STScI | SPACE TELESCOPE SCIENCE INSTITUTE

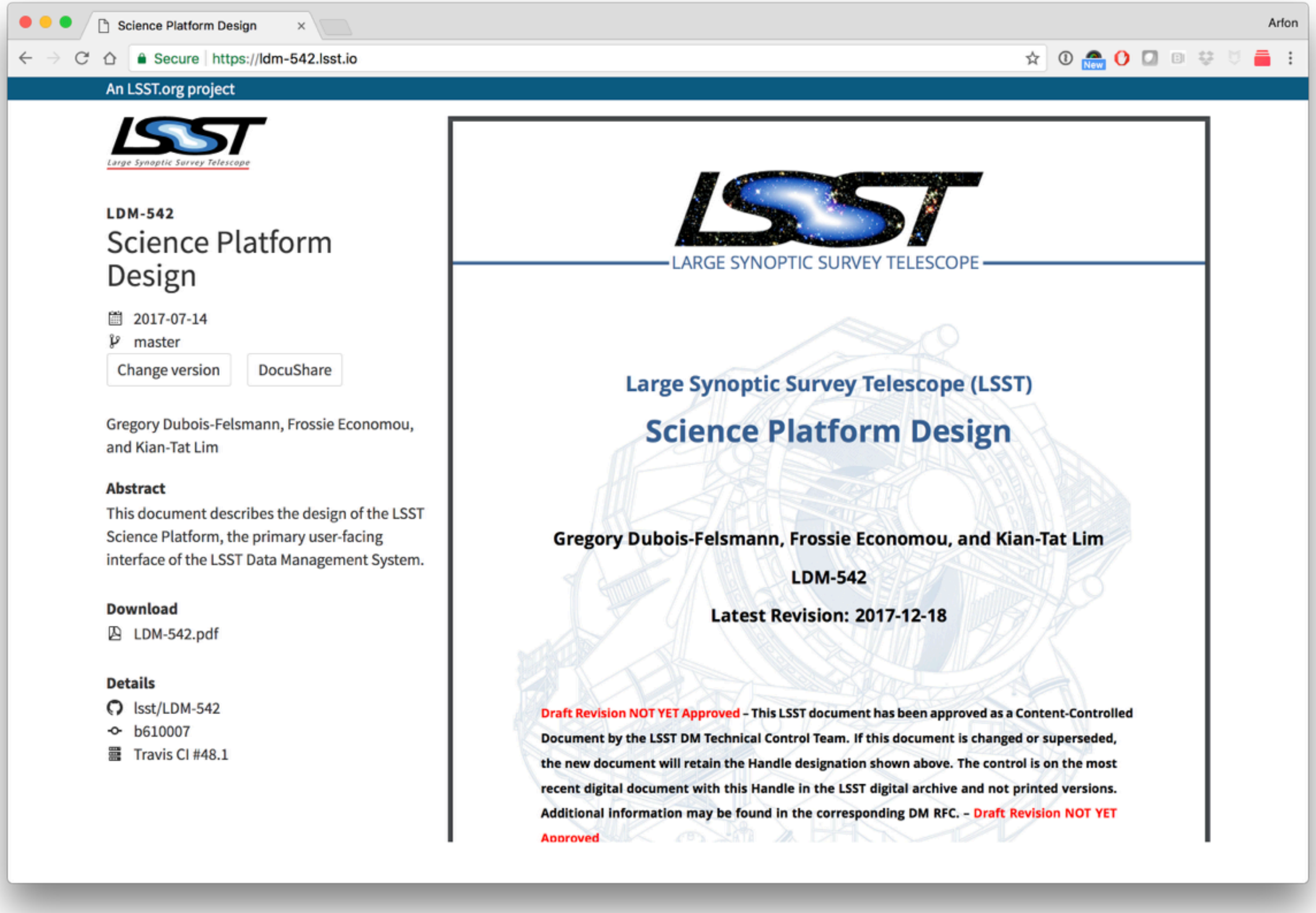
EXPANDING THE FRONTIERS OF SPACE ASTRONOMY

Server-Side Analytics

~~Science Platforms~~

Arfon & Mike

Some definitions



Science Platform Design
Arfon
Secure https://ldm-542.lsst.io
An LSST.org project
Large Synoptic Survey Telescope

LDM-542

Science Platform Design

2017-07-14

master

Change version DocuShare

Gregory Dubois-Felsmann, Frossie Economou, and Kian-Tat Lim

Abstract

This document describes the design of the LSST Science Platform, the primary user-facing interface of the LSST Data Management System.

Download

LDM-542.pdf

Details

lsst/LDM-542

b610007

Travis CI #48.1

LARGE SYNOPTIC SURVEY TELESCOPE

Large Synoptic Survey Telescope (LSST)

Science Platform Design

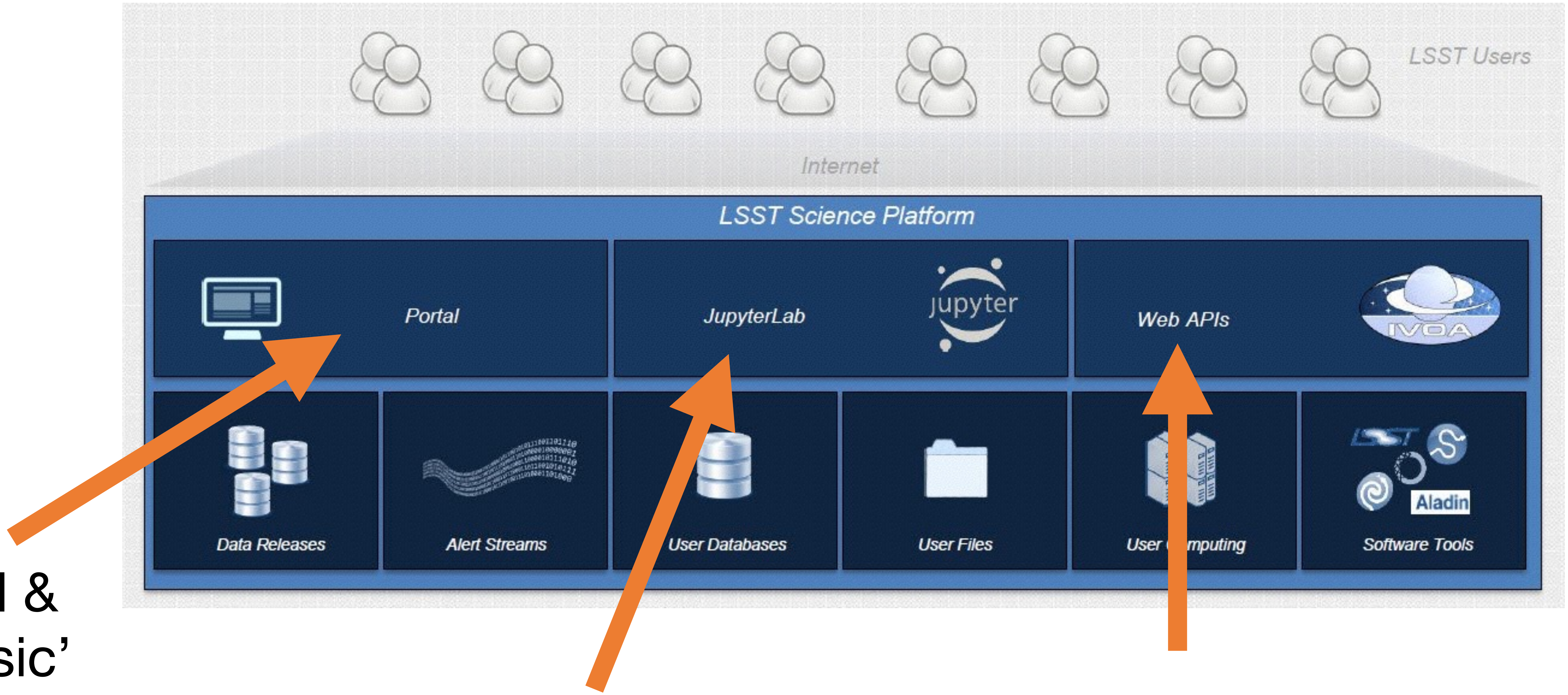
Gregory Dubois-Felsmann, Frossie Economou, and Kian-Tat Lim

LDM-542

Latest Revision: 2017-12-18

Draft Revision NOT YET Approved – This LSST document has been approved as a Content-Controlled Document by the LSST DM Technical Control Team. If this document is changed or superseded, the new document will retain the Handle designation shown above. The control is on the most recent digital document with this Handle in the LSST digital archive and not printed versions. Additional information may be found in the corresponding DM RFC. – Draft Revision NOT YET Approved

Some definitions



MAST Portal &
MAST ‘Classic’

Focus of this presentation

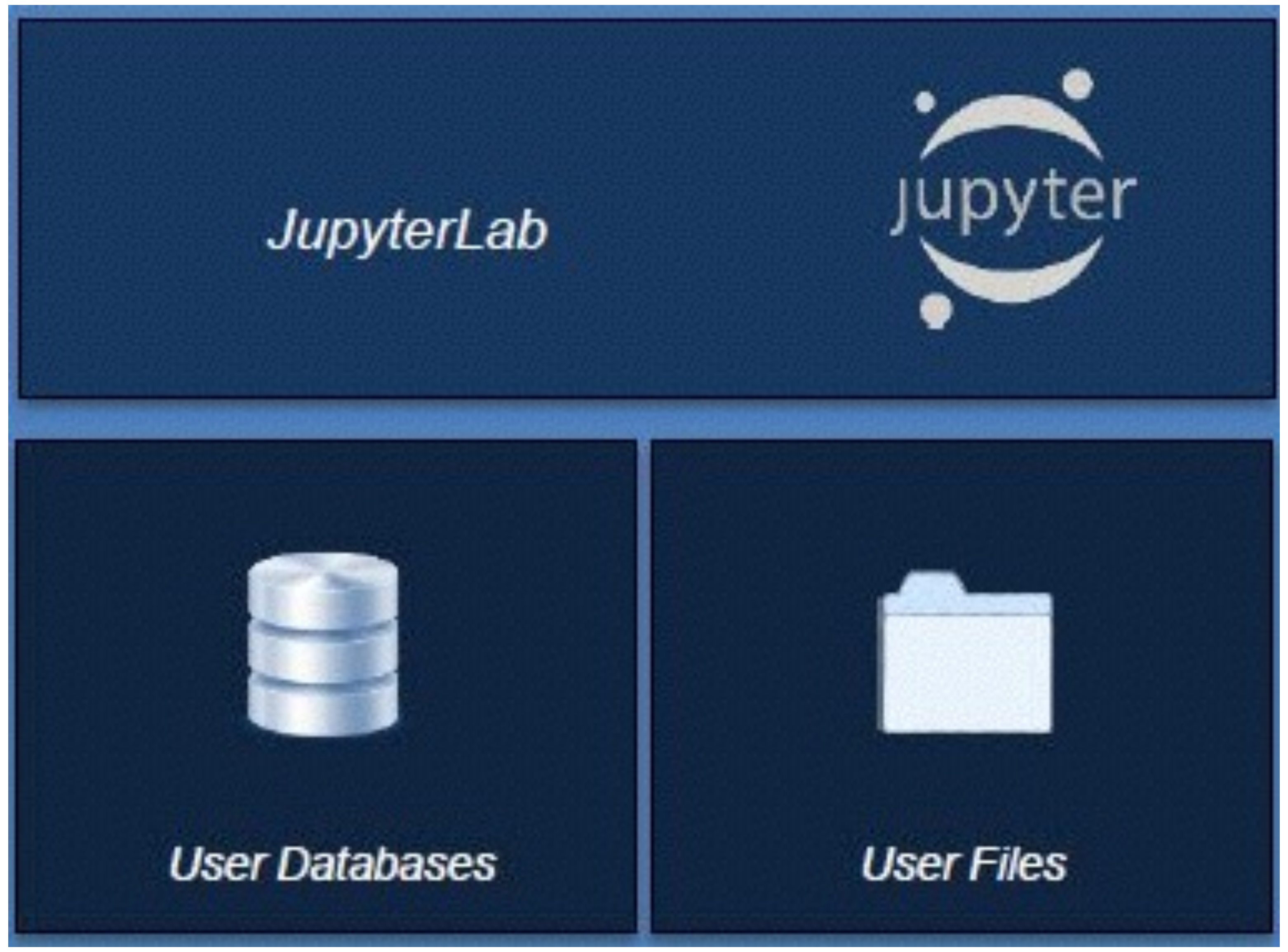
MAST APIs

Common technologies, many implementers

docker

jupyterhub

kubernetes



JupyterLab

jupyter

User Databases

User Files

SciServer

NOAO DATALAB

LSST

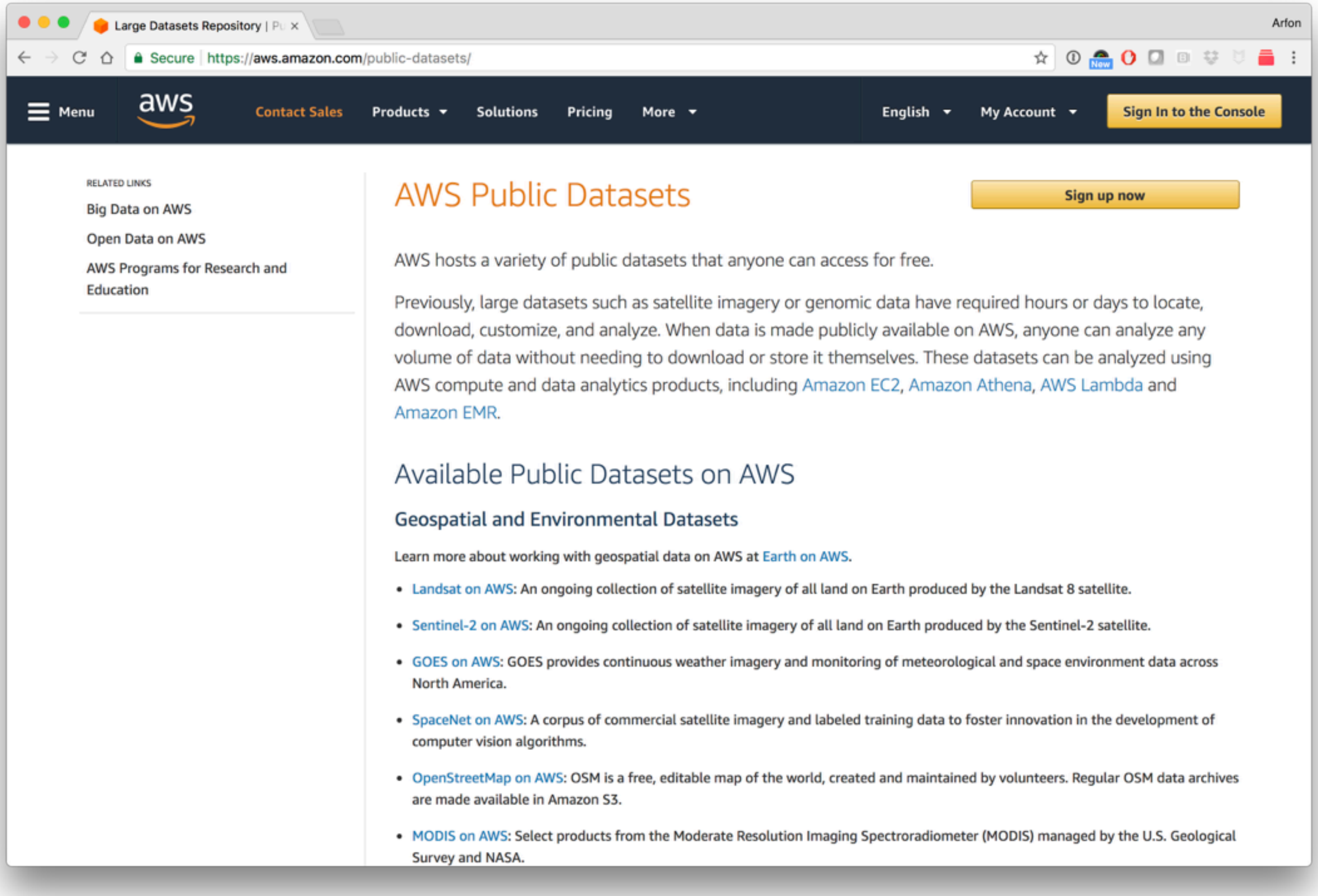
Large Synoptic Survey Telescope

STScI | SPACE TELESCOPE SCIENCE INSTITUTE

Highlights of what we're building

- Cloud-hosted copy of all HST public data
- (Live) JupyterLab environment with *some* compute/storage
- Collection of Docker containers installed with common tools

Amazon Web Services: Public Dataset Program



AWS Public Datasets

AWS hosts a variety of public datasets that anyone can access for free.

Previously, large datasets such as satellite imagery or genomic data have required hours or days to locate, download, customize, and analyze. When data is made publicly available on AWS, anyone can analyze any volume of data without needing to download or store it themselves. These datasets can be analyzed using AWS compute and data analytics products, including Amazon EC2, Amazon Athena, AWS Lambda and Amazon EMR.

Available Public Datasets on AWS

Geospatial and Environmental Datasets

Learn more about working with geospatial data on AWS at Earth on AWS.

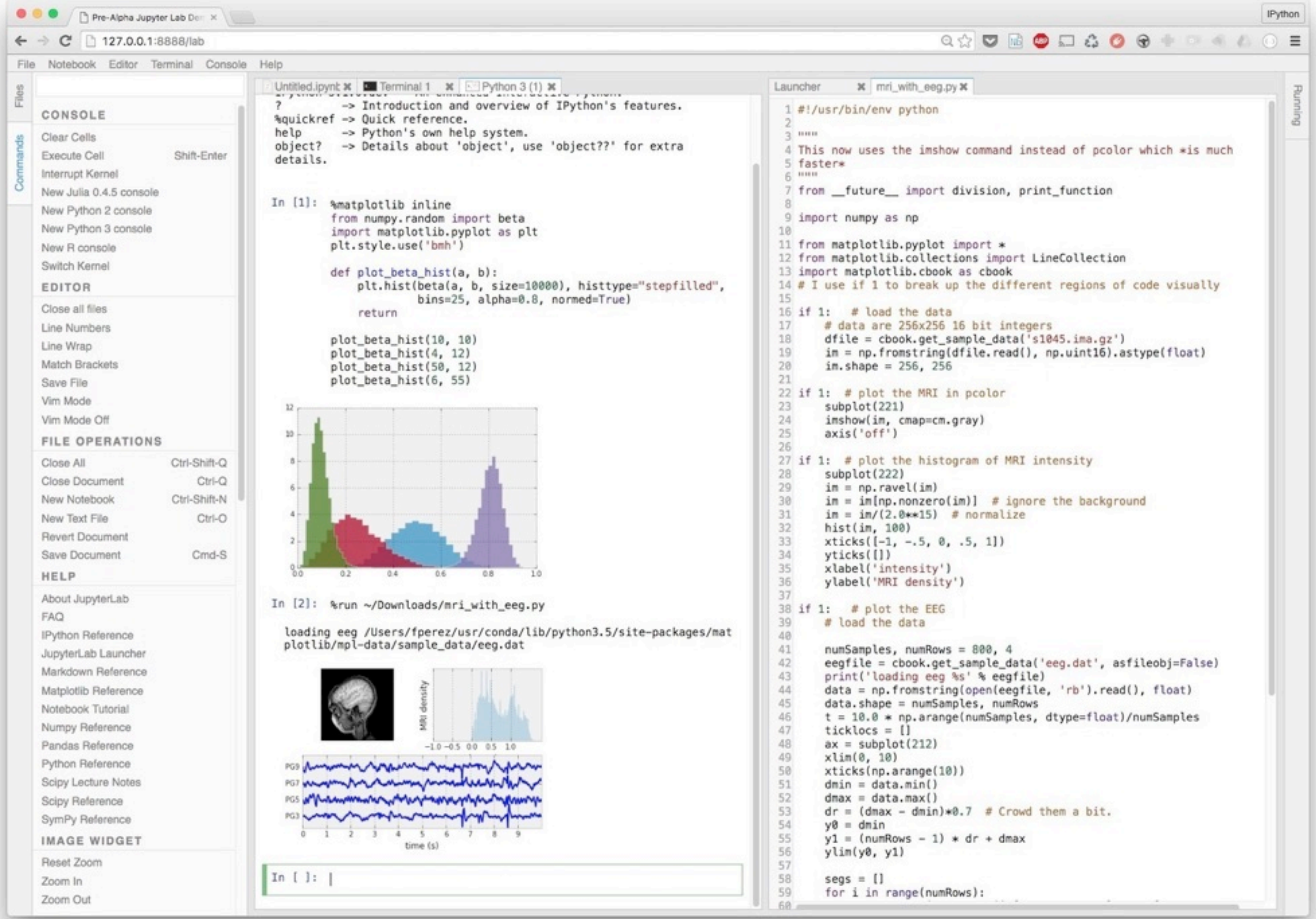
- Landsat on AWS: An ongoing collection of satellite imagery of all land on Earth produced by the Landsat 8 satellite.
- Sentinel-2 on AWS: An ongoing collection of satellite imagery of all land on Earth produced by the Sentinel-2 satellite.
- GOES on AWS: GOES provides continuous weather imagery and monitoring of meteorological and space environment data across North America.
- SpaceNet on AWS: A corpus of commercial satellite imagery and labeled training data to foster innovation in the development of computer vision algorithms.
- OpenStreetMap on AWS: OSM is a free, editable map of the world, created and maintained by volunteers. Regular OSM data archives are made available in Amazon S3.
- MODIS on AWS: Select products from the Moderate Resolution Imaging Spectroradiometer (MODIS) managed by the U.S. Geological Survey and NASA.

- ~120TB public HST data for ACS, COS, STIS, WFC3, FGS
- Range of high-impact datasets
- Hosted in cloud - 'highly available'
- Enable new types of data analyses
- Hosted at no cost to STScI/NASA

Amazon Web Services: Public Dataset Program

- Data is hosted in an S3 region (for ‘free’)
- Conditions of inclusion in program: make the data useful:
  - An AMI with a demonstration of how to use the public dataset must be provided
  - AWS recover costs by making access to the data free from AWS services (EC2), making it cost effective for researchers to buy AWS computing time
  - Enables new types of analyses

JupyterLab environment



```
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra
details.
```

```
In [1]:  %matplotlib inline
         from numpy.random import beta
         import matplotlib.pyplot as plt
         plt.style.use('bmh')

         def plot_beta_hist(a, b):
             plt.hist(beta(a, b, size=10000), histtype="stepfilled",
                      bins=25, alpha=0.8, normed=True)
             return

         plot_beta_hist(10, 10)
         plot_beta_hist(4, 12)
         plot_beta_hist(50, 12)
         plot_beta_hist(6, 55)
```

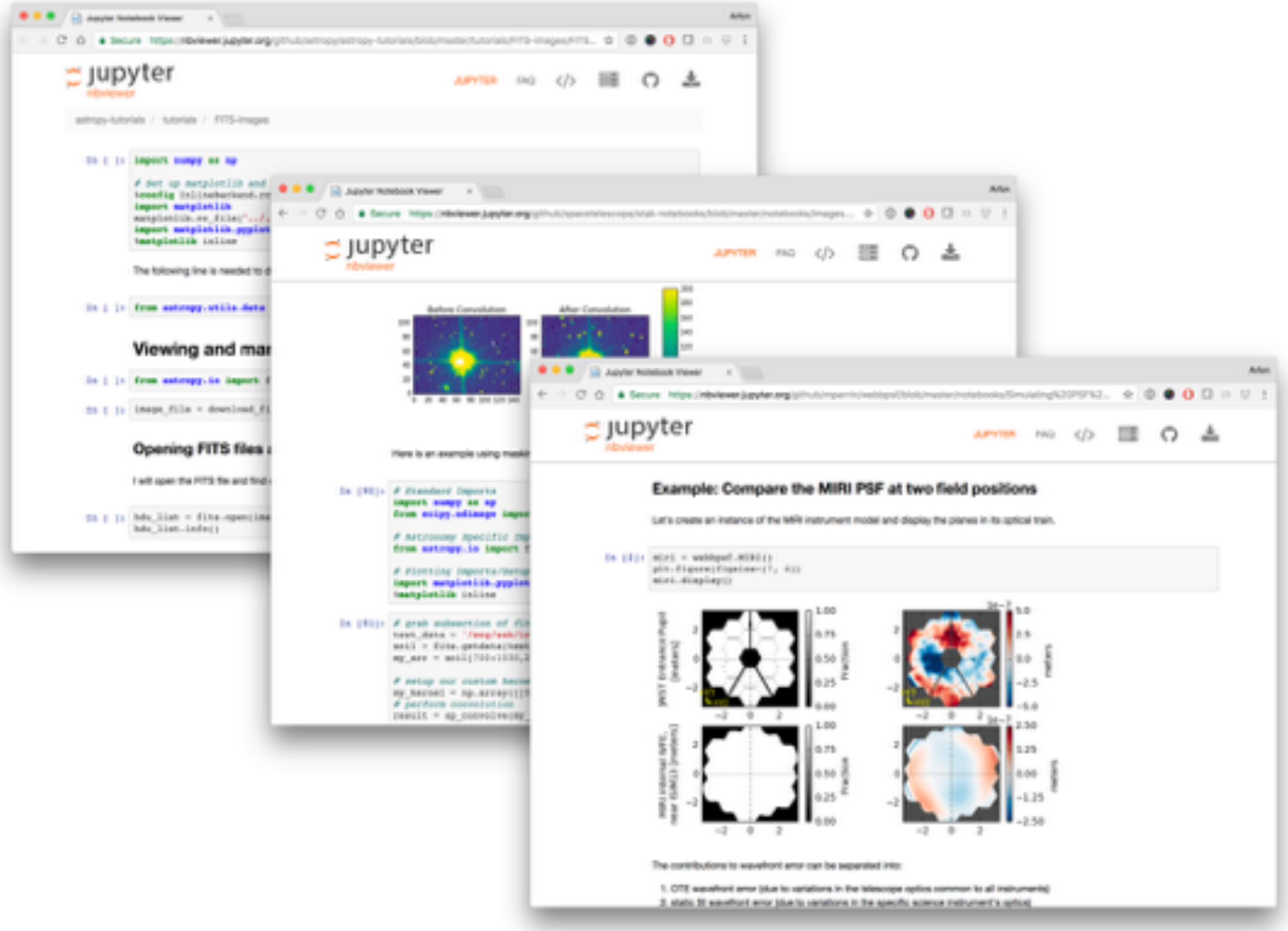
```
In [2]:  %run ~/Downloads/mri_with_eeg.py

  loading eeg /Users/fperez/usr/conda/lib/python3.5/site-packages/mat
  plotlib/mpl-data/sample_data/eeg.dat
```

```
In [ ]: |
```

```
#!/usr/bin/env python

"""
This now uses the imshow command instead of pcolor which *is much
faster*
"""
from __future__ import division, print_function

import numpy as np

from matplotlib.pyplot import *
from matplotlib.collections import LineCollection
import matplotlib.cbook as cbook
# I use if 1 to break up the different regions of code visually

if 1:   # load the data
    # data are 256x256 16 bit integers
    dfile = cbook.get_sample_data('s1045.ima.gz')
    im = np.fromstring(dfile.read(), np.uint16).astype(float)
    im.shape = 256, 256

if 1:  # plot the MRI in pcolor
    subplot(221)
    imshow(im, cmap=cm.gray)
    axis('off')

if 1:  # plot the histogram of MRI intensity
    subplot(222)
    im = np.ravel(im)
    im = im[np.nonzero(im)]  # ignore the background
    im = im/(2.0**15)  # normalize
    hist(im, 100)
    xticks([-1, -.5, 0, .5, 1])
    yticks([])
    xlabel('intensity')
    ylabel('MRI density')

if 1:   # plot the EEG
    # load the data

    numSamples, numRows = 800, 4
    eegfile = cbook.get_sample_data('eeg.dat', asfileobj=False)
    print('loading eeg %s' % eegfile)
    data = np.fromstring(open(eegfile, 'rb').read(), float)
    data.shape = numSamples, numRows
    t = 10.0 * np.arange(numSamples, dtype=float)/numSamples
    ticklocs = []
    ax = subplot(212)
    xlim(0, 10)
    xticks(np.arange(10))
    dmin = data.min()
    dmax = data.max()
    dr = (dmax - dmin)*0.7  # Crowd them a bit.
    y0 = dmin
    y1 = (numRows - 1) * dr + dmax
    ylim(y0, y1)

    segs = []
    for i in range(numRows):
```

- Interactive computing environment
- Where most development work is going from the core Jupyter team
- Works with community tools (e.g. Astropy)

Containers for different environments



import numpy as np

Viewing and ma

Opening FITS files

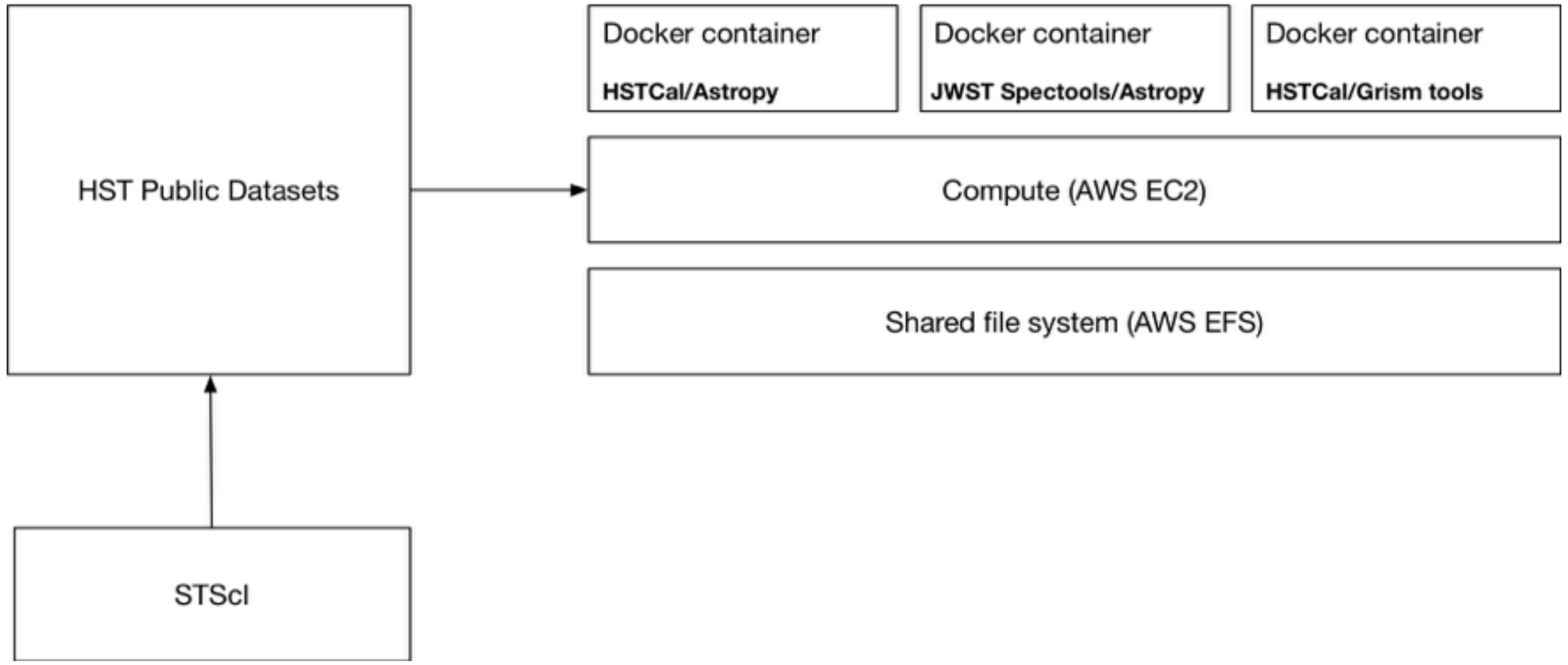
Example: Compare the MIRI PSF at two field positions

Let's create an instance of the MIRI instrument model and display the planes in its optical train.

```
miri = webbpsf.MIRI()
plt.figure(figsize=(7, 4))
miri.display()
```

The contributions to wavefront error can be separated into:

1. OTE wavefront error (due to variations in the telescope optics common to all instruments)
2. static SI wavefront error (due to variations in the specific science instrument's optics)



HST Public Datasets

STScI

Docker container HSTCal/Astropy

Docker container JWST Spectools/Astropy

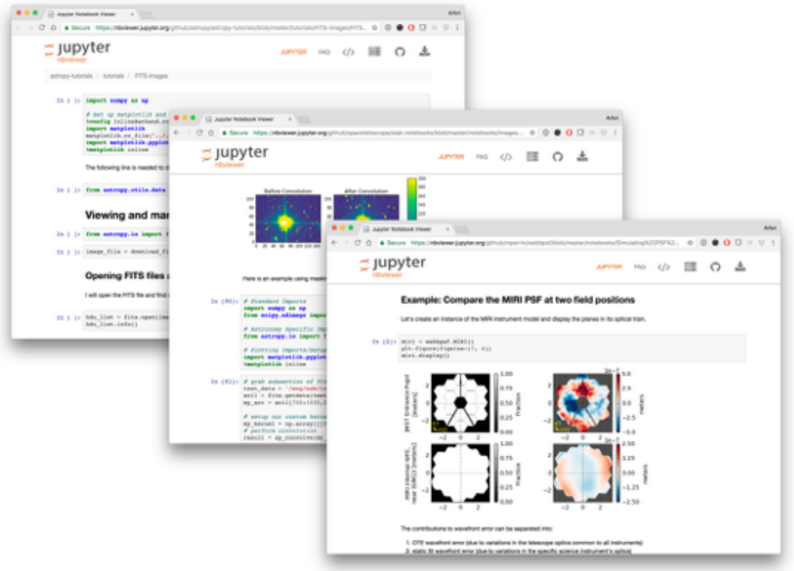
Docker container HSTCal/Grism tools

Compute (AWS EC2)

Shared file system (AWS EFS)

Technical details:

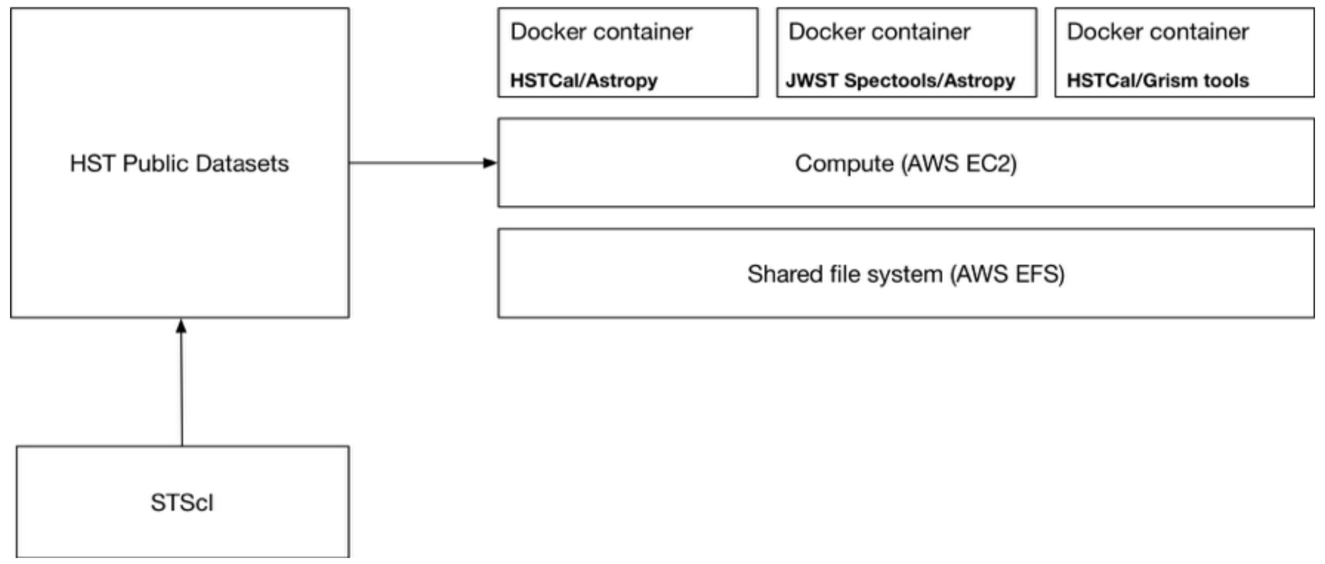
- JupyterHub, a multi-user Hub, spawns, manages, and proxies multiple instances of the single-user Jupyter notebook server.
- JupyterLab frontend provides notebook server, file management, and a terminal shell
- Using Docker to containerize science environments, allows a verified computing environment to be instantiated rapidly.
- Containers are versioned providing precise reproducibility of the research environment
- AWS computing resources scale with user load, providing good cost efficiency
- Container orchestration provides high availability, healing the cluster when there are hardware failures



Viewing and ma

Opening FITS files

Example: Compare the MIRI PSF at two field positions



Docker container
HSTCal/Astropy

Docker container
JWST Spectools/Astropy

Docker container
HSTCal/Grism tools

HST Public Datasets

Compute (AWS EC2)

Shared file system (AWS EFS)

STScI

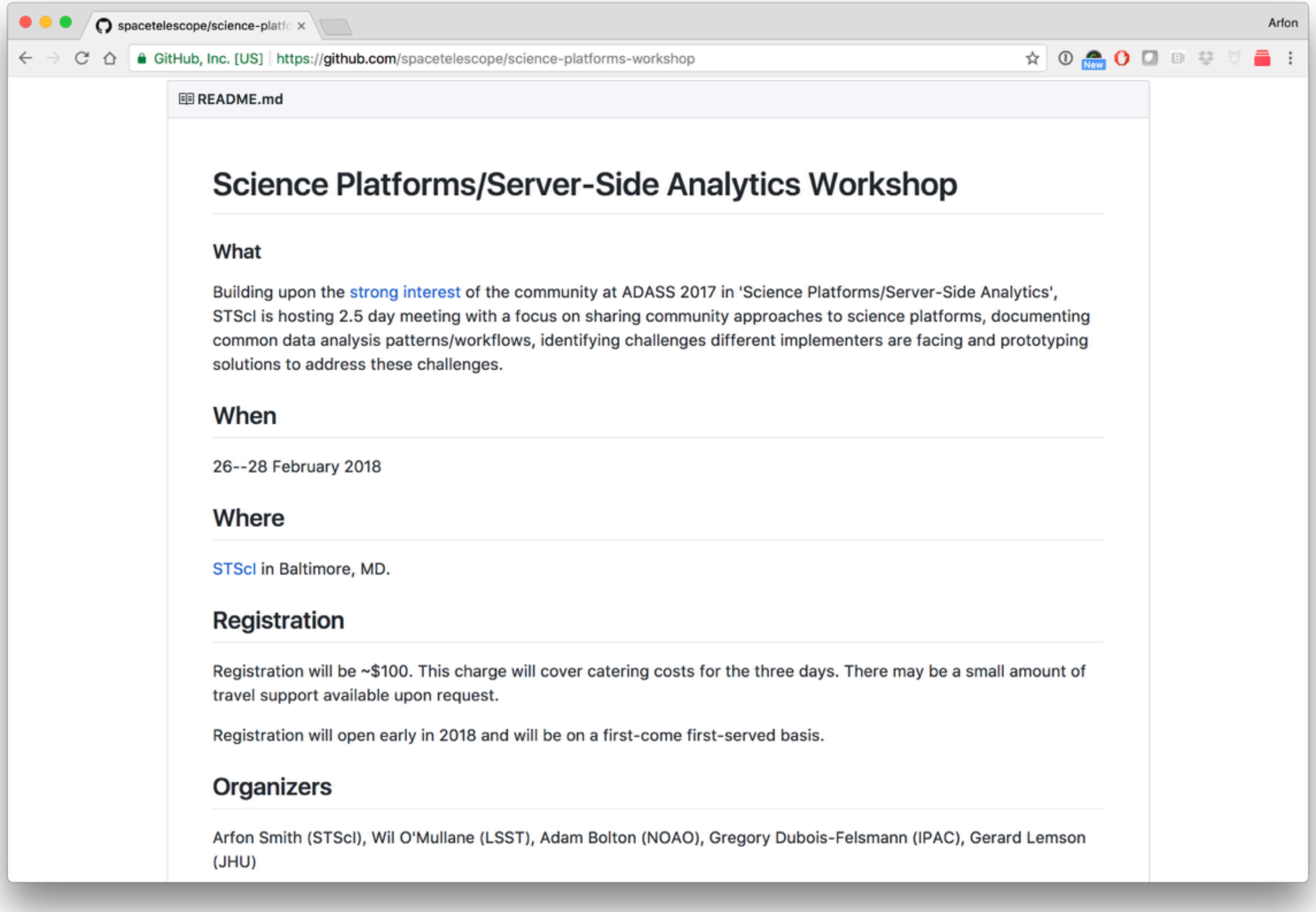
Core technical challenges

- Creating containers with pipeline/common software stacks
- Managing the cloud environment well:
  - User quotas (storage, compute etc.)
  - User storage (home directories), backups
  - Scalable, highly-available infrastructure (with cost caps/alerts)
- Relatively few large-scale JupyterHub deployments on AWS
- Inexperience of STScI with commercial cloud

Community coordination (i)

Iva++

Community coordination (ii)



spacetelescope/science-platfo ×
Arfon
GitHub, Inc. [US] https://github.com/spacetelescope/science-platforms-workshop
README.md

Science Platforms/Server-Side Analytics Workshop

What

Building upon the strong interest of the community at ADASS 2017 in 'Science Platforms/Server-Side Analytics', STScI is hosting 2.5 day meeting with a focus on sharing community approaches to science platforms, documenting common data analysis patterns/workflows, identifying challenges different implementers are facing and prototyping solutions to address these challenges.

When

26--28 February 2018

Where

STScI in Baltimore, MD.

Registration

Registration will be ~$100. This charge will cover catering costs for the three days. There may be a small amount of travel support available upon request.

Registration will open early in 2018 and will be on a first-come first-served basis.

Organizers

Arfon Smith (STScI), Wil O'Mullane (LSST), Adam Bolton (NOAO), Gregory Dubois-Felsmann (IPAC), Gerard Lemson (JHU)

Questions for the MUG (i)

- What problems do you foresee with our approach?
  - Who is this useful for?
  - What are sensible defaults for a service like this?
  - Does staging data in commercial cloud confuse our value proposition as a NASA archive?
  - Is asking the community to use AWS a problem? How can we lower the barrier to entry?

Questions for the MUG (ii)

- What extensions should we be thinking about?
  - Other missions (TESS?)
  - Other functionality? (e.g. batch processing)
  - Joint processing? (LSST/WFIRST/Euclid/JWST)
  - Accelerating JWST Early Release Science?
  - User support environment?
  - GO data delivery & archive proposals?
  - WFIRST (re)processing in the High Level Processing Partition