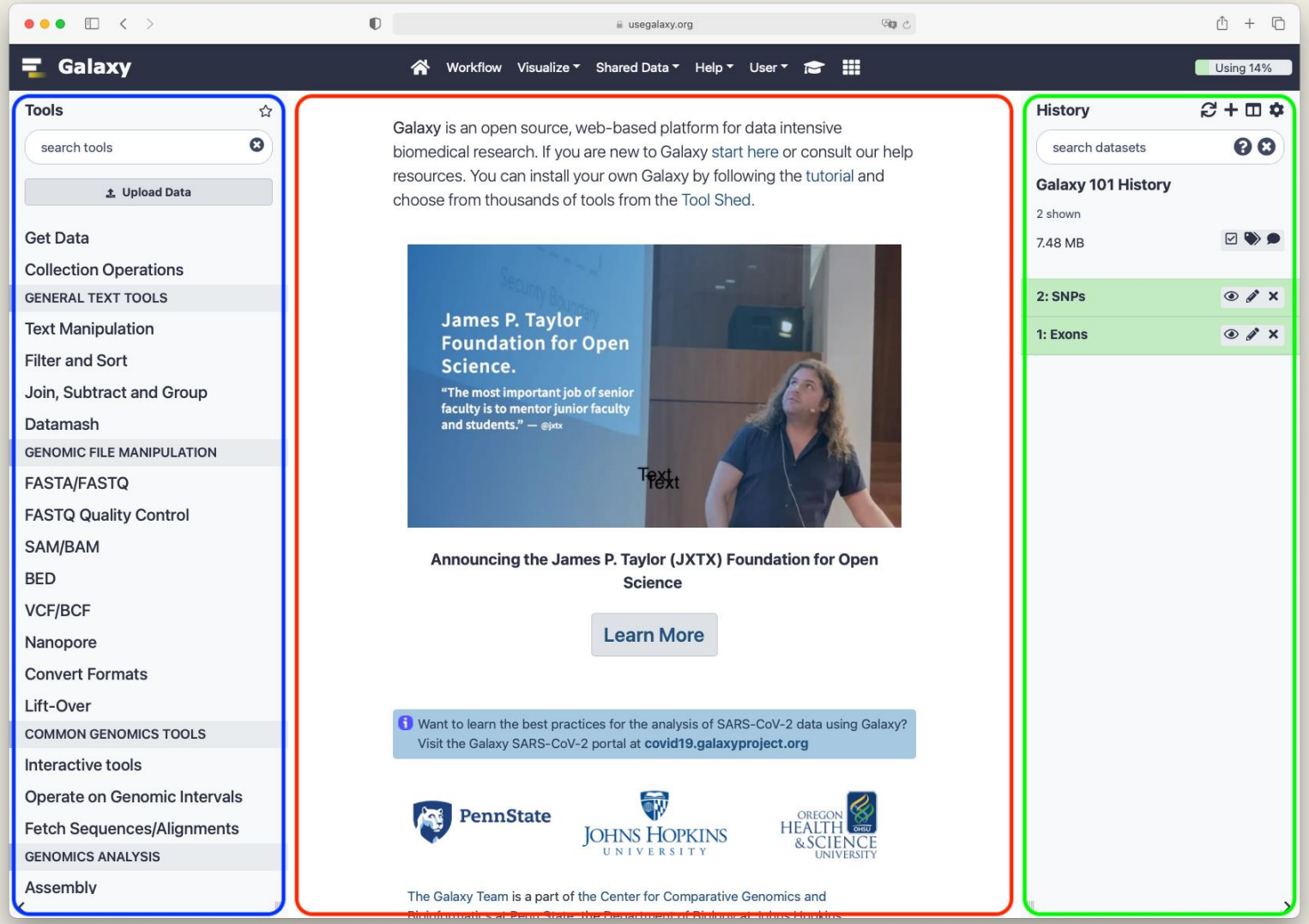
iRODS as an Object Store for the Galaxy Platform

Kaivan Kamali, Nate Coraor, Marius van den Beek, John Chilton, Anton Nekrutenko
Penn State University

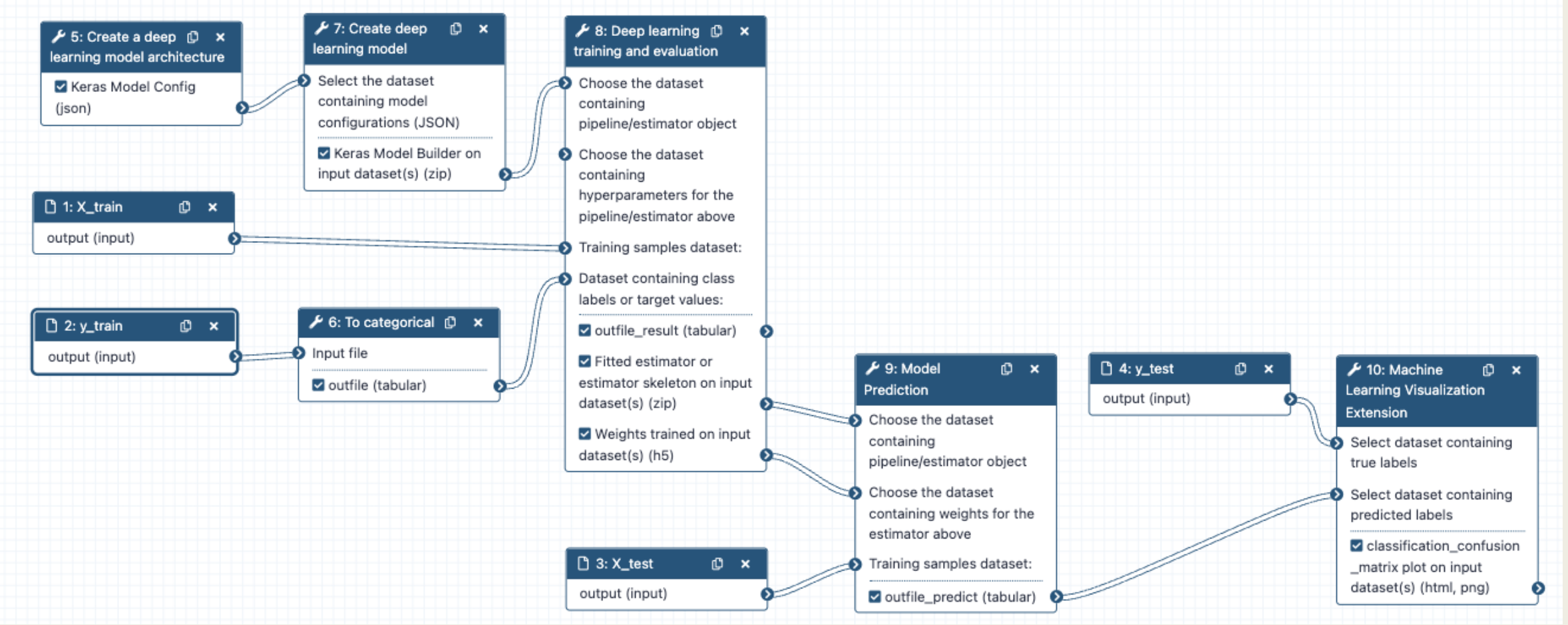
What is Galaxy?

- https://galaxyproject.org/
- Computational workbench used by thousands of scientists across the world to analyze large heterogeneous datasets, e.g., bioinformatics, chemistry, ecology, climate science, images [1]
- Easy to use [2, 3]
  - GUI based. No programming skills required
  - Web based. No system administration skills required
- Free and Open Source
- Many tools (~8000)
- Popular (>10.000 citations)
- Extensive tutorials available via Galaxy Training Network (GTN)

Galaxy Interface



Reproducible Research w/ Workflows



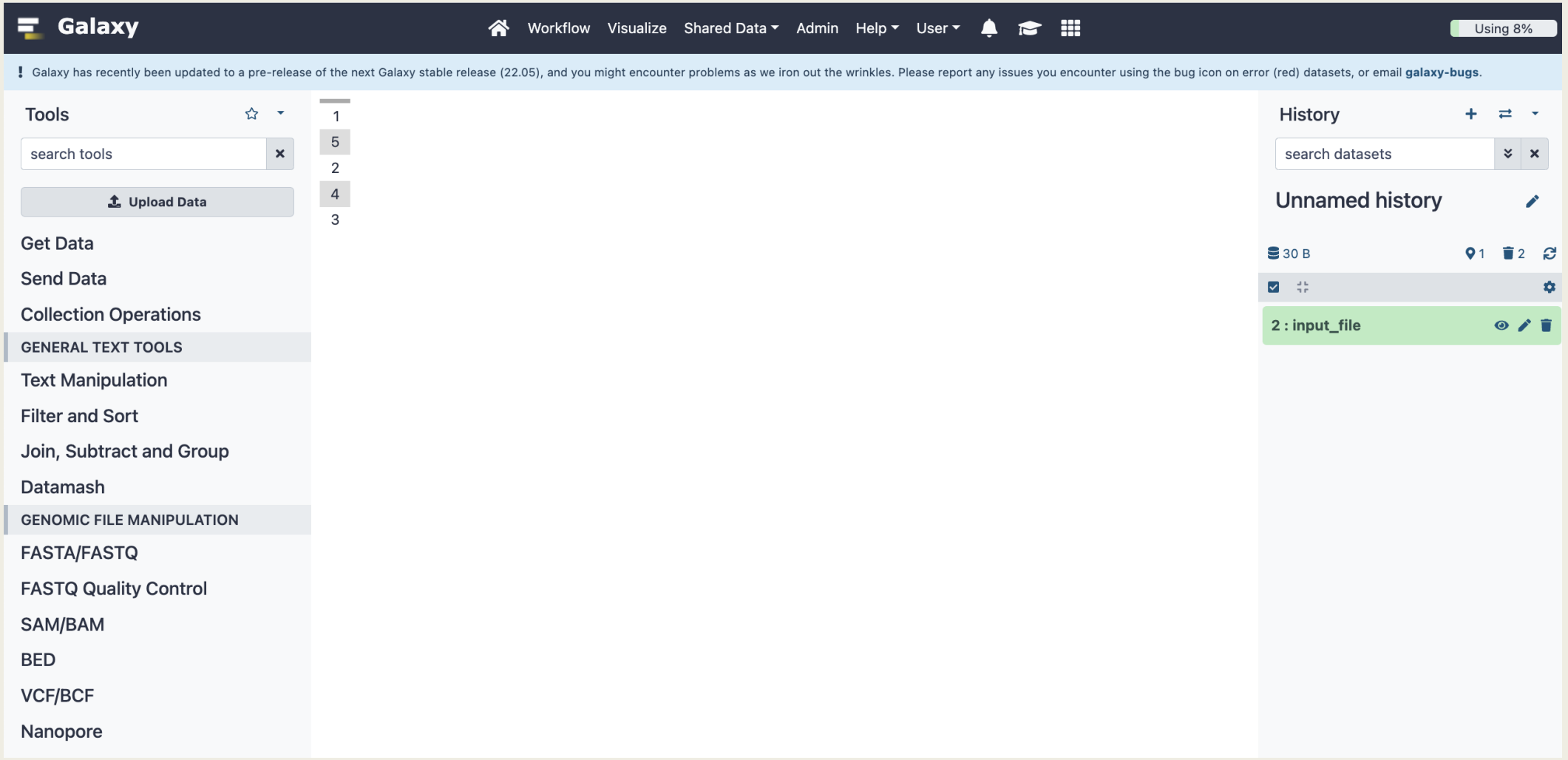
Transparency

- Galaxy Lets you share your histories, workflows, etc., enabling transparent research
- Sharing options
  - *Share with specific users*
  - *Share via link, with anyone who knows the link*
  - *Publish to make it visible to everybody*

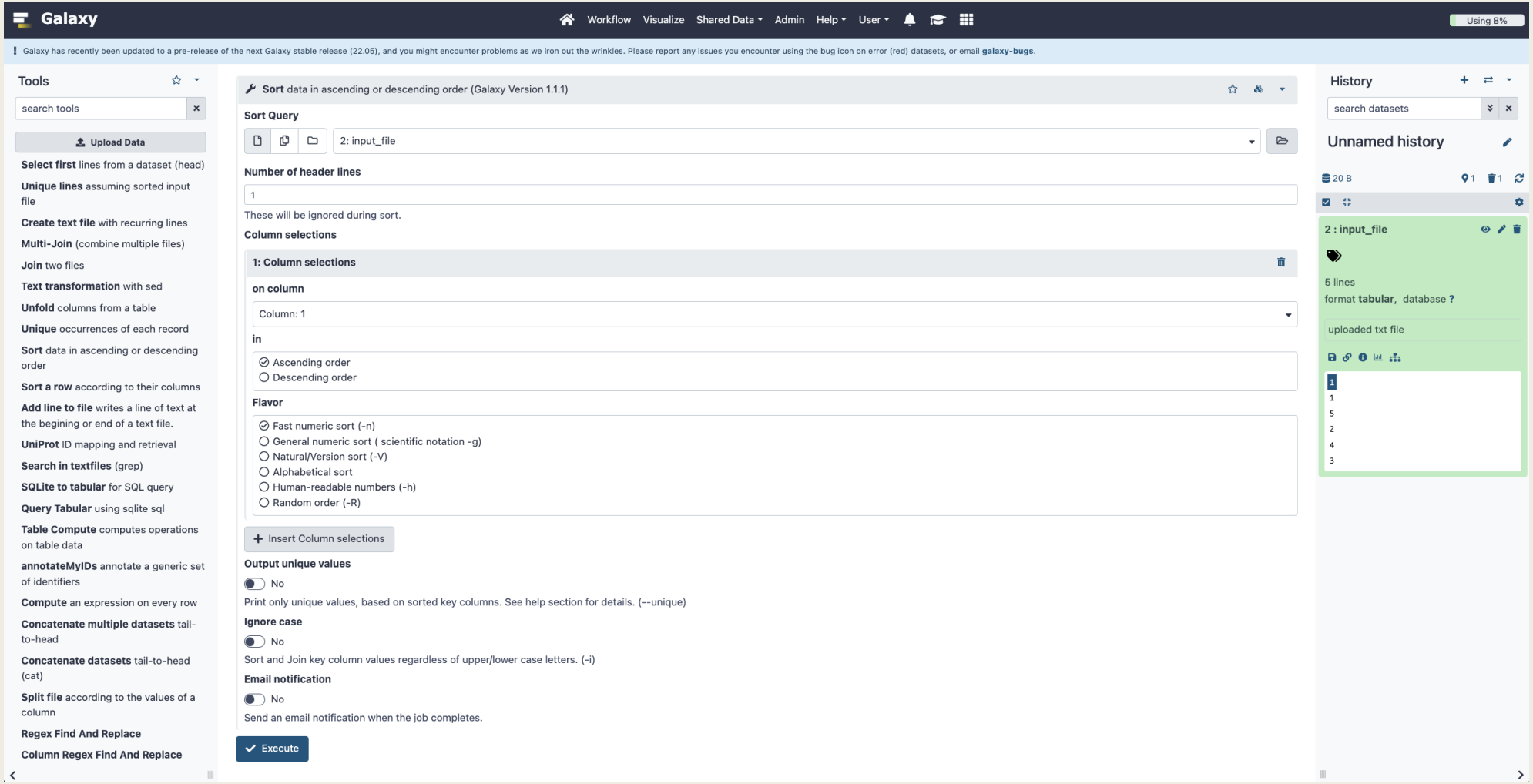
Uploading Data

- Galaxy supports data imports from
  - *The user's computer,*
  - *By URL,*
  - *And, directly from many online resources, e.g., UCSC, NCBI, etc.*
- Galaxy supports a range data formats, and translation between those formats
- The Galaxy servers provide substantial CPU and disk space
  - On usegalaxy.org, the median size of the datasets created by all users per day is 8.12 TB.

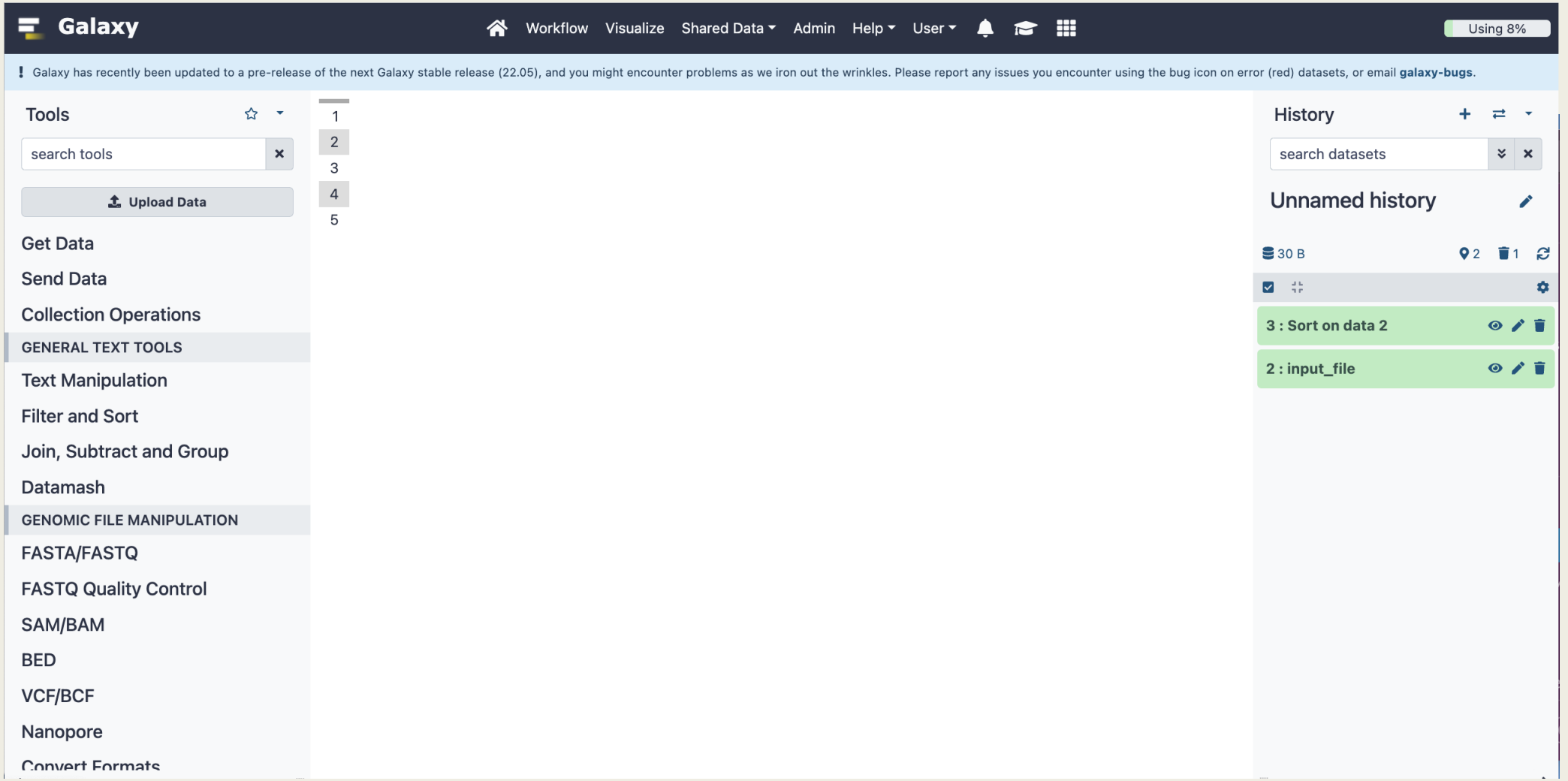
Uploaded File



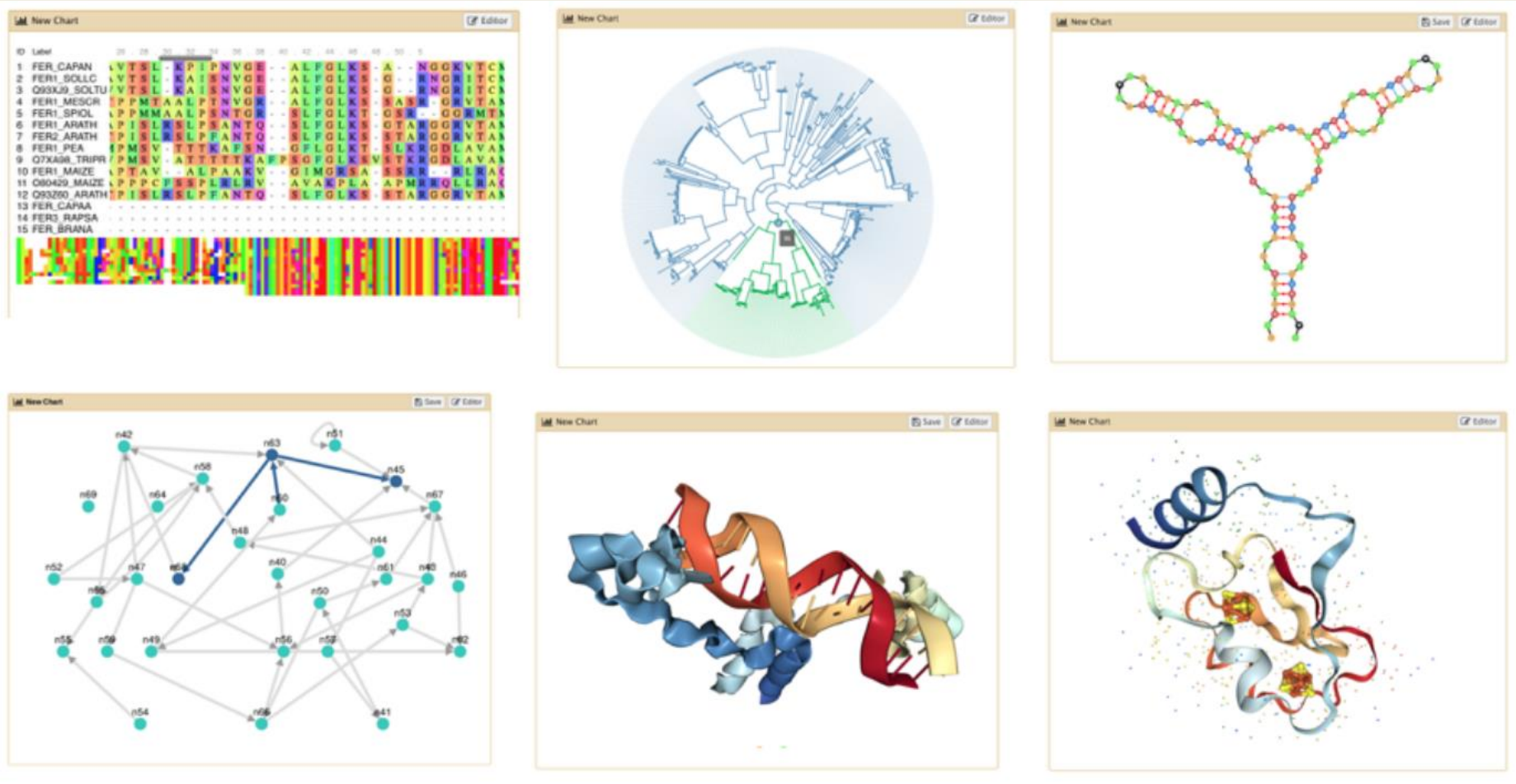
Run a Sort Tool



Output File


Galaxy
Workflow
Visualize
Shared Data
Admin
Help
User
Using 8%
Galaxy has recently been updated to a pre-release of the next Galaxy stable release (22.05), and you might encounter problems as we iron out the wrinkles. Please report any issues you encounter using the bug icon on error (red) datasets, or email galaxy-bugs.
Tools
search tools
Upload Data
Get Data
Send Data
Collection Operations
GENERAL TEXT TOOLS
Text Manipulation
Filter and Sort
Join, Subtract and Group
Datamash
GENOMIC FILE MANIPULATION
FASTA/FASTQ
FASTQ Quality Control
SAM/BAM
BED
VCF/BCF
Nanopore
Convert Formats
1
2
3
4
5
History
search datasets
Unnamed history
30 B
2
1
3 : Sort on data 2
2 : input_file

Data Visualization in Galaxy



Galaxy Servers

Galaxy Training Network (GTN)

- Collection of tutorials developed & maintained by the worldwide Galaxy community
- Tutorials for scientists, developers, and admins
- Tutorials have slides, hands on section, datasets, workflows, and videos

Galaxy ObjectStore

- ObjectStore is Galaxy's data virtualization technology
  - Abstracts Galaxy's business logic for data persistence technology and topology
  - Makes it possible to store data on a wide-variety of persistence media, and define any data distribution policy
    - Local storage, cloud-based solutions, *etc.*
- Enables Galaxy admins to add additional persistence media to existing file system
  - Expanding a mounted filesystem (e.g., when it runs out of space) w/o moving data
  - Enables replicating data onto multiple persistence media
    - Data access fault-tolerance

Galaxy Backend

- Backend: any persistence media that ObjectStore can be configured to read/write from/to
- List of backends that ObjectStore currently supports:
  - *Local storage (e.g., disk)*
  - *Network attached storage (NAS)*
  - *Cloud (e.g. S3)*
  - *Integrated Rule-Oriented Data Store (iRODS)*

Data Distribution

- When you have multiple backends, can define nested relationship between them
  - *Hierarchical backends*
  - *Distributed backends*

	Where data is read from?	Where data is written to?
Hierarchical	first backend where data exists	always the first backend
Distributed	first backend where data exists	pseudo-randomly selected backend

Data Distribution

- Hierarchical
  - Useful when you have been using a backend for a while
  - Then decide to "extend" it by adding a new backend
  - **But without having to move data from previous backend to the new backend**
- Distributed
  - Randomly selects a backend to which it should persist data
  - The random selection is based on admin-specified weights for backends

Galaxy's iRODS ObjectStore

- iRODS parameters are specified in an ObjectStore XML configuration file

```xml
<?xml version="1.0"?>
<object_store type="irods">
    <auth username="rods" password="rods" />
    <resource name="demoResc" />
    <zone name="tempZone" />
    <connection host="localhost" port="1247" timeout="30" refresh_time="30"/>
    <cache path="database/object_store_cache_irods" size="1000" />
    <extra_dir type="job_work" path="database/job_working_directory_irods"/>
    <extra_dir type="temp" path="database/tmp_irods"/>
</object_store>
```

iRODS ObjectStore Instatiation

- Galaxy reads/parses object store XML configuration file
- Instantiates an iRODS ObjectStore class
- iRODS ObjectStore class provides various methods
  - create/get/delete a file
  - get_file_name, get_file_size, get_file_path
  - is_file_empty, does_file_exists, is_file_in_cache
  - *Etc.*
- iRODS ObjectStore class uses Python iRODS Client to interact with iRODS server

Galaxy w/ iRODS Support to Test Server

- We implemented iRODS ObjectStore class
- Wrote unit/integration tests
- Had the PR reviewed/merged
- Configured Galaxy to use iRODS server
  - Hosted on Texas Advanced Computing Center (TACC)
- Deployed Galaxy server w/ iRODS support to our Test server
  - *test.galaxyproject.org*

First Challenge

- Galaxy creates and maintains a Python iRODS client Session object upon startup
  - Session object maintains a pool of connections
  - Saw occasional NetworkException errors in the Galaxy log
  - Seemed older connections would get dropped
- Initial solution
  - Per discussions on iRODS-Chat (https://groups.google.com/g/iROD-Chat) created a Session object for each interaction with iRODS
    - The NetworkException errors disappeared
  - However, creating a Session object for each interaction with iRODS was not performant
    - Creating/destroying thousands of iRODS Session objects was costly

First Challenge -- Continued

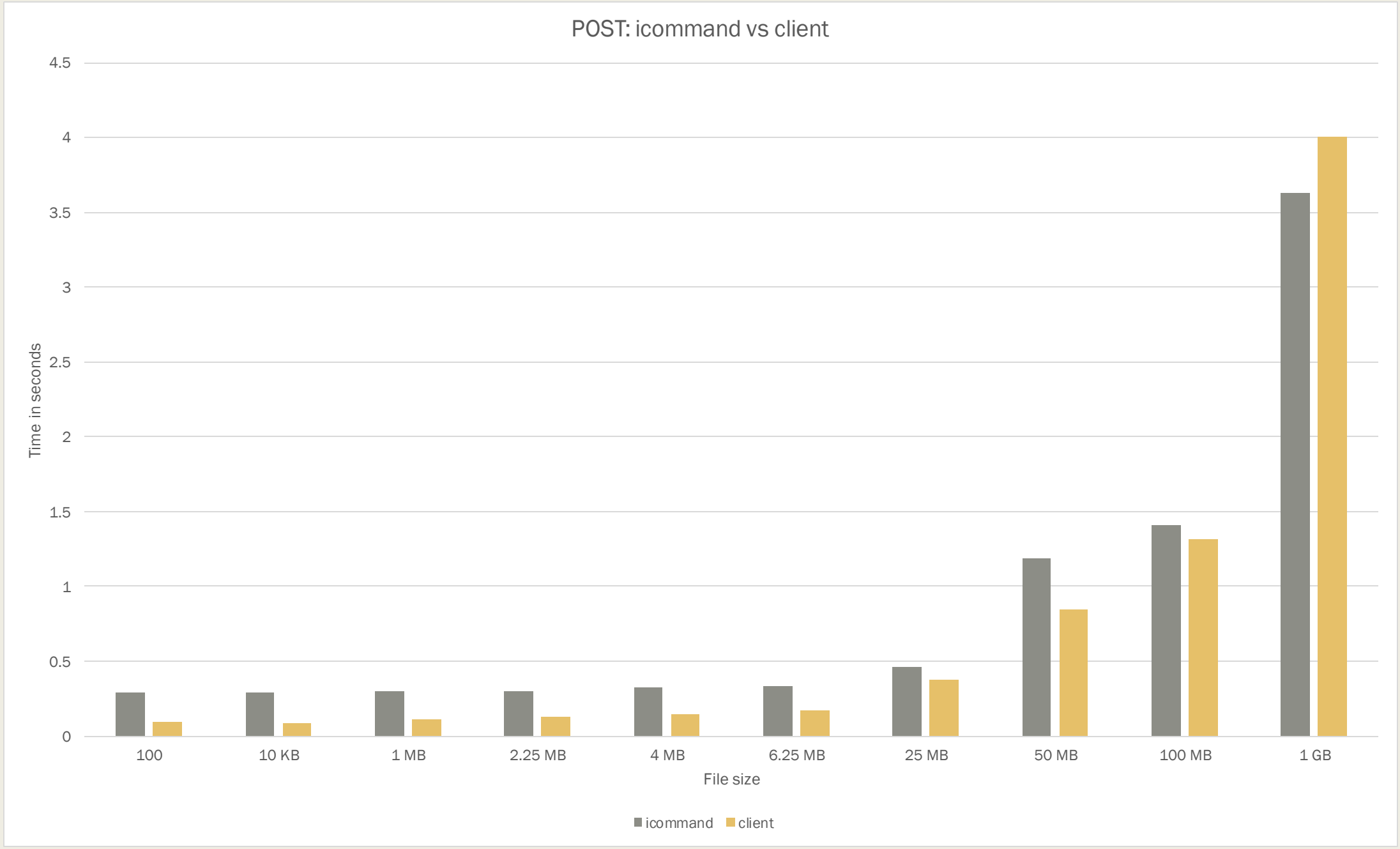
- Subsequent solution
  - Maintain the session object
    - Do not create/destroy the session object for each iRODS interaction
  - Record connection creation time for each connection in the Pool
  - When interacting with iRODS, if the connection was created more than *N* seconds ago (*N* configurable, default value is 300 seconds), recreate the connection
    - Otherwise, use the connection as is
  - This resulted in NetworkException errors disappearing
  - And, the solution is also performant

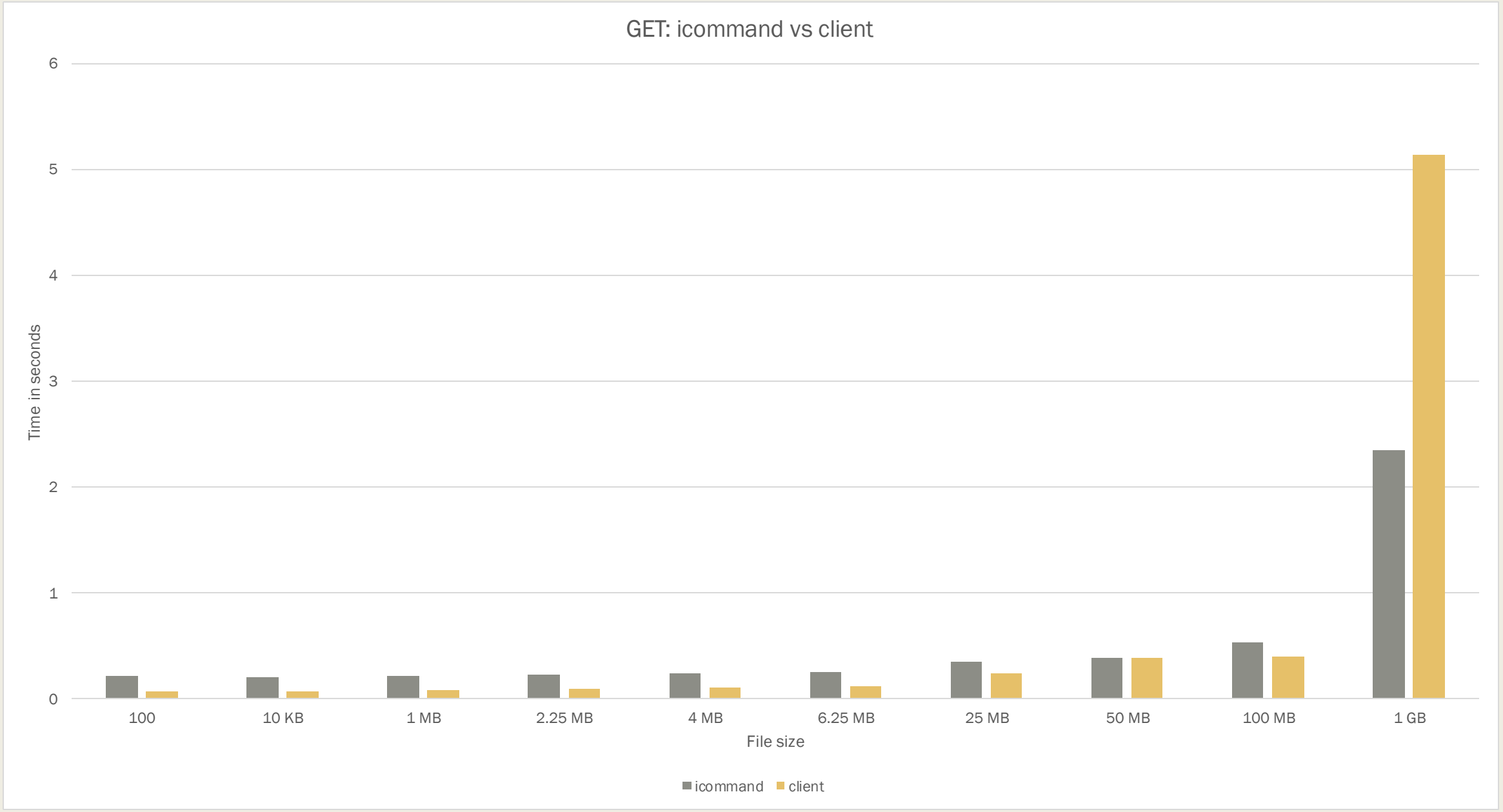
Second Challenge

- Python iRODS client's Post and Get methods were not performant
  - Slowed down Galaxy Post/Get operations
  - Specially for larger files
- The client's Post/Get runtime was much slower than iCommands equivalent
- Proposed solution
  - Discussed supporting multi-threaded Post/Get on the client side only
    - Eventually decided to scrap client only solution
  - Decided on multi-threaded Post/Get support on both client AND server sides
    - Python iRODS client: v0.9.0+
    - iRODS server: v4.2.9+
    - Data object transfers using put()/get() will spawn a number of threads to optimize performance
    - File sizes larger than a default threshold value of 32 MB

Python iRODS Client Vs iCommands

- Wrote scripts that generate M (say, 50) files of size N (say, 100 GB) with random content
- Post files to iRODS server using both Python iRODS client and iCommands
- Averaged the runtime of each to compare
- Get files from iRODS server using both Python iRODS client and iCommands
- Averaged the runtime of each to compare
- Repeated the steps above for file sizes 100 bytes, 10 KB, 1 MB, 2.25 MB, 4 MB, 6.25 MB, 25 MB, 50 MB, 100 MB, and 1 GB


POST: icommand vs client
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
Time in seconds
100
10 KB
1 MB
2.25 MB
4 MB
6.25 MB
25 MB
50 MB
100 MB
1 GB
File size
icommand
client


GET: icommand vs client
6
5
4
3
2
1
0
Time in seconds
100
10 KB
1 MB
2.25 MB
4 MB
6.25 MB
25 MB
50 MB
100 MB
1 GB
File size
icommand
client

Third Challenge

- Galaxy server became unresponsive
  - Call stack showed it's hung in Python iRODS client code
- After extensive debugging, found a bug in client code (in _recv_message_in_len())
  - The loop that iteratively reads from the socket, depended on the number of bytes read each time, to terminate the loop
  - If the iRODS server is down, the number of bytes read in each iteration of the loop is 0, causing it to loop indefinitely, halting the app
  - Revised the logic to account for this scenario
- Also, found and fixed a bug with Connection destructor
  - Memory leak fixed

Current Status

- Configured/deployed Galaxy with iRODS object store to Main (https://usegalaxy.org)
- Created a group of users that their object store access is overridden to iRODS
  - This is to gradually release iRODS to only a select few
  - Testing has been promising with no performance issues
- After this stage of our testing is complete, we plan to use a distributed object store
  - We use weights to split the writes between iRODS and disk
    - Say, start with 1 write to iRODS, for every 9 writes to disk
    - Reads go against both backends, depending on where the data resides

Current Status -- Continued

- We plan to incrementally increase iRODS weights and decrease disk weights
  - Say, 10% every month
  - Until all writes go to iRODS
    - Again, reads go against both backends, depending on where the data resides
  - *Can a real-time multi-user application run on top of iRODS?*
- Finally, we plan on migrating the data from disk to iRODS
  - Data migration happens via a separate script
  - Can then retire disk object store

Thank you!

- We would like to thank all the members of the iRODS team for their support and always being available
  - Special thanks to Daniel and Terrell!

- Questions/Comments?

References

1. Vahid Jalili, et. al. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2020 update, *Nucleic Acids Research*, Volume 48, Issue W1, 02 July 2020, Pages W395–W402, https://doi.org/10.1093/nar/gkaa434
2. A Short Introduction to Galaxy (https://training.galaxyproject.org/training-material/topics/introduction/tutorials/galaxy-intro-short/slides.html#1)
3. Introduction to Galaxy. https://training.galaxyproject.org/training-material/topics/introduction/slides/introduction.html#1
4. Galaxy ObjectStore. https://galaxyproject.org/admin/objectstore/
5. Dockerized iRODS Server. https://github.com/kxk302/irods