Руководство по эксплуатации

Компонента Сервис трассировки (TRAS)

Продукта Platform V Synapse Service Mesh (SSM)

ОГЛАВЛЕНИЕ

Руководство по эксплуатации

Руководство по системному администрированию

Термины и определения

Термин/Аббревиатура	Определение
Платформа	Платформа оркестрации приложений с средствами автоматизации и управления на основе политик, например Kubernetes
URL	Система унифицированных адресов электронных ресурсов
Трейс	Цепочка единичных операций (спанов), представляющих часть потока выполнения запроса. Трейс позволяет визуализировать активность запроса при его перемещении по системе и является самостоятельной единицей трассировки
Нода Платформы	Единица Kubernetes, кластера инфраструктуры Платформы, предоставляющая среду для работы контейнеров
Zipkin хранилище	Внешняя система хранения и отображения данных трассировки совместимая с форматом Zipkin
Platform V Synapse Service Mesh	Программный продукт на базе Istio SE, обеспечивающий возможность создания сервисной сети поверх Платформенной в Kubernetes
Контрольная панель	Проект, где запущены управляющие приложения Synapse Service

Термин/Аббревиатура	Определение
	Mesh (компонент POLM)
Istio SE	Настраиваемая сервисная сетка с открытым исходным кодом, служащая для взаимодействия, мониторинга и обеспечения безопасности контейнеров в кластере Kubernetes
Управление политиками / POLM	Компонент Управление политиками из состава продукта Platform V Synapse Service Mesh
Сервис трассировки / TRAS	Компонент Сервис трассировки продукта Platform V Synapse Service Mesh

Сервис трассировки не имеет пользовательского интерфейса для администрирования. Для управления используйте интерфейсы, предоставляемые Платформой.

Ниже приведены возможные сценарии администрирования с использованием Web-интерфейса Платформы и консоли клиента Платформы. Название Деплоймента или Пода — synapse-tracer.

Сценарии администрирования

Сценарий 1. Запуск

Для запуска через веб-интерфейс Платформы:

Шаг	Действие
Авторизуйтесь в веб-консоли Платформы	Перейдите по ссылке URL веб-консоли нужного кластера Платформы. В окне ввода учетных данных введите логин и пароль.
Перейдите в нужный проект	Выполните следующие действия: 1. В пункте меню **Projects** выберите из списка нужный проект.
Запустите Под	2. В меню выберите пункт **Workload/Deployments**. 3. На странице найдите нужный **Deployment**. При необходимости

Шаг	Действие
	воспользуйтесь поиском по имени. 4. Пройдите по ссылке в наименовании на вкладку **Detail**. Стрелкой вверх (**Increase the pod count**) установите нужное количество Подов.
Выйдите из веб-консоли Платформы	5. Раскройте меню нажатием на имя пользователя и выберите пункт **Log out**. 6. Закройте окно браузера.

Для запуска через консоль Платформы:

Шаг	Действие

Войдите в консольного клиента платформы	В окне командной строки в приглашении введите команды:
	`kubectl config set-credentials <user>/<host-alias-без-точек>:<port> \--token=<token>`
	`kubectl config set-cluster <host-alias-без-точек>:<port> \--insecure-skip-tls-verify=true --server=https://<host>:<port>`
	`kubectl config set-context <namespace>/<host-alias-без-точек>:<port>/<user> \--user=<user>/<host-alias-без-точек>:<port> \--namespace=maximov-test --cluster=<host-alias-без-точек>:<port>`
	`kubectl config use-context <namespace>/<host-alias-без-точек>:<port>/<user>`

Запустите компонент	В консоли выполните команду: `kubectl scale --replicas=<N> deployment/<имя Деплоймента>`, где: `N`>0 — требуемое количество запущенных Подов.

Сценарий 2. Остановка

Для остановки через веб-интерфейс Платформы:

Шаг	Действие
Авторизуйтесь в веб-консоли Платформы	Перейдите по ссылке URL веб-консоли нужного кластера Платформы. В окне ввода учетных данных введите логин и пароль.

Шаг	Действие
Перейдите в нужный проект	Выполните следующие действия: 1. В пункте меню **Projects** выберите из списка нужный проект.
Остановите компоненты	2. В меню выберите пункт **Workload/Deployments**. 3. На странице найдите нужный **Deployment**. При необходимости воспользуйтесь поиском по имени. 4. Пройдите по ссылке в наименовании на вкладку **Detail** и стрелкой вниз (**Decrease the pod count**) установите значение **0**.
Выйдите из веб-консоли Платформы	5. Раскройте меню нажатием на имя пользователя и выберите пункт **Log out**. 6. Закройте окно браузера.

Для остановки через консоль Платформы:

Шаг	Действие
Войдите в консольного клиента платформы	В окне командной строки в приглашении введите команды: kubectl config set-credentials /<host-alias-без-точек>: --token= kubectl config set-cluster <host-alias-без-точек>: --insecure-skip-tls-verify=true --server=https://: kubectl config set-context /<host-alias-без-точек>:/ --user=/<host-alias-без-точек>: --namespace=maximov-test --cluster=<host-alias-без-точек>: kubectl config use-context /<host-alias-без-точек>:/
Остановите компоненты	В консоли выполните команду: `kubectl scale --replicas=0 deployment/<имя Деплоймента>`
Завершите сеанс работы	В консоли выполните команду: `kubectl logout`

Сценарий 3. Настройка выделения ресурсов

Для настройки через веб-интерфейс Платформы:

Шаг	Действие
Авторизуйтесь в веб-консоли Платформы	Перейдите по ссылке URL веб-консоли нужного кластера Платформы. В окне ввода учетных данных введите логин и пароль.
Перейдите в проект	В пункте меню **Projects** выберите из списка нужный проект.
Откройте **Deployment**	Выполните следующие действия: 1. В меню выберите пункт **Workload/Deployments** 2. На странице найдите нужный **Deployment**. При необходимости воспользуйтесь поиском по имени. 3. Пройдите по ссылке в наименовании на вкладку **Detail** → **YAML**.

Скорректируйте параметры	В окне редактирования найдите параметры:
	`spec.template.spec.containers[0].resources.limits.cpu`
	`spec.template.spec.containers[0].resources.limits.memory`
	`spec.template.spec.containers[0].resources.requests.cpu`
	`spec.template.spec.containers[0].resources.requests.memory`
	Установите нужные значения.

Сохраните изменения	Нажмите **Save**.
Проверьте конфигурацию	Нажмите **Reload**. Проверьте сохранение изменений.
Выйдите из веб-консоли Платформы	Раскройте меню нажатием на имя пользователя и выберите пункт **Log out**. 8. Закройте окно браузера.

События системного журнала

Для просмотра системного журнала через веб-интерфейс Платформы:

Шаг	Действие
Авторизуйтесь в веб-	Перейдите по ссылке URL веб-консоли нужного кластера Платформы. В окне ввода учетных данных введите логин и

Шаг	Действие
консоли Платформы	пароль.
Перейдите в проект	В пункте меню **Projects** выберите из списка нужный проект.
Перейдите в **Pods**	Выполнить следующие действия: 1. В меню выберите пункт **Workload/Pods**; 2. На странице найдите нужный Под . При необходимости воспользуйтесь поиском по имени. 3. Пройдите по ссылке в наименовании на вкладку **Logs**.
Посмотрите события	4. В терминале проверьте системные журналы Пода.
Выйдите из веб-консоли Платформы	5. Раскройте меню нажатием на имя пользователя и выберите пункт **Log out**. 6. Закройте окно браузера.

Стандартные события системного журнала

Предусмотрено два уровня логирования событий системного журнала: info и debug.

В режиме "info" (режим по умолчанию), в событиях системного журнала периодически (в соответсвии с заданным в конфигурации приложения TRAS периодом) выводятся сообщения о статистике отправляемых трейсов:

- bandle sends (all time) - количество отправленных блоков(объединений) спанов в Zipkin хранилище, счетчик сквозной, показывает общее количество за время существования пода
- spanConverts (in second for logTime period) - количество обработанных спанов в секунду за период логирования
- error send to collector (all time) - количество ошибок при попытке отправить бандл, показывает общее количество за время существования пода
- maximal time of request per logTime period - максимальное время отклика Zipkin хранилища при отправке в него бандла, за период логирования

а также выводятся сообщения об ошибках в процессе работы приложения:

- non 200 response from collector: - ошибка отправки бандла в Zipkin хранилище. После двоеточия выводится текст ошибки.
- non 200 bundle: - ошибка отправки в Zipkin хранилище сопровождается выводом неотправленного бандла в консоль.

Режим debug используется для отладки приложения и выводит полную информацию о пересылаемых трейсах. Режим debug включается администратором проекта из консоли приложения TRAS, командой curl localhost:9192/loglevel -d "debug".

События мониторинга

Компонент TRAS не интегрирован с компонентом "Объединенный мониторинг Unimon" Plaform V Monitor (MONA). Для наблюдения за состоянием подов используйте существующие средства мониторинга Платформы, описанные в документации на конкретную платформу.

Вы также можете увидеть полный трейс в веб-интерфейсе Zipkin хранилища

Часто встречающиеся проблемы и пути их устранения

Проблема	Решение
В логах подов встречаются ошибки типа:<br 2022-05-20T10:03:54.718Z warn non 200 response from collector: no healthy upstream	Убедитесь, что эндпойнт Zipkin хранилища, настроен и готов к работе, убедитесь в наличии физического доступа до эндпойнта из кластера Платформы
Не стартуют поды	Проверьте события во вкладке Event и в разделах Status/Yaml на страницах Deployment ReplicaSet и Pod. Устраните ошибки согласно описанию найденых ошибок
Отсутствует информация о трейсах во вкладке Log в режиме debug	Убедитесь, что в проектах подключенных к контрольной панели Synapse Service Mesh (POLM) есть трафик Убедитесь, что функционал трейсинга включен на уровне контрольной панели панели Synapse Service Mesh (POLM)

Руководство прикладного разработчика

Термины и определения

Термин/Аббревиатура	Определение
Платформа	Платформа оркестрации приложений с средствами автоматизации и управления на основе политик, например Kubernetes
Трейс	Цепочка единичных операций (спанов), представляющих часть потока выполнения запроса. Трейс позволяет визуализировать активность запроса при его перемещении по системе и является самостоятельной единицей трассировки
Сэмплирование	Процесс выборки определённых запросов в соответствии с установленными пользователем правилами, позволяющий производить отслеживание и контроль трассировки
Platform V Synapse Service Mesh	Программный продукт на базе Istio SE, обеспечивающий возможность создания сервисной сети поверх Платформенной в Kubernetes
Контрольная панель	Проект, где запущены управляющие приложения Synapse Service Mesh
Zipkin хранилище	Внешняя система хранения и отображения данных трассировки совместимая с форматом Zipkin
Управление политиками / POLM	Компонент Управление политиками из состава продукта Platform V Synapse Service Mesh

Термин/Аббревиатура	Определение
Граничный прокси / IGEG	Компонент Граничный прокси продукта Platform V Synapse Service Mesh
Сервисный прокси / SVPX	Компонент Сервисный прокси продукта Platform V Synapse Service Mesh
Сервис трассировки / TRAS	Компонент Сервис трассировки продукта Platform V Synapse Service Mesh
Span	Логическая единица работы в трейсинге, имеющая название, время начало операции и её продолжительность

Системные требования

- Развернутый и настроенный кластер Платформы (Kubernetes 1.19 и выше)
- В кластере создан проект для развертывания контрольной панели Synapse Service Mesh
- В проекте создана учетная запись с правами на загрузку артефактов (администратор проекта).
- Свободные в проекте ресурсы, по лимитам и реквестам по размеру равные или превышающие зарезервированные в конфигурационных артефактах продукта Platform V Synapse Service Mesh.
- Добавленный в проект секрет для загрузки docker-образов из целевого docker-репозитория.
- Установленный на рабочем месте консольный клиент Платформы для доступа к Платформе через консоль.

Подключение и конфигурирование

Подключение осуществляется администраторами контрольной панели.

Для конфигурации компонента воспользуйтесь различными артефактами:

Артефакт	Содержание	Описание
Deployment	Параметры запуска контейнера приложения	Артефакт устанавливает наименование экземпляра приложения, ссылку на образ контейнера

Артефакт	Содержание	Описание
	в Платформе	приложения, запрашиваемые ресурсы, публикуемые порты, параметры liveness- и readiness-проб, параметры подключения sidecar-контейнеров и необходимость их использования, точки монтирования конфигурационных артефактов в файловую систему контейнера
Config Map	tracerconfig.json tracerconfigfilter.json	Файл, содержащий параметры конфигурации приложения
Secret	Ключи и сертификаты	Файлы, содержащие ключи и сертификаты. Содержат конфиденциальные данные, поэтому загружается в виде секрета
Service	Артефакт для регистрации приложения в service discovery Платформы	Селекторы и порты для подключения приложения к механизмам распределения трафика Платформы

Миграция на текущую версию

Этот раздел не применим к данному компоненту.

Разработка первого приложения с использованием программного продукта

Этот раздел не применим к данному компоненту.

Использование программного продукта

Компонент нужен для получения трейсов от Граничного и Сервисного прокси и передачи их в Zipkin хранилище.

На каждом этапе цепочки вызовов микросервисов необходима передача следующих HTTP-заголовков с момента их первой генерации:

Заголовок	Описание
`x-request-id`	Уникальный идентификатор сообщения. Формат: GUID. Генерируется с помощью Envoy, если запрос еще не имеет такого заголовка
`x-b3-traceid`	Уникальный идентификатор Trace. Формат: строка с hex-представлением 64- или 128-битного целого числа. Генерируется с помощью Envoy, если запрос еще не имеет такого заголовка
`x-b3-spanid`	Уникальный идентификатор Span. Формат: строка с hex-представлением 64-битного целого числа
`x-b3-parentspanid`	Уникальный идентификатор родительского Span. Формат: строка с hex-представлением 64-битного целого числа
`x-b3-sampled`	Признак сэмплирования запроса. Если значение равно `1`, то трассировка этого запроса будет отправлена в систему сбора информации
`x-b3-flags`	Дополнительные флаги: отладка и другие

Сервисный прокси генерирует данные заголовки автоматически, если они отсутствуют при вызове компонента.

Стандартные заголовки

Каждый сервис, который вызывается в единой цепочке вызовов, должен передавать полученные при его вызове заголовки.

- Для HTTP-вызова:

Пример передачи заголовков для HTTP:

```
    @RequestMapping(method = RequestMethod.POST, path = "/")
public ResponseEntity postInfo(@RequestHeader HttpHeaders inHeaders) {
Map<String, String> outHeaders = new HashMap<>();
outHeaders.put("x-request-id", inHeaders.getFirst("x-request-id"));
outHeaders.put("x-b3-traceid", inHeaders.getFirst("x-b3-traceid"));
outHeaders.put("x-b3-spanid", inHeaders.getFirst("x-b3-spanid"));
outHeaders.put("x-b3-parentspanid", inHeaders.getFirst("x-b3-parentspanid"));
outHeaders.put("x-b3-sampled", inHeaders.getFirst("x-b3-sampled"));
outHeaders.put("x-b3-flags", inHeaders.getFirst("x-b3-flags"));
outHeaders.put("x-ot-span-context", inHeaders.getFirst("x-ot-span-context"));
HttpEntity<Object> requestHttpEntity = new HttpEntity<>(requestBody, outHeaders);
ResponseEntity<Object> responseHttpEntity = restTemplate.postForEntity(url,
requestHttpEntity, Object.class);
}
```

- Для gRPC-вызова:

Пример передачи заголовков gRPC:

```
    @Grpc(serviceName = "com.sbt.synapse.gateway.MessageAsyncChannel/processMessage")
public class GrpcService implements BaseSynapseService<Message.ProtoMessage,
HeartbeatOuterClass.Heartbeat> {
   @Grpc(serviceName = "com.sbt.synapse.gateway.MessageAsyncChannel/processMessage")
   private GrpcService<Message.ProtoMessage, HeartbeatOuterClass.Heartbeat> grpcService;

   ...
   @Override
   public HeartbeatOuterClass.Heartbeat processSync(SynapseMessage<Message.ProtoMessage>
message) {
     MessageHeaders outHeaders = message.getHeaders();
             grpcService.invoke(protoMessage)
                 .authority(authority)
                 .headers(outHeaders);
     ...
       }
 ...
}
```

Для передачи gRPC-заголовков используется библиотека `com.sbt.synapse:synapse-grpc-core:2.0.12`.

Пользовательские заголовки (необязательные)

На данный момент в состав Span включаются следующие пользовательские теги:

Имя тега	Значение
`synapse.corellationid`	`request.headers["x-synapse-corellationid"]`
`synapse.messageid`	`request.headers["x-synapse-messageid"]`
`synapse.rquid`	`request.headers["x-synapse-rquid"]`
`synapse.rqtm`	`request.headers["x-synapse-rqtm"]`
`synapse.scname`	`request.headers["x-synapse-scname"]`
`synapse.servicename`	`request.headers["x-synapse-servicename"]`
`synapse.serviceversion`	`request.headers["x-synapse-serviceversion"]`
`synapse.spname`	`request.headers["x-synapse-spname"]`
`synapse.status_code`	`response.headers["x-synapse-status-code"]` или `request.headers["x-synapse-status-code"]`
`synapse.status_desc_bin`	`response.headers["x-synapse-status-desc-bin"]` или

Имя тега	Значение
	`request.headers["x-synapse-status-desc-bin"]`