STScI | SPACE TELESCOPE SCIENCE INSTITUTE

EXPANDING THE FRONTIERS OF SPACE ASTRONOMY

JupyterHub on AWS

Tom Donaldson, on behalf of
Christian Mesh, Mike Fox, Iva Momcheva,
Josh Peek, Arfon Smith

Highlights of what we're building

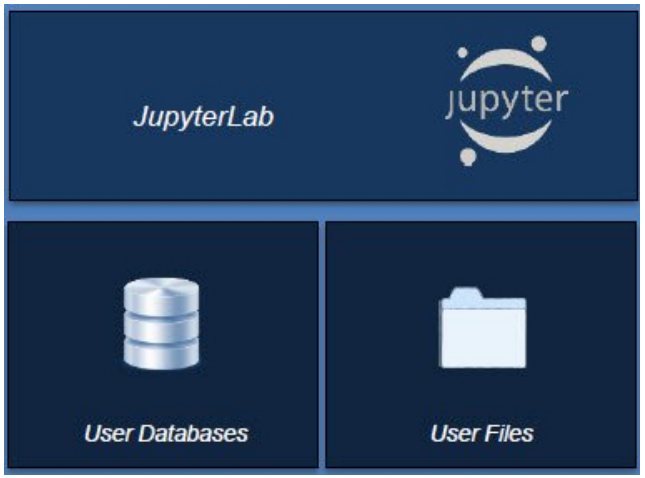
- Cloud-hosted copy of all HST public data
- (Live) JupyterLab environment with *some* compute/storage
- Collection of Docker containers installed with common tools

Common technologies, many implementers

docker

jupyterhub

kubernetes



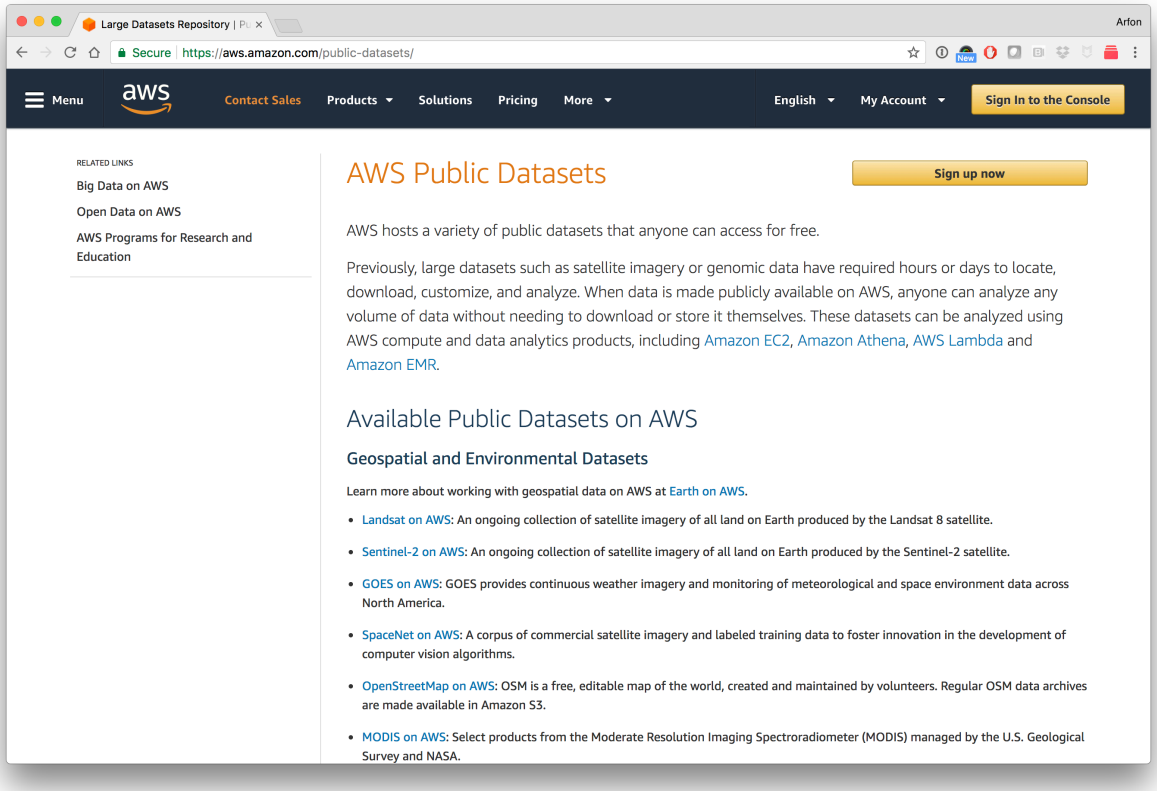
SciServer

NOAO DATALAB

LSST
Large Synoptic Survey Telescope

STScI | SPACE TELESCOPE SCIENCE INSTITUTE

Amazon Web Services: Public Dataset Program

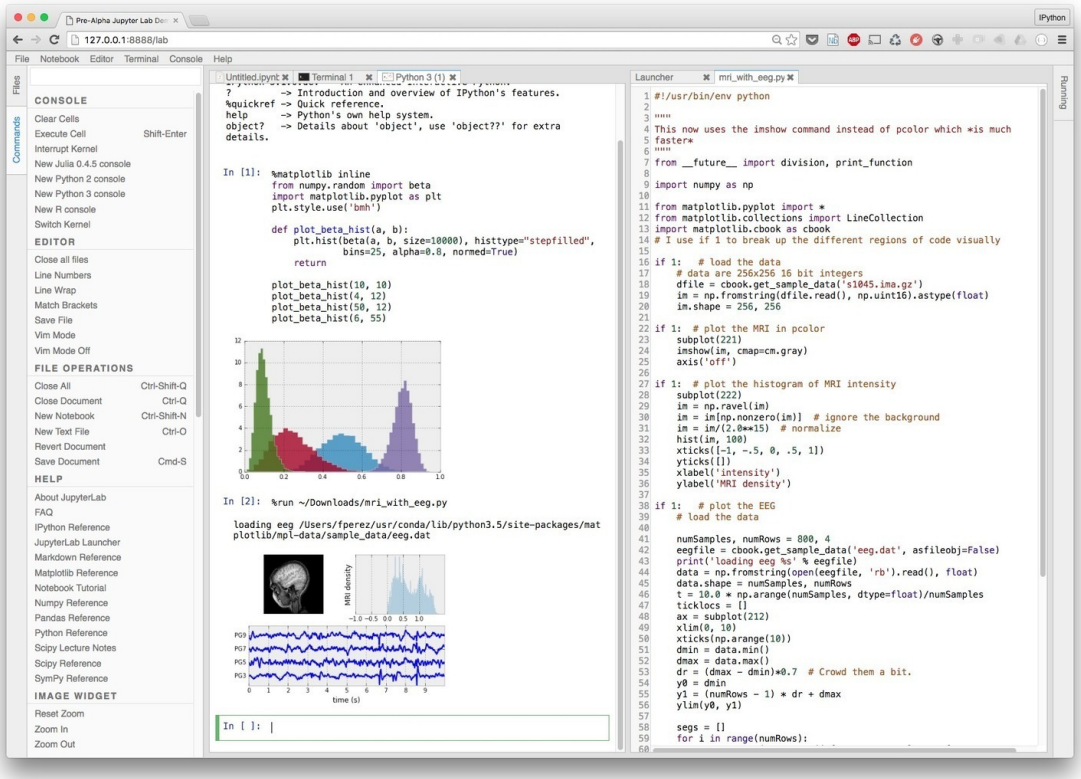


- ~120TB public HST data for ACS, COS, STIS, WFC3, FGS
- Range of high-impact datasets
- Hosted in cloud - ‘highly available’
- Enable new types of data analyses
- Hosted at no cost to STScI/NASA

Amazon Web Services: Public Dataset Program

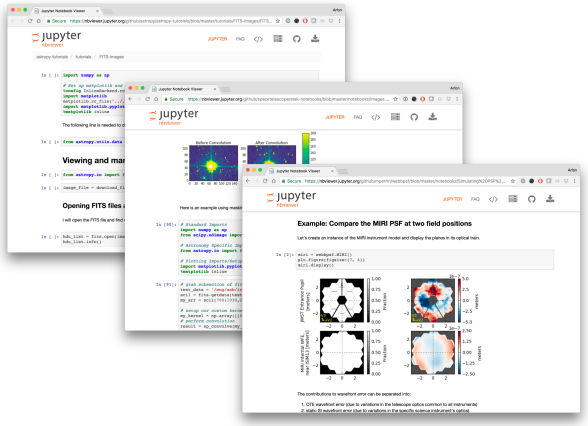
- Data is hosted in an S3 region (for ‘free’)
- Conditions of inclusion in program: make the data useful:
  - An AMI with a demonstration of how to use the public dataset must be provided
  - AWS recover costs by making access to the data free from AWS services (EC2), making it cost effective for researchers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

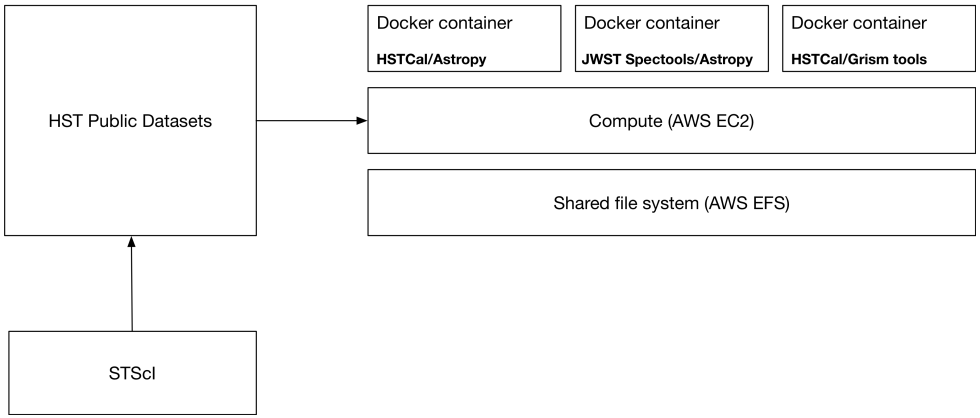
JupyterLab environment



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

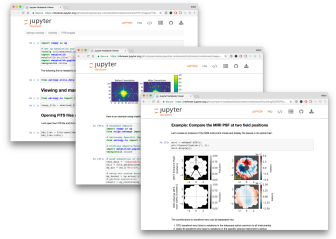
Containers for different environments

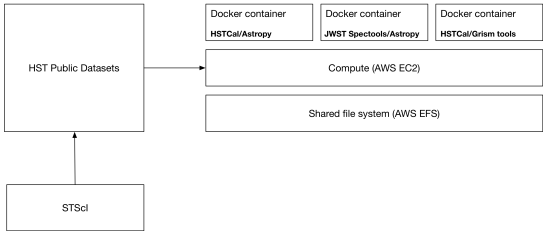




Technical details:

- 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





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)
- Relative inexperience of STScI with commercial cloud

** Relatively few large-scale JupyterHub deployments on AWS

Jupyter on AWS

- Started with Zero to JupyterHub
  - http://zero-to-jupyterhub.readthedocs.io/en/latest/
  - For installing and managing JupyterHub with Kubernetes
  - Documentation was sparse.
  - Many teething pains, especially in managing storage.
    - There were hundreds of lines of comments in k8s regarding how to handle stoarge, but with no decisions.
  - Worked through many issues, then contributed back rewritten documentation and code (via open source pull requests).
- Goal is to have an out-of-the-box solution anyone could spin up.
- Users have begun exercising the system.
- Automated tests being added to really push the system, especially with scalability.
- Using github authN now, but work has begun on integrating with STScI SSO
  - STScI SSO uses Shib, but we will probably create a bridge to that to leverage predominance of OAuth elsewhere.