Инструкция по развертыванию экземпляра ПО

“Сервис по приему и верификации запросов”

1. Введение

Наименование программного обеспечения “Сервис по приему и верификации запросов”.

1.1 Описание документа

Документ содержит сведения о порядке установки программного обеспечения “Сервис по приему и верификации запросов”. Также в документе приводятся требования к программному и аппаратному обеспечению.

1.2 Уровень подготовки пользователей

Документ предназначен для администратора АС, выполняющего установку программного обеспечения “Сервис отправки уведомлений при изменении статусов запросов”/

2. Требования к эксплуатации серверной части

Обеспечение функционирования ПО серверной части “Сервис по приему и верификации запросов” 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).

3. Установка

3.1 Описание компонентов

Состав компонентов для установки:

- `multihub-mock-service` – сервис создания тестового окружения для внешних мерчантов;
- `gateway_callback_worker` – 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.

3.2 Порядок установки

Для разворачивание 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.itisphera-test.ru/paymentboom_multihub-mock-service'
```

#Сервис создания тестового окружения для внешних мерчантов.

#Для тестирования ошибок, нужно в сумме платежа ввести "номер ошибки * 100".

#Для оплаты создается платежная страница и после создания транзакции в ответе будет на нее ссылка, где можно либо выбрать каким образом зафиналить транзакцию, ответ должен прийти в `call-back-worker`.

#Для выплаты создатается транзакция `created` и приходит ответ, далее через несколько секунд транзакция финалится success и приходит ответ в `call-back-worker`.

```
  mem_limit: 256m

    mem_reservation: 128M
```

```
    cpus: 0.2
    environment:  # необходимые переменные для запуска сервиса
      - LOGGER_LEVEL=debug
      - TRANSPORT_CONNECTION_PROTOCOL=amqp
      - TRANSPORT_CONNECTION_HOSTNAME=rabbitmq-service
      - TRANSPORT_CONNECTION_PORT=5672
      - TRANSPORT_CONNECTION_VHOST=/
      - TRANSPORT_CONNECTION_USERNAME=user
      - TRANSPORT_CONNECTION_PASSWORD=STRONG_password
      - 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=3000
      - 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="https://test.test/v1"
      - MULTIHUB_DEFAULT_SCHEMA_ENABLED=false
      - KEY_VALUE_STORAGE_REDIS_HOST=redis-cache-service
      - 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
    ports: # порты для подключения к сервису
     - "4000:4000"


  gateway_callback_worker:
      image: 'docker.itisphera-
test.ru/paymentboom_gateway_callback_worker'
```

#Сервис, который читает сообщения из сервера очереди и отправляет информацию об изменении состояния транзакции на Callback API клиента.

```
    mem_limit: 256m
      mem_reservation: 128M
      cpus: 0.2
    environment: # необходимые переменные для запуска сервиса
    - NGINX_WORKERS=2
        - DB_NAME=db_pay
        - DB_READ_HOST=db-pay-service
        - DB_READ_DB_PORT=5432
        - DB_WRITE_HOST=db-pay-service
        - DB_WRITE_DB_PORT=5432


        - DB_CONN_TIMEOUT_MILLIS_MASTER=3000
        - DB_IDLE_TIMEOUT_MILLIS_MASTER=10000
        - DB_MAX_CLIENTS_MASTER=10
        - DB_PING_TIMEOUT_MILLIS_MASTER=60000
```

```
        - DB_CONN_TIMEOUT_MILLIS_SLAVE=3000
        - DB_IDLE_TIMEOUT_MILLIS_SLAVE=10000
        - DB_MAX_CLIENTS_SLAVE=10
        - DB_PING_TIMEOUT_MILLIS_SLAVE=60000

    - AMQP_HOST=rabbitmq-service
        - AMQP_PORT=5672
        - AMQP_AUTORECONNECT=true
        - AMQP_AUTORECONNECT_TIMEOUT=5000
        - AMQP_SERVICE_NAME=user
        - AMQP_SHARED_SERVICE_QUEUE_ACK=true
        - AMQP_SHARED_SERVICE_QUEUE=true
        - AMQP_ADDITIONAL_GARBAGE_QUEUES_NAMES='["ph_callback-
worker_garbage"]'
        - AMQP_PREFETCH=100
        - AMQP_MAX_SEND_ATTEMPTS=200
        - AMQP_MAIN_SCHEDULE_QUEUE_NAME=ph_callback-worker_schedule
        - HTTP_REQUEST_TIMEOUT=7000
    - PROMETHEUS_HTTP_PORT=9696
    ports: # порты для подключения к сервису
        - "3000:3000"

  rabbitmq-service:
   # программный брокер сообщений на основе стандарта AMQP
    image: rabbitmq:3.11.8-management
      hostname: rabbitmq
```

```
      restart: always
      mem_limit: 256m
      mem_reservation: 128M
      cpus: 0.2
      environment: # необходимые переменные для запуска сервиса
      - RABBITMQ_DEFAULT_USER=rmuser
        - RABBITMQ_DEFAULT_PASS=rmpassword
        - RABBITMQ_SERVER_ADDITIONAL_ERL_ARGS=-rabbit channel_max 0
      volumes: # внешнее хранилище данных
        - ./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"
      ports:
        - "15672:15672" # порт для GUI
       - "5672:5672" # порт для подключения к сервису


  redis:
      image: 'bitnami/redis:latest'
     # резидентная система управления базами данных класса NoSQL
      mem_limit: 256m
      mem_reservation: 128M
```

```
    cpus: 0.2
    environment:  # необходимые переменные для запуска сервиса
      - ALLOW_EMPTY_PASSWORD=yes
    volumes:  # внешнее хранилище данных
      - ./redis:/var/lib/redis


    ports:  # порты для подключения к сервису
      -  "6379:6379"


  db-pay-service:
    # свободная объектно-реляционная система управления базами данных
    container_name: db-pay-service
    image: 'docker.itisphera-test.ru/postgres10'
    mem_limit: 512m
    mem_reservation: 256M
    cpus: 0.5
    environment: # необходимые переменные для запуска сервиса
      POSTGRES_PASSWORD: STRONG_password
      POSTGRES_USER: postgres
    POSTGRES_DB: db-pay
    restart: on-failure  # рестарт контейнера при сбое
    ports:  # порты для подключения к сервису
      - "5432:5432"


volumes: #внешние хранилища данных
```

```
  rabbitmq:
    external: true
  redis:
    external: true
```

Запустить командой:

```
docker-compose up -d
```