Lesson 02 Demo 03

Writing Advanced Queries for Real-Life Use Cases

Objective: To write complex PromQL queries for detecting performance issues, monitoring resource utilization, and setting up alerts

Tools required: Linux operating system

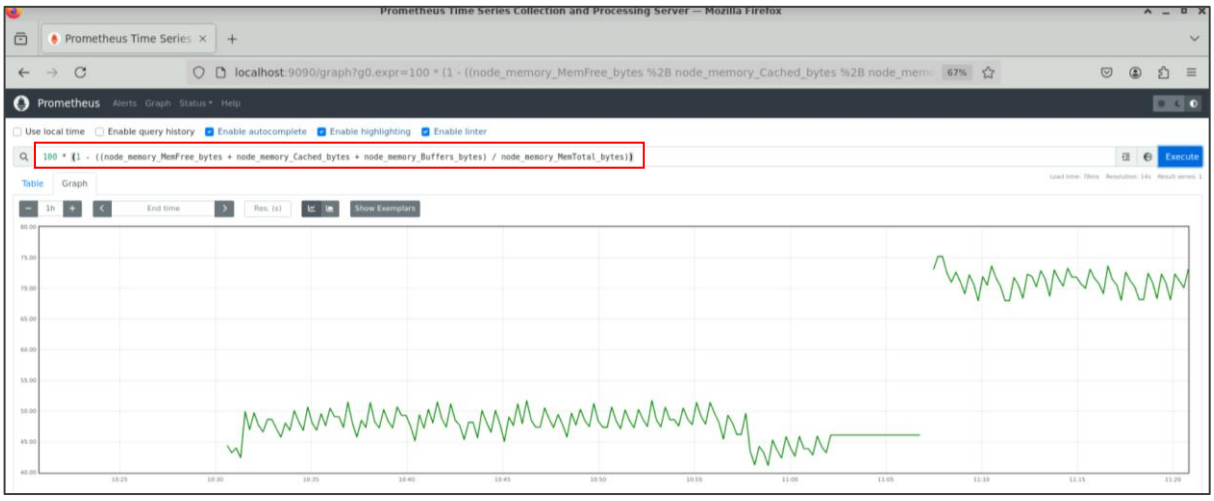
Prerequisites: Refer to Demo 02 of Lesson 02 for understanding basic PromQL queries

Steps to be followed:

1. Set alerts for high memory usage
2. Analyze disk I/O patterns for performance issues
3. Track node uptime for maintenance alerts
4. Monitor network traffic patterns to identify bottlenecks

Step 1: Set alerts for high memory usage

1.1 Execute the following query in the expression browser to identify nodes with high memory utilization:
100 * (1 - ((node_memory_MemFree_bytes + node_memory_Cached_bytes + node_memory_Buffers_bytes) / node_memory_MemTotal_bytes))



Note: This query calculates the memory utilization percentage for each node by subtracting the ratio of available memory from the total memory. An alert rule can also be set to notify when memory utilization exceeds a defined threshold.

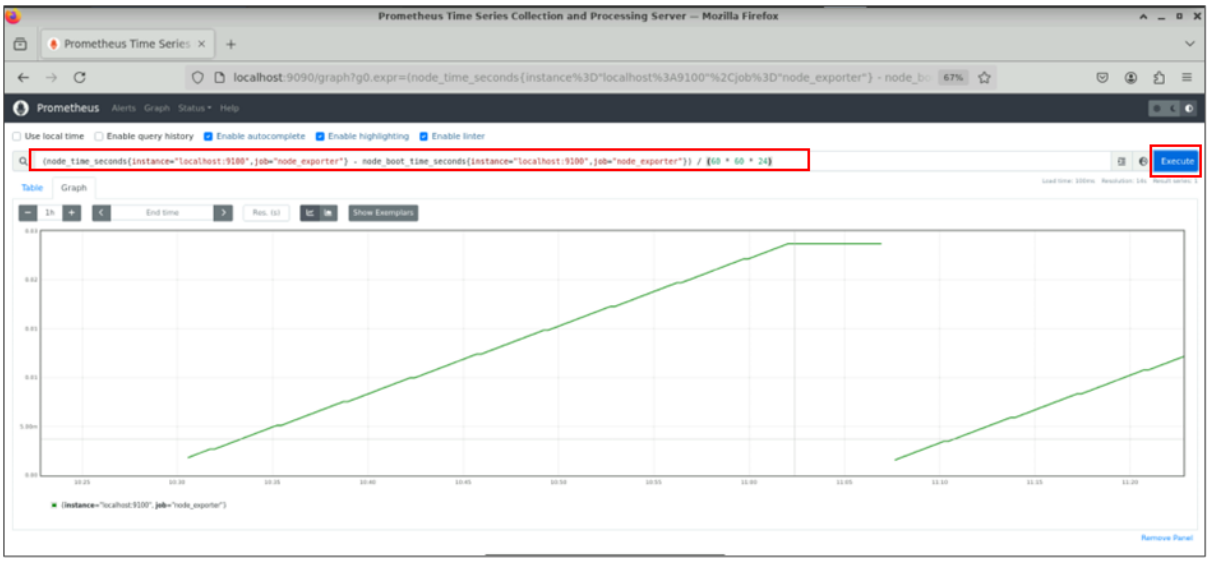
Step 2: Analyze disk I/O patterns for performance issues

2.1 Execute the following query to analyze the disk I/O patterns across the infrastructure:
sum by(instance) (rate(node_disk_read_bytes_total[5m]) + rate(node_disk_written_bytes_total[5m]))

Note: This query sums up the per-second disk read and write rates for each instance over the last 5 minutes, grouped by instance label. It helps identify unusual spikes or trends in disk I/O activity across nodes or the infrastructure.

Step 3: Track node uptime for maintenance alerts

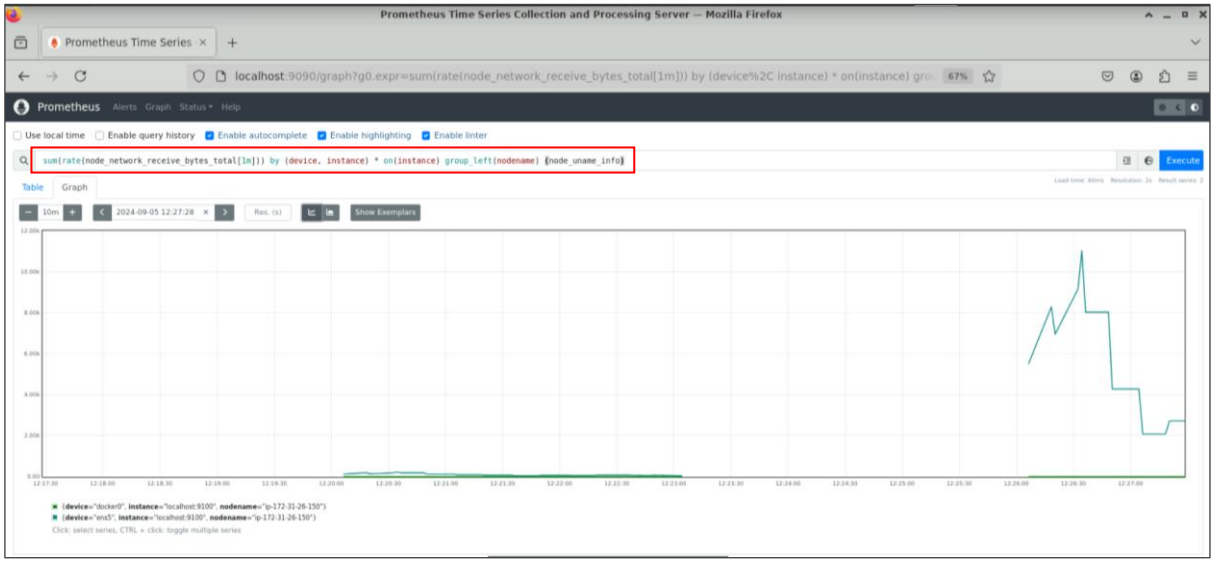
3.1 In the expression browser, enter the following query and click on **Execute** to calculate node uptime:
(node_time_seconds{instance="localhost:9100",job="node_exporter"} - node_boot_time_seconds{instance="localhost:9100",job="node_exporter"}) / (60 * 60 * 24)



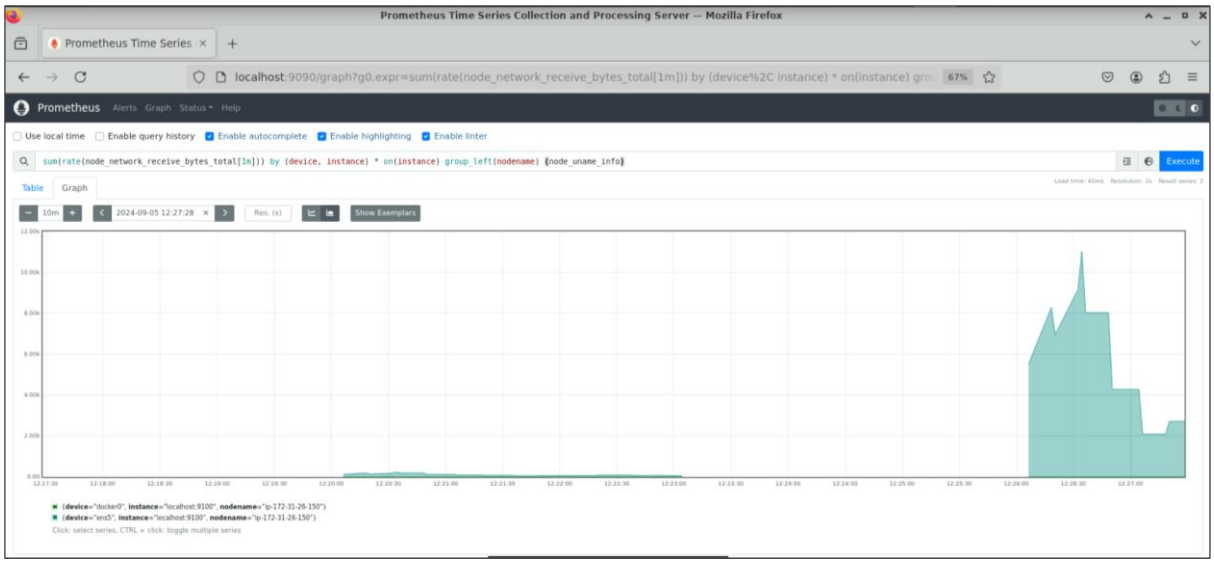
Note: This query divides the result of **node_time_seconds - node_boot_time_seconds** by **60 * 60 * 24** to convert the uptime from seconds to days. It can also be used to set up alerting mechanisms to notify when a node has been running for an extended period without a reboot.

Step 4: Monitor network traffic patterns to identify bottlenecks

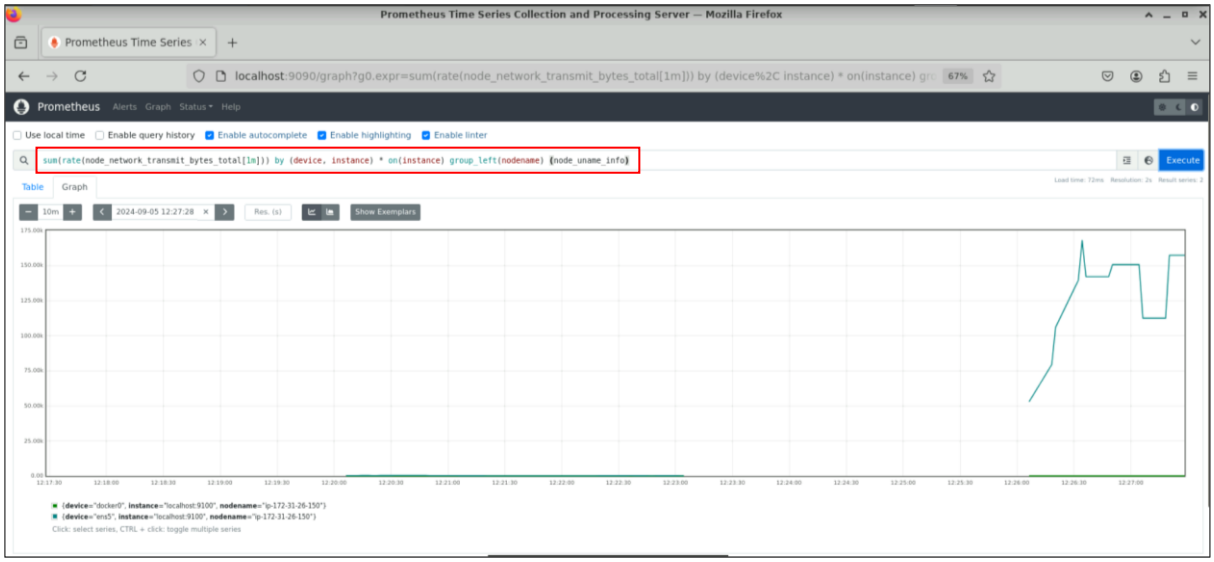
4.1 Execute the following query to calculate the per-second network receive rate for each device and instance:
sum(rate(node_network_receive_bytes_total[1m])) by (device, instance) * on(instance) group_left(nodename) (node_uname_info)



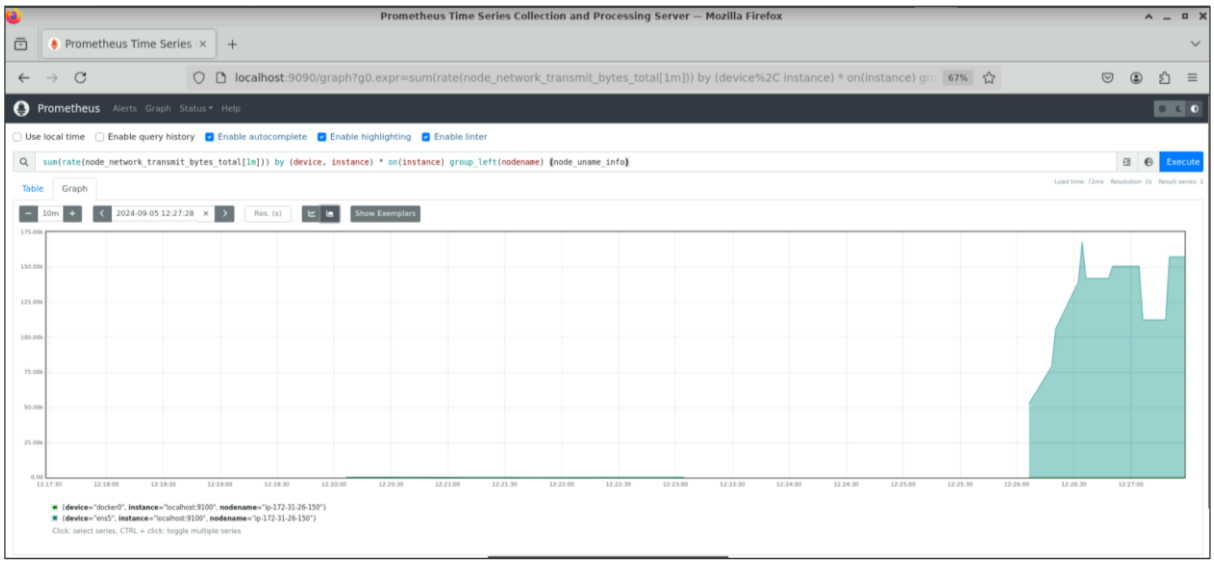
The stacked graph for this query is then displayed as shown below:



4.2 Run the following query to calculate the per-second network transmit rate for each device and instance:
sum(rate(node_network_transmit_bytes_total[1m])) by (device, instance) * on(instance) group_left(nodename) (node_uname_info)



The stacked graph for this query is then displayed as shown below:



Note: These queries monitor network traffic patterns across nodes to identify bottlenecks, plan capacity upgrades, and troubleshoot issues. It also helps analyze receive and transmit rates for each network device, including node names for easy identification.

By following these steps, you have efficiently monitored resource utilization, identified performance issues, and set up alerts to ensure smooth and proactive system operations.