Visual Analytics and User Experience Design(IT4031)

Project ID: 2023-A2-G35

Sri Lanka Institute of Information Technology4th Year 1st Semester

Assignment 2

Submitted by: IT20033828 – P.A.D.N.I. Ariyasinghe

Table of Contents

Table of Figures

List of Tables

Project Links

Below links can be used to access the project implementation of the group:

- Node exporter metrics - http://192.168.8.179:9100/metrics
- Prometheus - http://192.168.8.179:9090/targets?search=
- Grafana - http://192.168.8.179:3000/
- Prometheus_Node_MYSQL_MainDashboard - http://192.168.8.179:3000/d/e38d7c7a-afc4-459e-833e-e73bd5f6d5d2/prometheus?orgId=1&from=1684472675903&to=1684476275905

  Username : admin

  Password : vauedgdb

- Node_Expoter_Full_AdditionalDashboard - http://192.168.8.179:3000/d/rYdddlPWk/node-exporter-full?orgId=1&from=1684390188900&to=1684476588911
- Alerts - http://192.168.8.179:9090/alerts?search=

1. Architecture Diagram

- **Objective -** Implement a dashboard using Grafana to visualize the metrics of MySQLapplication, captured from Prometheus:
- **Deployment** – Virtual Machine
- **Application -** MYSQL
- **Language Used -** PROMQL
- **Exporters –** Node Exporter, MySQL Exporter

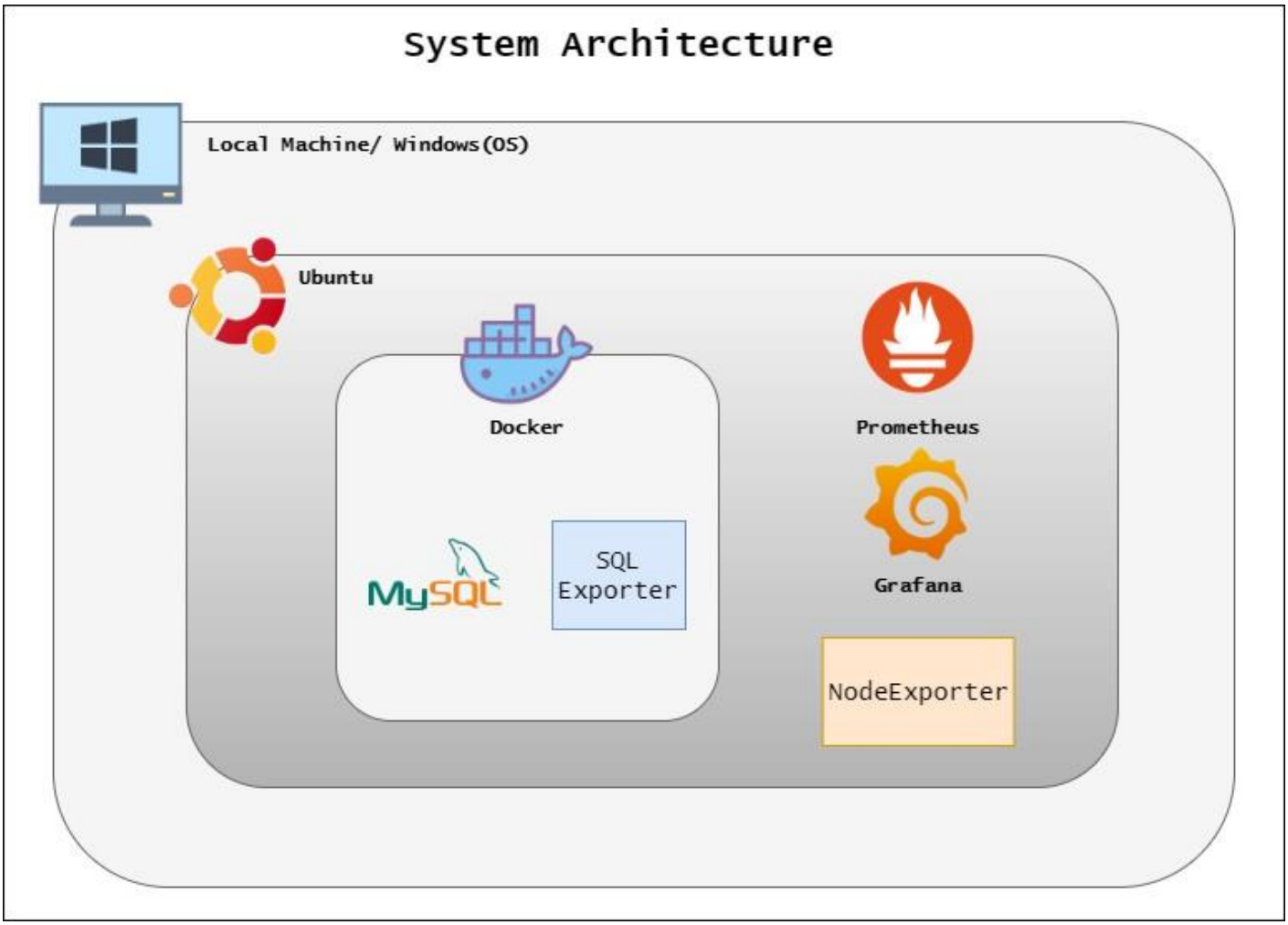


Figure 1- System Architecture

2. Virtual Machine Configuration

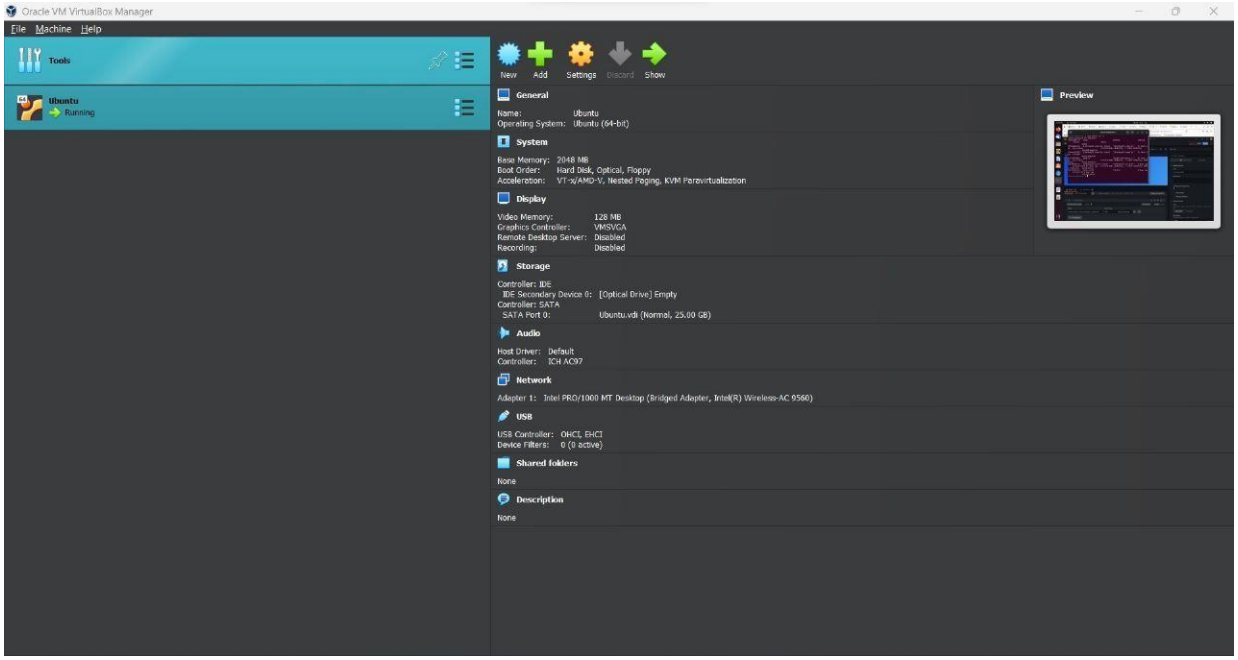


Figure 2- Adding Virtual Machine to Virtual Box in the Local Machine

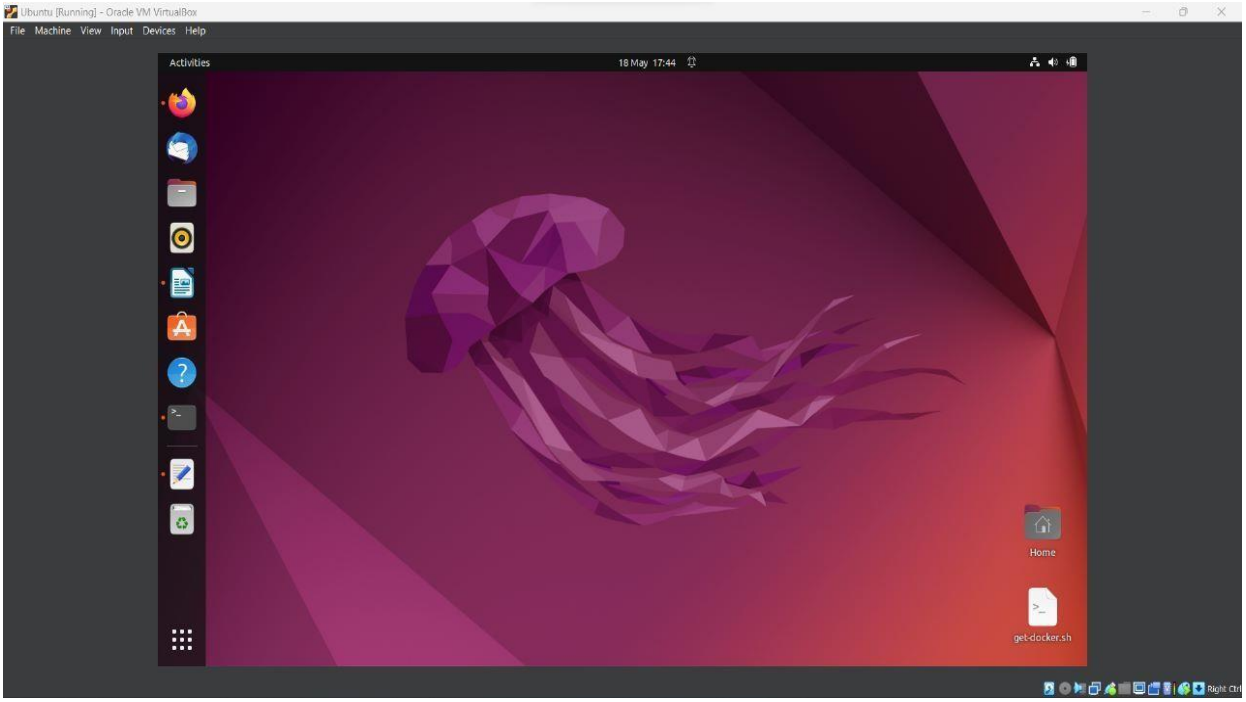


Figure 3- After configuring the Virtual Machine

2.1 Ubuntu Image Configuration

An Ubuntu Image was installed and configured on EC2 instance in order to deploy the applications in cloud environment.

3. Deploy API to Fetch Metrics

3.1 Install Docker on Ubuntu

Docker was installed on Ubuntu and the steps are as below.

- Update ***package***
  ***sudo apt-get update sudo apt-get install \ ca-certificates ***
  curl \ gnupg \ lsb-release

- Add Docker's official GPG key:
  curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

- Set up the stable repository
  ***echo ***
  "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive- keyring.gpg]
  ***https://download.docker.com/linux/ubuntu ***
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

- Install Docker Engine
  sudo apt-get update
  sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin

- Testing
  sudo docker run hello-world

3.2 Install MySQL on Docker

MySQL has installed on Docker by using below steps.

- Create a docker network

  sudo docker network create db_network

- Pull MySQL 8 Image and run

  ***sudo docker run -d ***
  ***--name mysql_instance ***
  ***--publish 3306 ***
  ***--network db_network ***
  ***--restart unless-stopped ***
  ***--env MYSQL_ROOT_PASSWORD=mypassword ***
  ***--volume mysql80-datadir:/var/lib/mysql \ mysql:8 ***
  --default-authentication-plugin=mysql_native_password

- Verify if our MySQL server is running Okay

  sudo docker ps | grep mysql

- Create monitoring User

  sudo docker exec -it mysql80 mysql -uroot -p
  On MYSQL >

CREATE USER 'exporter'@'%' IDENTIFIED BY 'exporterpassword' WITH MAX_USER_CONNECTIONS 3;CREATE USER 'exporter'@'%' IDENTIFIED BY 'exporterpassword' WITH MAX_USER_CONNECTIONS 3;

- Return to shell using exit command

3.3 Setup MySQL Exporter

Configured MySQL Exporter container on Linux shell.

***sudo docker run -d ***

***--name mysql80-exporter ***

***--publish 9104 ***

***--network db_network ***

***--restart always ***

--env

***DATA_SOURCE_NAME="exporter:exporterpassword@(mysql_instance:3306)/" ***

***prom/mysqld-exporter:latest ***

***--collect.info_schema.processlist ***

***--collect.info_schema.innodb_metrics ***

***--collect.info_schema.tablestats ***

***--collect.info_schema.tables ***

***--collect.info_schema.userstats ***

--collect.engine_innodb_status

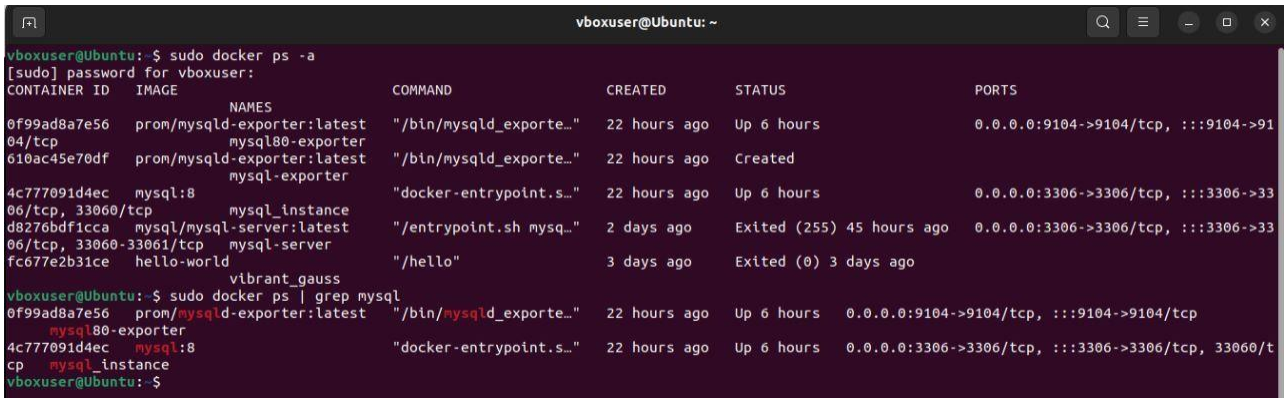


Figure 4- After configuring MySQL image and exporter

3.4 Install Node Exporter on Ubuntu

Node Exporter was installed on Ubuntu as a service in order to monitor metrics of the machine.

- Use wget command to download binary.
  wget https://github.com/prometheus/node_exporter/releases/download/v1.3.1/node_exporter-1.3.1.linux-amd64.tar.gz

- Extract node exporter from the archive.
  tar -xvf node_exporter-1.3.1.linux-amd64.tar.gz

- Move binary to the /usr/local/bin
  ***sudo mv ***
  ***node_exporter-1.3.1.linux-amd64/node_exporter ***
  /usr/local/bin/

- Clean up, delete node_exporter archive and a folder.
  rm -rf node_exporter*

- Verify that you can run the binary
  node_exporter –version

- Node Exporter has a lot of plugins that we can enable. If you run Node Exporter help you will get all the options.
  node_exporter –help

- Next, create similar systemd unit file.
  sudo vim /etc/systemd/system/node_exporter.service

- **node_exporter.service**
  [Unit]
  Description=Node Exporter
  Wants=network-online.target
  After=network-online.target

  StartLimitIntervalSec=500
  StartLimitBurst=5

  [Service]
  User=node_exporter
  Group=node_exporter
  Type=simple
  Restart=on-failure
  RestartSec=5s
  ***ExecStart=/usr/local/bin/node_exporter ***
  --collector.logind

  [Install]
  WantedBy=multi-user.target

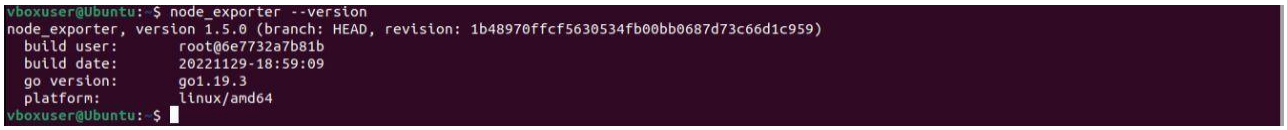


Figure 5- After configuring node exporter

- To automatically start the Node Exporter after reboot, enable the service

  sudo systemctl enable node_exporter

- Then start the Node Exporter

  sudo systemctl start node_exporter

- Check the status of Node Exporter with the following command:

  sudo systemctl status node_exporter

4. Configure Prometheus

4.1 Installing Prometheus

Installed Prometheus on Ubuntu and following are the steps used for the configuration.

- curl or wget command to download Prometheus.
  wget https://github.com/prometheus/prometheus/releases/download/v2.32.1/prometheus-2.32.1.linux-amd64.tar.gz

- Then, we need to extract all Prometheus files from the archive.
  tar -xvf prometheus-2.32.1.linux-amd64.tar.gz

- Usually, you would have a disk mounted to the data directory. For this tutorial, I will simply create a /data director. Also, you need a folder for Prometheus configuration files.
  sudo mkdir -p /data /etc/Prometheus

- Now, let's change the directory to Prometheus and move some files.
  cd prometheus-2.32.1.linux-amd64

- promtool is used to check configuration files and Prometheus rules.
  sudo mv prometheus promtool /usr/local/bin/

- Finally, let's move the example of the main Prometheus configuration file.
  sudo mv prometheus.yml /etc/prometheus/prometheus.yml

- Verify that you can execute the Prometheus binary by running the following command.
  prometheus –version

- **prometheus.service**

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

StartLimitIntervalSec=50
0StartLimitBurst=5

[Service]
User=prometheus
Group=prometheus
Type=simple
Restart=on-failure
RestartSec=5s
ExecStart=/usr/local/bin/prometheus \
  --config.file=/etc/prometheus/prometheus.yml \
  --storage.tsdb.path=/data \
  --web.console.templates=/etc/prometheus/consoles \
  --web.console.libraries=/etc/prometheus/console_libraries \
  --web.listen-address=0.0.0.0:9090 \
```

--web.enable-lifecycle

[Install]
WantedBy=multi-user.target

- To automatically start the Prometheus after reboot, run enable
  sudo systemctl enable prometheus

- Then just start the Prometheus
  sudo systemctl start prometheus

- To check the status of Prometheus run following command:
  sudo systemctl status prometheus

4.2 Setting the Targets in Prometheus

Target ports have configured in .yml file as follows.

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


# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets:
          - localhost:9093


# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  - dead-mans-snitch-rule.yml
  - target-down.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"
    static_configs:
      - targets: ["localhost:9090"]

  - job_name: "node_export"
    static_configs:
      - targets: ["localhost:9100"]

  - job_name: "mysql_instance"
    static_configs:
      - targets: ["localhost:9104"]
```

Figure 6- Prometheus.yaml file

After setting the targets, we can see all the targets and metrics being monitored by Prometheus.

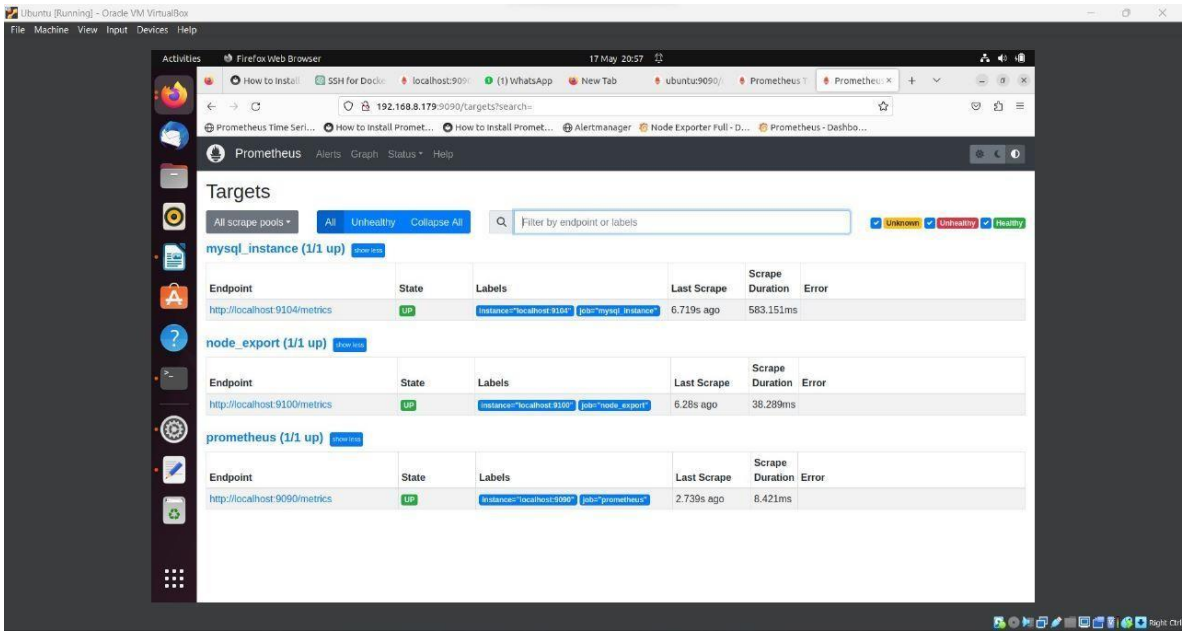


Figure 7- Prometheus Target Page

4.3 Retrieving Metrics to Prometheus

Metrics can be retrieved after clicking the endpoint link as below.

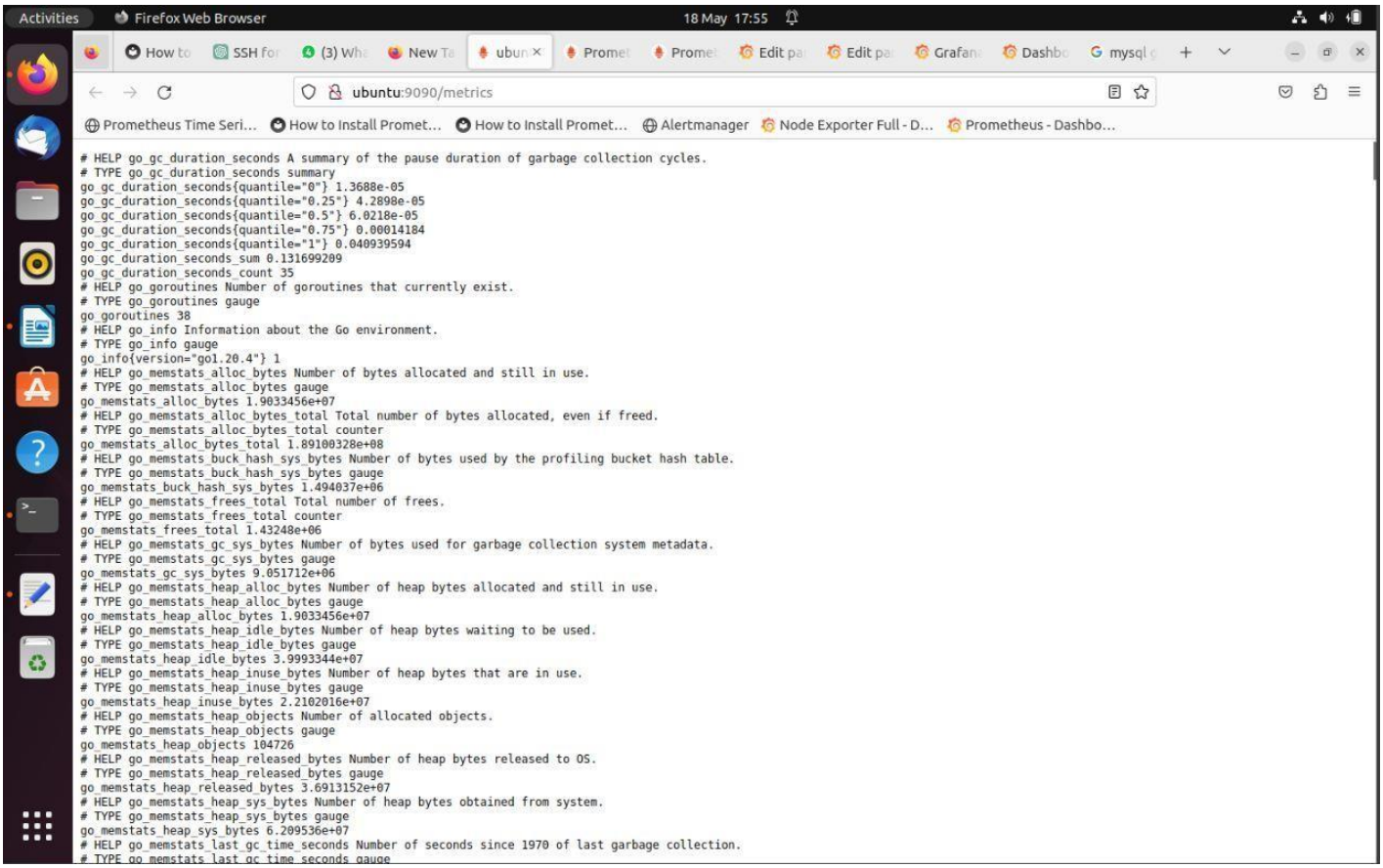


Figure 8- Prometheus Metrics

5. Grafana Dashboard

5.1 Install and Configure Grafana on Ubuntu

Grafana was installed on Ubuntu using the following commands.

- First, let's make sure that all the dependencies are installed.
  sudo apt-get install -y apt-transport-https software-properties-common

- Next, add GPG key.
  wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add –

- Add this repository for stable releases.
  echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list

- After you add the repository, update, and install Grafana.
  sudo apt-get update
  sudo apt-get -y install Grafana

- To automatically start the Grafana after reboot, enable the service.
  sudo systemctl enable grafana-server

- Then start the Grafana.
  sudo systemctl start grafana-server

- To check the status of Grafana, run the following command:
  sudo systemctl status grafana-server

5.2 Create Prometheus as the Data Source

Prometheus data source was created as below in order to retrieve metrics from Prometheus to Grafana.

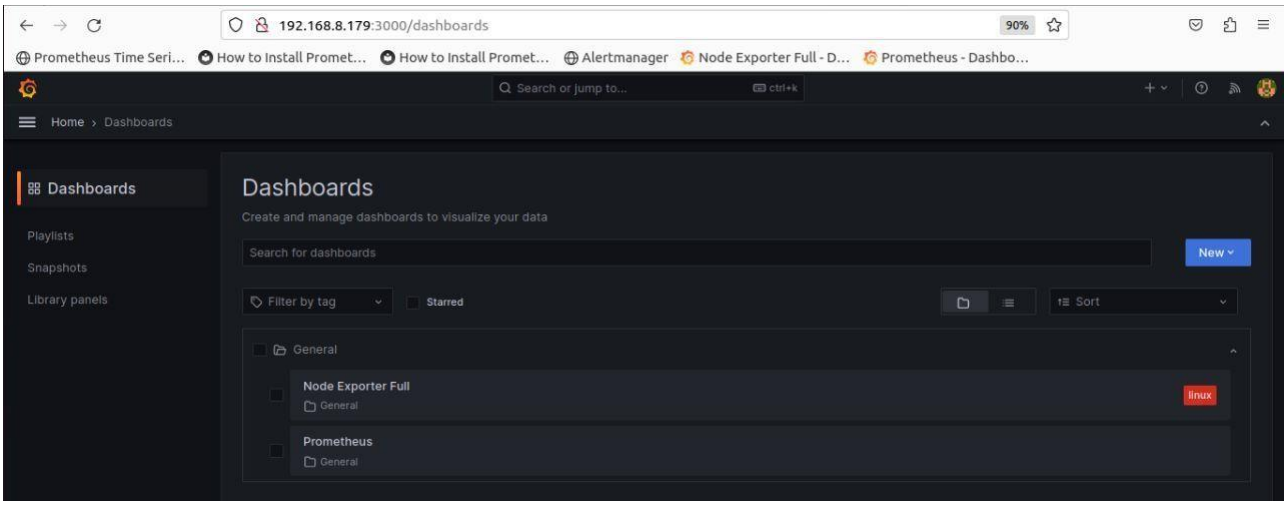


Figure 9- Grafana Dashboard Page

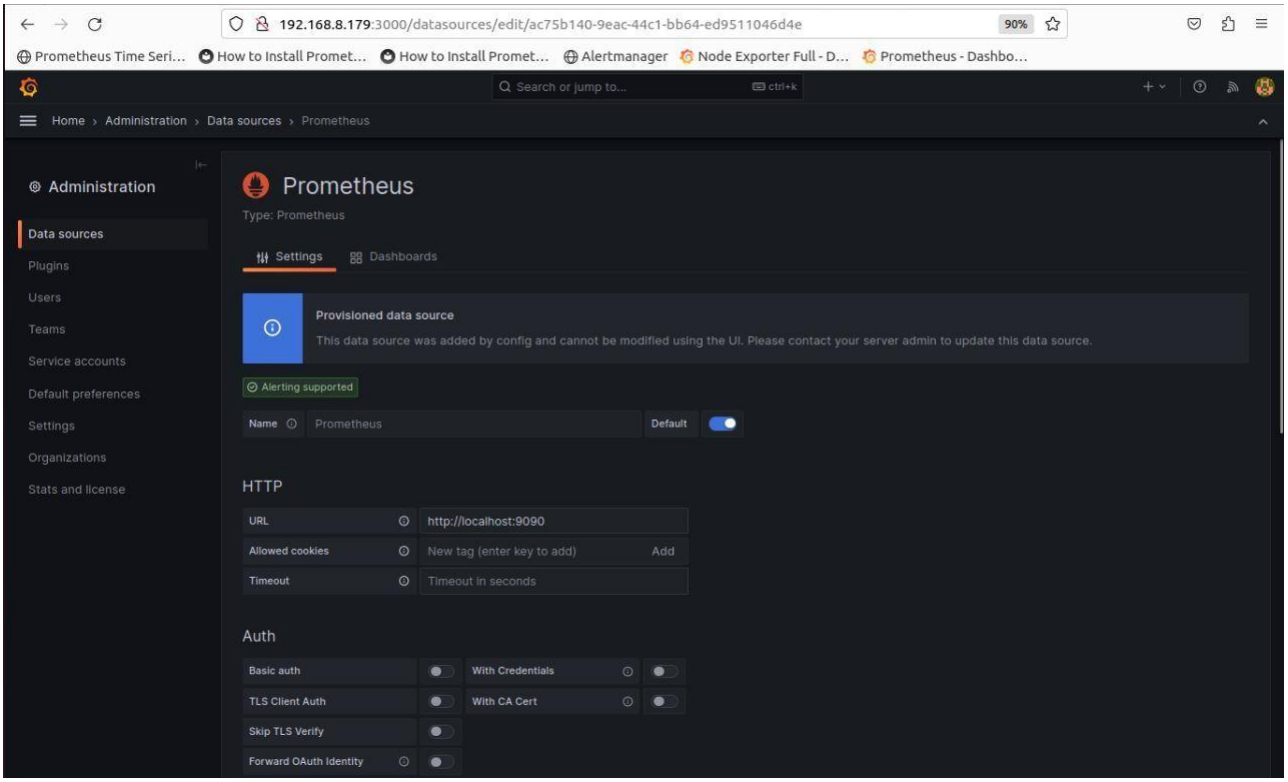


Figure 10- Setting Prometheus as a Datasource

6. Install Alert manager on Ubuntu

- Download Alertmanager from the same downloads page.
  wget https://github.com/prometheus/alertmanager/releases/download/v0.23.0/alertmanager-0.23.0.linux-amd64.tar.gz

- Extract Alertmanager binary.
  tar -xvf alertmanager-0.23.0.linux-amd64.tar.gz

- For Alertmanager, we need storage. It is mandatory (it defaults to "data/") and is used to store Alertmanager's notification states and silences. Without this state (or if you wipe it), Alertmanager would not know across restarts what silences were created or what notifications were already sent.
  sudo mkdir -p /alertmanager-data /etc/alertmanager

- Now, let's move Alermanager's binary to the local bin and copy sample config.
  sudo mv alertmanager-0.23.0.linux-amd64/alertmanager /usr/local/bin/
  sudo mv alertmanager-0.23.0.linux-amd64/alertmanager.yml /etc/alertmanager/

- Remove downloaded archive and a folder.
  rm -rf alertmanager*

- Check if we can run Alertmanager.
  alertmanager –version

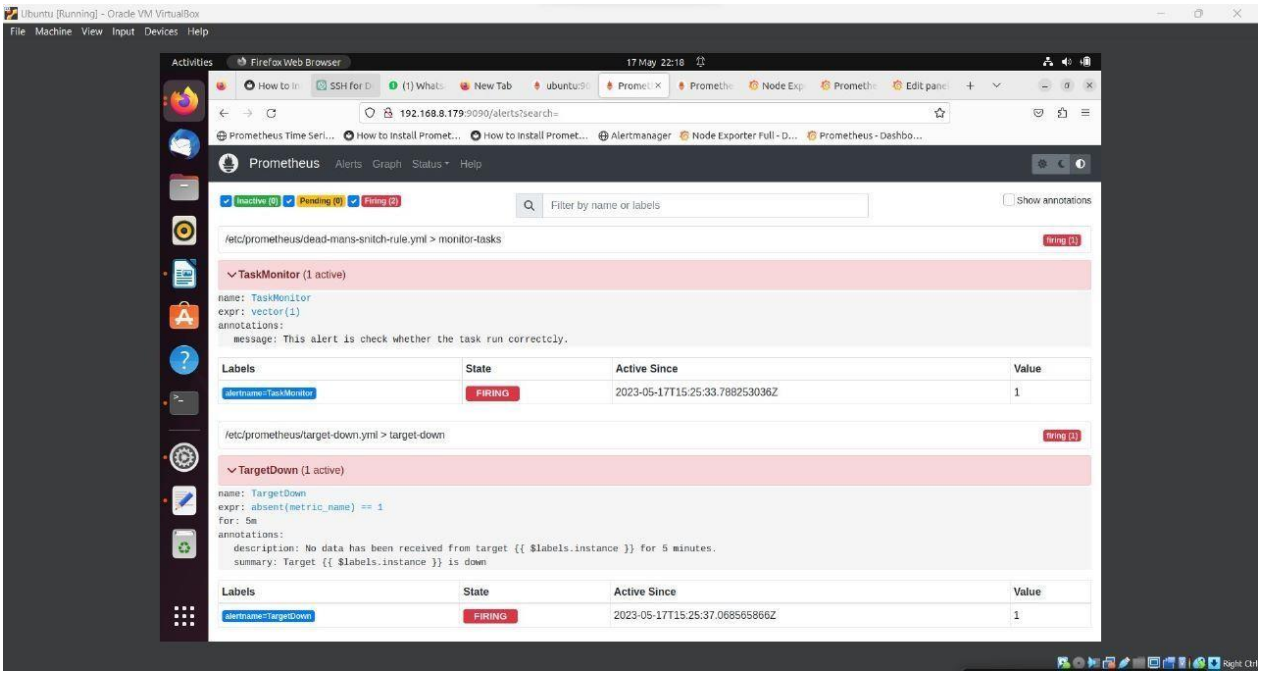


Figure 11- Alerts page in Prometheus 1

6.1 Used Metrics

Table 1– MySQL Metrics

MySQL Metrics	
Metrics Used & PromQL Queries	**Justification**
mysql_global_status_queries	
mysql_global_status_queries{job="mysql_instance"}	Check Queries Per Second(QPS).
mysql_global_status_uptime	
mysql_global_status_uptime{job="mysql_instance"}	Check the uptime for MySQL in your server via Seconds since system boot(server has been up)
mysql_global_variables_innodb_buffer_pool_size	
mysql_global_status_innodb_buffer_pool_bytes_data{job="mysql_instance"}	MySQL configuration parameter that specifies the amount of memory allocated to the InnoDB buffer pool by MySQL. This is the MySQL hosting configuration and should be configured based on the available system RAM.
mysql_global_status_threads_connected	
mysql_global_status_threads_connected{job="mysql_instance"}	The number of currently open connections.
mysql_global_variables_max_connections	
mysql_global_variables_max_connections{job="mysql_instance"}	Maximum number of connections configured.
mysql_global_status_max_used_connections	
mysql_global_status_max_used_connections{job="mysql_instance"}	Maximum Used Connections.
mysql_global_status_bytes_received	
mysql_global_status_bytes_received{job="mysql_instance"}	The number of bytes received from all clients.
mysql_global_status_bytes_sent	
mysql_global_status_bytes_sent{job="mysql_instance"}	The number of bytes sent to all clients.
mysql_global_status_table_locks_immediate	
mysql_global_status_table_locks_immediate{job="mysql_instance"}	Represents the total number of row locks.
mysql_global_status_table_locks_waited	

mysql_global_status_table_locks_waited{job="mysql_instance"}	Number of table locks.
mysql_global_variables_key_buffer_size	
mysql_global_variables_key_buffer_size{job="mysql_instance"}	Key Buffer Size.
mysql_global_variables_query_cache_size	
mysql_global_variables_have_query_cache{job="mysql_instance"}	Query Cache Size.
mysql_global_variables_innodb_log_buffer_size	
mysql_global_variables_innodb_log_buffer_size{job="mysql_instance"}	InnoDB Log Buffer Size
mysql_global_status_innodb_mem_adaptive_hash	
mysql_global_status_innodb_mem_adaptive_hash{job="mysql_instance"}	Adaptive Hash Index Size.

Table 2– Prometheus Metrics

Prometheus Metrics	
Metrics Used & PromQL Queries	**Justification**
scrape_duration_seconds	
scrape_duration_seconds{job="prometheus"}	Duration of the scrape in seconds.
prometheus_tsdb_head_series	
prometheus_tsdb_head_series{job="prometheus"}	Covers every series that has existed in the last 1 hour.
prometheus_tsdb_head_chunks	
prometheus_tsdb_head_chunks{job="prometheus"}	Total number of chunks in the head block.
prometheus_engine_query_duration_seconds_sum	
prometheus_engine_query_duration_seconds_sum{job="prometheus"}	The sum of the duration of all engine query processes.

Table 3– Node Metrics

Node Metrics	
Metrics Used & PromQL Queries	**Justification**
node_cpu_seconds_total	
#CPU Busy((count(count(node_cpu_seconds_total{job="node_e xport"}) by (cpu)) - avg(sum by (mode)(rate(node_cpu_seconds_total{job="node_ export", mode="idle"}[5m])))) * 100) / count(count(node_cpu_seconds_total{job="node_export"}) by (cpu))	This is a counter metric that counts. The number of seconds the CPU has been running.
count(count(node_cpu_seconds_total{job="node _export"}) by (cpu))	Number of CPU cores.
node_time_seconds	
node_time_seconds{job="node_export"} - node_boot_time_seconds{job="node_export"}	System up time in seconds.
node_memory_MemAvailable_bytes	
# RAM Used100 - ((node_memory_MemAvailable_bytes{job="node_export"} * 100) /node_memory_MemTotal_bytes{job="node_export"})	Memory information field MemAvailable_bytes.
node_memory_MemTotal_bytes	
node_memory_MemTotal_bytes{job="node_export"}	Memory Information field MemTotal_bytes.
node_filesystem_avail_bytes	
100 - ((node_filesystem_avail_bytes{job="node_export",mountpoi nt="/",fstype!="rootfs"} * 100) / node_filesystem_size_bytes{job="node_export",mountpoint ="/",fstype!="rootfs"})	FileSystem space available to non-root users in bytes.
node_filesystem_size_bytes{job="node_export",mountpoint ="/",fstype!="rootfs"}	Total FileSystem space available.
node_scrape_collector_duration_seconds	
node_scrape_collector_duration_seconds{job="node_export "}	Duration of a collector scrape.

node_network_receive_bytes_total	
irate(node_network_receive_bytes_total{job="node_export"}[$__rate_interval])*8	The average network traffic received.
node_network_transmit_bytes_total	
irate(node_network_transmit_bytes_total{job="node_export"}[$_rate_interval])*8	Network device statistic transmit_bytes.

6.2 Dashboard Creation

MySQL, Node and Prometheus metrics were monitored separately from Grafana as below.

MySQL Metrics

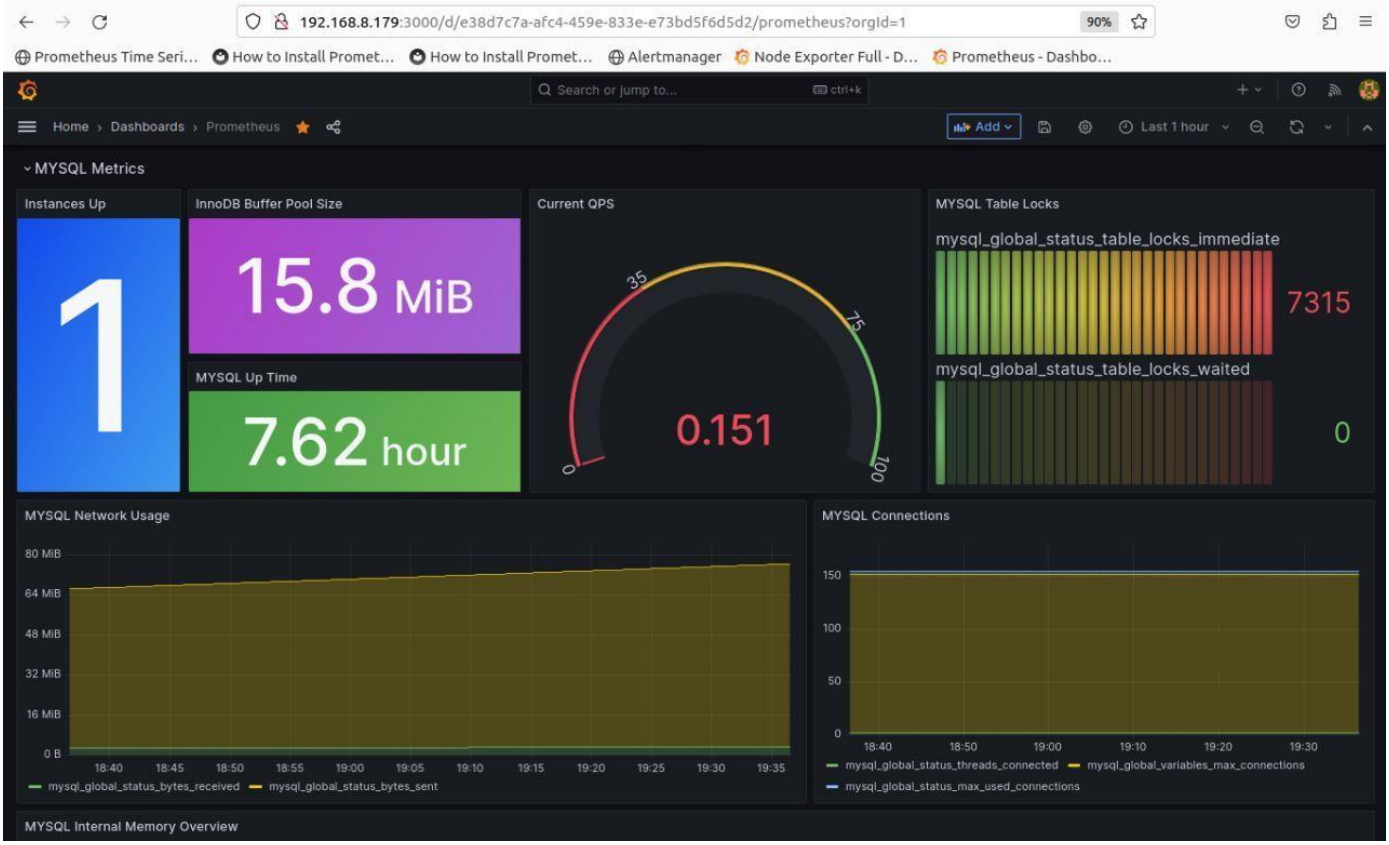


Figure 12- MySQL metrics in dashboard part1

Figure 13- MySQL metrics in dashboard part2

Node Metrics

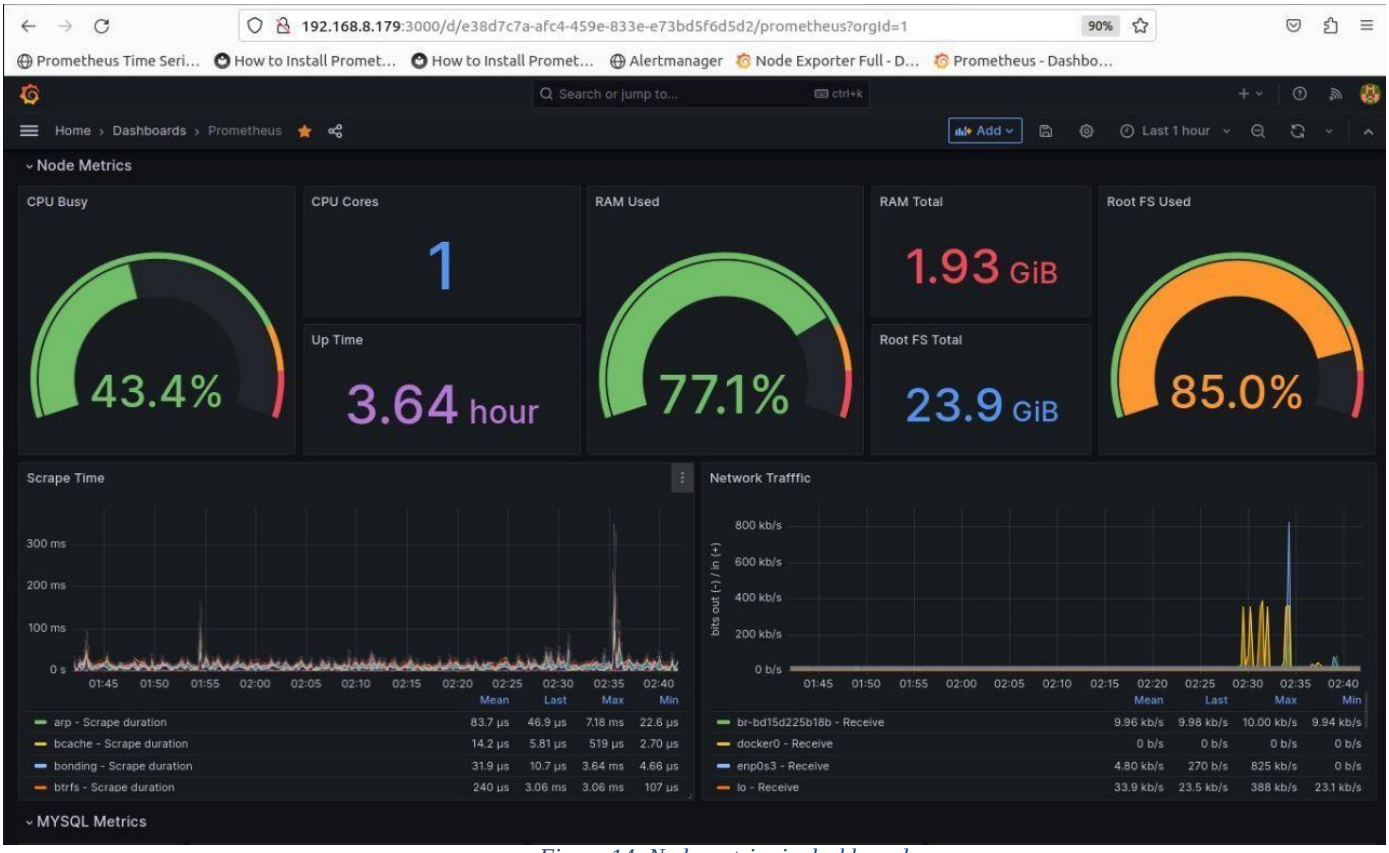


Figure 14- Node metrics in dashboard

Prometheus Metrics

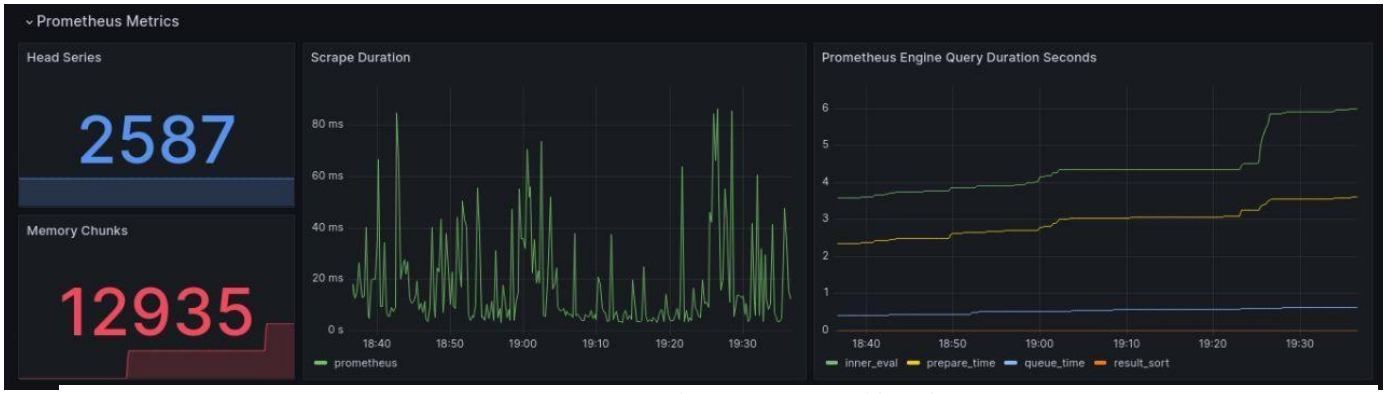
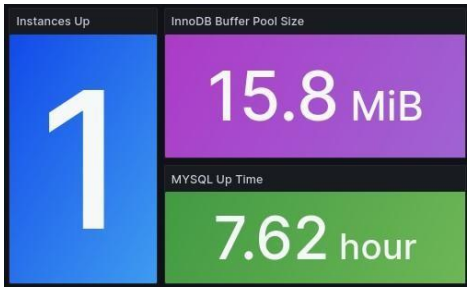


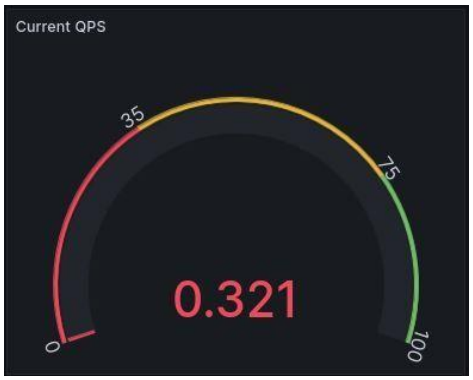
Figure 15- Prometheus metrics in Dashboard

6.3 MySQL Dashboard Explanation



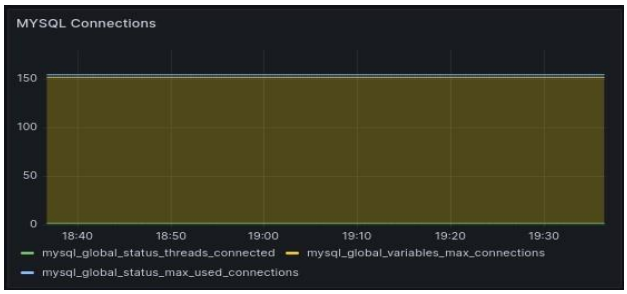
Stats

- Instances up - mysql_up
- MySQL Up Time - mysql_global_status_uptime
- InoDB Buffer Pool Size - mysql_global_variables_innodb_buffer_pool_size



Display Gauge

- Current QPS up - mysql_global_status_queries

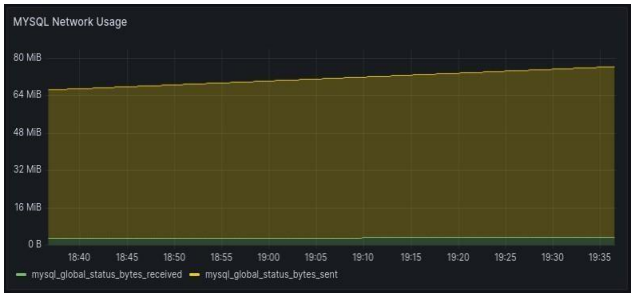


Graph

- MySQL Connections - mysql_global_status_threads_connected

mysql_global_variables_max_connections

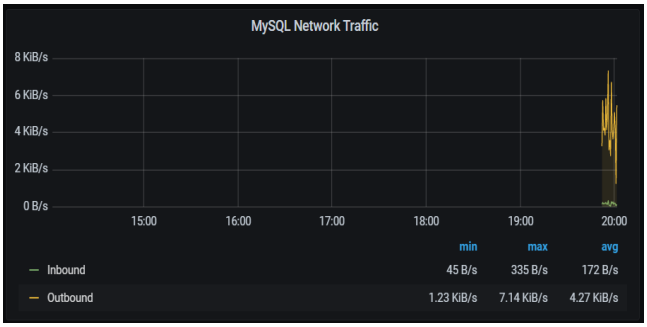
mysql_global_status_max_used_connections



Graph

- MySQL Network Usage Hourly - mysql_global_status_bytes_received

mysql_global_status_bytes_sent



Graph

- MySQL Network Traffic - mysql_global_status_bytes_received

mysql_global_status_bytes_sent

Graph

- MySQL Table Locks - mysql_global_status_table_locks

mysql_global_status_table_locks_waited

Graph

- MySQL Internal Memory Overview –

  mysql_global_status_innodb_page_size

  mysql_global_status_buffer_pool_pages

  mysql_global_variables_key_buffer_size

  mysql_global_variables_query_cache_size

  mysql_global_variables_innodb_log_buffer_size

  mysql_global_status_innodb_mem_adaptive_hash

  mysql_global_status_innodb_mem_dictionary

  node_memory_MemTotal_bytes

  mysql_global_variables_innodb_additional_mem_pool_size

  mysql_global_variables_tokudb_cache_size

7. Appendix


192.168.8.179:3000/d/e38d7c7a-afc4-459e-833e-e73bd5f6d5d2/prometheus?orgId=1#
How to Install Promet...
Alerts - Prometheus T...
Targets - Prometheus ...
Prometheus Time Seri...
Prometheus - Dashbo...
Node Exporter Full - D...
Home › Dashboards › Prometheus
Add
Last 1 hour
Node Metrics
CPU Busy
21.1%
CPU Cores
1
Up Time
4.12 hour
RAM Used
82.1%
RAM Total
1.93 GiB
Root FS Total
23.9 GiB
Root FS Used
85.0%
Scrape Time
arp - Scrape duration
bcache - Scrape duration
bonding - Scrape duration
btrfs - Scrape duration
Network Trafffic
br-bd15d225b18b - Receive
docker0 - Receive
enp0s3 - Receive
lo - Receive
MYSQL Metrics
Instances Up
1
InnoDB Buffer Pool Size
15.5 MiB
MYSQL Up Time
11.9 hour
Current QPS
0.321
MYSQL Table Locks
mysql_global_status_table_locks_immediate
3951
mysql_global_status_table_locks_waited
0
MYSQL Network Usage
mysql_global_status_bytes_received
mysql_global_status_bytes_sent
MYSQL Connections
mysql_global_status_threads_connected
mysql_global_variables_max_connections
mysql_global_status_max_used_connections

Figure 16- Dashboard

8. References

https://docs.splunk.com/observability/gdi/prometheus-node/prometheus-node.html

https://docs.docker.com/storage/bind-mounts/

https://prometheus.io/docs/prometheus/latest/installation/

https://docs.docker.com/engine/install/ubuntu/

https://www.server-world.info/en/note?os=Ubuntu_18.04&p=prometheus&f=2

https://devconnected.com/complete-mysql-dashboard-with-grafana-prometheus/

https://github.com/prometheus/mysqld_exporter/blob/main/mysqld-mixin/dashboards/mysql-overview.json

https://severalnines.com/database-blog/how-monitor-mysql-containers-prometheus-deployment-standalone-and-swarm-part-one

https://hub.docker.com/_/mysql

https://grafana.com/docs/grafana/latest/

https://ubuntu.com/download/desktop