

# VisionLabs FaceStream

Installation manual

v.5.1.13

## Contents

<b>Glossary</b>	<b>4</b>
<b>1 Introduction</b>	<b>5</b>
<b>2 System requirements</b>	<b>7</b>
2.1 Minimum requirements . . . . .	7
<b>3 Software requirements</b>	<b>8</b>
<b>General information</b>	<b>9</b>
Sequence of actions before launching . . . . .	9
Sequence of actions before manual launching . . . . .	10
Sequence of actions before automatic launching . . . . .	10
<b>4 Before launching</b>	<b>11</b>
4.1 Unpacking . . . . .	11
4.2 Symbolic link creation . . . . .	11
4.3 Enable logging to file for FaceStream . . . . .	12
4.4 Docker installation . . . . .	12
4.5 Docker Compose installation . . . . .	13
4.6 GPU dependencies installation . . . . .	14
4.6.1 Actions to launch FaceStream with GPU through Docker Compose . . . . .	14
4.7 Login to registry . . . . .	15
4.8 License key activation . . . . .	16
4.8.1 License key activation when LP is launched . . . . .	16
4.8.2 License key activation when LP is not launched . . . . .	16
4.8.2.1 Install HASP utility . . . . .	17
4.8.2.2 Configure HASP utility . . . . .	17
4.8.2.3 Add LP vendor library . . . . .	18
4.8.2.4 Create fingerprint for LUNA PLATFORM . . . . .	18
4.8.2.5 Add a license file manually using user interface . . . . .	18
4.8.2.6 Specify license server address . . . . .	19
4.8.2.7 Delete LP hasp utility . . . . .	20
4.9 Launching LP services . . . . .	21
4.9.1 InfluxDB OSS 2 container launching . . . . .	21
4.9.2 PostgreSQL container launching . . . . .	21
4.9.3 LUNA Configurator database creating . . . . .	22
4.9.4 LUNA Configurator DB initialization . . . . .	22
4.9.5 LUNA Configurator container launching . . . . .	23

4.9.6	LUNA Licenses container launching . . . . .	24
4.9.6.1	Specifying LUNA Licenses server address . . . . .	24
<b>5</b>	<b>FaceStream manual launching</b>	<b>25</b>
5.1	Upload settings to LUNA Configurator . . . . .	25
5.1.1	Upload LUNA Streams settings . . . . .	25
5.1.2	Upload FaceStream settings . . . . .	26
5.2	LUNA Streams database preparing . . . . .	26
5.2.1	LUNA Streams database creating . . . . .	26
5.2.2	LUNA Streams DB initialization . . . . .	27
5.3	Command for launching LUNA Streams container . . . . .	27
5.4	Commands for launching FaceStream container . . . . .	28
5.4.1	Command for launching container using CPU . . . . .	28
5.4.2	Command for launching container using GPU . . . . .	28
5.4.3	Launching keys . . . . .	29
5.4.4	Description of container launch parameters . . . . .	31
<b>6</b>	<b>Launching FaceStream using Docker Compose</b>	<b>32</b>
6.1	Launching FaceStream command . . . . .	33
<b>7</b>	<b>Docker commands</b>	<b>34</b>
7.1	Show containers . . . . .	34
7.2	Copy files to container . . . . .	34
7.3	Enter container . . . . .	34
7.4	Images names . . . . .	34
7.5	Show container logs . . . . .	34
7.6	Delete image . . . . .	35
7.7	Stop container . . . . .	35
7.8	Delete container . . . . .	35

## 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).
Bestshot	Best shot is selected from all frames of the track. The main conditions for best shot selection are appropriate quality and the presence of a face with best aspect angle. Such conditions are set through FaceStream configuration.
Detection	FaceStream entity that contains the coordinates of face or body and the estimated value of the object that determines the bestshot.
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.
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.

# 1 Introduction

This document describes:

- the manual process of launching the FaceStream application and the required dependencies using Docker.
- the automatic FaceStream application deployment with the LUNA Streams service using Docker Compose.

Before launching, it is necessary to explore the general information and sequence of actions.

FaceStream licensing is managed by a special parameter of the LUNA PLATFORM 5 license key, which determines the number of streams for LUNA Streams operation. Thus, a LUNA PLATFORM 5 license is required for FaceStream to work.

If FaceStream is launched without LP services launched in advance, then you need to request a new LUNA PLATFORM 5 license with a parameter that determines the streams number for LUNA Streams operation, and go through the full activation process.

If FaceStream is launched with LP services launched in advance, then you need to make sure that the license key contains a parameter that determines the streams number for LUNA Streams operation. If LUNA Licenses is launched on another server, you will also need to specify the address of this server in the LUNA Streams settings in the LUNA Configurator service.

Docker enables you to create a container that already has the required service, environment for the service, and a minimum number of required tools.

Each LUNA PLATFORM and FaceStream service has its own image. Docker images are the basis of containers. Each container includes libraries required for running services and execution parameters for use within a container runtime.

Docker Compose is used for automated containers deployment. The Docker Compose scenario from this distribution is used for deploying the LUNA Streams and FaceStream on a single server.

It is considered that launching is performed on the server with CentOS OS, where FaceStream was not installed.

Docker images of the LP and FaceStream containers are required for the FaceStream launching. 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.

Firewall and SELinux should be manually configured on the server by the administrator. Their configuration is not described in this document.

This document does not include a tutorial for Docker usage. Please refer to the Docker documentation to find more information about Docker:

<https://docs.docker.com>

This document includes examples of FaceStream deployment in a minimal 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 Putty (when you remotely connect to the server). The provided commands were tested with these utilities only. The use of other shells or emulators can lead to errors when executing commands.

For more information on general operation and application settings, see the FaceStream administrator manual.

## 2 System requirements

### 2.1 Minimum requirements

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:

#### **Video card resources usage**

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 470.103.01.

Now only one video card is supported per FaceStream instance.

#### **AVX2 instructions usage**

CPU with AVX2 support is required.

The system automatically detects available instructions and runs best performance.

### 3 Software requirements

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

Docker should be installed for containers launch. To upload settings to the LUNA Configurator service, Python version 2.x or 3.x is required.



## General information

It is recommended to read and understand this document. It will help you to find out what components FaceStream consists of and what tasks they solve.

Deploy should be performed in the order specified in the document.

All the procedures in the following manual are described for CentOS. If it is required to deploy dockers on any other OS, please refer to the Docker Compose documentation for information:

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

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: 11.2

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

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

## Sequence of actions before launching

Before launching FaceStream, you should perform a number of actions described in the “[Before launching](#)” section.

Some of the actions are mandatory, and some of the actions depend on whether the LUNA PLATFORM services are launched and whether the license is activated.

## Sequence of actions before manual launching

If the LUNA PLATFORM services are not launched and the license is not activated at the time of the FaceStream launch, then the following steps should be performed before manual or automatic launch:

- Activate the license (see the [“License key activation when LP is not launched”](#) section);
- Run the required components (see the [“Launching LP Services”](#) section):
  - Influx OSS 2 container
  - PostgreSQL container
  - LUNA Configurator container
  - LUNA Licenses container

After the completed steps, you can proceed with manual or automatic launch of LUNA Streams and FaceStream.

## Sequence of actions before automatic launching

If the LUNA PLATFORM services are launched and the LUNA PLATFORM license is activated at the time of the FaceStream launch, then you need to make sure that the license specifies a parameter that determines the number of streams to be processed by the LUNA Streams service and specify the address of the server with the LUNA Licenses service launched in the LUNA Configurator service (see the [“License key activation when LP is launched”](#)).

After the completed steps, you can proceed with manual or automatic launch of LUNA Streams and FaceStream.

## 4 Before launching

Before launching FaceStream, you need to complete a number of additional steps.

### 4.1 Unpacking

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.5.1.13.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.5.1.13.zip
```

You will need to configure FaceStream before launching it.

The unpacked archive includes all the configuration files required for the FaceStream launch. The configuration parameters description is given further in this document.

### 4.2 Symbolic link creation

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.5.1.13 fs-current
```

### 4.3 Enable logging to file for FaceStream

**Note.** Use the steps below only if you need to save logs to a file. By default, logs are output to the console. To view the logs from the console, use the `docker logs <container_name>` command.

If necessary, you can write the FaceStream logs in separate files. To do this, you should first create directory to save logs.

The log directory is created with the following command:

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

The directory for storing logs can be changed if necessary.

To enable logging, you should enable logging to a file in the FaceStream settings. To do this, you need to set the value of the “mode” parameter to “l2f” (output logs only to file) or “l2b” (output logs and file to the console).

To enable logging, you need to run the following command when starting the container:

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

Data from the specified folder is added to the Docker container when it is launched. All the data from the specified Docker container folder is saved to this directory.

To write LUNA service logs, follow the same steps.

By default, only system warnings are displayed in the FaceStream logs. By setting the “severity” parameter, you can enable error output (see the parameter description in the administrator manual).

### 4.4 Docker installation

Docker is required for launching of the FaceStream container.

The Docker installation is described in the official documentation:

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

You do not need to install Docker if you already have an installed Docker of the latest version on your server.

Quick installation commands are listed below.

Check the official documentation for updates if you have any problems with the installation.

Install dependencies.

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

Add repository.

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

Install Docker.

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

Launch Docker.

```
systemctl start docker
```

```
systemctl enable docker
```

Check Docker status.

```
systemctl status docker
```

## 4.5 Docker Compose installation

**Note.** Install Docker Compose only if you are going to use the FaceStream automatic launching script.

Install 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
```

See the official documentation for details:

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

## 4.6 GPU dependencies installation

**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-base 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.

### 4.6.1 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 `handlers` field to the `docker-compose.yml` file.

```
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 - 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>.

## 4.7 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.

Enter login <username>.

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

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

The login and password are received from VisionLabs.

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.

## 4.8 License key activation

### 4.8.1 License key activation when LP is launched

If the LUNA PLATFORM services are already launched and the license with a parameter that determines the streams number for LUNA Streams operation is already activated, then you need to make sure that the current LUNA PLATFORM key contains this parameter. The information can be provided by VisionLabs specialists.

If this parameter is not contained in the key, then you need to request a new key and contact VisionLabs specialists for advice on updating the license key.

If LUNA Streams is launched on a server other than the one on which LUNA Licenses is launched, then you should perform the steps described in the section [“Specifying LUNA Licenses server address”](#).

### 4.8.2 License key activation when LP is not launched

If the LUNA PLATFORM services are not launched or the launch is performed using Docker Compose, then it is assumed that the license has not yet been activated and it is necessary to request a new LUNA PLATFORM 5 license with a parameter determining the streams number for LUNA Streams operation, and go through the full LUNA PLATFORM 5 licensing process.

The HASP service is used for LUNA PLATFORM licensing. Without a license, you will be unable to run and use LUNA services and create streams for FaceStream.

There is a HASP key that enables you to use LUNA PLATFORM and create streams in FaceStream. It uses the `haspplib_x86_64_30147.so` vendor library.

You can find the vendor libraries in the `“/var/hasplm/”` directory.

License keys are provided by VisionLabs separately upon request. The utilized Liveness version is specified in the LUNA PLATFORM license key.

A network license is required to use LUNA PLATFORM and FaceStream in Docker containers.

The license key is created using the fingerprint. The fingerprint is created based on the information about hardware characteristics of the server. Therefore, the received license key will only work on the same server where the fingerprint was obtained. There is a possibility that a new license key will be required when you perform any changes on the license server.

Follow these steps:

- Install HASP utility on your server. HASP utility is usually installed on a separate server;
- Start the HASP utility;
- Create the fingerprint of your server and send it to VisionLabs;
- Activate your key, received from VisionLabs;
- Specify your HASP server address in a special file.



The Sentinel Keys tab of the user interface ( <server\_host\_address>:1947) shows activated keys.

LP uses HASP utility of a certain version. If an older version of HASP utility is installed, it is required to delete it before installation of a new version. See [“Delete LP hasp utility”](#).

#### 4.8.2.1 Install HASP utility

Go to the HASP directory.

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

Install HASP utility on you server.

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

Launch HASP utility.

```
systemctl daemon-reload
```

```
systemctl start aksusbd
```

```
systemctl enable aksusbd
```

```
systemctl status aksusbd
```

#### 4.8.2.2 Configure HASP utility

You can configure the HASP utility using the “/etc/hasplm/hasplm.ini” file.

**Note!** You do not need to perform this action if you already have the configured INI file for the HASP utility.

Delete the old file if necessary.

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

Copy the INI file with configurations. Its parameters are not described in this document.

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

#### 4.8.2.3 Add LP vendor library

Copy LP vendor library (x32 and x64). This library is required for using LP license key.

```
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/
```

Remove old version LP libraries if present:

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

Restart the utility

```
systemctl restart aksusbd
```

#### 4.8.2.4 Create fingerprint for LUNA PLATFORM

Go to the HASP directory.

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

Add permissions to the script.

Run the script.

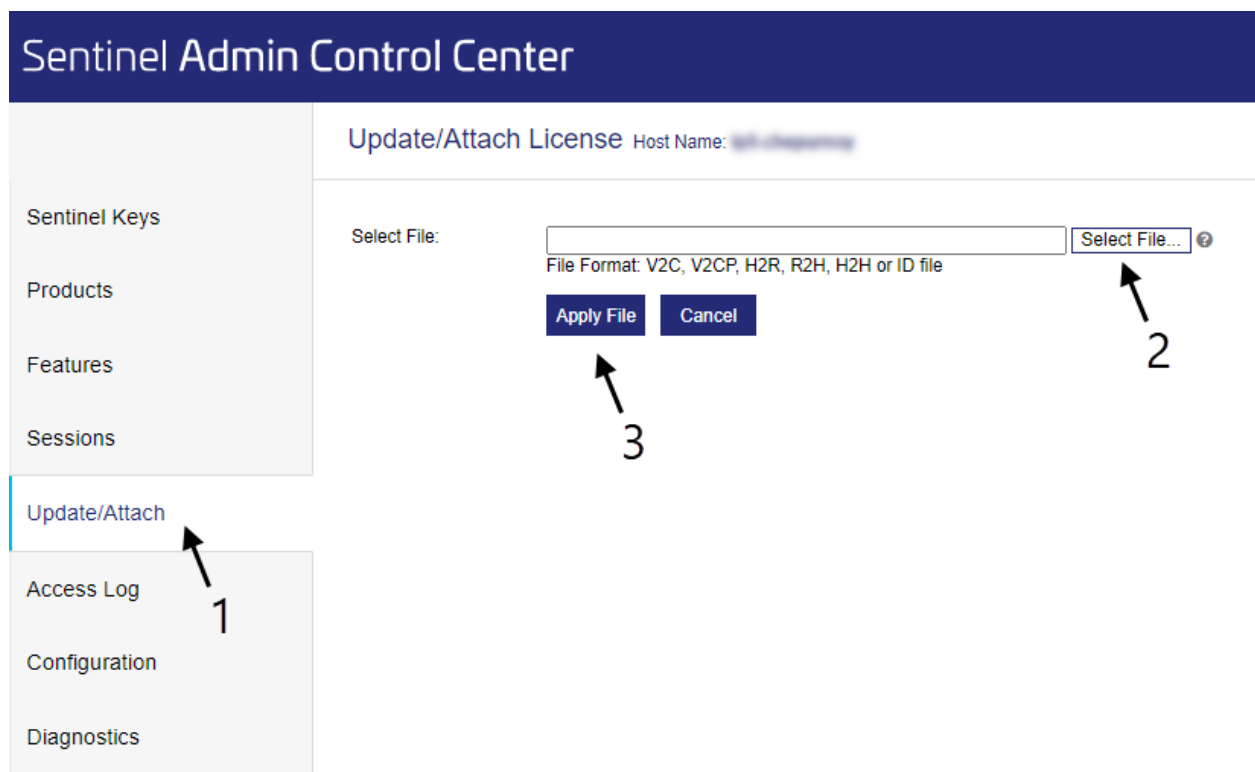
```
./LicenseAssist fingerprint > fingerprint_30147.c2v
```

The fingerprint is saved to file “fingerprint\_30147.c2v”.

**Send the file to VisionLabs. Your license key will be created using this fingerprint.**

#### 4.8.2.5 Add a license file manually using user interface

- Go to: <host\_address>:1947 (if access is denied check your Firewall/ SELinux settings (the procedure is not described in this document));
- Select the **Update/Attach** at the left pane;
- Press the “Select File...” button and select a license file(s) in the appeared window;
- Press the “Apply File” button.



**Figure 1:** License file is added manually

#### 4.8.2.6 Specify license server address

Specify your license service IP address in the configuration file in the following directory:

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

Change address to the HASP server in the following documents:

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

Change the server address in “hasp\_30147.ini” file.

```
serveraddr = <HASP_server_address>
```

The “hasp\_30147.ini” file is used by the Licenses service upon its container launch. It is required to restart the launched container when the server is changed.

HASP\_server\_address - the IP address of the server with your HASP key. You must use an IP address, not a server name.

#### 4.8.2.7 Delete LP hasp utility

This action is performed to delete HASP utility.

Stop and disable the utility.

```
systemctl stop aksusbd
```

```
systemctl disable aksusbd
```

```
systemctl daemon-reload
```

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

## 4.9 Launching LP services

**Note.** Before launching LP services, make sure that the license is activated (see the ["License key activation when LP is not launched"](#) section).

The launch of LUNA PLATFORM services is necessary if the following additional components are not launched:

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

If the above components are already launched, skip this section.

### 4.9.1 InfluxDB OSS 2 container launching

InfluxDB 2.0.8-alpine is required for LP monitoring purpose (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
```

### 4.9.2 PostgreSQL container launching

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-vlmatch: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.

#### 4.9.3 LUNA Configurator database creating

Create a database for LUNA Configurator:

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

Allow the user to log in to the database:

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

#### 4.9.4 LUNA Configurator DB initialization

Use the `docker run` command with these parameters to create the 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 \
-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.2.0.72 \
-c "python3 ./base_scripts/db_create.py; cd /srv/luna_configurator/configs/
  configs/; python3 -m configs.migrate --config /srv/luna_configurator/
  configs/config.conf head; cd /srv; python3 ./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  
- enables you to specify the path to the dump file with LP configurations.

./base\_scripts/db\_create.py; - creates database structure.

python3 -m configs.migrate head; - performs settings migrations in Configurator DB and sets revision for migration.

--dump-file /srv/luna\_configurator/used\_dumps/platform\_settings.json - updates settings in the Configurator DB with values from the provided file.

#### 4.9.5 LUNA Configurator container launching

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 \
--name=luna-configurator \
--restart=always \
--detach=true \
--network=host \

```

```
dockerhub.visionlabs.ru/luna/luna-configurator:v.2.0.72
```

#### 4.9.6 LUNA Licenses container launching

Make sure that you have specified the license server address in the “hasp\_30147.ini” file. See section [“Specify license server address”](#).

Add the access right for the “luna” user to the “hasp\_redirect” directory.

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

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 /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.0.3.79
```

##### 4.9.6.1 Specifying LUNA Licenses server address

**Note.** Perform the following actions only if you launch LUNA Streams not on a server with LUNA Licenses. Otherwise, you can skip this step.

Go to the user interface of the LUNA Configurator service and select “luna-streams” from the “Service name” drop-down list.

In the “LUNA\_LICENSES\_ADDRESS” section, specify the address of the server where the LUNA Licenses service is launched.



## 5 FaceStream manual launching

Before manual launching FaceStream, the following actions should be performed:

- The license key is activated.  
  
If the LP services are not launched and the LP license was not activated at the start of the FaceStream launch, see the [“License key activation when LP is not launched”](#) section.  
  
If LP services are launched and the LP license was activated at the start of the FaceStream launch, see the [“License key activation when LP is launched”](#) section.
- The required components are launched (see the [“Launching LP services”](#) section):
  - Influx OSS 2 container
  - PostgreSQL container
  - LUNA Configurator container
  - LUNA Licenses container

### 5.1 Upload settings to LUNA Configurator

The main settings of LUNA Streams and FaceStream should be set in the Configurator service after its launch. The exception is the FaceEngine settings, which are set in the configuration file “faceengine.conf” and transferred during the launch of the FaceStream container.

If necessary, you can use configuration files instead of the Configurator service settings and transfer them during container launching (for more information, see the “Use FaceStream with configuration files” section of the administrator manual).

FaceStream and LUNA Streams settings are uploaded into the Configurator in different ways.

#### 5.1.1 Upload LUNA Streams settings

To upload LUNA Streams settings into the Configurator service, you should use the configuration migration mechanism.

```
docker run \  
-v /etc/localtime:/etc/localtime:ro \  
--rm \  
--entrypoint=' ' \  
--network=host \  
dockerhub.visionlabs.ru/luna/streams-configs:v.0.5.5 \  
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

### 5.1.2 Upload FaceStream settings

FaceStream settings are located in a special file “facestream\_dump.json”. To upload these settings into the Configurator service, you should use the “load\_dump.py” script.

The “load\_dump.py” script requires Python version 2.x or 3.x. If version 2.x is installed, then the script should be run using the `python` command. If version 3.x is installed, then the script should be run using the `python3` command.

- Go to the directory with script and dump file:

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

- Run the script to upload FaceStream settings into the Configurator service, specifying the installed version of Python (the command below gives an example of running the script for Python version 2.x):

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

All necessary parameters will be automatically added to the Configurator service.

## 5.2 LUNA Streams database preparing

To launch FaceStream, you need to launch the LUNA Streams service by creating and initializing a database for it. This service is not included in the LUNA PLATFORM 5 distribution, so it should be launched separately.

### 5.2.1 LUNA Streams database creating

**Note.** Run the commands below only if you have not previously launched the PostgreSQL container as described in [“PostgreSQL container launching”](#). When performing the PostgreSQL container launch command, the LUNA Streams database is automatically created.

Create a database for LUNA Streams:

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

Allow the user to log in to the database:

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

Activate PostGIS:

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

### 5.2.2 LUNA Streams DB initialization

Initialize the data in the LUNA Streams database:

```
docker run -v /etc/localtime:/etc/localtime:ro \
--rm \
--network=host \
dockerhub.visionlabs.ru/luna/luna-streams:v.0.5.5 \
python3 ./base_scripts/db_create.py --luna-config http://localhost:5070/1
```

### 5.3 Command for launching 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 \
--name=luna-streams \
--restart=always \
--detach=true \
--network=host \
dockerhub.visionlabs.ru/luna/luna-streams:v.0.5.5
```

## 5.4 Commands for launching FaceStream container

### 5.4.1 Command for launching 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 /etc/localtime:/etc/localtime:ro \  
--env=PORT=34569 \  
--detach=true \  
--restart=always \  
--name=facestream \  
--network=host \  
dockerhub.visionlabs.ru/luna/facestream:v.5.1.13
```

For a description of the remaining parameters and launching keys, see the relevant sections below.

The list of streams is available at <http://127.0.0.1:34569/api/1/streams/>. Viewing the stream in the browser is available at [http://127.0.0.1:34569/api/1/streams/preview/<stream\\_id>](http://127.0.0.1:34569/api/1/streams/preview/<stream_id>).

### 5.4.2 Command for launching 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 “[GPU dependencies installation](#)” 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 /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.5.1.13
```

--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 relevant sections below.

The list of streams is available at <http://127.0.0.1:34569/api/1/streams/>. Viewing the stream in the browser is available at [http://127.0.0.1:34569/api/1/streams/preview/<stream\\_id>](http://127.0.0.1:34569/api/1/streams/preview/<stream_id>).

### 5.4.3 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.

#### 5.4.4 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”.

`--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.

## 6 Launching FaceStream using Docker Compose

Before automatic launching FaceStream, the following actions should be performed:

- The license key is activated.  
  
If the LP services are not launched and the LP license was not activated at the start of the FaceStream launch, see the “[License key activation when LP is not launched](#)” section.  
  
If LP services are launched and the LP license was activated at the start of the FaceStream launch, see the “[License key activation when LP is launched](#)” section.
- The required components are launched (see the “[Launching LP services](#)” section):
  - Influx OSS 2 container
  - PostgreSQL container
  - LUNA Configurator container
  - LUNA Licenses container

Additional notes about Docker Compose script. The script:

- loads LUNA Streams and FaceStream settings into Configurator
- creates and initializes the LUNA Streams database
- launches the LUNA Streams and FaceStream
- is tested using the default LUNA Streams and FaceStream configurations.
- is not intended to be used for FaceStream scaling:
  - It is not used for the deployment of FaceStream services on several servers.
  - It is not used for deployment and balancing of several FaceStream services on a single service.
- supports GPU utilization for FaceStream calculations.
- does not provide the possibility to use external database for LUNA Streams already installed on the server.
- does not perform migrations from previous FaceStream versions and updates from the previous FaceStream build.

See the “docker-compose.yml” file and other files in the “example-docker” directory for the information about launched services and performed actions.

You can write your scenario that deploys and configures all the required services. This document does not include information about scenario creation or tutorial for Docker usage. Please refer to the Docker documentation to find more information about Docker and Docker Compose:

<https://docs.docker.com>



## 6.1 Launching FaceStream command

Go to the Docker Compose folder:

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

Make sure that FS container are not launched before executing the script. An error will occur if you try to run a container with the same name as an existing container. If one or several LP containers are launched, you should stop them using the `docker container rm -f <container_name>` command. To stop all the containers, use `docker container rm -f $(docker container ls -aq)`.

To launch FaceStream with GPU using Docker Compose, you need to follow the steps in [“GPU dependencies installation”](#).

Launch Docker Compose:

You should be logged in the VisionLabs registry (see section [“Login to registry”](#))

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

--configurator\_address=127.0.0.1 - LUNA Configurator service address

Check the state of launched Docker containers.

```
docker ps
```

The list of streams is available at <http://127.0.0.1:34569/api/1/streams/>. Viewing the stream in the browser is available at [http://127.0.0.1:34569/api/1/streams/preview/<stream\\_id>](http://127.0.0.1:34569/api/1/streams/preview/<stream_id>).

## 7 Docker commands

### 7.1 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
```

### 7.2 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>
```

### 7.3 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
```

### 7.4 Images names

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

```
docker images
```

### 7.5 Show container logs

You can view the container logs with the following command:

```
docker logs <container_name>
```

## 7.6 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.5.1.13`
- 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)
```

## 7.7 Stop container

You can stop the container using the command:

```
docker stop <container_name>
```

Stop all the containers:

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

## 7.8 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.5.1.13`
- 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)
```