Руководство администратора ПО

«Идемпотентный HTTP-прокси сервис»

1. Введение

1.1. Область применения

Настоящий документ предназначен для администраторов, эксплуатирующих программное обеспечение «Идемпотентный HTTP-прокси сервис» (далее — «Сервис»). Руководство содержит порядок установки, настройки, администрирования и взаимодействия с API Сервиса.

1.2. Перечень выполняемых функций администратора/оператора

В перечень выполняемых функций администратора Сервиса входят:

- Установка и настройка Сервиса в контейнерной среде
- Настройка переменных окружения и подключение к внешним сервисам (Redis)
- Мониторинг работы Сервиса и анализ логов
- Восстановление работоспособности при сбоях
- Конфигурация параметров безопасности (TTL, таймауты, блокировки)
- Масштабирование системы при увеличении нагрузки

1.3. Уровень подготовки администратора/оператора

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

Рекомендуемая численность персонала для эксплуатации Сервиса — 1 штатная единица.

Администраторы Сервиса должны пройти обязательную общую и специальную подготовку для работы с Сервисом:

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 Сервиса входят:

- Функциональная спецификация ПО «Идемпотентный HTTP-прокси сервис»
- Техническое задание на разработку сервиса
- Описание процессов, обеспечивающих поддержание жизненного цикла программного обеспечения

2. Установка Системы

2.1. Системные требования

2.1.1. Аппаратные требования

- Количество логических ядер процессора: 2
- Семейство процессоров: x86
- Частота процессора: 2.0 ГГц
- Объём оперативной памяти: 4 ГБ
- Сетевой интерфейс: 1 Гбит/с
- Дисковое пространство: 10 ГБ

2.1.2. Программные требования

- Операционная система: Debian 11 или аналогичная, поддерживающая Docker
- Docker: 20.10+ (open-source community edition)
- Redis (для продакшена): 6.0+ или совместимая СУБД
- Дополнительно (опционально):
    - Prometheus + Grafana для мониторинга метрик
    - ELK Stack или Grafana Loki для централизованного логирования
    - Kubernetes для оркестрации (опционально)

2.1.3. Язык программирования

Сервис разработан на Go 1.24.4.

2.2. Порядок установки

1. Смонтируйте диск с дистрибутивом в папку /mnt
2. Скопируйте из дистрибутива исходники из папки /mnt в папку /root
3. Отредактируйте файл docker-compose.yml, в соответствии с пунктами 3.2 данного
4. документа
5. Смените текущую папку на /root и выполните в ней команду
6. docker compose -up -d --build

3. Настройка Системы

3.1. Общие сведения

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

3.2. Конфигурируемые параметры

Для корректной работы сервиса необходимо настроить следующие переменные окружения:

Переменная	Назначение	Значение по умолчанию	Обязательность
PORT	Порт HTTP-сервера	8080	Нет

Переменная	Назначение	Значение по умолчанию	Обязательность
STORAGE	Тип хранилища (redis/memory)	redis	Нет
REDIS_ADDR	Адрес Redis сервера	127.0.0.1:6379	Да (если STORAGE=redis)
REDIS_PASSWORD	Пароль Redis	(пусто)	Нет
REDIS_DB	Номер базы данных Redis	0	Нет
IDEMPOTENCY_TTL	Время жизни кэшированных ответов	10m	Нет
LOCK_TTL	Время удержания блокировки	30s	Нет
UPSTREAM_TIMEOUT	Таймаут запросов к апстриму	15s	Нет

Формат значений времени:

- 10m - 10 минут
- 30s - 30 секунд
- 1h - 1 час
- 500ms - 500 миллисекунд

3.3. Пример настройки в Docker Compose

```
version: '3.8'

services:
  redis:
    image: redis:7-alpine
    restart: always
    ports:
      - "6379:6379"
    volumes:
      - redis_data:/data
    environment:
      - REDIS_PASSWORD=secretpassword
    command: redis-server --requirepass secretpassword --appendonly yes

  proxy:
    build: .
    restart: always
    ports:
      - "8080:8080"
    environment:
      - PORT=8080
      - STORAGE=redis
      - REDIS_ADDR=redis:6379
      - REDIS_PASSWORD=secretpassword
      - REDIS_DB=0
      - IDEMPOTENCY_TTL=15m
      - LOCK_TTL=45s
      - UPSTREAM_TIMEOUT=20s
    depends_on:
      - redis
```

```
    healthcheck:
      test: ["CMD", "wget", "--no-verbose", "--tries=1", "--spider", "http://localhost:8080/proxy?url=http://example.com"]
      interval: 30s
      timeout: 10s
      retries: 3
      start_period: 40s

volumes:
  redis_data:
```

3.4. Конфигурация для локальной разработки (in-memory)

```
version: '3.8'

services:
  proxy:
    build: .
    restart: always
    ports:
      - "8080:8080"
    environment:
      - PORT=8080
      - STORAGE=memory
      - IDEMPOTENCY_TTL=5m
      - LOCK_TTL=10s
      - UPSTREAM_TIMEOUT=5s
```

4. Описание функциональности

4.1. Общие сведения

Сервис предоставляет HTTP API для проксирования запросов с обеспечением идемпотентности.

- Базовый URL: http://localhost:8080/proxy
- Поддерживаемые методы: POST, PUT, PATCH
- Обязательные заголовки: Idempotency-Key
- Обязательные параметры: ?url={target_url}

4.2. Структура запроса

4.2.1. Проксирование запроса

Метод: POST|PUT|PATCH /proxy

Параметры URL:

- url - целевой URL для проксирования (обязательный)

Заголовки запроса:

- Idempotency-Key - уникальный ключ идемпотентности (обязательный)
- Остальные заголовки (кроме hop-by-hop) передаются на апстрим

Тело запроса: Любое (передается на апстрим без изменений)

Пример запроса:

```
curl -X POST \
  -H "Idempotency-Key: user-123-order-456" \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer token123" \
  -d '{"amount": 100, "currency": "USD"}' \
  "http://localhost:8080/proxy?url=https://api.example.com/payments"
```

4.2.2. Проверка доступности

Метод: OPTIONS /proxy

Назначение: Проверка доступности сервиса

Ответ: 204 No Content

Пример запроса:

```
curl -X OPTIONS http://localhost:8080/proxy
```

4.3. Коды ответов

Код	Описание	Причина
200 OK	Успешный ответ	Ответ из кэша или от апстрима
201 Created	Создан ресурс	Ответ от апстрима
204 No Content	Успешно, без тела	Ответ на OPTIONS
400 Bad Request	Неверный запрос	Отсутствует Idempotency-Key или параметр url
405 Method Not Allowed	Метод не поддерживается	Использован GET, DELETE и т.д.
409 Conflict	Конфликт	Превышено время ожидания блокировки
502 Bad Gateway	Ошибка апстрима	Ошибка сети или апстрима
504 Gateway Timeout	Таймаут	Превышен UPSTREAM_TIMEOUT

4.4. Логирование

Сервис выводит логи в формате:

[proxy] method=POST url=https://api.example.com/payments key=user-123-order-456 status=201 source=upstream duration=245.7ms

Поля лога:

- method - HTTP-метод
- url - целевой URL
- key - Idempotency-Key
- status - HTTP-статус ответа
- source - источник ответа (upstream или await)
- duration - время обработки запроса
- err - ошибка (при наличии)

Приложение А. Примеры использования

А.1. Пример для платежной системы

```
# Создание платежа с идемпотентностью
curl -X POST \
  -H "Idempotency-Key: payment-$(date +%Y%m%d)-12345" \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer payment_token" \
  -d '{
    "amount": 100.50,
    "currency": "USD",
    "recipient": "merchant-123",
    "description": "Purchase #12345"
  }' \
  "http://localhost:8080/proxy?url=https://payments.example.com/api/v1/transactions"
```

А.2. Пример для уведомлений

```
# Отправка уведомления с защитой от дублирования
curl -X POST \
  -H "Idempotency-Key: notification-user-456-email-$(date +%Y%m%d)" \
  -H "Content-Type: application/json" \
  -d '{
    "to": "user@example.com",
    "subject": "Ваш заказ подтвержден",
    "template": "order_confirmation",
    "variables": {"order_id": "12345"}
  }' \
  "http://localhost:8080/proxy?url=https://notifications.example.com/api/send"
```