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

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 Postgresql в MySQL» (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 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)
- Postgresql 14 (Открытая лицензия PostgreSQL license)
- MySQL 8.0 (open-source community edition Открытая лицензия GNU)

При необходимости внешнего логгирования и мониторинга допускается использовать дополнительное сервисное ПО:

- Grafana Loki 2.6.1 (Открытая лицензия GNU)
- Grafna 9.2.2 (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. Смонтируйте диск с дистрибутивом в папку /mnt
2. Скопируйте из дистрибутива исходники из папки /mnt в папку /root
3. Смените текущую папку на /root
4. Отредактируйте файл docker-compose.yml, в соответствии с пунктами 3.3 и 3.4.1 данного документа
5. Создайте и отредактируйте файлы настроек для модуля, в соответствии с пунктами 3.4.2 данного документа
6. Выполните команду
   docker compose -up -d --build

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. Однако, для начала работы, требуется предварительно настроить, ранее установленную, СУБД Postgresql.

3.2. Настройка Postgresql

Необходимо создать пользователя pglogrepl, ис под которого будет работать репликация:

create user pglogrepl with replication password 'secret';

Необходимо дать права этому пользователю:

grant all on schema public to pglogrepl;

Необходимо дать разрешение этому пользователю подключаться к базе данных. Для этого надо отредактировать файл pg_hba.conf, добавив туда строки (может потребоваться исправить адресацию. В текущем примере указана возможность доступа как напрямую к локальному серверу, так и из среды Docker Compose):

host replication pglogrepl 127.0.0.1/32 md5
host replication pglogrepl 172.0.0.1/32 md5

Измените следующие строки в файле postgresql.conf:

wal_level=logical
max_wal_senders=5
max_replication_slots=5
Теперь Ваш сервер готов к запуску Системы.

3.3. Модуль чтения данных из PostgreSQL

Модуль 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
- PG_HOST — имя хоста или IP PostgreSQL сервера
- PG_PORT – номер порта PostgreSQL сервера. Значение по умолчанию 5432.
- PG_USER — имя пользователя PostgreSQL сервера
- PG_PASSWORD — пароль пользователя PostgreSQL сервера
- PG_DATABASE — имя базы данных
- PG_REPLICATION_NAME – название публикации. Значение по умолчанию pglogrepl_demo.
- PG_TABLES — список таблиц, с указанием схемы, подлежащих репликации через точку с запятой
- LOG_LEVEL - уровень логгирования. Поддерживаемые значения:
    - error
    - warn
    - info
    - debug

Пример настройки модуля:

```
pg_in:
build:
  context: ./pg_in/
restart: always
environment:
  PG_HOST: host.docker.internal
  PG_PORT: 5432
  PG_USER: postgres
  PG_PASSWORD: LDP8brN7B8ZLzf5Q879QXuJYXCwm9nP
  PG_DATABASE: test
  PG_TABLES: public.test;public.test1
  AMQP_HOST: amqp://rabbit:Q3ij6X4DZnb9uPT@host.docker.internal:5672/
  AMQP_EXCHANGE: transport
  LOG_LEVEL: debug
extra_hosts:
  - "host.docker.internal:host-gateway"
depends_on:
  - myout
```

3.4. Модуль сохранения в MySQL

3.4.1. 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.
- 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 MySQL
- 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

Пример настройки модуля:

```
  myout:
    build:
      context: ./my_out/
    restart: always
    volumes:
      - ./my_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:3306
      DB_USER: root
      DB_PASSWORD: VEr2sation
      DB_NAME: test
      LOG_LEVEL: debug
    extra_hosts:
      - "host.docker.internal:host-gateway"
```

3.4.2. Обработка поступающих запросов

Каждая запись таблицы базы данных, полученная от модуля чтения данных из PostgreSQL содержит в себе тип операции: инициализация, вставка, обновление, удаление или очистка; а также 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:

- init_sql – содержит SQL-запросы на первоначальную вставку данных в таблицу, инициируемые при начале процесса репликации.
  Ключами объекта insert_sql являются имена таблиц в базе PostgreSQL сервера.
  Запросы на вставку должны содержать конструкцию on conflict для обработки действий с записями, уже существующими в таблицах базы данных.
- insert_sql — содержит SQL-запросы на вставку данных в таблицу.

  Ключами объекта insert_sql являются имена таблиц в базе PostgreSQL сервера.
- update_sql — содержит SQL-запросы на модификацию данных в таблице. Ключами объекта являются имена таблиц в базе PostgreSQL сервера.
- delete_sql — содержит SQL-запросы на удаление данных из таблицы. Ключами объекта являются имена таблиц в базе PostgreSQL сервера.
- insert_sql — содержит SQL-запросы на вставку данных в таблицу.

  Ключами объекта insert_sql являются имена таблиц в базе PostgreSQL сервера.

Пример файла настройки обработчика запросов:

```
{    "init_sql": {
       "test":"insert into test (id, caption) values (:id, :caption) ON DUPLICATE KEY UPDATE caption=:caption;",
       "test1":"insert into test1 (id, caption) values (:id, :caption) ON DUPLICATE KEY UPDATE caption=:caption;"
    },
    "insert_sql": {
       "test":"insert into test (id, caption) values (:id, :caption) ON DUPLICATE KEY UPDATE caption=:caption;",
       "test1":"insert into test1 (id, caption) values (:id, :caption) ON DUPLICATE KEY UPDATE caption=:caption;"
    },
    "update_sql": {
       "test":"update test set caption = :caption where id = :id;",
       "test1":"update test1 set caption = :caption where id = :id;"
    },
    "delete_sql": {
       "test":"delete from test where id = :id;",
       "test1":"delete from test1 where id = :id;"
    },
    "truncate_sql": {
       "test":"delete from test;",
       "test1":"delete from test1;"
    }
}
```