

VisionLabs FaceStream

Installation manual

v.5.1.23

Contents

Glossary	4
System requirements	6
LP services and third-party applications	6
Processors	7
CPU	7
GPU	7
Introduction	9
1 Before launch	10
1.1 Unpack distribution	11
1.2 Create symbolic link	11
1.3 Install Docker	11
1.4 Install Docker Compose	12
1.5 Choose logging method	13
1.5.1 Logging to stdout	13
1.5.2 Logging to file	13
1.6 Install GPU dependencies	14
1.6.1 Actions to launch FaceStream with GPU through Docker Compose	14
1.7 Login to registry	15
1.8 Check for license	16
2 Launch FaceStream	17
2.1 Launch FaceStream manually	18
2.1.1 Upload settings to LUNA Configurator	18
2.1.1.1 Upload LUNA Streams settings	18
2.1.1.2 Upload FaceStream settings	18
2.1.2 Prepare LUNA Streams database	19
2.1.2.1 Create LUNA Streams database	19
2.1.2.2 Initialize LUNA Streams database	19
2.1.3 Launch LUNA Streams container	20
2.1.4 Launch FaceStream container	20
2.1.4.1 Launch FaceStream container using CPU	20
2.1.4.2 Launch FaceStream container using GPU	21
2.2 Launch FaceStream using Docker Compose	22
2.2.1 Launch FaceStream command	22
3 Next actions	24

4	Additional information	25
4.1	Docker commands	26
4.1.1	Show containers	26
4.1.2	Copy files to container	26
4.1.3	Enter container	26
4.1.4	Images names	26
4.1.5	Show container logs	26
4.1.6	Delete image	27
4.1.7	Stop container	27
4.1.8	Delete container	27
4.2	Launching keys	29
4.2.1	Description of container launch parameters	30
4.3	Logging to server	32
4.3.1	Create logs directory	32
4.3.2	Logging activation	32
4.3.2.1	LUNA Streams logging activation	32
4.3.2.2	FaceStream logging activation	32
4.3.3	Mounting directories with logs when starting services	33
4.4	Docker log rotation	34

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	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 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.
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**: 11.2

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 470.103.01.

Now only one video card is supported per FaceStream instance.

Introduction

To launch FaceStream using this documentation, you must have LUNA PLATFORM launched in accordance with the LUNA PLATFORM installation manual. If necessary, you can launch FaceStream by first launching the minimum required services LUNA Configurator, LUNA Licenses, PostgreSQL, and InfluxDB (see the “Installation manual without LP launched” document).

This document describes:

- the manual process of launching the FaceStream application and LUNA Streams service 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. The license must be activated before launching FaceStream.

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.

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 in a program for working with network protocols (when you remotely connect to the server), for example, Putty.

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

1 Before launch

Make sure you are the **root** user before launch!

Before launch FaceStream, you need to do the following:

1. [Unpack the FaceStream distribution](#)
2. [Create symbolic link](#)
3. [Install Docker](#)
4. [Install Docker Compose](#) if you plan to start FaceStream using Docker Compose script
5. [Choose logging method](#)
6. [Set up GPU computing](#) if you plan to use GPU
7. [Login to VisionLabs registry](#)
8. [Make sure that the license contains a parameter that determines the number of streams to be processed by the LUNA Streams service](#)

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

1.1 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.5.1.23.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.5.1.23.zip
```

1.2 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.5.1.23 fs-current
```

1.3 Install Docker

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 20.10.8 on your server. Not guaranteed to work with higher versions of Docker.

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
```

1.4 Install Docker Compose

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/>

1.5 Choose logging method

There are two methods to output logs

- standard log output (stdout);
- log output to a file.

Log output settings for LUNA PLATFORM services and LUNA Streams service are set in section `<SERVICE_NAME>_LOGGER` of LUNA Configurator service.

Log output settings for FaceStream are set in the settings `logging` of section `FACE_STREAM_CONFIG` of LUNA Configurator service.

If necessary, you can use both methods of displaying logs.

1.5.1 Logging to stdout

This method is used by default and requires no further action.

It is recommended to configure Docker log rotation to limit log sizes (see [“Docker log rotation”](#)).

1.5.2 Logging to file

Note. When you enable saving logs to a file, you should remember that logs occupy a certain place in the storage, and the process of logging to a file negatively affects system performance.

To use this method, you need to perform the following additional actions:

- **before launching the services:** create directories for logs on the server;
- **after launching the services:** activate log recording and set the location of log storage inside LP service containers;
- **during the launch of services:** configure synchronization of log directories in the container with logs on the server using the `volume` argument at the start of each container.

In the Docker Compose script, synchronization of directories with folders **is not configured**. You need to manually add folder mounting to the `docker-compose.yml` file.

See the instructions for enabling logging to files in the [“Logging to server”](#) section.

1.6 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-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.

1.6.1 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: ${REGISTRY_ADDRESS}:${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>.

1.7 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.

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.

1.8 Check for license

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.

2 Launch FaceStream

There are two ways to launch FaceStream - manual and automatic using a Docker Compose script.

Use the hyperlinks below to go to the instructions for the required launch method:

- [manual way to launch FaceStream](#)
- [automatic way to launch FaceStream](#)

2.1 Launch FaceStream manually

Note. Perform these actions only if you are going to launch FaceStream manually. If you are going to launch FaceStream using the Docker Compose script, go to [“Launch FaceStream using Docker Compose”](#).

2.1.1 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.

2.1.1.1 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 \
-v /tmp/logs/streams:/srv/logs \
--rm \
--network=host \
dockerhub.visionlabs.ru/luna/streams-configs:v.0.6.6 \
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

2.1.1.2 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 3.x):

```
python3 -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.

2.1.2 Prepare LUNA Streams database

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.

2.1.2.1 Create LUNA Streams database

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;"
```

2.1.2.2 Initialize LUNA Streams database

Initialize the data in the LUNA Streams database:

```
docker run -v /etc/localtime:/etc/localtime:ro \  
--rm \  
--network=host \  
python3 -m load_dump --dump-file=facestream_dump.json --luna-config=http://127.0.0.1:5070/1
```

```
-v /tmp/logs/streams:/srv/logs \  
dockerhub.visionlabs.ru/luna/luna-streams:v.0.6.6 \  
python3 ./base_scripts/db_create.py --luna-config http://localhost:5070/1
```

2.1.3 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.0.6.6
```

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.0.6.6.

2.1.4 Launch FaceStream container

2.1.4.1 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.5.1.23
```

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}}`.

2.1.4.2 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.5.1.23
```

`--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}}`.

2.2 Launch FaceStream using Docker Compose

The Docker Compose 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>

2.2.1 Launch 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 [“Install GPU dependencies”](#).

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>.

3 Next actions

To work further with FaceStream, you need to create a stream using a POST request “create stream” to the LUNA Streams service. The stream contains policies for processing a video stream/video file/set of images. If the stream is created with the status “pending” (by default), then the FaceStream worker will automatically start processing the stream.

See “Interaction of FaceStream with LUNA Streams” in the administrator manual for details on working with streams and LUNA Streams.

The list of streams to process is available at <http://127.0.0.1:34569/api/1/streams/>. Previewing a stream in a browser is available at http://127.0.0.1:34569/api/1/streams/preview/<stream_id>.

4 Additional information

This section provides the following additional information:

- [Useful commands for working with Docker](#)
- [Launching keys description](#)
- [Actions to enable saving logs to files](#)
- [Configuring Docker log rotation](#)

4.1 Docker commands

4.1.1 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
```

4.1.2 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>
```

4.1.3 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
```

4.1.4 Images names

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

```
docker images
```

4.1.5 Show container logs

You can view the container logs with the following command:

```
docker logs <container_name>
```

4.1.6 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.5.1.23`
- 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)
```

4.1.7 Stop container

You can stop the container using the command:

```
docker stop <container_name>
```

Stop all the containers:

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

4.1.8 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.5.1.23`
- 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)
```

4.2 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.

4.2.1 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.

4.3 Logging to server

To enable saving logs to the server, you should:

- create directories for logs on the server;
- activate log recording and set the location of log storage inside containers;
- configure synchronization of log directories in the container with logs on the server using the `volume` argument at the start of each container.

In the Docker Compose script, synchronization of directories with folders **is not configured**. You need to manually add folder mounting to the `docker-compose.yml` file.

4.3.1 Create logs directory

Below are examples of commands for creating directories for saving logs and assigning rights to them for LUNA Streams and FaceStream.

```
mkdir -p /tmp/logs/facestream /tmp/logs/streams
```

```
chown -R 1001:0 /tmp/logs/facestream /tmp/logs/streams
```

4.3.2 Logging activation

4.3.2.1 LUNA Streams logging activation

To enable logging to file, you need to set the `log_to_file` and `folder_with_logs` settings in the `<SERVICE_NAME>_LOGGER` section of the LUNA Streams settings.

Go to the Configurator service interface (`127.0.0.1:5070`) and set the logs path in the container in the `folder_with_logs` parameter for LUNA Streams service. For example, you can use the path `/srv/logs`.

Set the `log_to_file` option to `true` to enable logging to file.

4.3.2.2 FaceStream logging activation

To enable logging to file, you need to set the value of the `logging > mode` setting in the `FACE_STREAM_CONFIG` section to `l2f` (output logs only to file) or `l2b` (output logs both to file and to console).

Go to the Configurator service interface (`127.0.0.1:5070`) and specify the required setting value. The location path of logs in the FaceStream container cannot be changed. It is necessary to specify the path `/srv/logs` when mounting.

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).

4.3.3 Mounting directories with logs when starting services

The log directory is mounted with the following argument when starting the container:

```
-v <server_logs_folder>:<container_logs_folder> \
```

where <server_logs_folder> is the directory created in the [create logs directory](#) step, and <container_logs_folder> is the directory created in the [activate logging](#) step.

Example of command to launch the FaceStream with mounting a directory with logs:

```
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.5.1.23
```

Examples of manual container launch commands contain these arguments.

4.4 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.