
renku Documentation

Release 0.14.2

Swiss Data Science Center

May 19, 2021

1	Installation	3
1.1	Prerequisites	3
1.2	pipx	3
1.3	pip	4
1.4	Windows	4
1.5	Docker	5
2	Getting Started	7
3	Renku Command Line	9
3.1	renku (base command)	9
3.2	renku init	10
3.3	renku clone	11
3.4	renku config	12
3.5	renku dataset	13
3.6	renku run	17
3.7	renku log	20
3.8	renku status	22
3.9	renku update	22
3.10	renku rerun	23
3.11	renku rm	24
3.12	renku mv	24
3.13	renku workflow	24
3.14	renku save	25
3.15	renku show	26
3.16	renku storage	28
3.17	renku doctor	28
3.18	renku migrate	28
3.19	renku githooks	29
3.20	Error Tracking	29
4	Renku Python API	31
4.1	Project	31
4.2	Dataset	31
4.3	Inputs, Outputs, and Parameters	32
5	Internals	33

5.1	Models	33
5.2	Repository API	56
5.3	Plugin Support	63
6	Changes	65
6.1	'< https://github.com/SwissDataScienceCenter/renku-python/compare/v0.14.1...v0.14.2 >'__ (2021-04-16)	65
6.2	0.14.1 (2021-03-24)	65
6.3	0.14.0 (2021-03-05)	66
6.4	0.13.0 (2021-01-29)	66
6.5	'< https://github.com/SwissDataScienceCenter/renku-python/compare/v0.12.2...v0.12.3 >'__ (2021-01-05)	67
6.6	'< https://github.com/SwissDataScienceCenter/renku-python/compare/v0.12.1...v0.12.2 >'__ (2020-12-02)	67
6.7	'< https://github.com/SwissDataScienceCenter/renku-python/compare/v0.12.0...v0.12.1 >'__ (2020-11-16)	68
6.8	'< https://github.com/SwissDataScienceCenter/renku-python/compare/v0.11.6...v0.12.0 >'__ (2020-11-03)	68
6.9	0.11.6 (2020-10-16)	69
6.10	0.11.5 (2020-10-13)	69
6.11	0.11.4 (2020-10-05)	69
6.12	0.11.3 (2020-09-29)	70
6.13	0.11.2 (2020-09-24)	70
6.14	0.11.1 (2020-08-18)	70
6.15	0.11.0 (2020-08-14)	70
6.16	0.10.5 (2020-07-16)	71
6.17	0.10.4 (2020-05-18)	71
6.18	0.10.3 (2020-04-22)	72
6.19	0.10.1 (2020-03-31)	73
6.20	0.10.0 (2020-03-25)	73
6.21	0.9.1 (2020-02-24)	74
6.22	0.9.0 (2020-02-07)	74
6.23	0.8.0 (2019-11-21)	75
6.24	0.7.0 (2019-10-15)	76
6.25	0.6.1 (2019-10-10)	76
6.26	0.6.0 (2019-09-18)	76
6.27	0.5.2 (2019-07-26)	77
6.28	0.5.1 (2019-07-12)	77
6.29	0.5.0 (2019-03-28)	78
6.30	v0.4.0	79
6.31	v0.3.3	80
6.32	v0.3.2	80
6.33	v0.3.1	80
6.34	v0.3.0	80
6.35	v0.2.0	80
6.36	v0.1.0	81
7	License	83
8	Authors	85
9	Prerequisites	87
10	Renku Python Library, CLI and Service	91
10.1	Prerequisites	91

Python Module Index

95

Index

97

A Python library for the [Renku collaborative data science platform](#). It includes a CLI and SDK for end-users as well as a service backend. It provides functionality for the creation and management of projects and datasets, and simple utilities to capture data provenance while performing analysis tasks.

NOTE: `renku-python` is the python library and core service for Renku - it *does not* start the Renku platform itself - for that, refer to the Renku docs on [running the platform](#).

Renku releases and development versions are available from [PyPI](#). You can install it using any tool that knows how to handle PyPI packages. Our recommendation is to use `code:pix`.

Note: We do not officially support Windows at this moment. The way Windows handles paths and symlinks interferes with some Renku functionality. We recommend using the Windows Subsystem for Linux (WSL) to use Renku on Windows.

1.1 Prerequisites

Renku depends on Git under the hood, so make sure that you have [Git installed on your system](#).

Renku also offers support to store large files in [Git LFS](#), which is used by default and should be installed on your system. If you do not wish to use Git LFS, you can run Renku commands with the `-S` flag, as in `renku -S <command>`. More information on Git LFS usage in renku can be found in the [Data in Renku](#) section of the docs.

1.2 pipx

First, [install pipx](#) and make sure that the `$PATH` is correctly configured.

```
$ python3 -m pip install --user pipx
$ python3 -m pipx ensurepath
```

Once `pipx` is installed use following command to install `renku`.

```
$ pipx install renku
$ which renku
~/local/bin/renku
```

`pipx` installs Renku into its own virtual environment, making sure that it does not pollute any other packages or versions that you may have already installed.

Note: If you install Renku as a dependency in a virtual environment and the environment is active, your shell will default to the version installed in the virtual environment, *not* the version installed by `pipx`.

To install a development release:

```
$ pipx install --pip-args pre renku
```

1.3 pip

```
$ pip install renku
```

The latest development versions are available on PyPI or from the Git repository:

```
$ pip install --pre renku
# - OR -
$ pip install -e git+https://github.com/SwissDataScienceCenter/renku-python.git
↳#egg=renku
```

Use following installation steps based on your operating system and preferences if you would like to work with the command line interface and you do not need the Python library to be importable.

1.4 Windows

Note: We don't officially support Windows yet, but Renku works well in the Windows Subsystem for Linux (WSL). As such, the following can be regarded as a best effort description on how to get started with Renku on Windows.

Renku can be run using the Windows Subsystem for Linux (WSL). To install the WSL, please follow the [official instructions](#).

We recommend you use the Ubuntu 20.04 image in the WSL when you get to that step of the installation.

Once WSL is installed, launch the WSL terminal and install the packages required by Renku with:

```
$ sudo apt-get update && sudo apt-get install git python3 python3-pip python3-venv
↳pipx
```

Since Ubuntu has an older version of git LFS installed by default which is known to have some bugs when cloning repositories, we recommend you manually install the newest version by following [these instructions](#).

Once all the requirements are installed, you can install Renku normally by running:

```
$ pipx install renku
$ pipx ensurepath
```

After this, Renku is ready to use. You can access your Windows in the various mount points in `/mnt/` and you can execute Windows executables (e.g. `*.exe`) as usual directly from the WSL (so `renku run myexecutable.exe` will work as expected).

1.5 Docker

The containerized version of the CLI can be launched using Docker command.

```
$ docker run -it -v "$PWD":"$PWD" -w="$PWD" renku/renku-python renku
```

It makes sure your current directory is mounted to the same place in the container.

CHAPTER 2

Getting Started

Interaction with the platform can take place via the command-line interface (CLI).

Start by creating for folder where you want to keep your Renku project:

```
$ renku init my-renku-project
$ cd my-renku-project
```

Create a dataset and add data to it:

```
$ renku dataset create my-dataset
$ renku dataset add my-dataset https://raw.githubusercontent.com/
↳SwissDataScienceCenter/renku-python/master/README.rst
```

Run an analysis:

```
$ renku run wc < data/my-dataset/README.rst > wc_readme
```

Trace the data provenance:

```
$ renku log wc_readme
```

These are the basics, but there is much more that Renku allows you to do with your data analysis workflows.

For more information about using *renku*, refer to the *renku -help*.

Renku Command Line

The base command for interacting with the Renku platform.

3.1 renku (base command)

To list the available commands, either run `renku` with no parameters or execute `renku help`:

```
$ renku help
Usage: renku [OPTIONS] COMMAND [ARGS]...

Check common Renku commands used in various situations.

Options:
  --version                Print version number.
  --global-config-path    Print global application's config path.
  --install-completion    Install completion for the current shell.
  --path <path>          Location of a Renku repository.
                        [default: (dynamic)]
  --external-storage / -S, --no-external-storage
                        Use an external file storage service.
  -h, --help              Show this message and exit.

Commands:
  # [...]
```

3.1.1 Configuration files

Depending on your system, you may find the configuration files used by Renku command line in a different folder. By default, the following rules are used:

MacOS: `~/Library/Application Support/Renku`

Unix: ~/.config/renku

Windows: C:\Users\

If in doubt where to look for the configuration file, you can display its path by running `renku --global-config-path`.

3.2 renku init

Create an empty Renku project or reinitialize an existing one.

3.2.1 Start a Renku project

If you have an existing directory which you want to turn into a Renku project, you can type:

```
$ cd ~/my_project
$ renku init
```

or:

```
$ renku init ~/my_project
```

This creates a new subdirectory named `.renku` that contains all the necessary files for managing the project configuration.

If provided directory does not exist, it will be created.

3.2.2 Use a different template

Renku is installed together with a specific set of templates you can select when you initialize a project. You can check them by typing:

```
$ renku init --list-templates
```

INDEX	ID	DESCRIPTION	PARAMETERS
1	python	The simplest Python-based [...]	description: project des[...]
2	R	R-based renku project with [...]	description: project des[...]

If you know which template you are going to use, you can provide either the id `--template-id` or the template index number `--template-index`.

You can use a newer version of the templates or even create your own one and provide it to the `init` command by specifying the target template repository source `--template-source` (both local path and remote url are supported) and the reference `--template-ref` (branch, tag or commit).

You can take inspiration from the [official Renku template repository](#)

```
$ renku init --template-ref master --template-source \
https://github.com/SwissDataScienceCenter/renku-project-template

Fetching template from
https://github.com/SwissDataScienceCenter/renku-project-template@master
... OK
```

(continues on next page)

(continued from previous page)

INDEX ID	DESCRIPTION	PARAMETERS
1	python-minimal	Basic Python Project: [...] description: proj[...]
2	R-minimal	Basic R Project: The [...] description: proj[...]

Please choose a template by typing the index:

Provide parameters ~~~~~

Some templates require parameters to properly initialize a new project. You can check them by listing the templates `--list-templates`.

To provide parameters, use the `--parameter` option and provide each parameter using `--parameter "param1"="value1"`.

```
$ renku init --template-id python-minimal --parameter \
"description"="my new shiny project"
```

```
Initializing new Renku repository... OK
```

If you don't provide the required parameters through the option `-parameter`, you will be asked to provide them. Empty values are allowed and passed to the template initialization function.

Note: Every project requires a name that can either be provided using `--name` or automatically taken from the target folder. This is also considered as a special parameter, therefore it's automatically added to the list of parameters forwarded to the `init` command.

3.2.3 Update an existing project

There are situations when the required structure of a Renku project needs to be recreated or you have an **existing** Git repository. You can solve these situation by simply adding the `--force` option.

```
$ git init .
$ echo "# Example\nThis is a README." > README.md
$ git add README.md
$ git commit -m 'Example readme file'
# renku init would fail because there is a git repository
$ renku init --force
```

You can also enable the external storage system for output files, if it was not installed previously.

```
$ renku init --force --external-storage
```

3.3 renku clone

Clone a Renku project.

3.3.1 Cloning a Renku project

To clone a Renku project use `renku clone` command. This command is preferred over using `git clone` because it sets up required Git hooks and enables Git-LFS automatically.

```
$ renku clone <repository-url> <destination-directory>
```

It creates a new directory with the same name as the project. You can change the directory name by passing another name on the command line.

By default, `renku clone` pulls data from Git-LFS after cloning. If you don't need the LFS data, pass `--no-pull-data` option to skip this step.

Note: To move a project to another Renku deployment you need to create a new empty project in the target deployment and push both the repository and Git-LFS objects to the new remote. Refer to Git documentation for more details.

```
$ git lfs fetch --all
$ git remote remove origin
$ git remote add origin <new-repository-url>
$ git push --mirror origin
```

3.4 renku config

Get and set Renku repository or global options.

3.4.1 Set values

You can set various Renku configuration options, for example the image registry URL, with a command like:

```
$ renku config set interactive.default_url "/tree"
```

By default, configuration is stored locally in the project's directory. Use `--global` option to store configuration for all projects in your home directory.

3.4.2 Remove values

To remove a specific key from configuration use:

```
$ renku config remove interactive.default_url
```

By default, only local configuration is searched for removal. Use `--global` option to remove a global configuration value.

3.4.3 Query values

You can display all configuration values with:

```
$ renku config show
[renku "interactive"]
default_url = /lab
```

Both local and global configuration files are read. Values in local configuration take precedence over global values. Use `--local` or `--global` flag to read corresponding configuration only.

You can provide a **KEY** to display only its value:

```
$ renku config show interactive.default_url
default_url = /lab
```

3.4.4 Available configuration values

The following values are available for the `renku config` command:

Name	Description	Default
<code>show_ifs_message</code>	Whether to show messages about files being added to git LFS or not	True
<code>ifs_threshold</code>	Threshold file size below which files are not added to git LFS	100kb
<code>zenodo.access_token</code>	Access token for Zenodo API	None
<code>dataverse.access_token</code>	Access token for Dataverse API	None
<code>dataverse.server_url</code>	URL for the Dataverse API server to use	None
<code>interactive.default_url</code>	URL for interactive environments	None
<code>interactive.cpu_request</code>	CPU quota for environments	None
<code>interactive.mem_request</code>	Memory quota for environments	None
<code>interactive.gpu_request</code>	GPU quota for environments	None
<code>interactive.ifs_auto_fetch</code>	Whether to automatically fetch ifs files on environments startup	None
<code>interactive.image</code>	Pinned Docker image for environments	None

3.5 renku dataset

Renku CLI commands for handling of datasets.

3.5.1 Manipulating datasets

Creating an empty dataset inside a Renku project:

```
$ renku dataset create my-dataset
Creating a dataset ... OK
```

You can pass the following options to this command to set various metadata for the dataset.

Option	Description
<code>-t, --title</code>	A human-readable title for the dataset.
<code>-d, --description</code>	Dataset's description.
<code>-c, --creator</code>	Creator's name, email, and an optional affiliation. Accepted format is 'Forename Surname <email> [affiliation]'. Pass multiple times for a list of creators.
<code>-k, --keyword</code>	Dataset's keywords. Pass multiple times for a list of keywords.

Editing a dataset's metadata

Use `edit` subcommand to change metadata of a dataset. You can edit the same set of metadata as the `create` command by passing the options described in the table above.

```
$ renku dataset edit my-dataset --title 'New title'
Successfully updated: title.
```

Listing all datasets:

```
$ renku dataset ls
ID          NAME          TITLE          VERSION
-----
0ad1cb9a   some-dataset  Some Dataset
9436e36c   my-dataset    My Dataset
```

You can select which columns to display by using `--columns` to pass a comma-separated list of column names:

```
$ renku dataset ls --columns id,name,date_created,creators
ID          NAME          CREATED          CREATORS
-----
0ad1cb9a   some-dataset  2020-03-19 16:39:46  sam
9436e36c   my-dataset    2020-02-28 16:48:09  sam
```

Displayed results are sorted based on the value of the first column.

To inspect the state of the dataset on a given commit we can use `--revision` flag for it:

```
$ renku dataset ls --revision=1103a42bd3006c94ef2af5d6a5e03a335f071215
ID          NAME          TITLE          VERSION
-----
a1fd8ce2   201901_us_flights_1  2019-01 US Flights  1
c2d80abe   ds1            ds1
```

Showing dataset details:

. code-block:: console

```
$ renku dataset show some-dataset Name: some-dataset Created: 2020-12-09 13:52:06.640778+00:00
Creator(s): John Doe<john.doe@example.com> [SDSC] Keywords: Dataset, Data Title: Some Dataset
Description: Just some dataset
```

Deleting a dataset:

```
$ renku dataset rm some-dataset
OK
```

3.5.2 Working with data

Adding data to the dataset:

```
$ renku dataset add my-dataset http://data-url
```

This will copy the contents of `data-url` to the dataset and add it to the dataset metadata.

You can create a dataset when you add data to it for the first time by passing `--create` flag to add command:

```
$ renku dataset add --create new-dataset http://data-url
```

To add data from a git repository, you can specify it via `https` or `git+ssh` URL schemes. For example,

```
$ renku dataset add my-dataset git+ssh://host.io/namespace/project.git
```

Sometimes you want to add just specific paths within the parent project. In this case, use the `--source` or `-s` flag:

```
$ renku dataset add my-dataset --source path/within/repo/to/datafile \
  git+ssh://host.io/namespace/project.git
```

The command above will result in a structure like

```
data/
  my-dataset/
    datafile
```

You can use shell-like wildcards (e.g. `*`, `*`, `?`) when specifying paths to be added. Put wildcard patterns in quotes to prevent your shell from expanding them.

```
$ renku dataset add my-dataset --source 'path/**/datafile' \
  git+ssh://host.io/namespace/project.git
```

You can use `--destination` or `-d` flag to set the location where the new data is copied to. This location will be under the dataset's data directory and will be created if it does not exist. You will get an error message if the destination exists and is a file.

```
$ renku dataset add my-dataset \
  --source path/within/repo/to/datafile \
  --destination new-dir/new-subdir \
  git+ssh://host.io/namespace/project.git
```

will yield:

```
data/
  my-dataset/
    new-dir/
      new-subdir/
        datafile
```

To add a specific version of files, use `--ref` option for selecting a branch, commit, or tag. The value passed to this option must be a valid reference in the remote Git repository.

Adding external data to the dataset:

Sometimes you might want to add data to your dataset without copying the actual files to your repository. This is useful for example when external data is too large to store locally. The external data must exist (i.e. be mounted) on your filesystem. Renku creates a symbolic link to your data and you can use this symbolic link in renku commands as a normal file. To add an external file pass `--external` or `-e` when adding local data to a dataset:

```
$ renku dataset add my-dataset -e /path/to/external/file
```

Updating a dataset:

After adding files from a remote Git repository or importing a dataset from a provider like Dataverse or Zenodo, you can check for updates in those files by using `renku dataset update` command. For Git repositories, this command checks all remote files and copies over new content if there is any. It does not delete files from the local dataset if they are deleted from the remote Git repository; to force the delete use `--delete` argument. You can update to a specific branch, commit, or tag by passing `--ref` option. For datasets from providers like Dataverse or Zenodo, the whole dataset is updated to ensure consistency between the remote and local versions. Due to this limitation, the `--include` and `--exclude` flags are not compatible with those datasets. Modifying those datasets locally will prevent them from being updated.

You can limit the scope of updated files by specifying dataset names, using `--include` and `--exclude` to filter based on file names, or using `--creators` to filter based on creators. For example, the following command updates only CSV files from `my-dataset`:

```
$ renku dataset update -I '*.csv' my-dataset
```

Note that putting glob patterns in quotes is needed to tell Unix shell not to expand them.

External data are not updated automatically because they require a checksum calculation which can take a long time when data is large. To update external files pass `--external` or `-e` to the update command:

```
$ renku dataset update -e
```

Tagging a dataset:

A dataset can be tagged with an arbitrary tag to refer to the dataset at that point in time. A tag can be added like this:

```
$ renku dataset tag my-dataset 1.0 -d "Version 1.0 tag"
```

A list of all tags can be seen by running:

```
$ renku dataset ls-tags my-dataset
CREATED          NAME      DESCRIPTION      DATASET      COMMIT
-----
2020-09-19 17:29:13  1.0      Version 1.0 tag  my-dataset  6c19a8d31545b...
```

A tag can be removed with:

```
$ renku dataset rm-tags my-dataset 1.0
```

Importing data from other Renku projects:

To import all data files and their metadata from another Renku dataset use:

```
$ renku dataset import \
  https://renkulab.io/projects/<username>/<project>/datasets/<dataset-id>
```

or

```
$ renku dataset import \
  https://renkulab.io/datasets/<dataset-id>
```

You can get the link to a dataset from the UI or you can construct it by knowing the dataset's ID.

Importing data from an external provider:

```
$ renku dataset import 10.5281/zenodo.3352150
```

This will import the dataset with the DOI (Digital Object Identifier) `10.5281/zenodo.3352150` and make it locally available. Dataverse and Zenodo are supported, with DOIs (e.g. `10.5281/zenodo.3352150` or `doi:10.5281/zenodo.3352150`) and full URLs (e.g. `http://zenodo.org/record/3352150`). A tag with the remote version of the dataset is automatically created.

Exporting data to an external provider:

```
$ renku dataset export my-dataset zenodo
```

This will export the dataset `my-dataset` to `zenodo.org` as a draft, allowing for publication later on. If the dataset has any tags set, you can choose if the repository *HEAD* version or one of the tags should be exported. The remote version will be set to the local tag that is being exported.

To export to a Dataverse provider you must pass Dataverse server's URL and the name of the parent dataverse where the dataset will be exported to. Server's URL is stored in your Renku setting and you don't need to pass it every time.

Listing all files in the project associated with a dataset.

```
$ renku dataset ls-files
DATASET NAME      ADDED              PATH              LFS
-----
my-dataset        2020-02-28 16:48:09 data/my-dataset/add-me      *
my-dataset        2020-02-28 16:49:02 data/my-dataset/weather/file1 *
my-dataset        2020-02-28 16:49:02 data/my-dataset/weather/file2
my-dataset        2020-02-28 16:49:02 data/my-dataset/weather/file3 *
```

You can select which columns to display by using `--columns` to pass a comma-separated list of column names:

```
$ renku dataset ls-files --columns name,creators, path
DATASET NAME      CREATORS    PATH
-----
my-dataset        sam         data/my-dataset/add-me
my-dataset        sam         data/my-dataset/weather/file1
my-dataset        sam         data/my-dataset/weather/file2
my-dataset        sam         data/my-dataset/weather/file3
```

Displayed results are sorted based on the value of the first column.

Sometimes you want to filter the files. For this we use `--dataset`, `--include` and `--exclude` flags:

```
$ renku dataset ls-files --include "file*" --exclude "file3"
DATASET NAME      ADDED              PATH              LFS
-----
my-dataset        2020-02-28 16:49:02 data/my-dataset/weather/file1 *
my-dataset        2020-02-28 16:49:02 data/my-dataset/weather/file2 *
```

Unlink a file from a dataset:

```
$ renku dataset unlink my-dataset --include file1
OK
```

Unlink all files within a directory from a dataset:

```
$ renku dataset unlink my-dataset --include "weather/*"
OK
```

Unlink all files from a dataset:

```
$ renku dataset unlink my-dataset
Warning: You are about to remove following from "my-dataset" dataset.
../my-dataset/weather/file1
../my-dataset/weather/file2
../my-dataset/weather/file3
Do you wish to continue? [y/N]:
```

Note: The unlink command does not delete files, only the dataset record.

3.6 renku run

Track provenance of data created by executing programs.

3.6.1 Capture command line execution

Tracking execution of your command line script is done by simply adding the `renku run` command before the actual command. This will enable detection of:

- arguments (flags),
- string and integer options,
- input files or directories if linked to existing paths in the repository,
- output files or directories if modified or created while running the command.

Note: If there were uncommitted changes in the repository, then the `renku run` command fails. See `git status` for details.

Warning: Input and output paths can only be detected if they are passed as arguments to `renku run`.

Warning: Circular dependencies are not supported for `renku run`. See *Circular Dependencies* for more details.

Warning: When using output redirection in `renku run` on Windows (with “> file” or “2> file”), all Renku errors and messages are redirected as well and `renku run` produces no output on the terminal. On Linux, this is detected by `renku` and only the output of the command to be run is actually redirected. Renku specific messages such as errors get printed to the terminal as usual and don’t get redirected.

3.6.2 Detecting input paths

Any path passed as an argument to `renku run`, which was not changed during the execution, is identified as an input path. The identification only works if the path associated with the argument matches an existing file or directory in the repository.

The detection might not work as expected if:

- a file is **modified** during the execution. In this case it will be stored as an **output**;
- a path is not passed as an argument to `renku run`.

Specifying auxiliary inputs (`--input`)

You can specify extra inputs to your program explicitly by using the `--input` option. This is useful for specifying hidden dependencies that don’t appear on the command line. Explicit inputs must exist before execution of `renku run` command. This option is not a replacement for the arguments that are passed on the command line. Files or directories specified with this option will not be passed as input arguments to the script.

Disabling input detection (`--no-input-detection`)

Input paths detection can be disabled by passing `--no-input-detection` flag to `renku run`. In this case, only the directories/files that are passed as explicit input are considered to be file inputs. Those passed via command arguments are ignored unless they are in the explicit inputs list. This only affects files and directories; command options and flags are still treated as inputs.

3.6.3 Detecting output paths

Any path **modified** or **created** during the execution will be added as an output.

Because the output path detection is based on the Git repository state after the execution of `renku run` command, it is good to have a basic understanding of the underlying principles and limitations of tracking files in Git.

Git tracks not only the paths in a repository, but also the content stored in those paths. Therefore:

- a recreated file with the same content is not considered an output file, but instead is kept as an input;
- file moves are detected based on their content and can cause problems;
- directories cannot be empty.

Note: When in doubt whether the outputs will be detected, remove all outputs using `git rm <path>` followed by `git commit` before running the `renku run` command.

Command does not produce any files (`--no-output`)

If the program does not produce any outputs, the execution ends with an error:

```
Error: There are not any detected outputs in the repository.
```

You can specify the `--no-output` option to force tracking of such an execution.

Specifying outputs explicitly (`--output`)

You can specify expected outputs of your program explicitly by using the `--output` option. These output must exist after the execution of the `renku run` command. However, they do not need to be modified by the command.

Disabling output detection (`--no-output-detection`)

Output paths detection can be disabled by passing `--no-output-detection` flag to `renku run`. When disabled, only the directories/files that are passed as explicit output are considered to be outputs and those passed via command arguments are ignored.

3.6.4 Detecting standard streams

Often the program expect inputs as a standard input stream. This is detected and recorded in the tool specification when invoked by `renku run cat < A`.

Similarly, both redirects to standard output and standard error output can be done when invoking a command:

```
$ renku run grep "test" B > C 2> D
```

Warning: Detecting inputs and outputs from pipes | is not supported.

3.6.5 Specifying inputs and outputs programmatically

Sometimes the list of inputs and outputs are not known before execution of the program. For example, a program might accept a date range as input and access all files within that range during its execution.

To address this issue, the program can dump a list of input and output files that it is accessing in `inputs.txt` and `outputs.txt`. Each line in these files is expected to be the path to an input or output file within the project's directory. When the program is finished, Renku will look for existence of these two files and adds their content to the list of explicit inputs and outputs. Renku will then delete these two files.

By default, Renku looks for these two files in `.renku/tmp` directory. One can change this default location by setting `RENKU_INDIRECT_PATH` environment variable. When set, it points to a sub-directory within the `.renku/tmp` directory where `inputs.txt` and `outputs.txt` reside.

3.6.6 Exit codes

All Unix commands return a number between 0 and 255 which is called “exit code”. In case other numbers are returned, they are treated modulo 256 (-10 is equivalent to 246, 257 is equivalent to 1). The exit-code 0 represents a *success* and non-zero exit-code indicates a *failure*.

Therefore the command specified after `renku run` is expected to return exit-code 0. If the command returns different exit code, you can specify them with `--success-code=<INT>` parameter.

```
$ renku run --success-code=1 --no-output fail
```

3.6.7 Circular Dependencies

Circular dependencies are not supported in `renku run`. This means you cannot use the same file or directory as both an input and an output in the same step, for instance reading from a file as input and then appending to it is not allowed. Since `renku` records all steps of an analysis workflow in a dependency graph and it allows you to update outputs when an input changes, this would lead to problems with circular dependencies. An update command would change the input again, leading to `renku` seeing it as a changed input, which would run update again, and so on, without ever stopping.

Due to this, the `renku` dependency graph has to be *acyclic*. So instead of appending to an input file or writing an output file to the same directory that was used as an input directory, create new files or write to other directories, respectively.

3.7 renku log

Show provenance of data created by executing programs.

3.7.1 File provenance

Unlike the traditional file history format, which shows previous revisions of the file, this format presents tool inputs together with their revision identifiers.

A `*` character shows to which lineage the specific file belongs to. A `@` character in the graph lineage means that the corresponding file does not have any inputs and the history starts there.

When called without file names, `renku log` shows the history of most recently created files. With the `--revision <refname>` option the output is shown as it was in the specified revision.

Provenance examples

renku log B Show the history of file B since its last creation or modification.

renku log --revision HEAD~5 Show the history of files that have been created or modified 5 commits ago.

renku log --revision e3f0bd5a D E Show the history of files D and E as it looked in the commit e3f0bd5a.

3.7.2 Output formats

Following formats supported when specified with `--format` option:

- *ascii*
- *dot*
- *dot-full*
- *dot-landscape*
- *dot-full-landscape*
- *dot-debug*
- *json-ld*
- *json-ld-graph*
- *Makefile*
- *nt*
- *rdf*

You can generate a PNG of the full history of all files in the repository using the **dot** program.

```
$ FILES=$(git ls-files --no-empty-directory --recurse-submodules)
$ renku log --format dot $FILES | dot -Tpng > /tmp/graph.png
$ open /tmp/graph.png
```

3.7.3 Output validation

The `--strict` option forces the output to be validated against the Renku SHACL schema, causing the command to fail if the generated output is not valid, as well as printing detailed information on all the issues found. The `--strict` option is only supported for the `jsonld`, `rdf` and `nt` output formats.

3.8 renku status

Show status of data files created in the repository.

3.8.1 Inspecting a repository

Displays paths of outputs which were generated from newer inputs files and paths of files that have been used in diverent versions.

The first paths are what need to be recreated by running `renku update`. See more in section about [renku update](#).

The paths mentioned in the output are made relative to the current directory if you are working in a subdirectory (this is on purpose, to help cutting and pasting to other commands). They also contain first 8 characters of the corresponding commit identifier after the # (hash). If the file was imported from another repository, the short name of is shown together with the filename before @.

3.9 renku update

Update outdated files created by the “run” command.

3.9.1 Recreating outdated files

The information about dependencies for each file in the repository is generated from information stored in the underlying Git repository.

A minimal dependency graph is generated for each outdated file stored in the repository. It means that only the necessary steps will be executed and the workflow used to orchestrate these steps is stored in the repository.

Assume that the following history for the file H exists.



The first example shows situation when D is modified and files E and H become outdated.



In this situation, you can do effectively two things:

- Recreate a single file by running

```
$ renku update E
```

- Update all files by simply running

```
$ renku update --all
```

Note: If there were uncommitted changes then the command fails. Check `git status` to see details.

3.9.2 Pre-update checks

In the next example, files A or B are modified, hence the majority of dependent files must be recreated.

```

      (C) -- (D) -- (E)
     /
*A*--*B*--(F) -- (G) -- (H)
     \

```

To avoid excessive recreation of the large portion of files which could have been affected by a simple change of an input file, consider specifying a single file (e.g. `renku update G`). See also [renku status](#).

3.9.3 Update siblings

If a tool produces multiple output files, these outputs need to be always updated together.

```

      (B)
     /
*A*--[step 1]--(C)
     \
      (D)

```

An attempt to update a single file would fail with the following error.

```

$ renku update C
Error: There are missing output siblings:

  B
  D

Include the files above in the command or use --with-siblings option.

```

The following commands will produce the same result.

```

$ renku update --with-siblings C
$ renku update B C D

```

3.10 renku rerun

Recreate files created by the “run” command.

3.10.1 Recreating files

Assume you have run a step 2 that uses a stochastic algorithm, so each run will be slightly different. The goal is to regenerate output C several times to compare the output. In this situation it is not possible to simply call `renku update` since the input file A has not been modified after the execution of step 2.

```
A-[step 1]-B-[step 2*]-C
```

Recreate a specific output file by running:

```
$ renku rerun C
```

If you would like to recreate a file which was one of several produced by a tool, then these files must be recreated as well. See the explanation in [updating siblings](#).

3.11 renku rm

Remove a file, a directory, or a symlink.

Removing a file that belongs to a dataset will update its metadata. It also will attempt to update tracking information for files stored in an external storage (using Git LFS).

3.12 renku mv

Move or rename a file, a directory, or a symlink.

Moving a file that belongs to a dataset will update its metadata to include its new path and commit. Moreover, tracking information in an external storage (e.g. Git LFS) will be updated. Move operation fails if a destination already exists in the repo; use `--force` flag to overwrite them.

If you want to move files to another dataset use `--to-dataset` along with destination's dataset name. This removes source paths from all datasets' metadata that include them (if any) and adds them to the destination's dataset metadata.

The following command moves `data/src` and `README` to `data/dst` directory and adds them to `target-dataset`'s metadata. If the source files belong to one or more datasets then they will be removed from their metadata.

```
$ renku mv data/src README data/dst --to-dataset target-dataset
```

3.13 renku workflow

Manage the set of CWL files created by renku commands.

3.13.1 Manipulating workflows

Listing workflows:

```
$ renku workflow ls
26be2e8d66f74130a087642768f2cef0_rerun.yaml:
199c4b9d462f4b27a4513e5e55f76eb2_cat.yaml:
9bea2eccf9624de387d9b06e61eec0b6_rerun.yaml:
b681b4e229764ceda161f6551370af12_update.yaml:
25d0805243e3468d92a3786df782a2c4_rerun.yaml:
```

Each *.yaml file corresponds to a renku run/update/rerun execution.

Exporting workflows:

You can export the workflow to create a file as Common Workflow Language by using:

```
$ renku workflow set-name create output_file
baseCommand:
- cat
class: CommandLineTool
cwlVersion: v1.0
id: 22943eca-fa4c-4f3b-a92d-f6ac7badc0d2
inputs:
- default:
  class: File
  path: /home/user/project/intermediate
id: inputs_1
inputBinding:
  position: 1
type: File
- default:
  class: File
  path: /home/user/project/intermediate2
id: inputs_2
inputBinding:
  position: 2
type: File
outputs:
- id: output_stdout
streamable: false
type: stdout
requirements:
InitialWorkDirRequirement:
  listing:
  - entry: $(inputs.inputs_1)
    entryname: intermediate
    writable: false
  - entry: $(inputs.inputs_2)
    entryname: intermediate2
    writable: false
stdout: output_file
```

You can use `--revision` to specify the revision of the output file to generate the workflow for. You can also export to a file directly with `-o <path>`.

3.14 renku save

Convenience method to save local changes and push them to a remote server.

If you have local modification to files, you can save them using

```
$ renku save
Username for 'https://renkulab.io': my.user
Password for 'https://my.user@renkulab.io':
Successfully saved:
  file1
```

(continues on next page)

(continued from previous page)

```
file2
OK
```

Warning: The username and password for renku save are your gitlab user/password, not your renkulab login!

You can additionally supply a message that describes the changes that you made by using the `-m` or `--message` parameter followed by your message.

```
$ renku save -m "Updated file1 and 2."
Successfully saved:
  file1
  file2
OK
```

If no remote server has been configured, you can specify one by using the `-d` or `--destination` parameter. Otherwise you will get an error.

```
$ renku save
Error: No remote has been set up for the current branch

$ renku save -d https://renkulab.io/gitlab/my.user/my-project.git
Successfully saved:
  file1
  file2
OK
```

You can also specify which paths to save:

```
$ renku save file1
Successfully saved:
  file1
OK
```

3.15 renku show

Show information about objects in current repository.

3.15.1 Siblings

In situations when multiple outputs have been generated by a single `renku run` command, the siblings can be discovered by running `renku show siblings PATH` command.

Assume that the following graph represents relations in the repository.

```

      D---E---G
     /      \
A---B---C   F
```

Then the following outputs would be shown.


```

$ renku show siblings C
C
D
$ renku show siblings G
F
G
$ renku show siblings A
A
$ renku show siblings C G
C
D
---
F
G
$ renku show siblings
A
---
B
---
C
D
---
E
---
F
G

```

You can use the `-f` or `--flat` flag to output a flat list, as well as the `-v` or `--verbose` flag to also output commit information.

3.15.2 Input and output files

You can list input and output files generated in the repository by running `renku show inputs` and `renku show outputs` commands. Alternatively, you can check if all paths specified as arguments are input or output files respectively.

```

$ renku run wc < source.txt > result.wc
$ renku show inputs
source.txt
$ renku show outputs
result.wc
$ renku show outputs source.txt
$ echo $? # last command finished with an error code
1

```

You can use the `-v` or `--verbose` flag to print detailed information in a tabular format.

```

$ renku show inputs -v
PATH          COMMIT      USAGE TIME          WORKFLOW
-----
source.txt    6d10e05    2020-09-14 23:47:17  .renku/workflow/388...d8_head.yaml

```

3.16 renku storage

Manage an external storage.

3.16.1 Pulling files from git LFS

LFS works by checking small pointer files into git and saving the actual contents of a file in LFS. If instead of your file content, you see something like this, it means the file is stored in git LFS and its contents are not currently available locally (they are not pulled):

```
version https://git-lfs.github.com/spec/v1
oid sha256:42b5c7fb2acd54f6d3cd930f18fee3bdcb20598764ca93bdfb38d7989c054bcf
size 12
```

You can manually pull contents of file(s) you want with:

```
$ renku storage pull file1 file2
```

3.16.2 Removing local content of files stored in git LFS

If you want to restore a file back to its pointer file state, for instance to free up space locally, you can run:

```
$ renku storage clean file1 file2
```

This removes any data cached locally for files tracked in in git LFS.

3.16.3 Migrate large files to git LFS

If you accidentally checked a large file into git or are moving a non-LFS renku repo to git LFS, you can use the following command to migrate the files to LFS:

```
$ renku storage migrate --all
```

This will move all files that are bigger than the renku *lfs_threshold* config value and are not excluded by *.renkulfsignore* into git LFS.

To only migrate specific files, you can also pass their paths to the command like:

```
$ renku storage migrate big_file other_big_file
```

3.17 renku doctor

Check your system and repository for potential problems.

3.18 renku migrate

Migrate project to the latest Renku version.

When the way Renku stores metadata changes or there are other changes to the project structure or data that are needed for Renku to work, `renku migrate` can be used to bring the project up to date with the current version of Renku. This does not usually affect how you use Renku and no data is lost.

In addition, `renku migrate` will update your `Dockerfile` to install the latest version of `renku-python`, if supported, making sure your renku version is up to date in interactive environments as well.

If you created your repository from a project template and the template has changed since you created the project, it will also update files with their newest version from the template, without overwriting local changes if there are any.

You can check if a migration is necessary and what migrations are available by running

```
$ renku migrate -c
```

3.19 renku githooks

Install and uninstall Git hooks.

3.19.1 Prevent modifications of output files

The commit hooks are enabled by default to prevent situation when some output file is manually modified.

```
$ renku init
$ renku run echo hello > greeting.txt
$ edit greeting.txt
$ git commit greeting.txt
You are trying to update some output files.

Modified outputs:
  greeting.txt

If you are sure, use "git commit --no-verify".
```

3.20 Error Tracking

Renku is not bug-free and you can help us to find them.

3.20.1 GitHub

You can quickly open an issue on GitHub with a traceback and minimal system information when you hit an unhandled exception in the CLI.

```
Ahhhhhhhh! You have found a bug.

1. Open an issue by typing "open";
2. Print human-readable information by typing "print";
3. See the full traceback without submitting details (default: "ignore").

Please select an action by typing its name (open, print, ignore) [ignore]:
```

3.20.2 Sentry

When using `renku` as a hosted service the Sentry integration can be enabled to help developers iterate faster by showing them where bugs happen, how often, and who is affected.

1. Install Sentry-SDK with `python -m pip install sentry-sdk`;
2. Set environment variable `SENTRY_DSN=https://<key>@sentry.<domain>/<project>`.

<p>Warning: User information might be sent to help resolving the problem. If you are not using your own Sentry instance you should inform users that you are sending possibly sensitive information to a 3rd-party service.</p>
--

4.1 Project

Renku API Project.

Project class acts as a context for other Renku entities like Dataset, or Inputs/Outputs. It provides access to internals of a Renku project for such entities.

Normally, you do not need to create an instance of Project class directly unless you want to have access to Project metadata (e.g. path). To separate parts of your script that uses Renku entities, you can create a Project context manager and interact with Renku inside it:

```
from renku.api import Project, Input

with Project():
    input_1 = Input("data_1")
```

4.2 Dataset

Renku API Dataset.

Dataset class allows listing datasets and files inside a Renku project and accessing their metadata.

To get a list of available datasets in a Renku project use `list` method:

```
from renku.api import Dataset

datasets = Dataset.list()
```

You can then access metadata of a dataset like `name`, `title`, `keywords`, etc. To get the list of files inside a dataset use `files` property:

```
for dataset_file in dataset.files:
    print(dataset_file.path)
```

4.3 Inputs, Outputs, and Parameters

Renku API Workflow Models.

Input and Output classes can be used to define inputs and outputs of a script within the same script. Paths defined with these classes are added to explicit inputs and outputs in the workflow's metadata. For example, the following mark a `data/data.csv` as an input to the script:

```
from renku.api import Input

with open(Input("data/data.csv")) as input_data:
    for line in input_data:
        print(line)
```

Users can track parameters' values in a workflow by defining them using `Parameter` function.

```
from renku.api import Parameter

nc = Parameter(name="n_components", value=10)
```

Internals of the `renku-python` library.

5.1 Models

Model objects used in Python SDK.

5.1.1 Projects

Model objects representing projects.

```
class renku.core.models.projects.Project (name=None, created=NOTHING, version='8',  
agent_version='pre-0.11.0', template_source:  
str = None, template_ref: str = None, tem-  
plate_id: str = None, template_version:  
str = None, template_metadata: str = '{}',  
immutable_template_files=NOTHING, auto-  
mated_update=False, *, client=None, cre-  
ator=None, id=None)
```

Represent a project.

Method generated by attrs for class Project.

```
as_jsonld ()  
    Create JSON-LD.
```

```
classmethod from_jsonld (data, client=None)  
    Create an instance from JSON-LD data.
```

```
classmethod from_yaml (path, client=None)  
    Return an instance from a YAML file.
```

```
project_id  
    Return the id for the project.
```

to_yaml (*path=None*)

Write an instance to the referenced YAML file.

class `renku.core.models.projects.ProjectCollection` (*client=None*)

Represent projects on the server.

Example

Create a project and check its name.

```
# >>> project = client.projects.create(name='test-project') # >>> project.name # 'test-project'
```

Create a representation of objects on the server.

class **Meta**

Information about individual projects.

model

alias of *Project*

create (*name=None, **kwargs*)

Create a new project.

Parameters **name** – The name of the project.

Returns An instance of the newly create project.

Return type *renku.core.models.projects.Project*

class `renku.core.models.projects.ProjectSchema` (**args, commit=None, client=None, **kwargs*)

Project Schema.

Create an instance.

class **Meta**

Meta class.

model

alias of *Project*

fix_datetimes (*obj, many=False, **kwargs*)

Pre dump hook.

`renku.core.models.projects.generate_project_id` (*client, name, creator*)

Return the id for the project based on the repo origin remote.

5.1.2 Datasets

Model objects representing datasets.

Dataset object

```
class renku.core.models.datasets.Dataset (*, commit=None, client=None, path=None,
project: renku.core.models.projects.Project
= None, parent=None, checksum: str =
None, creators, id=None, label=None,
date_published=None, description=None,
identifer=NOTHING, in_language=None, im-
ages=None, keywords=None, license=None,
title: str = None, url=None, version=None,
date_created=NOTHING, files=NOTHING,
tags=NOTHING, same_as=None, name=None,
derived_from=None, immutable=False)
```

Represent a dataset.

Method generated by attrs for class Dataset.

as_jsonld()
Create JSON-LD.

contains_any(*files*)
Check if files are already within a dataset.

creators_csv
Comma-separated list of creators associated with dataset.

creators_full_csv
Comma-separated list of creators with full identity.

data_dir
Directory where dataset files are stored.

default_id()
Configure calculated ID.

default_label()
Generate a default label.

editable
Subset of attributes which user can edit.

entities
Yield itself.

find_file(*path, return_index=False*)
Find a file in files container using its relative path.

find_files(*paths*)
Return all paths that are in files container.

classmethod from_jsonld(*data, client=None, commit=None, schema_class=None*)
Create an instance from JSON-LD data.

classmethod from_revision(*client, path, revision='HEAD', parent=None, find_previous=True,*
***kwargs*)
Return dependency from given path and revision.

classmethod from_yaml(*path, client=None, commit=None*)
Return an instance from a YAML file.

keywords_csv
Comma-separated list of keywords associated with dataset.

mutate ()

Update mutation history and assign a new identifier.

Do not mutate more than once before committing the metadata or otherwise there would be missing links in the chain of changes.

name_validator (*attribute, value*)

Validate name.

original_identifier

Return the first identifier of the dataset.

parent

Return the parent object.

set_client (*client*)

Sets the clients on this entity.

short_id

Shorter version of identifier.

submodules

Proxy to client submodules.

tags_csv

Comma-separated list of tags associated with dataset.

to_yaml (*path=None, immutable=False*)

Write an instance to the referenced YAML file.

unlink_file (*path, missing_ok=False*)

Unlink a file from dataset.

Parameters **path** – Relative path used as key inside files container.

update_files (*files*)

Update files with collection of DatasetFile objects.

update_metadata (***kwargs*)

Updates instance attributes.

update_metadata_from (*other_dataset*)

Updates instance attributes with other dataset attributes.

Parameters **other_dataset** – *Dataset*

Returns *self*

Dataset file

Manage files in the dataset.

```
class renku.core.models.datasets.DatasetFile (*, commit=None, client=None, path=None,
id=None, label=NOTHING, project=
renku.core.models.projects.Project
=
None, parent=None, added=NOTHING,
checksum=None, filename=NOTHING,
name=None, filesize=None, filetype=None,
url=None, based_on=None, external=False, source=None)
```

Represent a file in a dataset.

Method generated by attrs for class DatasetFile.

as_jsonld()

Create JSON-LD.

commit_sha

Return commit hash.

default_filename()

Generate default filename based on path.

default_id()

Configure calculated ID.

default_label()

Generate a default label.

default_url()

Generate default url based on project's ID.

entities

Yield itself.

classmethod from_jsonld(data)

Create an instance from JSON-LD data.

classmethod from_revision(client, path, revision='HEAD', parent=None, find_previous=True, **kwargs)

Return dependency from given path and revision.

full_path

Return full path in the current reference frame.

parent

Return the parent object.

set_client(client)

Sets the clients on this entity.

size_in_mb

Return file size in megabytes.

submodules

Proxy to client submodules.

update_commit(commit)

Set commit and update associated fields.

update_metadata(path, commit)

Update files metadata.

5.1.3 Provenance

Extract provenance information from the repository.

Activities

```
class renku.core.models.provenance.activities.Activity(*,  
                                                    commit=None,  
                                                    client=None, path=None,  
                                                    label=NOTHING, project:  
                                                    renku.core.models.projects.Project  
                                                    = None, id=None,  
                                                    message=NOTHING,  
                                                    was_informed_by=NOTHING,  
                                                    part_of=None, gen-  
                                                    erated=None, inval-  
                                                    idated=None, influ-  
                                                    enced=NOTHING,  
                                                    started_at_time=NOTHING,  
                                                    ended_at_time=NOTHING,  
                                                    agents=NOTHING)
```

Represent an activity in the repository.

Method generated by attrs for class Activity.

as_jsonld()

Create JSON-LD.

default_agents()

Set person agent to be the author of the commit.

default_ended_at_time()

Configure calculated properties.

default_generated()

Create default generated.

default_id()

Configure calculated ID.

default_influenced()

Calculate default values.

default_invalidated()

Entities invalidated by this Action.

default_label()

Generate a default label.

default_message()

Generate a default message.

default_started_at_time()

Configure calculated properties.

default_was_informed_by()

List parent actions.

classmethod from_jsonld(*data, client=None, commit=None*)

Create an instance from JSON-LD data.

classmethod from_yaml(*path, client=None, commit=None*)

Return an instance from a YAML file.

classmethod generate_id(*commitsha*)

Calculate action ID.

get_output_paths ()
Gets all output paths generated by this run.

nodes
Return topologically sorted nodes.

parents
Return parent commits.

paths
Return all paths in the commit.

removed_paths
Return all paths removed in the commit.

submodules
Proxy to client submodules.

to_yaml (path=None)
Write an instance to the referenced YAML file.

```
class renku.core.models.provenance.activities.ProcessRun(*,
    commit=None,
    client=None,
    path=None,
    label=NOTHING,
    project=renku.core.models.projects.Project,
    = None,
    id=None,
    message=NOTHING,
    was_informed_by=NOTHING,
    part_of=None,
    invalidated=None,
    influenced=NOTHING,
    started_at_time=NOTHING,
    ended_at_time=NOTHING,
    agents=NOTHING,
    generated=None,
    association=None,
    annotations=None,
    qualified_usage=None,
    run_parameter=None)
```

A process run is a particular execution of a Process description.

Method generated by attrs for class ProcessRun.

add_annotations (annotations)
Adds annotations from an external tool.

as_jsonld ()
Create JSON-LD.

default_agents ()
Set person agent to be the author of the commit.

default_ended_at_time ()
Configure calculated properties.

default_generated ()
Create default generated.

default_id ()
Configure calculated ID.

default_influenced()
Calculate default values.

default_invalidated()
Entities invalidated by this Action.

default_label()
Generate a default label.

default_message()
Generate a default message.

default_started_at_time()
Configure calculated properties.

default_was_informed_by()
List parent actions.

classmethod from_jsonld(*data*, *client=None*, *commit=None*)
Create an instance from JSON-LD data.

classmethod from_run(*run*, *client*, *path*, *commit=None*, *subprocess_index=None*, *update_commits=False*)
Convert a Run to a ProcessRun.

classmethod from_yaml(*path*, *client=None*, *commit=None*)
Return an instance from a YAML file.

classmethod generate_id(*commitsha*)
Calculate action ID.

get_output_paths()
Gets all output paths generated by this run.

nodes
Return topologically sorted nodes.

parents
Return parent commits.

paths
Return all paths in the commit.

plugin_annotations()
Adds Annotation`s from plugins to a ``ProcessRun.

removed_paths
Return all paths removed in the commit.

submodules
Proxy to client submodules.

to_yaml(*path=None*)
Write an instance to the referenced YAML file.

```

class renku.core.models.provenance.activities.WorkflowRun (*,
                                                           commit=None,
                                                           client=None,
                                                           path=None,      la-
                                                           bel=NOTHING,
                                                           project:
                                                           renku.core.models.projects.Project
                                                           = None, id=None,
                                                           message=NOTHING,
                                                           was_informed_by=NOTHING,
                                                           part_of=None, inval-
                                                           idated=None,      in-
                                                           fluenced=NOTHING,
                                                           started_at_time=NOTHING,
                                                           ended_at_time=NOTHING,
                                                           agents=NOTHING,
                                                           generated=None,
                                                           association=None,
                                                           annotations=None,
                                                           qualified_usage=None,
                                                           run_parameter=None,
                                                           processes=NOTHING)

```

A workflow run typically contains several subprocesses.

Method generated by attrs for class WorkflowRun.

add_annotations (*annotations*)

Adds annotations from an external tool.

as_jsonld ()

Create JSON-LD.

default_agents ()

Set person agent to be the author of the commit.

default_ended_at_time ()

Configure calculated properties.

default_generated ()

Create default *generated*.

default_id ()

Configure calculated ID.

default_influenced ()

Calculate default values.

default_invalidated ()

Entities invalidated by this Action.

default_label ()

Generate a default label.

default_message ()

Generate a default message.

default_started_at_time ()

Configure calculated properties.

default_was_informed_by ()

List parent actions.

classmethod from_jsonld (*data, client=None, commit=None*)
Create an instance from JSON-LD data.

classmethod from_run (*run, client, path, commit=None, subprocess_index=None, update_commits=False*)
Convert a Run to a WorkflowRun.

classmethod from_yaml (*path, client=None, commit=None*)
Return an instance from a YAML file.

classmethod generate_id (*commitsha*)
Calculate action ID.

get_output_paths ()
Gets all output paths generated by this run.

nodes
Yield all graph nodes.

parents
Return parent commits.

paths
Return all paths in the commit.

plugin_annotations ()
Adds Annotation`s from plugins to a ``ProcessRun.

removed_paths
Return all paths removed in the commit.

submodules
Proxy to client submodules.

subprocesses
Subprocesses of this WorkflowRun.

to_yaml (*path=None*)
Write an instance to the referenced YAML file.

Entities

class `renku.core.models.entities.Entity` (*, *commit=None, client=None, path=None, id=None, label=NOTHING, project: renku.core.models.projects.Project = None, parent=None, checksum: str = None*)

Represent a data value or item.

Method generated by attrs for class Entity.

default_id ()
Configure calculated ID.

default_label ()
Generate a default label.

entities
Yield itself.

classmethod from_revision (*client, path, revision='HEAD', parent=None, find_previous=True, **kwargs*)
Return dependency from given path and revision.

parent

Return the parent object.

set_client (*client*)

Sets the clients on this entity.

submodules

Proxy to client submodules.

```
class renku.core.models.entities.Collection(*, commit=None, client=None, path=None,
                                           id=None, label=NOTHING, project:
                                           renku.core.models.projects.Project = None,
                                           parent=None, checksum: str = None,
                                           members=None)
```

Represent a directory with files.

Method generated by attrs for class Collection.

default_id ()

Configure calculated ID.

default_label ()

Generate a default label.

default_members ()

Generate default members as entities from current path.

entities

Recursively return all files.

```
classmethod from_revision(client, path, revision='HEAD', parent=None, find_previous=True,
                           **kwargs)
```

Return dependency from given path and revision.

parent

Return the parent object.

set_client (*client*)

Sets the clients on this entity.

submodules

Proxy to client submodules.

Agents

```
class renku.core.models.provenance.agents.Person(*, client=None, name, email=None,
                                                  label=NOTHING, affiliation=None,
                                                  alternate_name=None, id=None)
```

Represent a person.

Method generated by attrs for class Person.

check_email (*attribute, value*)

Check that the email is valid.

default_id ()

Set the default id.

default_label ()

Set the default label.

classmethod from_commit (*commit*)

Create an instance from a Git commit.

classmethod from_dict (*obj*)

Create and instance from a dictionary.

classmethod from_git (*git*)

Create an instance from a Git repo.

classmethod from_jsonld (*data*)

Create an instance from JSON-LD data.

classmethod from_string (*string*)

Create an instance from a 'Name <email>' string.

full_identity

Return name, email, and affiliation.

short_name

Gives full name in short form.

class `renku.core.models.provenance.agents.SoftwareAgent` (*, *label*, *id*)

Represent executed software.

Method generated by attrs for class SoftwareAgent.

as_jsonld ()

Create JSON-LD.

classmethod from_commit (*commit*)

Create an instance from a Git commit.

classmethod from_jsonld (*data*)

Create an instance from JSON-LD data.

Relations

class `renku.core.models.provenance.qualified.Usage` (*, *entity*, *role=None*, *id=None*)

Represent a dependent path.

Method generated by attrs for class Usage.

as_jsonld ()

Create JSON-LD.

classmethod from_jsonld (*data*)

Create an instance from JSON-LD data.

classmethod from_revision (*client*, *path*, *revision='HEAD'*, ***kwargs*)

Return dependency from given path and revision.

class `renku.core.models.provenance.qualified.Generation` (*entity*, *role=None*, *, *activity=None*, *id=NOTHING*)

Represent an act of generating a file.

Method generated by attrs for class Generation.

activity

Return the activity object.

as_jsonld ()

Create JSON-LD.

```
default_id()
    Configure calculated ID.

classmethod from_jsonld(data)
    Create an instance from JSON-LD data.
```

5.1.4 Renku Workflow

Renku uses PROV-O and its own Renku ontology to represent workflows.

Run

Represents a workflow template.

```
class renku.core.models.workflow.run.OrderedSubprocess(*id, index: int, process)
    A subprocess with ordering.

    Method generated by attrs for class OrderedSubprocess.

    static generate_id(parent_id, index)
        Generate an id for an OrderedSubprocess.

class renku.core.models.workflow.run.OrderedSubprocessSchema(*args,          commit=None,
                                                                client=None,
                                                                **kwargs)

    OrderedSubprocess schema.

    Create an instance.

    class Meta
        Meta class.

        model
            alias of OrderedSubprocess

class renku.core.models.workflow.run.Run(*commit=None, client=None, path=None,
                                           id=None, label=NOTHING, project:
                                           renku.core.models.projects.Project = None,
                                           command: str = None, successcodes: list =
                                           NOTHING, subprocesses=NOTHING, argu-
                                           ments=NOTHING, inputs=NOTHING, out-
                                           puts=NOTHING, run_parameters=NOTHING,
                                           activity=None)
```

Represents a *renku run* execution template.

Method generated by attrs for class Run.

```
activity
    Return the activity object.

add_subprocess(subprocess)
    Adds a subprocess to this run.

as_jsonld()
    Create JSON-LD.

classmethod from_factory(factory, client, commit, path)
    Creates a Run from a CommandLineToolFactory.
```

classmethod `from_jsonld (data)`

Create an instance from JSON-LD data.

static `generate_id (client, identifier=None)`

Generate an id for an argument.

to_argv ()

Convert run into argv list.

to_stream_repr ()

Input/output stream representation.

update_id_and_label_from_commit_path (*client, commit, path, is_subprocess=False*)

Updates the `_id` and `_label` using supplied commit and path.

class `renku.core.models.workflow.run.RunSchema (*args, commit=None, client=None, **kwargs)`

Run schema.

Create an instance.

class `Meta`

Meta class.

model

alias of `Run`

Parameters

Represents a workflow template.

class `renku.core.models.workflow.parameters.CommandArgument (*, id=None, label=None, position: int = None, prefix: str = None, value: str = None)`

An argument to a command that is neither input nor output.

Method generated by attrs for class `CommandArgument`.

as_jsonld ()

Create JSON-LD.

default_label ()

Set default label.

classmethod `from_jsonld (data)`

Create an instance from JSON-LD data.

static `generate_id (run_id, position=None)`

Generate an id for an argument.

to_argv ()

String representation (sames as `cmd` argument).

class `renku.core.models.workflow.parameters.CommandArgumentSchema (*args, commit=None, client=None, **kwargs)`

`CommandArgument` schema.

Create an instance.

class Meta

Meta class.

model

alias of *CommandArgument*

class renku.core.models.workflow.parameters.**CommandInput** (*, *id=None, label=None, position: int = None, prefix: str = None, consumes, mapped_to=None*)

An input to a command.

Method generated by attrs for class CommandInput.

as_jsonld()

Create JSON-LD.

default_label()

Set default label.

classmethod from_jsonld(data)

Create an instance from JSON-LD data.

static generate_id(run_id, position=None)

Generate an id for an argument.

to_argv()

String representation (sames as cmd argument).

to_stream_repr()

Input stream representation.

class renku.core.models.workflow.parameters.**CommandInputSchema** (*args, *commit=None, client=None, **kwargs*)

CommandArgument schema.

Create an instance.

class Meta

Meta class.

model

alias of *CommandInput*

class renku.core.models.workflow.parameters.**CommandInputTemplate** (*, *id=None, label=None, position: int = None, prefix: str = None, consumes, mapped_to=None*)

Template for inputs of a Plan.

Method generated by attrs for class CommandInputTemplate.

as_jsonld()

Create JSON-LD.

default_label()

Set default label.

classmethod from_jsonld(data)

Create an instance from JSON-LD data.

static generate_id(plan_id, position=None, id_=None)

Generate an id for an argument.

to_argv()

String representation (sames as cmd argument).

to_stream_repr()

Input stream representation.

class renku.core.models.workflow.parameters.**CommandInputTemplateSchema** (*args, commit=None, client=None, **kwargs)

CommandInputTemplateSchema schema.

Create an instance.

class **Meta**

Meta class.

model

alias of *CommandInputTemplate*

class renku.core.models.workflow.parameters.**CommandOutput** (*, id=None, label=None, position: int = None, prefix: str = None, create_folder: bool = False, produces, mapped_to=None)

An output of a command.

Method generated by attrs for class CommandOutput.

as_jsonld()

Create JSON-LD.

default_label()

Set default label.

classmethod from_jsonld(data)

Create an instance from JSON-LD data.

static generate_id(run_id, position=None)

Generate an id for an argument.

to_argv()

String representation (sames as cmd argument).

to_stream_repr()

Input stream representation.

class renku.core.models.workflow.parameters.**CommandOutputSchema** (*args, commit=None, client=None, **kwargs)

CommandArgument schema.

Create an instance.

```
class Meta
    Meta class.
```

```
    model
        alias of CommandOutput
```

```
class renku.core.models.workflow.parameters.CommandOutputTemplate (*, id=None,
                                                                    la-
                                                                    bel=None,
                                                                    position:
                                                                    int = None,
                                                                    prefix: str =
                                                                    None, cre-
                                                                    ate_folder:
                                                                    bool =
                                                                    False,
                                                                    mapped_to=None,
                                                                    produces:
                                                                    str = None)
```

Template for outputs of a Plan.

Method generated by attrs for class CommandOutputTemplate.

```
as_jsonld()
    Create JSON-LD.
```

```
default_label()
    Set default label.
```

```
classmethod from_jsonld(data)
    Create an instance from JSON-LD data.
```

```
static generate_id(plan_id, position=None, id=None)
    Generate an id for an argument.
```

```
to_argv()
    String representation (sames as cmd argument).
```

```
to_stream_repr()
    Input stream representation.
```

```
class renku.core.models.workflow.parameters.CommandOutputTemplateSchema (*args,
                                                                    com-
                                                                    mit=None,
                                                                    client=None,
                                                                    **kwargs)
```

CommandOutputTemplateSchema schema.

Create an instance.

```
class Meta
    Meta class.
```

```
    model
        alias of CommandOutputTemplate
```

```
class renku.core.models.workflow.parameters.CommandParameter (*, id=None,
                                                                    label=None, posi-
                                                                    tion: int = None,
                                                                    prefix: str = None)
```

Represents a parameter for an execution template.

Method generated by attrs for class `CommandParameter`.

sanitized_id

Return `_id` sanitized for use in non-jsonld contexts.

```
class renku.core.models.workflow.parameters.CommandParameterSchema (*args,
                                                                    com-
                                                                    mit=None,
                                                                    client=None,
                                                                    **kwargs)
```

`CommandParameter` schema.

Create an instance.

class Meta

Meta class.

model

alias of `CommandParameter`

```
class renku.core.models.workflow.parameters.MappedIOStream(*,          client=None,
                                                           id=None, label=None,
                                                           stream_type: str)
```

Represents an IO stream (stdin, stdout, stderr).

Method generated by attrs for class `MappedIOStream`.

as_jsonld()

Create JSON-LD.

default_id()

Generate an id for a mapped stream.

default_label()

Set default label.

classmethod from_jsonld(data)

Create an instance from JSON-LD data.

```
class renku.core.models.workflow.parameters.MappedIOStreamSchema(*args, com-
                                                                    mit=None,
                                                                    client=None,
                                                                    **kwargs)
```

`MappedIOStream` schema.

Create an instance.

class Meta

Meta class.

model

alias of `MappedIOStream`

```
class renku.core.models.workflow.parameters.RunParameter(*, id=None, label=None,
                                                           name: str = None, value:
                                                           str = None, type: str =
                                                           None)
```

A run parameter that is set inside the script.

Method generated by attrs for class `RunParameter`.

as_jsonld()
Create JSON-LD.

default_label()
Set default label.

classmethod from_jsonld(data)
Create an instance from JSON-LD data.

static generate_id(run_id, name)
Generate an id.

```
class renku.core.models.workflow.parameters.RunParameterSchema (*args,    com-
                                                                    mit=None,
                                                                    client=None,
                                                                    **kwargs)
```

CommandParameter schema.

Create an instance.

class Meta
Meta class.

model
alias of *RunParameter*

5.1.5 Renku Workflow Conversion

Renku allows conversion of tracked workflows to runnable workflows in supported tools (Currently CWL)

CWL

Converter for workflows to cwl.

```
class renku.core.models.workflow.converters.cwl.CWLConverter
    Converts a Run to cwl file(s).
```

```
static convert (run, client, path=None)
    Convert the workflow to one ore more .cwl files.
```

5.1.6 Tools and Workflows

Manage creation of tools and workflows for workflow tracking.

Command-line tool

Represent a `CommandLineToolFactory` for tracking workflows.

```
class renku.core.models.cwl.command_line_tool.CommandLineToolFactory (command_line,  
                                                                    ex-  
                                                                    plicit_inputs=NOTHING,  
                                                                    ex-  
                                                                    plicit_outputs=NOTHING,  
                                                                    no_input_detection=False,  
                                                                    no_output_detection=False,  
                                                                    direc-  
                                                                    tory='',  
                                                                    work-  
                                                                    ing_dir='',  
                                                                    stdin=None,  
                                                                    stderr=None,  
                                                                    std-  
                                                                    out=None,  
                                                                    suc-  
                                                                    cess-  
                                                                    Codes=NOTHING,  
                                                                    an-  
                                                                    nota-  
                                                                    tions=None,  
                                                                    mes-  
                                                                    sages=None,  
                                                                    warn-  
                                                                    ings=None)
```

Command Line Tool Factory.

Method generated by attrs for class CommandLineToolFactory.

add_indirect_inputs ()

Read indirect inputs list and add them to explicit inputs.

add_indirect_outputs ()

Read indirect outputs list and add them to explicit outputs.

find_explicit_inputs ()

Yield explicit inputs and command line input bindings if any.

generate_process_run (*client, commit, path*)

Return an instance of ProcessRun.

guess_inputs (**arguments*)

Yield command input parameters and command line bindings.

guess_outputs (*candidates*)

Yield detected output and changed command input parameter.

guess_type (*value, ignore_filenames=None*)

Return new value and CWL parameter type.

is_existing_path (*candidate, ignore=None*)

Return a path instance if it exists in current directory.

iter_input_files (*basedir*)

Yield tuples with input id and path.

split_command_and_args ()

Return tuple with command and args from command line arguments.

validate_command_line (*attribute, value*)

Check the command line structure.

validate_path (*attribute, value*)

Path must exist.

watch (*client, no_output=False*)

Watch a Renku repository for changes to detect outputs.

`renku.core.models.cwl.command_line_tool.add_indirect_parameter` (*working_dir, name, value*)

Add a parameter to indirect parameters.

`renku.core.models.cwl.command_line_tool.delete_indirect_files_list` (*working_dir*)

Remove indirect inputs, outputs, and parameters list.

`renku.core.models.cwl.command_line_tool.get_indirect_inputs_path` (*client_path*)

Return path to file that contains indirect inputs list.

`renku.core.models.cwl.command_line_tool.get_indirect_outputs_path` (*client_path*)

Return path to file that contains indirect outputs list.

`renku.core.models.cwl.command_line_tool.get_indirect_parameters_path` (*client_path*)

Return path to file that contains indirect parameters list.

`renku.core.models.cwl.command_line_tool.read_indirect_parameters` (*working_dir*)

Read and return indirect parameters.

Annotation

Represent an annotation for a workflow.

class `renku.core.models.cwl.annotation.Annotation` (*, *id, body=None, source=None*)

Represents a custom annotation for a research object.

Method generated by attrs for class Annotation.

as_jsonld ()

Create JSON-LD.

classmethod from_jsonld (*data*)

Create an instance from JSON-LD data.

class `renku.core.models.cwl.annotation.AnnotationSchema` (*args, *commit=None, client=None, **kwargs*)

Annotation schema.

Create an instance.

class Meta

Meta class.

model

alias of `Annotation`

Parameter

Represent parameters from the Common Workflow Language.

```
class renku.core.models.cwl.parameter.CommandInputParameter (id=None, streamable=None, type='string', description=None, default=None, inputBinding=None)
```

An input parameter for a CommandLineTool.

Method generated by attrs for class CommandInputParameter.

```
classmethod from_cwl (data)  
    Create instance from type definition.
```

```
to_argv (**kwargs)  
    Format command input parameter as shell argument.
```

```
class renku.core.models.cwl.parameter.CommandLineBinding (position=None, prefix=None, separate: bool = True, itemSeparator=None, valueFrom=None, shellQuote: bool = True)
```

Define the binding behavior when building the command line.

Method generated by attrs for class CommandLineBinding.

```
to_argv (default=None)  
    Format command line binding as shell argument.
```

```
class renku.core.models.cwl.parameter.CommandOutputBinding (glob=None)  
    Define the binding behavior for outputs.
```

Method generated by attrs for class CommandOutputBinding.

```
class renku.core.models.cwl.parameter.CommandOutputParameter (id=None, streamable=None, type='string', description=None, format=None, outputBinding=None)
```

Define an output parameter for a CommandLineTool.

Method generated by attrs for class CommandOutputParameter.

```
class renku.core.models.cwl.parameter.InputParameter (id=None, streamable=None, type='string', description=None, default=None, inputBinding=None)
```

An input parameter.

Method generated by attrs for class InputParameter.

```
class renku.core.models.cwl.parameter.OutputParameter (id=None, streamable=None, type='string', description=None, format=None, outputBinding=None)
```

An output parameter.

Method generated by attrs for class OutputParameter.

```
class renku.core.models.cwl.parameter.Parameter (streamable=None)  
    Define an input or output parameter to a process.
```

Method generated by attrs for class Parameter.

```
class renku.core.models.cwl.parameter.RunParameter (name=None, value=None)
    Define a parameter for a Workflow that is not passed via command-line.
```

Method generated by attrs for class RunParameter.

```
class renku.core.models.cwl.parameter.WorkflowOutputParameter (id=None, stream-
    able=None,
    type='string', de-
    scription=None,
    format=None,
    outputBind-
    ing=None,
    output-
    Source=None)
```

Define an output parameter for a Workflow.

Method generated by attrs for class WorkflowOutputParameter.

```
renku.core.models.cwl.parameter.convert_default (value)
    Convert a default value.
```

Types

Represent the Common Workflow Language types.

```
class renku.core.models.cwl.types.Directory (path=None, listing=NOTHING)
    Represent a directory.
```

Method generated by attrs for class Directory.

```
class renku.core.models.cwl.types.Dirent (entryname=None, entry=None, writable=False)
    Define a file or subdirectory.
```

Method generated by attrs for class Dirent.

```
class renku.core.models.cwl.types.File (path)
    Represent a file.
```

Method generated by attrs for class File.

Workflow

Represent workflows from the Common Workflow Language.

```
class renku.core.models.cwl.workflow.Workflow (steps=NOTHING)
    Define a workflow representation.
```

Method generated by attrs for class Workflow.

```
add_step (**kwargs)
    Add a workflow step.
```

```
class renku.core.models.cwl.workflow.WorkflowStep (run, id=NOTHING, in_=None,
    out=None)
```

Define an executable element of a workflow.

Method generated by attrs for class WorkflowStep.

```
renku.core.models.cwl.workflow.convert_run (value)
    Convert value to CWLClass if dict is given.
```

5.1.7 File References

Manage names of Renku objects.

class `renku.core.models.refs.LinkReference` (*client, name*)
Manage linked object names.

Method generated by attrs for class LinkReference.

REFS = 'refs'

Define a name of the folder with references in the Renku folder.

classmethod `check_ref_format` (*name, no_slashes=False*)
Ensures that a reference name is well formed.

It follows Git naming convention:

- any path component of it begins with “.”, or
- it has double dots “..”, or
- it has ASCII control characters, or
- it has “:”, “?”, “[“, “|”, “^”, “~”, SP, or TAB anywhere, or
- it has “*” anywhere, or
- it ends with a “/”, or
- it ends with “.lock”, or
- it contains a “@{” portion

classmethod `create` (*client, name, force=False*)
Create symlink to object in reference path.

delete ()
Delete the reference at the given path.

classmethod `iter_items` (*client, common_path=None*)
Find all references in the repository.

name_validator (*attribute, value*)
Validate reference name.

path
Return full reference path.

reference
Return the path we point to relative to the client.

rename (*new_name, force=False*)
Rename self to a new name.

set_reference (*reference*)
Set ourselves to the given reference path.

5.2 Repository API

This API is built on top of Git and Git-LFS.

Renku repository management.

```
class renku.core.management.LocalClient (path=<function default_path>,
                                         renku_home='.renku', parent=None, com-
                                         mit_activity_cache=NOTHING, activ-
                                         ity_index=None, remote_cache=NOTHING,
                                         migration_type=<MigrationType.ALL: 7>, exter-
                                         nal_storage_requested=True, *, data_dir='data')
```

A low-level client for communicating with a local Renku repository.

Method generated by attrs for class LocalClient.

5.2.1 Datasets

Client for handling datasets.

```
class renku.core.management.datasets.DatasetsApiMixin
```

Client for handling datasets.

Method generated by attrs for class DatasetsApiMixin.

```
CACHE = 'cache'
```

Directory to cache transient data.

```
DATASETS = 'datasets'
```

Directory for storing dataset metadata in Renku.

```
DATASETS_PROVENANCE = 'dataset.json'
```

File for storing datasets' provenance.

```
DATASET_IMAGES = 'dataset_images'
```

Directory for dataset images.

```
POINTERS = 'pointers'
```

Directory for storing external pointer files.

```
add_data_to_dataset (dataset, urls, force=False, overwrite=False, sources=(), destination="",
                    ref=None, external=False, extract=False, all_at_once=False, destina-
                    tion_names=None)
```

Import the data into the data directory.

```
add_dataset_tag (dataset, tag, description="", force=False)
```

Adds a new tag to a dataset.

Validates if the tag already exists and that the tag follows the same rules as docker tags. See <https://docs.docker.com/engine/reference/commandline/tag/> for a documentation of docker tag syntax.

Raises errors.ParameterError

```
clear_temporary_datasets_path ()
```

Clear path to Renku dataset metadata directory.

```
create_dataset (name=None, title=None, description=None, creators=None, keywords=None, im-
                ages=None, safe_image_paths=[])
```

Create a dataset.

```
dataset_commits (dataset, max_results=None)
```

Gets the newest commit for a dataset or its files.

Commits are returned sorted from newest to oldest.

```
datasets
```

Return mapping from path to dataset.

datasets_from_commit (*commit=None*)
Return datasets defined in a commit.

datasets_provenance
Return dataset provenance if available.

datasets_provenance_path
Path to store activity files.

get_dataset_path (*name*)
Get dataset path from name.

get_datasets_metadata_files ()
Return a generator of datasets metadata files.

has_datasets_provenance ()
Return true if dataset provenance exists.

has_external_files ()
Return True if project has external files.

initialize_datasets_provenance ()
Create empty dataset provenance file.

is_protected_path (*path*)
Checks if a path is a protected path.

is_using_temporary_datasets_path ()
Return true if temporary datasets path is set.

load_dataset (*name=None, strict=False*)
Load dataset reference file.

load_dataset_from_path (*path, commit=None*)
Return a dataset from a given path.

load_dataset_from_provenance (*name, strict=False*)
Load latest dataset's metadata from dataset provenance file.

move_files (*files, to_dataset, commit*)
Move files and their metadata from one or more datasets to a target dataset.

prepare_git_repo (*url, ref=None*)
Clone and cache a Git repo.

remove_dataset_tags (*dataset, tags*)
Removes tags from a dataset.

remove_datasets_provenance_file ()
Remove dataset provenance.

static_remove_file (*filepath*)
Remove a file/symlink and its pointer file (for external files).

renku_dataset_images_path
Return a `Path` instance of Renku dataset metadata folder.

renku_datasets_path
Return a `Path` instance of Renku dataset metadata folder.

renku_pointers_path
Return a `Path` instance of Renku pointer files folder.

set_dataset_images (*dataset, images, safe_image_paths=[]*)

Set the images on a dataset.

set_temporary_datasets_path (*path*)

Set path to Renku dataset metadata directory.

update_dataset_git_files (*files, ref, delete=False*)

Update files and dataset metadata according to their remotes.

Parameters

- **files** – List of files to be updated
- **delete** – Indicates whether to delete files or not

Returns List of files that should be deleted

update_datasets_provenance (*dataset, remove=False*)

Update datasets provenance for a dataset.

update_external_files (*records*)

Update files linked to external storage.

with_dataset (*name=None, create=False, immutable=False*)

Yield an editable metadata object for a dataset.

with_dataset_provenance (*name=None, create=False*)

Yield a dataset's metadata from dataset provenance.

5.2.2 Repository

Client for handling a local repository.

class `renku.core.management.repository.PathMixin` (*path=<function default_path>*)

Define a default path attribute.

Method generated by attrs for class PathMixin.

class `renku.core.management.repository.RepositoryApiMixin` (*renku_home='.renku', parent=None, commit_activity_cache=NOTHING, activity_index=None, remote_cache=NOTHING, migration_type=<MigrationType.ALL: 7>, *, data_dir='data')*

Client for handling a local repository.

Method generated by attrs for class RepositoryApiMixin.

ACTIVITY_INDEX = `'activity_index.yaml'`

Caches activities that generated a path.

DEPENDENCY_GRAPH = `'dependency.json'`

File for storing dependency graph.

DOCKERFILE = `'Dockerfile'`

Name of the Dockerfile in the repo.

LOCK_SUFFIX = `'.lock'`

Default suffix for Renku lock file.

METADATA = `'metadata.yml'`

Default name of Renku config file.

PROVENANCE_GRAPH = `'provenance.json'`

File for storing ProvenanceGraph.

WORKFLOW = `'workflow'`

Directory for storing workflow in Renku.

activities_for_paths (*paths, file_commit=None, revision='HEAD'*)

Get all activities involving a path.

activity_index_path

Path to the activity filepath cache.

add_to_activity_index (*activity*)

Add an activity and its generations to the cache.

cwl_prefix

Return a CWL prefix.

data_dir = `None`

Define a name of the folder for storing datasets.

dependency_graph

Return dependency graph if available.

dependency_graph_path

Path to the dependency graph file.

docker_path

Path to the Dockerfile.

find_previous_commit (*paths, revision='HEAD', return_first=False, full=False*)

Return a previous commit for a given path starting from *revision*.

Parameters

- **revision** – revision to start from, defaults to HEAD
- **return_first** – show the first commit in the history
- **full** – return full history

Raises `KeyError` – if path is not present in the given commit

has_graph_files ()

Return true if dependency or provenance graph exists.

import_from_template (*template_path, metadata, force=False*)

Render template files from a template directory.

init_repository (*force=False, user=None*)

Initialize an empty Renku repository.

initialize_graph ()

Create empty graph files.

is_project_set ()

Return if project is set for the client.

is_workflow (*path*)

Check if the path is a valid CWL file.

latest_agent

Returns latest agent version used in the repository.

lock

Create a Renku config lock.

migration_type

Type of migration that is being executed on this client.

parent = None

Store a pointer to the parent repository.

path_activity_cache

Cache of all activities and their generated paths.

process_and_store_run (*command_line_tool, name, client*)

Create Plan and Activity from CommandLineTool and store them.

process_commit (*commit=None, path=None*)

Build an *Activity*.

Parameters

- **commit** – Commit to process. (default: HEAD)
- **path** – Process a specific CWL file.

project

Return the Project instance.

provenance_graph_path

Path to store activity files.

remote

Return host, owner and name of the remote if it exists.

remove_graph_files ()

Remove all graph files.

renku_home = None

Define a name of the Renku folder (default: `.renku`).

renku_metadata_path

Return a `Path` instance of Renku metadata file.

renku_path = None

Store a `Path` instance of the Renku folder.

resolve_in_submodules (*commit, path*)

Resolve filename in submodules.

subclients (*parent_commit*)

Return mapping from submodule to client.

submodules

Return list of submodules it belongs to.

template_checksums

Return a `Path` instance to the template checksums file.

update_graphs (*activity_run*)

Update Dependency and Provenance graphs from a `ProcessRun/WorkflowRun`.

with_commit (*commit*)

Yield the state of the repo at a specific commit.

with_metadata (*read_only=False, name=None*)

Yield an editable metadata object.

with_workflow_storage ()

Yield a workflow storage.

workflow_names

Return index of workflow names.

workflow_path

Return a Path instance of the workflow folder.

`renku.core.management.repository.default_path` (*path='.'*)

Return default repository path.

`renku.core.management.repository.path_converter` (*path*)

Converter for path in PathMixin.

5.2.3 Git Internals

Wrap Git client.

class `renku.core.management.git.GitCore`

Wrap Git client.

Method generated by attrs for class GitCore.

candidate_paths

Return all paths in the index and untracked files.

commit (*commit_only=None, commit_empty=True, raise_if_empty=False, commit_message=None, abbreviate_message=True*)

Automatic commit.

dirty_paths

Get paths of dirty files in the repository.

ensure_clean (*ignore_std_streams=False*)

Make sure the repository is clean.

ensure_unstaged (*path*)

Ensure that path is not part of git staged files.

ensure_untracked (*path*)

Ensure that path is not part of git untracked files.

find_attr (**paths*)

Return map with path and its attributes.

find_ignored_paths (**paths*)

Return ignored paths matching `.gitignore` file.

modified_paths

Return paths of modified files.

remove_unmodified (*paths, autocommit=True*)

Remove unmodified paths and return their names.

repo = None

Store an instance of the Git repository.

setup_credential_helper ()
Setup git credential helper to `cache` if not set already.

transaction (*clean=True, commit=True, commit_empty=True, commit_message=None, commit_only=None, ignore_std_streams=False, raise_if_empty=False*)
Perform Git checks and operations.

worktree (*path=None, branch_name=None, commit=None, merge_args=('--ff-only',)*)
Create new worktree.

`renku.core.management.git.get_mapped_std_streams` (*lookup_paths, streams=('stdin', 'stdout', 'stderr')*)

Get a mapping of standard streams to given paths.

Git utilities.

class `renku.core.models.git.GitURL` (*href, pathname=None, protocol='ssh', hostname='localhost', username=None, password=None, port=None, owner=None, name=None, regex=None*)

Parser for common Git URLs.

Method generated by attrs for class `GitURL`.

image
Return image name.

classmethod `parse` (*href*)
Derive URI components.

class `renku.core.models.git.Range` (*start, stop*)
Represent parsed Git revision as an interval.

Method generated by attrs for class `Range`.

classmethod `rev_parse` (*git, revision*)
Parse revision string.

`renku.core.models.git.filter_repo_name` (*repo_name*)
Remove the `.git` extension from the repo name.

`renku.core.models.git.get_user_info` (*git*)
Get Git repository's owner name and email.

5.3 Plugin Support

5.3.1 Runtime Plugins

Runtime plugins are supported using the `pluggy` library.

Runtime plugins can be created as Python packages that contain the respective entry point definition in their `setup.py` file, like so:

```
from setuptools import setup

setup(
    ...
    entry_points={"renku": ["name_of_plugin = myproject.pluginmodule"]},
    ...
)
```

where *myproject.pluginmodule* points to a Renku *hookimpl* e.g.:

```
from renku.core.plugins import hookimpl

@hookimpl
def plugin_hook_implementation(param1, param2):
    ...
```

5.3.2 renku run hooks

Plugin hooks for renku run customization.

`renku.core.plugins.run.cmdline_tool_annotations(tool)`

Plugin Hook to add Annotation entry list to a WorkflowTool.

Parameters `run` – A WorkflowTool object to get annotations for.

Returns A list of `renku.core.models.cwl.annotation.Annotation` objects.

`renku.core.plugins.run.pre_run(tool)`

Plugin Hook that gets called at the start of a `renku run` call.

Can be used to setup plugins that get executed during the run.

Parameters `run` – A WorkflowTool object that will get executed by `renku run`.

`renku.core.plugins.run.process_run_annotations(run)`

Plugin Hook to add Annotation entry list to a ProcessRun.

Parameters `run` – A ProcessRun object to get annotations for.

Returns A list of `renku.core.models.cwl.annotation.Annotation` objects.

5.3.3 CLI Plugins

Command-line interface plugins are supported using the *click-plugins* <<https://github.com/click-contrib/click-plugins>> library.

As in case the runtime plugins, command-line plugins can be created as Python packages that contain the respective entry point definition in their *setup.py* file, like so:

```
from setuptools import setup

setup(
    ...
    entry_points={"renku.cli_plugins": ["mycmd = myproject.pluginmodule:mycmd"]},
    ...
)
```

where *myproject.pluginmodule:mycmd* points to a click command e.g.:

```
import click

@click.command()
def mycmd():
    ...
```

6.1 <https://github.com/SwissDataScienceCenter/renku-python/compare/v0.14.1...v0.14.2> (2021-04-16)

6.1.1 Highlights

- Ability to update local project from its template and to update the Dockerfile to install the current version of renku-python using *renku migrate*.
- Support for Unicode paths in *renku run* (including emojis).

6.1.2 Bug Fixes

- **cli:** fix renku rerun/update with unicode input/output paths (#1963) (9859b62)
- **service:** fix project_clone with git ref specified (#2008) (c072286)

6.1.3 Features

- **cli:** support template and docker migration (#2019) (ed87770)
- **dataset:** support moving files between datasets with renku mv (#1993) (a715b70)

6.2 0.14.1 (2021-03-24)

6.2.1 Bug Fixes

- **core:** Add error handling if push of temporary branch fails (#1979) (f8d7285)
- **core:** fix handling of '@' in filenames (#1982) (41316b4)

- **core:** fix template update if same filename was added locally (#1974) (5b47ddc)
- **core:** fixes save and push to correctly handle merge conflicts (#1925) (fdac171)
- **service:** sync service cache with remote before operations to prevent cache getting out of sync (#1972) (34ec5d6)

6.2.2 Features

- **dataset:** dataset import enhancements (#1970) (b3df7b8)
- **service:** renku service up/down/ps/restart/logs commands (#1899) (d9e49ae)
- **service:** add support for storing remote dataset images in the repo (#1878) (3862c2e)

6.3 0.14.0 (2021-03-05)

6.3.1 Bug Fixes

- **core:** call git commands for batches of files to prevent hitting argument length limits (#1893) (deaf055)
- **dataset:** change renku dataset import to move temporary files and become more resilient to errors (#1894) (279407e)
- **service** correctly address HTTP server errors (#1872) (2fd5052)
- **service** correctly handle ref on project.clone (#1888) (7f30404)
- **service** use project_id as part of project filesystem path (#1754) (391a14a)

6.3.2 Features

- **cli:** add renku storage migrate command to migrate git files to lfs (#1869) (bed1358)
- **cli:** add service component management commands (#1867) (928baf9)
- **core:** exclude renku metadata from being added to git lfs (#1898) (8046edb)
- **core:** add oauth authentication for KG access (#1881) (a568d31)
- **dataset:** improve naming for imported datasets (#1900) (9beb654)
- **service:** add build graph endpoint (#1571) (a7bfe3d)
- **service:** add renku config endpoints (#1834) (c09ca6b)
- **service:** add helm 3 values schema to chart (#1835) (57f6aee)
- **service** add root redirect to swagger docs (#1871) (1abd4f6)
- **service:** add support for adding images to datasets (#1850) (c3caafd)

6.4 0.13.0 (2021-01-29)

6.4.1 Bug Fixes

- **core:** fix renku save with deleted files (#1849) (93348f9)

- **core:** migration error when multiple outputs bind to the same input (#1832) (bb19b47)
- **core:** output git lfs error messages when there is an error (#1838) (e2b5421)
- **service:** reset cache after failed push (#1836) (f41df17)

6.4.2 Features

- **cli:** add dataset show command (#1798) (31e87f7)
- **core:** cache for workflow migrations (#1853) (f240d95)
- **service:** gzip decompression support (#1784) (59db473)
- **service:** use jwt sub claim as user-id (#1793) (06f815a)

6.5 ‘ [<https://github.com/SwissDataScienceCenter/renku-python/compare/v0.12.2...v0.12.3>](https://github.com/SwissDataScienceCenter/renku-python/compare/v0.12.2...v0.12.3) ‘__ (2021-01-05)

6.5.1 Bug Fixes

- **core:** fix gitlab ID parsing when GITLAB_BASE_URL is set without port (#1823) (4f94165)
- **service:** add datasets.remove to swagger docs (#1778) (631e6f5)
- **service:** correctly handle cloning of project with no commits (#1790) (440b238)

6.6 ‘ [<https://github.com/SwissDataScienceCenter/renku-python/compare/v0.12.1...v0.12.2>](https://github.com/SwissDataScienceCenter/renku-python/compare/v0.12.1...v0.12.2) ‘__ (2020-12-02)

6.6.1 Bug Fixes

- **core:** correctly generate project id for gitlab (sub)groups (#1746) (3fc29ad)
- **core:** fixes renku save to work with already staged changes (#1739) (1a8b7ad)
- **core:** adds pre-commit hook message for unsupported projects (#1730) (7f1731d)
- **service:** removes chdir calls in service (#1767) (4da22cb)

6.6.2 Features

- **api:** adds user-api parameters support (#1723) (6ee2862)
- **cli:** adds migrationscheck command (#1761) (b33ed35)
- **cli:** automatically track files in git-lfs if necessary (#1775) (866163a)
- **cli:** better error messages for renku clone (#1738) (78bb2ad)
- **core:** shorten commit messages to 100 characters for readability (#1749) (af50947)
- **service:** move user identification to jwt (#1520) (d45c4c3)

6.7 ‘ <https://github.com/SwissDataScienceCenter/renku-python/compare/v0.12.0...v0.12.1>’ __ (2020-11-16)

6.7.1 Bug Fixes

- **core:** re-raise renku handled exception on network failure (#1623) (4856a05)
- **dataset:** no commit if nothing is edited (#1706) (a68edf6)
- **service:** correctly determine resource age (#1695) (40153f0)
- **service:** correctly set project_name slug on project create (#1691) (234e1b3)
- **service:** set template version and metadata correctly (#1708) (ed98be3)

6.7.2 Features

- renku API (#1665) (949a4aa)

6.8 ‘ <https://github.com/SwissDataScienceCenter/renku-python/compare/v0.11.6...v0.12.0>’ __ (2020-11-03)

6.8.1 Bug Fixes

- **core:** fix bug where remote_cache caused project ids to leak (#1618) (3ef04fb)
- **core:** fix graph building for nodes with same subpath (#1625) (7cae9be)
- **core:** fix importing a dataset referenced from non-existent projects (#1574) (92b8bf8)
- **core:** fix old dataset migration and activity dataset outputs (#1603) (a5339e2)
- **core:** fix project migration getting overwritten with old metadata (#1581) (c5a5960)
- **core:** fix update creating a commit when showing help (#1627) (529e582)
- **core:** fixes git encoding of paths with unicode characters (#1538) (053dac9)
- **core:** make Run migration ids unique by relative path instead of absolute (#1573) (cf96310)
- **dataset:** broken directory hierarchy after renku dataset imports (#1576) (9dcffce)
- **dataset:** deserialization error (#1675) (420653f)
- **dataset:** error when adding same file multiple times (#1639) (05bfde7)
- **dataset:** explicit failure when cannot pull LFS objects (#1590) (3b05816)
- **dataset:** invalid generated name in migration (#1593) (89b2e43)
- **dataset:** remove blank nodes (#1602) (478f08c)
- **dataset:** set isBasedOn for renku datasets (#1617) (3aee6b8)
- **dataset:** update local files metadata when overwriting (#1582) (59eaf25)
- **dataset:** various migration issues (#1620) (f24c2e4)
- **service:** correctly set job timeout (#1677) (25f0eb6)

- **service:** dataset rm endpoint supports new core API (#1622) (e71916e)
- **service:** push to protected branches (#1614) (34c7f92)
- **service:** raise exception on uninitialized projects (#1624) (a2025c3)

6.8.2 Features

- **cli:** add click plugin support (#1604) (47b007f)
- **cli:** adds consistent behaviour for cli commands (#1523) (20b7248)
- **cli:** show lfs status of dataset files (#1575) (a1c3e2a)
- **cli:** verbose output for renku show (#1524) (dae968c)
- **core:** Adds renku dataset update for Zenodo and Dataverse (#1331) (e38c51f)
- **dataset:** list dataset description (#1588) (7e13857)
- **service:** adds template and dockerfile migration to migration endpoint (#1509) (ea01795)
- **service:** adds version endpoint (#1548) (6193df6)

6.9 0.11.6 (2020-10-16)

6.9.1 Bug Fixes

- **core:** fix bug where remote_cache caused project ids to leak (#1618) (3ef04fb)
- **dataset:** fix a bug where datasets imported from renku project won't update (#1615) (309eb2f)
- **service:** fixes pushing to protected branches (#1614) (34c7f92)

6.10 0.11.5 (2020-10-13)

6.10.1 Bug Fixes

- **core:** fix importing a dataset referenced from non-existent projects (#1574) (4bb13ef)
- **core:** fixes git encoding of paths with unicode characters (#1538) (9790707)
- **dataset:** fix broken directory hierarchy after renku dataset imports (#1576) (41e3e72)
- **dataset:** abort importing a dataset when cannot pull LFS objects (#1590) (9877a98)
- **dataset:** fix invalid dataset name after migration (#1593) (c7ec249)
- **dataset:** update dataset files metadata when adding and overwriting local files (#1582) (0a23e82)

6.11 0.11.4 (2020-10-05)

6.11.1 Bug Fixes

- **core:** fix project migration getting overwritten with old metadata (#1580) (dcc1541)

6.12 0.11.3 (2020-09-29)

6.12.1 Bug Fixes

- **core:** make Run migration ids unique by relative path instead of absolute (686b9f9)

6.13 0.11.2 (2020-09-24)

6.13.1 Bug Fixes

- **cli:** fixes libxslt dependency in docker image (#1534) (491bae7)
- **core:** fixes 'doi:...' import (#1536) (f653c79)
- **core:** fixes duplicate 'renku:Run' ids on repeat execution of migrations (#1532) (4ce6f3c)

6.13.2 Features

- **cli:** show existing paths when initializing non-empty dir (#1535) (07c559f)
- **core:** follow URL redirections for dataset files (#1516) (5a37b3c)
- **dataset:** flattened JSON-LD metadata (#1518) (458ddb9)
- **service:** add additional template parameters (#1469) (6372a32)
- **service:** adds additional fields to datasets listings (#1508) (f8a395f)
- **service:** adds project details and renku operation on jobs endpoint (#1492) (6b3fafd)
- **service:** execute read operations via git remote (#1488) (84a0eb3)
- **workflow:** avoid unnecessary parent runs (#1476) (b908ffd)

6.14 0.11.1 (2020-08-18)

6.14.1 Bug Fixes

- fixes shacl for DatasetFile when used inside a qualifiedGeneration (#1477) (99dd4a4)

6.15 0.11.0 (2020-08-14)

6.15.1 Bug Fixes

- **cli:** disable version check in githook calls (#1300) (5132db3)
- **core:** fix paths in migration of workflows (#1371) (8c3d34b)
- **core:** Fixes SoftwareAgent person context (#1323) (a207a7f)
- **core:** Only update project metadata if any migrations were executed (#1308) (1056a03)
- **service:** adds more custom logging and imp. except handling (#1435) (6c3adb5)

- **service:** fixes handlers for internal loggers (#1433) (a312f7c)
- **service:** move project_id to query string on migrations check (#1367) (0f89726)
- **tests:** integration tests (#1351) (3974a39)

6.15.2 Features

- **cli:** Adds renku save command (#1273) (4ddc1c2)
- **cli:** prompt for missing variables (1e1d408), closes #1126
- **cli:** Show detailed commands for renku log output (#1345) (19fb819)
- **core:** Calamus integration (#1281) (bda538f)
- **core:** configurable data dir (#1347) (e388773)
- **core:** disabling of inputs/outputs auto-detection (#1406) (3245ca0)
- **core:** migration check in core (#1320) (4bc52f4)
- **core:** Move workflow serialisation over to calamus (#1386) (f0fbc49)
- **core:** save and load workflow as jsonld (#1185) (d403289)
- **core:** separate models for migrations (#1431) (127d606)
- **dataset:** source and url for DatasetFile (#1451) (b4fa5db)
- **service:** added endpoints to execute all migrations on a project (#1322) (aca8cc2)
- **service:** adds endpoint for explicit migrations check (#1326) (146b1a7)
- **service:** adds source and destination versions to migrations check (#1372) (ea76b48)
- decode base64 headers (#1407) (9901cc3)
- **service:** adds endpoints for dataset remove (#1383) (289e4b9)
- **service:** adds endpoints for unlinking files from a dataset (#1314) (1b78b16)
- **service:** async migrations execution (#1344) (ff66953)
- **service:** create new projects from templates (#1287) (552f85c), closes #862

6.16 0.10.5 (2020-07-16)

6.16.1 Bug Fixes

- **core:** Pin dependencies to prevent downstream dependency updates from breaking renku. Fix pyshacl dependency. (#785) (30beedd)
- **core:** Fixes SoftwareAgent person context. (#1323) (fa62f58)

6.17 0.10.4 (2020-05-18)

6.17.1 Bug Fixes

- **dataset:** update default behaviour and messaging on dataset unlink (#1275) (98d6728)

- **dataset:** correct url in different domain (#1211) (49e8b8b)

6.17.2 Features

- **cli:** Adds warning messages for LFS, fix output redirection (#1199) (31969f5)
- **core:** Adds lfs file size limit and lfs ignore file (#1210) (1f3c81c)
- **core:** Adds renku storage clean command (#1235) (7029400)
- **core:** git hook to avoid committing large files (#1238) (e8f1a8b)
- **core:** renku doctor check for lfs migrate info (#1234) (480da06)
- **dataset:** fail early when external storage not installed (#1239) (e6ea6da)
- **core:** project clone API support for revision checkout (#1208) (74116e9)
- **service:** protected branches support (#1222) (8405ce5)
- **dataset:** doi variations for import (#1216) (0f329dd)
- **dataset:** keywords in metadata (#1209) (f98a800)
- **dataset:** no failure when adding ignored files (#1213) (b1e275f)
- **service:** read template manifest (#1254) (7eac85b)

6.18 0.10.3 (2020-04-22)

6.18.1 Bug Fixes

- **dataset:** avoid recursive addition of data directory (#1163) (79e3b03)
- **dataset:** commit after unlinking files (#1120) (97e8754)
- **dataset:** Dataverse export (#1028) (737cecf)

6.18.2 Features

- **core:** CLI warning when in non-root directory (#1162) (115e462)
- **dataset:** migrate submodule-based datasets (#1092) (dba20c4)
- **dataset:** no failure when adding existing files (#1177) (a68dcb7)
- **dataset:** remove `-link` flag (#1164) (969d4f8)
- **dataset:** show file size in ls-files (#1123) (0951930)
- **datasets:** specify title on dataset creation (#1204) (fb70ac5)
- **init:** read and display template variables (#1134) (0f86dc5), closes #1126
- **service:** add remote files to dataset (#1139) (f6bebfe)

6.19 0.10.1 (2020-03-31)

6.19.1 Bug Fixes

- **core:** directory input regression (#1155) (b17c843)
- **dataset:** correct url when importing with DOI (#1156) (025b735)

6.19.2 Features

- renku init options refactor (#1127) (78b208b)
- **datasets:** add files from dropbox (#1135) (bf5f2db)

6.20 0.10.0 (2020-03-25)

This release brings about several important Dataset features:

- importing renku datasets (#838)
- working with data external to the repository (#974)
- editing dataset metadata (#1111)

Please see the [Dataset documentation](#) for details.

Additional features were implemented for the backend service to facilitate a smoother user experience for dataset file manipulation.

IMPORTANT: starting with this version, a new metadata migration mechanism is in place (#1003). Renku commands will insist on migrating a project immediately if the metadata is found to be outdated.

6.20.1 Bug Fixes

- **cli:** consistently show correct contexts (#1096) (b333f0f)
- **dataset:** `--no-external-storage` flag not working (#1130) (c183e97)
- **dataset:** commit only updated dataset files (#1116) (d9739df)
- **datasets:** fixed importing large amount of small files (#1119) (8d61473)
- **datasets:** raises correct error message on import of protected dataset (#1112) (e579904)

6.20.2 Features

- **core:** new migration mechanism (#1003) (1cc33d4)
- **dataset:** adding external data without copying (#974) (6a17512)
- **dataset:** bypass import confirmation (#1124) (947210a)
- **dataset:** import renku datasets (#838) (6aa3651)
- **dataset:** metadata edit (#1111) (66cfbbc)
- **dataset:** wildcard support when adding data from git (#1128) (baa1c9f)

6.21 0.9.1 (2020-02-24)

6.21.1 Bug Fixes

- added test utility functions and cleanup (#1014) (f41100d)
- cache instance cleanup (#1051) (12f5446)
- enable dataset cmd in sub directories (#1012) (e3191e1)
- fields with default need to come last (#1046) (649b159)
- fixes renku show sibling handling with no paths (#1026) (8df678f)
- flush old keys for user projects and files (#1002) (7438c73)
- generate https IDs for entities instead of file:// (#1009) (87f7750)
- handle errors correctly (#1040) (950eeac)
- improved list datasets and files (#1034) (fd96d68)
- pin idna to 2.8 (#1020) (19ea7af)
- resync repo after import action (#1052) (b38341b)
- standardize test assertions (#1016) (16e8e63)
- temporarily disable integration tests (#1036) (8c8fd7a)
- updated readme to include local testing (#1000) (351a650)
- run tests via pipenv run commands (#999) (d8095e3)

6.21.2 Features

- **svc:** adds job details endpoint (#1050) (9c58a08)
- **svc:** added list user jobs endpoint (#1001) (f3c200c)
- **svc:** dataset import via service (#1023) (d6c670a)

6.22 0.9.0 (2020-02-07)

6.22.1 Bug Fixes

- adds git user check before running renku init (#892) (2e52dff)
- adds sorting to file listing (#960) (bcf6bcd)
- avoid empty commits when adding files (#842) (8533a7a)
- Fixes dataset naming (#898) (418deb3)
- Deletes temporary branch after renku init --force (#887) (eac0463)
- enforces label on SoftwareAgent (#869) (71badda)
- Fixes JSON-LD translation and related issues (#846) (65e5469)
- Fixes renku run error message handling (#961) (81d31ff)

- Fixes renku update workflow failure handling and renku status error handling (#888) (3879124)
- Fixes sameAs property to follow schema.org spec (#944) (291380e)
- handle missing renku directory (#989) (f938be9)
- resolves symlinks when pulling LFS (#981) (68bd8f5)
- serializes all zenodo metadata (#941) (787978a)
- Fixes various bugs in dataset import (#882) (be28bf5)

6.22.2 Features

- add project initialization from template (#809) (4405744)
- added renku service with cache and datasets (#788) (7a7068d), closes #767 #846
- Adds protection for renku relevant paths in dataset add (#939) (a3c02e8)
- Adds prov:Invalidated output to renku log (008ab20)
- better UX when adding to a dataset (#911) (c6ac967)
- check for required git hooks (#854) (54ba91d)
- Dataverse export (#909) (7e9e647)
- improve dataset command output (#927) (c7639d3)
- metadata on dataset creation (#850) (b357ee7)
- Plugin support for renku-run (#883) (7dbda83)
- python 3.8 compatibility (#861) (4aaac8d)
- SHACL Validation (#767) (255a01d)
- update bug_report template to be more renku-relevant (#988) (e00ded7)

6.23 0.8.0 (2019-11-21)

6.23.1 Bug Fixes

- addressed CI problems with git submodules (#783) (0d3eeb7)
- adds simple check on empty filename (#786) (8cd061b)
- ensure all Person instances have valid ids (4f80efc), closes #812
- Fixes jsonld issue when importing from dataverse (#759) (ffe36c6)
- fixes nested type scoped handling if a class only has a single class (#804) (16d03b6)
- ignore deleted paths in generated entities (86fedaf), closes #806
- integration tests (#831) (a4ad7f9)
- make Creator a subclass of Person (ac9bac3), closes #793
- Redesign scoped context in jsonld (#750) (2b1948d)

6.23.2 Features

- avoid creation of nested datasets (#796) (6084c87)
- do not create dataset implicitly (#779) (84e59d0)
- local git repo not treated as remote (8cc2834)
- renku clone command (#828) (4b3b615)

6.24 0.7.0 (2019-10-15)

6.24.1 Bug Fixes

- use UI-resolved project path as project ID (#701) (dfcc9e6)

6.25 0.6.1 (2019-10-10)

6.25.1 Bug Fixes

- add .renku/tmp to default .gitignore (#728) (6212148)
- dataset import causes renku exception due to duplicate LocalClient (#724) (89411b0)
- delete new dataset ref if file add fails (#729) (2dea711)
- fixes bug with deleted files not getting committed (#741) (5de4b6f)
- force current project for entities (#707) (538ef07)
- integration tests for #681 (#747) (b08435d)
- use commit author for project creator (#715) (1a40ebe), closes #713
- zenodo dataset import error (f1d623a)

6.25.2 Features

- adds basic QA checklist (#698) (c97e9bd)
- dataset tagging (#693) (797161f)
- include creator in project metadata (#687) ([9c7753e](https://git

6.26 0.6.0 (2019-09-18)

6.26.1 Bug Fixes

- adds _label and commit data to imported dataset files, single commit for imports (#651) (75ce369)
- always add commit to dataset if possible (#648) (7659bc8), closes #646
- cleanup needed for integration tests on py35 (#653) (fdd7215)
- fixed serialization of datetime to iso format (#629) (693d59d)

- fixes broken integration test (#649) (04eba66)
- hide image, pull, runner, show, workon and deactivate commands (#672) (a3e9998)
- integration tests fixed (#685) (f0ea8f0)
- migration of old datasets (#639) (4d4d7d2)
- migration timezones (#683) (58c2de4)
- Removes unnecessary call to git lfs with no paths (#658) (e32d48b)
- renku home directory overwrite in tests (#657) (90e1c48)
- upload metadata before actual files (#652) (95ed468)
- use latest_html for version check (#647) (c6b0309), closes #641
- user-related metadata (#655) (44183e6)
- zenodo export failing with relative paths (d40967c)

6.26.2 Features

- dataverse import (#626) (9f0f9a1)
- enable all datasets command to operate on dirty repository (#607) (74e328b)
- explicit input output specification (#598) (ce8ba67)
- export filename as schema:name (#643) (aed54bf), closes #640
- support for indirect inputs and outputs (#650) (e960a98)

6.27 0.5.2 (2019-07-26)

6.27.1 Bug Fixes

- safe_path check always operates on str (#603) (7c1c34e)

6.27.2 Features

- add SoftwareAgent to Activity (#590) (a60c20c), closes #508

6.28 0.5.1 (2019-07-12)

6.28.1 Bug Fixes

- ensure external storage is handled correctly (#592) (7938ac4)
- only check local repo for lfs filter (#575) (a64dc79)
- **cli**: allow renku run with many inputs (f60783e), closes #552
- added check for overwriting datasets (#541) (8c697fb)
- escape whitespaces in notebook name (#584) (0542fcc)

- modify json-ld for datasets (#534) (ab6a719), closes #525 #526
- refactored tests and docs to align with updated pydocteststyle (#586) (6f981c8)
- **cli**: add check of missing references (9a373da)
- **cli**: fail when removing non existing dataset (dd728db)
- **status**: fix renku status output when not in root folder (#564) (873270d), closes #551
- added dependencies for SSL support (#565) (4fa0fed)
- **datasets**: strip query string from data filenames (450898b)
- fixed serialization of creators (#550) (6a9173c)
- updated docs (#539) (ff9a67c)
- **cli**: remove dataset aliases (6206e62)
- **cwl**: detect script as input parameter (e23b75a), closes #495
- **deps**: updated dependencies (691644d)

6.28.2 Features

- add dataset metadata to the KG (#558) (fb443d7)
- **datasets**: export dataset to zenodo (#529) (fc6fd4f)
- added support for working on dirty repo (ae67be7)
- **datasets**: edit dataset metadata (#549) (db39083)
- integrate metadata from zenodo (#545) (4273d2a)
- **config**: added global config manager (#533) (938f820)
- **datasets**: import data from zenodo (#509) (52b2769)

6.29 0.5.0 (2019-03-28)

6.29.1 Bug Fixes

- **api**: make methods lock free (1f63964), closes #486
- use safe_load for parsing yaml (5383d1e), closes #464
- **datasets**: link flag on dataset add (eae30f4)

6.29.2 Features

- **api**: list datasets from a commit (04a9fe9)
- **cli**: add dataset rm command (a70c7ce)
- **cli**: add rm command (cf0f502)
- **cli**: configurable format of dataset output (d37abf3)
- **dataset**: add existing file from current repo (575686b), closes #99

- **datasets:** added `ls-files` command ([ccc4f59](#))
- **models:** reference context for relative paths ([5d1e8e7](#)), closes [#452](#)
- add JSON-LD output format for datasets ([c755d7b](#)), closes [#426](#)
- generate Makefile with `log -format Makefile` ([1e440ce](#))

6.30 v0.4.0

(released 2019-03-05)

- Adds `renku mv` command which updates dataset metadata, `.gitattributes` and symlinks.
- Pulls LFS objects from submodules correctly.
- Adds listing of datasets.
- Adds reduced dot format for `renku log`.
- Adds `doctor` command to check missing files in datasets.
- Moves dataset metadata to `.renku/datasets` and adds `migrate datasets` command and uses UUID for metadata path.
- Gets git attrs for files to prevent duplicates in `.gitattributes`.
- Fixes `renku show outputs` for directories.
- Runs Git LFS checkout in a worktrees and lazily pulls necessary LFS files before running commands.
- Asks user before overriding an existing file using `renku init` or `renku runner template`.
- Fixes `renku init --force` in an empty dir.
- Renames `CommitMixin._location` to `_project`.
- Addresses issue with commits editing multiple CWL files.
- Exports merge commits for full lineage.
- Exports path and parent directories.
- Adds an automatic check for the latest version.
- Simplifies issue submission from traceback to GitHub or Sentry. Requires `SENTRY_DSN` variable to be set and `sentry-sdk` package to be installed before sending any data.
- Removes outputs before run.
- Allows update of directories.
- Improves readability of the status message.
- Checks ignored path when added to a dataset.
- Adds API method for finding ignored paths.
- Uses branches for `init --force`.
- Fixes CVE-2017-18342.
- Fixes regex for parsing Git remote URLs.
- Handles `--isolation` option using `git worktree`.
- Renames `client.git` to `client.repo`.

- Supports `python -m renku`.
- Allows `'` and `-` in repo path.

6.31 v0.3.3

(released 2018-12-07)

- Fixes generated Homebrew formula.
- Renames `renku pull path` to `renku storage pull` with deprecation warning.

6.32 v0.3.2

(released 2018-11-29)

- Fixes display of workflows in `renku log`.

6.33 v0.3.1

(released 2018-11-29)

- Fixes issues with parsing remote Git URLs.

6.34 v0.3.0

(released 2018-11-26)

- Adds JSON-LD context to objects extracted from the Git repository (see `renku show context --list`).
- Uses PROV-O and WFPROV as provenance vocabularies and generates “stable” object identifiers (`@id`) for RDF and JSON-LD output formats.
- Refactors the log output to allow linking files and directories.
- Adds support for aliasing tools and workflows.
- Adds option to install shell completion (`renku --install-completion`).
- Fixes initialization of Git submodules.
- Uses relative submodule paths when appropriate.
- Simplifies external storage configuration.

6.35 v0.2.0

(released 2018-09-25)

- Refactored version using Git and Common Workflow Language.

6.36 v0.1.0

(released 2017-09-06)

- Initial public release as Renga.

CHAPTER 7

License

Copyright 2017–2021 – Swiss Data Science Center (SDSC)
A partnership between École Polytechnique Fédérale de Lausanne (EPFL) and
Eidgenössische Technische Hochschule Zürich (ETHZ).

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

CHAPTER 8

Authors

Python SDK and CLI for the Renku platform.

- Swiss Data Science Center <contact@datascience.ch>

Renku releases and development versions are available from [PyPI](#). You can install it using any tool that knows how to handle PyPI packages. Our recommendation is to use `pipx`.

Note: We do not officially support Windows at this moment. The way Windows handles paths and symlinks interferes with some Renku functionality. We recommend using the Windows Subsystem for Linux (WSL) to use Renku on Windows.

CHAPTER 9

Prerequisites

Renku depends on Git under the hood, so make sure that you have [Git installed on your system](#).

Renku also offers support to store large files in [Git LFS](#), which is used by default and should be installed on your system. If you do not wish to use Git LFS, you can run Renku commands with the `-S` flag, as in `renku -S <command>`. More information on Git LFS usage in renku can be found in the [Data in Renku](#) section of the docs.

First, [install pipx](#) and make sure that the `$PATH` is correctly configured.

```
$ python3 -m pip install --user pipx
$ python3 -m pipx ensurepath
```

Once pipx is installed use following command to install renku.

```
$ pipx install renku
$ which renku
~/.local/bin/renku
```

pipx installs Renku into its own virtual environment, making sure that it does not pollute any other packages or versions that you may have already installed.

Note: If you install Renku as a dependency in a virtual environment and the environment is active, your shell will default to the version installed in the virtual environment, *not* the version installed by pipx.

To install a development release:

```
$ pipx install --pip-args pre renku
```

```
$ pip install renku
```

The latest development versions are available on PyPI or from the Git repository:

```
$ pip install --pre renku
# - OR -
$ pip install -e git+https://github.com/SwissDataScienceCenter/renku-python.git
↪#egg=renku
```

Use following installation steps based on your operating system and preferences if you would like to work with the command line interface and you do not need the Python library to be importable.

Note: We don't officially support Windows yet, but Renku works well in the Windows Subsystem for Linux (WSL). As such, the following can be regarded as a best effort description on how to get started with Renku on Windows.

Renku can be run using the Windows Subsystem for Linux (WSL). To install the WSL, please follow the [official instructions](#).

We recommend you use the Ubuntu 20.04 image in the WSL when you get to that step of the installation.

Once WSL is installed, launch the WSL terminal and install the packages required by Renku with:

```
$ sudo apt-get update && sudo apt-get install git python3 python3-pip python3-venv
↪pipx
```

Since Ubuntu has an older version of git LFS installed by default which is known to have some bugs when cloning repositories, we recommend you manually install the newest version by following [these instructions](#).

Once all the requirements are installed, you can install Renku normally by running:

```
$ pipx install renku
$ pipx ensurepath
```

After this, Renku is ready to use. You can access your Windows in the various mount points in `/mnt/` and you can execute Windows executables (e.g. `*.exe`) as usual directly from the WSL (so `renku run myexecutable.exe` will work as expected).

The containerized version of the CLI can be launched using Docker command.

```
$ docker run -it -v "$PWD":"$PWD" -w="$PWD" renku/renku-python renku
```

It makes sure your current directory is mounted to the same place in the container.

Interaction with the platform can take place via the command-line interface (CLI).

Start by creating for folder where you want to keep your Renku project:

```
$ renku init my-renku-project
$ cd my-renku-project
```

Create a dataset and add data to it:

```
$ renku dataset create my-dataset
$ renku dataset add my-dataset https://raw.githubusercontent.com/
↪SwissDataScienceCenter/renku-python/master/README.rst
```

Run an analysis:

```
$ renku run wc < data/my-dataset/README.rst > wc_readme
```

Trace the data provenance:

```
$ renku log wc_readme
```

These are the basics, but there is much more that Renku allows you to do with your data analysis workflows.

For more information about using *renku*, refer to the *renku -help*.

Renku Python Library, CLI and Service

A Python library for the [Renku collaborative data science platform](#). It includes a CLI and SDK for end-users as well as a service backend. It provides functionality for the creation and management of projects and datasets, and simple utilities to capture data provenance while performing analysis tasks.

NOTE: `renku-python` is the python library and core service for Renku - it *does not* start the Renku platform itself - for that, refer to the [Renku docs on running the platform](#).

Renku releases and development versions are available from [PyPI](#). You can install it using any tool that knows how to handle PyPI packages. Our recommendation is to use `pipx`.

Note: We do not officially support Windows at this moment. The way Windows handles paths and symlinks interferes with some Renku functionality. We recommend using the Windows Subsystem for Linux (WSL) to use Renku on Windows.

10.1 Prerequisites

Renku depends on Git under the hood, so make sure that you have [Git installed on your system](#).

Renku also offers support to store large files in [Git LFS](#), which is used by default and should be installed on your system. If you do not wish to use Git LFS, you can run Renku commands with the `-S` flag, as in `renku -S <command>`. More information on Git LFS usage in renku can be found in the [Data in Renku](#) section of the docs.

First, install `pipx` and make sure that the `$PATH` is correctly configured.

```
$ python3 -m pip install --user pipx
$ python3 -m pipx ensurepath
```

Once `pipx` is installed use following command to install `renku`.

```
$ pipx install renku
$ which renku
~/.local/bin/renku
```

`pipx` installs Renku into its own virtual environment, making sure that it does not pollute any other packages or versions that you may have already installed.

Note: If you install Renku as a dependency in a virtual environment and the environment is active, your shell will default to the version installed in the virtual environment, *not* the version installed by `pipx`.

To install a development release:

```
$ pipx install --pip-args pre renku
```

```
$ pip install renku
```

The latest development versions are available on PyPI or from the Git repository:

```
$ pip install --pre renku
# - OR -
$ pip install -e git+https://github.com/SwissDataScienceCenter/renku-python.git
↪#egg=renku
```

Use following installation steps based on your operating system and preferences if you would like to work with the command line interface and you do not need the Python library to be importable.

Note: We don't officially support Windows yet, but Renku works well in the Windows Subsystem for Linux (WSL). As such, the following can be regarded as a best effort description on how to get started with Renku on Windows.

Renku can be run using the Windows Subsystem for Linux (WSL). To install the WSL, please follow the [official instructions](#).

We recommend you use the Ubuntu 20.04 image in the WSL when you get to that step of the installation.

Once WSL is installed, launch the WSL terminal and install the packages required by Renku with:

```
$ sudo apt-get update && sudo apt-get install git python3 python3-pip python3-venv
↪pipx
```

Since Ubuntu has an older version of git LFS installed by default which is known to have some bugs when cloning repositories, we recommend you manually install the newest version by following [these instructions](#).

Once all the requirements are installed, you can install Renku normally by running:

```
$ pipx install renku
$ pipx ensurepath
```

After this, Renku is ready to use. You can access your Windows in the various mount points in `/mnt/` and you can execute Windows executables (e.g. `*.exe`) as usual directly from the WSL (so `renku run myexecutable.exe` will work as expected).

The containerized version of the CLI can be launched using Docker command.

```
$ docker run -it -v "$PWD":"$PWD" -w="$PWD" renku/renku-python renku
```

It makes sure your current directory is mounted to the same place in the container.

10.1.1 Getting Started

Interaction with the platform can take place via the command-line interface (CLI).

Start by creating for folder where you want to keep your Renku project:

```
$ renku init my-renku-project
$ cd my-renku-project
```

Create a dataset and add data to it:

```
$ renku dataset create my-dataset
$ renku dataset add my-dataset https://raw.githubusercontent.com/
↳SwissDataScienceCenter/renku-python/master/README.rst
```

Run an analysis:

```
$ renku run wc < data/my-dataset/README.rst > wc_readme
```

Trace the data provenance:

```
$ renku log wc_readme
```

These are the basics, but there is much more that Renku allows you to do with your data analysis workflows.

For more information about using *renku*, refer to the *renku -help*.

r

- renku.api.models.dataset, 31
- renku.api.models.project, 31
- renku.api.models.run, 32
- renku.cli, 9
- renku.cli.clone, 11
- renku.cli.config, 12
- renku.cli.dataset, 13
- renku.cli.doctor, 28
- renku.cli.exception_handler, 29
- renku.cli.githooks, 29
- renku.cli.init, 10
- renku.cli.log, 20
- renku.cli.migrate, 28
- renku.cli.move, 24
- renku.cli.remove, 24
- renku.cli.rerun, 23
- renku.cli.run, 17
- renku.cli.save, 25
- renku.cli.show, 26
- renku.cli.status, 22
- renku.cli.storage, 28
- renku.cli.update, 22
- renku.cli.workflow, 24
- renku.core.management, 56
- renku.core.management.datasets, 57
- renku.core.management.git, 62
- renku.core.management.repository, 59
- renku.core.models, 33
- renku.core.models.cwl, 51
- renku.core.models.cwl.annotation, 53
- renku.core.models.cwl.command_line_tool, 51
- renku.core.models.cwl.parameter, 53
- renku.core.models.cwl.types, 55
- renku.core.models.cwl.workflow, 55
- renku.core.models.datasets, 34
- renku.core.models.entities, 42
- renku.core.models.git, 63
- renku.core.models.projects, 33
- renku.core.models.provenance, 37
- renku.core.models.provenance.activities, 38
- renku.core.models.provenance.agents, 43
- renku.core.models.provenance.qualified, 44
- renku.core.models.refs, 56
- renku.core.models.workflow, 45
- renku.core.models.workflow.converters, 51
- renku.core.models.workflow.converters.cwl, 51
- renku.core.models.workflow.parameters, 46
- renku.core.models.workflow.run, 45
- renku.core.plugins.run, 64

A

- `activities_for_paths()`
 (*renku.core.management.repository.RepositoryApiMixin* method), 60
- `Activity` (class in *renku.core.models.provenance.activities*), 38
renku.core.models.cwl.annotation, 53
- `activity` (*renku.core.models.provenance.qualified.Generation* attribute), 44
renku.core.models.cwl.annotation, 53
- `activity` (*renku.core.models.workflow.run.Run* attribute), 45
renku.core.models.cwl.annotation, 53
- `ACTIVITY_INDEX` (*renku.core.management.repository.RepositoryApiMixin* attribute), 59
- `activity_index_path`
 (*renku.core.management.repository.RepositoryApiMixin* attribute), 60
- `add_annotations()`
 (*renku.core.models.provenance.activities.ProcessRun* method), 39
- `add_annotations()`
 (*renku.core.models.provenance.activities.WorkflowRun* method), 41
- `add_data_to_dataset()`
 (*renku.core.management.datasets.DatasetsApiMixin* method), 57
- `add_dataset_tag()`
 (*renku.core.management.datasets.DatasetsApiMixin* method), 57
- `add_indirect_inputs()`
 (*renku.core.models.cwl.command_line_tool.CommandLineToolFactory* method), 52
- `add_indirect_outputs()`
 (*renku.core.models.cwl.command_line_tool.CommandLineToolFactory* method), 52
- `add_indirect_parameter()` (in module *renku.core.models.cwl.command_line_tool*), 53
- `add_step()` (*renku.core.models.cwl.workflow.Workflow* method), 55
- `add_subprocess()` (*renku.core.models.workflow.run.Run* method), 45
- `add_to_activity_index()`
 (*renku.core.management.repository.RepositoryApiMixin* method), 60
- `Annotation` (class) in *renku.core.models.cwl.annotation*, 53
- `AnnotationSchema` (class) in *renku.core.models.cwl.annotation*, 53
- `AnnotationSchema.Meta` (class) in *renku.core.models.cwl.annotation*, 53
- `as_jsonld()` (*renku.core.models.cwl.annotation.Annotation* method), 53
- `as_jsonld()` (*renku.core.models.datasets.Dataset* method), 35
- `as_jsonld()` (*renku.core.models.datasets.DatasetFile* method), 37
- `as_jsonld()` (*renku.core.models.projects.Project* method), 33
- `as_jsonld()` (*renku.core.models.provenance.activities.Activity* method), 38
- `as_jsonld()` (*renku.core.models.provenance.activities.ProcessRun* method), 39
- `as_jsonld()` (*renku.core.models.provenance.activities.WorkflowRun* method), 41
- `as_jsonld()` (*renku.core.models.provenance.agents.SoftwareAgent* method), 44
- `as_jsonld()` (*renku.core.models.provenance.qualified.Generation* method), 44
- `as_jsonld()` (*renku.core.models.provenance.qualified.Usage* method), 44
- `as_jsonld()` (*renku.core.models.workflow.parameters.CommandArgument* method), 46
- `as_jsonld()` (*renku.core.models.workflow.parameters.CommandInput* method), 47
- `as_jsonld()` (*renku.core.models.workflow.parameters.CommandInputTool* method), 47
- `as_jsonld()` (*renku.core.models.workflow.parameters.CommandOutput* method), 48
- `as_jsonld()` (*renku.core.models.workflow.parameters.CommandOutputTool* method), 49
- `as_jsonld()` (*renku.core.models.workflow.parameters.MappedIOStream* method), 49

<i>method</i>), 50	CommandOutput	(class	in
as_jsonld() (<i>renku.core.models.workflow.parameters.RunParameter</i>), <i>method</i>), 50	48		
as_jsonld() (<i>renku.core.models.workflow.run.Run</i> <i>method</i>), 45	CommandOutputBinding	(class	in
	<i>renku.core.models.cwl.parameter</i>), 54		
C	CommandOutputParameter	(class	in
CACHE (<i>renku.core.management.datasets.DatasetsApiMixin</i> <i>attribute</i>), 57	48	<i>renku.core.models.cwl.parameter</i>), 54	
candidate_paths (<i>renku.core.management.git.GitCore</i> <i>attribute</i>), 62	CommandOutputSchema	(class	in
check_email() (<i>renku.core.models.provenance.agents.Person</i> <i>method</i>), 43	48	<i>renku.core.models.workflow.parameters</i>),	
check_ref_format() (<i>renku.core.models.refs.LinkReference</i> class <i>method</i>), 56	CommandOutputSchema.Meta	(class	in
	<i>renku.core.models.workflow.parameters</i>),		
clear_temporary_datasets_path() (<i>renku.core.management.datasets.DatasetsApiMixin</i> <i>method</i>), 57	49	CommandOutputTemplateSchema (class	in
cmdline_tool_annotations() (in module <i>renku.core.plugins.run</i>), 64	49	<i>renku.core.models.workflow.parameters</i>),	
Collection (class in <i>renku.core.models.entities</i>), 43	49	CommandOutputTemplateSchema.Meta (class in	in
CommandArgument (class in <i>renku.core.models.workflow.parameters</i>), 46	49	<i>renku.core.models.workflow.parameters</i>),	
CommandArgumentSchema (class in <i>renku.core.models.workflow.parameters</i>), 46	49	CommandParameter (class in	in
CommandArgumentSchema.Meta (class in <i>renku.core.models.workflow.parameters</i>), 46	50	<i>renku.core.models.workflow.parameters</i>),	
CommandInput (class in <i>renku.core.models.workflow.parameters</i>), 47	50	CommandParameterSchema.Meta (class in	in
CommandInputParameter (class in <i>renku.core.models.cwl.parameter</i>), 53	50	<i>renku.core.models.workflow.parameters</i>), 50	
CommandInputSchema (class in <i>renku.core.models.workflow.parameters</i>), 47	50	commit() (<i>renku.core.management.git.GitCore</i> <i>method</i>), 62	
CommandInputSchema.Meta (class in <i>renku.core.models.workflow.parameters</i>), 47	50	commit_sha (<i>renku.core.models.datasets.DatasetFile</i> <i>attribute</i>), 37	
CommandInputTemplate (class in <i>renku.core.models.workflow.parameters</i>), 47	50	contains_any() (<i>renku.core.models.datasets.Dataset</i> <i>method</i>), 35	
CommandInputTemplateSchema (class in <i>renku.core.models.workflow.parameters</i>), 48	50	convert() (<i>renku.core.models.workflow.converters.cwl.CWLConverter</i> <i>static method</i>), 51	
CommandInputTemplateSchema.Meta (class in <i>renku.core.models.workflow.parameters</i>), 48	50	convert_default() (in module <i>renku.core.models.cwl.parameter</i>), 55	
CommandLineBinding (class in <i>renku.core.models.cwl.parameter</i>), 54	50	convert_run() (in module <i>renku.core.models.cwl.workflow</i>), 55	
CommandLineToolFactory (class in <i>renku.core.models.cwl.command_line_tool</i>), 51	50	create() (<i>renku.core.models.projects.ProjectCollection</i> <i>method</i>), 34	
	50	create() (<i>renku.core.models.refs.LinkReference</i> class <i>method</i>), 56	
	50	create_dataset() (<i>renku.core.management.datasets.DatasetsApiMixin</i> <i>method</i>), 57	
	50	creators_csv (<i>renku.core.models.datasets.Dataset</i> <i>attribute</i>), 35	
	50	creators_full_csv (<i>renku.core.models.datasets.Dataset</i> attribute), 35	
	50	cwl_prefix (<i>renku.core.management.repository.RepositoryApiMixin</i> <i>attribute</i>), 60	
	50	CWLConverter (class in	in

renku.core.models.workflow.converters.cwl),
51

D

data_dir (*renku.core.management.repository.RepositoryApiMixin* attribute), 60

data_dir (*renku.core.models.datasets.Dataset* attribute), 35

Dataset (class in *renku.core.models.datasets*), 35

dataset_commits ()
(*renku.core.management.datasets.DatasetsApiMixin* method), 57

DATASET_IMAGES (*renku.core.management.datasets.DatasetsApiMixin* attribute), 57

DatasetFile (class in *renku.core.models.datasets*), 36

DATASETS (*renku.core.management.datasets.DatasetsApiMixin* attribute), 57

datasets (*renku.core.management.datasets.DatasetsApiMixin* attribute), 57

datasets_from_commit ()
(*renku.core.management.datasets.DatasetsApiMixin* method), 57

DATASETS_PROVENANCE
(*renku.core.management.datasets.DatasetsApiMixin* attribute), 57

datasets_provenance
(*renku.core.management.datasets.DatasetsApiMixin* attribute), 58

datasets_provenance_path
(*renku.core.management.datasets.DatasetsApiMixin* attribute), 58

DatasetsApiMixin (class in *renku.core.management.datasets*), 57

default_agents () (*renku.core.models.provenance.activities.Activity* method), 38

default_agents () (*renku.core.models.provenance.activities.ProcessRun* method), 39

default_agents () (*renku.core.models.provenance.activities.WorkflowRun* method), 41

default_ended_at_time ()
(*renku.core.models.provenance.activities.Activity* method), 38

default_ended_at_time ()
(*renku.core.models.provenance.activities.ProcessRun* method), 39

default_ended_at_time ()
(*renku.core.models.provenance.activities.WorkflowRun* method), 41

default_filename ()
(*renku.core.models.datasets.DatasetFile* method), 37

default_generated ()
(*renku.core.models.provenance.activities.Activity* method), 38

default_generated ()
(*renku.core.models.provenance.activities.ProcessRun* method), 39

default_generated ()
(*renku.core.models.provenance.activities.WorkflowRun* method), 41

default_id () (*renku.core.models.datasets.Dataset* method), 35

default_id () (*renku.core.models.datasets.DatasetFile* method), 37

default_id () (*renku.core.models.entities.Collection* method), 43

default_id () (*renku.core.models.entities.Entity* method), 42

default_id () (*renku.core.models.provenance.activities.Activity* method), 38

default_id () (*renku.core.models.provenance.activities.ProcessRun* method), 39

default_id () (*renku.core.models.provenance.activities.WorkflowRun* method), 41

default_id () (*renku.core.models.provenance.agents.Person* method), 43

default_id () (*renku.core.models.provenance.qualified.Generation* method), 44

default_id () (*renku.core.models.workflow.parameters.MappedIOStream* method), 50

default_influenced ()
(*renku.core.models.provenance.activities.Activity* method), 38

default_influenced ()
(*renku.core.models.provenance.activities.ProcessRun* method), 39

default_influenced ()
(*renku.core.models.provenance.activities.WorkflowRun* method), 41

default_invalidated ()
(*renku.core.models.provenance.activities.ProcessRun* method), 40

default_invalidated ()
(*renku.core.models.provenance.activities.WorkflowRun* method), 41

default_label () (*renku.core.models.datasets.Dataset* method), 35

default_label () (*renku.core.models.datasets.DatasetFile* method), 37

default_label () (*renku.core.models.entities.Collection* method), 43

default_label () (*renku.core.models.entities.Entity* method), 42

default_label () (*renku.core.models.provenance.activities.Activity* method), 38

- default_label() (renku.core.models.provenance.activities.ProcessRun method), 40
- default_label() (renku.core.models.provenance.activities.WorkflowRun method), 41
- default_label() (renku.core.models.provenance.agents.DependencyGraph method), 43
- default_label() (renku.core.models.workflow.parameters.CommandArgument method), 46
- default_label() (renku.core.models.workflow.parameters.CommandInput method), 47
- default_label() (renku.core.models.workflow.parameters.CommandOutput method), 47
- default_label() (renku.core.models.workflow.parameters.DirectoryTemplate method), 48
- default_label() (renku.core.models.workflow.parameters.DockerContainer method), 49
- default_label() (renku.core.models.workflow.parameters.MappedOSExec method), 50
- default_label() (renku.core.models.workflow.parameters.RunParameter method), 51
- default_members() (renku.core.models.entities.Collection method), 43
- default_message() (renku.core.models.provenance.activities.Activity method), 38
- default_message() (renku.core.models.provenance.activities.ProcessRun method), 40
- default_message() (renku.core.models.provenance.activities.WorkflowRun method), 41
- default_path() (in module renku.core.management.repository), 62
- default_started_at_time() (renku.core.models.provenance.activities.Activity method), 38
- default_started_at_time() (renku.core.models.provenance.activities.ProcessRun method), 40
- default_started_at_time() (renku.core.models.provenance.activities.WorkflowRun method), 41
- default_url() (renku.core.models.datasets.DatasetFile method), 37
- default_was_informed_by() (renku.core.models.provenance.activities.Activity method), 38
- default_was_informed_by() (renku.core.models.provenance.activities.ProcessRun method), 40
- default_was_informed_by() (renku.core.models.provenance.activities.WorkflowRun method), 41
- LinkReference method), 56
- renku.core.models.cwl.command_line_tool), 53
- renku.core.management.repository.RepositoryApi attribute), 59
- renku.core.management.repository.RepositoryApi attribute), 60
- renku.core.management.repository.RepositoryApiMixin attribute), 60
- Directory (class in renku.core.models.cwl.types), 55
- renku.core.models.cwl.types), 55
- renku.core.management.git.GitCore attribute), 62
- renku.core.management.repository.RepositoryApiMixin attribute), 60
- renku.core.management.repository.RepositoryApiMixin attribute), 59
- ## E
- editable (renku.core.models.datasets.Dataset attribute), 35
- ensure_clean() (renku.core.management.git.GitCore method), 62
- ensure_unstaged() (renku.core.management.git.GitCore method), 62
- ensure_untracked() (renku.core.management.git.GitCore method), 62
- entities (renku.core.models.datasets.Dataset attribute), 35
- entities (renku.core.models.datasets.DatasetFile attribute), 37
- entities (renku.core.models.entities.Collection attribute), 43
- entities (renku.core.models.entities.Entity attribute), 42
- Entity (class in renku.core.models.entities), 42
- ## F
- File (class in renku.core.models.cwl.types), 55
- filter_repo_name() (in module renku.core.models.git), 63
- find_attr() (renku.core.management.git.GitCore method), 62
- find_explicit_inputs() (renku.core.models.cwl.command_line_tool.CommandLineTool method), 52
- find_file() (renku.core.models.datasets.Dataset method), 35
- find_files() (renku.core.models.datasets.Dataset method), 35

find_ignored_paths() (renku.core.management.git.GitCore method), 62
 find_previous_commit() (renku.core.management.repository.RepositoryApi method), 60
 fix_datetimes() (renku.core.models.projects.ProjectSchema method), 34
 from_commit() (renku.core.models.provenance.agents.Person class method), 43
 from_commit() (renku.core.models.provenance.agents.SoftwareAgent class method), 44
 from_cwl() (renku.core.models.cwl.parameter.CommandArgument class method), 54
 from_dict() (renku.core.models.provenance.agents.Person class method), 44
 from_factory() (renku.core.models.workflow.run.Run class method), 45
 from_git() (renku.core.models.provenance.agents.Person class method), 44
 from_jsonld() (renku.core.models.cwl.annotation.Annotation class method), 53
 from_jsonld() (renku.core.models.datasets.Dataset class method), 35
 from_jsonld() (renku.core.models.datasets.DatasetFile class method), 37
 from_jsonld() (renku.core.models.projects.Project class method), 33
 from_jsonld() (renku.core.models.provenance.activities.Activity class method), 38
 from_jsonld() (renku.core.models.provenance.activities.ProcessRun class method), 40
 from_jsonld() (renku.core.models.provenance.activities.WorkflowRun class method), 41
 from_jsonld() (renku.core.models.provenance.agents.Person class method), 44
 from_jsonld() (renku.core.models.provenance.agents.SoftwareAgent class method), 44
 from_jsonld() (renku.core.models.provenance.qualified.Usage class method), 45
 from_jsonld() (renku.core.models.provenance.qualified.Usage class method), 44
 from_jsonld() (renku.core.models.workflow.parameters.CommandArgument class method), 46
 from_jsonld() (renku.core.models.workflow.parameters.CommandInput class method), 47
 from_jsonld() (renku.core.models.workflow.parameters.CommandInputTemplate class method), 48
 from_jsonld() (renku.core.models.workflow.parameters.CommandOutput class method), 48
 from_jsonld() (renku.core.models.workflow.parameters.CommandOutputTemplate class method), 49
 from_jsonld() (renku.core.models.workflow.parameters.MappedIOStream class method), 50
 from_jsonld() (renku.core.models.workflow.parameters.RunParameter class method), 51
 from_jsonld() (renku.core.models.workflow.run.Run class method), 45
 from_revision() (renku.core.models.datasets.Dataset class method), 35
 from_revision() (renku.core.models.datasets.DatasetFile class method), 37
 from_revision() (renku.core.models.entities.Collection class method), 43
 from_revision() (renku.core.models.entities.Entity class method), 42
 from_run() (renku.core.models.provenance.activities.ProcessRun class method), 40
 from_run() (renku.core.models.provenance.activities.WorkflowRun class method), 42
 from_string() (renku.core.models.provenance.agents.Person class method), 44
 from_yaml() (renku.core.models.datasets.Dataset class method), 35
 from_yaml() (renku.core.models.projects.Project class method), 33
 from_yaml() (renku.core.models.provenance.activities.Activity class method), 38
 from_yaml() (renku.core.models.provenance.activities.ProcessRun class method), 40
 from_yaml() (renku.core.models.provenance.activities.WorkflowRun class method), 42
 from_yaml() (renku.core.models.provenance.agents.Person attribute), 44
 from_yaml() (renku.core.models.datasets.DatasetFile attribute), 37

G

generate_id() (renku.core.models.provenance.activities.Activity class method), 38
 generate_id() (renku.core.models.provenance.activities.ProcessRun class method), 40
 generate_id() (renku.core.models.provenance.activities.WorkflowRun class method), 42
 generate_id() (renku.core.models.workflow.parameters.CommandArgument static method), 46
 generate_id() (renku.core.models.workflow.parameters.CommandInput static method), 47
 generate_id() (renku.core.models.workflow.parameters.CommandInputTemplate static method), 48
 generate_id() (renku.core.models.workflow.parameters.CommandOutput static method), 48
 generate_id() (renku.core.models.workflow.parameters.CommandOutputTemplate static method), 49
 generate_id() (renku.core.models.workflow.parameters.RunParameter static method), 51

generate_id() (*renku.core.models.workflow.run.OrderedSubprocess* static method), 45
 generate_id() (*renku.core.models.workflow.run.Run* static method), 46
 generate_process_run() (*renku.core.models.cwl.command_line_tool.CommandLineToolFactory* method), 52
 generate_project_id() (in module *renku.core.models.projects*), 34
 Generation (class in *renku.core.models.provenance.qualified*), 44
 get_dataset_path() (*renku.core.management.datasets.DatasetsApiMixin* method), 58
 get_datasets_metadata_files() (*renku.core.management.datasets.DatasetsApiMixin* method), 58
 get_indirect_inputs_path() (in module *renku.core.models.cwl.command_line_tool*), 53
 get_indirect_outputs_path() (in module *renku.core.models.cwl.command_line_tool*), 53
 get_indirect_parameters_path() (in module *renku.core.models.cwl.command_line_tool*), 53
 get_mapped_std_streams() (in module *renku.core.management.git*), 63
 get_output_paths() (*renku.core.models.provenance.activities.Activity* method), 38
 get_output_paths() (*renku.core.models.provenance.activities.ProcessRun* method), 40
 get_output_paths() (*renku.core.models.provenance.activities.WorkflowRun* method), 42
 get_user_info() (in module *renku.core.models.git*), 63
 GitCore (class in *renku.core.management.git*), 62
 GitURL (class in *renku.core.models.git*), 63
 guess_inputs() (*renku.core.models.cwl.command_line_tool.CommandLineToolFactory* method), 52
 guess_outputs() (*renku.core.models.cwl.command_line_tool.CommandLineToolFactory* method), 52
 guess_type() (*renku.core.models.cwl.command_line_tool.CommandLineToolFactory* method), 52
 H
 has_datasets_provenance() (*renku.core.management.datasets.DatasetsApiMixin* method), 58
 has_external_files() (*renku.core.management.datasets.DatasetsApiMixin* method), 58
 has_graph_files() (*renku.core.management.datasets.DatasetsApiMixin* method), 58
 I
 image (*renku.core.models.git.GitURL* attribute), 63
 initialize_repository() (*renku.core.management.repository.RepositoryApiMixin* method), 60
 initialize_datasets_provenance() (*renku.core.management.datasets.DatasetsApiMixin* method), 58
 initialize_graph() (*renku.core.management.repository.RepositoryApiMixin* method), 60
 InputParameter (class in *renku.core.models.cwl.parameter*), 54
 is_existing_path() (*renku.core.models.cwl.command_line_tool.CommandLineToolFactory* method), 52
 is_project_set() (*renku.core.management.repository.RepositoryApiMixin* method), 60
 is_protected_path() (*renku.core.management.datasets.DatasetsApiMixin* method), 58
 is_using_temporary_datasets_path() (*renku.core.management.datasets.DatasetsApiMixin* method), 58
 iter_workflow() (*renku.core.management.repository.RepositoryApiMixin* method), 60
 iter_input_files() (*renku.core.models.cwl.command_line_tool.CommandLineToolFactory* method), 52
 iter_items() (*renku.core.models.refs.LinkReference* class method), 56
 K
 keywords_csv (*renku.core.models.datasets.Dataset* attribute), 35
 L
 latest_agent (*renku.core.management.repository.RepositoryApiMixin* attribute), 61
 LinkReference (class in *renku.core.models.refs*), 56
 load_dataset() (*renku.core.management.datasets.DatasetsApiMixin* method), 58
 load_dataset_from_path() (*renku.core.management.datasets.DatasetsApiMixin* method), 58
 load_dataset_from_provenance() (*renku.core.management.datasets.DatasetsApiMixin* method), 58

- LocalClient (*class in renku.core.management*), 56
- lock (*renku.core.management.repository.RepositoryApiMixin attribute*), 61
- LOCK_SUFFIX (*renku.core.management.repository.RepositoryApiMixin attribute*), 59
- ## M
- MappedIOStream (*class in renku.core.models.workflow.parameters*), 50
- MappedIOStreamSchema (*class in renku.core.models.workflow.parameters*), 50
- MappedIOStreamSchema.Meta (*class in renku.core.models.workflow.parameters*), 50
- METADATA (*renku.core.management.repository.RepositoryApiMixin attribute*), 60
- migration_type (*renku.core.management.repository.RepositoryApiMixin attribute*), 61
- model (*renku.core.models.cwl.annotation.AnnotationSchema.Meta attribute*), 53
- model (*renku.core.models.projects.ProjectCollection.Meta attribute*), 34
- model (*renku.core.models.projects.ProjectSchema.Meta attribute*), 34
- model (*renku.core.models.workflow.parameters.CommandArgumentSchema.Meta attribute*), 47
- model (*renku.core.models.workflow.parameters.CommandInputSchema.Meta attribute*), 47
- model (*renku.core.models.workflow.parameters.CommandInputTemplateSchema.Meta attribute*), 48
- model (*renku.core.models.workflow.parameters.CommandOutputSchema.Meta attribute*), 49
- model (*renku.core.models.workflow.parameters.CommandOutputTemplateSchema.Meta attribute*), 49
- model (*renku.core.models.workflow.parameters.CommandParameterSchema.Meta attribute*), 50
- model (*renku.core.models.workflow.parameters.MappedIOStreamSchema.Meta attribute*), 50
- model (*renku.core.models.workflow.parameters.RunParameterSchema.Meta attribute*), 51
- model (*renku.core.models.workflow.run.OrderedSubprocessSchema.Meta attribute*), 45
- model (*renku.core.models.workflow.run.RunSchema.Meta attribute*), 46
- modified_paths (*renku.core.management.git.GitCore attribute*), 62
- move_files () (*renku.core.management.datasets.DatasetsApiMixin method*), 58
- mutate () (*renku.core.models.datasets.Dataset method*), 35
- ## N
- name_validator () (*renku.core.models.datasets.Dataset method*), 36
- name_validator () (*renku.core.models.refs.LinkReference method*), 56
- nodes (*renku.core.models.provenance.activities.Activity attribute*), 39
- nodes (*renku.core.models.provenance.activities.ProcessRun attribute*), 40
- nodes (*renku.core.models.provenance.activities.WorkflowRun attribute*), 42
- ## O
- OrderedSubprocess (*class in renku.core.models.workflow.run*), 45
- OrderedSubprocessSchema (*class in renku.core.models.workflow.run*), 45
- OrderedSubprocessSchema.Meta (*class in renku.core.models.workflow.run*), 45
- original_identifier (*renku.core.models.datasets.Dataset attribute*), 36
- OutputParameter (*class in renku.core.models.cwl.parameter*), 54
- ## P
- Parameter (*class in renku.core.models.cwl.parameter*), 54
- parent (*renku.core.management.repository.RepositoryApiMixin attribute*), 61
- parent (*renku.core.models.datasets.Dataset attribute*), 36
- parent (*renku.core.models.datasets.DatasetFile attribute*), 37
- parent (*renku.core.models.entities.Collection attribute*), 43
- parent (*renku.core.models.entities.Entity attribute*), 42
- parents (*renku.core.models.provenance.activities.Activity attribute*), 39
- parents (*renku.core.models.provenance.activities.ProcessRun attribute*), 40
- parents (*renku.core.models.provenance.activities.WorkflowRun attribute*), 42
- parse () (*renku.core.models.git.GitURL class method*), 63
- path (*renku.core.models.refs.LinkReference attribute*), 56
- path_activity_cache (*renku.core.management.repository.RepositoryApiMixin attribute*), 61
- path_converter () (*in module renku.core.management.repository*), 62
- PathMixin (*class in renku.core.management.repository*), 59

paths (*renku.core.models.provenance.activities.Activity* reference (*renku.core.models.refs.LinkReference* attribute), 39 attribute), 56
 paths (*renku.core.models.provenance.activities.ProcessRun* REFS (*renku.core.models.refs.LinkReference* attribute), attribute), 40 56
 paths (*renku.core.models.provenance.activities.WorkflowRun* Remote (*renku.core.management.repository.RepositoryApiMixin* attribute), 42 attribute), 61
 Person (*class in renku.core.models.provenance.agents*), remove_dataset_tags ()
 43 (*renku.core.management.datasets.DatasetsApiMixin* method), 58
 plugin_annotations () remove_datasets_provenance_file ()
 (*renku.core.models.provenance.activities.ProcessRun* method), 40 (*renku.core.management.datasets.DatasetsApiMixin* method), 58
 plugin_annotations () remove_file () (*renku.core.management.datasets.DatasetsApiMixin* method), 42 static method), 58
 POINTERS (*renku.core.management.datasets.DatasetsApiMixin* remove_graph_files ()
 attribute), 57 (*renku.core.management.repository.RepositoryApiMixin* method), 61
 pre_run () (*in module renku.core.plugins.run*), 64
 prepare_git_repo () remove_unmodified ()
 (*renku.core.management.datasets.DatasetsApiMixin* method), 58 (*renku.core.management.git.GitCore* method), 62
 process_and_store_run () removed_paths (*renku.core.models.provenance.activities.Activity* attribute), 39
 (*renku.core.management.repository.RepositoryApiMixin* method), 61 removed_paths (*renku.core.models.provenance.activities.ProcessRun* attribute), 40
 process_commit () (*renku.core.management.repository.RepositoryApiMixin* method), 61 removed_paths (*renku.core.models.provenance.activities.WorkflowRun* attribute), 42
 process_run_annotations () (*in module renku.core.plugins.run*), 64
 ProcessRun (*class in renku.core.models.provenance.activities*), 39 rename () (*renku.core.models.refs.LinkReference* method), 56
 Project (*class in renku.core.models.projects*), 33 renku.api.models.dataset (*module*), 31
 project (*renku.core.management.repository.RepositoryApiMixin* attribute), 61 renku.api.models.project (*module*), 31
 project_id (*renku.core.models.projects.Project* attribute), 33 renku.api.models.run (*module*), 32
 ProjectCollection (*class in renku.core.models.projects*), 34 renku.cli (*module*), 9
 ProjectCollection.Meta (*class in renku.core.models.projects*), 34 renku.cli.clone (*module*), 11
 ProjectSchema (*class in renku.core.models.projects*), 34 renku.cli.config (*module*), 12
 ProjectSchema.Meta (*class in renku.core.models.projects*), 34 renku.cli.dataset (*module*), 13
 PROVENANCE_GRAPH (*renku.core.management.repository.RepositoryApiMixin* attribute), 60 renku.cli.doctor (*module*), 28
 provenance_graph_path renku.cli.exception_handler (*module*), 29
 (*renku.core.management.repository.RepositoryApiMixin* attribute), 61 renku.cli.githooks (*module*), 29
 renku.cli.init (*module*), 10
 renku.cli.log (*module*), 20
 renku.cli.migrate (*module*), 28
 renku.cli.move (*module*), 24
 renku.cli.remove (*module*), 24
 renku.cli.run (*module*), 17
 renku.cli.save (*module*), 25
 renku.cli.show (*module*), 26
 renku.cli.status (*module*), 22
 renku.cli.storage (*module*), 28
 renku.cli.update (*module*), 22
 renku.cli.workflow (*module*), 24
 renku.core.management (*module*), 56
 renku.core.management.datasets (*module*), 57

R

Range (*class in renku.core.models.git*), 63
 read_indirect_parameters () (*in module renku.core.models.cwl.command_line_tool*), 53

renku.core.management.git (module), 62
 renku.core.management.repository (module), 59
 renku.core.models (module), 33
 renku.core.models.cwl (module), 51
 renku.core.models.cwl.annotation (module), 53
 renku.core.models.cwl.command_line_tool (module), 51
 renku.core.models.cwl.parameter (module), 53
 renku.core.models.cwl.types (module), 55
 renku.core.models.cwl.workflow (module), 55
 renku.core.models.datasets (module), 34
 renku.core.models.entities (module), 42
 renku.core.models.git (module), 63
 renku.core.models.projects (module), 33
 renku.core.models.provenance (module), 37
 renku.core.models.provenance.activities (module), 38
 renku.core.models.provenance.agents (module), 43
 renku.core.models.provenance.qualified (module), 44
 renku.core.models.refs (module), 56
 renku.core.models.workflow (module), 45
 renku.core.models.workflow.converters (module), 51
 renku.core.models.workflow.converters.cwl (module), 51
 renku.core.models.workflow.parameters (module), 46
 renku.core.models.workflow.run (module), 45
 renku.core.plugins.run (module), 64
 renku_dataset_images_path (renku.core.management.datasets.DatasetsApiMixin attribute), 58
 renku_datasets_path (renku.core.management.datasets.DatasetsApiMixin attribute), 58
 renku_home (renku.core.management.repository.RepositoryApiMixin attribute), 61
 renku_metadata_path (renku.core.management.repository.RepositoryApiMixin attribute), 61
 renku_path (renku.core.management.repository.RepositoryApiMixin attribute), 61
 renku_pointers_path (renku.core.management.datasets.DatasetsApiMixin attribute), 58
 repo (renku.core.management.git.GitCore attribute), 62
 RepositoryApiMixin (class in renku.core.management.repository), 59
 resolve_in_submodules () (renku.core.management.repository.RepositoryApiMixin method), 61
 rev_parse () (renku.core.models.git.Range class method), 63
 Run (class in renku.core.models.workflow.run), 45
 RunParameter (class in renku.core.models.cwl.parameter), 55
 RunParameter (class in renku.core.models.workflow.parameters), 50
 RunParameterSchema (class in renku.core.models.workflow.parameters), 51
 RunParameterSchema.Meta (class in renku.core.models.workflow.parameters), 51
 RunSchema (class in renku.core.models.workflow.run), 46
 RunSchema.Meta (class in renku.core.models.workflow.run), 46
S
 sanitized_id (renku.core.models.workflow.parameters.CommandParameter attribute), 50
 set_client () (renku.core.models.datasets.Dataset method), 36
 set_client () (renku.core.models.datasets.DatasetFile method), 37
 set_client () (renku.core.models.entities.Collection method), 43
 set_client () (renku.core.models.entities.Entity method), 43
 set_dataset_images () (renku.core.management.datasets.DatasetsApiMixin method), 58
 set_reference () (renku.core.models.refs.LinkReference method), 56
 set_temporary_datasets_path () (renku.core.management.datasets.DatasetsApiMixin method), 59
 setup_credential_helper () (renku.core.management.git.GitCore method), 62
 short_id (renku.core.models.datasets.Dataset attribute), 36
 short_name (renku.core.models.provenance.agents.Person attribute), 44
 size_in_mb (renku.core.models.datasets.DatasetFile attribute), 37
 SoftwareAgent (class in renku.core.models.provenance.agents), 44
 split_command_and_args ()

W

`watch()` (*renku.core.models.cwl.command_line_tool.CommandLineToolFactory* method), 53

`with_commit()` (*renku.core.management.repository.RepositoryApiMixin* method), 61

`with_dataset()` (*renku.core.management.datasets.DatasetsApiMixin* method), 59

`with_dataset_provenance()` (*renku.core.management.datasets.DatasetsApiMixin* method), 59

`with_metadata()` (*renku.core.management.repository.RepositoryApiMixin* method), 62

`with_workflow_storage()` (*renku.core.management.repository.RepositoryApiMixin* method), 62

`Workflow` (class in *renku.core.models.cwl.workflow*), 55

`WORKFLOW` (*renku.core.management.repository.RepositoryApiMixin* attribute), 60

`workflow_names` (*renku.core.management.repository.RepositoryApiMixin* attribute), 62

`workflow_path` (*renku.core.management.repository.RepositoryApiMixin* attribute), 62

`WorkflowOutputParameter` (class in *renku.core.models.cwl.parameter*), 55

`WorkflowRun` (class in *renku.core.models.provenance.activities*), 40

`WorkflowStep` (class in *renku.core.models.cwl.workflow*), 55

`worktree()` (*renku.core.management.git.GitCore* method), 63