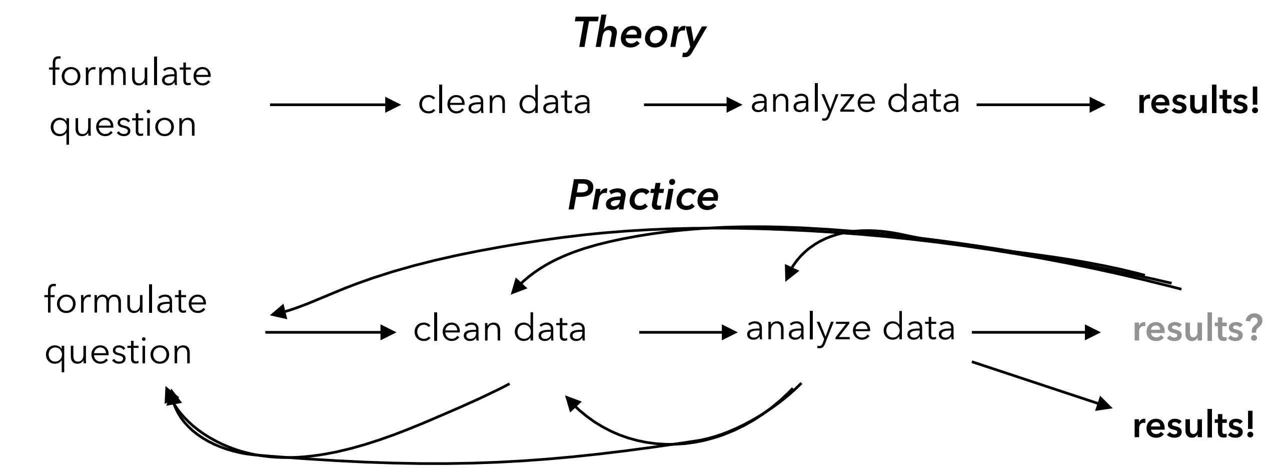
RENKU

SDSC

https://renkulab.io

Swiss Data Science Center

The Data Science Process



RENKU: reproducibility made easy

Renku is a tool for **reproducible data science.** Renku makes it possible to track your work and easily share your process and results with others.

RenkuLab adds hosted code environments and features for collaboration. Define a Julia environment and share it with others. Everyone works with the same versions of packages and tools, and it all happens at the click of a button: nothing to manage or install.

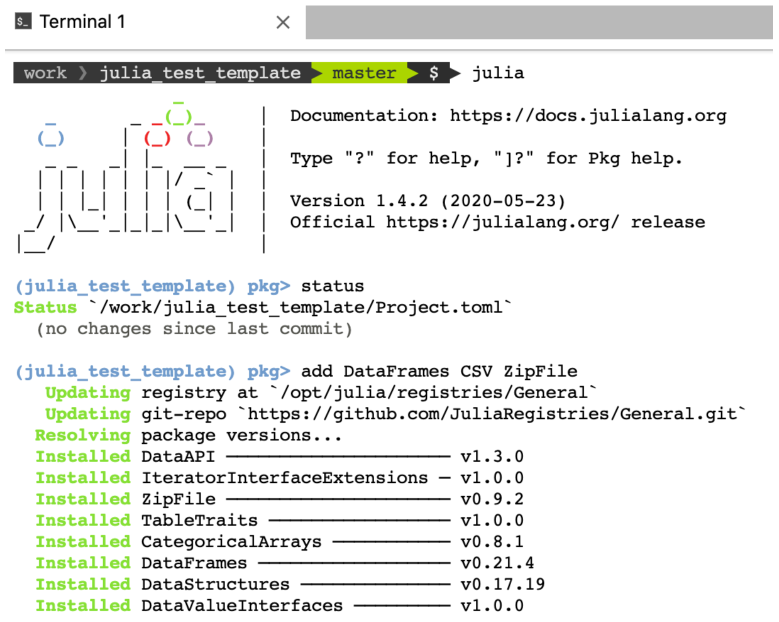
Taming the data-science process

In theory, the data science process is straightforward and linear.

In practice, it is messy. As you work, you backtrack, revise your ideas, update your approach.

With Renku, your work is tracked and reproducible, giving you the confidence to try things out and explore avenues.

You can always return to the last working state of your code or data if necessary. And months later, you will still know why you did what you did. Your colleagues will thank you; and your future self will thank you as well!



```
work 〉 julia_test_template  master  $  julia
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.4.2 (2020-05-23)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

(julia_test_template) pkg> status
Status `/work/julia_test_template/Project.toml`
  (no changes since last commit)

(julia_test_template) pkg> add DataFrames CSV ZipFile
   Updating registry at `/opt/julia/registries/General`
   Updating git-repo `https://github.com/JuliaRegistries/General.git`
  Resolving package versions...
  Installed DataAPI ──────────────────────── v1.3.0
  Installed IteratorInterfaceExtensions ─ v1.0.0
  Installed ZipFile ──────────────────────── v0.9.2
  Installed TableTraits ──────────────────── v1.0.0
  Installed CategoricalArrays ───────────── v0.8.1
  Installed DataFrames ───────────────────── v0.21.4
  Installed DataStructures ───────────────── v0.17.19
  Installed DataValueInterfaces ─────────── v1.0.0
```

Renku and Julia

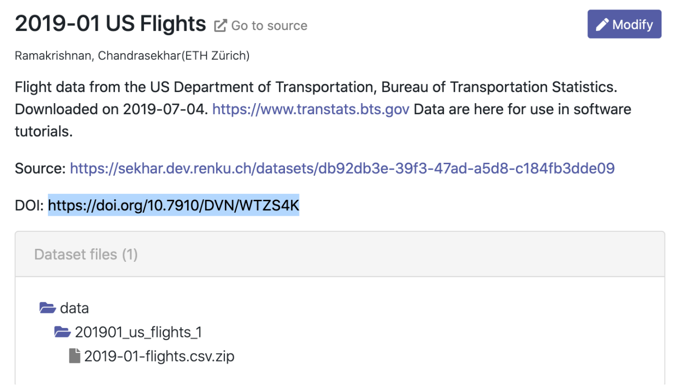
With RenkuLab, you can work in your favorite language: **Julia**. Your colleagues can view your results and step through your analysis and execute your code, even if they are not Julia developers themselves.

Example Renku Scenario

1. Import data

```
renku dataset import https://doi.org/10.7910/DVN/WTZS4K
```

Structure data in datasets. Datasets group related files and have metadata, for example, what URL a file was downloaded from. With datasets, you can change the physical layout of your data with minimal impact on the rest of your project.



2019-01 US Flights Go to source

Modify

Ramakrishnan, Chandrasekhar(ETH Zürich)

Flight data from the US Department of Transportation, Bureau of Transportation Statistics. Downloaded on 2019-07-04. https://www.transtats.bts.gov Data are here for use in software tutorials.

Source: https://sekhar.dev.renku.ch/datasets/db92db3e-39f3-47ad-a5d8-c184fb3dde09

DOI: https://doi.org/10.7910/DVN/WTZS4K

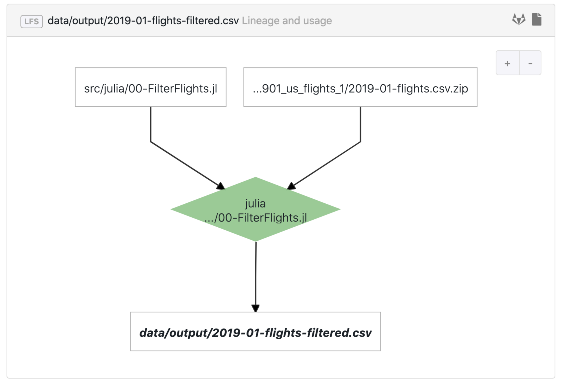
Dataset files (1)

```
data
  201901_us_flights_1
    2019-01-flights.csv.zip
```

2. Run reproducibly

```
renku run julia 00-FilterFlights.jl -i 2019-01-flights.csv.zip -o
2019-01-flights-filtered.csv
```

To get Renku to keep track of a script execution, prefix your command with `renku run`. Executed this way, Renku will remember the command with its inputs and outputs. Renku uses information to track provenance within a project, for example, to visualize how code and data are used, and to create workflows for your project artifacts.

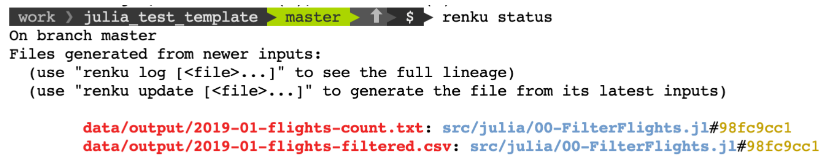


LFS data/output/2019-01-flights-filtered.csv Lineage and usage
src/julia/00-FilterFlights.jl
...901_us_flights_1/2019-01-flights.csv.zip
julia .../00-FilterFlights.jl
data/output/2019-01-flights-filtered.csv

3. Develop with confidence

You can make changes to your project with the confidence that nothing will be lost. If the *00-FilterFlights.jl* script is changed or the *2019-01-flights.csv.zip* file is modified, you don't need to remember what needs to be updated. You can ask using:

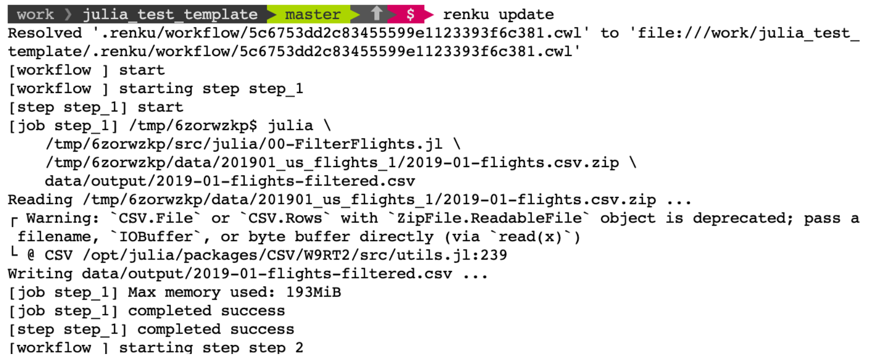
```
renku status
```



```
work 〉 julia_test_template  master  ⬆  $  renku status
On branch master
Files generated from newer inputs:
  (use "renku log [<file>...]" to see the full lineage)
  (use "renku update [<file>...]" to generate the file from its latest inputs)

        data/output/2019-01-flights-count.txt: src/julia/00-FilterFlights.jl#98fc9cc1
        data/output/2019-01-flights-filtered.csv: src/julia/00-FilterFlights.jl#98fc9cc1
```

And you can update everything downstream of a changed file with:

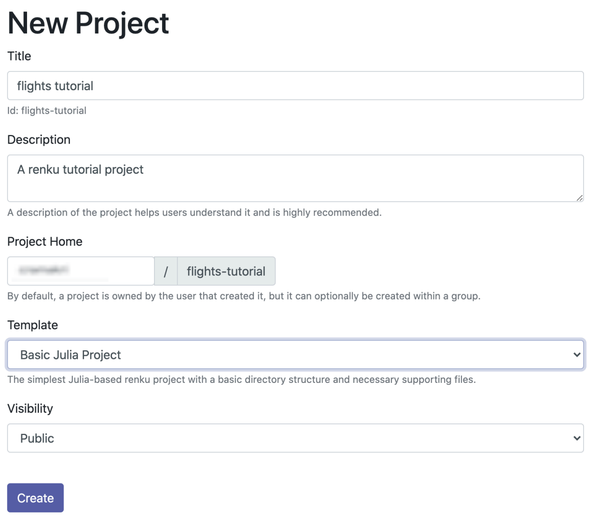
```
renku update
```



```
work 〉 julia_test_template  master  ⬆  $  renku update
Resolved '.renku/workflow/5c6753dd2c83455599e1123393f6c381.cwl' to 'file:///work/julia_test_
template/.renku/workflow/5c6753dd2c83455599e1123393f6c381.cwl'
[workflow ] start
[workflow ] starting step step_1
[step step_1] start
[job step_1] /tmp/6zorwzkp$ julia \
    /tmp/6zorwzkp/src/julia/00-FilterFlights.jl \
    /tmp/6zorwzkp/data/201901_us_flights_1/2019-01-flights.csv.zip \
    data/output/2019-01-flights-filtered.csv
Reading /tmp/6zorwzkp/data/201901_us_flights_1/2019-01-flights.csv.zip ...
┌ Warning: `CSV.File` or `CSV.Rows` with `ZipFile.ReadableFile` object is deprecated; pass a
 filename, `IOBuffer`, or byte buffer directly (via `read(x)`)
└ @ CSV /opt/julia/packages/CSV/W9RT2/src/utils.jl:239
Writing data/output/2019-01-flights-filtered.csv ...
[job step_1] Max memory used: 193MiB
[job step_1] completed success
[step step_1] completed success
[workflow ] starting step step_2
```

Getting Started on RenkuLab

Using Julia on RenkuLab is easy. Just create a new project, and select the Julia project template. This will give you create a project and install Julia and any necessary dependencies.



New Project

Title
flights tutorial
Id: flights-tutorial

Description
A renku tutorial project
A description of the project helps users understand it and is highly recommended.

Project Home
[redacted] / flights-tutorial
By default, a project is owned by the user that created it, but it can optionally be created within a group.

Template
Basic Julia Project
The simplest Julia-based renku project with a basic directory structure and necessary supporting files.

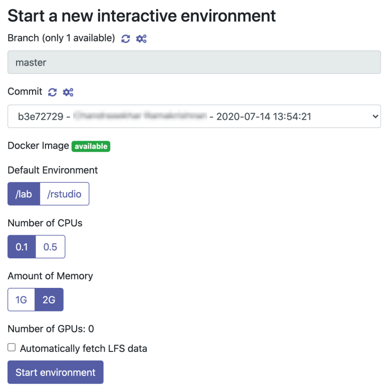
Visibility
Public

Create

Working in Julia

Once you have a project, you can launch an *Interactive Environment*. This will give you access to a JupyterLab UI to work on your project.

In JupyerLab, you work as you normally would, for example, using **Pkg** to install project dependencies and **git** to version your code.



Start a new interactive environment

Branch (only 1 available)
master

Commit
b3e72729 - [redacted] - 2020-07-14 13:54:21

Docker Image available

Default Environment
/lab /rstudio

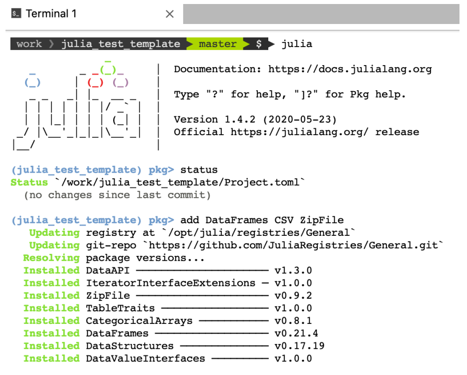
Number of CPUs
0.1 0.5

Amount of Memory
1G 2G

Number of GPUs: 0

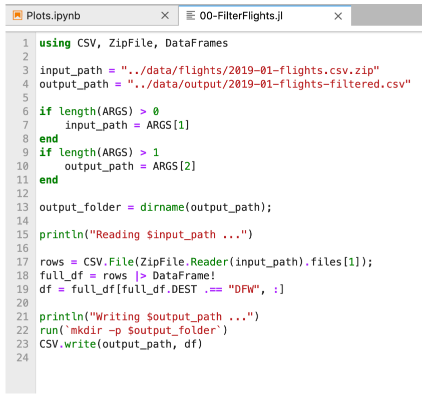
Automatically fetch LFS data

Start environment



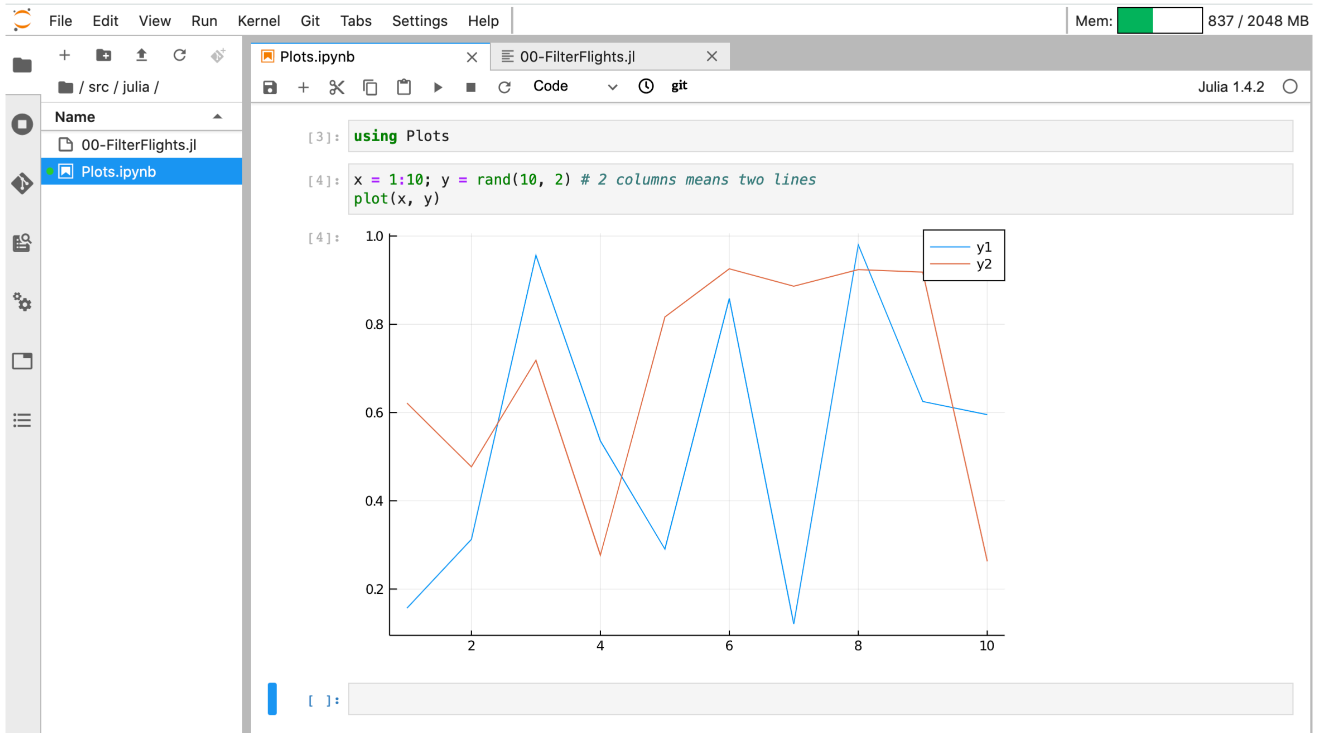
Terminal 1

```
work 〉 julia_test_template  master  $  julia
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.4.2 (2020-05-23)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

(julia_test_template) pkg> status
Status `/work/julia_test_template/Project.toml`
  (no changes since last commit)

(julia_test_template) pkg> add DataFrames CSV ZipFile
   Updating registry at `/opt/julia/registries/General`
   Updating git-repo `https://github.com/JuliaRegistries/General.git`
  Resolving package versions...
  Installed DataAPI ───────────────────── v1.3.0
  Installed IteratorInterfaceExtensions ─ v1.0.0
  Installed ZipFile ───────────────────── v0.9.2
  Installed TableTraits ───────────────── v1.0.0
  Installed CategoricalArrays ─────────── v0.8.1
  Installed DataFrames ────────────────── v0.21.4
  Installed DataStructures ────────────── v0.17.19
  Installed DataValueInterfaces ───────── v1.0.0
```



Plots.ipynb   00-FilterFlights.jl

```julia
using CSV, ZipFile, DataFrames

input_path = "../data/flights/2019-01-flights.csv.zip"
output_path = "../data/output/2019-01-flights-filtered.csv"

if length(ARGS) > 0
    input_path = ARGS[1]
end
if length(ARGS) > 1
    output_path = ARGS[2]
end

output_folder = dirname(output_path);

println("Reading $input_path ...")

rows = CSV.File(ZipFile.Reader(input_path).files[1]);
full_df = rows |> DataFrame!
df = full_df[full_df.DEST .== "DFW", :]

println("Writing $output_path ...")
run(`mkdir -p $output_folder`)
CSV.write(output_path, df)

```

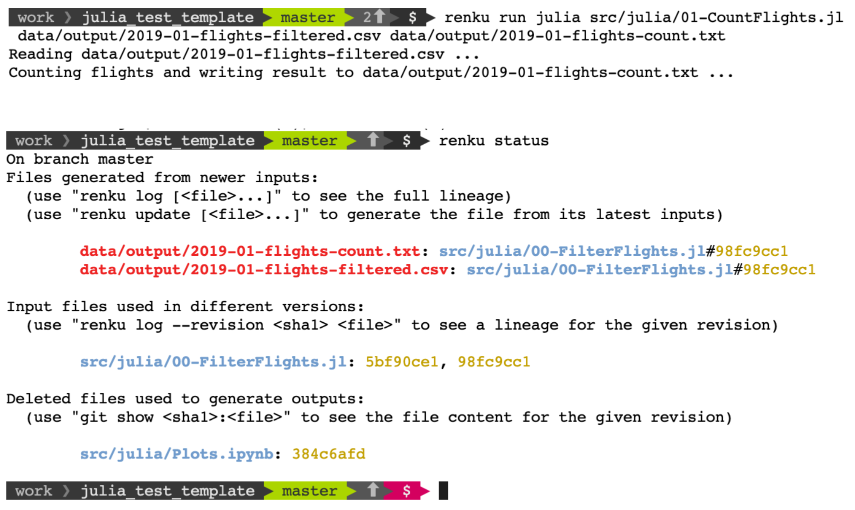


Plots.ipynb   00-FilterFlights.jl

```
[3]: using Plots

[4]: x = 1:10; y = rand(10, 2) # 2 columns means two lines
     plot(x, y)

[4]:

[ ]:
```

Running Renku Commands

Renku is a command-line tool and is already available in RenkuLab environments. Run **renku** commands from the terminal or console.



```
 work 〉 julia_test_template  master  2⬆  $  renku run julia src/julia/01-CountFlights.jl
 data/output/2019-01-flights-filtered.csv data/output/2019-01-flights-count.txt
Reading data/output/2019-01-flights-filtered.csv ...
Counting flights and writing result to data/output/2019-01-flights-count.txt ...


 work 〉 julia_test_template  master  ⬆  $  renku status
On branch master
Files generated from newer inputs:
  (use "renku log [<file>...]" to see the full lineage)
  (use "renku update [<file>...]" to generate the file from its latest inputs)

        data/output/2019-01-flights-count.txt: src/julia/00-FilterFlights.jl#98fc9cc1
        data/output/2019-01-flights-filtered.csv: src/julia/00-FilterFlights.jl#98fc9cc1

Input files used in different versions:
  (use "renku log --revision <sha1> <file>" to see a lineage for the given revision)

        src/julia/00-FilterFlights.jl: 5bf90ce1, 98fc9cc1

Deleted files used to generate outputs:
  (use "git show <sha1>:<file>" to see the file content for the given revision)

        src/julia/Plots.ipynb: 384c6afd

 work 〉 julia_test_template  master  ⬆  $
```

Building Blocks

Publishing code and data alone does not make a project reproducible or even *replicable*. For this, it is necessary to provide additional information.

What environment does the code run in?

How are code and data combined to generate results?

How did code and data evolve to the present state?

Fortunately, there are existing tools for each of these. Renku bundles **git** and **Docker** together with **workflow** and **semantic web** technologies to construct a **knowledge graph** that binds research artifacts with their provenance and metadata, containing the information necessary to automatically replicate a result and conceptually understand how it was created.

Renku projects use **Docker** to capture and recreate the runtime environment necessary to run them.

RenkuLab supports **Jupyter** and RStudio as working environments.

Renku uses **git** for version control of code/text and **git-lfs** for data. This makes it possible to see how code and data evolved in a project.

COMMON WORKFLOW LANGUAGE

Workflows describe the steps that produce an output. Renku uses **CWL** to run workflows on laptops, servers, or HPC infrastructure.

PROV-O is an ontology for describing how research artifacts are created. It forms the backbone of the Renku *Knowledge Graph*.

For collaboration features, RenkuLab integrates with **GitLab**, the popular open-source git repository manager.

Open Source

Renku is open-source software.
https://github.com/SwissDataScienceCenter/renku

Julia Showcase

Explore our Julia showcase project to try out Renku's features.
https://renkulab.io/projects/renku-tutorial/flights-tutorial-julia

Connect

Ask questions on our forum https://renku.discourse.group or chat https://gitter.im/SwissDataScienceCenter/renku

Authors

Mohammad Alisafaee, Dr. Andreas Bleuler, Lorenzo Cavazzi
Jakub Chrobasik, Dr. Pamela Delgado, Virginia Friedrich
Ralf Grubenmann, Emma Jablonski, Samuel Picek
Dr. Christine Choirat christine.choirat@epfl.ch
Chandrasekhar Ramakrishnan cramakri@ethz.ch
Dr. Rok Roškar roskarr@ethz.ch

Swiss Data Science Center

ETH zürich · EPFL · Empa Materials Science and Technology · eawag aquatic research · WSL Eidg. Forschungsanstalt für Wald Schnee und Landschaft WSL · PAUL SCHERRER INSTITUT PSI · SDSC