Sri Lanka Institute of Information Technology

Visual Analytics and User Experience Design

(IT4031)

4th Year 2nd Semester

Assignment 2

Group ID: 2024_A2_G14

Sewwandi W.M.C - IT20298494

Munasinghe M.G.P - IT20667450

Purnamal M.C.P - IT20655334

Gangoda G.G.W.N - IT20916626

Contents

Project Links

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

- Node exporter metrics - http://localhost:9100/metrics
- Prometheus - http://localhost:9090/targets
- Grafana - http://localhost:3000/?orgId=1
- Dashboard - http://localhost:3000/d/c08C_7UVk1/vaued-dashboard-latest?orgId=1
- Project folder drive link - https://mysliit-my.sharepoint.com/:f:/g/personal/it20655334_my_sliit_lk/EgtGx592DJVEpYsz8TcsEeYBYe5k2aPD_WQgvAK7a6OcKA?e=U1uVfd

Introduction

This project undertook the implementation of a metrics monitoring and visualization system utilizing Prometheus and Grafana. The primary objective was to capture metrics from an application and present them through interactive dashboards, facilitating effective analysis and monitoring of key performance indicators.

The core components deployed included an application, Prometheus server, and Grafana server. The developed application exposed Prometheus-compatible metrics through integration with third-party exporters. Prometheus, a robust monitoring system, collected metrics from the application at regular intervals and stored them for analysis. Grafana, a powerful visualization platform, enabled the creation of dynamic and insightful dashboards using the collected metrics. Additionally, alerting mechanisms were implemented within Grafana, enabling proactive notification of performance anomalies or issues.

Throughout the project, a meticulous approach was followed to meet the assignment's requirements. The deployment steps, configurations, and challenges encountered were thoroughly documented, serving as a valuable reference for future deployments. The exploration of containerization using Docker further showcased the system's portability and scalability.

In conclusion, the implementation of the metrics monitoring and visualization system using Prometheus and Grafana yielded valuable insights into application performance. The seamless integration of these tools demonstrated the power of metrics-driven monitoring and visualization. This report will delve into all assignment checkpoints, providing a comprehensive overview of the project's execution.

Architecture Diagram

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

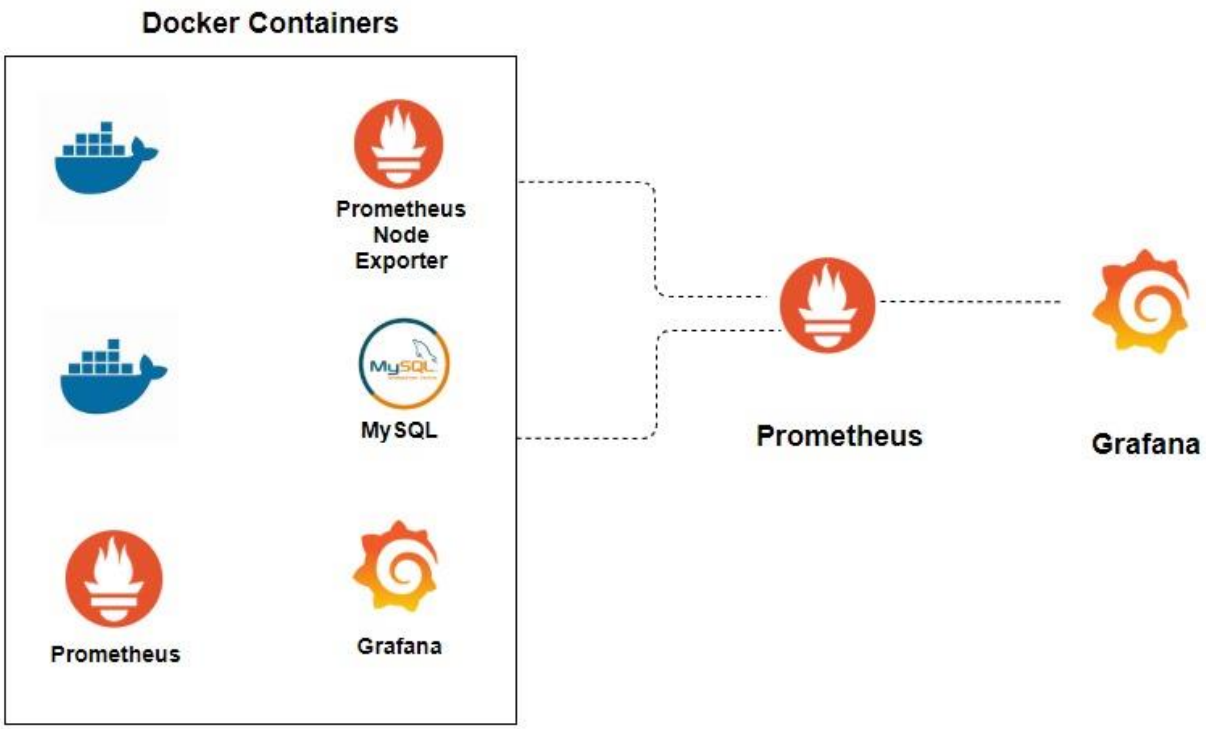

Figure 1: System Architecture

Install MySQL on Docker

- MySQL has been installed on Docker by using the steps below.

```
mysql-server:
  image: mysql/mysql-server:latest
  container_name: mysql-server
  restart: always
  environment:
    - MYSQL_ROOT_PASSWORD=1234
    - MYSQL_USER=user
    - MYSQL_PASSWORD=vaued2
    - MYSQL_DATABASE=vaued
  ports:
    - 3306:3306
  volumes:
    - ./mysql_data:/var/lib/mysql
  networks:
    - monitoring
```

- Verify if our MySQL server is running Okay.

  docker ps

- Create monitoring User.
  - **docker exec -it mysql-server mysql -uroot -p**
  - On MYSQL > **CREATE USER 'user'@'%' IDENTIFIED BY 'vaued2' WITH MAX_USER_CONNECTIONS 3;**

Setup MySQL Exporter

- Docker composes configuration for MySQL Exporter.

```
mysql-exporter:
  image: prom/mysqld-exporter
  container_name: mysql-exporter
  environment:
    - MYSQLD_EXPORTER_PASSWORD=vaued2
    - DATA_SOURCE_NAME=user:vaued2@(mysql-server:3306)/vaued
    - collect.info_schema.tablestats=true
    - collect.info_schema.userstats=true
    - collect.info_schema.query_response_time=true
    - collect.auto_increment.columns=true
    - collect.binlog_size=true
    - collect.perf_schema.eventsstatements=true
    - collect.perf_schema.eventswaits=true
    - collect.perf_schema.file_events=true
    - collect.perf_schema.indexiowaits=true
    - collect.perf_schema.tableiowaits=true
    - collect.perf_schema.tablelocks=true
  depends_on:
    - mysql-server
  ports:
    - 9104:9104
  networks:
    - monitoring
```

Install Node Exporter on Windows

- Node Exporter was installed on Windows as a service to monitor metrics of the machine.

```
node-exporter:
  image: prom/node-exporter:latest
  container_name: node-exporter
  restart: unless-stopped
  volumes:
    - /proc:/host/proc:ro
    - /sys:/host/sys:ro
    - /:/rootfs:ro
  command:
    - '--path.procfs=/host/proc'
    - '--path.rootfs=/rootfs'
    - '--path.sysfs=/host/sys'
    - '--collector.filesystem.mount-points-exclude=^/(sys|proc|dev|host|etc)($$|/)'
  ports:
    - 9100:9100
  expose:
    - 9100
  networks:
    - monitoring
```

Configure Prometheus

Installing Prometheus

- Docker compose configuration for Prometheus.

```
prometheus:
  image: prom/prometheus:v2.20.1
  container_name: prometheus
  volumes:
    - ./prometheus:/etc/prometheus
    - prometheus_data:/prometheus
  ports:
    - 9090:9090
  expose:
    - 9090
  networks:
    - monitoring
```

Setting the Targets in Prometheus

- Target ports have configured in .yml file as follows.

```
global:
 scrape_interval: 5s
scrape_configs:
 - jobnames: "prometheus"
  static_configs:
    - targets: ["prometheus:9090"]
 - job_name: "vaued-nodejs-app"
  static_configs:
    - targets: ["vaued-nodejs-app:8080"]
 - job_name: 'mysql-exporter'
```

```
  static_configs:
    - targets: ['mysql-exporter:9104']
  - job_name: 'node'
  static_configs:
    - targets: ['node-exporter:9100']
```

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

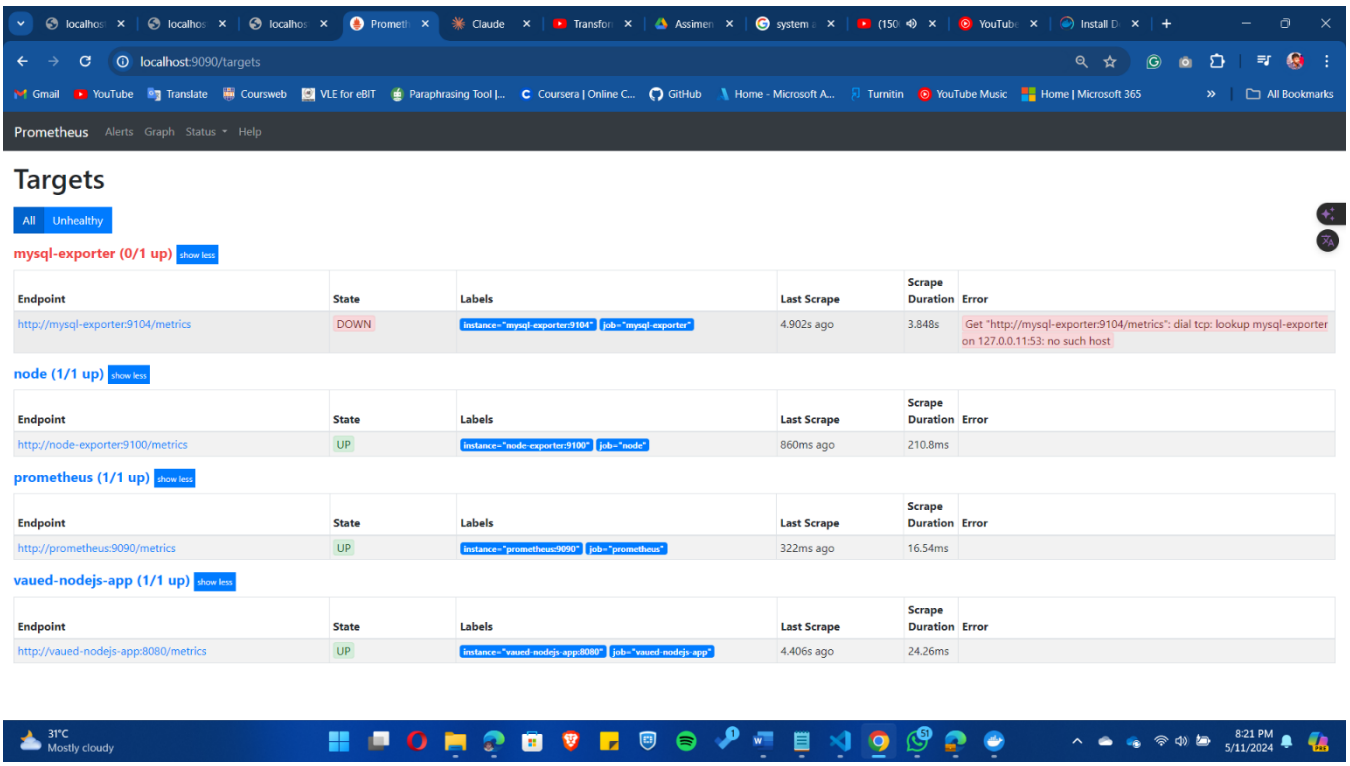

Figure 5: Prometheus monitoring targets

Retrieving Metrics

- Metrics can be retrieved as follows.

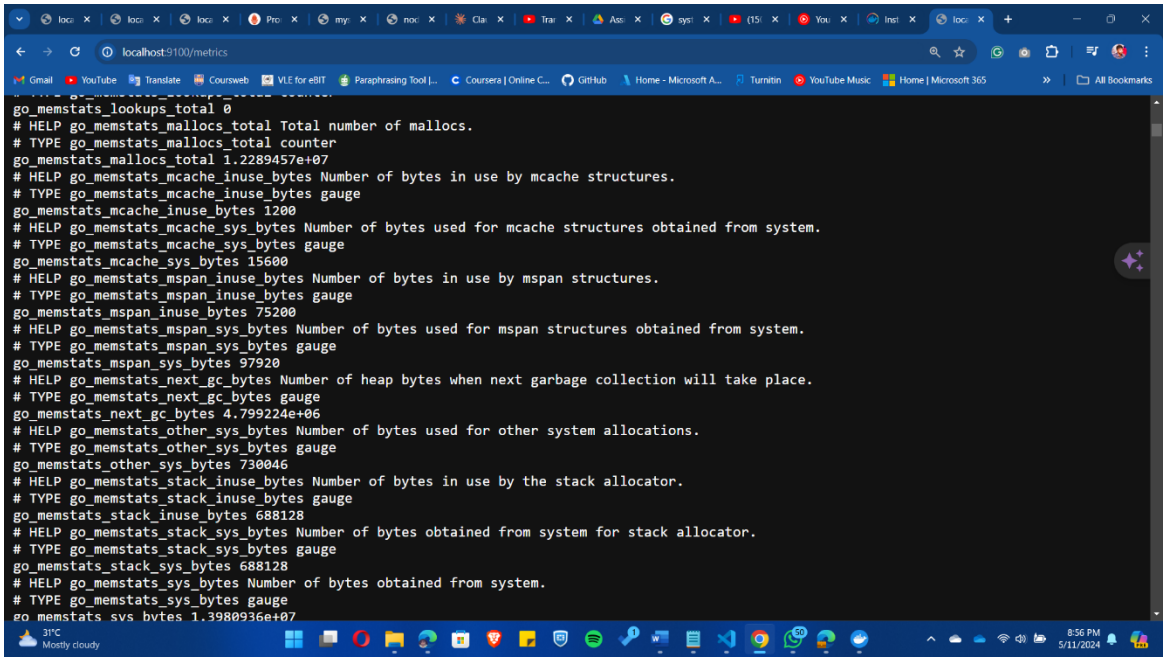

Figure 7: Node Exporter metrics

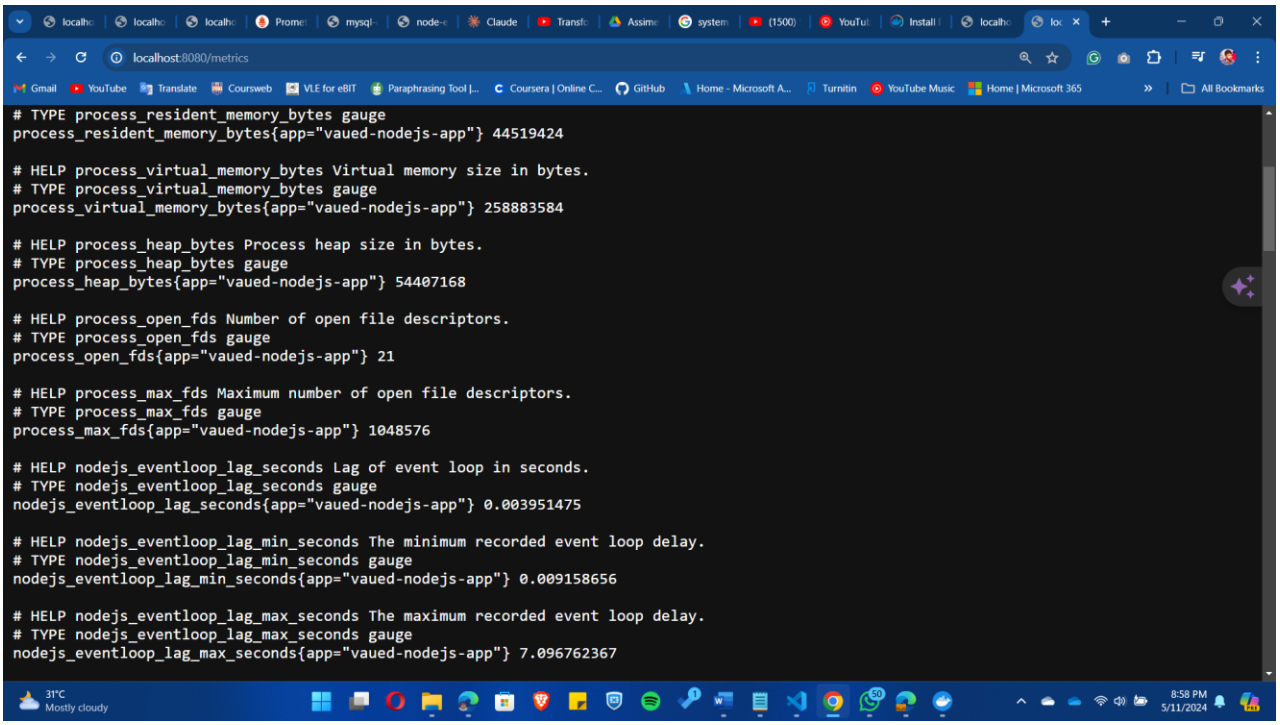

Figure 8: nodeJS metrics

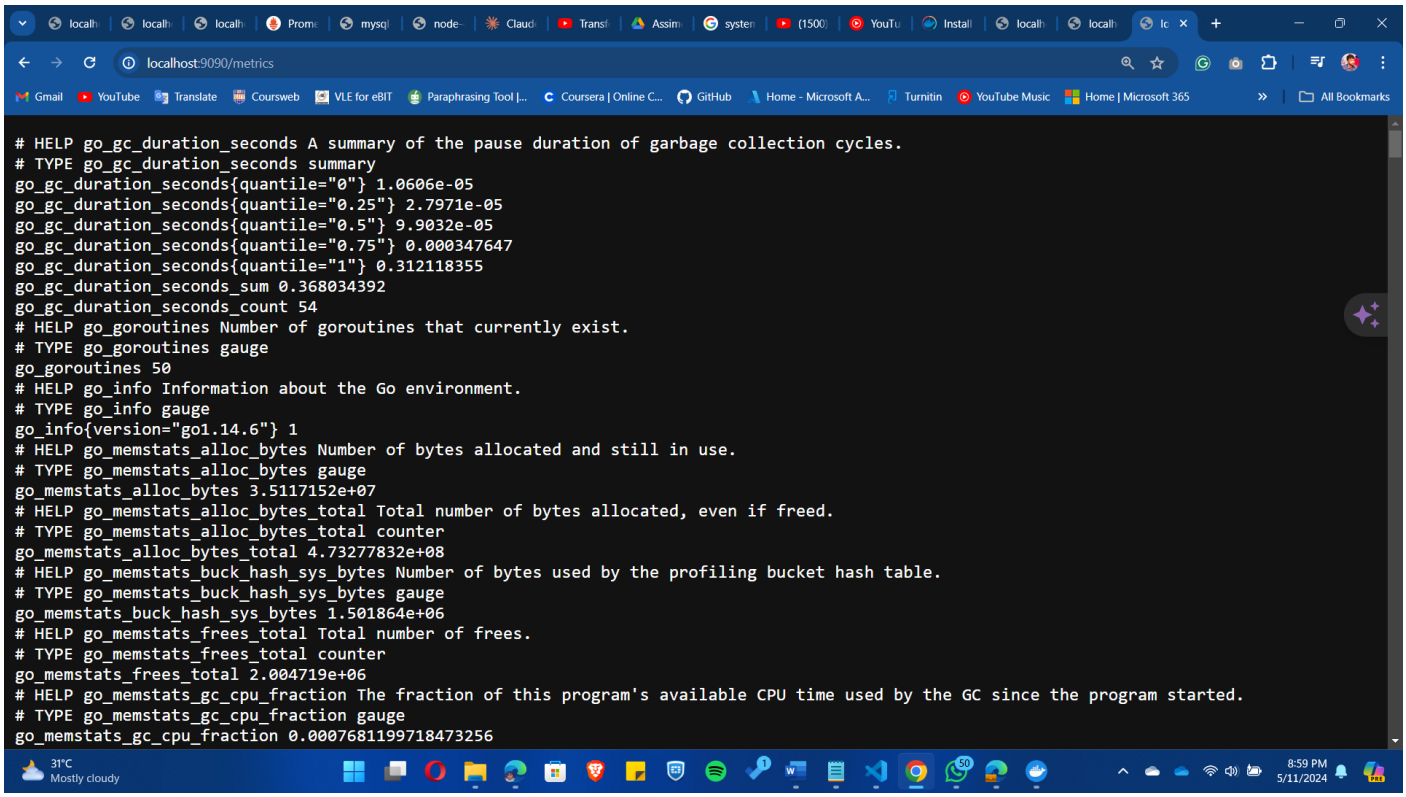

Figure 9: Prometheus metrics

Grafana Dashboard

Install and Configure Grafana on Docker

- Grafana was installed on Docker using the following Docker compose configuration.

  grafana:

  image: grafana/grafana:9.1.4

  container_name: grafana

  volumes:

  - grafana_data:/var/lib/grafana

  - ./grafana/provisioning:/etc/grafana/provisioning

  environment:

  - GF_AUTH_DISABLE_LOGIN_FORM=true

  - GF_AUTH_ANONYMOUS_ENABLED=true

  - GF_AUTH_ANONYMOUS_ORG_ROLE=Admin

ports:

- 3000:3000

expose:

- 3000

networks:

- monitoring

Create Prometheus as the Data Source

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

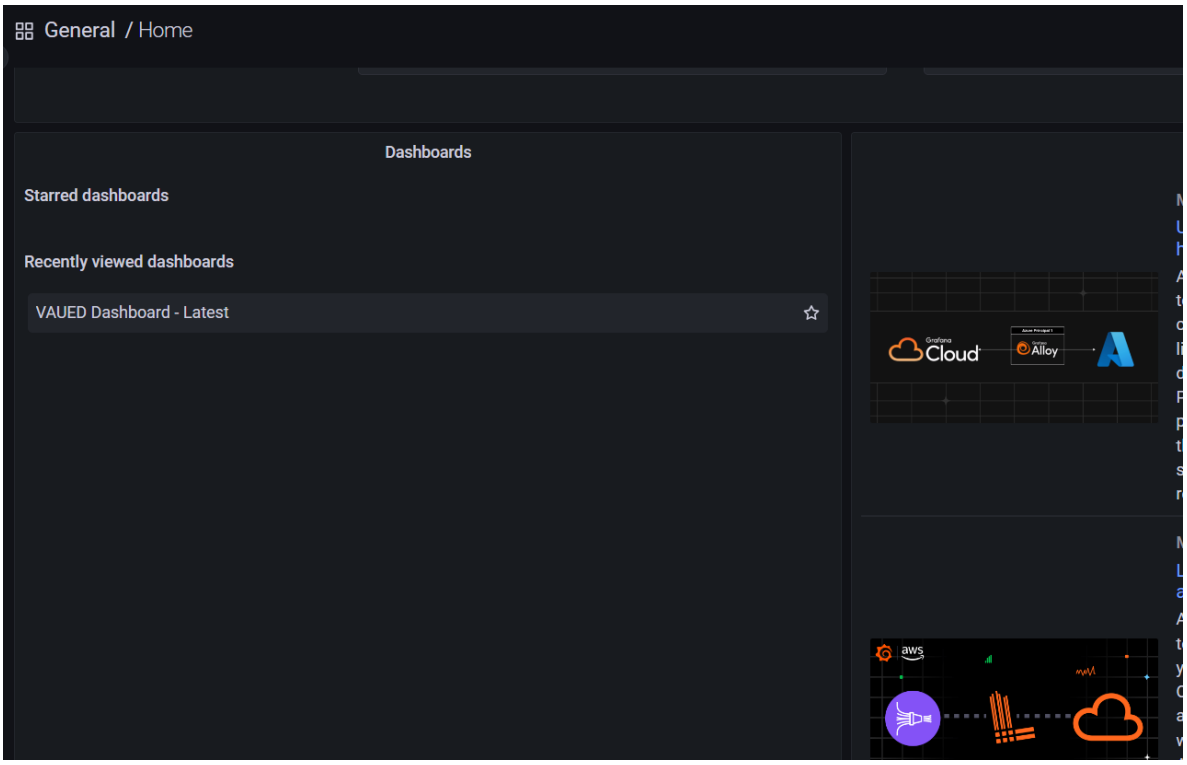

Figure 10: Grafana Dashboard Creation

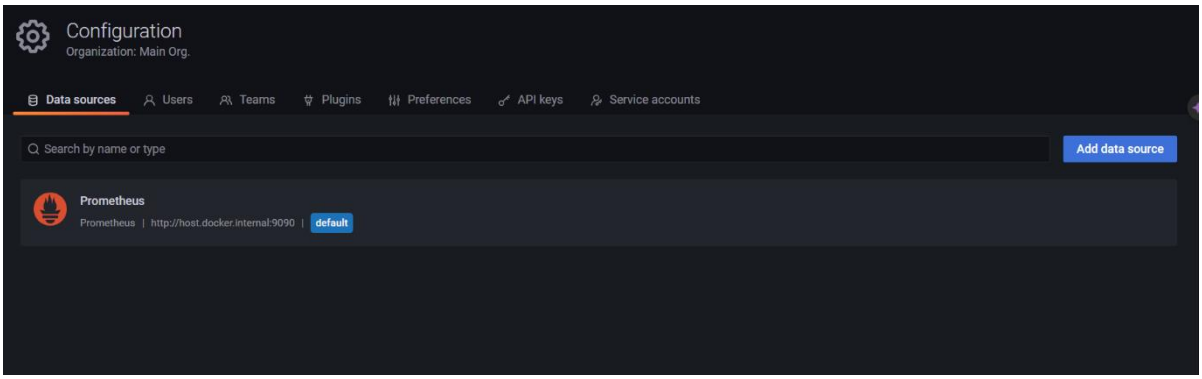

Figure 11: Setting Prometheus as Data source

Used Metrics

Table 1: MySQL Metrics

MySQL Metrics	
Metrics Used & PromQL Queries	**Justification**
mysql_global_status_queries	
irate(mysql_global_status_queries[5m])	Check Queries Per Second(QPS).
mysql_up	
mysql_up	Check the availability.
mysql_global_status_uptime	
mysql_global_status_uptime	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_variables_innodb_buffer_pool_size	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	5-mysql_global_status_threads_connected The number of currently open connections.
mysql_global_status_bytes_received	
increase(mysql_global_status_bytes_received[1h])	The number of bytes received from all clients.
mysql_global_status_bytes_sent	
increase(mysql_global_status_bytes_sent[1h])	The number of bytes sent to all clients.
mysql_global_status_table_locks_immediate	
irate(mysql_global_status_table_locks_immediate[5m])	Represents the total number of row locks.
mysql_global_status_table_locks_waited	

irate(mysql_global_status_table_locks_wa ited[5m])	Number of table locks.
mysql_global_status_innodb_page_size	
mysql_global_status_innodb_page_size * on (instance) mysql_global_status_buffer_pool_pages	Total size of buffer pool in bytes.
mysql_global_variables_key_buffer_size	
mysql_global_variables_key_buffer_size	Key Buffer Size.
mysql_global_variables_query_cache_size	
mysql_global_variables_query_cache_size	Query Cache Size.
mysql_global_variables_innodb_log_buffer_size	
mysql_global_variables_innodb_log_buffer_size	InnoDB Log Buffer Size
mysql_global_status_innodb_mem_adaptive_hash	
mysql_global_status_innodb_mem_adaptive_hash	Adaptive Hash Index Size.
mysql_global_status_innodb_mem_dictionary	
mysql_global_status_innodb_mem_dictionary	InnoDB Dictionary Size.

Table 2: Prometheus Metrics

Prometheus Metrics	
Metrics Used & PromQL Queries	**Justification**
prometheus_tsdb_head_series	
sum(prometheus_tsdb_head_series)	Covers every series that has existed in the last 1-3 hours
prometheus_tsdb_head_chunks	
sum(prometheus_tsdb_head_chunks)	Total number of chunks in the head block.
prometheus_engine_query_duration_seconds_sum	
sum(prometheus_engine_query_duration_seconds_sum) by (slice)	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) by (cpu))) - avg(sum by (mode)(rate(node_cpu_seconds_total{mode='idle'}[5m])))) * 100) / count(count(node_cpu_seconds_total) by (cpu))	This is a counter metric that counts. The number of seconds the CPU has been running.
node_time_seconds	
node_time_seconds -node_boot_time_seconds	System time in seconds epoch(1970).
node_memory_MemAvailable_bytes	
# RAM Used 100 - ((node_memory_MemAvailable_bytes * 100) / node_memory_MemTotal_bytes)	Memory information field MemAvailable_bytes.
node_memory_MemTotal_bytes	
node_memory_MemTotal_bytes	Memory Information field MemTotal_bytes.
node_filesystem_avail_bytes	
100 - ((node_filesystem_avail_bytes{fstype!="rootfs", mountpoint="/mnt"} * 100) / node_filesystem_size_bytes{fstype!="rootfs", mountpoint="/mnt"})	FileSystem space available to non-root users in bytes.

Dashboard Creation

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

MySQL

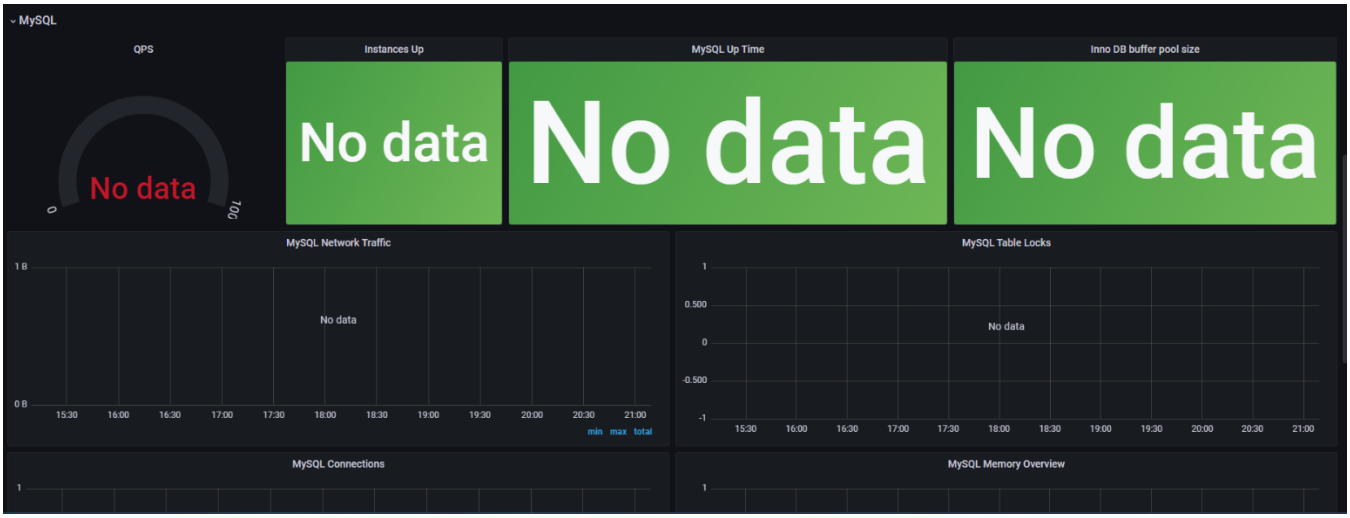

Figure 12: Grafana Dashboard Visualizing MySQL Metrics

System Metrics

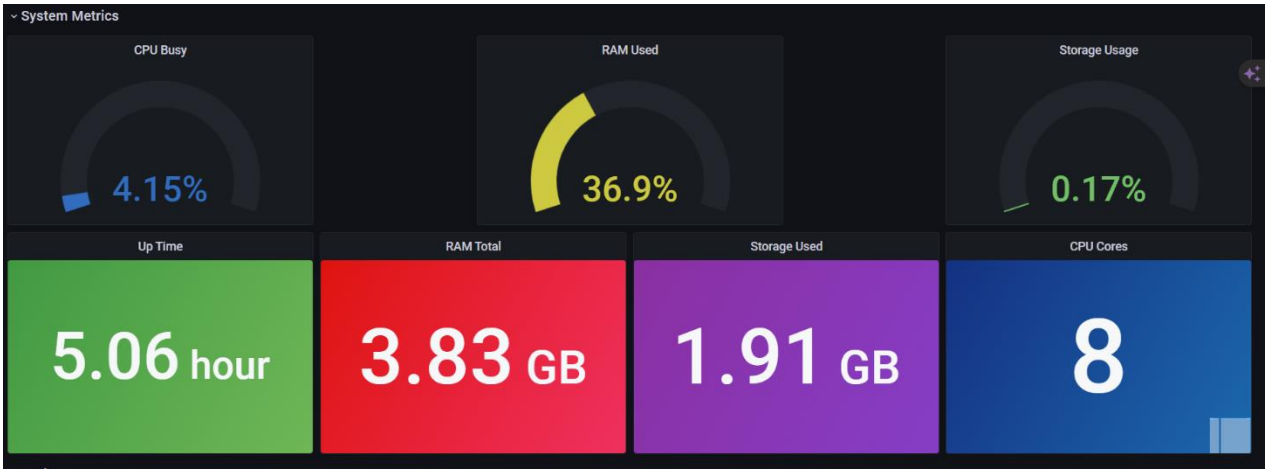

Figure 13: Grafana Dashboard Visualizing System Metrics

Prometheus

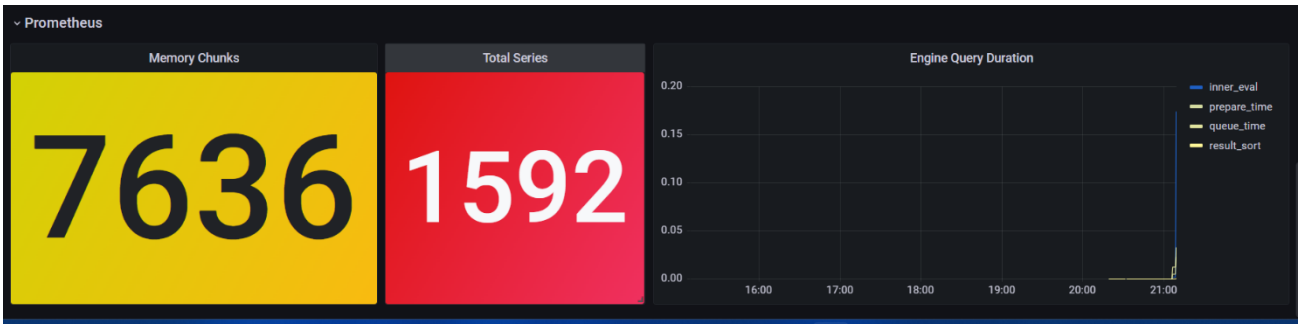

Figure 14: Grafana Dashboard Visualizing Prometheus Metrics

MySQL Dashboard Explanation

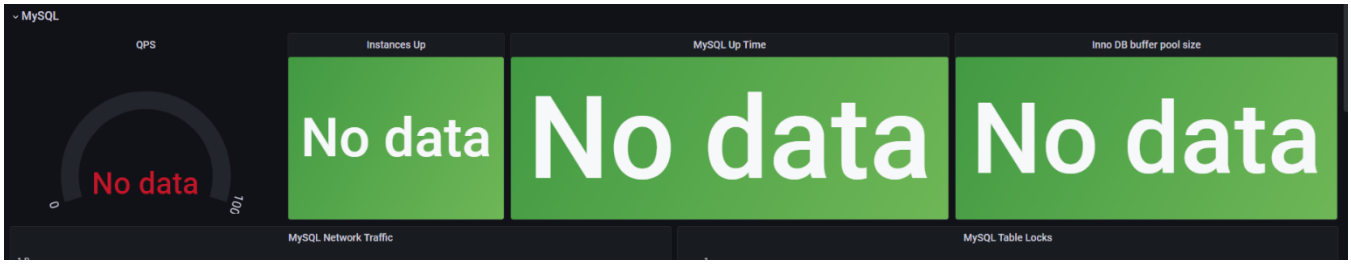

Figure 15: MySQL dashboard

<u>Stats</u>

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

<u>Gauge</u>

- QPS - mysql_global_status_queries

Member Contributions to the Project

IT Number	Name	Contribution
IT20655334	Purnamal M C P	• Docker set up • Implementing Node JS application • Document creation.
IT20298494	Sewwandi W.M.C	• Setting up alerts • Configure Node exporter. • Document creation
IT20667450	Munasinghe M.G.P	• Setting up Grafana Dashboard. • Document creation.
IT20916626	Gangoda G.G.W.N	• Installing and setting up Prometheus. • Document creation.

References

[1] https://prometheus.io/docs/introduction/overview/

[2] https://docs.docker.com/guides/