VisionLabs
MACHINES CAN SEE

VisionLabs LUNA PLATFORM 5

Руководство по утилите Storages

v.5.94.0

Содержание

Введение **3**

1 Сценарии применения Storages **4**

1.1 Запуск с нуля . . . 4

1.2 Обновление/даунгрейд . . . 5

1.2.1 Рекомендации по поведению сервисов во время подготовки окружения . . . 5

2 Настройка конфигурации Storages **7**

3 Команды утилиты **8**

3.1 Команда list . . . 8

3.2 Команда check . . . 9

3.3 Команда prepare . . . 10

3.3.1 Типы передаваемых аргументов . . . 11

3.4 Задание пользовательской схемы для баз данных . . . 13

3.5 Команда load_dump . . . 14

3.6 Команда logs . . . 15

3.6.1 Пример команды подготовки окружения с записью логов . . . 15

3.6.2 Пример команды получения логов подготовки окружения . . . 16

4 Именованные аргументы **18**

5 Сценарий обновления окружения **21**

Введение

В данном документе приводится описание утилиты Storages и способы её использования. Рекомендуется изучить данный документ перед использованием утилиты.

Утилита Storages позволяет проверить и/или подготовить окружение для сервисов LUNA PLATFORM перед их непосредственным запуском. В качестве подготовки окружения понимается следующее:

- Подготовка бакетов в InfluxDB для работы мониторинга
- Подготовка бакетов для сервиса Image Store, позволяющих хранить пользовательские данные (изображение, метаданные, архивы и пр.)
- Подготовка БД Influx для сбора агрегированной статистики сервисом Admin (см. раздел «Подсчет статистики выполненных запросов и оценок» в руководстве администратора)
- Подготовка баз данных, добавление функций VLMatch, создание сценариев миграции для баз данных LUNA PLATFORM (PostgreSQL или Oracle) и управление ими
- Выполнение миграции/загрузка настроек в БД Configurator
- Загрузка дамп-файлов в БД сервиса Configurator

Основное преимущество Storages перед ручной подготовкой окружения в том, что утилите известны ревизии настроек сервиса Configurator и сценарии миграции баз данных сервисов LP, что значительно упрощает процесс обновления и даунгрейда LUNA PLATFORM. Кроме того, с помощью утилиты можно оценить текущее состояние окружения LUNA PLATFORM и определить что именно будет нужно обновить.

Утилита поставляется в Docker-контейнере и может быть использована в качестве инструмента подготовки окружения LUNA PLATFORM, разворачиваемой в Docker-контейнерах, скриптом Docker Compose, системой оркестрации Kubernetes и др. Основная задача администратора — указать утилите Storages адреса баз данных, бакетов и пр. (см. раздел «Настройка конфигурации Storages»).

1 Сценарии применения Storages

Утилиту Storages можно использовать как для обновления или даунгрейда, так и для установки с нуля.

> Рекомендуется создавать бекапы перед любыми действиями, связанными с обновлением или даунгрейдом. См. рекомендации к созданию бекапов и полезные ссылки в руководстве по обновлению.

Примеры команд для обновления и установки с нуля приведены в соответствующих документах. Даунгрейд фактически не отличается от обновления.

В разделах ниже приведен общий порядок действий при выполнении соответствующих задач.

1.1 Запуск с нуля

Общий порядок действий при установке с нуля следующий:

1. Запустить и настроить СУБД PostgreSQL/Oracle и InfluxDB — создать пользователей, настроить права и т.д (см. раздел «Базы данных» в руководстве администратора).
2. Скомпилировать библиотеку VLMatch для пользовательской версии БД PostgeSQL или Oracle (см. раздел «Базы данных» в руководстве администратора). Утилита Storages автоматически добавляет функцию VLMatch в базы данных Faces и Events на этапе подготовки окружения баз данных.
3. Корректно заполнить конфигурацию Storages, указав все необходимые данные — адреса и аутентификационные данные БД, настройки подключения к S3 (при необходимости), адреса бакетов и пр.
4. Подготовить окружение с помощью команды prepare.
5. Убедиться, что окружение интересующей версии успешно подготовлено с помощью команды check.
6. Корректно заполнить пользовательские настройки сервисов (соединение с сервисами, базами данных и пр.) одним из следующих способов:
   - Выполнить команду load_dump, загрузив дамп-файл с пользовательскими настройками *
   - Настроить и запустить сервис Configurator, указав в нем пользовательские настройки
   - Заполнить конфигурационные файлы сервисов, указав в них пользовательские настройки
7. Запустить сервисы LUNA PLATFORM.

* можно отредактировать полный дамп-файл `luna_platform_<version>_dump.json` из комплекта поставки, либо использовать пользовательский дамп-файл (см. пример пользовательского дамп-файла `platform_settings.json` в комплекте поставки). См. подробную информацию про дампы в разделе «Сервис Configurator» в руководстве администратора

1.2 Обновление/даунгрейд

Общий порядок действий при обновлении или понижении версии следующий:

1. Корректно заполнить конфигурацию Storages, указав все необходимые данные — адреса и аутентификационные данные БД, настройки подключения к S3 (при необходимости), адреса бакетов и пр.
2. Убедиться, что Storages поддерживает подготовку окружения для LUNA PLATFORM интересующей версии с помощью команды list.
3. Проверить текущее окружение LUNA PLATFORM с помощью команды check.
4. Настроить поведение сервисов во время подготовки окружения
5. Подготовить окружение с помощью команды prepare.
6. Убедиться, что окружение интересующей версии успешно подготовлено с помощью команды check.
7. Актуализировать пользовательские настройки сервисов (соединение с сервисами, базами данных и пр.) одним из следующих способов (опционально):
    - Выполнить команду load_dump, загрузив дамп-файл с пользовательскими настройками *
    - Настроить и запустить сервис Configurator, указав в нем пользовательские настройки
    - Заполнить конфигурационные файлы сервисов, указав в них пользовательские настройки
8. Запустить сервисы LUNA PLATFORM новой версии.

* можно отредактировать полный дамп-файл `luna_platform_<version>_dump.json` из комплекта поставки, либо использовать пользовательский дамп-файл (см. пример пользовательского дамп-файла `platform_settings.json` в комплекте поставки). См. подробную информацию про дампы в разделе «Сервис Configurator» в руководстве администратора

1.2.1 Рекомендации по поведению сервисов во время подготовки окружения

В ходе подготовки окружения важно тщательно планировать воздействие на сервисы, использующие базы данных.

Ниже представлены рекомендации по управлению сервисами в различных сценариях миграции.

- **Снижение нагрузки:** Попробуйте сократить количество запросов до минимума во время подготовки окружения. Это поможет уменьшить вероятность возникновения конфликтов и проблем с целостностью данных.
- **Ограниченный доступ:** Рассмотрите возможность временного ограничения доступа к сервисам для избежания несогласованных изменений во время миграции.
- **Остановка сервисов:** Рассмотрите вариант остановки сервисов, активно взаимодействующих с БД. Это может способствовать уменьшению возможных конфликтов и упростить про-

цесс миграции.

- **Осторожное обращение:** Если оставляете сервис работающим во время миграции, будьте готовы к возможным задержкам в работе LUNA PLATFORM.

Конечное решение о выборе способа управления сервисами должно быть принято на основе конкретных требований вашей системы и бизнес-процессов.

2 Настройка конфигурации Storages

Для корректной подготовки окружения необходимо настроить конфигурацию Storages так, чтобы утилита имела возможность взаимодействовать с различными сервисами, базами данных, бакетами и другими ресурсами. Необходимые настройки Storages можно указать с помощью:

- конфигурационного файла сервиса Storages
- запущенного сервиса Configurator (при обновлении/даунгрейде)

Способ передачи настроек задается с помощью аргументов соответствующих команд (см. ниже). Если никакой аргумент не задан, то утилита Storages будет использовать конфигурационный файл по умолчанию. Если заданы оба аргумента, то приоритет будет отдан получению настроек из запущенного сервиса Configurator.

Использование конфигурационного файла Storages

По умолчанию конфигурационный файл расположен в контейнере Storages. При необходимости задания настроек, отличных от настроек по умолчанию, можно изменить копию дефолтного конфигурационного файла, расположенного в комплекте поставки по пути `luna_v.5.94.0/extras/conf/storages_config.conf`, и указать его с помощью аргумента –config. Конфигурационный файл нужно также примонтировать к контейнеру Storages.

Такой способ указан в качестве примера в руководствах по установке и обновлению с помощью Storages.

Использование настроек из запущенного сервиса Configurator

Адрес запущенного сервиса Configurator передается в аргументе –luna-config соответствующей команды.

Такой способ требует предварительной подготовки окружения для новой версии сервиса Configurator. Пример использования данного способа приведен в разделе «Сценарий обновления окружения».

3 Команды утилиты

Работа по подготовке окружения выполняется с помощью скрипта `luna_prepare`, которому передаются определенные команды, позволяющие детально настроить процесс подготовки.

Список доступных команд:

- check — команда, позволяющая проверить имеющееся окружение в соответствии с указанными сведениями о версии LUNA PLATFORM
- list — команда, позволяющая показать список доступных версий LUNA PLATFORM для подготовки окружения
- prepare — **основная команда**, позволяющая подготовить окружение в соответствии с указанными сведениями о версии LUNA PLATFORM
- load_dump — команда, позволяющая загрузить настройки из пользовательского дамп-файла в БД Configurator
- logs — команда, позволяющая отобразить или сохранить в файл специальные логи, полученные на этапе подготовки окружения

Для каждой команды кроме команды `list` есть свои аргументы. Список возможных аргументов с их базовым описанием можно получить с помощью справочного аргумента `--help`.

> Расширенное описание аргументов приведено в разделе «Именованные аргументы».

Кроме того, в команде `prepare` есть два типа аргументов — позиционные и именованные (см. раздел «Команда prepare») и для каждого типа аргумента также можно получить справочную информацию.

Например, можно получить список аргументов для команды `load_dump` с помощью следующей команды:

```
docker run \
--rm \
--network=host \
dockerhub.visionlabs.ru/luna/storages:v.0.4.94 \
bash -c "luna_prepare load_dump --help"
```

Важно! Крайне рекомендуется пользоваться справочными аргументами во время работы с утилитой Storages.

3.1 Команда list

Утилита Storages работает не со всеми версиями LUNA PLATFORM. Команда `list` позволяет получить список поддерживаемых версий выполнения подготовки окружения.

Важно! Список версий также содержит релизы, предназначенные для внутреннего использования. Следует использовать только публичные версии LUNA PLATFORM.

Пример команды получения списка версий:

```
docker run \
--rm \
--network=host \
dockerhub.visionlabs.ru/luna/storages:v.0.4.94 \
bash -c "luna_prepare list"
```

Пример успешного выполнения команды:

```
[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 Команда check

Команда check позволяет проверить текущее окружение и получить информацию о том, что нужно для установки окружения другой версии. С помощью данной команды, например, можно убедиться действительно ли у пользователя установлено окружение определенной версии.

Для команды check доступны следующие аргументы:

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

Пример команды проверки текущего окружения:

```
docker run \
--rm \
--network=host \
dockerhub.visionlabs.ru/luna/storages:v.0.4.94 \
bash -c "luna_prepare check \
    --platform_version=v.5.53.0 \
    --profile=common \
```

```
      --luna-config=http://10.16.5.177:5070"
```

В примере выполняется проверка окружения для всех сервисов LUNA PLATFORM v.5.53.0, кроме сервисов Backport 3 и Backport 4, а также проверка наличия бакетов по соответствующим настройкам «LUNA_IMAGE_STORE_bucket_name_ADDRESS» из сервиса Configurator, запущенного по адресу из аргумента `--luna-config`.

Если установленное окружение подходит под версию, указанную в аргументе `--platform_version`, то в логах появится сообщение подобного содержания:

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

Если установленное окружение не подходит под версию, указанную в аргументе `--platform_version`, то в логах появится сообщение подобного содержания:

```
[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'
```

Это означает, что при выполнении команды prepare будет выполнена миграция БД Tasks и обновлена ревизия миграции настроек Configurator на указанную.

3.3 Команда prepare

Prepare — это команда, которая подготавливает окружение в соответствии с указанной версией LUNA PLATFORM.

В качестве подготовки окружения понимается следующее:

- Подготовка бакетов в InfluxDB для работы мониторинга
- Подготовка бакетов для сервиса Image Store, позволяющих хранить пользовательские данные (изображение, метаданные, архивы и пр.)
- Подготовка БД Influx для сбора агрегированной статистики сервисом Admin (см. раздел «Подсчет статистики выполненных запросов и оценок» в руководстве администратора)
- Подготовка баз данных, добавление функций VLMatch, создание сценариев миграции для баз данных LUNA PLATFORM (PostgreSQL или Oracle) и управление ими
- Выполнение миграции/загрузка настроек в БД Configurator
- Загрузка дамп-файлов в БД сервиса Configurator

Все вышеописанные задачи указываются в качестве отдельных аргументов команды (см. ниже).

Примечание. При необходимости можно записывать логи выполнения подготовки окружения для решения специфических проблем, требующих обращения к специалистам VisionLabs. Для этого необходимо указать дополнительные данные в команде подготовки окружения. По умолчанию логи не сохраняются. См. раздел «Команда logs» для более подробной информации.

Важно! Утилита Storages не будет выполнять подготовку окружения для сервисов, которые отключены в настройке «ADDITIONAL_SERVICES_USAGE».

3.3.1 Типы передаваемых аргументов

Для команды `prepare` можно передать дополнительные аргументы, позволяющие детально настроить подготовку окружения.

Аргументы могут быть:

- позиционные (обязательные)
- именованные (необязательные, имеющие значение по умолчанию)

Позиционный аргумент должен находиться строго после команды `prepare`. Именованные аргументы указываются после позиционных.

В таблице ниже представлен список возможных позиционных и именованных аргументов для команды `prepare`.

Позиционный аргумент	Описание
	Подготовка сущностей
`lis_bucket`	Позволяет создать бакеты в контейнере Image Store. Настройки бакетов указываются в секциях «LUNA_IMAGE_STORE_bucket_name_ADDRESS» в настройках утилиты Storages (`--config`) или в настройках сервиса Configurator (`--luna-config`). Список доступных именованных аргументов: `--help`, `--verbose`, `--config`, `--luna-config`, `--ignore-integrity`, `--platform_version`, `--s3-buckets`, `--local-buckets`, `--profile`, `--ADDITIONAL_SERVICES_USAGE`, аргументы для передачи тегов настроек с адресами бакетов.

Позиционный аргумент	Описание
s3_bucket	Позволяет создать бакеты в хранилище S3 без обращения к сервису Image Store. Настройки бакетов указываются в секции «S3», а названия бакетов в секциях «LUNA_IMAGE_STORE_bucket_name_ADDRESS» в настройках утилиты Storages (--config) или в настройках сервиса Configurator (--luna-config). Список доступных именованных аргументов: --help, --verbose, --config, --luna-config, --ignore-integrity, --platform_version, --profile, --ADDITIONAL_SERVICES_USAGE, --LAMBDA_S3.
influx_bucket	Позволяет создать бакеты в БД Influx. Список доступных именованных аргументов: --help, --verbose, --config, --luna-config, --ignore-integrity, --platform_version, --profile, --LUNA_MONITORING.
aggregated_inf	Позволяет создать бакет «luna_monitoring_aggregated» в БД Influx и включить сбор статистики (см. раздел «Подсчет статистики выполненных запросов и оценок» в руководстве администратора). Список доступных именованных аргументов: --help, --verbose, --config, --luna-config, --ignore-integrity, --platform_version, --profile, --dry, --LUNA_MONITORING.
database	Обеспечивает создание баз данных, добавление функций VLMatch, создание сценариев миграции для баз данных LUNA PLATFORM (PostgreSQL или Oracle) и управление ими. Список доступных именованных аргументов: --help, --verbose, --config, --luna-config, --ignore-integrity, --platform_version, --profile, --db-password, --db-user, --ADDITIONAL_SERVICES_USAGE, аргументы для передачи тегов настроек баз данных.
configs	Выполняет миграцию настроек сервиса Configurator. Список доступных именованных аргументов: --help, --verbose, --config, --luna-config, --ignore-integrity, --platform_version, --profile, --configs-revision.
all_entities	Использует позиционные аргументы lis_bucket, influx_bucket, aggregated_influx_bucket, database и configs. Список доступных именованных аргументов: все вышеперечисленные и --dump-file.

Позиционный аргумент	Описание
	Указание сервиса для подготовки окружения по одному позиционному аргументу из списка выше
`<service_name>`	Название сервиса (`configurator`, `remote_sdk` и пр.) для которого нужно подготовить окружение. Сущность для подготовки окружения задается отдельно в именованном аргументе `--entity`, доступном для каждого сервиса. См. список доступных именованных аргументов в команде `luna_prepare prepare <service> --help`.

Описание всех именованных аргументов приведено в таблице «Именованные аргументы».

Важно! Не существует позиционного аргумента, подготавливающего окружение для всех сервисов. Профиль (ссылка на список сервисов) задается в именованном аргументе `--profile`, который доступен для всех позиционных аргументов кроме `<service_name>`. Для позиционного аргумента `<service_name>` отдельная логика выбора окружения для подготовки, указываемая во флаге `--entity`. Если именованный аргумент `--entity` не задан, то будет подготовлено окружение для всех сущностей.

> См. пример команды `prepare` в разделе «Сценарий обновления окружения».

3.4 Задание пользовательской схемы для баз данных

Пользовательскую схему для баз данных можно задать с помощью переменной окружения `LUNA_PG_SCHEMA`.

`LUNA_PG_SCHEMA` задает имя схемы PostgreSQL при создании таблицы. Значение по умолчанию - `luna`. Если значение `LUNA_PG_SCHEMA` отсутствует, то эта схема по умолчанию будет использоваться для новых таблиц, что делает ее схемой по умолчанию для соответствующего пользователя.

Важно! Если имя схемы отличается от имени пользователя (имя пользователя `luna` задается в команде запуска для PostgreSQl), таблица будет создана в схеме `public`.

Пример команды для назначения имени схемы:

```
docker run \
--rm \
--network=host \
--env=LUNA_PG_SCHEMA=custom_schema \
```

```
  -v /var/lib/luna/current/extras/conf/storages_config.conf:/srv/
     storages_config.conf \
  dockerhub.visionlabs.ru/luna/storages:v.0.4.94 \
  bash -c "luna_prepare prepare all_entities \
      --platform_version=v.5.94.0"
```

Здесь:

- `--env=LUNA_PG_SCHEMA=custom_schema` — указание переменной окружения `LUNA_PG_SCHEMA`, содержащей имя схемы «custom_schema».

3.5 Команда load_dump

Команда `load_dump` позволяет загрузить пользовательские настройки в сервис Configurator.

Для команды `load_dump` доступны следующие аргументы:

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

Например, можно загрузить пользовательские настройки из комплекта поставки с помощью следующей команды:

```
  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.0.4.94 \
  bash -c "luna_prepare load_dump \
      -v \
      --dump-file=/srv/platform_settings.json"
```

Пример успешного выполнения:

```
  [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
```

3.6 Команда logs

Команда `logs` позволяет получить специальные логи в JSON-формате, созданные на этапе подготовки окружения. Каждый лог содержит следующую информацию о выполненном действии:

- `migration_action` — выполненное действие (`create`, `update`, `downgrade` или `delete`)
- `migration_datetime` — время выполнения действия
- `migration_problems` — возникшие проблемы
- `migration_source` — источник (например, исходная версия ревизии)
- `migration_target` — цель (например, версия ревизии, на которую была выполнена миграция)
- `target_version` — версия LUNA PLATFORM, соответствующая выполняемому действию

Не все поля лога могут быть заполнены (например, `migration_problems': None`). Также некоторые логи могут отсутствовать, если не возникло никаких проблем (например, логи подготовки сущности influx_bucket).

> Получение логов имеет необязательный характер. Рекомендуется настраивать получение логов в случае возникновения проблем, которые требуют обращения к специалистам VisionLabs.

Логи сохраняются в отдельный файл формата БД SQLite.

Для команды `logs` доступны следующие аргументы:

- –save-file
- –tail

Для обеспечения записи логов во время подготовки окружения, требуется:

- создать директорию для хранения файла БД SQLite
- изменить владельца и группу для указанной директории
- в команде подготовки окружения примонтировать созданную директорию
- в команде подготовки окружения установить переменную окружения «DB_PATH», которая будет указывать путь к примонтированной директории (по умолчанию «/srv/sqlite_db/data»)

3.6.1 Пример команды подготовки окружения с записью логов

Создайте директорию для хранения файла БД SQLite:

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

Измените владельца и группу для указанной директории:

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

Выполните подготовку окружения с сохранением логов в файл БД SQLite:

```
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.0.4.94 \
bash -c "luna_prepare prepare all_entities \
    --platform_version=v.5.57.0"
```

Здесь:

- `-v /root/my_data/:/srv/sqlite_db/data \` — монтирование директории для хранения файла БД SQLite
- `--env=DB_PATH="/srv/my_data"\` — указание переменной окружения, содержащей путь до директории с файлом БД SQLite внутри контейнера

3.6.2 Пример команды получения логов подготовки окружения

После подготовки окружения можно получить логи с помощью следующей команды:

```
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.0.4.94 \
bash -c "luna_prepare logs \
    --save-file=/srv/sqlite_db/data/storages_logs.log \
    --tail=30"
```

Здесь:

- `--save-file=/srv/sqlite_db/data/storages_logs.log` — адрес внутри контейнера, куда должен сохраниться файл с логами. В данном случае файл сохранится в примонтированную директорию и будет доступен на хост-машине.

- `--tail=30` — количество строк с логами в файле `storages_logs.log`.

4 Именованные аргументы

Именованные аргументы предназначены для детальной настройки всех команд для скрипта `luna_prepare`.

Для каждой команды доступен определенный набор именованных аргументов. Список именованных аргументов для каждой команды можно получить с помощью справочного аргумента:

```
docker run \
--rm \
--network=host \
dockerhub.visionlabs.ru/luna/storages:v.0.4.94 \
bash -c "luna_prepare <command> <named_argument> --help"
```

> Если используется команда `prepare`, то именованные аргументы задаются после позиционных (см. раздел «Типы передаваемых аргументов»).

В таблице ниже приведено описание для всех именованных аргументов.

Именованные аргументы	Описание	Значение/поведение по умолчанию
`--help` или `-h`	Показать справочную информацию.	•
`--verbose` или `-v`	Включить вывод отладки.	Не используется
`--config` *	Путь до конфигурационного файла `config.conf` утилиты Storages, содержащего настройки, необходимые утилите для выполнения подготовки окружения.	`/srv/storages/storages/config`
`--luna-config` *	Адрес запущенного сервиса Configurator для считывания настроек, необходимых утилите Storages для выполнения подготовки окружения.	`127.0.0.1:5070`
`--profile`	Профиль: • `common` — позволяет создать окружение для всех сервисов LUNA PLATFORM, за исключением сервисов Backport 3 и Backport 4 • `backports` — позволяет создать окружение для всех сервисов LUNA PLATFORM, включая сервисы Backport 3 и Backport 4.	`common`

`--platform_vers`	Версия LUNA PLATFORM.	Последняя версия из списка команды `list`
`--s3-buckets`	Позволяет использовать прямые запросы к S3 вместо использования локальных бакетов Image Store.	Настройки из конфигурации
`--local-buckets`	Путь до локальной директории с бакетами. Директория должна быть примонтирована к контейнеру Storages.	Директория из конфигурации
`--bucket-ttl`	Время жизни объектов в бакете.	Не используется
`--clear-database`	Удалить все существующие настройки и ограничения из базы данных Configurator перед добавлением их из дамп-файла.	Не используется
`--entity`	Выбор сущности для подготовки окружения отдельного сервиса.	Все сущности
`--dump-file`	Путь до дамп-файла с настройками для Configurator. Дамп-файл должен быть примонтирован к контейнеру Storages.	•
`--configs-revision`	Ревизия настроек для выполнения миграции. Принимает следующие значения: • хеш ревизии • `head` — самая последняя ревизия** • `-1` — предыдущая относительно текущей	
`--db-user`	Пользователь БД. Требуется для переопределения пользователя по умолчанию в настройках.***	Не используется
`--db-password`	Пароль БД. Требуется для переопределения пароля по умолчанию в настройках.***	Не используется
`--dry`	Аргумент, позволяющий не модифицировать данные во время подготовки агрегированных бакетов InfluxDB.	`false`
`--ignore-integrity` или `-ii`	Аргумент, определяющий следует ли игнорировать ошибки существования объектов (баз данных, бакетов и пр.). Если аргумент отключен, то существование объектов будет расцениваться как ошибка.	Не используется
`--save-file`	Имя файла, куда будут сохраняться логи.	•

`--tail`	Количество логов.	10

* для получения настроек утилиты Storages можно использовать либо аргумент `--config`, либо аргумент `--luna-config`. См. подробную информацию в разделе «Настройка конфигурации Storages».

** последняя ревизия не всегда означает ревизию для последней версии LUNA PLATFORM. Если нужна ревизия последней версии LP, то можно не указывать флаг `--configs-revision`, т.к. значение по умолчанию означает использование ревизии, соответствующей версии LUNA PLATFORM, указываемой во флаге `--platform_version`.

*** именованные аргументы могут быть использованы в случае, когда нужно создать БД от одного пользователя, а использовать от другого

Также при использовании аргумента `--luna-config` можно передать следующие именованные аргументы, содержащие тег настройки в сервисе Configurator:

- `--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`
- `--LUNA_MONITORING`
- `--ADDITIONAL_SERVICES_USAGE`

5 Сценарий обновления окружения

Например, пользователь хочет обновиться с версии LUNA PLATFORM v.5.53.0 на версию LUNA PLATFORM v.5.57.0 и имеет четыре сервера, где развернуты старые сервисы LUNA PLATFORM:

- сервер A с базами данных — СУБД PostgreSQL, InfluxDB и Redis
- сервер B с сервисом Image Store
- сервер C с сервисом Licenses
- сервер D со всеми остальными сервисами LUNA PLATFORM

Фактически всю подготовку окружения можно выполнить с помощью одной или двух команд. Главное — правильно указать необходимые настройки для утилиты Storages. Утилита Storages может получить настройки из запущенного сервиса Configurator или из конфигурационного файла (см. «Настройка конфигурации Storages»). Если есть возможность указать все необходимые настройки в конфигурационном файле, то достаточно будет выполнения одной команды с указанием аргумента –config. Если все параметры соединения с БД, сервисами и пр. уже заданы в имеющемся сервисе Configurator и нет желания повторно заполнять конфигурацию Storages, то для подготовки окружения всех сервисов можно использовать имеющиеся настройки Configurator (аргумент –luna-config), предварительно отдельно обновив окружение для сервиса Configurator. В данном примере мы сначала обновим окружение для сервиса Configurator, а затем будем использовать все его настройки для обновления окружения остальных сервисов.

Для подготовки окружения для сервиса Configurator, нужно выполнить следующие действия на любом из серверов:

- Настроить группу параметров «LUNA_CONFIGURATOR_DB» для получения адреса БД Configurator в конфигурационном файле Storages:

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

> По умолчанию в конфигурационном файле указано расположение БД Configurator по адресу 127.0.0.1. В нашем примере PostgreSQL и Configurator расположены на разных серверах, поэтому необходимо явно задать адрес базы данных. Если бы Configurator и PostgreSQL были на одном сервере, то можно было бы не редактировать конфигурационный файл и использовать дефолтный внутри контейнера, не передавая аргумент `--config`.

> При необходимости можно задать настройки подготовки окружения для всех сервисов сразу в конфигурационном файле Storages и сразу перейти к подготовке окружения всех сервисов. Данный пример специально отражает оба варианта использования настроек.

- Выполнить миграцию БД сервиса Configurator:

```
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.0.4.94 \
bash -c "luna_prepare prepare configurator \
    --entity=database \
    --platform_version=v.5.57.0 \
    --config=/srv/storages_config.conf"
```

> Обратите внимание, что при использовании конфигурационного файла, отличного от дефолтного, его нужно примонтировать к контейнеру Storages.

- Выполнить миграцию настроек сервиса Configurator:

```
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.0.4.94 \
bash -c "luna_prepare prepare configurator \
    --entity=configs \
    --platform_version=v.5.57.0 \
    --config=/srv/storages_config.conf"
```

> При необходимости можно выполнить обе вышеописанные команды как одну команду с аргументом `--entity=all_entities`, т.к. `all_entities` для сервиса Configurator означают использование сущностей `database`, `configs` и `influx_bucket`. Однако бакет InfluxDB (последняя сущность) будет подготовлен в команде `prepare all_entities --profile=common` ниже, также что нет смысла его подготавливать на данном этапе. Если же все-таки подготовить его на данном этапе, то в команде подготовки окружения для всех сервисов он будет создан повторно, но только если указан именованный аргумент `--ii` (следует ли игнорировать существующие объекты).

- Остановить старую версию Configurator и запустить новую версию сервиса.

Теперь когда есть актуальная версия сервиса Configurator с актуальным окружением, то можно использовать его настройки при подготовке окружения для остальных сервисов.

Команда подготовки окружения выглядит следующим образом:

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

```
  --network=host \
  dockerhub.visionlabs.ru/luna/storages:v.0.4.94 \
  bash -c "luna_prepare prepare all_entities \
      --platform_version=v.5.57.0 \
      --profile=common \
      --luna-config=<configurator_address>"
```

> Команду подготовки окружения можно запустить с любого сервера, если в настройках Configurator все адреса указаны правильно.

Данная команда выполнит следующие действия на соответствующих серверах, используя настройки Configurator на сервере D:

- Создаст бакеты (если они отсутствуют) в InfluxDB для работы мониторинга на сервере A (позиционный аргумент `influx_bucket`)
- Создаст бакеты (если они отсутствуют) в сервисе Image Store на сервере D (позиционный аргумент `lis_bucket`)
- Подготовит БД Influx (если ранее не была подготовлена) для сбора агрегированной статистики сервисом Admin на сервере A (позиционный аргумент `aggregated_influx_bucket`)
- Выполнит миграцию баз данных на сервере A (позиционный аргумент `database`)
- Выполнит миграцию настроек в БД Configurator на сервере D (позиционный аргумент `configs`)

> Миграция настроек в БД Configurator в данном случае не будет выполнена, т.к. они уже были мигрированы на этапе подготовки окружения для сервиса Configurator.

> Утилита Storages будет проверять наличие бакетов в запущенном сервисе Image Store в соответствии с настройками Configurator. Если сервис Image Store не запущен, то можно явно указать расположение бакетов с помощью команды `--local-buckets`.

Пример выполнения команды:

```
  [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`
  ...
```

После выполнения данной команды нужно остановить все сервисы LUNA PLATFORM на соответствующих серверах, актуализировать настройки в Configurator (опционально) и запустить их новые версии.