

VisionLabs LUNA PLATFORM 5

Руководство по обновлению с использованием утилиты Storages

v.5.88.0

Содержание

Порты сервисов по умолчанию	6
Названия сервисов в Configurator	7
Введение	8
1 Подготовка к обновлению	9
1.1 Ключевые изменения предыдущих версий	10
1.2 Создание резервных копий	12
1.2.1 Копия СУБД PostgreSQL	12
1.2.2 Копия базы данных Influx	12
1.2.3 Копия бакетов Image Store	12
1.2.4 Дамп-файл с настройками сервисов	12
1.3 Подготовка к смене версии нейронной сети	13
1.3.1 Смена версии нейронной сети в Configurator	15
1.4 Удаление символической ссылки	15
1.5 Распаковка дистрибутива	16
1.6 Создание символической ссылки	16
1.7 Изменение группы и владельца для директорий	17
1.8 Сохранение пользовательских настроек сервиса Configurator	17
1.9 Создание директории логов для новых сервисов	18
1.10 Обновление лицензии	18
1.10.1 Действия из руководства по активации лицензии	19
1.11 Вычисления с помощью GPU	19
1.12 Удаление старых контейнеров (опционально)	20
2 Запуск сторонних сервисов	21
2.1 InfluxDB	21
2.2 PostgreSQL	22
2.2.1 Миграция с PostgreSQL 12 на PostgreSQL 16	22
2.2.2 Запуск контейнера PostgreSQL	22
2.3 Redis	22
3 Обновление окружения	24
3.1 Команда обновления окружения	24
4 Запуск сервисов	26
4.1 Configurator	28
4.1.1 Запуск контейнера Configurator	28

4.2	Image Store	29
4.2.1	Запуск контейнера Image Store	29
4.3	Accounts	30
4.3.1	Запуск контейнера Accounts	30
4.4	Licenses	31
4.4.1	Задание настроек лицензии с помощью Configurator	31
4.4.1.1	Задание настроек лицензии HASP	31
4.4.1.2	Задание настроек лицензии Guardant	31
4.4.2	Запуск контейнера Licenses	32
4.5	Faces	33
4.5.1	Запуск контейнера Faces	33
4.6	Events	34
4.6.1	Запуск контейнера Events	34
4.7	Сервисы Python Matcher	35
4.7.1	Использование Python Matcher без Python Matcher Proxy	35
4.7.2	Запуск контейнера Python Matcher	35
4.8	Remote SDK	37
4.8.1	Смена модели нейронной сети для извлечения биометрических шаблонов	37
4.8.2	Запуск контейнера Remote SDK	37
4.8.2.1	Запуск Remote SDK с использованием CPU	37
4.8.2.2	Запуск Remote SDK с использованием GPU	38
4.8.2.3	Запуск облегченной версии Remote SDK	39
4.9	Handlers	42
4.9.1	Запуск контейнера Handlers	42
4.10	Tasks	43
4.10.1	Запуск контейнеров Tasks и Tasks Worker	43
4.10.1.1	Запуск контейнера Tasks worker	43
4.10.1.2	Запуск контейнера Tasks	43
4.11	Sender	45
4.11.1	Запуск контейнера Sender	45
4.12	Сервисы видеоаналитики	46
4.12.1	Запуск контейнера Video Manager	46
4.12.2	Запуск контейнера Video Agent	46
4.13	API	48
4.13.1	Запуск контейнера API	48
4.14	Admin	49
4.14.1	Запуск контейнера Admin	49
4.15	Backport 3	50
4.15.1	Запуск контейнера Backport 3	50

4.15.2	User Interface 3	51
4.15.2.1	Запуск контейнера User Interface 3	51
4.16	Backport 4	52
4.16.1	Запуск контейнера Backport 4	52
4.16.2	User Interface 4	53
4.16.2.1	Запуск контейнера User Interface 4	53
5	Дополнительная информация	54
5.1	Создание аккаунта	55
5.2	Создание расписания задачи GC	56
5.3	Управление шифрованием биометрических шаблонов	57
5.3.0.1	Запуск скрипта миграции биометрических шаблонов	57
5.4	Визуализация мониторинга и логов с помощью Grafana	59
5.4.1	LUNA Dashboards	59
5.4.1.1	Запуск контейнера LUNA Dashboards	59
5.4.2	Grafana Loki	59
5.4.2.1	Запуск контейнера Grafana Loki	59
5.4.2.2	Запуск контейнера Promtail	60
5.5	Команды Docker	61
5.5.1	Показать контейнеры	61
5.5.2	Копировать файлы в контейнер	61
5.5.3	Вход в контейнер	61
5.5.4	Имена образов	61
5.5.5	Удаление образа	61
5.5.6	Остановка контейнера	62
5.5.7	Удаление контейнера	62
5.5.7.1	Проверка логов сервисов	63
5.6	Описание параметров запуска	64
5.6.1	Параметры запуска сервисов	64
5.6.1.1	Аргументы сервисов	65
5.6.2	Параметры создания баз данных	67
5.7	Запись логов на сервер	69
5.7.1	Создание директории логов	69
5.7.2	Активация записи логов	69
5.7.2.1	Активация записи логов сервисов LP	69
5.7.2.2	Активация записи логов сервиса Configurator (перед/после запуска Configurator)	70
5.7.3	Мониторинг директорий с логами при старте сервисов	70
5.8	Настройка ротации логов Docker	72
5.8.1	Задание пользовательских настроек InfluxDB	73

5.9	Использование Python Matcher с Python Matcher Proxy	75
5.9.1	Запуск контейнера Python Matcher Proxy	75

Порты сервисов по умолчанию

Название сервиса	Порт
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 Video Manager	5230
LUNA PLATFORM Video Agent	5240
LUNA PLATFORM 3 User Interface	4100
LUNA PLATFORM 4 User Interface	4200
Oracle DB	1521
PostgreSQL	5432
Redis DB	6379
InfluxDB	8086
Grafana	3000

Названия сервисов в Configurator

Таблица ниже включает в себя названия сервисов в сервисе Configurator. Данные параметры используются для конфигурации сервисов.

Сервис	Название сервиса в 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
Video Agent	luna-video-agent
Video Manager	luna-video-manager
Backport 3	luna-backport3
Backport 4	luna-backport4

Настройки для сервиса Configurator устанавливаются в его конфигурационном файле.

Введение

В данном документе приводятся примеры шагов для обновления окружения с помощью утилиты Storages и последующего запуска контейнеров LUNA PLATFORM.

Рекомендуется ознакомиться с руководством по утилите Storages перед развертыванием LUNA PLATFORM.

Данный документ включает в себя пример развертывания LUNA PLATFORM. LUNA PLATFORM разворачивается в минимальной рабочей конфигурации для использования в демонстрационных целях. Данная конфигурация не является достаточной для реальной эксплуатации системы в продуктивном контуре.

Важно! Обновление с помощью Storages возможно только для версий LUNA PLATFORM v.5.46.1 и выше. Если вы обновляете с более ранней версии, то воспользуйтесь обычным руководством по обновлению для обновления до последней версии или обновитесь до версии v.5.46.1, а затем воспользуйтесь данным руководством.

Обратите внимание, что начиная с версий 5.46.1 могли произойти критические изменения, такие как обновления порогов, версий нейронных сетей, прекращение поддержки FaceDetV1 и FaceDetV2 и другие (см. полный перечень критических изменений в разделе [«Ключевые изменения предыдущих версий»](#)). Такое изменение как обновление порогов может давать другой результат при выполнении оценивания, нежели в старой сборке. Данные команды помечены соответствующим образом. Будьте внимательны и не выполняйте лишних действий если обновляете с версии LUNA PLATFORM v.5.86.0.

Для обновления LUNA PLATFORM нужно выполнить действия из следующих разделов:

- [«Подготовка к обновлению»](#) — действия по распаковке архивов, подготовке директорий, настройке лицензии и пр. Некоторые действия могут быть опциональными.
- [«Запуск сторонних сервисов»](#) — запуск СУБД PostgreSQL и БД Redis, Influx
- [«Обновление окружения»](#) — обновление окружения (миграция баз данных, настроек и пр.)
- [«Запуск сервисов»](#) — запуск контейнеров с сервисами LUNA PLATFORM

В разделе [«Дополнительная информация»](#) приводится полезная информация по описанию параметров запуска сервисов, командах Docker, включении Grafana для визуализации мониторинга и пр.

Данное руководство написано с предположением, что:

- предыдущая минорная версия LUNA PLATFORM уже установлена, и требуемое окружение на сервере готово к работе.
- LP 5 установлена в соответствии с руководством по установке, и используются пути по умолчанию. В противном случае следует внести изменения вручную в процессе обновления.

1 Подготовка к обновлению

Убедитесь в том, что вы являетесь **root**-пользователем перед тем, как начать обновление!

Перед обновлением необходимо выполнить следующие действия:

1. [Ознакомиться с ключевыми изменениями предыдущих версий](#), если выполняется обновление с версии, отличной от версии LUNA PLATFORM v.5.86.0
2. [Создать резервные копии](#)
3. [Подготовиться к смене версии нейронной сети для извлечения биометрических шаблонов](#), если это необходимо
4. [Удалить старую символическую ссылку](#)
5. [Распаковать дистрибутив новой версии LUNA PLATFORM](#)
6. [Создать новую символическую ссылку](#)
7. [Изменить группу и владельца для новых директорий](#)
8. [Сохранить пользовательские настройки сервиса Configurator](#), если они изменялись
9. [Создать директории с логами](#), если ранее использовалась запись логов в файлы
10. [Обновить лицензию](#), если это необходимо
11. [Настроить вычисления с помощью GPU](#), если планируется использовать GPU
12. [Удалить старые контейнеры](#), если это необходимо

1.1 Ключевые изменения предыдущих версий

Примечание. При обновлении LUNA PLATFORM с предыдущей версии, пропустите данный раздел.

Ниже перечислены ключевые изменения предыдущих версий, на которые надо обратить внимание при выполнении обновления со старых версий LUNA PLATFORM. Для некоторых из этих изменений требуется выполнить обязательные действия, иначе LUNA PLATFORM может не запуститься или функционировать некорректно.

В таблице ниже перечислены не все изменения. См. подробную информацию о всех изменениях в примечаниях к выпуску LUNA PLATFORM.

Версия	Изменения	Обязательные действия
5.84.0	Из контейнера Remote SDK удалена 110ая модель нейронной сети для извлечения биометрических шаблонов тел. Теперь по умолчанию используется 116ая модель нейронной сети для извлечения биометрических шаблонов тел.	Ознакомиться с информацией, описанной в разделе «Подготовка к смене версии нейронной сети» , и выполнить определенные действия перед запуском Remote SDK, т.к. изменилась версия по умолчанию.
5.76.0	Группа параметров «INFLUX_MONITORING» переименована в «LUNA_MONITORING».	При обновлении окружения с помощью Storages с указанием разных сущностей (например, «luna_prepare configs», «luna_prepare database» и др.), необходимо выполнять обновление окружения для InfluxDB (аргументы «aggregated_influx_bucket» и «influx_bucket») строго после миграции настроек (аргумент «configs»). В противном случае подготовка окружения не будет выполнена корректно. При выполнении всего окружения с помощью аргумента «all_entities» никаких дополнительных действий делать не нужно.

Версия	Изменения	Обязательные действия
5.72.1	Из контейнера Remote SDK удалена 107ая модель нейронной сети для извлечения биометрических шаблонов тел.	Ознакомиться с информацией, описанной в разделе «Подготовка к смене версии нейронной сети» , и выполнить определенные действия перед запуском Remote SDK, если использовалась 107ая модель.
5.67.0	Обновлено значение порога из настройки «redetect_score_threshold» группы настроек «LUNA_REMOTE_SDK_FACE_DETECTOR_SETTINGS» с «0.3» до «0.5».	Обновить значение порога в соответствии с пользовательской логикой.
5.62.3	Теперь по умолчанию используется модель 62 нейронной сети для извлечения биометрических шаблонов.	Ознакомиться с информацией, описанной в разделе «Подготовка к смене версии нейронной сети» , и выполнить определенные действия перед запуском Remote SDK, т.к. изменилась версия по умолчанию.
5.53.0	Обновлен образ VisionLabs для PostgreSQL с 12 версии на 16 версию.	Если ранее использовался данный образ, то необходимо самостоятельно выполнить миграцию согласно официальной документации .

1.2 Создание резервных копий

Рекомендуется создать следующие резервные копии:

- резервные копии всех баз данных, используемых с LUNA PLATFORM;
- резервную копию бакетов Image Store;
- резервную копию настроек сервисов LUNA PLATFORM.

Создание резервных копий позволит восстановить в случае возникновения каких-либо проблем в процессе миграции.

1.2.1 Копия СУБД PostgreSQL

Резервная копия СУБД PostgreSQL выполняется с помощью утилит [pg_dumpall](#) или [pg_dump](#).

Воспользуйтесь официальной инструкцией для выполнения резервной копии.

1.2.2 Копия базы данных Influx

Резервная копия БД Influx выполняется с помощью команды [influxd backup](#).

Воспользуйтесь официальной инструкцией для выполнения резервной копии.

1.2.3 Копия бакетов Image Store

Создайте резервную копию бакетов с помощью следующей команды:

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

1.2.4 Дамп-файл с настройками сервисов

Пользовательские значения настроек сервисов LUNA PLATFORM (всех, кроме сервиса Configurator) автоматически мигрируются с помощью механизма миграции сервиса Configurator.

Если миграция сервиса по каким-либо причинам затерла пользовательскую настройку или пользователь просто хочет хранить старые настройки сервисов для различных версий LP, то можно создать дамп-файл.

Чтобы создать дамп-файл, используйте следующие команды (можно выполнить из любой директории на сервере):

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

или

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

Важно! Данный файл не будет использован в процессе нормальной установки LUNA PLATFORM. Чтобы применить сохраненные настройки, нужно использовать скрипт `db_create.py` с аргументом командной строки `--dump-file` (за которым следует имя созданного дампа-файла): `base_scripts/db_create.py --dump-file settings_dump.json`. Применить дамп-файл можно **только к пустой базе данных с созданными таблицами**. См. подробную информацию в разделе «Дамп-файл с настройками LP» руководства администратора.

1.3 Подготовка к смене версии нейронной сети

В некоторых сборках LUNA PLATFORM удаляются модели нейронных сетей для извлечения биометрических шаблонов лиц и тел, а также меняются настройки использования моделей по умолчанию. См. раздел «[Ключевые изменения предыдущих версий](#)» для подробной информации о данных изменениях.

Если выполняется обновление с версии, где были удалены нейронные сети, а в предыдущей сборке в настройках «DEFAULT_FACE_DESCRIPTOR_VERSION» или «DEFAULT_HUMAN_DESCRIPTOR_VERSION» была указана какая-то из удаленных моделей, то сервис Remote SDK **не запустится**.

В текущей сборке LUNA PLATFORM поддерживаются модели нейросетей для извлечения биометрических шаблонов:

Объект, из которого извлекается БШ	Модели нейронных сетей	Модель по умолчанию
Лицо	59, 60, 62	62
Тело	116	116

Необходимо выполнить одно из дополнительных действий в зависимости от следующих сценариев работы:

Продолжение использования отсутствующих нейронных сетей

Запросите у VisionLabs старую модель нейронной сети и подготовьте её к переносу в новый контейнер Remote SDK или Video Agent после его запуска (см. инструкцию по переносу в разделе «Использование модели нейросети не из поставки» руководства администратора).

Переход на новую версию нейронной сети с продолжением использования старых биометрических шаблонов

Перейти на новую версию с продолжением использования старых шаблонов можно выполнив задачу Additional extraction. Это позволит преобразовать старые шаблоны на новую версию нейронной сети.

В релизах LUNA PLATFORM сначала добавляется поддержка какой-либо модели, а только потом спустя несколько версий обновляется модель по умолчанию, что обеспечивает наличие новой модели в контейнере при обновлении с предыдущей версии. Однако если выполняется обновление со старой версии LUNA PLATFORM, то важно убедиться, что в старой версии контейнера Remote SDK имеется требуемая модель для выполнения задачи Additional extraction. Модель можно запросить у специалистов VisionLabs или перенести из более новой версии контейнера Remote SDK.

Важно! Исключением является процесс повторного извлечения биометрических шаблонов тел, позволяющий использовать задачу Additional extraction уже после запуска нового контейнера Remote SDK, который гарантированно имеет новую модель. В таком случае можно не думать о том есть ли модель нейронной сети в старом контейнере.

Шаги для выполнения перехода на новую версию нейронной сети с продолжением использования старых биометрических шаблонов:

- выполните предварительные действия если обновляетесь не с предыдущей версии LUNA PLATFORM:
 - **повторное извлечение биометрических шаблонов тел:** если предпочитаете выполнять задачу Additional extraction перед обновлением, то обязательно убедитесь, что в старом контейнере Remote SDK есть новая модель нейронной сети. Если предпочитаете выполнять задачу Additional extraction после обновления (рекомендуется), то никаких дополнительных действий не требуется.
 - **повторное извлечение биометрических шаблонов лиц:** обязательно убедитесь, что в контейнере Remote SDK есть новая модель нейронной сети.

Убедиться в наличии модели можно с помощью следующей команды:

```
docker exec -it luna-remote-sdk ls fsdk/data
```

В директории должны быть файлы вида `cnndescriptor_<nn_version>.conf`.

- выполните задачу Additional extraction:
 - **повторное извлечение биометрических шаблонов тел:** перед или после обновления
 - **повторное извлечение биометрических шаблонов лиц:** перед обновлением

См. раздел «Задача Additional extraction» в руководстве администратора.

- укажите новую версию нейронной сети в настройках «DEFAULT_FACE_DESCRIPTOR_VERSION» или «DEFAULT_HUMAN_DESCRIPTOR_VERSION» в соответствии с разделом [«Смена версии нейронной сети в Configurator»](#):

– **повторное извлечение биометрических шаблонов тел:**

- * после обновления если задача Additional extraction выполняется перед обновлением
- * перед обновлением если задача Additional extraction выполняется после обновления (иначе сервис Remote SDK может не запуститься если модель была удалена из контейнера)

– **повторное извлечение биометрических шаблонов лиц:** после обновления

Переход на новую версию нейронной сети с прекращением использования старых биометрических шаблонов

Укажите новую версию нейронной сети в настройках «DEFAULT_FACE_DESCRIPTOR_VERSION» или «DEFAULT_HUMAN_DESCRIPTOR_VERSION» перед запуском сервиса Remote SDK или Video Agent в соответствии с разделом [«Смена версии нейронной сети в Configurator»](#).

Если не требуется использовать извлечение биометрических шаблонов тел, то можно отключить использование нейронной сети с помощью команды `--env=EXTEND_CMD="--enable-body-descriptor-estimator=0"` при запуске контейнера Remote SDK (см. подробную информацию в разделе «Включение/отключение некоторых эстиматоров и детекторов» руководства администратора).

1.3.1 Смена версии нейронной сети в Configurator

Для смены версии нейронной сети необходимо выполнить следующие действия:

- откройте пользовательский интерфейс Configurator `http://<configurator_server_ip>:5070`;
- введите название настройки «DEFAULT_FACE_DESCRIPTOR_VERSION» или «DEFAULT_HUMAN_DESCRIPTOR_VERSION» в поле «Setting name» и нажмите «Apply Filters»;
- установите нужную модель нейронной сети в настройке «DEFAULT_FACE_DESCRIPTOR_VERSION» или «DEFAULT_HUMAN_DESCRIPTOR_VERSION»;
- сохраните изменения, нажав кнопку «Save».

Важно! Изменение настройки «DEFAULT_FACE_DESCRIPTOR_VERSION» распространяется на два контейнера - Remote SDK и Video Agent. Соответственно, модель нейронной сети для извлечения лиц должна присутствовать в обоих контейнерах. Если она присутствует только в одном контейнере, то второй не сможет запуститься.

1.4 Удаление символической ссылки

Удалите символическую ссылку в директорию предыдущей минорной версии с помощью следующей команды:

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

1.5 Распаковка дистрибутива

Дистрибутив представляет собой архив **luna_v.5.88.0**, где **v.5.88.0** это числовой идентификатор, обозначающий версию LUNA PLATFORM.

Архив включает в себя конфигурационные файлы, требуемые для установки и использования. Он не включает в себя Docker образы сервисов, их требуется скачать из Интернета отдельно.

Переместите дистрибутив в директорию на вашем сервере перед установкой. Например, переместите файлы в директорию `/root/`. В ней не должно быть никакого другого дистрибутива или файлов лицензии кроме целевых.

Переместите дистрибутив в директорию с LUNA PLATFORM.

```
mv /root/luna_v.5.88.0.zip /var/lib/luna
```

Установите приложение для распаковки архива при необходимости

```
yum install -y unzip
```

Откройте папку с дистрибутивом

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

Распакуйте файлы

```
unzip luna_v.5.88.0.zip
```

1.6 Создание символической ссылки

Создайте символическую ссылку. Она показывает, что актуальная версия файла дистрибутива используется для запуска LUNA PLATFORM.

```
ln -s luna_v.5.88.0 current
```


1.7 Изменение группы и владельца для директорий

Сервисы LP запускаются внутри контейнеров пользователем «luna». Таким образом, требуется установить разрешения для данного пользователя на работу с примонтированными директориями.

Откройте директорию LP «example-docker»:

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

Создайте директорию для хранения настроек:

```
mkdir luna_configurator/used_dumps
```

Установите для пользователя с UID 1001 и группой 0 разрешения на работу с примонтированными директориями.

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

1.8 Сохранение пользовательских настроек сервиса Configurator

Примечание. Пропустите данный шаг, если настройки Configurator не изменялись.

Настройки сервиса Configurator не мигрируются автоматически, в отличие от настроек всех остальных сервисов.

Если предыдущая версия LP использовалась с настройками сервиса Configurator, отличных от настроек по умолчанию, нужно создать резервную копию файла конфигурации «luna_configurator_postgres.conf» в отдельной директории на сервере.

```
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
```

Эта резервная копия должна быть примонтирована к запускаемому контейнеру сервиса Configurator.

Если вы не уверены, менялись ли настройки сервиса Configurator, то можете сравнить созданную резервную копию с настройками Configurator из текущей поставки с помощью следующей команды:

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

1.9 Создание директории логов для новых сервисов

Пропустите этот раздел в случае, если ранее логи не сохранялись на сервере.

В новой версии LUNA PLATFORM могли появиться новые сервисы, для которых нужно создать директории с логами. Это зависит от версии, с которой выполняется обновление. Например, в версии 5.30.0 появился сервис Accounts.

См. раздел «[Запись логов на сервер](#)» если вы ранее не использовали запись логов в файл, но хотите включить её.

Ниже приведены команды для создания директорий для всех существующих сервисов. Данные команды создадут и присвоят права только отсутствующим директориям.

```
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 /tmp/logs/luna-video-agent /tmp/logs/luna-  
video-manager
```

```
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 /tmp/logs/luna-video-agent /tmp/logs/  
luna-video-manager
```

Если необходимо использовать сервис Python Matcher Proxy, то нужно дополнительно создать директорию /tmp/logs/python-matcher-proxy и установить ей разрешения.

1.10 Обновление лицензии

Для обновления лицензии необходимо выполнить следующие действия:

- выполнить действия из [руководства по активации лицензии](#)
- задать настройки лицензирования [HASP](#) или [Guardant](#) перед запуском контейнера Licenses

1.10.1 Действия из руководства по активации лицензии

Откройте руководство по активации лицензии и выполните необходимые шаги.

Примечание. Это действие является обязательным. Лицензия не будет работать без выполнения шагов по активации лицензии из соответствующего руководства.

Примечание. При обновлении Guardant Control Center необходимо заново выписать лицензионный ключ.

1.11 Вычисления с помощью GPU

Для основных вычислений, выполняемых сервисом Remote SDK, можно использовать GPU.

Пропустите данный раздел, если не собираетесь использовать GPU для вычислений.

Для использования GPU с Docker-контейнерами необходимо установить NVIDIA Container Toolkit. Пример установки приведен ниже.

```
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
```

Проверьте работу NVIDIA Container toolkit, запустив базовый контейнер CUDA (он не входит в дистрибутив LP, его необходимо загрузить из Интернета):

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

См. [документацию NVIDIA](#) для дополнительной информации.

Извлечение атрибутов на GPU разработано для максимальной пропускной способности. Выполняется пакетная обработка входящих изображений. Это снижает затраты на вычисления для изображения, но не обеспечивает минимальную задержку для каждого изображения.

GPU-ускорение разработано для приложений с высокой нагрузкой, где количество запросов в секунду достигает тысяч. Нецелесообразно использовать ускорение GPU в сценариях

с небольшой нагрузкой, когда задержка начала обработки имеет значение.

1.12 Удаление старых контейнеров (опционально)

Примечание. Удаление старых контейнеров не является обязательным при подготовке окружения с помощью утилиты Storages. Достаточно просто ограничить количество запросов к БД, однако в таком случае необходимо понимать определенные последствия. См. раздел «Рекомендации по поведению сервисов во время подготовки окружения» в руководстве по утилите Storages.

Важно! Исключением является сервис Image Store. Во время подготовки окружения бакетов, Storages выполняет запрос к сервису Image Store для получения информации о бакетах. Если контейнер Image Store удален, то запрос будет выполнен неудачно и возникнет ошибка миграции. Чтобы выполнить миграцию бакетов при удаленном контейнере Image Store, необходимо явно указать утилите Storages расположение бакетов с помощью аргументов `--local-buckets` или `--s3-buckets`. При использовании локальных бакетов необходимо также примонтировать к контейнеру Storages директорию с бакетами.

Остановите все контейнеры, относящиеся к предыдущей версии LUNA PLATFORM. Контейнеры сторонних приложений удалять необязательно.

Например, можно использовать следующую команду:

```
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
```

Чтобы посмотреть имена запущенных контейнеров или их ID, используйте следующую команду:

```
docker ps -a
```

Также рекомендуется удалить старые образы контейнеров для освобождения места. Можно использовать следующую команду для удаления всех неиспользуемых образов.

Если на сервере достаточно места, рекомендуется выполнить это действие только после успешного запуска новой версии LP.

Данная команда удаляет все неиспользуемые образы, а не только образы, относящиеся к LP.

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

2 Запуск сторонних сервисов

В данном разделе описывается запуск баз данных в Docker-контейнерах. Они должны быть запущены перед подготовкой окружения и сервисами LP.

2.1 InfluxDB

Примечание. Если вы не удаляли старый контейнер, пропустите данный шаг.

Для мониторинга сервисов LUNA PLATFORM требуется наличие запущенной базы данных Influx 2.0.8-alpine. Ниже приведены команды по запуску контейнера InfluxDB.

Дополнительную информацию см. в разделе «Мониторинг» в руководстве администратора.

При необходимости можно настроить визуализацию данных мониторинга с помощью сервиса LUNA Dashboards, включающего в себя настроенную систему визуализации данных Grafana. Кроме того, можно запустить инструмент для расширенной работы с логами Grafana Loki. См. инструкцию по запуску LUNA Dashboards и Grafana Loki в разделе «Визуализация мониторинга и логов с помощью Grafana».

Примечание. При необходимости можно использовать внешнюю БД InfluxDB 2.0.8-alpine. В таком случае можно пропустить команду ниже, однако вам придется задать [пользовательские настройки](#) для каждого сервиса LUNA PLATFORM.

Используйте команду `docker run` со следующими параметрами:

```
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 Миграция с PostgreSQL 12 на PostgreSQL 16

В LUNA PLATFORM v.5.53.0 обновился образ VisionLabs для PostgreSQL с 12 версии на 16 версию.

Если ранее использовался данный образ, то необходимо самостоятельно выполнить миграцию согласно [официальной документации](#). При необходимости можно продолжить использовать PostgreSQL 12, указав образ «postgis-vmatch:12» в команде запуска контейнера.

Монтирование данных PostgreSQL 12 из директории «/var/lib/luna/postgres» в контейнер для PostgreSQL 16 приведет к ошибке.

2.2.2 Запуск контейнера PostgreSQL

Примечание. Если вы не удаляли старый контейнер, пропустите данный шаг.

Используйте следующую команду для запуска 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-vmatch:16
```

—v /var/lib/luna/current/example-docker/postgresql/data:/var/lib/postgresql/data/ — эта команда позволяет монтировать директорию «data» в контейнер PostgreSQL. Директория на сервере и директория в контейнере будут синхронизированы. Данные PostgreSQL из контейнера будут сохраняться в эту директорию.

--network=host — при необходимости изменить порт для PostgreSQL, следует изменить эту строку на -p 5440:5432. Здесь первый порт 5440 — локальный, а 5432 — порт в контейнере.

Все базы данных для сервисов LP следует создавать вручную, если используется уже установленный PostgreSQL.

2.3 Redis

Примечание. Если вы не удаляли старый контейнер, пропустите данный шаг.

Примечание. При запуске Redis по умолчанию задан пароль `lunapass`. Крайне рекомендуется сменить пароль для повышения безопасности. Обратите внимание, что пароль должен быть указан в соответствующей настройке сервиса, например, в параметре «password» группы «LUNA_ATTRIBUTES_DB» сервиса Events. По умолчанию в дамп-файле уже прописан пароль `lunapass`.

Используйте следующую команду для запуска Redis.

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

3 Обновление окружения

Окружение обновляется с помощью сервиса Storages. С помощью команды ниже будет выполнено:

- Создание бакетов в InfluxDB для работы мониторинга (если не было выполнено ранее)
- Создание бакетов для сервиса Image Store (если не было выполнено ранее)
- Подготовка БД Influx для сбора агрегированной статистики сервисом Admin (если не было выполнено ранее)
- Миграция баз данных
- Миграция настроек в БД Configurator

При обновлении окружения будет использован дефолтный конфигурационный файл сервиса Storages, содержащий все стандартные настройки подключения к базам данных, бакетов и пр. Если необходимо использовать нестандартные настройки или обновлять LUNA PLATFORM, развернутую на разных серверах, то нужно отредактировать конфигурационный файл перед запуском команды обновления окружения:

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

При необходимости можно сначала обновить окружение для сервиса Configurator, запустить его, а затем обновить окружение для всех остальных сервисов, используя настройки из запущенного сервиса Configurator. См. примеры и подробную информацию о Storages в руководстве по утилите Storages.

3.1 Команда обновления окружения

Примечание. Если старый контейнер Image Store не удален, то можно не указывать расположение бакетов с помощью аргументов `--local-buckets` или `--s3-buckets` и не монтировать к контейнеру Storages директорию с локальными бакетами. См. раздел «Удаление старых контейнеров (опционально)».

Подготовьте окружение с помощью следующей команды:

```
docker run \
--rm \
--network=host \
-v /var/lib/luna/current/extras/conf/storages_config.conf:/srv/
storages_config.conf \
-v /var/lib/luna/image_store:/srv/local_storage/ \
dockerhub.visionlabs.ru/luna/storages:v.0.4.37 \
bash -c "luna_prepare prepare all_entities \
--platform_version=v.5.88.0 \
```



```
--profile=common \  
--config=/srv/storages_config.conf \  
--local-buckets=/srv/local_storage"
```

Здесь:

- `luna_prepare prepare all_entities` — команда «prepare» для подготовки всех сущностей
- `--platform_version` — именованный аргумент, содержащий версию LUNA PLATFORM
- `--profile` — именованный аргумент, содержащий профиль (ссылка на список сервисов) backports, означающий, что будет подготовлено окружение для всех сервисов, включая сервисы Backport 3 и Backport 4
- `-v /var/lib/luna/image_store:/srv/local_storage/` — команда монтирования директории с бакетами Image Store
- `-v /var/lib/luna/current/extras/conf/storages_config.conf:/srv/storages_config.conf` — команда монтирования конфигурационного файла Storages
- `-local-buckets=/srv/local_storage` — именованный аргумент, содержащий директорию, в которой должны быть созданы бакеты Image Store
- `--config=/srv/storages_config.conf` — именованный аргумент, содержащий адрес конфигурационного файла для использования сервисом Storages

Примечание. На данном этапе можно выполнить определенные действия по шифрованию/дешифрованию/обновлению шифрования существующих биометрических шаблонов. См. раздел [«Управление шифрованием биометрических шаблонов»](#) для более подробной информации.

4 Запуск сервисов

В данном разделе приведены примеры команд запуска сервисов LUNA PLATFORM.

Сервисы LUNA PLATFORM должны запускаться в следующем порядке:

- Базы данных, балансировщики, HASP сервис и прочие сторонние сервисы
- [Configurator](#)
- [Image Store](#)
- [Accounts](#)
- [Licenses](#)
- [Faces](#)
- [Events](#)
- [Python Matcher](#)
- [Python Matcher Proxy](#). Сервис отключен по умолчанию.
- [Remote SDK](#)
- [Handlers](#)
- [Tasks](#)
- [Sender](#)
- [Сервисы видеоаналитики](#)
- [API](#)
- [Admin](#)

Следующие сервисы используются, когда требуется обеспечить совместимость с запросами формата LUNA PLATFORM 3:

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

Следующие сервисы используются, когда требуется обеспечить совместимость с запросами формата LUNA PLATFORM 4:

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

Рекомендуется запускать контейнеры один за другим и ожидать отображения статуса контейнера «up» (команда `docker ps`).

Некоторые из этих сервисов не являются обязательными к запуску и можно отключить их использование. Рекомендуется использовать сервисы Events, Tasks, Sender и Admin по умолчанию. См. раздел [«Использование необязательных сервисов»](#) для более подробной информации.

При запуске каждого сервиса используются определенные параметры, например, `--detach`, `--network` и др. См. раздел [«Описание параметров запуска»](#) для получения более подробной информации о всех параметрах запуска сервисов LUNA PLATFORM и баз данных.

См. раздел [«Команды Docker»](#) для получения более подробной информации о работе с контейнерами.

4.1 Configurator

4.1.1 Запуск контейнера Configurator

Примечание. Настройки сервиса Configurator не мигрируются автоматически, в отличие от настроек всех остальных сервисов. Если в предыдущей версии LP изменялись настройки сервиса Configurator и необходимо сохранить пользовательские значения, то в команде ниже необходимо заменить путь `/var/lib/luna/current/example-docker/luna_configurator/configs/luna_configurator_postgres.conf` на путь с резервной копией настроек сервиса Configurator `/var/lib/luna/BACKUP_luna_configurator_postgres.conf` (см. раздел [«Сохранение пользовательских настроек сервиса Configurator»](#)).

Используйте команду `docker run` со следующими параметрами для запуска 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.2.2.74
```

На данном этапе можно активировать запись логов в файл, если необходимо сохранять их на сервере (см. раздел [«Запись логов на сервер»](#)).

4.2 Image Store

4.2.1 Запуск контейнера Image Store

Примечание. Если вы не собираетесь использовать сервис Image Store, не запускайте этот контейнер и отключите использование сервиса в сервисе Configurator. См. раздел [«Использование необязательных сервисов»](#).

Используйте следующую команду для запуска сервиса Image Store:

```
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.3.14.35
```

-v /var/lib/luna/image_store:/srv/local_storage/ — данные из указанной директории добавляются в Docker-контейнер, когда он запущен. Все данные из указанной директории Docker-контейнера сохраняются в данную директорию.

Если директория с бакетами LP уже создана, укажите ее вместо /var/lib/luna/image_store/.

4.3 Accounts

4.3.1 Запуск контейнера Accounts

Используйте следующую команду для запуска сервиса Accounts:

```
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.0.3.42
```

4.4 Licenses

Примечание. Для использования триальной лицензии необходимо запускать сервис Licenses на том же сервере, на котором она используется.

4.4.1 Задание настроек лицензии с помощью Configurator

Выполните действия по заданию настроек для [HASP-ключа](#) или [Guardant-ключа](#).

4.4.1.1 Задание настроек лицензии HASP

Примечание. Выполняйте данные действия только если используется лицензия HASP. См. раздел «[Задание настроек лицензии Guardant](#)», если используется ключ Guardant.

Для задания адреса сервера лицензирования нужно выполнить следующие действия:

- перейдите в интерфейс сервиса Configurator `http://<configurator_server_ip>:5070/`
- введите в поле «Setting name» значение «LICENSE_VENDOR» и нажмите «Apply Filters»
- задайте IP-адрес сервера с вашим ключом HASP в поле «server_address» в формате «127.0.0.1».
- нажмите «Save»

Обратите внимание, что если лицензия активируется с помощью ключа HASP, то должно быть указано два параметра «vendor» и «server_address». Если вы хотите изменить защиту HASP на Guardant, то необходимо добавить поле «license_id».

4.4.1.2 Задание настроек лицензии Guardant

Примечание. Выполняйте данные действия только если используется ключ Guardant. См. раздел «[Задание настроек лицензии HASP](#)», если используется ключ HASP.

Для задания адреса сервера лицензирования нужно выполнить следующие действия:

- перейдите в интерфейс сервиса Configurator `http://<configurator_server_ip>:5070/`
- введите в поле «Setting name» значение «LICENSE_VENDOR» и нажмите «Apply Filters»
- задайте IP-адрес сервера с вашим ключом Guardant в поле «server_address»
- задайте идентификатор лицензии в формате `0x<your_license_id>`, полученный в разделе «Сохранение идентификатора лицензии» руководства по активации лицензии, в поле «license_id»
- нажмите «Save»

Обратите внимание, что если лицензия активируется с помощью ключа Guardant, то должно

быть указано три параметра «vendor», «server_address» и «license_id». Если вы хотите изменить защиту Guardant на HASP, то необходимо удалить поле «license_id».

4.4.2 Запуск контейнера Licenses

Используйте следующую команду для запуска сервиса:

```
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.0.11.6
```


4.5 Faces

4.5.1 Запуск контейнера Faces

Используйте следующую команду для запуска сервиса:

```
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.4.13.0
```

4.6 Events

4.6.1 Запуск контейнера Events

Примечание. Если вы не собираетесь использовать сервис Events, не запускайте этот контейнер и отключите использование сервиса в сервисе Configurator. См. раздел [«Использование необязательных сервисов»](#).

Используйте следующую команду для запуска сервиса:

```
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.4.17.0
```

4.7 Сервисы Python Matcher

Для задач сравнения можно использовать либо только сервис Python Matcher, либо дополнительно использовать сервис Python Matcher Proxy, который перенаправляет запросы сравнения либо сервису Python Matcher либо плагинам сравнения. В данном разделе описывается использование Python Matcher без Python Matcher Proxy.

Необходимо использовать сервис Python Matcher Proxy только если собираетесь использовать плагины сравнения. Использование Python Matcher Proxy и запуск соответствующего docker-контейнера описаны в разделе [«Использование Python Matcher с Python Matcher Proxy»](#).

См. описание и использование плагинов сравнения в руководстве администратора.

4.7.1 Использование Python Matcher без Python Matcher Proxy

Сервис Python Matcher со сравнением посредством базы данных Faces включен по умолчанию при запуске.

Сервис Python Matcher со сравнением посредством Events также включен по умолчанию. Его можно отключить, указав «USE_LUNA_EVENTS = 0» в разделе «ADDITIONAL_SERVICES_USAGE» настроек Configurator (см. раздел [«Использование необязательных сервисов»](#)). Таким образом, сервис Events не будет использоваться для LUNA PLATFORM.

Python Matcher, который производит сравнение с помощью библиотеки сравнений, включается когда «CACHE_ENABLED» установлен как «true» в настройке «DESCRIPTORS_CACHE».

Для сервисов Python Matcher и Python Matcher Proxy загружается одно изображение.

4.7.2 Запуск контейнера Python Matcher

Используйте следующую команду для запуска сервиса:

```
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.1.12.2
```

4.8 Remote SDK

4.8.1 Смена модели нейронной сети для извлечения биометрических шаблонов

В некоторых сборках LUNA PLATFORM удаляются модели нейронных сетей для извлечения биометрических шаблонов лиц и тел, а также меняются настройки использования моделей по умолчанию.

Если выполняется обновление с версии, где были удалены нейронные сети, а в предыдущей сборке в настройках «DEFAULT_FACE_DESCRIPTOR_VERSION» или «DEFAULT_HUMAN_DESCRIPTOR_VERSION» была указана какая-то из удаленных моделей, то сервис Remote SDK **не запустится**, если не выполнить одно из нижеперечисленных действий:

- запросить у специалистов VisionLabs старую модель нейронной сети и перенести её в новый контейнер
- перейти на новую версию нейронной сети, предварительно выполнив задачу Additional Extraction для повторного извлечения существующих БШ с использованием новой версии нейронной сети для их дальнейшего использования
- перейти на новую версию нейронной сети, прекратив использование старых биометрических шаблонов

См. подробную информацию в разделе [«Подготовка к смене версии нейронной сети»](#).

4.8.2 Запуск контейнера Remote SDK

Вы можете запустить сервис Remote SDK, используя CPU (задано по умолчанию) или GPU.

По умолчанию сервис Remote SDK запускается со всеми включенными эстиматорами и детекторами. При необходимости можно отключить использование некоторых эстиматоров или детекторов при запуске контейнера Remote SDK. Отключение ненужных эстиматоров позволяет экономить оперативную память или память GPU, поскольку при старте сервиса Remote SDK выполняется проверка возможности выполнения указанных оценок и загрузка нейронных сетей в память. При отключении эстиматора или детектора можно также удалить его нейронную сеть из контейнера Remote SDK. См. подробную информацию в разделе «Включение/отключение некоторых эстиматоров и детекторов» руководства администратора.

Запустите сервис Remote SDK, используя одну из следующих команд в соответствии с используемым процессором.

4.8.2.1 Запуск Remote SDK с использованием CPU

Используйте следующую команду для запуска сервиса:

```

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.0.15.0

```

4.8.2.2 Запуск Remote SDK с использованием GPU

Сервис Remote SDK не использует GPU по умолчанию. Если вы собираетесь использовать GPU, то следует включить его использование для сервиса Remote SDK в сервисе Configurator.

Если необходимо использовать GPU сразу для всех эстиматоров и детекторов, то необходимо использовать параметр «global_device_class» в секции «LUNA_REMOTE_SDK_RUNTIME_SETTINGS». Все эстиматоры и детекторы будут использовать значение данного параметра, если в параметре «device_class» их собственных настроек выставлено значение «global» (по умолчанию).

Если необходимо использовать GPU для определенного эстиматора или детектора, то необходимо использовать параметр «device_class» в секциях вида "LUNA_REMOTE_SDK_<estimator-or-detector-name>_SETTINGS.runtime_settings".

См. раздел «[Вычисления с помощью GPU](#)» для получения дополнительных требований к использованию GPU.

Используйте следующую команду для запуска сервиса Remote SDK с помощью 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.0.15.0
```

- `--gpus device=0` — параметр указывает используемое устройство GPU и позволяет использовать GPU. Один GPU используется для одного экземпляра Remote SDK. Использование множества GPU для одного экземпляра невозможно.

4.8.2.3 Запуск облегченной версии Remote SDK

Можно запустить облегченную версию сервиса Remote SDK, содержащую только конфигурационные файлы без нейронных сетей. Предполагается, что пользователь сам добавит в контейнер необходимые ему нейронные сети.

Запуск облегченной версии сервиса Remote SDK предназначен для продвинутых пользователей.

Для успешного запуска контейнера Remote SDK с пользовательским набором нейронных сетей нужно выполнить следующие действия:

- запросить у VisionLabs требуемые нейронные сети
- поместить нейронные сети в папку с установленной LUNA PLATFORM
- присвоить соответствующие права для файлов нейронных сетей
- смонтировать файлы нейронных сетей в папку `/srv/fsdk/data` контейнера Remote SDK
- с помощью аргументов переменной «`EXTEND_CMD`» явно указать какие из нейронных сетей должны использоваться

Обратите внимание, что с помощью флага «`enable-all-estimators-by-default`» для переменной «`EXTEND_CMD`» можно выключить по умолчанию использование всех нейронных сетей (эстиматоров), а затем с помощью специальных флагов явно указывать какие нейронные сети должны быть использованы. Если не указывать данный флаг или выставить значение «`-enable-all-estimators-by-default=1`», то сервис Remote SDK будет пытаться найти в контейнере все нейронные сети. Если какая-то из нейронных сетей не будет найдена, то сервис Remote SDK не запустится.

Список доступных аргументов для запуска:

Аргумент	Описание
<code>--enable-all-estimators-by-default</code>	включить все эстиматоры по умолчанию
<code>--enable-human-detector</code>	одновременный детектор
<code>--enable-face-detector</code>	детектор лиц

Аргумент	Описание
--enable-body-detector	детектор тел
--enable-face-landmarks5-estimator	эстиматор 5 контрольных точек лица
--enable-face-landmarks68-estimator	эстиматор 68 контрольных точек лица
--enable-head-pose-estimator	эстиматор положения головы
--enable-liveness-estimator	эстиматор OneShotLiveness
--enable-fisheye-estimator	эстиматор бочообразной дисторсии (эффекта FishEye)
--enable-face-detection-background-estimator	эстиматор фона изображения
--enable-face-warp-estimator	эстиматор биометрического образца лица
--enable-body-warp-estimator	эстиматор биометрического образца тела
--enable-quality-estimator	эстиматор качества изображения
--enable-image-color-type-estimator	эстиматор типа цвета по лицу
--enable-face-natural-light-estimator	эстиматор естественности освещения
--enable-eyes-estimator	эстиматор глаз
--enable-gaze-estimator	эстиматор направления взгляда
--enable-mouth-attributes-estimator	эстиматор атрибутов рта
--enable-emotions-estimator	эстиматор эмоций
--enable-mask-estimator	эстиматор маски
--enable-glasses-estimator	эстиматор очков
--enable-eyebrow-expression-estimator	эстиматор бровей
--enable-red-eyes-estimator	эстиматор красных глаз
--enable-headwear-estimator	эстиматор головного убора
--enable-basic-attributes-estimator	эстиматор базовых атрибутов
--enable-face-descriptor-estimator	эстиматор извлечения биометрического шаблона лица
--enable-body-descriptor-estimator	эстиматор извлечения биометрического шаблона тела
--enable-body-attributes-estimator	эстиматор атрибутов тел
--enable-people-count-estimator	эстиматор количества людей

Аргумент	Описание
<code>--enable-deepfake-estimator</code>	эстиматор Deepfake

См. подробную информацию включения и выключения определенных эстиматоров в разделе «Включение/отключение некоторых эстиматоров и детекторов» руководства администратора.

Ниже приведен пример команды для присвоения прав файлу нейронной сети:

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

Пример команды запуска контейнера Remote SDK с монтированием нейронных сетей для детекции лиц и извлечения биометрических шаблонов лиц:

```
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.0.15.0
```

4.9 Handlers

Примечание. Если вы не собираетесь использовать сервис Handlers, не запускайте этот контейнер и отключите использование сервиса в сервисе Configurator. См. раздел [«Использование необязательных сервисов»](#).

4.9.1 Запуск контейнера Handlers

Используйте следующую команду для запуска сервиса:

```
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.3.14.0
```

4.10 Tasks

Примечание. Если вы не собираетесь использовать сервис Tasks, не запускайте контейнер Tasks и контейнер Tasks Worker. Отключите сервис Tasks в сервисе Configurator. См. раздел «[Использование необязательных сервисов](#)».

4.10.1 Запуск контейнеров Tasks и Tasks Worker

Образ сервиса Tasks включает в себя сервисы Tasks и Tasks Worker («рабочие процессы сервиса Tasks»). Они оба должны быть запущены.

Если необходимо использовать задачу Estimator с использованием сетевого диска, то необходимо предварительно смонтировать директорию с изображениями с сетевого диска в специальные директории контейнеров Tasks и Tasks Worker. См. подробную информацию в разделе «Задача Estimator» в руководстве администратора.

4.10.1.1 Запуск контейнера Tasks worker

Используйте следующую команду для запуска сервиса:

```
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.3.24.2
```

4.10.1.2 Запуск контейнера Tasks

Используйте следующую команду для запуска сервиса:

```
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.3.24.2
```

4.11 Sender

4.11.1 Запуск контейнера Sender

Примечание. Если вы не собираетесь использовать сервис Sender, не запускайте этот контейнер и отключите этот сервис в Configurator. См. раздел «[Использование необязательных сервисов](#)».

Используйте следующую команду для запуска сервиса:

```
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.2.13.2
```

4.12 Сервисы видеоаналитики

Примечание. Если вы не собираетесь использовать сервисы видеоаналитики, не запускайте контейнер Video Manager и контейнер Video Agent. См. раздел «[Использование необязательных сервисов](#)».

4.12.1 Запуск контейнера Video Manager

Используйте следующую команду для запуска сервиса:

```
docker run \
--env=CONFIGURATOR_HOST=127.0.0.1 \
--env=CONFIGURATOR_PORT=5070 \
--env=PORT=5230 \
--env=WORKER_COUNT=1 \
--env=RELOAD_CONFIG=1 \
--env=RELOAD_CONFIG_INTERVAL=10 \
-v /etc/localtime:/etc/localtime:ro \
-v /tmp/logs/video-manager:/srv/logs \
--name=luna-video-manager \
--restart=always \
--detach=true \
--network=host \
dockerhub.visionlabs.ru/luna/luna-video-manager:v.0.2.10
```

4.12.2 Запуск контейнера Video Agent

Используйте следующую команду для запуска сервиса:

```
docker run \
--env=CONFIGURATOR_HOST=127.0.0.1 \
--env=CONFIGURATOR_PORT=5070 \
--env=PORT=5240 \
--env=WORKER_COUNT=1 \
--env=RELOAD_CONFIG=1 \
--env=RELOAD_CONFIG_INTERVAL=10 \
-v /etc/localtime:/etc/localtime:ro \
-v /tmp/logs/video-agent:/srv/logs \
--name=luna-video-agent \
--restart=always \
--detach=true \
--network=host \
```

`dockerhub.visionlabs.ru/luna/luna-video-agent:v.0.5.2`

4.13 API

4.13.1 Запуск контейнера API

Используйте следующую команду для запуска сервиса:

```
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.6.41.0
```

См. пример команды для создания нового аккаунта в разделе [«Создание аккаунта»](#).

Рекомендуется создать расписание для задачи очистки мусора, если оно не было ранее создано. См. пример команды в разделе [«Создание расписания задачи GC»](#).

4.14 Admin

4.14.1 Запуск контейнера Admin

Примечание. Если вы не собираетесь использовать сервис Admin, не запускайте этот контейнер.

Используйте следующую команду для запуска сервиса:

```
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.5.10.0
```

4.15 Backport 3

В данном разделе описывается запуск сервиса Backport 3.

Сервис не обязателен для использования LP5 и требуется только для эмуляции LP 3 API.

4.15.1 Запуск контейнера Backport 3

Используйте следующую команду для запуска сервиса:

```
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.0.11.42
```

4.15.2 User Interface 3

User Interface 3 используется только с сервисом Backport 3.

4.15.2.1 Запуск контейнера User Interface 3

Используйте следующую команду для запуска сервиса:

```
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.0.5.14
```

--env=LUNA_API_URL — URL сервиса Backport 3.

--env=PORT — порт сервиса User Interface 3.

4.16 Backport 4

В данном разделе описывается запуск сервиса Backport 4.

Этот сервис необязателен для использования LP5 и требуется только для эмуляции LP 4 API.

4.16.1 Запуск контейнера Backport 4

Используйте следующую команду для запуска сервиса:

```
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.1.6.4
```

4.16.2 User Interface 4

User Interface 4 используется только с сервисом Backport 4.

4.16.2.1 Запуск контейнера User Interface 4

Примечание. Перед запуском контейнера User Interface 4 необходимо наличие аккаунта типа **user**. Его логин и пароль в формате Base64 будут использованы для работы с пользовательским интерфейсом.

Используйте следующую команду для запуска сервиса:

```
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.0.1.9
```

--env=PORT — задает порт для запуска User Interface 4.

--env=BASIC_AUTH — задает авторизацию типа Basic для аккаунта, данные которого отображаются в пользовательском интерфейсе.

--env=LUNA_API_URL — задает URL сервиса Backport 4.

- Необходимо использовать внешний IP сервиса, а не локальный хост.
- Необходимо указать порт сервиса Backport 4 (5130 задан по умолчанию).

5 Дополнительная информация

В данном разделе приводится следующая дополнительная информация:

- [Создание аккаунта](#)
- [Создание расписания для задачи по очистке мусора](#)
- [Управление шифрованием биометрических шаблонов](#)
- [Визуализация мониторинга и логов с помощью Grafana](#)
- [Полезные команды для работы с Docker](#)
- [Описание параметров запуска сервисов LUNA PLATFORM и создания баз данных](#)
- [Действия по включению сохранения логов сервисов LP в файлы](#)
- [Настройка ротации логов Docker](#)
- [Задание пользовательских настроек InfluxDB](#)
- [Использование сервиса Python Matcher с сервисом Python Matcher Proxy](#)

5.1 Создание аккаунта

Аккаунт создается с помощью HTTP-запроса к ресурсу «create account».

Аккаунт также можно создать с помощью сервиса Admin. Данный способ требует наличия существующих логина и пароль (или логина и пароля по умолчанию) и позволяет создать аккаунты типа «admin». См. подробную информацию в разделе «Сервис Admin» руководства администратора.

Для создания аккаунта с помощью запроса к сервису API необходимо указать следующие обязательные данные:

- login — электронный адрес
- password — пароль
- account_type — тип аккаунта («user» или «advanced_user»)

Создайте аккаунт, используя свои аутентификационные данные.

Пример CURL-запроса к ресурсу «create account»:

```
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"  
}'
```

Необходимо заменить аутентификационные данные из примера на свои.

См. подробную информацию об аккаунтах в разделе «Аккаунты и типы авторизации» руководства администратора.

Для работы с токенами необходимо наличие аккаунта.

5.2 Создание расписания задачи GC

Перед началом работы с LUNA PLATFORM можно создать расписание для задачи Garbage collection.

Для этого следует выполнить запрос «create tasks schedule» к сервису API, указав необходимые правила для расписания.

Пример команды создания расписания для аккаунта из раздела «Создание аккаунта», приведен ниже.

В примере задается расписание для задачи Garbage collection для событий старше 30 дней с удалением БО и исходных изображений. Задача будет повторяться **один раз в сутки в 05:30 утра**.

```
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}
}'
```

При необходимости можно создать расписание без его автоматической активации. Для этого нужно указать параметр «create_stopped»: «true». В таком случае после создания расписания его необходимо активировать вручную с помощью параметра «action» = «start» запроса «patch tasks schedule».

См. подробную информацию в разделе «Запуск задач по расписанию» руководства администратора.

5.3 Управление шифрованием биометрических шаблонов

Примечание. См. подробную информацию в разделе «Шифрование биометрических шаблонов» в руководстве администратора.

Для обновления существующих биометрических шаблонов в базе данных Faces/Attributes/Events необходимо запустить скрипт `descriptors_encryption.py`. Этот скрипт предоставляет три варианта работы:

- Шифрование исходных биометрических шаблонов.
- Замена текущего ключа шифрования на новый.
- Дешифрование зашифрованных биометрических шаблонов.

Скрипт требует установки трех переменных окружения вне зависимости от выбранного варианта:

- `OLD_ENCRYPTION_KEY` (может быть пустым)
- `NEW_ENCRYPTION_KEY` (может быть пустым)
- `ENCRYPTION_ALGORITHM` (всегда обязателен и не может быть пустым)

Шифрование исходных биометрических шаблонов

Этот вариант позволяет зашифровать все биометрические шаблоны в базе данных, если они ранее не были зашифрованы. Уже зашифрованные биометрические шаблоны будут проигнорированы, поэтому скрипт можно запускать несколько раз. Для этого варианта необходимо заполнить переменную `NEW_ENCRYPTION_KEY` и оставить переменную `OLD_ENCRYPTION_KEY` пустой.

Замена текущего ключа шифрования на новый

Этот вариант предполагает, что шифрование уже было выполнено, и у вас есть текущий ключ шифрования, который вы хотите обновить. Уже зашифрованные биометрические шаблоны будут проигнорированы, поэтому скрипт можно запускать несколько раз. Необходимо указать обе переменные: `OLD_ENCRYPTION_KEY` и `NEW_ENCRYPTION_KEY`.

Дешифрование зашифрованных биометрических шаблонов

Этот вариант вернет биометрические шаблоны в их исходное состояние. Скрипт обработает только зашифрованные биометрические шаблоны и оставит неизменными оригинальные. Для выполнения этого варианта скрипта необходимо указать переменную `OLD_ENCRYPTION_KEY` и оставить переменную `NEW_ENCRYPTION_KEY` пустой.

5.3.0.1 Запуск скрипта миграции биометрических шаблонов

Важно! Убедитесь, что сделали резервную копию базы данных перед запуском скрипта. Также учтите, что все три варианта выше очистят временные атрибуты.

Для выполнения миграции необходимо выполнить скрипт `descriptors_encryption.py`, передав ему соответствующие аргументы. Список доступных аргументов приведен ниже:

- `--config` — Путь к файлу конфигурации.

- `--luna-config` — Адрес и версия API сервиса Configurator для получения настроек (например, «`http://127.0.0.1:5070/1`»).
- `--chunk-size` — Размер пакета для обновления.
- `--LUNA_FACES_DB` — Тег для установки LUNA_FACES_DB для получения настроек из Configurator (только при миграции БД Faces).
- `--DATABASE_NUMBER` — Тег для установки DATABASE_NUMBER для получения настроек из Configurator.
- `--LUNA_ATTRIBUTES_DB` — Тег для установки LUNA_ATTRIBUTES_DB для получения настроек из Configurator (только при миграции БД Faces).
- `--LUNA_EVENTS_DB` — Тег для установки LUNA_EVENTS_DB для получения настроек из Configurator (только при миграции БД Events).
- `-v`, `--verbose` — Включение режима отладки.

Также необходимо передать соответствующие переменные окружения в команду запуска контейнера (см. выше).

Важно! Скрипт необходимо выполнять когда сервис остановлен, после миграции БД и перед запуском повторным запуском сервиса.

Пример команды выполнения миграции биометрических шаблонов в БД сервиса Faces:

```
docker run \
--env=OLD_ENCRYPTION_KEY=<your_old_encryption_key> \
--env=NEW_ENCRYPTION_KEY=<your_new_encryption_key> \
--env=ENCRYPTION_ALGORITHM=aes256-gcm \
-v /etc/localtime:/etc/localtime:ro \
-v /tmp/logs/faces:/srv/logs \
--rm \
--network=host \
dockerhub.visionlabs.ru/luna/luna-faces:v.4.13.0 \
python3 ./base_scripts/descriptors_encryption.py --luna-config=http
://127.0.0.1:5070/1
```

5.4 Визуализация мониторинга и логов с помощью Grafana

Визуализация мониторинга выполняется за счет сервиса LUNA Dashboards, который содержит в себе платформу для визуализации данных мониторинга Grafana с настроенными дашбордами LUNA PLATFORM.

При необходимости можно отдельно установить настроенные дашборды для Grafana. См. дополнительную информацию в разделе «LUNA Dashboards» в руководстве администратора.

Вместе с Grafana можно использовать систему агрегации логов Grafana Loki, позволяющую гибко работать с логами LUNA PLATFORM. Для доставки логов LUNA PLATFORM в Grafana Loki используется агент Promtail (дополнительную информацию см. в разделе «Grafana Loki» в руководстве администратора).

5.4.1 LUNA Dashboards

Примечание. Для работы с Grafana необходимо использовать InfluxDB версии 2.

Примечание. Перед обновлением убедитесь, что старый контейнер LUNA Dashboards удален.

5.4.1.1 Запуск контейнера LUNA Dashboards

Используйте команду `docker run` со следующими параметрами для запуска LUNA Dashboards:

```
docker run \
--restart=always \
--detach=true \
--network=host \
--name=grafana \
-v /etc/localtime:/etc/localtime:ro \
dockerhub.visionlabs.ru/luna/luna-dashboards:v.0.1.3
```

Для использования веб-интерфейса Grafana нужно перейти по адресу «`http://IP_ADDRESS:3000`», при условии, что контейнеры LUNA Dashboards и InfluxDB были запущены.

5.4.2 Grafana Loki

Примечание. Для запуска Grafana Loki требуется наличие запущенного сервиса LUNA Dashboards.

Примечание. Перед обновлением убедитесь, что старые контейнеры Grafana Loki и Promtail удалены.

5.4.2.1 Запуск контейнера Grafana Loki

Используйте команду `docker run` со следующими параметрами для запуска 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
```

5.4.2.2 Запуск контейнера Promtail

Используйте команду `docker run` со следующими параметрами для запуска 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=
```

`-v /var/lib/luna/current/example-docker/logging/promtail.yml:/etc/promtail/luna.yml` — монтирование конфигурационного файла в контейнер Promtail

`-config.file=/etc/promtail/luna.yml` — флаг с адресом конфигурационного файла

`-client.url=http://127.0.0.1:3100/loki/api/v1/push` — флаг с адресом развернутой Grafana Loki

`-client.external-labels=job=containerlogs,pipeline_id=,job_id=,version=` — статические метки для добавления ко всем логам, отправляемым в Grafana Loki

5.5 Команды Docker

5.5.1 Показать контейнеры

Чтобы показать список запущенных Docker-контейнеров, используйте команду:

```
docker ps
```

Чтобы показать все имеющиеся Docker-контейнеры, используйте команду:

```
docker ps -a
```

5.5.2 Копировать файлы в контейнер

Можно переносить файлы в контейнер. Используйте команду `docker cp` для копирования файла в контейнер.

```
docker cp <file_location> <container_name>:<folder_inside_container>
```

5.5.3 Вход в контейнер

Можно входить в отдельные контейнеры с помощью следующей команды:

```
docker exec -it <container_name> bash
```

Для выхода из контейнера используйте следующую команду:

```
exit
```

5.5.4 Имена образов

Можно увидеть все имена образов с помощью команды

```
docker images
```

5.5.5 Удаление образа

Если требуется удаление образа:

- запустите команду `docker images`
- найдите требуемый образ, например `dockerhub.visionlabs.ru/luna/luna-image-store`
- скопируйте соответствующий ID образа из IMAGE ID, например, «61860d036d8c»
- укажите его в команде удаления:

```
docker rmi -f 61860d036d8c
```

Удалите все существующие образы:

```
docker rmi -f $(docker images -q)
```

5.5.6 Остановка контейнера

Контейнер можно остановить с помощью следующей команды:

```
docker stop <container_name>
```

Остановить все контейнеры:

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

5.5.7 Удаление контейнера

Если необходимо удалить контейнер:

- запустите команду «`docker ps`»
- остановите контейнер (см. [Остановка контейнера](#))
- найдите требуемый образ, например: `dockerhub.visionlabs.ru/luna/luna-image-store`
- скопируйте соответствующий ID контейнера из столбца CONTAINER ID, например, «23f555be8f3a»
- укажите его в команде удаления:

```
docker container rm -f 23f555be8f3a
```

Удалить все контейнеры:

```
docker container rm -f $(docker container ls -aq)
```

5.5.7.1 Проверка логов сервисов

Чтобы показать логи сервиса, используйте команду:

```
docker logs <container_name>
```

5.6 Описание параметров запуска

При запуске Docker-контейнера для какого-либо из сервисов LUNA PLATFORM необходимо задать дополнительные параметры, требуемые для работы этого сервиса.

Параметры, требуемые для конкретного контейнера, описаны в разделе, посвященном запуску этого контейнера.

Все параметры, приведенные в примере запуска сервиса, необходимы для корректного запуска и работы сервиса.

5.6.1 Параметры запуска сервисов

Пример команды запуска контейнеров сервисов LP:

```
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>
```

Следующие параметры используются при запуске контейнеров сервисов LP:

- `docker run` — команда для запуска выбранного образа в качестве нового контейнера.
- `dockerhub.visionlabs.ru/luna/<service-name>:<version>` — позволяет задать образ, требуемый для запуска контейнера.

Ссылки для загрузки требуемых образов контейнера доступны в описании запуска соответствующего контейнера.

- `--network=host` — указывает, что отсутствует симуляция сети и используется серверная сеть. При необходимости изменить порт для сторонних контейнеров следует заменить эту строку на `-p 5440:5432`. Здесь первый порт 5440 — это локальный порт, а 5432 — это порт, используемый в контейнере. Пример приведен для PostgreSQL.

- `--env=` — задает переменные окружения, требуемые для запуска контейнера (см. раздел «Аргументы сервисов»).
- `--name=<service_container_name>` — задает имя запускаемого контейнера. Имя должно быть уникальным. Если уже существует контейнер с таким же именем, произойдет ошибка.
- `--restart=always` — определяет политику перезагрузки. Демон всегда перезагружает контейнер вне зависимости от кода завершения.
- `--detach=true` — позволяет запустить контейнер в фоновом режиме.
- `-v` — позволяет загружать содержимое серверной папки в объем контейнера. Таким образом содержимое синхронизируется. Загружаются следующие общие данные:
- `/etc/localtime:/etc/localtime:ro` — задает текущий часовой пояс, используемый системой контейнера.
- `/tmp/logs/<service>:/srv/logs/` — позволяет копировать папку с записями (логами) сервиса на сервер в директорию `/tmp/logs/<service>`. Директорию для хранения логов можно изменить при желании.

5.6.1.1 Аргументы сервисов

Каждый сервис в LUNA PLATFORM имеет свои собственные аргументы запуска. Эти аргументы можно передать через:

- задание флага для скрипта запуска (`run.py`) соответствующего сервиса
- установку отдельных переменных окружения (`--env`) в командной строке Docker

Например, с использованием флага `--help` можно получить список всех доступных аргументов. Пример передачи аргумента для сервиса API может выглядеть следующим образом:

```
docker run --rm dockerhub.visionlabs.ru/luna/luna-api:v.6.41.0 python3 /srv/luna_api/run.py --help
```

Список основных аргументов:

Флаг в строке запуска	Переменная окружения	Описание
<code>--port</code>	PORT	Порт, на котором сервис будет ожидать подключений.
<code>--workers</code>	WORKER_COUNT	Количество «рабочих процессов» для сервиса.
<code>--log_suffix</code>	LOG_SUFFIX	Суффикс, добавляемый к именам файлов логов (при включенном параметре записи логов в файл).

<code>--config-reload</code>	RELOAD_CONFIG	Включение автоматической перезагрузки конфигураций. См. раздел «Автоматическая перезагрузка конфигураций» в руководстве администратора LUNA PLATFORM 5.
<code>--pulling-time</code>	RELOAD_CONFIG_INTERVAL	Период проверки конфигураций (по умолчанию 10 секунд). См. раздел «Автоматическая перезагрузка конфигураций» в руководстве администратора LUNA PLATFORM 5.
<code>--luna-config</code>	CONFIGURATOR_HOST, CONFIGURATOR_PORT	Адрес сервиса Configurator для загрузки настроек. Для <code>--luna-config</code> передается в формате <code>http://localhost:5070/1</code> . Для переменных окружения хост и порт задаются явно. Если аргумент не задан, то будет использован конфигурационный файл по умолчанию.
<code>--config</code>	Нет	Путь до конфигурационного файла с настройками сервиса.
<code>--<config_name></code>	Нет	Тег указанной настройки в Configurator. При задании данной настройки будет использовано значение тегированной настройки. Пример: <code>--LUNA_MONITORING TAG_1</code> . Примечание. Необходимо заранее присвоить тег соответствующим настройкам в Configurator. Примечание. Работает только с флагом <code>--luna-config</code> .
<code>--tls-cert</code>	Нет	Путь к SSL-сертификату для запуска сервиса с использованием протокола HTTPS.

<code>--tls_key</code>	Нет	Путь к SSL-закрытому ключу для запуска сервиса с использованием протокола HTTPS.
<code>--tls_key_pass</code>	Нет	Пароль для SSL-закрытого ключа для запуска сервиса с использованием протокола HTTPS.

Перечень аргументов может отличаться в зависимости от сервиса.

Также доступна возможность переопределить настройки сервисов при их старте с помощью переменных окружения.

Для переопределения настроек используется префикс `VL_SETTINGS`. Примеры:

- `--env=VL_SETTINGS.LUNA_MONITORING.SEND_DATA_FOR_MONITORING=0`. Использование переменной окружения из данного примера установит значение настройки «`SEND_DATA_FOR_MONITORING`» для секции «`LUNA_MONITORING`» равным «0».
- `--env=VL_SETTINGS.OTHER.STORAGE_TIME=LOCAL`. Для несоставных настроек (настроек, которые расположены в секции «`OTHER`» в конфигурационном файле) необходимо указать префикс «`OTHER`». Использование переменной окружения из данного примера установит значение настройки «`STORAGE_TIME`» (если сервис использует данную настройку) на значение «`LOCAL`».

Передача флагов с использованием переменной окружения

Флаги, для которых явно не выделена переменная окружения, можно передать с помощью переменной окружения `EXTEND_CMD`.

Например, можно передать тег настроек следующим способом:

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

5.6.2 Параметры создания баз данных

Пример команды запуска контейнеров для миграции баз данных или их создания:

```
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
```

Следующие параметры используются при запуске контейнеров для миграции баз данных или их создания:

`--rm` — этот параметр указывает, удаляется ли контейнер после завершения обработки всех заданных скриптов.

`python3 ./base_scripts/db_create.py` — этот параметр содержит версию Python и скрипт `db_create.py`, запускаемый в контейнере. Этот скрипт используется для создания структуры базы данных.

`--luna-config http://localhost:5070/1` — этот параметр указывает, откуда запущенный скрипт должен получать конфигурации. По умолчанию конфигурации запрашиваются сервисами от сервиса Configurator.

5.7 Запись логов на сервер

Чтобы включить сохранение логов на сервер, необходимо:

- создать директории для логов на сервере;
- активировать запись логов и задать расположение хранения логов внутри контейнеров сервисов LP;
- настроить синхронизацию директорий логов в контейнере с логами на сервере с помощью аргумента `volume` при старте каждого контейнера.

5.7.1 Создание директории логов

Ниже приведены примеры команд для создания директорий для хранения логов и присвоения им прав для всех сервисов LUNA PLATFORM.

```
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 /tmp/logs/luna-video-agent /tmp/logs/luna-  
video-manager
```

```
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 /tmp/logs/luna-video-agent /tmp/logs/  
luna-video-manager
```

Если необходимо использовать сервис Python Matcher Proxy, то нужно дополнительно создать директорию `/tmp/logs/python-matcher-proxy` и установить ей разрешения.

5.7.2 Активация записи логов

5.7.2.1 Активация записи логов сервисов LP

Для активации записи логов в файл необходимо задать настройки `log_to_file` и `folder_with_logs` в секции `<SERVICE_NAME>_LOGGER` настроек каждого сервиса.

Автоматический способ (перед/после запуска Configurator)

Для обновления настроек ведения логов можно использовать файл настроек `logging.json`, предоставленный в комплекте поставки.

Выполните следующую команду после запуска сервиса Configurator:

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

Обновите настройки записи логов с помощью скопированного файла.

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

Ручной способ (после запуска Configurator)

Перейдите в интерфейс сервиса Configurator (127.0.0.1:5070) и задайте путь расположения логов в контейнере в параметре `folder_with_logs` для всех сервисов, чьи логи необходимо сохранить. Например, можно использовать путь `/srv/logs`.

Установите параметр `log_to_file` как `true` чтобы активировать запись логов в файл.

5.7.2.2 Активация записи логов сервиса Configurator (перед/после запуска Configurator)

Настроек сервиса Configurator нет в пользовательском интерфейсе Configurator, они расположены в следующем файле:

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

Следует изменить параметры логирования в этом файле перед запуском сервиса Configurator или перезапустить его после внесения изменений.

Задайте путь расположения логов в контейнере в параметре `FOLDER_WITH_LOGS` = `./` файла. Например, `FOLDER_WITH_LOGS` = `/srv/logs`.

Установите параметр `log_to_file` как `true` чтобы активировать запись логов в файл.

5.7.3 Монтирование директорий с логами при старте сервисов

Директория с логами монтируется с помощью следующего аргумента при старте контейнера:

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

где `<server_logs_folder>` директория, созданная на этапе [создания директории логов](#), а `<container_logs_folder>` директория, созданная на этапе [активации записи логов](#).

Пример команды запуска сервиса API с монтированием директории с логами:

```
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.6.41.0
```

Примеры команд запуска контейнеров в данной документации содержат эти аргументы.

5.8 Настройка ротации логов Docker

Чтобы ограничить размер логов, генерируемых Docker, можно настроить автоматическую ротацию логов. Для этого необходимо добавить в файл `/etc/docker/daemon.json` следующие данные:

```
{
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m",
    "max-file": "5"
  }
}
```

Это позволит Docker хранить до 5 файлов логов на контейнер, при этом каждый файл будет ограничен 100 МБ.

После изменения файла необходимо перезапустить Docker:

```
systemctl reload docker
```

Вышеописанные изменения являются значениями по умолчанию для любого вновь созданного контейнера, они не применяются к уже созданным контейнерам.

5.8.1 Задание пользовательских настроек InfluxDB

Для InfluxDB OSS 2 доступны следующие настройки:

```
"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>
```

Можно обновить настройки InfluxDB для сервисов LP в сервисе Configurator, выполнив следующие действия:

- откройте следующий файл:

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

- задайте необходимые данные;
- сохраните изменения;
- скопируйте файл в контейнер InfluxDB:

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

- обновите настройки в сервисе Configurator.

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

Также можно вручную обновить настройки в пользовательском интерфейсе сервиса Configurator.

Настройки сервиса Configurator задаются отдельно.

- откройте файл с настройками Configurator:

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

- задайте необходимые данные;
- сохраните изменения;
- перезапустите Configurator:

```
docker restart luna-configurator
```

5.9 Использование Python Matcher с Python Matcher Proxy

Как было сказано ранее, вместе с сервисом Python Matcher можно дополнительно использовать сервис Python Matcher Proxy, который будет перенаправлять запросы сравнения либо сервису Python Matcher, либо плагинам сравнения. Использование плагинов может значительно ускорить выполнение запросов на сравнение. Например, с помощью плагинов возможно организовать хранение необходимых для выполнения операций сравнения данных и дополнительных полей объектов в отдельном хранилище, что позволит ускорить доступ к данным по сравнению с использованием стандартной БД LUNA PLATFORM.

Для использования сервиса Python Matcher с Python Matcher Proxy необходимо дополнительно запустить соответствующий контейнер, а затем выставить определенную настройку в сервисе Configurator. Выполняйте нижеперечисленные действия только если собираетесь использовать плагины сравнения.

См. описание и использование плагинов сравнения в руководстве администратора.

5.9.1 Запуск контейнера Python Matcher Proxy

Используйте следующую команду для запуска сервиса:

```
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.1.12.2
```

После запуска контейнера необходимо выставить следующее значение в сервисе Configurator.

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