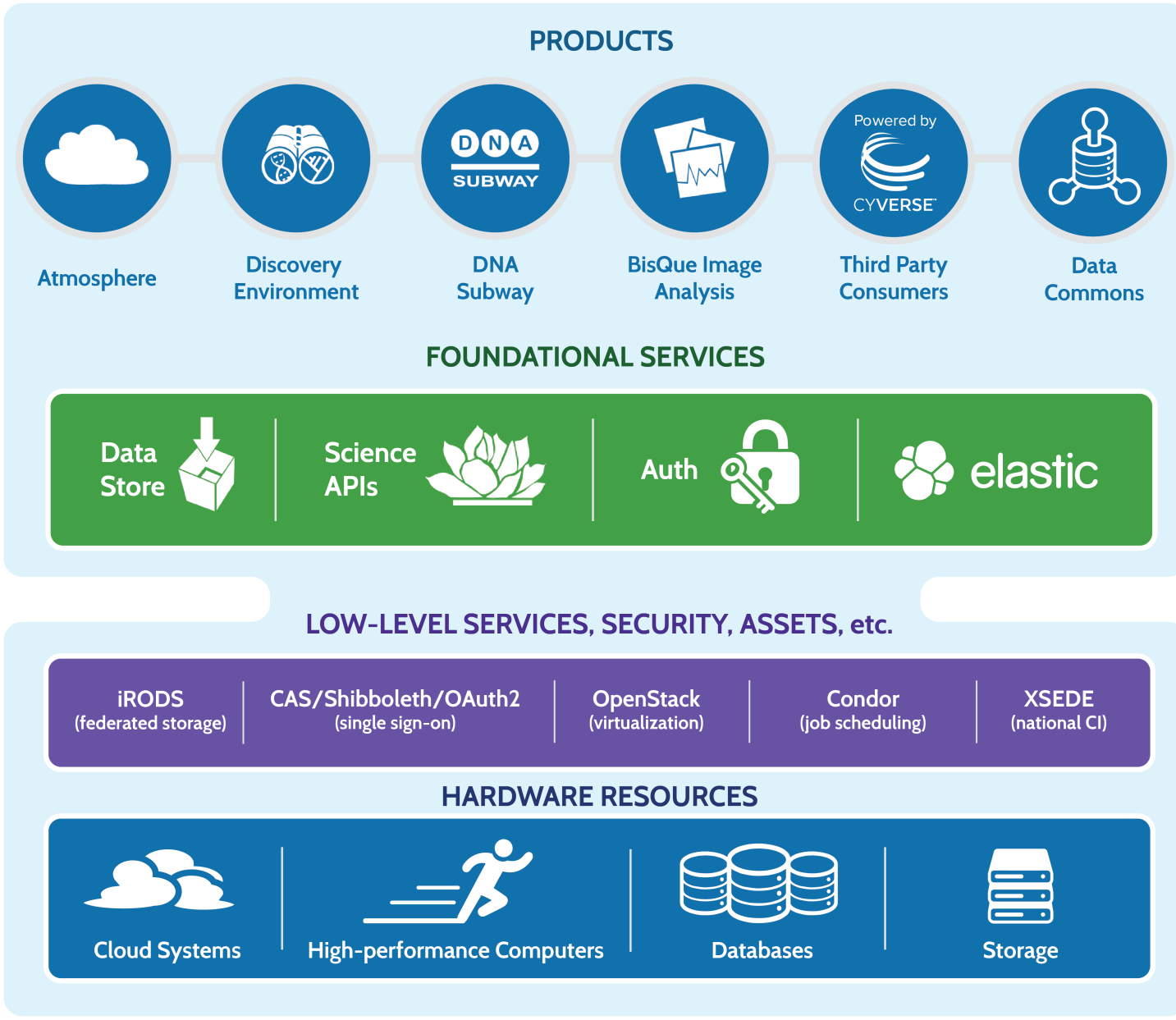
CYVERSE™

Transforming Science Through Data-driven Discovery

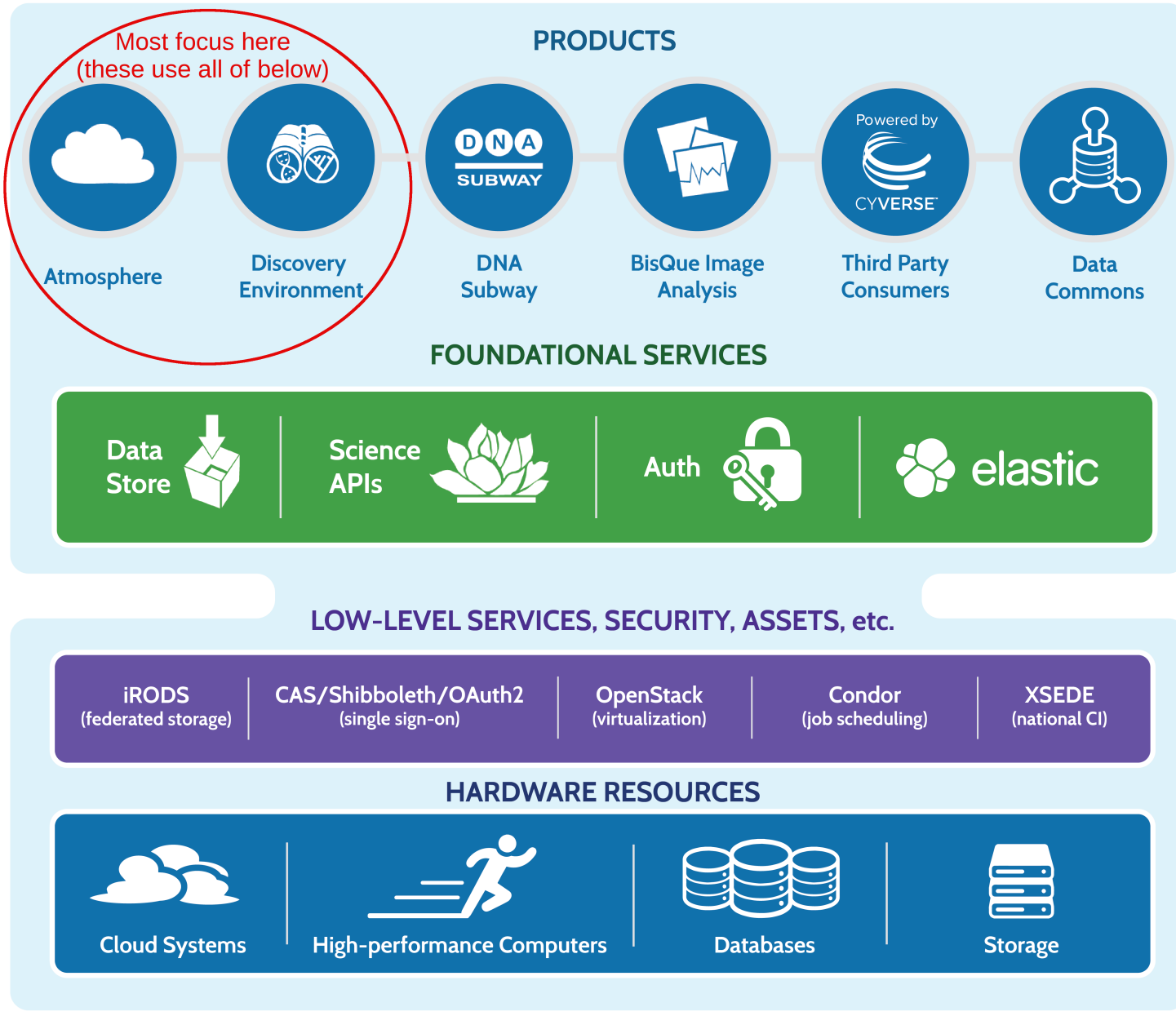
Containers & beyond at CyVerse

Ian McEwen
(Software Engineer, University of Arizona)

CyVerse high-level architecture



CyVerse high-level architecture

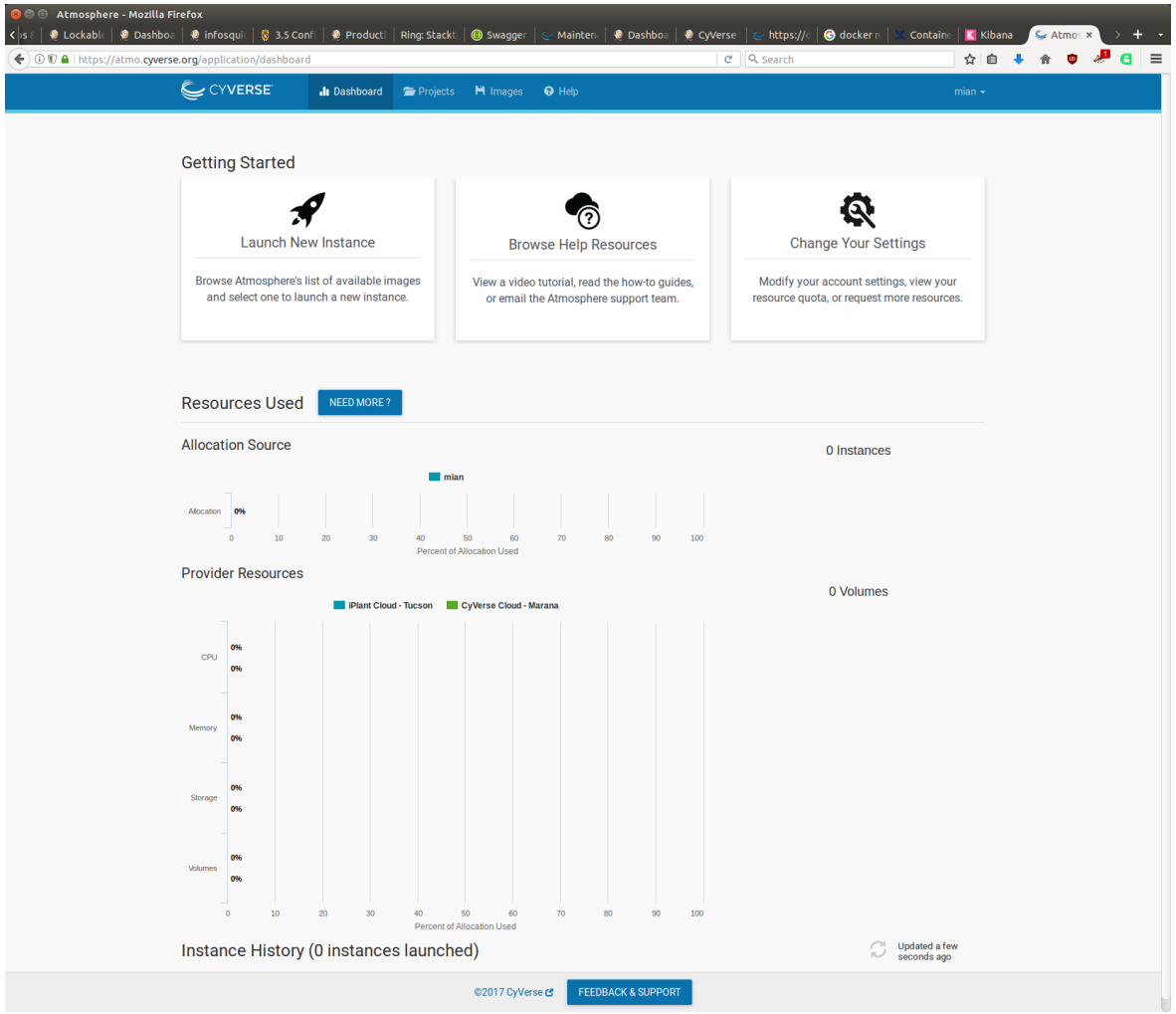


Atmosphere

Atmosphere provisions VMs for use by users; catalog of precreated images to make it easy to get started.

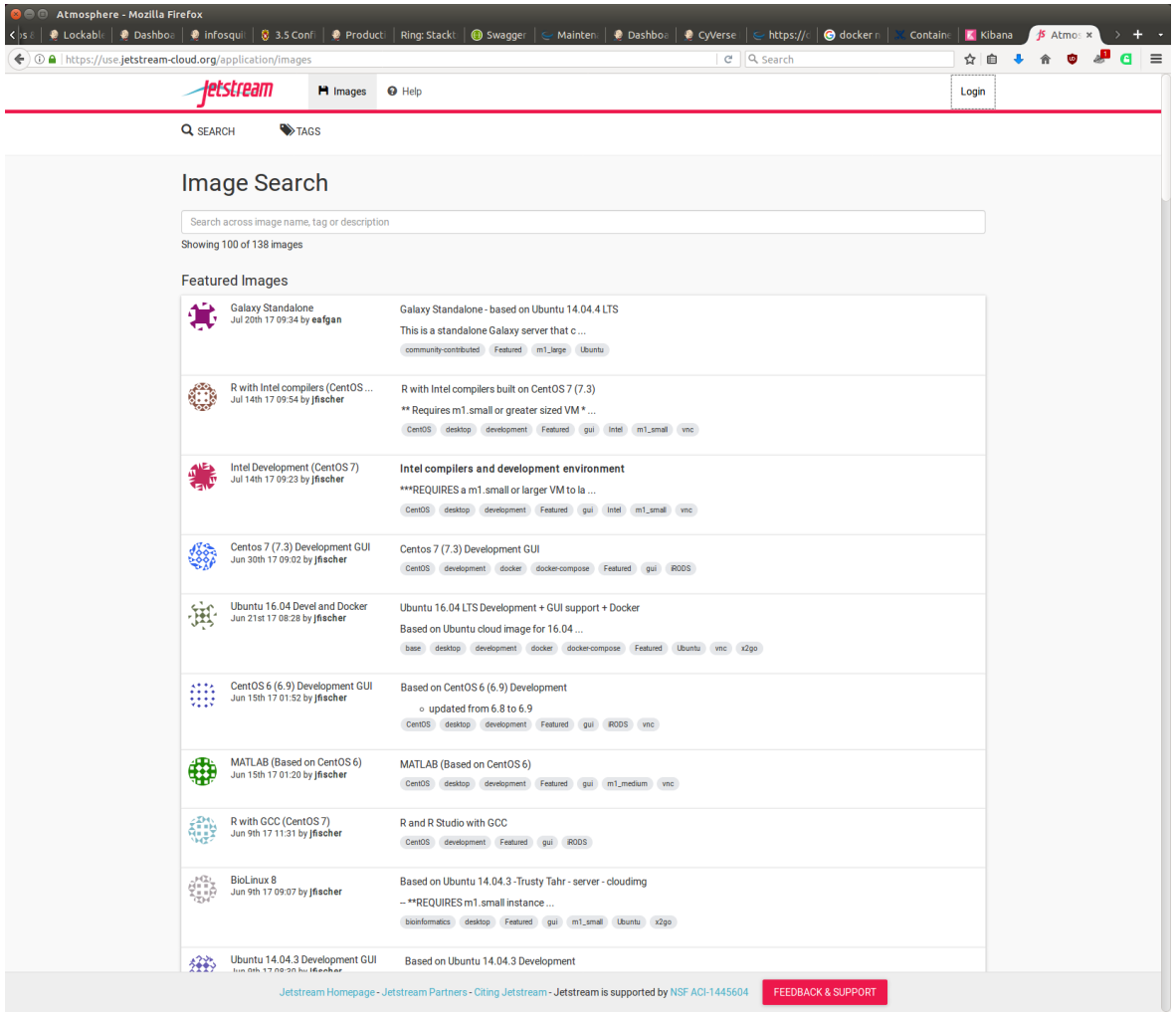
Atmosphere provides a place for technically savvy scientists and affiliates to experiment with technologies, e.g. containerization.

Data in the CyVerse Data Store can be accessed via an automatic FUSE mount and/or command line and GUI tools.



Jetstream

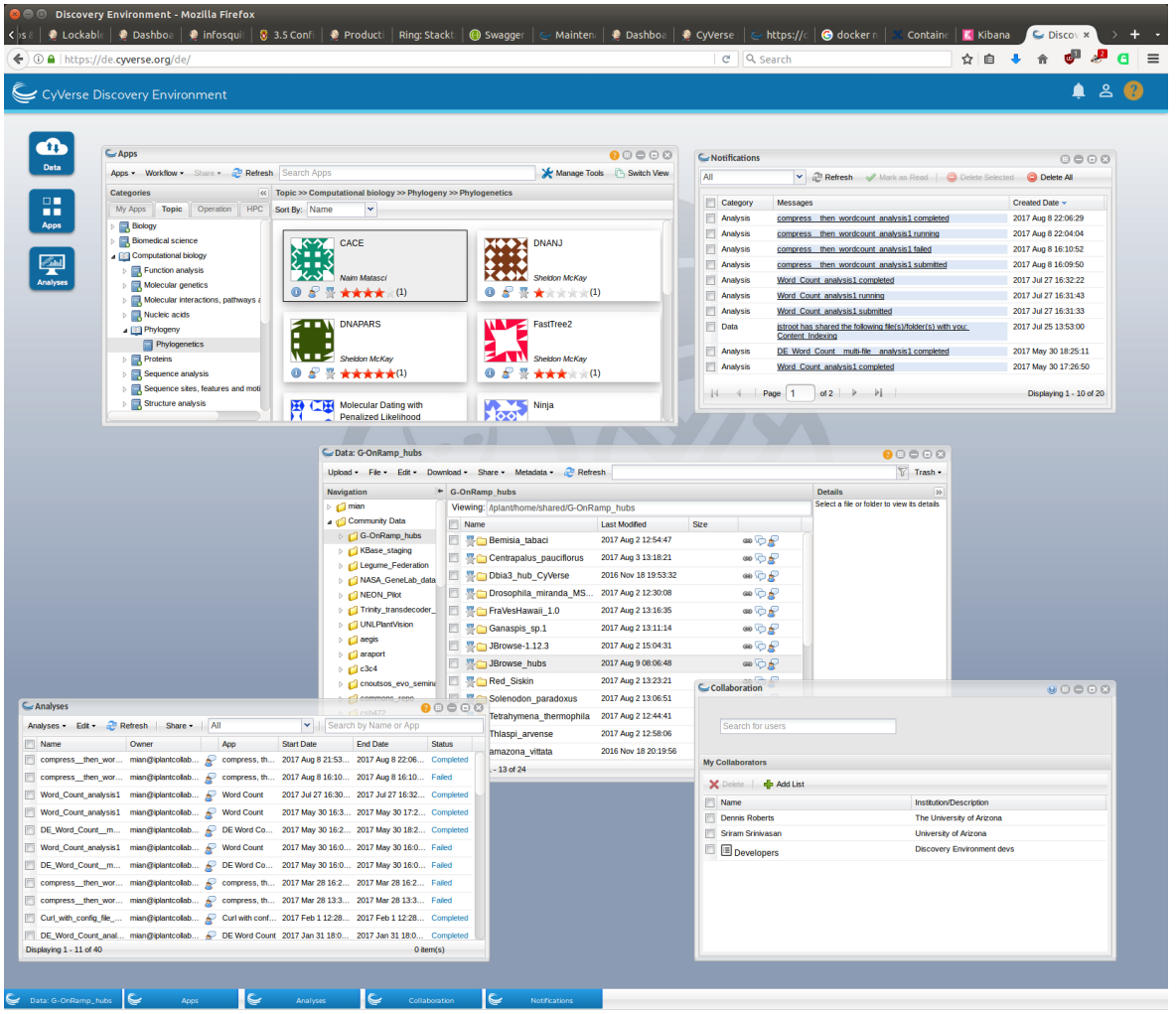
Atmosphere, but using XSEDE login/others' (TACC/Indiana University) hardware.



Discovery Environment (DE)

(the part I work on directly!)

The DE is the main workbench & entry-point into CyVerse infrastructure, aimed at less technical users and more established processes.



Managing lifecycles & pipelines

Since the DE aims to make the technical details largely transparent to the user, it broadly manages smaller and larger lifecycles and pipelines necessary to the process of doing computational work. For example:

- Data flows in, moves through various applications and analyses, management of data and metadata, eventually going on to be published with permanent identifiers in the CyVerse Data Commons or other repositories
- Apps/Analyses show a simple UI and job execution system handles staging and returning data, coordinating between systems (local vs. HPC), resource scheduling, permissions, notification, etc. This system is the main application of containerization in order to aid reproducibility, deployment, flexibility, etc.

Ongoing/future directions about expanding these ideas in many directions.

Recent & Ongoing Work

- Private Tools – resources more heavily limited, but just paste in a docker image to use (*done, just released)*
- Groups & Teams – ongoing work, being integrated at low levels like resource allocation and data permissions as well as higher levels like a concept of a project, team, etc.
- Bring your own compute – enabling users to provide computational resources (perhaps via Jetstream) into the DE for their own use, bringing the DE and its features closer to them
- Interactive jobs – spin up e.g. Jupyter & RStudio images with dependencies of relevance to the science task inside the system which already manages permissions, data staging, notification, etc.
- Better and more flexible tools for high-throughput processing, resource allocation, etc.
- Integrating deeper with the broader community, especially common repositories like bioconda/biocontainers

Perpetually, much more to talk about than time!

(Questions?)