PostgreSQL Performance Metrics Scraping into Prometheus and Grafana

Objective:

You want to scrape the following PostgreSQL query performance metrics into Prometheus -> Grafana:

- pg_stat_statements_query_stats_calls
- pg_stat_statements_query_stats_max_time
- pg_stat_statements_query_stats_mean_time
- pg_stat_statements_query_duration_histogram_bucket
- pg_stat_statements_query_duration_histogram_count
- pg_stat_statements_query_duration_histogram_sum
- p95 / p99 percentiles for query duration

Scenario:

PostgreSQL is running on a separate virtual machine (VM), while Prometheus is running as a pod in a Kubernetes cluster.

To collect the required metrics from PostgreSQL, we install an exporter (agent as docker container) on the same VM where PostgreSQL is running.

Steps from Database Side:

1. Enable pg_stat_statements extension:
   - docker exec -it postgres psql -U admin -d monitoringdb
   - CREATE EXTENSION IF NOT EXISTS pg_stat_statements;

2. Confirm:
   - SELECT * FROM pg_stat_statements LIMIT 5;

3. Update postgresql.conf:

- shared_preload_libraries = 'pg_stat_statements'

- track_activity_query_size = 2048

4. Apply via sed if editing is not available, then restart container.

5. Check pg_stat_statements.track = 'all' and save = on

6. Restart and confirm:

- docker restart postgres

7. Validate functionality:

- SELECT query, calls, mean_exec_time, total_exec_time FROM pg_stat_statements ORDER BY total_exec_time DESC LIMIT 10;

Exporter Setup:

1. Create `queries.yaml` with custom SQL queries to collect pg_stat_statements metrics.

2. Run postgres-exporter container:

- docker run -itd --name pg-exporter --network=host ...

- Mount queries.yaml file and set env variables

3. Confirm with:

- docker logs pg-exporter | grep -i error

4. Verify exporter endpoint: http://<VM-IP>:9187/metrics

5. Update Prometheus scrape configs:

```
scrape_configs:
  - job_name: 'postgres-exporter'
    static_configs:
      - targets: ['<your-host-ip>:9187']
```

Grafana Dashboards:

- Panel Title: Top Queries by Avg/Max Execution Time

  Query: topk(10, pg_stat_statements_query_stats_mean_time)

- Panel for p95 / p99:

  Query: histogram_quantile(0.99, sum by (le, queryid)(rate(pg_stat_statements_query_duration_histogram_bucket_count[5m])))

Optional:

- Top slow queries by p95:

  Query: topk(10, histogram_quantile(0.95, sum by (le, queryid)(rate(...))))