Руководство администратора ПО «Система логической репликации данных из MySQL в Postgresql»

1.Введение

1.1. Область применения

Настоящий документ предназначен для сотрудников эксплуатирующей организации и отражает основные функциональные возможности и порядок действий при выполнении операций, связанных с администрированием программного обеспечения «Система логической репликации данных из MySQL в Postgresql» (далее - «Система»)

1.2. Перечень выполняемых функций администратора/оператора

В перечень выполняемых функций администратора Системы входят:

- Установка и настройка Системы
- Реализация планов устранения сбоев и нетиповых нештатных ситуаций
- Выполнение сбора и предоставление в вышестоящую линию технической поддержки информации для воспроизведения технических проблем и выработки решений по их разрешению
- Реализация рекомендаций по устранению нештатных ситуаций, полученных с вышестоящей линии поддержки
- Восстановление работоспособности Системы при сбоях в работе функциональных модулей
- Разработка решения по устранению технических проблем в работе функциональных модулей

1.3. Уровень подготовки администратора/оператора

Администратор/оператор (далее по тексту Администратор) Системы должен обладать знаниями Javascript, уметь пользоваться и настраивать среду функционирования контейнеров или систему оркестрации, используемую на предприятии.

Рекомендуемая численность персонала для эксплуатации Системы — 1 штатная единица.

Администраторы Системы должны пройти обязательную общую и специальную подготовку для работы с Системой.

Общая подготовка должна включать в себя получение знаний и навыков работы с Системой в качестве администратора.

Специальная подготовка должна включать в себя получение знаний и навыков в объеме, необходимом для выполнения своих должностных обязанностей

1.4. Перечень документации

В состав документации, с которой необходимо ознакомиться администратору Системы входят:

- описание функциональных характеристик Системы
- описание процессов, обеспечивающих поддержание жизненного цикла программного обеспечения.

2. Установка Системы

В данном разделе будет описана установка Системы на Debian Linux. Предполагается, что были предварительно установлены также Docker, Docker Compose, RabbitMQ, а также, используемые СУБД: PostgreSQL и MySQL.

2.1. Системные требования к ПО

Минимальные аппаратные требования:

- Операционная система, способная запускать контейнеры. Предпочтительно Linux.
- Система управления контейнерной виртуализацией. Предпочтительно Docker Swarm или Kubernetes.
- Подключение к серверу очередей RabbitMQ
- Количество логических ядер процессора: 4
- Семейство процессоров: x86
- Частота процессора: 3.0. ГГц
- Объем установленной памяти: 16 Гб

2.1.2. Минимальные требования к сторонним компонентам и/или системам, необходимым для установки и работы ПО

- Debian 11 (Открытая лицензия GNU)
- Docker 24.0.2 (open-source community edition)
- RabbitMQ (Открытая лицензия Mozilla Public License)
- Grafana Loki 2.6.1 (Открытая лицензия GNU)
- Grafna 9.2.2 (Открытая лицензия GNU)
- Postgresql 14 (Открытая лицензия PostgreSQL license)
- MySQL 8.0 (open-source community edition Открытая лицензия GNU)

2.1.3. Языки программирования

При разработке Системы был использован язык программирования GoLang 1.20 (открытая лицензия BSD)

2.2. Порядок установки

1. Создайте папку /home/app
2. Смонтируйте диск с дистрибутивом в папку /mnt
3. Скопируйте из дистрибутива исходники из папки /mnt в папку /home/app
4. Смените текущую папку на /home/app и выполните команды
   sudo chown 10001:10001 ./volumes/loki
   sudo chown 472:472 ./volumes/grafana
5. Отредактируйте файл docker-compose.yml, в соответствии с пунктами 3.2 и 3.3 данного документа
6. Создайте и отредактируйте файлы настроек для модулей, в соответствии с пунктами 3.2.1, 3.3.1 и 3.3.2 данного документа
7. Смените текущую папку на /home/app и выполните в ней команду
   docker compose -up -d --build
8. Войдите браузером на ваш сервер на порт 3000 в систему мониторинга с пользователем admin и паролем admin. Измените пароль на безопасный.

3. Настройка Системы

3.1. Общие сведения

В данном документе приводятся примеры настройки Системы с использованием среды Docker Compose. Настройка операционной системы, сервера очередей RabbitMQ, СУБД, а также возможная настройка использования систем оркестрации, находятся вне компетенции этого документа и не будут тут описаны.

3.2. Модуль чтения данных из MySQL

3.2.1. Конфигурируемые параметры

Для корректной работы модуля чтения данных из MySQL требуется настроить сервер MySQL как master-сервер для репликации. Модуль также требует настройки следующих переменных окружения:

- AMQP_HOST — имя хоста или IP сервера RabbitMQ, с указанием порта и учетной записи. Пример смотрите ниже.
- AMQP_EXCHANGE — точка обмена на сервере RabbitMQ
- MYSQL_ADDRESS — адрес MySQL сервера
- MYSQL_USER — имя пользователя MySQL сервера
- MYSQL_PASSWORD — пароль пользователя MySQL сервера
- MYSQL_DATABASE — имя базы данных на MySQL сервере
- MYSQL_TABLES — список таблиц на MySQL сервере, подлежащих репликации через точку с запятой
- MYSQL_DUMP_PATH — путь к файлу mysqldump. Настройка модуля по умолчанию включает в себя версию этого файла для MySQL версии 8.0. Для этой версии сервера, заполнение переменной является необязательным. В случае использования другой версии сервера MySQL требуется указать эту переменную и указать в ней путь к файлу mysqldump с совпадающей версией.
- METRICS_PORT - порт к подсистеме проверки работоспособности.
- LOG_LEVEL - уровень логгирования. Поддерживаемые значения:
    - error
    - warn
    - info
    - debug
    - trace

Пример настройки модуля:

```
 replicator:
   build:
    context: ./replicator/
   restart: always
   ports:
    - '80:80'
   volumes:
    - ./replicator/config.json:/app/config.json:ro
   environment:
    MYSQL_ADDRESS: host.docker.internal:3306
```

```
  MYSQL_USER: root
  MYSQL_PASSWORD: 2j@m%TDcwTZ5dWA
  MYSQL_DATABASE: lpg
  MYSQL_TABLES: test;test1
  AMQP_HOST: amqp://rabbit:Q3ij6X4DZnb9uPT@host.docker.internal:5672/
  AMQP_EXCHANGE: transport
  MYSQL_DUMP_PATH: mysqldump
  LOG_LEVEL: trace
extra_hosts:
  - "host.docker.internal:host-gateway"
depends_on:
  - pgout
```

3.3. Модуль сохранения в PostgreSQL

3.3.1. Конфигурируемые параметры

Для корректной работы модуля, необходимо настроить для него следующие переменные окружения:

- SETTINGS_FILE - путь к файлу с настройками запросов. Файл подключается к контейнеру с помощью volume. В данной переменной окружения указывается локальный путь внутри контейнера, к которому был примонтирован volume.
- METRICS_PORT - порт к подсистеме проверки работоспособности.
- AMQP_HOST — имя хоста или IP сервера RabbitMQ, с указанием порта и учетной записи. Пример смотрите ниже.
- AMQP_EXCHANGE — точка обмена на сервере RabbitMQ
- DB_HOST — адрес сервера PostgreSQL
- DB_PORT— порт сервера PostgreSQL
- DB_USER — пользователь базы данных
- DB_PASSWORD — пароль пользователя базы данных
- DB_NAME — имя базы данных
- DB_CONN_MAX_TIME — устанавливает максимальное время перед переиспользованием соединения
- DB_MAX_OPEN_CONNECTIONS — максимальное количество соединений с базой данных
- DB_MAX_IDLE_CONNECTIONS — максимальное количество неиспользуемых соединений с базой данных
- LOG_LEVEL - уровень логгирования. Поддерживаемые значения:
    - error
    - warn
    - info
    - debug
    - trace

Пример настройки модуля:

pgout:
build:
context: ./pg_out/
restart: always
volumes:
- ./pg_out/config.json:/app/config.json:ro
environment:
AMQP_HOST: amqp://rabbit:Q3ij6X4DZnb9uPT@host.docker.internal:5672/
AMQP_EXCHANGE: transport
SETTINGS_FILE: /app/config.json
DB_HOST: host.docker.internal
DB_PORT: 5432
DB_USER: postgres
DB_PASSWORD: LDP8brN7B8ZLzf5Q879QXuJYXCwm9nP
DB_NAME: test
LOG_LEVEL: trace
extra_hosts:
- "host.docker.internal:host-gateway"

3.3.2. Обработка поступающих запросов

Каждая запись таблицы базы данных, полученная от модуля чтения данных из MySQL содержит в себе тип операции: вставка, обновление, удаление и название таблицы, для которой должна производиться эта операция.

Для каждой таблицы и для каждого типа операции файл настройки запросов должен содержать текст соответствующего SQL-запроса.

Для того, чтобы модуль обработал конкретный поступающий запрос, его следует прописать в файле, подключенном к модулю и указанном в переменной окружения SETTINGS_FILE. Этот файл содержит в текстовом формате json — объект, который в свою очередь, содержит объекты:

- insert_sql — содержит SQL-запросы на вставку данных в таблицу.

  Ключами объекта insert_sql являются имена таблиц в базе MySQL сервера. Запросы на вставку должны содержать конструкцию on conflict для обработки действий с записями, уже существующими в таблицах базы данных. Это необходимо по той причине, что при перезапуске модуля чтения данных из MySQL, он перезапускает процесс репликации и передает все записи из таблиц в запросах на вставку.

- update_sql — содержит SQL-запросы на модицикацию данных в таблице. Ключами объекта являются имена таблиц в базе MySQL сервера.

- delete_sql — содержит SQL-запросы на удаление данных из таблицы. Ключами объекта являются имена таблиц в базе MySQL сервера.

- insert_sql — содержит SQL-запросы на вставку данных в таблицу.

  Ключами объекта insert_sql являются имена таблиц в базе MySQL сервера.

Принимаемые сообщения могут быть также обработаны с помощью JavaScript-кода, который позволит модифицировать запросы к базе данных. Для этих целей используется объект requests. Значениями объекта requests являются строки, содержащие в себе JavaScript-код. Для каждой команды должен быть написан JavaScript-код, который должен вернуть SQL-запрос к базе данных в параметре sql. Для обработки запросов, в JavaScript передаются прараметры:

raw_data - RAW-содержимое запроса от модуля чтения из MySQL. Представляет собой json-закодированную строку. Передается в JavaScript как массив байт.

В JavaScript дополнительно передаются внешние функции:

- atob - преобразует строку в base64
- btoa - преобразует base64 в строку

Пример файла настройки обработчика запросов:

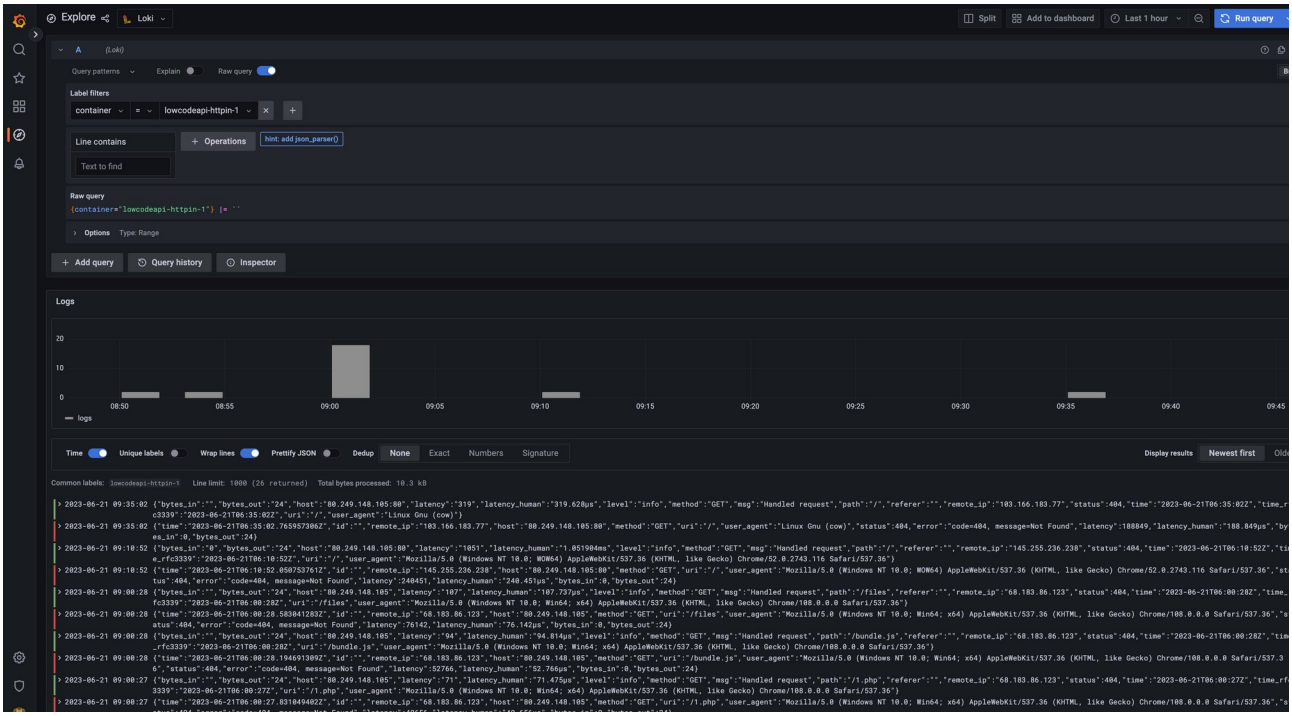
```
{
"requests": { },
"insert_sql": {
"test":"insert into test (id, caption) values (:id, :caption) on conflict (id) do update set
caption=EXCLUDED.caption",
"test1":"insert into test1 (id, name) values (:id, :name) on conflict (id) do update set
name=EXCLUDED.name"
},
"update_sql": {
"test":"update test set caption = :caption where id = :id",
"test1":"update test1 set name = :name where id = :id"
},
"delete_sql": {
"test":"delete from test where id = :id",
"test1":"delete from test1 where id = :id"
}
}
```

4. Система мониторинга

В качестве системы мониторинга используется Grafana Loki — это набор компонентов для полноценной системы работы с логами. Loki-стек состоит из трёх компонентов: Promtail, Loki, Grafana. Promtail собирает логи, обрабатывает их и отправляет в Loki. Loki их хранит. А Grafana умеет запрашивать данные из Loki и показывать их. Loki можно использовать не только для хранения логов и поиска по ним. Весь стек даёт большие возможности по обработке и анализу поступающих данных

Чтобы открыть интерфейс системы мониторинга, перейдите в браузере на IP Вашего сервера и порт 3000. Если Вы входите туда в первый раз, используйте логин admin и пароль admin. После первого входа система попросит Вас изменить пароль на безопасный.

Интерфейс выглядит так:



Выберите в меню пункт «Explore» - Вы увидите страницу поиска логов.

Сам запрос состоит из двух частей: selector и filter. Selector — это поиск по индексированным метаданным (лейблам), которые присвоены логам, а filter — поисковая строка или регэксп, с помощью которого отфильтровываются записи, определённые селектором.

Выберите в разделе Label filters в ниспадающем списке Label значение container, а в ниспадающем списке value выберите нужный контейнер. Выполните запрсо Run query и Вы увидите логи выбранного контейнера.