

VisionLabs FaceStream

Руководство по обновлению с версии v.5.1.16 на v.5.1.18 без запущенной LP

v.5.1.18

Содержание

Глоссарий	4
Системные требования	7
Сервисы LP и сторонние приложения	7
Процессоры	8
CPU	8
GPU	9
Введение	10
1 Подготовка к обновлению	11
1.1 Создание резервных копий	12
1.1.1 Копия настроек сервисов	12
1.2 Удаление символической ссылки	12
1.3 Распаковка архива	12
1.4 Создание символической ссылки	13
1.5 Перенос данных	13
1.5.1 Копирование данных PostgreSQL	14
1.5.2 Копирование данных InfluxDB	14
1.6 Сохранение пользовательских настроек сервиса Configurator	14
1.7 Установка зависимостей для GPU	15
1.7.1 Действия для запуска FaceStream с GPU через Docker Compose	15
1.8 Вход в registry	16
1.9 Активация лицензионного ключа	17
1.9.1 Установка утилиты HASP	17
1.9.2 Конфигурация утилиты HASP	18
1.9.3 Добавление библиотеки LP	18
1.9.4 Создание отпечатка системы для LUNA PLATFORM	18
1.9.5 Добавление файла с лицензией вручную с помощью пользовательского интерфейса	19
1.9.6 Выбор способа задания адреса сервера лицензирования	19
1.9.6.1 Задание адреса сервера лицензирования с помощью дамп-файла	20
1.9.7 Удаление старой утилиты LP HASP	20
1.10 Удаление старых контейнеров	21
1.11 Запуск контейнера InfluxDB OSS 2	21
1.12 Запуск контейнера PostgreSQL	22
1.13 Миграция базы данных LUNA Configurator	23
1.14 Запуск контейнера LUNA Configurator	24

1.15	Сервис LUNA Licenses	24
1.15.1	Задание адреса сервера лицензирования с помощью Configurator	24
1.15.2	Запуск контейнера LUNA Licenses	24
2	Обновление FaceStream	26
2.1	Миграция настроек	26
2.2	Миграция базы данных LUNA Streams	26
2.3	Команда запуска контейнера LUNA Streams	27
2.4	Команды запуска контейнера FaceStream	27
2.4.1	Команда запуска контейнера с использованием CPU	27
2.4.2	Команда запуска контейнера с использованием GPU	28
3	Дополнительная информация	29
3.1	Команды Docker	30
3.1.1	Показать контейнеры	30
3.1.2	Копировать файлы в контейнер	30
3.1.3	Вход в контейнер	30
3.1.4	Имена образов	30
3.1.5	Просмотр логов контейнера	30
3.1.6	Удаление образа	31
3.1.7	Остановка контейнера	31
3.1.8	Удаление контейнера	31
3.2	Ключи запуска	33
3.2.1	Расшифровка параметров запуска контейнера	34
3.3	Настройка ротации логов Docker	36

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	Кадр видеопотока, на котором лицо/тело зафиксировано в оптимальном ракурсе для дальнейшей обработки.
Портрет	Изображение лица или тела, трансформированное под определенный формат. Портрет имеет два типа - «waip» (изображение трансформируется в формат биометрического образца), «ghost» (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.
Трекинг	Функция отслеживания объекта (лица) на последовательности кадров.

Термин	Значение термина
LUNA Streams	Сервис для создания и управления потоками, которые содержат политики обработки видеопотока/видеофайла/набора изображений.

Аббревиатура	Расшифровка
БД, DB	База данных
LP	LUNA PLATFORM

Системные требования

FaceStream поставляется в Docker-контейнерах и может запускаться на CPU и GPU. Для установки необходимы образы Docker-контейнеров. Для загрузки образов Docker на сервере требуется подключение к сети Интернет, либо образы следует загрузить на любое другое устройство и перенести на сервер. Требуется вручную указать логин и пароль для загрузки образов Docker.

FaceStream можно запустить с помощью скрипта Docker Compose.

Рекомендуется использовать следующие версии Docker и Docker Compose для запуска FaceStream:

- Docker: 20.10.8 (для ручного запуска контейнеров)
- Docker Compose: 1.29.2 (для автоматического запуска контейнеров)

Запуск контейнеров FaceStream и LUNA Streams был протестирован на:

- CentOS Linux release 7.8.2003 (Core)

В контейнере FaceStream используется следующая ОС:

- CentOS Linux release 8.3.2011

Сервисы LP и сторонние приложения

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.
- **InfluxDB** используется для мониторинга сервисов LUNA PLATFORM. При необходимости мониторинг можно отключить.

Следующие версии баз данных рекомендованы к использованию с LUNA Streams:

- **PostgreSQL**: 12
- **Oracle**: 11.2

Для загрузки настроек в сервис LUNA Configurator требуется наличие **Python версии 2.x или 3.x**.

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

Балансировщики нагрузки (например, Nginx) и другие программы могут использоваться при масштабировании системы для обеспечения отказоустойчивости. Их конфигурация не описывается в данном руководстве.

Процессоры

Ниже приведены требования для запуска FaceStream в минимальной конфигурации. Требования для использования системы в продуктивном контуре рассчитываются в зависимости от предполагаемой нагрузки.

CPU

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 (см. ниже) и использования AVX2 инструкций. Требуется CPU с поддержкой AVX2. Система автоматически определяет наличие инструкций и запускается в оптимальном режиме.

GPU

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. Требуются 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.

Введение

В данном документе приводится пример шагов, необходимых для обновления на новую сборку FaceStream при условии, что приложение было ранее запущено без установленной LUNA PLATFORM (см. документ «Руководство по установке без запущенной LP»). Если FaceStream запускался в соответствии с руководством по установке FaceStream с запущенной LUNA PLATFORM, то необходимо использовать документ «Руководство по обновлению».

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 без запущенной LUNA PLATFORM, 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 в программе для работы с сетевыми протоколами (в случае удаленного подключения к серверу), например, Putty.

1 Подготовка к обновлению

Убедитесь в том, что вы являетесь **root**-пользователем перед тем, как начать обновление!

Перед обновлением FaceStream необходимо выполнить ряд дополнительных действий:

1. Создать резервные копии
2. Удалить старую символическую ссылку
3. Распаковать дистрибутив новой версии FaceStream
4. Создать новую символическую ссылку
5. Перенести старые данные БД PostgreSQL
6. Перенести старые данные БД Influx
7. Сохранить пользовательские настройки сервиса Configurator, если они изменялись
8. Настроить вычисления с помощью GPU, если планируется использовать GPU
9. Авторизоваться в registry VisonLabs, если ранее не была выполнена авторизация
10. Обновить лицензию, если необходимо
11. Удалить старые контейнеры
12. Запустить контейнер Influx OSS 2
13. Запустить контейнер PostgreSQL
14. Выполнить миграцию БД для LUNA Configurator
15. Запустить контейнер LUNA Configurator
16. Запустить контейнер LUNA Licenses

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

1.1 Создание резервных копий

Рекомендуется создать следующие резервные копии:

- резервная копия БД LUNA Streams (не описано в данной документации);
- резервную копию настроек сервисов LUNA PLATFORM, LUNA Streams и FaceStream.

Создание резервных копий позволит восстановить в случае возникновения каких-либо проблем в процессе миграции.

1.1.1 Копия настроек сервисов

Пользовательские значения настроек сервисов LUNA PLATFORM (всех, кроме сервиса Configurator), LUNA Streams и FaceStream автоматически мигрируются с помощью механизма миграции сервиса Configurator. Данная резервная копия не будет использована в процессе нормальной установки FaceStream.

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
```

1.2 Удаление символической ссылки

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

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

1.3 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.5.1.18.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.5.1.18.zip
```

1.4 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.5.1.18 fs-current
```

1.5 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.5.1.16. Следует выполнить эти действия для сборки FaceStream, установленной на вашем сервере. Измените v.5.1.16 в командах ниже на текущую сборку.

Следует скопировать директорию с данными вашей базы данных из директории «facestream_docker_v.5.1.16» в текущий корневой каталог для того, чтобы использовать эти данные в новой сборке FaceStream.

1.5.1 Копирование данных PostgreSQL

Следующий шаг необходим если используется PostgreSQL в Docker контейнере.

Скопируйте директорию «data»:

```
cp -r /var/lib/fs/facestream_docker_v.5.1.16/example-docker/postgresql /var/lib/fs/fs-current/example-docker/
```

1.5.2 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.5.1.16/example-docker/influx /var/lib/fs/fs-current/example-docker/
```

1.6 Сохранение пользовательских настроек сервиса Configurator

Настройки сервиса Configurator не мигрируются автоматически, в отличие от настроек всех остальных сервисов.

Если предыдущая версия FaceStream использовалась с настройками сервиса Configurator, отличных от настроек по умолчанию, нужно 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
```

Эта резервная копия должна быть примонтирована к запускаемому контейнеру сервиса Configurator.

Если вы не уверены, менялись ли настройки сервиса Configurator, то можете сравнить созданную резервную копию с настройками Configurator из текущей поставки с помощью следующей команды:

```
diff /var/lib/fs/<your_previous_fs_version>/extras/conf/configurator_configs
/luna_configurator_postgres.conf /var/lib/fs/
BACKUP_luna_configurator_postgres.conf
```

1.7 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.

1.7.1 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>.

1.8 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.

Логин и пароль можно запросить у представителя VisionLabs.

Введите логин <username>.

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

После выполнения команды будет запрошен ввод пароля. Введите пароль.

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

1.9 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.

Последовательность действий:

1. Установите на сервер утилиту HASP. Обычно утилита HASP устанавливается на отдельный сервер;
2. Запустите утилиту HASP;
3. Создайте отпечаток системы для вашего сервера и отправьте его в VisionLabs;
4. Активируйте свой ключ, полученный от VisionLabs;
5. Укажите адрес вашего сервера лицензирования.

Вкладка Sentinel Keys пользовательского интерфейса (<server_host_address>:1947) отображает активированные ключи.

1.9.1 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
```

1.9.2 Конфигурация утилиты HASP

Обновите конфигурацию утилиты HASP 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/
```

1.9.3 Добавление библиотеки LP

Скопируйте библиотеки LP (x32 и 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
```

1.9.4 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 представителю VisionLabs. Ваш лицензионный ключ будет создан с использованием данного отпечатка.

1.9.5 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».

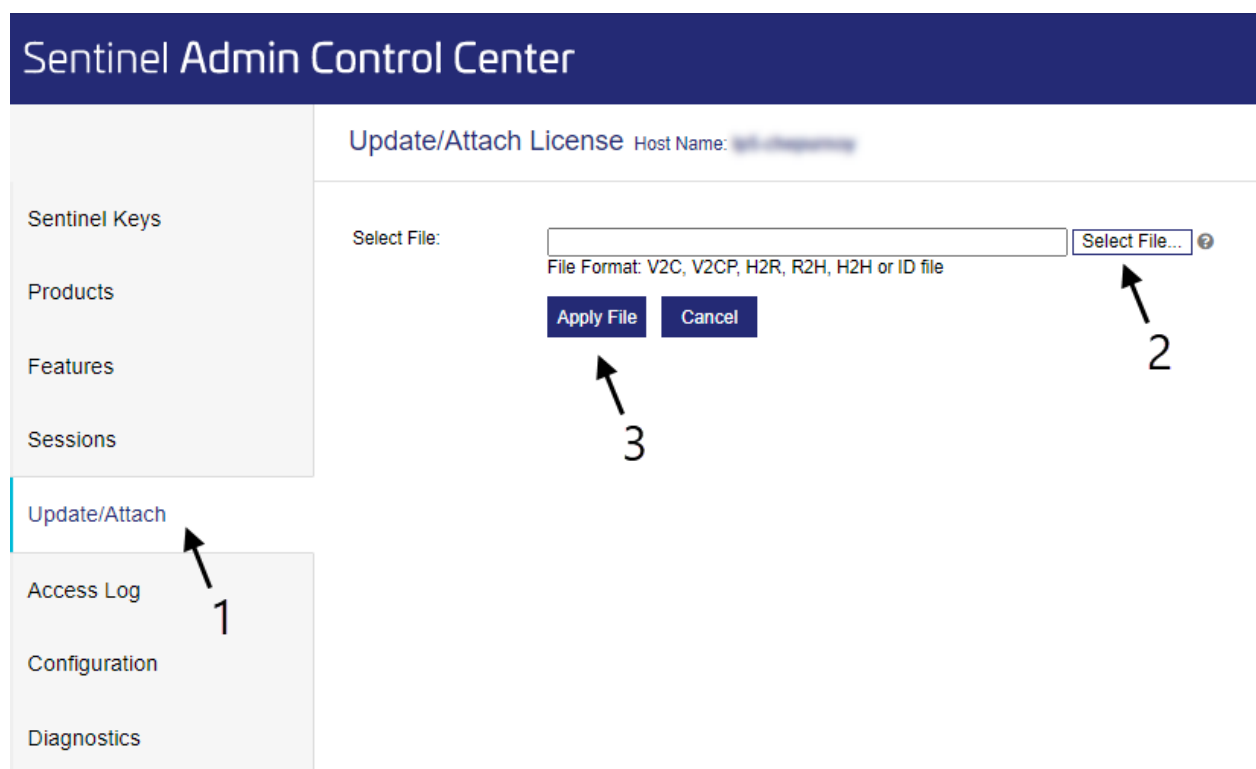


Рис. 1: Добавление лицензионного файла

1.9.6 Выбор способа задания адреса сервера лицензирования

IP-адрес сервера лицензирования можно задать одним из двух способов:

- в дамп-файле «platform_settings.json» (см. ниже). Содержимое стандартных настроек будет перезаписано содержимым этого файла на этапе запуска сервиса Configurator.
- в настройках сервиса Licenses в пользовательском интерфейсе Configurator (см. раздел [«Задание адреса сервера лицензирования с помощью Configurator»](#)).

Выберите наиболее удобный способ и выполните действия, описанные в соответствующих разделах.

1.9.6.1 Задание адреса сервера лицензирования с помощью дамп-файла

Откройте файл «platform_settings.json»:

```
vi /var/lib/fs/fs-current/extras/conf/configurator_configs/platform_settings.json
```

Задайте IP-адрес сервера с вашим ключом HASP в поле «server_address»:

```
{
  "value": {
    "vendor": "hasp",
    "server_address": "127.0.0.1"
  },
  "description": "License vendor config",
  "name": "LICENSE_VENDOR",
  "tags": []
},
```

Сохраните файл.

1.9.7 Удаление старой утилиты LP HASP

Ниже описаны действия, необходимые для удаления утилиты HASP.

```
systemctl stop aksusbd
```

```
systemctl disable aksusbd
```

```
systemctl daemon-reload
```

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

1.10 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
```

1.11 Запуск контейнера InfluxDB OSS 2

InfluxDB 2.0.8-alpine требуется для мониторинга минимально необходимых 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
```

1.12 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.

1.13 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.

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 \
-v /tmp/logs/configurator:/srv/logs \
--network=host \
--rm \
--entrypoint bash \
dockerhub.visionlabs.ru/luna/luna-configurator:v.2.1.0 \
-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.

1.14 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 \
-v /tmp/logs/configurator:/srv/logs \
--name=luna-configurator \
--restart=always \
--detach=true \
--network=host \
dockerhub.visionlabs.ru/luna/luna-configurator:v.2.1.0
```

1.15 Сервис LUNA Licenses

1.15.1 Задание адреса сервера лицензирования с помощью Configurator

Для задания адреса сервера лицензирования нужно выполнить следующие действия:

- перейдите в интерфейс сервиса Configurator http://<configurator_server_ip>:5070/
- введите в поле «Setting name» значение «LICENSE_VENDOR» и нажмите «Apply Filters»
- задайте IP-адрес сервера с вашим ключом HASP в поле «server_address»
- нажмите «Save»

1.15.2 Запуск контейнера LUNA Licenses

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 /tmp/logs/licenses:/srv/logs \  
--name=luna-licenses \  
--restart=always \  
--detach=true \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-licenses:v.0.5.0
```

2 Обновление FaceStream

Перед запуском FaceStream должны быть выполнены действия, описанные в разделе [«Подготовка к обновлению»](#).

2.1 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 \
-v /tmp/logs/streams:/srv/logs \
--rm \
--entrypoint=' ' \
--network=host \
dockerhub.visionlabs.ru/luna/streams-configs:v.0.5.17 \
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

2.2 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 \
-v /tmp/logs/streams:/srv/logs \
--rm \
--network=host \
dockerhub.visionlabs.ru/luna/luna-streams:v.0.5.17 \
alembic -x luna-config=http://127.0.0.1:5070/1 upgrade head
```

2.3 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 \
-v /tmp/logs/streams:/srv/logs \
--name=luna-streams \
--restart=always \
--detach=true \
--network=host \
dockerhub.visionlabs.ru/luna/luna-streams:v.0.5.17
```

Для проверки корректности запуска сервиса можно выполнить GET-запрос `http://127.0.0.1:5160/version`. В ответе должна вернуться версия LUNA Streams v.0.5.17.

2.4 Команды запуска контейнера FaceStream

2.4.1 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 \
-v /tmp/logs/facestream:/srv/logs \
--env=PORT=34569 \
--detach=true \
--restart=always \
--name=facestream \
--network=host \
dockerhub.visionlabs.ru/luna/facestream:v.5.1.18
```

Описание остальных параметров и ключей запуска см. в разделе «[Описание ключей запуска](#)».

Для проверки корректности запуска можно выполнить GET-запрос `http://127.0.0.1:34569/version`. В ответе должна вернуться версия FaceStream.

2.4.2 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 \
-v /tmp/logs/facestream:/srv/logs \
--env=PORT=34569 \
--gpus device=0 \
--detach=true \
--restart=always \
--name=facestream \
--network=host \
dockerhub.visionlabs.ru/luna/facestream:v.5.1.18
```

`--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. в разделе «[Описание ключей запуска](#)».

Для проверки корректности запуска можно выполнить GET-запрос `http://127.0.0.1:34569/version`. В ответе должна вернуться версия FaceStream.

3 Дополнительная информация

В данном разделе приводится следующая дополнительная информация:

- [Полезные команды для работы с Docker](#)
- [Описание ключей запуска](#)
- [Настройка ротации логов Docker](#)

3.1 Команды Docker

3.1.1 Показать контейнеры

Чтобы показать список запущенных Docker контейнеров, используйте команду:

```
docker ps
```

Чтобы показать все имеющиеся Docker контейнеры, используйте команду:

```
docker ps -a
```

3.1.2 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>
```

3.1.3 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
```

3.1.4 Имена образов

Можно увидеть все имена образов с помощью команды

```
docker images
```

3.1.5 Просмотр логов контейнера

Просмотреть логи контейнера можно с помощью следующей команды:

```
docker logs <container_name>
```

3.1.6 Удаление образа

Если требуется удаление образа:

- запустите команду `docker images`
- найдите требуемый образ, например: `dockerhub.visionlabs.ru/luna/facestream:v.5.1.18`
- 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)
```

3.1.7 Остановка контейнера

Контейнер можно остановить с помощью следующей команды:

```
docker stop <container_name>
```

Остановка всех контейнеры:

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

3.1.8 Удаление контейнера

Если необходимо удалить контейнер:

- запустите команду "docker ps"
- остановите контейнер (см. [Остановка контейнера](#))
- найдите требуемый образ, например: `dockerhub.visionlabs.ru/luna/facestream:v.5.1.18`
- 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)
```


3.2 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.

3.2.1 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.

3.3 Настройка ротации логов Docker

Чтобы ограничить размер логов, генерируемых Docker, можно настроить автоматическую ротацию логов. Для этого необходимо добавить в файл `/etc/docker/daemon.json` следующие данные:

```
{
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m",
    "max-file": "5"
  }
}
```

Это позволит Docker хранить до 5 файлов логов на контейнер, при этом каждый файл будет ограничен 100 МБ.

После изменения файла необходимо перезапустить Docker:

```
systemctl reload docker
```

Вышеописанные изменения являются значениями по умолчанию для любого вновь созданного контейнера, они не применяются к уже созданным контейнерам.