Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет Информационных технологий и программирования

Лабораторная работа №4
Мониторинг СУБД PostgreSQL

Выполнил: студент группы
M3203
Костыгов Андрей Константинович
Проверила:
Шевчик Софья Владимировна

Санкт-Петербург
2024 г.

1) На свой выбор установить инструменты мониторинга.
2) Экспортировать полезные данные в Prometheus.
3) Установить Grafana.

Дополним файл docker-compose:

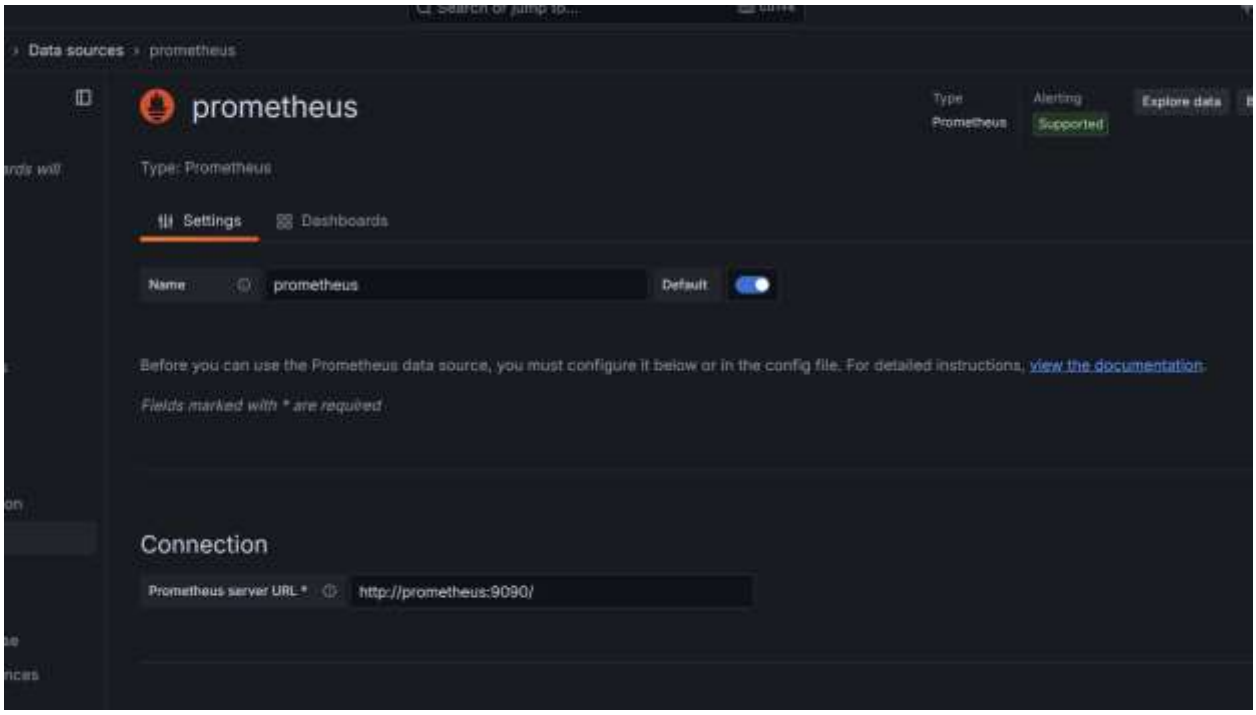
…

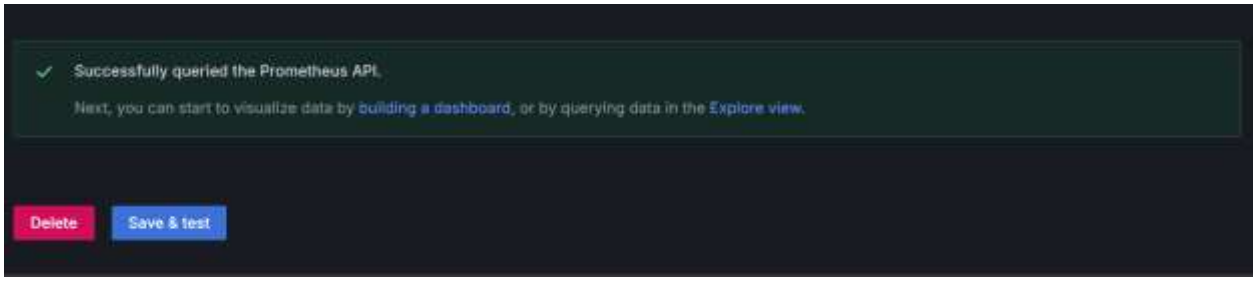
```
  grafana:
    image: grafana/grafana
    networks: [ patronistream ]
    ports:
      - 3000:3000

  prometheus:
    image: prom/prometheus
    networks: [ patronistream ]
    ports:
      - 9090:9090
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml:ro

  postgres-exporter:
    image: prometheuscommunity/postgres-exporter
    networks: [ patronistream ]
    ports:
      - 9187:9187
    volumes:
      - ./settings/queries.yml:/home/queries.yml
    env_file:
        - .env
    environment:
      DATA_SOURCE_NAME:
"postgresql://kosandron:1234@${POSTGRES_HOST}:${POSTGRES_PORT}/${
POSTGRES_DB}?sslmode=disable"
      PG_EXPORTER_EXTEND_QUERY_PATH: ./home/queries.yml
    links:
      - haproxy
      - prometheus
```

После этого в Grafana на порту 3000 добавим источник данных:

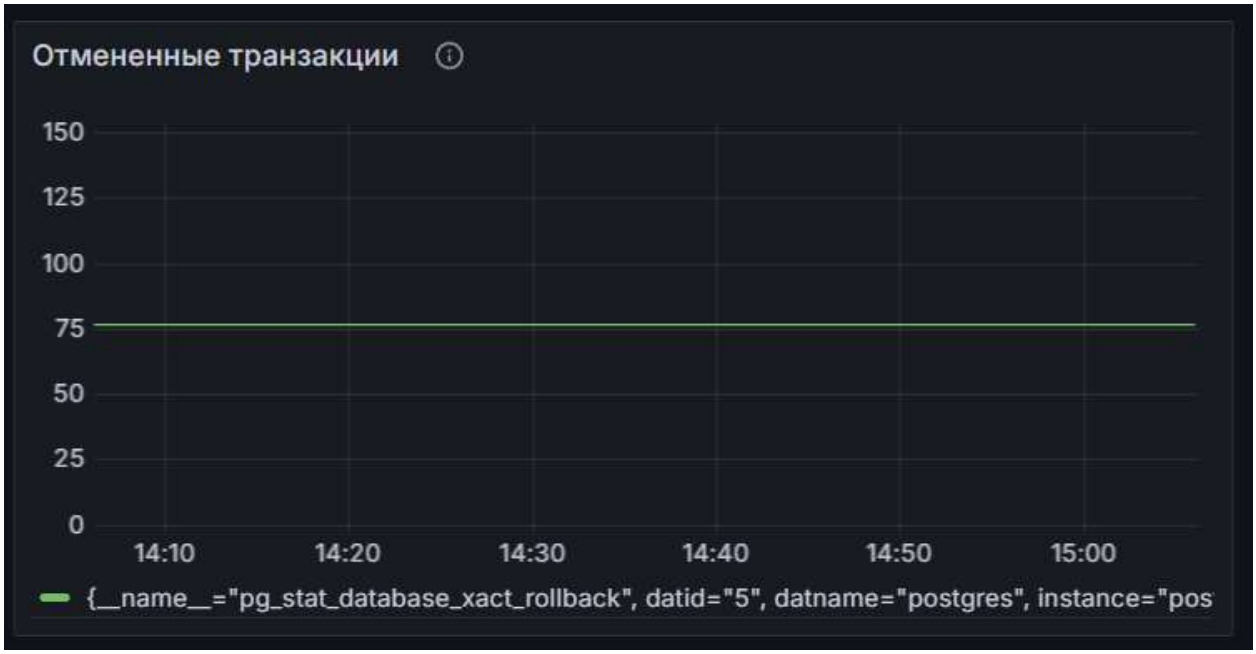




Далее создадим дашбоард с метриками:
Queries per second:
irate(pg_stat_statements_qps[1m])

Transactions per second:
irate(pg_stat_database_tps[1m])

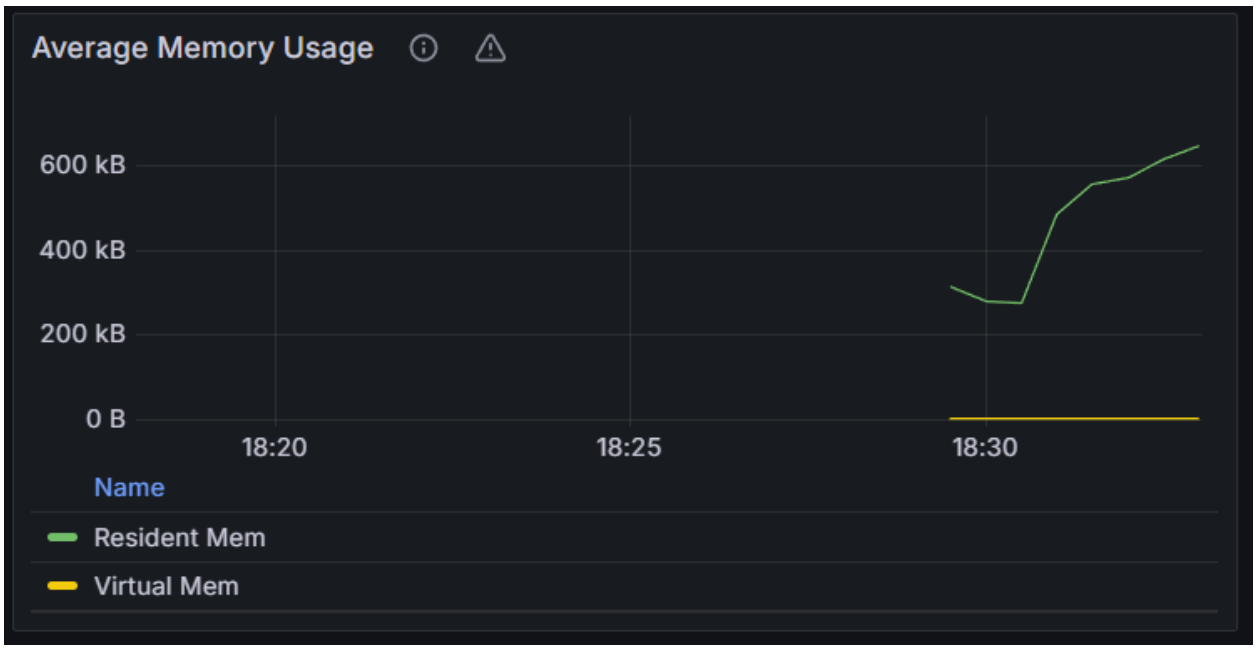
Rollbacks:
pg_stat_database_xact_rollback{datname="postgres"}



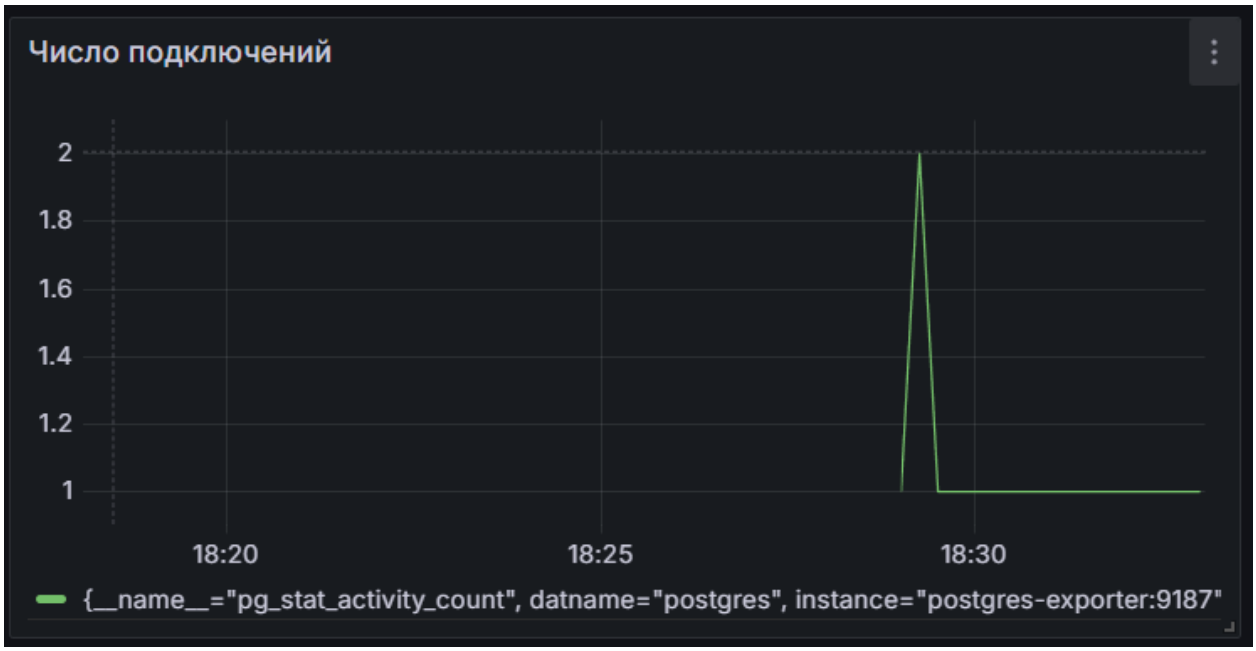
Среднее время отклика (расчет смотри ниже):

CPU Usage (процессорное время в секундах):
avg(rate(process_cpu_seconds_total[5m]) * 1000)

Использование памяти:



Число подключений к СУБД:



С помощью TPS, QPS и среднего времени отклика мы можем замечать новые транзакции, и если их много, то точно заметим возможную задержку. С помощью графика Rollbacks можем заметить, есть ли какая-то поломка или сбой, что происходит увеличение числа Rollback. С помощью мониторинга CPU мы можем следить за потреблением

Некоторые метрики мы можем сами создать через файл queries.yml:

```
games_count:
  query: "SELECT COUNT(*) AS games_count FROM games"
  metrics:
    - games_count:
        usage: "GAUGE"
        description: "Games count"
```

```
pg_stat_statements:
  query: "SELECT 1.0 * sum(total_exec_time) / sum(calls) as reply_time,
sum(calls) as QPS FROM pg_stat_statements"
  metrics:
    - reply_time:
        usage: "GAUGE"
        description: "Average reply time"
    - QPS:
        usage: "GAUGE"
        description: "Queries per second"
pg_stat_database:
  query: "SELECT sum(xact_commit) + sum(xact_rollback) as TPS FROM
pg_stat_database"
  metrics:
    - TPS:
        usage: "GAUGE"
        description: "Transactions per second"
```

Общий вид дашборда:

Экспортируем дашборд в JSON-файл: