VisionLabs FaceStream

Upgrade manual

v.5.84.0

Contents

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.

Introduction

This document provides an example of the steps required to upgrade to a new build of FaceStream, provided that the application was previously launched with the LUNA PLATFORM installed according to the appropriate manual. If FaceStream was launched in accordance with the FaceStream installation manual without launching LUNA PLATFORM (with the installation of the minimum required components), then you must use the appropriate document in the distribution.

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 and LUNA Streams are installed according to corresponding installation manuals, and the default paths are used. Otherwise, you should consider your manual changes during the update.

When updating, keep in mind that the PostgreSQL and InfluxDB data for the LUNA Streams service is stored in the directory with the LUNA PLATFORM installed. Therefore, it is assumed that when updating FaceStream after updating LUNA PLATFORM, PostgreSQL and InfluxDB data have already been moved to the directory with the new LUNA PLATFORM distribution (see the section "Move data" of the LUNA PLATFORM installation manual).

> 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 upgrade!

Before upgrade FaceStream, you need to do the following:

1. Create backups.
2. Delete symbolic link.
3. Unpack the distribution of the new version of FaceStream.
4. Create new symbolic link.
5. Set up GPU computing if you plan to use GPU.
6. Login to VisionLabs registry, if authorization was not performed earlier.
7. Remove old containers.

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_v.5.84.0.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_v.5.84.0.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_v.5.84.0 fs-current
```

1.5 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.

1.5.1 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
```

Here:

- `driver` — Specifies the driver for the reserved device(s).
- `count` — Specifies the number of GPU devices that should be reserved (providing the host holds that number of GPUs).
- `capabilities` — 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.

1.6 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.

1.7 Remove old containers

Before launching the containers of the current minor version, stop all FaceStream related containers of the previous minor version.

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.

To remove the FaceStream and LUNA Streams containers, use the following command:

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

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
```

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.1.1.18.

> 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.1.1.18 \
-c "python3 -m streams_configs.migrate head --config_db_url postgres://luna:
    luna@127.0.0.1:5432/luna_configurator"
```

Here:

- `python3 -m streams_configs.migrate` — Migration script.
- `--config_db_url postgres://luna:luna@127.0.0.1:5432/luna_configurator` — The "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.1.1.18 \
```

```
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.1.1.18
```

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.1.1.18.

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.5.2.8
```

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 v.5.2.8.

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.5.2.8
```

Here `--gpus device=0` is 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 v.5.2.8.

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.5.2.8.
- 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.5.2.8.
- 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` — 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` — Listening port for the Configurator service. By default, port 5070 is used.

  - `PORT` — 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-9 d7c-91fd8dfc9593 4caf7cf7-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

Below is a description of the container launch commands:

- `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.