RENKU CHEATSHEET

Getting Started

```
$ pip install renku
```

Install with pip

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

Starting a Renku project.

Datasets

```
$ renku dataset create <dataset>
```

Create a new dataset.

```
$ renku dataset ls
```

List all datasets in the project.

```
$ renku dataset rm <dataset>
```

Remove a dataset.

```
$ renku dataset add <dataset> <url>
```

Add data from <url> to a dataset. <url> can be a local file path, an http(s) address or a Git git+http or git+ssh repository.

```
$ renku dataset add <dataset> --source <path>
[--destination <rel-path>] <git-url>
```

Add only data in <path> from Git. With –destination: location the data is copied to.

```
$ renku dataset update <dataset>
```

Update files in a dataset based on their source.

```
$ renku dataset tag <dataset> <tag> [-d <desc>]
```

Add a tag to the current version of the dataset, with description <desc>.

```
$ renku dataset ls-tags <dataset>
```

List all tags for a dataset.

```
$ renku dataset rm-tags <dataset> <tags...>
```

Remove tags from a dataset.

```
$ renku dataset import <uri>
```

Import a dataset. <uri> can be a Renku, Zenodo or Dataverse URL or DOI.

```
$ renku dataset export <dataset> <provider>
```

Export the dataset <dataset> to <provider>. Providers: Zenodo, Dataverse.

```
$ renku dataset ls-files
```

List all dataset files in project.

```
$ renku dataset unlink <dataset> [--include
<path|pattern>]
```

Remove files from a dataset.

Running

```
$ renku rerun <path>
```

Recreate the file(s) <path> by rerunning the commands that created them.

```
$ renku run --name <name> <command> [--input
<in_file>...] [--output <out_file>...]
```

Execute a <command> with Renku tracking inputs and outputs. Input and output files are automatically detected from the command string. Creates a workflow template named <name>. With –input and/or –output: Manually specify input or output files to track.

```
$ renku run --name <name> <command> --no-output
```

Run a <command> that produces no output.

```
$ renku status
```

The the status of generated output files in the project.

```
$ renku update [--all] [<path>...]
```

Update outdated output files created by renku run. With <path>'s: Only recreate these files. With –all: Update all outdated output files.

Workflows

```
$ renku workflow ls
```

List Plans (workflow templates).

```
$ renku workflow show <name>
```

Show details for Plan <name>.

```
$ renku workflow execute --provider <provider>
[--set <param-name>=<value>...] <name>
```

Execute a Plan using <provider> as a backend, overriding parameter <param-name>'s value.

```
$ renku workflow iterate [--map
<param-name>=[value,value,...]] <name>
```

Repeatedly execute a Plan, taking values from the list specified with –map.

ETH zürich SDSC EPFL

RENKU CHEATSHEET

```
$ renku workflow export --format <format> <plan>
```

Export a Plan in a given format (e.g. 'cwl').

```
$ renku workflow compose <composed-name> <plan>
<plan>
```

Create a new Plan composed of child Plans.

```
$ renku workflow edit <plan>
```

Create a new Plan composed of child Plans.

```
$ renku workflow delete <plan>
```

Remove a Plan.

```
$ renku workflow visualize [--interactive]
```

Show linked workflows as a graph.

```
$ renku workflow revert <activity ID>
```

Undo a Run.

```
$ renku workflow inputs
$ renku workflow outputs
```

Show input respectively output files used by workflows.

CONFIG

```
$ renku config set <key> <value>
```

Set entry <key> to <value> in renku config.

```
$ renku config remove <key>
```

Unset entry <key> renku config.

MISC

```
$ renku doctor
```

Check your system and repository for potential problems.

```
$ renku gc
```

Free up disk space used for caches and temporary files.

```
$ renku log
```

Show a history of renku actions.

```
$ renku migrate
```

Migrate old metadata to the current Renku version.

```
$ renku mv <path>... <destination>
```

Safely move files within a project.

```
$ renku rm <path>...
```

Safely delete files from a project.

```
$ renku save [-m <message>]
```

Save (commit) and push all local changes. with optional message.

```
$ renku storage pull <path>...
```

Pull <path>'s from external storage (LFS).

CONFIGURATION OPTIONS

- **registry**: Docker image registry
- **zenodo.access_token**: Token for Zenodo export
- **dataverse.access_token**: Token for Dataverse export
- **show_lfs_message**: Whether to show the LFS warning message
- **lfs_threshold**: Size threshold below which files aren't added to LFS

RESOURCES

Public instance of Renku:
`https://renkulab.io/`
Github:
`https://github.com/SwissDataScienceCenter/renku`
Documentation:
`https://renku.readthedocs.io/en/latest/`
Renku-Python Documentation:
`https://renku-python.readthedocs.io/en/latest/`
SDSC:
`https://datascience.ch/`

ETH zürich SDSC EPFL