Установка системы с SSL (https)

1. Для разворачивания контейнеров создаем директории для баз данных и обратного прокси

```
mkdir resources
mkdir traefik
```

2. Переходим в директорию баз данных и создаем директорию для postgres

```
cd resources/
mkdir postgres
```

3. Переходим в директорию postgres, копируем в нее файл docker-compose.yml из предоставленного дистрибутива.

Создаем сеть для контейнеров базами данных и запускаем контейнер с postgres

```
cd postgres/
docker network create resources
docker-compose up -d
```

После запуска, скачается необходимый образ и запустится контейнер postgres.
Логин по умолчанию: postgres
Пароль: postgres
*изменить можно в файле **docker-compose.yml** перед запуском контейнера*

4. Создаем базу данных postgres, например с именем fastboard_back, создать можно разными способами, например подключиться к контейнеру при помощи PgAdmin

Восстанавливаем базу postgres, файл с бэкапом чистой базы **fastboard_back.sql,** который находится в директории 1. Postgres переданного дистрибутива, восстанавливать например через PgAdmin.
В момент запуска конейнера, пройдут миграции и создастся база данных с логином и паролем по умолчанию.
Логин: admin
Пароль: YYYY-MM-DD (дата первого запуска контейнеров)

5. Переходим на директорию выше, т.е. в директорию resources

```
cd ..
```

Создаем директории для Clickhouse переходим в нее

```
mkdir clickhouse
cd  clickhouse/
```

6. Копируем в директорию clickhouse файл docker-compose.yml и директорию с конфигами etc из предоставленного дистрибутива Clickhouse.

Меняем параметры выделенной памяти для контейнера в файле **docker-compose.yml** (по умолчанию выставлено от 2 до 8 Гб ОЗУ) и запускаем контейнер с clickhouse

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

После запуска, скачается необходимый образ и запустится контейнер с clickhouse.
Логин: admin
Пароль: Passw0rd
*изменить можно в файле **docker-compose.yml** перед запуском контейнера*

7. Переходим на директорию выше, т.е. в директорию resources

```
cd ..
```

8. Создаем директории для Redis и переходим в нее

```
mkdir redis
cd  redis/
```

9. Создаем директории для баз и логов redis

```
mkdir -p data/{bases,log}
```

10. Копируем в директорию redis файл docker-compose.yml и директорию с конфигами etc из предоставленного дистрибутива Redis. Запускаем контейнер с Redis

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

После запуска, скачается необходимый образ и запустится контейнер с redis.
Пароль по умолчанию: Passw0rd,
*изменить можно в файле **etc/redis.conf** перед запуском контейнера*

11. Создаем директории для RabbitMQ и переходим в нее

Для доступа к веб-интерфейсу Traefik нужно сгенерировать и подставить пароль в файл docker-compose.yml, для этого выполняем следующие шаги:

```
mkdir rabbitmq
cd rabbitmq/
```

Копируем в директорию RabbitMQ файл **docker-compose.yml** из предоставленного дистрибутива RabbitMQ. Запускаем контейнер с RabbitMQ

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

После запуска, скачается необходимый образ и запустится контейнер с RabbitMQ.
Логин и пароль по умолчанию: fb_rabbit,
*изменить можно в файле **docker-compose.yml** перед запуском контейнера*
Теперь нужно создать VHOST для контейнера бэк:

```
docker exec -it rabbitmq_1 bash
    rabbitmqctl add_vhost fb1
    rabbitmqctl list_vhosts
    rabbitmqctl set_permissions -p "fb1" "fb_rabbit" ".*" ".*" ".*"
```

12. Переходим в директорию traefik и копируем в нее файл docker-compose.yml из из предоставленного дистрибутива Traefik

Для доступа к веб-интерфейсу Traefik нужно сгенерировать и подставить пароль в файл docker-compose.yml, для этого выполняем следующие шаги:

```
cd ../../traefik
apt install apache2-utils
```

> **ВАЖНО!** Мы должны экранировать каждый символ “$” в нашем зашифрованном пароле (заменить $ на $$), так как мы используем пароль напрямую в docker-compose.yml

```
echo $(htpasswd -nb admin Passw0rd) | sed -e s/\\$/\\$\\$/g
```

Заменить Passw0rd на свой пароль.

Пример вывода команды (результат будет разный при каждом запуске команды):

```
admin:$$2y$$05$$iSGcI0SpukDoOZolGkfghIFe31e47F5vewcjlhzhgf0EHo45H.dFyKW
```

Вывод команды нужно поместить в наш docker-compose.yml внутрь traefik метки, заменив <USER-PASSWORD-OUTPUT> в примере ниже.

```
 "traefik.http.middlewares.auth.basicauth.users=<USER-PASSWORD-OUTPUT>"
```

После создания и установки пользователя и пароля, создаем сеть докер для Traefik и запускаем контейнер с Traefik

```
docker network create traefik
docker-compose up -d
```

После запуска, скачается необходимый образ и запустится контейнер с Traefik.

> **ВАЖНО! Для разворачивания проекта необходимо заранее иметь две DNS-записи на IP сервера, где разворачивается проект. Записи нужны для получения сертификатов от Let's Encrypt и шифрования трафика Они включаются в сборку программы для развертывания на конкретном сервере.**

Например:
front.example.com
back.example.com

13. Создаем директорию для FastBoard, переходим в нее, копируем docker-compose.yml и образы docker в рабочую директорию проекта

Создаем директорию для FastBoard, переходим в нее, копируем **docker-compose.yml** и образы docker в рабочую директорию проекта

```
cd ..
mkdir fastboard
cd fastboard
```

Проверяем и устанавливаем данные для авторизации в базах данных и параметры подключения к контейнерам с базами в файле **docker-compose.yml**, в данный момент там установлены данные для подключения по умолчанию.

Далее импортируем образы контейнеров **fastboard:back** и **fastboard:front**, для этого выполняем команды:

```
docker load -i fastboard-back.tar
docker load -i fastboard-front.tar
```

После загрузки образов, запускаем контейнеры бэк и фронт, а также проверяем логи из запуска

```
docker-compose up -d
docker-compose logs -f
```

или

```
docker-compose up -d
docker-compose ps
docker logs -f container_name
```

Если контейнеры запустились без проблем, то проверяем работу пройдя по ссылке указанной в **docker-compose.yml** для контейнера фронт: https://example.com

14. Активация первого ключа

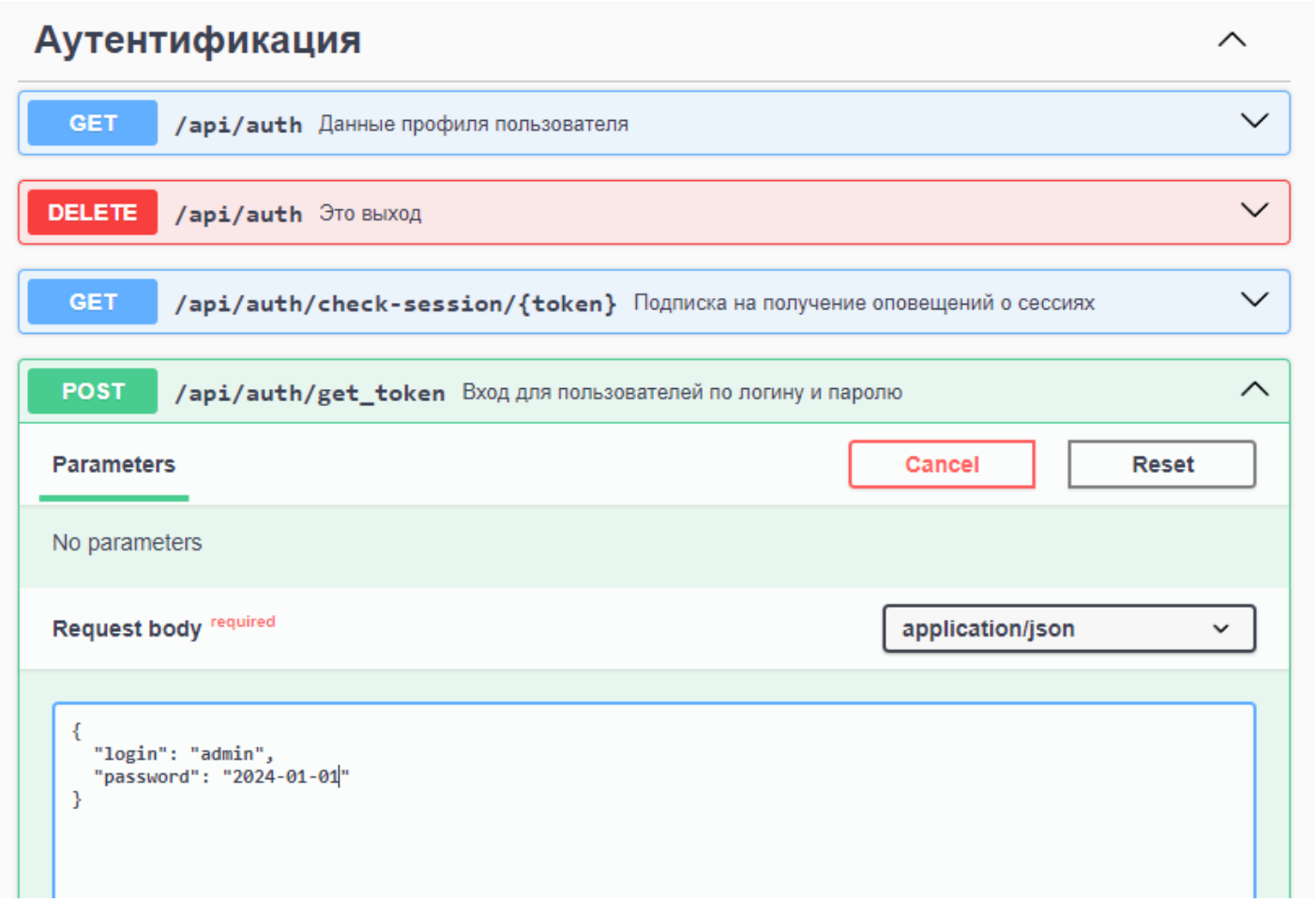
После успешного развертывания нужно войти в API системы под техническим пользователем для активации первого лицензионного ключа. Этот пользователь имеет права администратора и остается в системе.

Входим в API

Переходим по адресу: **ваш_бэкенд/docs/swagger,** находим блок аутентификации и метод **get_token.**
Жмем кнопку **TRY OUT**, вводим учетные данные:

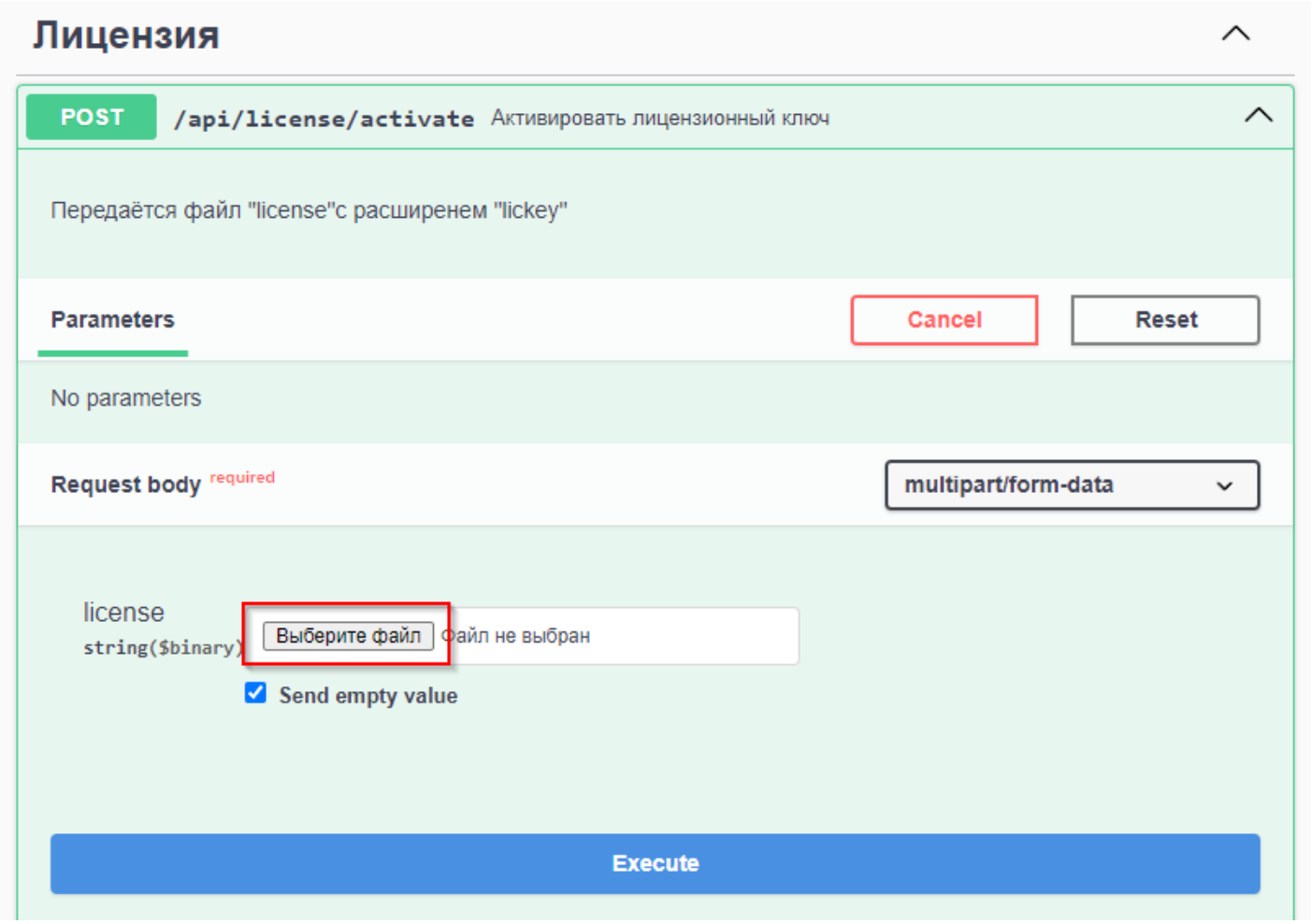
Логин: admin
Пароль: дата первого запуска контейнеров ГГГГ-ММ-ДД

Далее нажимаем кнопку **EXECUTE**



Далее активируем лицензионный ключ

В блоке «Лицензия» находим метод Acivate, жмем кнопку **TRY OUT**, выбираем файл лицензии и нажимаем кнопку **EXECUTE**



Система готова к работе

Теперь можно войти спомощью интерфейса и создать пользователей через панель администратора и выдавать им лицензии

Revision #17
Created 14 November 2023 13:14:46 by Станислав
Updated 12 September 2024 08:00:01 by Станислав