VK WorkSpace

Корпоративный мессенджер VK Teams

Инструкция по подключению к внешней системе мониторинга

Оглавление

Назначение документа

Система предоставляет возможность настроить сбор метрик внешней системой мониторинга с сервиса Prometheus VK Teams. Сбор метрик с осуществляется через механизм Federation.

В данном документе рассматривается настройка подключения к следующим внешним системам мониторинга:

1. VictoriaMetrics (см. раздел Настройка подключения к VictoriaMetrics). Документация по продукту — https://docs.victoriametrics.com/.
2. Prometheus (см. раздел Настройка подключения к внешнему Prometheus). Документация по продукту — https://prometheus.io/docs/introduction/overview/, https://prometheus.io/docs/alerting/latest/configuration/.

Визуализация собранных метрик возможна при помощи сервиса VK Teams Grafana или при помощи внешнего сервиса Grafana. Документация по продукту представлена https://grafana.com/docs/grafana/latest.

Документ предназначен для использования системные администраторами.

Дополнительная документация

Мониторинг VK Teams — в документе представлено описание мониторинга параметров инсталляции при помощи встроенных сервисов мониторинга (VictoriaMetrics VK Teams и Grafana VK Teams).

Подготовка к установке

Настройку сбора метрик внешней системой мониторинга можно произвести как при первой инсталляции VK Teams, так после завершения установки.

Перед установкой/настройкой сбора метрик внешней системой мониторинга необходимо разрешить доступ с сервера мониторинга заказчика до инсталляции VK Teams:

1. Прописать в файле **/usr/local/etc/premsetup/defaults.yaml** IP-адрес сервера мониторинга, к которому будет обращаться сервис Prometheus VK Teams. Управление осуществляется через параметр myteam_admin_acl:

   ```
   myteam_admin_acl:
     - '10.11.12.102'
   ```

2. Применить указанные настройки:

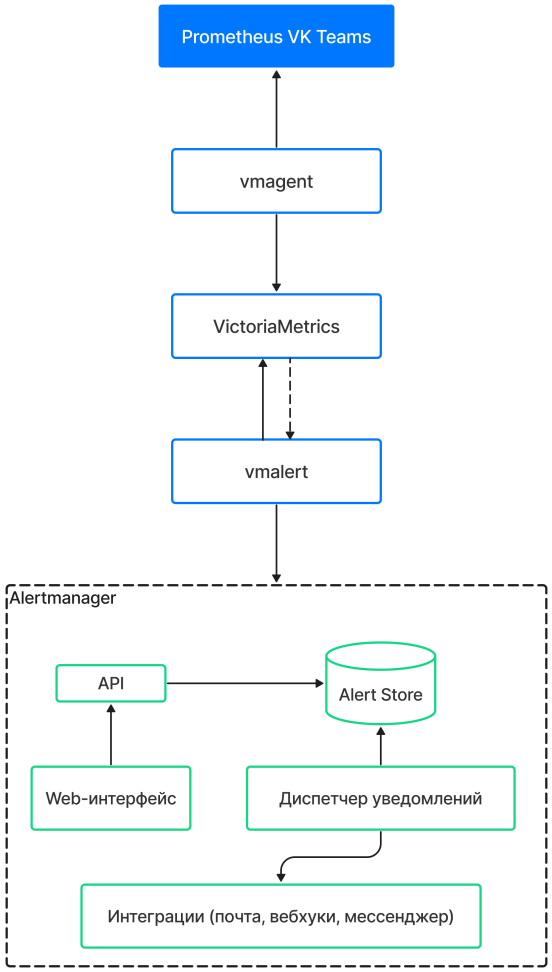
   ```
   premsetup.py --install -m myteam-admin
   ```

3. Перезапустить сервис Nginx:

   ```
   nginx.sh restart
   ```

Настройка подключения к VictoriaMetrics

Схема взаимодействия Prometheus VK Teams с сервисами VictoriaMetrics и Alertmanager представлена ниже:



vmagent собирает метрики от Prometheus VK Teams и сохраняет их в VictoriaMetrics.

vmalert — агрегатор алертов, который хранит правила уведомлений об изменении метрик. Делает запросы в TSDB VictoriaMetrics и отправляет по API информацию о метриках в сервис Alertmanager, который в зависимости от конфигурации перенаправляет уведомления с помощью настроенного способа оповещения.

Для установки VictoriaMetrics можно воспользоваться официальным репозиторием https://github.com/VictoriaMetrics/VictoriaMetrics.

Пример установки и настройки с использованием Docker Compose и установки в режиме single:

1. Клонировать репозиторий https://github.com/VictoriaMetrics/VictoriaMetrics.

2. В файле **VictoriaMetrics/deployment/docker/prometheus.yml** в секцию scrape_configs: добавить:

```
scrape_configs:
  - job_name: 'onpremise-federate-DOMAIN'
    scheme: https
    static_configs:
      - targets: ['admin.DOMAIN']
    metrics_path: /myteam-prometheus/federate # для версии VK Teams 24.2 и ниже
    metrics_path: /mon/federate # начиная с версии VK Teams 24.3
    params:
      match[]:
        - '{job=~".+"}'
```

, где DOMAIN — ваш домен.

Пример файла **VictoriaMetrics/deployment/docker/prometheus.yml**:

```
global:
  scrape_interval: 15s
  scrape_timeout: 15s

scrape_configs:
  - job_name: onpremise-federate-DOMAIN'
    scheme: https
    static_configs:
      - targets: ["admin.DOMAIN"]
    metrics_path: /myteam-prometheus/federate # для версии VK Teams 24.2 и ниже
    metrics_path: /mon/federate # начиная с версии VK Teams 24.3
    params:
      match[]:
        - '{job=~".+"}'
    # если используется self-signed сертификат на onpremise
    tls_config:
      insecure_skip_verify: true
  - job_name: "vmagent"
    static_configs:
      - targets: ["vmagent:8429"]
  - job_name: "vmalert"
    static_configs:
      - targets: ["vmalert:8880"]
  - job_name: "victoriametrics"
    static_configs:
      - targets: ["victoriametrics:8428"]
```

3. Если не используется сертификат от Let's Encrypt, то необходимо прокинуть ваш корневой сертификат в контейнер vmagent:

   - Создать в директории **deployment/docker** директорию **certs** и положить туда ваш корневой сертификат;
   - В файле **VictoriaMetrics/deployment/docker/docker-compose.yml** добавить параметр **- ./certs/root-ca.crt:/etc/ssl/certs/root-ca.crt** в контейнер vmagent в секцию volume.

Пример сервиса vmagent в **VictoriaMetrics/deployment/docker/docker-compose.yml**:

```
services:
  vmagent:
    container_name: vmagent
    image: victoriametrics/vmagent:v1.90.0
    depends_on:
      - "victoriametrics"
    ports:
      - 8429:8429
    volumes:
      - vmagentdata:/vmagentdata
      - ./prometheus.yml:/etc/prometheus/prometheus.yml
      - ./certs/root-ca.crt:/etc/ssl/certs/root-ca.crt
    command:
      - "--promscrape.config=/etc/prometheus/prometheus.yml"
      - "--remoteWrite.url=http://victoriametrics:8428/api/v1/write"
    networks:
      - vm_net
    restart: always
```

> **Внимание**
>
> Если этого не сделать, то при выполнении запроса на получение метрик scrape будет появляться ошибка **x509: certificate signed by unknown authority.**

4. При необходимости скорректировать параметры scrape_interval или scrape_timeout в файле **VictoriaMetrics/deployment/docker/prometheus.yml.**
5. Для избежания провалов метрик необходимо добавить параметры к запуску контейнера victoriametrics в **VictoriaMetrics/deployment/docker/docker-compose.yml** или **VictoriaMetrics/deployment/docker/docker-compose-cluster.yml**:

```
- "-promscrape.maxScrapeSize=50Mb" // увеличивает размер запросов на получение метрик
 - "-search.minStalenessInterval=60s" // фикс проблемы с провалом метрик
```

Пример сервиса victoriametrics в **VictoriaMetrics/deployment/docker/docker-compose.yml**:

```
services:
  victoriametrics:
    container_name: victoriametrics
    image: victoriametrics/victoria-metrics:v1.90.0
    ports:
      - 8428:8428
      - 8089:8089
      - 8089:8089/udp
      - 2003:2003
      - 2003:2003/udp
      - 4242:4242
    volumes:
      - vmdata:/storage
    command:
      - "-promscrape.maxScrapeSize=50Mb"
      - "-search.minStalenessInterval=60s"
      - "--storageDataPath=/storage"
      - "--graphiteListenAddr=:2003"
      - "--opentsdbListenAddr=:4242"
      - "--httpListenAddr=:8428"
      - "--influxListenAddr=:8089"
      - "--vmalert.proxyURL=http://vmalert:8880"
    networks:
      - vm_net
    restart: always
```

6. Для запуска выполнить команду:

```
docker compose up -d
```

7. Для настройки уведомлений об изменении метрик необходимо в **VictoriaMetrics/deployment/docker/alertmanager.yml** указать route и recivers. Например, при использовании telegram-бота для отправки уведомлений в чат:

```
route:
  receiver: "telegram"
receivers:
  - name: "telegram"
    telegram_configs:
      - bot_token: TELEGRAM_BOT_TOKEN
        api_url: https://api.telegram.org
        chat_id: TELEGRAM_CHAT_ID
        parse_mode: "HTML"
```

8. Для созданий правил уведомлений об изменении метрик создать файл **alerts-onpremise.yml** в директории **deployment/docker** и заполнить его правилами. Пример файла **deployment/docker/alerts-onpremise.yml:**

```
groups:
  - name: onpremise
    interval: 30s
    concurrency: 2
    rules:
      - alert: WARNING
        expr: localcheck_status == 1
```

```
        for: 3m
        labels:
          severity: warning
        annotations:
          summary: Instance {{ $labels.instance }})
          description: "status = {{ $value }} \nLABELS = {{ $labels }}"
      - alert: CRITICAL
        expr: localcheck_status == 2
```

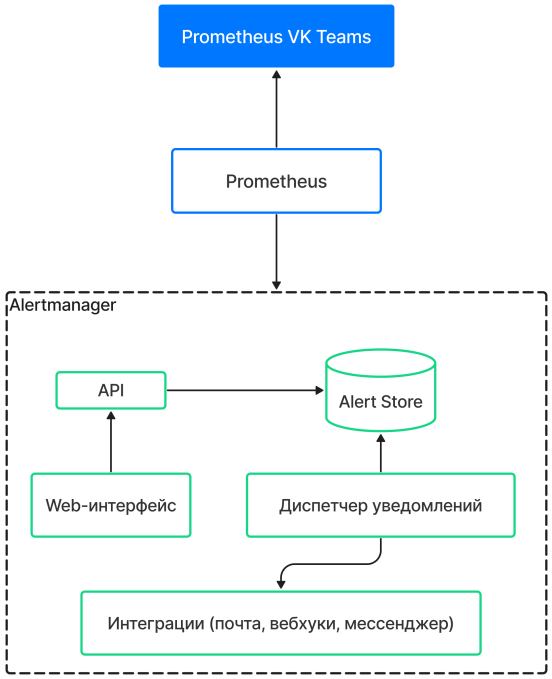
9. В файле в **VictoriaMetrics/deployment/docker/docker-compose.yml** добавить параметр **- ./alerts-onpremise.yml:/etc/alerts/alerts-onpremise.yml** к сервису vmalert в секцию volume.

   Пример сервиса vmalert в **VictoriaMetrics/deployment/docker/docker-compose.yml:**

```
services:
  vmalert:
    container_name: vmalert
    image: victoriametrics/vmalert:v1.90.0
    depends_on:
      - "victoriametrics"
      - "alertmanager"
    ports:
      - 8880:8880
    volumes:
      - ./alerts.yml:/etc/alerts/alerts.yml
      - ./alerts-health.yml:/etc/alerts/alerts-health.yml
      - ./alerts-vmagent.yml:/etc/alerts/alerts-vmagent.yml
      - ./alerts-vmalert.yml:/etc/alerts/alerts-vmalert.yml
      - ./alerts-onpremise.yml:/etc/alerts/alerts-onpremise.yml
```

Настройка подключения к внешнему Prometheus

Схема взаимодействия Prometheus VK Teams с внешним Prometheus и Alertmanager представлена ниже:



TSDB Prometheus собирает метрики от Prometheus VK Teams и отправляет по API информацию об алертах в сервис Alertmanager, который в зависимости от конфигурации перенаправляет уведомления с помощью настроенного способа оповещения.

Для установки Prometheus stack можно воспользоваться репозиторием https://github.com/vegasbrianc/prometheus.

Пример установки с использованием Docker Compose:

1. Клонировать репозиторий https://github.com/vegasbrianc/prometheus.

2. В файле **prometheus/prometheus.yml** в секцию scrape_configs добавить:

```
scrape_configs:
  - job_name: 'onpremise-federate-DOMAIN'
    scheme: https
    static_configs:
      - targets: ["admin.DOMAIN"]
    metrics_path: /myteam-prometheus/federate # для версии VK Teams 24.2 и ниже
    metrics_path: /mon/federate # начиная с версии VK Teams 24.3
    params:
      match[]:
        - '{job=~".+"}'
```

, где DOMAIN — ваш домен.

Пример файла **prometheus/prometheus.yml:**

```
global:
  scrape_interval:   15s
  evaluation_interval: 15s

rule_files:
  - 'alert.rules'

alerting:
  alertmanagers:
  - scheme: http
    static_configs:
    - targets:
      - "alertmanager:9093"

scrape_configs:
  - job_name: onpremise-federate-DOMAIN'
    scheme: https
    static_configs:
      - targets: ["admin.DOMAIN"]
    metrics_path: /myteam-prometheus/federate # для версии VK Teams 24.2 и ниже
    metrics_path: /mon/federate # начиная с версии VK Teams 24.3
    params:
      match[]:
        - '{job=~".+"}'
```

3. Если не используется сертификат от Let's Encrypt, то необходимо прокинуть ваш корневой сертификат в контейнер prometheus:
   - Создать в корне директорию **certs** и положить туда ваш корневой сертификат;
   - В файле **prometheus/docker-compose.yml** добавить параметр **- ./certs/root-ca.crt:/etc/ssl/certs/root-ca.crt** в контейнер prometheus в секцию volume.

Пример сервиса prometheus в **prometheus/docker-compose.yml:**

```
services:
  prometheus:
    image: prom/prometheus:v2.36.2
    volumes:
      - ./prometheus/:/etc/prometheus/
      - prometheus_data:/Prometheus
      - ./certs/root-ca.crt:/etc/ssl/certs/root-ca.crt
command:
      - '--config.file=/etc/prometheus/prometheus.yml'
      - '--storage.tsdb.path=/prometheus'
      - '--web.console.libraries=/usr/share/prometheus/console_libraries'
      - '--web.console.templates=/usr/share/prometheus/consoles'
    ports:
      - 9090:9090
    links:
      - cadvisor:cadvisor
      - alertmanager:alertmanager
    depends_on:
      - cadvisor
    networks:
      - back-tier
    restart: always
```

> **Внимание**
>
> Если этого не сделать то при scrape будет появляться ошибка **x509: certificate signed by unknown authority.**

4. При необходимости скорректировать или добавить параметры scrape_interval или scrape_timeout в файле **prometheus/prometheus.yml.**
5. Для настройки уведомлений об изменении метрик в файле **prometheus/alertmanager/config.yml** указать route и recivers. Например, при использовании telegram-бота для отправки уведомлений в чат:

```
route:
  receiver: "telegram"
receivers:
  - name: "telegram"
    telegram_configs:
      - bot_token: TELEGRAM_BOT_TOKEN
        api_url: https://api.telegram.org
        chat_id: TELEGRAM_CHAT_ID
        parse_mode: "HTML"
```

6. Для созданий правил уведомлений об изменении метрик заполнить файл **prometheus/ prometheus/ alert.rules:**

```
groups:
  - name: onpremise
    interval: 30s
    concurrency: 2
    rules:
      - alert: WARNING
        expr: localcheck_status == 1
        for: 3m
        labels:
          severity: warning
        annotations:
          summary: Instance {{ $labels.instance }})
          description: "status = {{ $value }} \nLABELS = {{ $labels }}"
      - alert: CRITICAL
        expr: localcheck_status == 2
```

7. Для запуска выполнить команду:

```
docker compose up -d
```

Дата обновления документа: 18.04.2024 г.