Comprehensive Production-Grade Monitoring and Logging Setup

Tools Used:

- **Prometheus (Port: 9090):** Real-time metrics collection and storage.
- **Node Exporter (Port: 9100):** Collects system metrics and exposes them to Prometheus.
- **Grafana (Port: 3000):** Visualizes metrics in customizable dashboards.
- **Alertmanager (Port: 9093):** Sends email notifications when CPU, memory, or RAM usage exceeds defined thresholds.
- **Loki (Port: 3100):** Log aggregation and storage solution.
- **Promtail (Port: 9080):** Collects and pushes application and script logs to Loki.

Install NodeExporter

```
wget https://github.com/prometheus/node_exporter/releases/download/v1.8.2/node_exporter-1.8.2.linux-amd64.tar.gz
```

1. **Extract the downloaded archive:**

   ```
   tar -xvzf node_exporter-1.8.2.linux-amd64.tar.gz
   ```

2. **Move the binary to a directory in your PATH:**

   ```
   sudo mv node_exporter-1.8.2.linux-amd64/node_exporter /usr/local/bin/
   ```

3. **Verify the installation:**

   ```
   node_exporter --version
   ```

Step 2: Run Node Exporter in the Background

To ensure that Node Exporter runs in the background and continues to send metrics to Prometheus, you can use **systemd** (if your EC2 instance uses a systemd-based OS like Ubuntu or CentOS).

1. **Create a systemd service file:**

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

2. **Add the following configuration to the service file:**

   ```
   [Unit]
   Description=Prometheus Node Exporter
   ```

```
Documentation=https://github.com/prometheus/node_exporter
After=network.target

[Service]
User=nobody
ExecStart=/usr/local/bin/node_exporter
Restart=always
RestartSec=3
LimitNOFILE=4096

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

- The ExecStart line points to the location of the Node Exporter binary.
- Restart=always ensures that Node Exporter restarts if it crashes.

3. **Reload systemd, enable and start Node Exporter service:**

```
sudo systemctl daemon-reload
sudo systemctl enable node_exporter
sudo systemctl start node_exporter
```

4. **Verify the service is running:**

```
sudo systemctl status node_exporter
```

Prometheus:

To install Prometheus using the provided link and run it in the background, follow these steps:

1. Download the Prometheus Tarball

First, download the Prometheus tarball from the specified URL:

```
cd /tmp

wget https://github.com/prometheus/prometheus/releases/download/v2.53.3/prometheus-2.53.3.linux-amd64.tar.gz
```

2. Extract the Tarball

After the tarball is downloaded, extract it:

```
tar -xvzf prometheus-2.53.3.linux-amd64.tar.gz
```

3. Move Prometheus Files

Move the extracted files to /usr/local/bin to make them globally accessible:

```
sudo mv prometheus-2.53.3.linux-amd64/prometheus /usr/local/bin/
sudo mv prometheus-2.53.3.linux-amd64/promtool /usr/local/bin/
```

4. Create Directories for Prometheus Data and Config

Create the necessary directories for Prometheus configuration and data storage:

```
sudo mkdir -p /etc/prometheus
sudo mkdir -p /var/lib/prometheus
```

5. Create a Prometheus Configuration File

Create a basic configuration file for Prometheus:

```
sudo nano /etc/prometheus/prometheus.yml
```

Add the following configuration to the file:

```
global:
  scrape_interval: 15s

rule_files:
  - /etc/prometheus/alert_rules.yml

scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']

  - job_name: 'node_exporter-jenkins'
    static_configs:
      - targets:
          - '<ec2-ip>:9100'
        labels:
          instance: 'jenkins_instance'

  - job_name: 'node_exporter-neprakabirbal'
    static_configs:
      - targets:
          - '<ec2-ip>:9100'
        labels:
          instance: 'neprakabirbal_instance'

  - job_name: 'Microservices-Envx'
    static_configs:
      - targets:
          - '10.0.4.234:9100'
        labels:
          instance: 'Microservices-Envx'
alerting:
  alertmanagers:
    - static_configs:
        - targets: ['localhost:9093']  # Replace with your Alertmanager's address
```

6. Create a Prometheus Service

Now, create a systemd service to run Prometheus in the background. Create a new file for the service:

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

Add the following content to the file:

```
[Unit]
Description=Prometheus
Documentation=https://prometheus.io/docs/introduction/overview/
After=network.target

[Service]
User=prometheus
Group=prometheus
ExecStart=/usr/local/bin/prometheus --config.file=/etc/prometheus/prometheus.yml --
storage.tsdb.path=/var/lib/prometheus/ --web.listen-address=0.0.0.0:9090

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

7. Set Permissions

Create a Prometheus user and set the appropriate permissions:

```
sudo useradd --no-create-home --shell /bin/false prometheus
sudo chown -R prometheus:prometheus /etc/prometheus /var/lib/prometheus
```

8. Reload Systemd and Start Prometheus

Reload the systemd manager to recognize the new service:

```
sudo systemctl daemon-reload
```

Now, start Prometheus as a background service:

```
sudo systemctl start prometheus
```

9. Enable Prometheus to Start on Boot

To ensure Prometheus starts automatically on system boot:

```
sudo systemctl enable prometheus
```

10. Check Prometheus Status

Verify that Prometheus is running:

```
sudo systemctl status prometheus
```

11. Access Prometheus Web Interface

Prometheus should now be running and accessible via http://<your_server_ip>:9090. You can open it in a web browser to check the interface and explore the metrics.

Alertmanager:

1. Download the Alertmanager Tarball

First, download the Alertmanager tarball from the specified URL:

```
cd /tmp
wget https://github.com/prometheus/alertmanager/releases/download/v0.27.0/alertmanager-0.27.0.linux-amd64.tar.gz
```

2. Extract the Tarball

After the tarball is downloaded, extract it:

```
tar -xvzf alertmanager-0.27.0.linux-amd64.tar.gz
```

3. Move Alertmanager Files

Move the extracted files to /usr/local/bin to make them globally accessible:

```
sudo mv alertmanager-0.27.0.linux-amd64/alertmanager /usr/local/bin/
sudo mv alertmanager-0.27.0.linux-amd64/amtool /usr/local/bin/
```

4. Create Directories for Alertmanager

Create the necessary directories for Alertmanager configuration and data storage:

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

5. Create an Alertmanager Configuration File

Create a basic configuration file for Alertmanager:

```
sudo nano /etc/alertmanager/config.yml
```

Add the following configuration to the file:

```
---
global:
  resolve_timeout: 5m
  smtp_smarthost: 'smtp.example.com:587'  # Replace with your SMTP server address and port
  smtp_from: 'alertmanager@example.com'   # Replace with the sender email address
```

```
  smtp_auth_username: 'your-username'     # Replace with your SMTP username
  smtp_auth_password: 'your-password'     # Replace with your SMTP password

route:
  group_by: ['alertname']
  group_wait: 10s
  group_interval: 5m
  repeat_interval: 3h
  receiver: 'default-receiver'

receivers:
  - name: 'default-receiver'
    email_configs:
      - to: 'madeep9347@gmail.com
        send_resolved: true
```

This configuration is a basic setup, which sends alerts to a specified email address. You can replace the email configuration with other receivers based on your setup.

6. Create an Alertmanager Service

Create a systemd service to run Alertmanager in the background. Create a new file for the service:

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

Add the following content to the file:

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

[Service]
User=alertmanager
Group=alertmanager
ExecStart=/usr/local/bin/alertmanager --config.file=/etc/alertmanager/config.yml --
storage.path=/var/lib/alertmanager/

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

7. Set Permissions

Create an Alertmanager user and set the appropriate permissions:

```
sudo useradd --no-create-home --shell /bin/false alertmanager
sudo chown -R alertmanager:alertmanager /etc/alertmanager /var/lib/alertmanager
```

8. Reload Systemd and Start Alertmanager

Reload the systemd manager to recognize the new service:

```
sudo systemctl daemon-reload
```

Now, start Alertmanager as a background service:

```
sudo systemctl start alertmanager
```

9. Enable Alertmanager to Start on Boot

To ensure Alertmanager starts automatically on system boot:

```
sudo systemctl enable alertmanager
```

10. Check Alertmanager Status

Verify that Alertmanager is running:

```
sudo systemctl status alertmanager
```

Grafana:

1. Update the package info

```
sudo apt-get install -y apt-transport-https
sudo apt-get install -y software-properties-common wget
wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add –
```

2. Add stable repository of Grafana

```
echo "deb https://packages.grafana.com/enterprise/deb stable main" | sudo tee -a
/etc/apt/sources.list.d/grafana.list
```

3. Update repository and Install Grafana

```
sudo apt-get update
sudo apt-get install grafana-enterprise
```

4. Start the Grafana Server

```
sudo systemctl daemon-reload
sudo systemctl start grafana-server
sudo systemctl status grafana-server
sudo systemctl enable grafana-server.service
```

5. Access the Grafana Dashboard using http://localhost:3000/login

---

Step 1: Install Loki

You can run **Loki** on any server (including your EC2 instance or container) to aggregate logs. We'll use Docker for simplicity, but you can also install Loki directly on the host if preferred.

Step 1.1: Pull the Loki Docker Image

1. **Pull the Loki Docker image**: Run the following command to pull the latest **Loki** image from Docker Hub:

```
docker pull grafana/loki:latest
```

Step 1.2: Create the Configuration File

2. **Create the Loki configuration file (loki-config.yaml)**:

You can use the default configuration or customize it based on your requirements. Here's a basic configuration to get you started:

```
auth_enabled: false

server:
  http_listen_port: 3100
  grpc_listen_port: 9095

ingester:
  chunk_target_size: 1048576

storage_config:
  boltdb_shipper:
    active_index_directory: /tmp/loki/index
    cache_location: /tmp/loki/cache
    shared_store: filesystem

compactor:
  working_directory: /tmp/loki/compactor
  retention_enabled: true
  retention_delete_delay: 24h
  retention_period: 4d  # Logs older than 4 days will be deleted automatically

limits_config:
  max_entries_limit: 500000
```

Save this file as loki-config.yaml in the directory where you will run the **Loki** container. This configuration file tells **Loki** where to store the logs and how to manage the data.

Step 1.3: Run Loki in Docker

3. **Run the Loki Docker container**:

Run **Loki** in a Docker container using the following command, assuming the loki-config.yaml is in the current directory:

```
docker run -d --name=loki -p 3100:3100 \
  -v $(pwd)/loki-config.yaml:/etc/loki/loki-config.yaml \
```

```
grafana/loki:latest
```

---

1. Install Promtail

Step 1.1: Download Promtail

First, you need to download the appropriate version of **Promtail** for your EC2 instance.

1. **Download the Promtail tarball**:

```
wget https://github.com/grafana/loki/releases/download/v2.9.11/promtail-linux-amd64.zip
```

Step 1.2: Extract the Tarball

2. **Extract the downloaded archive**:

```
unzip promtail-linux-amd64.zip
```

Step 1.3: Move the Binary to a Directory in Your PATH

3. **Move the Promtail binary to /usr/local/bin/**:

```
sudo mv promtail-linux-amd64 /usr/local/bin/promtail
```

Step 1.4: Verify the Installation

4. **Verify the installation**:

```
promtail –version
```

You should see the version of **Promtail** you just installed.

---

2. Create Systemd Service for Promtail

Now, let's set up **Promtail** to run as a service using **systemd**, similar to how you did with **Node Exporter**. This ensures that **Promtail** starts automatically when the system boots.

Step 2.1: Create the systemd Service File

1. **Create a new service file for Promtail**:

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

2. **Add the following configuration to the service file**:

```
[Unit]
Description=Promtail
Documentation=https://github.com/grafana/loki
After=network.target
```

```
[Service]
User=nobody
ExecStart=/usr/local/bin/promtail -config.file=/etc/promtail/promtail.yaml
Restart=always
RestartSec=3
LimitNOFILE=4096
[Install]
WantedBy=multi-user.target
```

- The ExecStart line points to the location of the **Promtail** binary and the **promtail.yaml** configuration file.
- Restart=always ensures **Promtail** restarts if it crashes.

Step 2.2: Create the Promtail Configuration File

1. **Create a configuration file for Promtail**:

```
mkdir -p /etc/promtail
sudo nano /etc/promtail/promtail.yaml
```

2. **Add the following basic configuration to promtail.yaml**: Make sure to replace <Loki-Server-IP> with the actual IP address of your **Loki** server.

```
server:
  http_listen_port: 9080
  grpc_listen_port: 0

positions:
  filename: /tmp/positions.yaml

clients:
  - url: http://<loki-ip>:3100/api/prom/push

scrape_configs:
  - job_name: system
    static_configs:
      - targets:
          - localhost
        labels:
          job: varlogs-jenkins
          instance: "jenkins"
          __path__: /var/log/*log

  - job_name: cpu_logs
    static_configs:
      - targets:
          - localhost
        labels:
          job: cpu_logs-jenkins-instance
          __path__: /var/log/cpu_usage.log
```

```
  pipeline_stages:
   - json:
       expressions:
         pid: pid
         ppid: ppid
         user: user
         cpu: cpu
         mem: mem
         cmd: cmd
```

- **url**: The URL of your **Loki** server.
- **__path__**: The path where the log files are stored. This is set to collect all logs in /var/log/*log. You can adjust this to suit your needs (e.g., specific application logs or other log files).

```
sudo usermod -aG adm nobody
cd /var/log/
sudo touch cpu_usage.log
sudo chown $USER:$USER /var/log/cpu_usage.log
```

Step 2.3: Reload systemd, Enable, and Start Promtail Service

1. **Reload systemd** to apply the changes:

```
sudo systemctl daemon-reload
```

2. **Enable Promtail to start at boot**:

```
sudo systemctl enable promtail
```

3. **Start the Promtail service**:

```
sudo systemctl start promtail
```

Step 2.4: Verify the Promtail Service is Running

1. **Check the status of Promtail**:

```
sudo systemctl status promtail
```

You should see output indicating that **Promtail** is running.

---

3. Verify Promtail is Sending Logs to Loki

After configuring **Promtail** and starting the service, you can verify that it is properly sending logs to **Loki**.

Step 3.1: Check Promtail Logs

To ensure **Promtail** is correctly sending logs to **Loki**, check the logs for **Promtail**:

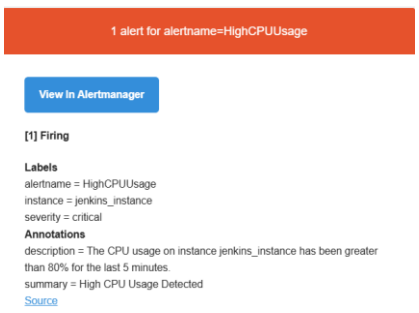
```
journalctl -u promtail -f
```

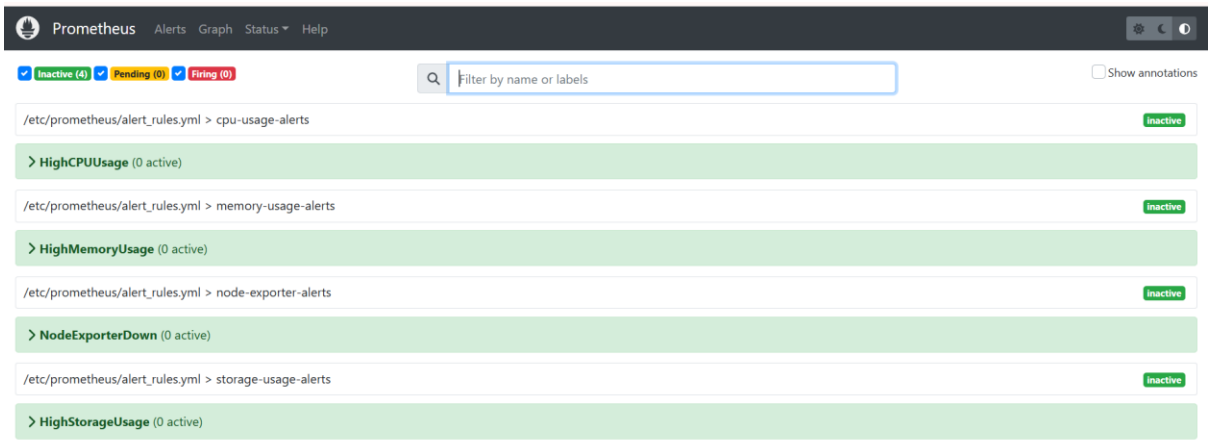
This Script is used for continuously send the CPU, RAM usage logs to loki
cpu_logger.sh

```
cd /var/log/
#!/bin/bash

# Define the log file path
LOG_FILE="/var/log/cpu_usage.log"

# Set the timezone to Indian Standard Time (IST)
export TZ='Asia/Kolkata'

# Create or clear the log file
> "$LOG_FILE"

# Run indefinitely
while true; do
   # Log the timestamp and log header (only once for each iteration)
   echo "$(date +'%Y-%m-%dT%H:%M:%S') | [CPU_LOG]" >> "$LOG_FILE"

   # Get the top 15 processes by CPU usage, formatted as JSON (no timestamp included here)
   ps -eo pid,ppid,%cpu,%mem,cmd --sort=-%cpu | head -n 16 | tail -n +2 | awk '{
      printf "{\"pid\":\"%s\",\"ppid\":\"%s\",\"cpu\":\"%s\",\"mem\":\"%s\",\"cmd\":\"%s\"}\n",
      $1, $2, $3, $4, $5
   }' >> "$LOG_FILE"

   # Add a separator line for readability
   echo "----" >> "$LOG_FILE"

   # Sleep for 30 seconds before the next iteration
   sleep 30
done
```

```
chmod +x /home/ubuntu/cpu_logger.sh
```

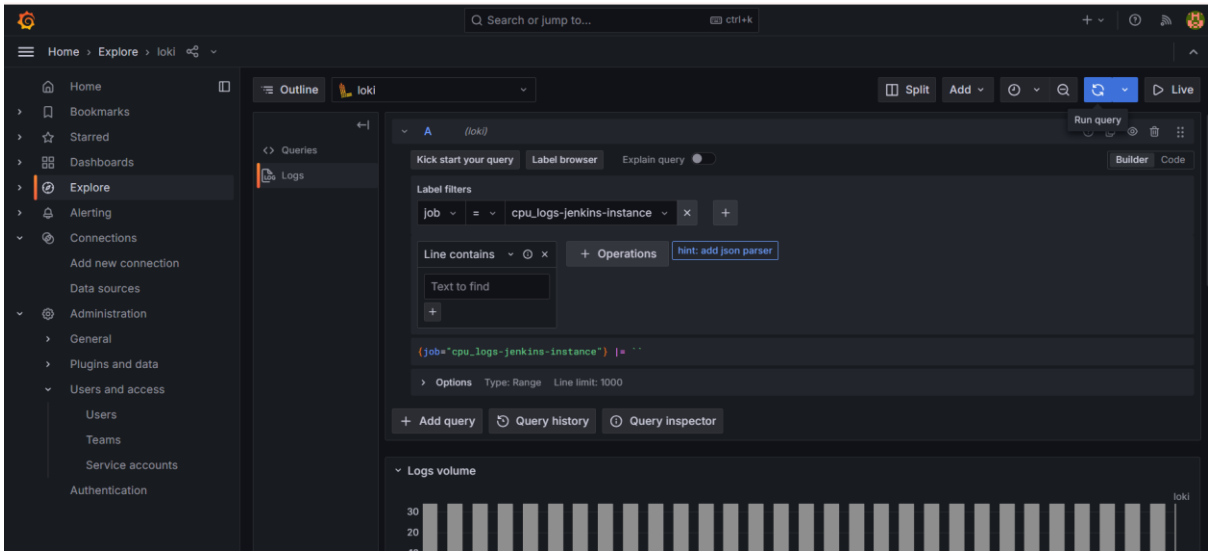
```
nohup /home/ubuntu/cpu_logger.sh &
```

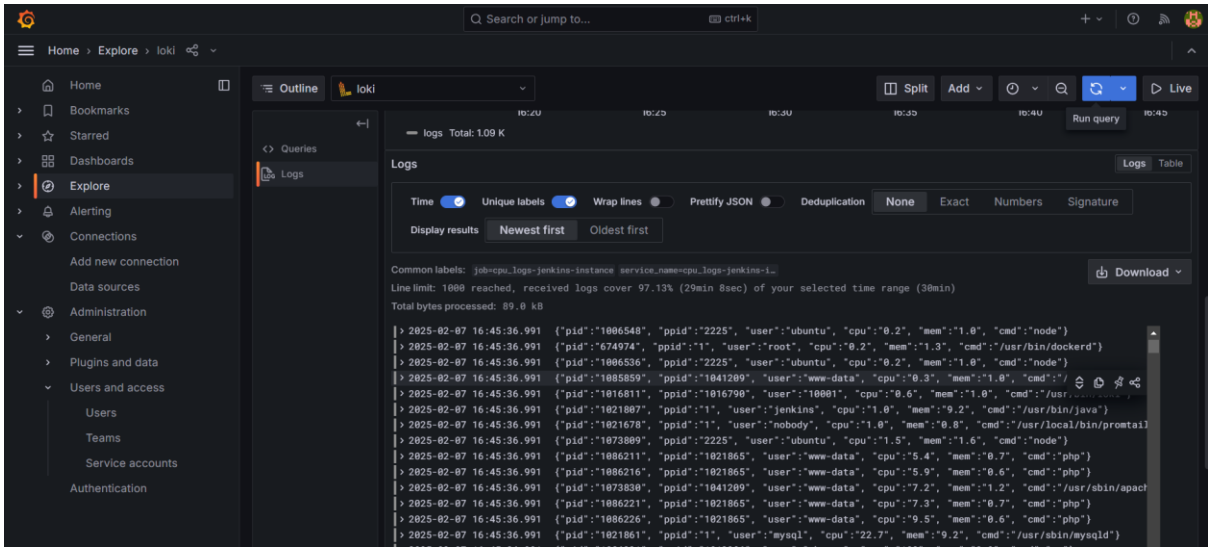




in Grafana metrics it showing the cpu increased to 59%

Check the CPU logs:





Check the application logs