VisionLabs
MACHINES CAN SEE

VisionLabs LUNA PLATFORM

Руководство по установке

ООО «ВижнЛабс»

123458, г. Москва, ул. Твардовского д. 8, стр. 1

+7 (499) 399 3361

info@visionlabs.ru

www.visionlabs.ru

Содержание

Глоссарий

Термин	Сокращение
LUNA PLATFORM	LP, LUNA
LUNA PLATFORM CORE	CORE
LUNA PLATFORM API	API
LUNA PLATFORM Faces	Faces
LUNA PLATFORM Image Store	Image Store
LUNA PLATFORM Broker	Broker
LUNA PLATFORM Extractor	Extractor
LUNA PLATFORM Matcher	Matcher
LUNA PLATFORM Administration Panel Service	Administration Panel Service
LUNA PLATFORM User Interface	User Interface, UI
LUNA PLATFORM Event & Statistic Service	Event & Statistic Service
LUNA PLATFORM Index Building and Index Search Services	Index Building and Index Search Services
LUNA PLATFORM Matcher Daemon	Matcher Daemon
LUNA PLATFORM Index Manager	Index Manager
LUNA PLATFORM Indexer	Indexer
LUNA PLATFORM Indexed Matcher	Indexed Matcher

Порты по умолчанию для сервисов

Название сервиса	Порт
LUNA PLATFORM API	5000
LUNA PLATFORM Faces	5030
LUNA PLATFORM Image Store	5020
Administration Panel Service Backend	5010
Administration Panel Service Tasks	5011
User Interface	80
Event Service	5009
Statistic Service	5008
Matcher Daemon	6001

Введение

Данный документ описывает процедуру установки пакета программного обеспечения VisionLabs LUNA PLATFORM и дополнительных сервисов.

В комплект поставки входят основные и дополнительные сервисы.

Основные сервисы и необходимые для их работы компоненты:

- сервис HASP,
- CORE:
  - СУБД Aerospike,
  - сервис Broker,
  - RabbitMQ,
  - сервис обработчика Extractor,
  - сервис обработчика Matcher.
- API:
  - СУБД PostgreSQL или СУБД Oracle,
  - сервис Faces,
  - сервис Image Store.

Дополнительные сервисы и их компоненты:

- сервис Index:
  - Matcher Daemon,
  - Index Manager,
  - Indexed Matcher,
  - Indexer.
- сервис User Interface,
- сервис Administration Panel,
- сервис Event& Statistic:
  - InfluxDB,
  - СУБД Redis.

Установка дополнительных сервисов не обязательна для работы LUNA PLATFORM.

Для работы LUNA PLATFORM требуется файл лицензии, который поставляется компанией VisionLabs отдельно по запросу.

Выполнять установку компонентов следует в указанном в документе порядке.

Все команды, приведенные в документе, должны выполняться пользователем с правами администратора или пользователем root.

Рекомендуется ознакомиться с документом SystemOverview.pdf перед установкой. Это позволит понять, из каких компонентов состоит LUNA PLATFORM и какие задачи они решают.

Системные требования

Для установки полного пакета ПО LUNA должны выполняться следующие системные требования:

- ОС CentOS 7.4 x86_64;
- Центральный процессор Intel, не менее 4 физических ядер с тактовой частотой не менее 2,0 ГГц;
- Поддержка инструкций:
  - Минимальные требования: SSE 4.2;
  - Рекомендуемые требования: AVX 2;
- оперативная память DDR3 (рекомендуется DDR4), не менее 8 Гб;
- доступ в интернет (для установки зависимостей);
- наличие свободного пространства не менее 10 Гб. Рекомендуется использовать SSD.

Приведенная выше конфигурация обеспечит работу программного комплекса на минимальной мощности и не предназначена для систем в продуктивном контуре. Конфигурация продуктивного контура высчитывается на основании предполагаемой нагрузки на систему.

Соглашения

Комментарии и пояснения к коду начинаются с символа # и выделены серым цветом.

1 Selinux и Firewall

Необходимо настроить Selinux и Firewall, чтобы они не блокировали работу сервисов LUNA PLATFORM.

Примечание. Настройка Selinux и Firewall не описана в данной инструкции.

2 Подготовка и распаковка дистрибутива

Дистрибутив представляет собой архив вида luna_v.X.Y.ZZ, где X.Y.ZZ – численный идентификатор, обозначающий версию продукта.

Архив содержит все компоненты, необходимые для установки и эксплуатации системы. Архив не включает зависимости, которые входят в стандартную поставку репозитория CentOS 7.4 x86_64 и могут быть загружены из открытых источников.

Перед процессом установки поместите файлы дистрибутива и лицензии в директорию /root/. В данной директории не должно быть других файлов дистрибутива и лицензии кроме целевых, используемых для установки конечного продукта

```
# Создайте директорию для распаковки дистрибутива
mkdir -p /var/lib/luna
# Переместите дистрибутив в созданную директорию
mv /luna* /var/lib/luna
# Установите архиватор unzip, если он не установлен
yum install -y unzip
# Перейдите в папку с дистрибутивом
cd /var/lib/luna
# Разархивируйте файлы
unzip luna*.zip
```

Создайте символьную ссылку, указав вместо X.Y.ZZ версию продукта. Ссылка указывает, что именно текущая версия дистрибутива используется для запуска.

```
ln -s luna_v.X.Y.ZZ current
```

Если отсутствует ссылка /var/lib/luna/current, скрипты запуска не смогут определить расположение бинарных файлов текущего релиза, а также будет невозможна дальнейшая установка.

```
# Вернитесь в корневой каталог
cd
```

2.1 Сервисные файлы

Для каждого из сервисов в поставке присутствует unit-файл. Следует скопировать эти файлы в папку system для запуска сервисов после их установки.

```
# Скопируйте файлы сервисов из комплекта поставки в системную директорию
cp /var/lib/luna/current/extras/systemd/luna-*.service
/etc/systemd/system/
# Перезагрузите системные сервисы
systemctl daemon-reload
```

3 Основные сервисы

3.1 Установка зависимостей и программного обеспечения

На этом шаге нужно установить расширенный пакет CentOS (epel-release) и все необходимые зависимости из него. Кроме того, нужно установить зависимости для LUNA PLATFORM, поставляемые с дистрибутивом и зависимости Python.

Примечание. Убедитесь, что у Вас есть права администратора, и подключён интернет.

```
# Переключитесь на пользователя root
sudo su

# Обновите системные пакеты до последних версий
yum -y --nogpgcheck update

# Установите epel-release для доступа к расширенному репозиторию пакетов (требуется для RabbitMQ и др. зависимостей)
yum -y --nogpgcheck install epel-release

# Установите необходимые зависимости из репозитория CentOS
yum -y --nogpgcheck install boost-system boost-chrono jemalloc libexif qpid-cpp-client boost-context boost-regex boost-thread double-conversion-devel glog libevent pyOpenSSL gcc rabbitmq-server libjpeg-devel zlib-devel ImageMagick-devel libwebp openssl-devel libatomic libarchive

# Установите зависимости для Postgresql. Следует уточнить актуальную версию у VisionLabs

wget https://download.postgresql.org/pub/repos/yum/9.6/redhat/rhel-7-x86_64/pgdg-redhat96-9.6-3.noarch.rpm
rpm -ivh pgdg-redhat96-9.6-3.noarch.rpm
yum install -y postgresql96-devel
yum install -y postgresql96-server

# По умолчанию в качестве системы обмена сообщениями используется RabbitMQ. Если необходимо использовать IbmMQ, установите клиентские пакеты MQSeries:
MQSeriesClient-9.0.4-0.x86_64.rpm
MQSeriesRuntime-9.0.4-0.x86_64.rpm
MQSeriesSDK-9.0.4-0.x86_64.rpm
#с помощью команды yum -y install <путь_к_файлу_пакета>
#(пакеты можно загрузить с ресурса http://www-01.ibm.com/support/docview.wss?uid=swg24042009)

# Установите зависимости, поставляемые с дистрибутивом. Следующая команда должна быть выполнена одной строкой:
yum -y install /var/lib/luna/current/extras/rpms/haspd-*.rpm
yum -y install /var/lib/luna/current/extras/rpms/tbb*.rpm
yum -y install /var/lib/luna/current/extras/rpms/aerospike*/*.rpm


# Получите менеджер пакетов для Python

yum -y install python36 python36-devel

python3.6 /var/lib/luna/current/extras/get-pip.py
```

3.2 Настройка сервиса HASP

Для лицензирования LUNA PLATFORM используется HASP сервис. Без лицензии будет невозможно запустить и использовать сервисы LUNA.

Примечание. Файл лицензии поставляется компанией VisionLabs отдельно по запросу.

```
# Добавьте сервис в автозагрузку и стартуйте сервис
systemctl daemon-reload
systemctl start aksusbd
systemctl enable aksusbd
```

Активировать лицензию можно одним из следующих способов:

1. Добавить файл лицензии через консоль

```
# Добавьте файл лицензии в систему. Файл должен находиться в директории /root. Команда найдёт файл и активирует лицензию
find /root/ -type f -name "Unlocked_*.v2c" -exec hasp_update u {} \;
# Перезапустите сервис
systemctl restart aksusbd

# Проверьте, что сервис работает
systemctl status aksusbd
```

2. Добавить файл лицензии вручную через пользовательский интерфейс (Рис. 1):
   - Перейдите по адресу: <host_address>:1947;
   - Выберите вкладку **Update/Attach** на панели слева;
   - Нажмите кнопку «Browse» и выберите файл лицензии в открывшемся окне;
   - Нажмите кнопку «Apply file».

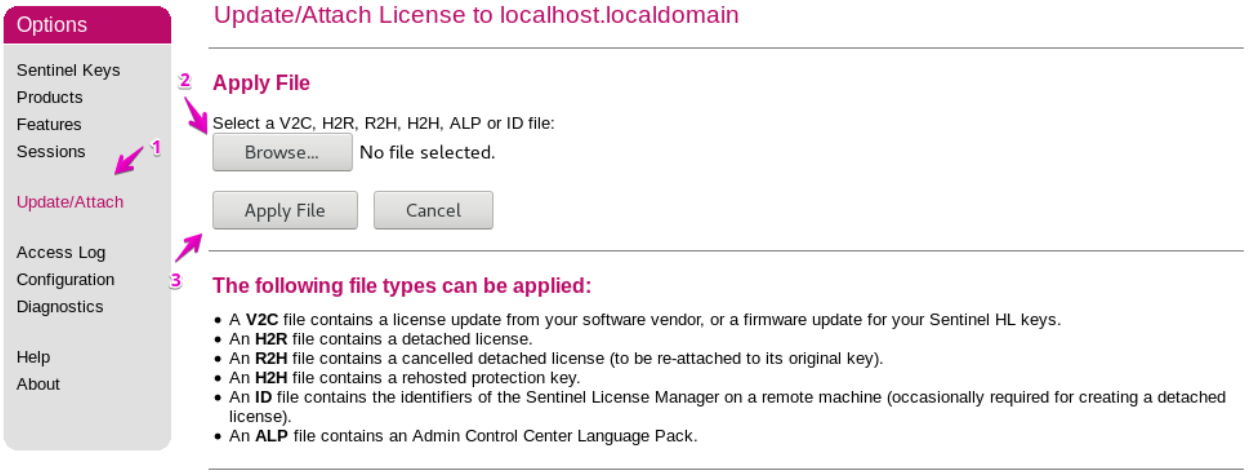

Рисунок 1 – Ручное добавление файла лицензии

3. Получить доступ к сетевой лицензии на сервере лицензирования (Рис. 2):
   - Перейдите по адресу: <host_address>:1947;
   - Выберите вкладку **Configuration** на панели слева;
   - Перейдите на вкладку **Access to Remote License Managers**;

- Введите адрес сервера лицензирования в поле «Remote License Search Parameters»;
- Нажмите кнопку «Submit».

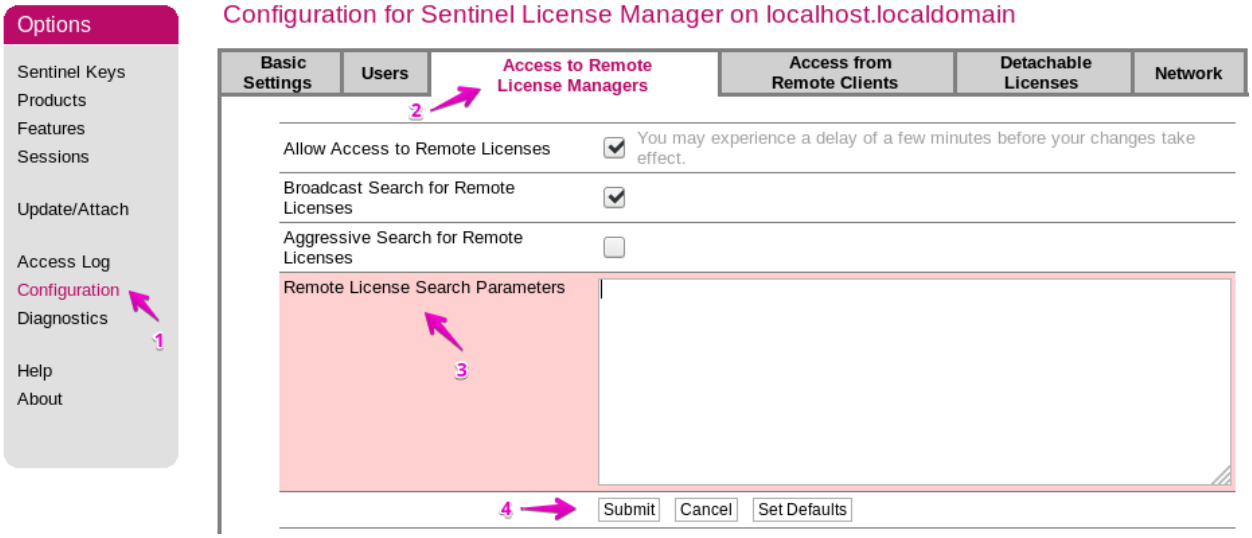

Рисунок 2 – Получение сетевого доступа к лицензии

Примечание. Для просмотра активных ключей нужно перейти на вкладку Sentinel Keys.

3.3 CORE

3.3.1 Настройка брокера сообщений RabbitMQ

RabbitMQ используется для обмена сообщениями между сервисами LUNA PLATFORM. RabbitMQ необходимо для установки других сервисов.

```
    # Добавьте сервис в автозагрузку и стартуйте сервис
    systemctl start rabbitmq-server
    systemctl enable rabbitmq-server

    # Создайте пользователя “luna” и задайте пароль “luna"
    rabbitmqctl add_user luna luna

    # Назначьте пользователю расширенные права
    # set_permissions [-p <vhostpath>] <user> <conf> <write> <read>
    rabbitmqctl set_permissions -p / luna ".*" ".*" ".*"

    # Присвойте пользователю статус администратора
    # set_user_tags <username> <tag>
    rabbitmqctl set_user_tags luna administrator

    # Активируйте управляющий модуль для того, чтобы вносить изменения в
конфигурацию брокера сообщений
    rabbitmq-plugins enable rabbitmq_management
    systemctl restart rabbitmq-server

    # Получите доступ к утилите rabbitmqadmin
    find /var/lib/rabbitmq/ -name 'rabbitmqadmin' -exec chmod 0777 {} \;
    # Добавьте путь к файлу rabbitmqadmin в переменную окружения, чтобы
система могла найти и выполнить команду rabbitmqadmin
    PATH="$PATH:/var/lib/rabbitmq/mnesia/rabbit@`hostname -s`-plugins-
expand/rabbitmq_management-`rpm -qa | grep rabbitmq | cut -d - -f
3`/priv/www/cli/"

    # Добавьте в конфигурацию брокера необходимые очереди для экстракции и
матчинга, а также службы обмена для коммуникации между компонентами
    rabbitmqadmin -u luna -p luna declare exchange name=luna.extract
type=direct
    rabbitmqadmin -u luna -p luna declare exchange name=luna.match
type=direct
    rabbitmqadmin -u luna -p luna declare exchange name=luna.index
type=direct
    rabbitmqadmin -u luna -p luna declare queue name=extractor durable=false
auto_delete=true
    rabbitmqadmin -u luna -p luna declare queue name=matcher durable=false
auto_delete=true
    rabbitmqadmin -u luna -p luna declare queue name=indexer durable=false
auto_delete=true
    rabbitmqadmin -u luna -p luna declare queue name=indexed_matcher
durable=false auto_delete=true

    # Выключаем модуль конфигурирования и перезагружаем сервис
    rabbitmq-plugins disable rabbitmq_management
    systemctl restart rabbitmq-server
```

3.3.2 Настройка СУБД AeroSpike

Настройте конфигурацию СУБД AeroSpike. Aerospike используется для хранения дескрипторов.

```
    # Удалите конфигурацию по умолчанию
    rm -f /etc/aerospike/aerospike.conf

    # Скопируйте подготовленную конфигурацию из комплекта поставки в
конфигурационный каталог aerospike
    cp -f /var/lib/luna/current/extras/aerospike.conf /etc/aerospike/

    # Запустите сервис и добавьте его в автозагрузку
    systemctl start aerospike
    systemctl enable aerospike
```

Примечание: Убедитесь, что сетевые порты по умолчанию доступны, либо проведите настройку в соответствии с документацией, доступной по ссылке

3.4 API

3.4.1 Настройка СУБД PostgreSQL/Oracle

В качестве БД LUNA API можно использовать PostgreSQL или Oracle. По умолчанию сервис настроен на работу с БД PostgreSQL.

Настроить используемую БД можно в файле *config.conf*, который расположен в папке */var/lib/luna/current/luna-api/luna_api/configs*.

Настройки для PostgreSQL, заданные по умолчанию:

```
DB = "postgres"
DB_USER_NAME = "faceis"
DB_PASSWORD = "faceis"
DB_NAME = "faceis_db"
DB_HOST = "127.0.0.1"
DB_PORT = 5432
```

Настройки для Oracle:

```
DB = "oracle"
DB_USER_NAME = "faceis"
DB_PASSWORD = "faceis"
DB_NAME = "XE"
DB_HOST = "127.0.0.1"
DB_PORT = 1521
```

3.4.1.1 Настройка СУБД Oracle

Установка СУБД Oracle описана в документации на сайте корпорации Oracle:

https://docs.oracle.com/en/database/oracle/oracle-database/index.html

Во все сервисы с именами *luna-*.service* в папке */etc/systemd/system/* для Oracle необходимо добавить переменную окружения NLS_LANG перед запуском сервисов. Это необходимо для корректной работы базы данных.

```
Environment=NLS_LANG=RUSSIAN
```

3.4.1.2 Настройка СУБД PostgreSQL

Примечание. Мы рекомендуем использовать версию PostgreSQL 9.6, минимально допустимая версия 9.2, но она не поддерживается командой postgres.

```
# Инициализируйте базу данных
/usr/pgsql-9.6/bin/postgresql96-setup initdb
```

После этого нужно провести настройку доступности базы данных. Для этого откройте файл /var/lib/pgsql/9.6/data/pg_hba.conf.

```
vim /var/lib/pgsql/9.6/data/pg_hba.conf
```

Примечание. Ниже приведён пример конфигурации.

В файле нужно изменить значения колонки «Method» как в примере ниже:

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD
# "local" is for Unix domain socket connections only
local   all             all                                     trust
# IPv4 local connections:
```

```
host    all             all             127.0.0.1/32            trust
# IPv6 local connections:
host    all             all             ::1/128                 trust
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local   replication     postgres                                peer
#host    replication     postgres        127.0.0.1/32            ident
```

```
# Запустите сервис и добавьте его в автозагрузку
systemctl start postgresql-9.6
systemctl enable postgresql-9.6
```

Создайте нового пользователя и базу данных. Для пользователя задаются логин, пароль и привилегии. По умолчанию принято создавать пользователя “faceis”.

```
# Войдите в интерфейс базы данных
psql -U postgres
# Создайте пользователя БД
create role faceis;

# Назначьте для пользователя пароль
ALTER USER faceis WITH PASSWORD 'faceis';

# Создайте базу данных
CREATE DATABASE faceis_db;

# Назначьте права для пользователя, созданного ранее
GRANT ALL PRIVILEGES ON DATABASE faceis_db TO faceis;

# Дайте возможность пользователю авторизоваться в БД
ALTER ROLE faceis WITH LOGIN;
# Выйдите
\q
```

3.4.2 Установка зависимостей для LUNA API

LUNA API предоставляет пользователям ограниченный доступ к системным ресурсам.

```
# Перейдите в директорию модуля
cd /var/lib/luna/current/luna-api

 # Создайте виртуальное окружение
python3.6 -m venv venv

 # Активируйте виртуальное окружение
source venv/bin/activate

# Запустите установку зависимостей Python
pip3.6 install -r requirements.txt

# Запустите наполнение базы данных структурой
python3.6 ./base_scripts/db_create.py

# Деактивируйте виртуальную среду
deactivate
```

3.4.3 Сервис Faces

Сервис Faces хранит лица, персоны и списки.

3.4.3.1 Настройка СУБД

```
# Войдите в интерфейс базы данных
psql -U postgres
# Создайте пользователя БД
create role luna;

# Назначьте для пользователя пароль
ALTER USER luna WITH PASSWORD 'luna';

# Создайте базу данных
CREATE DATABASE luna_faces;

# Назначьте права для пользователя, созданного ранее
GRANT ALL PRIVILEGES ON DATABASE luna_faces TO luna;

# Дайте возможность пользователю авторизоваться в БД
ALTER ROLE luna WITH LOGIN;
# Выйдите
\q
```

3.4.3.2 Установка

Сервис используется для хранения лиц и списков.

```
# Перейдите в директорию модуля
cd /var/lib/luna/current/luna-faces

 # Создайте виртуальное окружение
python3.6 -m venv venv

 # Активируйте виртуальное окружение
source venv/bin/activate

# Запустите установку зависимостей Python
pip3.6 install -r requirements.txt

# Запустите наполнение базы данных структурой
python3.6 ./base_scripts/db_create.py

# Деактивируйте виртуальную среду
deactivate
```

3.4.4 Сервис Image Store

Для создания дескрипторов используются специальные нормализованные изображения лиц, полученные из фотографий. Нормализованные изображения хранятся в IMAGE STORE и используются при обновлении нейронной сети для повторного извлечения дескрипторов.

3.4.4.1 Настройка портретного хранилища

Изображения могут храниться на жёстком диске или в Image Store.

Для отправки портретов LUNA в хранилище Image Store необходимо изменить параметр SEND_TO_LUNA_IMAGE_STORE в конфигурационном файле luna-api/luna_api/configs/config.conf.

```
SEND_TO_LUNA_IMAGE_STORE = 1 #: flag, which indicates whether send portraits to LUNA Image Store or not
```

Портреты можно сохранять не только в Image Store, но и на жёстком диске. Для хранения портретов на жестком диске установите флаг ENABLE_PLUGINS в файле luna-api/luna_api/configs/config.conf.

```
ENABLE_PLUGINS = 1 #: flag to enable plug-ins
```

3.4.4.2 Установка

```
# Перейдите в директорию модуля
cd /var/lib/luna/current/luna-image-store

 # Создайте виртуальное окружение
python3.6 -m venv venv

 # Активируйте виртуальное окружение
source venv/bin/activate

# Запустите установку зависимостей Python
pip3.6 install -r requirements.txt

# Деактивируйте виртуальное окружение
deactivate
```

3.4.4.3 Запуск Image Store

```
# Запустите и активируйте сервис
systemctl start luna-image-store
systemctl enable luna-image-store
```

3.4.4.4 Создание базы данных

Примечание. Сервис Image Store должен быть запущен перед созданием базы данных.

```
# Перейдите в директорию модуля API
cd /var/lib/luna/current/luna-api

# Активируйте виртуальное окружение
source venv/bin/activate
```

```
# Инициируйте создание хранилища Image Store
python3.6 ./base_scripts/lis_bucket_create.py

# Деактивируйте виртуальное окружение
deactivate
```

3.5 Запуск сервисов LUNA PLATFORM

```
    # Запустите все вервисы LUNA PLATFORM и добавьте их в автозагрузку

    systemctl start luna-faces
    systemctl enable luna-faces


    systemctl start luna-broker
    systemctl enable luna-broker

    systemctl start luna-api
    systemctl enable luna-api


    systemctl start luna-extractor@1
    systemctl enable luna-extractor@1
    systemctl start luna-matcher@1
    systemctl enable luna-matcher@1

    # Каждый из экземпляров Extractor и Matcher нужно запускать отдельно,
указав номер экземпляра после символа “@”
    # Пример:
    # systemctl start luna-extractor@1
    # systemctl enable luna-extractor@1
    # systemctl start luna-extractor@2
    # systemctl enable luna-extractor@2
```

4 Дополнительные сервисы

Дополнительные сервисы включают сервисы построения индекса и поиска по индексу, пользовательский интерфейс, средства работы со статистикой и инструменты администратора. Они не обязательны к установке.

Для установки всех сервисов требуются python3.6. Для их установки выполните следующие команды:

```
yum -y --nogpgcheck install python36 python36-setuptools
```

Перед началом установки дополнительных сервисов проверьте, что CORE и API работают.

```
systemctl status luna-extractor@1 luna-matcher@1 luna-broker luna-api
luna-faces luna-image-store
```

В случае, если Вы устанавливаете дополнительные сервисы на отдельных серверах или виртуальных машинах, необходимо выполнить следующие действия:

Скопировать дистрибутив на сервер/виртуальную машину, распаковать и создать символьную ссылку на папку. Данные процессы описаны в разделе «Подготовка и распаковка дистрибутива».

Кроме того, может потребоваться обновление CentOs, установка epel-release и установка зависимостей. Они описаны в главе «Установка зависимостей и программного обеспечения».

Также необходимо скопировать все **необходимые** сервисные файлы из комплекта поставки в директорию /etc/system/system/. Данный процесс описан в разделе «Сервисные файлы».

4.1 Сервисы построения индекса и поиска по индексу

Перечисленные сервисы позволяют индексировать дескрипторы в списках и проводить поиск по сформированному индексу:

- Indexer создаёт индекс на основе списка дескрипторов;
- Indexed Matcher проводит поиск в индексах;
- Matcher Daemon:
  - отвечает за копирование индекса с сервера, на котором установлен Indexer, на сервер поиска по индексу (Index Matcher),
  - перезапускает Indexed Matcher с новым индексом;
- Index Manager формирует задачи на построение индекса и координирует процесс доставки индекса на сервера Index Matcher;

Поиск по индексированным дескрипторам проводится в разы быстрее.

Перед поиском в индексированном списке Indexed Matcher получает из сервиса Faces информацию о недавно добавленных в список дескрипторах. Если новые дескрипторы были добавлены, Indexed Matcher получает их из БД. Таким образом поиск проводится по индексированному списку и по всем недавно добавленным дескрипторам.

Примечание: Index Matcher и Matcher Daemon необходимо устанавливать вместе на один хост;

Indexer должен быть установлен на отдельном сервере.

4.1.1 Установка зависимостей Matcher Daemon

Сервер с Matcher Daemon и Indexed Matcher

Зависимости требуются для установки Matcher Daemon.

```
# Установите Python и Python-setuptools
yum -y --nogpgcheck install python36 python36-setuptools
# Установите pip
python3.6 /var/lib/luna/current/extras/get-pip.py
# Установите зависимость из комплекта поставки
yum -y install /var/lib/luna/current/extras/rpms/matcher-daemon-*.rpm
```

4.1.2 Настройка Matcher Daemon

Нужно настроить доступ Matcher Daemon к серверу с Indexer и дать Matcher Daemon права на запуск Indexed Matcher. Matcher Daemon должен быть установлен на одном сервере с Indexed Matcher.

Сервер с Indexer

Примечание. Необходимо настроить авторизацию по публичному ключу на сервере с Index. Для этой цели следует отредактировать файл /etc/ssh/sshd и перезапустить сервис: systemctl restart sshd. Эта процедура не описана в данной инструкции.

```
# На сервере с Indexer следует создать пользователя matcher-daemon
adduser matcher-daemon
```

Сервер с Matcher Daemon и Indexed Matcher

```
# На этом шаге необходимо сгенерировать ssh ключ для предоставления
доступа Matcher Daemon к серверу с Indexer
# Смените пользователя на matcher-daemon
su matcher-daemon
# Сгенерируйте ssh-ключ
ssh-keygen
# Система запросит ввести путь к ключу. Нажмите Enter, чтобы система
создала ключ в директории по умолчанию /home/<current_user>/.ssh/id_rsa.
# Система запросит ввести ключевую фразу, но это необязательно. Нажмите
Enter три раза.

# Задайте адрес сервера, на котором установлен LUNA Indexer
ssh-keyscan -H <indexer_host> >> ~/.ssh/known_hosts
# Далее необходимо скопировать файл публичного ключа на сервер с Indexer
вручную или командой:
scp /home/matcher-daemon/.ssh/id_rsa.pub matcher-
daemon@<indexer_host>:<Directory_for_key>
```

Примечание. Необходимо создавать ключ и в случае, когда все сервисы установлены на одном сервере и в случае, когда сервисы распределены по разным серверам.

Сервер с Indexer

```
# На этом шаге необходимо скопировать публичную часть ключа в файл /home/matcher-daemon/.ssh/authorized_keys

# Создайте пользователя matcher-daemon:
adduser matcher-daemon
# Смените пользователя на matcher-daemon
su matcher-daemon

# Создайте папку
mkdir /home/matcher-daemon/.ssh
# Задайте права доступа для папки
chmod 700 /home/matcher-daemon/.ssh

# Добавьте содержимое файла «id_rsa.pub» в файл /home/matcher-daemon/.ssh/authorized_keys. Его можно скопировать вручную или с помощью команды:
cat <path_to_file_id_rsa.pub> >> /home/matcher-daemon/.ssh/authorized_keys
# Установите права доступа к папке
chmod 600 /home/matcher-daemon/.ssh/authorized_keys
```

Сервер с Matcher Daemon и Indexed Matcher

```
# Добавьте пользователю matcher-daemon разрешение запускать/останавливать Indexed Matcher без ввода пароля

# Переключитесь на пользователя root
sudo su

# Откройте файл в редакторе
visudo

# после следующих строк:

## Allow root to run any commands anywhere
root    ALL=(ALL)       ALL

# Добавьте строку:
matcher-daemon ALL=(ALL) NOPASSWD: /bin/systemctl start luna-indexed-matcher, /bin/systemctl stop luna-indexed-matcher, /bin/systemctl restart luna-indexed-matcher
```

```
# Отредактируйте конфигурационный файл matcher-daemon:
vim /etc/matcher-daemon/matcher-daemon-config.json
# В указанных ниже полях пропишите в файле адрес сервера с Indexer, путь к папке с индексом и путь к ключу (приведён пример, который используется по умолчанию):

  "indexer_host": "127.0.0.1",
  "Index_holding_dir": "/var/lib/luna/index",
  "ssh_key": "/home/matcher-daemon/.ssh/id_rsa"
```

"indexer_host" содержит адрес сервера Index.

“Index_holding_dir" содержит директорию, в которой должен храниться индекс. Одна и та же директория должна быть указана здесь и в конфигурационном файле Indexer: */var/lib/luna/current/conf/indexer.conf.* По умолчанию задаётся директория “/var/lib/luna/index”.

"ssh_key" содержит путь до файла ключа id_rsa.

```
    # Директорию для индекса необходимо создать вручную и задать права
доступа для неё
    mkdir /var/lib/luna/index
    chmod 700 /var/lib/luna/index
```

4.1.3 Настройка БД для LUNA Index Manager

```
    # Войдите в интерфейс базы данных
    psql -U postgres

    # Создайте пользователя БД
    create role luna;

    # Назначьте для пользователя пароль
    ALTER USER luna WITH PASSWORD 'luna';

    # Создайте базу данных
    CREATE DATABASE luna_index_manager;

    # Назначьте права для пользователя, созданного ранее
    GRANT ALL PRIVILEGES ON DATABASE luna_index_manager TO luna;

    # Дайте возможность пользователю авторизоваться в БД
    ALTER ROLE luna WITH LOGIN;

    # Выйдите
    \q
```

4.1.4 Установка LUNA Index Manager

Измените порт для Matcher Daemon в конфигурационном файле : */var/lib/luna/current/luna-index-manager/luna_index_manager/configs/config.conf*. Должен быть указан порт 6001.

```
    #: luna-matcher_daemons
    LUNA_MATCHER_DAEMONS = ["http://127.0.0.1:6001/1"]    #: list of luna-
matcher-daemon endpoint
```

Убедитесь, что остальные пути указаны корректно.

```
    # Перейдите в директорию модуля
    cd /var/lib/luna/current/luna-index-manager
     # Создайте виртуальное окружение
    python3.6 -m venv venv
     # Активируйте виртуальное окружение
    source venv/bin/activate
    # Запустите установку зависимостей Python
    pip3.6 install -r requirements.txt

    # Запустите наполнение базы данных структурой
    python3.6 ./base_scripts/db_create.py

    # Деактивируйте виртуальную среду
    deactivate
```

4.1.5 Запуск сервисов построения индекса и поиска по индексу

Запустите сервисы.

```
systemctl start luna-indexer
systemctl enable luna-indexer
systemctl start matcher-daemon
systemctl enable matcher-daemon
systemctl start luna-index-manager
systemctl enable luna-index-manager

# Проверьте статусы всех сервисов
systemctl status luna-indexer matcher-daemon luna-index-manager
```

4.2 Сервис User Interface

Сервис UI предоставляет графический интерфейс для пользователей LUNA PLATFORM.

4.2.1 Установка

Примечание. Настоятельно рекомендуется создать виртуальную среду для установки зависимостей Python.

Установите зависимости.

```
# Установите python, если он ещё не установлен
yum -y install python36 python36-devel

 # Перейдите в директорию модуля
cd /var/lib/luna/current/luna-ui2

 # Создайте виртуальное окружение
python3.6 -m venv venv

 # Активируйте виртуальное окружение
source venv/bin/activate
 # Установите зависимости из файла
pip3.6 install -r requirements.txt
```

4.2.2 Настройка

Сгенерируйте ключ, чтобы обеспечить безопасные сеансы работы.

```
# Перейдите в директорию со скриптом
cd /var/lib/luna/current/luna-ui2/app/scripts

# Сгенерируйте ключ
python3.6 generate_secret_key.py
# Деактивируйте виртуальную среду
deactivate
```

Скопируйте ключ и добавьте его в файл */var/lib/luna/current/luna-ui2/app/config.ini*

```
vim /var/lib/luna/current/luna-ui2/app/config.ini
```

в поле:

```
COOKIE_SECRET = <secret_key>
```

Задайте правильный IP адрес сервиса API:

```
LUNA_API_URI = http://<correct_address>:5000/4/
```

4.2.3 Запуск сервиса User Interface

Если все предыдущие шаги выполнены без ошибок, то сервер готов к работе.

```
# Запустите сервис
systemctl start luna-ui.service
systemctl enable luna-ui.service
```

```
# Проверьте статус сервиса
systemctl status luna-ui.service
```

По умолчанию UI использует порт 80. Для перехода в UI необходимо ввести http://<correct_address>:80

4.3 Сервис Administration Panel

Сервис панели администратора предоставляет графический интерфейс для административных задач. Также сервис может взаимодействовать с Grafana (стороннее приложение) для просмотра системных метрик.

Перед началом установки, убедитесь, что LUNA Core и LUNA API корректно установлены и функционируют.

4.3.1 Системные требования

Для установки сервиса панели администратора необходимо следующее ПО:

- Пакет разработчика PostgeSQL;
- Экземпляр СУБД PostgeSQL;
- Python 3.6;
- Пакет Python setuptools.

4.3.2 Создание базы данных

Создайте базу данных для Сервиса панели администратора.

```
# Войдите в интерфейс базы данных
psql -U postgres;
# Создайте пользователя базы данных
create role faceis;
# Задайте пароль для пользователя
ALTER USER faceis WITH PASSWORD 'faceis';
# Создайте базу данных
CREATE DATABASE admin_faceis_db;
# Назначьте права пользователю
GRANT ALL PRIVILEGES ON DATABASE admin_faceis_db TO faceis;
# Дайте пользователю разрешение на авторизацию в БД
ALTER ROLE faceis WITH LOGIN;
# Выйдите из интерфейса
\q
```

4.3.3 Установка

Установите зависимости:

Примечание. Для установки настойчиво рекомендуется использование виртуальной среды.

```
# Установите python, если он ещё не установлен
yum -y install python36 python36-devel
#Перейдите в директорию модуля
cd /var/lib/luna/current/luna-admin

# Создайте виртуальное окружение
python3.6 -m venv venv

# Активируйте виртуальное окружение
source venv/bin/activate
```

```
#Установка зависимостей
pip3.6 install -r requirements.txt

# Инициализируйте наполнение структуры базы данных
python3.6 bases_scripts/db_create.py


# Деактивируйте виртуальную среду
deactivate
```

Примечание. db_create.py также создает скрипты для последующей миграции БД (db_repository). Без этих скриптов db_migrate.py не может быть выполнен.

4.3.4 Настройка

Все настройки содержатся в файле configs/config.py. Файл может быть изменен в соответствии с вашими потребностями.

Примечании: Все настройки, кроме настроек сборщиков мусора, ***обязательно должны быть*** *синхронизированы с настройками LUNA API.*

4.3.5 Запуск сервиса Administration Panel

Запуск сервиса:

```
systemctl enable luna-admin_back
systemctl start luna-admin_back
systemctl enable luna-admin_tasks
systemctl start luna-admin_tasks

# Проверьте статусы сервисов
systemctl status luna-admin_back luna-admin_tasks
```

4.3.6 Создание дашборда Grafana

Для создания дашборда:

```
    #Перейдите в директорию модуля
    cd /var/lib/luna/current/luna-admin
    # Запустите скрипт
    python3.6 ./bases_scripts/create_grafana_dashboards.py --
config=./configs/config.conf
```

Примечание. Создание дашборда не является обязательным.

4.4 Сервис Event & Statistic

Сервис **Event & Statistic** реализует RESTful API и позволяет отслеживать все запросы, которые обрабатывает LUNA.

В состав Event & Statistic включены два сервиса:

- Сервис событий LUNA API Python Server Events;
- Сервис управления статистикой Statistic Manager.

4.4.1 Системные требования

Поддерживаются следующие операционные системы:

- RedHat Linux 7;
- CentOS Linux 7.

Требуется установка следующих сторонних зависимостей:

- Python 3.6;
- Redis v. 3.2.3 или выше;
- InfluxDB v. 1.3.2 или выше;
- Дополнительные зависимости библиотек, перечисленные в **requirements.txt**.

4.4.2 Установка пакетов и требований

Для установки пакетов и требований необходимо:

- Установить Python 3.6;
- Установить InfluxDB;
- Установить Redis;
- Установить зависимости;
- Настроить базу данных.

4.4.2.1 Установка и запуск InfluxDB

InfluxDB используется для хранения данных о выполнении пользовательских запросов с привязкой ко времени. Это позволяет получать статистику за указанный период.

```
    # Загрузите и установите .rpm пакет
```

```
    yum -y install https://dl.influxdata.com/influxdb/releases/influxdb-
1.3.2.x86_64.rpm
    # Включите автозагрузку
    systemctl enable influxdb.service
    # Стартуйте сервис
    systemctl start influxdb
```

4.4.2.2 Установка Redis

БД Redis используется для временного хранения данных и создания событий, совпадающих с ответами LUNA на пользовательский запрос. События доставляются в режиме реального времени через web-сокеты.

Для установки последовательно выполните следующие команды:

```
    # Установите Redis
    yum -y install redis

    # Включите автозагрузку
    systemctl enable redis.service

    # Стартуйте сервис
    systemctl start redis
```

4.4.2.3 Установка библиотеки Python

Примечание. Настоятельно рекомендуется установка виртуальной среды.
Для установки последовательно выполните следующие команды:

```
    # Установите python, если он ещё не установлен
    yum -y install python36 python36-devel
    # Перейдите в папку с проектом
    cd /var/lib/luna/current/luna-stat-server

    # Создайте виртуальное окружение
    python3.6 -m venv venv

    # Активируйте виртуальное окружение.
    source venv/bin/activate

    # Установите зависимости из файла
    pip3.6 install -r requirements.txt


    # Деактивируйте виртуальную среду
    deactivate
```

4.4.3 Настройка

Проверьте и, в случае необходимости, измените следующие настройки в конфигурационном файле /var/lib/luna/current/luna-stat-server/config.conf и проверьте актуальность указанных портов для баз данных:

```
    INFLUX_LOGIN="root"
    INFLUX_PASSWORD="root"
    INFLUX_DATABASE="stat_service"
    INFLUX_URL="localhost:8086"

    REDIS_LOGIN=""
```

```
REDIS_PASSWORD=""
REDIS_DATABASE="0"
REDIS_URL="localhost:6379"

LPS_URL="localhost:5000"
LPS_API_VERSION=3

IGNORE_AGE_TAG=0
IGNORE_FACE_SCORE_TAG=0
IGNORE_GENDER_TAG=0
IGNORE_GLASSES_TAG=0
IGNORE_SIMILARITY_TAG=0

COOKIE_SECRET="very secret keyword 21c54-nx3xr3mx3em-0uy9nryn1370"
```

После этого оба компонента сервиса Event & Statistic могут быть запущены.

4.4.4 Создание базы данных

Создайте базу данных.

```
# Перейдите в папку с проектом
cd /var/lib/luna/current/luna-stat-server
# Активируйте виртуальное окружение.
source venv/bin/activate
# Создайте базу данных
python3.6 db_create.py
# Деактивируйте виртуальную среду
deactivate
```

4.4.5 Запуск сервиса Event & Statistic

Запустите сервис **Events**:

```
systemctl start luna-stat-lpse.service
systemctl enable luna-stat-lpse.service
```

Запустите сервис **Statistic Manager**:

```
systemctl start luna-stat-sm.service
systemctl enable luna-stat-sm.service
```

Сервис получает события от LUNA API по порту 5009. Подписка и статистика доступны по 5008 порту. Вы можете использовать любой свободный порт.

Примечание. Можно запустить несколько экземпляров обоих сервисов.

```
# Проверьте статусы сервисов
systemctl status luna-stat-lpse.service luna-stat-sm.service
```

5 Настройка LUNA API для отправки статистики в установленные сервисы

Для отправки статистики администраторов необходимо (подробнее установка и конфигурация сервиса панели администратора описаны в **п.** 4.3 данного документа):

- создать БД в InfluxDB;

```
# Перейдите в папку с проектом
cd /var/lib/luna/current/luna-api

# Активируйте виртуальную среду
source venv/bin/activate

# Запустите тест
python3.6 ./base_scripts/influx_db_create.py

# Деактивируйте виртуальную среду
deactivate
```

- изменить параметры SEND_ADMIN_STATS, ADMIN_STATISTICS_SERVER, ADMIN_STATISTICS_DB в конфигурационном файле **luna-api/luna_api/configs/config.conf.**

```
SEND_ADMIN_STATS = 1
ADMIN_STATISTICS_SERVER_ORIGIN = "http://<influxdb-ip>:<influxdb-port>"
ADMIN_STATISTICS_DB = "luna_api_admin"
```

Для отправки статистики событий в сервис **Event & Statistic** необходимо изменить параметры SEND_ACCOUNT_STATS и ACCOUNTS_STATISTICS_SERVER в конфигурационном файле var/lib/luna/current/luna-api/luna_api/configs/config.conf.

```
SEND_ACCOUNT_STATS = 1
ACCOUNTS_STATISTICS_SERVER = "http://<influxdb-ip>:<influxdb-port>/internal/lps_event"
```

6 Тестирование LUNA PLATFORM

Тестирование проводится для всех компонентов LUNA PLATFORM, кроме сервисов пользовательского интерфейса и Event & Statistic.

Для проведения тестирования выполните следующие шаги:

1. Убедитесь, что все сервисы запущены:

```
systemctl status luna-extractor@1 luna-matcher@1 luna-broker luna-api luna-faces luna-image-store
```

Примечание. Необходимо проверить все экземпляры сервисов luna-extractor и luna-matcher.

2. Проверьте и, если необходимо, измените настройки сервиса в конфигурационном файле /var/lib/luna/current/luna-api/tests/config.py :

```
    LUNA_API_URL="<Luna API url>/<Luna API version>/";
    SS_BASE_URL="<Statistic Manager service URL>:<Statistic Manager service port>";
```

3. Запустите тестирование. Если сервисы работают корректно, будет выведено сообщение “Ok”:

```
    # Перейдите в папку с проектом
    cd /var/lib/luna/current/luna-api
```

```
    # Активируйте виртуальное окружение
    source venv/bin/activate


    # Запустите тестирование
    python3 -m unittest tests.unittests_main
```

```
    # Деактивируйте виртуальную среду
    deactivate
```

Приложение. История изменений

Дата	Версия	Описание
22.05.17	1	Первая версия документа
23.08.17	2	Добавлены инструкции для модулей UI и панели администрирования
24.08.17	3	Добавлены инструкции для модуля events & statistics
06.07.18	4	Добавлены инструкции для сервисов Indexer, Index Matcher, Matcher Daemon, Index Manager и Faces
16.08.18	5	Документ был обновлён
12.09.18	6	Добавлены: раздел Глоссарий, раздел Общие сведения, информация о СУБД Oracle