# Инструкция по развертыванию экземпляра «Системы гарантированной доставки http - сообщений»

# 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 гарантированной доставки http — сообщений» (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:

- инструкция по развертыванию экземпляра Системы
- 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, Apache Kafka.

## 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 Apache Kafka
- Количество логических ядер процессора: 4
- Семейство процессоров: x86
- Частота процессора: 3.0. ГГц
- Объем установленной памяти: 16 Гб

### 2.1.2. Минимальные ребования 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)
- Apache Kafka 2.13-2.8.1 (Открытая лицензия Apache License)
- Grafana Loki 2.6.1 (Открытая лицензия GNU)
- Grafna 9.2.2 (Открытая лицензия GNU)

## 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 3.2.1 и 3.3.1 данного документа
6. Создайте и отредактируйте файлы настроек для обоих модулей, в соответствии с пунктами 3.2.2, 3.2.3 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 Apache Kafka, а также возможная настройка использования систем оркестрации, находятся вне компетенции этого документа и не будут тут описаны.

## 3.2. Модуль приема сообщений

### 3.2.1 Конфигурируемые параметры

Для корректной работы модуля приема сообщений, 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.
- HOST - адрес сервиса, который будет слушать модуль. На этот адрес следует пробросить внешний порт или настроить проксирующий сервер для поддержки протокола https.
- KAFKA_HOST — имя хоста или IP сервера Kafka
- KAFKA_TOPIC - топик на сервере Kafka
- KAFKA_PARTITION — имя разделения (partition) на сервере Kafka
- KAFKA_TIMEOUT - таймаут запросов к серверу Kafka
- LOG_LEVEL - уровень логгирования. Поддерживаемые значения:
    - error
    - warn
    - info
    - debug
    - trace

**Пример настройки модуля:**

```
  httpin:
    build:
      context: ./http-in/
    restart: always
    ports:
      - '80:80'
    volumes:
      - ./http-in/config.json:/app/config.json:ro
    environment:
      HOST: :80
      KAFKA_HOST: host.docker.internal:9092
      KAFKA_TOPIC: test
```

```
    KAFKA_PARTITION: 0
    KAFKA_TIMEOUT: 10
    LOG_LEVEL: trace
    SETTINGS_FILE: /app/config.json
  extra_hosts:
    - "host.docker.internal:host-gateway"
```

### 3.2.2. Настройка маршрутизации

Для того, чтобы сервер обрабатывал конкретный URL, 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 еще два объекта:

- requests — содержит описание маршрутов и обработчики поступающих данных

  Ключами объекта requests являются описатели маршрутов. Они формируются путем конкатенации HTTP-метода, разделителя и относительного пути.

  Префиксом названия команды является HTTP-метод. Поддерживаемые методы: GET, POST, PUT, DELETE. После префикса должен находиться разделитель |. Затем должен быть указан маршрут. Маршрут может включать в себя как параметры, так и символ *, позволяющий обрабатывать различные маршруты одной командой. Примеры:

  - GET|/users
  - GET|/users/:id
  - GET|/users/files/*
  - POST|/users/:id

  Значениями объекта requests являются строки, содержащие в себе JavaScript-код.

- responses — содержит обработчики ответов, отправляемых модулем клиенту.

  Ключами объекта responses являются описатели маршрутов, описанные выше.

  Значениями объекта responses являются строки, содержащие в себе JavaScript-код.

**Пример настройки маршрутизации:**

```
{
  "requests": {
    "GET|/ping":""
  },
  "responses": {
    "GET|/ping":" "
  }
}
```

### 3.2.3. Обработка запросов

Для каждого поступающего запроса может быть написан JavaScript-код, который позволит модифицировать как входящие параметры, так и ответ на запрос. В силу ограничений формата json, в коде не поддерживаются переносы строк и кавычки. Эти символы могут быть вставлены в код с помощью символа экранирования, и соответственно, будут выглядеть как \n и \" .

Для обработки входящего запроса, в JavaScript передаются прараметры:

- raw_data - RAW-содержимое запроса. Представляет собой json-закодированную строку. Передается в JavaScript как массив байт.
- body - тело запроса. Передается в JavaScript как массив байт.
- cookies - массив cookie
- headers - массив с заголовками запроса
- method - HTTP-метод
- query_params - параметры запроса в URL
- url - URL запроса

JavaScript-код может проанализировать эти параметры и должен вернуть параметр raw_data, который будет отправлен модулю доставки сообщений.

Для обработки ответа на запрос, в JavaScript передается параметр raw_data - RAW-содержимое ответа.

В JavaScript дополнительно передаются внешние функции:

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

Если в разделе responses не указать соответствующий запросу маршрут, то модуль ответит кодом 200 Ok с пустым телом ответа.

## 3.3. Модуль доставки сообщений

### 3.3.1. Конфигурируемые параметры

- 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.
- ALLOW_INSECURE - флаг контролировать ли сертификат системы получателя
- HTTP_HOST — имя хоста или IP адрес системы получателя
- HTTP_TIMEOUT - таймаут запроса
- KAFKA_HOST - имя хоста или IP адрес сервера Kafka
- KAFKA_TOPIC - топик на сервере Kafka
- KAFKA_TIMEOUT - таймаут запросов к серверу Kafka
- KAFKA_GROUP - группа подписчиков сервера Kafka

- LOG_LEVEL - уровень логгирования. Поддерживаемые значения:
    - error
    - warn
    - info
    - debug
    - trace

**Пример настройки модуля:**

```
httpout:
  build:
    context: ./http-out/
  restart: always
  volumes:
    - ./http-out/config.json:/app/config.json:ro
  environment:
    KAFKA_HOST: host.docker.internal:9092
    KAFKA_TOPIC: test
    KAFKA_GROUP: consumer-group-id-1
    KAFKA_TIMEOUT: 10
    LOG_LEVEL: trace
    SETTINGS_FILE: /app/config.json
    HTTP_HOST: google.com
    HTTP_TIMEOUT: 10
  extra_hosts:
    - "host.docker.internal:host-gateway"
```

### 3.3.2. Обработка поступающих запросов

Каждый запрос, полученный от модуля приема сообщений, приходит с теми же настройками маршрутизации, которые были описаны выше в разделе 3.2.2.

Принимаемые сообщения могут быть обработаны с помощью JavaScript-кода, который позволит модифицировать как входящие параметры, так и проанализировать ответ на команду.

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 еще два объекта:

- requests — содержит описание маршрутов и обработчики поступающих данных

    Ключами объекта requests являются описатели маршрутов. Они должны совпадать с аналогичными маршрутами, указанными у модуля приема сообщений.

    Значениями объекта requests являются строки, содержащие в себе JavaScript-код.

- responses — содержит обработчики ответов, отправляемых модулем клиенту.

Ключами объекта responses являются описатели маршрутов, описанные выше.

Значениями объекта responses являются строки, содержащие в себе JavaScript-код.

Для каждой команды может быть написан JavaScript-код, который позволит модифицировать как входящие параметры, так и позволит изменить поведение при ответе на команду.

Для обработки входящей команды, в JavaScript передаются прараметры:

- raw_data - RAW-содержимое запроса. Представляет собой json-закодированную строку. Передается в JavaScript как массив байт.
- body - тело запроса. Передается в JavaScript как массив байт.
- cookies - массив cookie
- headers - массив с заголовками запроса
- method - HTTP-метод
- query_params - параметры запроса в URL
- url - URL запроса JavaScript-код может проанализировать и изменить эти параметры.

Для обработки ответа команды, в JavaScript передаются параметры:

- raw_data - RAW-содержимое ответа. Представляет собой json-закодированную строку. Передается в JavaScript как массив байт.
- body - тело ответа. Передается в JavaScript как массив байт.
- status_code - HTTP-код ответа
- headers - массив заголовков
- cookies - массив cookie

JavaScript-код может проанализировать эти параметры и должен вернуть параметр status_code.

Если status_code будет 2xx, сообщение будет помечено как обработанное. В противном случае, через некоторое время будет произведена попытка повторной доставки сообщения.

В JavaScript дополнительно передаются внешние функции:

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

Если в разделе requests не указать соответствующий запросу маршрут, то модуль отправит запрос в систему получатель в том виде, в котором его получил модуль приема сообщений

Если в разделе responses не указать соответствующий запросу маршрут, то модуль будет проверять статус ответа на запрос и если его status_code будет 2xx, сообщение будет помечено как обработанное. В противном случае, через некоторое время будет произведена попытка повторной доставки сообщения.

Пример файла настройки обработчика запросов:

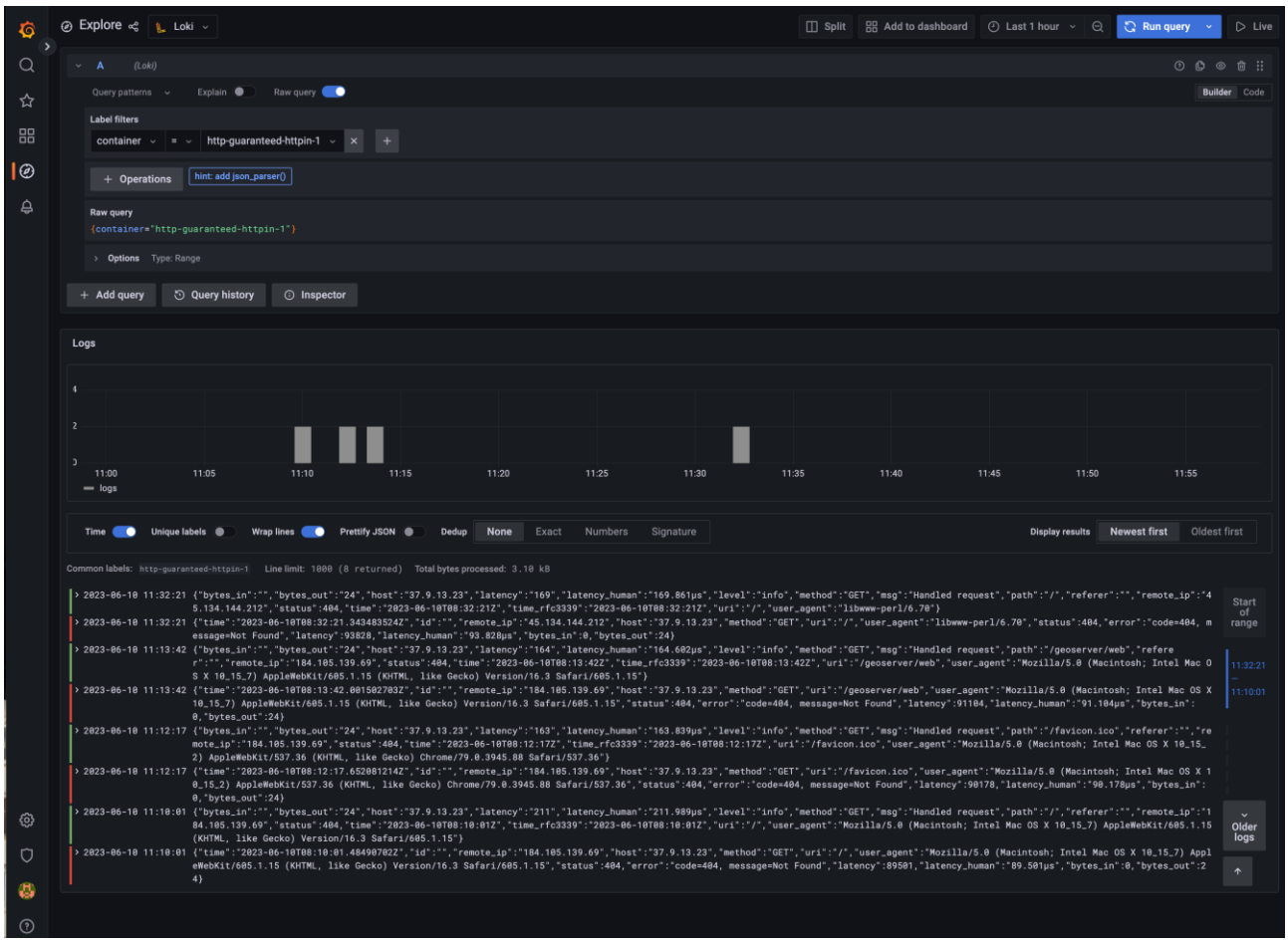
```
{
   "requests": {
      "default":" ",
      "GET|/ping":"method='GET';\nurl='https://google.com/';"
   },
   "responses": {
      "default":" ",
      "GET|/ping":" "
   }
}
```


# 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.