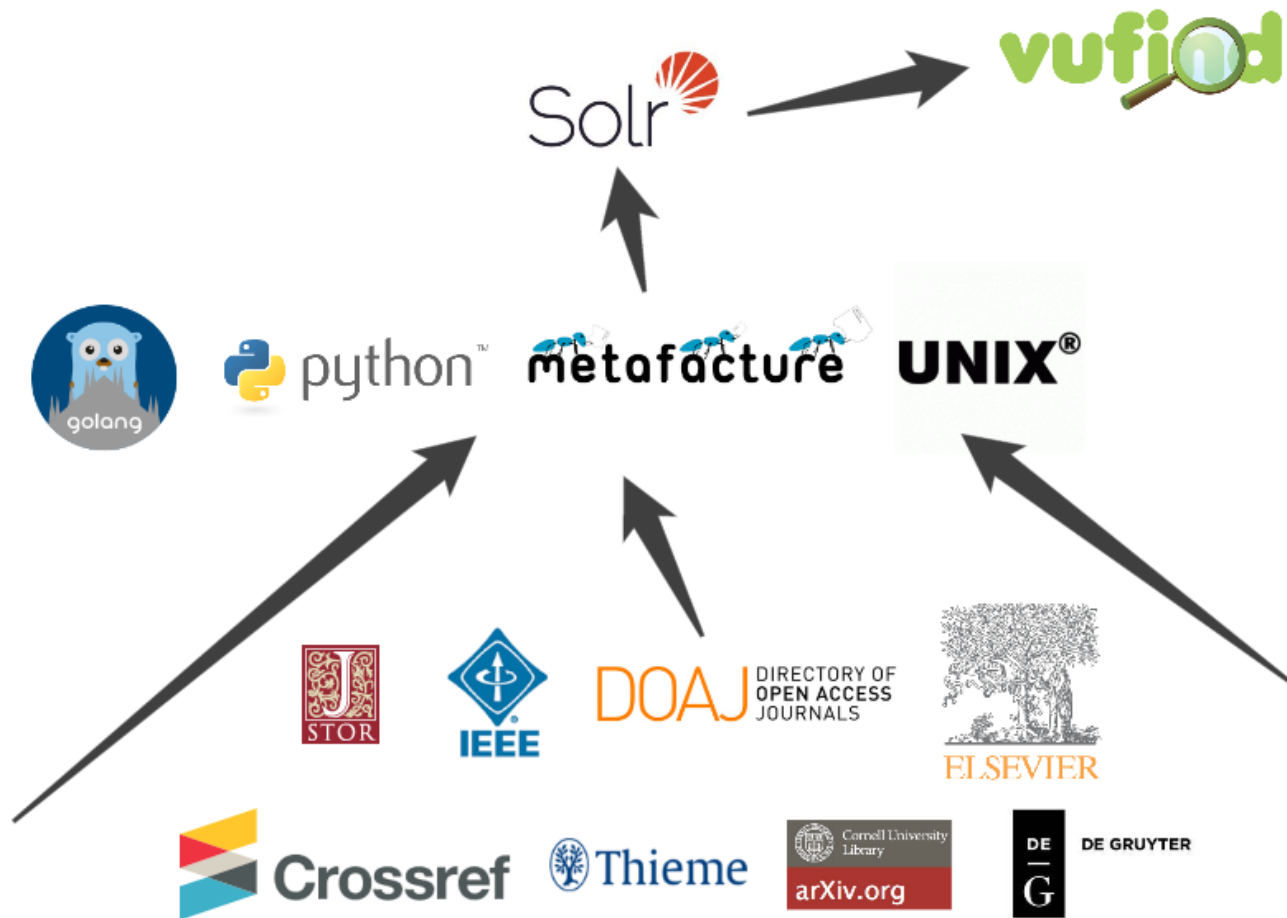


# Building Metadata Indices

2017-05-17, 12:30-15:00, Leipzig University Library

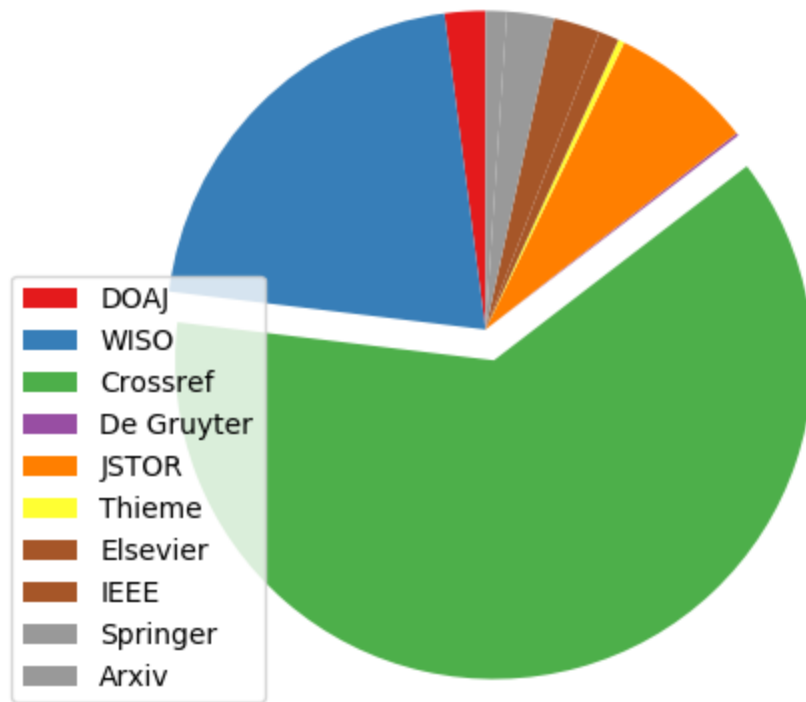
Martin Czygan, Tracy Hoffmann, Robert Schenk, et al.

# Overview



# Source distribution

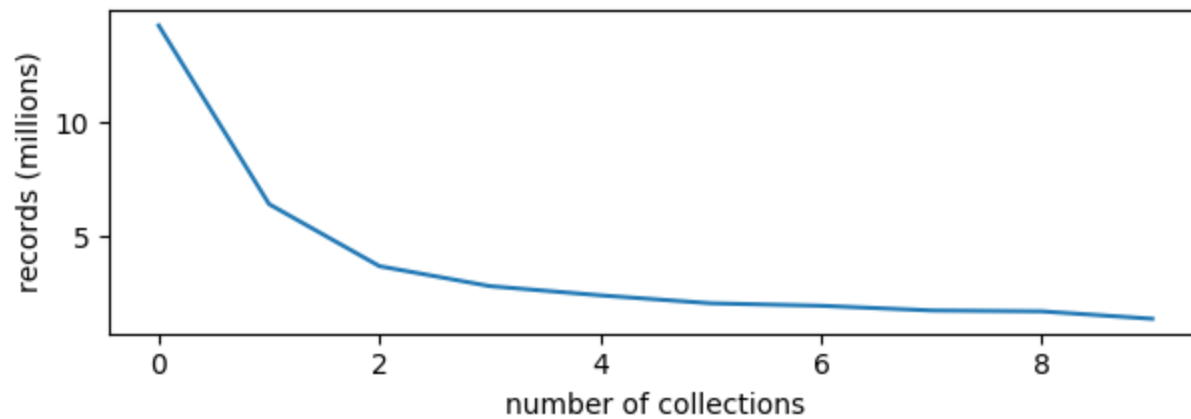
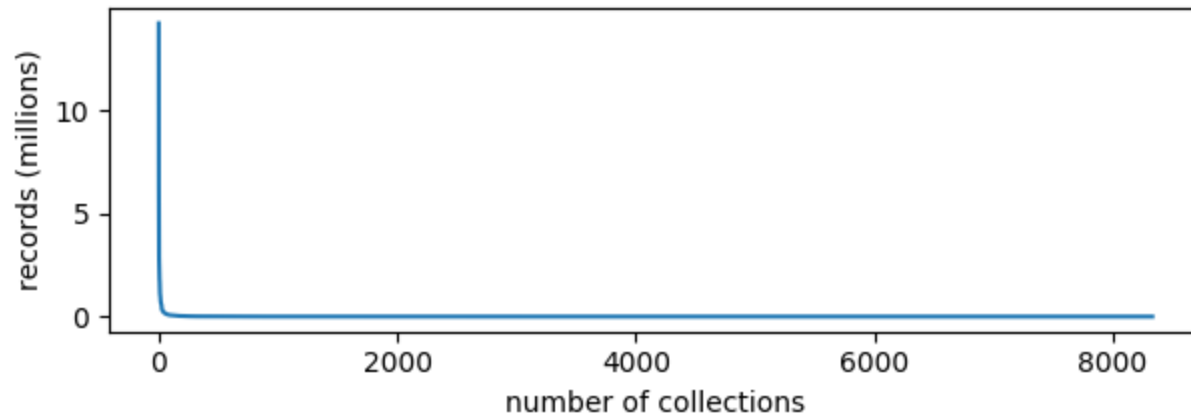
Article Metadata Index Sources (2017)



Around 118,221,121 articles in total. Only parts visible to libraries.

# Collection distribution

Crossref collection size distribution (2017)



# Diversity

- HTTP API (Crossref)
- FTP-Pull (Elsevier, IEEE)
- FTP-Push (WISO)
- Download (Springer)
- OAI-PMH (Arxiv, Thieme)
- Elasticsearch Backup (DOAJ)

## Formats

- XML
- JSON

# Gallery of raw data

- Examples from the data as provided.

```
<GENIOS Profile="manuel_IJAR" Dateissue="20170301T000311">
<Document ID="20051001" IDNAME="NO" DB="IJAR">
<Abstract>
This interview deals with the issue of ...
</Abstract>
<Authors>
<Author>Lola Cendales, Angelo Torres</Author>
</Authors>
<Descriptors><Descriptor>n.n.</Descriptor></Descriptors>
<Date>20050101</Date>
<Issue>1</Issue>
<ISSN>1862-1303</ISSN>
<ISBN>n.n.</ISBN>
<Subtitle>n.n.</Subtitle>
<Series-Title>n.n.</Series-Title>
<Editors><Editor>n.n.</Editor></Editors>
<Edition>n.n.</Edition>
<Language>n.n.</Language>
...
```

```
<article dtd-version="1.0" article-type="misc">
  <front>
    <journal-meta>
      <journal-id>14centyxqweeesdd</journal-id>
      <journal-id>j50000837</journal-id>
      <journal-title-group>
        <journal-title>
          14th Century English Newsletter
        </journal-title>
      </journal-title-group>
      <publisher>...
    </publisher>
    <issn pub-type="ppub">01375840</issn>
    </journal-meta>
    <article-meta>
      ...
    </article-meta>
  </front>
</article>
...
```



```
{
  "status": "ok",
  "message-type": "work-list",
  "message-version": "1.0.0",
  "message": {
    "items-per-page": 1000,
    "items": [
      {
        "indexed": {
          "date-parts": [
            [
              2016,
              1,
              1
            ]
          ],
          "date-time": "2016-01-01T06:10:02Z",
          "timestamp": 1451628602718
        },
        ....
      }
    ],
    ....
  }
}
```

```
{
  "sort": [
    "000011857dbc42afb0f1a8c7e35ab46f"
  ],
  "_type": "article",
  "_index": "doaj_v1",
  "_score": null,
  "_source": {
    "index": {
      "publisher": [
        "Press of International Journal of Ophthalmology"
      ],
      "schema_subject": [
        "LCC:Medicine",
        "LCC:Ophthalmology"
      ],
      "classification": [
        "Medicine",
        "Ophthalmology"
      ],
      ...
    }
  }
}
```

```
[
  {
    "shardLabel": "SLUB-dbod",
    "sourceID": "64",
    "megaCollection": "Perinorm - Datenbank Normen ...",
    "productISIL": null,
    "externalLinkToContentFile": null,
    "contentFileLabel": null,
    "contentFileURI": null,
    "linkToContentFile": null,
    "ISIL": "DE-105",
    "evaluateHoldingsFileForLibrary": "no",
    "holdingsFileLabel": null,
    "holdingsFileURI": null,
    "linkToHoldingsFile": null
  },
  ...
]
```

# Basic steps

- Synchronize data
- Normalize various formats
- Apply licencing information
- Postprocessing steps (e.g. deduplication)
- Create solr-importable format

# Synchronize data

- <https://lftp.yar.ru/> (FTP)
- <https://github.com/ubleipzig/metha> (OAI)
- Requests: HTTP for Humans
- `pyelasticsearch`

# Synchronize data

```
$ lftp -u xxxxx,xxxxx -e "  
set sftp:auto-confirm yes;  
set net:max-retries 5;  
set net:timeout 10;  
set mirror:parallel-directories 1;  
set ssl:verify-certificate no;  
set ftp:ssl-protect-data true;  
mirror --verbose=0 --only-newer -I '*' /  
/tmp/siskin-data/common/FTPMirror/b57115...;  
exit"  
ftp.ieee.org  
...
```

# Synchronize data

```
$ metha-sync http://www.intechopen.com/oai/
...
$ metha-cat http://www.intechopen.com/oai/
<Record>
  <header status="">
    <identifier>oai:intechopen.com:4119</identifier>
    <datestamp>2005-03-01</datestamp>
  </header>
  <metadata>
    <oai_dc:dc>
      <dc:title>
        Realization of a Service Robot for Cleaning ...
      </dc:title>
      <dc:creator>Jianwei Zhang</dc:creator>
      <dc:subject>
        International Journal of Advanced
        Robotic Systems
      </dc:subject>
      <dc:coverage>Volume 2</dc:coverage>
    ...
  </metadata>
</Record>
```

# Synchronize data

```
while True:
    params = {
        'rows': rows,
        'filter': filter,
        'cursor': cursor
    }

    url = 'http://api.crossref.org/works?%s' % (...)

    for attempt in range(1, self.attempts):
        if not cache.is_cached(url):
            time.sleep(self.sleep)
            body = cache.get(url)
```



# Normalization

- metafacture
- span

# Normalization

Metafacture: tool suite for metadata processing (German National Library)

Components: FLUX and MORPH.

```
// Example flux script.
```

```
fileName |  
open-file |  
decode-xml |  
handle-generic-xml("Record") |  
morph(FLUX_DIR + "morph.xml", *) |  
encode-json |  
write("stdout");
```

# Normalization

MORPH example:

```
<entity name="url[]" flushWith="record">
  <data source="metadata.dc.identifier.value">
    <regexp match="^(http.*)" format="${1}"/>
  </data>
</entity>
...
<data source="metadata.dc.type.value" name="rft.genre">
  <lookup in="genre_liste"/>
</data>
...
```

# Normalization

Everything normalized into a so-called *intermediate schema*.

```
{
  "finc.format": "ElectronicArticle",
  "finc.mega_collection": "DOAJ Directory of ...",
  "finc.record_id": "ai-28-000011857dbc42afb0f1a8c7e35ab46f",
  "finc.source_id": "28",
  "ris.type": "EJOUR",
  "rft.atitle": "Study progresses on continuous ...",
  "rft.genre": "article",
  "rft.issn": [
    "1672-5123"
  ],
  "rft.jtitle": "Guoji Yanke Zazhi",
  "rft.pages": "1737-1740",
  "rft.pub": [
    "Press of International Journal of Ophthalmology ..."
  ],
  ...
}
```

# Normalization

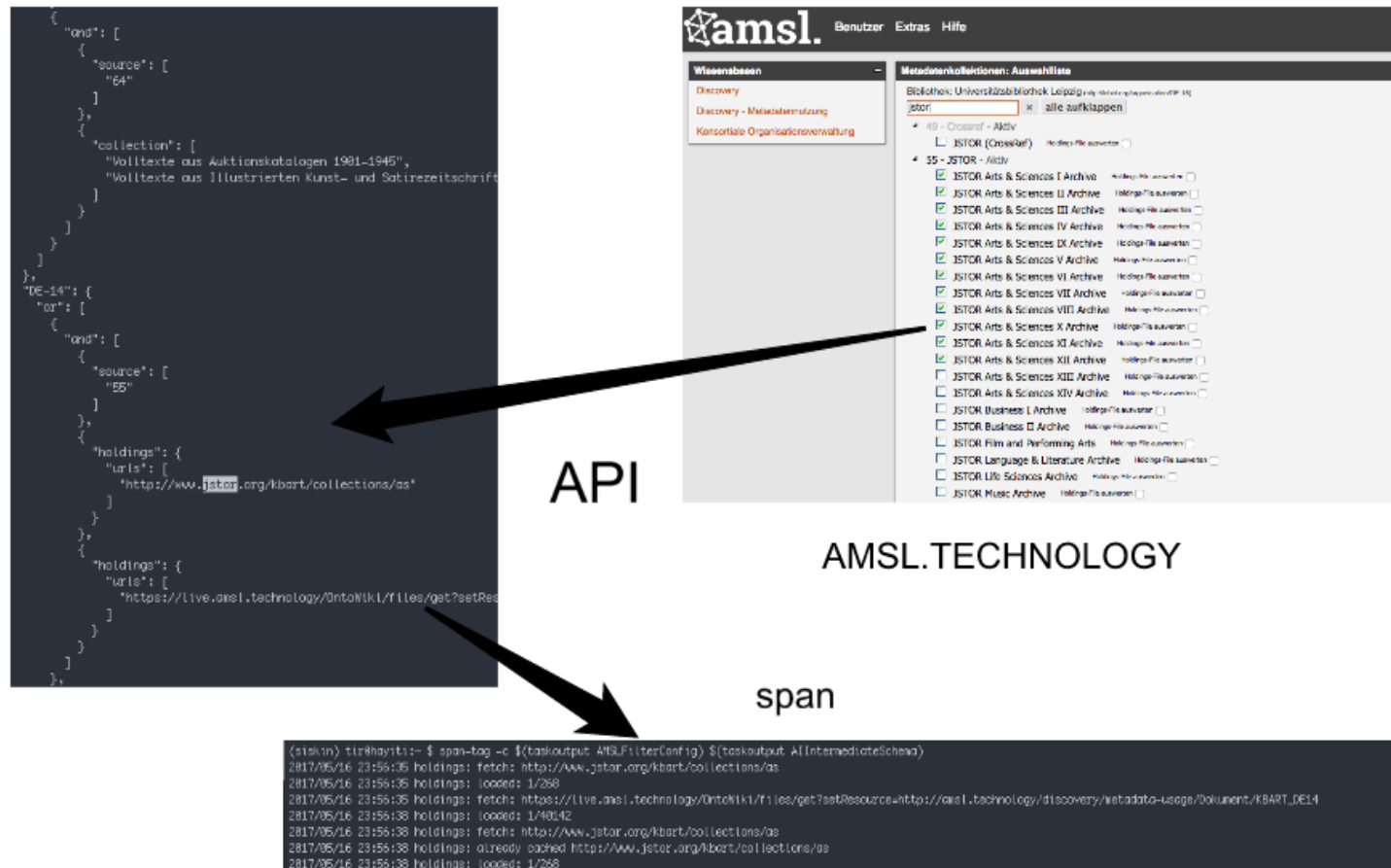
Usually, we put all data into single file, once normalized.

```
$ taskls AIIntermediateSchema --date 2017-05-08  
... 30G May 12 16:48 /.../AII.../date-2017-05-08.1dj.gz
```

Around 200G uncompressed. Includes abstracts, sometimes even full texts.

# Licensing

A complex piece.



# Licensing

A complex piece. But each library has fine-grained control over what is included in their catalog (sources, collections, licensing data, issn lists, ...)

# Postprocessing

Currently only DOI-based deduplication over all data sets. Using a custom command line [tool](#).

We also tested a few other things, e.g. include citation data from Microsoft Academic Graph.

Occasional automated quality checks – but not yet part of default workflow.



# Postprocessing

Deduplication via custom [tool](#) (similar to `uniq`):

```
$ cat x.csv
...
"ai-49-aHR0cDo...", "49", "10.1006/bulm.2002.0328", \
    "DE-14", "DE-Br1"
"dswarm-105-MT...", "105", "10.1006/bulm.2002.0328", \
    "DE-Mit1", "DE-14", "DE-Br1", "DE-520", "DE-15", \
    "DE-540", "DE-D275"
...

$ groupcover -prefs '85 55 89 60 50 105 49 28 48 121' < x.csv
```

# Last Steps

Convert data into something SOLR understands, then index the data.

Move data from a single file into SOLR fast with [solrbulk](#):

```
$ solrbulk -server localhost:8983/solr/biblio file.ldj  
...
```

```
[2017/05/13 21:52:17 [worker-6] @15117877  
[2017/05/13 21:52:17 [worker-4] @14941000  
[2017/05/13 21:52:17 120029017 docs in 5h41m24.697311325s at 5859.448 docs/s with 8 workers  
[2017/05/13 21:52:37 final commit: 200 OK  
|  
|real    341m44.788s  
|user    74m15.708s  
|sys     10m16.548s
```

# Last Steps

We use a another key-value store, to keep the intermediate schema files. This is not strictly necessary, but might increase performance, as SOLR index is a bit smaller.

- used [memcachedb](#), switched to [microblob](#)

# Orchestration

Many different (usually small) tasks: Sync this FTP server, convert to this format, apply some licensing, do deduplication, export to some other format.

# Orchestration

- How to document these?

# Orchestration

We use a dedicated orchestration framework written in Python. It documents the workflows and makes them executable.

- <https://github.com/spotify/luigi>

Luigi is a Python module that helps you build complex pipelines of batch jobs. It handles dependency resolution, workflow management, visualization etc. It also comes with Hadoop support built in.

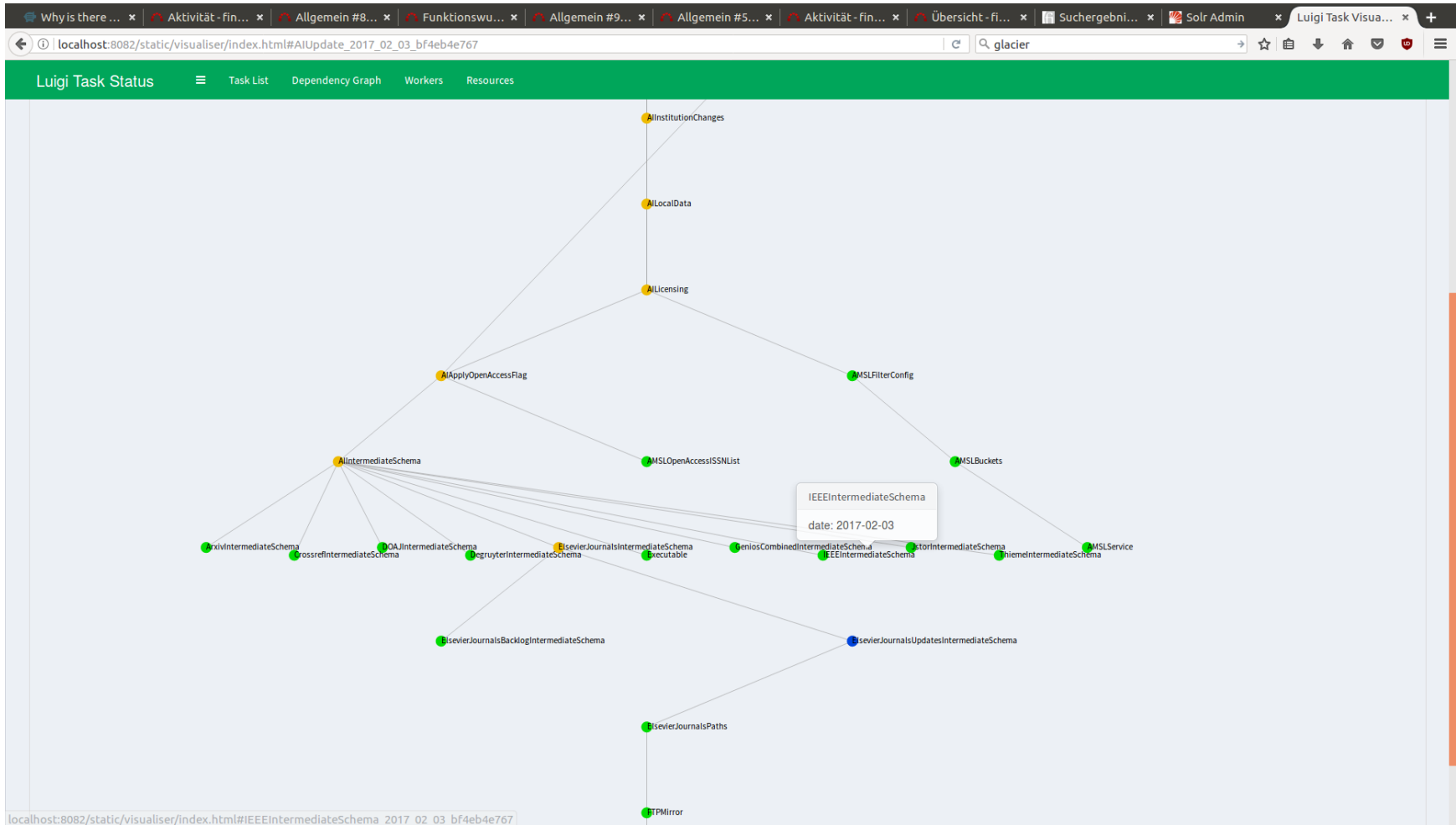
Uses a dependency graph (DAG).

# Orchestration

Dependencies (analogy):

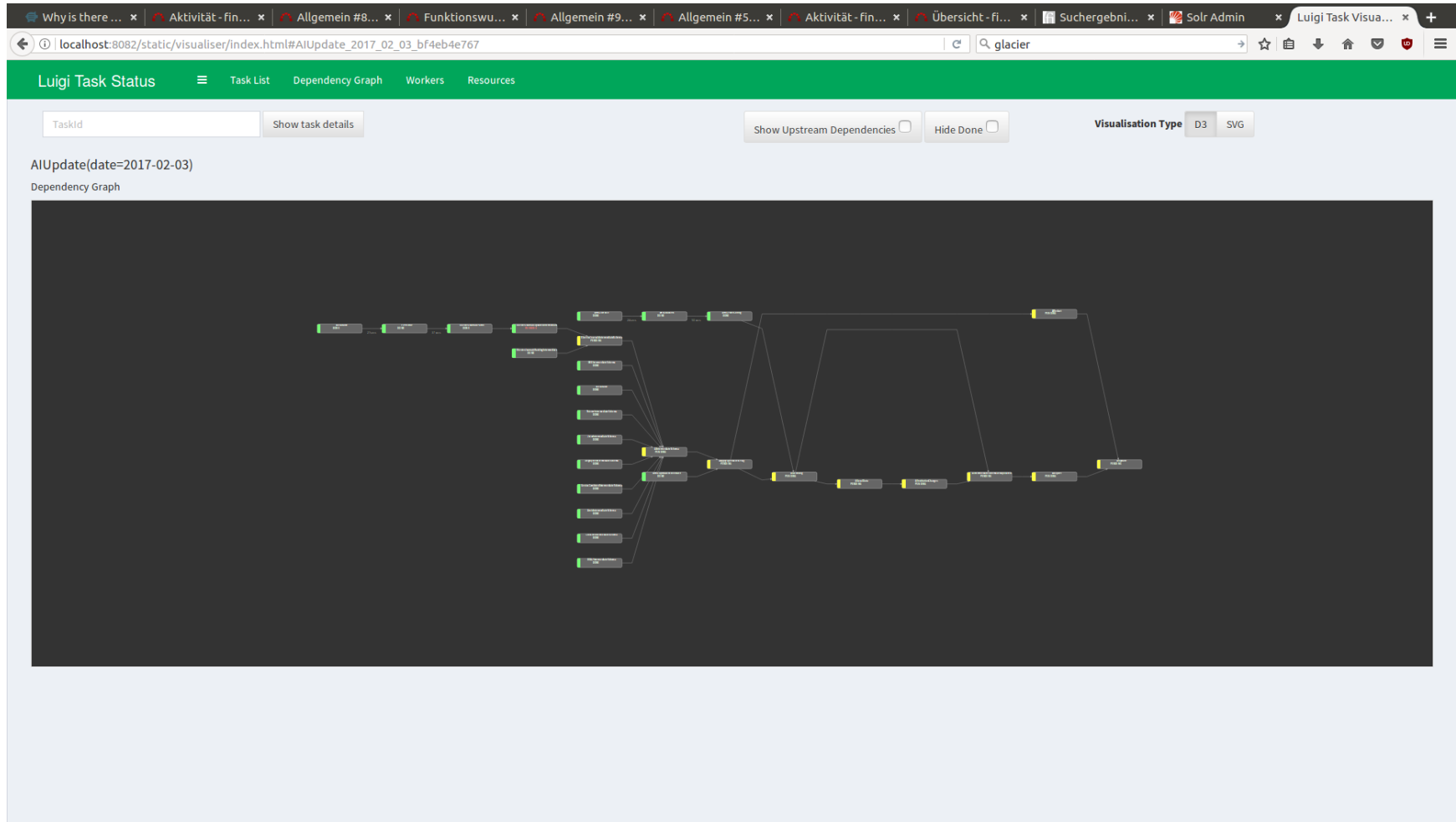
- Pizza

# Orchestration

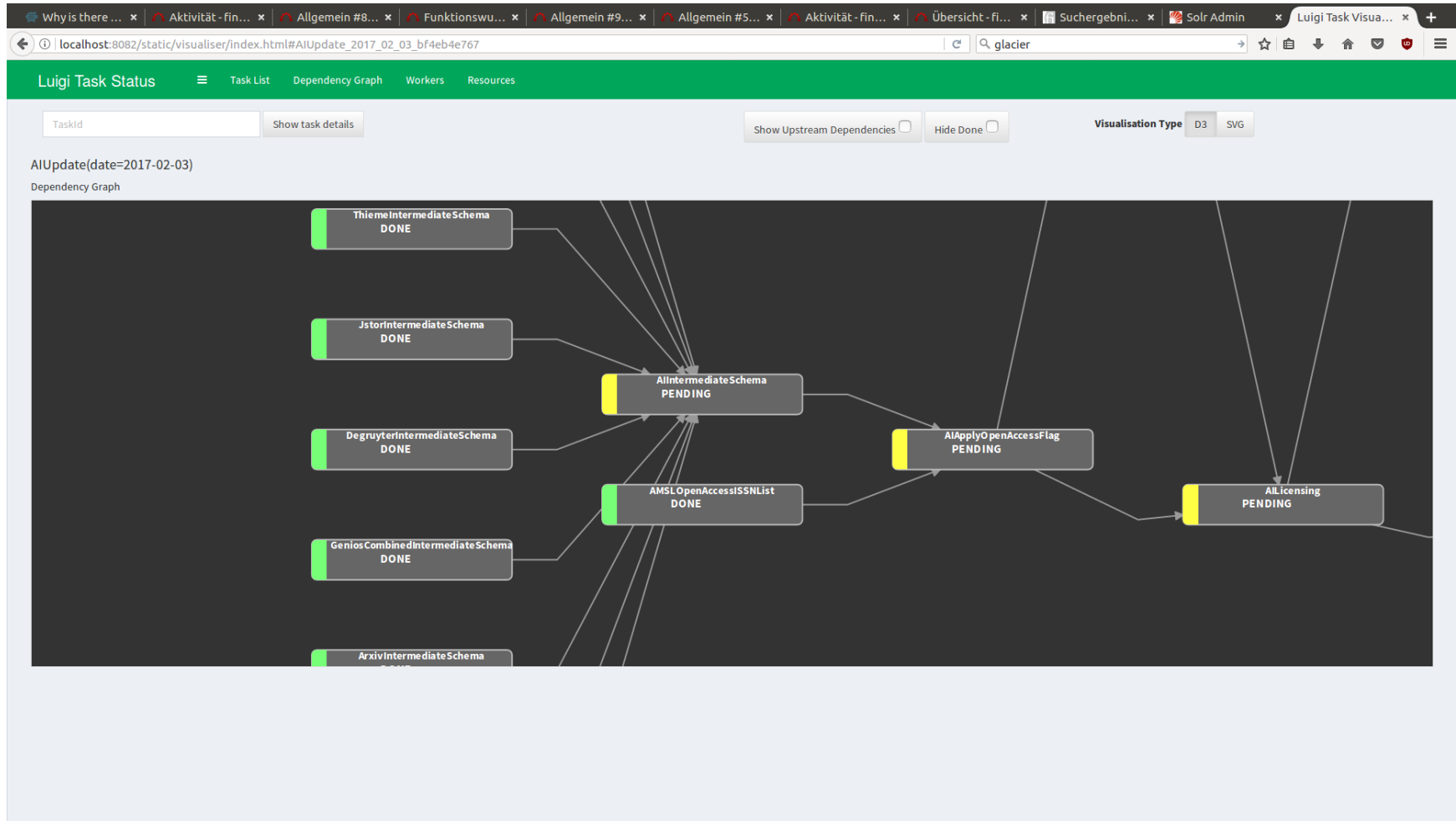




# Orchestration



# Orchestration



# Orchestration

```
Terminal x czygan@pre_Ai:- x czygan@ai-blob2:- x czygan@ai-blob2:- x +
Every 2,0s: tasksps Fri Feb 24 13:34:45 2017

>> localhost:8082, 2017-02-24 13:34:45.453272

RUNNING (4)
-----
r * FTPMirror
r * FTPMirror
r * FTPMirror
r * ThiemeCombine_2017_02_24_tm_journalarticles_992bf29c27

PENDING (26)
-----
p * AIApplyOpenAccessFlag_2017_02_24_342defe6d0
p * AIExport_2017_02_24_solrsvu3_62ae7b767c
p * AIInstitutionChanges_2017_02_24_342defe6d0
p * AIIntermediateSchemaDeduplicated_2017_02_24_342defe6d0
p * AIIntermediateSchema_2017_02_24_342defe6d0
p * AILicensing_2017_02_24_342defe6d0
p * AILocalData_2017_02_24_342defe6d0
p * AIRedact_2017_02_24_342defe6d0
p * AIUpdate_2017_02_24_342defe6d0
p * AMSLBuckets_2017_02_24_UBL_ai_5698e87fe9
p * AMSLFilterConfig_2017_02_24_UBL_ai_5698e87fe9
p * AMSLServices_2017_02_24_outboundServices_819f56d258
p * ArxivCombine_2017_02_24_342defe6d0
p * ArxivIntermediateSchema_2017_02_24_342defe6d0
p * ElsevierJournalsIntermediateSchema_2017_02_24_342defe6d0
p * ElsevierJournalsPaths_2017_02_24_342defe6d0
p * ElsevierJournalsUpdatesIntermediateSchema_2017_02_24_342defe6d0
p * IEEEIntermediateSchema_2017_02_24_342defe6d0
p * IEEEPaths_2017_02_24_342defe6d0
p * IEEEUpdatesIntermediateSchema_2017_02_24_342defe6d0
p * JstorIntermediateSchema_2017_02_24_342defe6d0
p * JstorLatestMembers_2017_02_24_342defe6d0
p * JstorMembers_2017_02_24_342defe6d0
p * JstorPaths_2017_02_24_342defe6d0
p * JstorXML_2017_02_24_342defe6d0
p * ThiemeIntermediateSchema_2017_02_24_journalarticles_48322b1169


DONE (11)
-----
d * AMSLOpenAccessISSNList_2017_02_24_342defe6d0
d * CrossrefIntermediateSchema_2006_01_01_2017_02_24_9c410bb3e4
d * DOAJIntermediateSchema_2017_02_24_342defe6d0
d * DegruyterIntermediateSchema_2017_02_24_342defe6d0
d * ElsevierJournalsBacklogIntermediateSchema_99914b932b
d * Executable_http__git_io_v1_span_import_92204838d2
d * Executable_http__lftp_yar_lftp_0e52caac58
d * Executable_http__zlib_net_pigz_500b032025
d * Executable_https__github_c_metha_sync_122b54c67f
d * GeniosCombinedIntermediateSchema_2017_02_24_342defe6d0
d * IEEEBacklogIntermediateSchema_99914b932b
```

# Orchestration

- workflows are **extensible**
- all tasks executable via command line
- distributed as a normal **python package**

# Development

- open source
- with git (and github)
- each data source has a file containing *recipes*

 miku pqdt: attach isils and export, refs #10495

..

 <a href="#">__init__.py</a>	pylint checks
 <a href="#">amsl.py</a>	amsl: fix field name
 <a href="#">arxiv.py</a>	amsl: add support for source stamping via --stamp, refs. #9886
 <a href="#">base.py</a>	base: add TODO
 <a href="#">crossref.py</a>	crossref: code style fixes
 <a href="#">datacite.py</a>	apply most pylint suggestions
 <a href="#">dawson.py</a>	dawson: update docs
 <a href="#">dblp.py</a>	dblp: style fix
 <a href="#">degruyter.py</a>	degruyter: allow long lines
 <a href="#">doaj.py</a>	unify import formatting with isort -rc --atomic .
 <a href="#">dummy.py</a>	apply most pylint suggestions
 <a href="#">elsevierjournals.py</a>	apply most pylint suggestions
 <a href="#">gbi.py</a>	genios: use sid as tag
 <a href="#">genios.py</a>	genios: possibly, maybe, #9534
 <a href="#">highwire.py</a>	highwire: add missing tag
 <a href="#">ieee.py</a>	ieee: add IEEEDOIList
 <a href="#">ijoc.py</a>	ijoc: add missing -with-fullrecord
 <a href="#">izi.py</a>	izi: add missing -with-fullrecord
 <a href="#">jstor.py</a>	apply most pylint suggestions
 <a href="#">mag.py</a>	mag: update docs

# Development

- can create or modify new sources independent of existing workflows
- no central database
- strive for reproducibility
- run on various OS (different Linux distributions, Mac OS X)

# Development

- Why [Python](#)? It's [popular](#) and has wide range of application in the scientific domain.
- Why [Metafacture](#)? A comprehensive and declarative tool.
- Why UNIX? 40 plus years of [text processing](#).
- Why [Go](#)? It's fast and easy to deploy.



# Outlook

- more (new) sources
- migrating existing data sources into the DAG
- automated quality checks
- add more external information (e.g. citation graph) to the metadata

# More

- <https://github.com/ubleipzig>
- <https://github.com/finc>