Archiving workflow for provenance

CLARIAH - Provenance workshop

03-09-2018

Jerry de Vries, Vyacheslav Tykhonov, Emil Bode, Dirk Roorda
Agenda

• Introduction

• Archiving workflow for provenance information at DANS
  • Collection side
  • Research side

• Examples

• Questions
Introduction

• Why an archiving workflow for provenance?

• Provenance important for reproducible research

• Workflow important for each research project internally/externally (NARCIS, Freya, PARTHENOS)
Provenance workflow on collection side

- Metadata harvesting from NARCIS (R-function)
- Prepare for data analysis
  - Transform raw data to own format (store mongoDB)
  - Standardize data
  - Keep track of original metadata harvest as reference
- Analysis
  - Try to enrich metadata by finding links (linking with external sources)
  - Store results (now local)
- Publish data and provenance (R-code, git commit, etc)
Provenance workflow on research side

- Getting data files (metadata + provenance information from data repository)
- Loading files into local storage
- Processing of data files
  - Statistical exploration of the NARCIS metadata
- Produce new dataset files
- Updating provenance with information about versions of processing tools
- Storing dataset files including updated provenance information in github
Keeping provenance information research side

Table of Contents
- 1. Hello NARCIS
- 2. Preparation
  - 2.1. Config
  - 2.2. Start Docker
  - 2.3. Connect to MongoDB
- 3. Key value generation
  - 3.1. Load data
    - 4.1. Top level fields
  - 5. Inventory of keys
    - 6.1. Nested keys
  - 6. Results
- 7. Appendix
  - 7.1. Use variety
  - 7.2. Preparation
  - 7.3. Run variety
  - 7.4. Results

Hello NARCIS

We explore a dump of the NARCIS data. The dump has been taken by Emil Bode in December 2017. He has prepared it as a MongoDB database contents in a Docker container. This container is obtainable from DataVerse: NARCIS metadata in rdditnorm format. Next to the dump is a readme that tells you exactly how to query the data locally.

The purpose of this notebook is an initial scan of the full database contents. The database is called NARCIS, and in it is just one collection, dce2017, with over 1.5 million documents.

A document is a dictionary of keys and values, where values may be numbers, strings, dates, but also lists of values and also dictionaries of keys and values. The nesting of lists and dictionaries can be arbitrarily deep.

Documents do not have to conform to any schema. Most likely, there are a few dominant schemata, but also likely; there might be outliers.

We assume no previous knowledge of NARCIS, so we want to explore the data from scratch.

This notebook performs a first step:

Start Docker

Below are the magic commands to start and stop the relevant Docker container. They are shell commands, not Python commands.

In case you want to stop Docker

```
In [3]:
  
  docker stop NARCIS
  docker rm NARCIS

NARCIS

In the next cell a Docker container is started.
```

```
In [4]:
  
  docker run --name NARCIS --v (DUMP_DIR)/data/db -p 27019:27017 -d mongo --logpath /data/db/log.1

0e5307c30296a13fe7905dcbb0ae0425d4858a6a4ce08376f5516b9d491b2e1e8
```

Connect to MongoDB

Now there is a MongoDB behind port 27019 that we can connect to.

We make the connection and get some very basic statistics about the contents of the database.

```
In [5]:
  
  client = MongoClient('mongodb://localhost:27019/')
  client.database_names()

Out[5]:

['NARCIS', 'admin', 'config', 'local']
```

Navigate to the Dce2017 collection

```
In [6]:
  
  DB = client.NARCIS
  print(DB.collection_names())
  DB = DB.Dce2017
  DB.count()

Out[6]:

['Dce2017']
```

Key value generation

We want to treat a document as a generator of key-value pairs, where keys are hierarchical keys and values are scalar values. We convert scalar values to strings, and escape newlines.

We then write the key value pairs to file. If we just write them naively as tuples (_id, bkey, value) where bkey is the hierarchical key within the document, we end up with a file of SGB.
Archiving data and tools

Goal: Reproducible research with possibility to execute the same analysis to obtain the same result

• Archiving dataset consisting data and software
  • Software archived as docker images with provenance information
  • Data files have different revisions and can be attached as volumes to docker images

Result

• Researchers can go to any dataset revision applying the same tool
Docker image with provenance

LOD dataset: Docker image of Virtuoso and its database with data.

Subject
Computer and Information Science

Notes
Docker image can be restored from file virt-image-arc.tar.gz:

```
$ docker load < virt-image-arc.tar.gz
```

You should see something like that:

```
036fbc493384: Loading layer
[==================================================================================================] 9.216kB/9.216kB
```

Loaded image ID:

```
sha256:b25b7c7c1a5411d1ae31be86c442c6d674cb7546d4b05ff722c188419dc642e
```

Run docker images to find IMAGE_ID

```
$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
b25b7c7c1a54 4 hours ago 606MB
```

Copy string below IMAGE_ID and use as Docker container identification.

Unpack db-virtuoso.tar.gz in the current folder and start container:
Archiving tools and data with Docker

$ docker logs my-virtuoso
should show something like:
15:18:16 HTTP/WebDAV server online at 8890
15:18:16 Server online at 1111 (pid 1)

If you will open http://localhost:8890/sparql you’ll get access to SPARQL endpoint ready to query dataset.

$ docker exec -i --user root -t my-virtuoso /bin/bash
to access running container
Questions?