



**VisionLabs**  
MACHINES CAN SEE

# **VisionLabs FaceStream**

**Upgrade manual from v.5.1.30 to v.5.1.32 without LP launched**

**v.5.1.32**

## Contents

<b>Glossary</b>	<b>4</b>
<b>System requirements</b>	<b>6</b>
LP services and third-party applications . . . . .	6
Processors . . . . .	7
CPU . . . . .	7
GPU . . . . .	7
<b>Introduction</b>	<b>9</b>
<b>1 Before upgrade</b>	<b>10</b>
1.1 Create backups . . . . .	11
1.1.1 Backup of services configurations . . . . .	11
1.2 Delete symbolic link . . . . .	11
1.3 Unpack distribution . . . . .	11
1.4 Create symbolic link . . . . .	12
1.5 Move data . . . . .	12
1.5.1 Move PostgreSQL data . . . . .	12
1.5.2 Move InfluxDB Data . . . . .	13
1.6 Save user configurations of the Configurator . . . . .	13
1.7 Install GPU dependencies . . . . .	13
1.7.1 Actions to launch FaceStream with GPU through Docker Compose . . . . .	14
1.8 Login to registry . . . . .	15
1.9 License activation . . . . .	16
1.10 Remove old containers . . . . .	16
1.11 Launch InfluxDB OSS 2 container . . . . .	17
1.12 Launch PostgreSQL container . . . . .	17
1.13 LUNA Configurator database migration . . . . .	18
1.13.1 Initialize LUNA Configurator database . . . . .	18
1.14 Launch LUNA Configurator container . . . . .	19
1.15 LUNA Licenses service . . . . .	19
1.15.1 Specify license settings using Configurator . . . . .	19
1.15.1.1 Specify HASP license settings . . . . .	19
1.15.1.2 Specify Guardant license settings . . . . .	20
1.15.2 Launch LUNA Licenses container . . . . .	20
<b>2 Upgrade FaceStream</b>	<b>21</b>
2.1 Migrate settings . . . . .	21
2.2 Migrate LUNA Streams database . . . . .	21

2.3	Launch LUNA Streams container . . . . .	22
2.4	Launch FaceStream container . . . . .	22
2.4.1	Launch FaceStream container using CPU . . . . .	22
2.4.2	Launch FaceStream container using GPU . . . . .	23
<b>3</b>	<b>Additional information</b>	<b>24</b>
3.1	Docker commands . . . . .	25
3.1.1	Show containers . . . . .	25
3.1.2	Copy files to container . . . . .	25
3.1.3	Enter container . . . . .	25
3.1.4	Images names . . . . .	25
3.1.5	Show container logs . . . . .	25
3.1.6	Delete image . . . . .	26
3.1.7	Stop container . . . . .	26
3.1.8	Delete container . . . . .	26
3.2	Launching keys . . . . .	28
3.2.1	Description of container launch parameters . . . . .	29
3.3	Docker log rotation . . . . .	31

## Glossary

Term	Meaning
Aspect angle	Head rotation degree (in degrees) on each of the three axes (up/down tilt relative to the horizontal axis; left/right tilt, relative to the vertical axis; a rotation about the vertical axis).
Batch	Group of data processed simultaneously.
Best shot	The frame of the video stream on which the face/body is fixed in the optimal angle for further processing.
Detection	FaceStream entity that contains the coordinates of face or body and the estimated value of the object that determines the best shot.
Descriptor	A set of unique features received from the warp. A descriptor requires much less storage memory in comparison with the sample and is used for comparison of faces.
Event	LUNA PLATFORM entity, which contains information (city, user data, track id, etc.) about one face and/or body. This information is transferred to the LUNA PLATFORM by the FaceStream application. For a complete list of the transferred information, see the OpenAPI LUNA PLATFORM documentation.
LUNA Streams	Service for creating and managing streams that contain policies for processing a video stream/video file/set of images.
Normalized image, warp	Images containing a face or body and corresponding to VisionLabs standard. Used when working with LUNA PLATFORM.
Portrait	Image of face or body that has been transformed to a specific format. The portrait has two types - “warp” (the image is transformed into warp format), “gost” (detection is cut out from the source frame, considering indentation).
Track	Information about object’s position (face of a person) in a sequence of frames. If the object leaves the frame zone, the track doesn’t discontinue right away. For some time, the system expects the object to return and if it does, the track continues.
Tracking	Object (face) tracking function in the frame sequence.

Abbreviation	Term
DB	Database
LP	LUNA PLATFORM

## System requirements

FaceStream is delivered in Docker containers and can be launched on CPU and GPU. Docker images of the containers are required for the installation. Internet connection is required on the server for Docker images download, or the images should be downloaded on any other device and moved to the server. It is required to manually specify login and password for Docker images downloading.

FaceStream can be launched with a Docker Compose script.

The following Docker and Docker Compose versions are recommended for FaceStream launching:

- Docker: 20.10.8 (to manually launch containers)
- Docker Compose: 1.29.2 (to automatically launch containers)

FaceStream and LUNA Streams containers launch were tested on the following operating systems:

- CentOS Linux release 7.8.2003 (Core)

The following OS is used inside the FaceStream container:

- CentOS Linux release 8.3.2011

## LP services and third-party applications

FaceStream requires LUNA PLATFORM components, additional databases, and the LUNA Streams service. Basic information about this software is contained in this document.

LUNA Streams is not a component of the LUNA PLATFORM.

The following LUNA PLATFORM components are used by default with FaceStream:

- **LUNA Licenses** is used to license the LUNA Streams service.
- **LUNA Configurator** is used for quick access to the basic FaceStream settings and LUNA PLATFORM service settings.
- **PostgreSQL** is used as the default database for the LUNA Streams service. It is also possible to use an Oracle database instead of PostgreSQL.
- **InfluxDB** is used for monitoring. If necessary, monitoring can be disabled.

The following database versions are recommended for use with LUNA Streams:

- **PostgreSQL**: 12
- **Oracle**: 21c

To upload settings to the LUNA Configurator service, **Python version 2.x or 3.x** is required.

Installation and configuration of Oracle is not described in this manual. Further in the document, examples of launching using PostgreSQL will be given.

Balancers (for example, Nginx) and other software can be used when scaling the system to provide fail-safety. Their configuration is not described in this document.

## Processors

Below are the requirements to launch FaceStream in a minimal configuration. System requirements for the production system are calculated based on the intended system load.

### CPU

The following minimum requirements are given per FaceStream instance.

For the application to work correctly, the hardware must meet the following minimum requirements:

- 2 GHz or faster processor;
- 4 Gb RAM or higher;
- 10 Gb available hard disk space.
- Access to the Internet (for containers and additional software download).

Hardware requirements can be affected by several factors:

- Number of video streams;
- Frame frequency and resolution of video streams;
- FaceStream settings. The default settings are the most versatile. Depending on the operating conditions of the application, using their values can affect the quality, or performance.

Hardware should be selected based on the above factors.

FaceStream can also work in the computation speedup mode due to usage of video card resources or AVX2 instructions. CPU with AVX2 support is required. The system automatically detects available instructions and runs best performance.

### GPU

GPU calculations are supported for FaceDetV3 only. See “defaultDetectorType” parameter in the FaceEngine configuration (“faceengine.conf”).

A minimum of 6GB or dedicated video RAM is required. 8 GB or more VRAM recommended.

Pascal, Volta, Turing architectures are supported.

Compute Capability 6.1 or higher and CUDA 11.4 are required.

The recommended NVIDIA driver is r470.

Now only one video card is supported per FaceStream instance.



## Introduction

This document provides an example of the steps required to upgrade to a new FaceStream build, provided that the application was previously launched without LUNA PLATFORM installed (see the document “Installation manual without LP launched”). If FaceStream was launched in accordance with the FaceStream installation manual with the LUNA PLATFORM launched, then the “Upgrade manual” document must be used.

This manual is intended with an assumption that:

- you already have a previous minor version of FaceStream distribution installed and the required environment is up and running at your server(s).
- FaceStream is installed according to FaceStream installation manual without LP launched, and the default paths are used. Otherwise, you should consider your manual changes during the update.

This document includes an example of FaceStream deployment. It implements LUNA PLATFORM minimum power operating for demonstration purposes and cannot be used for the production system.

All the provided commands should be executed using the Bash shell (when you launch commands directly on the server) or in a program for working with network protocols (when you remotely connect to the server), for example, Putty.

## 1 Before upgrade

Make sure you are the **root** user before starting the upgrade!

Before upgrade FaceStream, you need to do the following:

1. [Create backups](#)
2. [Delete old symbolic link](#)
3. [Unpack the distribution of the new version of FaceStream](#)
4. [Create new symbolic link](#)
5. [Move old data of the PostgreSQL DB](#)
6. [Move old data of the Influx DB](#)
7. [Save user configurations of the Configurator service](#) if they have changed
8. [Set up GPU computing](#), если планируется использовать GPU
9. [Login to VisionLabs registry](#) if authorization was not previously performed
10. [Update the license](#) if necessary
11. [Remove old containers](#)
12. [Launch Influx OSS 2 container](#)
13. [Launch PostgreSQL container](#)
14. [Perform LUNA Configurator DB migration](#)
15. [Launch LUNA Configurator container](#)
16. [Launch LUNA Licenses container](#)

After the steps have been performed, you can start manually or automatically launching LUNA Streams and FaceStream.

## 1.1 Create backups

It is recommended to create the following backups:

- backup of LUNA Streams database (not described in this documentation);
- backup of LUNA PLATFORM services, LUNA Streams service and FaceStream configurations.

Creating backups will enable you to restore in case of any problems during the migration process.

### 1.1.1 Backup of services configurations

Custom configurations for LUNA PLATFORM services (all except the Configurator service), LUNA Streams and FaceStream are automatically migrated using the Configurator service migration mechanism. This backup will not be used during the normal installation of FaceStream.

To create a dump-file, use the following options (may be executed from anywhere on your server):

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

or

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

## 1.2 Delete symbolic link

Delete the symbolic link to the previous minor version directory using the following command:

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

## 1.3 Unpack distribution

It is recommended to move the archive to a pre-created directory for FaceStream and unpack the archive there.

The following commands should be performed under the root user.

Create a directory for FaceStream.

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

Move the archive to the created directory. It is considered that the archive is saved to the “/root” directory.

```
mv /root/facestream_docker_v.5.1.32.zip /var/lib/fs/
```

Go to the directory.

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

Install the unzip utility if it is not installed.

```
yum install unzip
```

Unpack the archive.

```
unzip facestream_docker_v.5.1.32.zip
```

## 1.4 Create symbolic link

Create a symbolic link. The link indicates that the current version of the distribution file is used to run the software package.

```
ln -s facestream_docker_v.5.1.32 fs-current
```

## 1.5 Move data

Move the data for your databases to the directory with new distribution.

It is considered, that you use the default paths for storing databases and buckets.

The example is provided for updating from version v.5.1.30. You should perform these actions for the FaceStream build installed on your server. Change v.5.1.30 in commands below to your currently installed build.

You should copy the data folder of your database from “facestream\_docker\_v.5.1.30” directory to the current root. Thus you can use your data in the new FaceStream build.

### 1.5.1 Move PostgreSQL data

The following step is required if you are using PostgreSQL in Docker container.

Copy the “data” folder:

```
cp -r /var/lib/fs/facestream_docker_v.5.1.30/example-docker/postgresql /var/lib/fs/fs-current/example-docker/
```

### 1.5.2 Move InfluxDB Data

The following step is required if you are using InfluxDB in Docker container.

Copy the “influx” folder with all its buckets:

```
cp -r /var/lib/fs/facestream_docker_v.5.1.30/example-docker/influx /var/lib/fs/fs-current/example-docker/
```

## 1.6 Save user configurations of the Configurator

The configurations of the Configurator service are not automatically migrated, unlike the configurations of all other services.

If your previous LP version was used with non-default Configurator service configurations, back up your /var/lib/fs/fs-current/extras/conf/configurator\_configs/luna\_configurator\_postgres.conf config file in the separate directory on your server.

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

This backup must be mounted to the Configurator service container that is being run.

If you are not sure if the Configurator service configurations have changed, you can compare the created backup with the Configurator configurations from the current distribution using the following command:

```
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 Install GPU dependencies

**Skip this section if you are not going to utilize GPU for your calculations.**

You need to install NVIDIA Container Toolkit to use GPU with Docker containers.

The example of the installation is given below.

```
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
```

Check the NVIDIA Container toolkit operating by running a base CUDA container (this container is not provided in the FaceStream distribution and should be downloaded from the Internet):

```
docker run --rm --gpus all nvidia/cuda:11.4.3-base-centos7 nvidia-smi
```

See the documentation for additional information:

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

Attributes extraction on the GPU is engineered for maximum throughput. The input images are processed in batches. This reduces computation cost per image but does not provide the shortest latency per image.

GPU acceleration is designed for high load applications where request counts per second consistently reach thousands. It won't be beneficial to use GPU acceleration in non-extensively loaded scenarios where latency matters.

### 1.7.1 Actions to launch FaceStream with GPU through Docker Compose

To launch FaceStream with GPU through Docker Compose, it is necessary, in addition to the above actions, to add the `deploy` section in the `facestream` field to the `docker-compose.yml` file.

Before starting the FaceStream container with GPU, it is required to **enable GPU** for calculations in the FaceStream settings using the “`enable_gpu_processing`” parameter (see the “FaceStream configuration” section in the administrator manual).

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

```

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

```

driver - this field specifies the driver for the reserved device(s);

count - this field specifies the number of GPU devices that should be reserved (providing the host holds that number of GPUs);

capabilities - this field expresses both generic and driver specific capabilities. It must be set, otherwise, an error will be returned when deploying the service.

See the documentation for additional information:

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

## 1.8 Login to registry

When launching containers, you should specify a link to the image required for the container launching. This image will be downloaded from the VisionLabs registry. Before that, you should login to the registry.

Login and password can be requested from the VisionLabs representative.

Enter login <username>.

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

After running the command, you will be prompted for a password. Enter password.

In the `docker login` command, you can enter the login and password at the same time, but this does not guarantee security because the password can be seen in the command history.

## 1.9 License activation

To activate/upgrade a license, follow the steps in the license activation manual included in the distribution package.

## 1.10 Remove old containers

Before launching the containers of the current minor version, stop all FaceStream related containers of the previous minor version and third-party software containers.

PostgreSQL and InfluxDB containers can also be removed as their versions can be updated in new build.

PostgreSQL and InfluxDB require restarting even if their containers were not changed. This is related to the transfer of their data folders. See [“Move data”](#).

To delete a container use the next command:

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

where [container\_name] is the service docker container name or ID.

For example, to remove the FaceStream, LUNA Streams, LUNA Configurator and LUNA Licenses containers, use the following command:

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

To see the containers names or IDs, use the following command:

```
docker ps -a
```

It is also recommended to delete old images of the containers to free space. You can use the following command to delete all unused images.

If there is enough space on the server it is recommended to perform this action only after new version of FaceStream is successfully launched.

The command deletes all the unused images, not only the images related to FaceStream.

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



## 1.11 Launch InfluxDB OSS 2 container

InfluxDB 2.0.8-alpine is required to monitor the minimum required LP services (for more information, see the “Monitoring” section in the LUNA PLATFORM administrator manual).

**Note!** If you already have InfluxDB 2.0.8-alpine installed, skip this step.

Use the `docker run` command with these parameters:

```
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 Launch PostgreSQL container

If you already have PostgreSQL installed, skip this step.

Use the following command to launch PostgreSQL.

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

```
--network=host \  
dockerhub.visionlabs.ru/luna/postgis-vmatch:12
```

`-v /var/lib/luna/current/example-docker/postgresql/data:/var/lib/postgresql/data/` - the volume command enables you to mount the “data” folder to the PostgreSQL container. The folder on the server and the folder in the container will be synchronized. The PostgreSQL data from the container will be saved to this directory.

`--network=host` - if you need to change the port for PostgreSQL, you should change this string to `-p 5440:5432`. Where the first port 5440 is the local port and 5432 is the port used inside the container.

### 1.13 LUNA Configurator database migration

The following instruction for migrating the LUNA Configurator service database assumes that the configuration migration revision is already installed in the database. The revision is set using the `configs.migrate head;` script. This script is included in the FaceStream installation manual. If the installation was performed according to the manual, no additional steps are required. Settings will be migrated automatically.

If there is no revision, you should re-create the database structure. See the FaceStream installation manual, section “Initialize LUNA Configurator database”. Then you need to set all the necessary settings manually.

#### 1.13.1 Initialize LUNA Configurator database

When upgrading the LUNA Configurator database with existing settings, you should perform database structure migration and saved settings migration.

Your current database should already have settings migration revision.

Use the `docker run` command with these parameters to create the LUNA Configurator database tables.

```
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.2.1.41 \  
-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;` - upgrades already existing database structure.

`python3 -m configs.migrate head;` - performs settings migrations in LUNA Configurator DB and sets revision for migration. The revision will be required during the upgrade to the new LUNA Configurator build.

## 1.14 Launch LUNA Configurator container

Use the `docker run` command with these parameters to launch 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.2.1.41
```

## 1.15 LUNA Licenses service

### 1.15.1 Specify license settings using Configurator

Follow the steps below to set the settings for [HASP-key](#) or [Guardant-key](#).

#### 1.15.1.1 Specify HASP license settings

**Note.** Perform these actions only if the HASP key is used. See the “[Specify Guardant license settings](#)” section if the Guardant key is used.

To set the license server address, follow these steps:

- go to the Configurator service interface `http://<configurator_server_ip>:5070/`
- specify the “LICENSE\_VENDOR” value in the “Setting name” field and click “Apply Filters”
- set the IP address of the server with your HASP key in the field “server\_address”
- click “Save”

If the license is activated using the HASP key, then two parameters “vendor” and “server\_address” must be specified. If you want to change the HASP protection to Guardant, then you need to add the “license\_id” field.

#### 1.15.1.2 Specify Guardant license settings

**Note.** Perform these actions only if the Guardant key is used. See the “Specify HASP license settings” section if the HASP key is used.

To set the license server address, follow these steps:

- go to the Configurator service interface `http://<configurator_server_ip>:5070/`
- enter the value “LICENSE\_VENDOR” in the “Setting name” field and click “Apply Filters”
- set the IP address of the server with your Guardant key in the “server\_address” field
- set the license ID in the format `0x<your_license_id>`, obtained in the section “Save license ID” in the license activation manual, in the “license\_id” field
- click “Save”

If the license is activated using the Guardant key, then three parameters “vendor”, “server\_address” and “license\_id” must be specified. If you want to change the Guardant protection to HASP, then you need to delete the “license\_id” field.

#### 1.15.2 Launch LUNA Licenses container

Use the following command to launch the service:

```
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.0.7.18
```

## 2 Upgrade FaceStream

Before launching FaceStream, the steps described in the “[Before upgrade](#)” section must be completed.

### 2.1 Migrate settings

To preserve the possibility of using the LUNA Streams user settings from the previous version, you should perform a migration. In the current release, you need to migrate twice - to version 0.5.17, and then to version v.0.6.21.

If you are upgrading from FaceStream v.5.1.18 and below, you must first run the following command for LUNA Streams v.0.5.17, and then for the latest version of LUNA Streams.

FaceStream settings do not require migration in the current release.

```
docker run \
-v /etc/localtime:/etc/localtime:ro \
-v /tmp/logs/streams:/srv/logs \
--entrypoint=/bin/bash \
--rm \
--network=host \
dockerhub.visionlabs.ru/luna/streams-configs:v.0.6.21 \
-c "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 - luna\_configurator  
database address flag

### 2.2 Migrate LUNA Streams database

Run migration script to update the LUNA Streams database structure.

It is recommended that you back up your database before taking any changes.

```
docker run \
-v /etc/localtime:/etc/localtime:ro \
-v /tmp/logs/streams:/srv/logs \
--rm \
--network=host \
dockerhub.visionlabs.ru/luna/luna-streams:v.0.6.21 \
alembic -x luna-config=http://127.0.0.1:5070/1 upgrade head
```

## 2.3 Launch LUNA Streams container

The container is launched with the following command:

```
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.0.6.21
```

To check if the service started correctly, you can perform a GET request <http://127.0.0.1:5160/> version. The response should return the LUNA Streams version v.0.6.21.

## 2.4 Launch FaceStream container

### 2.4.1 Launch FaceStream container using CPU

The container is launched as follows:

```
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.5.1.32
```

For a description of the remaining parameters and launching keys, see the [“Launching keys”](#) section.

To verify that the application was launched correctly, you can perform a GET request `http://127.0.0.1:34569/version`. The response should return the FaceStream `{{Version_of_FS}}`.

## 2.4.2 Launch FaceStream container using GPU

**Note.** Use this command only if you are going to use FaceStream with GPU.

Before launching FaceStream in GPU mode, additional dependencies should be installed (see [“Install GPU dependencies”](#) section).

Before starting the FaceStream container with GPU, it is required to **enable GPU** for calculations in the FaceStream settings using the `“enable_gpu_processing”` parameter (see the [“FaceStream configuration”](#) section in the administrator manual).

```
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.5.1.32
```

`--gpus device=0` - the parameter specifies the used GPU device and enables GPU utilization. A single GPU can be utilized per FaceStream instance. Multiple GPU utilization per instance is not available.

For a description of the remaining parameters and launching keys, see the [“Launching keys”](#) section.

To verify that the application was launched correctly, you can perform a GET request `http://127.0.0.1:34569/version`. The response should return the FaceStream `{{Version_of_FS}}`.

### 3 Additional information

This section provides the following additional information:

- [Useful commands for working with Docker](#)
- [Launching keys description](#)
- [Configuring Docker log rotation](#)



### 3.1 Docker commands

#### 3.1.1 Show containers

To show the list of launched Docker containers use the command:

```
docker ps
```

To show all the existing Docker containers use the command:

```
docker ps -a
```

#### 3.1.2 Copy files to container

You can transfer files into the container. Use the `docker cp` command to copy a file into the container.

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

#### 3.1.3 Enter container

You can enter individual containers using the following command:

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

To exit the container, use the command:

```
exit
```

#### 3.1.4 Images names

You can see all the names of the images using the command

```
docker images
```

#### 3.1.5 Show container logs

You can view the container logs with the following command:

```
docker logs <container_name>
```

### 3.1.6 Delete image

If you need to delete an image:

- run the `docker images` command
- find the required image, for example: `dockerhub.visionlabs.ru/luna/v.5.1.32`
- copy the corresponding image ID from the IMAGE ID, for example, "61860d036d8c"
- specify it in the deletion command:

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

Delete all the existing images:

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

### 3.1.7 Stop container

You can stop the container using the command:

```
docker stop <container_name>
```

Stop all the containers:

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

### 3.1.8 Delete container

If you need to delete a container:

- run the `docker ps` command
- stop the container (see [Stop container](#))
- find the required image, for example: `dockerhub.visionlabs.ru/luna/v.5.1.32`
- copy the corresponding container ID from the CONTAINER ID column, for example, "23f555be8f3a"
- specify it in the deletion command:

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

Delete all the containers:

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

## 3.2 Launching keys

To launch FaceStream with Configurator, the keys are set using environment variables:

--env= - this parameter sets the environment variables required to start the container. The following basic values are specified:

- CONFIGURATOR\_HOST=127.0.0.1 - host on which the Configurator service is running. The local host is set if the container is running on the same server where the Configurator is running.
- CONFIGURATOR\_PORT=5070 - listening port for the Configurator service. By default, port 5070 is used.
- PORT=34569 - port where FaceStream will listen.
- STREAMS\_ID="" - tag specifies a list of stream IDs that will be requested from LUNA Streams for processing. Other streams will be filtered. The "stream\_id" parameter is given in response to the "create stream" request.

If the value is "" or the STREAMS\_ID tag is not set, then FaceStream will take all existing "stream\_id" from the queue.

If a non-existent value is set, an error about an incorrect UUID will be indicated when launching FaceStream.

By default, the value equals "".

To use the key, the CONFIGURATOR\_HOST and CONFIGURATOR\_PORT variables should be specified.

- STREAMS\_NAME="" - list of streams names sets in this tag. Streams names are set using the "name" parameter at the time of their creation ("create streams" request). Streams with these names will be requested from LUNA Streams for processing. Other streams will be filtered.

Otherwise, the principle of operation is similar to the "STREAMS\_ID" tag.

- GROUPS\_ID="" and GROUPS\_NAME="" - tags specify a list of group IDs or a list of group names. The parameters "group\_id" or "group\_name" are set during stream creation ("create stream" request). Streams with these parameters will be requested from LUNA Streams for processing. Other streams will be filtered.

If the value is "" or the GROUPS\_ID/GROUPS\_NAME tags are not set, then FaceStream will not filter streams by groups.

If a non-existent value is set, an error about an incorrect UUID will be indicated when launching FaceStream.

By default, the value equals "".

To use the keys, the CONFIGURATOR\_HOST and CONFIGURATOR\_PORT variables should be specified.

You can set multiple values for “STREAMS\_NAME”, “STREAMS\_ID”, “GROUPS\_NAME” and “GROUPS\_ID” tags. Syntax example: `--env=STREAMS_ID="037f3196-c874-4eca-9d7c-91fd8dfc95934caf7cf7-dd0d-4ad5-a35e-b263e742e28a"`

- `CONFIGS_ID=""` - tag is used to set a LUNA Configurator tag, which relates to the FaceStream main configurations. The same tag should be set for “TRACK\_ENGINE\_CONFIG” and “FACE\_STREAM\_CONFIG”.

If the value is set to "" then the “TRACK\_ENGINE\_CONFIG” and “FACE\_STREAM\_CONFIG” records will be used by default. If the record by default does not exist or has an invalid JSON syntax, the configuration file from the distribution package will be used.

By default, the value equals "".

To use the key, the `CONFIGURATOR_HOST` and `CONFIGURATOR_PORT` variables should be specified.

- `CONFIG_RELOAD = 1` - tag that enables checking for changes in the “FACE\_STREAM\_CONFIG” section of the LUNA Configurator service and takes the following values:
  - 1 - change tracking is enabled, if there are changes in the configuration, all FaceStream containers will be automatically restarted;
  - 0 - change tracking is disabled.

By default, the value equals 1.

- `PULLING_TIME = 10` - tag that sets the period for receiving new parameters from the “FACE\_STREAM\_CONFIG” section of the LUNA Configurator service in the range [1...3600] sec. Used in conjunction with the `CONFIG_RELOAD` tag.

By default, the value equals 10.

`--device=` - this parameter is required to specify the address to the USB device. The address must be specified in the stream source when it is created. Example: `--device=/dev/video0`.

See how FaceStream works with LUNA Configurator in the section “Use FaceStream with LUNA Configurator” of the administrator manual.

### 3.2.1 Description of container launch parameters

`docker run` - command to launch the selected image as a new container.

`-v` - enables you to load the contents of the server folder into the volume of the container. This way the content is synchronized.

`-v /var/lib/fs/fs-current/extras/conf/configs/faceengine.conf:/srv/facestream/data/faceengine.conf \` - this parameter enables you to use the FaceEngine settings from the configuration file “faceengine.conf”.

`-v /var/lib/fs/fs-current/extras/conf/configs/runtime.conf:/srv/facestream/data/runtime.conf \` - this parameter enables you to mount the runtime configuration file into the FaceStream container. Before changing the default settings, you need to consult with VisionLabs specialists.

`--network=host` - this parameter specifies that there is no network simulation and a server network is used. If you need to change the port for third-party containers, replace this line with `-p 5440:5432`. Here, the first port 5440 is the local port, and 5432 is the port used in the container.

`/etc/localtime:/etc/localtime:ro` - sets the current time zone used by the container system.

`--name=facestream` - this parameter specifies the name of the container to be launched. The name must be unique. If a container with the same name already exists, an error will occur.

`--restart=always` - this parameter defines the restart policy. Daemon always restarts the container regardless of the completion code.

`--detach=true` - running the container in the background.

### 3.3 Docker log rotation

To limit the size of logs generated by Docker, you can set up automatic log rotation. To do this, add the following data to the `/etc/docker/daemon.json` file:

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

This will allow Docker to store up to 5 log files per container, with each file being limited to 100MB.

After changing the file, you need to restart Docker:

```
systemctl reload docker
```

The above changes are the default for any newly created container, they do not apply to already created containers.