

Мониторинг: Prometheus, часть 2



Артур

Сагутдинов



Артур Сагутдинов

Инженер DevOps
департамента голосовых
цифровых технологий

Banks Soft Systems



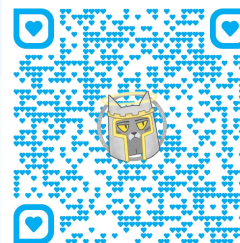
15+ лет в сфере ИТ



Разрабатываю и внедряю
линуксовую инфраструктуру



[Сисадминский блог](#)

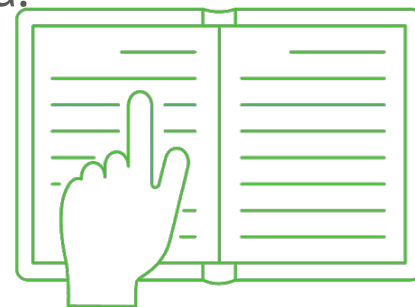


https://t.me/belf_igor

Предисловие

На этом занятии мы:

- продолжим освоение Prometheus;
- поговорим об экспортерах;
- узнаем, что такое инструментирование;
- установим Alertmanager;
- поставим Docker на мониторинг в Prometheus;
- узнаем, как создавать свои Dashboard в Grafana.





План занятия

1. [Alertmanager](#)
2. [Экспортеры](#)
3. [Инструментирование](#)
4. [Мониторинг Docker с помощью Prometheus](#)
5. [Создание своего Dashboard с помощью Grafana](#)
6. [Итоги](#)
7. [Домашнее задание](#)



Alertmanager

Alertmanager

Alertmanager — это ПО, которое позволяет:

- обрабатывать оповещения, отправляемые из клиентских приложений,
- консолидировать эти оповещения;
- перенаправлять их по необходимым каналам доставки сообщений.

Классическая установка Alertmanager

Качаем последнюю версию приложения с [GitHub](https://github.com/prometheus/alertmanager/releases/download/v0.22.2/alertmanager-0.22.2.linux-amd64.tar.gz). У нас 64 битный виртуальный сервер, потому мы будем использовать amd64 и сразу же извлекаем её:

```
wget  
https://github.com/prometheus/alertmanager/releases/download/v0.22.2/alertmanag  
er-0.22.2.linux-amd64.tar.gz  
tar -xvf alertmanager-*linux-amd64.tar.gz
```

Копируем содержимое архива в нужные папки:

```
sudo cp ./alertmanager-*linux-amd64/alertmanager /usr/local/bin  
sudo cp ./alertmanager-*linux-amd64/amtool /usr/local/bin
```

Классическая установка Alertmanager

Config файл копируем в папку Prometheus:

```
sudo cp ./alertmanager-*.linux-amd64/alertmanager.yml /etc/prometheus
```

Передаём пользователю Prometheus права на файл:

```
sudo chown -R prometheus:prometheus /etc/prometheus/alertmanager.yml
```


Создаём сервис для работы с Node Exporter

```
nano /etc/systemd/system/prometheus-alertmanager.service
```

Вставляем в файл сервиса следующее содержимое:

```
[Unit]
Description=Alertmanager Service
After=network.target
[Service]
EnvironmentFile=-/etc/default/alertmanager
User=prometheus
Group=prometheus
Type=simple
ExecStart=/usr/local/bin/alertmanager
--config.file=/etc/prometheus/alertmanager.yml
--storage.path=/var/lib/prometheus/alertmanager $ARGS
ExecReload=/bin/kill -HUP $MAINPID
Restart=on-failure
[Install]
WantedBy=multi-user.target
```

Тестируем сервис Alertmanager

Прописываем автозапуск:

```
sudo systemctl enable prometheus-alertmanager
```

Запускаем сервис:

```
sudo systemctl start prometheus-alertmanager
```

Проверяем статус сервиса:

```
sudo systemctl status prometheus-alertmanager
```

Подключаем Prometheus к Alertmanager

В Config файл Prometheus добавляем подключение к Alertmanager:

```
sudo nano /etc/prometheus/prometheus.yml
```

Нужно привести раздел # Alertmanager configuration к виду:

```
alerting:
  alertmanagers:
    - static_configs:
      - targets: # Можно указать как targets: ['localhost"9093']
        - localhost:9093
```

localhost:9093 — это адрес локального Alertmanager.

Перезапускаем Prometheus:

```
sudo systemctl restart prometheus
```

Создаём правило оповещения

Создадим файл с конфигурацией оповещения:

```
sudo nano /etc/prometheus/netology-test.yml
```

Нужно привести файл к следующему виду:

```
groups:
- name: netology-test
  rules:
  - alert: InstanceDown
    expr: up == 0 # Если лежит
    for: 1m # 1 минуту и более
    labels:
      severity: critical # Критический статус
    annotations: # Описание
      description: '{{ $labels.instance }} of job {{ $labels.job }} has been
down
      for more than 1 minute.'
    summary: Instance {{ $labels.instance }} down
```

Подключаем правило к Prometheus

Для этого отредактируем prometheus.yml:

```
sudo nano /etc/prometheus/prometheus.yml
```

Добавим в раздел rule_files запись:

```
- "netology-test.yml"
```

Перезапустим Prometheus:

```
systemctl restart prometheus
```

Настроим оповещения в Alertmanager

Откроем Config файл Alertmanager:

```
sudo nano /etc/prometheus/alertmanager.yml
```

И приведём к следующему виду:

```
global:
route:
  group_by: ['alertname']
  group_wait: 30s
  group_interval: 10m
  repeat_interval: 60m
  receiver: 'email'
receivers:
- name: 'email'
  email_configs:
  - to: 'yourmailto@tomain.com'
    from: 'yourmailfrom@fromdomain.com'
    smarthost: 'mailserver:25'
    auth_username: 'user'
    auth_identity: 'user'
    auth_password: 'paS$w0rd'
```

Проверка оповещений Alertmanager

После внесения всех изменений, перезапустим Alertmanager и выключим экспортер, стоящий на сервере Prometheus:

```
sudo systemctl restart prometheus-alertmanager  
sudo systemctl stop node-exporter
```

Теперь можно проверить интерфейсы Prometheus и Alertmanager, расположенные на стандартных портах 9090 и 9093.

Проверка оповещений Alertmanager

Prometheus Alerts Graph Status ▾ Help Classic UI

✓ Inactive (0)

✓ Pending (0)

✓ Firing (1)

☐ Show annotations

/etc/prometheus/netology-test.yml > netology-test **firing (1)**

▼ InstanceDown (1 active)

name: `InstanceDown`
expr: `up == 0`
for: 1m
labels:
 severity: critical
annotations:
 description: {{ \$labels.instance }} of job {{ \$labels.job }} has been down for more than 1 minute.
 summary: Instance {{ \$labels.instance }} down

Labels	State	Active Since	Value
<code>alertname=InstanceDown</code> <code>instance=localhost:9100</code> <code>job=prometheus</code> <code>severity=critical</code>	FIRING	2021-07-11T13:35:53.373996788Z	0

Проверка оповещений Alertmanager

Alertmanager

Alerts

Silences

Status

Help

New Silence

Filter

Group

Receiver: All

☐ Silenced

☐ Inhibited

+

Silence

Custom matcher, e.g. `env="production"`

+ Expand all groups


alertname="InstanceDown"


+

1 alert

2021-07-11T13:36:53.373Z

+ Info

 Source

 Silence

instance="localhost:9100"

+

job="prometheus"

+

severity="critical"

+



Экспортеры



Экспортеры

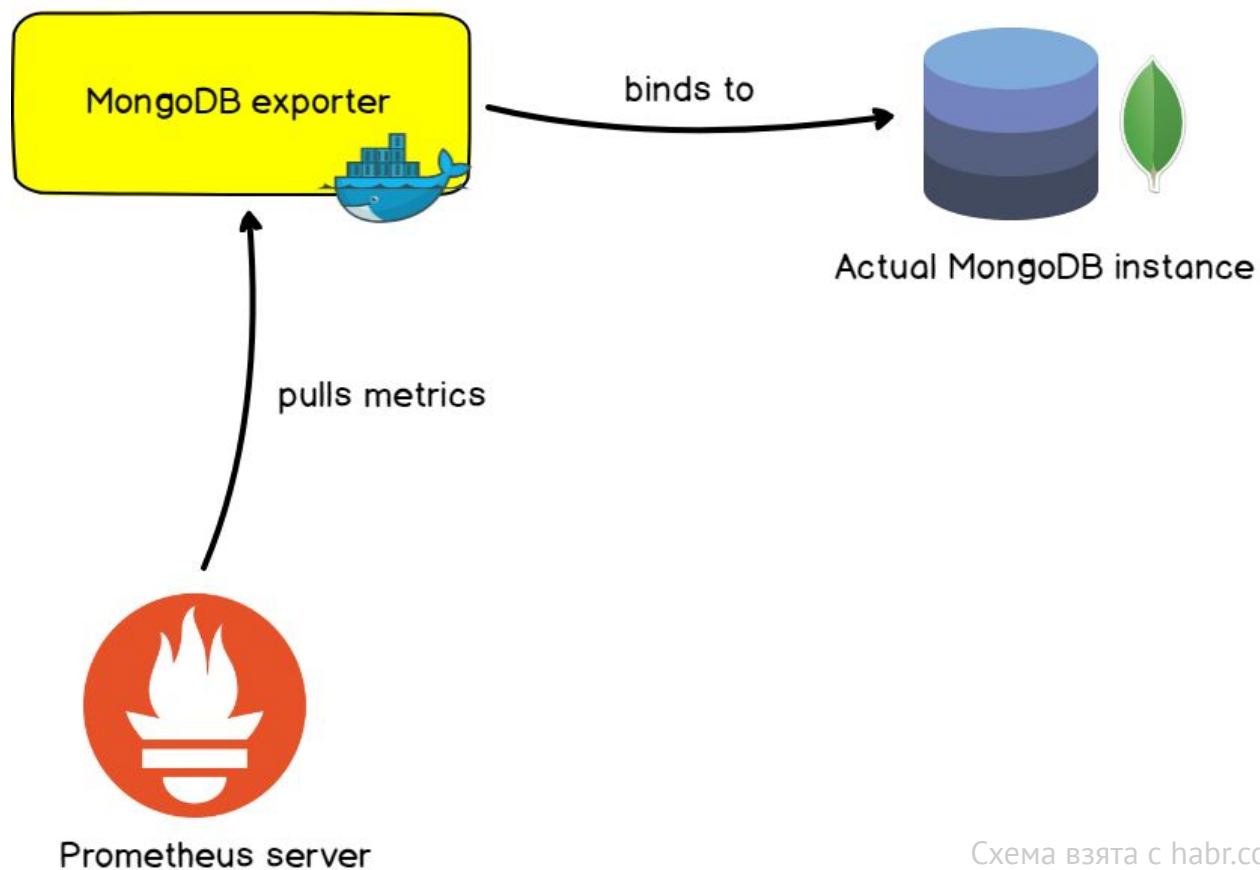
Мы уже сталкивались с Node Exporter. Это был лишь один из множества представителей семейства экспортеров.

Экспортер — это набор predetermined метрик, силами которого можно поставить совместимую с ним среду\приложение на мониторинг, не прибегая к излишним ухищрениям.

Экспортеры бывают: как в виде отдельных приложений, например, то, которое мы качали для постановки на мониторинг нашего сервера с Prometheus, так и в виде Docker контейнеров.

Экспортеры

Exporters with Prometheus





Инструментирование



Инструментирование

Инструментированием в разрезе Prometheus называется процесс доработки какого-либо приложения таким образом, чтобы оно предоставляло по заданному URL адресу метрики, с которыми может работать Prometheus.

Это очень важный функционал, дающий возможность извлекать данные из разрабатываемых вами приложений максимально простым способом, и в дальнейшем отслеживать их с помощью Grafana. При этом вы не будете тратить время на разработку своей собственной системы мониторинга с нуля.

Инструментирование

Prometheus официально поддерживает библиотеки инструментирования для языков Python, Java, Ruby, Go, Node и C#.

С помощью сторонних библиотек можно инструментировать и приложения на других языках.

Instrumenting your app

Official client libraries



Third-party client libraries





Мониторинг Docker в Prometheus

Для начала установим репозиторий Docker

Устанавливаем пакеты для работы apt через HTTPS:

```
# Обновляем кеш
sudo apt update
# Устанавливаем необходимые пакеты
sudo apt install apt-transport-https ca-certificates curl gnupg lsb-release -y
```

Добавляем GPG ключ:

```
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o \
    /usr/share/keyrings/docker-archive-keyring.gpg
```

Установим Docker Engine

Добавляем stable репозиторий для x86_64 / amd64:

```
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
\
    https://download.docker.com/linux/debian $(lsb_release -cs) stable" | sudo
\
    tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Устанавливаем Docker Engine:

```
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io -y
```

Запускаем Docker:

```
sudo systemctl enable docker
sudo systemctl start docker
sudo systemctl status docker
```

Установка Docker Compose

1. Скачиваем последний стабильный релиз из репозитория:

```
sudo curl -L  
"https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(una  
me -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

2. Устанавливаем права на запуск:

```
sudo chmod +x /usr/local/bin/docker-compose
```

3. Проверяем, что всё работает:

```
docker-compose --version
```

Мониторинг Docker в Prometheus

Docker из коробки поддерживает мониторинг с помощью Prometheus. Для того чтобы включить выгрузку данных на хосте с Docker, нужно создать файл `daemon.json`:

```
sudo nano /etc/docker/daemon.json
```

И указать внутри него `ip` и порт, на котором Docker будет отдавать метрики. Указываем в формате «`server_ip:port`»

```
{  
  "metrics-addr" : "ip_нашего_сервера:9323",  
  "experimental" : true  
}
```

Перезапустим Docker:

```
sudo systemctl restart docker
```

Для проверки можно открыть адрес `http://server_ip:port/metrics`

Добавим endpoint Docker в Prometheus

Чтобы поставить только что организованный нами endpoint на мониторинг, нам необходимо отредактировать файл prometheus.yml:

```
sudo nano /etc/prometheus/prometheus.yml
```

В раздел static_configs добавим новый endpoint.

Это может выглядеть так:

```
static_configs:  
- targets: ['localhost:9090', 'localhost:9100', 'server_ip:9323']
```

Или так:

```
static_configs:  
- targets:  
  - localhost:9090  
  - localhost:9100  
  - server_ip:9323
```

Перезапустим prometheus

```
sudo systemctl restart prometheus
```



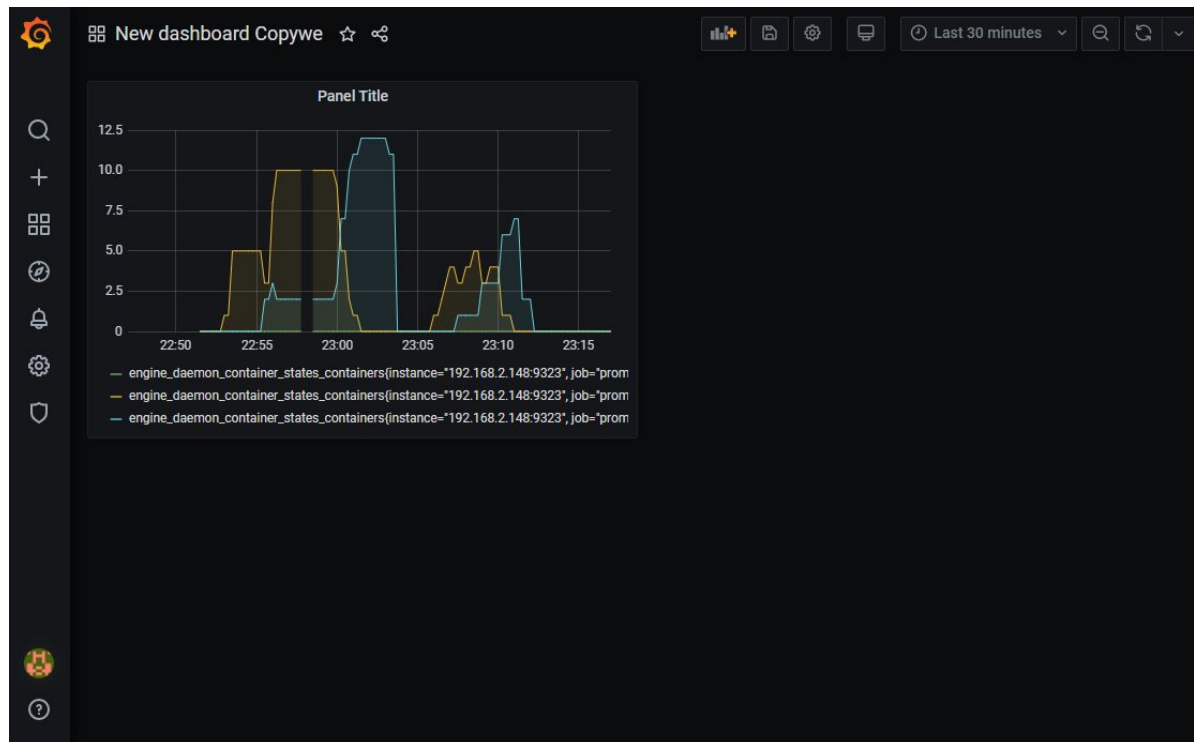
Создание своего Dashboard с помощью Grafana

Настраиваем свой Dashboard

- В интерфейсе Grafana нажимаем на **+** и выбираем **Dashboard**;
- Нажимаем на **+** **Add new panel**;
- В выпадающем меню **Metrics** выбираем: engine > engine_daemon_container_states_container;
- Нажимаем на Apply и вываливаемся в интерфейс Dashboard;
- Остаётся сохранить Dashboard, чтобы позже к нему вернуться

Настраиваем свой Dashboard

В итоге мы добавим в наш Dashboard одну панельку с графиком. Дальнейшие манипуляции ограничиваются лишь вашей фантазией.





Итоги

Итоги

Сегодня мы:

- научились устанавливать Alertmanager и интегрировать Alertmanager и Prometheus;
- создали своё правило для оповещения;
- получили метрики из Docker;
- добавили метрику Docker на свой Dashboard.





Домашнее задание

Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачу можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты задача полностью**.

**Задавайте вопросы и
пишите отзыв о лекции!**

Артур Сагутдинов



https://t.me/belf_igor

