

# VisionLabs FaceStream

Руководство по обновлению с версии v.5.1.13 на v.5.1.14

v.5.1.14

Глоссарий	4
1 Введение	6
2 Аппаратные требования	7
2.1 Минимальные аппаратные требования	7
3 Программные требования	8
Общая информация	9
4 Подготовка к обновлению	10
4.1 Создание резервной копии	10
4.2 Копирование настроек LUNA Configurator	10
4.3 Подготовка файла с настройками сервисов для миграции	10
4.4 Удаление символической ссылки	11
4.5 Распаковка архива	11
4.6 Создание символической ссылки	12
4.7 Перенос данных	12
4.7.1 Копирование данных PostgreSQL	12
4.7.2 Копирование данных InfluxDB	12
4.8 Удаление старых контейнеров	13
4.9 Включение записи логов в файл для FaceStream	13
4.10 Установка зависимостей для GPU	14
4.10.1 Действия для запуска FaceStream с GPU через Docker Compose	15
4.11 Активация лицензионного ключа	16
4.11.1 Активация ключа при запущенной LP	16
4.11.2 Активация ключа без запущенной LP	16
4.11.2.1 Установка утилиты HASP	17
4.11.2.2 Конфигурация утилиты HASP	18
4.11.2.3 Добавление библиотеки LP	18
4.11.2.4 Создание отпечатка системы для LUNA PLATFORM	19
4.11.2.5 Добавление файла с лицензией вручную с помощью пользовательского интерфейса	19
4.11.2.6 Задание адреса лицензированного сервера	20
4.11.2.7 Удаление утилиты LP HASP	21
4.12 Обновление и запуск сервисов LP	21
4.12.1 Запуск контейнера InfluxDB OSS 2	21
4.12.2 Запуск контейнера PostgreSQL	22

4.12.3	Миграция базы данных LUNA Configurator . . . . .	22
4.12.3.1	Инициализация базы данных LUNA Configurator . . . . .	22
4.12.4	Запуск контейнера LUNA Configurator . . . . .	23
4.12.5	Запуск контейнера LUNA Licenses . . . . .	24
4.12.5.1	Указание адреса сервиса LUNA Licenses . . . . .	24
<b>5</b>	<b>Обновление FaceStream</b>	<b>25</b>
5.1	Миграция настроек . . . . .	25
5.2	Миграция базы данных LUNA Streams . . . . .	25
5.3	Команда запуска контейнера LUNA Streams . . . . .	25
5.4	Команды запуска контейнера FaceStream . . . . .	26
5.4.1	Команда запуска контейнера с использованием CPU . . . . .	26
5.4.2	Команда запуска контейнера с использованием GPU . . . . .	27
5.4.3	Ключи запуска . . . . .	27
5.4.4	Расшифровка параметров запуска контейнера . . . . .	29
<b>6</b>	<b>Команды Docker</b>	<b>31</b>
6.1	Показать контейнеры . . . . .	31
6.2	Копировать файлы в контейнер . . . . .	31
6.3	Вход в контейнер . . . . .	31
6.4	Имена образов . . . . .	31
6.5	Просмотр логов контейнера . . . . .	31
6.6	Удаление образа . . . . .	32
6.7	Остановка контейнера . . . . .	32
6.8	Удаление контейнера . . . . .	32

## Глоссарий

Термин	Значение термина
Биометрический образец	Изображения, содержащие лицо или тело и соответствующие стандарту VisionLabs. Используется при работе с LUNA PLATFORM.
Биометрический шаблон	Набор уникальных свойств, получаемых в LUNA PLATFORM из биометрического образца.
Детекция	Сущность FaceStream, содержащая координаты лица или тела и оценочное значение объекта, по которому определяется лучший кадр.
Лучший кадр	Лучший кадр выбирается из всех кадров одного трека. Основными условиями выбора лучшего кадра являются приемлемое качество изображения и наличие на нём лица или тела с наилучшим ракурсом. Условия выбора лучшего кадра задаются в настройках FaceStream.
Портрет	Изображение лица или тела, трансформированное под определенный формат. Портрет имеет два типа - «warp» (изображение трансформируется в формат биометрического образца), «gost» (из исходного кадра вырезается детекция с учетом отступов).
Ракурс	Степень поворота головы (в градусах) по каждой из трех осей вращения (наклон вверх/вниз относительно горизонтальной оси; наклон влево/вправо относительно вертикальной оси; поворот относительно вертикальной оси).
Событие	Сущность LUNA PLATFORM, которая содержит информацию (город, пользовательские данные, номер трека и т.д.) об одном лице и/или теле. Данная информация передается в LUNA PLATFORM приложением FaceStream. Полный перечень передаваемой информации см. в документации OpenAPI LUNA PLATFORM.
Трек	Информация о положении объекта (лица) одного человека на последовательности кадров. Если объект покидает зону кадра, то трек прерывается не сразу. Некоторое время он ожидает возвращения объекта в кадр. Если объект вернулся, то трек продолжается.

Термин	Значение термина
Трекинг	Функция отслеживания объекта (лица) на последовательности кадров.

## 1 Введение

В данном документе приводится пример шагов, необходимых для обновления на новую сборку FaceStream. Сборка FaceStream включает в себя сам FaceStream и сервис LUNA Streams. Поскольку для запуска FaceStream требуется наличие сервисов InfluxDB, PostgreSQL, LUNA Configurator и LUNA Licenses, то в данном руководстве приводятся команды для сохранения данных этих сервисов.

Данное руководство написано с предположением, что:

- предыдущая версия сборки FaceStream уже установлена, и требуемое окружение на сервере готово к работе.
- FaceStream установлен в соответствии с руководством по установке, и используются пути по умолчанию. В противном случае следует внести изменения вручную в процессе обновления.

В данном документе приведены примеры разворачивания сборки FaceStream в минимальной рабочей конфигурации для использования в демонстрационных целях. Данная конфигурация не является достаточной для реальной эксплуатации системы в продуктивном контуре.

Все описываемые команды необходимо исполнять в оболочке Bash (когда команды запускаются напрямую на сервере) или Putty (в случае удаленного подключения к серверу). Описываемые команды тестировались только с помощью этих средств. Использование других оболочек или эмуляторов может привести к ошибкам при выполнении команд.

## 2 Аппаратные требования

### 2.1 Минимальные аппаратные требования

Дальнейшие минимальные требования приведены для использования одного экземпляра FaceStream.

Для корректной работы приложения аппаратное обеспечение должно отвечать следующим минимальным требованиям:

- CPU с частотой 2 ГГц и выше;
- 4 Гб оперативной памяти и выше;
- 10 Гб свободного места на жестком диске.
- Доступ к Интернету (для контейнеров и дополнительных загрузок ПО).

На аппаратные требования влияют несколько факторов:

- Количество обрабатываемых потоков;
- Частота и разрешение кадров потоков;
- Параметры настройки FaceStream. Настройки по умолчанию являются наиболее универсальными. В зависимости от условий эксплуатации приложения с помощью их значений можно повлиять на качество или производительность.

Следует подбирать аппаратное обеспечение на основе вышеперечисленных факторов.

FaceStream также может работать в режиме ускорения вычислений за счет:

#### Использования ресурсов видеокарты

Вычисления с использованием видеокарты поддерживаются только для детектора FaceDetV3. См. параметр «defaultDetectorType» в настройках FaceEngine («faceengine.conf»).

Требуется минимум 6Гб оперативной или выделенной видеопамяти. Рекомендуется 8 Гб VRAM или более.

Поддерживаются архитектуры Pascal, Volta, Turing. Требуется Compute Capability 6.1 или выше и CUDA версии 11.4.

Рекомендуемый драйвер NVIDIA - 470.103.01.

В данный момент для одного экземпляра FaceStream поддерживается только одна видеокарта.

#### Использования AVX2 инструкций

Требуется CPU с поддержкой AVX2. Система автоматически определяет наличие инструкций и запускается в оптимальном режиме.

### 3 Программные требования

Запуск контейнеров FaceStream и LUNA Streams был протестирован на:

- CentOS Linux release 7.8.2003 (Core)

В контейнере FaceStream используется для запуска следующая ОС:

- CentOS Linux release 8.3.2011

Для запуска контейнеров должен быть установлен Docker. Для загрузки настроек в сервис LUNA Configurator требуется наличие Python версии 2.x или 3.x.



## Общая информация

Рекомендуется внимательно изучить данный документ. Документ даёт общее представление о том, из каких компонентов состоит FaceStream и какие задачи они решают.

Развертывание следует выполнять в порядке, указанном в данном документе.

Все процедуры в данном руководстве описаны для CentOS. Если требуется развернуть Docker контейнеры на любой другой ОС, следует обратиться к документации по Docker Compose:

<https://docs.docker.com/compose/install/>

Для работы FaceStream требуются компоненты LUNA PLATFORM, дополнительные базы данных и сервис LUNA Streams. Основная информация об этом ПО содержится в данном документе.

LUNA Streams не является компонентом LUNA PLATFORM.

Следующие компоненты LUNA PLATFORM используются по умолчанию с FaceStream.

- LUNA Licenses используется для лицензирования сервиса LUNA Streams.
- LUNA Configurator используется быстрого доступа к основным настройкам FaceStream и настройкам сервисов LUNA PLATFORM.
- PostgreSQL используется в качестве базы данных по умолчанию для сервиса LUNA Streams. Также возможно использование базы данных Oracle вместо PostgreSQL.
- Для мониторинга сервисов LUNA PLATFORM используется InfluxDB. При необходимости мониторинг можно отключить.

Следующие версии баз данных рекомендованы к использованию с LUNA Streams:

- PostgreSQL: 12
- Oracle: 11.2

Установка и конфигурация Oracle не описывается в данном руководстве. Далее в документе будут приводиться примеры запуска с использованием PostgreSQL.

Balancers и другие программы могут использоваться при масштабировании системы для обеспечения отказоустойчивости. Их конфигурация не описывается в данном руководстве.

## 4 Подготовка к обновлению

Перед обновлением FaceStream необходимо выполнить ряд дополнительных действий.

### 4.1 Создание резервной копии

Создайте резервную копию БД LUNA Streams перед выполнением миграции. Данные можно восстановить в случае возникновения каких-либо проблем в процессе миграции.

Создание резервной копии базы данных не описано в данном документе.

### 4.2 Копирование настроек LUNA Configurator

Для сохранения настроек Configurator, используемых в прошлой версии FaceStream следует создать резервную копию файла конфигурации `/var/lib/fs/fs-current/extras/conf/configurator_configs/luna_configurator_postgres.conf` в отдельной директории на сервере.

```
cp /var/lib/fs/fs-current/extras/conf/configurator_configs/  
luna_configurator_postgres.conf /var/lib/fs/  
BACKUP_luna_configurator_postgres.conf
```

### 4.3 Подготовка файла с настройками сервисов для миграции

Если предыдущая версия FaceStream использовалась с настройками других сервисов, отличных от настроек по умолчанию, сделайте резервную копию файла с настройками сервисов в отдельной директории на сервере вне контейнера сервиса Configurator.

Чтобы создать файл настроек, используйте следующие опции (можно выполнить из любой директории на сервере):

```
wget -O /var/lib/fs/settings_dump_backup.json 127.0.0.1:5070/1/dump
```

или

```
curl 127.0.0.1:5070/1/dump > /var/lib/fs/settings_dump_backup.json
```

Эту копию не требуется использовать во время обновления. Она нужна только для ручного восстановления настроек в случае непредвиденных проблем.

#### 4.4 Удаление символической ссылки

Удалите символическую ссылку в директорию предыдущей минорной версии с помощью следующей команды:

```
rm -f /var/lib/fs/fs-current
```

#### 4.5 Распаковка архива

Рекомендуется переместить архив в предварительно созданную директорию для FaceStream и распаковать архив в этой директории.

Указанные команды следует выполнять под пользователем root.

Создайте директорию для FaceStream.

```
mkdir -p /var/lib/fs
```

Переместите архив в созданную директорию. Предполагается, что архив сохранён на сервере в директории «/root».

```
mv /root/facestream_docker_v.5.1.14.zip /var/lib/fs/
```

Перейдите в директорию.

```
cd /var/lib/fs/
```

Установите утилиту unzip, если она ещё не установлена.

```
yum install unzip
```

Распакуйте архив.

```
unzip facestream_docker_v.5.1.14.zip
```

Перед запуском FaceStream потребуется выполнить его настройку.

В распакованном архиве находятся конфигурационные файлы, которые необходимы для запуска FaceStream. Описание параметров из данных файлов приведены далее в документе.

## 4.6 Создание символической ссылки

Создайте символическую ссылку. Символическая ссылка указывает на директорию, в которой хранятся файлы для запуска нужной версии программного продукта.

```
ln -s facestream_docker_v.5.1.14 fs-current
```

## 4.7 Перенос данных

Переместите данные для базы данных LUNA Streams в директорию с новым дистрибутивом.

Считается, что для хранения базы данных используются пути по умолчанию.

В приведенном примере происходит обновление с версии v.5.1.13. Следует выполнить эти действия для сборки FaceStream, установленной на вашем сервере. Измените v.5.1.13 в командах ниже на текущую сборку.

Следует скопировать директорию с данными вашей базы данных из директории «facestream\_docker\_v.5.1.13» в текущий корневой каталог для того, чтобы использовать эти данные в новой сборке FaceStream.

### 4.7.1 Копирование данных PostgreSQL

Следующий шаг необходим если используется PostgreSQL в Docker контейнере.

Скопируйте директорию «data»:

```
cp -r /var/lib/fs/facestream_docker_v.5.1.13/example-docker/postgresql /var/lib/fs/fs-current/example-docker/
```

### 4.7.2 Копирование данных InfluxDB

Следующий шаг необходим если используется InfluxDB в Docker контейнере.

Скопируйте директорию «influx» со всеми бакетами:

```
cp -r /var/lib/fs/facestream_docker_v.5.1.13/example-docker/influx /var/lib/fs/fs-current/example-docker/
```

## 4.8 Удаление старых контейнеров

Перед запуском контейнеров текущей минорной версии необходимо остановить все контейнеры, относящиеся к предыдущей минорной версии FaceStream, а также контейнеры сторонних приложений.

Контейнеры PostgreSQL и InfluxDB также можно удалить, т.к. их версии будут обновлены в новой сборке.

PostgreSQL и InfluxDB требуют перезагрузки даже если их контейнеры не изменялись. Это связано с переносом данных из их директорий. См. [«Перенос данных»](#).

Для удаления контейнера используйте следующую команду:

```
docker container rm -f [container_name]
```

где [container\_name] это имя контейнера сервиса docker или ID.

Например, для удаления контейнеров FaceStream, LUNA Streams, LUNA Configurator и LUNA Licenses используйте следующую команду:

```
docker container rm -f facestream luna-streams luna-configurator luna-licenses
```

Чтобы посмотреть имена контейнеров или их ID, используйте следующую команду:

```
docker ps -a
```

Также рекомендуется удалить старые образы контейнеров для освобождения места. Можно использовать следующую команду для удаления всех неиспользуемых образов.

Если на сервере достаточно места, рекомендуется выполнить это действие только после успешного запуска новой версии FaceStream.

Данная команда удаляет все неиспользуемые образы, а не только образы, относящиеся к FaceStream.

```
docker image prune -a -f
```

## 4.9 Включение записи логов в файл для FaceStream

**Примечание.** Используйте нижеописанные действия только если необходимо сохранять логи в файл. По умолчанию логи выводятся в консоль. Для просмотра логов из консоли используйте команду `docker logs <container_name>`.

При необходимости можно записывать логи FaceStream в отдельные файлы. Для этого необходимо предварительно создать директорию для сохранения логов.

Директория для логов создается с помощью следующей команды:

```
mkdir -p /var/lib/fs/fs-current/logs/
```

Директорию для хранения логов можно изменить при желании.

Для включения записи логов необходимо включить логирование в файл в настройках FaceStream. Для этого нужно выставить значение параметра «mode» на «l2f» (выводить логи только в файл) или «l2b» (выводить логи и файл и в консоль).

Для активации записи логов нужно выполнить следующую команду при запуске контейнера:

```
-v /var/lib/fs/fs-current/logs:/srv/logs/ \
```

Данные из указанной директории добавляются в Docker контейнер, когда он запущен. Все данные из указанной директории Docker контейнера сохраняются в данную директорию.

Для записи логов сервисов LUNA необходимо выполнить аналогичные действия.

По умолчанию в логах FaceStream выводятся только предупреждения системы. С помощью настройки параметра «severity» можно включить вывод ошибок (см. описание параметра в руководстве администратора).

## 4.10 Установка зависимостей для GPU

**Пропустите данный раздел если не собираетесь использовать FaceStream с GPU.**

Для использования GPU с Docker контейнерами необходимо установить NVIDIA Container Toolkit.

Пример установки приведен ниже.

```
distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
```

```
curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-  
docker.repo | tee /etc/yum.repos.d/nvidia-docker.repo
```

```
yum install -y nvidia-container-toolkit
```

```
systemctl restart docker
```

Проверьте работу NVIDIA Container toolkit, запустив базовый контейнер CUDA (он не входит в дистрибутив FaceStream, его необходимо загрузить из Интернета):

```
docker run --rm --gpus all nvidia/cuda:11.4-base nvidia-smi
```

Для дополнительной информации см. следующую документацию:

<https://github.com/NVIDIA/nvidia-docker#centos-7x8x-docker-ce-rhel-7x8x-docker-ce-amazon-linux-12>.

Извлечение атрибутов на GPU разработано для максимальной пропускной способности. Выполняется пакетная обработка входящих изображений. Это снижает затраты на вычисления для изображения, но не обеспечивает минимальную задержку для каждого изображения.

GPU-ускорение разработано для приложений с высокой нагрузкой, где количество запросов в секунду достигает тысяч. Нецелесообразно использовать ускорение GPU в сценариях с небольшой нагрузкой, когда задержка начала обработки имеет значение.

#### 4.10.1 Действия для запуска FaceStream с GPU через Docker Compose

Для запуска FaceStream с GPU через Docker Compose необходимо, кроме вышеописанных действий, добавить секцию `deploy` в поле `handlers` в файл `docker-compose.yml`.

```
vi /var/lib/fs/fs-current/example-docker/docker-compose.yml
```

```
facestream:
  image: ${DOCKER_REGISTRY_TEST}:${DOCKER_REGISTRY_PORT}/facestream:${FACESTREAM_TAG}
  deploy:
    resources:
      reservations:
        devices:
          - driver: nvidia
            count: all
            capabilities: [gpu]
  restart: always
  environment:
    CONFIGURATOR_HOST: ${HOST_CONFIGURATOR}
    CONFIGURATOR_PORT: 5070
```

driver - в данном поле указывается драйвер для зарезервированного устройства(устройств);

count - в данном поле задается количество графических процессоров, которые должны быть зарезервированы (при условии, что хост содержит такое количество графических процессоров);

capabilities - данное поле выражает как общие, так и специфические возможности драйвера. Его необходимо задать, иначе будет возвращена ошибка при развертывании сервиса.

Для дополнительной информации см. следующую документацию:

<https://docs.docker.com/compose/gpu-support/#enabling-gpu-access-to-service-containers>.

## 4.11 Активация лицензионного ключа

В большинстве случаев нет необходимости получать отдельную лицензию при обновлении минорных версий. Единственный случай, когда это может понадобиться - при обновлении лицензии. Например, изменилась библиотека LP.

Если лицензия LUNA PLATFORM 5 не изменилась, то пропустите данный шаг.

### 4.11.1 Активация ключа при запущенной LP

Если сервисы LUNA PLATFORM уже запущены и лицензия с параметром, регулирующим количество потоков для работы LUNA Streams, уже активирована, то необходимо убедиться в том, что текущий ключ LUNA PLATFORM содержит данный параметр. Информация может быть предоставлена специалистами VisionLabs.

Если данный параметр не содержится в ключе, то необходимо запросить новый ключ и обратиться к специалистам VisionLabs для консультации по обновлению лицензионного ключа.

Если LUNA Streams запускается на сервере, отличном от того, на котором запущен LUNA Licenses, то необходимо выполнить действия, описанные в разделе ["Указание адреса сервиса LUNA Licenses"](#).

### 4.11.2 Активация ключа без запущенной LP

Если сервисы LUNA PLATFORM не запущены или запуск осуществляется с помощью Docker Compose, то подразумевается, что лицензия ещё не была активирована и необходимо запросить новую лицензию LUNA PLATFORM 5 с параметром, регулирующим количество потоков для работы LUNA Streams, и пройти полный процесс лицензирования LUNA PLATFORM 5.

Для лицензирования LP и используется сервис HASP. Без лицензии невозможно запускать и использовать сервисы LUNA и создавать потоки для FaceStream.

Существует ключ HASP, который позволяет пользователям работать в LUNA PLATFORM и создавать потоки в FaceStream. Он использует библиотеку haspvlib\_x86\_64\_30147.so.



Библиотеки находятся в директории «/var/hasplm/».

Лицензионные ключи предоставляются компанией VisionLabs по запросу отдельно от поставки. Количество одновременно обрабатываемых потоков указано в лицензионном ключе LUNA PLATFORM.

Для использования LUNA PLATFORM и FaceStream в Docker контейнерах требуется сетевая лицензия.

Лицензионный ключ создается с помощью отпечатка системы. Этот отпечаток создается на базе информации об аппаратных характеристиках сервера. Таким образом, полученный лицензионный ключ будет работать только на том же сервере, с которого был получен отпечаток системы.

Существует вероятность, что потребуется новый лицензионный ключ при внесении каких-либо изменений на сервере лицензии.

Последовательность действий:

- Установите на сервер утилиту HASP. Обычно утилита HASP устанавливается на отдельный сервер;
- Запустите утилиту HASP;
- Создайте отпечаток системы для вашего сервера и отправьте его в VisionLabs;
- Активируйте свой ключ, полученный от VisionLabs;
- Укажите адрес вашего сервера HASP в специальном файле.

Вкладка Sentinel Keys пользовательского интерфейса ( <server\_host\_address>:1947) отображает активированные ключи.

LP использует утилиту HASP определённой версии. Если на сервере установлена более старая версия утилиты, её следует удалить перед установкой новой версии. См. раздел «Удаление утилиты LP HASP».

#### 4.11.2.1 Установка утилиты HASP

Откройте директорию HASP.

```
cd /var/lib/fs/fs-current/extras/hasp/
```

Установите утилиту HASP на сервер.

```
yum -y install /var/lib/fs/fs-current/extras/hasp/aksusbd-*.rpm
```

Запустите утилиту HASP.

```
systemctl daemon-reload
```

```
systemctl start aksusbd
```

```
systemctl enable aksusbd
```

```
systemctl status aksusbd
```

#### 4.11.2.2 Конфигурация утилиты HASP

Осуществить конфигурацию утилиты HASP можно с помощью файла «/etc/hasplm/hasplm.ini».

**Примечание!** Не выполняйте это действие, если INI файл для утилиты HASP уже сконфигурирован.

Удалите старый файл настроек, если необходимо.

```
rm -rf /etc/hasplm/hasplm.ini
```

Скопируйте INI файл с конфигурациями. Параметры не описаны в данном документе.

```
cp /var/lib/fs/fs-current/extras/hasp/hasplm.ini /etc/hasplm/
```

#### 4.11.2.3 Добавление библиотеки LP

Скопируйте библиотеку LP (x32 and x64). Она требуется для использования лицензионного ключа LP.

```
cp /var/lib/fs/fs-current/extras/hasp/haspvlib_30147.so /var/hasplm/
```

```
cp /var/lib/fs/fs-current/extras/hasp/haspvlib_x86_64_30147.so /var/hasplm/
```

Удалите библиотеки LP старой версий если они есть:

```
rm -f /var/hasplm/haspvlib_x86_64_111186.so /var/hasplm/haspvlib_111186.so
```

Перезапустите утилиту

```
systemctl restart aksusbd
```

#### 4.11.2.4 Создание отпечатка системы для LUNA PLATFORM

Откройте директорию HASP.

```
cd /var/lib/fs/fs-current/extras/hasp/licenseassist
```

Запустите скрипт.

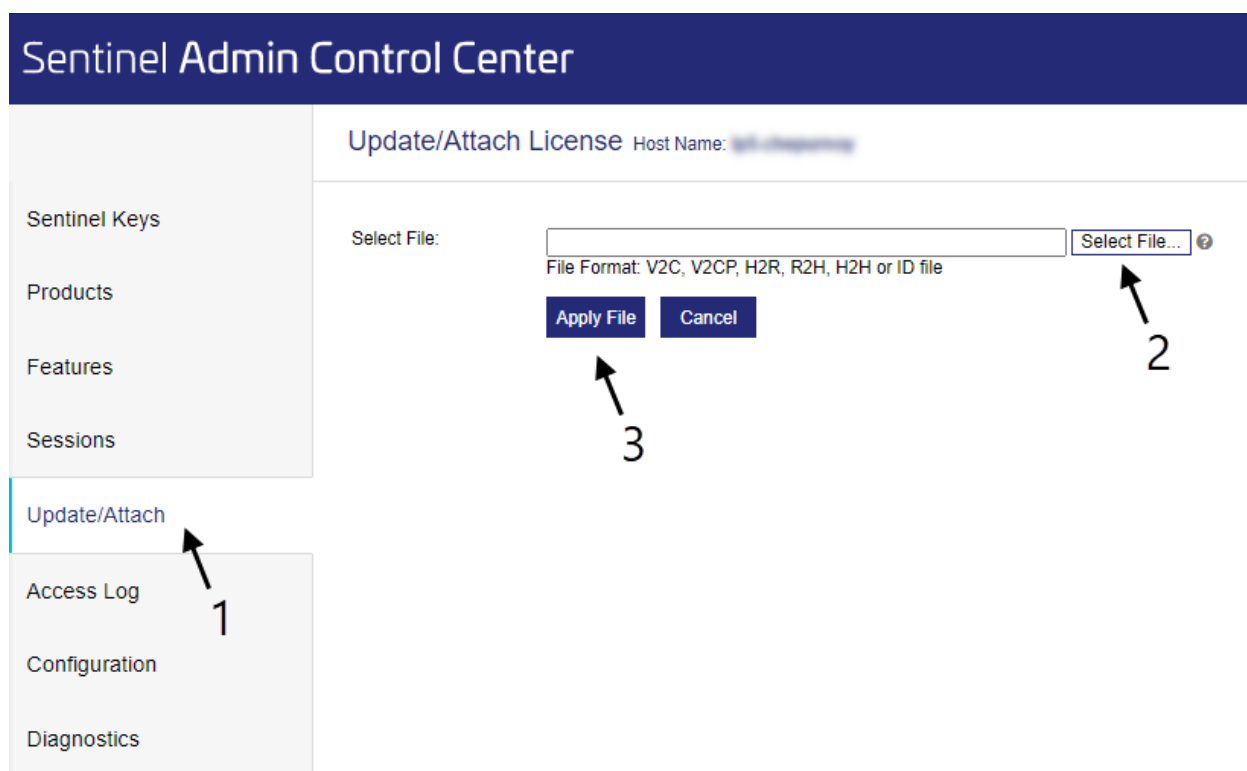
```
./LicenseAssist fingerprint > fingerprint_30147.c2v
```

Отпечаток системы сохраняется в файл «fingerprint\_30147.c2v».

**Отправьте файл в VisionLabs. Ваш лицензионный ключ будет создан с использованием данного отпечатка.**

#### 4.11.2.5 Добавление файла с лицензией вручную с помощью пользовательского интерфейса

- Перейдите: <host\_address>:1947 (если доступ запрещен, проверьте настройки Firewall/SELinux (данная процедура не описана в этом документе));
- Выберите **Update/Attach** в левой панели;
- Нажмите «Select File...» и выберите файл(ы) лицензии в появившемся окне;
- Нажмите «Apply File».



**Рис. 1:** Лицензионный файл добавляется вручную

#### 4.11.2.6 Задание адреса лицензированного сервера

Укажите IP адрес сервера с лицензией в конфигурационном файле в следующей директории:

`/var/lib/fs/fs-current/extras/hasp_redirect/`

Поменяйте адрес сервера HASP в следующих документах:

```
vi /var/lib/fs/fs-current/extras/hasp_redirect/hasp_30147.ini
```

Поменяйте адрес сервера в файле «hasp\_30147.ini».

```
serveraddr = <HASP_server_address>
```

Файл «hasp\_30147.ini» используется сервисом Licenses при запуске его контейнера. Требуется перезапустить уже запущенный контейнер при изменении сервера.

HASP\_server\_address - IP адрес сервера с вашим ключом HASP. Необходимо использовать IP адрес, а не имя сервера.

#### 4.11.2.7 Удаление утилиты LP HASP

Данное действие требуется выполнить для удаления утилиты.

Остановите и отключите утилиту.

```
systemctl stop aksusbd
```

```
systemctl disable aksusbd
```

```
systemctl daemon-reload
```

```
yum -y remove aksusbd haspd
```

## 4.12 Обновление и запуск сервисов LP

### 4.12.1 Запуск контейнера InfluxDB OSS 2

InfluxDB 2.0.8-alpine требуется для мониторинга сервисов LP (дополнительную информацию см. в разделе «Мониторинг» в руководстве администратора LUNA PLATFORM 5).

**Примечание!** Если у вас уже установлен InfluxDB 2.0.8-alpine, пропустите этот шаг.

Используйте команду `docker run` со следующими параметрами:

```
docker run \
-e DOCKER_INFLUXDB_INIT_MODE=setup \
-e DOCKER_INFLUXDB_INIT_BUCKET=luna_monitoring \
-e DOCKER_INFLUXDB_INIT_USERNAME=luna \
-e DOCKER_INFLUXDB_INIT_PASSWORD=password \
-e DOCKER_INFLUXDB_INIT_ORG=luna \
-e DOCKER_INFLUXDB_INIT_ADMIN_TOKEN=
  kofqt4Pfqn6o0RBtMDQqVoJLgHoxxDUmmhiAZ7JS6VmEnrqZXQhxDhad8AX9tmiJH6CjM7Y1U8p5eSEocG
  == \
-v /etc/localtime:/etc/localtime:ro \
-v /var/lib/fs/fs-current/example-docker/influx:/var/lib/influxdb2 \
--restart=always \
--detach=true \
--network=host \
--name influxdb \
dockerhub.visionlabs.ru/luna/influxdb:2.0.8-alpine
```

### 4.12.2 Запуск контейнера PostgreSQL

Если БД PostgreSQL уже установлена, пропустите этот шаг.

Используйте следующую команду для запуска PostgreSQL.

```
docker run \
--env=POSTGRES_USER=luna \
--env=POSTGRES_PASSWORD=luna \
--shm-size=1g \
-v /var/lib/fs/fs-current/example-docker/postgresql/entrypoint-initdb.d:/
  docker-entrypoint-initdb.d/ \
-v /var/lib/fs/fs-current/example-docker/postgresql/data:/var/lib/
  postgresql/data/ \
-v /etc/localtime:/etc/localtime:ro \
--name=postgres \
--restart=always \
--detach=true \
--network=host \
dockerhub.visionlabs.ru/luna/postgis-vlmatch:12
```

`-v /var/lib/luna/current/example-docker/postgresql/data:/var/lib/postgresql/data/` - эта команда позволяет монтировать директорию «data» в контейнер PostgreSQL. Директория на сервере и директория в контейнере будут синхронизированы. Данные PostgreSQL из контейнера будут сохраняться в эту директорию.

`--network=host` - при необходимости изменить порт для PostgreSQL, следует изменить эту строку на `-p 5440:5432`. Здесь первый порт 5440 - локальный, а 5432 - порт в контейнере.

### 4.12.3 Миграция базы данных LUNA Configurator

Следующая инструкция для миграции базы данных сервиса LUNA Configurator предполагает, что в базе данных уже установлена ревизия миграции настроек. Ревизия устанавливается с помощью скрипта `configs.migrate head`; . Этот скрипт включен в руководство по установке FaceStream. Если установка выполнялась в соответствии с руководством, дополнительных действий не понадобится. Миграция настроек выполнится автоматически.

При отсутствии ревизии следует заново создать структуру базы данных. См. руководство по установке FaceStream, раздел «Инициализация БД Configurator». Затем следует задать все необходимые настройки вручную.

#### 4.12.3.1 Инициализация базы данных LUNA Configurator

При обновлении базы данных сервиса LUNA Configurator с существующими настройками необходимо выполнить миграцию структуры базы данных, а также миграцию сохраненных настроек.

Текущая база данных должна уже содержать ревизию миграции настроек.

Используйте команду `docker run` со следующими параметрами для создания таблиц базы данных сервиса LUNA Configurator.

```
docker run \
-v /etc/localtime:/etc/localtime:ro \
-v /var/lib/fs/fs-current/extras/conf/configurator_configs/
  luna_configurator_postgres.conf:/srv/luna_configurator/configs/config.
  conf \
--network=host \
--rm \
--entrypoint bash \
dockerhub.visionlabs.ru/luna/luna-configurator:v.2.0.82 \
-c "alembic upgrade head; cd /srv/luna_configurator/configs/configs/;
  python3 -m configs.migrate --config /srv/luna_configurator/configs/config.
  .conf head;"
```

`alembic upgrade head`; - обновляет структуру существующей базы данных.

`python3 -m configs.migrate head`; - выполняет миграции настроек в базе данных сервиса LUNA Configurator и устанавливает ревизию для миграции. Ревизия потребуется в процессе обновления на новую версию LUNA Configurator.

#### 4.12.4 Запуск контейнера LUNA Configurator

Используйте следующую команду для запуска LUNA Configurator.

```
docker run \
--env=PORT=5070 \
--env=WORKER_COUNT=1 \
--env=RELOAD_CONFIG=1 \
--env=RELOAD_CONFIG_INTERVAL=10 \
-v /etc/localtime:/etc/localtime:ro \
-v /var/lib/fs/BACKUP_luna_configurator_postgres.conf:/srv/luna_configurator
  /configs/config.conf \
--name=luna-configurator \
--restart=always \
--detach=true \
--network=host \
dockerhub.visionlabs.ru/luna/luna-configurator:v.2.0.82
```

#### 4.12.5 Запуск контейнера LUNA Licenses

Убедитесь в том, что задали адрес сервера с лицензиями в файле «hasp\_30147.ini». См. раздел [«Задание адреса лицензированного сервера»](#).

Добавьте право доступа для пользователя «luna» в директорию «hasp\_redirect».

```
chown -R 1001:0 /var/lib/fs/fs-current/extras/hasp_redirect/
```

Используйте следующую команду для запуска сервиса:

```
docker run \  
--env=CONFIGURATOR_HOST=127.0.0.1 \  
--env=CONFIGURATOR_PORT=5070 \  
--env=PORT=5120 \  
--env=WORKER_COUNT=1 \  
--env=RELOAD_CONFIG=1 \  
--env=RELOAD_CONFIG_INTERVAL=10 \  
-v /etc/localtime:/etc/localtime:ro \  
-v /var/lib/fs/fs-current/extras/hasp_redirect/hasp_30147.ini:/home/luna/.  
    hasplm/hasp_30147.ini \  
--name=luna-licenses \  
--restart=always \  
--detach=true \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-licenses:v.0.3.89
```

##### 4.12.5.1 Указание адреса сервиса LUNA Licenses

**Примечание.** Выполняйте нижеперечисленные действия только если запускаете LUNA Streams не на сервере с LUNA Licenses. В противном случае данный шаг можно пропустить.

Перейдите в пользовательский интерфейс сервиса LUNA Configurator и выберите «luna-streams» в выпадающем списке «Service name».

В секции «LUNA\_LICENSES\_ADDRESS» укажите адрес сервера, где запущен сервис LUNA Licenses.



## 5 Обновление FaceStream

Перед запуском FaceStream должны быть выполнены действия, описанные в разделе «Подготовка к обновлению».

### 5.1 Миграция настроек

Для сохранения возможности использования пользовательских настроек LUNA Streams из предыдущей версии, необходимо выполнить миграцию.

Настройки FaceStream не требуют миграции в текущем релизе.

Выполните следующую команду для миграции настроек LUNA Streams:

```
docker run \
-v /etc/localtime:/etc/localtime:ro \
--rm \
--entrypoint=' ' \
--network=host \
dockerhub.visionlabs.ru/luna/streams-configs:v.0.5.9 \
python3 -m streams_configs.migrate head --config_db_url postgres://luna:
luna@127.0.0.1:5432/luna_configurator
```

--config\_db\_url postgres://luna:luna@127.0.0.1:5432/luna\_configurator - флаг указания адреса БД luna\_configurator

### 5.2 Миграция базы данных LUNA Streams

Необходимо выполнить скрипт миграции для обновления структуры базы данных LUNA Streams.

Рекомендуется создать резервную копию базы данных перед тем, как принимать какие-либо изменения.

```
docker run \
-v /etc/localtime:/etc/localtime:ro \
--rm \
--network=host \
dockerhub.visionlabs.ru/luna/luna-streams:v.0.5.9 \
alembic -x luna-config=http://127.0.0.1:5070/1 upgrade head
```

### 5.3 Команда запуска контейнера LUNA Streams

Запуск контейнера осуществляется следующей командой:

```
docker run \
--env=CONFIGURATOR_HOST=127.0.0.1 \
--env=CONFIGURATOR_PORT=5070 \
--env=PORT=5160 \
--env=WORKER_COUNT=1 \
--env=RELOAD_CONFIG=1 \
--env=RELOAD_CONFIG_INTERVAL=10 \
-v /etc/localtime:/etc/localtime:ro \
--name=luna-streams \
--restart=always \
--detach=true \
--network=host \
dockerhub.visionlabs.ru/luna/luna-streams:v.0.5.9
```

## 5.4 Команды запуска контейнера FaceStream

### 5.4.1 Команда запуска контейнера с использованием CPU

Запуск контейнера осуществляется следующим образом:

```
docker run \
--env=CONFIGURATOR_HOST=127.0.0.1 \
--env=CONFIGURATOR_PORT=5070 \
-v /var/lib/fs/fs-current/extras/conf/configs/faceengine.conf:/srv/
  facestream/data/faceengine.conf \
-v /var/lib/fs/fs-current/extras/conf/configs/runtime.conf:/srv/facestream/
  data/runtime.conf \
-v /etc/localtime:/etc/localtime:ro \
--env=PORT=34569 \
--detach=true \
--restart=always \
--name=facestream \
--network=host \
dockerhub.visionlabs.ru/luna/facestream:v.5.1.14
```

Описание остальных параметров и ключей запуска см. ниже в соответствующих разделах.

Список потоков доступен по адресу <http://127.0.0.1:34569/api/1/streams/>. Просмотр потока в браузере доступен по адресу [http://127.0.0.1:34569/api/1/streams/preview/<stream\\_id>](http://127.0.0.1:34569/api/1/streams/preview/<stream_id>).

### 5.4.2 Команда запуска контейнера с использованием GPU

**Примечание.** Используйте данную команду только если собираетесь использовать FaceStream с GPU.

Перед запуском FaceStream в режиме GPU необходимо установить дополнительные зависимости (см. раздел «[Установка зависимостей для GPU](#)»).

Перед запуском контейнера FaceStream с GPU требуется **включить использование GPU** для вычислений в настройках FaceStream с помощью параметра «enable\_gpu\_processing» (см. раздел «Настройки FaceStream» в руководстве администратора).

```
docker run \
--env=CONFIGURATOR_HOST=127.0.0.1 \
--env=CONFIGURATOR_PORT=5070 \
-v /var/lib/fs/fs-current/extras/conf/configs/faceengine.conf:/srv/
  facestream/data/faceengine.conf \
-v /var/lib/fs/fs-current/extras/conf/configs/runtime.conf:/srv/facestream/
  data/runtime.conf \
-v /etc/localtime:/etc/localtime:ro \
--env=PORT=34569 \
--gpus device=0 \
--detach=true \
--restart=always \
--name=facestream \
--network=host \
dockerhub.visionlabs.ru/luna/facestream:v.5.1.14
```

--gpus device=0 - параметр указывает используемое устройство GPU и позволяет использовать GPU. Один GPU используется для одного экземпляра FaceStream. Использование множества GPU для одного экземпляра невозможно.

Описание остальных параметров и ключей запуска см. ниже в соответствующих разделах.

Список потоков доступен по адресу <http://127.0.0.1:34569/api/1/streams/>. Просмотр потока в браузере доступен по адресу [http://127.0.0.1:34569/api/1/streams/preview/<stream\\_id>](http://127.0.0.1:34569/api/1/streams/preview/<stream_id>).

### 5.4.3 Ключи запуска

Ключи запуска задаются с помощью переменных окружения:

—env= - этот параметр задает переменные окружения, требуемые для запуска контейнера. Указываются следующие основные значения:

- CONFIGURATOR\_HOST=127.0.0.1 - хост, на котором запущен сервис Configurator. Локальный хост задается в случае, если контейнер запущен на том же сервере, где работает Configurator.
- CONFIGURATOR\_PORT=5070 - порт прослушивания для сервиса Configurator. По умолчанию используется порт 5070.
- PORT=34569 - порт, где FaceStream будет слушать.
- STREAMS\_ID=« » - в теге задается список идентификаторов потоков, которые будут запрошены из LUNA Streams для обработки. Остальные потоки будут отфильтрованы. Параметр «stream\_id» выдается в ответе на запрос «create stream».

Если значение равно « » или тег «STREAMS\_ID» не задан, то FaceStream будет брать из очереди все существующие «stream\_id».

Если задано несуществующее значение, то при запуске FaceStream будет указана ошибка о некорректном UUID.

По умолчанию значение равно « ».

Для использования ключа должны быть указаны переменные «CONFIGURATOR\_HOST» и «CONFIGURATOR\_PORT».

- STREAMS\_NAME=« » - в теге задается список имен потоков. Имена потоков задаются с помощью параметра «name» во время их создания (запрос «create streams»). Потоки с данными именами будут запрошены из LUNA Streams для обработки. Остальные потоки будут отфильтрованы.

В остальном принцип работы схож с тегом «STREAMS\_ID».

- GROUPS\_ID=« » и GROUPS\_NAME=« » - в тегах задаются список идентификаторов групп или список имён групп. Параметры «group\_id» или «group\_name» задаются во время создания потока (запрос «create stream»). Потоки с данными параметрами будут запрошены из LUNA Streams для обработки. Остальные потоки будут отфильтрованы.

Если значение равно « » или теги «GROUPS\_ID»/«GROUPS\_NAME» не заданы, то FaceStream не будет фильтровать потоки по группам.

Если задано несуществующее значение, то при запуске FaceStream будет указана ошибка о некорректном UUID.

По умолчанию значение равно « ».

Для использования ключей должны быть указаны переменные «CONFIGURATOR\_HOST» и «CONFIGURATOR\_PORT».

Для тегов «STREAMS\_NAME», «STREAMS\_ID», «GROUPS\_NAME» и «GROUPS\_ID» можно задать несколько значений. Пример синтаксиса: `-env=STREAMS_ID=«037f3196-c874-4eca-9d7c-91fd8dfc9593 4caf7cf7-dd0d-4ad5-a35e-b263e742e28a»`

- CONFIGS\_ID=« » - тег LUNA Configurator, который связан с основными конфигурациями для работы FaceStream. Единый тег должен быть задан для «TRACK\_ENGINE\_CONFIG» и «FACE\_STREAM\_CONFIG».

Если задано значение « », то будут использованы записи по умолчанию «TRACK\_ENGINE\_CONFIG» и «FACE\_STREAM\_CONFIG» из LUNA Configurator. Если запись по умолчанию не существует или содержит недопустимый для JSON синтаксис, то будет использован конфигурационный файл из комплекта поставки.

По умолчанию значение равно « ».

Для использования ключа должны быть указаны переменные «CONFIGURATOR\_HOST» и «CONFIGURATOR\_PORT».

- CONFIG\_RELOAD = 1 - тег, включающий проверку наличия изменений в секции «FACE\_STREAM\_CONFIG» сервиса LUNA Configurator и принимающий следующие значения:
  - «1» - отслеживание изменений включено, при наличии изменений в конфигурации будут автоматически перезапущены все контейнеры FaceStream;
  - «0» - отслеживание изменений отключено.

По умолчанию значение равно «1».

- PULLING\_TIME = 10 - тег, задающий период получения новых параметров из секции «FACE\_STREAM\_CONFIG» сервиса LUNA Configurator в диапазоне [1...3600] сек. Используется совместно с тегом «CONFIG-RELOAD».

По умолчанию значение равно «10».

–device= - данный параметр необходим для указания адреса USB устройства. Адрес должен быть указан в источнике потока при его создании. Пример: --device=/dev/video0.

См. принцип работы FaceStream с LUNA Configurator в разделе «Использование FaceStream с LUNA Configurator» руководства администратора.

#### 5.4.4 Расшифровка параметров запуска контейнера

`docker run` - команда для запуска выбранного образа в качестве нового контейнера.

–v - параметр volume позволяет загружать содержимое серверной папки в объем контейнера. Таким образом содержимое синхронизируется.

–v /var/lib/fs/fs-current/extras/conf/configs/faceengine.conf:/srv/facestream/data/faceengine.conf \ - этот параметр позволяет использовать настройки FaceEngine из конфигурационного файла «faceengine.conf».

–v /var/lib/fs/fs-current/extras/conf/configs/runtime.conf:/srv/facestream/data/runtime.conf \ - этот параметр позволяет монтировать конфигурационный файл среды

выполнения в контейнер FaceStream. Перед изменением стандартных настроек требуется проконсультироваться со специалистами VisionLabs.

`--network=host` - этот параметр указывает, что отсутствует симуляция сети и используется серверная сеть. При необходимости изменить порт для сторонних контейнеров следует заменить эту строку на `-p 5440:5432`. Здесь первый порт 5440 - это локальный порт, а 5432 - это порт, используемый в контейнере.

`/etc/localtime:/etc/localtime:ro` - задает текущий часовой пояс, используемый системой контейнера.

`--name=facestream` - этот параметр задает имя запускаемого контейнера. Имя должно быть уникальным. Если уже существует контейнер с таким же именем, произойдет ошибка.

`--restart=always` - этот параметр определяет политику перезагрузки. Демон всегда перезагружает контейнер вне зависимости от кода завершения.

`--detach=true` - запуск контейнера в фоновом режиме.

## 6 Команды Docker

### 6.1 Показать контейнеры

Чтобы показать список запущенных Docker контейнеров, используйте команду:

```
docker ps
```

Чтобы показать все имеющиеся Docker контейнеры, используйте команду:

```
docker ps -a
```

### 6.2 Копировать файлы в контейнер

Можно переносить файлы в контейнер. Используйте команду `docker cp` для копирования файла в контейнер.

```
docker cp <file_location> <container_name>:<folder_inside_container>
```

### 6.3 Вход в контейнер

Можно входить в отдельные контейнеры с помощью следующей команды:

```
docker exec -it <container_name> bash
```

Для выхода из контейнера используйте следующую команду:

```
exit
```

### 6.4 Имена образов

Можно увидеть все имена образов с помощью команды

```
docker images
```

### 6.5 Просмотр логов контейнера

Просмотреть логи контейнера можно с помощью следующей команды:

```
docker logs <container_name>
```

## 6.6 Удаление образа

Если требуется удаление образа:

- запустите команду `docker images`
- найдите требуемый образ, например: `dockerhub.visionlabs.ru/luna/v.5.1.14`
- скопируйте соответствующий ID образа из IMAGE ID, например, "61860d036d8c"
- укажите его в команде удаления:

```
docker rmi -f 61860d036d8c
```

Удаление всех существующих образов:

```
docker rmi -f $(docker images -q)
```

## 6.7 Остановка контейнера

Контейнер можно остановить с помощью следующей команды:

```
docker stop <container_name>
```

Остановка всех контейнеры:

```
docker stop $(docker ps -a -q)
```

## 6.8 Удаление контейнера

Если необходимо удалить контейнер:

- запустите команду "docker ps"
- остановите контейнер (см. [Остановка контейнера](#))
- найдите требуемый образ, например: `dockerhub.visionlabs.ru/luna/v.5.1.14`
- скопируйте соответствующий ID контейнера из столбца CONTAINER ID, например, "23f555be8f3a"
- укажите его в команде удаления:



```
docker container rm -f 23f555be8f3a
```

Удаление всех контейнеров:

```
docker container rm -f $(docker container ls -aq)
```