VisionLabs
MACHINES CAN SEE

VisionLabs LUNA PLATFORM 5

Пример развертывания с использованием Docker Compose

v.5.84.0

Содержание

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	5250
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

Данный документ описывает установку и использование Docker Compose для развертывания LUNA PLATFORM.

> Данный документ включает в себя пример развертывания LUNA PLATFORM с помощью скрипта Compose. Он использует минимальные ресурсы, необходимые для демонстрации работы LUNA PLATFORM и не может использоваться для реальных рабочих целей.

Считается, что установка выполняется на сервере с операционной системой Almalinux 8, где LP не была установлена.

Docker Compose используется для автоматического развертывания контейнеров. Скрипт Docker Compose из данного дистрибутива используется для развертывания сервисов LUNA PLATFORM на одном сервере.

Администратор должен вручную настроить Firewall и SELinux на сервере. В данном документе не описывается их настройка.

В данной инструкции по установке не предполагается резервное копирование или копирование баз данных для данных LP.

Для развертывания LUNA PLATFORM с помощью скрипта Docker Compose нужно выполнить действия из следующих разделов:

- «Подготовка к запуску» — действия по распаковке архивов, подготовке директорий, настройке лицензии и пр. Некоторые действия могут быть опциональными.
- «Запуск LUNA PLATFORM» — запуск скрипта Docker Compose для развертывания LUNA PLATFORM.

Данный документ также содержит инструкцию по автоматическому запуску LUNA Dashboards (Grafana) и Loki (см. раздел «Включение Grafana и Loki»).

В разделе «Дополнительная информация» приводится полезная информация по настройке логирования, включении ротации логов, командах Docker и пр.

Для использования скрипта Docker Compose требуется сетевая лицензия LUNA PLATFORM. Лицензия предоставляется компанией VisionLabs по запросу отдельно от поставки. Лицензионный ключ создается с помощью отпечатка системы. Этот отпечаток создается на базе информации об аппаратных характеристиках сервера. Таким образом, полученный лицензионный ключ будет работать только на том же сервере, с которого был получен отпечаток системы. LUNA PLATFORM можно активировать с помощью одной из двух утилит — HASP или Guardant. В разделе «Активация лицензии» приведены полезные ссылки на инструкции по активации лицензионного ключа для каждого способа.

Примечания о скрипте Docker Compose. Скрипт:

- Тестируется с использованием настроек сервисов по умолчанию.

- Готовит окружение с помощью утилиты Storages.
- Не предназначен для использования в целях масштабирования LP:
  - Не используется для развертывания сервисов LP на нескольких серверах.
  - Не используется для развертывания и балансирования нескольких сервисов LP на одном сервере.
- Запускает базы данных по умолчанию и не включает в себя встроенную возможность изменять используемые базы данных.
- Поддерживает использование GPU для вычислений LP.
- Не обеспечивает возможность использования внешних баз данных, уже установленных на сервере.
- Не выполняет миграции из предыдущих версий LP и обновления предыдущих сборок LP.

См. файл «docker-compose.yml» и другие файлы в директории «example-docker» для получения информации о запускаемых сервисах и выполненных действиях.

Можно написать собственный скрипт, который разворачивает и конфигурирует все необходимые сервисы. Данный документ не включает информацию о создании скриптов и не обучает использованию Docker. Обратитесь к документации Docker для получения подробной информации о Docker и Docker Compose:

https://docs.docker.com

> Рекомендуется использовать сервисы оркестрации для коммерческого использования LP. Их использование не описано в данном руководстве.

> Все описываемые команды необходимо исполнять в оболочке Bash (когда команды запускаются напрямую на сервере) или в программе для работы с сетевыми протоколами (в случае удаленного подключения к серверу), например, Putty.

Для активации LUNA PLATFORM требуется файл лицензии. Этот файл предоставляется компанией VisionLabs по запросу.

Все действия, описанные в данном руководстве, должны выполняться пользователем **root**. В данном документе не описывается создание пользователя с привилегиями администратора и последующая установка, выполняемая этим пользователем.

1 Подготовка к запуску

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

Перед запуском LUNA PLATFORM необходимо выполнить следующие действия:

1. Распаковать дистрибутив LUNA PLATFORM
2. Создать символическую ссылку
3. Настроить SELinux и Firewall
4. Активировать лицензию
5. Выполнить установку Docker
6. Выполнить установку Docker Compose
7. Выбрать способ записи логов
8. Авторизироваться в registry VisonLabs
9. Настроить вычисления с помощью GPU, если планируется использовать GPU

1.1 Распаковка дистрибутива

Дистрибутив представляет собой архив **luna_v.5.84.0**, где **v.5.84.0** 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.5.84.0.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.5.84.0.zip
```

1.2 Создание символической ссылки

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

```
ln -s luna_v.5.84.0 current
```

1.3 SELinux и Firewall

SELinux и Firewall необходимо настроить так, чтобы они не блокировали сервисы LUNA PLATFORM.

> Конфигурация SELinux и Firewall не описываются в данном руководстве.

Если SELinux и Firewall не настроены, дальнейшая установка невозможна.

1.4 Активация лицензии

Для активации лицензии необходимо выполнить следующие действия:

- выполнить действия из руководства по активации лицензии
- задать настройки лицензирования HASP или Guardant

1.4.1 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.

1.4.2 Задание настроек лицензии HASP

Для HASP-ключа нужно задать IP-адрес сервера лицензирования. Адрес задается в дамп-файле «platform_settings.json». Содержимое стандартных настроек будет перезаписано содержимым этого файла на этапе запуска сервиса Configurator.

Откройте файл «platform_settings.json»:

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

Задайте IP-адрес сервера с вашим ключом HASP в поле «server_address»:

```
{
    "value": {
        "vendor": "hasp",
        "server_address": "127.0.0.1"
    },
    "description":"License vendor config",
    "name":"LICENSE_VENDOR",
    "tags":[]
},
```

Сохраните файл.

> 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».

1.4.3 Задание настроек лицензии Guardant

Для Guardant-ключа нужно задать IP-адрес сервера лицензирования и идентификатор лицензии. Настройки задаются в дамп-файле «platform_settings.json». Содержимое стандартных настроек будет перезаписано содержимым этого файла на этапе запуска сервиса Configurator.

Откройте файл «platform_settings.json»:

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

Задайте следующие данные:

- IP-адрес сервера с вашим ключом Guardant в поле «server_address»
- идентификатор лицензии в формате `0x<your_license_id>`, полученный в разделе «Сохранение идентификатора лицензии» в руководстве по активацию лицензии, в поле «license_id»:

```
{
    "value": {
        "vendor": "guardant",
        "server_address": "127.0.0.1",
        "license_id": "0x92683BEA"
    },
    "description":"License vendor config",
    "name":"LICENSE_VENDOR",
    "tags":[]
},
```

Сохраните файл.

> 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».

1.5 Установка Docker

Установка Docker описана в официальной документации.

Примечание. При тестировании данной инструкции использовался Docker версии 25.0.3. Не гарантируется работа с более высокими версиями Docker.

Команды для быстрой установки приведены ниже.

> Проверьте официальную документацию на наличие обновлений при возникновении каких-либо проблем с установкой.

Установите зависимости:

```
yum install -y yum-utils device-mapper-persistent-data lvm2
```

Добавьте репозиторий:

```
yum-config-manager --add-repo https://download.docker.com/linux/centos/
   docker-ce.repo
```

Установите Docker:

```
yum -y install docker-ce docker-ce-cli containerd.io
```

Запустите Docker:

```
systemctl start docker
```

```
systemctl enable docker
```

Проверьте статус Docker:

```
systemctl status docker
```

1.6 Установка Docker Compose

Примечание. При тестировании данной инструкции использовался Docker Compose версии 2.24.6. Не гарантируется работа с более высокими версиями Docker Compose.

Загрузите бинарный файл Docker Compose:

```
curl -L "https://github.com/docker/compose/releases/download/v2.24.6/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

Установите разрешения на выполнение:

```
chmod +x /usr/local/bin/docker-compose
```

Создайте символическую ссылку:

```
ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

1.7 Выбор способа записи логов

В LUNA PLATFORM существует два способа вывода логов:

- стандартный вывод логов (stdout);
- вывод логов в файл.

Настройки вывода логов задаются в настройках каждого сервиса в секции `<SERVICE_NAME>_LOGGER`.

При необходимости можно использовать оба способа вывода логов.

> Для более подробной информации о системе логирования LUNA PLATFORM см. раздел «Логирование информации» в руководстве администратора.

1.7.1 Запись логов в stdout

Данный способ используется по умолчанию и для него не требуется выполнять дополнительных действий.

Рекомендуется настроить ротацию логов Docker для ограничения их размеров (см. раздел «Настройка ротации логов Docker»).

1.7.2 Запись логов в файл

Примечание. При включении сохранения логов в файле необходимо помнить о том, что логи занимают определенное место в хранилище, а процесс логирования в файл негативно влияет на производительность системы.

Для использования данного способа необходимо выполнить следующие дополнительные действия:

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

> В скрипте Docker Compose уже настроена синхронизация директорий логов, необходимо только создать директории и активировать запись логов.

См. инструкцию по включению записи логов в файлы в разделе «Запись логов на сервер».

1.8 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 необходимо установить Docker Compose v1.28.0+.

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
```

После обновления списка пакетов установите пакет `nvidia-docker2` и зависимости:

```
yum clean expire-cache
```

```
yum install -y nvidia-docker2
```

```
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.

Далее необходимо дополнительно добавить секцию `deploy` в поле `remote-sdk` в файл `docker-compose.yml`.

```
vi /var/lib/luna/current/example-docker/docker-compose.yml
```

```
remote-sdk:
   image: ${DOCKER_URL}/luna-remote-sdk:${LUNA_REMOTE_SDK_TAG}
   deploy:
     resources:
       reservations:
         devices:
         - driver: nvidia
           count: all
           capabilities: [gpu]
   restart: always
   ...
```

`driver` — в данном поле указывается драйвер для зарезервированного устройства(устройств);

`count` — в данном поле задается количество графических процессоров, которые должны быть зарезервированы (при условии, что хост содержит такое количество графических процессоров);

`capabilities` — данное поле выражает как общие, так и специфические возможности драйвера. Его необходимо задать, иначе будет возвращена ошибка при развертывании сервиса.

Для дополнительной информации см. следующую документацию:

https://docs.docker.com/compose/gpu-support/#enabling-gpu-access-to-service-containers.

> 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.

1.9 Авторизация в registry

При запуске контейнеров необходимо указать ссылку на образ, необходимый для запуска контейнера. Этот образ загружается из VisionLabs registry. Перед этим необходима авторизация.

> Логин и пароль можно запросить у представителя VisionLabs.

Введите логин <username>.

```
docker login dockerhub.visionlabs.ru --username <username>
```

После выполнения команды будет запрошен ввод пароля. Введите пароль.

> В команде `docker login` можно вводить логин и пароль одновременно, однако это не гарантирует безопасность, т.к. пароль можно будет увидеть в истории команд.

2 Запуск LUNA PLATFORM

Запуск Docker Compose осуществляется с помощью скрипта «start_platform.sh», расположенного в директории «example-docker».

> При необходимости можно модифицировать скрипт запуска Docker Compose под пользовательские нужды. Модификация скрипта предназначена только для опытных пользователей.

Во время запуска скрипта будет создан стандартный аккаунт типа **user** с логином `user@mail.com` и паролем `password`. Инструкция по созданию собственного аккаунта приведена ниже.

> См. подробную информацию об аккаунтах в разделе «Аккаунты, токены и способы авторизации» руководства администратора.

2.1 Запуск сервисов

Откройте директорию Docker Compose:

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

> Убедитесь в том, что контейнеры LP не запущены до выполнения скрипта. Попытка запустить контейнер с таким же именем, как существующий контейнер, приведет к ошибке. Если запущен один или несколько контейнеров LP, необходимо остановить их с помощью команды `docker container rm -f <container_name>`. Чтобы остановить все контейнеры, используйте `docker container rm -f $(docker container ls -aq)`.

Запуск Docker Compose:

> Необходимо выполнить вход в VisionLabs registry (см. раздел «Вход в registry»)

```
./start_platform.sh --common
```

Скрипт поддерживает следующие аргументы:

- `--common` - подготовка окружения с профилем `common` и запуск LUNA PLATFORM без Backport'ов и сервисов Video Manager и Video Agent
- `--extra backport3` - при активации вместе с `--common` будет использован профиль `backports` при подготовке окружения, а также запущен сервис Backport3 и его пользовательский интерфейс
- `--extra backport4` - при активации вместе с `--common` будет использован профиль `backports` при подготовке окружения, а также запущен сервис Backport3 и его пользовательский интерфейс

- `--extra videoanalytics` - при активации вместе с `--common` будет включено использование сервиса Video Manager и Video Agent в настройке «ADDITIONAL_SERVICES_USAGE», а также запущены сервисы Video Manager и Video Agent
- `--help/-h` - вывод справочной информации

Комбинирование флагов `--extra backport3`, `--extra backport4`, и `--extra videoanalytics` вместе с `--common` позволяет включить сразу несколько дополнительных сервисов.

Использование флагов `--extra` без флага `--common` невозможно.

Если аргументы не заданы, скрипт выполнит развертывание LUNA PLATFORM аналогично аргументу `--common`.

Развертывание контейнеров требует времени. Необходимо дождаться того, чтобы все сервисы были запущены перед началом работы с LUNA PLATFORM.

Проверьте статус всех запущенных Docker-контейнеров.

```
docker ps
```

2.1.1 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.

2.2 Создание аккаунта

Примечание. При запуске скрипта Docker Compose автоматически создается аккаунт типа «user» с логином «user@mail.com» и паролем «password». Инструкция по созданию аккаунта со своими аутентификационными данными приведена ниже.

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.

2.3 Активация 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.

2.4 Включение Grafana и Loki

Примечание. Выполняйте следующие действия если хотите использовать LUNA Dashboards (Grafana) и Loki. В противном случае, пропустите данный шаг.

Для использования Grafana и Loki можно выполнить скрипт `start_logging.sh`, запускающий сервис LUNA Dashboards, Loki и Promtail. Данный скрипт нужно выполнять после запуска основного скрипта Docker Compose.

> См. подробную информацию о визуализации мониторинга в разделах «LUNA Dashboards» и «Grafana Loki» руководства администратора.

Откройте директорию Docker Compose:

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

Запустите Docker Compose:

```
./start_logging.sh
```

Проверьте статус запущенных Docker-контейнеров.

```
docker ps
```

3 Дополнительная информация

В данном разделе приводится следующая дополнительная информация:

- Полезные команды для работы с Docker
- Действия по включению сохранения логов сервисов LP в файлы
- Настройка ротации логов Docker

3.1 Команды Docker

3.1.1 Показать контейнеры

Чтобы показать список запущенных Docker-контейнеров, используйте команду:

```
docker ps
```

Чтобы показать все имеющиеся Docker-контейнеры, используйте команду:

```
docker ps -a
```

3.1.2 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>
```

3.1.3 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
```

3.1.4 Имена образов

Можно увидеть все имена образов с помощью команды

```
docker images
```

3.1.5 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)
```

3.1.6 Остановка контейнера

Контейнер можно остановить с помощью следующей команды:

```
docker stop <container_name>
```

Остановить все контейнеры:

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

3.1.7 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)
```

3.1.7.1 Проверка логов сервисов

Чтобы показать логи сервиса, используйте команду:

```
docker logs <container_name>
```

3.2 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.

3.3 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.

> В скрипте Docker Compose уже настроена синхронизация директорий с папками, создаваемыми в разделе ниже.

3.3.1 Создание директории логов

Необходимо создать следующие директории для хранения логов и присвоить им соответствующие права.

```
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.

3.3.2 Активация записи логов

3.3.2.1 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

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

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.

3.3.2.2 Активация записи логов сервиса 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
```

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.

> Необходимо перезапустить Configurator после внесения изменений.