FAIR DATA WITH DATALAD

Findable

Accessible

Interoperable

Reusable

https://www.go-fair.org/fair-principles

FAIR PRINCIPLES

F1 (Meta)data are assigned a globally unique and persistent identifier
F2 Data are described with rich metadata
F3 Metadata clearly and explicitly include the identifier of the data they describe
F4 (Meta)data are registered or indexed in a searchable resource

---

A1 (Meta)data are retrievable by their identifier using a standardised ... protocol
A1.1 The protocol is open, free, and universally implementable
A1.2 The protocol allows for an authentication and authorisation procedure
A2 Metadata are accessible, even when the data are no longer available

---

I1 (Meta)data use a formal, accessible ... language for knowledge representation.
I2 (Meta)data use vocabularies that follow FAIR principles
I3 (Meta)data include qualified references to other (meta)data

---

EVOLUTION OF DATA AND TOOLS MUST BE ANTICIPATED

The utility of a resource declines in the absence of continued investment.
FAIR today is not FAIR forever.

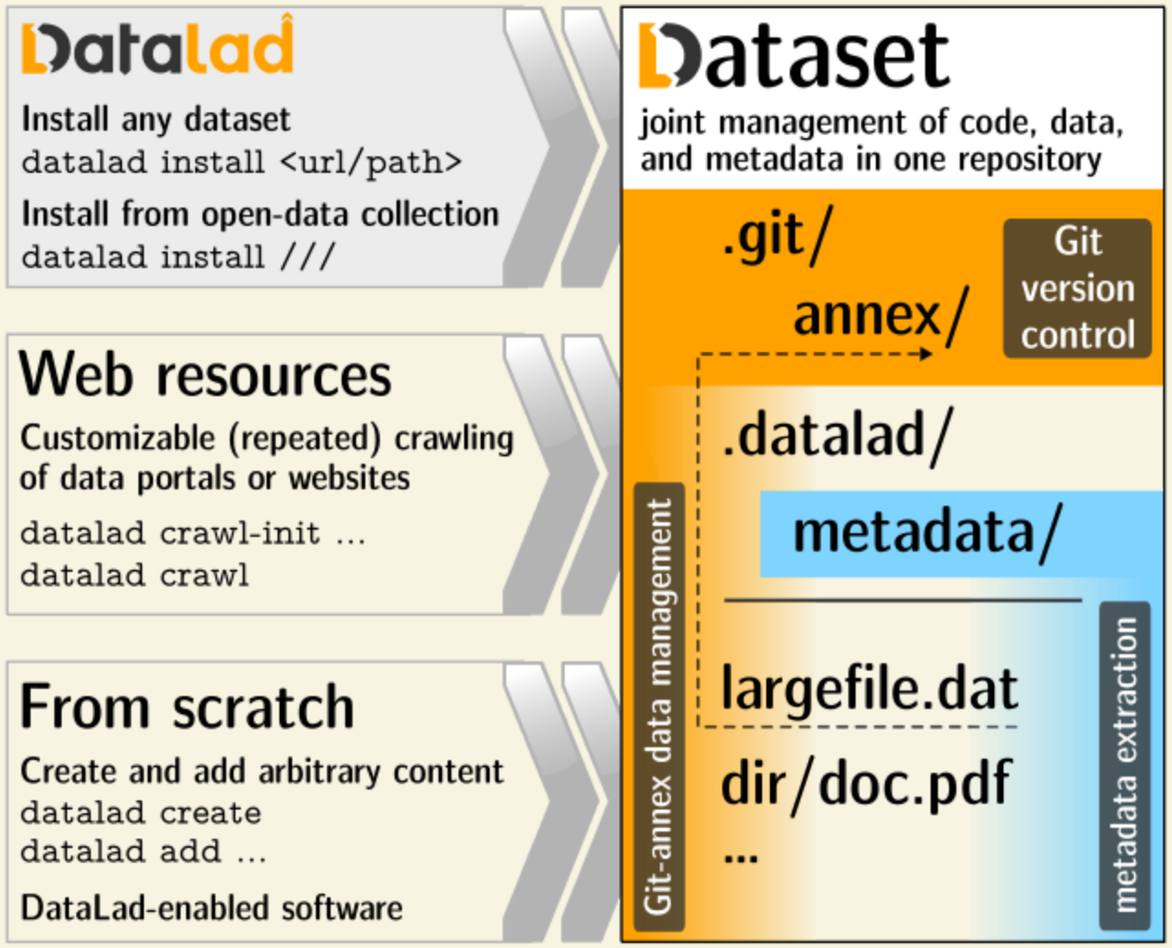
- what worked yesterday will eventually need updating to remain useful (especially analysis code)
- data can be "broken" too!
- sticking to "old" standards will ultimately make things special, and too expensive to work with

DATALAD PRINCIPLES

- There are only two things in the world: **datasets and files**.
- A **dataset is a Git repository**.
- A dataset can have an **optional *annex*** for (large) file content tracking (transport to and from the annex managed with Git-annex, https://git-annex.branchable.com).

- Minimization of custom procedures and data structures: **Users must not loose data or data access**, if DataLad would vanish.
- **Complete decentralization**, no required central server or service.
- Maximize use of existing 3rd-party infrastructure.

DATASET: TRACKING CONTENT AND/OR ITS IDENTIFIERS

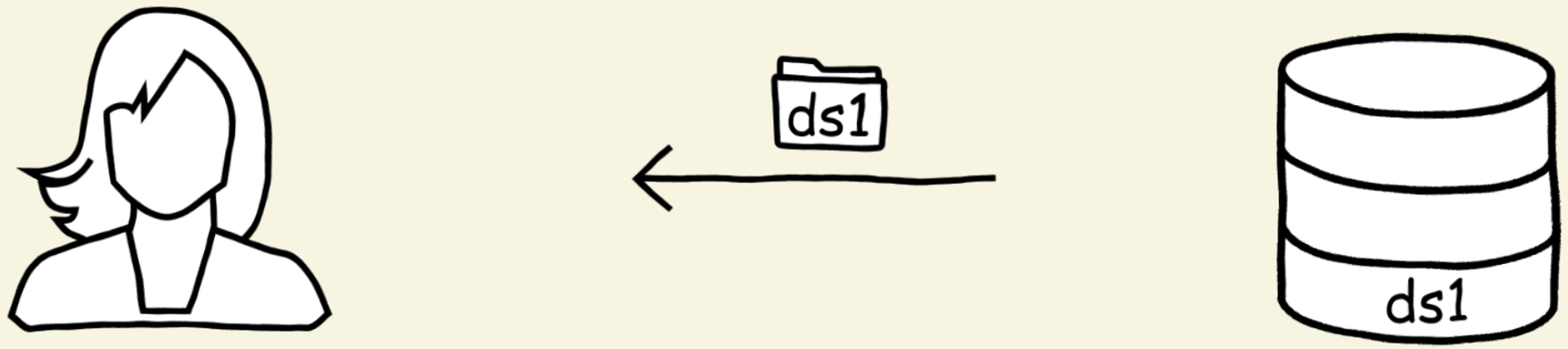


Dataset ID, Dataset annex (local) storage ID
two distinct UUIDs

Dataset version (Git repository state)
SHASUM

File key
configurable checksum (file content based)

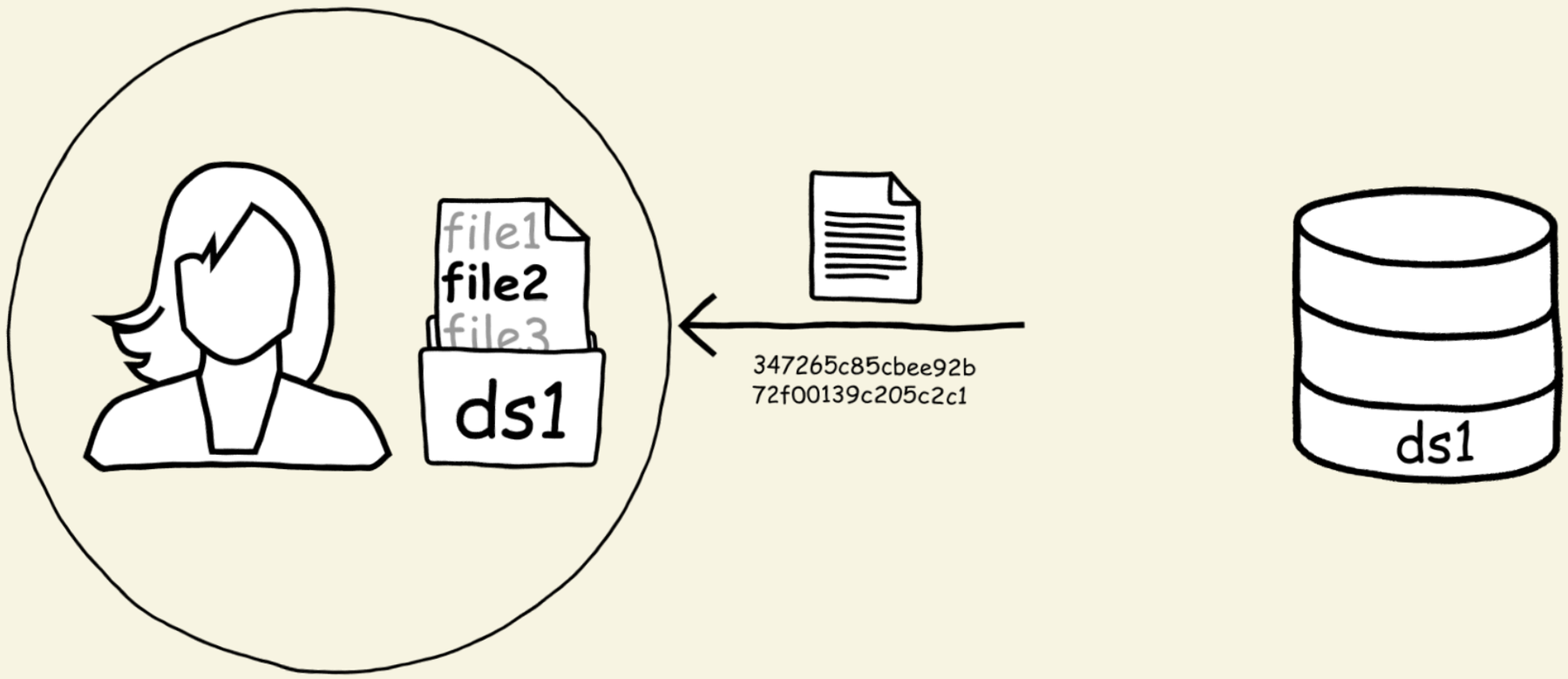
INSTALL AN EXISTING DATASET



request via standard URL,
but dataset ID resolver is possible

```
$ datalad install http://example.com/ds1
```

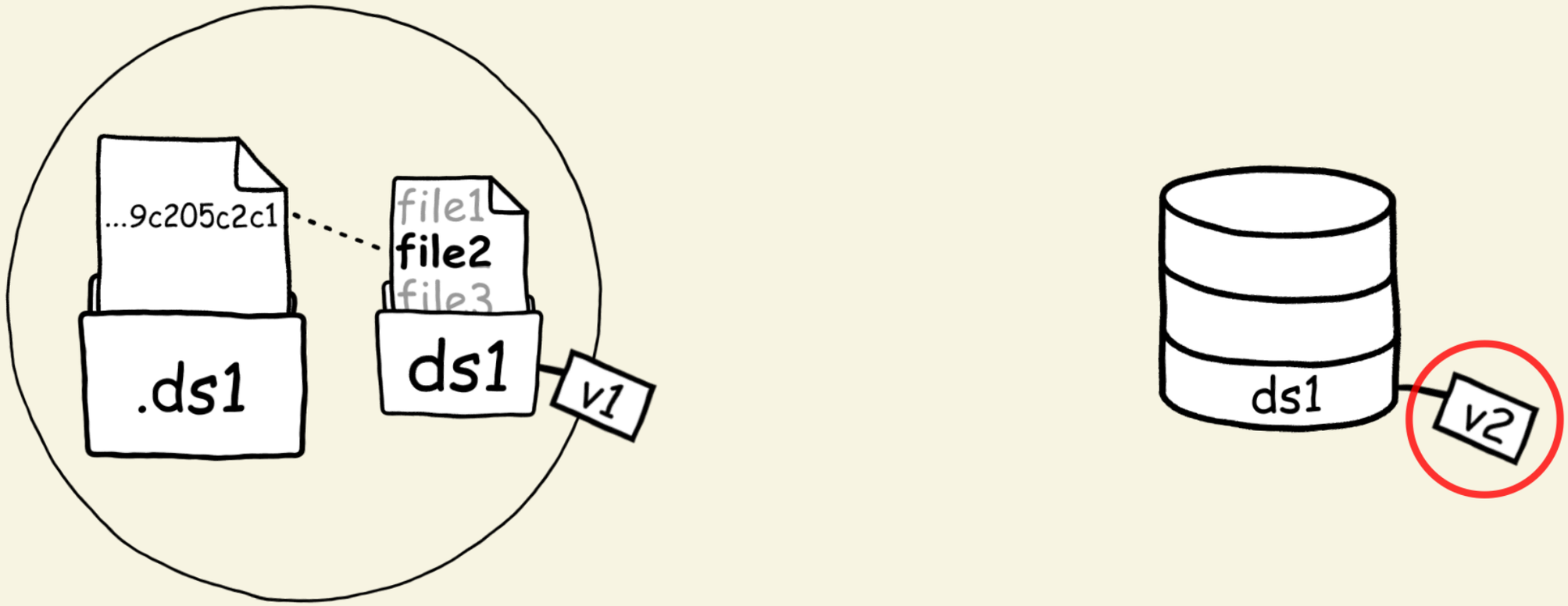
OBTAIN DATASET CONTENT



request via user-friendly local file path, not internal ID,
regardless of remote actual storage solution properties

```
ds1/ $ datalad get file2
```

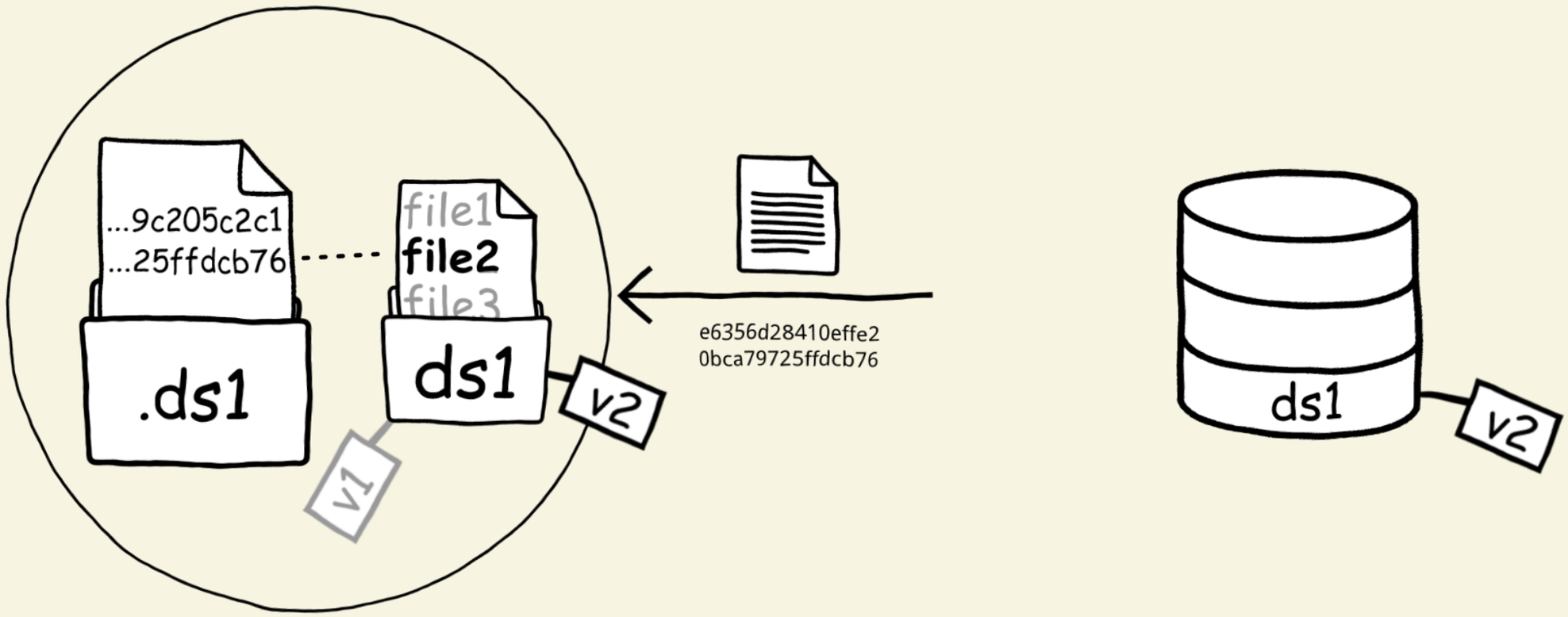
TRACKING "REMOTE" DATA EVOLUTION



ability to track any number of dataset "siblings",
in Git or non-Git data stores

```
ds1/ $ datalad update
```

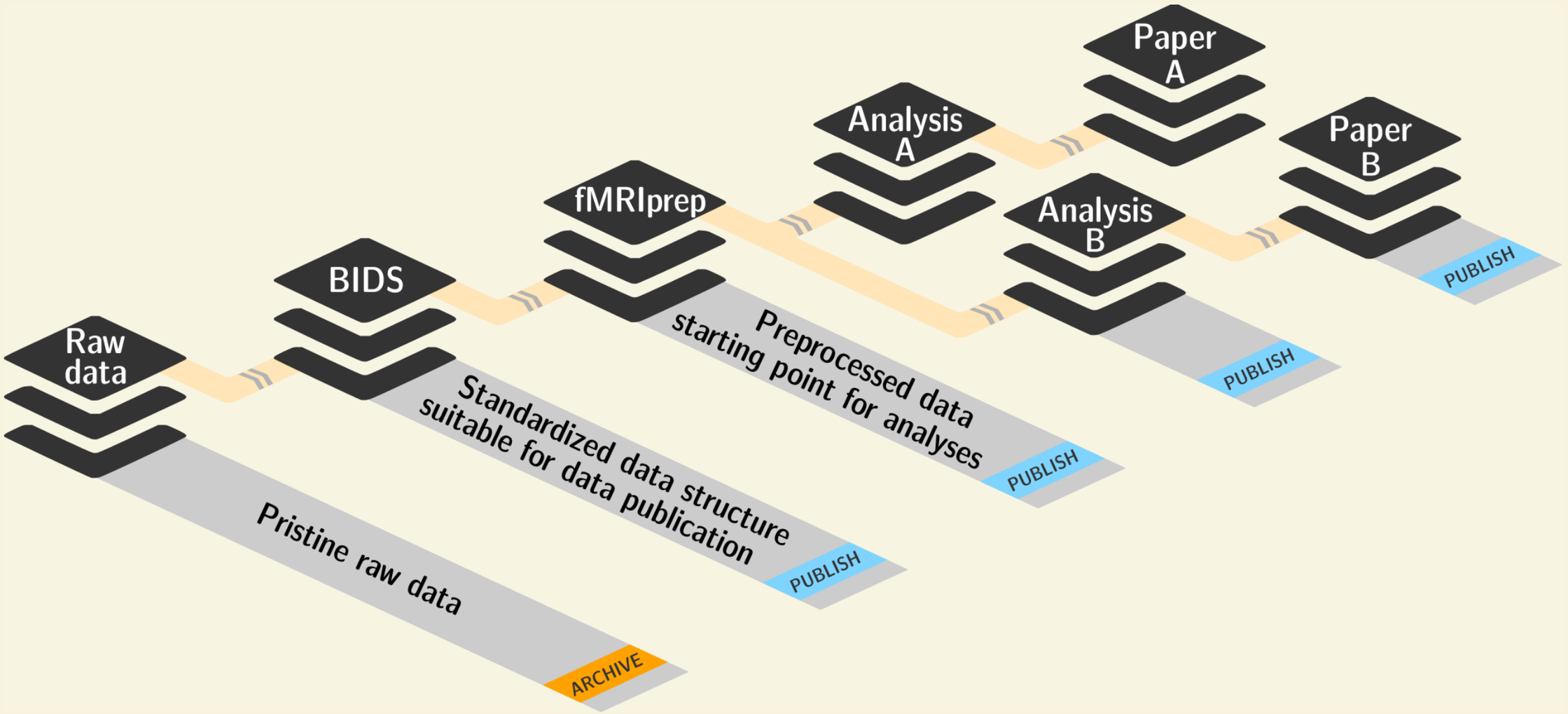
KEEP UP-TO-DATE



apply changes from default or selected sibling
while maintaining local data availability status

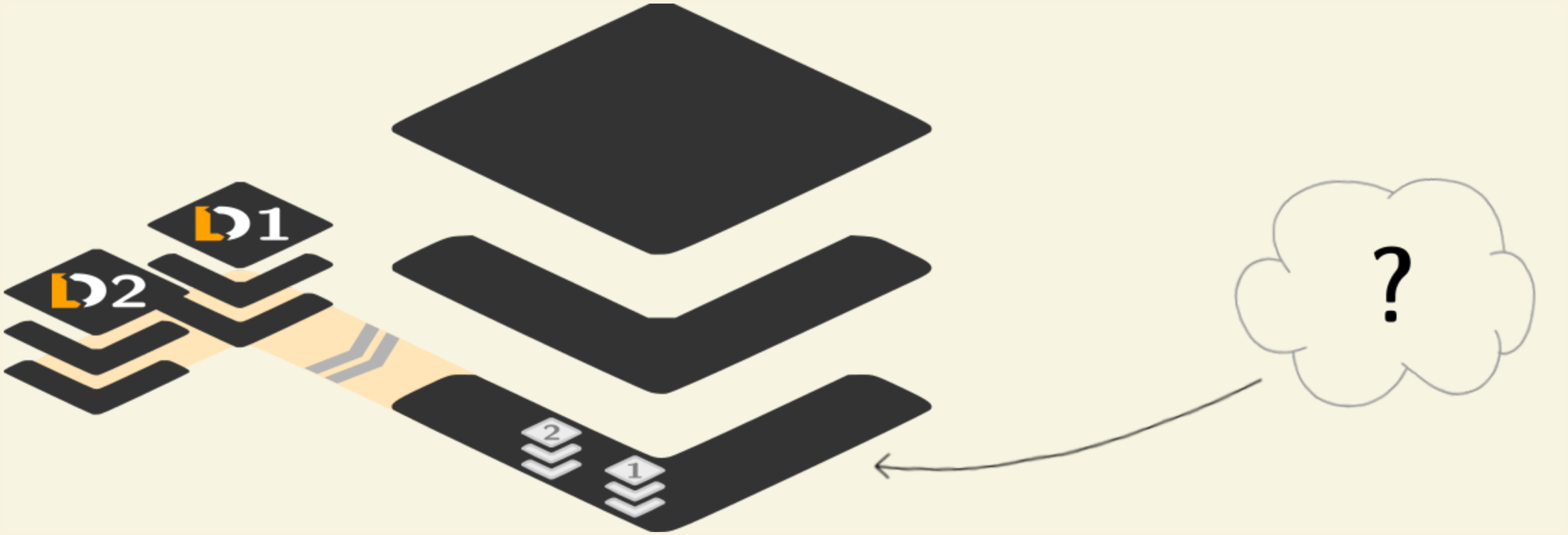
```
ds1/ $ datalad update --merge --reobtain-data
```

P1: ONE THING, ONE DATASET



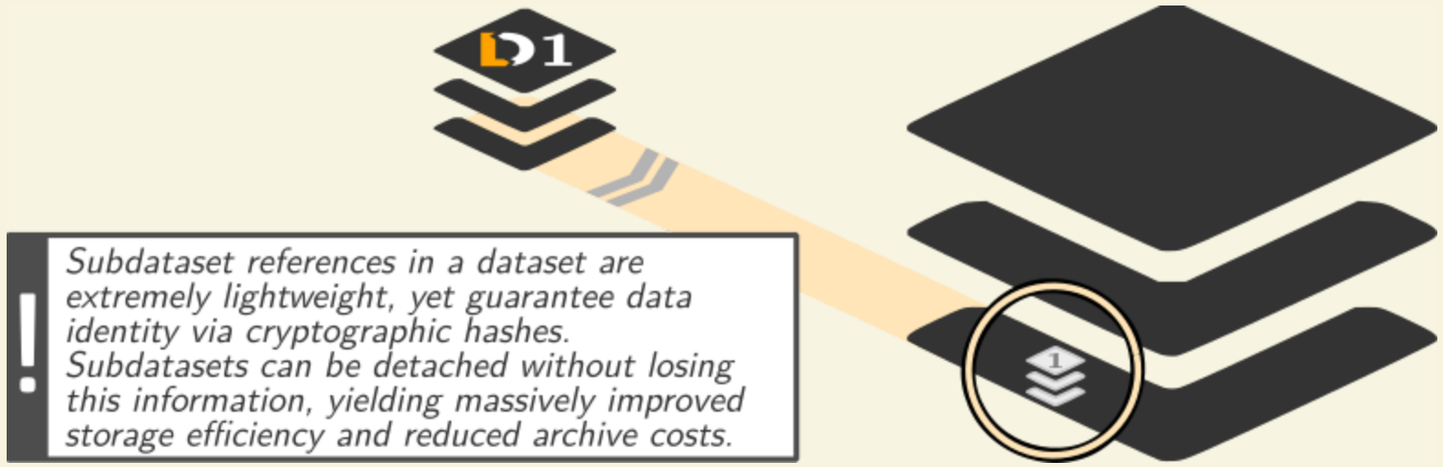
whenever a particular collection of files could anyhow be useful in more than one context, put them in their own dataset

P2: RECORD WHERE THINGS ARE



link individual datasets to declare data-dependencies, record access URLs for individual files obtained from (unstructured) sources "in the cloud"

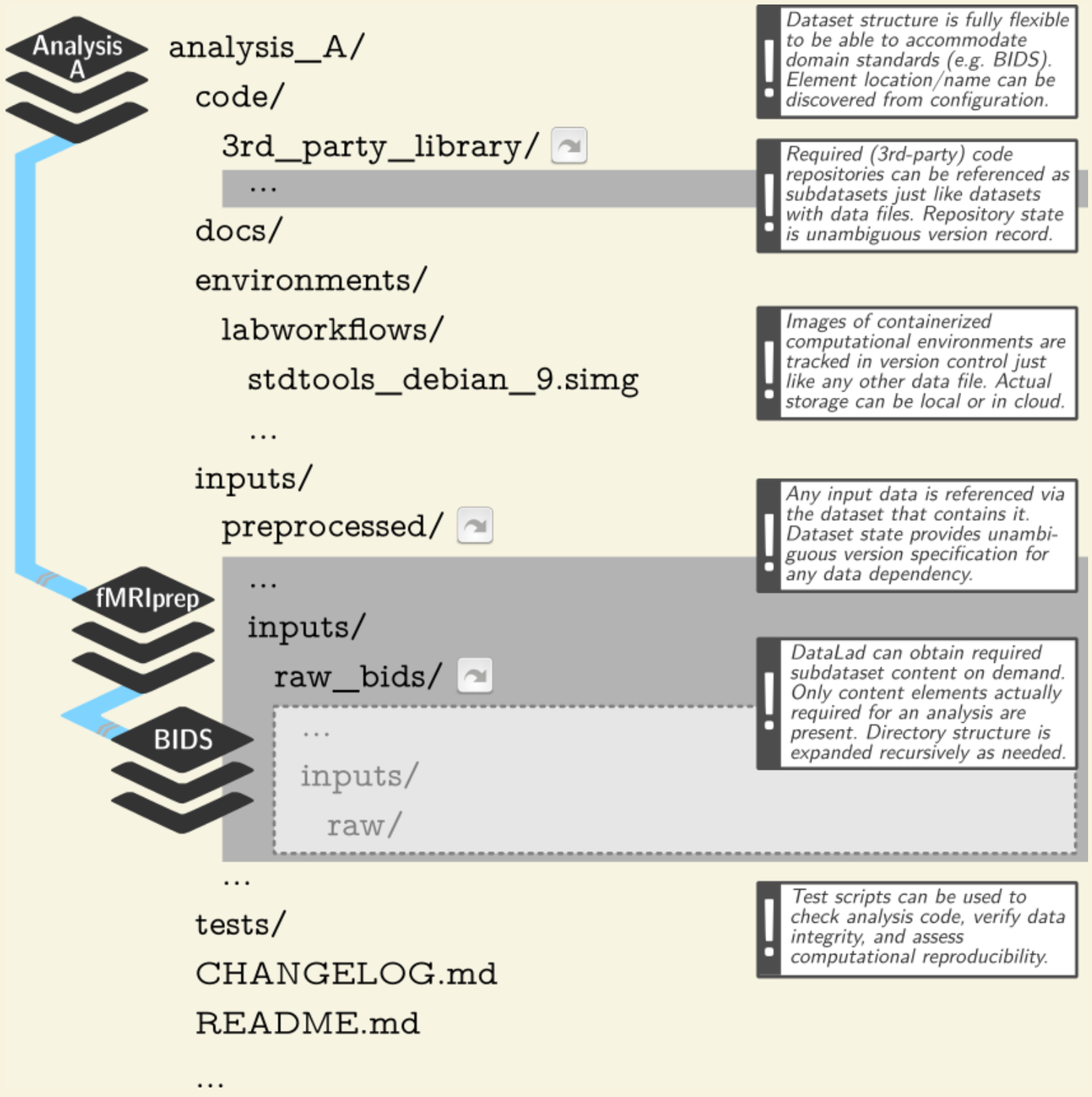
LINK RE-USABLE DATA RESOURCE UNITS




```
$ datalad install --dataset . --source http://example.com/ds inputs/rawdata
```

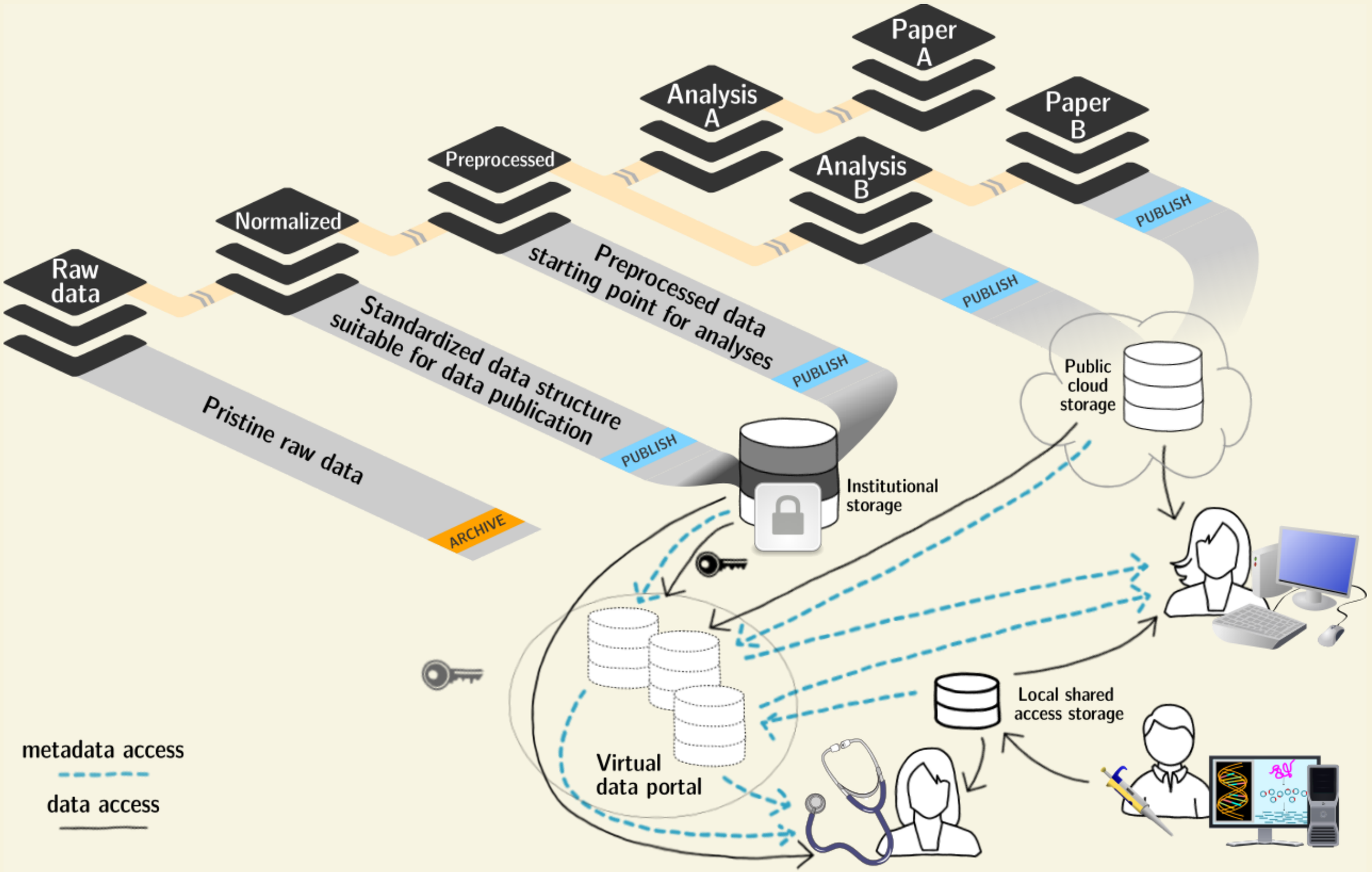
```
$ git diff HEAD~1
diff --git a/.gitmodules b/.gitmodules
new file mode 100644
index 0000000..c3370ba
--- /dev/null
+++ b/.gitmodules
@@ -0,0 +1,3 @@
+[submodule "inputs/rawdata"]
+       path = inputs/rawdata
+       url = http://example.com/importantds
diff --git a/inputs/rawdata b/inputs/rawdata
new file mode 160000
index 0000000..fabf852
--- /dev/null
+++ b/inputs/rawdata
@@ -0,0 +1 @@
+Subproject commit fabf8521130a13986bd6493cb33a70e580ce8572
```

MODULAR DATA STEWARDSHIP AND CURATION

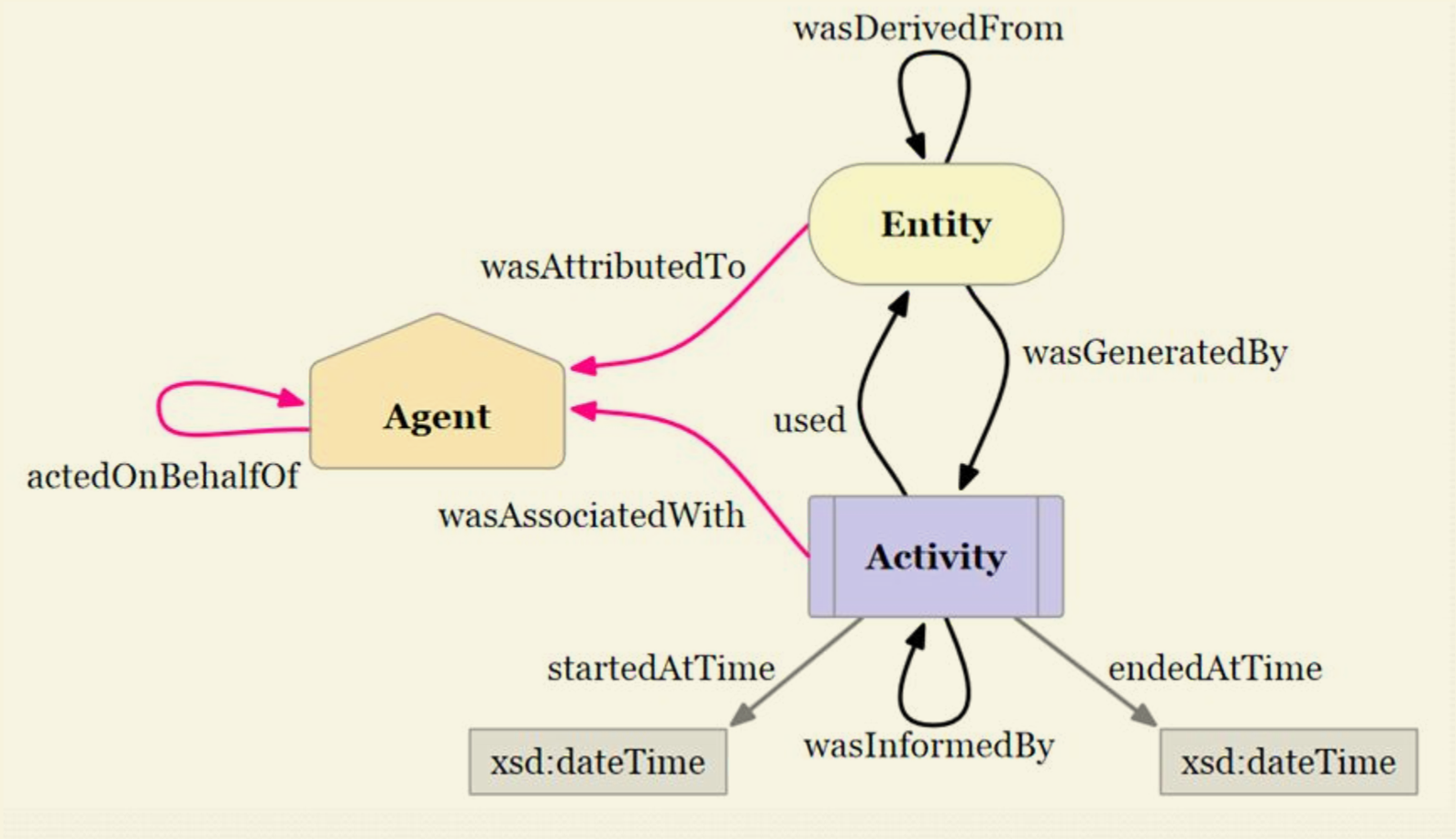


"actionable" links to subdatasets/files, seamless handling of dataset trees

MOVE/SHARE/PUBLISH DATA



P3: RECORD WHAT YOU DID TO IT, AND WITH WHAT

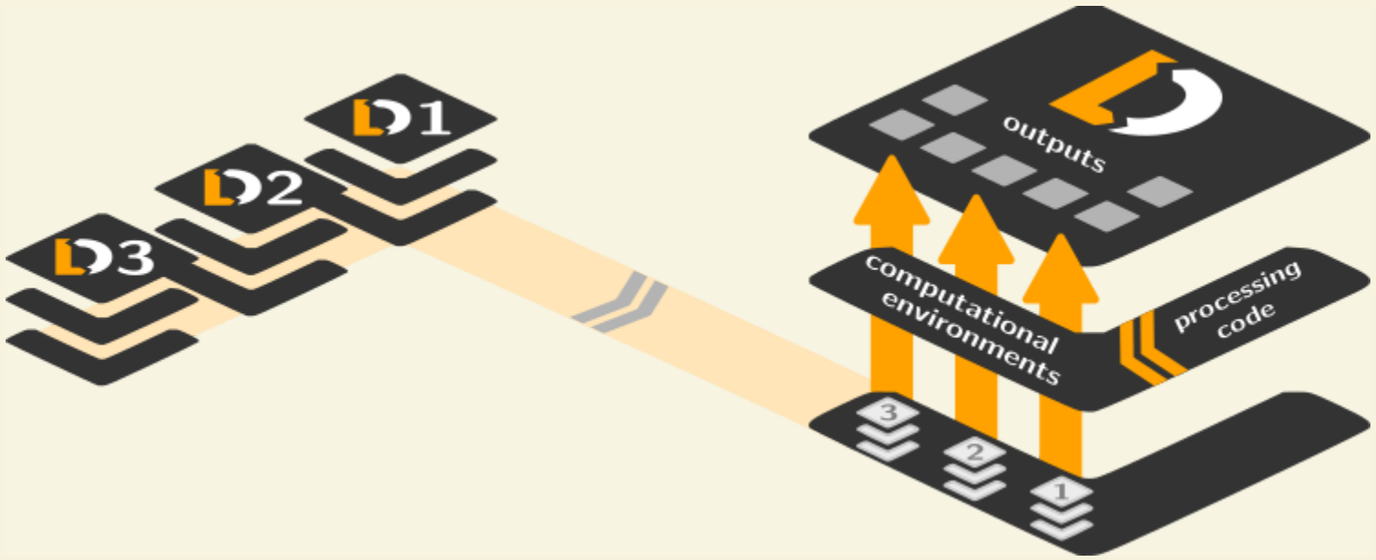


capture how exactly the content of every file

(that was not obtained from elsewhere)

came to be

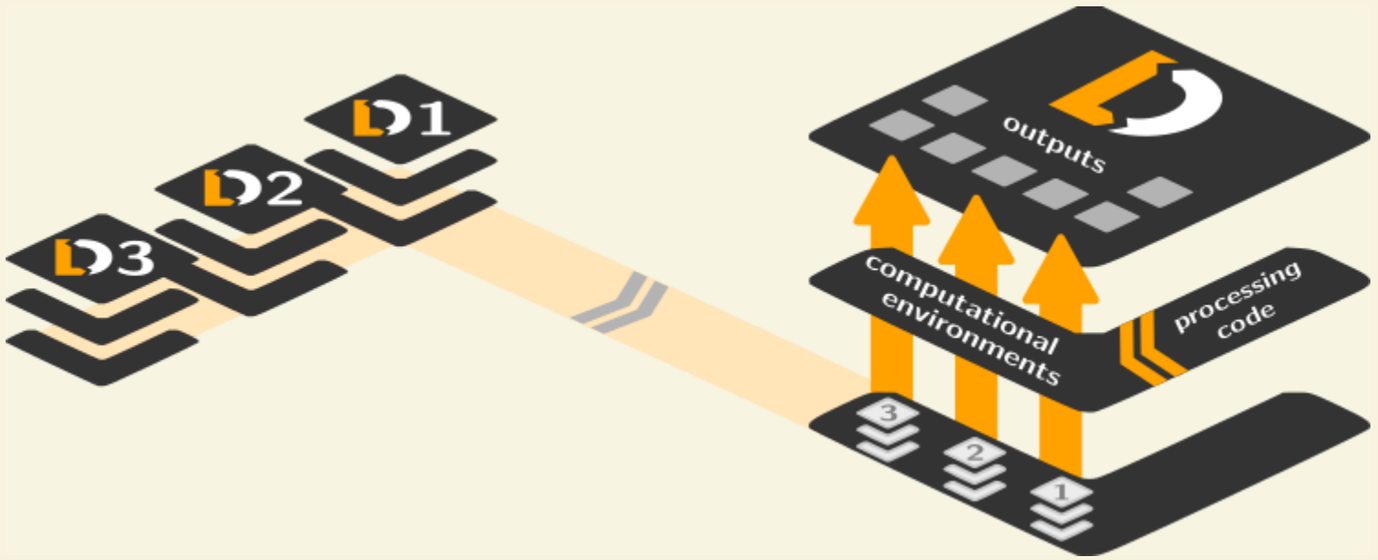
DATA PROVENANCE CAPTURE




```
$ datalad run -m "Perform eye movement event detection"\
  --input 'inputs/raw_eyegaze/sub-*/beh/sub-*...tsv.gz' \
  --output 'sub-*' \
  bash code/compute_all.sh

-- Git commit -- Michael Hanke <michael.hanke@gmail.com>; Fri Sep 21 22:00:47 2018
    [DATALAD RUNCMD] Perform eye movement event detection
    === Do not change lines below ===
    {
     "cmd": "bash code/compute_all.sh",
     "dsid": "d2b4b72a-7c13-11e7-9f1f-a0369f7c647e",
     "exit": 0,
     "inputs": ["inputs/raw_eyegaze/sub-*/beh/sub-*_task-movie_run-*_....tsv.gz"],
     "outputs": ["sub-*"],
     "pwd": "."
    }
    ^^^ Do not change lines above ^^^
---
 sub-01/sub-01_task-movie_run-1_events.png | 2 +-
 sub-01/sub-01_task-movie_run-1_events.tsv | 2 +-
...
```

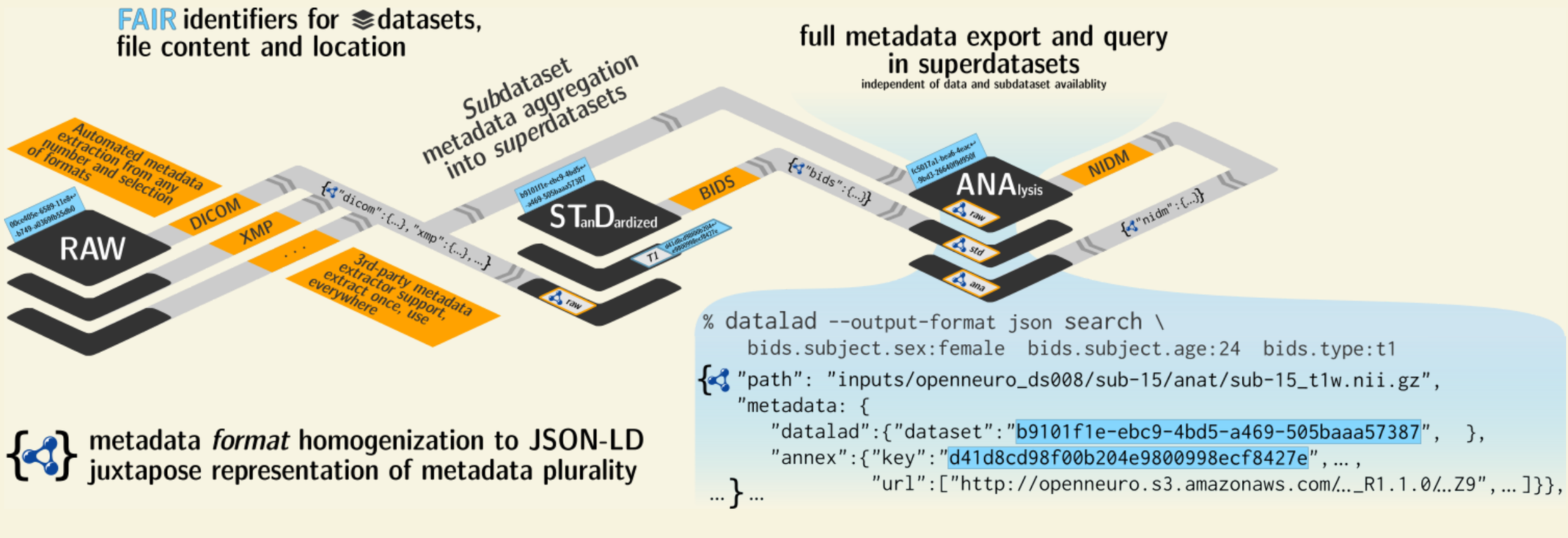
"COMPLETE" PROVENANCE CAPTURE




```
$ datalad containers-run -n nilearn \
  --input 'inputs/mri_aligned/sub-*/in_bold3Tp2/sub-*_task-avmovie_run-*_bold*' \
  --output 'sub-*/LC_timeseries_run-*.csv' \
  "bash -c 'for sub in sub-*; do for run in run-1 ... run-8;
     do python3 code/extract_lc_timeseries.py \$sub \$run; done; done'"

-- Git commit -- Michael Hanke <michael.hanke@gmail.com>; Fri Jul 6 11:02:28 2018
    [DATALAD RUNCMD] singularity exec --bind {pwd} .datalad/e...
    === Do not change lines below ===
    {
     "cmd": "singularity exec --bind {pwd} .datalad/environments/nilearn.simg bash..",
     "dsid": "92ea1faa-632a-11e8-af29-a0369f7c647e",
     "exit": 0,
     "inputs": [
      "inputs/mri_aligned/sub-*/in_bold3Tp2/sub-*_task-avmovie_run-*_bold*",
      ".datalad/environments/nilearn.simg"
     ],
     "outputs": ["sub-*/LC_timeseries_run-*.csv"],
     "pwd": "."
    }
    ^^^ Do not change lines above ^^^
---
 sub-01/LC_timeseries_run-1.csv | 1 +
 sub-01/LC_timeseries_run-2.csv | 1 +
...
```

SCALABLE AND ACTIONABLE (META)DATA REPRESENTATIONS



{} metadata *format* homogenization to JSON-LD
juxtapose representation of metadata plurality

- (meta)data logistics solution with **built-in provenance capture**
- automatic **metadata updates** to track evolution of standards
- integrates well with storage/hosting technology
- facilitates building metadata-driven applications
- viable system to **bring computation to data**

MINIMAL USAGE TO GET MOST OF THE FEATURES

- Create a dataset:

  `datalad create <name>`

- Save the state of a dataset after manual changes:

  `datalad save`

- Capture command output:

  `datalad run <shell command>`