

Отслеживание статуса сервера, просмотр статистики и отправка уведомлений в Telegram(если сервер упал или встал) с помощью Grafana, Prometheus и Node Exporter

Этот гайд поможет настроить систему мониторинга серверов с использованием Grafana, Prometheus и Node Exporter для сбора и отображения метрик, а также отправки уведомлений в Telegram в случае падения серверов. В гайде используется Ubuntu 22.04.

1. Установка и настройка Node Exporter на отслеживаемом сервере.

Node Exporter собирает системные метрики, такие как использование ЦП, памяти, диск и сеть, и передаёт их Prometheus для мониторинга.

1.1. Скачиваем Node Exporter на отслеживаемый сервер:

```
wget
https://github.com/prometheus/node_exporter/r
eleases/download/v1.8.2/node_exporter-1.8.2.l
inux-amd64.tar.gz
```

1.2. Распаковываем архив:

```
tar -xvf
node_exporter-1.8.2.linux-amd64.tar.gz
```

1.3. Удаляем архив:

```
rm node_exporter-1.8.2.linux-amd64.tar.gz
```

1.4. Перемещаем папку:

```
sudo mv node_exporter-1.8.2.linux-amd64
node_exporter
```

1.5. Делаем бинарник исполняемым:

```
chmod +x node_exporter/node_exporter
```

- 1.6. Перемещаем бинарник в /usr/bin для удобного запуска:

```
sudo mv node_exporter/node_exporter /usr/bin/
```
- 1.7. Удаляем временную папку:

```
rm -Rvf node_exporter/
```
- 1.8. Создаём systemd-сервис для автоматического запуска Node Exporter:

```
sudo tee
/etc/systemd/system/exporterd.service >
/dev/null <<EOF
[Unit]
Description=Node Exporter
After=network.target
[Service]
User=root
ExecStart=/usr/bin/node_exporter
[Install]
WantedBy=multi-user.target
EOF
```
- 1.9. Запускаем сервис и включаем его на автозапуск, команды вводим отдельно!!:

```
sudo systemctl daemon-reload
sudo systemctl enable exporterd.service
sudo systemctl start exporterd.service
```
- 1.10. Открываем порт 9100:

```
sudo ufw allow 9100
```

А еще лучше открывать 9100 не для всех подряд, а только для того айпишника, на котором сервер стоит Grafana и Prometheus:

```
sudo ufw allow from 192.168.1.100 to any port 9100
```

Вместо 192.168.1.100 ваш айпи сервера на котором будет стоять статистика.

2. Установка и настройка Prometheus и Grafana на сервере для сбора метрик.

Prometheus — это система мониторинга, которая будет собирать данные с Node Exporter, а Grafana — это визуализационный инструмент для отображения этих данных

Установка Prometheus

- 2.1. Скачиваем Prometheus на сервер где будет статистика либо на том же самом сервере:

```
wget
```

```
https://github.com/prometheus/prometheus/releases/download/v2.54.1/prometheus-2.54.1.linux-amd64.tar.gz
```

- 2.2. Распаковываем архив:

```
tar xvf prometheus-2.54.1.linux-amd64.tar.gz
```

- 2.3. Удаляем архив:

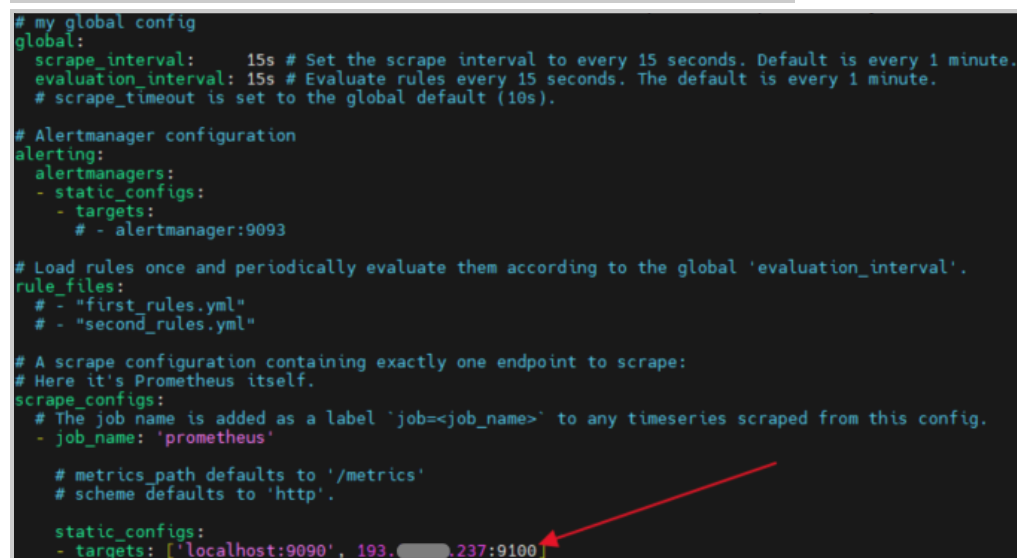
```
rm prometheus-2.54.1.linux-amd64.tar.gz
```

- 2.4. Перемещаем папку:

```
sudo mv prometheus-2.54.1.linux-amd64  
prometheus
```

- 2.5. Открываем конфиг нашего Prometheus для настройки:

```
vim /root/prometheus/prometheus.yml
```



```
# my global config
global:
  scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets:
            # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label 'job=<job_name>' to any timeseries scraped from this config.
  - job_name: 'prometheus'

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ['localhost:9090', 193.168.1.237:9100]
```

На место красной стрелки вписываем ip на который мы

устанавливали node_exporter

ВАЖНО! Все, блин, таргеты — в одинарных кавычках через запятую с пробелом!

**Пример: - targets: ['localhost:9090',
'айпи-вашего-сервера:9100']**

- 2.6. Далее выдаем права на исполнение нашего файла:

```
chmod +x /root/prometheus/prometheus.yml
```

- 2.7. Создаем сервисный файл, КОПИРУЕТЕ ВСЁ ЦЕЛИКОМ!:

```
sudo tee
```

```
/etc/systemd/system/prometheusd.service >
```

```
/dev/null <<EOF
```

```
[Unit]
```

```
Description=prometheus
```

```
After=network-online.target
```

```
[Service]
```

```
User=root
```

```
ExecStart=/root/prometheus/prometheus
```

```
--config.file="/root/prometheus/prometheus.yml"
```

```
Restart=always
```

```
RestartSec=3
```

```
LimitNOFILE=65535
```

```
[Install]
```

```
WantedBy=multi-user.target
```

```
EOF
```

- 2.8. После чего запускаем сервис:

```
sudo systemctl daemon-reload && sudo
```

```
systemctl enable prometheusd && sudo
```

```
systemctl restart prometheusd
```

3. Установка и настройка Grafana.

- 3.1. Если у вас RU сервер то начните с 4 пункта.

- 3.2. Установите необходимые пакеты:
- ```
sudo apt-get install -y apt-transport-https
software-properties-common wget
```
- 3.3. Импортируйте ключ GPG, *копируйте по одному!!!*:
- ```
sudo mkdir -p /etc/apt/keyrings/  
wget -q -O - https://apt.grafana.com/gpg.key  
| gpg --dearmor | sudo tee  
/etc/apt/keyrings/grafana.gpg > /dev/null
```
- 3.4. Добавим репозиторий для стабильных выпусков:
- ```
echo "deb
[signed-by=/etc/apt/keyrings/grafana.gpg]
https://apt.grafana.com stable main" | sudo
tee -a /etc/apt/sources.list.d/grafana.list
```
- 3.5. Выполните следующую команду, чтобы обновить список доступных пакетов:
- ```
sudo apt-get update
```
- 3.6. Чтобы установить операционную систему Grafana, выполните следующую команду:
- ```
sudo apt-get install grafana
```
- 3.7. Чтобы запустить службу, выполните следующую команду:
- ```
sudo systemctl daemon-reload && sudo  
systemctl start grafana-server && sudo  
systemctl enable grafana-server
```
- 3.8. Откройте порт 3000 для доступа к Grafana:
- ```
sudo ufw allow 3000
```

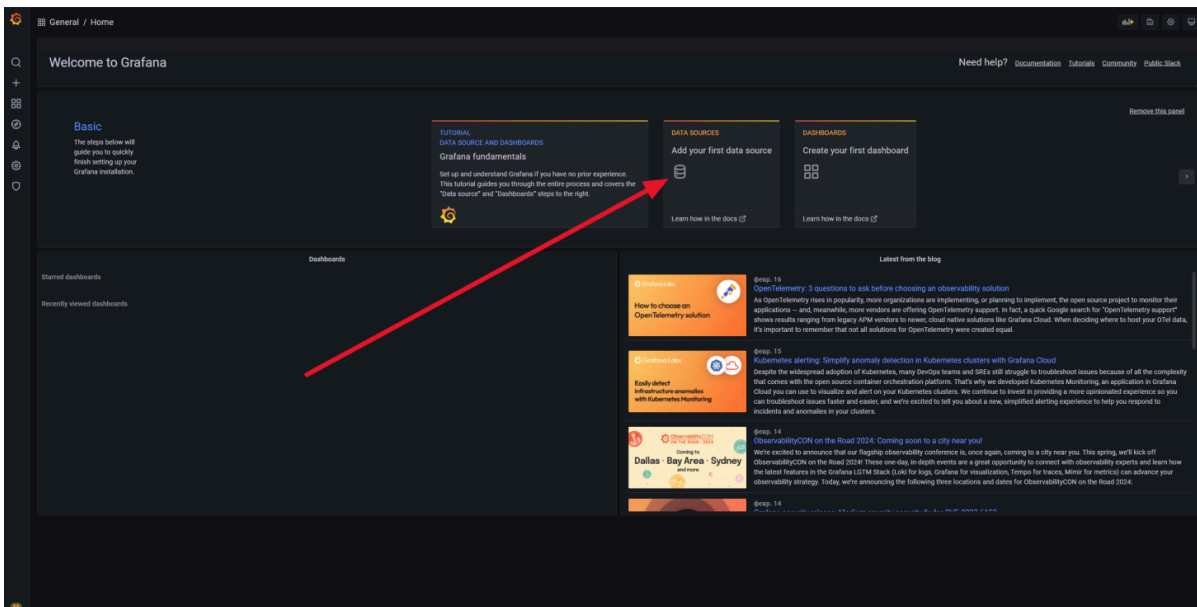
## 4. Установка Grafana на RU сервер.

- 4.1. Скачайте пакет напрямую на сервер:
- ```
wget  
https://dl.grafana.com/oss/release/grafana_9.  
5.6_amd64.deb
```
- 4.2. Установка зависимостей:
- ```
sudo apt-get update
sudo apt-get install -y adduser libfontconfig
```

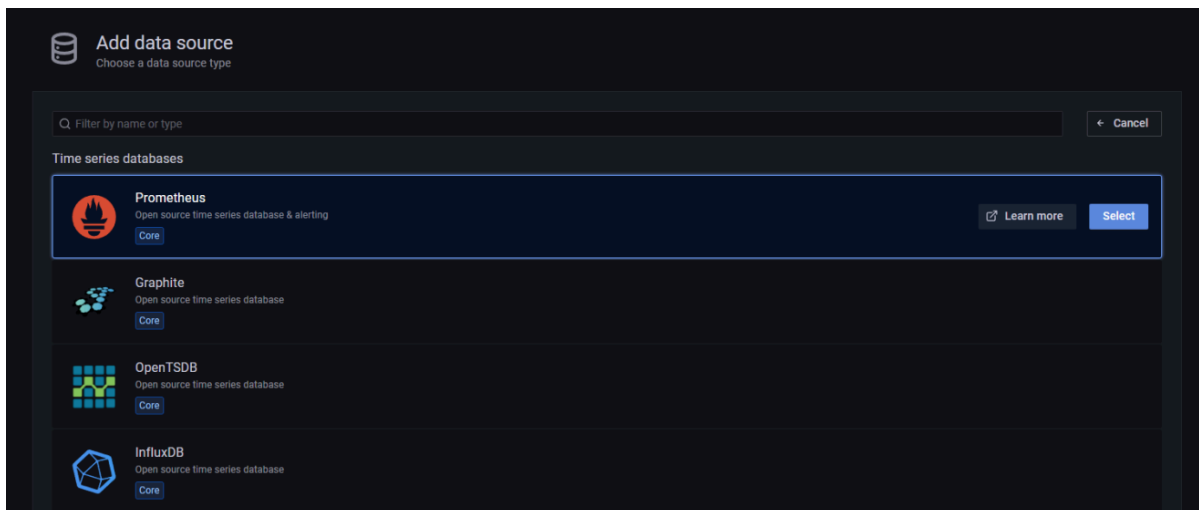
- 4.3. Установите скачанный **.deb** пакет с помощью **dpkg**:
- ```
sudo dpkg -i grafana_11.4.0_amd64.deb
```
- 4.4. Если в процессе установки возникают ошибки зависимостей, исправьте их командой:
- ```
sudo apt-get install -f
```
- 4.5. После установки запустите сервис Grafana и настройте его автозапуск при загрузке системы:
- ```
sudo systemctl start grafana-server  
sudo systemctl enable grafana-server
```
- 4.6. Убедитесь, что Grafana работает корректно:
- ```
sudo systemctl status grafana-server
```

## 5. Добавление Prometheus как источника данных в Grafana

- 5.1. Переходим в браузере на своём домашнем устройстве по адресу: `http://IP:3000`, где IP - адрес сервера, на который мы установили Prometheus и Grafana. Если не заходит то добавляем в файерволл порт 3000: `ufw allow 3000` Логин и пароль: `admin/admin` Далее задаем свой пароль и попадаем в Dashboard.
- 5.2. Нажимаем на кнопку Data Sources



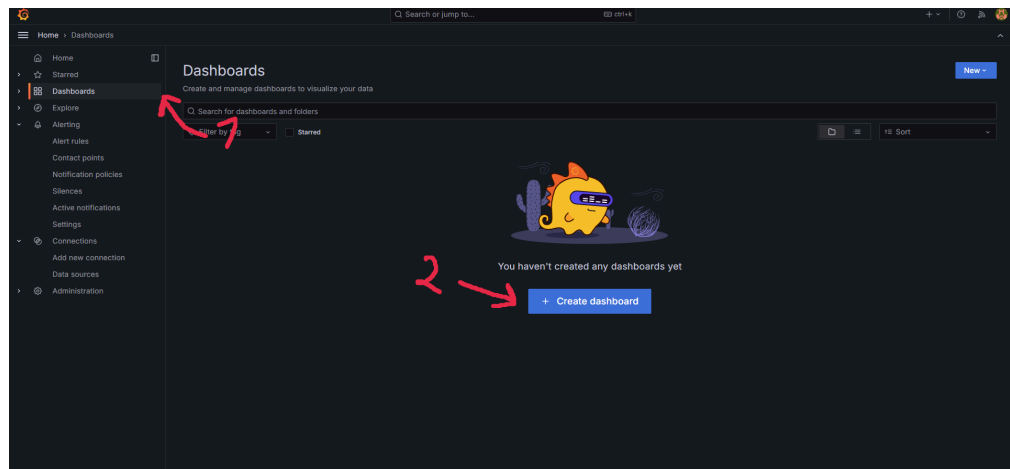
Выбираем Prometheus



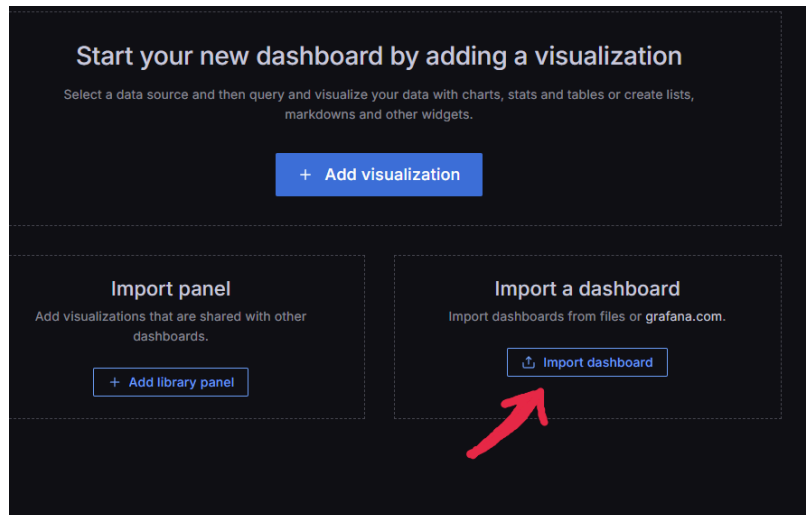
В графе URL вводим `http://IP:9090` , где IP - адрес сервера на котором стоит Prometheus и Grafana. После чего нажимаем внизу на **Save & test**

## 6. Импорт дашборда для мониторинга серверов

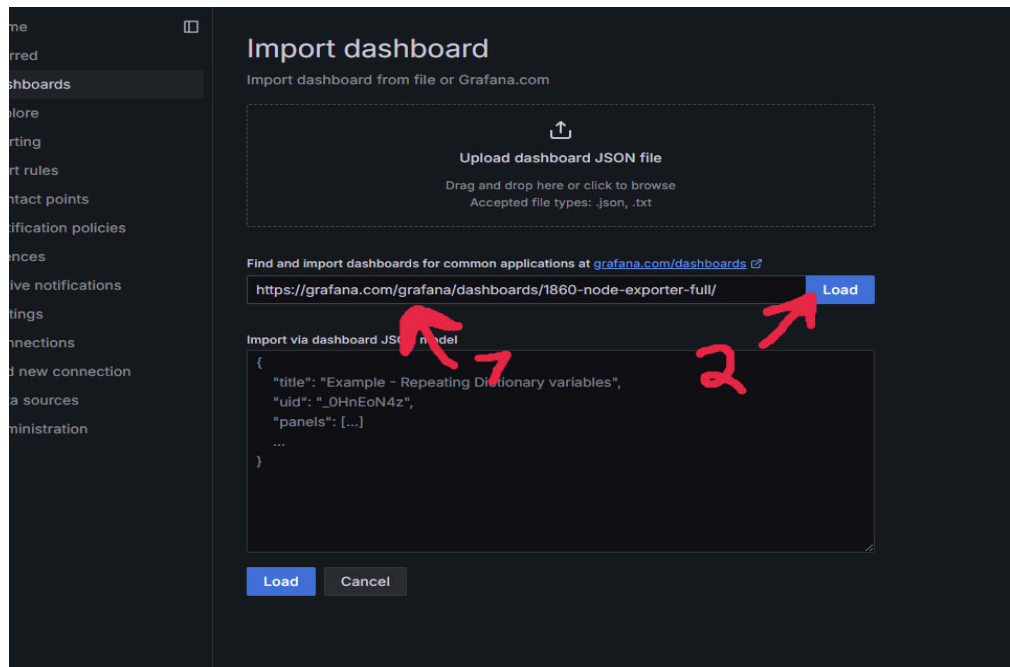
- 6.1. Скопируйте [ссылку на дашборд Node Exporter](#).
- 6.2. В Grafana перейдите в **Dashboards**, затем нажмите **Create dashboard**.



2) После чего выберите **Import dashboard**

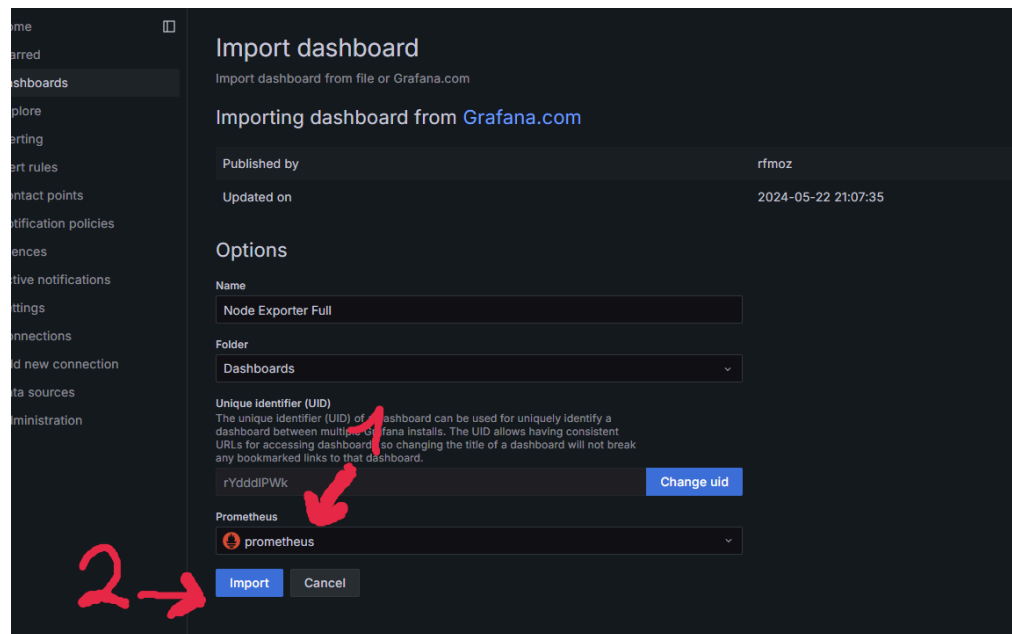


### 3) Вставьте ссылку и нажмите **Load**

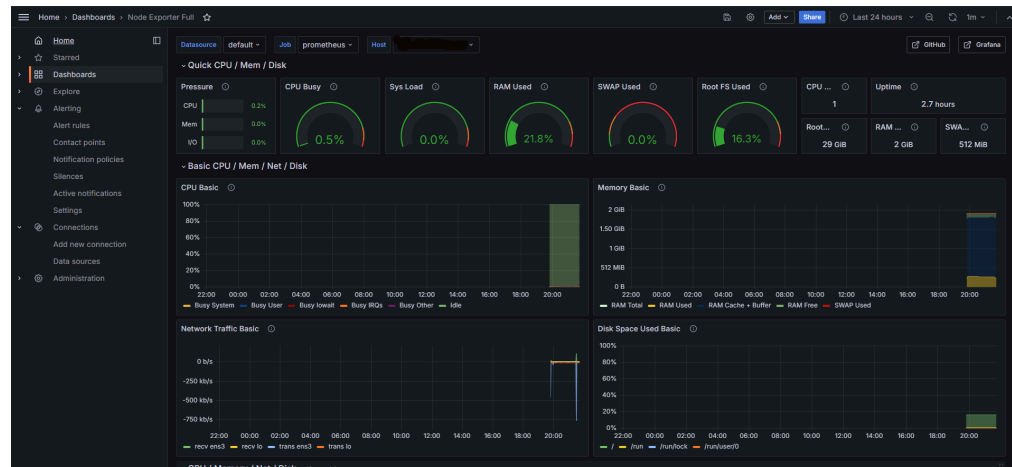


### 4) Выберите источник данных Prometheus из выпадающего списка и нажмите **Import**. При желании можете изменить название на другое.





5) У вас откроется статистика. В разделе Hosts можете выбирать сервера, на которые вы установили Node\_exporter. Сверху справа углу можете изменять данные: время, за которое отобразится статистика, ручное обновление статистики и изменить время обновления статистики. И всё, вы можете смотреть статистику сервера при переходе на <http://IP-server:3000/dashboards>



## 7. Настройка уведомлений в Telegram об падении сервера и когда он восстановился

Для того чтобы получать уведомления о падении серверов, необходимо настроить Telegram в Grafana.

В Grafana перейдите в раздел Contact points и создайте новый КОНТАКТ:

- 7.1. Выберите Telegram
- 7.2. В поле Token введите токен вашего бота (его можно получить у [BotFather](#)).
- 7.3. В поле Chat ID введите свой ID Telegram (узнать его можно через [UserInfoBot](#))

- 7.4. После нажимаете Save contact point

Дальше, после создания бота для уведомления, переходим в Notification Templates и создаём новый шаблон, чтобы отправлялись сообщения типа «Ваш сервер IP упал!!!», а не длинный текст со всякими ссылками на ваш Grafana.

Вставьте следующий код в Template:

```
{{- define "telegram.message_alert" -}}
🚨 {{ .Labels.alertname }} 🚨
{{- if eq .Status "firing" }}
Сервер {{ .Labels.instance }} упал!!! 😱{{- else
if eq .Status "resolved" }}
Сервер {{ .Labels.instance }} восстановлен! ✅
{{- end }}
{{- end -}}
```

```
{{ define "telegram" }}
{{ if .Alerts.Firing -}}
{{- range .Alerts.Firing }}
{{ template "telegram.message_alert" . }}
{{- end }}
{{- end }}
```

```
{{- end }}
{{ if .Alerts.Resolved -}}
{{- range .Alerts.Resolved }}
{{ template "telegram.message_alert" . }}
{{- end }}
{{- end }}
{{ end }}
```

Дальше нажимаем на «Save» сверху справа в углу.

## 8. Создание правила алертинга для уведомлений в Telegram.

Перейдите в **Alert rules** и создайте новое правило. И сделайте как на Изображениях:

1)

The screenshot shows the 'Define query and alert condition' step in the Grafana Alerting configuration interface. It is divided into two main sections: 'Query' and 'Alert condition'.

**Query Section:**

- 1. Enter alert rule name:** A text input field with the value 'Status servers'.
- 2. Define query and alert condition:** A section with a 'Define query and alert condition' link and a 'Need help?' link.
- Query Builder:** A panel with a 'prometheus' data source, a '10 minutes' interval, and a 'Set as alert condition' button. It includes a 'Metrics browser' and a query input field containing 'up{job="prometheus"}'. Below the query input are tabs for 'Options', 'Legend: Auto', 'Format: Time series', 'Step: auto', and 'Type: Instant'.
- Rule type:** A section with a 'Rule type' label and a 'Need help?' link. It has two radio buttons: 'Grafana-managed' (selected) and 'Data source-managed'.
- Expressions:** A section with an 'Add query' button and a 'Rule type' label. It contains two sub-sections:
  - B Reduce:** A section with a 'Reduce' label and a 'Set as alert condition' link. It has an 'Input' dropdown set to 'A', a 'Function' dropdown set to 'Last', and a 'Mode' dropdown set to 'Strict'.
  - C Threshold:** A section with a 'Threshold' label and a 'Set as alert condition' link. It has an 'Input' dropdown set to 'B', an 'IS BELOW' dropdown set to '1', and a 'Custom recovery threshold' toggle.

2) Вот тут создайте папки, во второй подгруппе поставьте время, через которое будут проверяться правила, другим словом, исполняться каждые N секунд/минут/часов. В Pending period ставьте None чтобы сразу отправлялось сообщение в Telegram.

The screenshot shows the 'Set evaluation behavior' step in the Grafana Alerting configuration interface. It contains the following sections:

- 3. Set evaluation behavior:** A section with a 'Define how the alert rule is evaluated.' link and a 'Need help?' link.
- Folder:** A section with a 'Folder' label and a 'Select a folder to store your rule.' label. It has a dropdown menu set to 'Telegramstatus' and a '+ New folder' button.
- Evaluation group and interval:** A section with an 'Evaluation group and interval' label and a 'Define how often the alert rule is evaluated.' label. It has a dropdown menu set to 'status' and a '+ New evaluation group' button.
- Pending period:** A section with a 'Pending period' label and a 'Period the threshold condition must be met to trigger the alert. Selecting "None" triggers the alert immediately once the condition is met.' label. It has a row of buttons: '0s', 'None' (selected), '15m', '30m', '45m', '1h', and '1h15m'.
- Configure no data and error handling:** A link to 'Configure no data and error handling'.

3) В 4 пункте в Contact Point выберите вашего Telegram-бота.

The screenshot shows the '4. Configure labels and notifications' page in Grafana's Alertmanager interface. The page has a dark theme. At the top, it says 'Select who should receive a notification when an alert rule fires.' Below this, there are two sections: 'Labels' and 'Notifications'. The 'Labels' section has a sub-header 'Add labels to your rule for searching, silencing, or routing to a notification policy.' and a button '+ Add labels'. The 'Notifications' section has a sub-header 'Select who should receive a notification when an alert rule fires.' and two buttons: 'Select contact point' (which is highlighted) and 'Use notification policy'. Below these buttons, it says 'Notifications for firing alerts are routed to a selected contact point.' and a link 'Need help?'. Under the 'Select contact point' button, there is a section for 'Alertmanager: grafana' with a gear icon. Below that, there is a 'Contact point' section with a dropdown menu showing 'Telegram bot' and a link 'View or create contact points'. At the bottom, there is a section for 'Telegram' with a link 'Muting, grouping and timings (optional)'.

4) Далее сохраняйте правило через кнопку сверху справа в углу Save rule and exit.

## 9. Заключение

Теперь вы создали полноценную систему мониторинга серверов, которая позволяет вам отслеживать важные метрики в режиме реального времени, визуализировать их в Grafana и мгновенно получать уведомления в Telegram в случае сбоя сервера.