GS LABS
GS GROUP

Публичный - L0

DRE Event Server

Руководство администратора

Индекс	2060-DREES-AG
Секретность	Публичный - L0
Ревизия	1.1
Статус	Согласован
Подразделение	Департамент по разработке сервисов
Компания	GS Labs

Содержание

1. Аннотация

Документ предназначен для технических специалистов, занимающихся администрированием системы мониторинга DRE Event Server (далее - DREES) и обладающих навыками работы с компьютером на профессиональном уровне.

2. Общее описание

DREES предназначена для эффективной передачи событий и метрик с клиентских устройств и обеспечивает сбор данных с клиентских устройств, единое хранилище данных, предоставление собранных данных внешним системам. Клиентские настройки для Consumer Client устанавливаются с помощью переменных окружения и параметров консольного исполняемого файла (раздел 5).

Настройка и управление серверной стороны (Server Side) осуществляется посредством внесения изменений в настройки платформы кластеров Kubernetes при помощи изменения конфигурационного файла helm системным администратором (раздел 3).

После установки продукта, имеется возможность протестировать его работоспособность через специальные UI, подробная информация приведена в разделе 4.

Данные о работе сервисов отображаются в лог-файлах (см. раздел 6).

3. Администрирование Server Side

3.1. Общие сведения

Установка и обновление сервера производится посредством Helm - менеджера пакетов (чартов) Kubernetes. При установке на основе чарта на кластере создаются его экземпляры - релизы (releases).

Список запущенных на кластере релизов, их состояние и версии соответствующих чартов можно получить, выполнив на мастер-ноде команду:

```
helm ls
```

Удаление компонентов сервера Event Server (компонента DREES) производится путем удаления соответствующих релизов:

```
helm delete RELEASENAME --purge
```

Администрирование установленного сервера Event Server (компонента DREES) производится через командный интерфейс Kubectl на любой ноде кластера. Подробную информацию по работе с Kubectl можно найти в документации продукта Kubernetes.

3.2. Конфигурация сервисов

Для конфигурации сервисов необходимо внести изменения в helmfile и перезапустить деплой.

Перечень доступных параметров переменных окружения сервисов DREES и их начальных значений предоставляется заказчику по требованию.

3.3. Конфигурация Nats Streaming Server

Данный сервер выполняет роль посредника (шина данных), в который отправляются сообщения и передаются клиентам, по шаблону Publish-Subscribe.

При первоначальной настройке сервера стоит обратить внимание на следующие опции, которые могу влиять на производительность сервера и гарантии доставки сообщений потребителям данных.

Особое внимание стоит уделить конфигурации Store Limits, поскольку устаревшие и не доставленные сообщения будут вытесняться при достижении лимитов хранилища.

Для оценки объема хранилища рекомендуется проанализировать статистику базы данных за требуемый для хранения интервал времени.

Опции "File Options Configuration" требуются в режиме работы кластера с типом хранилища "file".

Перечень переменных конфигурационных файлов Nats Streaming Server и их начальных значений предоставляется заказчику по требованию.

3.4. Конфигурация Premier adapter

3.4.1. Информация для развертывания

Сервис для отправки start watch и heartbeat событий на Premier платформу.

Сервис не масштабируется горизонтально. Т.к. имеет следующие механизмы которые будут не согласованы при разделении на поды:

- кэширование данных в redis
- вычитывание последовательных данных из grpc stream Premier-adapter справляется с актуальной нагрузкой без масштабирования.

Сервис взаимодействует со следующими узлами:

- **MDS**. Endpoint /api/premier/uma-content-ids для получения списка пар umaID + contentID
- **Redis**. Хранилище куда складываются вышеперечисленные данные дабы исключить нагрузку на MDS сервис
- **Consumer** server, развернутый для внутренних нужд DREES. Из него premier-adapter получает данные о heartbeat и start watch событиях в формате gRPC stream
- **Premier** платформа, куда отправляются события в случае если они относятся к данной платформе (есть связка umaID + contentID).

Перечень доступных параметров переменных окружения Premier adapter и их начальных значений предоставляется заказчику по требованию.

3.5. Конфигурация Mediascope adapter

3.5.1. Переменные окружения

Перечень доступных параметров переменных окружения Premier adapter и их начальных значений предоставляется заказчику по требованию.

3.5.2. Тестовые данные для интеграционного стенда

Значение параметров user, pass, endpoint, identifier необходимо запросить в Mediascope.

3.5.3. Информация для развертывания

Сервис не масштабируется горизонтально. Т.к. получает события из одно стрима consumer-server и не имеет механизма распределения этих событий между подами.

3.6. Конфигурация Appsflyer adapter

Данный адаптер подключается как consumer-клиент в режиме stream к consumer-серверу. Клиент отправляет запрос на получение новых событий от 2-х типов устройств `DT_IOS` и `DT_ANDROID`. Поступившие данные передаются в фоновые процессы для последующей фильтрации и подготовки к отправке во внешнюю платформу Appsflyer API.

3.6.1. Информация для развертывания

Сервис не предусмотрен для горизонтального масштабирования за счет увеличения реплик, поскольку имеет ограничения параллельного чтения из канала брокера для одного клиента. Для ускорения обработки большего количества событий рекомендуется увеличить численное значение переменной окружения `POOL_WORKERS`. За счет увеличения количества параллельных "рабочих", увеличится и скорость потребления данных из общей выгружаемой очереди сообщений. Так же следует скорректировать лимиты на ресурсы для приложения в helm чартах.

Сервис взаимодействует со следующими узлами:

- **Consumer** server, компонент экосистемы DREES.
- **Appsflyer** внешняя платформа, где регистрируются события для последующей аналитической работы с данными.

3.6.2. Переменные окружения

Перечень доступных параметров переменных окружения Premier adapter и их начальных значений предоставляется заказчику по требованию.

3.7. Базы данных

База данных DREES представляет собой две таблицы:

- Таблица events - гипертаблица, содержащая все события, генерируемые клиентскими устройствами.
- Таблица clients - содержит информацию о клиентских устройствах.

3.8. API сервиса

DREES имеет 3 протокола API:

1. gRPC - высокоэффективный бинарный протокол передачи данных. Используется клиентами, способными поддержать gRPC
2. gRPC Web - gRPC для браузерных JavaScript приложений (SmartTV, Web)
3. JSON API - отдельное API для клиентов, неспособных поддержать gRPC нативно (Stingray)

4. API тестирование

После установки продукта, имеется возможность протестировать его работоспособность через специальный UI, который активируется helm параметром swagger.enabled=true. В продуктовой эксплуатации данная опция должна быть выставлена в значение false.

4.1.1. Swagger UI

Swagger позволяет протестировать работу REST API. Ссылку на веб-интерфейс Swagger необходимо запросить у системного администратора.

Проверить работоспособность REST API можно следующим образом:

1. Перейти на Swagger UI
2. Вверху справа по очереди выбрать v1 и v2
3. Выбрать в выпадающем списке Schemes протокол HTTPS
4. Выбрать метод POST /v1/event или POST /v2/event
5. Нажать Try It Out
6. Вставить следующий код:

```
{
  "client": {
    "id": "test",
    "type": "DT_IOS"
  },
  "event": {
    "name": "test",
    "group": "test"
  }
}
```

В ответе должно быть:

5. Администрирование Consumer Client (выгрузки)

Для получения выгрузок с DREES, необходимо иметь утилиту consumer, которая представляет из себя исполняемый файл для Windows, Linux или MacOS систем с архитектурой amd64/386.

5.1. Отличия Windows версии

При использовании утилиты на Windows, имя исполняемого файла оканчивается на ".exe" (в приведенных примерах вместо consumer в начале строки будет consumer.exe, где под consumer понимается условное название исполняемого файла). Запуск утилиты должен производиться с правами администратора. Во всех примерах далее будет использоваться Linux версия исполняемого файла (в тексте примеров обозначается consumer в начале строки, где под consumer понимается условное название исполняемого файла).

5.2. Доступ к API

Доступ к API предоставляется при наличии ключа авторизации в переменной окружения CONSUMER_SERVER_KEY. Так же с помощью параметров CONSUMER_SERVER_ADDRESS и CONSUMER_SERVER_TLS можно направить Consumer на версию DREES, отличную от Production. По умолчанию утилита будет обращаться в Production окружение по адресу [domain.name]:*443* в режиме TLS соединения.

5.3. Описание возможностей утилиты

Чтобы получить описание возможностей, можно выполнить команду:

consumer help

```
consumer help
```

Данная команда выведет помощь по использованию на английском языке.

5.4. Общие параметры

5.4.1. Утилиты

Утилита поддерживает и требует на вход некоторые параметры:

1. Имя шаблона выгрузки (template) - это первый параметр любой конечной команды. Так же может являться путём до yaml файла, содержащего описание шаблона выгрузки.
    a. Пример: *consumer dump csv* ***vod.ContentWatched****2019-01 --group vod*
    b. Список доступных шаблонов можно получить выполнив команду *consumer tpl list*
2. Флаг *--group (-g)* - имя группы событий, из которой производится выгрузка данных
    a. Пример: *--group=vod*
3. Флаг *--name (-n)* - имя события. Может принимать несколько значений через запятую
    a. Пример: *--name=ContentWatchStarte*d
    b. Пример: *--name=ContentWatchStarted,ContentWatched*
4. Флаг *--filter (-f)* - позволяет фильтровать события по его свойствам. Поддерживаемые поля: clients.id, clients.dre_id, clients.hw_id. Разрешается использовать только одно поле
    a. Пример: *--filter=clients.dre_id=12345*
5. Флаг *--help* - позволяет вывести справку о любой команде, включая специальные параметры
    a. Пример: *consumer stream logstash --help*
6. Флаг *--debug* - позволяет выводить отладочную информацию во время работы утилиты
7. Флаг *--net-debug* - позволяет выводить отладочную информацию сети во время работы утилиты

5.5. Режимы работы

Утилита предоставляет 3 основных способа работы: дамп (dump), поллинг (polling) и стриминг (stream).

5.5.1. Dump

Данный режим работы позволяет скачать данные за определённый период времени. Для него поддерживаются следующие способы вывода:

- stdout - вывод в стандартный вывод терминала
- csv - вывод в партиционированные CSV файлы
- single-csv - вывод в единый CSV файл
- logstash - вывод в TCP сервер Logstash
- noop - поглощение вывода, никуда не выводит
- mssql - вставка данных в MSSQL

> **Примечание**
>
> Продукт DREES может выгружать данные, пригодные для вставки в стороннюю БД MSSQL, не являющуюся частью продукта.

Примеры вызова команды:

consumer dump stdout vod.ContentWatched 2020-01 #Выгрузка за период с [2020-01-01 00:00:00 по 2020-02-01 00:00:00) с партиционированием по суткам (d)

consumer dump csv vod.ContentWatched 2020-01-02 --partition-size=h #Выгрузка в CSV за период с [2020-01-02 00:00:00 по 2020-01-03 00:00:00) с партиционированием по часам (h)

consumer dump csv vod.ContentWatched 2020-01-02 --headers --partition-size=m #Выгрузка за период с [2020-01-02 00:00:00 по 2020-01-03 00:00:00) с партиционированием по минутам (m) и с добавлением называний столбцов в CSV файлах

5.5.1.1. Партиционирование данных

При дампе данных в CSV имеется специальная возможность партиционирования файлов по периоду времени - день (d), час (h) или минута (m). По умолчанию используется партиционирование по дням. Для избежания партиционирования используйте команду *consumer dump csv-single*.

5.5.1.2. Фильтрация данных

В утилите имеются возможности для фильтрации данных по параметрам HWID и DREID. Для этого при использовании команды Dump нужно использовать флаги *--filter=clients.dre_id=<dre_id>* или *--filter=clients.hw_id=<hw_id>*.

5.5.2. Polling

Данный режим работы похож на dump, но отличается механизмом выгрузки - периодический опрос сервера на поступление новых данных, без постоянного соединения.

Особенности работы режима Polling в следующем:

1. Если указать дату в прошлом, без указания даты завершения, то утилита будет ожидать новых данных и добавлять их к выгрузке.
2. Если даты не указывать совсем, то выгрузка начнется с текущего времени и будет периодически делать запросы на выгрузку новых данных.

Для него поддерживаются следующие способы вывода:

- stdout - вывод в стандартный вывод терминала
- logstash - вывод в TCP сервер Logstash
- noop - поглощение вывода, никуда не выводит
- mssql - вставка данных в MSSQL

> **Примечание**
>
> Продукт DREES может выгружать данные, пригодные для вставки в стороннюю БД MSSQL, не являющуюся частью продукта.

Примеры вызова команды:

consumer polling stdout vod.ContentWatched 2020-12-22T00:00:00 -g appregistration #Выгрузка за период с 2020-12-22 00:00:00 по текущее время и продолжение выгрузки до бесконечности

consumer polling logstash simple -g appregistration -n AuthorizationCompleted,RegistrationCompleted --address=<logstash-host:port> #команда начнёт выгрузку в logstash по указанному адресу начиная с текущего времени и до бесконечности

5.5.2.1. Фильтрация данных

Режим работы поддерживает одинаковые характеристики фильтрации с режимом Dump (п. 5.5.1.2)

5.5.2.2. Сохранение состояния выгрузки

Режим предусматривает сохранение состояния выгрузки на стороне сервера с возможностью последующей догрузки данных. При выполнении запроса с одинаковыми критериями выгрузки состояние будет сохраняться. Для принудительного сброса состояния необходимо добавить:

consumer polling stdout vod.ContentWatched 2020-01 -g appregistration --restart # Сброс состояния выгрузки на сервере, если такой запрос уже исполнялся ранее.

5.5.3. Stream

Данный режим работы позволяет передавать данные в около-реальном времени начиная с определённой даты. Для него поддерживаются следующие способы вывода:

- stdout - вывод в стандартный вывод терминала
- logstash - вывод в TCP сервер Logstash
- mssql - вставка данных в MSSQL

> **Примечание**
>
> Продукт DREES может выгружать данные, пригодные для вставки в стороннюю БД MSSQL, не являющуюся частью продукта.

- noop - поглощение вывода, никуда не выводит

Параметр "-g" и "-n" можно заменить на "–pattern", который имеет следующие опции:

- *all* - все доступные событий, пример *--pattern="all"*
- *group* - фильтрация событий по имени группы, пример *--pattern="group=vod"*
- *name* - фильтрация событий по имени события и имени группы, пример *--pattern="name=operationState.OTTMonitoring"*
- *client_id* - фильтрация по идентификатору клиента, пример *--pattern="client_id=my_client_id"*
- *hw_id* - фильтрация по атрибуту клиента hw_id, пример *--pattern="hw_id=my_hw_id"*
- *dre_id* - фильтрация по атрибуту клиента dre_id, пример *--pattern="dre_id=my_dre_id"*
- *device_type* - фильтрация по типу устройства device_type, пример *--pattern="device_type=DT_IOS"*.
  Примеры вызова команды:

consumer stream stdout simple --pattern="group=vod" #команда начнёт выводить в stdout терминала все поступающие события для группы событий vod

consumer stream logstash operationState.OTTMonitoring --pattern="group=operationState" --address=<logstash-host:port> # команда начнёт стриминг в logstash по указанному адресу

5.5.3.1. Сохранение состояния выгрузки

Режим поддерживает сохранение и сброс состояния выгрузки идентичный с polling (п. 5.5.2)

Для сброса состояния выгрузки и возобновление выгрузки с момента времени запуска утилиты используется флаг "*–restart*"

5.6. YAML Шаблоны

5.6.1. Общие сведения

С помощью шаблонов можно задавать какие именно данные необходимо выгружать и какие трансформации значений производить. В некоторых плагинах, таких как MS SQL, необходимо так же задавать в каком типе данных сохранять полученные значения для каждой из колонок (параметр *targetType*).

Для указания шаблона при выгрузке необходимо передать id, название или путь к YAML шаблону при запуске consumer после указания плагина:

consumer [режим работы] [название плагина] [шаблон] [ключи запуска]

5.6.2. Структура шаблонов

Шаблон состоит из нескольких разделов, описывающих необходимые данные:

- columns - список необходимых полей. Для каждого поля необходимо описывать:
  - dbName - название исходного поля
  - name - название поля в итоговых данных.
  - targetType - тип данных (string, int, float, bool, timestamp, json). Применимо при выгрузке данных для сторонней БД MSSQL.
- description - текстовое описание шаблона.
- name - название шаблона.
- encoding - кодировка при выводе в текстовый файл. Применимо для CSV
- pipeline - описание механизмов преобразования исходных данных (к пр. конвертация числовых типов устройств в строковые значения). Допустимые типы данных и их параметры:

1. intEnum - маппинг числовых значений к строковым.
   a. column - название поля.
   b. map - описание маппинга исходного цифрового значения к строковому.
   c. unknownValue - значение по-умолчанию для не описанных значений.
2. concat - объединение значений нескольких полей в одну.
   a. columns - список исходных полей.
   b. separator - разделитель.
   c. columnName - название итогового поля.
3. date - конвертация формата даты\времени.
   a. column - название поля.
   b. originalFormat - исходный формат даты\времени.
   c. format - итоговый формат даты\времени.
4. rawJson - конвертация исходных данных к JSON формату.
   a. column - название поля.
5. string - конвертация исходных данных к строковому типу.
   a. column - название поля.
6. coalesce - выбор первого встретившегося значения из нескольких полей.
   a. columns - поля в заданном порядке
   b. toColumn - имя поля, в которое поместить результат

5.6.2.1. Скрытые поля

В некоторых ситуациях может возникнуть необходимость обработать результат нескольких выбираемых полей и сложить его в результирующее. Для того чтобы избежать добавление исходных полей в результат выгрузки, можно скрывать такие поля двумя способами:

1. Если указать пустой 'dbName', то в результат добавится новое пустое поле
2. Если указать имя поля 'name', начинающееся с точки (например, '.dreId'), то оно не будет добавлено в итоговую выгрузку

Это особенно полезно при работе с преобразованиями данных "coalesce" и "concat".

coalesce example:

```
name: coalesce-example
description: Шаблон для демонстрации coalesce
columns:
# ...
  - dbName: clients.dre_id   # clients.dre_id - это последнее доступное значение dre_id
  name: .clientsDreId  # поле будет скрыто из выгрузки, т.к. имя начинается на "."
  targetType: string
  - dbName: events.dre_id  # events.dre_id - это dre_id, который был прислан в момент возникновения события
  name: .eventsDreId    # поле будет скрыто из выгрузки, т.к. имя начинается на "."
  targetType: string
  - name: dreId   # dbName не указан - пустое поле
  targetType: string
# ...

pipeline:
- coalesce:
columns:
- .eventsDreId
- .clientsDreId
toColumn: dreId  # заполняем пустое поле результатом
```

5.6.2.2. Фильтр по умолчанию

Так как зачастую шаблоны предназначены для определённой группы событий или события, можно задавать фильтры для них по умолчанию. Таким образом, при использовании consumer, не потребуется дополнительно указывать группу событий и/или его имя:

```
filter:
   group: vod    # соответствует флагу --group=vod
   name: ContentWatched  # соответствует флагу --name=ContentWatched
```

В случае если в шаблоне указан фильтр, и в consumer указан флаг, то приоритет будет отдан флагу.

В случае если в шаблоне задан фильтр, но при вызове consumer его нужно отключить, то можно использовать символ "-":

consumer dump stdout vod.ContentWatched 2020-06-25 --name=- # отключит фильтр по имени события

5.6.2.3. Форматы даты\времени (date)

Название	Пример
DF_UNIX_SECONDS	1591881177
DF_UNIX_MILLI	1591881177261
DF_UNIX_NANO	1591881177261967893
DF_RFC3339	2020-06-11T16:14:54+03:00
DF_RFC3339Nano	2020-06-11T16:14:54.808525904+03:00

5.6.3. Доступные исходные поля

Перечень доступных полей, использующихся для формирования шаблонов выгрузки, предоставляется заказчику по требованию.

5.6.4. Пример шаблона

```
columns:
- dbName: events.timestamp
  name: timestamp
  targetType: datetime
- dbName: events.attributes
  name: attributes
  targetType: json
- dbName: clients.device_type
  name: deviceType
  targetType: string
description: Primer shablona
id: "7"
name: example
encoding: windows-1251
pipeline:
  - date:
      column: timestamp
      format: DF_RFC3339
  - intEnum:
      column: deviceType
      map:
        "1": STB-GW
        "2": STB
      unknownValue: unknown
  - rawJson:
      column: attributes
```

5.7. Плагины

Описание доступных плагинов с рекомендациями по настройке предоставляются заказчику по требованию.

6. Логирование

Лог-файлы компонентов системы могут быть получены через стандартный механизм доступа к логам в среде kebernetes при помощи команды kubectl logs <pod name>.

6.1. Форматы лог файлов

6.1.1. Consumer Client

Лог-файл компонента Consumer Client имеет следующий формат:

[Time] [Level] Info

Time - Дата и время формирования записи с точностью до секунд и с дробными секундами.

Level - Уровень логирования. Один из уровней (по возрастанию): *trace, debug, info, warn, error, fatal*.

Info - Текстовая информация - Дополнительная информация, которую можно использовать для поиска и трассировки ошибок.

Пример записи нескольких строчек лог-файла Consumer Client:

```
[2021-04-23T14:20:04.74306876+02:00][debug] Sent keepalive message
[2021-04-23T14:20:04.7514284+02:00][trace] recv: finish streamLoaderId=c0b5e3d5-282d-4697-8c28-2279dae075f7
[2021-04-23T14:20:04.751475174+02:00][trace] msgpack: start     streamLoaderId=c0b5e3d5-282d-4697-8c28-
2279dae075f7
[2021-04-23T14:20:04.751734195+02:00][trace] msgpack: finish    streamLoaderId=c0b5e3d5-282d-4697-8c28-
2279dae075f7
```

6.1.2. Event Server

Лог-файл компонента Event Server имеет следующий формат:

[Level][msg] [Time] [Details]

Level - Уровень логирования. Один из уровней (по возрастанию): *trace, debug, info, warn, error, fatal*.

msg - Логируемое сообщение

Time - Дата и время формирования записи с точностью до секунд и с дробными секундами.

Details - Формат JSON, дополнительная информация к сообщению msg.

Пример записи нескольких строчек лог-файла компонента Event Server:

```
{"level":"debug","msg":"flush events","time":"2021-04-23T13:55:50.16944293Z"}
{"level":"debug","msg":"events saved","time":"2021-04-23T13:55:50.174235394Z"}
{"details":{"clientCount":1,"clientIds":["0B0026BEC1"],"eventIds":[4136908]},"level":"debug","msg":"Broker flush
debug","time":"2021-04-23T13:55:50.174281302Z"}
{"details":{"count":1},"level":"debug","msg":"events pushed","time":"2021-04-23T13:55:50.178871186Z"}
{"details":{"reason":"timeout"},"level":"debug","msg":"flush finished","time":"2021-04-23T13:55:50.178913526Z"}
```

6.1.3. Api Gateway

Формат лог-файла аналогичен формату описанному в разделе 6.1.2

6.1.4. Consumer Server

Формат лог-файла аналогичен формату описанному в разделе 6.1.2