HEASARC • IRSA
NED • MAST

NASA Astronomical Virtual Observatories

NAVO

NASA astro VO in the Cloud:

What we think we need and a proposed solution

Tess Jaffe
on behalf of
NASA's astrophysics archives: HEASARC, IRSA, and MAST
and
NASA's Astronomical Virtual Observatories (NAVO)

The Fornax project:

- HEASARC (High Energy Science Archive Research Center, NASA Goddard)
- IRSA (InfraRed Science Archive, Caltech)
- MAST (Mikulski Archive for Space Telescopes, Space Telescope Science Institute)
- NASA's CISTO (Goddard) and Navteca (contractor)

Collaborators:

- Other NASA science domains (Heliophysics, Earth Sciences, Planetary,...),
- SciServer @ Johns Hopkins

What we are building is not a domain-specific platform, though development is driven partly by specific NASA astrophysics use cases.

Science use cases: some examples

- Science goal: Identify periodic variable stars

  Technical Challenge: Mine enormous NEOWISE time series data (i.e., just "big data")

- Science goal: phenomenology of chaotic systems across decades of data

  Technical Challenge: extract spectra from every observation using mission- and HEA-specific tools, pass to ML algorithms running on HPC ("big software").

- Science goal: A host of demographic studies of galaxies

  Technical Challenge: Combining multi-wavelength images spanning a large range of resolution using forced photometry methods ("big data" plus "big software" plus "big variety").

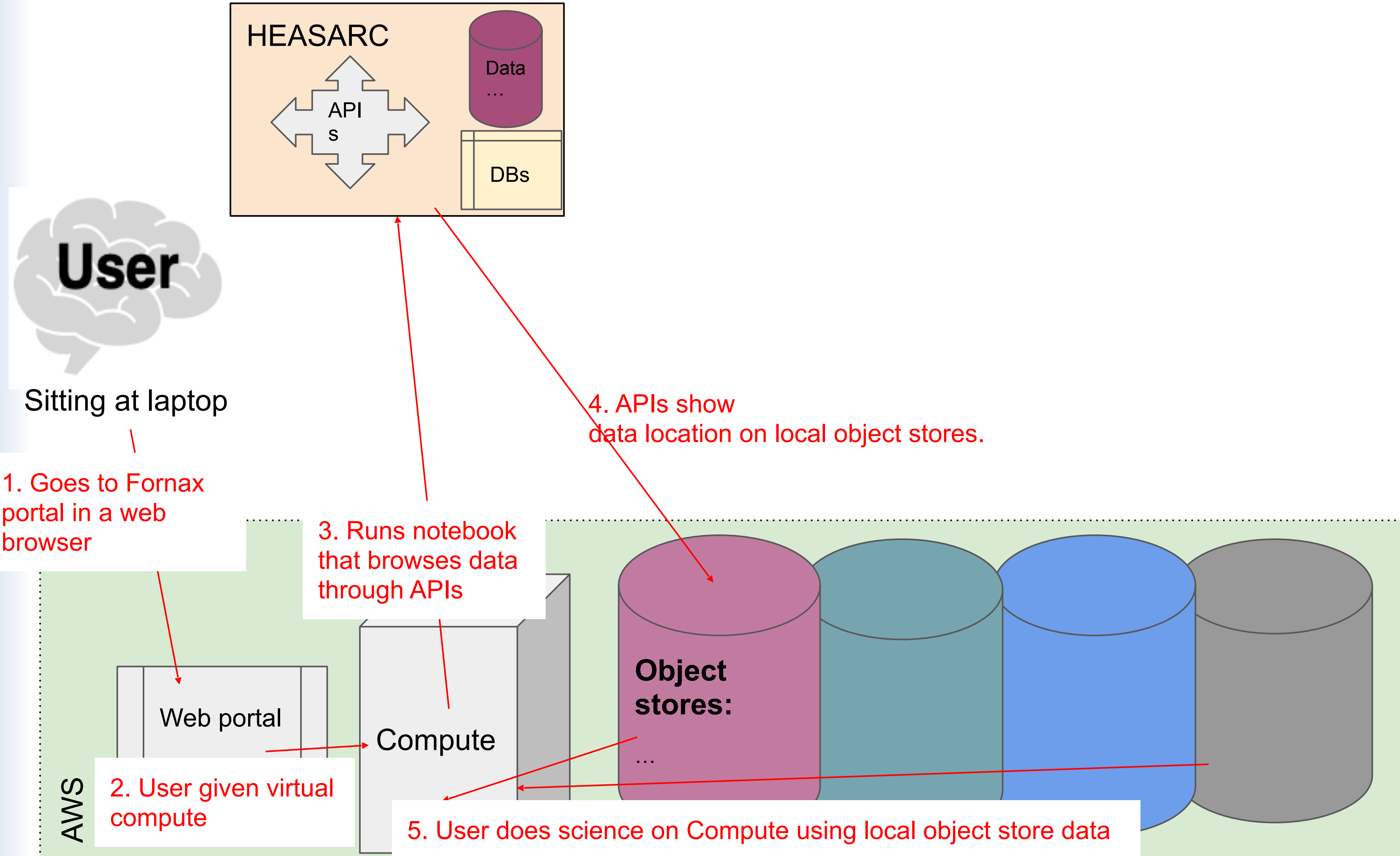
- **Inclusion goal: anybody anywhere can create an account, launch a container, open a tutorial notebook, and be doing analysis in 10 minutes.**

  Technical Challenge: security and cost controls (i.e., "big community")

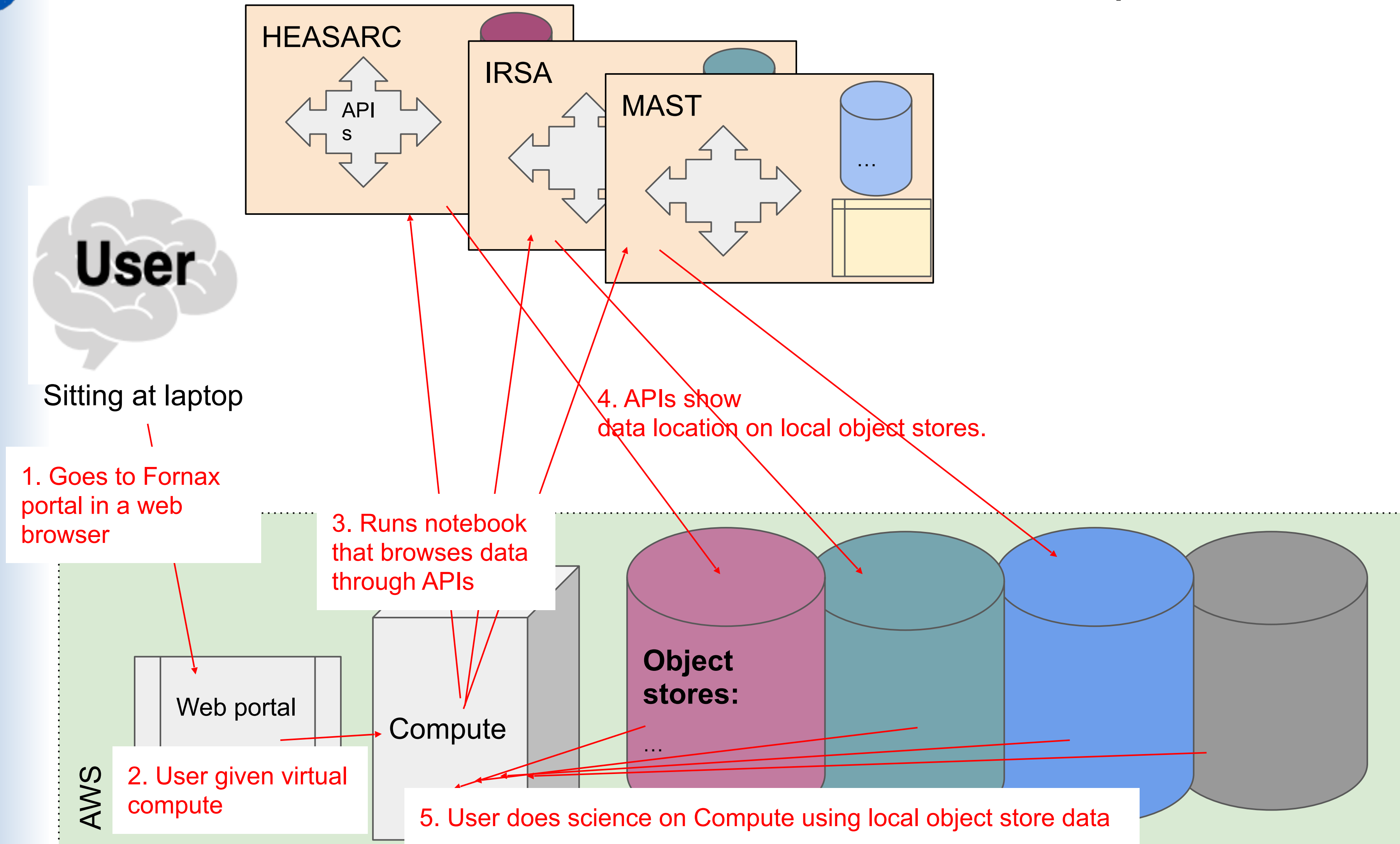
What is a “science platform”? No idea, but Fornax is:

- Fairly **generic**, not domain specific, because meant for *all NASA astrophysics users*.
    - Also developed in collaboration with NASA heliophysics, planetary, earth sciences, etc.
- To most of our target users: JupyterLab, Linux terminal, and Python with shared collaboration spaces.
    - We are not trying to anticipate what science they’ll want to do.
- Infrastructure components that will be part of the open source infrastructure-as-code
    - Cloud-agnostic where possible.
- Specific user interface apps can be deployed on top.

Future use case flow on Fornax cloud science platform



Future use case flow on Fornax cloud science platform

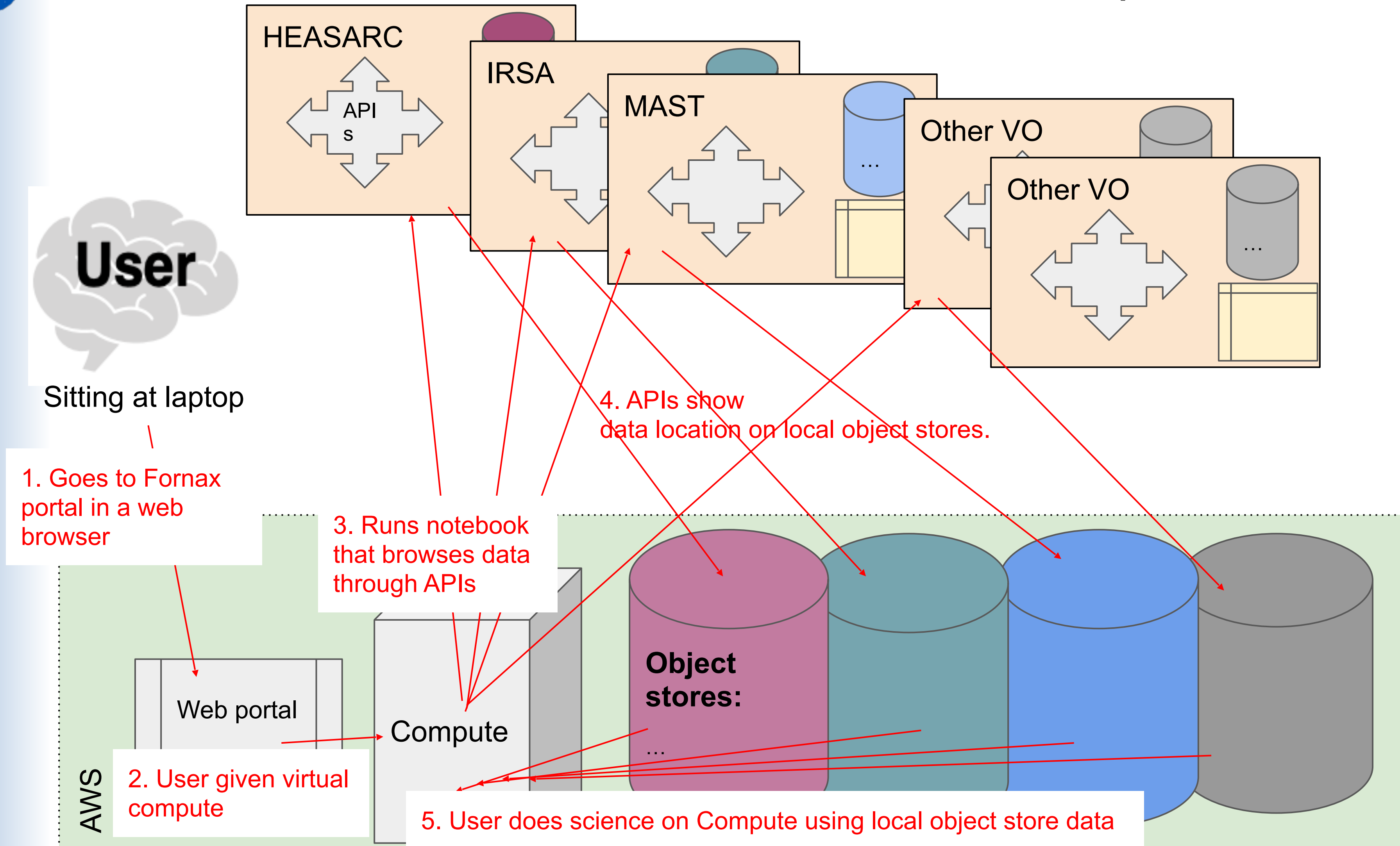


HEASARC • IRSA
NED • MAST

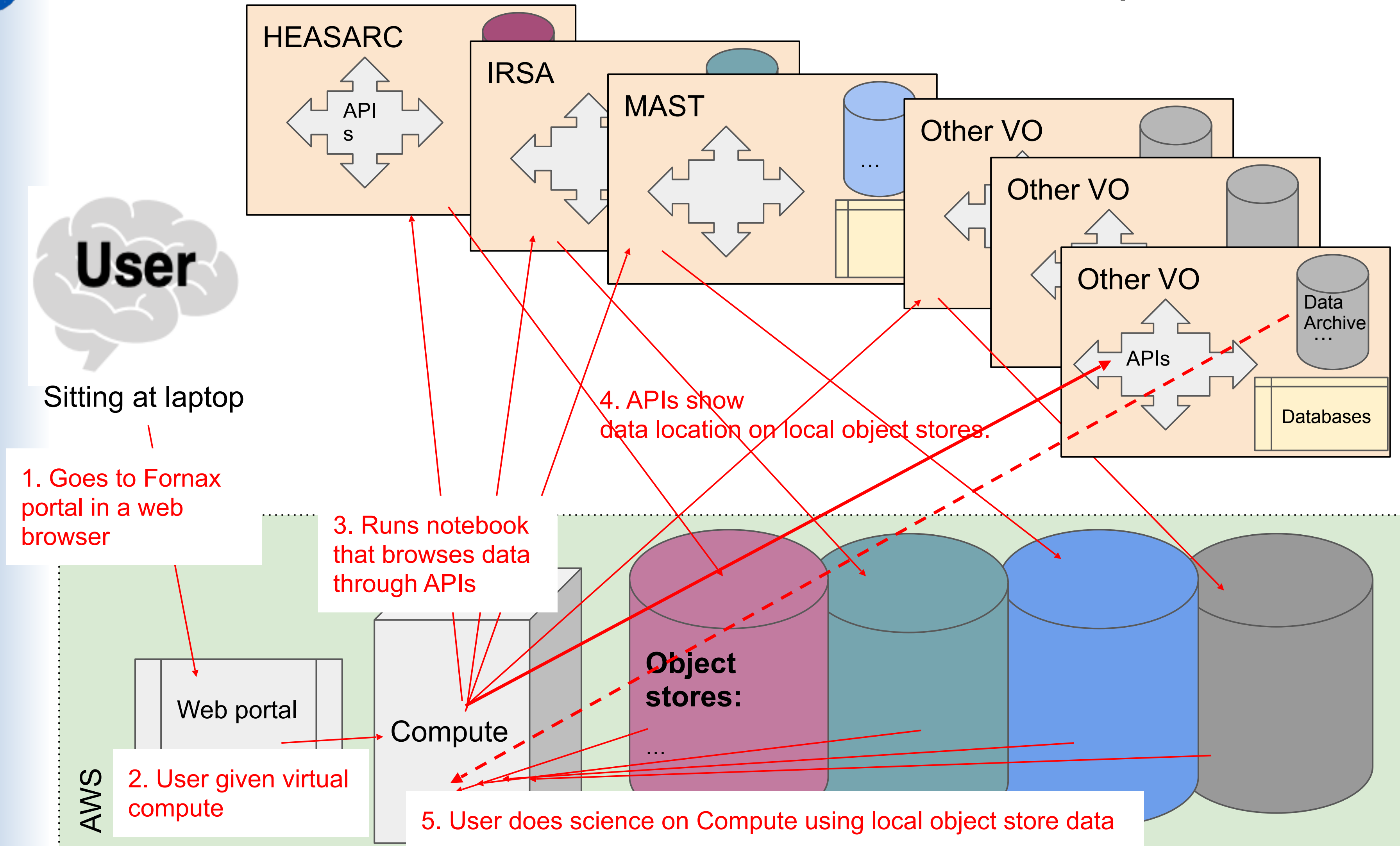
NASA Astronomical Virtual Observatories

NAVO

Future use case flow on Fornax cloud science platform



Future use case flow on Fornax cloud science platform



Client Side Implementation

https://github.com/zoghbi-a/pyvo/tree/cloud-links

```
import pyvo
from pyvo.utils import activate_features
activate_features('cloud')
```

Activate dev cloud feature

```
res = pyvo.dal.sia.search(query_url, pos=pos, size=0.0)
```

Basic SIA search

Process cloud information in the DALResult (through CloudMixin)

- **Look for JSON column**
- **Call datalinks**
- **etc**

```
r = res[0]
r.enable_cloud()

r.get_cloud_uris('aws')
```

List AWS URIs

```
['s3://fornaxdev-east1-curated/FTP/chandra/data/byobsid/2/3052/primary/acisf03052N004_cntr_img2.
 's3://heasarc-public/chandra/data/byobsid/2/3052/primary/acisf03052N004_cntr_img2.jpg']
```

Client Side Implementation

https://github.com/zoghbi-a/pyvo/tree/cloud-links

```
# this is a summary of the access point
r.access.summary()
```

```
|prem | https://heasarc.gsfc.nasa.gov/FTP/chandra/data/byobsid/2/3052/primary/acisf03052N004_cntr_img2.jpg
|aws  | s3://fornaxdev-east1-curated/FTP/chandra/data/byobsid/2/3052/primary/acisf03052N004_cntr_img2.jpg
|aws  | s3://heasarc-public/chandra/data/byobsid/2/3052/primary/acisf03052N004_cntr_img2.jpg
```

To download the data, we can call the download method on the record, specifying where we want the data to be download from.
For example, to download from `prem` servers, we do:

```
print('\n-- download from prem --')
path = r.download('prem')
print(path)
```

```
-- download from prem --
/Home/eud/azoghbi/.astropy/cache/download/url/24d5ebd459d6c99ca4427916aa8ac6df/contents
```

and to download from `aws`, we do:

```
print('\n-- download from aws --')
path = r.download('aws')
print(path)
```

```
-- download from aws --
Downloading s3://heasarc-public/chandra/data/byobsid/2/3052/primary/acisf03052N004_cntr_img2.jpg to acisf03052N004_c
ntr_img2.jpg ... [Done]
acisf03052N004_cntr_img2.jpg
```

Print access summary & download data

Server-side options:

- Datalinks:
  - - Requires another call
  - - No all providers have a Datalink service (NASA)
  - - Not clear how does a client distinguish between cloud data of some column vs other linked data that standard datalinks are used for (e.g. similar, parent data etc.).
  - - Need for more standards to be defined to handle the extra information (e.g. storage region etc)
  - Client/server need new standards.
  - + Flexible.
- Add a separate column with JSON text
  - - Maybe against VO ethos.
  - + Easy and flexible: define a few standards, and it works for all cloud providers (AWS, Google, Azure, or simple site mirrors).

What is a “science platform”? No idea, but Fornax is:

- Fairly **generic**, not domain specific, because meant for *all NASA astrophysics users*.
  - Also developed in collaboration with NASA heliophysics, planetary, earth sciences, etc.
- To most of our target users: JupyterLab, Linux terminal, and Python.
  - Our plan for user-friendly ‘services’ will initially be a library of Python notebooks people can use and modify.
  - Additional Python data access routines to be contributed into PyVO and/or astroquery.
  - Collaboration via user-defined groups and shareable private spaces.
- Infrastructure components that will be part of the open source infrastructure-as-code:
  - Open Science Studio (descended from Pangeo, DaskHub)
  - Security monitoring infrastructure (incl. both AWS and maybe some commercial products).
  - Cost tracking by individual science user plus guardrails plus a user cost dashboard.
  - Long jobs interface: queue-like system
  - Development to be upstreamed back into Open Science Studio or stand-alone open source software.
- User interface apps can be deployed on top. Planned:
  - NASA/Navteca developing an API Baker that can turn a notebook into a callable service.
  - (Before ChatGPT3), Navteca testing LLM for in-platform help bot trained on curated documentation.
  - NASA/Navteca’s Pasarela lets you create a URL from your website to open a session on the platform with the appropriate code and data ready to go as set by the website for the user.
  - Large catalog cross-matching collab. with LSST (LINCC), see M. Juric talk.
  - Other open source astronomy Jupyter apps as appropriate.

Summary: areas of common interest

- IVOA-related:
  - VO-compatible cloud data access
    - E.g., alternative/extension to ObsCore access_url?
  - VO-compatible discoverability of cloud data
    - E.g., how to find out if dataset X is on AWS eu-west-1 from Registry?
    - E.g., how to discover which platform will have dataset X located proximate to compute?
  - VO-compatible collaboration spaces on science platforms
    - (relation to VOSpace?)
- More general infrastructure:
  - Security and cost controls.
  - Authentication and Authorization, Single Sign On, etc.
  - Lots of other things I'm sure...

NAVO NASA Astronomical Virtual Observatories
HEASARC • IRSA NED • MAST

Extra slides with the gory details

Cloud Data & VO

Fornax Access Layer WG
(Slides by Abdu Zoghbi)

Context & Goal are broader than Fornax

- Many data providers are serving data from the cloud.
- The same data continue to be served from on-prem servers.
- Many cloud providers.
- The goal is to let users know about that data
  - Within current standards?
  - Creating new standards?

HEASARC • IRSA
NED • MAST

NASA Astronomical Virtual Observatories

NAVO

Current Standard

- Focus on SIA for now (look for solutions that are applicable elsewhere).
- In SIA1: the URL is in a column with UCD=VOX:Image_AccessReference
- In SIA2: Column name access_url (and access_format)

Cloud Cases

1. Simple URI to open data: we have a direct data URI
   - e.g. uri=s3://bucket_name/key/to/file

1. More information than a simple URI:
   - Bucket: bucket_name
   - Region: us-east-1
   - Key: key/to/file
   - Access_type: region-only (or open or restricted etc)

Cloud Cases: Simple URI

- Relatively easy to handle:
  - Serve with datalinks:
    - - Requires another call
    - - Not all providers have a Datalink service (NASA)
    - - Not clear how does a client distinguish between cloud data of some column vs other linked data that standard datalinks are used for (e.g. similar, parent data etc.).
    - + Flexible.
  - Add a separate column with a specific UCD?
    - + easy to implement.
    - - Require a separate column for every cloud provider serving the data.
    - User or client needs to parse URI and use the relevant cloud API.

Cloud Cases: More than A URI

- Datalinks:
  - - Requires another call
  - - No all providers have a Datalink service (NASA)
  - - Not clear how does a client distinguish between cloud data of some column vs other linked data that standard datalinks are used for (e.g. similar, parent data etc.).
  - - Need for more standards to be defined to handle the extra information (e.g. storage retion etc)
  - Client/server need new standards.
  - + Flexible.
- Add a separate column with JSON text?
  - - Maybe Against VO ethos.
  - + Very Flexible: define a few standards, and it works for all cloud providers (AWS, Google, Azure, or simple site mirrors).

Server Side Implementation

Server Side Implementation

Add a column with a specific UCD

Table Head

```
 <FIELD datatype="char" arraysize="*" ucd="meta.dataset;meta.ref.aws" name="aws"/>
▼<DATA>
  ▼<TABLEDATA>
    ▼<TR>
```

Table Data

```
<TD>182.63625</TD>
<TD>39.40544</TD>
<TD>CHANDRA ACIS-S</TD>

<TD>s3://heasarc-public/chandra/data/byobsid/2/3052/primary/acisf03052N004_cntr_img2.jpg</TD>
```

Server Side Implementation

Use Datalinks:

```
<RESOURCE utype="adhoc:service" ID="cloudlinks" type="meta">
  <PARAM datatype="char" arraysize="*" name="standardID" value="ivo://ivoa.net/std/DataLink#links-1.0"/>
  <PARAM datatype="char" arraysize="*" name="accessURL" value="https://heasarc.gsfc.nasa.gov/xamin/vo/datalink/chanmaster"/>
  <GROUP name="inputParams">
    <PARAM ref="DataLinkID" datatype="char" arraysize="*" name="id" value=""/>
    <PARAM datatype="char" arraysize="*" name="provider" value="prem">
      <VALUES>
        <OPTION name="On prem servers" value="prem"/>
        <OPTION name="AWS region 1" value="aws:us-east1"/>
        <OPTION name="AWS some other region" value="aws:us-east2"/>
        <OPTION name="GC some region" value="gc"/>
      </VALUES>
    </PARAM>
  </GROUP>
</RESOURCE>
```

Seperate ID so we don't call other datalink services

Reference to main table

New parameter

Provider options: Aws, gc, prem etc

Server Side Implementation

Use Datalinks:

https://datalink-url?provider=prem

```
<TABLE>
    <FIELD datatype="char" arraysize="*" ucd="meta.id;meta.main" name="ID"/>
    <FIELD datatype="char" arraysize="*" ucd="meta.ref.url" name="access_url"/>
    ...
    <DATA>
        <TABLEDATA>
            <TR>
                <TD>[SOME_ID]</TD>
                <TD>https://someurl/path/to/some/file.fits</TD>
                ...
            </TR>
        </TABLEDATA>
    </DATA>
</TABLE>
```

https://datalink-url?provider=aws

```
<TABLE>
    <FIELD datatype="char" arraysize="*" ucd="meta.id;meta.main" name="ID"/>
    <FIELD datatype="char" arraysize="*" ucd="meta.ref.url" name="access_url"/>
    ...
    <DATA>
        <TABLEDATA>
            <TR>
                <TD>[SOME_ID]</TD>
                <TD>s3://somebucket/path/to/some/file.fits</TD>
                ...
            </TR>
        </TABLEDATA>
    </DATA>
</TABLE>
```

HEASARC • IRSA
NED • MAST

NASA Astronomical Virtual Observatories

NAVO

Server Side Implementation

Use a Column with JSON text for serving detailed storage information

```
<FIELD datatype="char" arraysize="*" name="cloud_access"/>
<FIELD datatype="char" arraysize="*" ucd="meta.dataset;meta.ref.aws" name="aws"/>
▼<DATA>
 ▼<TABLEDATA>
  ▼<TR>

    <TD>182.63625</TD>
    <TD>39.40544</TD>
    <TD>CHANDRA ACIS-S</TD>
    <TD>{"aws": { "bucket_name": "fornaxdev-east1-curated", "region": "us-east-1",
        "key": "FTP/chandra/data/byobsid/2/3052/primary/acisf03052N004_cntr_img2.jpg" }}</TD>
```

Client Side Implementation

https://github.com/zoghbi-a/pyvo/tree/cloud-links

Client Side Implementation

```
import pyvo
from pyvo.utils import activate_features
activate_features('cloud')
```

Activate cloud feature

```
res = pyvo.dal.sia.search(query_url, pos=pos, size=0.0)

res
```

Basic SIA search

```
<Table length=2>
obsid   status   ...                                   aws
                 ...
object  object   ...                                  object
------ -------- ... ---------------------------------------------------------------------
  3052 archived ... s3://heasarc-public/chandra/data/byobsid/2/3052/primary/acisf03052N004_cntr_img2.jpg
  3480 archived ... s3://heasarc-public/chandra/data/byobsid/0/3480/primary/acisf03480N004_cntr_img2.jpg
```

Client Side Implementation

Process cloud information in the DALResult (through CloudMixin)

- **Look for Json Column**
- **Call datalinks**
- **etc**

```
r = res[0]
r.enable_cloud()

r.get_cloud_uris('aws')
```

List AWS URIs

```
['s3://fornaxdev-east1-curated/FTP/chandra/data/byobsid/2/3052/primary/acisf03052N004_cntr_img2.jpg',
 's3://heasarc-public/chandra/data/byobsid/2/3052/primary/acisf03052N004_cntr_img2.jpg']
```

Client Side Implementation

```
# this is a summary of the access point
r.access.summary()
```

```
|prem | https://heasarc.gsfc.nasa.gov/FTP/chandra/data/byobsid/2/3052/primary/acisf03052N004_cntr_img2.jpg
|aws  | s3://fornaxdev-east1-curated/FTP/chandra/data/byobsid/2/3052/primary/acisf03052N004_cntr_img2.jpg
|aws  | s3://heasarc-public/chandra/data/byobsid/2/3052/primary/acisf03052N004_cntr_img2.jpg
```

To download the data, we can call the download method on the record, specifying where we want the data to be download from. For example, to download from `prem` servers, we do:

```
print('\n-- download from prem --')
path = r.download('prem')
print(path)
```

Print access summary & download data

```
-- download from prem --
/Home/eud/azoghbi/.astropy/cache/download/url/24d5ebd459d6c99ca4427916aa8ac6df/contents
```

and to download from `aws`, we do:

```
print('\n-- download from aws --')
path = r.download('aws')
print(path)
```

```
-- download from aws --
Downloading s3://heasarc-public/chandra/data/byobsid/2/3052/primary/acisf03052N004_cntr_img2.jpg to acisf03052N004_c
ntr_img2.jpg ... [Done]
acisf03052N004_cntr_img2.jpg
```

Client Side Implementation

Use with non-pyvo tables (e.g. astroquery) using JSON column (no datalinks in tables)

Use case 4: No mixins; Work with Tables

In this case, we use assume the user is not passing a pyvo object (`Record` or `DALResults`), but instead is passing an astropy `Table` or `Row` .

In this case, datalinks and ucd cloud information are not available as they are pyvo features, but we can process the cloud JSON text in a cloumn named `cloud_acess` .

To do this, the `DALResult` is converted to a `Table` first, then it is passed to `pyvo.utils.cloud.enable_cloud` separately. The `handler` is `CloudHandler` object that has a download methods and allows access to the URIs etc..

```
from pyvo.utils.cloud import enable_cloud
```

```
tab = res.to_table()
handler = enable_cloud(tab)
```

```
handler.access_points[0].summary()
```

```
|prem  | https://heasarc.gsfc.nasa.gov/FTP/chandra/data/byobsid/2/3052/primary/acisf03052N004_cntr_img2.jpg
|aws   | s3://fornaxdev-east1-curated/FTP/chandra/data/byobsid/2/3052/primary/acisf03052N004_cntr_img2.jpg
```

```
handler.download('prem')
```