VisionLabs
MACHINES CAN SEE

VisionLabs LUNA PLATFORM 5

Руководство по установке с использованием Storages

v.5.59.0

Содержание

Стандартные порты для сервисов

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 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
Backport 3	luna-backport3
Backport 4	luna-backport4

Настройки для сервиса Configurator устанавливаются в его конфигурационном файле.

Системные требования

LUNA PLATFORM поставляется в Docker-контейнерах и может запускаться на CPU и GPU. Для установки необходимы образы Docker-контейнеров LP. Для загрузки образов Docker на сервере требуется подключение к сети Интернет, либо образы следует загрузить на любое другое устройство и перенести на сервер. Требуется вручную указать логин и пароль для загрузки образов Docker.

LUNA PLATFORM можно запустить с помощью скрипта Docker Compose.

Запуск LUNA PLATFORM был протестирован на следующих версиях Docker и Docker Compose:

- Docker: 25.0.3 (для ручного запуска контейнеров)
- Docker Compose: 2.24.6 (для автоматического запуска контейнеров)

Не гарантируется работа с более высокими версиями Docker или Docker Compose.

Для использования механизма плагинов требуется наличие Python 3.11 или выше.

Запуск контейнеров LUNA PLATFORM официально поддерживается на CentOS 7/8. Корректная работа на других системах не гарантируется. Все процедуры в руководстве по установке описаны для CentOS 7.

В сервисах LUNA PLATFORM используются операционная система CentOS Linux 8.3.2011.

Процессоры

Приведенная ниже конфигурация обеспечит минимальную мощность для работы ПО, но для использования системы в продуктивном контуре этого недостаточно. Требования для использования системы в продуктивном контуре рассчитываются в зависимости от предполагаемой нагрузки.

CPU

Следующие минимальные системные требования необходимы для установки программного пакета LUNA PLATFORM:

- CPU Intel, минимум 4 физических ядра с тактовой частотой 2.0 GHz или выше. Требуется поддержка набора инструкций AVX2 для CPU;
- RAM DDR3 (рекомендуется DDR4), 8 Гб или выше;
- Свободное место на диске — минимум 80 Гб.

> Необходимое количество свободного места на диске напрямую зависит от размера БД и бакетов Image Store. Если база данных очень велика, то может потребоваться более 80 Гб.

> Рекомендуется использование SSD для баз данных и хранилища Image Store.

GPU

Для ускорения GPU необходим NVIDIA GPU. Поддерживаются следующие архитектуры:

- Pascal или более новые.

Требуется Compute Capability 6.1 или выше.

Требуется минимум 6Гб оперативной или выделенной видеопамяти. Рекомендуется 8 Гб VRAM или более.

CUDA версии 11.4 должна быть установлена на сервере сервиса Remote SDK. Рекомендуемый драйвер NVIDIA — r470.

Сторонние приложения

Следующие сторонние приложения используются по умолчанию с LUNA PLATFORM 5.

- PostgreSQL используется в качестве базы данных по умолчанию для сервисов Faces, Configurator, Events, Handlers, Lambda, Tasks, Admin, и Backport3.

> Также возможно использование базы данных Oracle вместо PostgreSQL для всех сервисов кроме сервиса Events. Установка и конфигурация Oracle не описывается в данном руководстве.

- Для сервисов Faces и Sender используется БД Redis.
- Для мониторинга используется БД Influx.

Балансировщики и другие программы могут использоваться при масштабировании системы для обеспечения отказоустойчивости. В руководстве по установке приводятся рекомендации по запуску контейнера Nginx с конфигурационным файлом для балансировки запросов к сервисам API, Faces, Image Store и Events.

Для использования LP рекомендуются следующие версии сторонних приложений:

- PostgreSQL: 16
- Oracle: 21c (если используется вместо PostgreSQL)
- Redis: 7.2
- InfluxDB: 2.0.8-alpine
- Grafana: 8.5.20 (опционально)
- Grafana Loki: 2.7.1 (опционально)
- Nginx: 1.17.4-alpine (опционально)

> Эти версии протестированы специалистами VisionLabs. При необходимости можно использовать более новые версии, но их работоспособность не гарантируется.

Для распаковки дистрибутива рекомендуется использовать пакет `unzip`. Команда для скачивания пакета дана в инструкции по установке.

Если необходимо использовать внешнюю базу данных и функцию VLMatch, требуется загрузить дополнительные зависимости, описанные в руководстве по установке (см. раздел «Внешняя база данных»).

Docker-контейнеры PostgreSQL, Redis, InfluxDB, Grafana и Nginx можно загрузить из реестра VisionLabs.

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 обновляетесь 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 обновляетесь с версии LUNA PLATFORM 5.58.0.

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 баз данных PostgreSQL, 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 5.58.0
2. Создать резервные копии
3. Удалить старую символическую ссылку
4. Распаковать дистрибутив новой версии LUNA PLATFORM
5. Создать новую символическую ссылку
6. Изменить группу и владельца для новых директорий
7. Сохранить пользовательские настройки сервиса Configurator, если они изменялись
8. Создать директории с логами, если ранее использовалась запись логов в файлы
9. Обновить лицензию, если это необходимо
10. Настроить вычисления с помощью GPU, если планируется использовать GPU
11. 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
5.53.0	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 базы данных PostgreSQL

Резервная копия БД 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 дамп-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 Удаление символической ссылки

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

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

1.4 Распаковка дистрибутива

Дистрибутив представляет собой архив **luna_v.5.59.0**, где **v.5.59.0** 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.5.59.0.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.5.59.0.zip
```

1.5 Создание символической ссылки

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

```
ln -s luna_v.5.59.0 current
```

1.6 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
```

1.7 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
```

1.8 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/lambda
```

```
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/lambda
```

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

1.9 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

1.9.1 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.

1.10 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.

1.11 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.

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=
   kofqt4Pfqjn6o0RBtMDQqVoJLgHoxxDUmmhiAZ7JS6VmEnrqZXQhxDhad8AX9tmiJH6CjM7Y1U8p5eSEocG
   == \
-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-vlmatch: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-vlmatch: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.

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
```

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

Подготовьте окружение с помощью следующей команды:

```
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.0.2.0 \
bash -c "luna_prepare prepare all_entities \
    --platform_version=v.5.59.0 \
    --config=/srv/storages_config.conf \
    --profile=backports"
```

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/current/extras/conf/storages_config.conf:/srv/storages_config.conf` — команда монтирования конфигурационного файла Storages
- `--config=/srv/storages_config.conf` — именованный аргумент, содержащий адрес конфигурационного файла для использования сервисом Storages

3.2 Использование необязательных сервисов

Следующие сервисы необязательны для LP:

- Events
- Image Store
- Tasks
- Sender
- Handlers
- Python Matcher Proxy (отключен по умолчанию)
- Lambda (отключен по умолчанию)

Эти сервисы можно отключить при отсутствии необходимости в них.

Используйте секцию «ADDITIONAL_SERVICES_USAGE» в настройках сервиса API в сервисе Configurator, чтобы отключить ненужные сервисы.

Можно использовать файл сброса, предоставленный в комплекте поставки, для включения/отключения сервисов перед запуском сервиса Configurator.

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

> Отключение какого-либо из сервисов имеет определенные последствия. См. подробную информацию в разделе «Отключаемые сервисы» руководства администратора.

> Обратите внимание, что сервиса Storages не будет выполнять подготовку окружения для сервисов, которые отключены в настройке «ADDITIONAL_SERVICES_USAGE».

3.3 Загрузка дамп-файла

Примечание. Можете пропустить данное действие если нет необходимости в загрузке дамп-файла с настройками.

Выполните следующую команду для загрузки дамп-файла в 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.0.2.0 \
bash -c "luna_prepare load_dump \
    --dump-file=/srv/platform_settings.json"
```

Здесь:

- `luna_prepare load_dump` — команда «load_dump», позволяющая загрузить дамп-файл в БД Configurator
- `-v /var/lib/luna/current/extras/conf/platform_settings.json:/srv/platform_settings.json \` — команда монтирования дамп-файла `platform_settings.json`
- `--dump-file=/srv/platform_settings.json` — именованный аргумент, содержащий адрес дамп-файла внутри контейнера

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
- API
- Admin

> Сервис Lambda (отключен по умолчанию) можно запустить после сервисов Licenses и Configurator.

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.2.2.0
```

> 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.3.11.0
```

`-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.0.3.0
```

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.0.10.0
```

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.4.11.0
```

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.4.12.0
```

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.1.9.0
```

4.8 Remote SDK

4.8.1 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.

4.8.1.1 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.0.5.0
```

4.8.1.2 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.0.5.0
```

- `--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.

4.8.1.3 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
--enable-all-estimators-by-default	включить все эстиматоры по умолчанию
--enable-human-detector	одновременный детектор
--enable-face-detector	детектор лиц
--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

Аргумент	Описание
--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
--enable-deepfake-estimator	эстиматор Deepfake

> См. подробную информацию включении и выключении 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.0.5.0
```

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.3.5.0
```

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.3.21.0
```

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.3.21.0
```

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.2.11.0
```

4.12 API

4.12.1 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.6.26.0
```

> 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».

4.13 Admin

4.13.1 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.5.6.0
```

4.14 Backport 3

В данном разделе описывается запуск сервиса Backport 3.

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

4.14.1 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.0.11.0
```

4.14.2 User Interface 3

User Interface 3 используется только с сервисом Backport 3.

4.14.2.1 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.0.5.11
```

`--env=LUNA_API_URL` — URL сервиса Backport 3.

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

4.15 Backport 4

В данном разделе описывается запуск сервиса Backport 4.

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

4.15.1 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.1.5.0
```

4.15.2 User Interface 4

User Interface 4 используется только с сервисом Backport 4.

4.15.2.1 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.0.1.6
```

`--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).

4.16 Lambda

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

Работа с сервисом Lambda возможна только при разворачивании сервисов LUNA PLATFORM в Kubernetes. Для использования необходимо самостоятельно развернуть сервисы LUNA PLATFORM в Kubernetes или обратиться за консультацией к специалистам VisionLabs. Используйте команды ниже в качестве справочной информации.

Включите использование сервиса Lambda (см. раздел «Использование необязательных сервисов»).

4.16.1 Подготовка Docker registry

Необходимо подготовить registry для хранения образов Lambda. Перенесите базовые образы и образ Kaniko executor в свой registry с помощью нижеперечисленных команд.

Загрузите образы из удаленного репозитория в локальное хранилище образов:

```
docker pull dockerhub.visionlabs.ru/luna/lpa-lambda-base-fsdk:v.0.0.69
```

```
docker pull dockerhub.visionlabs.ru/luna/lpa-lambda-base:v.0.0.69
```

```
docker pull dockerhub.visionlabs.ru/luna/kaniko-executor:latest
```

Добавьте новые имена образам, заменив `new-registry` на свои. Имена базовых образов в пользовательском реестре должны быть такими же, как и в реестре `dockerhub.visionlabs.ru/luna`.

```
docker tag dockerhub.visionlabs.ru/luna/lpa-lambda-base-fsdk:v.0.0.69 new-
   registry/lpa-lambda-base-fsdk:v.0.0.69
```

```
docker tag dockerhub.visionlabs.ru/luna/lpa-lambda-base:v.0.0.69 new-
   registry/lpa-lambda-base:v.0.0.69
```

```
docker tag dockerhub.visionlabs.ru/luna/kaniko-executor:latest new-registry/
   kaniko-executor:latest
```

Отправьте локальные образы в свой удаленный репозиторий, заменив `new-registry` на свои.

```
docker push new-registry/lpa-lambda-base-fsdk:v.0.0.69
```

```
docker push new-registry/lpa-lambda-base:v.0.0.69
```

```
docker push new-registry/kaniko-executor:latest
```

4.16.2 Запуск контейнера Lambda

Используйте следующую команду для запуска сервиса:

```
docker run \
--env=CONFIGURATOR_HOST=127.0.0.1 \
--env=CONFIGURATOR_PORT=5070 \
--env=PORT=5210 \
--env=WORKER_COUNT=1 \
--env=RELOAD_CONFIG=1 \
--env=RELOAD_CONFIG_INTERVAL=10 \
-v /etc/localtime:/etc/localtime:ro \
-v /tmp/logs/lambda:/srv/logs \
--name=luna-lambda \
--restart=always \
--detach=true \
--network=host \
dockerhub.visionlabs.ru/luna/luna-lambda:v.0.3.0
```

5 Дополнительная информация

В данном разделе приводится следующая дополнительная информация:

- Создание аккаунта
- Создание расписания для задачи по очистке мусора
- Визуализация монтиторинга 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 Визуализация монтиторинга 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).

5.3.1 LUNA Dashboards

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

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

5.3.1.1 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.0.0.9
```

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

5.3.2 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.

5.3.2.1 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
```

5.3.2.2 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

5.4 Команды Docker

5.4.1 Показать контейнеры

Чтобы показать список запущенных Docker-контейнеров, используйте команду:

```
docker ps
```

Чтобы показать все имеющиеся Docker-контейнеры, используйте команду:

```
docker ps -a
```

5.4.2 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>
```

5.4.3 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
```

5.4.4 Имена образов

Можно увидеть все имена образов с помощью команды

```
docker images
```

5.4.5 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)
```

5.4.6 Остановка контейнера

Контейнер можно остановить с помощью следующей команды:

```
docker stop <container_name>
```

Остановить все контейнеры:

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

5.4.7 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)
```

5.4.7.1 Проверка логов сервисов

Чтобы показать логи сервиса, используйте команду:

```
docker logs <container_name>
```

5.5 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.

5.5.1 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. Daemon 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.

5.5.1.1 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.6.26.0 python3 /srv/
   luna_api/run.py --help
```

Список основных аргументов:

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

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

`--tls_key`	Нет	Путь к SSL-закрытому ключу для запуска сервиса с использованием протокола HTTPS.
`--tls_key_pass`	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.INFLUX_MONITORING.SEND_DATA_FOR_MONITORING=0`. Использование переменной окружения из данного примера установит значение настройки «SEND_DATA_FOR_MONITORING» для секции «INFLUX_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="--INFLUX_MONITORING=TAG_1 --LUNA_EVENTS_DB=TAG_2"
```

5.5.2 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.

5.6 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.

5.6.1 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/lambda
```

```
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/lambda
```

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

5.6.2 Активация записи логов

5.6.2.1 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.

5.6.2.2 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.

5.6.3 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.6.26.0
```

Примеры команд запуска контейнеров в данной документации содержат эти аргументы.

5.7 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.

5.7.1 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
```

5.8 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.

5.8.1 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.1.9.0
```

После запуска контейнера необходимо выставить следующее значение в сервисе Configurator.

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