

Системная информация

- [Системные требования](#)
- [Технические рекомендации](#)
- [Установка системы с SSL \(https\)](#)
- [Установка системы без SSL \(http\)](#)
- [Обновление системы](#)

Системные требования

Операционная система

Linux Ubuntu 22.04 или выше; Debian 12 или выше; Windows 7 или выше

Процессор

Рекомендуемое кол-во ядер: 16 или больше

Оперативная память

Рекомендуемый объем: 64 Gb или больше

Дисковое пространство

Тип диска: SSD

Рекомендуемый объем: 300 Gb

Технологический стэк

Docker version 20.10.17 или выше,

Docker-compose version 1.25.4 или выше

Базы данных

PostgreSQL 16

Clickhouse v23.8.2.7

Технические рекомендации

Процессор

Fastboard использует для обработки данных СУБД российской разработки Clickhouse. ClickHouse реализует параллельную обработку данных и использует все доступные аппаратные ресурсы. При выборе процессора учитывайте, что ClickHouse работает более эффективно в конфигурациях с большим количеством ядер, но с более низкой тактовой частотой, чем в конфигурациях с меньшим количеством ядер и более высокой тактовой частотой. Например, 16 ядер с 2600 MHz предпочтительнее, чем 8 ядер с 3600 MHz.

Рекомендуется использовать технологии Turbo Boost и hyper-threading. Их использование существенно улучшает производительность при типичной нагрузке.

RAM

Мы рекомендуем использовать как минимум 16 ГБ оперативной памяти, чтобы иметь возможность выполнять нетривиальные запросы. Сервер ClickHouse может работать с гораздо меньшим объёмом RAM, память требуется для обработки запросов.

Необходимый объём RAM зависит от:

- Сложности запросов.
- Объёма данных, обрабатываемых в запросах.

Для расчета объёма RAM необходимо оценить размер промежуточных данных для операций GROUP BY, DISTINCT, JOIN а также других операций, которыми вы пользуетесь.

ClickHouse может использовать внешнюю память для промежуточных данных.

Дисковое пространство

Для установки ClickHouse необходимо 2ГБ свободного места на диске.

Объём дискового пространства, необходимый для хранения ваших данных, необходимо рассчитывать отдельно. Расчёт должен включать:

- Приблизительную оценку объёма данных. Можно взять образец данных и получить из него средний размер строки. Затем умножьте полученное значение на количество строк, которое вы планируете хранить.
- Оценку коэффициента сжатия данных. Чтобы оценить коэффициент сжатия данных, загрузите некоторую выборку данных в ClickHouse и сравните действительный размер данных с размером сохранённой таблицы. Например, данные типа clickstream обычно сжимаются в 6-10 раз.

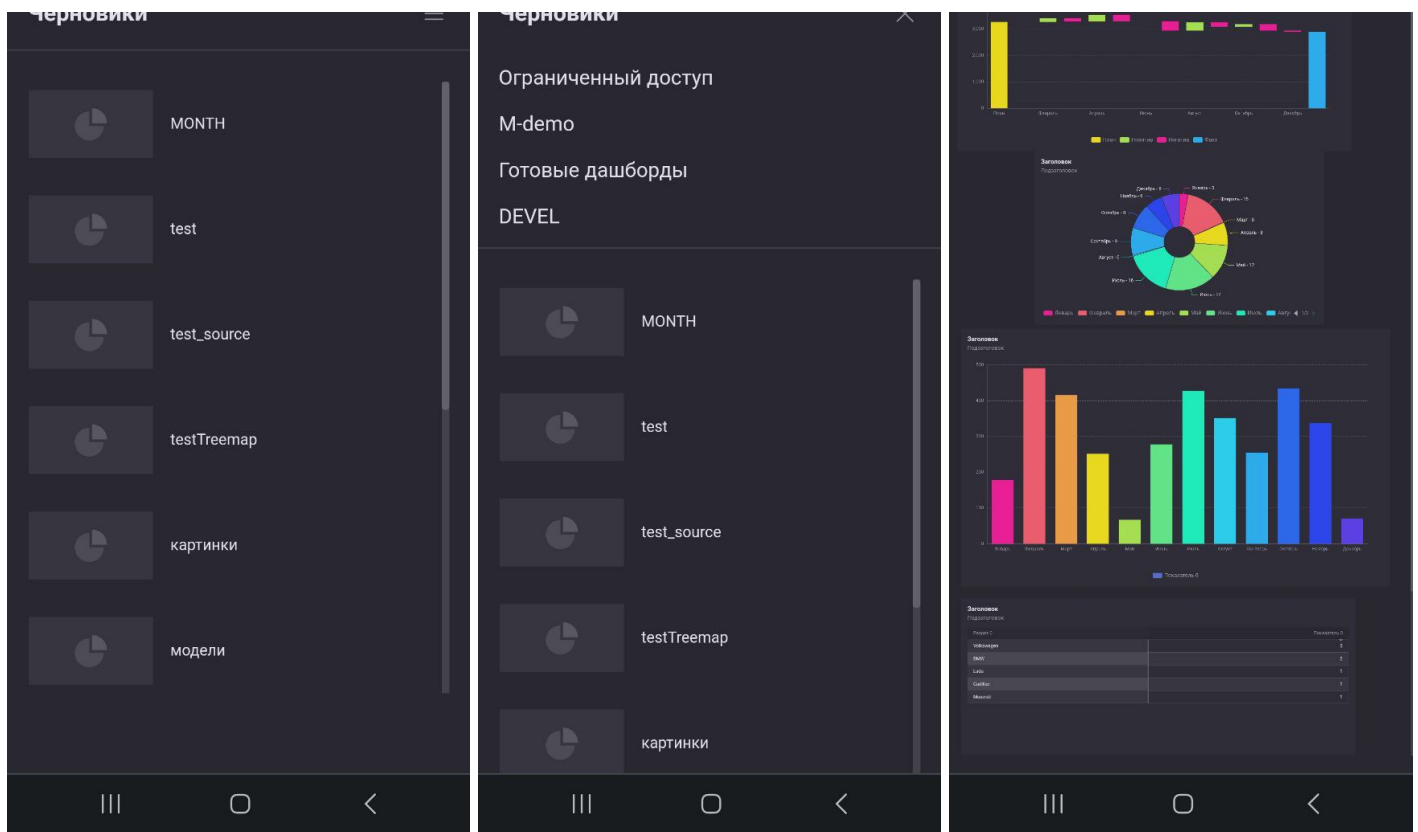
Для оценки объёма хранилища, примените коэффициент сжатия к размеру данных. Если вы планируете хранить данные в нескольких репликах, то необходимо полученный объём умножить на количество реплик.

Файл подкачки

Отключайте файл подкачки в продуктовых средах.

Поддержка мобильных устройств

Интерфейс Fastboard спроектирован с учетом поддержки работы на мобильных устройствах. Используемые элементы управления позволяют выбирать и просматривать существующие группы, проекты, страницы дашбордов.



Сеть

По возможности, используйте сети 10G и более высокого класса.

Пропускная способность сети критически важна для обработки распределенных запросов с большим количеством промежуточных данных. Также, скорость сети влияет на задержки в процессах репликации.

Продукт может быть развернут внутри закрытого контура без доступа к интернету.

Распределение нагрузки

Системное устройство Fastboard состоит из следующих компонентов:

- **5 сервисов:**

1. HTTP-сервис
2. Планировщик заданий (CRON)
3. Загрузчик данных из внешних источников
4. Блок трансформации файлов
5. Сервис трансформации данных для визуализаций

- **Сервисное хранилище PostgreSQL для настроек проектов и системных параметров**
- **Хранилище и расчеты данных для визуализаций ClickHouse**
- **Для обмена сообщений между сервисами и кеширование данных используется REDIS**
- **Менеджер процессов PM2 обеспечивает балансировку нагрузки между сервисами в рамках одного сервера**

Каждый из этих компонентов может быть запущен на разных серверах для оптимизации нагрузки

На одном сервере может быть запущено несколько экземпляров одного или нескольких компонентов

В мультисерверных решениях для работы системы можно использовать любые системы кластеризации и балансировщики нагрузки

Данная архитектура обеспечивает следующие преимущества:

Каждый узел (компонент/сервис) выполняет свою задачу. Например, расчет данных, в том числе многопоточный, выполняет ClickHouse, а преобразование данных в формат для визуализаций выполняет сервис трансформации. Таким образом достигается оптимальное распределение нагрузки между всеми компонентами системы.

Менеджер процессов PM2 обеспечивает балансировку нагрузки выделенных для системы ресурсов. Также система готова к работе с любой существующей системой управления кластерами. Установка максимально допустимых параметров потребления происходит на уровне операционной системы или выбранной пользователем системы кластеризации.

Установка системы с SSL (https)

1. Для разворачивания контейнеров создаем директорию 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 контейнера, 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 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 разным при каждом запуске команды):

```
admin:$2y$05$$iSGcl0SpukDoOZolGkfghlFe31e47F5vewcjlhzhgf0EHo45H.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**

Аутентификация

**GET****/api/auth** Данные профиля пользователя**DELETE****/api/auth** Это выход**GET****/api/auth/check-session/{token}** Подписка на получение оповещений о сессиях**POST****/api/auth/get_token** Вход для пользователей по логину и паролю**Parameters****Cancel****Reset**

No parameters

Request body required

application/json



```
{
  "login": "admin",
  "password": "2024-01-01"
}
```

Далее активируем лицензионный ключ

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

Лицензия



POST

/api/license/activate Активировать лицензионный ключ



Передаётся файл "license" с расширением "lickey"

Parameters

Cancel

Reset

No parameters

Request body required

multipart/form-data



license

string(\$binary)

Выберите файл

Файл не выбран

☒ Send empty value

Execute

Система готова к работе

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

Установка системы без SSL (http)

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

```
mkdir resources
```

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

С именем fastboard_back. Создать можно разными способами, например подключиться к контейнеру при помощи PgAdmin или psql-16

```
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" >
/etc/apt/sources.list.d/pgdg.list'

curl -fsSL https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo gpg --dearmor -o
/etc/apt/trusted.gpg.d/postgresql.gpg

apt update
apt install postgresql-client-16
psql -h 127.0.0.1 -U postgres
\l
CREATE DATABASE fastboard_back;
\l
\q
```

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

```
cd ..
```

Создаем директорию для Clickhouse переходим в нее

```
mkdir clickhouse
cd clickhouse/
```

6. Копируем в директорию clickhouse файл docker-compose.yml и директории с конфигами (etc, custom_config) 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 **etc/users.xml** 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. Создаем директорию для FastBoard,

Переходим в нее, копируем **`docker-compose.yml`**, **`license_rsa.pub`** и образы `docker` из архива **`fastboard.tar.gz`** 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
```

В момент запуска контейнера с бэкенд приложения, пройдут миграции и создастся база данных с логином и паролем по умолчанию. Логин admin пароль YYYY-MM-DD (дата первого запуска контейнеров)

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

13. Первый вход в систему и пользователи

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**

Аутентификация

**GET****/api/auth** Данные профиля пользователя**DELETE****/api/auth** Это выход**GET****/api/auth/check-session/{token}** Подписка на получение оповещений о сессиях**POST****/api/auth/get_token** Вход для пользователей по логину и паролю**Parameters****Cancel****Reset**

No parameters

Request body required

application/json



```
{
  "login": "admin",
  "password": "2024-01-01"
}
```

Далее активируем лицензионный ключ

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

Лицензия

POST

/api/license/activate Активировать лицензионный ключ

Передаётся файл "license" с расширением "lickey"

Parameters

Cancel

Reset

No parameters

Request body required

multipart/form-data

license

string(\$binary)

Выберите файл

Файл не выбран

☒ Send empty value

Execute

Система готова к работе

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

Обновление системы

1. Скачиваем архив с новой сборкой по ссылке (предоставляется отдельно)

2. Распаковываем

3. Заливаем на сервер новые образы докер

4. На сервере из директории с образами выполняем:

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

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

5. На сервере из деректории фастборда выполняем

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