VisionLabs LUNA PLATFORM 5

Example of deployment using Docker Compose

v.5.67.0

Contents

Default ports for services

Service name	Port
LUNA PLATFORM API	5000
LUNA PLATFORM Admin	5010
LUNA PLATFORM Image Store	5020
LUNA PLATFORM Faces	5030
LUNA PLATFORM Events	5040
LUNA PLATFORM Tasks	5050
LUNA PLATFORM Tasks Worker	5051
LUNA PLATFORM Configurator	5070
LUNA PLATFORM Sender	5080
LUNA PLATFORM Handlers	5090
LUNA PLATFORM Python Matcher	5100
LUNA PLATFORM Licenses	5120
LUNA PLATFORM Backport 4	5130
LUNA PLATFORM Backport 3	5140
LUNA PLATFORM Accounts	5170
LUNA PLATFORM Lambda	5210
LUNA PLATFORM Remote SDK	5220
LUNA PLATFORM 3 User Interface	4100
LUNA PLATFORM 4 User Interface	4200
Oracle DB	1521
PostgreSQL	5432
Redis DB	6379
InfluxDB	8086
Grafana	3000

Configuration names for services

The table below includes the service names in the Configurator service. Use these parameters to configure your services.

Service	Service name in Configurator
API	luna-api
Licenses	luna-licenses
Faces	luna-faces
Image Store	luna-image-store
Accounts	luna-accounts
Tasks	luna-tasks
Events	luna-events
Sender	luna-sender
Admin	luna-admin
Remote SDK	luna-remote-sdk
Handlers	luna-handlers
Lambda	luna-lambda
Python Matcher	luna-python-matcher
Backport 3	luna-backport3
Backport 4	luna-backport4

Settings for the Configurator service are set in its configuration file.

Introduction

This document describes the installation and usage of Docker Compose for LUNA PLATFORM deployment.

> This document includes an example of LUNA PLATFORM deployment using the Compose script example. It implements LUNA PLATFORM minimum power operating for demonstration purposes and cannot be used for the production system.

It is considered that installation is performed on the server with CentOS OS, where LP was not installed.

Docker Compose is used for automated containers deployment. The Docker Compose scenario from this distribution is used for deploying the LUNA PLATFORM services on a single server.

Firewall and SELinux should be manually configured on the server by the administrator. Their configuration is not described in this document.

No data backup or databases replication is implemented for LP data in this installation.

To deploy the LUNA PLATFORM using the Docker Compose script, follow the steps in the following sections:

- "Before launch" — Actions for unpacking archives, preparing directories, configuring licenses, etc. Some actions may be optional.
- "LUNA PLATFORM launch" — Launching the Docker Compose script to deploy the LUNA PLATFORM.

To use the Docker Compose script, a LUNA PLATFORM network license is required. The license is provided by VisionLabs on request separately from the delivery. The license key is created using the fingerprint of the system. This fingerprint is created based on information about the hardware characteristics of the server. Thus, the received license key will work only on the same server from which the system fingerprint was obtained. LUNA PLATFORM can be activated using one of two utilities — HASP or Guardant. The section "Activate license" provides useful links to instructions for activating the license key for each method.

This document also contains instructions for automatically launching LUNA Dashboards (Grafana) and Loki (see the "Enable Grafana and Loki" section).

The "Additional information" section provides useful information on configuring logging, enabling log rotation, Docker commands, etc.

Additional notes about Docker Compose script. The script:

- Is tested using the default services configurations.
- Is not intended to be used for LP scaling:
    - It is not used for the deployment of LP services on several servers.
    - It is not used for deployment and balancing of several LP services on a single service.

- Launches default databases and does not include a build-in possibility to change the databases used.

- Supports GPU utilization for LP calculations.

- Does not provide the possibility to use external databases already installed on the server.

- Does not perform migrations from previous LP versions and updates from the previous LP build.

- Does not run the services Backport 3, Backport 4, User Interface 3, User Interface 4.

See the "docker-compose.yml" file and other files in the "example-docker" directory for the information about launched services and performed actions.

You can write your scenario that deploys and configures all the required services. This document does not include information about scenario creation or tutorial for Docker usage. Please refer to the Docker documentation to find more information about Docker and Docker Compose:

https://docs.docker.com

> It is recommended to use orchestration services for the commercial usage of LP. Their utilization is not described in this manual.

> All the provided commands should be executed using the Bash shell (when you launch commands directly on the server) or in a program for working with network protocols (when you remotely connect to the server), for example, Putty.

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

All actions described in this manual must be performed by the **root** user. This document does not describe the creation of the user with administrator privileges and the following installation by this user.

1 Before launch

Make sure that you are the **root** user before launch!

Before launching the LUNA PLATFORM, you must perform the following actions:

1. Unpack the LUNA PLATFORM distribution.
2. Create symbolic link.
3. Configure SELinux and Firewall.
4. Activate license.
5. Install Docker.
6. Install Docker Compose.
7. Choose logging method.
8. Set up GPU computing if you plan to use GPU.
9. Login to VisionLabs registry.

1.1 Distribution unpacking

The distribution package is an archive **luna_v.5.67.0**, where **v.5.67.0** is a numerical identifier, describing the current LUNA PLATFORM version.

The archive includes configuration files, required for installation and exploitation. It does not include Docker images for the services. They should be downloaded from the Internet.

Move the distribution package to the directory on your server before the installation. For example, move the files to `/root/` directory. The directory should not contain any other distribution or license files except the target ones.

Move the distribution to the created directory.

```
mv /root/luna_v.5.67.0.zip /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_v.5.67.0.zip
```

1.2 Symbolic link creation

Create a symbolic link.

> The link indicates that the current version of the distribution file is used to run LUNA PLATFORM.

```
ln -s luna_v.5.67.0 current
```

1.3 SELinux and Firewall

You must configure SELinux and Firewall so that they do not block LUNA PLATFORM services.

> SELinux and Firewall configurations are not described in this guide.

If SELinux and Firewall are not configured, the installation cannot be performed.

1.4 License activation

To activate the license, follow these steps:

- Follow the steps from license activation manual.
- Set settings for HASP license or Guardant license.

1.4.1 Actions from License activation manual

Open the license activation manual and follow the necessary steps.

Note: This action is mandatory. The license will not work without following the steps to activate the license from the corresponding manual.

1.4.2 Specify HASP license settings

For the HASP key, you need to specify the IP address of the licensing server. The address is set in the dump file "platform_settings.json". The contents of the default settings will be overwritten by the contents of this file when the Configurator service starts.

Open the "platform_settings.json" file.

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

Set the server IP address with your HASP key in the "server_address" field.

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

Save the file.

> If the license is activated using the HASP key, then two parameters "vendor" and "server_address" must be specified. If you want to change the HASP protection to Guardant, then you need to add the "license_id" field.

1.4.3 Specify Guardant license settings

For the Guardant key, you need to specify the IP address of the licensing server and the license ID. The settings are set in the dump file "platform_settings.json". The contents of the standard settings will be overwritten by the contents of this file at the launch stage of the Configurator service.

Open the file "platform_settings.json".

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

Enter the following data:

- IP address of the server with your Guardant key in the "server_address" field.
- License ID in the format `0x<your_license_id>`, obtained in the section "Save license ID" of license activation manual, in the "license_id" field.

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

Save the file.

> If the license is activated using the Guardant key, then three parameters "vendor", "server_address" and "license_id" must be specified. If you want to change the Guardant protection to HASP, then you need to delete the "license_id" field.

1.5 Docker installation

The Docker installation is described in the official documentation.

Note: Docker version 25.0.3 was used when testing this instruction. It is not guaranteed to work with higher versions of Docker.

Quick installation commands are listed below.

> Check the official documentation for updates if you have any problems with the installation.

Install dependencies:

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

Add repository:

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

Install Docker:

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

Launch Docker:

```
systemctl start docker
```

```
systemctl enable docker
```

Check Docker status:

```
systemctl status docker
```

1.6 Docker Compose installation

Note: Docker Compose version 2.24.6 was used when testing this instruction. It is not guaranteed to work with higher versions of Docker Compose.

Download Docker Compose binary:

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

Set execution permissions:

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

Create symbolic link:

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

1.7 Choose logging method

There are two methods to output logs in LUNA PLATFORM:

- Standard log output (stdout).
- Log output to a file.

Log output settings are set in the settings of each service in the `<SERVICE_NAME>_LOGGER` section.

If necessary, you can use both methods of displaying logs.

> For more information about the LUNA PLATFORM logging system, see the "Logging" section in the administrator manual.

1.7.1 Logging to stdout

This method is used by default and requires no further action.

It is recommended to configure Docker log rotation to limit log sizes (see "Docker log rotation").

1.7.2 Logging to file

Note: When you enable saving logs to a file, you should remember that logs occupy a certain place in the storage, and the process of logging to a file negatively affects system performance.

To use this method, you need to perform the following additional actions:

- **Before launching the services:** Create directories for logs on the server.
- **After launching the services:** Activate log recording and set the location of log storage inside LP service containers.
- **During the launch of services:** Configure synchronization of log directories in the container with logs on the server using the `volume` argument at the start of each container.

> Synchronization of log directories is already configured in the Docker Compose script, you only need to create directories and activate logging.

See the instructions for enabling logging to files in the "Logging to server" section.

1.8 Calculations using GPU

You can use GPU for the general calculations performed by Remote SDK.

Skip this section if you are not going to utilize GPU for your calculations.

> Docker Compose v1.28.0+ is required to use the GPU.

You need to install NVIDIA Container Toolkit to use GPU with Docker containers. The example of the installation is given below.

```
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
```

Install the nvidia-docker2 package (and dependencies) after updating the package listing:

```
yum clean expire-cache
```

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

```
systemctl restart docker
```

Check the NVIDIA Container toolkit operating by running a base CUDA container (this container is not provided in the LP distribution and should be downloaded from the Internet):

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

See the NVIDIA documentation for additional information.

Next, you should additionally add a `deploy` section to the `remote-sdk` field in the `docker-compose.yml` file.

```
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
     ...
```

Here:

- driver — This field specifies the driver for the reserved device(s).

- count — This field specifies the number of GPU devices that should be reserved (providing the host holds that number of GPUs).

- capabilities — This field expresses both generic and driver specific capabilities. It must be set, otherwise, an error will be returned when deploying the service.

See the Docker documentation for additional information.

> Attributes extraction on the GPU is engineered for maximum throughput. The input images are processed in batches. This reduces computation cost per image but does not provide the shortest latency per image.

> GPU acceleration is designed for high load applications where request counts per second consistently reach thousands. It won't be beneficial to use GPU acceleration in non-extensively loaded scenarios where latency matters.

1.9 Login to registry

When launching containers, you should specify a link to the image required for the container launching. This image will be downloaded from the VisionLabs registry. Before that, you should login to the registry.

> Login and password can be requested from the VisionLabs representative.

Enter login <username>.

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

After running the command, you will be prompted for a password. Enter password.

In the `docker login` command, you can enter the login and password at the same time, but this does not guarantee security because the password can be seen in the command history.

2 LUNA PLATFORM launch

The launch of Docker Compose is performed using a script "start_platform.sh", located in the "example-docker" directory.

> If necessary, you can modify the Docker Compose startup script for user needs. Modification of the script is intended only for experienced users.

When the script is run, a default account of type **user** will be created with login `user@mail.com` and password `password`. Instructions for creating your own account are given below.

> See detailed information about accounts in the "Accounts, tokens and authorization types" section of the administrator manual.

2.1 Launch services

Go to the Docker Compose folder.

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

> Make sure that LP containers are not launched before executing the script. An error will occur if you try to run a container with the same name as an existing container. If one or several LP containers are launched, you should stop them using the `docker container rm -f <container_name>` command. To stop all the containers, use `docker container rm -f $(docker container ls -aq)`.

Launch Docker Compose.

> You must be logged in the VisionLabs registry (see section "Login to registry").

```
./start_platform.sh
```

Deploying containers takes some time. You have to wait until all the services are running before working with LUNA PLATFORM.

Check the state of launched Docker containers.

```
docker ps
```

2.1.1 Run Remote SDK utilizing GPU

The Remote SDK service does not utilize GPU by default. If you are going to use the GPU, then you should enable its use for the Remote SDK service in the Configurator service.

If you need to use the GPU for all estimators and detectors at once, then you need to use the "global_device_class" parameter in the "LUNA_REMOTE_SDK_RUNTIME_SETTINGS" section. All estimators and detectors will use the value of this parameter if the "device_class" parameter of their settings like "LUNA_REMOTE_SDK_<estimator-or-detector-name>_SETTINGS.runtime_settings" is set to "global" (by default for all estimators and detectors).

If you need to use the GPU for a specific estimator or detector, then you need to use the "device_class" parameter in sections like "LUNA_REMOTE_SDK_<estimator/detector-name>_SETTINGS.runtime_settings".

See section "Calculations using GPU" for additional requirements for GPU utilization.

2.2 Account creation

Note: When you run the Docker Compose script, an account of type "user" is automatically created with login "user@mail.com" and password "password". Instructions for creating an account with your authentication data are given below.

The account is created using an HTTP request to the "create account" resource of the API service.

> You can also create an account using the Admin service. This method requires an existing login and password (or the default login and password) and enables you to create an "admin" account. See the "Admin service" section of the administrator manual for details.

To create the account using a request to the API service, you need to provide the following mandatory data:

- "login" — Email address.
- "password" — Password.
- "account_type" — Account type ("user" or "advanced_user").

Create the account using your authentication details.

Example of CURL-request to the "create account" resource:

```
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"
}'
```

It is necessary to replace the authentication data from the example with your own.

> To work with tokens, you must have an account.

2.3 Activate GC task schedule

Before you start working with the LUNA PLATFORM, you can create a schedule for the Garbage collection task.

To do this, make a "create tasks schedule" request to the API service, specifying the necessary rules for the schedule.

An example of a schedule creation command for an account created in section "Account creation" is given below.

The example sets a schedule for the Garbage collection task for events older than 30 days with the removal of the samples and the source images. The task will be repeated **once a day at 05:30 am**.

```
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}
}'
```

If necessary, you can create a schedule without automatically activating it. To do this, specify the parameter "create_stopped": "true". In this case, after creating the schedule, it must be activated manually using the "action" = "start" parameter of the "patch tasks schedule" request.

For more information, see the "Running scheduled tasks" section of the administrator manual.

2.4 Enable Grafana and Loki

Note: Follow these steps if you want to use LUNA Dashboards (Grafana) and Loki. Otherwise, skip this step.

To use Grafana and Loki, you can execute a script `start_logging.sh`, launching the LUNA Dashboards, Loki and Promtail service. This script must be executed after executing the main Docker Compose script.

> See detailed information about monitoring visualization in the "LUNA Dashboards" and "Grafana Loki" section of the administrator manual.

Go to the Docker Compose folder.

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

Launch Docker Compose.

```
./start_logging.sh
```

Check the state of launched Docker containers.

```
docker ps
```

3 Additional information

This section provides the following additional information:

- Useful commands for working with Docker.
- Actions to enable saving LP service logs to files.
- Configuring Docker log rotation.

3.1 Docker commands

3.1.1 Show containers

To show the list of launched Docker containers use the command:

```
docker ps
```

To show all the existing Docker containers use the command:

```
docker ps -a
```

3.1.2 Copy files to container

You can transfer files into the container. Use the `docker cp` command to copy a file into the container.

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

3.1.3 Enter container

You can enter individual containers using the following command:

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

To exit the container, use the command:

```
exit
```

3.1.4 Images names

You can see all the names of the images using the command:

```
docker images
```

3.1.5 Delete image

If you need to delete an image:

- Run the `docker images` command.

- Find the required image, for example dockerhub.visionlabs.ru/luna/luna-image-store.
- Copy the corresponding image ID from the IMAGE ID, for example, "61860d036d8c".
- Specify it in the deletion command:

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

Delete all the existing images.

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

3.1.6 Stop container

You can stop the container using the command:

```
docker stop <container_name>
```

Stop all the containers:

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

3.1.7 Delete container

If you need to delete a container:

- Run the "docker ps" command.
- Stop the container (see Stop container).
- Find the required image, for example dockerhub.visionlabs.ru/luna/luna-image-store.
- Copy the corresponding container ID from the CONTAINER ID column, for example, "23f555be8f3a".
- Specify it in the deletion command:

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

Delete all the containers.

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

3.1.7.1 Check service logs

You can use the following command to show logs for the service:

```
docker logs <container_name>
```

3.2 Docker log rotation

To limit the size of logs generated by Docker, you can set up automatic log rotation. To do this, add the following data to the `/etc/docker/daemon.json` file:

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

This will allow Docker to store up to 5 log files per container, with each file being limited to 100MB.

After changing the file, you need to restart Docker:

```
systemctl reload docker
```

The above changes are the default for any newly created container, they do not apply to already created containers.

3.3 Logging to server

To enable saving logs to the server, you should:

- Create directories for logs on the server.
- Activate log recording and set the location of log storage inside LP service containers.
- Configure synchronization of log directories in the container with logs on the server using the `volume` argument at the start of each container.

> Docker Compose script is already configured to synchronize directories with the folders created in the section below.

3.3.1 Create logs directory

You need to create the following directories for storing logs and assign them the appropriate rights.

```
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
```

```
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
```

If you need to use the Python Matcher Proxy service, then you need to additionally create the `/tmp/logs/python-matcher-proxy` directory and set its permissions.

3.3.2 Logging activation

3.3.2.1 LP services logging activation

To enable logging to file, you need to set the `log_to_file` and `folder_with_logs` settings in the `<SERVICE_NAME>_LOGGER` section of the settings for each service.

Automatic method

To update logging settings, you can use the `logging.json` settings file provided with the distribution package.

Run the following command after starting the Configurator service:

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

Update your logging settings with the copied file.

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

Manual method

Go to the Configurator service interface (`127.0.0.1:5070`) and set the logs path in the container in the `folder_with_logs` parameter for all services whose logs need to be saved. For example, you can use the path `/srv/logs`.

Set the `log_to_file` option to `true` to enable logging to file.

3.3.2.2 Configurator service logging activation

The Configurator service settings are not located in the Configurator user interface, they are located in the following file:

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

Set the path to the logs location in the container in the `FOLDER_WITH_LOGS = ./` parameter of the file. For example, `FOLDER_WITH_LOGS = /srv/logs`.

Set the `log_to_file` option to `true` to enable logging to file.

> You should restart Configurator after making changes.