Министерство науки и образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э.

Баумана

(национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)



**МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ**

**ПО КУРСУ «АДМИНИСТРИРОВАНИЕ БАЗЫ ДАННЫХ»**

**Лабораторная работа №2 «****Введение в мониторинг РСУБД»**

Авторы:

Кудрявцев А.П., [kudryavtsevap@bmstu.ru](mailto:kudryavtsevap@bmstu.ru)

Фомин М.М.

Москва, 2023

## Лабораторная работа №2. Введение в мониторинг РСУБД

Данная лабораторная состоит из 5 взаимосвязанных частей. В первых трех частях лабораторной работы описывается процесс развертывания программ для мониторинга РСУБД PostgreSQL. Далее излагаются теоретические основы мониторинга и начало работы с визуализацией данных в Grafana. После теоретической части предлагаются студентам задания для самостоятельного выполнения в рамках лабораторной работы.

**Цель:**

Данная лабораторная работа призвана сформировать у студента навыки развертывания программ в ОС Linux, основы мониторинга и использования статистики, предоставляемой РСУБД PostgreSQL.

**Задачи:**

* Получить навыки установки программ для мониторинга РСУБД на ОС Linux.
* Ознакомиться с метриками РСУБД PostgreSQL, предоставляемые postgres\_exporter.
* Получить начальные практические основы организации мониторинга с помощью Grafana и Prometheus.

1. Краткая информация о мониторинге

Мониторинг в ИТ сегодня – это система, которая позволяет в режиме реального времени выявлять проблемы в ИТ инфраструктуре, а также оценивать тренды использования ресурсов. Как правило состоит из нескольких базовых компонентов – сбора сырых данных, обработки данных с целью их анализа, рассылки уведомлений и пользовательского интерфейса для просмотра графиков и отчетов. В настоящее время существует большое количество систем для мониторинга различных категорий – сети, серверной инфраструктуры, производительности приложений, реального пользователя, безопасности и др.

Мониторинг — это сбор метрик и представление этих метрик в удобном виде (таблицы, графики, шкалы, уведомления, отчёты). Метрики — это абстракция, с которой мы имеем дело, когда говорим о мониторинге. Это какие-то числа, описывающие состояние интересующей нас системы. Самый простой и понятный мониторинг следит за ресурсами компьютера: загрузкой процессора, памяти, диска, сети. Аналогично можно следить за чем-то более высокоуровневым, вроде количества пользователей БД или времени выполнения транзакций.

В качестве платформы для мониторинга установим Grafana. Grafana является прекрасной системой визуализации и анализа информации, которая позволяет работать с широким спектром источников данных MS SQL, MySQL, PostgreSQL, Prometheus и др. А источником данных будет служить Prometheus.

Рассмотрим подробнее схему взаимодействия компонентов системы мониторинга на основе Grafana/Prometheus. Базовая конфигурация состоит из трех компонентов.

**Экспортер** собирает данные и возвращает их в виде набора метрик. Экспортеры делятся на официальные (написанные командой Prometheus) и неофициальные (написанные разработчиками различного программного обеспечения для интеграции с Prometheus). При необходимости есть возможность писать свои экспортеры и расширять существующие дополнительными метриками. В нашем случае будем использовать postgresql\_exporter, который собирает метрики работы PostgreSQL сервера.

**Prometheus** получает метрики от экспортеров и сохраняет их в БД временных рядов. Поддерживает мощный язык запросов PromQL (Prometheus Query Language) для выборки и агрегации метрик. Позволяет строить простые графики и формировать правила уведомлений (alerts) на основе выражений PromQL для отправки через Alertmanager.

**Grafana** предоставляет средства визуализации и дополнительного анализа информации из Prometheus. Есть примеры Dashboard практически под любые задачи, которые при необходимости можно легко доработать. Создание собственных Dashboard.

1. Развертывание Prometheus

Перейдите на официальный сайт Prometheus и скопируйте файл **prometheus-2.45.2.linux-amd64.tar.gz:**

<https://prometheus.io/download/#prometheus/>

Войдите с правами root:

**sudo -i**

В дальнейшем предстоит создавать папки и, если предпочитаете работать в графической оболочке, а не командной строке, то установите nautilus-admin, позволяющий работать с правами администратора:

**sudo apt install nautilus-admin**

Откройте nautilus (проводник) и в папке Downloads найдите скопированный файл **prometheus-2.45.2.linux-amd64.tar.gz** и распакуйте архив. Это можно сделать помощью nautilus, нажав правую клавишу мышки и выбрав один из вариантов Extract, либо с помощью командной строки:

**tar -xvzf prometheus-2.45.2.linux-amd64.tar.gz prometheus-2.45.2.linux-amd64/**

Скопируйте исполняемые файлы в **prometheus** и **promtool** в **/usr/local/bin/**.

Следующим шагом, если работаете с nautilus, создайте папки:

/etc/prometheus/

/etc/prometheus/consoles/

/etc/prometheus/console\_libraries/

И скопируйте файлы из соответствующих папок в  
**Downloads/prometheus-2.45.2.linux-amd64/.**

При работе с терминалом совместим создание папок и копирование:

cd /home/<user>/Downloads/prometheus-2.45.2.linux-amd64/

cp -r consoles/ /etc/prometheus/

cp -r console\_libraries/ /etc/prometheus/

cp prometheus.yml /etc/prometheus/

Создайте папку **/var/lib/prometheus** для хранения данных:

mkdir /var/lib/prometheus

Создайте пользователя без права подключения **prometheus** и назначьте владельцем файлов и папок:

useradd -M -r -s /bin/nologin prometheus

chown -R prometheus:prometheus /etc/prometheus /var/lib/prometheus

Проверьте права папок **/etc/prometheus** и **/var/lib/prometheus.**

Создайте systemd-юнит, чтобы управлять сервисом:

nano /etc/systemd/system/prometheus.service

Редактор создаст файл, скопируйте в файл конфигурацию сервиса Prometheus текст:

[Unit]

Description=Prometheus systemd service unit

Wants=network-online.target

After=network-online.target

[Service]

Type=simple

User=prometheus

Group=prometheus

ExecReload=/bin/kill -HUP $MAINPID

ExecStart=/usr/local/bin/prometheus \

--config.file=/etc/prometheus/prometheus.yml \

--storage.tsdb.path=/var/lib/prometheus \

--web.console.templates=/etc/prometheus/consoles \

--web.console.libraries=/etc/prometheus/console\_libraries \

--web.listen-address=0.0.0.0:9090

SyslogIdentifier=prometheus

Restart=always

[Install]

WantedBy=multi-user.target

Сохраните изменения в файле и выйдите.

Обновите список юнитов:

systemctl daemon-reload

Запустите Prometheus:

systemctl start prometheus.service

Проверьте запустилась ли служба:

systemctl status prometheus.service

…

● prometheus.service - Prometheus systemd service unit

Loaded: loaded (/etc/systemd/system/prometheus.service; enabled; vendor preset: enabled)

Active: active (running) since Mon 2023-12-25 08:34:15 MSK; 29min ago

Main PID: 1005 (prometheus)

…

Для автоматической загрузки Prometheus введите:

systemctl enable prometheus.service

1. Развертывание postgres\_exporter

Перейдите на официальный сайт Prometheus <https://github.com/prometheus-community/postgres_exporter/releases/> и скопируйте файл **postgres\_exporter-0.15.0.linux-amd64.tar.gz** в папку Downloads.

Войдите с правами root:

**sudo -i**

Перейдем в папку Downloads:

cd /home/<user>/Downloads/

Распакуем файл в папку **/opt/**:

**tar xzf postgres\_exporter-0.15.0.linux-amd64.tar.gz -C /opt/**

Назначаем необходимые для работы права:

**chown -R postgres:postgres /opt/postgres\_exporter-0.15.0.linux-amd64**

Создадим файл конфигурации сервиса

**nano /etc/systemd/system/postgres\_exporter.service**

со следующим содержимым:

[Unit]

Description=Prometheus PostgreSQL Exporter

After=network.target

[Service]

Type=simple

Restart=always

User=postgres

Group=postgres

Environment=DATA\_SOURCE\_NAME="user=postgres host=/var/run/postgresql sslmode=disable"

#Environment=DATA\_SOURCE\_NAME="user=postgres port=5432 dbname=demo host=/var/run/postgresql sslmode=disable"

#Environment=PG\_EXPORTER\_EXTEND\_QUERY\_PATH=/opt/postgres\_exporter-0.15.0.linux-amd64/queries.yaml

ExecStart=/opt/postgres\_exporter-0.15.0.linux-amd64/postgres\_exporter

[Install]

WantedBy=multi-user.target

Устанавливаем и запускаем сервис:

systemctl daemon-reload

systemctl enable postgres\_exporter.service

systemctl start postgres\_exporter.service

Проверяем статус:

systemctl status postgres\_exporter.service

При необходимости открываем порт на межсетевом экране:

ufw allow 9187/tcp

Настройка Prometheus для получения данных postgres\_exporter

Открываем на редактирование файл конфигурации prometheus:

nano /opt/configs/prometheus.yml

в секции **scrape\_configs** добавляем (важно: соблюдать отступы – каждый отступ два пробела):

scrape\_configs:

# The job name is added as a label `job=<job\_name>` to any timeseries scraped from this config.

- job\_name: 'prometheus'

scrape\_interval: 5s

static\_configs:

- targets: ['localhost:9090']

- job\_name: 'postgresql'

scrape\_interval: 5s

static\_configs:

- targets: ['localhost:9187']

Перезапустите Prometheus:

systemctl restart prometheus.service

Проверьте статус:

systemctl status prometheus.service

1. Развертывание Grafana

Скачайте файл **grafana-enterprise\_10.2.3\_amd64.deb** с применением VPN:

<https://grafana.com/grafana/download?pg=get&platform=linux&plcmt=selfmanaged-box1-cta1>

Если не удается скопировать, то это можно сделать с сайта кафедры.

Перейдите в папку Downloads:

cd /home/<user>/Downloads/

Инсталлируйте:

dpkg -i grafana-enterprise\_10.2.3\_amd64.deb

Запустите сервис:

systemctl daemon-reload

systemctl enable grafana-server

systemctl start grafana-server

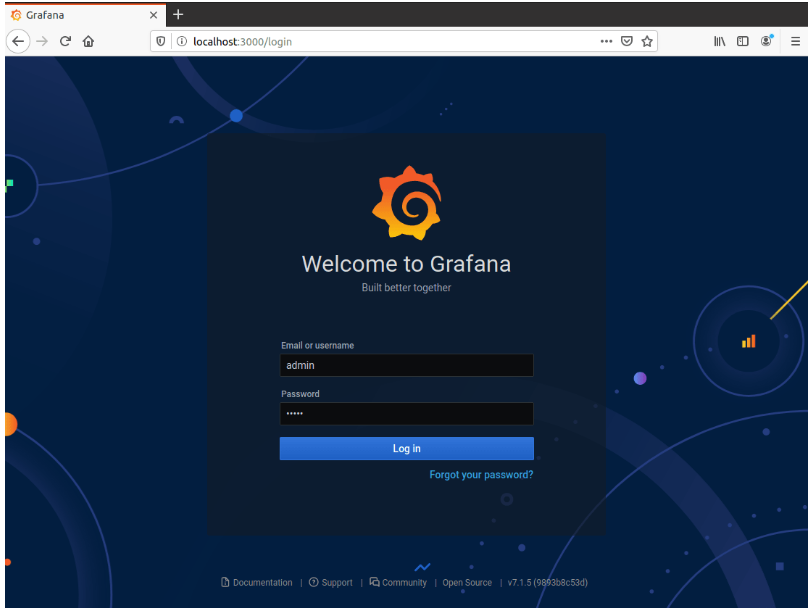
systemctl start grafana-server

Чтобы установить соединение, создайте правило для порта 3000:

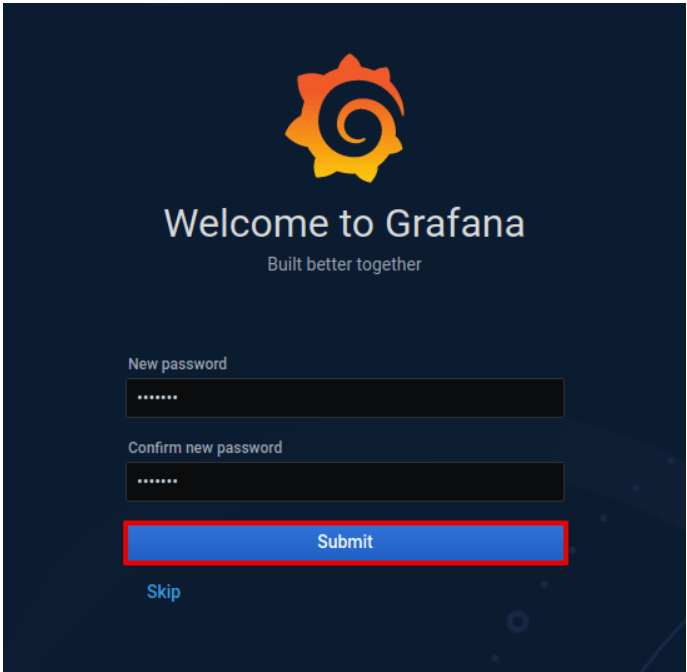
ufw allow 3000/tcp

4.1. Регистрация

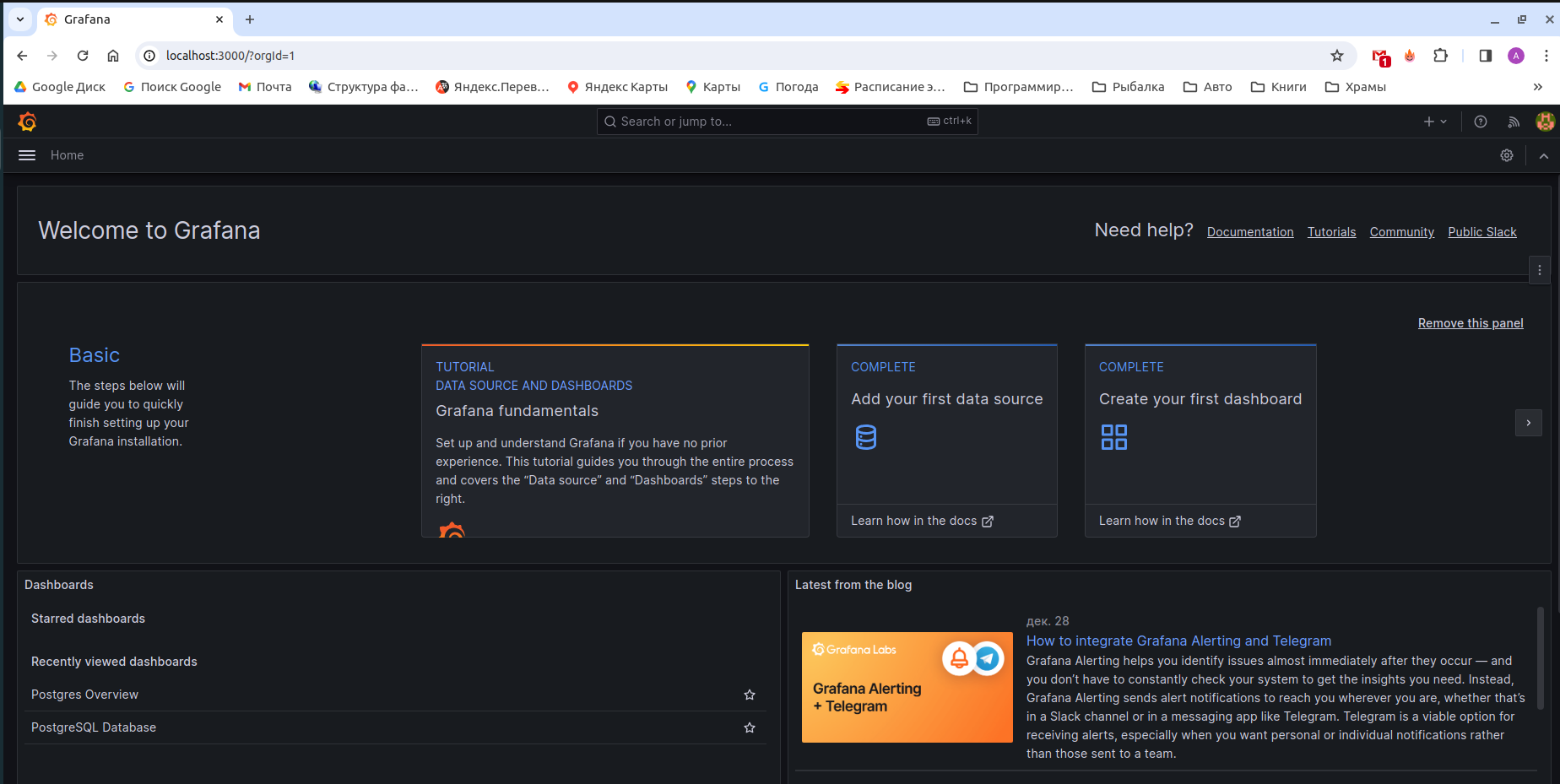
Откройте браузер и адресной строке наберите http://localhost:3000. Откроется страница:



По умолчанию имя пользователя и пароль **admin**. После ввода предложат изменить пароль:



После завершения формальных процедур войдите на домашнюю страницу.

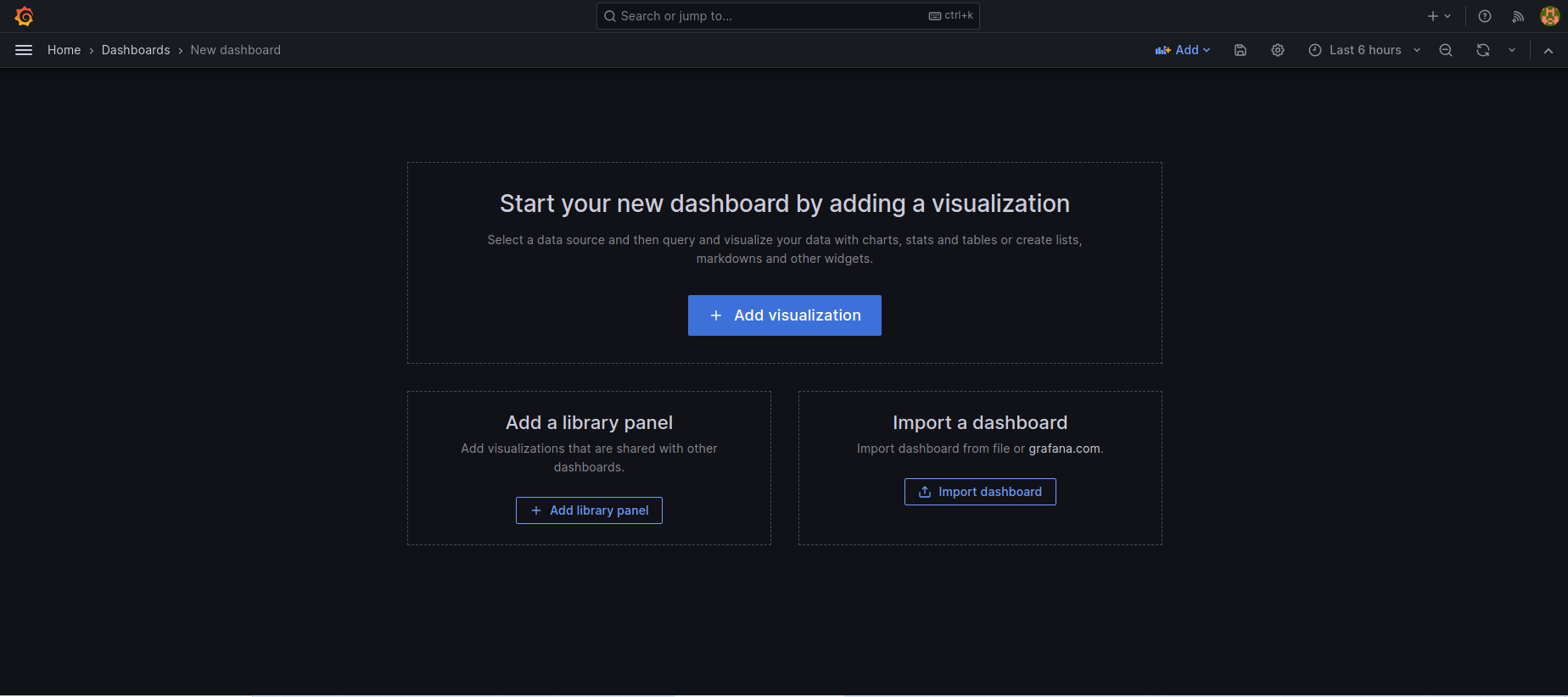


4.2. Знакомство с Grafana

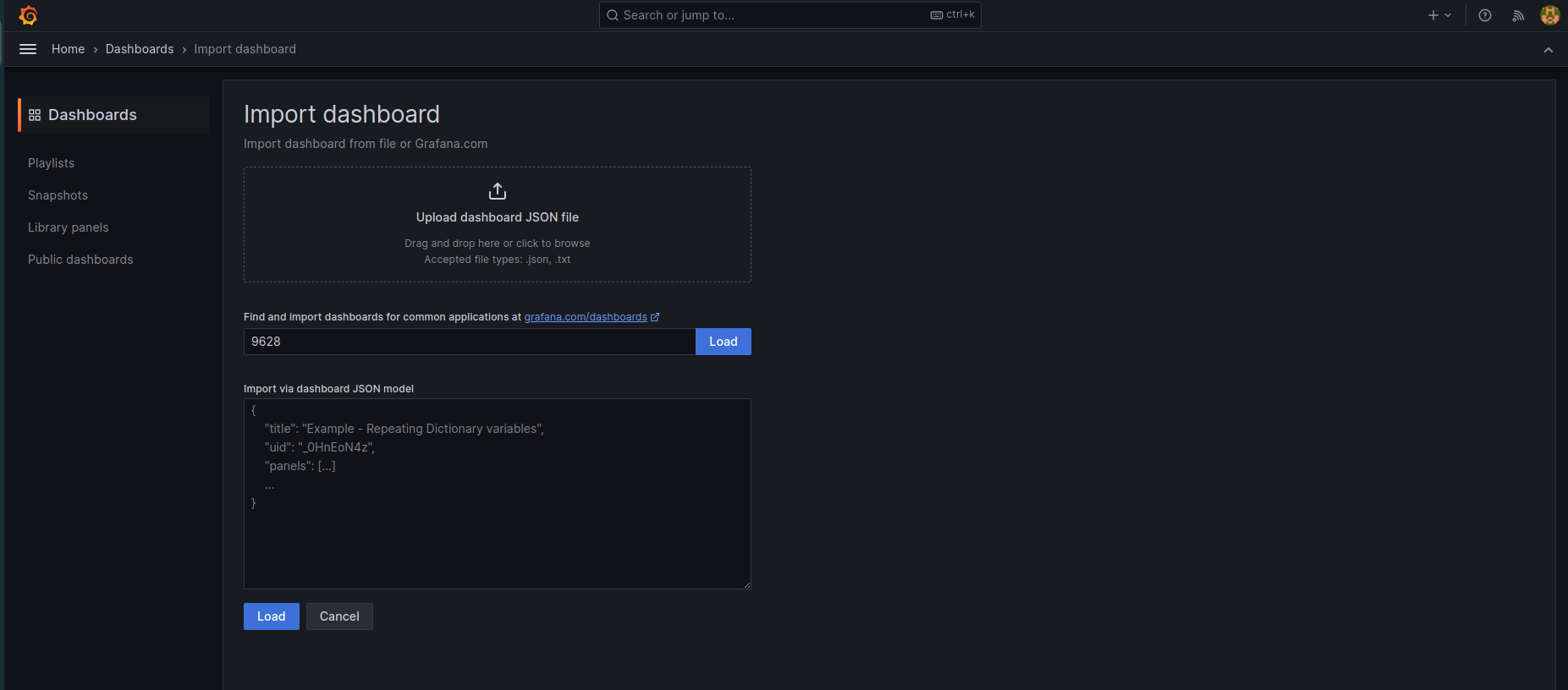
Для Prometheus и Grafana многие пользователи разработали свои Dashboard и разместили их на сайте Grafana. Для получения первого впечатления от возможностей этих программ загрузим эти Dashboard, тем более, что это делается достаточно просто (нас интересуют те Dashboard, которые обрабатывают метрики PostgreSQL). Обращу внимание, что метрики, отображаемые в Dashboard, снимаются с вашего сервера PostgreSQL, и чем активнее он работает, тем динамичнее изменяются метрики.

Первым шагом надо добавить источник данных – это Prometheus. Порядок действий описан в п. 6.1.

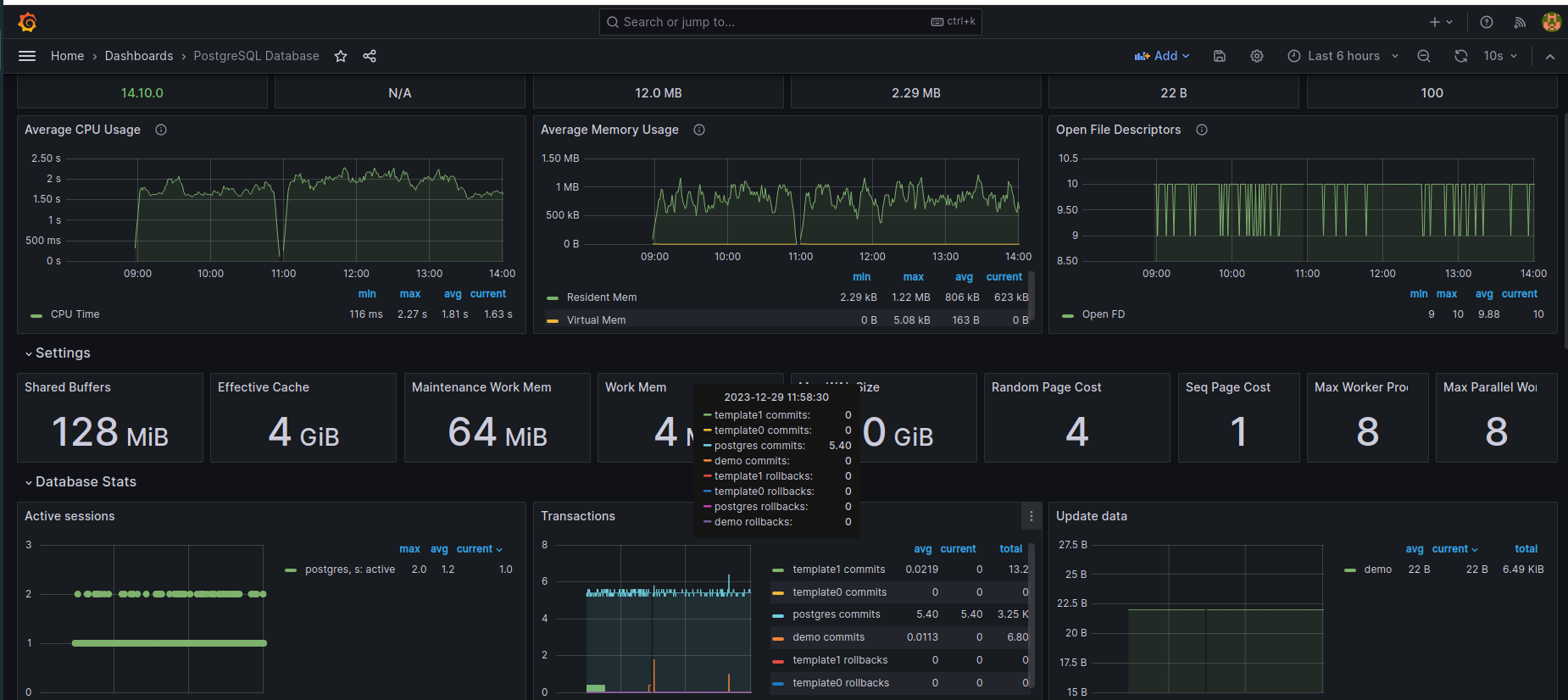
На домашней странице «**Create your first dashboard»**. Выходим на страницу выбора вариантов создания Dashboard.



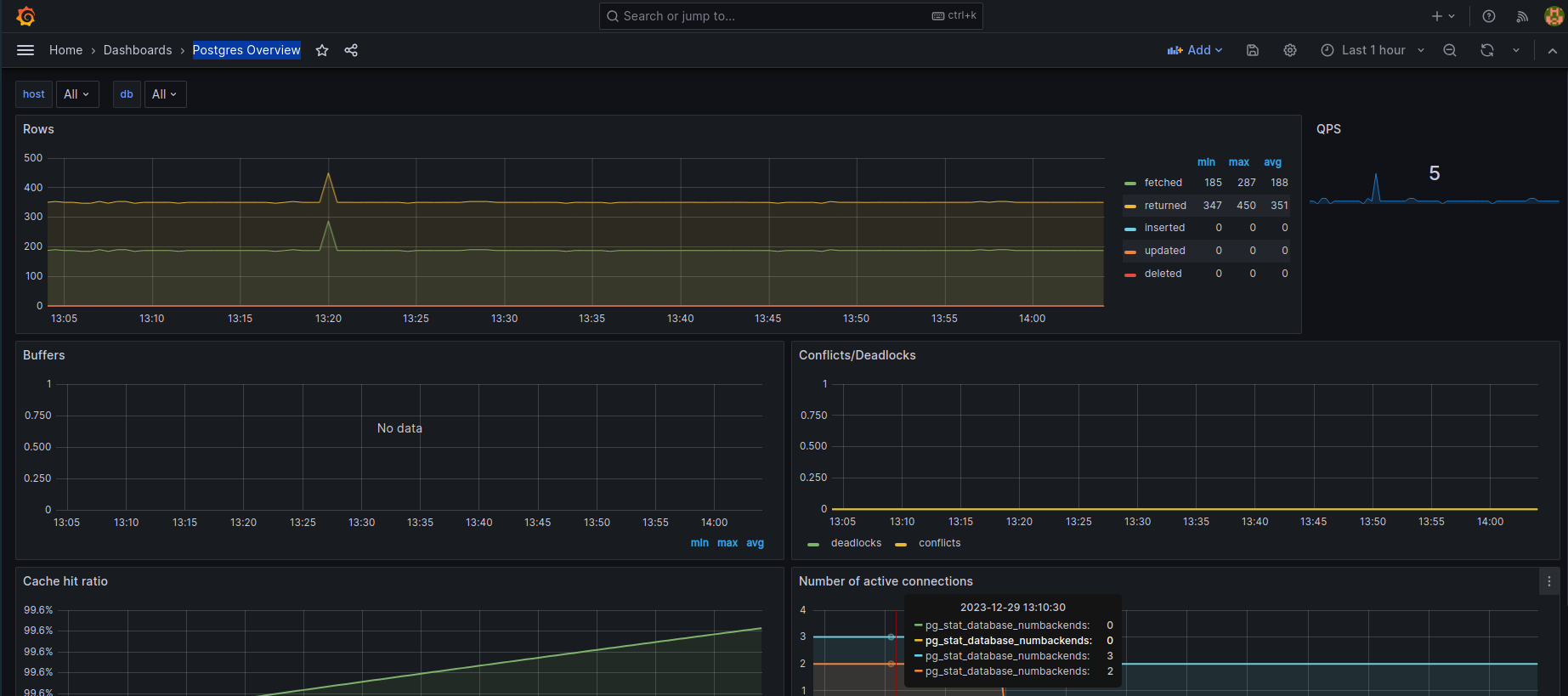
Будем импортировать, нажимаем на кнопку «**Import dashboard**».



И в строку вводим ID 9628 и нажимаем «**Load**». Отобразится Dashboard «PostgreSQL Database» c несколькими панелями, каждая из которых отображает одну или несколько метрик PostgreSQL. Сохраните Dashboard, будет полезно посмотреть на возможности и способы реализации.



Повторим импорт Dashboard, но с ID равным 455. Загрузится «Postgres Overview».



1. Метрики

Метрики можно разделить на две группы: это метрики, которые формируются в postgres\_exporter, так называемые по умолчанию (default), и пользовательские.

Метрики по умолчанию — это некоторый набор метрик для мониторинга, сбор которых заложен непосредственно в коде. Сбор метрик по умолчанию, можно отключить, задав параметры командной строки --disable-default-metrics или определив переменные окружения PG\_EXPORTER\_DISABLE\_DEFAULT\_METRICS.

Для просмотра метрик надо войти на хост <http://localhost:9187/metrics>.

По умолчанию в мониторинг передаются метрики из следующих представлений:

* pg\_stat\_bgwriter;
* pg\_stat\_database;
* pg\_stat\_database\_conflicts;
* pg\_locks;
* pg\_stat\_replication;
* pg\_stat\_activity;
* pg\_settings.

Из того, на что стоит обратить внимание, это - pg\_stat\_replication и pg\_stat\_activity, так как набор возвращаемых полей зависит от версии PostgreSQL. Для этого, в экспортере производится проверка версии СУБД и выбирается соответствующий запрос. Указывать версию в пользовательских запросах, нельзя. Предполагается, что администратор заранее знает, какой версии экземпляр будет мониторится. Так же могут возникнуть проблемы, если пытаться собирать метрики одним экспортером с экземпляров различных версий.

Рассмотрим вывод экспортера. Для примера возьмем две метрики: pg\_settings\_shared\_buffers и pg\_settings\_wal\_sender\_timeout. В выводе экспортера на каждую метрику мы получаем три строки.

# HELP pg\_settings\_shared\_buffers\_bytes Sets the number of shared memory buffers used   
 by the server. [Units converted to bytes.]

# TYPE pg\_settings\_shared\_buffers\_bytes gauge

pg\_settings\_shared\_buffers\_bytes{server="/var/run/postgresql/:5432"} 1.34217728e+08

Первая строка — представляет подсказку и состоит из имени метрики в postgres\_exporter, описания (колонка short\_desc представления pg\_settings) и, если происходило приведение к базовому типу, указывается к какому именно (значение в квадратных скобках).

Во второй строке указывается тип значения в терминах **Prometheus**. И третья строка - метрика с набором меток и присвоенным значением.

**pg\_settings\_shared\_buffers\_bytes** — имя метрики. Оно, по правилам хорошего тона, составляется из имени таблицы, имени метрики и единицы измерения. Далее идет короткое описание из таблицы pg\_settings. Выражение Units converted to bytes означает то, что значении метрики было приведено к единице измерения байты. На последнее стоит обратить внимание, потому что непосредственно в базе значение выглядит несколько иначе, а именно:

name | setting | unit | short\_desc

----------------+---------+------+------------------------------------------------------------

shared\_buffers | 16384 | 8kB | Sets the number of shared memory buffers used by the server

Тип метрики у нас GAUGE, это значит она может принимать произвольные значения в течении времени. В нашем случае количество буферов, при подстройке, может меняется в обе стороны. О типах метрик и их отличиях будет изложено ниже.

Следующая метрика **pg\_settings\_wal\_sender\_timeout**. Информационные поля собираются по тому же принципу, что описан выше. Только в этом случае значение метрики приводится к секундам, а в базе хранится в миллисекундах. С этой особенностью нужно быть внимательными и учитывать при построении графиков.

# HELP pg\_settings\_wal\_sender\_timeout\_seconds Sets the maximum time to wait for WAL

replication. [Units converted to seconds.]

# TYPE pg\_settings\_wal\_sender\_timeout\_seconds gauge

pg\_settings\_wal\_sender\_timeout\_seconds{server="127.0.0.1:5432"} 60

Значение, хранимое в базе данных:

demo=# SELECT name, setting, COALESCE(unit, ''), short\_desc, vartype

FROM pg\_settings

WHERE name='wal\_sender\_timeout';

name | setting | coalesce | short\_desc | vartype

--------------------+---------+----------+----------------------------------------------------+---------

wal\_sender\_timeout | 60000 | ms | Sets the maximum time to wait for WAL replication. | integer

Все значения имеющие единицы измерения приводятся к двум типам: метрики объема к байтам; метрики времени к секундам.

1. Grafana

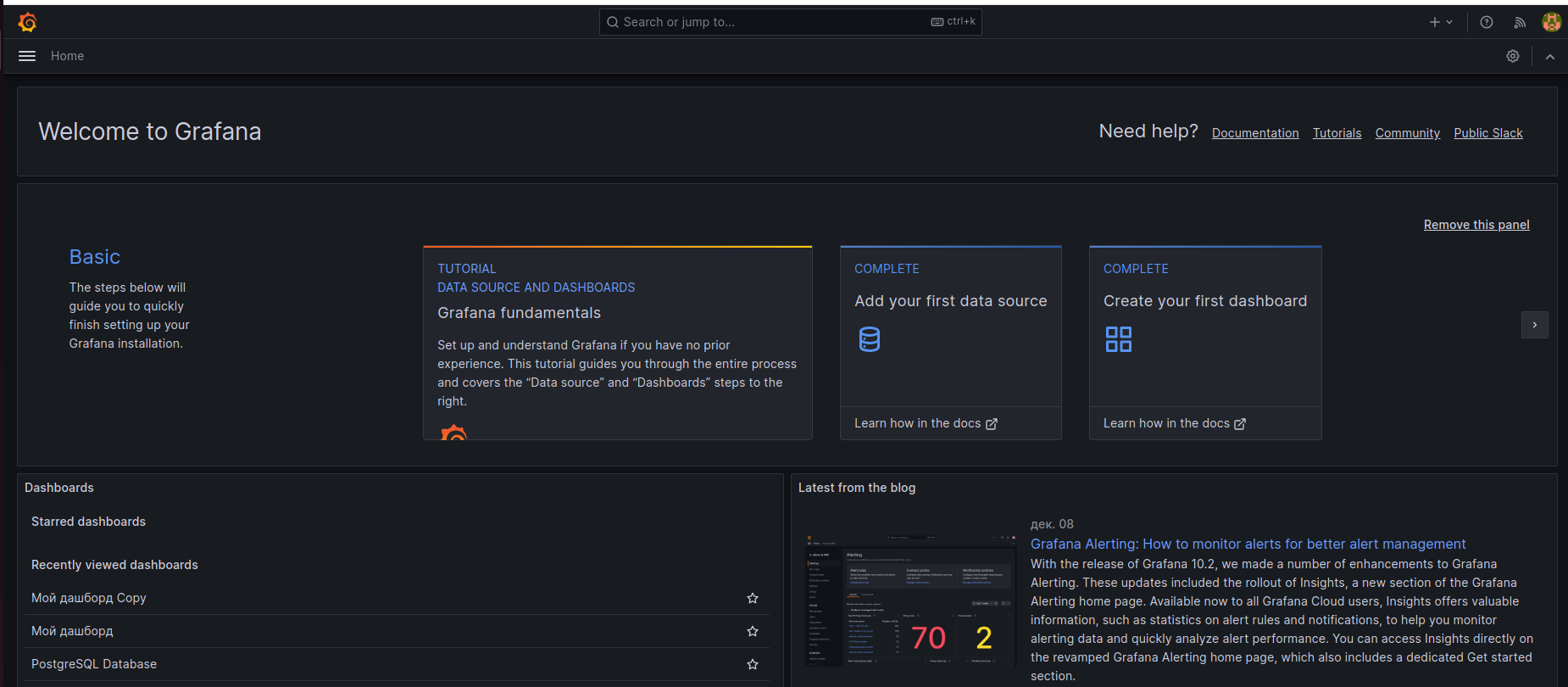
[Программное обеспечение Grafana с открытым исходным кодом](https://grafana.com/oss/) (OSS) позволяет вам запрашивать, визуализировать, оповещать и исследовать ваши метрики, журналы и трассировки, где бы они ни хранились. Grafana OSS предоставляет вам инструменты для представления временных рядов (TSDB) метрик базы данных в подробные графики, удобные для просмотра и быстрой оценки результата. Платформа плагинов Grafana OSS также позволяет подключать источники данных, такие как базы данных NoSQL/SQL, Prometheus и много других.

6.1. Выбор источника данных

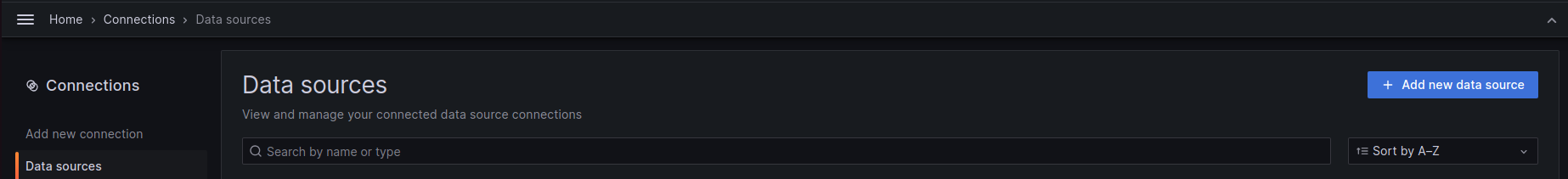
Grafana имеет встроенную поддержку многих источников данных. Если нужны другие источники данных, вы также можете установить один из множества плагинов источников данных. Если нужный вам плагин не существует, вы можете разработать собственный плагин.

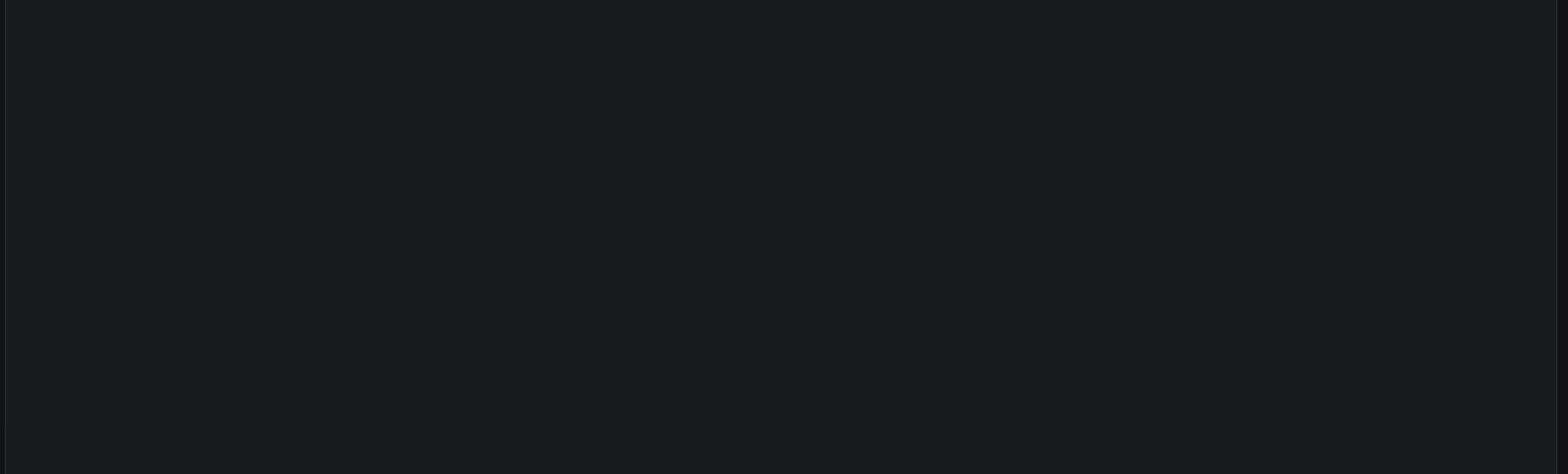
Каждый источник данных поставляется с редактором запросов, который формулирует пользовательские запросы в соответствии со структурой источника. После добавления и настройки источника данных его можно использовать в качестве входных данных для многих операций.

Перейдите в браузере по адресу <http://localhost:3000/>.

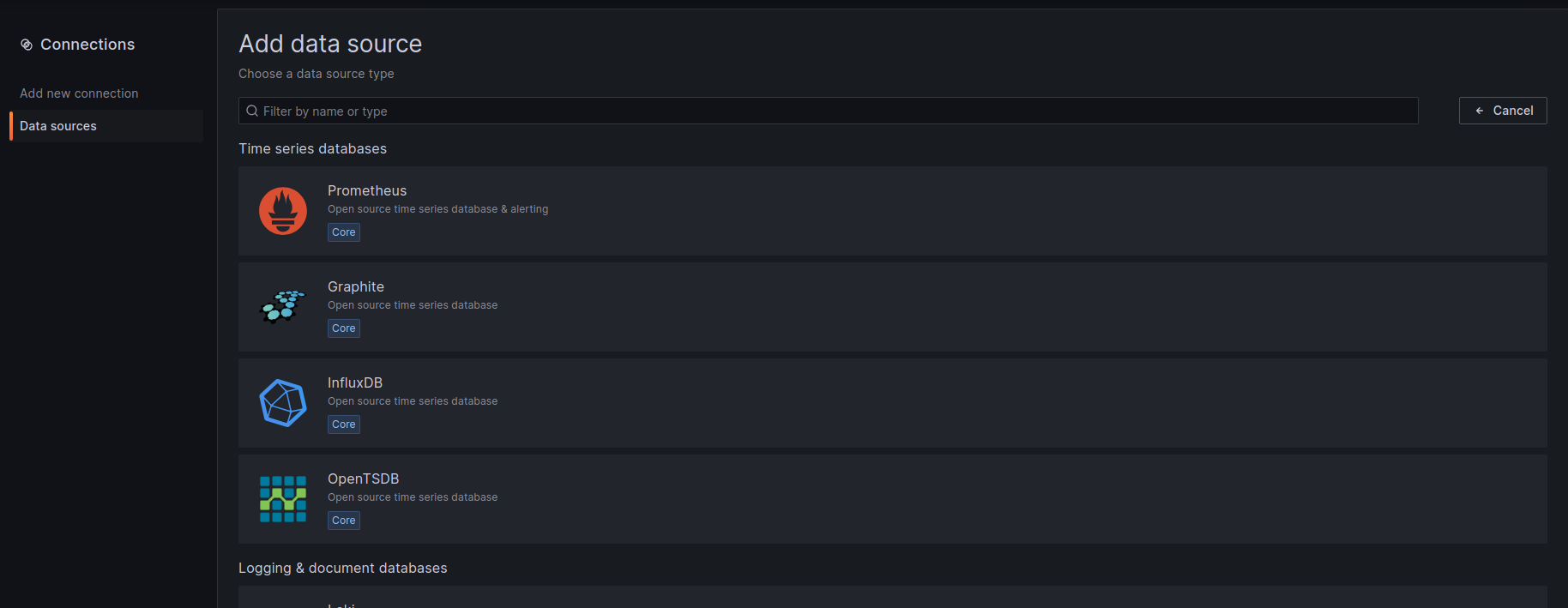


Добавить источник данных можно несколькими способами. На домашней странице нажать на **Add your first data source**. Или через меню. Что и сделаем. Войдем в меню, верхний левый угол. Далее – Connection и Data Sources (источник данных). Откроется страница.

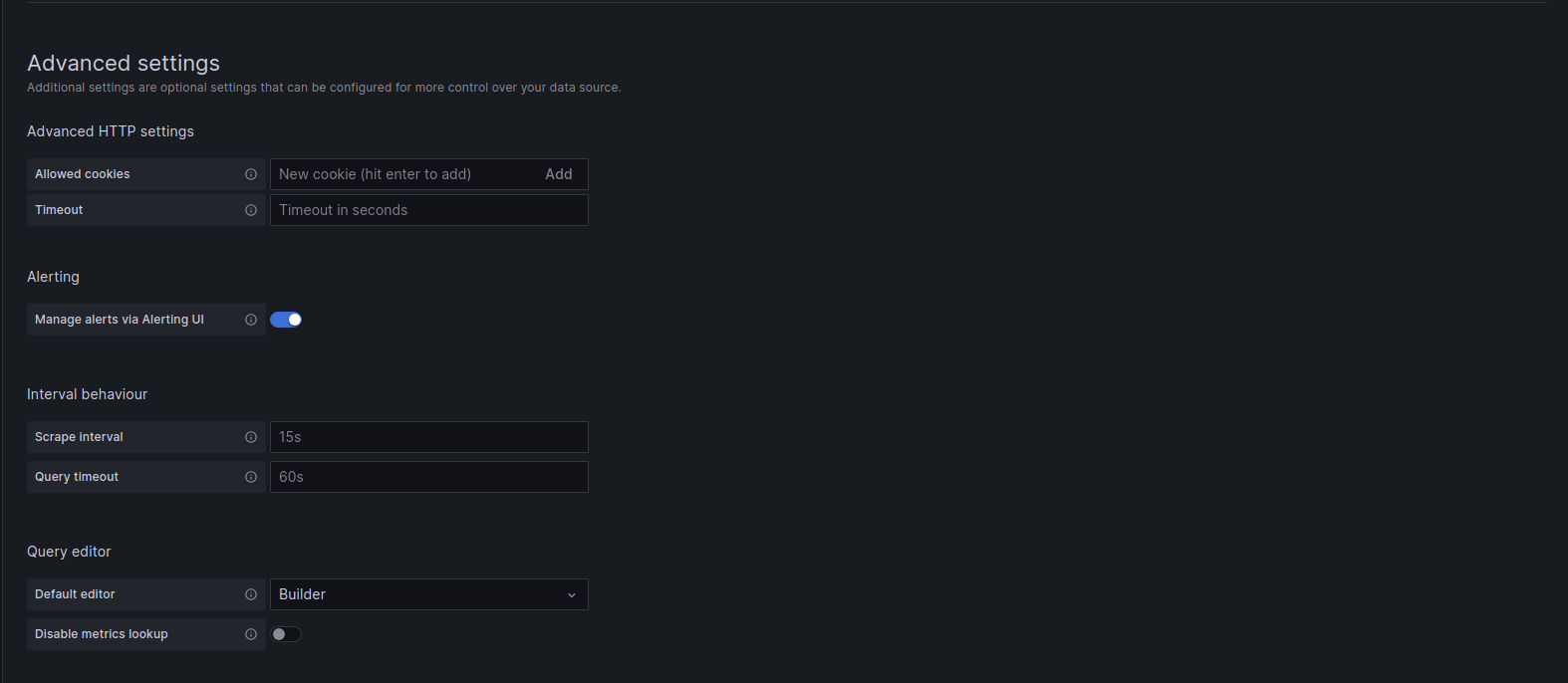
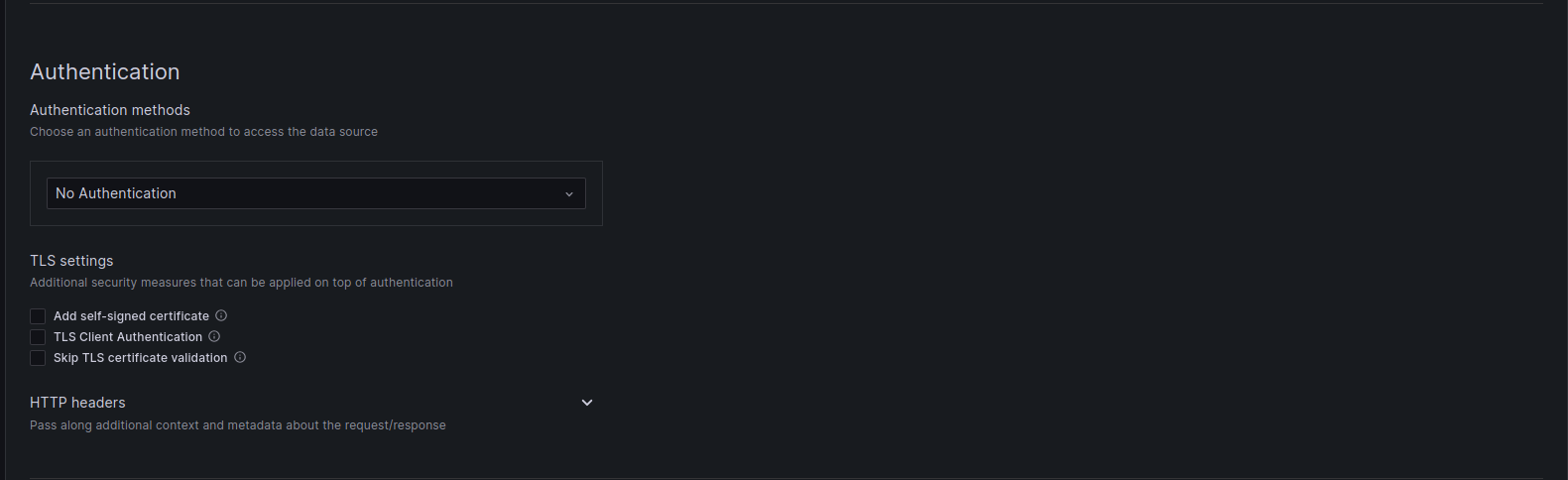
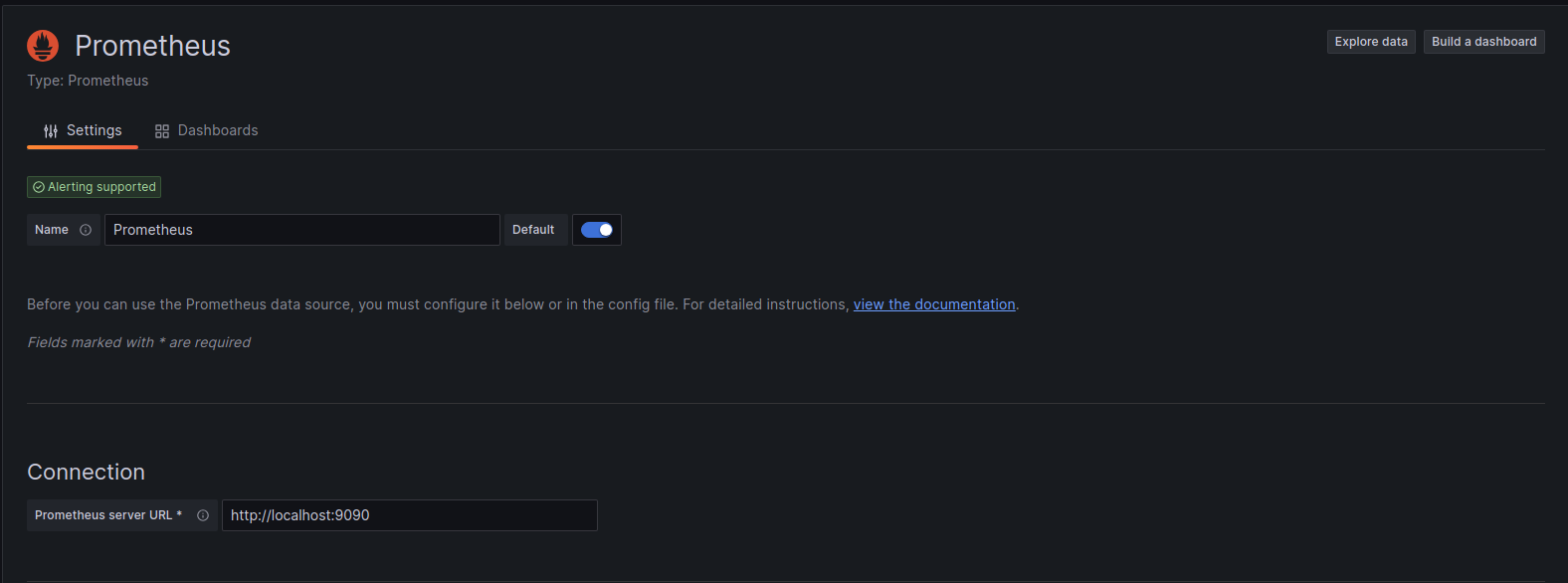


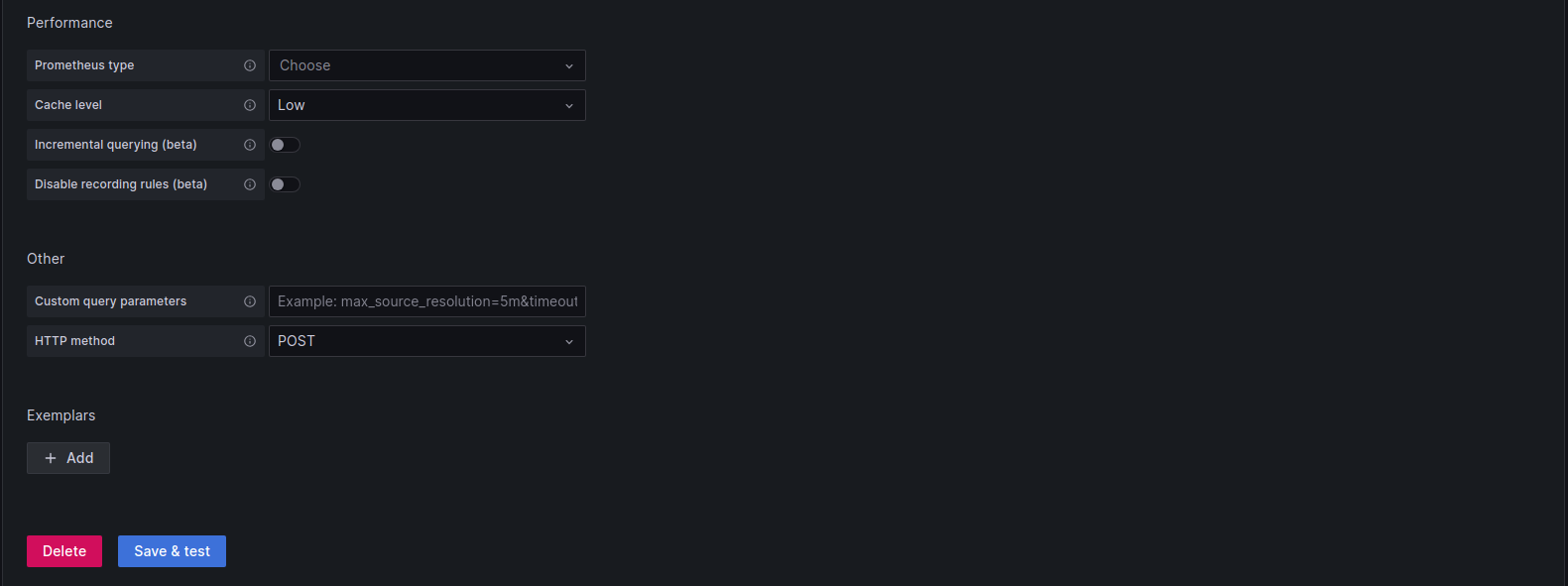


Источников данных, естественно, нет, добавим первый. Нажимаем кнопку **Add new data source.** Открывается страница выбора источника данных.



Выбираем в качестве источника данных **Prometheus.** После этого попадаем на страницу настройки источника данных **Prometheus.**



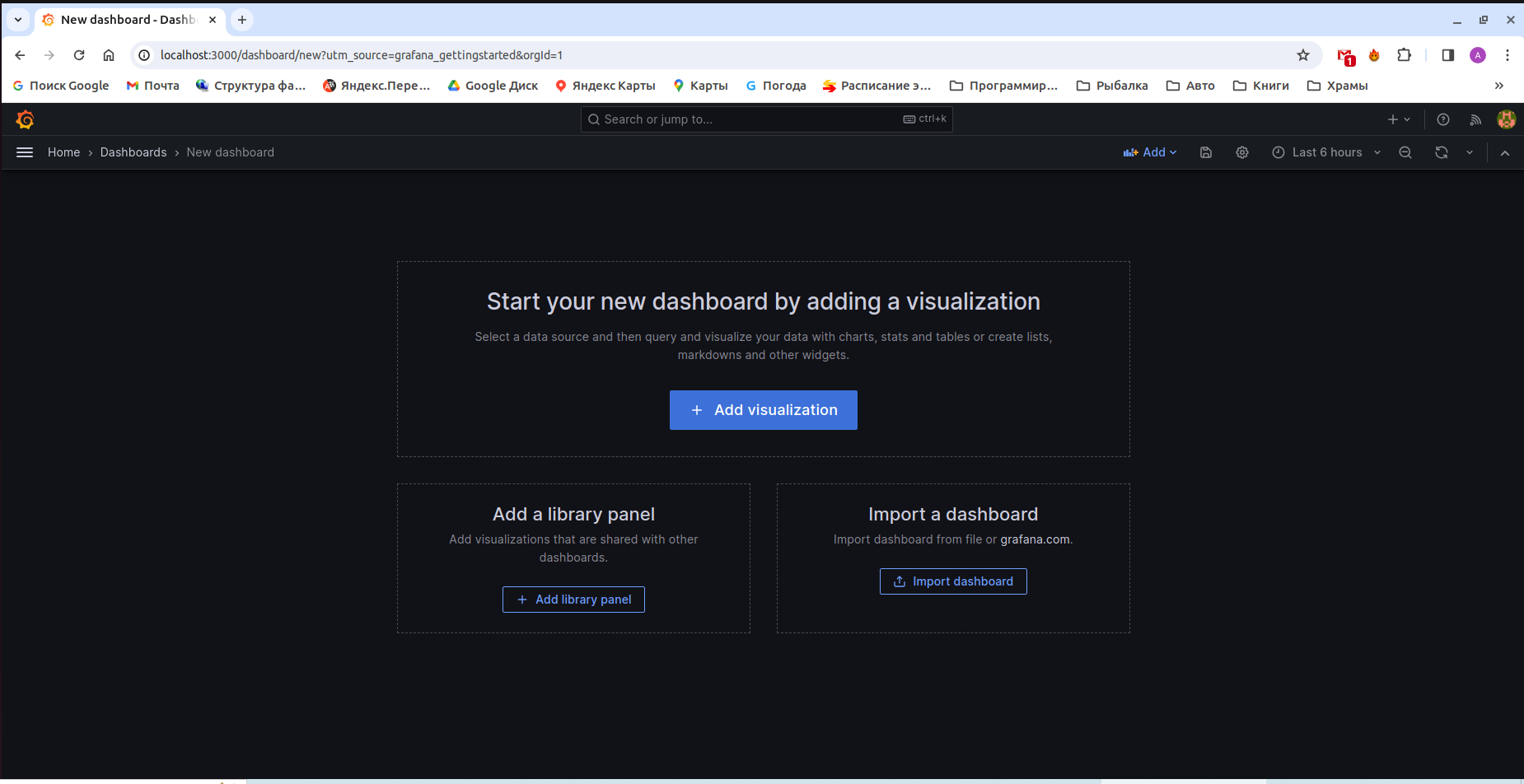
ы

Оставьте значения по умолчанию, кроме поля **Connection**, хотя там и введено требуемое значение http://localhost:9090, но это является подсказкой, и этот адрес надо ввести. И завершается выбор источника данных нажатием кнопки **Save & test**. После подтверждения об успешном прохождения тестирования перейти на станицу **Data Sources** и убедиться, что источник данных сохранен. Обратите внимание на кнопку **Delete.** Если ваш источник данных по каким- либо причинам не устраивает, то его можно удалить.

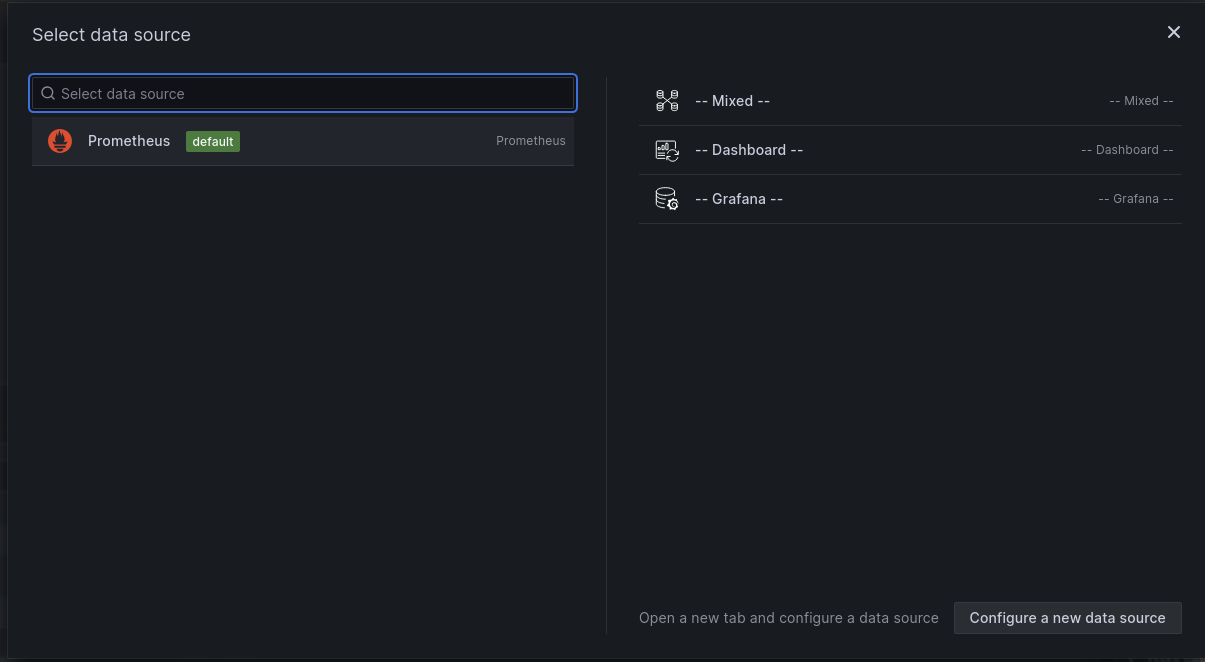
6.2. Создать панель (Dashboard) мониторинга

Панели – это прямоугольные области, содержащие графики, таблицы или другую визуальную информацию. Остальные панели (после первой) можно добавлять щелчком по кнопке Add visualization (название кнопок и их расположение может меняться от версии к версии). Панели организованы в виде сетки и могут переупорядочиваться перетаскиванием мышью.

Перейдем в браузере по адресу http://localhost:3000/ и щелкните по ссылке **Create your first dashboard**. В ответ откроется страница, как показано на рис.

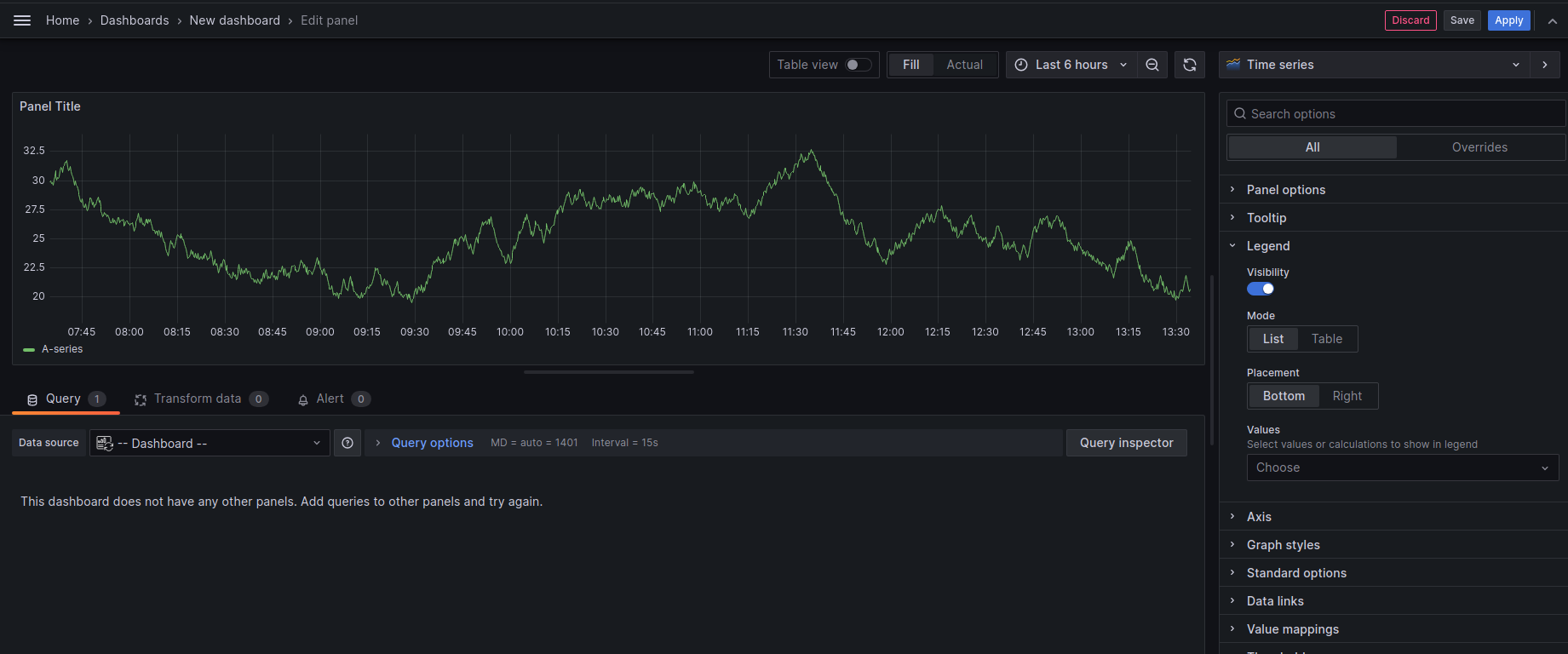


Здесь можно щелкнуть по ссылке **Add visualization**.



Источник данных выбрался Prometheus, в качестве панели выберем –Dashboard--.

И откроется редактор Dashboard.



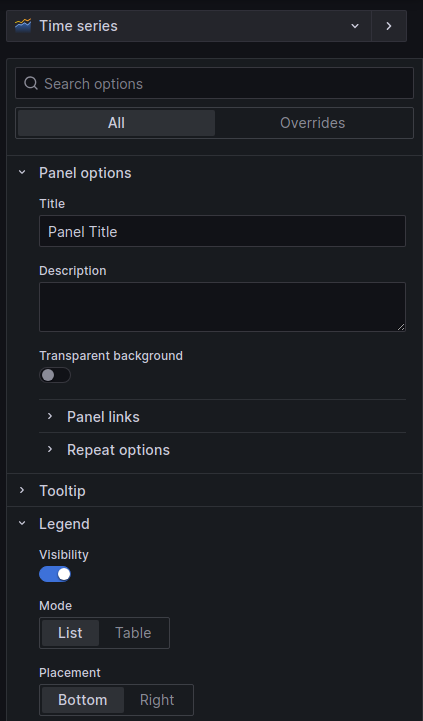
Не пугайтесь. График – это подсказка, говорящая о том, что выбрано представление данных в виде **Time series** (временных рядов), что подтверждается надписью в районе правого верхнего угла. Тут же можно изменить представление, выбрав из списка другое, нажав на стрелку combobox.

Нажмем на кнопку обновить в виде двух полуокружностей и получим начальный вид Dashboard.

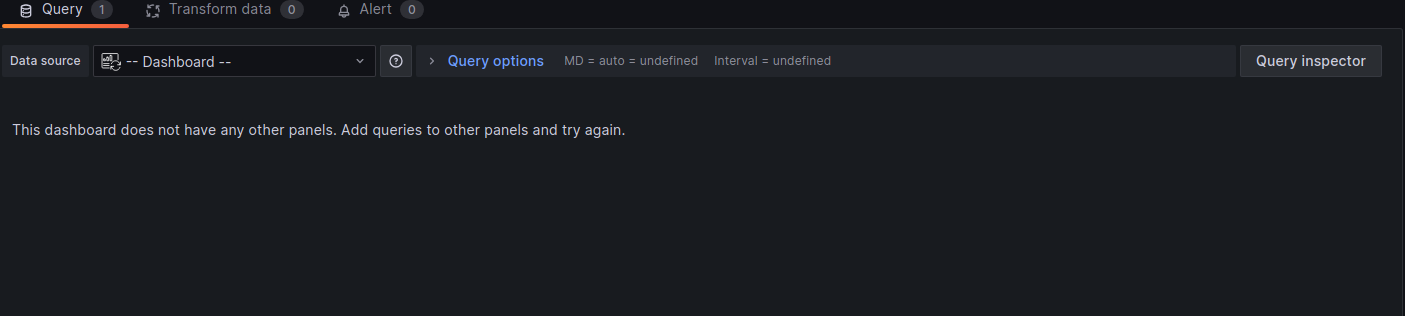


Сначала назовем Dashboard, чтобы в дальнейшем было проще его находить.

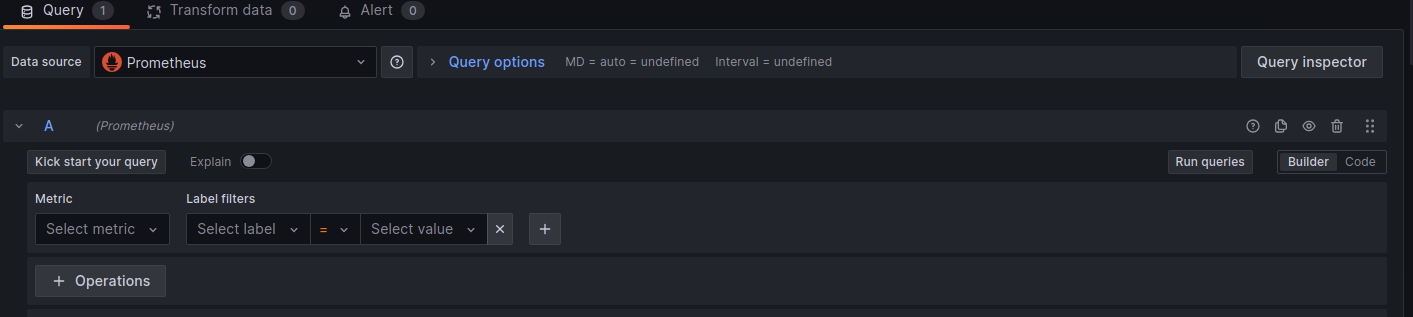
Для этого в редакторе панелей в секции **Panel options** в поле **Title** введем заголовок Dashboard, а ниже в поле **Description** описание.



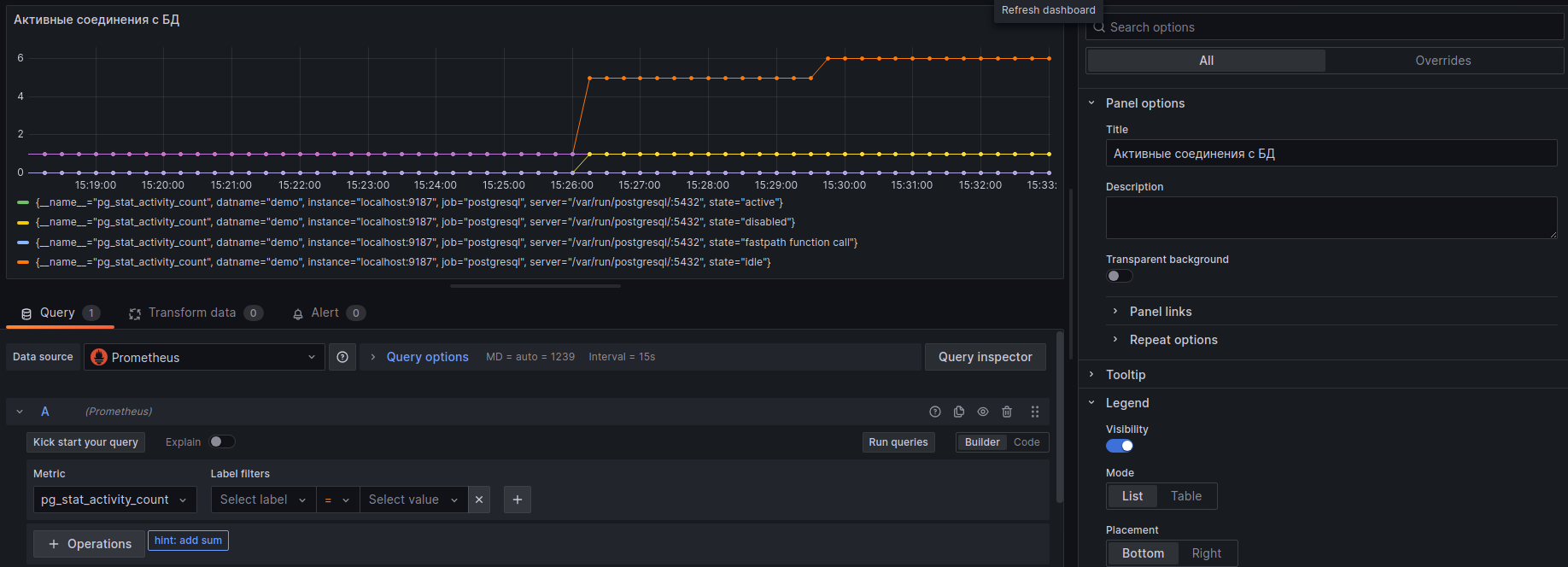
В панели Query в нижней части экрана выберите **Data source** Prometheus.



После этого откроется панель, содержащая поля для ввода и манипуляции с метриками.



В поле **Metrics** выберем метрику pg\_stat\_activity\_count (количество активных соединений). И после нажатия кнопки Run queries панель «заживет».



Если графиков нет, то необходимо сохранить Dashboard, нажав кнопку **Save** и введя название Dashboard. Затем остается только нажать кнопку **Apply**. Редактор закроется и отобразится актуальная страница Dashboard. В районе правого верхнего угла измените интервал времени с 6 часов на меньшее, скажем на 15 мин, а время авто обновления на 10 сек.

Откройте несколько соединений c помощью клиента PostgreSQL (psql/pgAdmin/DBeaver) и проследите за изменением графика.

Для возвращения в режим редактирования нажмите на значок меню и выберите команду Edit.



## Практическое задание

* Разобраться с панелью **Panel options** на странице редактирования Dashboard.
* Выбрать любые три метрики по умолчанию, разобраться в их назначении и создать для них три Dashboard. Протестировать их с использованием с использованием утилиты pgbench (<https://postgrespro.ru/docs/postgrespro/14/pgbench>).

**Пример.**

* Инициализация (достаточно выполнить один раз):  
  pgbench -i -U <пользователь> -d demo  
  Создаются четыре тестовые таблицы в БД demo pgbench\_...
* Тест:  
  pgbench -T 1200 -c 30 -U <пользователь> -d demo  
  -T 1200 – время теста в секундах (20 мин);  
  -c 30 – количество клиентов.

## Список использованных источников

1. Grafana Labs.   
   <https://grafana.com/>
2. Prometheus  
   <https://prometheus.io/>
3. Postgres\_exporter  
   <https://github.com/prometheus-community/postgres_exporter>