



Prometheus + Grafana + Alertmanager в Docker

🕒 Обновлено: 06.06.2022 🕒 Опубликовано: 05.06.2022

Используемые

термины: [Prometheus](#), [Grafana](#), [Docker](#).

В данной инструкции мы рассмотрим пример docker-compose файла для организации системы мониторинга на базе Prometheus. Мы получим:

- Базу данных Prometheus и интерфейс для выполнения PromQL.
- Визуализацию с помощью Grafana.
- Сбор метрик через Node Exporter.
- Мониторинг HTTP с использованием Blackbox.
- Отправку уведомлений в Telegram с помощью Alertmanager.

Мы будем работать в среде Linux, хотя полученный файл docker-compose можно применять где угодно. Останавливаться подробно на описании docker-compose мы не будем.

[Готовим систему](#)

[Запуск сервисов для Prometheus + Node Exporter](#)

[Добавляем Grafana](#)

[Настраиваем отправку оповещений в телеграм](#)

[Мониторинг сайта с помощью blackbox](#)

Подготовка системы

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

Any question? ►

Создаем каталоги, где будем создавать наши файлы:

```
mkdir -p  
/opt/prometheus_stack/{prometheus,grafana,alertmanager,blackbox}
```

Создаем файл:

```
touch /opt/prometheus_stack/docker-compose.yml
```

Переходим в каталог prometheus_stack:

```
cd /opt/prometheus_stack
```

Дальше будем работать относительно него.

Prometheus + Node Exporter

Открываем файл:

```
vi docker-compose.yml
```

```
version: '3.9'  
  
services:  
  
  prometheus:  
    image: prom/prometheus:latest  
    volumes:  
      - ./prometheus:/etc/prometheus/  
    container_name: prometheus  
    hostname: prometheus  
    command:  
      - --  
    config.file=/etc/prometheus/prometheus  
    .yml  
    ports:  
      - 9090:9090  
    restart: unless-stopped  
    environment:  
      TZ: "Europe/Moscow"
```

Any question? ►

```

    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/li
b/docker/containers|rootfs/var/lib/doc
ker/overlay2|rootfs/run/docker/netns|r
ootfs/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

```

** в данном примере мы создаем 2 сервиса — prometheus и node-exporter. Также мы отдельно определили подсеть **172.28.0.0/16**, в которой будут находиться наши контейнеры docker.*

Создаем конфигурационный файл для prometheus:

```
vi prometheus/prometheus.yml
```

Any question? ►

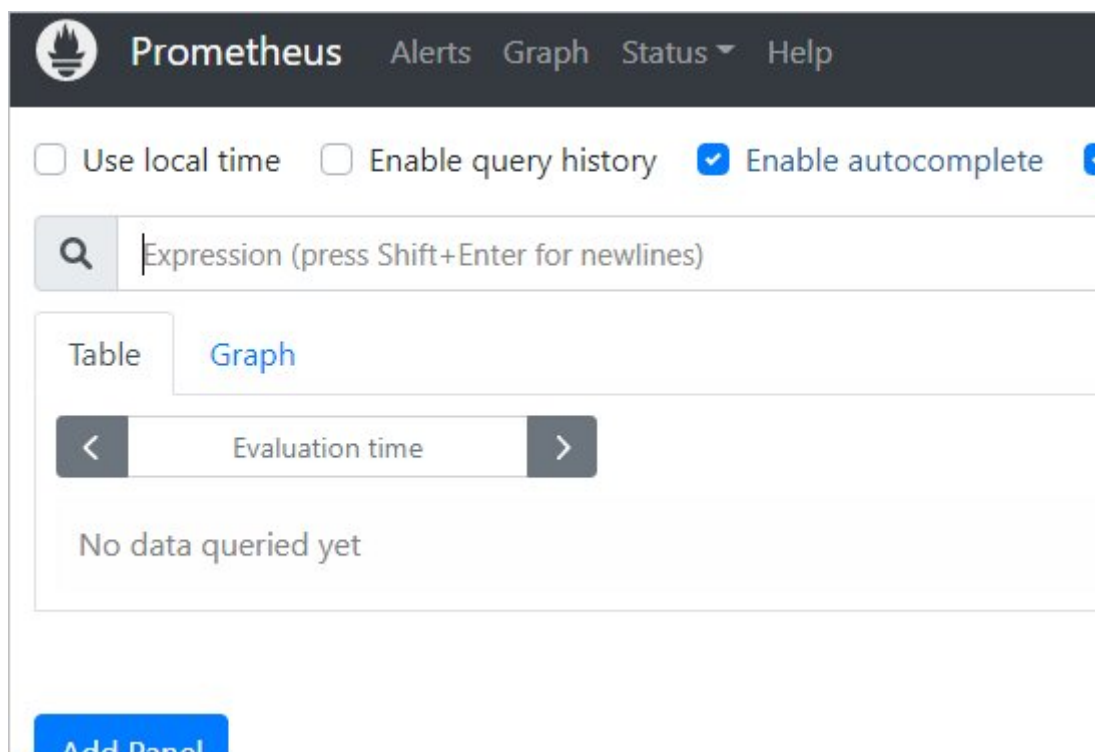
```
scrape_configs:
  - job_name: node
    scrape_interval: 5s
    static_configs:
      - targets: ['node-exporter:9100']
```

** в данном примере мы прописываем наш **node-exporter** в качестве таргета.*

Запускаем контейнеры:

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

Ждем несколько секунд и можно попробовать подключиться. Открываем браузер и переходим по адресу <http://<IP-адрес сервера>:9090> — мы должны увидеть страницу Prometheus:



Переходим по адресу <http://<IP-адрес сервера>:9100> — мы должны увидеть страницу Node Exporter:



Any question? ►

Мы движемся в правильном направлении. Идем дальше.

Grafana

Добавим к нашему стеку графану.

Открываем файл:

```
vi docker-compose.yml
```

Добавляем:

```
version: '3.9'

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

  ...
```

** по аналогии с другими сервисами, мы добавили сервис для графаны.*

Перезапустим контейнеры:

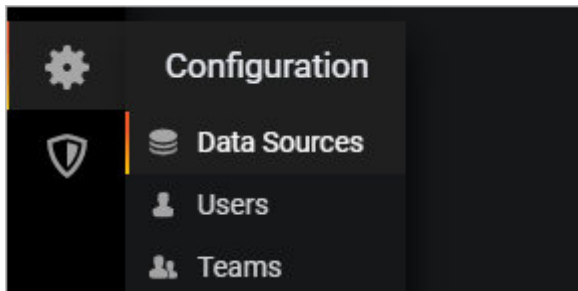
Any question? ►

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

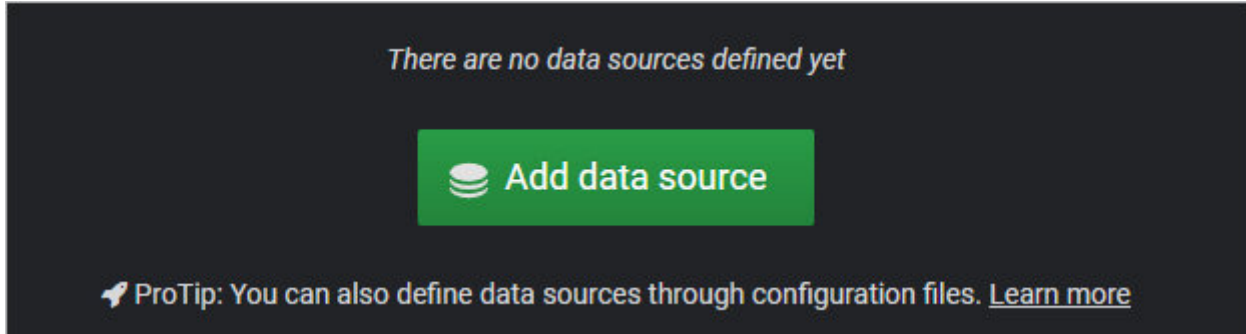
Открываем браузер и переходим по адресу <http://<IP-адрес сервера>:3000> — мы должны увидеть стартовую страницу Grafana.

Для авторизации вводим **admin / admin**. После система потребует ввести новый пароль.

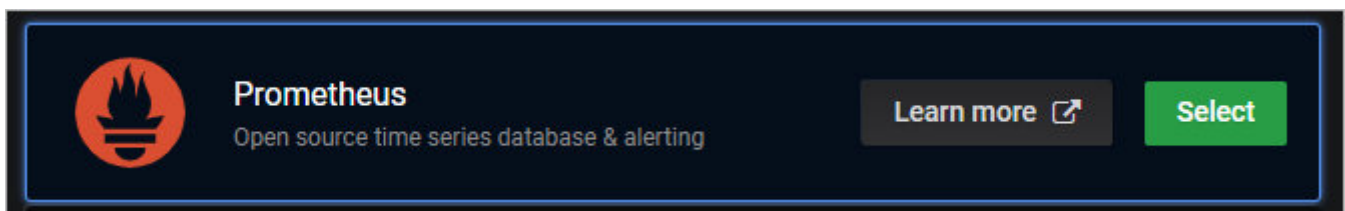
Настроим связку с Prometheus. Кликаем по иконке **Configuration - Data Sources**:



Переходим к добавлению источника, нажав по **Add data source**:

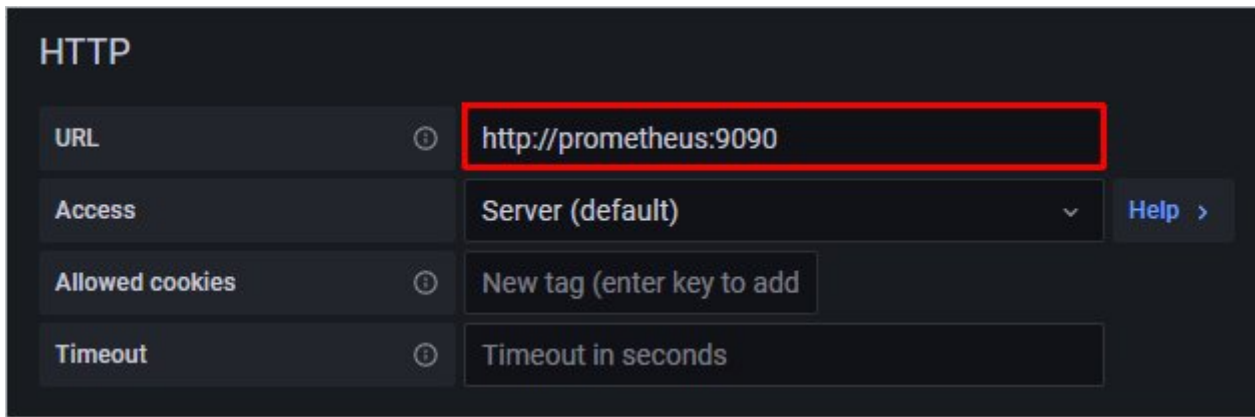


Среди списка источников данных находим и выбираем Prometheus, кликнув по **Select**:



Задаем параметры для подключения к Prometheus:

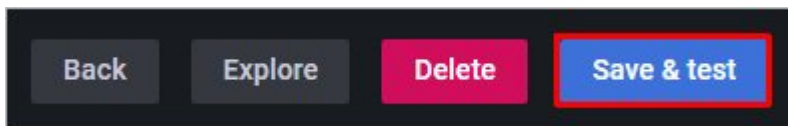
Any question? ►



HTTP

URL	<input type="text" value="http://prometheus:9090"/>
Access	Server (default) Help >
Allowed cookies	<input type="text" value="New tag (enter key to add)"/>
Timeout	<input type="text" value="Timeout in seconds"/>

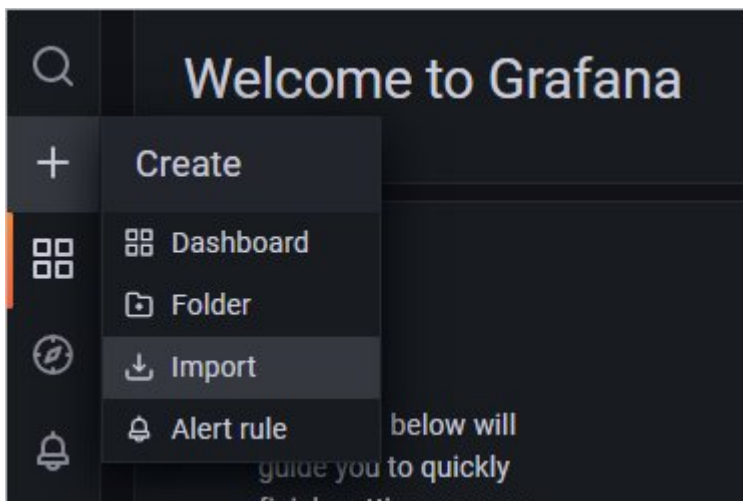
Сохраняем настройки, кликнув по **Save & Test**:



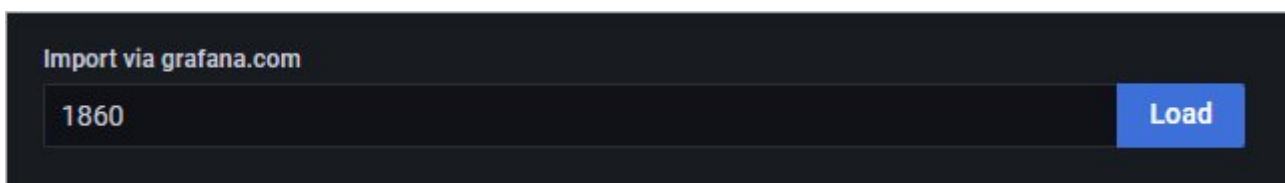
Back Explore Delete **Save & test**

Добавим дашборд для мониторинга с node exporter.
Для этого уже есть готовый вариант.

Кликаем по изображению плюса и выбираем
Import:



Вводим идентификатор дашборда. Для Node
Exporter это **1860**:



Import via grafana.com

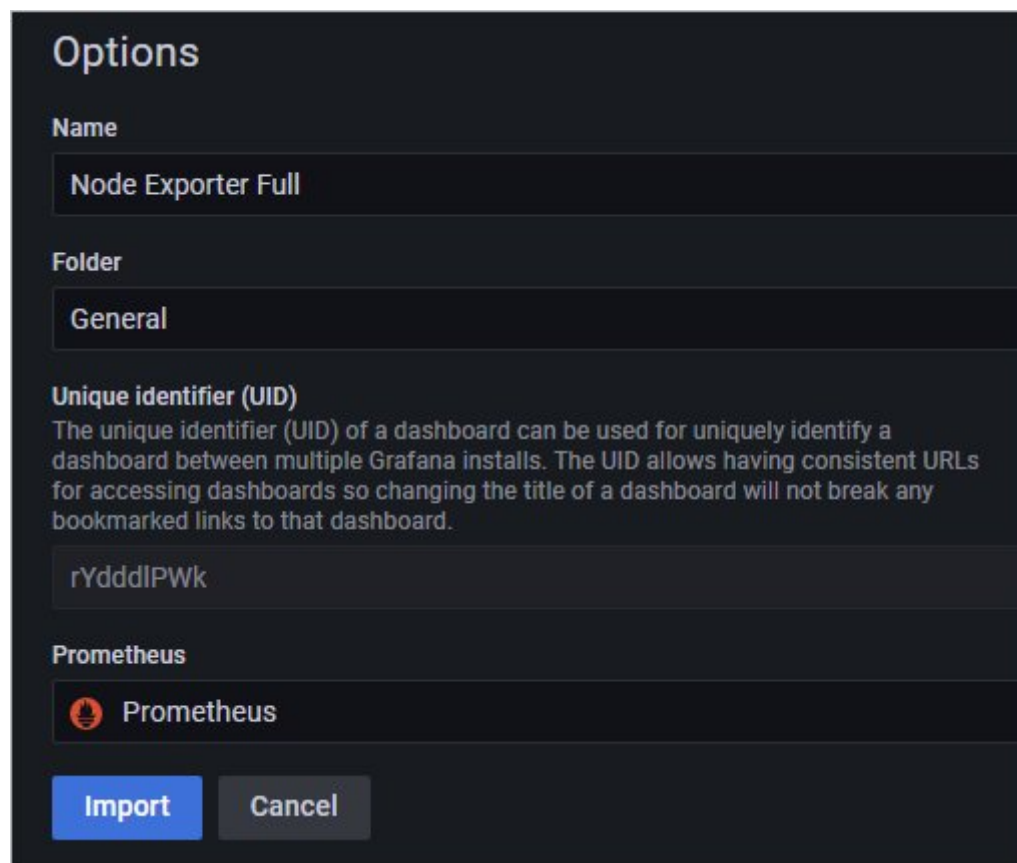
Load

Кликаем **Load** — Grafana подгрузит дашборд из
своего репозитория — выбираем в разделе

Any question? ►

Prometheus наш источник данных и кликаем по

Import:



The image shows the 'Options' dialog in Grafana for importing a dashboard. It has a dark theme. The 'Name' field contains 'Node Exporter Full'. The 'Folder' field contains 'General'. Below these is a section for 'Unique identifier (UID)' with a text description and a text input field containing 'rYdddIPWk'. At the bottom is a 'Prometheus' section with a Prometheus logo and the text 'Prometheus'. At the very bottom are two buttons: 'Import' (blue) and 'Cancel' (grey).

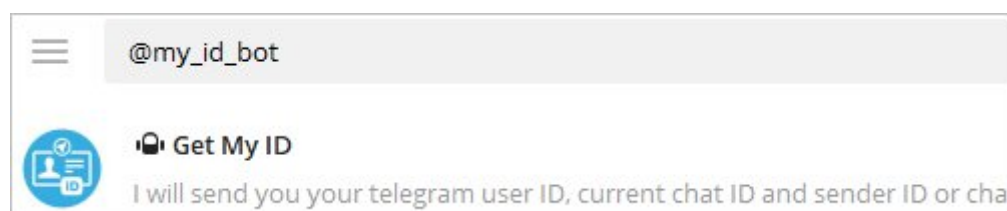
Мы увидим страницу с настроенными показателями метрик. Можно пользоваться.

Настройка оповещений с AlertManager

В нашем примере мы настроим отправку уведомлений на телеграм бот. Само оповещение будет выполнять AlertManager, который интегрируется с Prometheus.

Для начала подготовим наш бот телеграм.

Создадим нового — для этого открываем телеграм и ищем **@my_id_bot**:

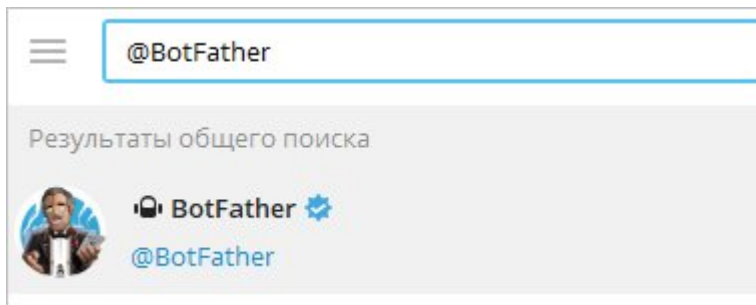


Any question? ►

Приложение покажет наш идентификатор.

Записываем — он нам потребуется позже.

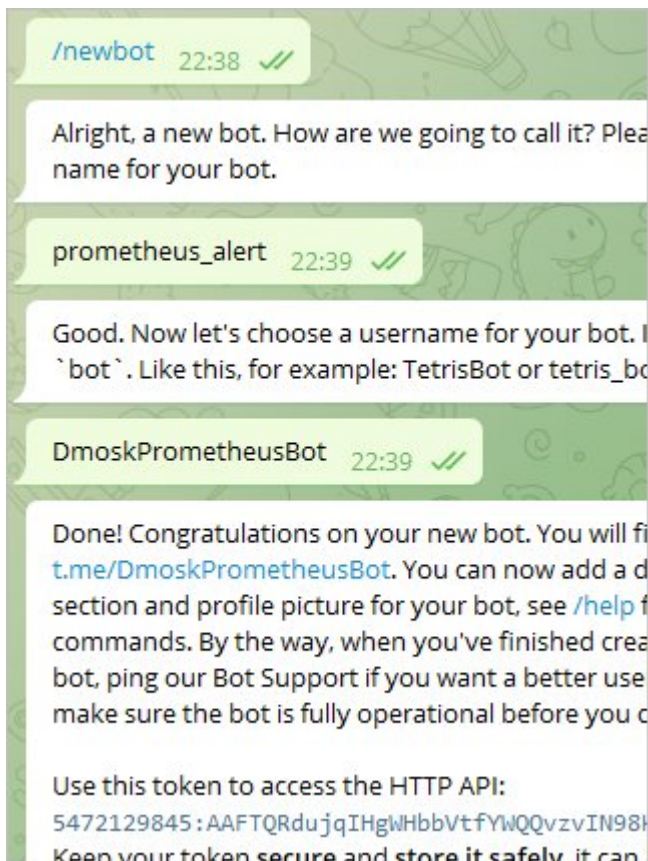
Теперь создаем бота. Ищем @BotFather:



Переходим в чат с найденным BotFather и запускаем бота:



Создаем бота, последовательно введя команду **/newbot** и отвечая на запросы мастера:



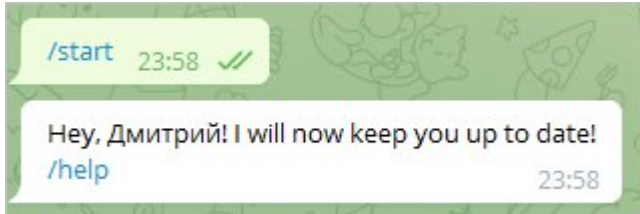
* в нашем примере мы создаем бота **`prometheus_alert`** с именем учетной записи **`DmoskPrometheusBot`**.

Any question? ►

Переходим в чат с созданным ботом, кликнув по его названию:



Запускаем бота:



С ботом мы закончили. Переходим к docker.

Открываем наш файл docker-compose:

```
vi docker-compose.yml
```

И добавляем:

```
version: '3.9'
...

alertmanager-bot:
  command:
    - --
  alertmanager.url=http://alertmanager:9093
    - --log.level=info
    - --store=bolt
    - --bolt.path=/data/bot.db
    - --telegram.admin=573454381
    - --
  telegram.token=5472129845:AAFTQRdujqIHgWHbbVtfYWQQvzvIN98KBqg
  image: metalmatze/alertmanager-bot:0.4.3
  user: root
  ports:
    - 8080:8080
  container_name: alertmanager-bot
  hostname: alertmanager-bot
```

Any question? ►

```
environment:
  TZ: "Europe/Moscow"
restart: unless-stopped
volumes:
  - ./data:/data
networks:
  - default

alertmanager:
  image: prom/alertmanager:v0.21.0
  user: root
  ports:
    - 127.0.0.1:9093:9093
  volumes:
    -
  ./alertmanager/:/etc/alertmanager/
  container_name: alertmanager
  hostname: alertmanager
  environment:
    TZ: "Europe/Moscow"
  restart: unless-stopped
  command:
    - '--
config.file=/etc/alertmanager/config.y
ml'
    - '--
storage.path=/etc/alertmanager/data'
  networks:
    - default

...

```

* мы добавили два сервиса — **alertmanager-bot** (телеграм бот для prometheus alertmanager) и **alertmanager** (система оповещений в prometheus).

Теперь открываем конфигурационный файл для prometheus:

```
vi prometheus/prometheus.yml
```

И добавим строки:

Any question? ►

```
...  
  
rule_files:  
  - 'alert.rules'  
  
alerting:  
  alertmanagers:  
    - scheme: http  
      static_configs:  
        - targets:  
          - "alertmanager:9093"
```

** в данном примере мы указали, что наш сервер мониторинга должен использовать в качестве системы оповещения **alertmanager**, который доступен по адресу **alertmanager:9093**. Также мы добавили файл **alert.rules** с описанием правил оповещения.*

Создаем файл с правилами оповещения:

```
vi prometheus/alert.rules
```

```
groups:  
- name: test  
  rules:  
    - alert: PrometheusTargetMissing  
      expr: up == 0  
      for: 0m  
      labels:  
        severity: critical  
      annotations:  
        summary: "Prometheus target  
missing (instance {{ $labels.instance  
}})"  
        description: "A Prometheus  
target has disappeared. An exporter  
might be crashed. VALUE = {{ $value }}  
LABELS: {{ $labels }}"
```

** в данном примере мы добавили правило, которое будет срабатывать при недоступности узла (node-exporter).*

Any question? ►

Переходим к конфигурированию alertmanager:

```
vi alertmanager/config.yml
```

```
route:
  receiver: 'alertmanager-bot'

receivers:
- name: 'alertmanager-bot'
  webhook_configs:
  - send_resolved: true
    url: 'http://alertmanager-bot:8080'
```

* данная конфигурация позволяет отправлять оповещения с помощью webhook на сервис **alertmanager-bot:8080** (наш телеграм бот для alertmanager).

Запускаем новые сервисы docker:

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

Открываем браузер и переходим по адресу <http://<IP-адрес сервера>:9090> — на вкладке Alerts мы должны увидеть нашу настройку:

Prometheus Alerts Graph Status Help

✓ Inactive (1) ✓ Pending (0) ✓ Firing (0)

/etc/prometheus/alert.rules > test

▼ PrometheusTargetMissing (0 active)

name: PrometheusTargetMissing
expr: up == 0
labels:
 severity: critical
annotations:
 description: A Prometheus target has disappeared. An exporter might be crashed
 summary: Prometheus target missing (instance {{ \$labels.instance }})

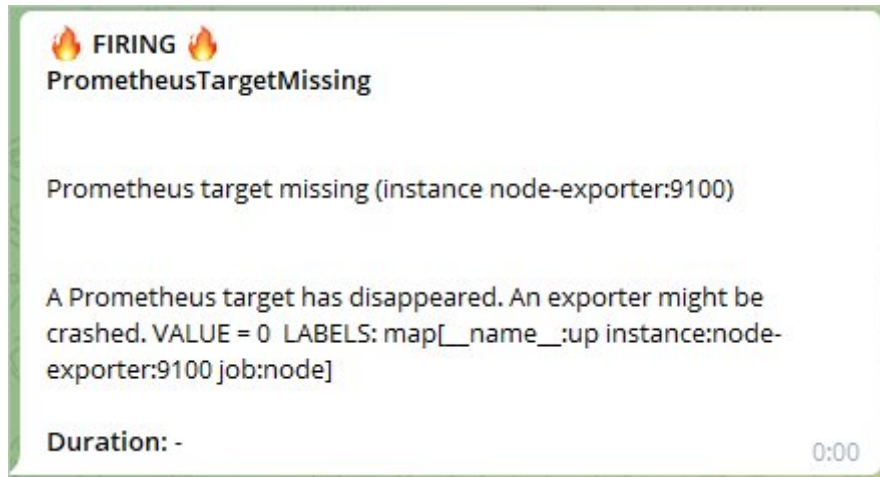
Any question? ►

Попробуем проверить работу оповещения.

Отключаем наш node exporter:

```
docker stop exporter
```

На телеграм должно прийти оповещение:



Отправка уведомлений настроена.

Blackbox exporter

Переходим к настройке мониторинга http сервисов.

В нашем примере мы будем проверять работу сайта dmosk.ru.

Открываем наш файл docker-compose:

```
vi docker-compose.yml
```

Добавим в него:

```
version: '3.9'

services:
  ...

  blackbox:
    image: prom/blackbox-exporter
    container_name: blackbox
    hostname: blackbox
    ports:
      - 9115:9115
```

Any question? ►

```
restart: unless-stopped
command:
  - "--"
config.file=/etc/blackbox/blackbox.yml
"
volumes:
  - ./blackbox:/etc/blackbox
environment:
  TZ: "Europe/Moscow"
networks:
  - default

...

```

* сервис **blackbox** является компонентом Prometheus для мониторинга сетевых сервисов по различным протоколам.

Открываем конфигурационный файл prometheus:

```
vi prometheus/prometheus.yml
```

И дописываем:

```
scrape_configs:
  ...
  - job_name: 'blackbox'
    metrics_path: /probe
    params:
      module: [http_2xx]
    static_configs:
      - targets:
        - https://www.dmosk.ru
    relabel_configs:
      - source_labels: [__address__]
        target_label: __param_target
      - source_labels:
        [__param_target]
        target_label: instance
      - target_label: __address__
        replacement: blackbox:9115
  ...

```

Any question? ►

** подобная конфигурация позволит создать дополнительное задание мониторинга сайта <https://www.dmosk.ru>.*

Создаем конфигурационный файл blackbox:

```
vi blackbox/blackbox.yml
```

```
modules:
  http_2xx:
    prober: http
    timeout: 5s
    http:
      valid_http_versions:
["HTTP/1.1", "HTTP/2.0"]
      valid_status_codes: [200]
      method: GET
      no_follow_redirects: true
      fail_if_ssl: false
      fail_if_not_ssl: false
      fail_if_body_matches_regexp:
        - "Could not connect to
database"
      fail_if_body_not_matches_regexp:
        - "Download the latest version
here"
      fail_if_header_matches: #
Verifies that no cookies are set
        - header: Set-Cookie
          allow_missing: true
          regexp: '.*'
      fail_if_header_not_matches:
        - header: Access-Control-
Allow-Origin
          regexp: '(\*|example\.com)'
      tls_config:
        insecure_skip_verify: false
        preferred_ip_protocol: "ip4"
        ip_protocol_fallback: false
```

** во многом, данный пример взят с официальной [страницы на Github](#).*

Теперь откроем конфигурационный файл с описанием правил для предупреждений:

Any question? ►


```
vi prometheus/alert.rules
```

Добавим:

```
...

- alert: BlackboxSlowProbe
  expr:
avg_over_time(probe_duration_seconds[1
m]) > 5
  for: 1m
  labels:
    severity: warning
  annotations:
    summary: Blackbox slow probe
(instance {{ $labels.instance }})
    description: "Blackbox probe
took more than 1s to complete\n  VALUE
= {{ $value }}\n  LABELS = {{ $labels
}}"

- alert: BlackboxProbeHttpFailure
  expr: probe_http_status_code <=
199 OR probe_http_status_code >= 400
  for: 0m
  labels:
    severity: critical
  annotations:
    summary: Blackbox probe HTTP
failure (instance {{ $labels.instance
}})
    description: "HTTP status code
is not 200-399\n  VALUE = {{ $value
}}\n  LABELS = {{ $labels }}"
```

* в данном примере мы создали два правила:

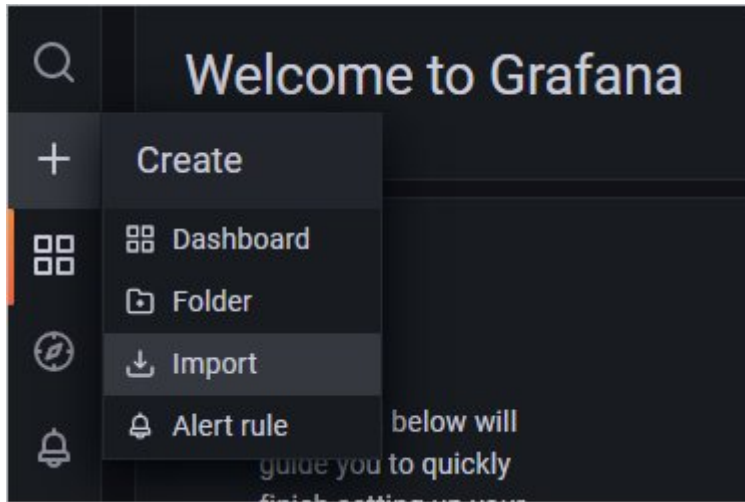
- **BlackboxSlowProbe** — предупреждать, если сайт открывается дольше 5 секунд.
- **BlackboxProbeHttpFailure** — реагировать, в случае получения кода ответа с ошибкой работы сайта (более 400).

Запускаем добавленный в докер сервис:

Any question? ►

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

Для визуализации мониторинга с помощью blackbox есть готовый дашборд в графанае. Снова открываем страницу импорта:



Вводим идентификатор **7587** и выбираем источник (как делали при добавлении дашборда node exporter).

Linux # Серверы # Мониторинг # DevOps



Была ли полезна вам эта инструкция?

Да

Нет

Like

2

[Tweet](#)

Any question? ►



*Дмитрий Моск — IT-специалист.
Настройка серверов, компьютерная помощь.*

Мини-инструкции

[Использование header_checks для замены заголовков в Postfix](#)

[Как установить и выполнить базовую настройку ноды для Ethereum под Linux Ubuntu](#)

[Как настроить отказоустойчивого кластер из двух серверов KeyDB](#)

Мониторинг под ключ с docker — Prometheus + Grafana + Alertmanager

[Как с помощью Gradle и плагина ospackage собрать пакеты RPM и Deb](#)

[Создание снапшотов на ZFS с их просмотром на шаре Samba](#)

[Примеры работы с Gitlab CI/CD — написание конвейеров для автоматизации разработки](#)

[Весь список ...](#)

Нужна помощь? Пишите:

Что хотите узнать...

Any question? ►

Контактная эл. почта

не обязательно, но для более быстрого ответа

Получить инструкцию

Реклама



[Настройка серверов](#)

Any question? ►