheus configuration we need to restart the prometheus service using **[sudo systemctl restart prometheus].**

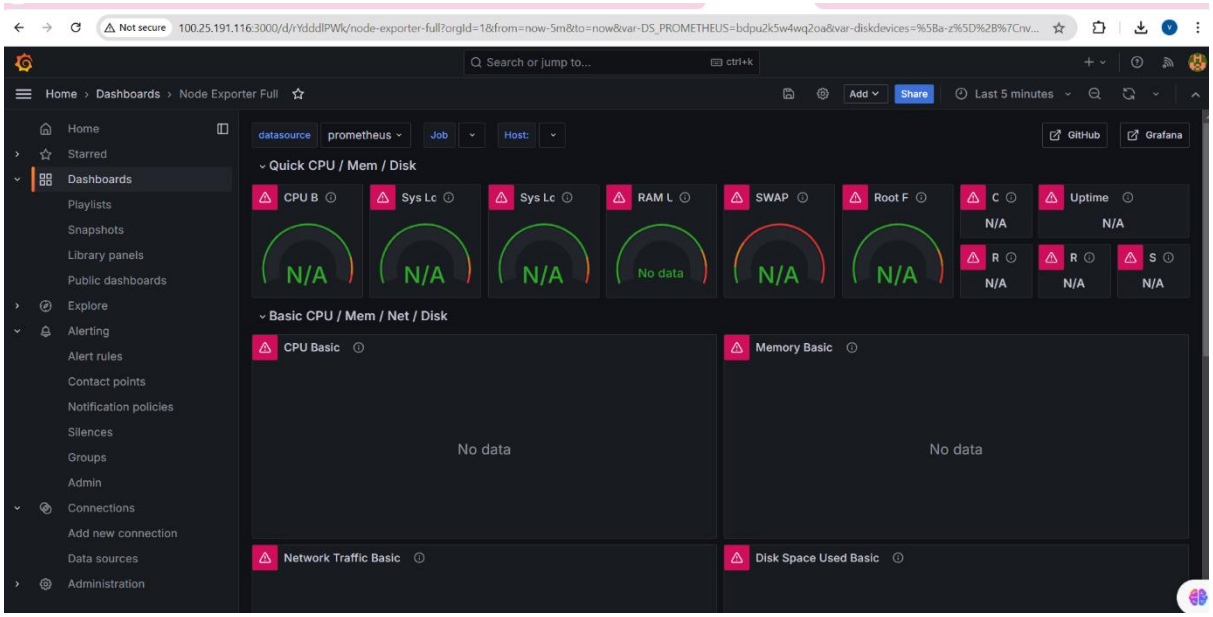
```
ubuntu@ip-172-31-60-253:~$ sudo systemctl restart prometheus.service
```

- Then we can able to Visualize Metrics in Grafana, for this we need to add prometheus data source in grafana, follow the below steps for this.

- **Add Prometheus as a Data Source in Grafana:**
  - Open Grafana in web browser **http://grafana-server-ip:3000**
  - Go to Configuration > Data Sources > Add data source.
  - Select Prometheus and enter the URL of the Prometheus server http://prometheus-server-ip:9090. Then save and test the configuration.
  - If it returns success then prometheus data source is added in grafana.
  - Then we need to import the dashboard in grafana to visualize the metrics.



- **Import a Node Exporter Dashboard:**
  - Go to Create > Import.
  - You can find pre-built Node Exporter dashboards on Grafana's dashboard repository.
  - Enter the dashboard ID and import it, then we can able to see the EC2 instance metrics visually from graphs in the imported grafana dashboard.



✓ Installing and configuring Alert Manager:

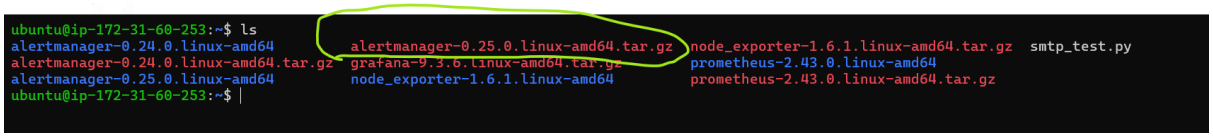
❖ Download Alert Manager:

- Download the tar file of Alert Manager from the official docs.

Download Alert Manager

Command:

[wget https://github.com/prometheus/alertmanager/releases/download/v0.25.0/alertmanager-0.25.0.linux-amd64.tar.gz]



- The above command will download the Alert Manager tar file.

❖ Extracting the downloaded Alert Manager tar file:

- After Downloading the tar file, we need to extract the tar file to get all the binaries required to install the Alert Manager.

Command:

[sudo tar xvfz alertmanager-0.25.0.linux-amd64.tar.gz]

- The above command will extract the tar file in the home path, after the extraction is done we will get a folder named alertmanager-0.25.0.linux-amd64 which contains all the binaries and dependencies required to install the Alert Manager.

❖ Move the Extracted Files:

- Move the extracted Alert Manager directory to **/usr/local/bin.**

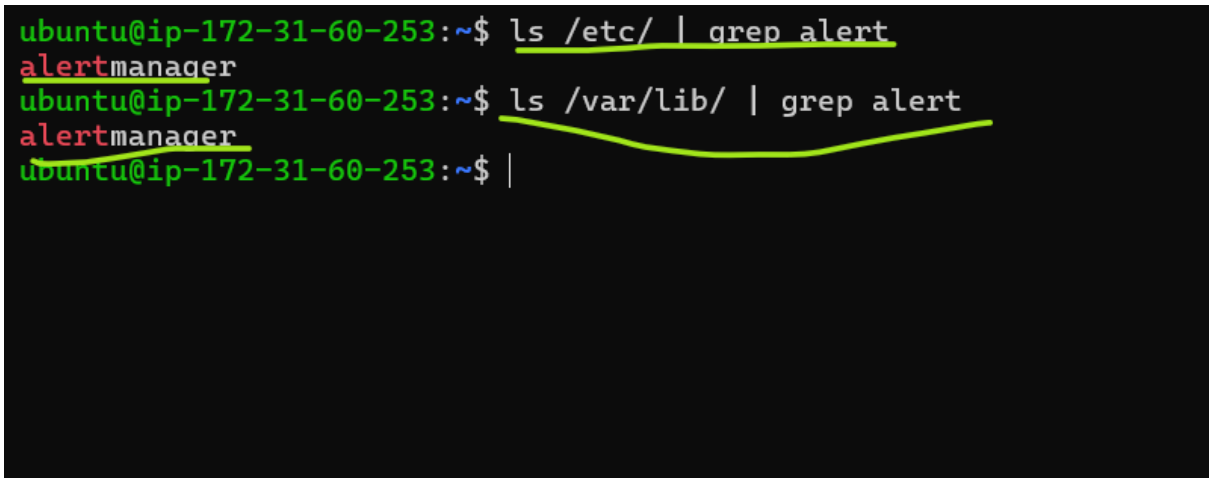
```
sudo mv alertmanager-0.25.0.linux-amd64/alertmanager /usr/local/bin/

sudo mv alertmanager-0.25.0.linux-amd64/amtool /usr/local/bin/
```

- The above commands will move the files to bin folder/PATH variable to let the PATH variable know the default location of the Alert Manager.
- Because Executables are accessible system-wide by placing them in **/usr/local/bin/.**

❖ Create Directories for Configuration and Data:

```
sudo mkdir /etc/alertmanager
sudo mkdir /var/lib/alertmanager
```

- **/etc/alertmanager** This directory will store the configuration file (alertmanager.yml).
- **/var/lib/alertmanager** This directory will store persistent data for Alertmanager.

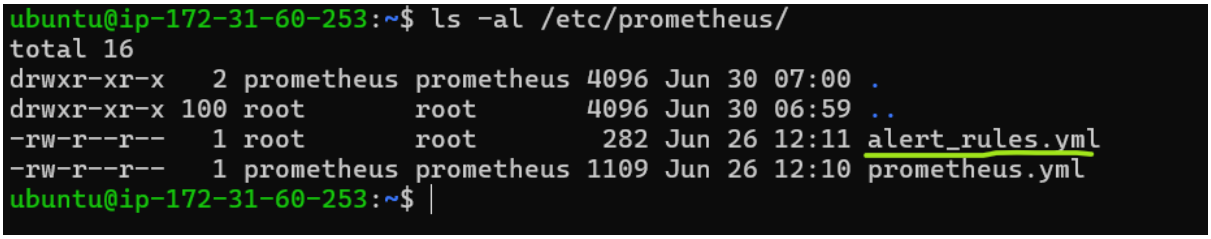


Note:

- Creating dedicated directories for configuration and data is a best practice for organizing, securing, and managing the Alertmanager installation effectively. It separates concerns, enhances security, and ensures persistent storage for critical application data.

❖ Setting up Alerting Rules in Prometheus:

- After the successful Installation we need to configure the Alert Manager to send the alerts for the events.
- Create an alerting rules file (e.g., alert_rules.yml) in /etc/prometheus that defines the conditions under which alerts are triggered.
- Edit the **prometheus.yml** file which is present in **/etc/prometheus** and edit the file with the below content to configure alert manager with prometheus for alerts.
- Ensure localhost:9093 is replaced with the correct address if Alertmanager is running on a different machine or port.



```
global:
  scrape_interval: 15s
scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']


alerting:
  alertmanagers:
    - static_configs:
        - targets: ['localhost:9093']


rule_files:
  - 'alert_rules.yml'
```

- Create the **alert_rules.yml** file and add the below context to get the alerts for instance down event.

```
groups:
  - name: example_alerts
    rules:
      - alert: InstanceDown
        expr: up == 0
        for: 5m
        labels:
          severity: critical
        annotations:
          summary: "Instance {{ $labels.instance }} down"
          description: "{{ $labels.instance }} of job {{ $labels.job }} has been down for more than 5 minutes."
```

```
ubuntu@ip-172-31-60-253:~$ sudo vi /etc/prometheus/alert_rules.yml
ubuntu@ip-172-31-60-253:~$
```

```
groups:
- name: example
  rules:
  - alert: NodeExporterDown
    expr: up{job="node_exporter"} == 0
    for: 1m
    labels:
      severity: critical
    annotations:
      summary: "Node Exporter down"
      description: "Node Exporter is down on instance {{ $labels.instance }}."

~
~
~
```

Note:

- Ensure the rule_files section in prometheus.yml includes the path to your alerting rules file as shown above.

❖ Visualize Alerts in Grafana:

- **Add Prometheus as a Data Source in Grafana:**
  - Open Grafana in web browser **http://grafana-server-ip:3000**
  - Go to Configuration > Data Sources > Add data source.
  - Select Prometheus and enter the URL of the Prometheus server http://prometheus-server-ip:9090.
  - Then save and test the configuration, if it returns success then prometheus data source is added in grafana.

- **Import Alert Manager Dashboard:**
  - Go to Create > Import.
  - You can find pre-built Alert Manager dashboards on Grafana's dashboard repository.
  - Enter the dashboard ID and import it.
  - Then we can able to see the alerts visually from graphs in the imported grafana dashboard.

❖ Create a Configuration File to send email notifications for alerts:

- Create the **alertmanager.yml** configuration file in **/etc/alertmanager.**

Command:

[sudo nano /etc/alertmanager/alertmanager.yml] (or)

[sudo vi /etc/alertmanager/alertmanager.yml]

- After create the above file add the below context to the file and save the file using **:wq!** In vi editor and by using **ctrl+X and Y** in nano editor to send the notifications to the mail for alerts based on events.
- In the below content adjust the email configuration settings as necessary for SMTP server and notification preferences.

```
global:
  smtp_smarthost: 'smtp.gmail.com:587'
  smtp_from: 'alertmanager@yourdomain.com'
  smtp_auth_username: 'your-email@gmail.com'
  smtp_auth_password: 'your-app-password'
  smtp_require_tls: true

route:
  receiver: 'email'

receivers:
  - name: 'email'
    email_configs:
      - to: 'destination-email@example.com'
        send_resolved: true

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

```
ubuntu@ip-172-31-60-253:~$ sudo vi /etc/alertmanager/alertmanager.yml
```

```
global:
  smtp_smarthost: '[redacted]smtp.mailtrap.io:2525'
  smtp_from: 'root@abe866dd5717'
  smtp_auth_username: 'b78ccf26dfb795'
  smtp_auth_password: '[redacted]'

route:
  receiver: 'email'

receivers:
- name: 'email'
  email_configs:
  - to: '[redacted]'
~
~
~
~
```

❖ Create a User for Alert Manager:

- Create a new system user for running Alert Manager
- We need to create a dedicated user for Alert Manager so that service follows best practices for Unix/Linux system administration.
- It helps in tracking and managing resources used by specific services and simplifies troubleshooting.
- So, we need create a user and change the ownership and permissions for the user accordingly.

Command:

[sudo useradd -rs /bin/false alertmanager]

❖ Create Service for Alert Manager:

- Now we need to create the service for Alert Manager. Creating a new systemd service file for Alert Manager is essential for managing Alert Manager as a background service on a Linux system.
- **[sudo nano /etc/systemd/system/alertmanager.service]** (or) **[sudo vi /etc/systemd/system/alertmanager.service]**
- This above command will create a service file for Alert Manager in **/etc/systemd/system** path with the service name as alertmanager (Generally any service file will present in this location in linux).

- In the service we need to add the below content will is essential to run alertmanager as a service, this service file contains the location of alertmanager and other details.

```
[Unit]
Description=Alertmanager
Documentation=https://prometheus.io/docs/alerting/latest/alertmanager/
Wants=network-online.target
After=network-online.target

[Service]
User=alertmanager
Group=alertmanager
Type=simple
ExecStart=/usr/local/bin/alertmanager --
config.file=/etc/alertmanager/alertmanager.yml --storage.path=/var/lib/alertmanager
ExecReload=/bin/kill -HUP $MAINPID
Restart=on-failure

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

- After adding the above content in alertmanager.service file we need to save the file and then save the file using **:wq!** in vi editor and by using **ctrl+X and Y** in nano editor.

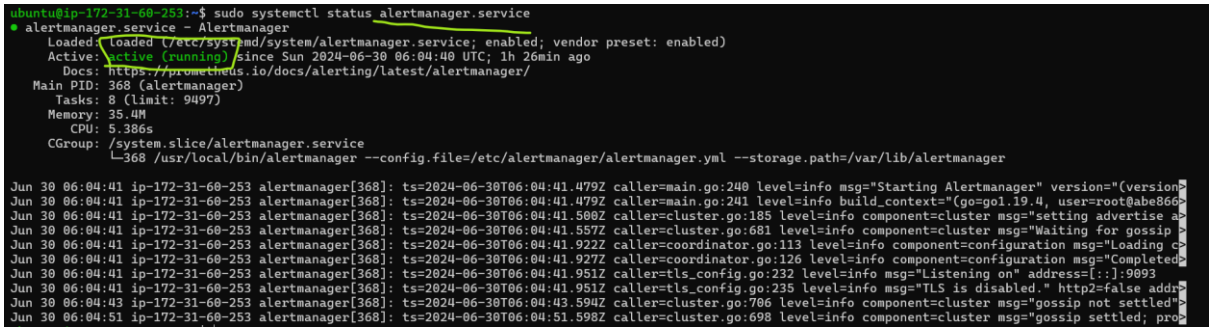
```
ubuntu@ip-172-31-60-253:~$ sudo vi /etc/systemd/system/alertmanager.service
ubuntu@ip-172-31-60-253:~$
```

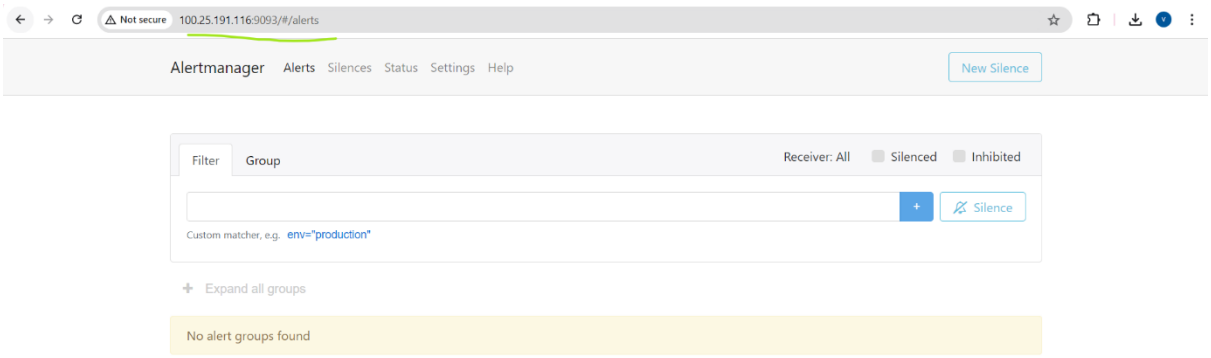
❖ **Run and start Alert Manager:**

- we need to run the below commands which will restart the daemon and start and enables the alertmanager service and will let us know the status of alert manager.

```
sudo systemctl daemon-reload

sudo systemctl start alertmanager

sudo systemctl enable alertmanager

sudo systemctl status alertmanager
```

- If our service is up and running, we can now able to access the alertmanager UI in the browser using the public IP of the server on port 9093.
- **http://<public-ip>:9093.**





❖ Configure Prometheus to Use Alertmanager:

- Now we need to configure Alertmanager with prometheus to send the alerts.
- Edit the Prometheus Configuration File **prometheus.yml**
- Add an alerting section to specify the Alertmanager instances as below.

```
alerting:
 alertmanagers:
  - static_configs:
     - targets:
        - 'localhost:9093'
```

Note:

- Replace localhost:9093 with the appropriate address if Alertmanager is running on a different machine.

- As we have changed the prometheus configuration we need to restart the prometheus using **[sudo systemctl restart prometheus command].**

- Once all the required tools installation and configuration is done check the status of each of the tool.

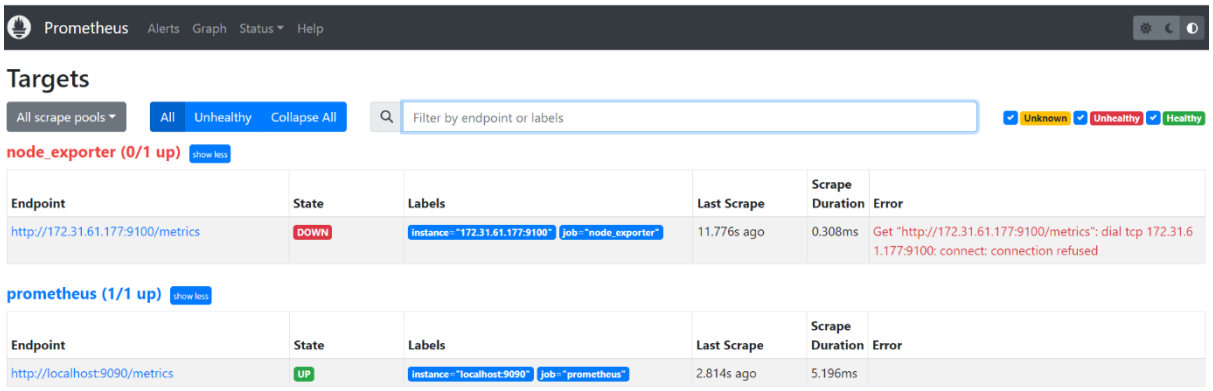
- If every tool is up and running then we are all set to monitor the resources from prometheus and grafana with alerts and notifications.

```
sudo systemctl status prometheus
sudo systemctl status grafana
sudo systemctl status node_exporter
sudo systemctl status alertmanager
```

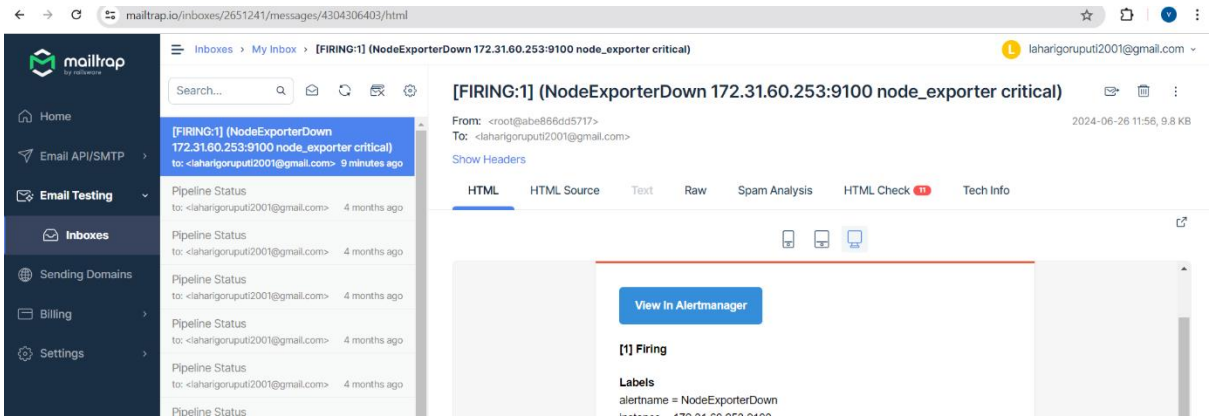
```
    - send_resolved: false
      to: laharigoruputi2001@gmail.com
      from: root@abe866dd5717
      hello: localhost
      smarthost: sandbox.smtp.mailtrap.io:2525
      auth_username: b78ccf26dfb795
      auth_password: <secret>
      headers:
        From: root@abe866dd5717
        Subject: '{{ template "email.default.subject" . }}'
        To: laharigoruputi2001@gmail.com
      html: '{{ template "email.default.html" . }}'
      require_tls: true
  templates: []
```

- After that to verify the alerts and notifications configurations we can stop the node exporter service for 1-5 minutes and then start the node exporter service again, then we can able to see the alerts in the prometheus and we will get the notifications emails also as shown in the below pics.

- **Alerts in prometheus for node stopping**



- **Notification Alerts in SMTP server mail**



- For trouble shooting use the below command **[sudo journalctl -u service-name].**

Thanks,

Lahari G