Data management

Local version control workflows

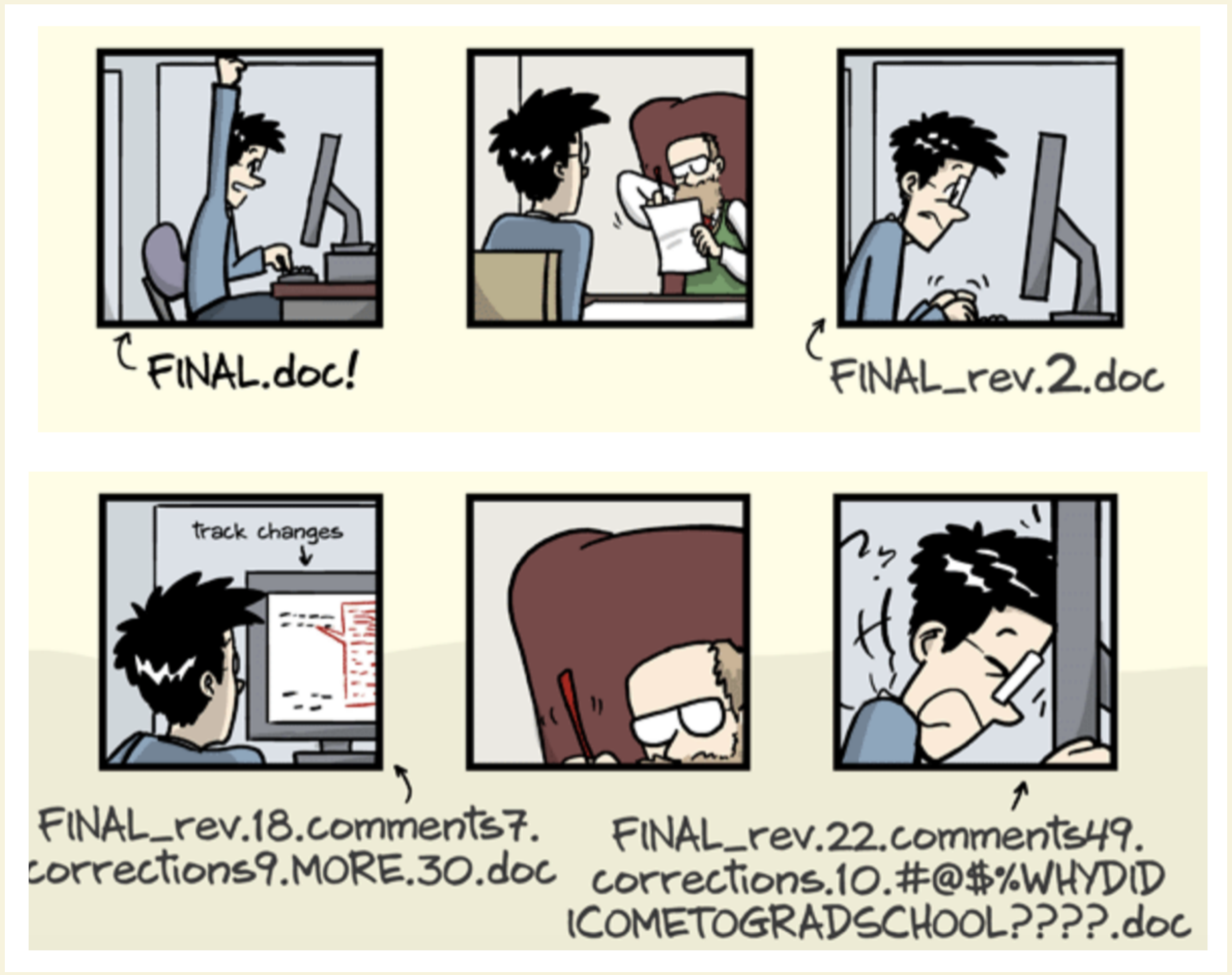

FINAL.doc!
FINAL_rev.2.doc
track changes
FINAL_rev.18.comments7.corrections9.MORE.30.doc
FINAL_rev.22.comments49.corrections.10.#@$%WHYDIDICOMETOGRADSCHOOL????.doc

Image credit: Full comic at http://phdcomics.com/comics.php?f=1531

LAST SESSION...

- **Prerequisites & technicalities**: Git identity, Gitlab/Github, Reathedocs, the Handbook
- **Theory**: DataLad datasets, YODA principles
- **Practice**: Create, structure, and install datasets

LAST SESSION...

- **Prerequisites & technicalities**: Git identity, Gitlab/Github, Reathedocs, the Handbook
- **Theory**: DataLad datasets, YODA principles
- **Practice**: Create, structure, and install datasets

CHANGE IN SESSION SET-UP:

LAST SESSION...

- **Prerequisites & technicalities**: Git identity, Gitlab/Github, Reathedocs, the Handbook
- **Theory**: DataLad datasets, YODA principles
- **Practice**: Create, structure, and install datasets

CHANGE IN SESSION SET-UP:

- Sessions oriented at handbook content (totalling 6 sessions)

LAST SESSION...

- **Prerequisites & technicalities**: Git identity, Gitlab/Github, Reathedocs, the Handbook
- **Theory**: DataLad datasets, YODA principles
- **Practice**: Create, structure, and install datasets

CHANGE IN SESSION SET-UP:

- Sessions oriented at handbook content (totalling 6 sessions)
- Domain-agnostic narrative: "Educational course" on DataLad

LAST SESSION...

- **Prerequisites & technicalities**: Git identity, Gitlab/Github, Reathedocs, the Handbook
- **Theory**: DataLad datasets, YODA principles
- **Practice**: Create, structure, and install datasets

CHANGE IN SESSION SET-UP:

- Sessions oriented at handbook content (totalling 6 sessions)
- Domain-agnostic narrative: "Educational course" on DataLad
- Live code-casts to follow along in your own terminal

OBJECTIVES

Local version control workflows

- Saving modifications to datasets: adding and changing files
- Installing (sub)datasets
- Dataset nesting

OBJECTIVES

Local version control workflows

- Saving modifications to datasets: adding and changing files
- Installing (sub)datasets
- Dataset nesting

Let's get into a terminal!

OBJECTIVES

Local version control workflows

- Saving modifications to datasets: adding and changing files
- Installing (sub)datasets
- Dataset nesting

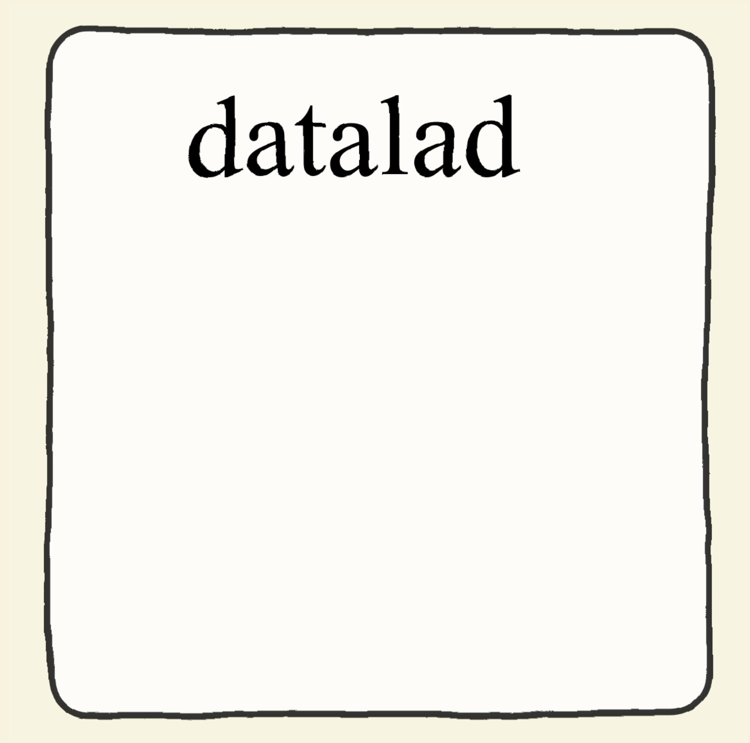
Let's get into a terminal!

-> handbook chapter

DATALAD DATASETS

DATALAD DATASETS

- DataLad is build on top of other tools:



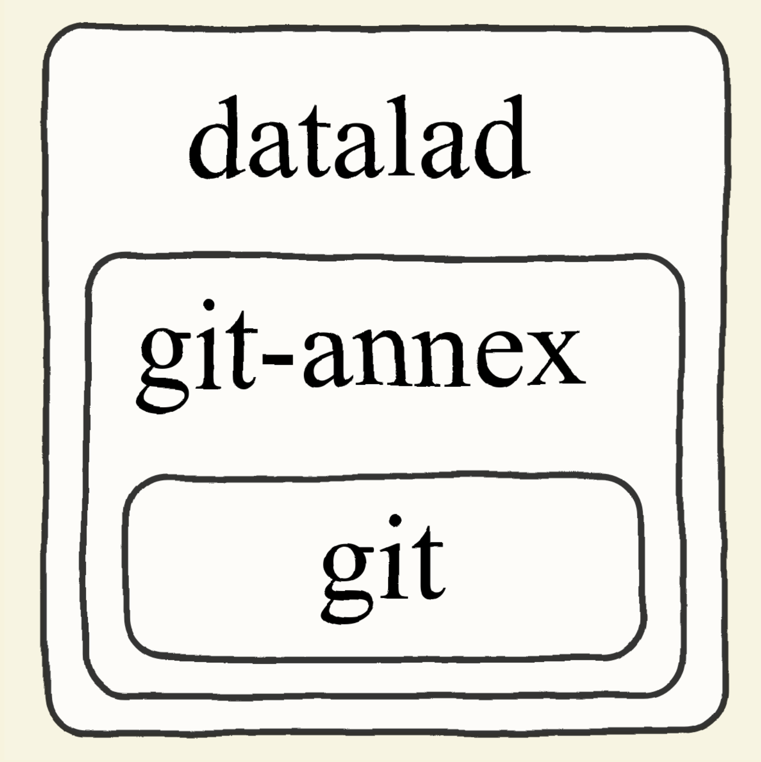
DATALAD DATASETS

The foundation is Git:

Datasets are Git repositories. If you want, use any Git command/feature!

Git-annex handles large file content:

(Large) file content is "annexed".

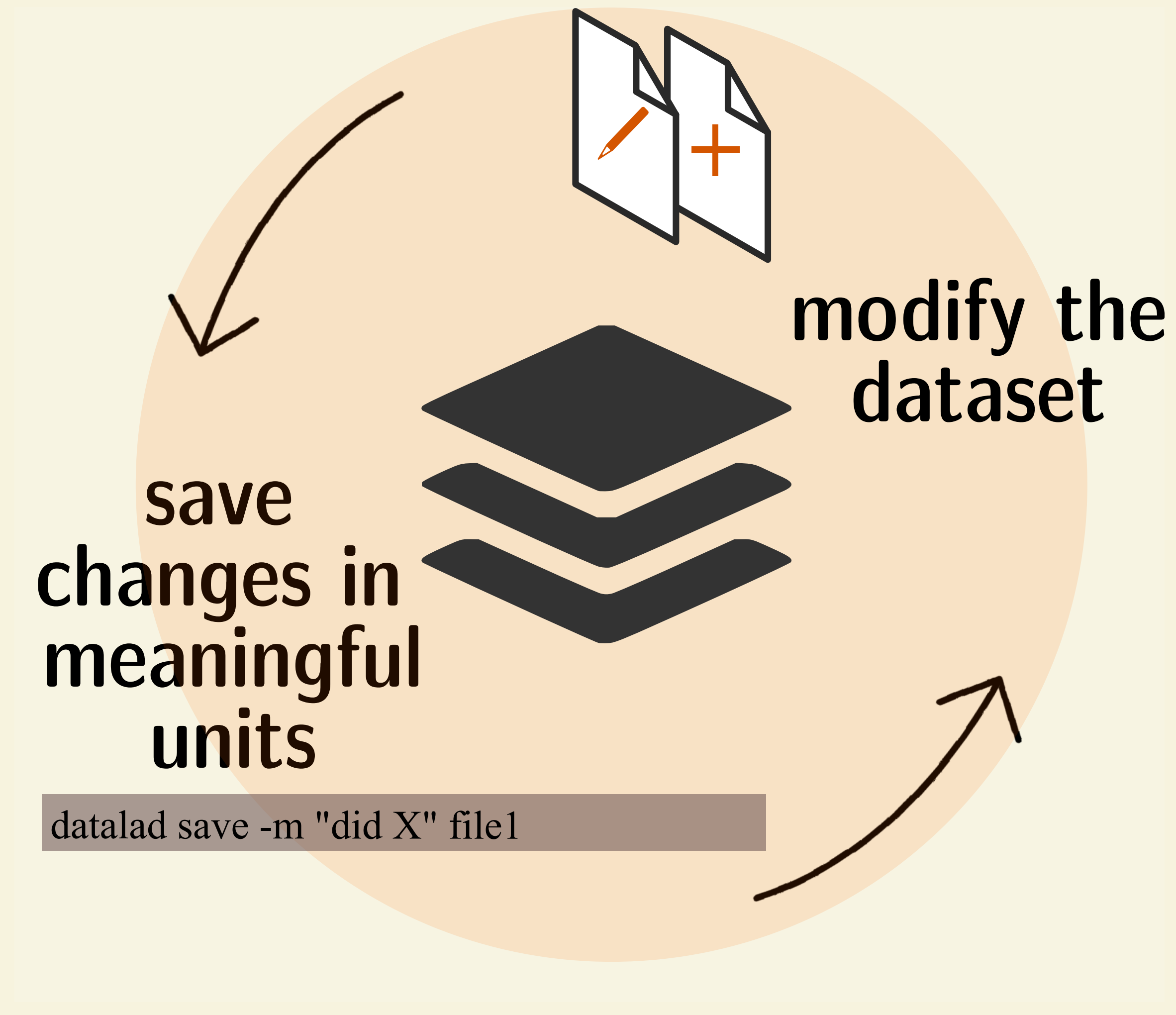


LOCAL VERSION CONTROL

Procedurally, version control is easy with DataLad!

LOCAL VERSION CONTROL

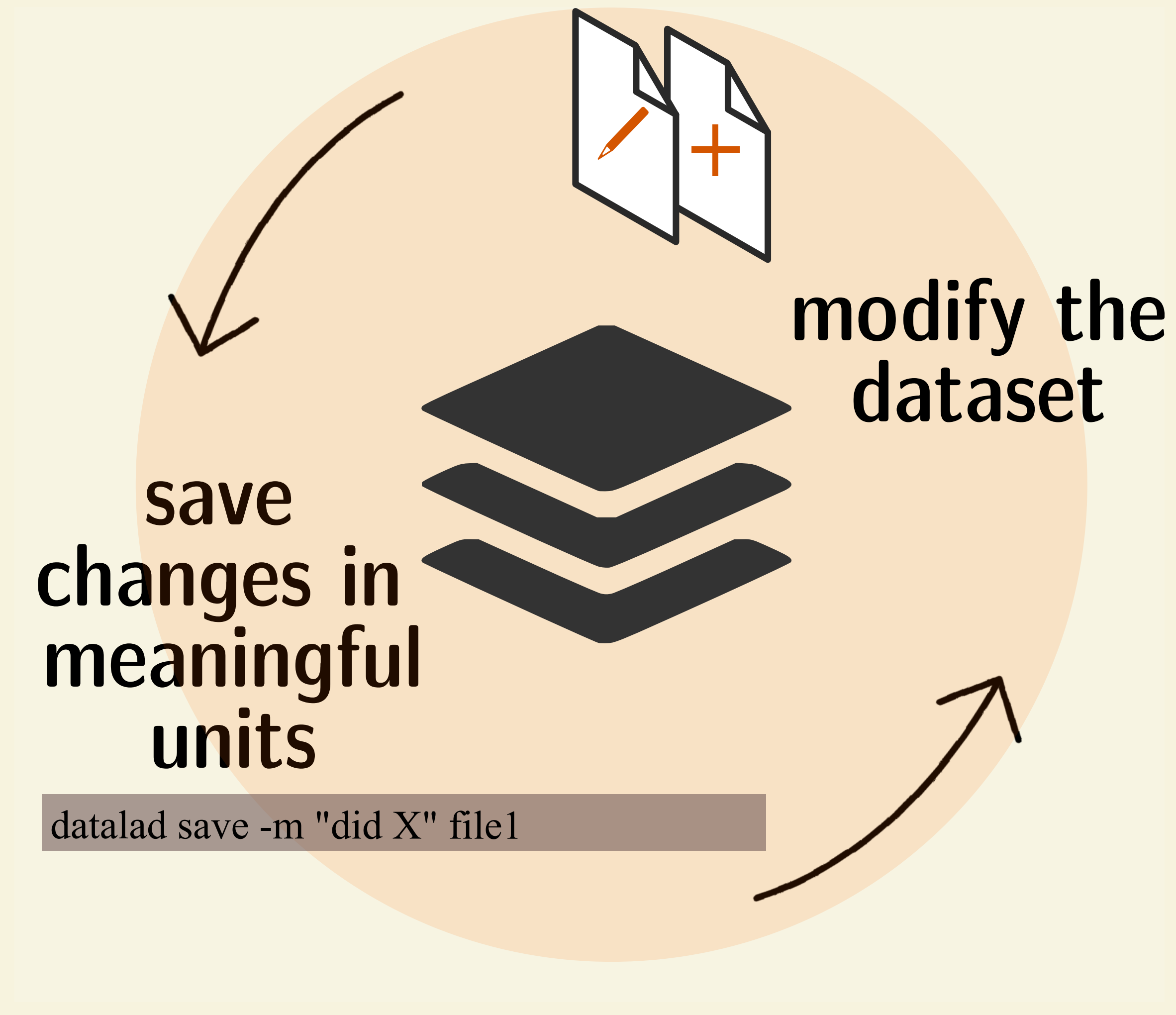
Procedurally, version control is easy with DataLad!



```
datalad save -m "did X" file1
```

LOCAL VERSION CONTROL

Procedurally, version control is easy with DataLad!

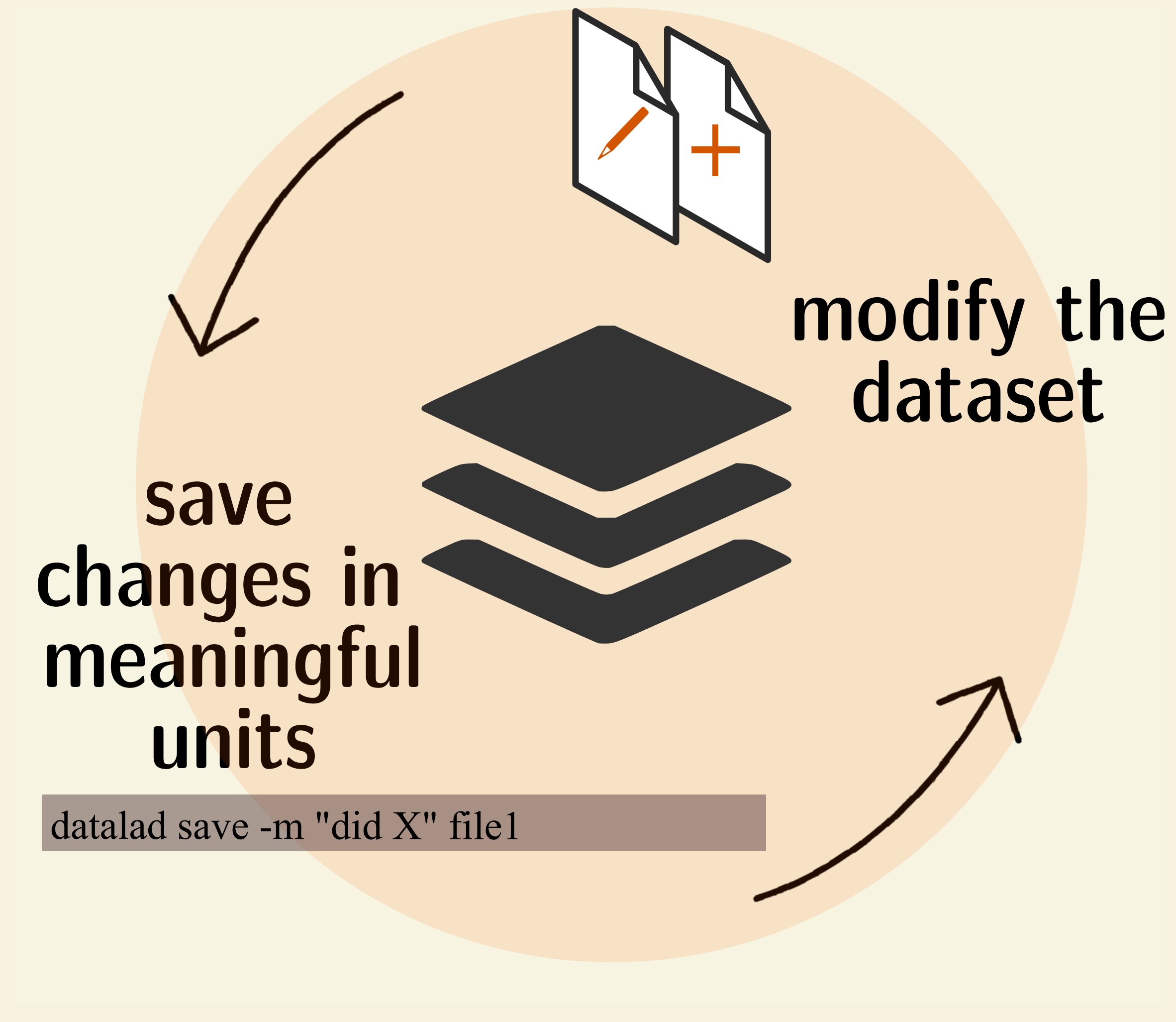


```
datalad save -m "did X" file1
```

Advice:

LOCAL VERSION CONTROL

Procedurally, version control is easy with DataLad!



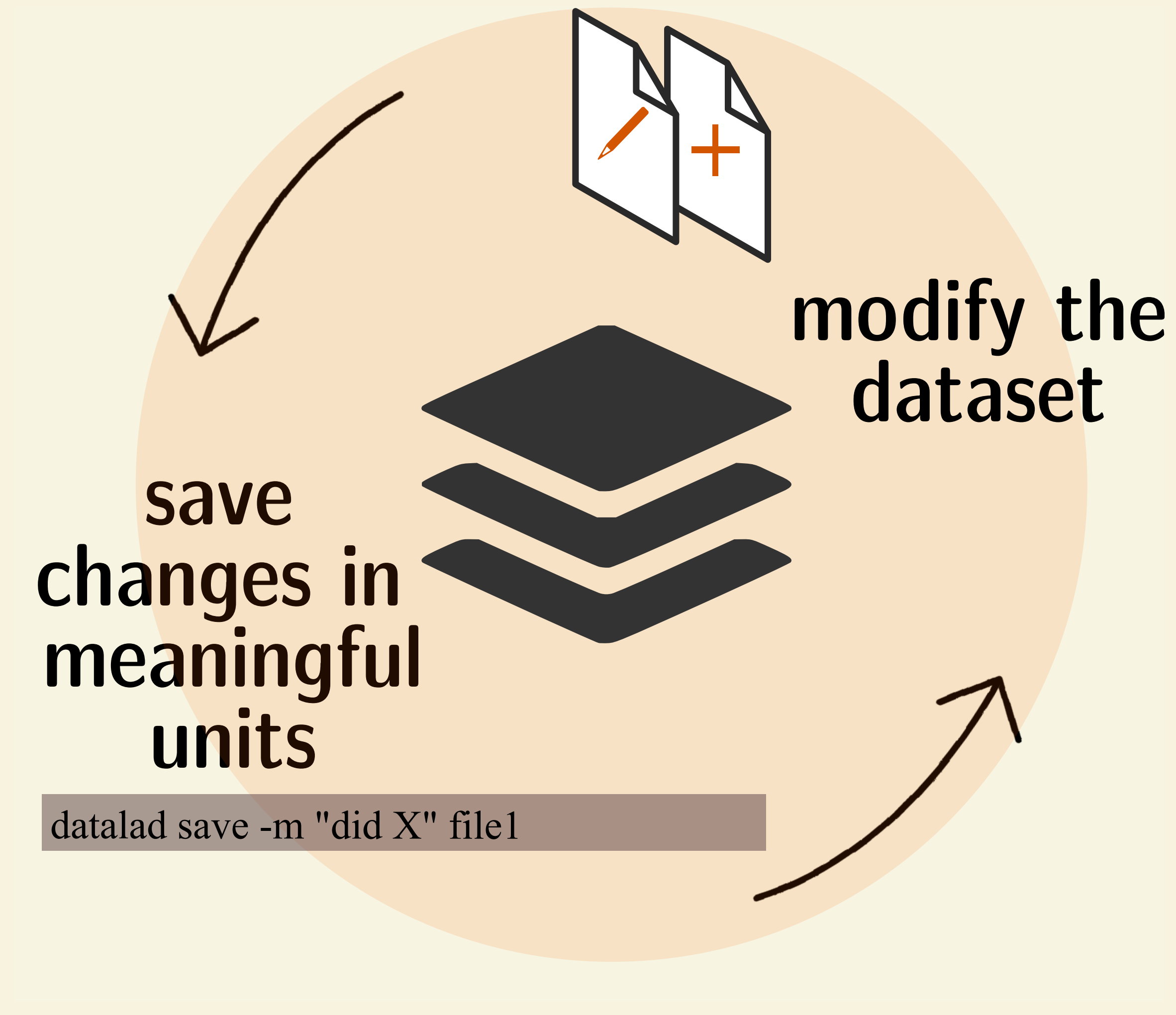
```
datalad save -m "did X" file1
```

Advice:

- Save *meaningful* units of change

LOCAL VERSION CONTROL

Procedurally, version control is easy with DataLad!



```
datalad save -m "did X" file1
```

Advice:

- Save *meaningful* units of change
- Attach helpful commit messages

SUMMARY - LOCAL VERSION CONTROL

SUMMARY - LOCAL VERSION CONTROL

`datalad create` creates an empty dataset.

SUMMARY - LOCAL VERSION CONTROL

`datalad create` **creates an empty dataset.**

- Configurations (**-c yoda**, **-c text2git**) are useful.

SUMMARY - LOCAL VERSION CONTROL

`datalad create` **creates an empty dataset.**

Configurations (**-c yoda**, **-c text2git**) are useful.

A dataset has a *history* to track files and their modifications.

SUMMARY - LOCAL VERSION CONTROL

`datalad create` creates an empty dataset.

Configurations (**-c yoda**, **-c text2git**) are useful.

A dataset has a *history* to track files and their modifications.

Explore it with Git (**git log**) or external tools (e.g., **tig**).

SUMMARY - LOCAL VERSION CONTROL

`datalad create` creates an empty dataset.

Configurations (**-c yoda**, **-c text2git**) are useful.

A dataset has a *history* to track files and their modifications.

Explore it with Git (**git log**) or external tools (e.g., **tig**).

`datalad save` records the dataset or file state to the history.

SUMMARY - LOCAL VERSION CONTROL

`datalad create` creates an empty dataset.

Configurations (**-c yoda**, **-c text2git**) are useful.

A dataset has a *history* to track files and their modifications.

Explore it with Git (**git log**) or external tools (e.g., **tig**).

`datalad save` records the dataset or file state to the history.

Concise **commit messages** should summarize the change for future you and others.

SUMMARY - LOCAL VERSION CONTROL

`datalad create` **creates an empty dataset.**

Configurations (**-c yoda**, **-c text2git**) are useful.

A dataset has a *history* to track files and their modifications.

Explore it with Git (**git log**) or external tools (e.g., **tig**).

`datalad save` **records the dataset or file state to the history.**

Concise **commit messages** should summarize the change for future you and others.

`datalad status` **reports the current state of the dataset.**

SUMMARY - LOCAL VERSION CONTROL

`datalad create` creates an empty dataset.
Configurations (**-c yoda**, **-c text2git**) are useful.

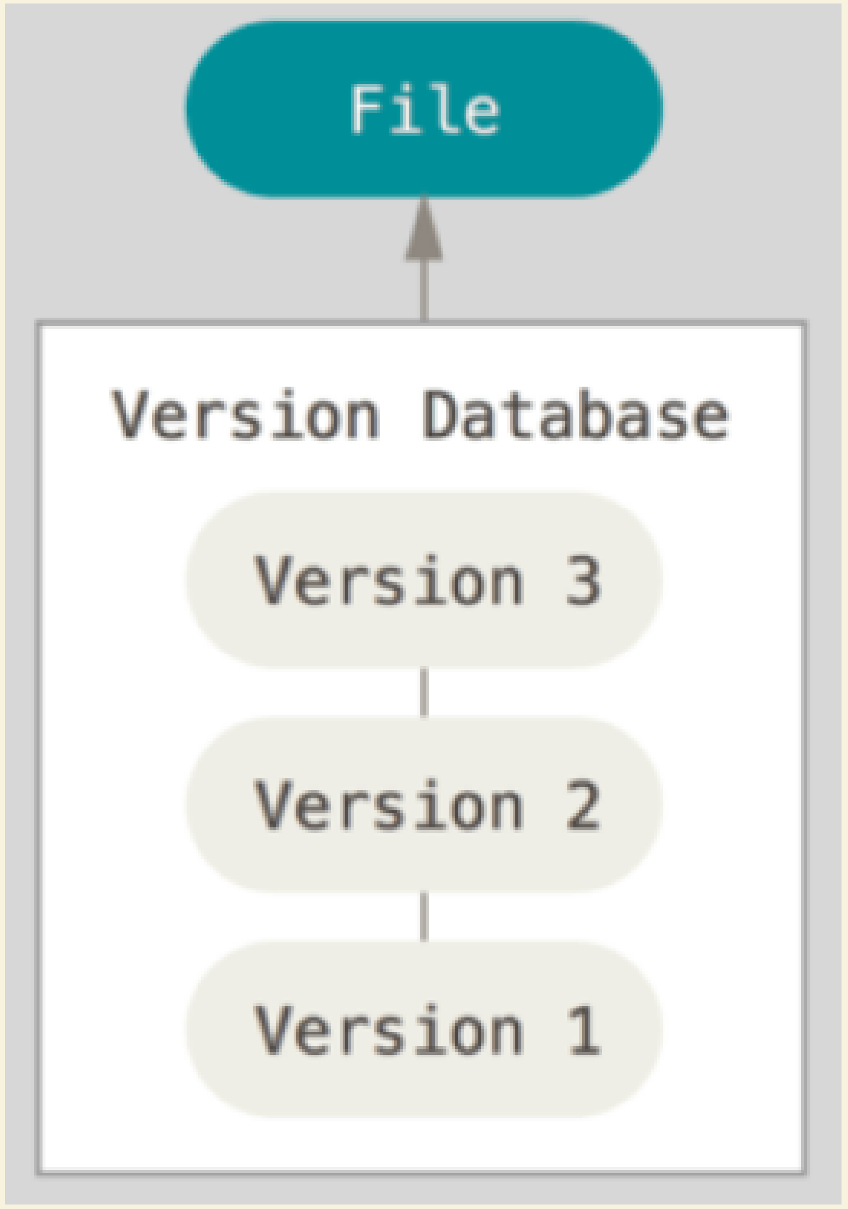
A dataset has a *history* to track files and their modifications.
Explore it with Git (**git log**) or external tools (e.g., **tig**).

`datalad save` records the dataset or file state to the history.
Concise **commit messages** should summarize the change for future you and others.

`datalad status` reports the current state of the dataset.
A clean dataset status is good practice.

SUMMARY - LOCAL VERSION CONTROL

SUMMARY - LOCAL VERSION CONTROL



INSTALLING DATASETS

INSTALLING DATASETS

```
$ datalad install --dataset . \
 --source https://github.com/datalad-datasets/longnow-podcasts.git \
 recordings/longnow
```

INSTALLING DATASETS

```
$ datalad install --dataset . \
 --source https://github.com/datalad-datasets/longnow-podcasts.git \
 recordings/longnow
```

- Installs a dataset (here: from Github) (--source/-s) ...

INSTALLING DATASETS

```
$ datalad install --dataset . \
 --source https://github.com/datalad-datasets/longnow-podcasts.git \
 recordings/longnow
```

- Installs a dataset (here: from Github) (--source/-s) ...
- into the *super*dataset Datalad-101 (--dataset/-d) under the path recordings/longnow ...

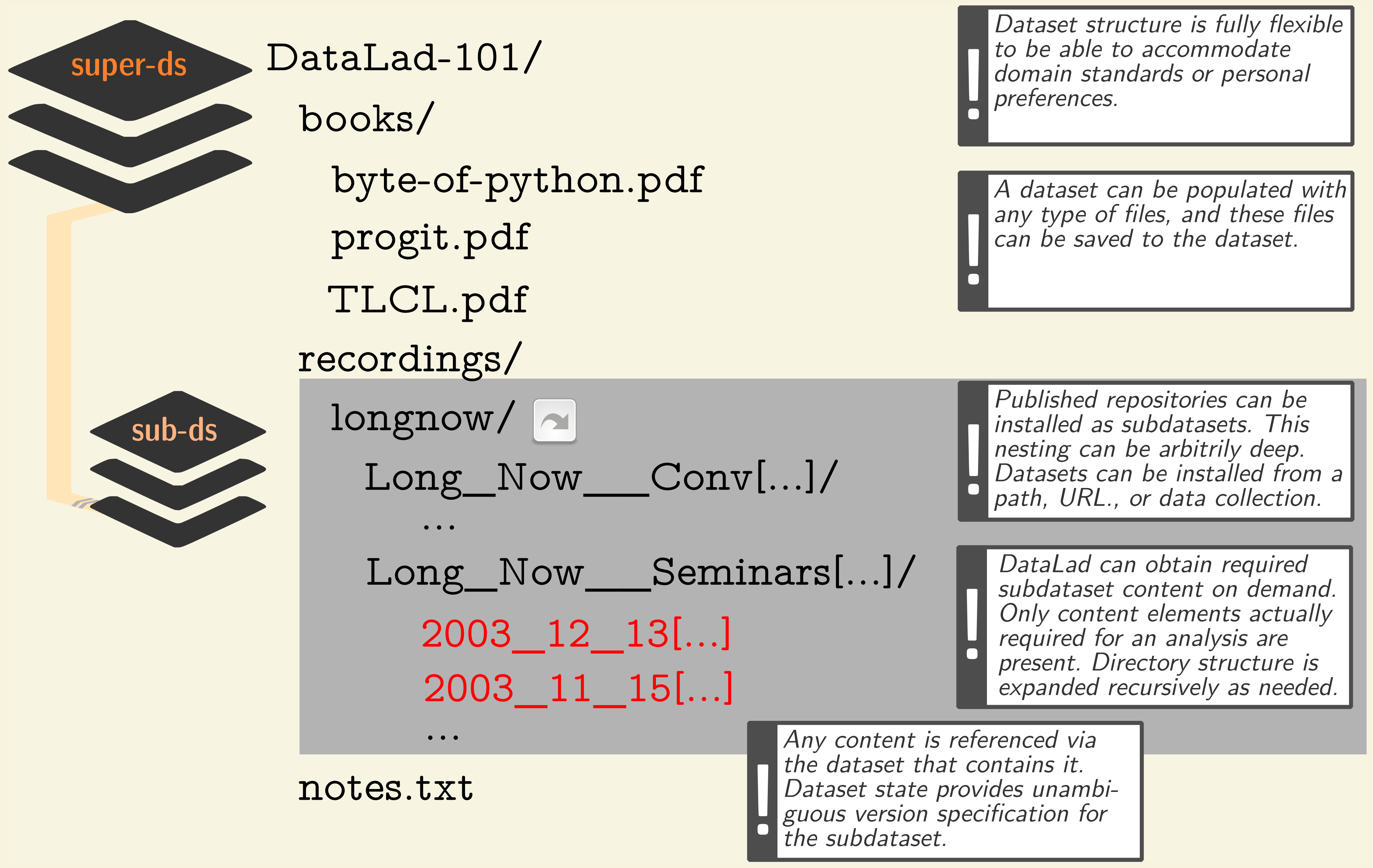
INSTALLING DATASETS

```
$ datalad install --dataset . \
 --source https://github.com/datalad-datasets/longnow-podcasts.git \
 recordings/longnow
```

- Installs a dataset (here: from Github) (--source/-s) ...
- into the *super*dataset Datalad-101 (--dataset/-d) under the path recordings/longnow ...
- and registers it as a *subdataset*.

INSTALLING DATASETS

INSTALLING DATASETS



```
DataLad-101/
  books/
    byte-of-python.pdf
    progit.pdf
    TLCL.pdf
  recordings/
    longnow/
      Long_Now__Conv[...]/
        ...
      Long_Now__Seminars[...]/
        2003_12_13[...]
        2003_11_15[...]
        ...
  notes.txt
```

! *Dataset structure is fully flexible to be able to accommodate domain standards or personal preferences.*

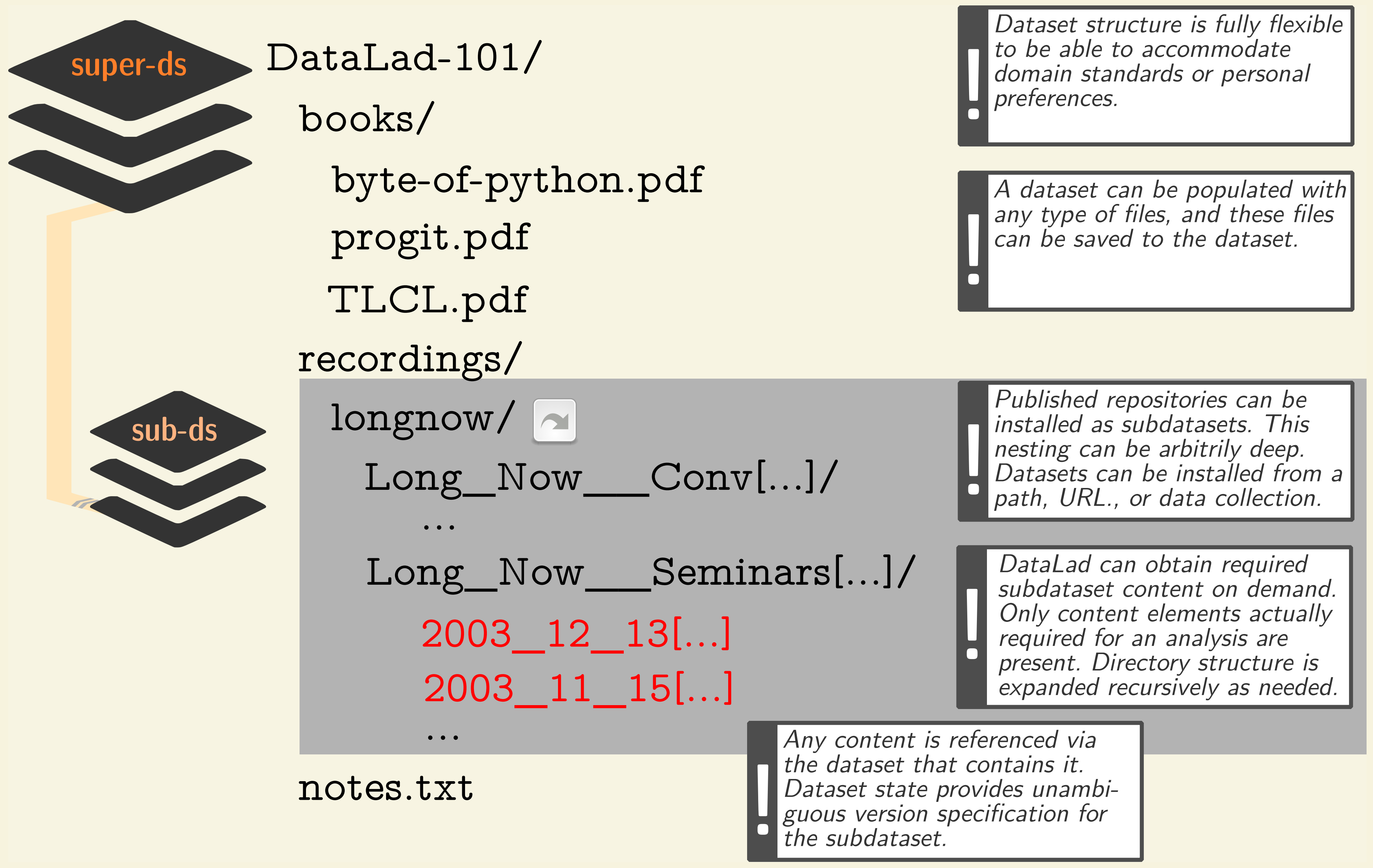
! *A dataset can be populated with any type of files, and these files can be saved to the dataset.*

! *Published repositories can be installed as subdatasets. This nesting can be arbitrily deep. Datasets can be installed from a path, URL., or data collection.*

! *DataLad can obtain required subdataset content on demand. Only content elements actually required for an analysis are present. Directory structure is expanded recursively as needed.*

! *Any content is referenced via the dataset that contains it. Dataset state provides unambiguous version specification for the subdataset.*

INSTALLING DATASETS



Dataset structure is fully flexible to be able to accommodate domain standards or personal preferences.

A dataset can be populated with any type of files, and these files can be saved to the dataset.

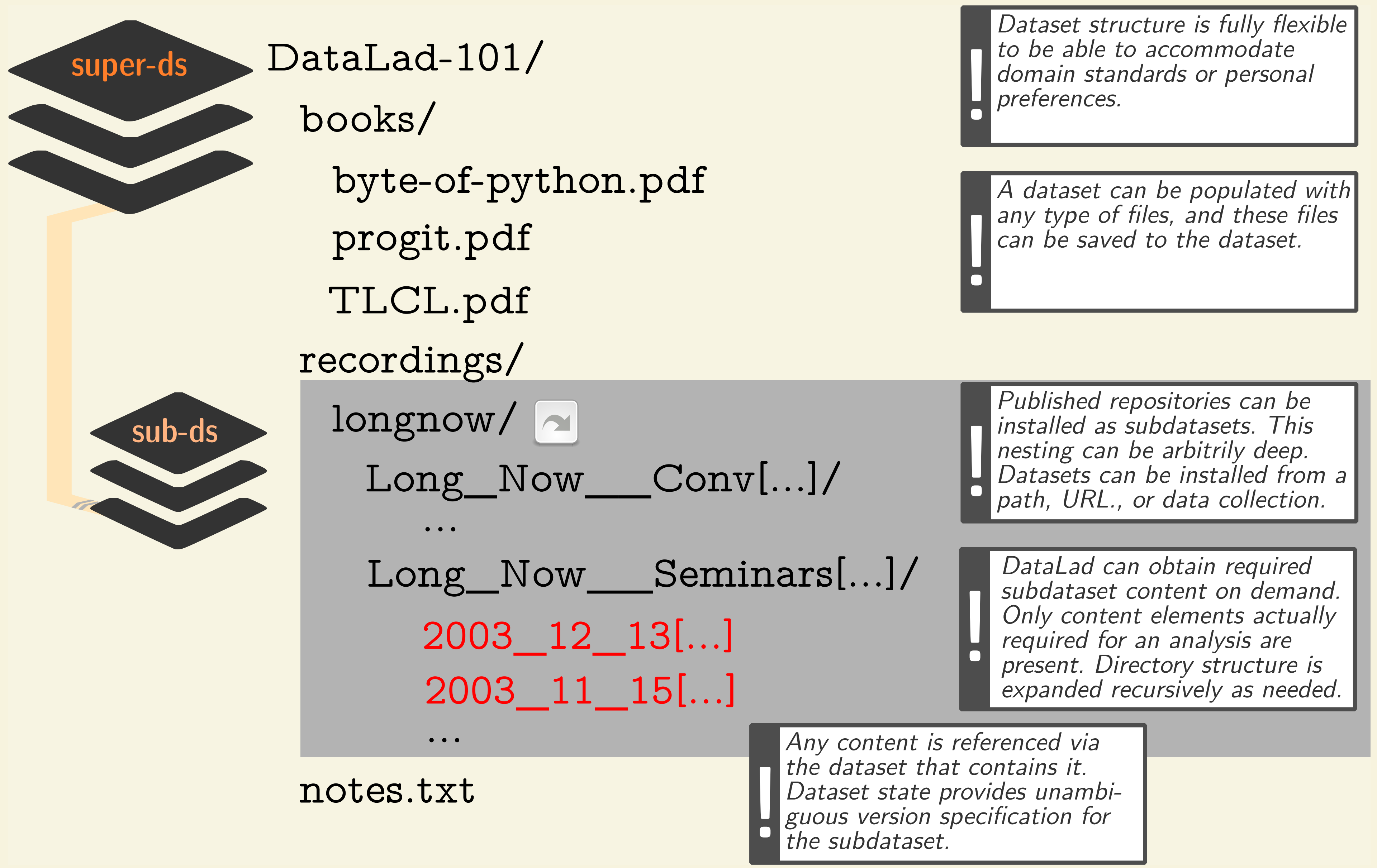
Published repositories can be installed as subdatasets. This nesting can be arbitrily deep. Datasets can be installed from a path, URL., or data collection.

DataLad can obtain required subdataset content on demand. Only content elements actually required for an analysis are present. Directory structure is expanded recursively as needed.

Any content is referenced via the dataset that contains it. Dataset state provides unambiguous version specification for the subdataset.

- Datasets are light-weight: Upon installation, only small files and meta data about file availability are retrieved.

INSTALLING DATASETS



! *Dataset structure is fully flexible to be able to accommodate domain standards or personal preferences.*

! *A dataset can be populated with any type of files, and these files can be saved to the dataset.*

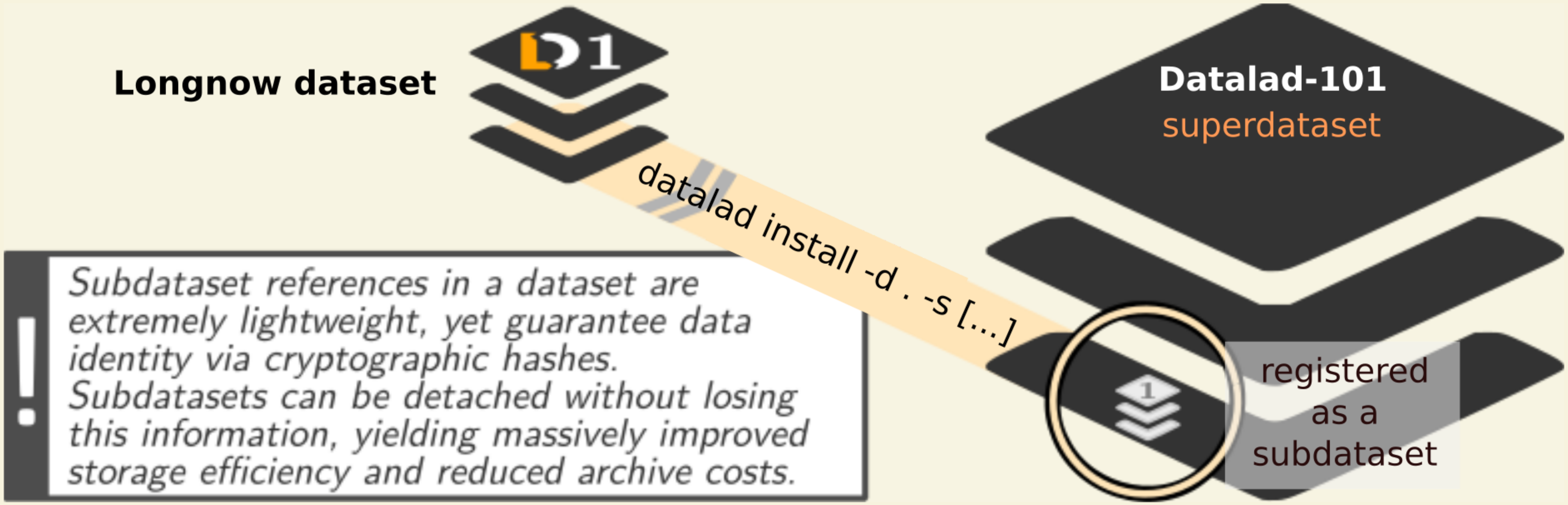
! *Published repositories can be installed as subdatasets. This nesting can be arbitrily deep. Datasets can be installed from a path, URL., or data collection.*

! *DataLad can obtain required subdataset content on demand. Only content elements actually required for an analysis are present. Directory structure is expanded recursively as needed.*

! *Any content is referenced via the dataset that contains it. Dataset state provides unambiguous version specification for the subdataset.*

- Datasets are light-weight: Upon installation, only small files and meta data about file availability are retrieved.
- Content can be obtained on demand via `datalad get`.

DATASET NESTING



Subdataset references in a dataset are extremely lightweight, yet guarantee data identity via cryptographic hashes. Subdatasets can be detached without losing this information, yielding massively improved storage efficiency and reduced archive costs.

```
commit 8fdf62acd0bf1e99ebcb6c466edc994a5f4013ba
Author: DataLad Demo demo@datalad.org
Date:   Sat Oct 26 15:54:44 2019 +0200

    [DATALAD] Recorded changes

diff --git a/.gitmodules b/.gitmodules
new file mode 100644
index 0000000..1b59b8c
--- /dev/null
+++ b/.gitmodules
@@ -0,0 +1,4 @@
+[submodule "recordings/longnow"]
+       path = recordings/longnow
+       url = https://github.com/datalad-datasets/longnow-podcasts.git
+       datalad-id = b3ca2718-8901-11e8-99aa-a0369f7c647e
diff --git a/recordings/longnow b/recordings/longnow
new file mode 160000
index 0000000..dcc34fb
--- /dev/null
```

```
+++ b/recordings/longnow
@@ -0,0 +1 @@
+Subproject commit dcc34fbe669b06ced84ced381ba0db21cf5e665f
```

SUMMARY - DATASET CONSUMPTION & NESTING

SUMMARY - DATASET CONSUMPTION & NESTING

`datalad install` installs a dataset.

SUMMARY - DATASET CONSUMPTION & NESTING

`datalad install` **installs a dataset.**

It can be installed “on its own”: Specify the **--source/-s** of the dataset, and an optional **path** for it to be installed to.

SUMMARY - DATASET CONSUMPTION & NESTING

`datalad install` installs a dataset.

> It can be installed “on its own”: Specify the **--source/-s** of the dataset, and an optional **path** for it to be installed to.

Datasets can be installed as subdatasets within an existing dataset.

SUMMARY - DATASET CONSUMPTION & NESTING

`datalad install` installs a dataset.

It can be installed “on its own”: Specify the **--source/-s** of the dataset, and an optional **path** for it to be installed to.

Datasets can be installed as subdatasets within an existing dataset.

The **--dataset/-d** option needs a path to the root of the superdataset.

SUMMARY - DATASET CONSUMPTION & NESTING

`datalad install` installs a dataset.

It can be installed “on its own”: Specify the **--source/-s** of the dataset, and an optional **path** for it to be installed to.

Datasets can be installed as subdatasets within an existing dataset.

The **--dataset/-d** option needs a path to the root of the superdataset.

Only small files and metadata about file availability are present locally after an install.

SUMMARY - DATASET CONSUMPTION & NESTING

`datalad install` installs a dataset.

It can be installed “on its own”: Specify the **--source/-s** of the dataset, and an optional **path** for it to be installed to.

Datasets can be installed as subdatasets within an existing dataset.

The **--dataset/-d** option needs a path to the root of the superdataset.

Only small files and metadata about file availability are present locally after an install.

To retrieve actual file content of larger files, `datalad get` downloads large file content on demand.

SUMMARY - DATASET CONSUMPTION & NESTING

`datalad install` installs a dataset.

It can be installed “on its own”: Specify the **--source/-s** of the dataset, and an optional **path** for it to be installed to.

Datasets can be installed as subdatasets within an existing dataset.

The **--dataset/-d** option needs a path to the root of the superdataset.

Only small files and metadata about file availability are present locally after an install.

To retrieve actual file content of larger files, `datalad get` downloads large file content on demand.

`datalad status` can report on total and retrieved repository size

SUMMARY - DATASET CONSUMPTION & NESTING

`datalad install` installs a dataset.

It can be installed “on its own”: Specify the **--source/-s** of the dataset, and an optional **path** for it to be installed to.

Datasets can be installed as subdatasets within an existing dataset.

The **--dataset/-d** option needs a path to the root of the superdataset.

Only small files and metadata about file availability are present locally after an install.

To retrieve actual file content of larger files, `datalad get` downloads large file content on demand.

`datalad status` can report on total and retrieved repository size

using `--annex` and `--annex all` options.

SUMMARY - DATASET CONSUMPTION & NESTING

`datalad install` installs a dataset.

It can be installed "on its own": Specify the **--source/-s** of the dataset, and an optional **path** for it to be installed to.

Datasets can be installed as subdatasets within an existing dataset.

The **--dataset/-d** option needs a path to the root of the superdataset.

Only small files and metadata about file availability are present locally after an install.

To retrieve actual file content of larger files, `datalad get` downloads large file content on demand.

`datalad status` can report on total and retrieved repository size

using `--annex` and `--annex all` options.

Datasets preserve their history.

SUMMARY - DATASET CONSUMPTION & NESTING

`datalad install` installs a dataset.

It can be installed “on its own”: Specify the **--source/-s** of the dataset, and an optional **path** for it to be installed to.

Datasets can be installed as subdatasets within an existing dataset.

The **--dataset/-d** option needs a path to the root of the superdataset.

Only small files and metadata about file availability are present locally after an install.

To retrieve actual file content of larger files, `datalad get` downloads large file content on demand.

`datalad status` can report on total and retrieved repository size

using `--annex` and `--annex all` options.

Datasets preserve their history.

The superdataset records only the *version state* of the subdataset.

NOW WHAT I CAN DO WITH THAT?

Local version control

- Version control changing small files (code, manuscripts (text!), ...)
- Add large files to a dataset history
- Meaninful and well-described commits will make future interactions with the dataset history easier

Dataset installation and nesting

- Consume existing datasets
- Link datasets together

PRACTICE @HOME

- Start a coding project or take an existing project and version control your work with DataLad. Remember to create datasets with the `text2git` or `yoda` configuration!

FURTHER READING

The basics on datasets:

- Chapter DataLad Datasets in the handbook.

How to get help on commands and their options:

- Section How to get help in the handbook

OUTLINE: WHAT COMES NEXT?

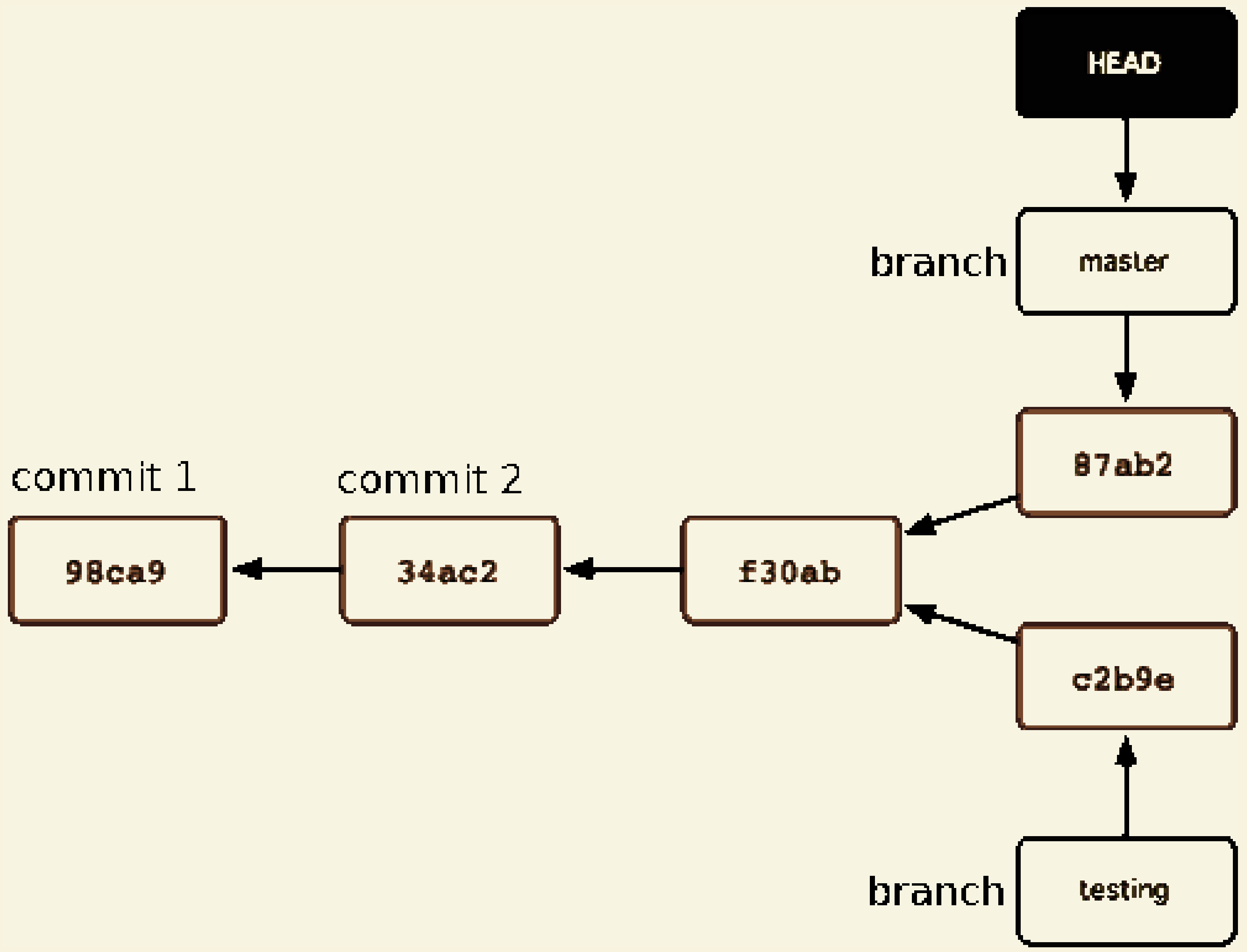
- Reproducible analyses with DataLad (chapter DataLad, Run! in the handbook).
- **Which date is suitable?** > Doodle poll <

OPEN QUESTION SESSION

BACKUP SLIDES FOR ANTICIPATED QUESTIONS

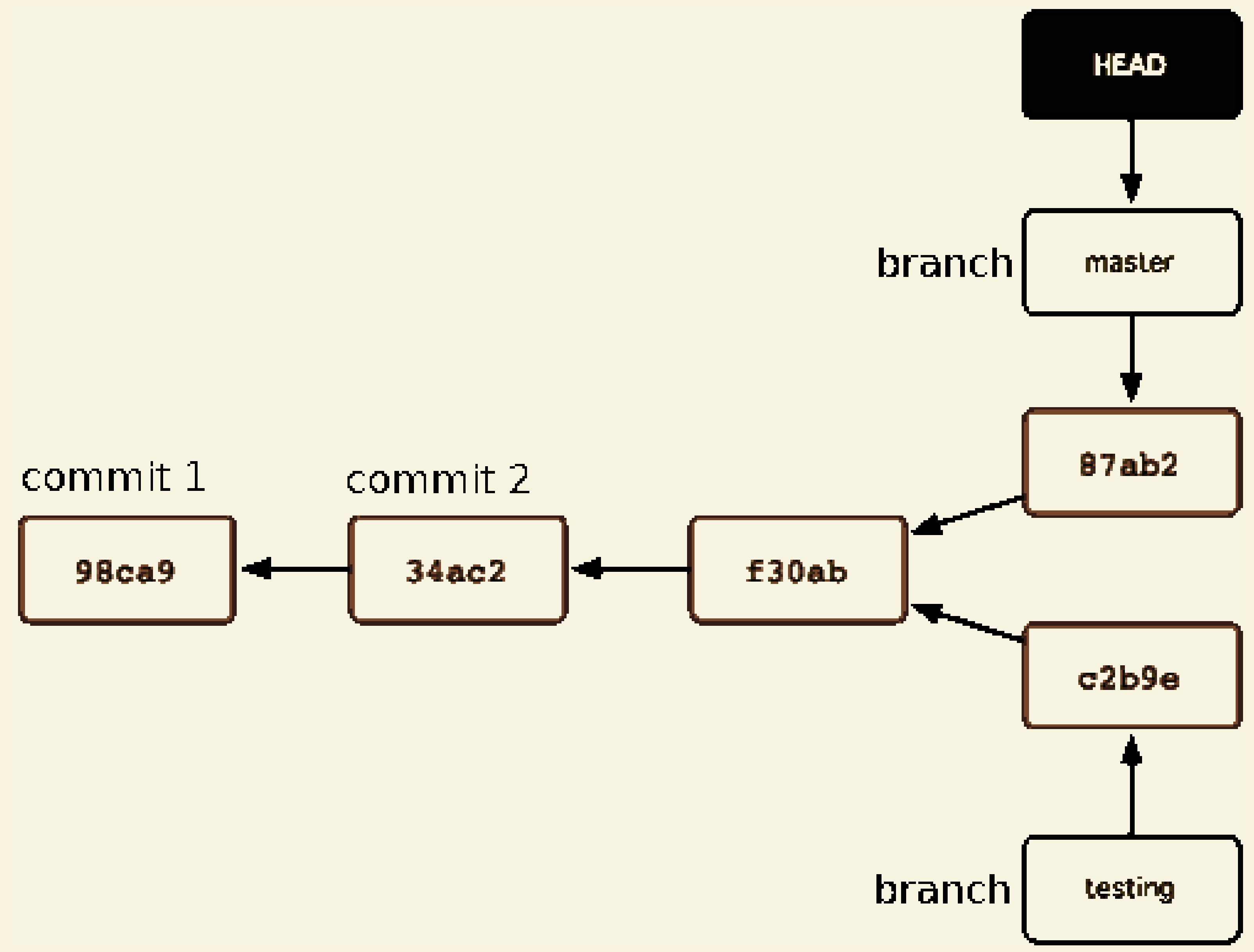
WHAT IS HEAD?

- A git repository is build up as a *tree* of **commits** (history entries).
- A **branch** is a named pointer (reference) to a commit, and allows to isolate developments. The default branch is called `master`.
- HEAD is a pointer to the branch you are on, and thus to the last commit in the given branch.



WHAT IS HEAD?

- A git repository is build up as a *tree* of **commits** (history entries).
- A **branch** is a named pointer (reference) to a commit, and allows to isolate developments. The default branch is called `master`.
- HEAD is a pointer to the branch you are on, and thus to the last commit in the given branch.



If you'd be on branch "testing", which commit would HEAD point to?

HOW DOES A HERE-DOCUMENT WORK?

```
$ cat << EOT > notes.txt
One can create a new dataset with 'datalad create [--description] PATH'.
The dataset is created empty

EOT
```

- Two *delimiting identifiers* (EOT) wrap any amount of text into a stream
- The << characters *redirect* the stream into *standard input* for the `cat` command
- The > character *redirects* the *standard output* of `cat` and writes it into a new file `notes.txt`

HOW DOES A HERE-DOCUMENT WORK?

```
$ cat << EOT > notes.txt
One can create a new dataset with 'datalad create [--description] PATH'.
The dataset is created empty

EOT
```

- Two *delimiting identifiers* (EOT) wrap any amount of text into a stream
- The << characters *redirect* the stream into *standard input* for the `cat` command
- The > character *redirects* the *standard output* of `cat` and writes it into a new file `notes.txt`

Why is it used?

HOW DOES A HERE-DOCUMENT WORK?

```
$ cat << EOT > notes.txt
One can create a new dataset with 'datalad create [--description] PATH'.
The dataset is created empty

EOT
```

- Two *delimiting identifiers* (EOT) wrap any amount of text into a stream
- The << characters *redirect* the stream into *standard input* for the `cat` command
- The > character *redirects* the *standard output* of `cat` and writes it into a new file `notes.txt`

Why is it used?

- Allows pretty formating (e.g., line breaks)
- Allows writing documents from the terminal

WHAT'S WITH ALL THOSE CONFIGURATIONS?

WHAT'S WITH ALL THOSE CONFIGURATIONS?

- Contents of a dataset can be stored in Git or Git-annex.

WHAT'S WITH ALL THOSE CONFIGURATIONS?

- Contents of a dataset can be stored in Git or Git-annex.
- Files stored in Git *can* be more easily modifiable.

WHAT'S WITH ALL THOSE CONFIGURATIONS?

- Contents of a dataset can be stored in Git or Git-annex.
- Files stored in Git *can* be more easily modifiable.
- By default, all contents of a dataset are stored in Git-annex.

WHAT'S WITH ALL THOSE CONFIGURATIONS?

- Contents of a dataset can be stored in Git or Git-annex.
- Files stored in Git *can* be more easily modifiable.
- By default, all contents of a dataset are stored in Git-annex.
- Configurations like `text2git` and `yoda` configure the dataset to store files of certain types or in certain locations to be stored in Git

WHAT'S WITH ALL THOSE CONFIGURATIONS?

- Contents of a dataset can be stored in Git or Git-annex.
- Files stored in Git *can* be more easily modifiable.
- By default, all contents of a dataset are stored in Git-annex.
- Configurations like `text2git` and `yoda` configure the dataset to store files of certain types or in certain locations to be stored in Git
- More about this in Under the hood: Git-annex.