VisionLabs
MACHINES CAN SEE

VisionLabs LUNA PLATFORM

Installation manual

VisionLabs B.V.

Keizersgracht 311, 1016 EE, Amsterdam, the Netherlands

+31 20 369 04 93

info@visionlabs.ai

www.visionlabs.ai

Table of contents

Glossary

Term	Abbreviation
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	Administration Panel
LUNA PLATFORM User Interface	User Interface, UI
LUNA PLATFORM Event & Statistic	Event & Statistic
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

Default Ports for Services

Service name	Port
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

Introduction

This manual describes all necessary procedures for a full installation of LUNA PLATFORM software package and additional services.

General Information

The distribution package includes general services and additional services.

General services and their components are:

- HASP service,
- CORE:
  - DBMS Aerospike,
  - Broker service,
  - RabbitMQ,
  - Extractor worker service,
  - Matcher worker service.
- API:
  - DBMS PostgreSQL or DBMS Oracle,
  - Faces service,
  - Image Store service.

Additional services and their components are:

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

Additional services installation is not mandatory.

A license file is required for LUNA PLATFORM activation. The file is provided by VisionLabs separately upon request.

The installation should be performed in the order specified in the document.

All actions described in this manual must be performed by a user with administrator permissions or by the **root** user.

It is recommended to read and understand SystemOverview.pdf. It will help you to find out what components LUNA PLATFORM consists of and what tasks they solve.

System requirements

The following system requirements should be met for the LUNA PLATFORM software package installation:

- OS CentOS 7.4 x86_64;
- CPU Intel, 4 physical cores minimum with clock frequency 2.0 GHz or higher;
- Instructions support:
    - Minimum requirements: SSE 4.2 instruction set support;
    - Recommended requirements: AVX2 instruction set support;
- RAM DDR3 (DDR4 recommended), 8 Gb or higher;
- Access to the Internet (for dependencies installation);
- Free storage 10 Gb or higher. SSD usage recommended.

The configuration above guarantees software package minimum power operating and cannot be used for production system. System requirements for production system are calculated based on intended system load.

Conventions

Comments and code explanations start with # and are in gray color.

1 Selinux and Firewall

You should configure Selinux and Firewall so that they do not block LUNA PLATFORM services.

Note. Selinux and Firewall configuration is not described in this guide.

2 Distribution Unpacking and Preparation

The distribution package is an archive luna_v.X.Y.ZZ, where X.Y.ZZ is numerical identifier, describing current LUNA PLATFORM version.

The archive includes all components, required for installation and exploitation. It does not include dependencies from standard CentOS 7.4 x86_64 repository. The dependencies are available in the Internet.

Move installation file and license file to /root/ directory before installation. The directory should not contain any other distribution or license files except the target ones.

```
# Create directory for distribution file unpacking
mkdir -p /var/lib/luna
# Move the distribution to the created directory
mv /luna* /var/lib/luna
# Install the unzip archiver, if it is necessary
yum install -y unzip
# Go to the folder with distribution
cd /var/lib/luna
# Unzip files
unzip luna*.zip
```

Create a symbolic link. Specify the current product version instead of X.Y.ZZ. The link indicates that the current version of the distribution file is used to run the software package.

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

If there is no /var/lib/luna/current link, launching scripts will not be able to determine location of the current release binary files. Further installation is also impossible.

```
# Return to the root catalog
cd
```

2.1 Service Files

Each service has a unit-file in the distribution package. You should copy the files in the system folder to be able to start services after their installation.

```
# Copy service files from the distribution archive to the system
directory
cp /var/lib/luna/current/extras/systemd/luna-*.service
/etc/systemd/system/
# Reload system services
systemctl daemon-reload
```

3 General Services

3.1 Software and Dependencies Installation

On this step you should install the extended CentOS repository (epel-release) and all required dependencies from it. Moreover, you should install dependencies for the LUNA PLATFORM from the distribution package and Python dependencies.

Note. Make sure that you have administrator rights and the Internet is connected.

```
# Switch to the root user
sudo su

# Update system packages to the latest versions
yum -y --nogpgcheck update

# Install epel-release for access to extended package repository (necessary for RabbitMQ and other dependencies)
yum -y --nogpgcheck install epel-release

# Install all necessary dependencies from CentOS repository
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
# Install dependencies for Postgresql. You should learn about the actual version from 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

#By default RabbitMQ is used for messages transferring. If you are going to use IbmMQ, you need to install the following client MQSeries packages:
MQSeriesClient-9.0.4-0.x86_64.rpm
MQSeriesRuntime-9.0.4-0.x86_64.rpm
MQSeriesSDK-9.0.4-0.x86_64.rpm
#using command:  yum -y install <path_to_package_file>
#(Packages can be downloaded from http://www-01.ibm.com/support/docview.wss?uid=swg24042009)

# Install dependencies from distribution archive. The command below should be typed in one line:
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


# Receive package manager for Python
yum -y install python36 python36-devel
python3.6 /var/lib/luna/current/extras/get-pip.py
```

3.2 HASP Service Configuration

The HASP service is used for LUNA PLATFORM licensing. Without a license you will be unable to run and use LUNA services.

Note. A *license file is provided by VisionLabs separately upon request.*

```
# Add service to autoload and start service
systemctl daemon-reload
systemctl start aksusbd
systemctl enable aksusbd
```

You can activate a license in one of the following ways:

1. Add a license file using the console.

```
# Add a license file to the system. The file should be in the /root
directory. The command will find the file and activate the license

find /root/ -type f -name "Unlocked_*.v2c" -exec hasp_update u {} \;

# Restart the service
systemctl restart aksusbd

# Check the service workability
systemctl status aksusbd
```

2. Add a license file manually using user interface (Fig. 1):
   - Go to: <host_address>:1947;
   - Select the **Update/Attach** at the left pane;
   - Press the “Browse” button and select a license file in the appeared window;
   - Press the “Apply file” button.

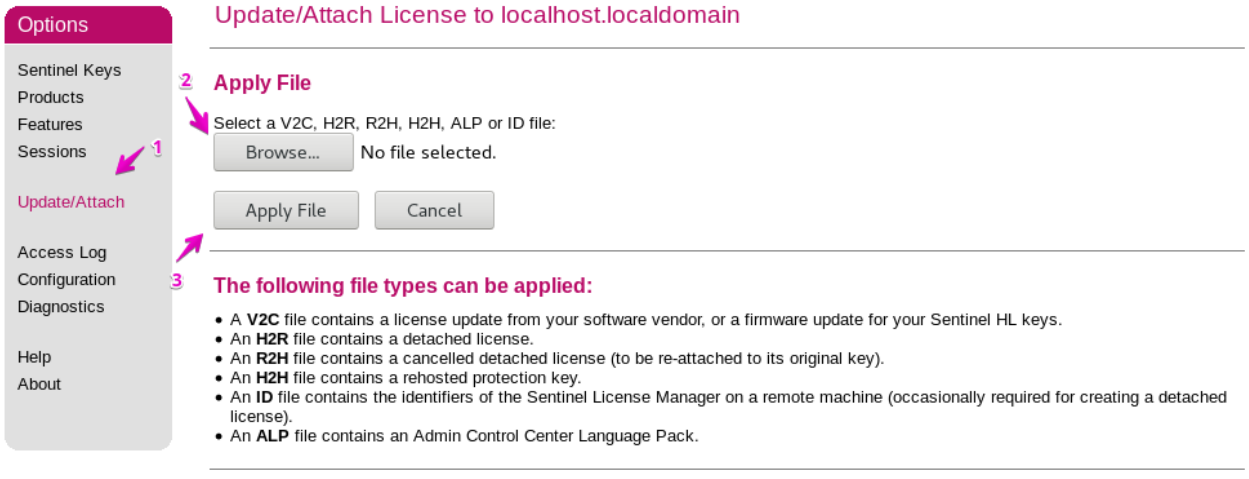

Figure 1 – License file is added manually

3. Gain access to a network license on the license server (Fig. 2):
   - Go to: <host_address>:1947;
   - Select the **Configuration** at the left pane;
   - Go to **Access to Remote License Managers** tab;
   - Enter address of the license server in the “Remote License Search Parameters” field;

- Press the «Submit» button.

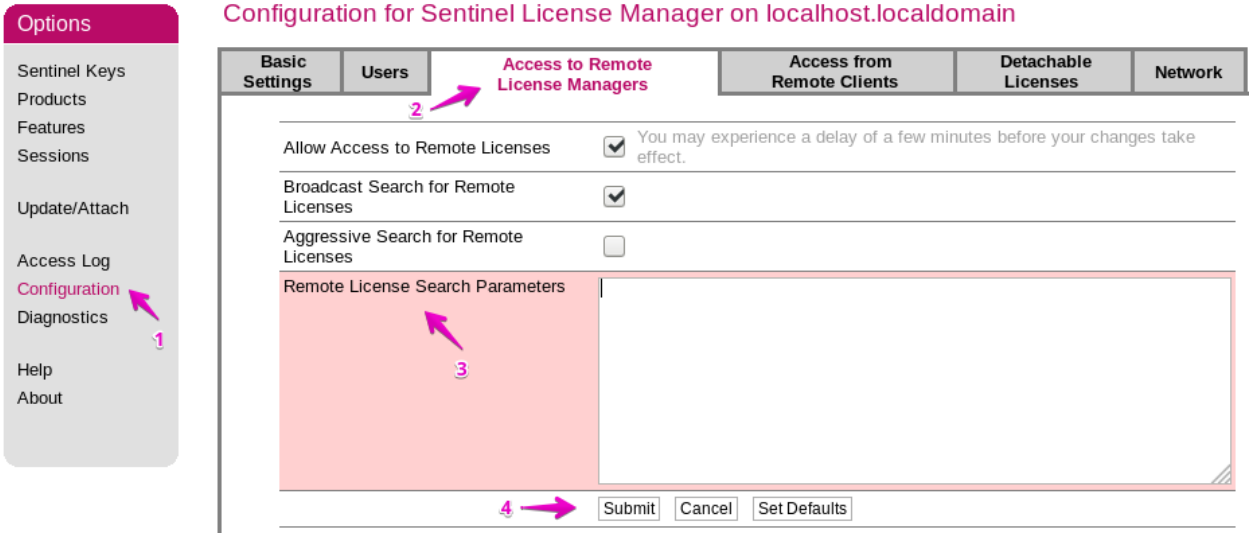

Figure 2 – Access to the network license

Note. The Sentinel Keys tab shows activated keys.

3.3 CORE

3.3.1 RabbitMQ Message Broker Configuration

RabbitMQ is used for messaging between LUNA PLATFORM services. RabbitMQ is required for installation of other services.

```
    # Add a service to autoload and start service
    systemctl start rabbitmq-server
    systemctl enable rabbitmq-server

    # Create a user “luna” with a password “luna”
    rabbitmqctl add_user luna luna

    # Assign extended rights to the user
    # set_permissions [-p <vhostpath>] <user> <conf> <write> <read>
    rabbitmqctl set_permissions -p / luna ".*" ".*" ".*"

    # Set the user as administrator
    # set_user_tags <username> <tag>
    rabbitmqctl set_user_tags luna administrator

    # Activate control module for changing message broker configuration
    rabbitmq-plugins enable rabbitmq_management
    systemctl restart rabbitmq-server

    # Get access to the utility rabbitmqadmin
    find /var/lib/rabbitmq/ -name 'rabbitmqadmin' -exec chmod 0777 {} \;
    # Add a path to the rabbitmqadmin file to the environment variable so
that the system could find and execute the rabbitmqadmin command
    PATH="$PATH:/var/lib/rabbitmq/mnesia/rabbit@`hostname -s`-plugins-
expand/rabbitmq_management-`rpm -qa | grep rabbitmq | cut -d - -f
3`/priv/www/cli/"

    # Create exchanges for service communication and add queues for
extraction and matching requests
    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

    # Turn off the configuration module and restart the service
    rabbitmq-plugins disable rabbitmq_management
    systemctl restart rabbitmq-server
```

3.3.2 AeroSpike DBMS Configuration

Configurate the AeroSpike DBMS. Aerospike is used to store descriptors.

```
    # Remove the default configuration
    rm -f /etc/aerospike/aerospike.conf
    # Copy the prepared configuration from the distributive archive to the
AeroSpike configuration catalogue
    cp -f /var/lib/luna/current/extras/aerospike.conf /etc/aerospike/

    # Launch the service and add it to autoload
    systemctl start aerospike
    systemctl enable aerospike
```

Note: Make sure default network ports are up and reachable, or adjust configuration as desired according to this manual https://www.aerospike.com/docs/operations/configure/network

3.4 API

3.4.1 PostgreSQL/Oracle DBMS

You can use PostgreSQL or Oracle as LUNA API DB. By default, the service uses PostgreSQL DB.

You can manage DB in the *config.conf* file. The file may be found in the */var/lib/luna/current/luna-api/luna_api/configs*.

PostgreSQL configuration set by default:

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

Oracle configuration:

```
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 Oracle DBMS Configuration

Installation of the Oracle DBMS is described in the documentation on the Oracle corporation website.

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

You should add the environment variable NLS_LANG to all services with names *luna-*.service* in the folder */etc/systemd/system/* for Oracle before the services start. It is required for correct operation of the database.

```
Environment=NLS_LANG=ENGLISH
```

3.4.1.2 PostgreSQL DBMS Configuration

Note. We recommend you to install PostgreSQL version 9.6, the minimum allowed version of postgresql is 9.2, but it is not supported by the postgres command.

```
# Initialize database
/usr/pgsql-9.6/bin/postgresql96-setup initdb
```

Next you should configure availability of the database. Otherwise, you will not be able to change it. Open /var/lib/pgsql/9.6/data/pg_hba.conf.

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

Note. An example of configuration is given below.

You should change “Method" column values according to the example:

```
# 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
```

```
# Launch service and add it to autoload
systemctl start postgresql-9.6
systemctl enable postgresql-9.6
```

Create a new user and database. The user has a login, a password and privileges. By default, you should create a user “faceis”.

```
# Enter database interface
psql -U postgres

# Create database user
create role faceis;

# Assign password to the user
ALTER USER faceis WITH PASSWORD 'faceis';

# Create database
CREATE DATABASE faceis_db;

# Grant privileges to database and the user
GRANT ALL PRIVILEGES ON DATABASE faceis_db TO faceis;

# Enable the user to login into DB
ALTER ROLE faceis WITH LOGIN;
# Exit
\q
```

3.4.2 LUNA API Dependencies Installation

LUNA API is designed to provide limited user access to system resources.

```
# Go to module directory
cd /var/lib/luna/current/luna-api

# Create a virtual environment
python3.6 -m venv venv

# Activate the virtual environment
source venv/bin/activate
```

```
# Initiate Python dependencies installation
pip3.6 install -r requirements.txt

# Initiate database structure filling
python3.6 ./base_scripts/db_create.py

# Deactivate virtual environment
deactivate
```

3.4.3 Faces Service

Faces service stores faces, persons and lists.

3.4.3.1 DBMS Configuration

```
# Enter database interface
psql -U postgres

# Create database user
create role luna;

# Assign password to the user
ALTER USER luna WITH PASSWORD 'luna';

# Create database
CREATE DATABASE luna_faces;

# Grant privileges to database and the user
GRANT ALL PRIVILEGES ON DATABASE luna_faces TO luna;

# Enable the user to login into DB
ALTER ROLE luna WITH LOGIN;
# Exit
\q
```

3.4.3.2 Installation

The service is used for storing faces and lists.

```
# Go to module directory
cd /var/lib/luna/current/luna-faces


# Create a virtual environment
python3.6 -m venv venv

# Activate the virtual environment
source venv/bin/activate
```

```
# Initiate Python dependencies installation
pip3.6 install -r requirements.txt

# Initiate database structure filling
python3.6 ./base_scripts/db_create.py

# Deactivate virtual environment
deactivate
```

3.4.4 Image Store Service

Special warped images received from photos are used for descriptors creation. Warped images are stored in the IMAGE STORE and are used upon neural network update for descriptors re-extraction.

3.4.4.1 Portrait Storage Configuration

Images can be stored in Image Store or on a hard drive.

To enable storage of LUNA images in Image Store please change the parameter SEND_TO_LUNA_IMAGE_STORE in the configuration file 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
```

As well as in Image Store, LUNA portraits can be stored on a hard drive. To store portraits on a hard drive, enable the flag ENABLE_PLUGINS in the configuration file luna-api/luna_api/configs/config.conf.

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

3.4.4.2 Installation

Note. It is strongly recommended to create a virtual environment for python dependencies installation.

```
# Go to module directory
cd /var/lib/luna/current/luna-image-store


# Create a virtual environment
python3.6 -m venv venv

# Activate the virtual environment
source venv/bin/activate
```

```
# Initiate Python dependencies installation
pip3.6 install -r requirements.txt
# Deactivate virtual environment
deactivate
```

3.4.4.3 Launch Image Store

```
# Launch the service and add it to the startup
systemctl start luna-image-store
systemctl enable luna-image-store
```

3.4.4.4 Create Database

Note. The Image Store service should be launched before database creation.

```
# Go to API module directory
cd /var/lib/luna/current/luna-api


# Activate the virtual environment
source venv/bin/activate
```

```
# Initiate Image Store storage creation
python3.6 ./base_scripts/lis_bucket_create.py

# Deactivate virtual environment
deactivate
```

3.5 LUNA PLATFORM Services Launch

```
    # Launch LUNA PLATFORM services and add them to the startup
    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

    # Each of the Extractor and Matcher instances should be run separately by
specifying the instance number after the “@” symbol
    # Example:
    # systemctl start luna-extractor@1
    # systemctl enable luna-extractor@1
    # systemctl start luna-extractor@2
    # systemctl enable luna-extractor@2
```

4 Additional Services

Additional services provide index building and index search services, graphical interface, administrator panel and services for events and statistics. Their installation is not mandatory.

Package for Python 3.6 required for installation. To install the package, please execute the following commands:

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

Make sure, that the CORE and the API work correctly before installing additional services.

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

In the case, when you install additional services on a separate server or virtual machine you should perform the following steps:

Copy the distribution package of the system to your server/virtual machine, unzip it and create a symbolic link to the folder. The processed are described in the “Distribution Unpacking and Preparation” section.

In addition, update of CentOS, installation of epel-release and dependencies installation may be required. They are described in the “Software and Dependencies Installation” section.

You should also copy all required service files from the distribution package to the /etc/systemd/system/ directory. The process Is described in the “Service Files” section.

4.1 Index Building and Index Search Services

The following services are used for descriptors indexing in the list and searching in the generated index:

- Indexer creates index based on a descriptor list;
- Indexed Matcher searches by indexes;
- Matcher Daemon:
  - copies the index from the server, on which Indexer is installed, to the server for search by an index (Index Matcher),
  - restarts Indexed Matcher with a new index generation;
- Index Manager forms tasks for index building and coordinates the process of index delivery to Index Matcher servers.

Search by indexed descriptors is considerably faster.

Indexed Matcher receives information from Faces service about descriptors recently added to the list before searching in an indexed list. If there are any new descriptors, Indexed Matcher will receive them from a DB. Search will be performed for the indexed list and all recently added descriptors.

Note: The Index Matcher and the Matcher Daemon should be installed on the same host;

Indexer should be installed on a separate server.

4.1.1 Matcher Daemon Dependencies Installation

Matcher Daemon and Indexed Matcher server

The dependencies are required for the Matcher Daemon installation.

```
# Install Python and Python-setuptools
yum -y --nogpgcheck install python36 python36-setuptools
# Install pip
python3.6 /var/lib/luna/current/extras/get-pip.py
# Install dependency from the distribution archive
yum -y install /var/lib/luna/current/extras/rpms/matcher-daemon-*.rpm
```

4.1.2 Matcher Daemon Configuration

You should configure the Matcher Daemon access to the server with Indexer and allow Matcher Daemon to start Indexed Matcher. Matcher Daemon should be installed on the same server with Indexed Matcher.

Indexer server

Note. You should enable authorization by public key on the sever with Index. For this purpose, you should edit /etc/ssh/sshd_config file and restart the service: systemctl restart sshd. The procedure is not described in the manual.

```
# First of all you should create matcher-daemon user on the Index server
adduser matcher-daemon
```

Matcher Daemon and Indexed Matcher server

```
# On this step you should generate a ssh-key to grant access for Matcher Daemon to the Indexer server

# Change user to matcher-daemon
su matcher-daemon

# Generate a ssh-key
ssh-keygen
# The system will prompt you to enter the path to the key. Press Enter to allow system creation in default directory /home/<current_user>/.ssh/id_rsa
# The system will prompt a passphrase. It is optional. Press Enter three times.

# Specify the adress of the server with LUNA Indexer
ssh-keyscan -H <indexer_host>  >> ~/.ssh/known_hosts
# You should copy the public key file to the server with Indexer manually or using command:
scp /home/matcher-daemon/.ssh/id_rsa.pub matcher-daemon@<indexer_host>:<Directory_for_key>
```

Note. You should create a ssh key both when all the services are installed to a single server and when they are installed to separate servers.

Indexer server

```
On this step you should copy the public key to the file /home/matcher-daemon/.ssh/authorized_keys
```

```
# Change user to matcher-daemon
su matcher-daemon
# Create a directory
mkdir /home/matcher-daemon/.ssh
# Set permissions for the folder
chmod 700 /home/matcher-daemon/.ssh

# Add the contents of the file «id_rsa.pub» to the file /home/matcher-daemon/.ssh/authorized_keys. You may copy it manually or using the command:
cat <path_to_file_id_rsa.pub> >> /home/matcher-daemon/.ssh/authorized_keys
# Set permissions
chmod 600 /home/matcher-daemon/.ssh/authorized_keys
```

Matcher Daemon and Indexed Matcher server

```
# You should add the matcher-daemon permission to start/stop the Indexed Matcher without entering a password
# Switch to root user
sudo su
# Open the sudoers file:
visudo
# After the lines:
# Allow root to run any commands anywhere
root    ALL=(ALL)       ALL
# Add the following line:
matcher-daemon ALL=(ALL) NOPASSWD: /bin/systemctl start luna-indexed-matcher, /bin/systemctl stop luna-indexed-matcher, /bin/systemctl restart luna-indexed-matcher
```

```
# Edit the matcher-daemon configuration file:
vim /etc/matcher-daemon/matcher-daemon-config.json
# Enter the indexer server adress, path to the index folder and path to the key in the following fields (the default example is given):
  "indexer_host": "127.0.0.1",
  "Index_holding_dir": "/var/lib/luna/index",
  "ssh_key": "/home/matcher-daemon/.ssh/id_rsa"

# "indexer_host" includes adress to the indexer server.
# In the "Index_holding_dir" field you should specify the directory for index. The path to the directory should be similar to the path set in the /var/lib/luna/current/conf/indexer.conf file. By default the directory is "/var/lib/luna/index".
# "ssh_key" should include the path to the id_rsa file.
```

```
# You should create the index directory and set permissions to it
mkdir /var/lib/luna/index
chmod 700 /var/lib/luna/index
```

4.1.3 LUNA Index Manager DB Configuration

```
# Enter database interface
psql -U postgres

# Create database user
create role luna;
```

```
# Assign password to the user
ALTER USER luna WITH PASSWORD 'luna';

# Create database
CREATE DATABASE luna_index_manager;
# Grant privileges to database and the user
GRANT ALL PRIVILEGES ON DATABASE luna_index_manager TO luna;

# Enable the user to login into DB
ALTER ROLE luna WITH LOGIN;

# Exit
\q
```

4.1.4 LUNA Index Manager Installation

Change the port for Matcher Daemon in the configuration file: */var/lib/luna/current/luna-index-manager/luna_index_manager/configs/config.conf*. The port should be set to 6001.

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

Make sure all other paths are set correctly.

```
# Go to module directory
cd /var/lib/luna/current/luna-index-manager

# Create a virtual environment
python3.6 -m venv venv
# Activate the virtual environment
source venv/bin/activate

# Initiate Python dependencies installation
pip3.6 install -r requirements.txt

# Initiate database structure filling
python3.6 ./base_scripts/db_create.py
```

```
# Deactivate virtual environment
deactivate
```

4.1.5 Launch Index Building and Index Search Services

Launch services.

```
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

# Check status of the services
systemctl status luna-indexer matcher-daemon luna-index-manager
```

4.2 User Interface Service

The UI service implements a graphical interface for LUNA PLATFORM users.

4.2.1 Installation

Note. It is strongly recommended to create a virtual environment for Python dependencies installation.

Install dependencies.

```
# Install python if it is not installed already
yum -y install python36 python36-devel

# Go to the folder with the project
cd /var/lib/luna/current/luna-ui2

# Create a virtual environment
python3.6 -m venv venv

# Activate the virtual environment
source venv/bin/activate

# Install dependencies from the file
pip3.6 install -r requirements.txt
```

4.2.2 Configuration

Generate a key to provide secure sessions.

```
# Go to the folder with the script
cd /var/lib/luna/current/luna-ui2/app/scripts
```

```
# Generate a key
python3.6 generate_secret_key.py
```

```
# Deactivate virtual environment
deactivate
```

Copy the key and paste it to the */var/lib/luna/current/luna-ui2/app/config.ini* file

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

to the field:

```
COOKIE_SECRET = <secret_key>
```

You should also specify a correct IP address of the API service:

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

4.2.3 Launch User Interface Service

If all the previous actions are executed successfully, the server is ready to work.

```
# Start the service:
systemctl start luna-ui.service
systemctl enable luna-ui.service
```

```
# Check the services status:
systemctl status luna-ui.service
```

UI uses port 80 by default. You can open UI by address: http://<correct address>:80

4.3 Administration Panel Service

The administration service is a web application that provides a graphical interface to some common administrative routines. Also, this service has ability to communicate with Grafana (a 3rd party tool) for viewing system metric.

Before installation, make sure that LUNA CORE and LUNA API work correctly.

4.3.1 System Requirements

Following software is required for administration panel service installation:

- PostgeSQL developer package;
- PostgeSQL DBMS instance;
- Python 3.6;
- Python setuptools.

4.3.2 Database Creation

Create database for Administration service.

```
# Enter database interface
psql -U postgres;

# Create database user
create role faceis;

# Assign password to the user
ALTER USER faceis WITH PASSWORD 'faceis';

# Create database
CREATE DATABASE admin_faceis_db;

# Grant privileges to database user
GRANT ALL PRIVILEGES ON DATABASE admin_faceis_db TO faceis;

# Allow user to authorize in the DB
ALTER ROLE faceis WITH LOGIN;

# Exit
\q
```

4.3.3 Installation

Install dependencies:
Note. *A virtual environment is strongly recommended for installation.*

```
 # Go to the folder with the project
 cd /var/lib/luna/current/luna-admin

 # Create a virtual environment
python3.6 -m venv venv

# Activate the virtual environment.
source venv/bin/activate
```

```
# Install dependencies from the file
pip3.6 install -r requirements.txt

# Initiate database structure filling
python3.6 bases_scripts/db_create.py

# Deactivate virtual environment
deactivate
```

Note. **db_create.py** also creates scripts for further database migration (**db_repository** folder), without them **db_migrate** script execution is impossible.

4.3.4 Configuration

All configurations are in the **configs/config.py** file. The file can be changed in accordance with your needs.

Note. All options except garbage collector options must be synchronized with LUNA API options.

4.3.5 Launch Administration Panel Service

Launch the service:

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

# Check the services status:
systemctl status luna-admin_back luna-admin_tasks
```

4.3.6 Grafana dashboard creation

To create the dashboard:

```
    # Go to the folder with the project
    cd /var/lib/luna/current/luna-admin
    # Run the script
    python3.6 ./bases_scripts/create_grafana_dashboards.py --
config=./configs/config.conf
```

Note. Grafana dashboard creation is ***optional****.*

4.4 Event & Statistic Service

Event & Statistic Service realizes RESTful API and allows keeping track of all requests that LUNA processes.

Two services are included into Event & Statistic:

- LUNA API Python Server Events – events service;
- Statistic Manager – statistics service.

4.4.1 System Requirements

Following operation systems are supported:

- RedHat Linux 7;
- CentOS Linux 7.

Third-party dependencies to be installed:

- Python 3.6;
- Redis v. 3.2.3 or higher;
- InfluxDB v. 1.3.2;
- Additional library dependencies listed in **requirements.txt**.

4.4.2 Packages and Requirements Installation

To install packages and requirements, you should:

- Install Python 3.6.;
- Install InfluxDB;
- Install Redis;
- Install requirements;
- Configure database.

4.4.2.1 Install InfluxDB

InfluxDB stores statistic with time reference about user requests handling. It allows retrieving statistics for the specified period.

```
    # Load and install .rpm package
    yum -y install https://dl.influxdata.com/influxdb/releases/influxdb-
1.3.2.x86_64.rpm
    # Add service to startup
    systemctl enable influxdb.service
    # Start service
    systemctl start influxdb
```

4.4.2.2 Install Redis

Redis DB is used for temporary data storage and events generation. The events coincide the LUNA reply on a user request. The events are delivered in real time by means of web sockets.

Perform the following commands:

```
    # Install
    yum -y install redis

    # Add service to the startup
    systemctl enable redis.service
    # Start the service
    systemctl start redis
```

4.4.2.3 Install Python libraries

Note. Virtual environment is strongly recommended for installation.

Perform the following commands:

```
    # Install python if it is not installed already
    yum -y --nogpgcheck install python36-setuptools

    # Go to the folder with the project
    cd /var/lib/luna/current/luna-stat-server

    # Create a virtual environment
    python3.6 -m venv venv

    # Activate the virtual environment.
    source venv/bin/activate

    # Install dependencies from the file
    pip3.6 install -r requirements.txt
    # Deactivate the environment
    deactivate
```

4.4.3 Configuration

You should check and, if necessary, change the following settings in the /var/lib/luna/current/luna-stat-server/config.conf configuration file and change ports of the databases, if it is necessary:

```
    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"
```

4.4.4 Database Creation

Create a database.

```
# Go to the folder with the project
cd /var/lib/luna/current/luna-stat-server
# Activate the virtual environment.
source venv/bin/activate
# Create DB
python3.6 db_create.py
# Deactivate the environment
deactivate
```

4.4.5 Launch Event & Statistic Service

Start the Events service:

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

Start the Statistic Manager service (service interface):

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

The event service receives events from LUNA API on the 5009 port. The statistics service provides subscription and statistics on the 5008 port. You can assign any free port.

Note. *You can start several instances of both services.*

```
# Check the services status:
systemctl status luna-stat-lpse.service luna-stat-sm.service
```

5 LUNA API Statistics Configuration

To enable administrator statistics accumulation (more information on administration panel service setup and configuration are provided in the section 6.3), one should:

- create a database in InfluxDB;

```
# Go to the folder with the project
cd /var/lib/luna/current/luna-api
# Activate the virtual environment.
source venv/bin/activate

# Create DB
python3.6 ./base_scripts/influx_db_create.py
# Deactivate environment
deactivate
```

- change the parameters SEND_ADMIN_STATS, ADMIN_STATISTICS_SERVER, ADMIN_STATISTICS_DB in the configuration file /var/lib/luna/current/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"
```

To enable events statistics accumulation in the **Event & Statistic service**, one should change the parameters SEND_ACCOUNT_STATS and ACCOUNTS_STATISTICS_SERVER in the configuration file 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 LUNA PLATFORM Testing

Tests are performed for all services of the LUNA PLATFORM except UI and Statistic services.

To perform tests, you should follow the steps:

1. Ensure that services and LUNA API are running:

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

Note. You should check all the required instances of luna-extractor and luna-matcher.

2. Check and, if necessary, change the service settings in the **/var/lib/luna/current/luna-api/tests/config.py** configuration file:

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

3. Run tests. If everything works, the std output will show “Ok”:

```
# Go to the folder with the project
cd /var/lib/luna/current/luna-api

# Activate the virtual environment
source venv/bin/activate

# Run tests
python3.6 -m unittest tests.unittests_main

# Deactivate virtual environment
deactivate
```

Appendix: Changelog

Date	Version	Notes
22.05.17	1	Initial release
23.08.17	2	Added instructions for UI and administration modules
24.08.17	3	Added instructions for events & statistics module
06.07.18	4	Added instructions for Indexer, Indexed Matcher, Matcher Daemon, Index Manager and Faces services
16.08.18	5	The document was updated
12.09.18	6	Glossary section, General information section and information about Oracle DBMS were added