NASA astro VO in the Cloud:

What we think we need and a proposed solution

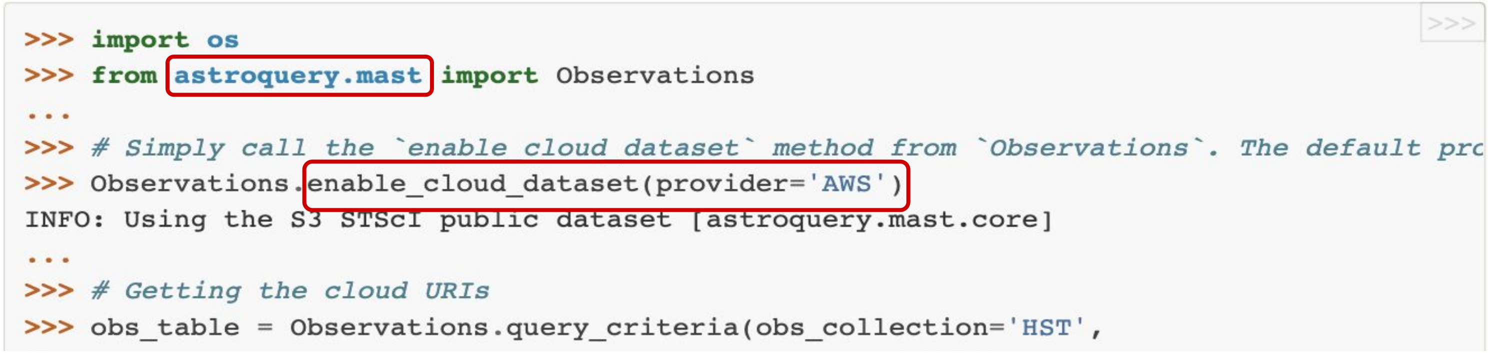
Tess Jaffe
on behalf of
HEASARC, IRSA, and MAST

A use case for VO in the cloud:

- MAST hosts images both on prem and in an AWS S3 bucket
- MAST’s SIA service currently returns an access URL pointing to the on prem version.
- User is working on AWS and wants to access the data from S3.
- We should make this easy and transparent.

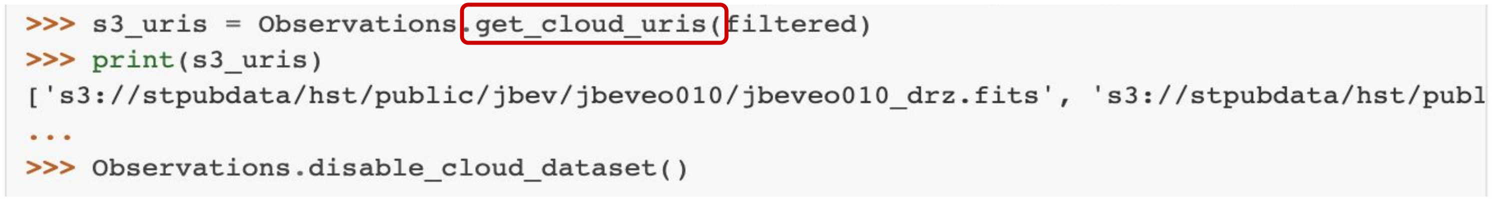
Current solution for MAST

- Separate MAST service specifically to serve S3 addresses
- Python client astroquery.mast module custom made for MAST



```
>>> import os
>>> from astroquery.mast import Observations
...
>>> # Simply call the `enable_cloud_dataset` method from `Observations`. The default pro
>>> Observations.enable_cloud_dataset(provider='AWS')
INFO: Using the S3 STScI public dataset [astroquery.mast.core]
...
>>> # Getting the cloud URIs
>>> obs_table = Observations.query_criteria(obs_collection='HST',
```

<... snip ...>



```
>>> s3_uris = Observations.get_cloud_uris(filtered)
>>> print(s3_uris)
['s3://stpubdata/hst/public/jbev/jbeveo010/jbeveo010_drz.fits', 's3://stpubdata/hst/publ
...
>>> Observations.disable_cloud_dataset()
```

Prototype VO-compatible solution

- SIA service returns extra information in the VOTable.
- Clients need to read the extra data and fetch from cloud where appropriate.

E.g., in pyvo, provide a utility to fetch the data product from AWS instead of on prem:

```
import pyvo
query_url = “https://mast.stsci.edu/portal_vo/Mashup/VoQuery.asmx/SiaV1?MISSION=HST&”
results = pyvo.dal.sia.search(query_url, pos=pos, size=0.0)
pyvo.utils.download_file( results[0], ‘aws’)
```

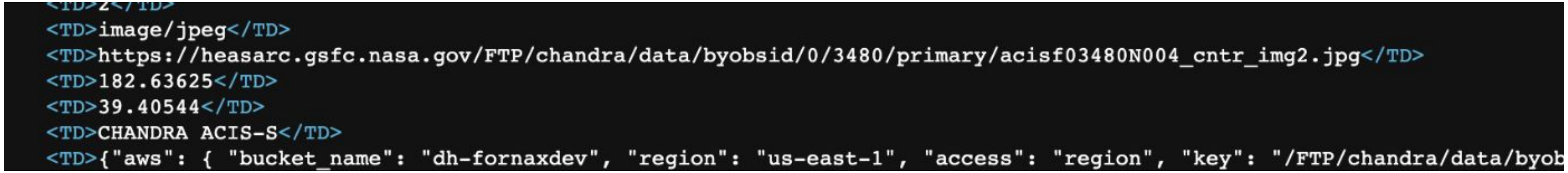
(See pyvo PR #369)

Proposed implementation: service side

1. Return `cloud_access` column in VOTable
   a. JSON content
   b. Dictionary of (list of) dictionaries for each cloud access possibility
   c. E.g.,

```
{“aws”: [{ “bucket_name”: “stpubdata”,
          “region”: “us-east-1”,
           “access” : “region”,
           “key” : “hst/foo/bar/to/image/file.fits”},
           { “bucket_name”: “anotherAWSbucket”, … }],
 “google”: {...},
 “azure”: {...}
}
```

(Try it at our test service.)



```
<TD>2</TD>
<TD>image/jpeg</TD>
<TD>https://heasarc.gsfc.nasa.gov/FTP/chandra/data/byobsid/0/3480/primary/acisf03480N004_cntr_img2.jpg</TD>
<TD>182.63625</TD>
<TD>39.40544</TD>
<TD>CHANDRA ACIS-S</TD>
<TD>{"aws": { "bucket_name": "dh-fornaxdev", "region": "us-east-1", "access": "region", "key": "/FTP/chandra/data/byob
```

Proposed implementation: client side

2. Clients can look for `cloud_access` information in addition to, e.g., `access_url`.

   a. In Python, we suggest a user friendly utility:

   ```
   pyvo.utils.download_file(record,’aws’)
   ```

   which under the hood selects between

   ```
   astropy.utils.data.download_file(record[‘access_url’])
   ```

   and

   ```
   boto3.client('s3').download_file(
        record[‘cloud_access’][‘aws’][‘bucket’],
       record[‘cloud_access’][‘aws’][‘path’],
        outfile)
   ```

   depending on the `cloud_access` information and the availability of the object.

   b. Likewise for other languages, clients, or user-written code.

Questions for IVOA

- Service issues:
    - Extra columns in any service that gives you a URL to a file?
        - Alternative ideas?
    - JSON content of extra column or other?
        - Is JSON in an XML element going to be a problem?
        - <![CDATA[...json…]]> ?
    - Integrate with DataLink?
- Python-specific issues:
    - where to put the client-side utility?
- What aren’t we thinking about?
- What are others doing or thinking about?