

Главная страница » Запуск Prometheus в Docker

Запуск Prometheus в Docker

автор: [Александр](#) / 01.11.2024



[Prometheus](#) — это приложение для мониторинга с открытым исходным кодом. Оно сканирует HTTP-конечные точки для сбора показателей в простом текстовом формате, охватывает множество аспектов мониторинга, таких как генерация и сбор показателей, построение графиков на информационных панелях и оповещение об аномалиях. С недавнего времени, Prometheus умеет работать с [Telegram](#) из коробки. Об этом ниже.

В этом руководстве мы узнаем, как установить три ключевых компонента для использования Prometheus в Docker. Это:

- Сервер Prometheus для сбора показателей и запроса к ним;
- Node Exporter для экспорта системных показателей в формате, совместимом с Prometheus;
- Grafana — это веб-приложение для создания графических информационных панелей, которое поддерживает Prometheus и другие серверные части.

Подразумевается, что у нас уже установлен Docker и docker-compose, в противном случае, можно воспользоваться инструкцией [Установка Docker на VPS/VDS](#).



1. Запуск Prometheus + Node Exporter в Docker
2. Запуск Grafana в Docker
3. Добавление источника данных Prometheus в Grafana

Запуск Prometheus + Node Exporter в Docker

Node Exporter — это небольшое приложение, которое собирает метрики операционной системы и предоставляет к ним доступ по HTTP.

Node Exporter измеряет различные метрики, такие как:

- общая память (RAM);
- используемая память (RAM);
- кэш памяти (RAM);
- свободный диск;
- пространство для IOPS;
- монтирования;
- загрузка процессора;
- сеть (трафик, поток TCP, соединения).

Node Exporter устанавливается на всех серверах или виртуальных машинах для сбора данных обо всех узлах. По умолчанию он выставляет метрики в по адресу: **http://IP-адрес:9100/metrics**.

На примере **Node Exporter** мы будем собирать метрики узла и передавать их в Prometheus и визуализируем их в Grafana.

И так приступим к установке, создадим директорию для размещения **docker-compose** файла:

```
mkdir -p /prometheus
```

Перейдем в созданный каталог:

```
cd /prometheus
```

Создадим файл **docker-compose.yml**, содержащий следующие сервисы:

```
nano docker-compose.yml
```

Перенесем содержимое ниже в файл docker-compose.yml:

```
version: '3.9'

services:

  prometheus:
    image: prom/prometheus:latest
    volumes:
      - ./configuration:/etc/prometheus/
      - ./data:/prometheus/
    container_name: prometheus
    hostname: prometheus
    command:
      - --config.file=/etc/prometheus/prometheus.yml
    ports:
      - 9090:9090
    restart: unless-stopped
    environment:
      TZ: "Europe/Moscow"
    networks:
      - default

  node-exporter:
    image: prom/node-exporter
    volumes:
      - /proc:/host/proc:ro
      - /sys:/host/sys:ro
      - /:/rootfs:ro
```

```
    container_name: exporter
    hostname: exporter
    command:
      - --path.procfs=/host/proc
      - --path.sysfs=/host/sys
      - --collector.filesystem.ignored-mount-points
      -
    ^/(sys|proc|dev|host|etc|rootfs/var/lib/docker/containers|rootfs/var/lib
    /docker/overlay2|rootfs/run/docker/netns|rootfs/var/lib/docker/aufs)
    ($$|/)
    ports:
      - 9100:9100
    restart: unless-stopped
    environment:
      TZ: "Europe/Moscow"
    networks:
      - default

networks:
  default:
    ipam:
      driver: default
      config:
        - subnet: 172.28.0.0/16
```

Для сбора статистики **Node Exporter** мы смонтировали `/proc` `/sys` и `/` в режиме чтения.

Находясь в директории **prometheus/**, создадим два каталога для хранения конфигурационных файлов и накопленных данных:

```
mkdir -p configuration
mkdir -p data
```

Выдадим права на созданный каталог **data**:

```
chown 65534:65534 data
```

В каталоге **configuration** создадим конфигурационный файл **prometheus.yml**:

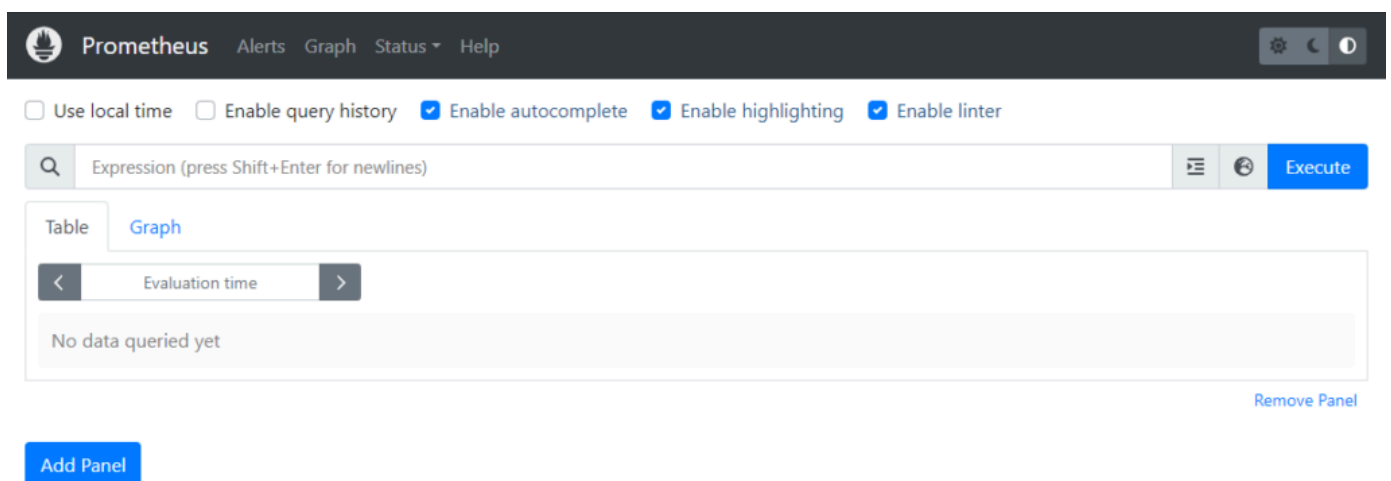
```
nano configuration/prometheus.yml
```

```
scrape_configs:
  - job_name: node
    scrape_interval: 5s
    static_configs:
      - targets: ['node-exporter:9100']
```

Из каталога **prometheus/** запустим **docker-compose**:

```
docker-compose up -d
```

http://<IP_узла_установки>:9090 — prometheus (по умолчанию аутентификация у сервиса отсутствует).



По адресу http://<IP_узла_установки>:9100 — страница Node Exporter

Node Exporter

Prometheus Node Exporter

Version: (version=1.8.2, branch=HEAD, revision=f1e0e8360aa60b6cb5e5cc1560bed348fc2c1895)

- [Metrics](#)

Значит все установлено верно.

Запуск Grafana в Docker

Теперь доработаем наш docker-compose.yml файл для запуска Grafana:

```
version: '3.9'

services:

  prometheus:
    image: prom/prometheus:latest
    volumes:
      - ./configuration:/etc/prometheus/
      - ./data:/prometheus/
    container_name: prometheus
    hostname: prometheus
    command:
      - --config.file=/etc/prometheus/prometheus.yml
    ports:
      - 9090:9090
    restart: unless-stopped
    environment:
      TZ: "Europe/Moscow"
    networks:
      - default

  node-exporter:
    image: prom/node-exporter
    volumes:
      - /proc:/host/proc:ro
```

```

    - /sys:/host/sys:ro
    - /:/rootfs:ro
  container_name: exporter
  hostname: exporter
  command:
    - --path.procfs=/host/proc
    - --path.sysfs=/host/sys
    - --collector.filesystem.ignored-mount-points
    -
  ^/(sys|proc|dev|host|etc|rootfs/var/lib/docker/containers|rootfs/var/lib
  /docker/overlay2|rootfs/run/docker/netns|rootfs/var/lib/docker/aufs)
  ($$|/)
  ports:
    - 9100:9100
  restart: unless-stopped
  environment:
    TZ: "Europe/Moscow"
  networks:
    - default

grafana:
  image: grafana/grafana
  user: root
  depends_on:
    - prometheus
  ports:
    - 3000:3000
  volumes:
    - ./grafana:/var/lib/grafana
    - ./grafana/provisioning/:/etc/grafana/provisioning/
  container_name: grafana
  hostname: grafana
  restart: unless-stopped
  environment:
    TZ: "Europe/Moscow"
  networks:
    - default

networks:
  default:

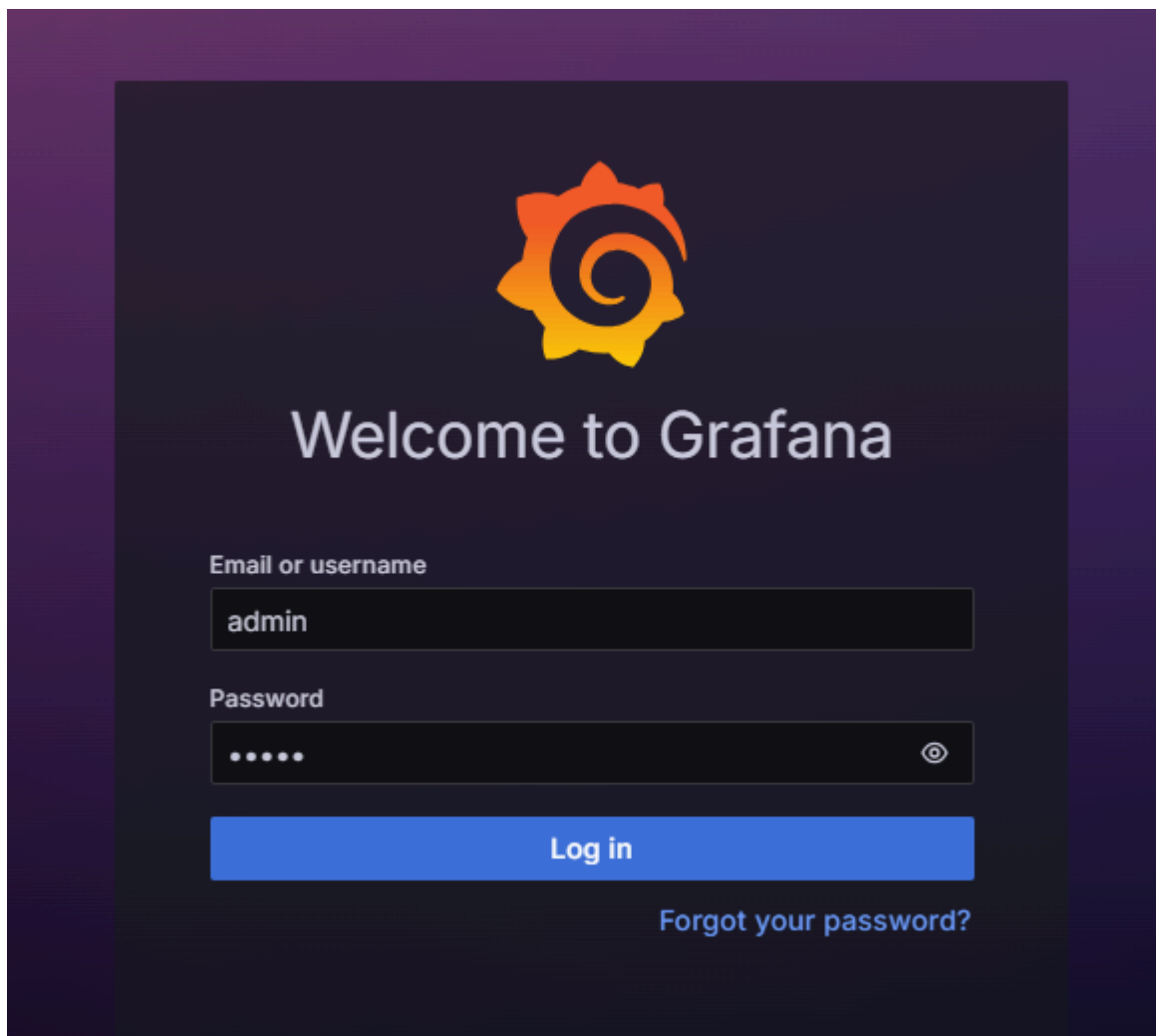
```

```
ipam:
  driver: default
  config:
    - subnet: 172.28.0.0/16
```

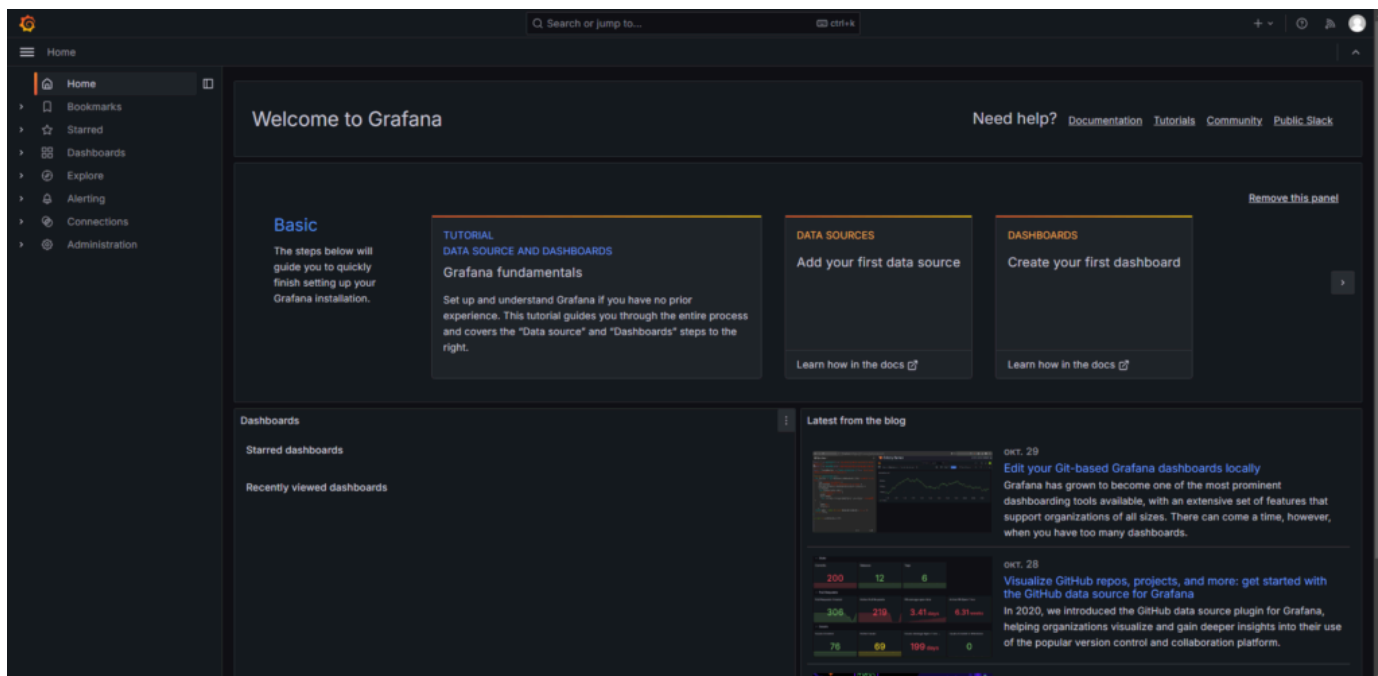
Выполним команду:

```
docker-compose up -d
```

Теперь **Grafana** на http://<IP_узла_установки>:3000 — grafana (данные для входа по умолчанию: login — admin; password — admin).

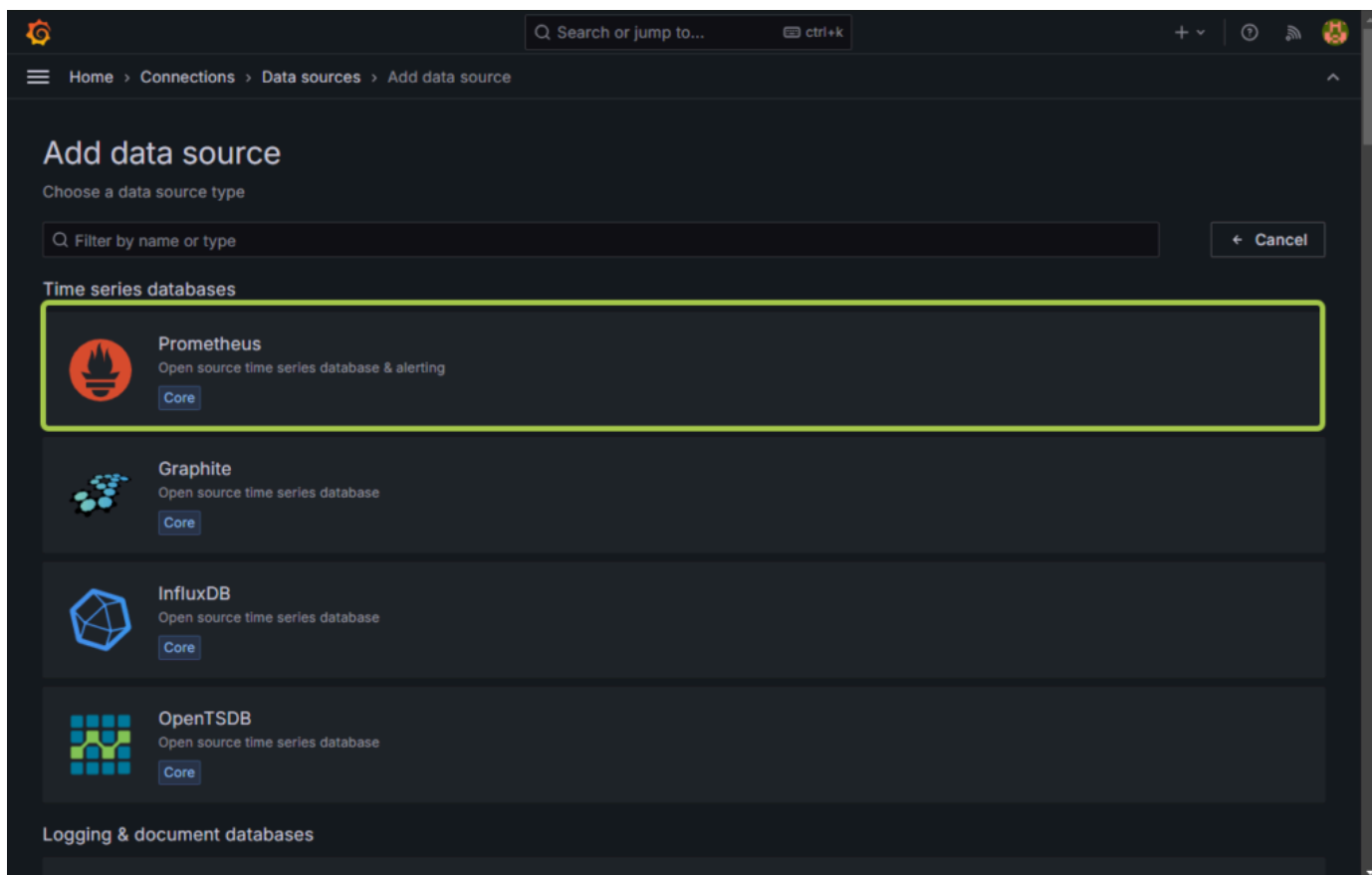


Система предложит придумать новый пароль для входа в систему.




Добавление источника данных Prometheus в Grafana

Теперь добавьте Prometheus в качестве источника данных на Grafana, нажав на **Data sources -> Prometheus**



Далее и введем URL Prometheus, название оставим по умолчанию.

 Search or jump to... ctrl+k

Home > Connections > Data sources > prometheus

Name ⓘ prometheus Default ☒

Before you can use the Prometheus data source, you must configure it below or in the config file. For detailed instructions, [view the documentation](#).

Fields marked with * are required

Connection

Prometheus server URL * ⓘ http://172.28.131.157:9090

Authentication


Authentication methods

Choose an authentication method to access the data source

No Authentication ▼

TLS settings

Additional security measures that can be applied on top of authentication

 Search or jump to... ctrl+k

Home > Connections > Data sources > prometheus

Prometheus type ⓘ Choose ▼

Cache level ⓘ Low ▼

Incremental querying (beta) ⓘ ☐

Disable recording rules (beta) ⓘ ☐

Other

Custom query parameters ⓘ Example: max_source_resolution=5m&timeout=

HTTP method ⓘ POST ▼

Exemplars

+ Add

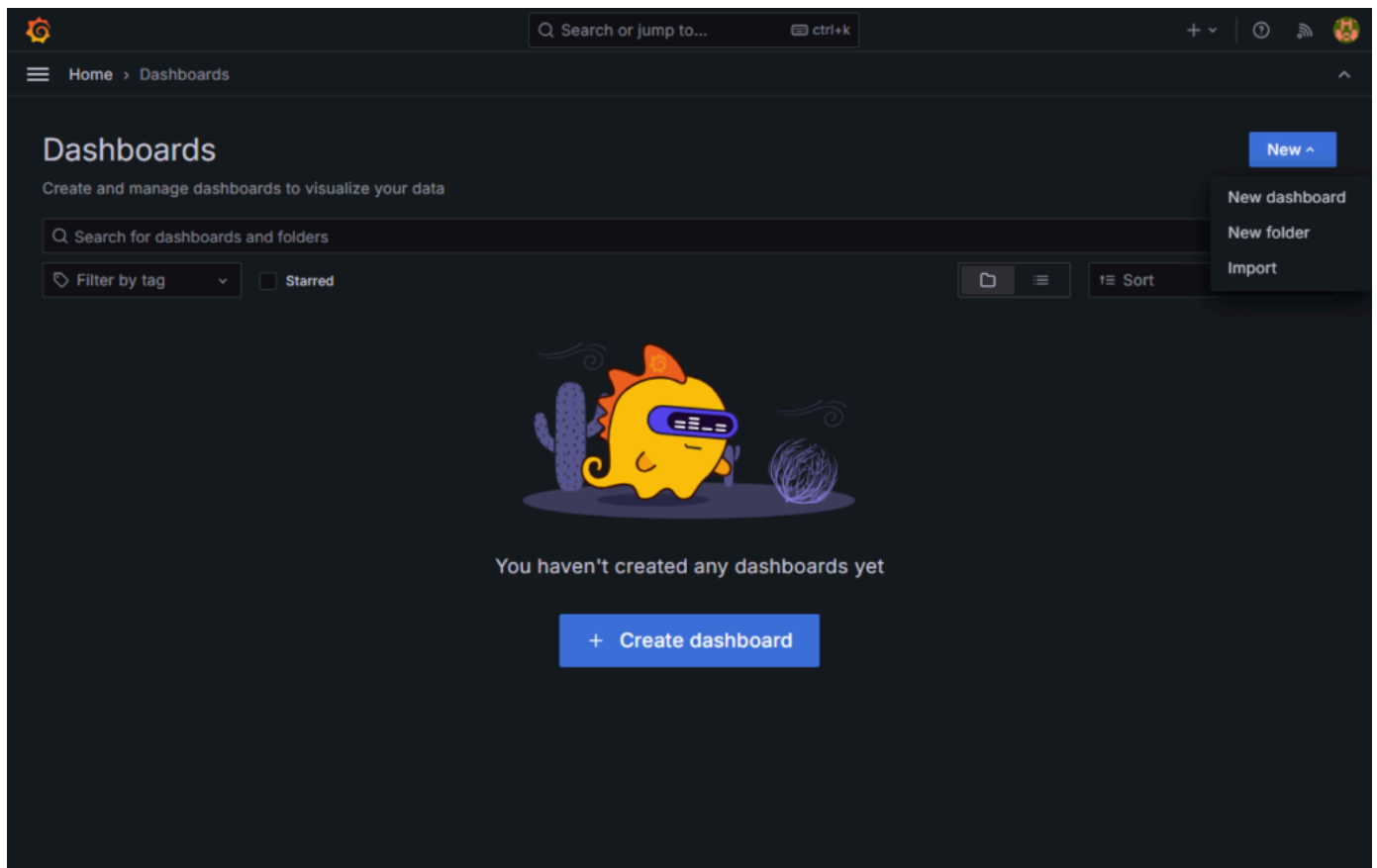
✓ Successfully queried the Prometheus API.

Next, you can start to visualize data by [building a dashboard](#), or by querying data in the [Explore view](#).

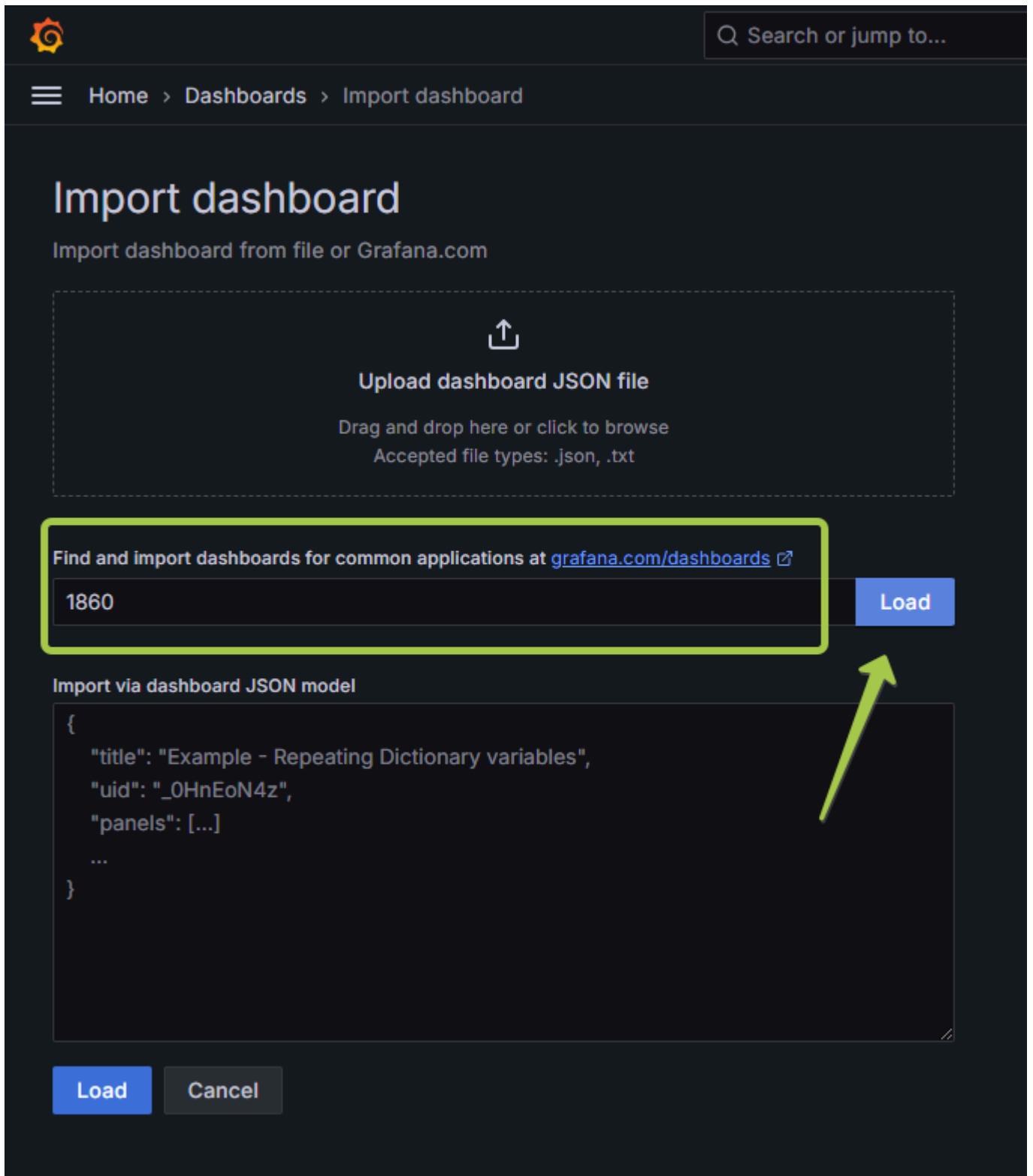
Delete Save & test

Сохраним и проверим источник данных по кнопке **Save & test**.

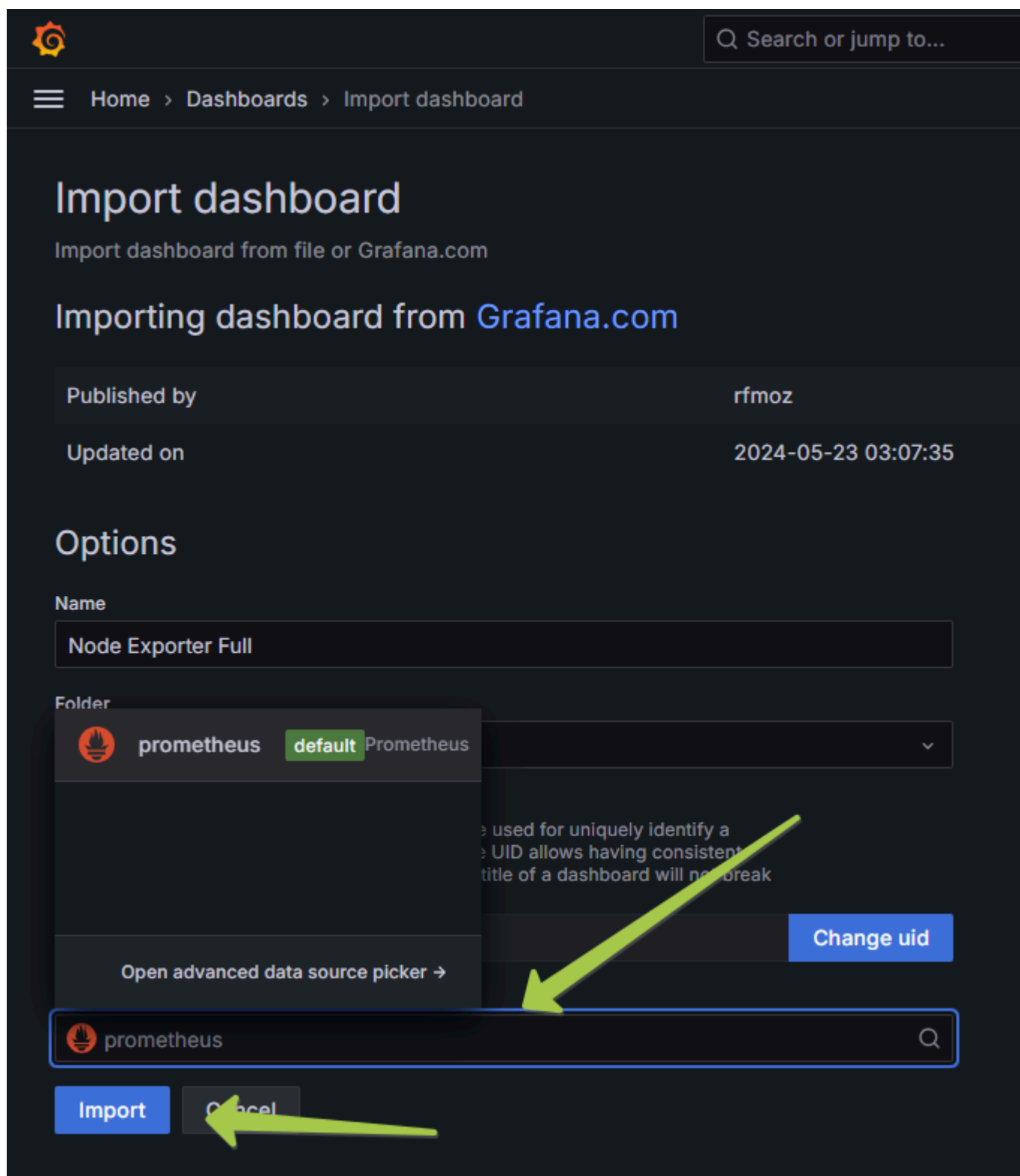
Теперь осталось добавить дашборды для мониторинга с node exporter. Для этого уже есть готовые варианты. Выберите пункт меню **Dashboards**. Нажмите на кнопку **New** и в выпадающем меню выберите пункт **Import**.



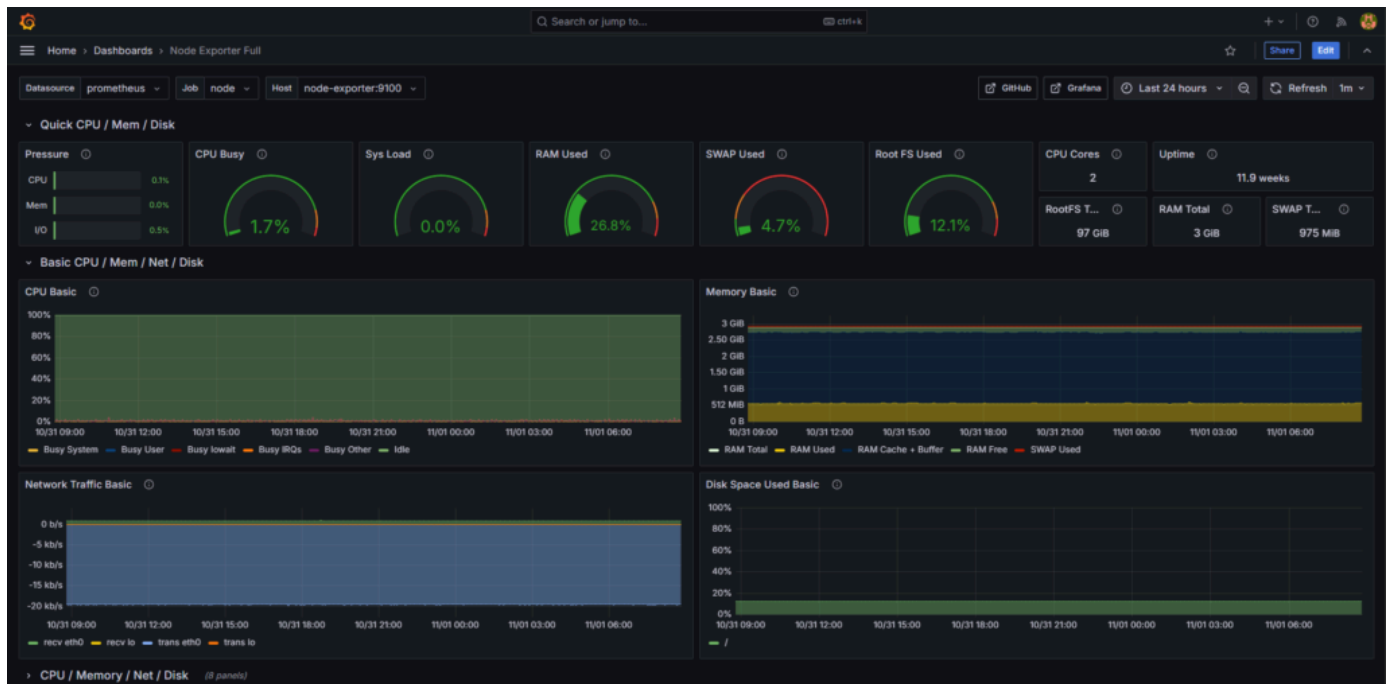
Вводим номер **1860** и нажимаем на **Load**. **Grafana** подгрузит дашборд из своего репозитория, далее в разделе **Prometheus** указываем наш источник данных и кликаем по **Import**.



Указываем наш источник **Prometheus** данных и кликаем по **Import**.



При переходе на импортированный дашборд откроется страница с данными по нему:



Ссылки:

- [Мониторинг состояния узлов при помощи Prometheus и Grafana](#)
- [Promethues оповещение через Telegram](#)
- [Установка и настройка Dashboard JVM Micrometer в Prometheus/Grafana](#)
- [Monitoring Stack](#)
- [Prometheus + Grafana + Alertmanager в Docker](#)
- [Promethues оповещение через Telegram](#)

Добавить комментарий

Ваш адрес email не будет опубликован. Обязательные поля помечены *

Имя *

Email *

Сайт

Комментарий *

☐

Сохранить моё имя, email и адрес сайта в этом браузере для последующих моих комментариев.

Отправить комментарий