Руководство прикладного разработчика

компонента Telemetry Collector (Единый коллектор телеметрии) (Код компонента: COTE)

продукта Platform V Monitor (Код продукта: OPM)

ОГЛАВЛЕНИЕ

Руководство прикладного разработчика

Термины и определения

Общие термины и определения, используемые в данном документе, представлены в общей документации продукта Platform V Monitor (OPM).

Ниже приведены специальные термины и определения для компонента Единый коллектор телеметрии (COTE) продукта Platform V Monitor (OPM).

Термин/Аббревиатура	**Определение**
Management application	Компонент Единого коллектора, предоставляющий пользователям возможность управления настройками и мониторинга сбора метрик
Prometheus-агент	Модуль клиентской части Prometheus, который собирает метрики с клиентских приложений
Pull-модель сбора	Модель сбора данных, где инициатором выступает серверная сторона, а не клиентская. Также модель называют **активной**
Push-модель сбора	Модель сбора данных, где клиент отправляет запросы на сервер, а сервер публикует API для их обработки. Также модель называют **пассивной**
Pipeline сбора телеметрических данных	Конвейер обработки информации. Представлен набором компонентов и связей между ними
Zipkin	Распределенная система отслеживания данных в реальном времени

Системные требования

В этом разделе представлены только минимальные требования КТС, прочие требования к окружению и конфигурации представлены в «Руководстве по установке».

Это руководство содержит названия параметров и переменных, содержащих наименование/префикс «ose» или «openshift», которые применимы для различных средств контейнеризации, указанных в документе «Руководство по установке».

Здесь и далее поддерживаемой системой приложений-контейнеров является Kubernetes (использование OSE – опционально).

Минимальные КТС

Сервис	**RAM**	**CPU**	**Кол-во инстансов**
Pipeline Management Application	2048	1	1
Pull Collector	4096	2	2
Push Collector	4096	2	2
Ingress/Egress/Istio/Fluent-bit/Unimon-Agent	4096	2	-

Зависимость пропускной способности от требуемых КТС представлена в отчетах НТ.

Подключение и конфигурирование push-api

Перед началом работы с Единым коллектором ваш проект должен быть зарегистрирован в Abyss, а также в проекте должен быть заведен пользователь с правом TC_PIPELINE_PUSH. Доступ к PUSH API Единого коллектора осуществляется по технологии API Key. Ключ выдается администраторами Abyss. После получения ключа можно обращаться к PUSH API Единого коллектора передавая ключ в заголовке X-PVM-API-KEY. Также необходимо настроить Egress на клиенте для исходящего трафика на PUSH API Единого коллектора.

Сбор Prometheus метрик: Настройка Prometheus агентов (Unimon-agent) в клиентском NameSpace

Все примеры приведены на базе сбора собственных метрик/логов/трассировок!
Чтобы обеспечить сбор прикладных метрик, приложение должно выставить метрики через HTTP-endpoint, например, `/metrics` (путь может быть любым). Prometheus-agent (или unimon-agent) находит HTTP-endpoint, собирает и обогащает метрики, а затем передает их в приложение. Данные предоставляются в формате Actuator Prometheus. Prometheus читает секцию конфигурации scrape_configs, согласно которой настраивает свой внутренний механизм обнаружения сервисов (Service Discovery). Механизм Service Discovery взаимодействует с Kubernetes API (в основном для получения endpoints). На основании данных из Kubernetes механизм Service Discovery обновляет Targets (список целей).

Структура конфигурации:

```
global:
  # How frequently to scrape targets by default.
  [ scrape_interval: <duration> | default = 1m ]

  # How long until a scrape request times out.
  [ scrape_timeout: <duration> | default = 10s ]

  # How frequently to evaluate rules.
  [ evaluation_interval: <duration> | default = 1m ]

  # The labels to add to any time series or alerts when communicating with
  # external systems (federation, remote storage, Alertmanager).
  external_labels:
    [ <labelname>: <labelvalue> ... ]

  # File to which PromQL queries are logged.
  # Reloading the configuration will reopen the file.
```

```
  [ query_log_file: <string> ]


# A list of scrape configurations.
scrape_configs:
  [ - <scrape_config> ... ]


# Settings related to the remote write feature.
remote_write:
  [ - <remote_write> ... ]
```

Global секция определяет параметры, которые действительны во всех других контекстах конфигурации. Они также служат значениями по умолчанию для других разделов конфигурации.

Секция scrape_config определяет список заданий сбора.

Секция relabel_configs позволяет добавлять настройки «фильтрации» – какие endpoints будет брать, а какие — нет, и "relabeling" – какие лейблы добавить или удалить (для всех получаемых метрик). Конфигурация Service Discovery Kubernetes определяется в секции kubernetes_sd_configs.

Секция remote_write определяет URL-адрес endpoint, на который отправляются метрики.

Подробная структура описана на странице конфигурации Prometheus.

Пример конфигурации Prometheus-агента (фактически в scrape_configs нужно оставить kubernetes-pods и kubernetes-pods-https и далее задать endpoint).

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: unimon-agent-config.r4
  labels:
    app: unimon-agent
data:
```

```
  prometheus.yml: |-
    global:
        scrape_interval: %GLOBAL_SCRAPE_INTERVAL%
        scrape_timeout: %GLOBAL_SCRAPE_TIMEOUT%
    scrape_configs:
        - job_name: 'kubernetes-pods'
          kubernetes_sd_configs:
            - role: endpoints
              namespaces:
                names:
                  - "ci02707148-idevgen-audit-mmv"
          relabel_configs:
            - source_labels:
[__meta_kubernetes_service_annotation_prometheus_io_scrape]
              regex: true
              action: keep
            - source_labels:
[__meta_kubernetes_service_annotation_prometheus_io_path]
              regex: (.+)
              target_label: __metrics_path__
              action: replace
            - source_labels:
[__meta_kubernetes_service_annotation_prometheus_io_scheme]
              action: drop
              regex: https
            - source_labels: [__address__,
__meta_kubernetes_service_annotation_prometheus_io_port]
```

```
    action: replace
    target_label: __address__
    regex: ([^:]+)(?::\d+)?;(\d+)
    replacement: $1:$2
  - source_labels: [__meta_kubernetes_namespace]
    separator: ;
    regex: (.*)
    target_label: namespace
    replacement: $1
    action: replace
  - source_labels: [__meta_kubernetes_pod_name]
    separator: ;
    regex: (.*)
    target_label: pod
    replacement: $1
    action: replace
  - source_labels: [__meta_kubernetes_service_name]
    separator: ;
    regex: (.*)
    target_label: service
    replacement: $1
    action: replace
  - source_labels: [__meta_kubernetes_pod_node_name]
    separator: ;
    regex: (.*)
```

```
      target_label: nodeName
      replacement: $1
      action: replace
    - action: labelmap
      regex: __meta_kubernetes_pod_label_(.+)
    - source_labels: [__address__]
      separator: ;
      regex: (.*)
      target_label: unimonVersion
      action: replace
      replacement: "D-04.000.00-2373_client"
    - source_labels: [__address__]
      separator: ;
      regex: (.*)
      target_label: tenant
      action: replace
      replacement: "undefined"
    - source_labels: [__address__]
      separator: ;
      regex: (.*)
      target_label: source
      action: replace
      replacement: OpenShift
  - job_name: 'kubernetes-pods-https'
    scheme: https
```

```
        tls_config:
          ca_file: /etc/prom-certs/root-cert.pem
          cert_file: /etc/prom-certs/cert-chain.pem
          key_file: /etc/prom-certs/key.pem
          insecure_skip_verify: true
        kubernetes_sd_configs:
        - role: endpoints
          namespaces:
            names:
              - "ci02707148-idevgen-audit-mmv"
        relabel_configs:
        - source_labels:
[__meta_kubernetes_service_annotation_prometheus_io_scrape]
          regex: true
          action: keep
        - source_labels:
[__meta_kubernetes_service_annotation_prometheus_io_path]
          regex: (.+)
          target_label: __metrics_path__
          action: replace
        - source_labels:
[__meta_kubernetes_service_annotation_prometheus_io_scheme]
          action: keep
          regex: https
        - source_labels: [__address__,
__meta_kubernetes_service_annotation_prometheus_io_port]
          action: replace
```

```
    target_label: __address__
    regex: ([^:]+)(?::\d+)?;(\d+)
    replacement: $1:$2
  - source_labels: [__meta_kubernetes_namespace]
    separator: ;
    regex: (.*)
    target_label: namespace
    replacement: $1
    action: replace
  - source_labels: [__meta_kubernetes_pod_name]
    separator: ;
    regex: (.*)
    target_label: pod
    replacement: $1
    action: replace
  - source_labels: [__meta_kubernetes_service_name]
    separator: ;
    regex: (.*)
    target_label: service
    replacement: $1
    action: replace
  - source_labels: [__meta_kubernetes_pod_node_name]
    separator: ;
    regex: (.*)
    target_label: nodeName
```

```
      replacement: $1
      action: replace
    - action: labelmap
      regex: __meta_kubernetes_pod_label_(.+)
    - source_labels: [__address__]
      separator: ;
      regex: (.*)
      target_label: unimonVersion
      action: replace
      replacement: "D-04.000.00-2373_client"
    - source_labels: [__address__]
      separator: ;
      regex: (.*)
      target_label: tenant
      action: replace
      replacement: "undefined"
    - source_labels: [__address__]
      separator: ;
      regex: (.*)
      target_label: source
      action: replace
      replacement: OpenShift
  remote_write:
    - url: "http://telemetry-collector-push-service:8083/project/Collector/pipeline/self-monitoring-pipe/prometheus-proto"
```

, где http://telemetry-collector-push-service:8083/project/Collector/pipeline/self-monitoring-pipe/prometheus-proto - это http-endpoint push-коллектора для отправки метрик.

Конфигурация pipeline для импорта:

```
{
  "quota": {
    "limitTrafficPerMin": 10000
  },
  "name": "self-monitoring-pipe",
  "input": {
    "typeName": "standard-http-api",
    "config": "{}"
  },
  "processors": [
    {
      "typeName": "prometheus-unimon-converter",
      "config": "{\"allowNan\": false, \"required\": false, \"headersToEnrich\": []}",
      "order": 1
    },
    {
      "typeName": "jackson-mapper-processor",
      "config": "{}",
      "order": 2
    },
    {
      "typeName": "pipeline-data-enrichment-processor",
```

```
      "config": "{}",

      "order": 3

    },

    {

      "typeName": "json-flattener-processor",

      "config": "{\"flattenMode\": \"KEEP_ARRAYS\",
\"standardFlatteningForFieldsWithDots\": false}",

      "order": 4

    }

  ],

  "outputs": [

    {

      "typeName": "kafka-output",

      "config": "{\"acks\": -1, \"topic\": \"METRICS_TOPIC_NAME\", \"clientId\":
\"cidddd\", \"lingerMs\": 0, \"batchSize\": 16384, \"maxBlockMs\": 60000,
\"keySerializer\": \"org.apache.kafka.common.serialization.StringSerializer\",
\"sslClientAuth\": false, \"requestTimeout\": 30000, \"deliveryTimeout\": 120000,
\"valueSerializer\": \"org.springframework.kafka.support.serializer.JsonSerializer\",
\"bootstrapServers\": \"KAFKA_ADDRESS_AND_PORT\", \"securityProtocol\":
\"SSL\", \"idempotenceEnabled\": true, \"sslKeystoreKeyPass\":
\"KAFKA_PASSWORD_ID\", \"sslEnabledProtocols\": \"TLSv1.2\",
\"sslKeystoreLocation\": \"/certificates/kafka.jks\", \"sslKeystorePassword\":
\"KAFKA_PASSWORD_ID\", \"sslTruststoreLocation\": \"/certificates/kafka.jks\",
\"sslTruststorePassword\": \"KAFKA_PASSWORD_ID\",
\"maxInFlightRequestsPerConnection\": 5, \"sslEndpointIdentificationAlgorithm\":
\"https\"}"

    }

  ]

}
```

, где METRICS_TOPIC_NAME - это имя топика Apache Kafka для записи метрик (топик создается с помощью UI Abyss),

KAFKA_PASSWORD_ID - идентификатор пароля, от сертификатов Apache Kafka. Подробная информации о создании/получении/редактировании паролей представлена в документации Indicator.

KAFKA_ADDRESS_AND_PORT - список bootstrap серверов Apache Kafka Abyss (эта информация должна быть известна Администраторам PVM).

Сбор логов: настройка FluentBit агента

Логи приложений записываются в файл в формате JSON и в лог в обычном формате.

На файлы логов подписан FluentBit Sidecar, который отправляет логи дальше по pipeline.

Конфигурация FluentBit, носит информативный характер:

```
[SERVICE]
    Flush        1
    Daemon       Off
    Parsers_File /fluent-bit/etc/parsers.conf
    HTTP_Server  On
    HTTP_Listen  0.0.0.0
    HTTP_PORT    8085
[INPUT]
    Name tail
    Tag file.tail
    Path /fluent-bit/etc/logs/*.json
    Mem_Buf_Limit 10MB
    Skip_Long_Lines On
    Refresh_Interval 2
    Rotate_Wait 1
    Read_from_Head Off
    DB /fluent-bit/etc/logs/kube.db
```

```
    Parser custom
  [OUTPUT]
    Name stdout
    Match *
  [OUTPUT]
    Name            http
    Match           file.tail
    Host            telemetry-collector-push-service
    Port            8083
    URI             /project/PROJECT_NAME/pipeline/PIPELINE_NAME
    Format          json
```

, где OUTPUT - это настройка вывода логов до Apache Kafka через единый коллектор.

Конфигурация pipeline для импорта:

1. Pipeline для обработки всех логов

```
{
  "quota": {
    "limitTrafficPerMin": 10000
  },
  "name": "PIPELINE_NAME",
  "input": {
    "typeName": "standard-http-api",
    "config": "{}"
  },
  "processors": [
```

```
    {
      "typeName": "http-jackson-converter",
      "config": "{\"required\": false, \"headersToEnrich\": []}",
      "order": 1
    },
    {
      "typeName": "pipeline-data-enrichment-processor",
      "config": "{}",
      "order": 2
    },
    {
      "typeName": "json-flattener-processor",
      "config": "{\"flattenMode\": \"KEEP_ARRAYS\",
\"standardFlatteningForFieldsWithDots\": false}",
      "order": 3
    }
  ],
  "outputs": [
    {
      "typeName": "kafka-output",
      "config": "{\"acks\": -1, \"topic\": \"LOGS_TOPIC_NAME\", \"clientId\": \"cid-3\",
\"lingerMs\": 0, \"batchSize\": 16384, \"maxBlockMs\": 60000, \"keySerializer\":
\"org.apache.kafka.common.serialization.StringSerializer\", \"sslClientAuth\": false,
\"requestTimeout\": 30000, \"deliveryTimeout\": 120000, \"valueSerializer\":
\"org.springframework.kafka.support.serializer.JsonSerializer\",
\"bootstrapServers\": \"KAFKA_ADDRESS_AND_PORT\", \"securityProtocol\":
\"SSL\", \"idempotenceEnabled\": true, \"sslKeystoreKeyPass\":
\"KAFKA_PASSWORD_ID\", \"sslEnabledProtocols\": \"TLSv1.2\",
```

```
\"sslKeystoreLocation\": \"/certificates/kafka.jks\", \"sslKeystorePassword\":
\"KAFKA_PASSWORD_ID\", \"sslTruststoreLocation\": \"/certificates/kafka.jks\",
\"sslTruststorePassword\": \"KAFKA_PASSWORD_ID\",
\"maxInFlightRequestsPerConnection\": 5, \"sslEndpointIdentificationAlgorithm\":
\"https\"}"

    }

  ]

}
```

, где LOGS_TOPIC_NAME - это имя топика Apache Kafka для записи логов (топик создается с помощью UI Abyss),

KAFKA_PASSWORD_ID - идентификатор пароля, от сертификатов Apache Kafka,

KAFKA_ADDRESS_AND_PORT - список bootstrap серверов Apache Kafka Abyss (эта информация должна быть известна Администраторам PVM).

Сбор трассировок

Конфигурирование приложения (источника телеметрии)

Сбор данных происходит с помощью Spring Zipkin. Трассировки пишутся для всех приложений сервиса: pull/push collector, management application.

Конфигурация (не требует изменения, информативный характер):

```
spring.zipkin.enabled: true

spring.zipkin.base-url: http://telemetry-collector-push-
service:8083/project/PROJECT_NAME/pipeline/PIPELINE_NAME

spring.zipkin.api-path: /

spring.zipkin.sender.type: WEB
```

Конфигурация pipeline для импорта:

```
{

  "quota": {

    "limitTrafficPerMin": 10000

  },
```

```
  "name": "PIPELINE_NAME",
  "input": {
    "typeName": "standard-http-api",
    "config": "{}"
  },
  "processors": [
    {
      "typeName": "http-jackson-converter",
      "config": "{\"required\": false, \"headersToEnrich\": []}",
      "order": 1
    },
    {
      "typeName": "pipeline-data-enrichment-processor",
      "config": "{}",
      "order": 2
    },
    {
      "typeName": "json-flattener-processor",
      "config": "{\"flattenMode\": \"KEEP_ARRAYS\",
\"standardFlatteningForFieldsWithDots\": false}",
      "order": 3
    }
  ],
  "outputs": [
    {
```

```
      "typeName": "kafka-output",

      "config": "{\"acks\": -1, \"topic\": \"TRACE_TOPIC_NAME\", \"clientId\": \"cid-
3\", \"lingerMs\": 0, \"batchSize\": 16384, \"maxBlockMs\": 60000, \"keySerializer\":
\"org.apache.kafka.common.serialization.StringSerializer\", \"sslClientAuth\": false,
\"requestTimeout\": 30000, \"deliveryTimeout\": 120000, \"valueSerializer\":
\"org.springframework.kafka.support.serializer.JsonSerializer\",
\"bootstrapServers\": \"KAFKA_ADDRESS_AND_PORT\", \"securityProtocol\":
\"SSL\", \"idempotenceEnabled\": true, \"sslKeystoreKeyPass\":
\"KAFKA_PASSWORD_ID\", \"sslEnabledProtocols\": \"TLSv1.2\",
\"sslKeystoreLocation\": \"/certificates/kafka.jks\", \"sslKeystorePassword\":
\"KAFKA_PASSWORD_ID\", \"sslTruststoreLocation\": \"/certificates/kafka.jks\",
\"sslTruststorePassword\": \"KAFKA_PASSWORD_ID\",
\"maxInFlightRequestsPerConnection\": 5, \"sslEndpointIdentificationAlgorithm\":
\"https\"}"

    }

  ]

}
```

, где TRACE_TOPIC_NAME - это имя топика Apache Kafka для записи трассировок (топик создается с помощью UI Abyss),

KAFKA_PASSWORD_ID - идентификатор пароля, от сертификатов Apache Kafka,

KAFKA_ADDRESS_AND_PORT - список bootstrap серверов Apache Kafka Abyss (эта информация должна быть известна Администраторам PVM).

Настройка FluentBit агента для сбора трассировок

Пример конфигурации FluentBit для отправки трейсов из системы распределенной трассировки Zipkin в FluentBit:

```
apiVersion: v1

kind: ConfigMap

metadata:

  name: volume-conf-fluent-bit-sidecar

data:

  fluent-bit.conf: |-
```

```
[SERVICE]
    Flush        1
    Daemon       Off
    Parsers_File /fluent-bit/etc/parsers.conf
    HTTP_Server  On
    HTTP_Listen  0.0.0.0
    HTTP_PORT    ${fluentbit_monitoring_port_number}
[INPUT]
    Name http
    Tag api_v2_spans
    Port ${spans_port_number}
    Buffer_max_size ${buffer_max_size}
    Buffer_chunk_size ${buffer_chunk_size}
[FILTER]
    Name nest
    Match *
    Operation lift
    Nested_under tags
    Add_prefix tag_
[FILTER]
    Name nest
    Match *
    Operation lift
    Nested_under localEndpoint
    Add_prefix localEndpoint_
```

```
  [FILTER]
    Name nest
    Match *
    Operation lift
    Nested_under remoteEndpoint
    Add_prefix remoteEndpoint_
  [OUTPUT]
    Name stdout
    Match *
  [OUTPUT]
    Name            http
    Match           file.tail
    Host            telemetry-collector-push-service
    Port            8083
    URI             /project/Collector/pipeline/tracing-pipe
    Format          json
parsers.conf: |-
  [PARSER]
    Name custom
    Format json
```

Здесь в секции [INPUT] для плагина HTTP, тег api_v2_spans такой же, какой динамически устанавливается путем добавления в конец URL-адреса запроса. За, Объединенный мониторинг Unimon (MONA), Журналирование (LOGA), входящих в состав продукта Platform V Monitor (OPM), которые затем могут быть визуализированы. Типы данных, которые собирает Единый коллектор:

- метрики, отражающие информацию о работе приложений;

- метрики для контроля бизнес-показателей, отражающие активность и опыт конечного пользователя или конечной точки подключения;
- трейсы и логи, формируемые в результате работы приложения.

Часто встречающиеся проблемы и пути их разрешения

Не отображаются Prometheus метрики в Grafana

1. Проверить работоспособность Grafana. Например, выбрать больший временной период отображения метрик или зайти на другие дашборды. Если на дашбордах есть сообщения об ошибках (обычно в правом верхнем углу), это говорит о проблемах с Grafana. Если метрики или графики за любой период отображаются, Grafana доступна;
2. Убедиться, что pipeline, настроенный в едином коллекторе, обеспечивает сбор метрик от приложения. Данную информацию необходимо уточнить у сотрудников, имеющих в Grafana доступ к разделу со списком задач сбора данных телеметрии, в рамках рассматриваемого проекта;
3. Проверить, что в Service вашего приложения указаны аннотации Prometheus;
4. Проверить работоспособность Prometheus-agent по логам. Необходимо убедиться в отсутствии сообщений об ошибках в Pod Prometheus-agent.

Не отображаются логи и трассировки

1. Проверить работоспособность средства отображения логов и трассировок;
2. Убедиться, что pipeline, настроенный в едином коллекторе, обеспечивает сбор логов/трассировок от приложения.

Остальные шаги выполняются при наличии доступов в Kafka и хранилище Abyss

1. Проверить метрики нужного приложения в топике;
2. Проверить метрики нужного приложения в хранилище Abyss (работоспособность перекладчика Kafka → хранилище Abyss).

Подробная информация представлена в документе «Руководство оператора» Abyss.