



VisionLabs LUNA PLATFORM 5

Upgrade manual

v.5.67.0

Contents

Default ports for services	5
Configuration names for services	6
Introduction	7
1 Before upgrade	8
1.1 Key changes from previous versions	9
1.2 Create backups	12
1.2.1 Backup of PostgreSQL DMBS	13
1.2.2 Backup of Influx database	13
1.2.3 Backup of Image Store buckets	13
1.2.4 Dump file with service settings	13
1.3 Prepare to change the neural network version	14
1.3.1 Change neural network version using Configurator	15
1.4 Delete old symbolic link	15
1.5 Distribution unpacking	15
1.6 Symbolic link creation	16
1.7 Changing group and owner for directories	16
1.8 Move data (versions 5.38.3 and below)	16
1.9 Save user configurations of the Configurator	17
1.10 Create log directory for new services	17
1.11 License update	18
1.11.1 Actions from License activation manual	18
1.12 Calculations using GPU	18
1.13 Remove old containers	19
2 Services launch	21
2.1 Third-party services	22
2.1.1 InfluxDB	22
2.1.2 PostgreSQL	23
2.1.3 Redis	24
2.2 Configurator	25
2.2.1 Optional services usage	25
2.2.2 Configurator database migration	25
2.2.3 Run Configurator container	26
2.3 Image Store	28
2.3.1 Image Store container launch	28
2.3.2 Buckets creation	28

2.3.3	Add TTL for local-buckets	30
2.3.4	Add TTL for S3 buckets	31
2.4	Accounts	33
2.4.1	Accounts DB tables creation	33
2.4.2	Accounts database migration	33
2.4.3	Accounts container launch	33
2.5	Licenses	35
2.5.1	Specify license settings using Configurator	35
2.5.2	Licenses container launch	36
2.6	Faces	37
2.6.1	Faces database migration	37
2.6.2	Faces container launch	37
2.7	Events	38
2.7.1	Events database migration	38
2.7.2	Events container launch	38
2.8	Python Matcher services	39
2.8.1	Use Python Matcher without Python Matcher Proxy	39
2.8.2	Python Matcher container launch	39
2.9	Remote SDK	40
2.9.1	Change the neural network model for extracting descriptors	40
2.9.2	Remote SDK container launch	40
2.10	Handlers	45
2.10.1	Handlers database migration	45
2.10.2	Handlers container launch	45
2.11	Tasks	46
2.11.1	Tasks database migration	46
2.11.2	Tasks and Tasks Worker containers launch	46
2.12	Sender	48
2.12.1	Sender container launch	48
2.13	API	49
2.13.1	API container launch	49
2.13.2	Account migration	49
2.14	Admin	51
2.14.1	Admin container launch	51
2.15	Backport 3	52
2.15.1	Backport 3 database migration	52
2.15.2	Accounts and tokens migration	52
2.15.3	Backport 3 container launch	53
2.15.4	User Interface 3	54

2.16	Backport 4	55
2.16.1	Backport 4 container launch	55
2.16.2	User Interface 4	56
3	Additional information	57
3.1	Account creation	58
3.2	GC task schedule creation	59
3.3	Monitoring and logs visualization using Grafana	60
3.3.1	LUNA Dashboards	60
3.3.2	Grafana Loki	60
3.4	Docker commands	62
3.4.1	Show containers	62
3.4.2	Copy files to container	62
3.4.3	Enter container	62
3.4.4	Images names	62
3.4.5	Delete image	62
3.4.6	Stop container	63
3.4.7	Delete container	63
3.5	Launching parameters description	65
3.5.1	Launching services parameters	65
3.5.2	Creating DB parameters	68
3.6	Logging to server	69
3.6.1	Create logs directory	69
3.6.2	Logging activation	69
3.6.3	Mounting directories with logs when starting services	70
3.7	Docker log rotation	72
3.8	Set custom InfluxDB settings	73
3.9	Use Python Matcher with Python Matcher Proxy	75
3.9.1	Python Matcher proxy container launch	75
3.10	Move old data to root directory	76
3.10.1	Move Image Store buckets	76
3.10.2	Move PostgreSQL data	76
3.10.3	Move InfluxDB Data	77

Default ports for services

Service name	Port
LUNA PLATFORM API	5000
LUNA PLATFORM Admin	5010
LUNA PLATFORM Image Store	5020
LUNA PLATFORM Faces	5030
LUNA PLATFORM Events	5040
LUNA PLATFORM Tasks	5050
LUNA PLATFORM Tasks Worker	5051
LUNA PLATFORM Configurator	5070
LUNA PLATFORM Sender	5080
LUNA PLATFORM Handlers	5090
LUNA PLATFORM Python Matcher	5100
LUNA PLATFORM Licenses	5120
LUNA PLATFORM Backport 4	5130
LUNA PLATFORM Backport 3	5140
LUNA PLATFORM Accounts	5170
LUNA PLATFORM Lambda	5210
LUNA PLATFORM Remote SDK	5220
LUNA PLATFORM 3 User Interface	4100
LUNA PLATFORM 4 User Interface	4200
Oracle DB	1521
PostgreSQL	5432
Redis DB	6379
InfluxDB	8086
Grafana	3000

Configuration names for services

The table below includes the service names in the Configurator service. Use these parameters to configure your services.

Service	Service name in Configurator
API	luna-api
Licenses	luna-licenses
Faces	luna-faces
Image Store	luna-image-store
Accounts	luna-accounts
Tasks	luna-tasks
Events	luna-events
Sender	luna-sender
Admin	luna-admin
Remote SDK	luna-remote-sdk
Handlers	luna-handlers
Lambda	luna-lambda
Python Matcher	luna-python-matcher
Backport 3	luna-backport3
Backport 4	luna-backport4

Settings for the Configurator service are set in its configuration file.

Introduction

This document gives an example of the steps for upgrading from previous build to a new build of LUNA PLATFORM.

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

This document also contains the commands required to upgrade from version 5.2.0 and higher to the current build. Note that starting from versions 5.2.0, there may have been critical changes, such as updates to thresholds or neural network versions, deprecation of FaceDetV1 and FaceDetV2 detectors, and others (see the full list in the [“Key changes from previous versions”](#) section). A change such as updating the thresholds may give a different result when performing estimation than in the old build. This manual is for upgrading from a previous build and these commands are for advanced users only. These commands are marked accordingly. Be careful and do not perform unnecessary actions if you are updating from from LUNA PLATFORM 5.64.0.

For LUNA PLATFORM upgrade, you need to perform the actions from the following sections:

- [“Before upgrade”](#) — Actions to unpack archives, prepare directories, configure a license, etc. Some actions may be optional.
- [“Services launch”](#) — Database migration actions and launching containers with LUNA PLATFORM services.

The [“Additional Information”](#) section provides useful information on describing service launch parameters, Docker commands, enabling Grafana for monitoring visualization, etc.

This manual is designed with an assumption that:

- You already have a previous minor version of LUNA PLATFORM installed and the required environment is up and running at your server(s).
- LP 5 is installed according to LP 5 installation manual, and the default paths are used. Otherwise, you should consider your manual changes during the update.

1 Before upgrade

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

Before upgrading the LUNA PLATFORM, you must perform the following actions:

1. [See key changes from previous versions](#) if you are upgrading from a version other than LUNA PLATFORM 5.64.0.
2. [Create backups](#).
3. [Prepare to change the version of the neural network to extract descriptors](#) if necessary.
4. [Delete old symbolic link](#).
5. [Unpack the distribution of the new version of LUNA PLATFORM](#).
6. [Create new symbolic link](#).
7. [Change group and owner for new directories](#).
8. [Move data](#) if you are upgrading from version 5.38.3 and below.
9. [Save user configurations of the Configurator service](#) if they have changed.
10. [Create log directories for new services](#) if you have previously used logging to file.
11. [Update license](#) if necessary.
12. [Set up GPU computing](#) if you plan to use GPU.
13. [Remove old containers](#).

1.1 Key changes from previous versions

Note: When updating LUNA PLATFORM from a previous version, skip this section.

The following are the key changes from previous versions that you should pay attention to when updating from older versions of LUNA PLATFORM. Some of these changes require mandatory actions to be performed, otherwise the LUNA PLATFORM may not start or function incorrectly.

Not all changes are listed in the table below. See the LUNA PLATFORM release notes for details on all changes.

Version	Changes	Mandatory actions
5.67.0	The threshold value from the “redetect_score_threshold” setting of the “LUNA_REMOTE_SDK_FACE_DETECTOR_SETTING” section has been updated from “0.3” to “0.5”.	Update the threshold value according to user logic.
5.62.3	Now, by default, the neural network model 62 for extracting face descriptors is used.	Read the information described in the section “ Prepare to change the neural network version ”, and perform certain actions before launching Remote SDK, because the default version has changed.
5.53.0	The VisionLabs image for PostgreSQL has been updated from version 12 to version 16.	If this image was previously used, then you need to perform the migration yourself according to official documentation .
5.46.0	The functionality for working with neural networks (detection, estimation and extraction) has been transferred from the Handlers service to the new Remote SDK service.	-
	The 105th neural network model for extracting body descriptors has been removed from the Remote SDK container. Now, by default, the 110th neural network model is used for extracting body descriptors.	Read the information described in the section “ Prepare to change the neural network version ”, and perform certain actions before launching Remote SDK, because the default version has changed.

Version	Changes	Mandatory actions
5.45.1	The default value of the “score_threshold” setting in the “LUNA_HANDLERS_FACE_DETECTOR_SETTINGS” section of the Configurator service has been changed from 0.42 to 0.5.	Check the face recognition logic if the “score_threshold” value is used that is different from the default value.
5.40.0	The PostgreSQL, InfluxDB and Image Store container launch commands now prescribe the paths of the directories for mounting located in the root directory <code>/var/lib/luna/<db_or_bucket_folder></code> , unlike previous versions, where paths were prescribed for a specific version of the LUNA PLATFORM <code>/var/lib/luna/current/example-docker/<db_or_bucket_folder></code> . This enables you not to move data with each update.	Move the old PostgreSQL, InfluxDB and Image Store data to the root directory (see “Move data”), and then delete and recreate the containers by specifying new directory paths for mounting.
5.38.3	The default value of the “score_threshold” setting in the “LUNA_HANDLERS_BODY_DETECTOR_SETTINGS” section of the Configurator service has been changed from 0.3 to 0.5.	Check the body recognition logic if the “score_threshold” value is used that is different from the default value.
	The default value of the “redetect_face_target_size” setting in the “LUNA_HANDLERS_FACE_DETECTOR_SETTINGS” section of the Configurator service has been changed from 45 to 64.	Check the face recognition logic if the “redetect_face_target_size” value is used that is different from the default value.
5.36.5	The address of the license server must now be set in the Configurator settings before starting the Licenses container.	Perform the steps described in the “Specify license server address using Configurator” section.
5.35.0	Support for old Services for index building and searching by index has been discontinued.	-

Version	Changes	Mandatory actions
5.34.0	The 54, 56 and 57 neural network models for extracting face descriptors have been removed from the Handlers container.	Read the information described in the section “Prepare to change the neural network version” , and perform certain actions before launching Handlers if one of the listed models was used.
	The 104 and 106 neural network models for extracting body descriptors has been removed from the Handlers container. Now the 107 model is used by default.	Read the information described in the section “Prepare to change the neural network version” , and perform certain actions before launching Handlers, because the default version has changed.
	Support for the Liveness V1 service has been discontinued.	-
5.30.0	Vendor libraries for HASP key and HASP utility have been updated (from 111186 to 30147).	Update the HASP service and issue a new license (see “License update”).
	The mechanism for creating and managing accounts has been changed.	Perform actions marked with the phrase Note. Accounts migration (upgrading from 5.2.0...5.28.0 only)
5.28.0	The default value of the “score_threshold” setting in the “FACE_DETECTOR_V3” section of the Configurator service has been changed from 0.89 to 0.42.	Check the face recognition logic if the “score_threshold” value is used that is different from the default value.
5.26.0	The presence of a mask on the chin from this version refers to the “missing” state. In previous versions, it referred to the “medical_mask” state.	Check the logic of the mask estimation.
5.24.0	Support for the Vertica database for the Events service has been discontinued.	-

Version	Changes	Mandatory actions
5.23.0	The default thresholds (recommended) have been updated for the following checks in the “face_quality” section and the “/iso” resource: “mouth_occluded” (old values: min=0, max=0.3; new values: min=0, max=0.5) and “mouth_open” (old values: min=0, max=0.64; new values: min=0, max=0.5)	Check the logic of “mouth_occluded” and “mouth_open” if the default values were used.
5.14.0	The Liveness V2 algorithm has been updated. The default thresholds for “liveness_threshold” and “quality_threshold” are now “0.5”. The recommended threshold “liveness_threshold” is now “0.5” instead of “0.88”.	Check the logic of the Liveness V2 algorithm.
5.6.0	Now, by default, the neural network model 59 for extracting face descriptors is used.	Read the information described in the section “Prepare to change the neural network version” , and perform certain actions before launching Handlers, because the default version has changed.
	The 101 model of the neural network for extracting body descriptors has been removed from the Handlers container. Now the 104 model is used by default.	Read the information described in the section “Prepare to change the neural network version” , and perform certain actions before launching Handlers, because the default version has changed.
5.3.0	The logic of launching LUNA PLATFORM services inside containers has been changed. Now applications are launched not from the “root” user, but from the “luna” user.	-
	Support for FaceDetV1 and FaceDetV2 detectors has been discontinued.	-

1.2 Create backups

It is recommended to create the following backups:

- Backups of all databases used with LUNA PLATFORM.
- Backup of Image Store buckets.

- Backup of LUNA PLATFORM services configurations.

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

1.2.1 Backup of PostgreSQL DMBS

A PostgreSQL database backup is performed using the [pg_dumpall](#) or [pg_dump](#) utilities.

Use the official instructions to perform a backup.

1.2.2 Backup of Influx database

An InfluxDB backup is performed using the [influxd backup](#) command.

Use the official instructions to perform a backup.

1.2.3 Backup of Image Store buckets

Create a backup of buckets using the following command:

```
cp -r /var/lib/luna/image_store /var/lib/luna/BACKUP_image_store
```

1.2.4 Dump file with service settings

Custom values of settings for LUNA PLATFORM services (all except the Configurator service) are automatically migrated using the Configurator service migration mechanism.

If the migration of the service for some reason has lost the user configuration or the user just wants to store the old service settings for different LP versions, then you can create a dump file.

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

```
wget -O /var/lib/luna/BACKUP_settings_dump.json 127.0.0.1:5070/1/dump
```

or

```
curl 127.0.0.1:5070/1/dump > /var/lib/luna/BACKUP_settings_dump.json
```

Important: This file will not be used during the normal installation of the LUNA PLATFORM. To apply the dumped settings use the `db_create.py` script with the `--dump-file` command line argument (followed with the created dump file name): `base_scripts/db_create.py --dump-file settings_dump.json`. You can apply full settings dump on an empty database only. See the detailed information in the “Settings dump” section of the administrator manual.

1.3 Prepare to change the neural network version

In some LUNA PLATFORM builds, neural network models for extracting face and body descriptors are removed, and the default model usage settings are changed. See the section [“Key changes from previous versions”](#) for more information about these changes.

If you are updating from a version where neural networks were removed, and in the previous build one of the deleted models was specified in the “DEFAULT_FACE_DESCRIPTOR_VERSION” or “DEFAULT_HUMAN_DESCRIPTOR_VERSION” settings, then the Remote SDK service **will not start**.

The current build of LUNA PLATFORM supports neural network models for extracting descriptors:

Object from which descriptor is extracted	Neural network models	Default model
Face	59, 60, 62	62
Body	107, 110	110

It is necessary to perform one of the additional actions depending on the following scenarios of the work:

Continuation of the use of missing neural networks

Request to VisionLabs an old neural network model and prepare it for transfer to the new Remote SDK container after its launch (see instructions in the section [“Use non-delivery neural network model”](#) of administrator manual).

Switching to new version of the neural network with continuation of the use of old descriptors

- Run the [“Additional extraction”](#) task before the update (see [“Additional extraction task”](#) in the administrator manual). This will convert old descriptors to a new version of the neural network.
- Specify the new version of the neural network in the settings [“DEFAULT_FACE_DESCRIPTOR_VERSION”](#) or [“DEFAULT_HUMAN_DESCRIPTOR_VERSION”](#) before launching the Remote SDK service in accordance with the section [“Change neural network version using Configurator”](#).

Switching to new version of the neural network with cessation of the use of old descriptors

Specify the new version of the neural network in the settings [“DEFAULT_FACE_DESCRIPTOR_VERSION”](#) or [“DEFAULT_HUMAN_DESCRIPTOR_VERSION”](#) before launching the Remote SDK service in accordance with the section [“Change neural network version using Configurator”](#).

If it is not necessary to extract body descriptors, then you can disable the use of the neural network using the command `--env=EXTEND_CMD="--enable-body-descriptor-estimator=0"` when launching Remote SDK container (see the detailed information in the section [“Enable/disable several estimators and detectors”](#) of the administrator manual).

1.3.1 Change neural network version using Configurator

To change the version of the neural network, you must perform the following steps:

- Open the Configurator user interface `http://<configurator_server_ip>:5070`.
- Enter the name of the setting “DEFAULT_FACE_DESCRIPTOR_VERSION” or “DEFAULT_HUMAN_DESCRIPTOR_VERSION” in the “Setting name” field and click “Apply Filters”.
- Set the necessary neural network model in the “DEFAULT_FACE_DESCRIPTOR_VERSION” or “DEFAULT_HUMAN_DESCRIPTOR_VERSION” setting.
- Save the changes by clicking the “Save” button.

1.4 Delete old symbolic link

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

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

1.5 Distribution unpacking

The distribution package is an archive **luna_v.5.67.0**, where **v.5.67.0** is a numerical identifier, describing the current LUNA PLATFORM version.

The archive includes configuration files, required for installation and exploitation. It does not include Docker images for the services. They should be downloaded from the Internet.

Move the distribution package to the directory on your server before the installation. For example, move the files to `/root/` directory. The directory should not contain any other distribution or license files except the target ones.

Move the distribution to the created directory.

```
mv /root/luna_v.5.67.0.zip /var/lib/luna
```

Install the unzip archiver if it is necessary.

```
yum install -y unzip
```

Go to the folder with distribution.

```
cd /var/lib/luna
```

Unzip files.

```
unzip luna_v.5.67.0.zip
```

1.6 Symbolic link creation

Create a symbolic link.

The link indicates that the current version of the distribution file is used to run LUNA PLATFORM.

```
ln -s luna_v.5.67.0 current
```

1.7 Changing group and owner for directories

LP services are launched inside the containers by the “luna” user. Therefore, it is required to set permissions for this user to use the mounted volumes.

Go to the LP “example-docker” directory.

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

Create a directory to store settings.

```
mkdir luna_configurator/used_dumps
```

Set permissions for the user with UID 1001 and group 0 to use the mounted directories.

```
chown -R 1001:0 luna_configurator/used_dumps
```

1.8 Move data (versions 5.38.3 and below)

In the official installation documentation for versions of LUNA PLATFORM v.5.38.3 and lower, the paths in the PostgreSQL, InfluxDB and Image Store container launch commands were specified for a specific version of LUNA PLATFORM `/var/lib/luna/current/example-docker/<db_or_bucket_folder>`. Starting from version LUNA PLATFORM v.5.40.0, the commands for launching PostgreSQL, InfluxDB and Image Store containers specify the paths of the directories for mounting, located in the root directory `/var/lib/luna/<db_or_bucket_folder>`.

Example commands for launching PostgreSQL, InfluxDB and Image Store containers contain arguments for mounting the corresponding data directories from the root directory. If you are upgrading from

version LUNA PLATFORM v.5.38.3 and lower and the previous version of LUNA PLATFORM was installed according to the official documentation, then in the container launch commands, specify the old directories with the data to be mounted or first transfer the old data to the root directory according to the section [“Move old data to root directory”](#).

1.9 Save user configurations of the Configurator

Note: Skip this step if the Configurator settings have not been changed.

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

If your previous LP version was used with non-default Configurator service configurations, back up your “luna_configurator_postgres.conf” config file in the separate directory on your server.

```
cp /var/lib/luna/<your_previous_lp_version>/example-docker/luna_configurator
  /configs/luna_configurator_postgres.conf /var/lib/luna/
  BACKUP_luna_configurator_postgres.conf
```

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

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

```
diff /var/lib/luna/current/example-docker/luna_configurator/configs/
  luna_configurator_postgres.conf /var/lib/luna/
  BACKUP_luna_configurator_postgres.conf
```

1.10 Create log directory for new services

Skip this section if no logs were previously stored on the server.

In the new version of LUNA PLATFORM, new services could appear, for which you need to create directories with logs. It depends on the version from which you are upgrading. For example, version 5.30.0 introduced the Accounts service.

See [“Logging to server”](#) section if you have not previously used logging to a file, but want to enable it.

Following are the commands to create directories for all existing services. These commands will create and assign permissions only to missing directories.

```
mkdir -p /tmp/logs/configurator /tmp/logs/image-store /tmp/logs/accounts /  
tmp/logs/faces /tmp/logs/licenses /tmp/logs/events /tmp/logs/python-  
matcher /tmp/logs/handlers /tmp/logs/remote-sdk /tmp/logs/tasks /tmp/logs  
/tasks-worker /tmp/logs/sender /tmp/logs/api /tmp/logs/admin /tmp/logs/  
backport3 /tmp/logs/backport4
```

```
chown -R 1001:0 /tmp/logs/configurator /tmp/logs/image-store /tmp/logs/  
accounts /tmp/logs/faces /tmp/logs/licenses /tmp/logs/events /tmp/logs/  
python-matcher /tmp/logs/handlers /tmp/logs/remote-sdk /tmp/logs/tasks /  
tmp/logs/tasks-worker /tmp/logs/sender /tmp/logs/api /tmp/logs/admin /tmp  
/logs/backport3 /tmp/logs/backport4
```

If you need to use the Python Matcher Proxy service, then you need to additionally create the `/tmp/logs/python-matcher-proxy` directory and set its permissions.

1.11 License update

To update the license, follow these steps:

- Follow the steps from [license activation manual](#).
- Set settings for [HASP](#) license or [Guardant](#) license before starting Licenses container.

1.11.1 Actions from License activation manual

Open the license activation manual and follow the necessary steps.

Note: This action is mandatory. The license will not work without following the steps to activate the license from the corresponding manual.

Note. When updating Guardant Control Center, you must re-issue the license key.

1.12 Calculations using GPU

You can use GPU for the general calculations performed by Remote SDK.

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 LP distribution and should be downloaded from the Internet):

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

See the [NVIDIA documentation](#) for additional information.

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.

1.13 Remove old containers

Before launching the containers of the current minor version, stop all LUNA PLATFORM related containers of the previous minor version. It is not necessary to delete third-party services containers.

For example, to remove LP containers only use the following command:

```
docker container rm -f luna-configurator luna-backport3 luna-backport4 luna-sender luna-tasks luna-handlers luna-remote-sdk luna-python-matcher luna-events luna-licenses luna-faces luna-image-store luna-ui-3 luna-ui-4 luna-admin luna-api luna-tasks-worker luna-accounts luna-lambda
```

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 LP is successfully launched.

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

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

2 Services launch

This section gives examples for:

- Databases tables migration.
- Buckets creation.
- Launching of containers.

LUNA PLATFORM services must be launched in the following sequence:

- Databases, Balancers, HASP service and other third-party party software.
- [Configurator](#)
- [Image Store](#)
- [Accounts](#)
- [Licenses](#)
- [Faces](#)
- [Events](#)
- [Python Matcher](#)
- [Python Matcher Proxy](#). The service is disabled by default
- [Remote SDK](#)
- [Handlers](#)
- [Tasks](#)
- [Sender](#)
- [API](#)
- [Admin](#)

The following service are used when emulation of LUNA PLATFORM 3 is required only:

- [Backport 3](#)
- [User Interface 3](#)

The following service are used when emulation of LUNA PLATFORM 4 is required only:

- [Backport 4](#)
- [User Interface 4](#)

It is recommended to launch containers one by one and wait for the container status to become “up” (use the `docker ps` command).

Some of these services are optional and you can disable their use. It is recommended to use Events, Tasks, Sender and Admin services by default. See the “[Optional services usage](#)” section for details.

When launching each service, certain parameters are used, for example, `--detach`, `--network`, etc. See the section “[Launching parameters description](#)” for more detailed information about all launch parameters of LUNA PLATFORM services and databases.

See the “[Docker commands](#)” section for details about working with containers.

2.1 Third-party services

This section describes the launching of databases in docker containers. They must be launched before LP services.

2.1.1 InfluxDB

Note: If you haven't deleted the old container, skip this step.

Monitoring LUNA PLATFORM services requires running the Influx 2.0.8-alpine database. Below are the commands to launch the InfluxDB container.

For more information, see the “Monitoring” section in the administrator manual.

If necessary, you can configure the visualization of monitoring data using the LUNA Dashboards service, which includes a configured Grafana data visualization system. In addition, you can launch the Grafana Loki tool for advanced work with logs. See the instructions for launching LUNA Dashboards and Grafana Loki in the “[Monitoring and logs visualization using Grafana](#)” section.

Note: If necessary, you can use an external InfluxDB 2.0.8-alpine. In this case, you can skip the command below, but you will have to set [custom settings](#) for each LUNA PLATFORM service.

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/luna/influx:/var/lib/influxdb2 \  
--restart=always \  
--detach=true \  
--network=host \  
--name influxdb \  
dockerhub.visionlabs.ru/luna/influxdb:2.0.8-alpine
```

2.1.2 PostgreSQL

2.1.2.1 Migrate PostgreSQL 12 to PostgreSQL 16

In LUNA PLATFORM v.5.53.0, the VisionLabs image for PostgreSQL has been updated from version 12 to version 16.

If this image was previously used, then you need to perform the migration yourself according to [official documentation](#). If necessary, you can continue using PostgreSQL 12 by specifying the image “postgis-`v1match:12`” in the container launch command.

Mounting PostgreSQL 12 data from the directory “`/var/lib/luna/postgres`” into a container for PostgreSQL 16 will result in an error.

2.1.2.2 Launch PostgreSQL

Note: If you haven't deleted the old container, 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/luna/postgresql/data:/var/lib/postgresql/data/ \  
  -v /var/lib/luna/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-v1match:16
```

`-v /var/lib/luna/current/example-docker/postgresql/entrypoint-initdb.d:/docker-entrypoint-initdb.d/` \ — The “`docker-entrypoint-initdb.d`” script includes the commands for the creation of services databases. During database creation, a default username and password are automatically used.

`-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.

You should create all the databases for LP services manually if you are going to use an already installed PostgreSQL.

2.1.3 Redis

Note: If you haven't deleted the old container, skip this step.

Use the following command to launch Redis.

```
docker run \  
-v /etc/localtime:/etc/localtime:ro \  
--name=redis \  
--restart=always \  
--detach=true \  
--network=host \  
dockerhub.visionlabs.ru/luna/redis:7.2
```

2.2 Configurator

2.2.1 Optional services usage

The listed below services are not mandatory for LP:

- Events
- Image Store
- Tasks
- Sender
- Handlers
- Python Matcher Proxy (disabled by default)
- Lambda (disabled by default)

Working with the Lambda service is possible only when deploying LUNA PLATFORM services in Kubernetes. See detailed information in the installation manual using Kubernetes.

You can disable them if their functionality is not required for your tasks.

Use the “ADDITIONAL_SERVICES_USAGE” section in the API service settings in the Configurator service to disable unnecessary services.

You can use the dump file provided in the distribution package to enable/disable services before Configurator launch.

```
vi /var/lib/luna/current/extras/conf/platform_settings.json
```

Disabling any of the services has certain consequences. For more information, see the “Disableable services” section of the administrator manual.

2.2.2 Configurator database migration

Migration from the previous LP build to LP build v.5.67.0 is described in this section.

The following instruction for Configurator DB migration assumes that you already have settings migration revision set in your database. This revision is set using the `configs.migrate` head script. This script is included in the installation manuals of LP, starting with build 5.1.1. If you have performed installation according to the manual, you do not need to perform additional actions. Settings migration will be performed automatically.

If there is no revision, you should recreate the database structure. See the LP 5 installation manual section “Configurator DB tables creation” for instructions. Then you should specify all the required settings manually.

The migration from version 5.1.0 is described in “LP_Upgrade_Manual.html” in the distribution package of version 5.1.1. You can update your Configurator database using this instruction and distribution

package of version 5.1.1. Otherwise, you should recreate the database structure for the Configurator service and specify all the required settings manually. See the LP 5 installation manual section “Configurator DB tables creation” for instructions.

Migration from pre-release builds (builds before v.5.1.0) of LP is not provided. You should transfer all the required settings to the Configurator DB of 5.1.1 build manually.

When upgrading the Configurator service database with existing service configurations, a database structure migration must be performed, which will also migrate the saved settings of all LP services except the Configurator service configurations.

Note on migrating Configurator service settings is described in the next section.

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/luna/current/example-docker/luna_configurator/configs/  
  luna_configurator_postgres.conf:/srv/luna_configurator/configs/config.  
  conf \  
--network=host \  
-v /tmp/logs/configurator:/srv/logs \  
--rm \  
--entrypoint bash \  
dockerhub.visionlabs.ru/luna/luna-configurator:v.2.2.18 \  
-c "alembic upgrade head; cd /srv/luna_configurator/configs/configs/  
  python3 -m configs.migrate --config /srv/luna_configurator/configs/config  
  .conf head;"
```

Here:

- `alembic upgrade head`; — Upgrades already existing database structure.
- `python3 -m configs.migrate head`; — Performs settings migrations in Configurator DB and sets revision for migration. The revision will be required during the upgrade to the new LP5 build.

2.2.3 Run Configurator container

Note: Configurator service configurations are not automatically migrated, unlike all other service configurations. If the configurations of the Configurator service were changed in the previous version of LP and you need to save custom values, then in the command below you need to replace the path `/var/lib/luna/current/example-docker/luna_configurator/configs/luna_configurator_postgres.conf` with the path with a backup copy of the configurations Configurator service `/var/lib/luna/BACKUP_luna_configurator_postgres.conf` (see “[Save user configurations of the Configurator](#)” section).

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/luna/current/example-docker/luna_configurator/configs/  
  luna_configurator_postgres.conf:/srv/luna_configurator/configs/config.  
  conf \  
-v /tmp/logs/configurator:/srv/logs \  
--name=luna-configurator \  
--restart=always \  
--detach=true \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-configurator:v.2.2.18
```

At this stage, you can activate logging to file if you need to save them on the server (see the [“Logging to server”](#) section).

2.3 Image Store

2.3.1 Image Store container launch

Note: If you are not going to use the Image Store service, do not launch this container and disable the service utilization in Configurator. See section “[Optional services usage](#)”.

Use the following command to launch the Image Store service:

```
docker run \  
--env=CONFIGURATOR_HOST=127.0.0.1 \  
--env=CONFIGURATOR_PORT=5070 \  
--env=PORT=5020 \  
--env=WORKER_COUNT=1 \  
--env=RELOAD_CONFIG=1 \  
--env=RELOAD_CONFIG_INTERVAL=10 \  
-v /var/lib/luna/image_store:/srv/local_storage/ \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/image-store:/srv/logs \  
--name=luna-image-store \  
--restart=always \  
--detach=true \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-image-store:v.3.14.2
```

Here `-v /var/lib/luna/image_store:/srv/local_storage/` is the 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.

If you already have a directory with LP buckets you should specify it instead of `/var/lib/luna/image_store/`.

2.3.2 Buckets creation

Buckets are required to store data in Image Store. The Image Store service should be launched before the commands execution.

When upgrading from the previous version, it is recommended to launch the bucket creation commands one more time. Hence you make sure that all the required buckets were created.

If the error with code `13006` appears during launching of the listed above commands, the bucket is already created.

Buckets in LP can be created in two ways:

- Using the `lis_bucket_create.py` script located in the Image Store service container.

- Using direct requests to the Image Store service.

Use script to create buckets

Run this script to create general buckets:

```
docker run \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/api:/srv/logs \  
--rm \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-api:v.6.29.0 \  
python3 ./base_scripts/lis_bucket_create.py -ii --luna-config http://  
localhost:5070/1
```

If you want to specify the storage time of an object in a bucket, you can additionally specify the number of days using the `--bucket-ttl` argument. See details in the “Object lifecycle” section of the administrator manual.

If you are going to use the Tasks service, use the following command to additionally create the “task-result” in the Image Store service:

```
docker run \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/tasks:/srv/logs \  
--rm \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-tasks:v.3.22.1 \  
python3 ./base_scripts/lis_bucket_create.py -ii --luna-config http://  
localhost:5070/1
```

If you are going to use the portraits, use the following command to additionally create the “portraits”.

```
docker run \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/api:/srv/logs \  
--rm \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-backport3:v.0.11.9 \  
python3 ./base_scripts/lis_bucket_create.py -ii --luna-config http://  
localhost:5070/1
```

Use direct requests

The curl utility is required for the following requests.

The “visionlabs-samples” bucket is used for face samples storage. The bucket is required for LP utilization.

```
curl -X POST http://127.0.0.1:5020/1/buckets?bucket=visionlabs-samples
```

If you want to specify the storage time of an object in a bucket, you can additionally specify the number of days using the `ttl` query parameter. See details in the “Object lifecycle” section of the administrator manual.

The “portraits” bucket is used for portraits storage. The bucket is required for Backport 3 utilization.

```
curl -X POST http://127.0.0.1:5020/1/buckets?bucket=portraits
```

The “visionlabs-bodies-samples” bucket is used for human bodies samples storage. The bucket is required for LP utilization.

```
curl -X POST http://127.0.0.1:5020/1/buckets?bucket=visionlabs-bodies-samples
```

The “visionlabs-image-origin” bucket is used for source images storage. The bucket is required for LP utilization.

```
curl -X POST http://127.0.0.1:5020/1/buckets?bucket=visionlabs-image-origin
```

The “visionlabs-objects” bucket is used for objects storage. The bucket is required for LP utilization.

```
curl -X POST http://127.0.0.1:5020/1/buckets?bucket=visionlabs-objects
```

The “task-result” bucket for the Tasks service. Do not use it if you are not going to use the Tasks service.

```
curl -X POST http://127.0.0.1:5020/1/buckets?bucket=task-result
```

2.3.3 Add TTL for local-buckets

Note. Perform the steps below only if the buckets are stored in local storage and you need to manage the lifetime of existing and/or new objects in the buckets.

In order to add TTLs for all objects in a bucket located in local storage, you must update it by specifying the `ttl` request parameter.

For example, you can add a lifetime of all objects equal to 2 days in the “visionlabs-samples” bucket using the following command:

```
curl -X PUT http://127.0.0.1:5020/1/buckets?bucket=visionlabs-samples?ttl=2
```

2.3.4 Add TTL for S3 buckets

Note: Perform the steps below only if the buckets are stored in S3-like storage and you need to manage the lifetime of existing and/or new objects in the bucket.

In order to add TTLs for objects in buckets located in S3-like storage, a special migration script `migrate_ttl_settings.py` must be executed.

Note: See the “Migration to apply TTL to objects in S3” section in the administrator manual for details on migration and its specifics.

The migration script should be run with the following arguments:

- `--bucket` – Bucket name, e.g. `visionlabs-samples`.
- `--update-tags` - Whether tags should be added to all existing objects.

If the `--update-tags` argument is 1, the tags required for TTL will be added to all existing objects. The duration of the migration will depend on the number of existing objects.

If the `--update-tags` argument is 0, the tags required for TTL will not be added to all existing objects. In this case, lifecycle management for existing objects will not be available.

Tags will be added automatically to new objects after migration.

The script also needs to fill the authorization data to the S3-like storage in the configuration file and mount it to the Image Store service container.

Fill the configuration file with the following command:

```
vi /var/lib/luna/current/extras/conf/s3_bucket.conf
```

Perform the migration of buckets to the S3 storage:

```
docker run \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/image-store:/srv/logs \  
-v /var/lib/luna/current/extras/conf/s3_bucket.conf:/srv/base_scripts/  
  migrate_ttl_settings/config.conf \  
--rm \  
--network=host \  

```

```
dockerhub.visionlabs.ru/luna/luna-image-store:v.3.14.2 \  
python3 ./base_scripts/migrate_ttl_settings/migrate_ttl_settings.py --bucket  
=<your_bucket_name> --update-tags=<your_value>
```

2.4 Accounts

2.4.1 Accounts DB tables creation

Note. Accounts migration (upgrading from 5.2.0...5.28.0 only). Perform the steps below only if you are upgrading from LP 5.2.0...5.28.0, where the **Admin service was not used**. If you have used the Admin service before, skip this section and perform the steps in the “[Accounts database migration](#)” section.

Use the following command to create Accounts DB tables:

```
docker run \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/accounts:/srv/logs \  
--rm \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-accounts:v.0.3.9 \  
python3 ./base_scripts/db_create.py --luna-config http://localhost:5070/1
```

2.4.2 Accounts database migration

You need to execute migration scripts to update your Accounts database structure.

It is recommended to create the back up of your database before applying any changes.

```
docker run \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/accounts:/srv/logs \  
--rm \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-accounts:v.0.3.9 \  
alembic -x luna-config=http://127.0.0.1:5070/1 upgrade head
```

In LUNA PLATFORM 5.30.0, the database that stores information about accounts has changed. In LUNA PLATFORM versions 5.2.0...5.28.0, information was stored in the Admin database (`luna_admin` database), and in versions 5.30.0 and higher, information is stored in the Accounts database (`luna_accounts` database). If you are upgrading from versions 5.2.0...5.28.0 to the current version, then the script above will transform the `luna_admin` database to work with new accounts (**the database name will remain the same**).

2.4.3 Accounts container launch

Use the following command to launch the service:

```
docker run \  
--env=CONFIGURATOR_HOST=127.0.0.1 \  
--env=CONFIGURATOR_PORT=5070 \  
--env=PORT=5170 \  
--env=WORKER_COUNT=1 \  
--env=RELOAD_CONFIG=1 \  
--env=RELOAD_CONFIG_INTERVAL=10 \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/accounts:/srv/logs \  
--name=luna-accounts \  
--restart=always \  
--detach=true \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-accounts:v.0.3.9
```

2.5 Licenses

Note: To use a trial license, it is required to launch the Licenses service on the same server where trial license is being used.

2.5.1 Specify license settings using Configurator

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

2.5.1.1 Specify HASP license settings

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

To set the license server address, follow these steps:

- Go to the Configurator service interface `http://<configurator_server_ip>:5070/`.
- Specify the “LICENSE_VENDOR” value in the “Setting name” field and click “Apply Filters”.
- Set the IP address of the server with your HASP key in the field “server_address” in the format “127.0.0.1”.
- Click “Save”.

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

2.5.1.2 Specify Guardant license settings

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

To set the license server address, follow these steps:

- Go to the Configurator service interface `http://<configurator_server_ip>:5070/`.
- Enter the value “LICENSE_VENDOR” in the “Setting name” field and click “Apply Filters”.
- Set the IP address of the server with your Guardant key in the “server_address” field.
- Set the license ID in the format `0x<your_license_id>`, obtained in the section “Save license ID” in the License activation manual, in the “license_id” field.
- Click “Save”.

If the license is activated using the Guardant key, then three parameters “vendor”, “server_address”

and “license_id” must be specified. If you want to change the Guardant protection to HASP, then you need to delete the “license_id” field.

2.5.2 Licenses container launch

Use the following command to launch the service:

```
docker run \  
--env=CONFIGURATOR_HOST=127.0.0.1 \  
--env=CONFIGURATOR_PORT=5070 \  
--env=PORT=5120 \  
--env=WORKER_COUNT=1 \  
--env=RELOAD_CONFIG=1 \  
--env=RELOAD_CONFIG_INTERVAL=10 \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/licenses:/srv/logs \  
--name=luna-licenses \  
--restart=always \  
--detach=true \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-licenses:v.0.10.18
```

2.6 Faces

2.6.1 Faces database migration

You need to execute migration scripts to update your Faces database structure.

It is recommended to create the back up of your database before applying any changes.

Run the following command to perform the Faces DB migration.

```
docker run \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/faces:/srv/logs \  
--rm \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-faces:v.4.11.9 \  
alembic -x luna-config=http://127.0.0.1:5070/1 upgrade head
```

2.6.2 Faces container launch

Use the following command to launch the service:

```
docker run \  
--env=CONFIGURATOR_HOST=127.0.0.1 \  
--env=CONFIGURATOR_PORT=5070 \  
--env=PORT=5030 \  
--env=WORKER_COUNT=2 \  
--env=RELOAD_CONFIG=1 \  
--env=RELOAD_CONFIG_INTERVAL=10 \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/faces:/srv/logs \  
--name=luna-faces \  
--restart=always \  
--detach=true \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-faces:v.4.11.9
```

2.7 Events

2.7.1 Events database migration

You need to execute migration scripts to update your Events database structure.

It is recommended to create the back up of your database before applying any changes.

```
docker run \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/events:/srv/logs \  
--rm \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-events:v.4.12.9 \  
alembic -x luna-config=http://127.0.0.1:5070/1 upgrade head
```

2.7.2 Events container launch

Note: If you are not going to use the Events service, do not launch this container and disable the service utilization in Configurator. See section “[Optional services usage](#)”.

Use the following command to launch the service:

```
docker run \  
--env=CONFIGURATOR_HOST=127.0.0.1 \  
--env=CONFIGURATOR_PORT=5070 \  
--env=PORT=5040 \  
--env=WORKER_COUNT=1 \  
--env=RELOAD_CONFIG=1 \  
--env=RELOAD_CONFIG_INTERVAL=10 \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/events:/srv/logs \  
--name=luna-events \  
--restart=always \  
--detach=true \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-events:v.4.12.9
```

2.8 Python Matcher services

For matching tasks, you can use either only the Python Matcher service, or additionally use the Python Matcher Proxy service, which redirects matching requests to either the Python Matcher service or matching plugins. This section describes how to use Python Matcher without Python Matcher Proxy.

You need to use the Python Matcher Proxy service only if you are going to use matching plugins. Using Python Matcher Proxy and running the corresponding docker container are described in the [“Use Python Matcher with Python Matcher Proxy”](#) section.

See the description and usage of matching plugins in the administrator manual.

2.8.1 Use Python Matcher without Python Matcher Proxy

The Python Matcher service with matching by the Faces DB is enabled by default after launching.

The Python Matcher service with matching by the Events is also enabled by default. You can disable it by specifying “USE_LUNA_EVENTS = 0” in the “ADDITIONAL_SERVICES_USAGE” settings of Configurator (see [“Optional services usage”](#) section). Thus, the Events service will not be used for LUNA PLATFORM.

The Python Matcher that matches using the matcher library is enabled when “CACHE_ENABLED” is set to “true” in the “DESCRIPTORS_CACHE” setting.

A single image is downloaded for the Python Matcher service and the Python Matcher Proxy service.

2.8.2 Python Matcher container launch

Use the following command to launch the service:

```
docker run \  
--env=CONFIGURATOR_HOST=127.0.0.1 \  
--env=CONFIGURATOR_PORT=5070 \  
--env=PORT=5100 \  
--env=WORKER_COUNT=1 \  
--env=RELOAD_CONFIG=1 \  
--env=RELOAD_CONFIG_INTERVAL=10 \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/python-matcher:/srv/logs \  
--name=luna-python-matcher \  
--restart=always \  
--detach=true \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-python-matcher:v.1.9.9
```

2.9 Remote SDK

2.9.1 Change the neural network model for extracting descriptors

Some builds of LUNA PLATFORM remove the neural network models for extracting descriptors of faces and bodies, and also change the default settings for using models.

If you are upgrading from a version where neural networks were removed, and in the previous build one of the deleted models was specified in the “DEFAULT_FACE_DESCRIPTOR_VERSION” or “DEFAULT_HUMAN_DESCRIPTOR_VERSION” settings, then the Remote SDK service **will not start** unless you do one of the following actions:

- request old neural network model from VisionLabs specialists and transfer it to new container
- switch to new version of the neural network, having previously completed the Additional extraction task to re-extract existing descriptors using the new version of the neural network for their further use
- switch to new version of the neural network with cessation of the use of old descriptors

See “[Prepare to change the neural network version](#)” for details.

2.9.2 Remote SDK container launch

You can run the Remote SDK service utilizing CPU (set by default) or GPU.

By default, the Remote SDK service is launched with all estimators and detectors enabled. If necessary, you can disable the use of some estimators or detectors when launching the Remote SDK container. Disabling unnecessary estimators enables you to save RAM or GPU memory, since when the Remote SDK service launches, the possibility of performing these estimates is checked and neural networks are loaded into memory. If you disable the estimator or detector, you can also remove its neural network from the Remote SDK container. See the “Enable/disable several estimators and detectors” section of the administrator manual for more information.

Run the Remote SDK service using one of the following commands according to the utilized processing unit.

2.9.2.1 Run Remote SDK utilizing CPU

Use the following command to launch the Remote SDK service using CPU:

```
docker run \  
--env=CONFIGURATOR_HOST=127.0.0.1 \  
--env=CONFIGURATOR_PORT=5070 \  
--env=PORT=5220 \  
--env=WORKER_COUNT=1 \  
--env=RELOAD_CONFIG=1 \  

```

```
--env=RELOAD_CONFIG_INTERVAL=10 \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/remote-sdk:/srv/logs \  
--network=host \  
--name=luna-remote-sdk \  
--restart=always \  
--detach=true \  
dockerhub.visionlabs.ru/luna/luna-remote-sdk:v.0.6.0
```

2.9.2.2 Run Remote SDK utilizing GPU

The Remote SDK service does not utilize GPU by default. If you are going to use the GPU, then you should enable its use for the Remote SDK service in the Configurator service.

If you need to use the GPU for all estimators and detectors at once, then you need to use the “global_device_class” parameter in the “LUNA_REMOTE_SDK_RUNTIME_SETTINGS” section. All estimators and detectors will use the value of this parameter if the “device_class” parameter of their settings like “LUNA_REMOTE_SDK_<estimator-or-detector-name>_SETTINGS.runtime_settings” is set to “global” (by default for all estimators and detectors).

If you need to use the GPU for a specific estimator or detector, then you need to use the “device_class” parameter in sections like “LUNA_REMOTE_SDK_<estimator/detector-name>_SETTINGS.runtime_settings”.

See section “[Calculations using GPU](#)” for additional requirements for GPU utilization.

Use the following command to launch the Remote SDK service using GPU:

```
docker run \  
--env=CONFIGURATOR_HOST=127.0.0.1 \  
--env=CONFIGURATOR_PORT=5070 \  
--env=PORT=5220 \  
--env=WORKER_COUNT=1 \  
--env=RELOAD_CONFIG=1 \  
--env=RELOAD_CONFIG_INTERVAL=10 \  
--gpus device=0 \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/remote-sdk:/srv/logs \  
--network=host \  
--name=luna-remote-sdk \  
--restart=always \  
--detach=true \  
dockerhub.visionlabs.ru/luna/luna-remote-sdk:v.0.6.0
```

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

2.9.2.3 Run slim version of Remote SDK

You can run a slim version of the Remote SDK service that contains only configuration files without neural networks. It is assumed that the user himself will add the neural networks he needs to the container.

The launch of the slim version of the Remote SDK service is intended for advanced users.

To successfully launch the Remote SDK container with a custom set of neural networks, you need to perform the following actions:

- Request the required neural networks from VisionLabs.
- Place neural networks in a folder with LUNA PLATFORM installed.
- Assign appropriate rights to neural network files.
- Mount neural network files to the `/srv/fsdk/data` folder of the Remote SDK container.
- Using the arguments of the variable “EXTEND_CMD” to explicitly specify which of the neural networks should be used.

Using the “enable-all-estimators-by-default” flag for the “EXTEND_CMD” variable, you can disable the use of all neural networks (estimators) by default, and then use special flags to explicitly specify which neural networks should be used. If you do not specify this flag or set the value “--enable-all-estimators-by-default=1”, the Remote SDK service will try to find all neural networks in the container. If one of the neural networks is not found, the Remote SDK service will not start.

List of available estimators:

Argument	Description
--enable-all-estimators-by-default	Enable all estimators by default.
--enable-human-detector	Simultaneous detector of bodies and bodies.
--enable-face-detector	Face detector.
--enable-body-detector	Body detector.
--enable-face-landmarks5-estimator	Face landmarks5 estimator.
--enable-face-landmarks68-estimator	Face landmarks68 estimator.
--enable-head-pose-estimator	Head pose estimator.
--enable-liveness-estimator	Liveness estimator.
--enable-fisheye-estimator	FishEye effect estimator.
--enable-face-detection-background-estimator	Image background estimator.
--enable-face-warp-estimator	Face sample estimator.
--enable-body-warp-estimator	Body sample estimator.
--enable-quality-estimator	Image quality estimator.
--enable-image-color-type-estimator	Face color type estimator.

Argument	Description
--enable-face-natural-light-estimator	Natural light estimator.
--enable-eyes-estimator	Eyes estimator.
--enable-gaze-estimator	Gaze estimator.
--enable-mouth-attributes-estimator	Mouth attributes estimator.
--enable-emotions-estimator	Emotions estimator.
--enable-mask-estimator	Mask estimator.
--enable-glasses-estimator	Glasses estimator.
--enable-eyebrow-expression-estimator	Eyebrow estimator.
--enable-red-eyes-estimator	Red eyes estimator.
--enable-headwear-estimator	Headwear estimator.
--enable-basic-attributes-estimator	Basic attributes estimator.
--enable-face-descriptor-estimator	Face descriptor extraction estimator.
--enable-body-descriptor-estimator	Body descriptor extraction estimator.
--enable-body-attributes-estimator	Body attributes estimator.
--enable-people-count-estimator	People count estimator.
--enable-deepfake-estimator	Deepfake estimator.

See the detailed information on enabling and disabling certain estimators in the section “Enable/disable several estimators and detectors” of the administrator manual.

Below is an example of a command to assign rights to a neural network file:

```
chown -R 1001:0 /var/lib/luna/current/<neural_network_name>.plan
```

Example of a command to run Remote SDK container with mounting neural networks for face detection and face descriptor extraction:

```
docker run \
  --env=CONFIGURATOR_HOST=127.0.0.1 \
  --env=CONFIGURATOR_PORT=5070 \
  --env=PORT=5220 \
  --env=WORKER_COUNT=1 \
  --env=RELOAD_CONFIG=1 \
```

```
--env=RELOAD_CONFIG_INTERVAL=10 \  
--env=EXTEND_CMD="--enable-all-estimators-by-default=0 --enable-face-  
detector=1 --enable-face-descriptor-estimator=1" \  
-v /var/lib/luna/current/cnn59b_cpu-avx2.plan:/srv/fsdk/data/cnn59b_cpu-avx2  
.plan \  
-v /var/lib/luna/current/FaceDet_v3_a1_cpu-avx2.plan:/srv/fsdk/data/  
FaceDet_v3_a1_cpu-avx2.plan \  
-v /var/lib/luna/current/FaceDet_v3_redetect_v3_cpu-avx2.plan:/srv/fsdk/data  
/FaceDet_v3_redetect_v3_cpu-avx2.plan \  
-v /var/lib/luna/current/slnet_v3_cpu-avx2.plan:/srv/fsdk/data/slnet_v3_cpu-  
avx2.plan \  
-v /var/lib/luna/current/LNet_precise_v2_cpu-avx2.plan:/srv/fsdk/data/  
LNet_precise_v2_cpu-avx2.plan \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/remote-sdk:/srv/logs \  
--network=host \  
--name=luna-remote-sdk \  
--restart=always \  
--detach=true \  
dockerhub.visionlabs.ru/luna/luna-remote-sdk:v.0.6.0
```

2.10 Handlers

Note: If you are not going to use the Handlers service, do not launch this container and disable the service utilization in Configurator. See section “[Optional services usage](#)”.

2.10.1 Handlers database migration

You need to execute migration scripts to update your Handlers database structure.

It is recommended to create the back up of your database before applying any changes.

```
docker run \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/handlers:/srv/logs \  
--rm \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-handlers:v.3.6.0 \  
alembic -x luna-config=http://127.0.0.1:5070/1 upgrade head
```

2.10.2 Handlers container launch

Use the following command to launch the service:

```
docker run \  
--env=CONFIGURATOR_HOST=127.0.0.1 \  
--env=CONFIGURATOR_PORT=5070 \  
--env=PORT=5090 \  
--env=WORKER_COUNT=1 \  
--env=RELOAD_CONFIG=1 \  
--env=RELOAD_CONFIG_INTERVAL=10 \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/handlers:/srv/logs \  
--name=luna-handlers \  
--restart=always \  
--detach=true \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-handlers:v.3.6.0
```

2.11 Tasks

Note: If you are not going to use the Tasks service, do not launch the Tasks container and the Tasks Worker container. Disable the service utilization in Configurator. See section “[Optional services usage](#)”.

2.11.1 Tasks database migration

You need to execute migration scripts to update your Tasks database structure.

It is recommended to create the back up of your database before applying any changes.

```
docker run \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/tasks:/srv/logs \  
--rm \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-tasks:v.3.22.1 \  
alembic -x luna-config=http://127.0.0.1:5070/1 upgrade head
```

2.11.2 Tasks and Tasks Worker containers launch

Tasks service image includes the Tasks service and the Tasks Worker. They both must be launched.

The “task-result” bucket should be created for the Tasks service before the service launch. The buckets creation is described in the “[Buckets creation](#)”.

If it is necessary to use the Estimator task using a network disk, then you should first mount the directory with images from the network disk into special directories of Tasks and Tasks Worker containers. See the “Estimator task” section in the administrator manual for details.

2.11.2.1 Tasks Worker launch

Use the following command to launch the service:

```
docker run \  
--env=CONFIGURATOR_HOST=127.0.0.1 \  
--env=CONFIGURATOR_PORT=5070 \  
--env=PORT=5051 \  
--env=WORKER_COUNT=1 \  
--env=RELOAD_CONFIG=1 \  
--env=RELOAD_CONFIG_INTERVAL=10 \  
--env=SERVICE_TYPE="tasks_worker" \  
-v /etc/localtime:/etc/localtime:ro \  

```

```
-v /tmp/logs/tasks-worker:/srv/logs \  
--name=luna-tasks-worker \  
--restart=always \  
--detach=true \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-tasks:v.3.22.1
```

2.11.2.2 Tasks launch

Use the following command to launch the service:

```
docker run \  
--env=CONFIGURATOR_HOST=127.0.0.1 \  
--env=CONFIGURATOR_PORT=5070 \  
--env=PORT=5050 \  
--env=WORKER_COUNT=1 \  
--env=RELOAD_CONFIG=1 \  
--env=RELOAD_CONFIG_INTERVAL=10 \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/tasks:/srv/logs \  
--name=luna-tasks \  
--restart=always \  
--detach=true \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-tasks:v.3.22.1
```

2.12 Sender

2.12.1 Sender container launch

Note: If you are not going to use the Sender service, do not launch this container and disable the service utilization in Configurator. See section [“Optional services usage”](#).

Use the following command to launch the service:

```
docker run \  
--env=CONFIGURATOR_HOST=127.0.0.1 \  
--env=CONFIGURATOR_PORT=5070 \  
--env=PORT=5080 \  
--env=WORKER_COUNT=1 \  
--env=RELOAD_CONFIG=1 \  
--env=RELOAD_CONFIG_INTERVAL=10 \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/sender:/srv/logs \  
--name=luna-sender \  
--restart=always \  
--detach=true \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-sender:v.2.11.9
```

2.13 API

2.13.1 API container launch

Use the following command to launch the service:

```
docker run \  
--env=CONFIGURATOR_HOST=127.0.0.1 \  
--env=CONFIGURATOR_PORT=5070 \  
--env=PORT=5000 \  
--env=WORKER_COUNT=1 \  
--env=RELOAD_CONFIG=1 \  
--env=RELOAD_CONFIG_INTERVAL=10 \  
--name=luna-api \  
--restart=always \  
--detach=true \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/api:/srv/logs \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-api:v.6.29.0
```

2.13.2 Account migration

Note. Accounts migration (upgrading from 5.2.0...5.28.0 only). If you want to keep the ability to use the “account_id” that was used as the “Luna-Account-Id” header in versions 5.2.0...5.28.0 (without creating an account in the Admin service), then you need to link the old “account_id” to the account being created.

Example of CURL-request to the “create account” resource:

```
curl --location --request POST 'http://127.0.0.1:5000/6/accounts' \  
--header 'Content-Type: application/json' \  
--header 'Luna-Account-Id: <your_old_account_id>' \  
--data '{  
  "login": "user@mail.com",  
  "password": "password",  
  "account_type": "user",  
  "description": "description"  
}'
```

It is necessary to replace the authentication data from the example with your own.

To work with tokens, you must have an account.

2.14 Admin

2.14.1 Admin container launch

Note: If you are not going to use the Admin service, do not launch this container.

Use the following command to launch the service:

```
docker run \  
--env=CONFIGURATOR_HOST=127.0.0.1 \  
--env=CONFIGURATOR_PORT=5070 \  
--env=PORT=5010 \  
--env=WORKER_COUNT=1 \  
--env=RELOAD_CONFIG=1 \  
--env=RELOAD_CONFIG_INTERVAL=10 \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/admin:/srv/logs \  
--name=luna-admin \  
--restart=always \  
--detach=true \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-admin:v.5.7.1
```

Monitoring data about the number of executed requests is saved in the luna-admin bucket of the InfluxDB. To enable data saving use the following command:

```
docker exec -it luna-admin python3 ./base_scripts/influx2_cli.py  
create_usage_task --luna-config http://127.0.0.1:5070/1
```

2.15 Backport 3

The section describes launching of Backport 3 service.

The service is not mandatory for utilizing LP5 and is required for emulation of LP 3 API only.

2.15.1 Backport 3 database migration

You need to execute migration scripts to update your Backport 3 DB structure.

It is recommended to create the back up of your database before applying any changes.

Run the following command to perform the Backport 3 DB migration.

```
docker run \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/backport3:/srv/logs \  
--rm \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-backport3:v.0.11.9 \  
alembic -x luna-config=http://127.0.0.1:5070/1 upgrade head
```

2.15.2 Accounts and tokens migration

Note. Accounts migration (upgrading from 5.2.0...5.28.0 only). Use the following command to migrate existing accounts and tokens from Backport 3 previous version:

```
docker run \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/backport3:/srv/logs \  
--rm \  
--network=host \  
--entrypoint bash dockerhub.visionlabs.ru/luna/luna-backport3:v.0.11.9 -c "  
  cd ./base_scripts/migrate_accounts/ && pip3 install -r requirements-  
  postgres.txt && python3 ./run.py --backport_db_url postgres://luna:  
  luna@127.0.0.1:5432/luna_backport3 --accounts_db_url postgres://luna:  
  luna@127.0.0.1:5432/luna_admin"
```

Here:

- `python3 ./run.py` — Running the script for migrating accounts and tokens.
- `--backport_db_url postgres://luna:luna@127.0.0.1:5432/luna_backport3` — “luna_backport3” DB address flag.

- `--accounts_db_url postgres://luna:luna@127.0.0.1:5432/luna_admin--"luna_admin"` DB address flag.

When migrating API and Faces service databases, all Backport 3 accounts will be migrated and stored in the database of the Accounts service. All migrated accounts will be of type "user". The fields "password", "email" and "organization_name" will be transferred from the "account" table of the Backport 3 database to the "account" table of the Accounts database under new names — "password", "login" and "description" respectively. Tokens will remain stored in the Backport3 database, but their identifier will also be entered in the Accounts database, where the necessary permissions will be automatically set.

2.15.3 Backport 3 container launch

Use the following command to launch the service:

```
docker run \  
--env=CONFIGURATOR_HOST=127.0.0.1 \  
--env=CONFIGURATOR_PORT=5070 \  
--env=PORT=5140 \  
--env=WORKER_COUNT=1 \  
--env=RELOAD_CONFIG=1 \  
--env=RELOAD_CONFIG_INTERVAL=10 \  
--name=luna-backport3 \  
--restart=always \  
--detach=true \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/backport3:/srv/logs \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-backport3:v.0.11.9
```

2.15.4 User Interface 3

The User Interface 3 is used with the Backport 3 service only.

2.15.4.1 User Interface 3 container launch

Use the following command to launch the service:

```
docker run \  
--env=PORT=4100 \  
--env=LUNA_API_URL=http://127.0.0.1:5140 \  
--name=luna-ui-3 \  
--restart=always \  
--detach=true \  
--network=host \  
-v /etc/localtime:/etc/localtime:ro \  
dockerhub.visionlabs.ru/luna/luna3-ui:v.0.5.13
```

Here:

- `--env=LUNA_API_URL` — Specifies the URL of the Backport 3 service.
- `--env=PORT` — Specifies the port of the User Interface 3 service.

2.16 Backport 4

The section describes launching of Backport 4 service.

The service is not mandatory for utilizing LP5 and is required for emulation of LP 4 API only.

2.16.1 Backport 4 container launch

Use the following command to launch the service:

```
docker run \  
--env=CONFIGURATOR_HOST=127.0.0.1 \  
--env=CONFIGURATOR_PORT=5070 \  
--env=PORT=5130 \  
--env=WORKER_COUNT=1 \  
--env=RELOAD_CONFIG=1 \  
--env=RELOAD_CONFIG_INTERVAL=10 \  
--name=luna-backport4 \  
--restart=always \  
--detach=true \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/backport4:/srv/logs \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-backport4:v.1.5.9
```

2.16.2 User Interface 4

The User Interface 4 is used with the Backport 4 service only.

2.16.2.1 User Interface 4 container launch

Note: You should have the account with account type **user** before launching the User Interface 4 container. Its login and password in Base64 format will be used to work with the user interface.

Use the following command to launch the service:

```
docker run \  
--env=PORT=4200 \  
--env=LUNA_API_URL=http://<server_external_ip>:5130 \  
--env=BASIC_AUTH=dXNlcjBtYWlsLmNvbTpwYXNzd29yZA== \  
--name=luna-ui-4 \  
--restart=always \  
--detach=true \  
--network=host \  
-v /etc/localtime:/etc/localtime:ro \  
dockerhub.visionlabs.ru/luna/luna4-ui:v.0.1.8
```

Here:

- --env=PORT — Sets the port for running User Interface 4.
- --env=BASIC_AUTH — Sets the Basic authorization for the account which data is displayed in the user interface.
- --env=LUNA_API_URL — Sets the URL of the Backport 4 service.

You should use the external IP of the service, not localhost.

You should specify the Backport 4 service port (“5130” is set by default).

3 Additional information

This section provides the following additional information:

- [Account creation.](#)
- [Creating a schedule for a garbage collection task.](#)
- [Monitoring and logs visualization using Grafana.](#)
- [Useful commands for working with Docker.](#)
- [Description of the parameters for launching LUNA PLATFORM services and creating databases.](#)
- [Actions to enable saving LP service logs to files.](#)
- [Configuring Docker log rotation.](#)
- [Setting custom InfluxDB settings.](#)
- [Using Python Matcher service with Python Matcher Proxy service.](#)
- [Move old LUNA PLATFORM data to root directory](#)

3.1 Account creation

The account is created using an HTTP request to the “create account” resource of the API service.

You can also create an account using the Admin service. This method requires an existing login and password (or the default login and password) and enables you to create an “admin” account. See the “Admin service” section of the administrator manual for details.

To create the account using a request to the API service, you need to provide the following mandatory data:

- “login” — Email address.
- “password” — Password.
- “account_type” — Account type (“user” or “advanced_user”).

Create the account using your authentication details.

Example of CURL-request to the “create account” resource:

```
curl --location --request POST 'http://127.0.0.1:5000/6/accounts' \  
--header 'Content-Type: application/json' \  
--data '{  
  "login": "user@mail.com",  
  "password": "password",  
  "account_type": "user",  
  "description": "description"  
}'
```

It is necessary to replace the authentication data from the example with your own.

To work with tokens, you must have an account.

3.2 GC task schedule creation

Before you start working with the LUNA PLATFORM, you can create a schedule for the Garbage collection task.

To do this, make a “create tasks schedule” request to the API service, specifying the necessary rules for the schedule.

An example of a schedule creation command for an account created in section “[Account creation](#)” is given below.

The example sets a schedule for the Garbage collection task for events older than 30 days with the removal of the samples and the source images. The task will be repeated **once a day at 05:30 am**.

```
curl --location --request POST 'http://127.0.0.1:5000/6/tasks/schedules' \
--header 'Authorization: Basic dXNlckBtYWlsLmNvbTpwYXNzd29yZA==' \
--header 'Content-Type: application/json' \
--data '{
  "task": {
    "task_type": 4,
    "content": {
      "target": "events",
      "filters": {
        "create_time__lt": "now-30d"
      },
      "remove_samples": true,
      "remove_image_origins": true
    }
  },
  "trigger": {"cron": "30 5 * * *", "cron_timezone": "utc"},
  "behaviour": {"start_immediately": false, "create_stopped": false}
}'
```

If necessary, you can create a schedule without automatically activating it. To do this, specify the parameter “create_stopped”: “true”. In this case, after creating the schedule, it must be activated manually using the “action” = “start” parameter of the “patch tasks schedule” request.

For more information, see the “Running scheduled tasks” section of the administrator manual.

3.3 Monitoring and logs visualization using Grafana

Monitoring visualization is performed by the LUNA Dashboards service, which contains the Grafana monitoring data visualization platform with configured LUNA PLATFORM dashboards.

If necessary, you can install customized dashboards for Grafana separately. See the “LUNA Dashboards” section in the administrator manual for more information.

Together with Grafana, you can use the Grafana Loki log aggregation system, which enables you to flexibly work with LUNA PLATFORM logs. The Promtail agent is used to deliver LUNA PLATFORM logs to Grafana Loki (for more information, see the “Grafana Loki” section in the administrator manual).

3.3.1 LUNA Dashboards

Note: To work with Grafana you need to use InfluxDB version 2.

Note: Before updating, make sure that the old LUNA Dashboards container is deleted.

3.3.1.1 Run LUNA Dashboards container

Use the `docker run` command with these parameters to run Grafana:

```
docker run \  
--restart=always \  
--detach=true \  
--network=host \  
--name=grafana \  
-v /etc/localtime:/etc/localtime:ro \  
dockerhub.visionlabs.ru/luna/luna-dashboards:v.0.1.0
```

Use “`http://IP_ADDRESS:3000`” to go to the Grafana web interface when the LUNA Dashboards and InfluxDB containers are running.

3.3.2 Grafana Loki

Note: Grafana Loki requires LUNA Dashboards to be running.

Note: Before updating, make sure that the old Grafana Loki and Promtail containers are removed.

3.3.2.1 Run Grafana Loki container

Use the `docker run` command with these parameters to run Grafana Loki:

```
docker run \  
--name=loki \  
--restart=always \  
--detach=true \  
--network=host \  
-v /etc/localtime:/etc/localtime:ro \  
dockerhub.visionlabs.ru/luna/loki:2.7.1
```

3.3.2.2 Run Promtail container

Use the `docker run` command with these parameters to run Promtail:

```
docker run \  
-v /var/lib/luna/current/example-docker/logging/promtail.yml:/etc/promtail/  
  luna.yml \  
-v /var/lib/docker/containers:/var/lib/docker/containers \  
-v /etc/localtime:/etc/localtime:ro \  
--name=promtail \  
--restart=always \  
--detach=true \  
--network=host \  
dockerhub.visionlabs.ru/luna/promtail:2.7.1 \  
-config.file=/etc/promtail/luna.yml -client.url=http://127.0.0.1:3100/loki/  
  api/v1/push -client.external-labels=job=containerlogs,pipeline_id=,job_id  
  =,version=
```

Here:

- `-v /var/lib/luna/current/example-docker/logging/promtail.yml:/etc/promtail/luna.yml` — Mounting the configuration file to the Promtail container.
- `-config.file=/etc/promtail/luna.yml` — Flag with the address of the configuration file.
- `-client.url=http://127.0.0.1:3100/loki/api/v1/push` — Flag with the address of deployed Grafana Loki.
- `-client.external-labels=job=containerlogs,pipeline_id=,job_id=,version=` — Static labels to add to all logs sent to Grafana Loki.

3.4 Docker commands

3.4.1 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
```

3.4.2 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>
```

3.4.3 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
```

3.4.4 Images names

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

```
docker images
```

3.4.5 Delete image

If you need to delete an image:

- Run the `docker images` command.

- Find the required image, for example dockerhub.visionlabs.ru/luna/luna-image-store.
- 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)
```

3.4.6 Stop container

You can stop the container using the command:

```
docker stop <container_name>
```

Stop all the containers:

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

3.4.7 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/luna-image-store.
- 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)
```

3.4.7.1 Check service logs

You can use the following command to show logs for the service:

```
docker logs <container_name>
```

3.5 Launching parameters description

When launching a Docker container for a LUNA PLATFORM service you should specify additional parameters required for the service launching.

The parameters specific for a particular container are described in the section about this container launching.

All the parameters given in the service launching example are required for proper service launching and utilization.

3.5.1 Launching services parameters

Example command of launching LP services containers:

```
docker run \  
--env=CONFIGURATOR_HOST=127.0.0.1 \  
--env=CONFIGURATOR_PORT=5070 \  
--env=PORT=<Port_of_the_launched_service> \  
--env=WORKER_COUNT=1 \  
--env=RELOAD_CONFIG=1 \  
--env=RELOAD_CONFIG_INTERVAL=10 \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/<service>:/srv/logs/ \  
--name=<service_container_name> \  
--restart=always \  
--detach=true \  
--network=host \  
dockerhub.visionlabs.ru/luna/<service-name>:<version>
```

The following parameters are used when launching LP services containers:

- `docker run` — Command for running the selected image as a new container.
- `dockerhub.visionlabs.ru/luna/<service-name>:<version>` — Sets the image required for the container launching.

Links to download the container images you need are available in the description of the corresponding container launching.

- `--network=host` — Sets that a network is not simulated and the server network is used. If you need to change the port for third-party party containers, 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. The example is given for PostgreSQL.

- `--env=` — Sets the environment variables required to run the container (see the “[Service arguments](#)” section).
- `--name=<service_container_name>` — Sets the name of the launched container. The name must be unique. If there is a container with the same name, an error will occur.
- `--restart=always` — Sets a restart policy. The daemon will always restart the container regardless of the exit status.
- `--detach=true` — Run the container in the background mode.
- `-v` — Enables you to mount the content of a server folder into a volume in the container. Thus their contents will synchronize. The following general data is mounted:
- `/etc/localtime:/etc/localtime:ro` — Sets the current time zone used by the system in the container.
- `/tmp/logs/<service>:/srv/logs/` — Enables copying of the folder with service logs to your server `/tmp/logs/<service>` directory. You can change the directory where the logs will be saved according to your needs.

3.5.1.1 Service arguments

Each service in LUNA PLATFORM has its own launch arguments. These arguments can be passed through:

- Setting a flag for the launch script (`run.py`) of the corresponding service.
- Setting environment variables (`--env`) on the Docker command line.

For example, using the `--help` flag you can get a list of all available arguments. An example of passing an argument to an API service:

```
docker run --rm dockerhub.visionlabs.ru/luna/luna-api:v.6.29.0 python3 /srv/luna_api/run.py --help
```

List of main arguments:

Launch flag	Environment variable	Description
<code>--port</code>	PORT	Port on which the service will listen for connections.
<code>--workers</code>	WORKER_COUNT	Number of workers for the service.
<code>--log_suffix</code> <code>--log_suffix</code>	LOG_SUFFIX LOG_SUFFIX	Suffix added to log file names (with the option to write logs to a file enabled).

<code>--config-reload</code>	RELOAD_CONFIG	Enable automatic configuration reload. See “Automatic configurations reload” in the LUNA PLATFORM 5 administrator manual.
<code>--pulling-time</code>	RELOAD_CONFIG_INTERVAL	Configuration checking period (default 10 seconds). See “Automatic configurations reload” in the LUNA PLATFORM 5 administrator manual.
<code>--luna-config</code> <code>--luna-config</code>	CONFIGURATOR_HOST, CONFIGURATOR_PORT	Address of the Configurator service for downloading settings. For <code>--luna-config</code> it is sent in the format <code>http://localhost:5070/1</code> . For environment variables, the host and port are set explicitly. If the argument is not given, the default configuration file will be used.
<code>--config</code>	None	Path to the file with service configurations.
<code>--<config_name></code>	<code>--EXTEND_CMD=<config_name></code>	Tag of the specified configuration in the Configurator. When setting this configuration, the value of the tagged configuration will be used. Example: <code>--INFLUX_MONITORING TAG_1</code> Note: You must pre-tag the appropriate configuration in. Configurator. Note: Only works with the <code>--luna-config</code> flag.
<code>--tls_cert</code>	None	Path to the SSL certificate for launching the service using the HTTPS protocol.
<code>--tls_key</code>	None	Path to the SSL private key for launching the service using the HTTPS protocol.
<code>--tls_key_pass</code>	None	Password for the SSL private key for launching the service using the HTTPS protocol.

The list of arguments may vary depending on the service.

It is also possible to override the settings of services at their start using environment variables.

The VL_SETTINGS prefix is used to redefine the settings. Examples:

- `--env=VL_SETTINGS.INFLUX_MONITORING.SEND_DATA_FOR_MONITORING=0`. Using the environment variable from this example will set the “SEND_DATA_FOR_MONITORING” setting for the INFLUX_MONITORING section to “0”.
- `--env=VL_SETTINGS.OTHER.STORAGE_TIME=LOCAL`. For non-compound settings (settings that are located in the “OTHER” section in the configuration file), you must specify the “OTHER” prefix. Using the environment variable from this example will set the value of the “STORAGE_TIME” setting (if the service uses this setting) to “LOCAL”.

Passing flags using environment variable

Flags for which an environment variable is not explicitly allocated can be passed using the environment variable EXTEND_CMD.

For example, you can pass the configurations tag in the following way:

```
--env=EXTEND_CMD="--INFLUX_MONITORING=TAG_1 --LUNA_EVENTS_DB=TAG_2"
```

3.5.2 Creating DB parameters

Example command of launching containers for database migration or database creation:

```
docker run \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/<service>:/srv/logs/ \  
--rm \  
--network=host \  
dockerhub.visionlabs.ru/luna/<service-name>:<version> \  
python3 ./base_scripts/db_create.py --luna-config http://localhost:5070/1
```

The following parameters are used when launching containers for database migration or database creation:

Here:

- `--rm` — Sets if the container is deleted after all the specified scripts finish processing.
- `python3 ./base_scripts/db_create.py` — Sets Python version and a script `db_create.py` launched in the container. The script is used for the database structure creation.
- `--luna-config http://localhost:5070/1` — Sets where the launched script should receive configurations. By default, the service requests configurations from the Configurator service.

3.6 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 LP service containers.
- Configure synchronization of log directories in the container with logs on the server using the `volume` argument at the start of each container.

3.6.1 Create logs directory

Below are examples of commands for creating directories for saving logs and assigning rights to them for all LUNA PLATFORM services.

```
mkdir -p /tmp/logs/configurator /tmp/logs/image-store /tmp/logs/accounts /  
tmp/logs/faces /tmp/logs/licenses /tmp/logs/events /tmp/logs/python-  
matcher /tmp/logs/handlers /tmp/logs/remote-sdk /tmp/logs/tasks /tmp/logs  
/tasks-worker /tmp/logs/sender /tmp/logs/api /tmp/logs/admin /tmp/logs/  
backport3 /tmp/logs/backport4
```

```
chown -R 1001:0 /tmp/logs/configurator /tmp/logs/image-store /tmp/logs/  
accounts /tmp/logs/faces /tmp/logs/licenses /tmp/logs/events /tmp/logs/  
python-matcher /tmp/logs/handlers /tmp/logs/remote-sdk /tmp/logs/tasks /  
tmp/logs/tasks-worker /tmp/logs/sender /tmp/logs/api /tmp/logs/admin /tmp  
/logs/backport3 /tmp/logs/backport4
```

If you need to use the Python Matcher Proxy service, then you need to additionally create the `/tmp/logs/python-matcher-proxy` directory and set its permissions.

3.6.2 Logging activation

3.6.2.1 LP services 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 settings for each service.

Automatic method (before/after starting Configurator)

To update logging settings, you can use the `logging.json` settings file provided with the distribution package.

Run the following command after starting the Configurator service:

```
docker cp /var/lib/luna/current/extras/conf/logging.json luna-configurator:/
srv/luna_configurator/used_dumps/logging.json
```

Update your logging settings with the copied file.

```
docker exec -it luna-configurator python3 ./base_scripts/db_create.py --dump
-file /srv/luna_configurator/used_dumps/logging.json
```

Manual method (after starting Configurator)

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 all services whose logs need to be saved. For example, you can use the path `/srv/logs`.

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

3.6.2.2 Configurator service logging activation (before/after Configurator start)

The Configurator service settings are not located in the Configurator user interface, they are located in the following file:

```
/var/lib/luna/current/example-docker/luna_configurator/configs/
luna_configurator_postgres.conf
```

You should change the logging parameters in this file before starting the Configurator service or restart it after making changes.

Set the path to the logs location in the container in the `FOLDER_WITH_LOGS = ./` parameter of the file. For example, `FOLDER_WITH_LOGS = /srv/logs`.

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

3.6.3 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 API service with mounting a directory with logs:

```
docker run \  
--env=CONFIGURATOR_HOST=127.0.0.1 \  
--env=CONFIGURATOR_PORT=5070 \  
--env=PORT=5000 \  
--env=WORKER_COUNT=1 \  
--env=RELOAD_CONFIG=1 \  
--env=RELOAD_CONFIG_INTERVAL=10 \  
--name=luna-api \  
--restart=always \  
--detach=true \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/api:/srv/logs \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-api:v.6.29.0
```

The example container launch commands in this documentation contain these arguments.

3.7 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.

3.8 Set custom InfluxDB settings

If you are going to use InfluxDB OSS 2, then you need to update the monitoring settings in Configurator service.

There are the following settings for InfluxDB OSS 2:

```
"send_data_for_monitoring": 1,
"use_ssl": 0,
"flushing_period": 1,
"host": "127.0.0.1",
"port": 8086,
"organization": "<ORGANIZATION_NAME>",
"token": "<TOKEN>",
"bucket": "<BUCKET_NAME>",
"version": <DB_VERSION>
```

You can update InfluxDB settings in the Configurator service by following these steps:

- Open the following file:

```
vi /var/lib/luna/current/extras/conf/influx2.json
```

- Set required data.
- Save changes.
- Copy the file to the influxDB container:

```
docker cp /var/lib/luna/current/extras/conf/influx2.json luna-configurator:/srv/
```

- Update settings in the Configurator.

```
docker exec -it luna-configurator python3 ./base_scripts/db_create.py --dump
-file /srv/influx2.json
```

You can also manually update settings in the Configurator service user interface.

The Configurator service configurations are set separately.

- Open the file with the Configurator configurations:

```
vi /var/lib/luna/current/example-docker/luna_configurator/configs/
luna_configurator_postgres.conf
```

- Set required data.
- Save changes.
- Restart Configurator:

```
docker restart luna-configurator
```

3.9 Use Python Matcher with Python Matcher Proxy

As mentioned earlier, along with the Python Matcher service, you can additionally use the Python Matcher Proxy service, which will redirect matching requests either to the Python Matcher service or to the matching plugins. Plugins may significantly improve matching processing performance. For example, it is possible to organize the storage of the data required for matching operations and additional objects fields in separate storage using plugins, which will speed up access to the data compared to the use of the standard LUNA PLATFORM database.

To use the Python Matcher service with Python Matcher Proxy, you should additionally launch the appropriate container, and then set a certain setting in the Configurator service. Follow the steps below only if you are going to use matching plugins.

See the description and usage of matching plugins in the administrator manual.

3.9.1 Python Matcher proxy container launch

Use the following command to launch the service:

After starting the container, you need to set the "luna_matcher_proxy":true parameter in the "ADDITIONAL_SERVICES_USAGE" section in the Configurator service.

```
docker run \  
--env=CONFIGURATOR_HOST=127.0.0.1 \  
--env=CONFIGURATOR_PORT=5070 \  
--env=PORT=5110 \  
--env=WORKER_COUNT=1 \  
--env=RELOAD_CONFIG=1 \  
--env=RELOAD_CONFIG_INTERVAL=10 \  
--env=SERVICE_TYPE="proxy" \  
-v /etc/localtime:/etc/localtime:ro \  
-v /tmp/logs/python-matcher-proxy:/srv/logs \  
--name=luna-python-matcher-proxy \  
--restart=always \  
--detach=true \  
--network=host \  
dockerhub.visionlabs.ru/luna/luna-python-matcher:v.1.9.9
```

After launching the container, you need to set the following value in the Configurator service.

```
ADDITIONAL_SERVICES_USAGE = "luna_matcher_proxy":true
```

3.10 Move old data to root directory

The official installation manual for LUNA PLATFORM v.5.40.0 and higher provides example commands for creating directories for storing PostgreSQL, InfluxDB and Image Store data in the root directory `/var/lib/luna/<db_or_bucket_folder>`. In older versions of LUNA PLATFORM, example commands were given to create directories within a specific LUNA PLATFORM version directory `/var/lib/luna/current/example-docker/<db_or_bucket_folder>`.

Storing directories in the root directory allows you not to transfer data during the next update.

Example commands for launching PostgreSQL, InfluxDB and Image Store containers contain arguments for mounting the corresponding data directories from the root directory. If you are upgrading from version LUNA PLATFORM v.5.38.3 and lower and the previous version of LUNA PLATFORM was installed according to the official documentation, then in the container launch commands, specify the old directories with the mounted data or first transfer the old data according to the sections below.

The sections below provide example commands for version 5.38.3.

Important: Before transferring old data, stop the corresponding containers, because New data may appear during the transfer process.

3.10.1 Move Image Store buckets

The following step is required if you are storing Image Store buckets in the default directory.

Copy the “image_store” folder with all its buckets.

```
mv /var/lib/luna/luna_v.5.38.3/example-docker/image_store /var/lib/luna/
```

Set rights for the luna user.

```
chown -R 1001:0 /var/lib/luna/image_store
```

3.10.2 Move PostgreSQL data

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

Copy the “data” folder.

```
mv /var/lib/luna/luna_v.5.38.3/example-docker/postgresql /var/lib/luna/
```

3.10.3 Move InfluxDB Data

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

Copy the “influx” folder with all its buckets.

```
mv /var/lib/luna/luna_v.5.38.3/example-docker/influx /var/lib/luna/
```