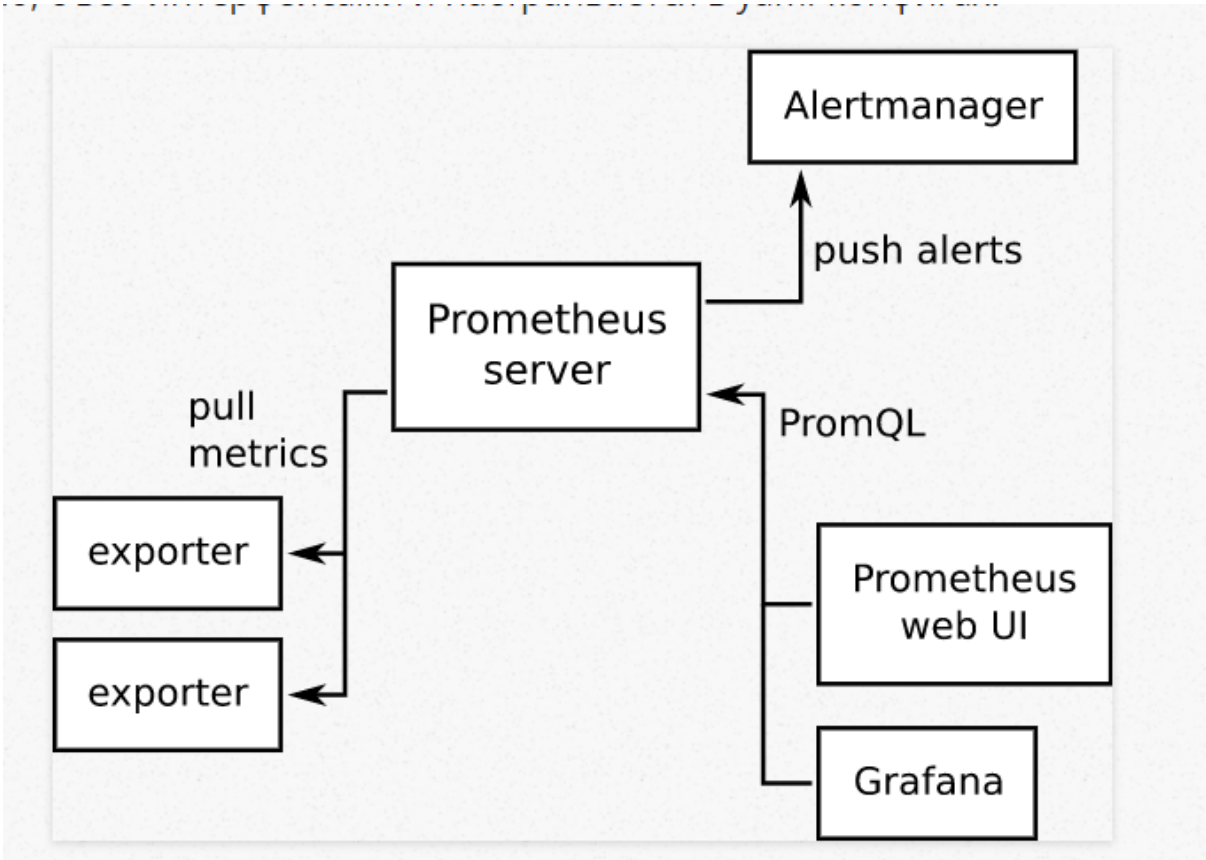
Что будет в лабе?

Простыми словами суть лабы – нам надо собрать метрики на врм1 и на врм2. Есть 4 сущеность, которые собирают метрики.

- node_exporter – чтобы собирать метрики с машины, если зайти на localhost:9100, то там будут все метрики, которые собрал этот node_exporter
- prometeus – ходит на порт 9100 и собирает метрики, которые выдает node_exporter, и кладет все метрики в одно место.
- grafana – все, что насобирал prometeus она красиво рисует.
- alert meneger – мониторит условия (например сервер не работает), и если условие выполняется, то выполняет какое-либо действие (напрмер, написать сообщение в телеграмм).

Что делает в лабе?
На первой машине запускаем node_exporter, который собирает метрики с врм1. на второй машине запускаем node_exporter, который собирает метрики на врм2. На первой машине запускаем prometeus, который собирает врм1 и к врм2 и собирает метрики в одно место (кладется в базу данных prometeus). На врм запускаем grafana, которая подключается к prometeus врм1.И на первой запускаем alurt meneger, который шлет сообщение в телеграмм.
https://laurvas.ru/prometheus-p1/



1. Вернуть интернет на врм2

sudo iptables -F
– чистит iptables , убирает оттуда все правила.

2. Установка всех 4х сущностей.
   cd Downloads
   wget
   https://github.com/prometheus/prometheus/releases/download/v2.42.0/prometheus-2.42.0.linux-amd64.tar.gz

   Если ошибка, что система read-only, то пересоздаем машины)

```
tar xf prometheus-*.tar.gz
cd prometheus-*
cp prometheus promtool /usr/local/bin
mkdir /etc/prometheus /var/lib/prometheus
cp prometheus.yml /etc/prometheus
useradd --no-create-home --home-dir / --shell /bin/false prometheus
chown -R prometheus:prometheus /var/lib/prometheus
```

systemd – системный демон линукса, который запускает сервисы. Нам надо создать команды start, stop, restart для всех 4х сущностей, чтобы можно было их удобно запускать. (по заданию все сущности должны работать с помощью systemd)

3. Создание команд управления

Создаем файл `/etc/systemd/system/prometheus.service` через sudo и редактируем его.
По заданию “Prometheus должен иметь циклическую запись 10-20 дней и/или ограничение по размеру” – мы делаем циклическую запись 10 дней., поэтому добавляем еще один флаг (--storage.tsdb.retention.time=10d)

```
[Unit]
Description=Prometheus
Wants=network-online.target
After=network-online.target

[Service]
User=prometheus
Group=prometheus
Restart=on-failure – если что-то сломается, то будет перезапускаться
ExecStart=/usr/local/bin/prometheus \
  --config.file /etc/prometheus/prometheus.yml \
  --storage.tsdb.path /var/lib/prometheus/ \
  --storage.tsdb.retention.time=10d – что будет запускать
ExecReload=/bin/kill -HUP $MAINPID
ProtectHome=true
ProtectSystem=full

[Install]
WantedBy=default.target
```

4. Перезагружаем системный демон, чтобы он прочитал новый конфиг файл.

```
# systemctl daemon-reload
# systemctl start prometheus
# systemctl status prometheus
```

5. Делаем запуск автоматически при запуске системы

sudo systemctl enable prometheus.service

6. Запускаем и проверяем статус

```
systemctl start prometheus.service
systemctl status prometheus.service
```

7. Чтобы посмотреть, что происходит в prometheus
    a. Открываем порт (приватный) 9090 через Базис, публичный(9096, любой)
    b. http://176.118.65.63:9096
    c. У нас откроется прометеус

8. Ставим node exporter

если # – то sudo

```
$ wget
https://github.com/prometheus/node_exporter/releases/download/v1.5.0/node_export
er-1.5.0.linux-amd64.tar.gz
$ tar xf node_exporter-1.5.0.linux-amd64.tar.gz
# cd node_exporter-*
# cp node_exporter /usr/local/bin
# useradd --no-create-home --home-dir / --shell /bin/false node_exporter
```

    a. Создаем сервер в systemd

Создаем файл /etc/systemd/system/node_exporter.service
Копируем в файл

```
[Unit]
Description=Prometheus Node Exporter
After=network.target

[Service]
Type=simple
User=node_exporter
```

```
Group=node_exporter
ExecStart=/usr/local/bin/node_exporter

SyslogIdentifier=node_exporter
Restart=always

# PrivateTmp=yes
# ProtectHome=yes
# NoNewPrivileges=yes

# ProtectSystem=strict
# ProtectControlGroups=true
# ProtectKernelModules=true
# ProtectKernelTunables=yes

[Install]
WantedBy=multi-user.target
```

b. По аналогии перезагружаем демона

```
# systemctl daemon-reload
# systemctl start node_exporter
# systemctl status node_exporter
```

c. Для проверки

```
curl http://localhost:9100
```

и там будут метрики, которые он насобирал

9. Настраиваем прометеус, чтобы он собирал метрики от node expoter
   a. Заходим в конфиг прометеуса

```
/etc/prometheus/prometheus.yml
```

Если он создан базово, то он нам пока подходит.

10. Устанавливаем алерт менеджер

если # — то sudo

```
$ wget
https://github.com/prometheus/alertmanager/releases/download/v0.25.0/alertmanage
r-0.25.0.linux-amd64.tar.gz
$ tar xf alertmanager-0.25.0.linux-amd64.tar.gz
# cd alertmanager-*
# cp alertmanager /usr/local/bin
# mkdir /etc/alertmanager /var/lib/alertmanager
# cp alertmanager.yml /etc/alertmanager
# useradd --no-create-home --home-dir / --shell /bin/false alertmanager
# chown -R alertmanager:alertmanager /var/lib/alertmanager
```

a. Создаем демона /etc/systemd/system/alertmanager.service\

b. Пишем туда

```
[Unit]
Description=Alertmanager for prometheus
After=network.target

[Service]
User=alertmanager
ExecStart=/usr/local/bin/alertmanager \
  --config.file=/etc/alertmanager/alertmanager.yml \
  --storage.path=/var/lib/alertmanager/
ExecReload=/bin/kill -HUP $MAINPID

NoNewPrivileges=true
ProtectHome=true
ProtectSystem=full

[Install]
WantedBy=multi-user.target
```

c. Запускаем сервер

```
# systemctl daemon-reload
# systemctl start alertmanager
# systemctl status alertmanager
```

d. Для проверки делаем status

11. Подключаемся ко 2 машине, чтобы скачать все там.

12. Ставим node exporter как и для первой машины (можно не комментить сроки)
13. Ставим grafana
    a. Скачиваем по этой ссылке

```
wget https://dl.grafana.com/oss/release/grafana_9.4.3_amd64.deb
```

b. Устанавливаем

```
sudo apt install ./grafana_9.4.3_amd64.deb
```

Везде нажимаем ОК в конце cancel

c. Запускаем

d. Как попасть в графана?

Через порт 3000, поэтому на второй машине прокидываем порт 3000 (локальный), публичный – любой.

e. В баузере на локальной машине http://176.118.165.63:3000

14. Настройка прометеуса на врм1
    a. Меняем его кофиг так, чтобы он ходил и на первую машину и на вторую

b. Заходим в файл /etc/systemd/system/prometheus.service
c. Все что было комментируем и вставляем новое

```
global:
  scrape_interval:     5s – интервал в которой прометеус будет собирать
метрики
  evaluation_interval: 5s

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

scrape_configs:
  - job_name: 'node'
    static_configs:
    - targets:
       - localhost:9100
       - 192.168.3.11:9100 – локал ip второй машины
```

d. Перезапустить прометеус
e. Зайдем на прометеус на локальной машине (через апти и хост). Можно поиском найти node_load1 и посмтреть загрузку процессора на двух машинах.

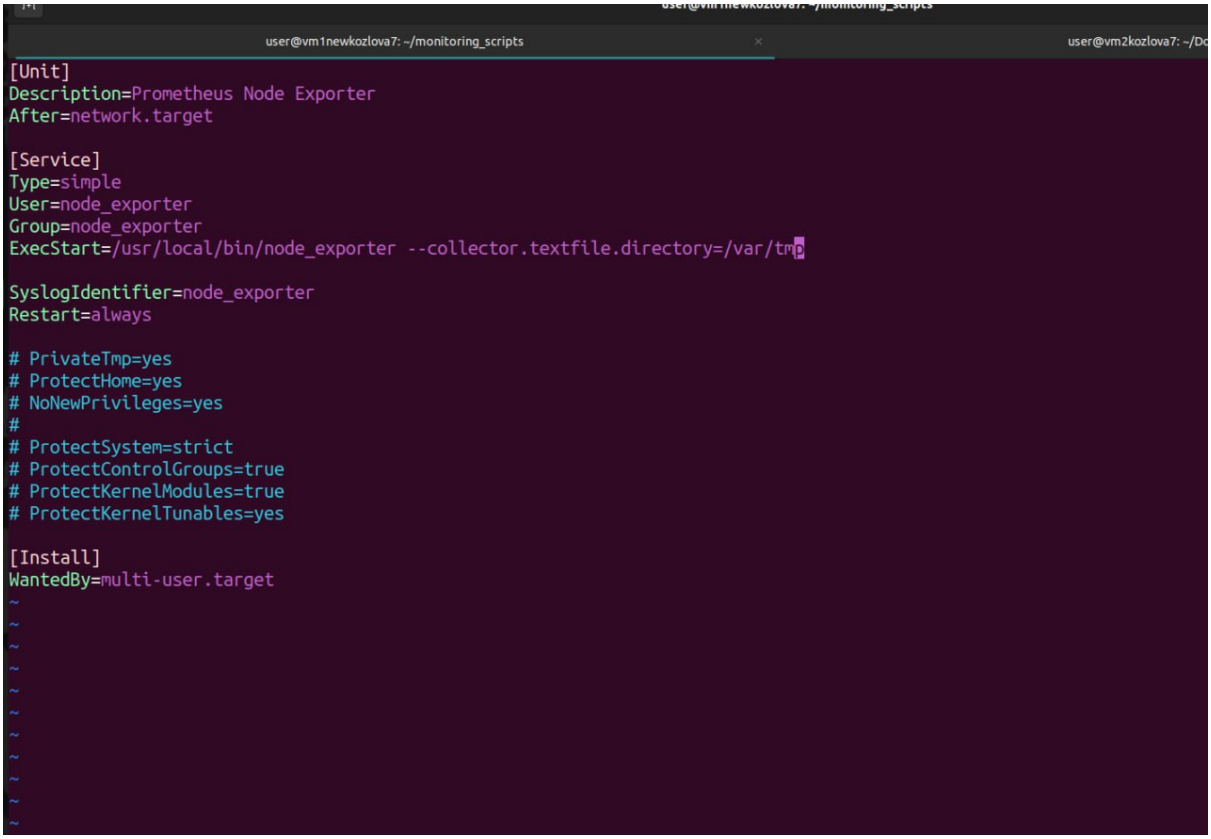
15. Надо построить красивый график в графане
    a. заходим admin admin
    b. предлагает новый пароль придумать, но можно и пропустить
    c. Нужно графану привязать к прометеусу
        i. Настройки
        ii. датасурсер
        iii. добавить датасурсер
        iv. выбираем прометеус
        v. url = http://192.168.3.10:9090
           srcape interval = 5s (необязательно)
    d. В графане будем отображать все стандартный характеристи с двух node_expoter
        i. Дашборт
        ii. импорт
        iii. Вводим 1860 – стандартный дашборд графаны для этих node_exporter
        iv. Загрузить
        v. Выбираем прометеус. который мы добавили
        vi. Импорт
        vii. мы видим все метрики, которые насобирали node_exporter с двух машин

16. ЧТобы собирать метрики с nginx мы должны сказать node_exporter читать данные из файла, который нужен нам. Так как файла с метриками nginx нет, то мы его создаем.
17. Создаем такой файл

a. Создаем папку (название любое)
   mkdir monitoring_scripts
   cd monitoring_scripts/
   vim monitor_nginx.py
b. В файле

```
import os
import re

STATUS_FILE = "/home/user/monitoring_scripts/nginx_status"
METRICS_FILE = "/var/tmp/nginx.prom"

os.system(f"systemctl status nginx > {STATUS_FILE}")

state = 1
tasks = 0
memory = 0
cpu = 0
with open(STATUS_FILE) as status_file:
    status = status_file.read()

    m = re.search(r"Active: (\w+)", status)
    state = int(m.group(1) == "active")
    if (state):
        m = re.search(r"Tasks: (\d+)", status)
        tasks = int(m.group(1))
        m = re.search(r"Memory: (\d+\.?\d*)", status)
        memory = float(m.group(1))

with open(METRICS_FILE, "w") as metrics_file:
    metrics_file.write(f"nginx_state {state}\n")
    metrics_file.write(f"nginx_tasks {tasks}\n")
    metrics_file.write(f"nginx_memory {memory}\n")
    metrics_file.write(f"nginx_cpu {cpu}\n")
```

c. Надо указать node_expoter читать созданный нами файл
   i. Заходим в файл node_expoter.service (выше местоположение)
   ii. Изменять файл как тут



iii. Перезапускаем systemd (выше есть команда)
iv. Перезапускаем node_exporter
v. Проверяем сатус и что есть есть флаг, кооторый мы указали
vi. Ждем 5 секунд
vii. Заходим в прометеус на локальной машине
viii. И можно посмотреть nginx метрики, которые мы создали

18. Нам надо пересоздавать наш файл, чтобы получать метрики.
    a. В линуксе есть cron , который запускает скрипты по определенному времени.
    b. Создаем cron
        i. crontab -e
        ii. выбираем редактор (nano)
        iii. Пишем команду
        * * * * python3 /home/user/monitoring_srcripts/nginx_monitor.py
    c. Мы написали – каждую минуту вызывается скрипт
    d. Мы хотим каждые 10 секунд. Обманываем cron
        * * * * sleep 10; python3 /home/user/monitoring_scripts/nginx_monitor.py
        * * * * sleep 20; python3 /home/user/monitoring_scripts/nginx_monitor.py
        * * * * sleep 30; python3 /home/user/monitoring_scripts/nginx_monitor.py
        * * * * sleep 40; python3 /home/user/monitoring_scripts/nginx_monitor.py
        * * * * sleep 50; python3 /home/user/monitoring_scripts/nginx_monitor.py
    e. Проверяем, что все работает
    f. Идем в прометеус на локальной машине
    g. находим nginx_state
    h. на врм1 останавливаем nginx и смотрит на локальной машине как сроко появится у нас 0
    i. УРА мы собрали метрики nginx (запускаем nginx)

19. Надо собрать метрики размера папки прометеус
    Для этого мы создаем скрипт monitor_prometheus.sh – как угодно файл можно назвать

```
#!/bin/bash

OUTPUT_FILE="/var/tmp/prometheus_size.prom"
SIZE=$(du -ms /var/lib/prometheus | grep -oE "[0-9]+")
printf "#HELP prometheus folder size counter\n#TYPE
prometheus_folder_size\nprometheus_folder_size %d\n" $SIZE >
$OUTPUT_FILE
```

    a. Надо сделать его исполняемым

chmod +x ./monitor_prometeus.sh

    b. Заходим в прометеус на локальной машине и смотрим, появилсь ли у нас наша метрика prometheus_filder_size
    c. Делаем запуск этого скрипта каждые 10 секунд с помощью cron
    d. * * * * * <путь до файла monitor_prometeus.sh>
    e. Делаем слипы аналогично питону

20. Надо, чтобы метрики отображались в графане
    a. Идем в графану
    b. Дашборд
    c. Нью дашборд
    d. add a new panel
    e. Выбираем метрику размера папки прометеус
    f. Запускаем, у нас метрика только началась собираться, поэтому будет совсем немного
    g. Пишем название (любое)
    h. Описание заполняем тоже (можно и нет)
    i. apply
    j. Метрика будет расти и пишется в мегабайтах
    k. Add panel
    l. add new panel
    m. выбираем nginx_state
    n. делаем ее не графиком (а что угодно)
    o. Делаем заголовок
    p. add new panel
    q. выбираем nginx memory
    r. Add new panel
    s. выбираем nginx_tasks
    t. Сделаем ее например бубликом

ПАПКА ПРОМЕТЕУСА – РАЗМЕР БД ПРОМЕТЕУСА

Для метрик хотса используем готовые

21. Надо сделат так, чтобы нам приходили сообщения о падении nginx

22. Заходим на статью https://1cloud.ru/help/monitoring_system_helps/receive_alerts_in_telegram и находим там бота и запускаем его
23. Создаем /etc/alertmanager/alertmanager.yml
    a. пишем туда

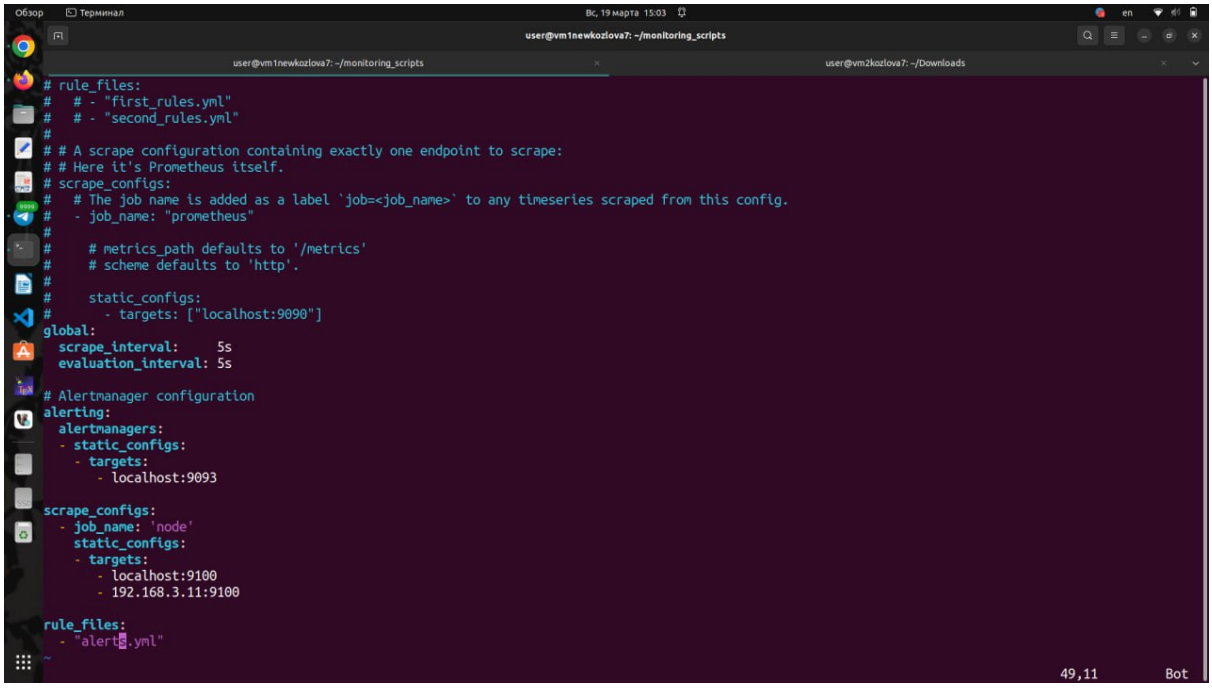
72da2f – токен от бота

```
route:
  group_by: ['alertname']
  group_wait: 10s
  group_interval: 10s
  repeat_interval: 1h
  receiver: 'telepush'
receivers:
  - name: 'telepush'
    webhook_configs:
      - url: 'https://telepush.dev/api/inlets/alertmanager/72da2f
```

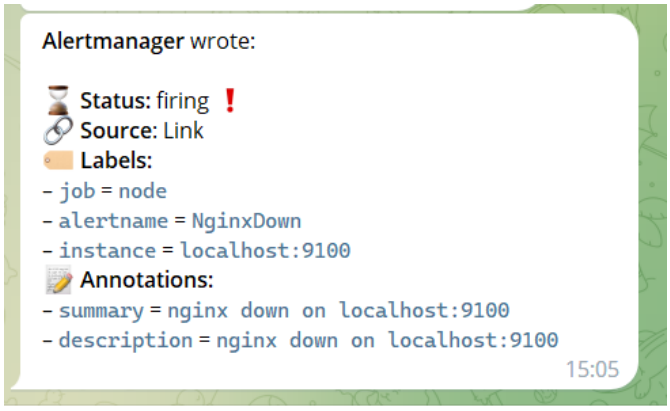
    b. Перезапускаем сервер алертменеджер
24. Надо сказать прометеусу, какие состояния нам надо присылать
25. Создаем файл
    /etc/prometheus/alerts.yml
    Вставляем туда

```
groups:
- name: example
  rules:

  # Alert for any instance that has a median request latency >1s.
  - alert: NginxDown
    expr: nginx_state{instance="localhost:9100"} == 0
    annotations:
      summary: "nginx down on {{ $labels.instance }}"
      description: "nginx down on {{ $labels.instance }}"
```

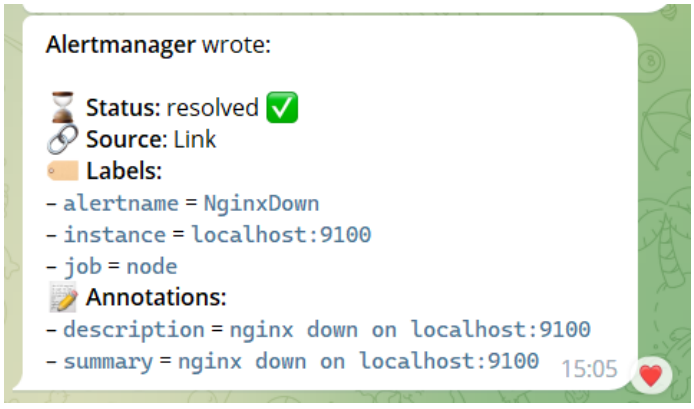
26. Надо прометеусу указать этот файл (у нас есть файл с правилами)
    Файл /etc/prometheus/prometheus.yml
    меняем



27. Перезагружаем прометеус
28. Для проверки – делаем остановку nginx
    и нам приходит сообщение



29. Запускаем nginx



30. ДУДОСИМ (при сдаче лабы тоже)
31. Скачиваем (на врм1)

```
sudo apt install apache2-utils
```

32. Скачиваем stress

```
sudo apt install stress
```

33. На врм2 скачиваем стресс

34. Стресс много че умеет))))))))))))

35. На врм2 запускаем нагрузку
    `stress -c 1 -i 2`

дудосится одно ядро и 2 i/o

36. На врм1 дудосим nginx
    ab -c 10 -n 100000 http://localhost/

    если быстро работает, то
    ab -c 100 -n 10000000 http://localhost/

37. Смотрим метрики на графане

ВСЕЕЕ