invenio-cli Documentation

Release 1.0.21

CERN

May 19, 2023

CONTENTS

Command-line tool to create and manage an InvenioRDM instance.

CHAPTER

ONE

INSTALLATION

```
$ pip install invenio-cli
```

CHAPTER

TWO

USAGE

2.1 Local Development environment

```
# Initialize environment and cd into <created folder>
$ invenio-cli init rdm
$ cd <created folder>

# Install locally
# install python dependencies (pre-release versions needed for now),
# link/copy assets + statics, install js dependencies, build assets and
# final statics
$ invenio-cli install --pre

# Start and setup services (database, Elasticsearch, Redis, queue)
$ invenio-cli services

# Optional: add demo data
$ invenio-cli demo --local

# Run the server
$ invenio-cli run

# Update assets or statics
$ invenio-cli update
```

2.2 Containerized 'Production' environment

```
# Initialize environment and cd into <created folder>
$ invenio-cli init rdm
$ cd <created folder>

# Spin-up InvenioRDM
$ invenio-cli containerize

# Optional: add demo data
$ invenio-cli demo --containers

# After updating statics or code, if you do not need to re-install JS
```

(continues on next page)

(continued from previous page)

```
# dependencies which can take time
$ invenio-cli containerize --no-install-js
```

2.3 More Help

```
# Get more help
$ invenio-cli --help
```

Further documentation is available on https://invenio-cli.readthedocs.io/

2.4 User's Guide

This part of the documentation will show you how to get started in using Invenio-Cli.

2.4.1 Installation

Invenio-Cli is on PyPI so all you need is:

```
$ pip install invenio-cli
```

2.4.2 Usage

Invenio module to ease the creation and management of applications.

2.5 API Reference

If you are looking for information on a specific function, class or method, this part of the documentation is for you.

2.5.1 API Docs

CLI

Invenio module to ease the creation and management of applications.

Commands

Invenio module to ease the creation and management of applications.

class invenio_cli.commands.**AssetsCommands**(*cli_config*)

Local installation commands.

Constructor.

link_js_module(*path*)

High-level command to install and build a JS module.

watch_assets()

High-level command to watch assets for changes.

watch_js_module(*path*, *link=True*)

High-level command to watch a JS module for changes.

class invenio_cli.commands.**Commands**(*cli_config*)

Abstraction over CLI commands that are either local or containerized.

Constructor.

Parameters
cli_config – :class:CLIConfig instance

destroy()

Destroys the instance's virtualenv.

NOTE: This function has no knowledge of the existence of services.
Refer to services.py to destroy services' containers.

classmethod pyshell(*debug=False*)

Start a Python shell.

classmethod shell()

Start a shell in the virtual environment.

class invenio_cli.commands.**ContainersCommands**(*cli_config*, *docker_helper=None*)

Containerized environment CLI commands.

Constructor.

build(*pull=True*, *cache=True*)

Return the steps to build images.

Parameters

- **pull** – Attempt to pull newer versions of the images.
- **cache** – Use cached images and layers.

declare_queues(*project_shortname*)

Steps to declare the MQ queues required for statistics, etc.

demo(*project_shortname*)

Steps to demo records into the instance.

fixtures(*project_shortname*)

Steps to set up the required fixtures for the instance.

rdm_fixtures(*project_shortname*)

Steps to set up the rdm fixtures for the instance.

setup(*force*, *demo_data=True*, *stop=False*, *services=True*)

Return the steps to setup containerize services.

Parameters

- **force** – Remove existing content (db, indices, etc.).
- **demo_data** – Include demo records.
- **stop** – Stop services after setup.

start(*lock=False*, *build=False*, *setup=False*, *demo_data=True*, *services=True*)

Return the steps to start service and application containers.

Parameters

- **lock** – Lock dependencies.
- **build** – Build containers if not built.
- **setup** – Setup services (db, indices, etc.).
- **demo_data** – Include demo records.
- **services** – Start services or only the application containers. This option is incompatible will all the other flags.

translations(*project_shortname*)

Steps to compile translations for the instance.

class invenio_cli.commands.**InstallCommands**(*cli_config*)

Local installation commands.

Constructor.

install(*pre*, *dev=False*, *flask_env='production'*)

Development installation steps.

install_py_dependencies(*pre*, *dev=False*)

Install Python dependencies.

symlink_project_file_or_folder(*target*)

Create symlink in instance pointing to project file or folder.

update_instance_path()

Update path to instance in config.

class invenio_cli.commands.**LocalCommands**(*cli_config*)

Local CLI commands.

Constructor.

run(*host*, *port*, *debug=True*, *services=True*, *celery_log_file=None*)

Run development server and celery queue.

update_statics_and_assets(*force*, *flask_env='production'*, *log_file=None*)

High-level command to update less/js/images/… files.

Needed here (parent) because is used by Assets and Install commands.

class invenio_cli.commands.**PackagesCommands**

Local installation commands.

static install_locked_dependencies(*pre*, *dev*)

Install dependencies from Pipfile.lock using sync.

static install_packages(*packages*, *log_file=None*)

Steps to install Python packages.

It is a class method since it does not require any configuration.

static is_locked()

Checks if the dependencies have been locked.

static lock(*pre*, *dev*)

Steps to lock Python dependencies.

static outdated_packages()

Steps to show outdated packages.

It is a class method since it does not require any configuration.

static update_package_new_version(*package*, *version*)

Update invenio-app-rdm version.

It is a class method since it does not require any configuration.

static update_packages()

Steps to update all Python packages.

It is a class method since it does not require any configuration.

class invenio_cli.commands.**RequirementsCommands**

Pre-requirements check.

classmethod check(*development=False*)

Steps to check the pre-requisites.

classmethod check_dev()

Steps to check the development pre-requisites.

classmethod check_docker_compose_version(*major*, *minor=-1*, *patch=-1*, *exact=False*)

Check the docker compose version.

classmethod check_docker_version(*major*, *minor=-1*, *patch=-1*, *exact=False*)

Check the docker version.

classmethod check_git_version(*major*, *minor=-1*, *patch=-1*, *exact=False*)

Check the git version.

classmethod check_imagemagick_version(*major*, *minor=-1*, *patch=-1*, *exact=False*)

Check the ImageMagick version.

classmethod check_node_version(*major*, *minor=-1*, *patch=-1*, *exact=False*)

Check the node version.

classmethod check_npm_version(*major*, *minor=-1*, *patch=-1*, *exact=False*)

Check the npm version.

classmethod check_pipenv_installed()

Check the pipenv version.

classmethod check_python_version(*major*, *minor=-1*, *patch=-1*, *exact=False*)

Check the python version.

class invenio_cli.commands.**ServicesCommands**(*cli_config*, *docker_helper=None*)

Service CLI commands.

Constructor.

declare_queues()

Steps to declare the MQ queues required for statistics, etc.

demo()

Steps to add demo records into the instance.

destroy()

Steps to destroy the services's containers.

ensure_containers_running()

Ensures containers are running.

fixtures()

Steps to set up the required fixtures for the instance.

rdm_fixtures()

Steps to set up the rdm fixtures for the instance.

services_expected_status(*expected*)

Checks if the services have the expected status.

setup(*force*, *demo_data=True*, *stop=False*, *services=True*)

Steps to setup services' containers.

A check in invenio-cli's config file is done to see if one-time setup has been executed before.

start()

Steps to start services' containers.

status(*services*, *verbose*)

Checks the status of the given service.

Returns

A list of the same length than services. Each item will be a code corresponding to: 0 success, 1 failure, 2 healthcheck not defined.

stop()

Stops containers.

translations()

Steps to compile translations.

class invenio_cli.commands.**TranslationsCommands**(*project_path*, *instance_path*)

Translations CLI commands.

Constructor.

compile(*directory=None*, *fuzzy=False*, *translation_folder='translations'*, *symlink=True*)

Compile the message catalog.

classmethod extract(*babel_file*, *output_file*, *input_dirs*, *msgid_bugs_address*, *copyright_holder*, *add_comments='NOTE'*)

Extract messages from source code and templates.

classmethod init(*output_dir*, *input_file*, *locale*)

Initialize a new language catalog.

classmethod update(*output_dir*, *input_file*)

Update the message catalog.

class invenio_cli.commands.**UpgradeCommands**

Local installation commands.

static upgrade(*script_path*)

Steps to perform an upgrade of the invenio instance.

First, and alembic upgrade is launched to allow alembic to migrate the database using SQLAlchemy. Then, the custom script is executed. Last, the search indices are destroyed, initialized and rebuilt. It is a class method since it does not require any configuration.

Helpers

Invenio CLI helpers module.

2.6 Additional Notes

Notes on how to contribute, legal information and changes are here for the interested.

2.6.1 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given. Additional documentation can be found in the Invenio maintainers guide.

Types of Contributions

Report Bugs

Report bugs at https://github.com/inveniosoftware/invenio-cli/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

Write Documentation

Invenio-Cli could always use more documentation, whether as part of the official Invenio-Cli docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/inveniosoftware/invenio-cli/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

Get Started!

Ready to contribute? Here's how to set up *invenio-cli* for local development.

1. Fork the *inveniosoftware/invenio-cli* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/invenio-cli.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv invenio-cli
$ cd invenio-cli/
$ pip install -e .[all]
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass tests:

```
$ ./run-tests.sh
```

The tests will provide you with test coverage and also check PEP8 (code style), PEP257 (documentation), flake8 as well as build the Sphinx documentation and run doctests.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -s
    -m "component: title without verbs"
    -m "* NEW Adds your new feature."
    -m "* FIX Fixes an existing issue."
    -m "* BETTER Improves and existing feature."
    -m "* Changes something that should not be visible in release notes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests and must not decrease test coverage.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring.
3. The pull request should work for Python 2.7, 3.5 and 3.6. Check https://travis-ci.org/inveniosoftware/invenio-cli/pull_requests and make sure that the tests pass for all supported Python versions.

2.6.2 Changes

Version 1.0.21 (released 2023-05-18)

- deps: support docker < 7 for compatibility with urllib3 v2

Version 1.0.20 (released 2023-03-134)

- setup: add queues initialisation to steps

Version 1.0.19 (released 2023-03-10)

- global: remove fail message on warning (i.e. soft failures)

Version 1.0.18 (released 2023-02-07)

- containerize: fix translation commands instance path

Version 1.0.17 (released 2023-01-30)

- requirements: check node version depending on app-rdm version

Version 1.0.16 (released 2023-01-30)

- bump cookiecutter to v11.0

Version 1.0.15 (released 2023-01-13)

- Setup: fix empty translation folder failing

Version 1.0.14 (released 2023-01-09)

- Add app-rdm fixtures to setup

Version 1.0.13 (released 2022-11-14)

- Allow compilation command to fail in case of missing catalogs.

Version 1.0.12 (released 2022-10-28)

- Adds support for translations (i18n) management commands.

Version 1.0.11 (released 2022-10-24)

- Add support for InvenioILS

Version 1.0.8 (released 2022-10-13)

- Fix issue when checking for services to be up and running correctly.

Version 1.0.7 (released 2022-10-10)

- Fix compat issue with RDM versions < v10

Version 1.0.6 (released 2022-10-10)

- Bump default RDM version.

Version 1.0.5 (released 2022-05-31)

- Bump click version.
- Bump default RDM version.
- Improve error handling.
- Add check for npm version.
- Move ImageMagick check to –development.

Version 1.0.4 (released 2022-02-14)

- Fixes an issue with virtualenv 20.13.1+ brining in setuptools 60.x which is incompatible with Celery v5.2.3. Once Celery v5.2.4 has been released, this fix is no longer needed.

Version 1.0.3 (released 2022-02-04)

- Added `--no-input` and `--config=` options to `init` to support running with predefined config and without requiring user input.

Version 1.0.0 (released 2021-08-05)

- Initial public release.

2.6.3 License

HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Note: In applying this license, CERN does not waive the privileges and immunities granted to it by virtue of its status as an Intergovernmental Organization or submit itself to any jurisdiction.

2.6.4 Authors

Invenio module that allows the creation of applications building workflows

- CERN <info@inveniosoftware.org>

PYTHON MODULE INDEX

i

INDEX