



**VisionLabs**  
MACHINES CAN SEE

# **VisionLabs LUNA PLATFORM 5**

**Installation manual using Storages utility**

**v.5.67.0**

## Contents

<b>Default ports for services</b>	<b>5</b>
<b>Configuration names for services</b>	<b>6</b>
<b>Introduction</b>	<b>7</b>
<b>1 Before upgrade</b>	<b>8</b>
1.1 Key changes from previous versions . . . . .	9
1.2 Create backups . . . . .	10
1.2.1 Backup of PostgreSQL DMBS . . . . .	10
1.2.2 Backup of Influx database . . . . .	10
1.2.3 Backup of Image Store buckets . . . . .	10
1.2.4 Dump file with service settings . . . . .	10
1.3 Prepare to change the neural network version . . . . .	11
1.3.1 Change neural network version using Configurator . . . . .	12
1.4 Delete old symbolic link . . . . .	12
1.5 Distribution unpacking . . . . .	12
1.6 Symbolic link creation . . . . .	13
1.7 Changing group and owner for directories . . . . .	13
1.8 Save user configurations of the Configurator . . . . .	13
1.9 Create log directory for new services . . . . .	14
1.10 License update . . . . .	15
1.10.1 Actions from License activation manual . . . . .	15
1.11 Calculations using GPU . . . . .	15
1.12 Remove old containers (optional) . . . . .	16
<b>2 Run third-party services</b>	<b>18</b>
2.1 InfluxDB . . . . .	18
2.2 PostgreSQL . . . . .	19
2.2.1 Migrate PostgreSQL 12 to PostgreSQL 16 . . . . .	19
2.2.2 Launch PostgreSQL . . . . .	19
2.3 Redis . . . . .	20
<b>3 Update environment</b>	<b>21</b>
3.1 Update environment command . . . . .	21
3.2 Optional services usage . . . . .	22
3.3 Load dump file . . . . .	22

<b>4</b>	<b>Services launch</b>	<b>24</b>
4.1	Configurator . . . . .	25
4.1.1	Run Configurator container . . . . .	25
4.2	Image Store . . . . .	26
4.2.1	Image Store container launch . . . . .	26
4.3	Accounts . . . . .	27
4.3.1	Accounts container launch . . . . .	27
4.4	Licenses . . . . .	28
4.4.1	Specify license settings using Configurator . . . . .	28
4.4.2	Licenses container launch . . . . .	29
4.5	Faces . . . . .	30
4.5.1	Faces container launch . . . . .	30
4.6	Events . . . . .	31
4.6.1	Events container launch . . . . .	31
4.7	Python Matcher services . . . . .	32
4.7.1	Use Python Matcher without Python Matcher Proxy . . . . .	32
4.7.2	Python Matcher container launch . . . . .	32
4.8	Remote SDK . . . . .	33
4.8.1	Change the neural network model for extracting descriptors . . . . .	33
4.8.2	Remote SDK container launch . . . . .	33
4.9	Handlers . . . . .	38
4.9.1	Handlers container launch . . . . .	38
4.10	Tasks . . . . .	39
4.10.1	Tasks and Tasks Worker containers launch . . . . .	39
4.11	Sender . . . . .	41
4.11.1	Sender container launch . . . . .	41
4.12	API . . . . .	42
4.12.1	API container launch . . . . .	42
4.13	Admin . . . . .	43
4.13.1	Admin container launch . . . . .	43
4.14	Backport 3 . . . . .	44
4.14.1	Backport 3 container launch . . . . .	44
4.14.2	User Interface 3 . . . . .	45
4.15	Backport 4 . . . . .	46
4.15.1	Backport 4 container launch . . . . .	46
4.15.2	User Interface 4 . . . . .	47
<b>5</b>	<b>Additional information</b>	<b>48</b>
5.1	Account creation . . . . .	49
5.2	GC task schedule creation . . . . .	50

5.3	Monitoring and logs visualization using Grafana . . . . .	51
5.3.1	LUNA Dashboards . . . . .	51
5.3.2	Grafana Loki . . . . .	51
5.4	Docker commands . . . . .	53
5.4.1	Show containers . . . . .	53
5.4.2	Copy files to container . . . . .	53
5.4.3	Enter container . . . . .	53
5.4.4	Images names . . . . .	53
5.4.5	Delete image . . . . .	53
5.4.6	Stop container . . . . .	54
5.4.7	Delete container . . . . .	54
5.5	Launching parameters description . . . . .	56
5.5.1	Launching services parameters . . . . .	56
5.5.2	Creating DB parameters . . . . .	59
5.6	Logging to server . . . . .	60
5.6.1	Create logs directory . . . . .	60
5.6.2	Logging activation . . . . .	60
5.6.3	Mounting directories with logs when starting services . . . . .	61
5.7	Docker log rotation . . . . .	63
5.8	Set custom InfluxDB settings . . . . .	64
5.9	Use Python Matcher with Python Matcher Proxy . . . . .	66
5.9.1	Python Matcher proxy container launch . . . . .	66

## 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 provides examples of steps to update the environment using the Storage utility and then launch LUNA PLATFORM containers.

It is recommended to familiarize yourself with the Storages utility manual before deploying the 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.

**Important:** Updating using Storages is only possible for versions of LUNA PLATFORM v.5.46.1 and higher. If you are upgrading from an earlier version, use the normal upgrade manual to upgrade to the current version, or upgrade to v.5.46.1 and then use this manual.

Note that starting from versions 5.46.1, 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 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.
- [“Run third-party services”](#) — Launching PostgreSQL, Redis, Influx databases.
- [“Update environment”](#) — Updating the environment (migration of databases, settings, etc.).
- [“Services launch”](#) — Launching containers with LUNA PLATFORM services.

The section [“Additional information”](#) provides useful information on the description of service launch parameters, Docker commands, using Grafana, 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. [Save user configurations of the Configurator service](#) if they have changed.
9. [Create log directories for new services](#) if you have previously used logging to file.
10. [Update license](#) if necessary.
11. [Set up GPU computing](#) if you plan to use GPU.
12. [Remove old containers](#) if necessary.



## 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 <b>62</b> for extracting face descriptors is used.	Read the information described in the section “ <a href="#">Prepare to change the neural network version</a> ”, 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 <a href="#">official documentation</a> .

## 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 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
```

## 1.9 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.

## 1.10 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.

### 1.10.1 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.

## 1.11 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.

## 1.12 Remove old containers (optional)

**Note:** Removing old containers is not necessary when preparing your environment using the Storages utility. It is enough to simply limit the number of queries to the database, but in this case it is necessary to understand certain consequences. See the “Recommendations services behavior during environment preparation” section in the Storages utility manual.

**Important:** The Image Store service is an exception. During the preparation of the bucket environment, Storages makes a request to the Image Store service to obtain information about the buckets. If the Image Store container is removed, the request will fail and a migration error will occur. To migrate buckets when the Image Store container is removed, you must explicitly tell the Storages utility the location of the buckets using the `--local-buckets` or `--s3-buckets` arguments. When using local buckets, you must also mount the directory with the buckets to the Storages container.

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 Run third-party services

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

### 2.1 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
```

## 2.2 PostgreSQL

### 2.2.1 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-`vlmatch: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.

### 2.2.2 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 /etc/localtime:/etc/localtime:ro \
--name=postgres \
--restart=always \
--detach=true \
--network=host \
dockerhub.visionlabs.ru/luna/postgis-vlmatch:16
```

`-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.

## 2.3 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
```

### 3 Update environment

The environment is prepared using the Storages service. Using the command below will execute:

- Creating buckets in InfluxDB for monitoring work (if not done previously)
- Creating buckets for the Image Store service (if not done previously)
- Preparing the Influx database for collecting aggregated statistics using the Admin service (if not done previously)
- Database migration
- Migration of settings to the Configurator database

When updating the environment, the default configuration file of the Storages service will be used, containing all the standard settings for connecting to databases, buckets, etc. If you need to use non-default settings or update LUNA PLATFORM deployed on different servers, you need to edit the configuration file before running the environment update command:

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

If necessary, you can first update the environment for the Configurator service, start it, and then update the environment for all other services using the settings from the running Configurator service. See the Storages utility manual for examples and detailed information about Storages.

#### 3.1 Update environment command

Prepare the environment with the following command:

```
docker run \
--rm \
-v /var/lib/luna/current/extras/conf/storages_config.conf:/srv/
storages_config.conf \
--network=host \
dockerhub.visionlabs.ru/luna/storages:v.0.2.51 \
bash -c "luna_prepare prepare all_entities \
--platform_version=v.5.67.0 \
--config=/srv/storages_config.conf \
--profile=backports"
```

Here:

- `luna_prepare prepare all_entities` — Command “prepare” to prepare all entities.
- `--platform_version` — Named argument containing the LUNA PLATFORM version.
- `--profile` — Named argument containing the profile (list of services) backports, which means that the environment will be prepared for all services, including Backport 3 and Backport 4

services.

- `-v /var/lib/luna/current/extras/conf/storages_config.conf:/srv/storages_config.conf` — Command to mount the Storages configuration file
- `--config=/srv/storages_config.conf` — Named argument containing the address of the configuration file for use by the Storages service.

If the old Image Store container is deleted, then you must explicitly specify the location of the buckets using the `--local-buckets` or `--s3-buckets` arguments, and also mount the directory with local buckets to the Storages container.

### 3.2 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.

Storages service will not prepare the environment for services that are disabled in the “ADDITIONAL\_SERVICES\_USAGE” setting.

### 3.3 Load dump file

Run the following command to upload the dump file to the Configurator:

```
docker run \
--rm \
--network=host \
-v /var/lib/luna/current/extras/conf/platform_settings.json:/srv/
platform_settings.json \
dockerhub.visionlabs.ru/luna/storages:v.0.2.51 \
bash -c "luna_prepare load_dump \
--dump-file=/srv/platform_settings.json"
```

Here:

- `luna_prepare load_dump` — Command “load\_dump”, which enables you to upload a dump file to the Configurator database.
- `-v /var/lib/luna/current/extras/conf/platform_settings.json:/srv/platform_settings.json \` — Command to mount the dump file `platform_settings.json`.
- `--dump-file=/srv/platform_settings.json` — Named argument containing the address of the dump file inside the container.

## 4 Services launch

This section provides examples of commands for launching LUNA PLATFORM services.

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.



## 4.1 Configurator

### 4.1.1 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).

## 4.2 Image Store

### 4.2.1 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/`.

## 4.3 Accounts

### 4.3.1 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
```

## 4.4 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.

### 4.4.1 Specify license settings using Configurator

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

#### 4.4.1.1 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.

#### 4.4.1.2 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.

#### 4.4.2 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
```

## 4.5 Faces

### 4.5.1 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
```

## 4.6 Events

### 4.6.1 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
```

## 4.7 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.

### 4.7.1 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.

### 4.7.2 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
```



## 4.8 Remote SDK

### 4.8.1 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.

### 4.8.2 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.

#### 4.8.2.1 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

```

#### 4.8.2.2 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.

#### 4.8.2.3 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
```

## 4.9 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”](#).

### 4.9.1 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
```

## 4.10 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”](#).

### 4.10.1 Tasks and Tasks Worker containers launch

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

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.

#### 4.10.1.1 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
```

#### 4.10.1.2 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
```



## 4.11 Sender

### 4.11.1 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
```

## 4.12 API

### 4.12.1 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
```

See the example command for creating a new account in the [“Account creation”](#) section.

It is recommended that you create a schedule for the garbage collection task if you have not already created one. See [“Create GC task schedule”](#) for an example command.

## 4.13 Admin

### 4.13.1 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
```

## 4.14 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.

### 4.14.1 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
```

### 4.14.2 User Interface 3

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

#### 4.14.2.1 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.

## 4.15 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.

### 4.15.1 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
```

### 4.15.2 User Interface 4

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

#### 4.15.2.1 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=dXNlcKBTYWlsLmNvbTpwYXNzd29yZA== \  
--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).

## 5 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.](#)



## 5.1 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.

## 5.2 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.

## 5.3 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).

### 5.3.1 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.

#### 5.3.1.1 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.

### 5.3.2 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.

#### 5.3.2.1 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
```

### 5.3.2.2 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.

## 5.4 Docker commands

### 5.4.1 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
```

### 5.4.2 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>
```

### 5.4.3 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
```

### 5.4.4 Images names

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

```
docker images
```

### 5.4.5 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](https://hub.docker.com/r/visionlabs/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)
```

#### 5.4.6 Stop container

You can stop the container using the command:

```
docker stop <container_name>
```

Stop all the containers:

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

#### 5.4.7 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](https://hub.docker.com/r/visionlabs/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)
```

##### 5.4.7.1 Check service logs

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

```
docker logs <container_name>
```

## 5.5 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.

### 5.5.1 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 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.

#### 5.5.1.1 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>--&lt;config_name&gt;</code>	<code>--EXTEND_CMD=&lt;config_name&gt;</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> <b>Note:</b> You must pre-tag the appropriate configuration in. Configurator. <b>Note:</b> 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"
```

### 5.5.2 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.

## 5.6 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.

### 5.6.1 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.

### 5.6.2 Logging activation

#### 5.6.2.1 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.

#### 5.6.2.2 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.

### 5.6.3 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.

## 5.7 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.

## 5.8 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
```

## 5.9 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.

### 5.9.1 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
```