Assignment #4

Important Information

1. Deadline — Week 9 (you can submit within 2 weeks after the topic explanation).
2. If you cannot attend in person for valid reasons, you can defend your project online only after getting approval in Teams. Without approval, online defense is not allowed.
3. All completed assignments must be uploaded to Moodle. If a student does not provide a GitHub link in Moodle or via Teams, and does not defend the project, the grade will be 0.
4. After each assignment, update your GitHub repository before uploading the work to Moodle.
5. The defense must be done by running the scripts live. All scripts must be fully written and saved in advance.
6. Late submission penalties: 1–2 days late — 5% deduction; 3–7 days late — 15% deduction; up to 2 weeks late — 30% deduction; more than 2 weeks late — 50% deduction.

Goal

In this assignment, you will learn to work with **Prometheus** — a system for real-time monitoring and metric collection. Unlike the previous task with **Apache Superset**, where you analyzed data inside databases, here the focus is on **system monitoring** and **collecting metrics from different sources.**

> Be sure to read Lectures 6–7–8 in order and check the FAQ.
>
> They explain step by step how to run Prometheus and Grafana, and how to launch DB Exporter, Node Exporter, and Custom Exporter.

What We Will Monitor

Tasks

General Conditions:

You must create **three dashboards:**

1. **Database Exporter (PostgreSQL/MySQL)** — 30 points
2. **Node Exporter (System Monitoring)** — 25 points
3. **Custom Exporter (External APIs)** — 45 points

To earn points for a dashboard, **all checklist items must be completed.** If even one point is missing — you get **0 points** for that dashboard. *(Partial defense is not allowed. Make sure everything is ready before your presentation.)*

№	Requirement	Done
1	Prometheus and Grafana are successfully running and connected	+/-
2	The corresponding Exporter (DB/Node/Custom) is running and available at `http://localhost:port`	+/-
3	At least 10 **PromQL queries** are created for this dashboard	+/-
4	At least **60% of the queries** use functions (`avg`, `rate`, `sum`, `count`, `time()`, etc.) or time filters (`[5m]`, `by()`, `grouping`)	+/-
5	All 10 PromQL queries are **tested in Prometheus** and correctly return data (demonstrated during defense)	+/-
6	Metrics were collected for **1–5 hours** (you can simulate load if needed)	+/-
7	Dashboard contains **≥10 visualizations**, with **at least 4 different types** (e.g., 3-time series, 3-gauge, 1-heatmap, 3-bar)	+/-
8	A **global filter (dashboard variable)** is configured and works across all panels	+/-
9	At least **one alert rule** in Grafana (with visible trigger condition and status)	+/-
10	All data is displayed correctly (values update in real time)	+/-
11	Dashboard JSON file is exported and uploaded to GitHub. The GitHub repo also includes: `docker-compose.yml`, `prometheus.yml`, `custom_exporter.py`, `README.md`	+/-
12	During defense, you show: container status, Targets (all “UP”), PromQL query results, and	+/-

	Grafana visualizations	

(30 points) Dashboard 1 — Database Exporter (PostgreSQL / MySQL)

This dashboard shows the performance and internal statistics of your SQL server (load, size, activity).

Key metrics to visualize in Grafana (you can add your own to reach 10 PromQL queries):

- Number of active connections
- Database size (bytes, GB)
- Uptime
- Read/write operations rate
- Query processing speed (QPS)
- Number of users and privileges
- Total number of tables and rows

(25 points) Dashboard 2 — Node Exporter (System Metrics)

This dashboard monitors your computer or server resources in real time.

Main metrics (you can add your own to reach 10 PromQL queries):

- CPU usage (per core)
- Load average (1, 5, 15 min)
- Total, available, and used memory (in GB)
- RAM usage (%)
- Free disk space (GB)
- Disk I/O — read and write (bytes/sec)
- Network traffic — incoming/outgoing (Mbit/sec)
- CPU temperature
- Battery level and health (for laptops)
- **Experiment:** baseline → load → analysis (measure changes after load)
- *(Optional)* number of active processes, system uptime, swap usage, CPU frequency, GPU usage

(45 points) Dashboard 3 — Custom Exporter (External APIs)

This dashboard demonstrates how to collect and visualize data from external APIs using a custom Python exporter.

You can choose any open API (OpenWeather, Exchange Rates, GitHub, NASA, Air Quality, etc.)

Additional Requirements:

- Develop and run a custom script `custom_exporter.py` that publishes metrics via `prometheus_client`. Update frequency - every **20 seconds**.
- Implement **at least 10 custom metrics** (e.g., temperature, currency rate, pollution level, user activity, number of commits, etc.)
- Create **at least 10 PromQL queries** with mathematical functions, filters, or groupings.
- All metrics must be successfully collected in Prometheus and visualized in Grafana.
- Other requirements follow the **general checklist above.**

Важная информация:

1. Дедлайн — 9-ая неделя (можно сдавать в течение 2 недель после объяснения темы).
2. Если вы не сможете присутствовать по уважительным причинам, можно защитить проект онлайн, но только после согласования в Teams. Без согласования онлайн-защита не допускается.
3. Все выполненные задания необходимо прикреплять в Moodle. Если студент не предоставит ссылку на GitHub в Moodle или не отправит её через Teams, а также не защитит проект, то оценка за задание будет 0 баллов.
4. После каждого выполненного ассайнмента обязательно обновляйте свой репозиторий перед загрузкой работы в Moodle.
5. Скриншоты с заданиями загружать в GitHub запрещено. Защита проводится только через запуск скриптов в реальном времени. Все скрипты должны быть полностью прописаны и сохранены заранее.
6. Также у нас будет система штрафов. За опоздание на 1-2 дня — штраф 5% от остатка, на 3-7 дней — 15% от остатка, на срок до 2 недель — 30% от остатка, свыше 2 недель — 50% от остатка.

Цель

В этом задании вы научитесь работать с Prometheus - системой мониторинга и сбора метрик в реальном времени. В отличие от предыдущего задания с Apache Superset, где вы анализировали данные внутри баз данных, здесь фокус на **мониторинге состояния систем** и сборе метрик из различных источников.

> Обязательно ознакомьтесь с Лекциями 6–7–8 по порядку и прочитайте FAQ.
>
> Там подробно описано, как поднять Prometheus и Grafana пошагово, а также как запускать DB Exporter, Node Exporter и Custom Exporter.

Что мы будем мониторить?

Задания

Общие условия:

Вы должны создать три дашборда:

1. Database Exporter (PostgreSQL/MySQL) - 30 баллов
2. Node Exporter (System Monitoring) - 25 баллов
3. Custom Exporter (External APIs) - 45 баллов

Чтобы получить баллы за конкретный дашборд, необходимо выполнить все пункты чеклиста ниже. Если хотя бы один пункт отсутствует — за данный дашборд выставляется **0 баллов** *(Частичная защита не допускается. Убедитесь, что все необходимые элементы подготовлены заранее.)*

№	Требование	Выполнено
1	Prometheus и Grafana успешно подняты и связаны между собой	+/-
2	Соответствующий Exporter (DB/Node/Custom) запущен и доступен по `http://localhost:port`	+/-
3	Создано не менее 10 **PromQL-запросов** для данного дашборда	+/-
4	Минимум **60% запросов** содержат функции (`avg`, `rate`, `sum`, `count`, `time()` и т.п.) или фильтры по времени (`[5m]`, `by()`, `grouping`)	+/-
5	**Все 10 PromQL-запросов проверены в Prometheus**, они корректно возвращают данные, и продемонстрированы во время защиты	+/-
6	Метрики собирались **в течение 1–5 часов** (допускается имитация нагрузки)	+/-
7	На дашборде **≥10 визуализаций**, из них **4 разных типов** минимум (например, 3-time series, 3-gauge, 1-heatmap, 3-bar)	+/-
8	Настроен **глобальный фильтр** (dashboard variable) для всего дашборда	+/-
9	Все данные отображаются корректно (значения обновляются в реальном времени)	+/-
10	Добавлен хотя бы один alert (правило оповещения) в Grafana с видимым условием срабатывания и текущим статусом.	+/-
11	Экспортирован JSON-дашборд и загружен в GitHub. А также в GitHub присутствуют все необходимые файлы: `docker-compose.yml`, `prometheus.yml`, `custom_exporter.py`, README.md	+/-
12	Во время защиты продемонстрировано: работа контейнеров, статус Targets (все “UP”), выполнение PromQL-запросов и визуализация в Grafana	+/-

(30 баллов) ДАШБОРД 1 — Database Exporter (PostgreSQL / MySQL)

Этот дашборд демонстрирует работу базы данных и внутренние показатели SQL-сервера (нагрузка, размер, активность).

Ключевые метрики для визуализации в Grafana ниже. Для достижения 10 PromQL-запросов по требованию можно дополнить список собственными идеями.

- Количество активных подключений
- Размер базы данных (в байтах, в гигабайтах)
- Время работы (Uptime)
- Интенсивность операций чтения/записи
- Скорость обработки запросов (QPS)
- Количество пользователей и их привилегии
- Общее количество таблиц и строк

(25 баллов) ДАШБОРД 2 — Node Exporter (Системные метрики)

Этот дашборд позволяет наблюдать ресурсы ноутбука или сервера в реальном времени.

Ключевые метрики ниже. Для достижения 10 PromQL-запросов по требованию можно дополнить список собственными идеями.

- CPU usage (по ядрам)
- Load average (1, 5, 15 минут)
- Общая, доступная и используемая память (в GB)
- RAM usage (в процентах)
- Свободное место на диске (в GB)
- Disk I/O — чтение и запись (байт/сек)
- Входящий и исходящий сетевой трафик (в Мбит/сек)
- Температура процессора
- Ёмкость и состояние батареи ноутбука
- **Эксперимент:** baseline → нагрузка → анализ (измерение изменений после нагрузки)
- (дополнительно по желанию) количество активных процессов, системный uptime, swap usage, частота CPU или использование GPU, если доступно

(45 баллов) ДАШБОРД 3 — Custom Exporter (внешние API)

Этот дашборд иллюстрирует сбор и визуализацию внешних данных через Custom Python Exporter.

Вы можете выбрать любой открытый API (OpenWeather, Exchange Rates, GitHub, NASA, Air Quality и т. д.)

Дополнительные требования:

- Разработайте и запустите собственный скрипт `custom_exporter.py`, который публикует метрики через `prometheus_client`. Частота обновления - каждые 20 секунд.
- Реализуйте **не менее 10 кастомных метрик** (например: температура, курс валют, уровень загрязнения, активность пользователей, количество коммитов и т. д.)
- Создайте **не менее 10 PromQL-запросов** с математическими функциями, фильтрами или группировками
- Все метрики должны успешно собираться в Prometheus и корректно отображаться в Grafana
- Остальные пункты требований смотрите в **общем чеклисте**