VisionLabs LUNA PLATFORM 5

Storages utility manual

v.5.67.0

Contents

Introduction

This document describes the Storages utility and how to use it. It is recommended to familiarize yourself with this document before using the utility.

The Storages utility enables you to check and/or prepare the environment for LUNA PLATFORM services before launching them. The preparation of the environment is understood as follows:

- Preparing buckets in InfluxDB for monitoring work.
- Preparing buckets for the Image Store service, enabling you to store user data (images, metadata, archives, etc.).
- Preparing the InfluxDB for collection of aggregated statistics by the Admin service (see "Requests and estimations statistics gathering" section in the administrator manual).
- Preparing databases, adding VLMatch functionality, creating and managing migration scripts for LUNA PLATFORM databases (PostgreSQL or Oracle).
- Performing migration/loading settings into the Configurator database.
- Loading dump files into the Configurator service database.

The main advantage of Storages over manual preparation of the environment is that the utility knows revisions of the Configurator service settings and migration scenarios for LP service databases, which greatly simplifies the process of updating and downgrading LUNA PLATFORM. In addition, using the utility, you can evaluate the current state of the LUNA PLATFORM environment and determine what exactly needs to be updated.

The utility is supplied in a Docker container and can be used as a tool for preparing the LUNA PLATFORM environment deployed in Docker containers, the Docker Compose script, the Kubernetes orchestration system, etc. The main task of the administrator is to indicate to the Storages utility the addresses of databases, buckets, etc. (see "Setting Storages configuration" section).

1 Use cases for Storages utility

The Storages utility can be used for updating or downgrading, or for installing from scratch.

> It is recommended to create backups before any actions related to updating or downgrading. See recommendations for creating backups and useful links in the update manual.

Example commands for updating and installing from scratch are provided in the corresponding documents. A downgrade is actually no different from an update.

The sections below provide general procedures for performing these tasks.

1.1 Launch from scratch

The general procedure for installing from scratch is as follows:

1. Launch and configure PostgreSQL/Oracle and InfluxDB databases — create users, configure rights, etc. (see the "Databases" section in the administrator manual).
2. Compile the VLMatch library for a user version of the PostgeSQL or Oracle database (see the "Databases" section in the administrator manual). The Storages utility automatically adds the VLMatch function to the Faces and Events databases during the database environment preparation phase.
3. Correctly fill out the Storages configuration, indicating all the necessary data — database addresses and authentication data, S3 connection settings (if necessary), bucket addresses, etc.
4. Prepare the environment using the prepare command.
5. Make sure that the environment of the version of interest has been successfully prepared using the check command.
6. Correctly fill in user settings for services (connection to services, databases, etc.) in one of the following ways:
   - Execute the command load_dump, loading a dump file with user settings *
   - Configure and launch the Configurator service, specifying user settings in it.
   - Fill in the service configuration files, specifying user settings in them.
7. Launch LUNA PLATFORM services.

* you can edit the full `luna_platform_<version>_dump.json` dump file included in the distribution package, or use a custom dump file (see example of a custom dump file `platform_settings.json` included in the distribution package). See the "Configurator service" section in the administrator manual for more information about dumps.

1.2 Upgrade/downgrade

The general procedure for upgrading or downgrading is as follows:

1. Correctly fill out the Storages configuration, indicating all the necessary data — database addresses and authentication data, S3 connection settings (if necessary), bucket addresses, etc.
2. Make sure that Storages supports preparing the environment for LUNA PLATFORM of the version you are interested in using the list command.
3. Check the current LUNA PLATFORM environment using the check command.
4. Configure the behavior of services during environment preparation
5. Prepare the environment using the prepare command.
6. Make sure that the environment of the version of interest has been successfully prepared using the check command.
7. Update user settings for services (connection to services, databases, etc.) in one of the following ways (optional):
   - Execute the command load_dump, loading a dump file with user settings *
   - Configure and launch the Configurator service, specifying user settings in it.
   - Fill in the service configuration files, specifying user settings in them.
8. Launch the new version of LUNA PLATFORM services.

* you can edit the full `luna_platform_<version>_dump.json` dump file included in the distribution package, or use a custom dump file (see example of a custom dump file `platform_settings.json` included in the distribution package). See the "Configurator service" section in the administrator manual for more information about dumps.

1.2.1 Recommendations services behavior during environment preparation

When preparing your environment, it is important to carefully plan the impact on services that use databases.

Below are recommendations for managing services in various migration scenarios.

- **Load reduction:** Try to reduce the number of requests to a minimum during environment preparation. This will help reduce the likelihood of conflicts and data integrity issues.

- **Restricted Access:** Consider temporarily restricting access to services to avoid inconsistent changes during migration.

- **Stopping services:** Consider stopping services that actively interact with the database. This can help reduce potential conflicts and simplify the migration process.

- **Caution:** If you leave the service running during migration, be prepared for possible delays in the operation of LUNA PLATFORM.

The final decision on how to manage services should be made based on the specific requirements of your system and business processes.

2 Setting Storages configuration

To correctly prepare the environment, you need to configure the Storages configuration so that the utility can interact with various services, databases, buckets and other resources. The required Storages settings can be specified using:

- Storages service configuration file.
- Running Configurator service (during update/downgrade).

The method for transferring settings is specified using the arguments of the corresponding commands (see below). If no argument is given, the Storages utility will use the default configuration file. If both arguments are given, priority will be given to receiving settings from the running Configurator service.

Using Storages configuration file

By default, the configuration file is located in the Storages container. If you need to specify settings other than the default settings, you can change a copy of the default configuration file located in the distribution package at the path `luna_v.5.67.0/extras/conf/storages_config.conf` and specify it using the argument –config. The configuration file must also be mounted to the Storages container.

This method is listed as an example in the installation and upgrade manuals using Storages.

Using settings from running Configurator service

The address of the running Configurator service is passed in the –luna-config argument of the corresponding command.

This method requires preliminary preparation of the environment for the new version of the Configurator service. An example of using this method is given in the section "Environment upgrade scenario".

3 Utility commands

The preparation of the environment is carried out using the `luna_prepare` script, to which specific commands are passed, enabling for detailed configuration of the preparation process.

List of available commands:

- check — Command that enables you to check the existing environment in accordance with the specified information about the LUNA PLATFORM version.
- list — Command that enables you to show a list of available versions of LUNA PLATFORM for preparing the environment.
- prepare — **Main command**, which enables you to prepare the environment in accordance with the specified LUNA PLATFORM version information.
- load_dump — Command that enables you to load settings from a user dump file into the Configurator database.
- logs — Command that enables you to display or save to a file special logs received during the environment preparation stage.

Each command except the `list` command has its own arguments. A list of possible arguments with their basic description can be obtained using the help argument `--help`.

> For an extended description of the arguments, see "Named arguments" section.

In addition, the `prepare` command has two types of arguments — positional and named (see "Prepare command") and you can also get help information for each type of argument.

For example, you can get a list of arguments for the `load_dump` command with the following command:

```
docker run\
--rm\
--network=host \
dockerhub.visionlabs.ru/luna/storages:v.0.2.51 \
bash -c "luna_prepare load_dump --help"
```

Important: It is highly recommended that you use help arguments when working with the Storages utility.

3.1 List command

The Storages utility does not work with all versions of LUNA PLATFORM. The `list` command enables you to get a list of supported versions of the environment preparation run.

Important: The version list also contains releases intended for internal use. Only public versions of LUNA PLATFORM must be used.

Example of a command to get a list of versions:

```
docker run\
--rm\
--network=host \
dockerhub.visionlabs.ru/luna/storages:v.0.2.51 \
bash -c "luna_prepare list"
```

Example of successful command execution:

```
[2024-02-07 16:20:50+0300] [storages] [INFO]: Available platform versions
    for surroundings preparation:
[2024-02-07 16:20:50+0300] [storages] [INFO]: v.5.46.1, v.5.47.1, v.5.47.4,
    v.5.49.1, v.5.51.0, v.5.51.4, v.5.51.6, v.5.53.0, v.5.54.0, v.5.55.0, v
    .5.56.0, v.5.57.0
```

3.2 Check command

The `check` command enables you to check the current environment and get information about what is needed to install an environment of a different version. Using this command, for example, you can verify whether the user really has a certain version of the environment installed.

The following arguments are available for the `check` command:

- –config
- –luna-config
- –profile
- –platform_version
- –s3-buckets
- –local-buckets

Example of a command to check the current environment:

```
docker run \
--rm \
--network=host \
dockerhub.visionlabs.ru/luna/storages:v.0.2.51 \
bash -c "luna_prepare check \
    --platform_version=v.5.53.0 \
    --profile=common \
    --luna-config=http://10.16.5.177:5070"
```

The example checks the environment for all LUNA PLATFORM v.5.53.0 services, except for the Backport 3 and Backport 4 services, and also checks the availability of buckets using the corresponding settings "LUNA_IMAGE_STORE_bucket_name_ADDRESS" from the Configurator service launched at the address from the `--luna-config` argument.

If the installed environment matches the version specified in the `--platform_version` argument, then a message similar to this will appear in the logs:

```
[2024-02-06 13:57:01+0300] [storages] [INFO]: The surrounding state is
    completely prepared for platform version 'v.5.53.0'
```

If the installed environment does not match the version specified in the `--platform_version` argument, then a message similar to this will appear in the logs:

```
[2024-02-08 18:19:48+0300] [storages] [INFO]: luna_tasks[v.3.18.0] will be
    added database entity: tasks_database_migration with revision 75
    d45d56f7f7
[2024-02-08 18:19:48+0300] [storages] [INFO]: luna_tasks[v.3.18.0] will be
    updated configs migration version from '6a3d8839' to 'e7490433'
```

This means that when the prepare command is executed, the Tasks database will be migrated and the Configurator settings migration revision will be updated to the specified one.

3.3 Prepare command

The `prepare` command prepares the environment according to the specified version of LUNA PLATFORM.

The preparation of the environment is understood as follows:

- Preparing buckets in InfluxDB for monitoring work.
- Preparing buckets for the Image Store service, enabling you to store user data (images, metadata, archives, etc.).
- Preparing the InfluxDB for collection of aggregated statistics by the Admin service (see "Requests and estimations statistics gathering" section in the administrator manual).
- Preparing databases, adding VLMatch functionality, creating and managing migration scripts for LUNA PLATFORM databases (PostgreSQL or Oracle).
- Performing migration/loading settings into the Configurator database.

All of the above tasks are specified as separate command arguments (see below).

Note: If necessary, you can record environment preparation execution logs to solve specific problems that require contacting VisionLabs specialists. To do this, you need to specify additional data in the

environment preparation command. By default, logs are not saved. See for more information.

Important: The Storages utility will not prepare the environment for services that are disabled in the "ADDITIONAL_SERVICES_USAGE" setting.

3.3.1 Types of passed arguments

For the `prepare` command, you can pass additional arguments that enable you to configure the preparation of the environment in detail.

Arguments can be:

- Positional (mandatory)
- Named (optional, having a default value)

The positional argument must appear exactly after the `prepare` command. Named arguments appear after positional ones.

The table below provides a list of possible positional and named arguments for the `prepare` command.

Positional argument	Description
	Preparing entities
`lis_bucket`	Enables you to create buckets in the Image Store container. Bucket settings are specified in the "LUNA_IMAGE_STORE_bucket_name_ADDRESS" sections in the Storages utility settings (`--config`) or in the Configurator service settings (`--luna-config`). List of available named arguments: `--help`, `--verbose`, `--config`, `--luna-config`, `--ignore-integrity`, `--platform_version`, `--s3-buckets`, `--local-buckets`, `--profile`, `--ADDITIONAL_SERVICES_USAGE`, arguments for passing configuration tags with bucket addresses.
`s3_bucket`	Enables you to create buckets in S3 storage without accessing the Image Store service. Bucket settings are specified in the "S3" section, and bucket names in the "LUNA_IMAGE_STORE_bucket_name_ADDRESS" sections in the Storages utility settings (`--config`) or in the Configurator service settings (`--luna-config`). List of available named arguments: `--help`, `--verbose`, `--config`, `--luna-config`, `--ignore-integrity`, `--platform_version`, `--profile`, `--ADDITIONAL_SERVICES_USAGE`, `--LAMBDA_S3`.

Positional argument	Description
influx_bucket	Enables you to create buckets in the InfluxDB. List of available named arguments: --help, --verbose, --config, --luna-config, --ignore-integrity, --platform_version, --profile, --INFLUX_MONITORING.
aggregated_inf	Allows you to create a "luna_monitoring_aggregated" bucket in the Influx database and enable statistics collection (see section "Requests and estimations statistics gathering" in the administrator manual). List of available named arguments: --help, --verbose, --config, --luna-config, --ignore-integrity, --platform_version, --profile, --dry, --INFLUX_MONITORING.
database	Enables creating databases, adding VLMatch functionality, creating and managing migration scripts for LUNA PLATFORM databases (PostgreSQL or Oracle). List of available named arguments: --help, --verbose, --config, --luna-config, --ignore-integrity, --platform_version, --profile, --db-password, --db-user, --ADDITIONAL_SERVICES_USAGE, Arguments for passing database settings tags.
configs	Migrates Configurator service settings. List of available named arguments: --help, --verbose, --config, --luna-config, --ignore-integrity, --platform_version, --profile, --configs-revision.
all_entities	Uses positional arguments lis_bucket, influx_bucket, aggregated_influx_bucket, database and configs. List of available named arguments: all of the above and --dump-file.
	Specifying service for preparing environment using one positional argument from the list above
<service_name >	Name of the service (configurator, remote_sdk, etc.) for which you need to prepare the environment. The entity for preparing the environment is specified separately in the named argument --entity, available for each service. See the luna_prepare prepare <service> --help command for a list of available named arguments.

A description of all named arguments is given in the table "Named arguments".

Important: There is no positional argument that prepares the environment for all services. The profile (a link to a list of services) is specified in the named argument `--profile`, which is available for all positional arguments except `<service_name>`. For the positional argument `<service_name>` there is a separate logic for selecting the environment for preparation, specified in the `--entity` flag. If the named argument `--entity` is not specified, then an environment for all entities will be prepared.

> See the example of the `prepare` command in the "Upgrade environment scenario" section.

3.4 Load_dump command

The `load_dump` command enables you to load user settings into the Configurator service.

The following arguments are available for the `load_dump` command:

- –help
- –verbose
- –config
- –clear-database
- –dump-file

For example, you can load the supplied custom settings using the following command:

```
docker run \
--rm \
--network=host \
-v /var/lib/luna/current/extras/conf/platform_settings.json:/srv/
    platform_settings.json \
dockerhub.visionlabs.ru/luna/storages:v.0.2.51 \
bash -c "luna_prepare load_dump \
    -v \
    --dump-file=/srv/platform_settings.json"
```

Example of successful command execution:

```
[2024-02-07 16:25:33+0300] [storages] [INFO]: getting settings-dump file
[2024-02-07 16:25:33+0300] [storages] [INFO]: start updating settings
[2024-02-07 16:25:34+0300] [storages] [INFO]: update setting with name '
    ADDITIONAL_SERVICES_USAGE'
[2024-02-07 16:25:34+0300] [storages] [INFO]: update setting with name '
    LICENSE_VENDOR'
...
[2024-02-07 16:25:34+0300] [storages] [INFO]: applied settings: 20 out of 20
[2024-02-07 16:25:34+0300] [storages] [INFO]: database is ready to use
```

3.5 Logs command

The `logs` command enables you to get special logs in JSON format created during the environment preparation stage. Each log contains the following information about the action performed:

- `migration_action` — Performed action (`create`, `update`, `downgrade` or `delete`).
- `migration_datetime` — Time of execution of the action.
- `migration_problems` — Problems encountered.
- `migration_source` — Source (for example, the source revision version).
- `migration_target` — Target (for example, the revision version to which the migration was performed).
- `target_version` — Version of LUNA PLATFORM corresponding to the action being performed.

Not all log fields may be filled in (for example, `migration_problems': None`). Also, some logs may be missing if no problems occurred (for example, entity preparation logs influx_bucket).

> Receiving logs is optional. It is recommended to configure the receipt of logs in case of problems that require contacting VisionLabs specialists.

Logs are saved in a separate file in SQLite database format.

The following arguments are available for the `logs` command:

- –save-file
- –tail

To ensure that logs are recorded during environment preparation, the following is required:

- Create a directory to store the SQLite database file.
- Change the owner and group for the specified directory.
- In the environment preparation command, mount the created directory.
- In the environment preparation command, set the environment variable "DB_PATH", which will indicate the path to the mounted directory (by default "/srv/sqlite_db/data").

3.5.1 Example of Prepare command with logging enabled

Create a directory to store the SQLite database file:

```
mkdir -p /var/lib/luna/sqlite_db/data
```

Change the owner and group for the specified directory:

```
chown -R 1001:0 /var/lib/luna/sqlite_db/data
```

Prepare the environment and save logs to a SQLite database file:

```
docker run \
--rm \
--env=DB_PATH="/srv/my_data" \
-v /var/lib/luna/sqlite_db/data/:/srv/my_data \
--network=host \
dockerhub.visionlabs.ru/luna/storages:v.0.2.51 \
bash -c "luna_prepare prepare all_entities \
    --platform_version=v.5.57.0"
```

Here:

- `-v /root/my_data/:/srv/sqlite_db/data \` — Mount the directory for storing the SQLite database file.
- `--env=DB_PATH="/srv/my_data"\` — Specifying an environment variable containing the path to the directory with the SQLite database file inside the container.

3.5.2 Example of Logs command to get environment preparation logs

After preparing the environment, you can get logs using the following command:

```
docker run \
--rm \
--network=host \
--env=DB_PATH="/srv/sqlite_db/data" \
-v /var/lib/luna/sqlite_db/data/:/srv/sqlite_db/data \
dockerhub.visionlabs.ru/luna/storages:v.0.2.51 \
bash -c "luna_prepare logs \
    --save-file=/srv/sqlite_db/data/storages_logs.log \
    --tail=30"
```

Here:

- `--save-file=/srv/sqlite_db/data/storages_logs.log` — Address inside the container where the log file should be saved. In this case, the file will be saved in a mounted directory and will be available on the host machine.
- `--tail=30` — Number of lines with logs in the `storages_logs.log` file.

4 Named arguments

Named arguments are intended for detailed configuration of all commands for the `luna_prepare` script.

Each command has a specific set of named arguments available. A list of named arguments for each command can be obtained using the reference argument:

```
docker run \
--rm \
--network=host \
dockerhub.visionlabs.ru/luna/storages:v.0.2.51 \
bash -c "luna_prepare <command> <named_argument> --help"
```

> If the `prepare` command is used, then named arguments are specified after positional ones (see "Types of passed arguments" section).

The table below provides a description for all named arguments.

Named arguments	Description	Default value/behavior
`--help` or `-h`	Show help information.	None
`--verbose` or `-v`	Enable debug output.	Not used
`--config *`	The path to the `config.conf` configuration file of the Storages utility, which contains the settings necessary for the utility to prepare the environment.	`/srv/storages/ storages/ config`
`--luna-config *`	The address of the running Configurator service for reading the settings required by the Storages utility to prepare the environment.	`127.0.0.1:5070`
`--profile`	Profile: • `common` — Create environment for all LUNA PLATFORM services, with the exception of Backport 3 and Backport 4 services. • `backports` — Create environment for all LUNA PLATFORM services, including Backport 3 and Backport 4 services.	`common`
`--platform_vers`	LUNA PLATFORM version.	Latest version from the `list` command

`--s3-buckets`	Enables you to use direct queries to S3 instead of using local Image Store buckets.	Settings from the configuration
`--local-buckets`	Path to the local directory with buckets. The directory must be mounted to the Storages container.	Directory from the configuration
`----bucket-ttl`	Lifetime of objects in a bucket.	Not used
`--clear-database`	Remove all existing settings and restrictions from the Configurator database before adding them from the dump file.	Not used
`--entity`	Selecting an entity to prepare the environment of a separate service.	All entities
`--dump-file`	Path to the dump file with settings for Configurator. The dump file must be mounted to the Storages container.	—
`--configs-revision`	Revision of settings for performing migration. Accepts the following values: • Revision hash • head — Latest revision** • -1 — Previous revision relative to current	Revision for specified LP version
`--db-user`	DB user. Required to override the default user in settings.***	Not used
`--db-password`	DB password. Required to override the default password in settings.***	Not used
`--dry`	Argument that enables you to avoid modifying data while preparing aggregated InfluxDB buckets.	`false`
`--ignore-integrity` or `-ii`	Argument that determines whether errors in the existence of objects (databases, buckets, etc.) should be ignored. If the argument is disabled, the existence of objects will be treated as an error.	Not used
`--save-file`	Name of the file where the logs will be saved.	—
`--tail`	Number of logs.	10

* to get the Storages utility settings, you can use either the `--config` argument or the `--luna-config` argument. See "Setting Storages configuration" for details.

** latest revision does not always mean a revision for the latest version of LUNA PLATFORM. If you need a revision of the latest version of LP, then you don't have to specify the `--configs-revision` flag,

because the default value means using the revision corresponding to the LUNA PLATFORM version specified in the `--platform_version` flag.

*** named arguments can be used in the case when you need to create a database from one user and use it from another.

Also, when using the `--luna-config` argument, you can pass the following named arguments containing the configuration tag in the Configurator service:

- `--LUNA_LAMBDA_DB`
- `--LUNA_FACES_DB`
- `--LUNA_BACKPORT3_DB`
- `--LUNA_ACCOUNTS_DB`
- `--LUNA_TASKS_DB`
- `--LUNA_HANDLERS_DB`

- `--LUNA_EVENTS_DB`
- `--DATABASE_NUMBER`
- `--LUNA_IMAGE_STORE_FACES_SAMPLES_ADDRESS`
- `--LUNA_IMAGE_STORE_TASK_RESULT_ADDRESS`
- `--LUNA_IMAGE_STORE_IMAGES_ADDRESS`
- `--LUNA_IMAGE_STORE_PORTRAITS_ADDRESS`
- `--LUNA_IMAGE_STORE_OBJECTS_ADDRESS`
- `--LUNA_IMAGE_STORE_BODIES_SAMPLES_ADDRESS`
- `--LAMBDA_S3`
- `--INFLUX_MONITORING`
- `--ADDITIONAL_SERVICES_USAGE`

5 Environment upgrade scenario

For example, a user wants to upgrade from LUNA PLATFORM v.5.53.0 to LUNA PLATFORM v.5.57.0 and has four servers where old LUNA PLATFORM services are deployed:

- Server A with databases — PostgreSQL, InfluxDB and Redis databases.
- Server B with the Image Store service.
- Server C with the Licenses service.
- Server D with all other LUNA PLATFORM services.

In fact, all environment preparation can be done with one or two commands. The main thing is to correctly specify the necessary settings for the Storages utility. The Storages utility can receive settings from the running Configurator service or from a configuration file (see "Setting Storages configuration"). If it is possible to specify all the necessary settings in the configuration file, then executing one command with the argument –config will be enough. If all parameters for connecting to the database, services, etc. have already been set in the existing Configurator service and there is no desire to refill the Storages configuration, then to prepare the environment for all services, you can use the existing Configurator settings (argument –luna-config), having previously separately updated the environment for the Configurator service. In this example, we will first update the environment for the Configurator service, and then we will use all its settings to update the environment of the remaining services.

To prepare the environment for the Configurator service, you need to perform the following steps on any of the servers:

- Configure the "LUNA_CONFIGURATOR_DB" section to obtain the Configurator DB address in the Storages configuration file:

```
vi /var/lib/luna/current/extras/conf/storages_config.conf
```

By default, the configuration file specifies the location of the Configurator database at 127.0.0.1. In our example, PostgreSQL and Configurator are located on different servers, so you need to explicitly set the database address. If Configurator and PostgreSQL were on the same server, then it would be possible not to edit the configuration file and use the default one inside the container without passing the `--config` argument.

If necessary, you can set the environment preparation settings for all services at once in the Storages configuration file and immediately proceed to preparing the environment for all services. This example specifically reflects both options for using settings.

- Migrate the Configurator service database:

```
docker run \
--rm \
```

```
--network=host \
- -v /var/lib/luna/current/extras/conf/storages_config.conf:/srv/
    storages_config.conf \
dockerhub.visionlabs.ru/luna/storages:v.0.2.51 \
bash -c "luna_prepare prepare configurator \
    --entity=database \
    --platform_version=v.5.57.0 \
    --config=/srv/storages_config.conf"
```

> Note that when using a configuration file other than the default one, it must be mounted to the Storages container.

- Migrate Configurator service settings:

```
docker run \
--rm \
--network=host \
- -v /var/lib/luna/current/extras/conf/storages_config.conf:/srv/
    storages_config.conf \
dockerhub.visionlabs.ru/luna/storages:v.0.2.51 \
bash -c "luna_prepare prepare configurator \
    --entity=configs \
    --platform_version=v.5.57.0 \
    --config=/srv/storages_config.conf"
```

> If necessary, you can run both of the above commands as one command with the argument `--entity=all_entities`, because `all_entities` for the Configurator service means using the `database`, `configs` and `influx_bucket` entities. However, the InfluxDB bucket (the last entity) will be prepared in the `prepare all_entities --profile=backports` command below, also that there is no point in preparing it at this stage. If you nevertheless prepare it at this stage, then in the environment preparation command for all services it will be re-created, but only if the named argument `--ii` is specified (whether existing objects should be ignored).

- Stop the old version of Configurator and start the new version of the service.

Now that there is a current version of the Configurator service with the current environment, you can use its settings when preparing the environment for other services.

The environment preparation command looks like this:

```
docker run \
--rm \
--network=host \
dockerhub.visionlabs.ru/luna/storages:v.0.2.51 \
```

```
bash -c "luna_prepare prepare all_entities \
    --platform_version=v.5.57.0 \
    --profile=backports \
    --luna-config=<configurator_address>"
```

> The environment preparation command can be launched from any server if all addresses are specified correctly in the Configurator settings.

This command will do the following on the respective servers using the Configurator settings on server D:

- Creates buckets (if they are missing) in InfluxDB for monitoring work on server A (positional argument `influx_bucket`).
- Creates buckets (if they are missing) in the Image Store service on server D (positional argument `lis_bucket`).
- Will prepare the InfluxDB (if it has not been prepared previously) for collecting aggregated statistics by the Admin service on server A (positional argument `aggregated_influx_bucket`).
- Will migrate the databases on server A (positional argument `database`).
- Will migrate settings to the Configurator database on server D (positional argument `configs`).

> Migration of settings to the Configurator database will not be performed in this case, because they have already been migrated at the stage of preparing the environment for the Configurator service.

> The Storages utility will check for the presence of buckets in the launched Image Store service in accordance with the Configurator settings. If the Image Store service is not launched, you can explicitly specify the location of the buckets using the `--local-buckets` command.

Example of command execution:

```
[2024-02-07 15:05:40+0300] [storages] [INFO]: The surrounding state was
    prepared according to platform version 'v.5.57.0'
[2024-02-07 15:05:40+0300] [storages] [INFO]: Actual services versions:
[2024-02-07 15:05:40+0300] [storages] [INFO]: Service `luna_accounts`
    version - `v.0.2.6`
[2024-02-07 15:05:40+0300] [storages] [INFO]: Service `luna_admin` version -
    `v.5.5.6`
...
```

After executing this command, you need to stop all LUNA PLATFORM services on the corresponding servers, update the settings in the Configurator (optional) and launch their new versions.