СЕРВИС ПРОВЕДЕНИЯ КАСКАДНЫХ ОПЕРАЦИЙ

Инструкция для развертывания экземпляра ПО

Оглавление

1. Введение

Настоящий документ (далее – Описание) распространяется на программное обеспечение (далее – ПО) “*Сервис проведения каскадных операций*” разработанное компанией ***ООО «Фарос Медиа».***

Контакты:
Сайт: https://faros-media.ru/
Телефон: +7 967 101-46-43
Почта: CEO@FAROS-MEDIA.RU

1.1 Описание документа

Документ содержит сведения о порядке установки программного обеспечения “*Сервис проведения каскадных операций*” Также в документе приводятся требования к программному и аппаратному обеспечению.

1.2 Общие сведения

“*Сервис проведения каскадных операций*” – это техническое решение, предоставляющее возможность дополнительных попыток проведения платежа без изменения параметров запроса в случае прерывания по различным причинам. Дополнительные последовательные попытки происходят через резервных провайдеров автоматически на стороне сервиса по заранее заданной конфигурации.

1.3 Назначение программы

Программа предназначена для случаев, когда проведение платежа прерывается по различным причинам. Функционал каскадного проведения операций включает в себя последовательные дополнительные попытки проведения платежа через резервных провайдеров без изменения платежного метода.

1.4 Уровень подготовки пользователей

Для интеграции API пользователь должен иметь квалификацию разработчика не ниже уровня Regular Middle.

1.5 Требования к эксплуатации серверной части

Обеспечение функционирования ПО серверной части “Сервис проведения каскадных операций” реализовано на базе серверной операционной системы Linux. Минимальной конфигурацией аппаратной составляющей являются:

- Современная ОС: Linux;
- Оперативная память: 2 ГБ;
- Свободное дисковое пространство: не менее 20 Gb;
- Количество логических ядер процессора: 2;
- Частота процессора: 3.50 GHz.

Возможно разворачивание экземпляра ПО и на других ОС, поддерживающих платформу для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации Docker, например Windows 10 (Профессиональная или Корпоративная).

1.6 Глоссарий

Термин	Значение
API	Описание способов взаимодействия одной компьютерной программы с другими.
БД	База данных – набор структурированных данных, хранящихся в виде таблицы.
Инстанс	Экземпляр класса (объекта) в объектно-ориентированном программировании.
Мерчант	Партнер, пользователь программного обеспечения “*Сервис проведения каскадных операций*”.
Очередь	Некая структура данных, которая обеспечивает хранение и передачу двоичных данных между различными участниками системы.
Сервис	Независимо компилируемый программный модуль, динамически подключаемый к основной программе и предназначенный для расширения и/или использования её возможностей. Сервисы обычно выполняются в виде библиотек общего пользования.
ПО	Программное обеспечение.
Каскадные операции	Дополнительные попытки проведения платежа без изменения параметров запроса в случае прерывания по различным причинам.
Конфигурация	Набор настроек, методов и параметров, направленных на систематизацию поведения программного обеспечения.

2. Установка

2.1 Описание компонентов

Состав компонентов для установки:

- multihub-mock-service – сервис создания тестового окружения для внешних мерчантов;
- payments-router – сервис отправки уведомлений;
- rabbitmq-service – программный брокер сообщений на основе стандарта AMQP;
- redis – резидентная система управления базами данных класса NoSQL;
- db-pay-service – свободная объектно-реляционная система управления базами данных.

2.2 Порядок установки

Для развертывания экземпляра **ПО,** необходим docker-compose (инструментальное средство, входящее в состав **Docker**. Оно предназначено для решения задач, связанных с развертыванием проектов). Его можно установить, следуя инструкции на официальном сайте https://docs.docker.com/compose/install/.

Необходимо создать файл docker-compose.yml с содержимым:

```
version: '2.4'

services:


 multihub-mock-service:

  image: 'docker.faros-media-test.ru/paymentboom_multihub-mock-service:feat-temp-remove-check-hash.22'

  restart: always

  environment:

    - AMQP_USER=rmuser

    - AMQP_PASSWORD=rmpassword

    - LOGGER_LEVEL=debug

    - TRANSPORT_CONNECTION_PROTOCOL=amqp

    - TRANSPORT_CONNECTION_HOSTNAME=rabbitmq-service
```

- TRANSPORT_CONNECTION_PORT=5672
- TRANSPORT_CONNECTION_VHOST=/
- TRANSPORT_CONNECTION_USERNAME=rmuser
- TRANSPORT_CONNECTION_PASSWORD=rmpassword
- TRANSPORT_CONNECTION_FRAME_MAX=0
- TRANSPORT_CONNECTION_CHANNEL_MAX=0
- TRANSPORT_CONNECTION_HEARTBEAT=0
- TRANSPORT_AUTO_RECONNECT=true
- TRANSPORT_AUTO_RECONNECT_TIMEOUT=5000
- TRANSPORT_SERVICE_NAME=multihub_mock_service
- TRANSPORT_SHARED_SERVICE_QUEUE=true
- TRANSPORT_SHARED_SERVICE_QUEUE_MESSAGE_TTL=10000
- TRANSPORT_EXCLUSIVE_QUEUE_ACK=false
- TRANSPORT_SHARED_SERVICE_QUEUE_ACK=false
- TRANSPORT_MAIN_EXCHANGE_NAME=main
- TRANSPORT_MAIN_REPLY_EXCHANGE_NAME=main_reply
- TRANSPORT_PUBLISH_AWAIT_TIMEOUT=10000
- TRANSPORT_PREFETCH=100
- TRANSPORT_MAIN_SCHEDULE_EXCHANGE_NAME=main_schedule
- TRANSPORT_MAIN_SCHEDULE_QUEUE_NAME=main_schedule
- TRANSPORT_MAIN_GARBAGE_EXCHANGE_NAME=main_garbage
- TRANSPORT_MAIN_GARBAGE_QUEUE_NAME=main_garbage
- DATABASE_MASTER_DATABASE=db_pay
- DATABASE_MASTER_HOST=db-pay-service
- DATABASE_MASTER_PORT=5432

- DATABASE_MASTER_USER=postgres
- DATABASE_MASTER_PASSWORD=STRONG_password
- DATABASE_SLAVE_DATABASE=db_pay
- DATABASE_SLAVE_HOST=db-pay-service
- DATABASE_SLAVE_PORT=5432
- DATABASE_SLAVE_USER=postgres
- DATABASE_SLAVE_PASSWORD=STRONG_password
- DATABASE_POOL_CONNECTION_TIMEOUT_MILLIS=2000
- DATABASE_POOL_IDLE_TIMEOUT_MILLIS=30000
- DATABASE_POOL_MAX=10
- DATABASE_LOGGING=false
- SERVER_HOST=0.0.0.0
- SERVER_PORT=4000
- SERVICE_NAMES_REFERENCE_API=ph_reference-service
- SERVICE_NAMES_PAYMENT_STATUS=ph_payment-status
- SERVICE_NAMES_GATEWAY_CALLBACK_WORKER=ph_callback-worker
- SERVICE_NAMES_PAYMENT_NOTIFICATION=ph_payment-notification
- MULTIHUB_MOCK_SERVICE_URL=http://5.182.4.89:4000/v1
- MULTIHUB_DEFAULT_SCHEMA_ENABLED=true
- KEY_VALUE_STORAGE_REDIS_HOST=redis
- KEY_VALUE_STORAGE_REDIS_PORT=6379
- KEY_VALUE_STORAGE_REDIS_TIMEOUT=5000
- KEY_VALUE_STORAGE_REDIS_CONNECTION_TIMEOUT=5000
- MONITORING_SERVICE_NAME=multihub-mock-service
- PROMETHEUS_HTTP_PORT=9696

```
    - K8S_POD_NAME=paymentboom_multihub-mock-service
  ports:
    - "4000:4000"
  depends_on:
    - rabbitmq-service
    - db-pay-service
    - redis



 payments-router-service:
  image: 'docker.faros-media-test.ru/paymentboom_payments-
router:master.119'
  restart: always
  environment:
    - NGINX_WORKERS=2
    - TRANSPORT_CONNECTION_PROTOCOL=amqp
    - TRANSPORT_CONNECTION_HOSTNAME=rabbitmq-service
    - TRANSPORT_CONNECTION_PORT=5672
    - TRANSPORT_CONNECTION_VHOST=/
    - TRANSPORT_CONNECTION_FRAME_MAX=0
    - TRANSPORT_CONNECTION_CHANNEL_MAX=0
    - TRANSPORT_CONNECTION_HEARTBEAT=0
    - TRANSPORT_AUTO_RECONNECT=true
    - TRANSPORT_AUTO_RECONNECT_TIMEOUT=5000
    - TRANSPORT_SERVICE_NAME=payments_router
```

- TRANSPORT_SHARED_SERVICE_QUEUE=true
- TRANSPORT_SHARED_SERVICE_QUEUE_MESSAGE_TTL=10000
- TRANSPORT_EXCLUSIVE_QUEUE_ACK=false
- TRANSPORT_SHARED_SERVICE_QUEUE_ACK=false
- TRANSPORT_MAIN_EXCHANGE_NAME=main
- TRANSPORT_MAIN_REPLY_EXCHANGE_NAME=main_reply
- TRANSPORT_PUBLISH_AWAIT_TIMEOUT=60000
- TRANSPORT_PREFETCH=100
- TRANSPORT_MAIN_SCHEDULE_EXCHANGE_NAME=main_schedule
- TRANSPORT_MAIN_SCHEDULE_QUEUE_NAME=main_schedule
- TRANSPORT_MAIN_GARBAGE_EXCHANGE_NAME=main_garbage
- TRANSPORT_MAIN_GARBAGE_QUEUE_NAME=main_garbage
- DATASOURCE_DBPAY_HOST=db-pay-service
- DATASOURCE_DBPAY_PORT=5432
- DATASOURCE_DBPAY_USER=postgres
- DATASOURCE_DBPAY_DATABASE=db_pay
- DATASOURCE_DBPAY_LOGGING=false
- DATASOURCE_DBPAY_POOLSIZE=30
- SERVICE_NAME_PAYMENT_STATUS=ph_payment-status
- SERVICE_NAME_CALLBACK_WORKER=ph_callback-worker
- TRANSACTION_WATCH_INTERVAL=10
- LOGGER_LEVEL=debug
- PROMETHEUS_HTTP_HOST=0.0.0.0
- PROMETHEUS_HTTP_PORT=9696

```
  - DATASOURCE_DBPAY_PASSWORD=STRONG_password
  - TRANSPORT_CONNECTION_USERNAME=rmuser
  - TRANSPORT_CONNECTION_PASSWORD=rmpassword
  - K8S_POD_NAME=payments-router-service-pod-name
 ports:
  - "8080:8080"
 depends_on:
  - rabbitmq-service
  - db-pay-service



rabbitmq-service:

 image: rabbitmq:3.11.8-management
 hostname: rabbitmq-service
 restart: always
 environment:
  - RABBITMQ_DEFAULT_USER=rmuser
  - RABBITMQ_DEFAULT_PASS=rmpassword
  - RABBITMQ_SERVER_ADDITIONAL_ERL_ARGS=-rabbit channel_max 0
 volumes:
  - ./rabbitmq:/var/lib/rabbitmq
 command:
  - bash
```

```
    - -c
    - |
      cd /opt/rabbitmq/plugins/
      apt update
      apt install -y wget
      wget https://github.com/rabbitmq/rabbitmq-delayed-message-exchange/releases/download/3.11.1/rabbitmq_delayed_message_exchange-3.11.1.ez
      rabbitmq-plugins enable rabbitmq_delayed_message_exchange
      rabbitmq-server start
  ports:
    - "15672:15672"
    - "5672:5672"

 db-pay-service:

  container_name: db-pay-service
  image: 'postgres:14'
  environment:
    POSTGRES_PASSWORD: STRONG_password
    POSTGRES_USER: postgres
    POSTGRES_DB: db_pay
  restart: on-failure
  volumes:
    - ./postgres:/var/lib/postgresql/
  ports:
```

```
    - "5432:5432"
  command: [ "postgres", "-c", "wal_level=logical" ]

 redis:
  image: 'redis:7.2-rc-bullseye'
  environment:
    - ALLOW_EMPTY_PASSWORD=yes
    - REDIS_AOF_ENABLED=no
    - vm.overcommit_memory=1
  volumes:
    - ./redis:/data
  ports:
   -  "6379:6379"


volumes:
  rabbitmq:
    external: true
  postgres:
    external: true
  redis:
    external: true
```