Leibniz-Institut für
Astrophysik Potsdam

Behind the scenes of AIP IVOA data services

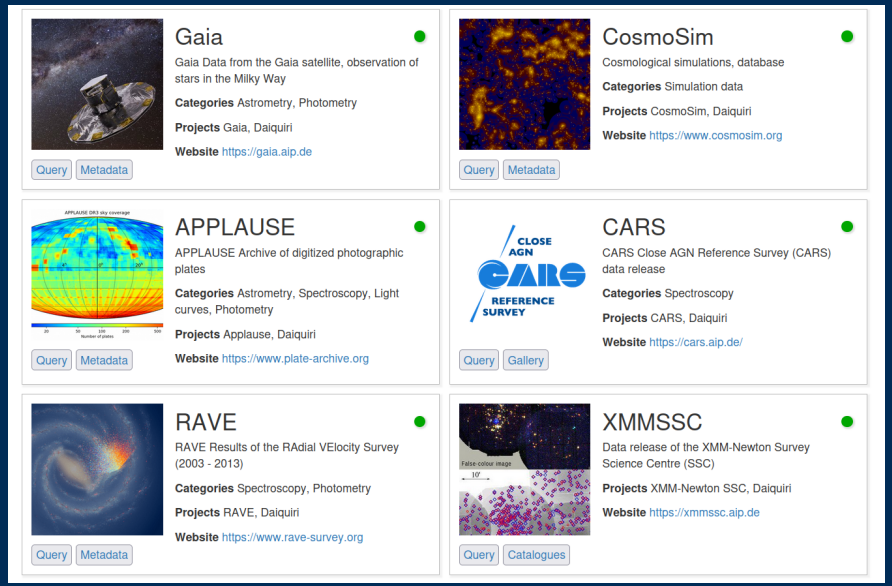
Providing DOI within the Datalink protocol and operations report - Django Daiquiri and dockerized DevOp environment

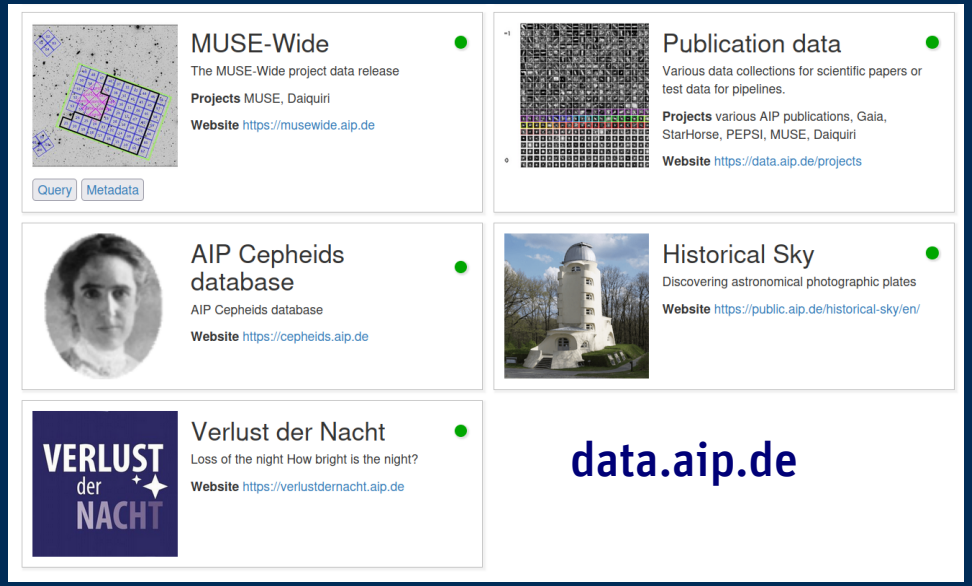
A. Galkin, K. Makan / IVOA Interop, Sydney, Australia / May 2024

Outline - Behind the scenes of AIP IVOA data services

- IVOA compliant data archives at AIP
- Providing DOI within the Datalink protocol
- Operations report – Django Daiquiri and Dockerized DevOp environment

Data services at AIP – data.aip.de





Data services at AIP powered by Django Daiquiri framework

IVOA interfaces

- ADQL query interface
- TAP protocol
- IVOA Simple Cone Search
- VO endpoints for IVOA registry of registries
- Datalink protocol

Supports further interfaces:

- Metadata management backend incl. DOI and UCD
- OAI-PMH2 endpoint

As providers, we have to handle metadata for different protocols, each of the metadata schemas have diverse structures. Our goal is to re-use the information already given in the archive and to avoid duplicating it.

Providing DOI within the Datalink protocol

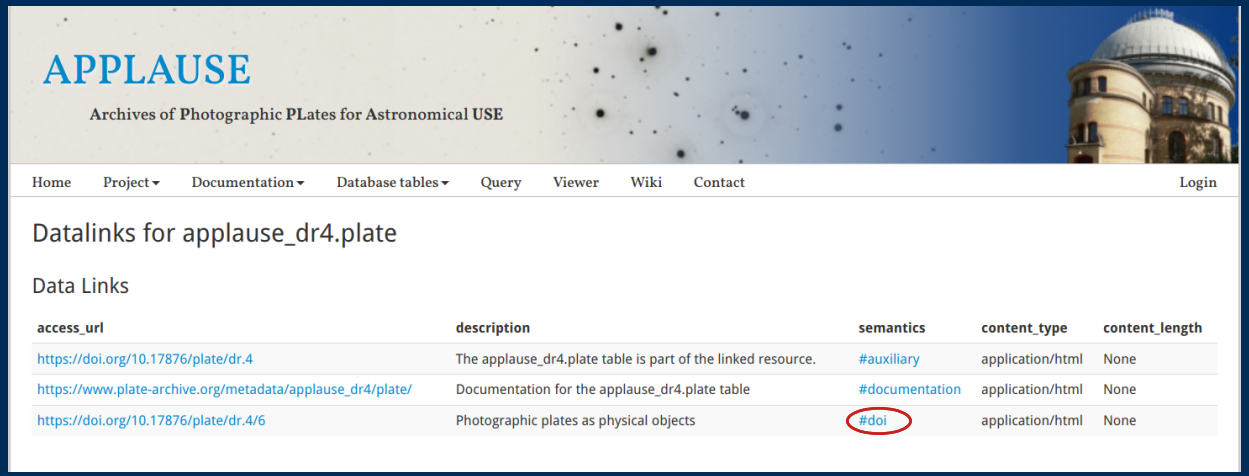
An example from the APPLAUSE archive (H. Enke et al. 2024)

- DOIs exist for schema and tables.
- Furthermore, DOIs exist for objects such as plates, previews, envelopes, logbooks.
- Instead of providing DOI via the science data base or an individual solution, we store the DOI information in the Datalink table.
- DOI was added as extension to the Datalink-core dictionary as DOI is not (yet?) part of the Datalink-core dictionary.

If a DOI is present, the Datalink tables are generated from already existing metadata either per script or on the fly (dynamic Datalink).

- Daiquiri framework automatically generates basic Datalink entries for schemas and tables which are present in each archive from the existing metadata.
- For other objects, our archives expect the DOI to be included directly in the Datalink tables.

Datalink Viewer

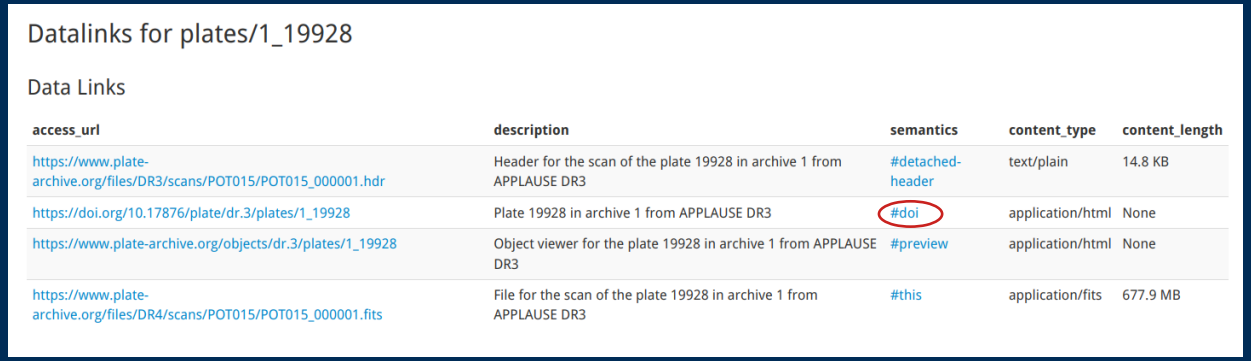


https://www.plate-archive.org/datalink/applause_dr4.plate/

applause_dr4.plate table

https://www.plate-archive.org/datalink/plates/1_19928/

Plate 1_19928 object



https://www.plate-archive.org/datalink/semantics#doi

DOI Datalink dictionary extension

Use of Datalink tables in OAI PMH protocol

The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) is a low-barrier mechanism for repository interoperability.

- DOI and the DOI landing page information come from the Datalink table
- Files for the object in OAI protocol also come from Datalink table
- OAI interface uses the „alternate identifier“ tag for an object to use with Datalink protocol

Plate 1_19928 object

```
▼<alternateIdentifiers>
   <alternateIdentifier alternateIdentifierType="datalink">plates/1_19928</alternateIdentifier>
   <alternateIdentifier alternateIdentifierType="DOI Landing Page">https://www.plate-archive.org/objects/dr.3/plates/1_19928</alternateIdentifier>
 </alternateIdentifiers>
```

https://www.plate-archive.org/oai/?verb=GetRecord&metadataPrefix=datacite&identifier=oai:www.plate-archive.org:plates/1_19928

Building a dockerized DevOp environment - from development to operation

Development on local machine → Development on a dev server:

- test data
- storage

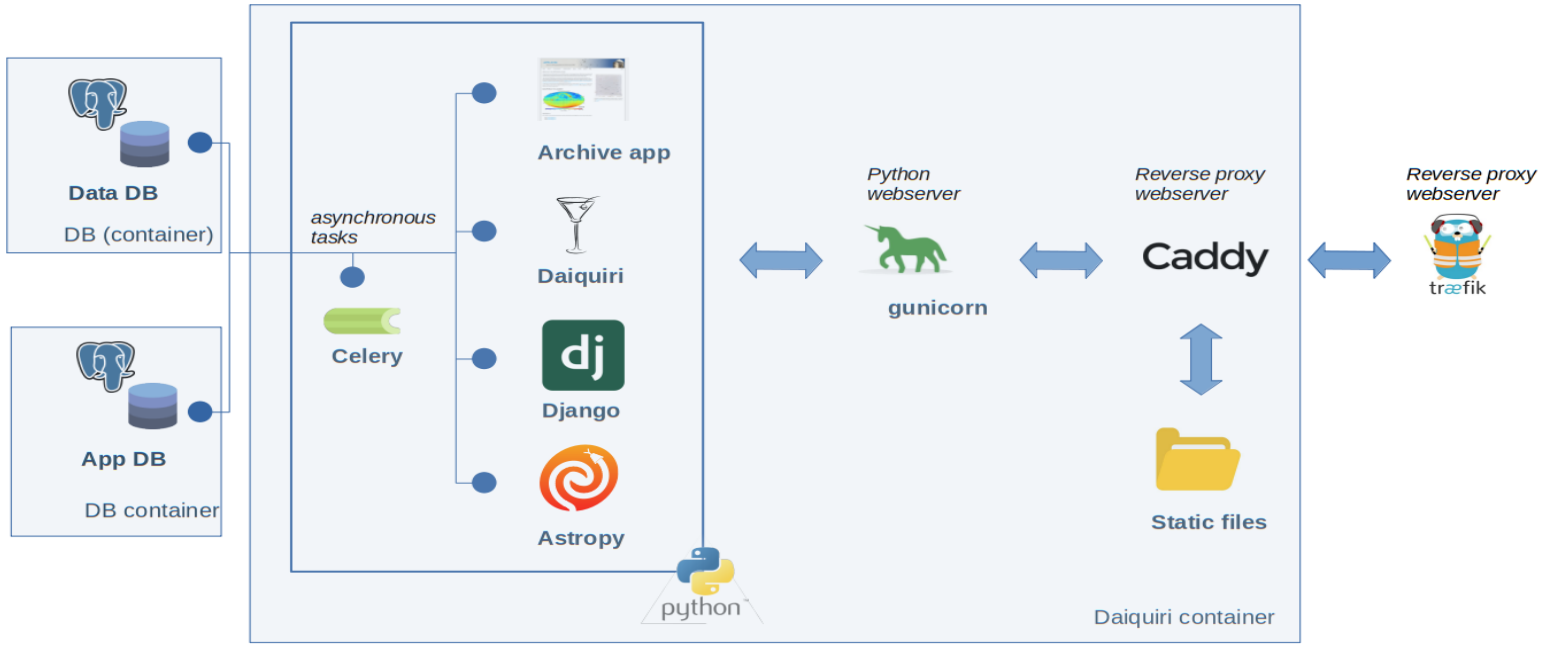
→ Test on the test instance:

- identical to the prod instance plus the new feature
- test data
- storage

→ Deployment to the productive instance

Ideally, a developer has the exact same environment as on the production instance.

Building a dockerized DevOp environment - an archive instance



The highs…

- The setup is reproducible. So are the bugs.
- Development, test and deployment setups are the same environment.
- Less downtime
- No more „changing configs“ on the productive instances.
- As many as dev or test instances as needed.
- Additional features for maintanance can be built into the setup – data volumes, custom installs for the app, cron jobs, etc.

Docker is a very fast developing technology – continuous updates are needed, more difficult to deploy on older OS. Same goes for web services.

With docker any updates or combinations of packages and features are possible and can be tested beforehand, as many dev or test instances as needed.

… and the lows ;)

Containerization does not mean simplicity. A lot of expertise is needed, but it is completely scripted / documented now.

Questions?

Anastasia Galkin

agalkin@aip.de

•

github.com/django-daiquiri