

VisionLabs FaceStream

Руководство по установке

v.5.1.9

Глоссарий	4
1 Введение	6
2 Аппаратные требования	8
2.1 Минимальные аппаратные требования	8
3 Программные требования	9
Общая информация	10
Последовательность действий перед запуском	10
Последовательность действий без запущенной LP	11
Последовательность действий при запущенной LP	11
4 Подготовка к запуску	12
4.1 Распаковка архива	12
4.2 Создание символической ссылки	12
4.3 Включение записи логов в файл для FaceStream	13
4.4 Установка Docker	13
4.5 Установка Docker Compose	14
4.6 Установка зависимостей для GPU	15
4.6.1 Действия для запуска FaceStream с GPU через Docker Compose	16
4.7 Вход в registry	16
4.8 Активация лицензионного ключа	17
4.8.1 Активация ключа при запущенной LP	17
4.8.2 Активация ключа без запущенной LP	17
4.8.2.1 Установка утилиты HASP	18
4.8.2.2 Конфигурация утилиты HASP	19
4.8.2.3 Добавление библиотеки LP	19
4.8.2.4 Создание отпечатка системы для LUNA PLATFORM	19
4.8.2.5 Добавление файла с лицензией вручную с помощью пользова- тельского интерфейса	20
4.8.2.6 Задание адреса лицензированного сервера	21
4.8.2.7 Удаление утилиты LP HASP	21
4.9 Запуск сервисов LP	22
4.9.1 Запуск контейнера InfluxDB OSS 2	22
4.9.2 Запуск контейнера PostgreSQL	22
4.9.3 Создание БД LUNA Configurator	23
4.9.4 Инициализация БД Configurator	24

4.9.5	Запуск контейнера LUNA Configurator	24
4.9.6	Запуск контейнера LUNA Licenses	25
4.10	Указание адреса сервиса LUNA Licenses	26
5	Ручной запуск FaceStream	27
5.1	Загрузка настроек в LUNA Configurator	27
5.2	Подготовка БД LUNA Streams	28
5.2.1	Создание БД LUNA Streams в контейнере PostgreSQL	28
5.2.2	Инициализация БД LUNA Streams	29
5.3	Команда запуска контейнера LUNA Streams	29
5.4	Команды запуска контейнера FaceStream	30
5.4.1	Команда запуска контейнера с использованием CPU	30
5.4.2	Команда запуска контейнера с использованием GPU	30
5.4.3	Ключи запуска	31
5.4.4	Расшифровка параметров запуска контейнера	33
6	Запуск FaceStream с помощью Docker Compose	34
6.1	Команда запуска FaceStream с помощью Docker Compose	35
7	Команды Docker	36
7.1	Показать контейнеры	36
7.2	Копировать файлы в контейнер	36
7.3	Вход в контейнер	36
7.4	Имена образов	36
7.5	Просмотр логов контейнера	36
7.6	Удаление образа	37
7.7	Остановка контейнера	37
7.8	Удаление контейнера	37

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	Фрагмент изображения лица с кадра, отобранного в соответствии с настройками алгоритма и максимально приближенный к требованиям ГОСТ 19794-5-2006 / ISO IEC 19794-5 2005(E).
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

Данный документ описывает:

- ручной процесс запуска приложения FaceStream и требуемых для него зависимостей с помощью Docker.
- автоматическое развертывание приложения FaceStream вместе с сервисом LUNA Streams с помощью Docker Compose.

Перед запуском необходимо ознакомиться с общей информацией и последовательностью действий.

Лицензирование FaceStream регулируется особым параметром лицензионного ключа LUNA PLATFORM 5, определяющим количество потоков для работы LUNA Streams. Таким образом, для работы FaceStream необходима лицензия LUNA PLATFORM 5.

Если FaceStream запускается без заранее запущенных сервисов LP, то необходимо запросить новую лицензию LUNA PLATFORM 5 с параметром, определяющим количество потоков для работы LUNA Streams, и пройти полный процесс активации.

Если FaceStream запускается с заранее запущенными сервисами LP то нужно убедиться, что лицензионный ключ содержит параметр, определяющий количество потоков для работы LUNA Streams. Если LUNA Licenses запущена на другом сервере, то также нужно будет указать адрес этого сервера в настройках LUNA Streams в сервисе LUNA Configurator.

Docker позволяет создать контейнер, в котором уже имеется требуемый сервис, программная среда для него и минимальный набор необходимых инструментов.

У каждого сервиса LUNA PLATFORM и FaceStream есть собственный образ. Образы Docker являются основой для контейнеров. Каждый контейнер включает в себя библиотеки, необходимые для функционирования сервисов, и параметры, используемых во время работы контейнера.

Docker Compose используется для автоматического развертывания контейнеров. Скрипт Docker Compose из данного дистрибутива используется для развертывания LUNA Streams и FaceStream на одном сервере.

Считается, что запуск выполняется на сервере с операционной системой CentOS, где FaceStream не был установлен.

Для запуска FaceStream необходимы образы Docker контейнеров LP и FaceStream. Для загрузки образов Docker на сервер требуется интернет-соединение или контейнеры должны быть заранее загружены на любом другом устройстве и перенесены на сервер. Требуется вручную задать логин и пароль для скачивания образов Docker.

Администратор должен вручную настроить Firewall и SELinux на сервере. В данном документе не описывается их настройка.

Данный документ не включает руководство по использованию Docker. Пожалуйста, обратитесь к документации Docker для более подробной информации:

<https://docs.docker.com>

В данном документе приведены примеры разворачивания 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.

Подробную информацию по общей работе и настройках приложения см. в руководстве администратора FaceStream.

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.

Последовательность действий перед запуском

Перед запуском FaceStream необходимо выполнить ряд действий, описанных в разделе «Подготовка к запуску».

Часть действий является обязательной, а часть действий зависит от того, запущены ли сервисы LUNA PLATFORM и активирована ли лицензия.

Последовательность действий без запущенной LP

Если на момент начала запуска FaceStream сервисы LUNA PLATFORM не запущены и не активирована лицензия, то перед ручным или автоматическим запуском нужно выполнить следующие действия:

- Активировать лицензию (см. раздел [«Активация ключа без запущенной LP»](#));
- Запустить требуемые компоненты (см. [«Запуск сервисов LP»](#)):
 - контейнер Influx OSS 2
 - контейнер PostgreSQL
 - контейнер LUNA Configurator
 - контейнер LUNA Licenses
- Загрузить настройки LUNA Streams и FaceStream в LUNA Configurator (см. [«Загрузка настроек в LUNA Configurator»](#));
- Создать и инициализировать базу данных LUNA Streams (см. [«Подготовка БД LUNA Streams»](#)).

После выполненных действий можно приступить к ручному или автоматическому запуску LUNA Streams и FaceStream.

Последовательность действий при запущенной LP

Если на момент начала запуска FaceStream сервисы LUNA PLATFORM запущены и активирована лицензия LUNA PLATFORM, то перед ручным или автоматическим запуском нужно выполнить следующие действия:

- Убедиться, что в лицензии указан параметр, определяющий количество потоков для обработки сервисом LUNA Streams и указать адрес сервера с запущенным сервисом LUNA Licenses в сервисе LUNA Configurator (см. раздел [«Активация ключа при запущенной LP»](#));
- Загрузить настройки LUNA Streams и FaceStream в LUNA Configurator (см. [«Загрузка настроек в LUNA Configurator»](#));
- Создать и инициализировать базу данных LUNA Streams (см. [«Подготовка БД LUNA Streams»](#)).

После выполненных действий можно приступить к ручному или автоматическому запуску LUNA Streams и FaceStream.

4 Подготовка к запуску

Перед запуском FaceStream необходимо выполнить ряд дополнительных действий.

4.1 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.5.1.9.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.5.1.9.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.

4.2 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.5.1.9 fs-current
```

4.3 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).

4.4 Установка Docker

Docker требуется для запуска контейнера FaceStream.

Установка Docker описана в официальной документации:

<https://docs.docker.com/engine/install/centos/>.

Если на сервере уже установлен Docker последней версии, то выполнять повторную установку не требуется.

Ниже перечислены команды для быстрой установки:

При возникновении проблем с установкой обратитесь к официальной документации Docker.

Установите зависимости.

```
yum install -y yum-utils device-mapper-persistent-data lvm2
```

Добавьте репозиторий.

```
yum-config-manager --add-repo https://download.docker.com/linux/centos/  
docker-ce.repo
```

Установите Docker.

```
yum -y install docker-ce docker-ce-cli containerd.io
```

Запустите Docker.

```
systemctl start docker
```

```
systemctl enable docker
```

Проверьте, запущен ли Docker.

```
systemctl status docker
```

4.5 Установка Docker Compose

Примечание. Выполняйте установку Docker Compose только если собираетесь использовать скрипт автоматического запуска FaceStream.

Установите Docker Compose.

```
curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-  
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

```
chmod +x /usr/local/bin/docker-compose
```

```
ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

Для получения более подробной информации обратитесь к официальной документации:

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

4.6 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.

4.6.1 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>.

4.7 Вход в registry

При запуске контейнеров необходимо указать ссылку на образ, необходимый для запуска контейнера. Этот образ загружается из VisionLabs registry. Перед этим необходима авторизация.

Введите логин `<username>`.


```
docker login dockerhub.visionlabs.ru --username <username>
```

После выполнения команды будет запрошен ввод пароля. Введите пароль.

Получить логин и пароль можно из VisionLabs.

В команде `docker login` можно вводить логин и пароль одновременно, однако это не гарантирует безопасность, т.к. пароль можно будет увидеть в истории команд.

4.8 Активация лицензионного ключа

4.8.1 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"](#).

4.8.2 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](#)».

4.8.2.1 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
```

4.8.2.2 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/
```

4.8.2.3 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
```

4.8.2.4 Создание отпечатка системы для LUNA PLATFORM

Откройте директорию HASP.

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

Добавьте разрешения для скрипта.

```
chmod +x 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.

4.8.2.5 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

4.8.2.6 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.

4.8.2.7 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
```

4.9 Запуск сервисов LP

Примечание. Перед запуском сервисов LP убедитесь, что лицензия активирована (см. раздел ["Активация ключа без запущенной LP"](#)).

Запуск сервисов LUNA PLATFORM необходим в случае, если не запущены следующие дополнительные компоненты:

- InfluxDB OSS 2
- PostgreSQL
- сервис LUNA Configurator
- сервис LUNA Licenses

Если вышеперечисленные компоненты уже запущены, пропустите данный раздел.

4.9.1 Запуск контейнера InfluxDB OSS 2

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

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
```

4.9.2 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.

`-v /var/lib/luna/current/example-docker/postgresql/entrypoint-initdb.d:/docker-entrypoint-initdb.d/` \ - скрипт «docker-entrypoint-initdb.d» включает команды для создания баз данных LUNA Streams и LUNA Configurator. При создании базы данных автоматически используются имя пользователя и пароль по умолчанию.

`--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.

Примечание! Необходимо вручную создать базу данных для сервиса LUNA Streams если собираетесь использовать уже установленную БД PostgreSQL. Команды для создания базы данных LUNA Streams приведены в разделе «[Создание БД LUNA Streams](#)»).

4.9.3 Создание БД LUNA Configurator

Создайте базу данных для LUNA Configurator в контейнере Postgres:

```
docker exec -i postgres psql -U luna -c "CREATE DATABASE luna_configurator;"
```

Дайте возможность пользователю авторизоваться в БД:

```
docker exec -i postgres psql -U luna -c "GRANT ALL PRIVILEGES ON DATABASE
luna_configurator TO luna;"
```

4.9.4 Инициализация БД Configurator

Используйте команду `docker run` со следующими параметрами для создания таблиц базы данных 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 \
-v /var/lib/fs/fs-current/extras/conf/configurator_configs/platform_settings
.json:/srv/luna_configurator/used_dumps/platform_settings.json \
--network=host \
--rm \
--entrypoint bash \
dockerhub.visionlabs.ru/luna/luna-configurator:v.2.0.58 \
-c "python3.9 ./base_scripts/db_create.py; cd /srv/luna_configurator/configs
/configs/; python3.9 -m configs.migrate --config /srv/luna_configurator/
configs/config.conf --profile platform head; cd /srv; python3.9 ./
base_scripts/db_create.py --dump-file /srv/luna_configurator/used_dumps/
platform_settings.json"
```

`/var/lib/fs/fs-current/extras/conf/configurator_configs/platform_settings.json`
- позволяет задавать путь к файлу с конфигурациями LP.

`./base_scripts/db_create.py`; - создает структуру базы данных.

`python3.9 -m configs.migrate --profile platform head`; - выполняет миграции настроек в базе данных Configurator и устанавливает ревизию для миграции.

`--dump-file /srv/luna_configurator/used_dumps/platform_settings.json` - обновляет настройки в базе данных Configurator значениями из предоставленного файла.

4.9.5 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/fs-current/extras/conf/configurator_configs/
  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.2.0.58

```

4.9.6 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.0.3.65

```

4.10 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 запущена сервис LUNA Licenses.

5 Ручной запуск FaceStream

Перед ручным запуском FaceStream обязательно должны быть выполнены следующие действия:

- Активирован лицензионный ключ.

Если сервисы LP не запущены и лицензия LP не была активирована на момент начала запуска см. раздел [«Активация ключа без запущенной LP»](#).

Если сервисы LP запущены и лицензия LP была активирована на момент начала запуска см. раздел [«Активация ключа при запущенной LP»](#).

- Запущены требуемые компоненты (см. [«Запуск сервисов LP»](#)):
 - контейнер Influx OSS 2
 - контейнер PostgreSQL
 - контейнер LUNA Configurator
 - контейнер LUNA Licenses
- Настройки LUNA Streams и FaceStream загружены в Configurator (см. [«Загрузка настроек в LUNA Configurator»](#)).
- Создана и инициализирована база данных LUNA Streams (см. [«Подготовка БД LUNA Streams»](#)).

5.1 Загрузка настроек в LUNA Configurator

Основные настройки LUNA Streams и FaceStream быть заданы в сервисе Configurator после его запуска. Исключением являются настройки FaceEngine, которые задаются в конфигурационном файле «faceengine.conf» и передаются во время запуска контейнера FaceStream.

При необходимости можно использовать вместо настроек сервиса Configurator конфигурационные файлы и передавать их во время запуска контейнера (для дополнительной информации см. раздел «Использование FaceStream с конфигурационными файлами» руководства администратора).

Для настройки LUNA Streams используются файлы «streams_postgres_dump.json» и «streams_dump.json». Файлы содержат настройки для соединения LUNA Streams с БД и общие настройки LUNA Streams.

Для настройки FaceStream используется файл «facestream_dump.json». Файл содержит настройки «FACE_STREAM_CONFIG» и «TRACK_ENGINE_CONFIG».

Файлы находятся в директории «example-docker/luna_configurator/dumps».

Для загрузки настроек в сервис LUNA Configurator, необходимо использовать скрипт «load_dump.py».

Для использования скрипта «load_dump.py» требуется наличие Python версии 2.x или 3.x. Если установлена версия 2.x, то скрипт нужно запускать с помощью команды `python`. Если же установлена версия 3.x, то скрипт нужно запускать с помощью команды `python3`.

- Перейдите в директорию со скриптом и файлами с настройками:

```
cd /var/lib/fs/fs-current/example-docker/luna_configurator/dumps/
```

- Запустите скрипт для загрузки настроек FaceStream и LUNA Streams в сервис LUNA Configurator, указав установленную версию Python (команда ниже приводит пример запуска скрипта для Python версии 2.x):

```
python -m load_dump --dump-file=streams_dump.json --luna-config=http://127.0.0.1:5070/1
```

```
python -m load_dump --dump-file=facestream_dump.json --luna-config=http://127.0.0.1:5070/1
```

```
python -m load_dump --dump-file=streams_postgres_dump.json --luna-config=http://127.0.0.1:5070/1
```

Все необходимые параметры будут автоматически добавлены в сервис LUNA Configurator.

5.2 Подготовка БД LUNA Streams

Для запуска FaceStream необходимо запустить сервис LUNA Streams, создав и инициализировав для него БД. Данный сервис не входит в поставку LUNA PLATFORM 5, поэтому должен быть запущен отдельно.

5.2.1 Создание БД LUNA Streams в контейнере PostgreSQL

Примечание. Выполняйте нижеописанные команды только если ранее не выполнялся запуск контейнера PostgreSQL, описанный в разделе «Запуск контейнера PostgreSQL». При выполнении команды запуска контейнера PostgreSQL автоматически создается база данных LUNA Streams.

Создайте базу данных для LUNA Streams:

```
docker exec -i postgres psql -U luna -c "CREATE DATABASE luna_streams;"
```

Дайте возможность пользователю авторизоваться в БД:

```
docker exec -i postgres psql -U luna -c "GRANT ALL PRIVILEGES ON DATABASE
luna_streams TO luna;"
```

Активируйте PostGIS:

```
docker exec -i postgres psql -U luna luna_streams -c "CREATE EXTENSION
postgis;"
```

5.2.2 Инициализация БД LUNA Streams

Инициализируйте данные в БД LUNA Streams:

```
docker run -v /etc/localtime:/etc/localtime:ro \
--rm \
--network=host \
dockerhub.visionlabs.ru/luna/luna-streams:v.0.5.0 \
python3.9 ./base_scripts/db_create.py --luna-config http://localhost:5070/1
```

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.0.5.0
```

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 /etc/localtime:/etc/localtime:ro \
--env=PORT=34569 \
--detach=true \
--restart=always \
--name=facestream \
--network=host \
dockerhub.visionlabs.ru/luna/facestream:v.5.1.9
```

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>.

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 «Настройка 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 /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.5.1.9
```

--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>.

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».

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».

`--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 Запуск FaceStream с помощью Docker Compose

Перед автоматическим запуском FaceStream обязательно должны быть выполнены следующие действия:

- Активирован ключ.

Если сервисы LP не запущены и лицензия LP не была активирована на момент начала установки см. раздел [«Активация ключа без запущенной LP»](#).

Если сервисы LP запущены и лицензия LP была активирована на момент начала установки см. раздел [«Активация ключа при запущенной LP»](#).

- Запущены требуемые компоненты (см. [«Запуск сервисов LP»](#)):
 - контейнер Influx OSS 2
 - контейнер PostgreSQL
 - контейнер LUNA Configurator
 - контейнер LUNA Licenses

Примечания о скрипте Docker Compose. Скрипт:

- загружает настройки LUNA Streams и FaceStream в Configurator
- создает и инициализирует БД LUNA Streams
- запускает LUNA Streams и FaceStream
- тестируется с использованием настроек LUNA Streams и FaceStream по умолчанию.
- не предназначен для использования в целях масштабирования FaceStream:
 - Не используется для развертывания FaceStream на нескольких серверах.
 - Не используется для развертывания и балансирования нескольких экземпляров FaceStream на одном сервере.
- поддерживает использование GPU для вычислений FaceStream.
- не обеспечивает возможность использования внешней базы данных для LUNA Streams, уже установленной на сервере.
- не выполняет миграции из предыдущих версий FaceStream и обновления предыдущих сборок FaceStream.

См. файл «docker-compose.yml» и другие файлы в директории «example-docker» для получения дополнительной информации.

Можно написать собственный скрипт, который разворачивает и конфигурирует все необходимые сервисы LP и FaceStream. Данный документ не включает информацию о создании скриптов и не обучает использованию Docker. Обратитесь к документации Docker для получения подробной информации о Docker и Docker Compose:

<https://docs.docker.com>

6.1 Команда запуска FaceStream с помощью Docker Compose

Откройте директорию Docker Compose:

```
cd /var/lib/fs/fs-current/example-docker
```

Убедитесь в том, что контейнер FS не запущен до выполнения скрипта. Попытка запустить контейнер с таким же именем, как существующий контейнер, приведет к ошибке. Если контейнер запущен, необходимо остановить его с помощью команды `docker container rm -f <container_name>`. Чтобы остановить все контейнеры, используйте `docker container rm -f $(docker container ls -aq)`.

Для запуска FaceStream с GPU с помощью Docker Compose необходимо выполнить действия, описанные в разделе «[Установка зависимостей для GPU](#)».

Запуск FaceStream через Docker Compose:

Необходимо выполнить вход в VisionLabs registry (см. раздел «[Вход в registry](#)»)

```
./start_facestream.sh --configurator_address=127.0.0.1
```

--configurator_address=127.0.0.1 - адрес сервиса Configurator

Проверьте статус всех запущенных Docker контейнеров.

```
docker ps
```

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>.

7 Команды Docker

7.1 Показать контейнеры

Чтобы показать список запущенных Docker контейнеров, используйте команду:

```
docker ps
```

Чтобы показать все имеющиеся Docker контейнеры, используйте команду:

```
docker ps -a
```

7.2 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>
```

7.3 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
```

7.4 Имена образов

Можно увидеть все имена образов с помощью команды

```
docker images
```

7.5 Просмотр логов контейнера

Просмотреть логи контейнера можно с помощью следующей команды:

```
docker logs <container_name>
```

7.6 Удаление образа

Если требуется удаление образа:

- запустите команду `docker images`
- найдите требуемый образ, например: `dockerhub.visionlabs.ru/luna/v.5.1.9`
- 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)
```

7.7 Остановка контейнера

Контейнер можно остановить с помощью следующей команды:

```
docker stop <container_name>
```

Остановка всех контейнеры:

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

7.8 Удаление контейнера

Если необходимо удалить контейнер:

- запустите команду "docker ps"
- остановите контейнер (см. [Остановка контейнера](#))
- найдите требуемый образ, например: `dockerhub.visionlabs.ru/luna/v.5.1.9`
- 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)
```