NIH HPC Object storage system overview

Tim Miller
NIH HPC Systems Staff
btmiller@helix.nih.gov
https://hpc.nih.gov
January 9, 2018

Outline

- Overview of the object storage
- A first practical example
- When would you want to use object storage?
- How do you get access to the object storage?
- Using the NIH HPC object storage
  - HPC staff developed tools
  - Rclone
- Programming your own tools

Outline

- Overview of the object storage
- A first practical example
- When would you want to use object storage?
- How do you get access to the object storage?
- Using the NIH HPC object storage
  - HPC staff developed tools
  - Rclone
- Programming your own tools

Basics of object storage

- “Web Scale” storage
  - Highly reliable (dispersed over multiple sites)
  - Easy to expand (just add more disks)
  - Accessed via simple list, put, get, delete semantics (examples forthcoming)
- Different from file based storage systems
  - Objects are accessed by **NAME**, not **PATH**
  - Completely flat name space
  - No concept of directories, but “/” is a valid character in object names
  - Data and metadata are stored together with the object (sometimes true in file storage systems as well)

The NIH HPC object storage system

- https://hpc.nih.gov/storage/object.html
- Dedicated for use by NIH HPC system (helix, biowulf) users
- Accessible from Helix, Biowulf, and compute nodes
  - No Globus or Helixdrive (more on this later)
- Geographically distributed (B12, Shady Grove)
  - However no off-site back-ups (tape or otherwise)

Components of the system

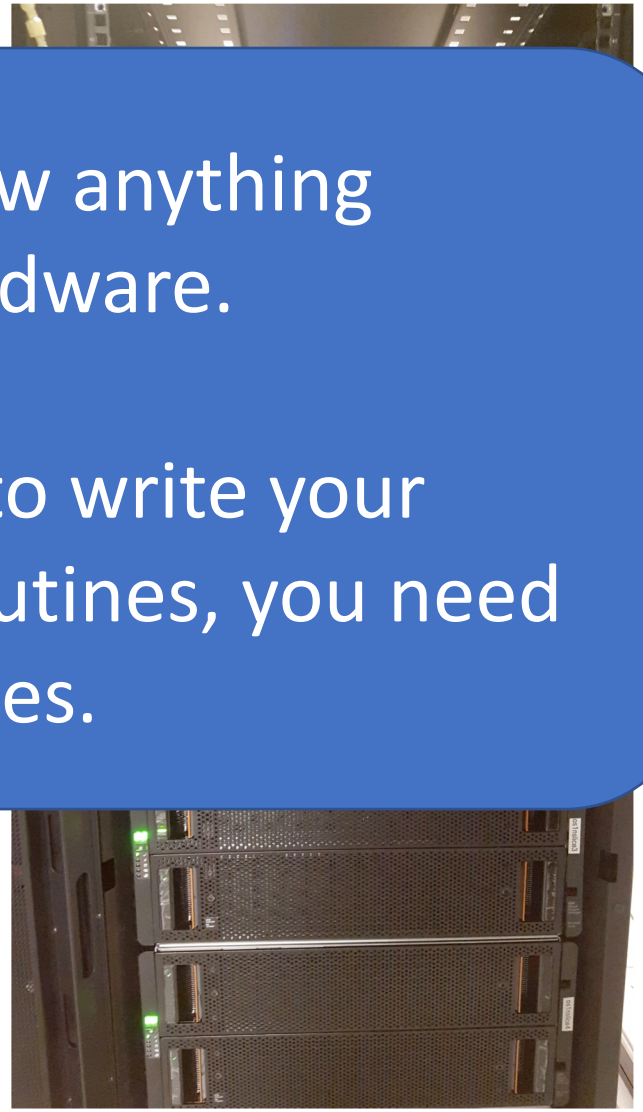
- Manager
  - Staff interacts with it
  - Used for provisioning, monitoring, etc.
- **Accessor**
  - Primary point of user interaction
- Storage server
  - Actually holds data
  - No direct user interaction

Components of the system

- Mana
  - St
  - U
    m
- **Acce**
  - Pr
    in
- Stora
  - Actu
  - No direct user interaction

You don’t need to know anything about the physical hardware.

However, if you want to write your own custom access routines, you need the accessors’ addresses.

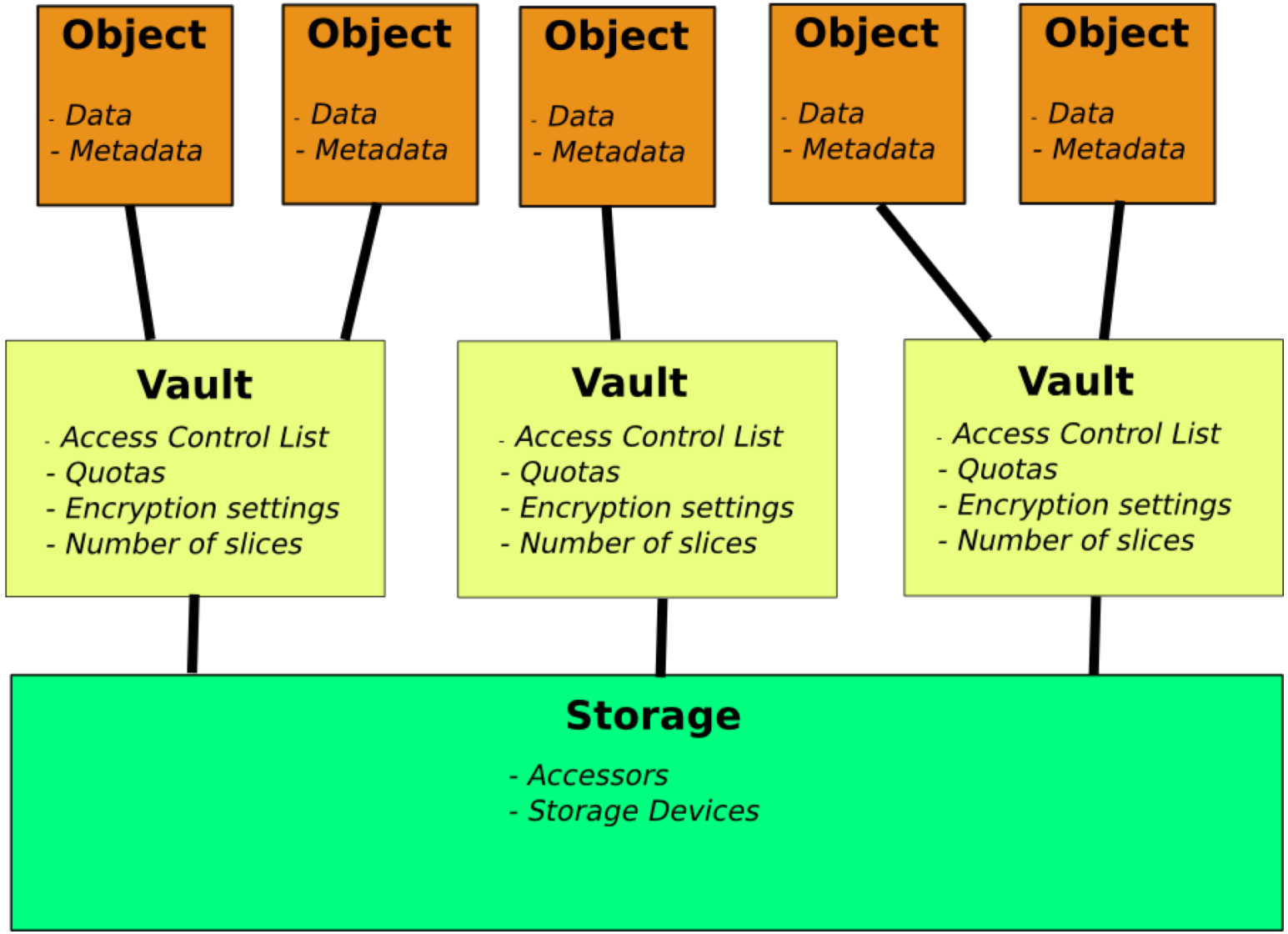


Manager

Accessors

Storage servers

Logical View



Logical view components (1)

- Each user has access to one or more **VAULTS**
  - These are like buckets, for those familiar with Amazon S3, except they are created by system administrators.
- The vaults are containers for objects
- Objects contain both data and metadata (we'll see examples)
  - Physically, objects are divided up and different parts are sent to multiple different storage servers
  - The system is designed to be able to lose a certain number of storage servers and still be able to reconstruct objects.

Logical view components (2)

- I/O operations are performed via **accessors**
  - Use S3 operations layered on top of the HTTP protocol
  - Six accessors: os{1,2}naccess{1,2,3}
  - A RESTful API (**RE**presentational **S**tate **T**ransfer) – see next bullet point!
  - When configuring some programs to access the object store, <u>you must</u> specify which accessor to use
    - We'll see examples later
- Take a REST!
  - A RESTful API handles <u>transactions</u>
  - PUT, GET, DELETE
  - No multiple-part requests!
- Pre-written programs (and your own)

Outline

- Overview of the object storage
- A first practical example
- When would you want to use object storage?
- How do you get access to the object storage?
- Using the NIH HPC object storage
  - HPC staff developed tools
  - Rclone
- Programming your own tools

Our first example: just showing off

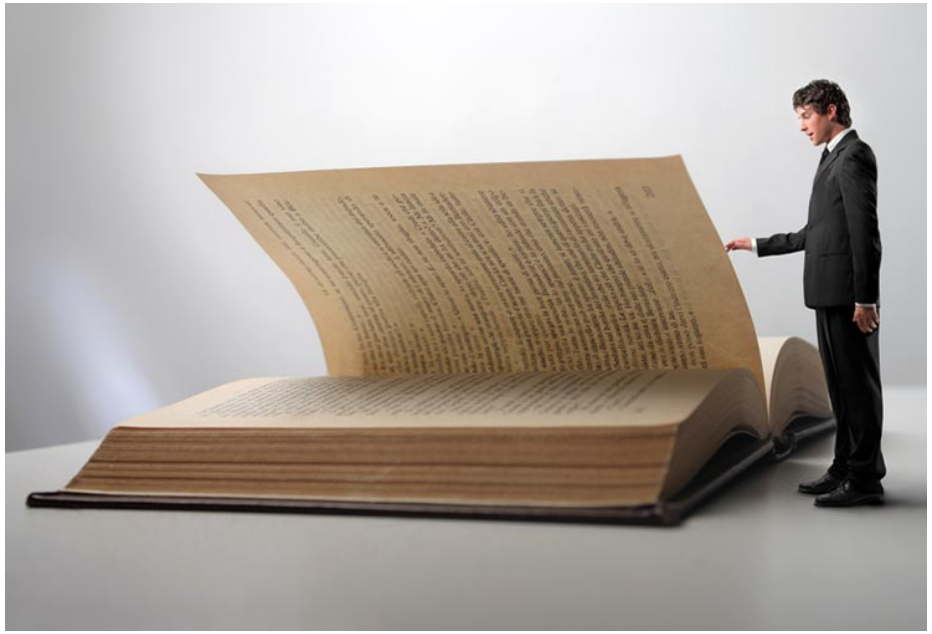
- Key commands
  - obj_df
  - obj_ls
  - obj_put
  - obj_get
- Key take-aways
  - Object storage is not accessed like disk storage
  - We have to use special tools (or write our own)
  - Programs need to be directly aware of the object store to use it, or files must be staged to an intermediate location.

Outline

- Overview of the object storage
- A first practical example
- When would you want to use object storage?
- How do you get access to the object storage?
- Using the NIH HPC object storage
  - HPC staff developed tools
  - Rclone
- Programming your own tools

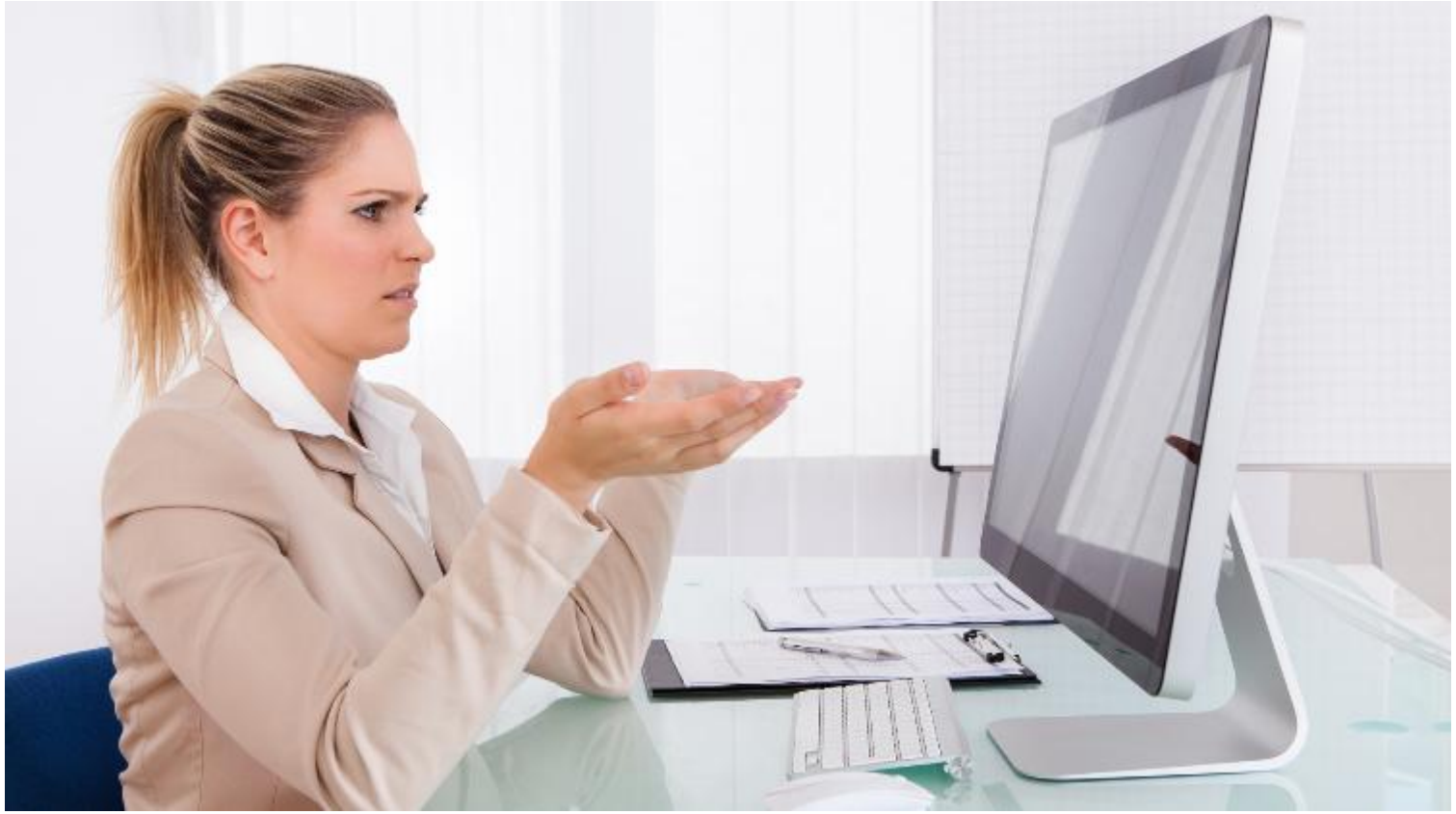
Use-cases for object storage

- Read-intensive workloads
  - Object storage is much more efficient at reading than writing.
  - An **entire** object has to be re-written for each change
    - Computationally expensive to process and disperse the data
    - Lots of over-writing
- Static data
  - Related to the above
  - Data that doesn’t change often, but still used
  - E.g. reference genomics files
- **LIMITED** archiving
  - Can’t guarantee data will remain forever.
  - E.g. intermediate files



When not to use object storage

- Content of data changes frequently (database updates)
- Data will not need to be repeatedly read (use scratch or lscratch)
- Only need to read part of each unit of data (object store will read the whole thing usually)
- Reads are performance critical



Rules, policies, and archiving

- ALL NIH HPC policies that apply to other HPC storage systems apply to the object store
  - No personally-identifiable information (PII)
  - No personal health information (PHI)
  - Archiving OK, but it's time-limited and should be discussed with HPC staff
- NO back-ups or snapshots
  - If you delete something from the object store, it's gone (unless you have another copy somewhere).
  - Likewise, if you overwrite an object, the original copy is unrecoverable.
- Reduce metadata operations
  - As much as possible, avoid listing vault contents – it's slow!
  - Use a regular scheme for naming objects or keep an off-line index

Outline

- Overview of the object storage
- A first practical example
- When would you want to use object storage?
- How do you get access to the object storage?
- Using the NIH HPC object storage
  - HPC staff developed tools
  - Rclone
- Programming your own tools

Requesting object storage

https://hpc.nih.gov/nih/object_request.html

- Accessible from within the NIH network (and VPN).
- If off-campus, e-mail staff@hpc.nih.gov.

HPC Object Storage Request

This form should be completed by NIH HPC users who wish to use the HPC object storage system or who wish to increase their object store quota.

The HPC object storage system is a large, highly-scalable, and highly reliable data storage system. However, it has substantially lower performance than the storage systems that serve /home and /data. In addition, it is accessed in a completely different manner than more traditional disk-based systems. **To access object storage, you must use special utilities that can communicate via the S3 protocol.** For more information, please see the HPC object storage page.

Quick Links
Storage on Biowulf & Helix
The NIH HPC object storage system
Request form for non-object storage
Request form for non-object shared data

The primary use case for the HPC object storage is for data that changes infrequently but still needs to be accessed for use in calculations (e.g. reference sequence files used in genomic analysis) or for future scientific work. Although there are no time restrictions on the use of HPC object storage space, users are expected to clean up data that is no longer necessary for ongoing projects. **Disk space on Helix and Biowulf, including on the object storage system, should never be used as archival storage.**

User Information:

Name:

Helx/Biowulf username:

Telephone:

Principal Investigator:

Brief description (one or two lines) of your research project:

Object storage vault name (if different than your user name):

Size of vault (if new) or quota increase (if existing):

How this space requirement was estimated and why your workload would benefit from object storage.
Example: I have 400 BAM files that will be written once and read numerous times by different batch jobs. Each BAM file takes 30 GB of space. Therefore, I need 12 TB of space on the object storage. Object storage is ideal for my workload because the data, once written, is static, and my analysis pipeline supports data access via the S3 protocol.

By clicking the check box below, you agree that you understand that object storage is different from disk storage in terms of how it is accessed and used. You acknowledge that you have read the documentation on the HPC object storage system and determined that your intended use is a good fit for the object storage system. You further acknowledge that you **cannot** use standard Unix/Linux utilities (e.g. ls, cat, vi, emacs) to interact with its data, but instead must use special utilities (e.g. the NIH HPC object tools, Rclone, S3 utilities, etc.) to interact with data on the object storage system. If you are not sure whether object storage would benefit your workload, please contact staff@hpc.nih.gov to discuss your proposed usage.

I have read the above (really!) and agree to it:

SUBMIT CLEAR

Standard information about yourself and what you want the object storage allocation for.

Search:

HPC @ NIH

High-Performance Computing at the NIH

Status | Applications | Storage | User Guides | User Dashboard | How To | About

HPC Object Storage Request

This form should be completed by NIH HPC users who wish to use the HPC object storage system or who wish to increase their object store quota.

The HPC object storage system is a large, highly-scalable, and highly reliable data storage system. However, it has substantially lower performance than the storage systems that serve /home and /data. In addition, it is accessed in a completely different manner than more traditional disk-based systems. **To access object storage, you must use special utilities that can communicate via the S3 protocol.** For more information, please see the HPC object storage page.

Quick Links

- Storage on Biowulf & Helix
- The NIH HPC object storage system
- Request form for non-object storage
- Request form for non-object shared data

The primary use case for the HPC object storage is for data that changes infrequently but still needs to be accessed for use in calculations (e.g. reference sequence files used in genomic analysis) or for future scientific work. Although there are no time restrictions on the use of HPC object storage space, users are expected to clean up data that is no longer necessary for ongoing projects. **Disk space on Helix and Biowulf, including on the object storage system, should never be used as archival storage.**

User Information:

Name:

Helx/Biowulf username:

Telephone:

Principal Investigator:

Brief description (one or two lines) of your research project:

Object storage vault name (if different than your user name):

Size of vault (if new) or quota increase (if existing):

How this space requirement was estimated and why your workload would benefit from object storage.
Example: I have 400 BAM files that will be written once and read numerous times by different batch jobs. Each BAM file takes 30 GB of space. Therefore, I need 12 TB of space on the object storage. Object storage is ideal for my workload because the data, once written, is static, and my analysis pipeline supports data access via the S3 protocol.

By clicking the check box below, you agree that you understand that object storage is different from disk storage in terms of how it is accessed and used. You acknowledge that you have read the documentation on the HPC object storage system and determined that your intended use is a good fit for the object storage system. You further acknowledge that you **cannot** use standard Unix/Linux utilities (e.g. ls, cat, vi, emacs) to interact with its data, but instead must use special utilities (e.g. the NIH HPC object tools, Rclone, S3 utilities, etc.) to interact with data on the object storage system. If you are not sure whether object storage would benefit your workload, please contact staff@hpc.nih.gov to discuss your proposed usage.

I have read the above (really!) and agree to it:

SUBMIT CLEAR

Last modified: 9 January 2017

Unlike your data directory, you can choose any name for your vault (within reason). If you leave this blank, the vault name will be the same as your user name.

You also have to tell us how big you would like your vault. We generally will not give out more than 20 TB until you show that you can make effective use of it.

Search:

HPC @ NIH

High-Performance Computing at the NIH

Status | Applications | Storage | User Guides | User Dashboard | How To | About

HPC Object Storage Request

This form should be completed by NIH HPC users who wish to use the HPC object storage system or who wish to increase their object store quota.

The HPC object storage system is a large, highly-scalable, and highly reliable data storage system. However, it has substantially lower performance than the storage systems that serve /home and /data. In addition, it is accessed in a completely different manner than more traditional disk-based systems. **To access object storage, you must use special utilities that can communicate via the S3 protocol.** For more information, please see the HPC object storage page.

Quick Links

- Storage on Biowulf & Helix
- The NIH HPC object storage system
- Request form for non-object storage
- Request form for non-object shared data

The primary use case for the HPC object storage is for data that changes infrequently but still needs to be accessed for use in calculations (e.g. reference sequence files used in genomic analysis) or for future scientific work. Although there are no time restrictions on the use of HPC object storage space, users are expected to clean up data that is no longer necessary for ongoing projects. **Disk space on Helix and Biowulf, including on the object storage system, should never be used as archival storage.**

User Information:

Name:

Helx/Biowulf username:

Telephone:

Principal Investigator:

Brief description (one or two lines) of your research project:

Object storage vault name (if different than your user name):

Size of vault (if new) or quota increase (if existing):

How this space requirement was estimated and why your workload would benefit from object storage.

GB of space. Therefore, I need 12 TB of space on the object storage. Object storage is ideal for my workload because the data, once written, is static, and my analysis pipeline supports data access via the S3 protocol.

By clicking the check box below, you agree that you understand that object storage is different from disk storage in terms of how it is accessed and used. You acknowledge that you have read the documentation on the HPC object storage system and determined that your intended use is a good fit for the object storage system. You further acknowledge that you **cannot** use standard Unix/Linux utilities (e.g. ls, cat, vi, emacs) to interact with its data, but instead must use special utilities (e.g. the NIH HPC object tools, Rclone, S3 utilities, etc.) to interact with data on the object storage system. If you are not sure whether object storage would benefit your workload, please contact staff@hpc.nih.gov to discuss your proposed usage.

I have read the above (really!) and agree to it:

SUBMIT | CLEAR

Last modified: 9 January 2017

For your justification, be sure to specify why you are requesting object storage rather than disk storage. Letting us know whether you plan to use your own programs or HPC developed or installed tools is also helpful.

Search:

HPC @ NIH

High-Performance Computing at the NIH

Status | Applications | Storage | User Guides | User Dashboard | How To | About

HPC Object Storage Request

This form should be completed by NIH HPC users who wish to use the HPC object storage system or who wish to increase their object store quota.

The HPC object storage system is a large, highly-scalable, and highly reliable data storage system. However, it has substantially lower performance than the storage systems that serve /home and /data. In addition, it is accessed in a completely different manner than more traditional disk-based systems. **To access object storage, you must use special utilities that can communicate via the S3 protocol.** For more information, please see the HPC object storage page.

Quick Links

- Storage on Biowulf & Helix
- The NIH HPC object storage system
- Request form for non-object storage
- Request form for non-object shared data

The primary use case for the HPC object storage is for data that changes infrequently but still needs to be accessed for use in calculations (e.g. reference sequence files used in genomic analysis) or for future scientific work. Although there are no time restrictions on the use of HPC object storage space, users are expected to clean up data that is no longer necessary for ongoing projects. **Disk space on Helix and Biowulf, including on the object storage system, should never be used as archival storage.**

User Information:

Name:

Helx/Biowulf username:

Telephone:

Principal Investigator:

Brief description (one or two lines) of your research project:

Object storage vault name (if different than your user name):

Size of vault (if new) or quota increase (if existing):

How this space requirement was estimated and why your workload would benefit from object storage.
Example: I have 400 BAM files that will be written once and read numerous times by different batch jobs. Each BAM file takes 30 GB of space. Therefore, I need 12 TB of space on the object storage. Object storage is ideal for my workload because the data, once written, is static, and my analysis pipeline supports data access via the S3 protocol.

By clicking the check box below, you agree that you understand that object storage is different from disk storage in terms of how it is accessed and used. You acknowledge that you have read the documentation on the HPC object storage system and determined that your intended use is a good fit for the object storage system. You further acknowledge that you **cannot** use standard Unix/Linux utilities (e.g. ls, cat, vi, emacs) to interact with its data, but instead must use special utilities (e.g. the NIH HPC object tools, Rclone, S3 utilities, etc.) to interact with data on the object storage system. If you are not sure whether object storage would benefit your workload, please contact staff@hpc.nih.gov to discuss your proposed usage.

I have read the above (really!) and agree to it:

SUBMIT | CLEAR

Last modified: 9 January 2017

Remember to check the box at the bottom indicating that you understand what object storage is and the policies associated with its use.

Search:

HPC @ NIH

High-Performance Computing at the NIH

Status | Applications | Storage | User Guides | User Dashboard | How To | About

HPC Object Storage Request

This form should be completed by NIH HPC users who wish to use the HPC object storage system or who wish to increase their object store quota.

The HPC object storage system is a large, highly-scalable, and highly reliable data storage system. However, it has substantially lower performance than the storage systems that serve /home and /data. In addition, it is accessed in a completely different manner than more traditional disk-based systems. **To access object storage, you must use special utilities that can communicate via the S3 protocol.** For more information, please see the HPC object storage page.

Quick Links

- Storage on Biowulf & Helix
- The NIH HPC object storage system
- Request form for non-object storage
- Request form for non-object shared data

The primary use case for the HPC object storage is for data that changes infrequently but still needs to be accessed for use in calculations (e.g. reference sequence files used in genomic analysis) or for future scientific work. Although there are no time restrictions on the use of HPC object storage space, users are expected to clean up data that is no longer necessary for ongoing projects. **Disk space on Helix and Biowulf, including on the object storage system, should never be used as archival storage.**

User Information:

Name:

Helx/Biowulf username:

Telephone:

Principal Investigator:

Brief description (one or two lines) of your research project:

Object storage vault name (if different than your user name):

Size of vault (if new) or quota increase (if existing):

How this space requirement was estimated and why your workload would benefit from object storage.
Example: I have 400 BAM files that will be written once and read numerous times by different batch jobs. Each BAM file takes 30 GB of space. Therefore, I need 12 TB of space on the object storage. Object storage is ideal for my workload because the data, once written, is static, and my analysis pipeline supports data access via the S3 protocol.

By clicking the check box below, you agree that you understand that object storage is different from disk storage in terms of how it is accessed and used. You acknowledge that you have read the documentation on the HPC object storage system and determined that your intended use is a good fit for the object storage system. You further acknowledge that you **cannot** use standard Unix/Linux utilities (e.g. ls, cat, vi, emacs) to interact with its data, but instead must use special utilities (e.g. the NIH HPC object tools, Rclone, S3 utilities, etc.) to interact with data on the object storage system. If you are not sure whether object storage would benefit your workload, please contact staff@hpc.nih.gov to discuss your proposed usage.

I have read the above (really!) and agree to it:

SUBMIT | CLEAR

Last modified: 9 January 2017

Once you have submitted the form

- You'll get an e-mail confirmation
- The HPC staff will contact you if there are any questions about your request
- Your storage will be set up. You'll be given a set of **access keys** that you can use to access your space.

Hands-on: setting up access

- I will distribute access keys to individuals/teams
  - Note: the vaults used in this class are **TEMPORARY** and will be deleted a day or so after the class ends.
  - In other words, don't store anything you actually want to keep (only) here.
- Create a file /home/$USER/.boto
  - Replace $USER with your user name
  - Make sure the file is <u>only readable by you</u>
- Put the following contents in the file:

```
[Credentials]
aws_access_key_id = your_key_id
aws_secret_access_key = your_secret_key
```

Hands-on: setting up access

- Check that you can “see” your vault when you do obj_df.
  - Note – obj_df will not work with temporary student vaults.
  - How much space do you have on the object store?
  - Tip: obj_df reports value in bytes, which is not very easy to read. Use “obj_df -h” to get human-readable values.
- Run “obj_ls” to see the contents of your vault
  - This **will** work with temporary student vaults
  - If your vault name is not the same as your username, use “-v <vaultname>” to specify which vault.
  - Is there anything in your vault?

Outline

- Overview of the object storage
- A first practical example
- When would you want to use object storage?
- How do you get access to the object storage?
- Using the NIH HPC object storage
  - HPC staff developed tools
  - Rclone
- Programming your own tools

Overview of staff-developed tools

Name	Description	Notes
obj_ls	Lists objects on the storage	Listing objects can be slow if there is a large number of them. Please use this sparingly, and find other mechanisms for keeping track of objects that you have stored.
obj_put	Copies data from a file storage system such as /data or /home to the object store.	Supports wildcards. Note that copied data is not deleted from the source file system and will still count against any quotas.
obj_get	Copies data from the object store to a file system file or standard output.	Also supports wildcards. There are several options for choosing where data from the object store is placed on the filesystem. If obj_put was used to copy the data to the object store, obj_get can attempt to restore the original file modification time and permissions. Data can also be written to standard output and piped to applications.
obj_rm	Removes data from the object store.	Removes an object or objects from the object store. For safety, only the "?" wildcard character is supported (matches any single character). Please note that there are no back-ups or snapshots of the object store, so once an object is deleted, it is gone forever.
obj_df	Shows space utilization on the object store.	Please be careful not to exceed the amount of space allocated to you; if you do, you will not be able to store new objects.

obj_chmod described later!

Using obj_df

```
[teacher@biowulf ~]$ obj_df
Name                                Quota              Used               Remaining
teacher                             500000000000       0                  500000000000
```

```
[teacher@biowulf ~]$ obj_df -h
Name                                Quota         Used          Remaining
teacher                             500.00 GB     0.00 B        500.00 GB
```

- Shows you how much space you have on vaults you have **write** access to.
- Can get the same information (in a slightly different format) via checkquota
- Does not work with the student vaults for this class!

Using obj_ls

- Lists objects in a vault
- Owner ID is specific to the object store (i.e. not a Linux UID)

```
biowulf:~ 3$ obj_ls
        Size - Owner ID                              - Modified              Object
     3398373 - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-10-25 13:36:39   10k_2_subs_sum.txt.gz
      184284 - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-10-25 13:36:39   2006498997.statistics_cmd.txt
       31293 - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-10-25 13:36:39   CHARMMinginstallerfixes.pdf
        2342 - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-10-25 13:36:39   Layer_3_config_v2.txt
         572 - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-10-25 13:36:38   blurb.txt
      268192 - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-10-25 13:36:39   bwulf_phase2.odt
      226802 - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-10-25 13:36:39   bwulf_phase2.pdf
          13 - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-10-25 13:36:38   city.txt
      462933 - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-10-25 13:36:39   cleversafe_aug31_log
          18 - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-10-25 13:36:38   clientlist.txt
       99370 - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-10-25 13:36:40   cuda/include/cudnn.h
    59909104 - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-10-25 13:36:45   cuda/lib64/libcudnn.so.5.0.5
    58775484 - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-10-25 13:36:45   cuda/lib64/libcudnn_static.a
```

A prettier view

- -h: sizes are human readable
- -m: only print out files that match a given pattern

```
biowulf:~ 7$ obj_ls -h -m "*slurm*"
        Size - Owner ID                              - Modified              Object
    542.00 B - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-10-25 13:36:39   slurm-24745791.out
    11.28 KB - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-10-25 13:36:39   slurm-25014754.out
     6.22 MB - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-10-25 13:36:45   ticket_23161/slurm-24632000.out
```

Listing different vaults

- Some users have access to multiple vaults (see obj_df)
- Use -v flag on **all** obj tools to specify a vault
  - The default vault is the same as your username (may not exist)

```
biowulf:~ 8$ obj_ls -v apivault1
        Size - Owner ID                              - Modified              Object
          30 - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-08-05 12:09:45   btmiller/a/tfoo.txt
          30 - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-08-05 12:09:49   btmiller/b/tfoo.txt
          30 - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-08-05 12:09:55   btmiller/c/tfoo.txt
          30 - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-08-05 12:10:00   btmiller/d/tfoo.txt
          30 - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-08-05 12:10:04   btmiller/e/tfoo.txt
          30 - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-08-05 12:10:10   btmiller/f/tfoo.txt
          30 - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-08-05 12:10:17   btmiller/g/tfoo.txt
          30 - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-08-05 12:10:22   btmiller/h/tfoo.txt
          30 - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-08-05 12:10:30   btmiller/i/tfoo.txt
          30 - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-08-05 12:10:38   btmiller/j/tfoo.txt
          30 - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-08-05 12:10:55   btmiller/k/tfoo.txt
         110 - 117792b4-fd5e-45ba-8d44-55fbb54802e4 - 2016-04-21 14:38:27   btmiller/lamb
```

Putting data onto the object store

- Use obj_put
- Numerous options – we'll go through some of the more important

```
biowulf:~ 1$ obj_put -h
usage: obj_put [-h] [-a] [-R] [-P AMAZONPROFILE] [-p PREFIX] [-s SUFFIX]
               [-v VAULT] [-d] [-V] [--force] [-F] [-m METADATA]
               files [files ...]

Put one or more objects into a vault

positional arguments:
  files                 Files to store on the object storage

optional arguments:
  -h, --help            show this help message and exit
  -a, --amazon          Use credentials from ~/.aws/credentials instead of
                        ~/.boto
  -R, --recursive       Recursively copy all files in a directory to the
                        object store
  -P AMAZONPROFILE, --profile AMAZONPROFILE
                        Profile to use from ~/.aws/credentials - only
                        applicable when the -a flag is used.
  -p PREFIX, --prefix PREFIX
                        Prefix to prepend to the files - may contain slashes
  -s SUFFIX, --suffix SUFFIX
                        Suffix to append to the files - may contain slashes
  -v VAULT, --vault VAULT
                        Which vault to put files to (default is your username)
  -d, --dry-run         Don't actually do anything, just sanity check
  -V, --verbose         Be very explicit about what is happening
  --force               Overwrite existing objects without prompting - BE
                        CAREFUL
  -F, --full-objname    Use the full filesystem path (on the command line) as
                        the object name
  -m METADATA, --metadata METADATA
                        A list of metadata variables, e.g.
                        var1:value1,var2:value2
```

Putting data onto the object store

- Use prefix (-p, --prefix) to make a “directory”, e.g. ”-p new_results/”
  - Trailing slash is important!
- Use -F (--full-objname) to have the object name be the full path on the system.
- Use -R to recursively copy data.
- Not shown: checksum and progress bar options

```
biowulf:~ 1$ obj_put -h
usage: obj_put [-h] [-a] [-R] [-P AMAZONPROFILE] [-p PREFIX] [-s SUFFIX]
               [-v VAULT] [-d] [-V] [--force] [-F] [-m METADATA]
               files [files ...]

Put one or more objects into a vault

positional arguments:
  files                 Files to store on the object storage

optional arguments:
  -h, --help            show this help message and exit
  -a, --amazon          Use credentials from ~/.aws/credentials instead of
                        ~/.boto
  -R, --recursive       Recursively copy all files in a directory to the
                        object store
  -P AMAZONPROFILE, --profile AMAZONPROFILE
                        Profile to use from ~/.aws/credentials - only
                        applicable when the -a flag is used.
  -p PREFIX, --prefix PREFIX
                        Prefix to prepend to the files - may contain slashes
  -s SUFFIX, --suffix SUFFIX
                        Suffix to append to the files - may contain slashes
  -v VAULT, --vault VAULT
                        Which vault to put files to (default is your username)
  -d, --dry-run         Don't actually do anything, just sanity check
  -V, --verbose         Be very explicit about what is happening
  --force               Overwrite existing objects without prompting - BE
                        CAREFUL
  -F, --full-objname    Use the full filesystem path (on the command line) as
                        the object name
  -m METADATA, --metadata METADATA
                        A list of metadata variables, e.g.
                        var1:value1,var2:value2
```

An example

```
[teacher@biowulf ~]$ obj_put -V -p "classfiles/" qbf.txt
Processing file qbf.txt - object name will be classfiles/qbf.txt.
Adding file qbf.txt as object classfiles/qbf.txt.
[teacher@biowulf ~]$ obj_ls -h
        Size - Owner ID                              - Modified              Object
     64.00 B - 8d17e967-d9b4-4bf3-929c-d489ea8d763b - 2017-02-13 17:42:48   classfiles/qbf.txt
```

Getting the data back with obj_get

- Like obj_put, has a lot of options
- Several are important!
- <u>If</u> you stored a checksum, use the -c flag to compare against it.

```
biowulf:~ 2$ obj_get -h
usage: obj_get [-h] [-v VAULT] [-d] [-r] [-R] [-o] [-p] [-D DIRECTORY] [-s]
               [-V] [-q] [--strip STRIP] [--force]
               objects [objects ...]

Gets an objects from the object store and puts its contents into a file on the filesystem.

positional arguments:
  objects               Objects to fetch from the object store - may contain
                        glob (*,?) characters

optional arguments:
  -h, --help            show this help message and exit
  -v VAULT, --vault VAULT
                        Which vault to put files to (default is your username)
  -d, --dry-run         Don't actually do anything, just sanity check
  -r, --restore         Restore original path, permissions, times, and
                        ownership (if possible) - equivalent to -op
  -R, --recursive       Recursively copy all files with the same leading
                        prefix from the object store back to disk
  -o, --original-path   Restore original file path, if possible
  -p, --preserve-attributes
                        Restore original permissions modification time, and
                        access time, if possible
  -D DIRECTORY, --directory DIRECTORY
                        Directory to put the objects (overrides -r and -o).
  -s, --stdout          Dump object contents to standard output, rather than
                        to a file - overrides, -D, -r, and -o
  -V, --verbose         Be very explicit about what is happening
  -q, --quiet           Only print errors (opposite of --verbose)
  --strip STRIP         Strip the leading N path elements when restoring an
                        object to disk.
  --force               Overwrite existing files on the filesystem without
                        prompting - BE CAREFUL
```

Getting the data back with obj_get

- Note use of -r, -o, and -p
  - Rely on having metadata associated w/ the object.
  - Probably will not work unless placed with obj_put.
  - Overriden by -D
- --strip requires -D
- --stdout is a very useful option
  - Stream object data to programs

```
biowulf:~ 2$ obj_get -h
usage: obj_get [-h] [-v VAULT] [-d] [-r] [-R] [-o] [-p] [-D DIRECTORY] [-s]
               [-V] [-q] [--strip STRIP] [--force]
               objects [objects ...]

Gets an objects from the object store and puts its contents into a file on the filesystem.

positional arguments:
  objects               Objects to fetch from the object store - may contain
                        glob (*,?) characters

optional arguments:
  -h, --help            show this help message and exit
  -v VAULT, --vault VAULT
                        Which vault to put files to (default is your username)
  -d, --dry-run         Don't actually do anything, just sanity check
  -r, --restore         Restore original path, permissions, times, and
                        ownership (if possible) - equivalent to -op
  -R, --recursive       Recursively copy all files with the same leading
                        prefix from the object store back to disk
  -o, --original-path   Restore original file path, if possible
  -p, --preserve-attributes
                        Restore original permissions modification time, and
                        access time, if possible
  -D DIRECTORY, --directory DIRECTORY
                        Directory to put the objects (overrides -r and -o).
  -s, --stdout          Dump object contents to standard output, rather than
                        to a file - overrides, -D, -r, and -o
  -V, --verbose         Be very explicit about what is happening
  -q, --quiet           Only print errors (opposite of --verbose)
  --strip STRIP         Strip the leading N path elements when restoring an
                        object to disk.
  --force               Overwrite existing files on the filesystem without
                        prompting - BE CAREFUL
```

Streaming to stdout example

```
[teacher@biowulf ~]$ obj_get -s classfiles/qbf.txt
The quick brown fox jumped over the lazy object storage system.
```

- Useful for piping into commands
- See the --stdin option to obj_put to stream data TO the object store
  - WARNING: this could be way too slow if your program outputs data quickly!

Practice time!

- Copy /data/classes/objectstore to your data directory
- Copy the file lamb.txt to your object store vault
- Verify that the copy is there. How did you do this?
- Read the data back from the object store two different ways
  - Stream it to standard output
  - Read it back to a file in a new directory called lamb2
- Upload the data fruits.txt in the “some_files” directory to your object store vault
- Sort the fruits.txt data alphabetically (use the Linux sort command), output the results to your data directory, and then copy the sorted file to the object store.

Permissions and obj_chmod

- By default, objects in a vault are only visible to users who have access rights to that vault.
  - Vaults are treated similarly to shared data directory: requestor becomes the vault owner who can add or delete users.
  - Users can have read-only or read-write permissions
- A user who has **write** permission on a vault can make an object publicly visible.
  - Use obj_chmod to set “private” or “public-read” permissions
  - Available via HTTP “wget -O output http://os1naccess1/vault/object”
  - Not accessible beyond the NIH HPC systems, but this will change

Using the obj command

- obj is a single command that gives access to all object commands
  - See “obj help” for usage
  - E.g. “obj put” calls “obj_put”
  - One additional function – “obj url”, prints a URL of a publicly-available object
    - Use in conjunction with obj_chmod
    - URL is only reachable within the HPC systems!

Overview of Rclone

- Another tool for copying/synchronizing data between file storage and object storage
- Specializes in synchronizing entire directories to/from object storage
  - Like rsync, but for the object store
  - Not the most convenient tool for single files, but more convenient than obj_* for lots of data
- In addition to the HPC object store, can talk to other systems (S3, DreamHost, etc.).
- Remember to use --no-check-certificate when using with the HPC object store (might want an alias)
- Web site https://hpc.nih.gov/apps/rclone.html

Configuring rclone

- Load the module
  - `module load rclone`
- Configure:
  - `rclone config`
  - Remember what you named your object store (or check ~/.rclone.cfg or ~/.config/rclone/)
- For HPC object store
  - Choose Amazon S3 storage
  - Enter object key and secret key
  - For region – choose S3 clone that understands v2 signatures (12)
  - Enter os1naccess2 as the endpoint (can use os{1,2}naccess{1,2,3})
  - Leave other items as default
  - See example at https://hpc.nih.gov/apps/rclone.html!

Using rclone

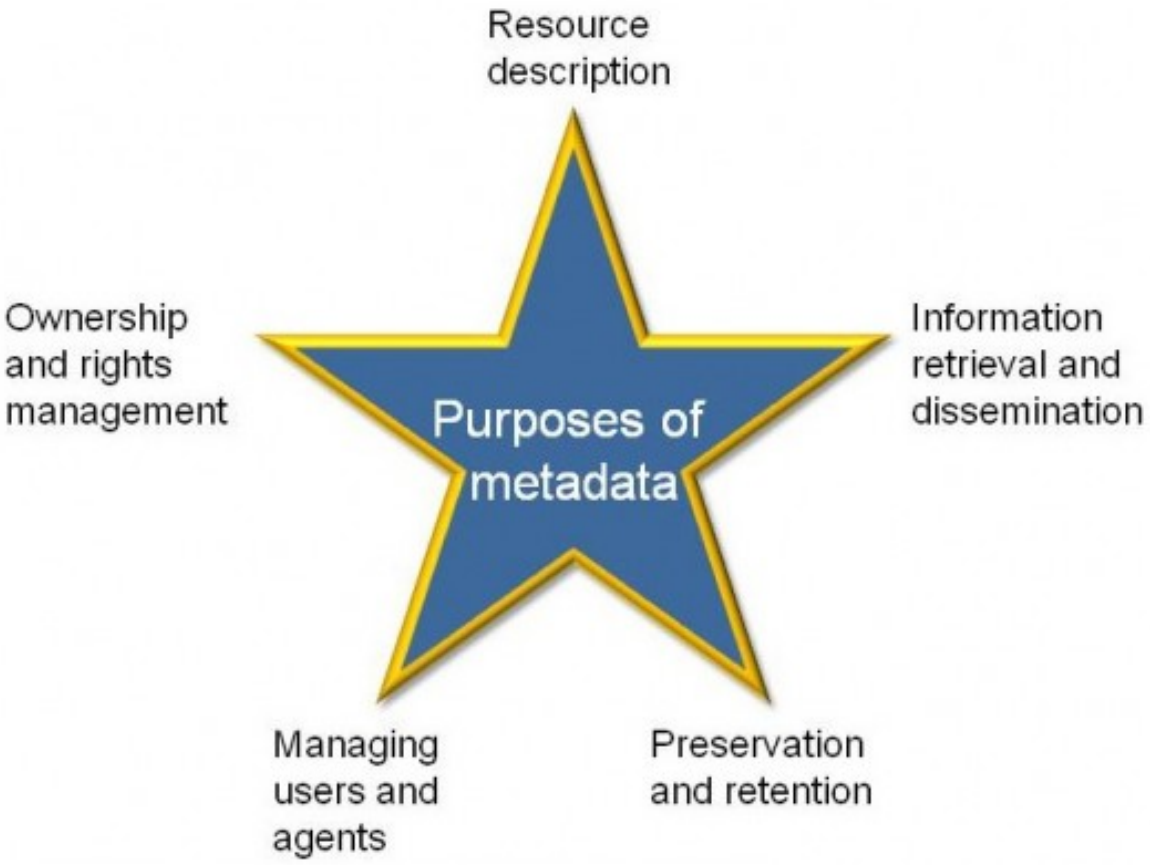
- Important argument “--no-check-certificate”
  - Needed for NIH HPC object store, since it uses self-signed SSL certificate
- Vaults are denoted <storage-system-name>:<vault-name>
  - E.g. nihhpc-obj:btmiller
  - The storage system name is what you defined it to be when configuring rclone!
- “rclone ls” – list a directory
  - E.g. “rclone ls nihhpc-obj:btmiller”
  - “rclone lsd” will list <u>only</u> directories
- “rclone copy” – copy from source to destination
- “rclone move” – move from source to destination
- “rclone (purge|rmdir)” – removes paths
- Sources and destinations can be
  - Object store (HPC or other)
  - Filesystem paths

Exercises

- Read “rclone --help”
- Configure rclone to access your object store vault
- Use rclone to view the contents of your object store vault.
- Using rclone, upload the “some_files” subdirectory of the object store class examples to your vault.
- Copy just the files with “some_files” in their object name to a new directory with rclone.
- Sort some_files/a_subdirectory/famous_computers.txt by system name, then upload the results (via rclone) to a new object named some_files/a_subdirectory/famous_computers_byname.txt

Some notes on metadata

- Metadata = data about data
- Traditional file metadata
  - Ownership
  - Permissions
  - Name
  - Disk location
  - Arbitrary?
- Object storage metadata
  - Arbitrary key-value pairs
  - Permissions handled separately via ACL
  - (Re-)Creation time set automatically



Using metadata on the object storage

- Needs to be accessed via the object API or via “-m” flag to obj_put
  - E.g. set_metadata, get_metadata methods in Boto (more on this later)
- obj_put, obj_get are examples – store (and retrieve) filesystem path for an object.
- You might want to use metadata for…
  - Anonymized sample identifiers
  - Type of machine/analysis program used to generate the data
  - Any input paths, objects
  - Just about anything else you can think of that is not part of the object’s data itself!

A practical example using the object store

- Put raw data (gzipped fastq files) into the object store
- Align the fastq files using STAR (read data directly from object storage)
- Create bigWig coverage
- Make coverage available via Genome browser
- Visualize the data

Putting example data into the object store

```
biowulf$ sinteractive -c12 --mem=40g --gres=lscratch:50
salloc.exe: Pending job allocation 34043956
salloc.exe: job 34043956 queued and waiting for resources
salloc.exe: job 34043956 has been allocated resources
...
node$ cd /lscratch/$SLURM_JOB_ID
node$ wget https://www.encodeproject.org/files/ENCFF398KGB/@@download/ENCFF398KGB.fastq.gz
...
node$ wget https://www.encodeproject.org/files/ENCFF058JYK/@@download/ENCFF058JYK.fastq.gz
...
node$ obj put --metadata read:1,exp:ENCSR862RGX ENCFF398KGB.fastq.gz
node$ obj put --metadata read:2,exp:ENCSR862RGX ENCFF058JYK.fastq.gz
node$ rm ENCFF398KGB.fastq.gz ENCFF058JYK.fastq.gz
node$ obj ls -h
        Size - Owner ID                                - Modified              Object
     3.75 GB - e277e9bd-d283-466b-ac0c-b0949ee73baf - 2017-02-22 18:03:45   ENCFF058JYK.fastq.gz
     3.69 GB - e277e9bd-d283-466b-ac0c-b0949ee73baf - 2017-02-22 18:02:09   ENCFF398KGB.fastq.gz
```

Align data from the object store with STAR

```
node$ module load STAR/2.5.2b
node$ STAR \
    --runThreadN $((SLURM_CPUS_PER_TASK - 2)) \
    --genomeDir /fdb/STAR_indices/2.5.2b/GENCODE/Gencode_human/release_25/genes-100 \
    --readFilesIn <(obj get -s ENCFF398KGB.fastq.gz) \
                  <(obj get -s ENCFF058JYK.fastq.gz) \
    --readFilesCommand zcat \
    --outSAMtype BAM SortedByCoordinate \
    --outFileNamePrefix ENCSR862RGX
Feb 22 18:18:10 ..... started STAR run
Feb 22 18:18:10 ..... loading genome
Feb 22 18:22:33 ..... started mapping
Feb 22 18:34:30 ..... started sorting BAM
Feb 22 18:52:07 ..... finished successfully

node$ ls -lh
total 8.1G
-rw-r--r-- 1 wresch staff 8.1G Feb 22 18:47 ENCSR862RGXAligned.sortedByCoord.out.bam
-rw-r--r-- 1 wresch staff 1.9K Feb 22 18:52 ENCSR862RGXLog.final.out
-rw-r--r-- 1 wresch staff  34K Feb 22 18:52 ENCSR862RGXLog.out
-rw-r--r-- 1 wresch staff 1.7K Feb 22 18:52 ENCSR862RGXLog.progress.out
-rw-r--r-- 1 wresch staff 9.0M Feb 22 18:52 ENCSR862RGXSJ.out.tab

node$ obj put ENCSR862RGXAligned.sortedByCoord.out.bam
```

Align data from the object store with STAR

```
node$ module load STAR/2.5.2b
node$ STAR \
    --runThreadN $((SLURM_CPUS_PER_TASK - 2)) \
    --genomeDir /fdb/STAR_indices/2.5.2b/GENCODE/Gencode_human/release_25/genes-100 \
    --readFilesIn <(obj get -s ENCFF398KGB.fastq.gz) \
                  <(obj get -s ENCFF058JYK.fastq.gz) \
    --readFilesCommand zcat \
    --outSAMtype BAM SortedByCoordinate \
    --outFileNamePrefix ENCSR862RGX
Feb 22 18:18:10 ..... started STAR run
Feb 22 18:18:10 ..... loading genome
Feb 22 18:22:33 ..... started mapping
Feb 22 18:34:30 ..... started sorting BAM
Feb 22 18:52:07 ..... finished successfully

node$ ls -lh
total 8.1G
-rw-r--r-- 1 wresch staff 8.1G Feb 22 18:47 ENCSR862RGXAligned.sortedByCoord.out.bam
-rw-r--r-- 1 wresch staff 1.9K Feb 22 18:52 ENCSR862RGXLog.final.out
-rw-r--r-- 1 wresch staff  34K Feb 22 18:52 ENCSR862RGXLog.out
-rw-r--r-- 1 wresch staff 1.7K Feb 22 18:52 ENCSR862RGXLog.progress.out
-rw-r--r-- 1 wresch staff 9.0M Feb 22 18:52 ENCSR862RGXSJ.out.tab

node$ obj put ENCSR862RGXAligned.sortedByCoord.out.bam
```

Note the use of input redirection combined with obj get

Create bigWig of coverage

```
node$ module load bedtools/2.26.0-38-gf3db04e ucsc

node$ bedtools genomecov -ibam ENCSR862RGXAligned.sortedByCoord.out.bam \
          -g gencode_25.genome -split -bg \
        | grep '^chr' \
        | sort -k1,1 -k2,2n -S 3G \
        > temp.bedGraph
node$ bedGraphToBigWig temp.bedGraph gencode_25.genome ENCSR862RGX.bw
node$ obj put ENCSR862RGX.bw
```

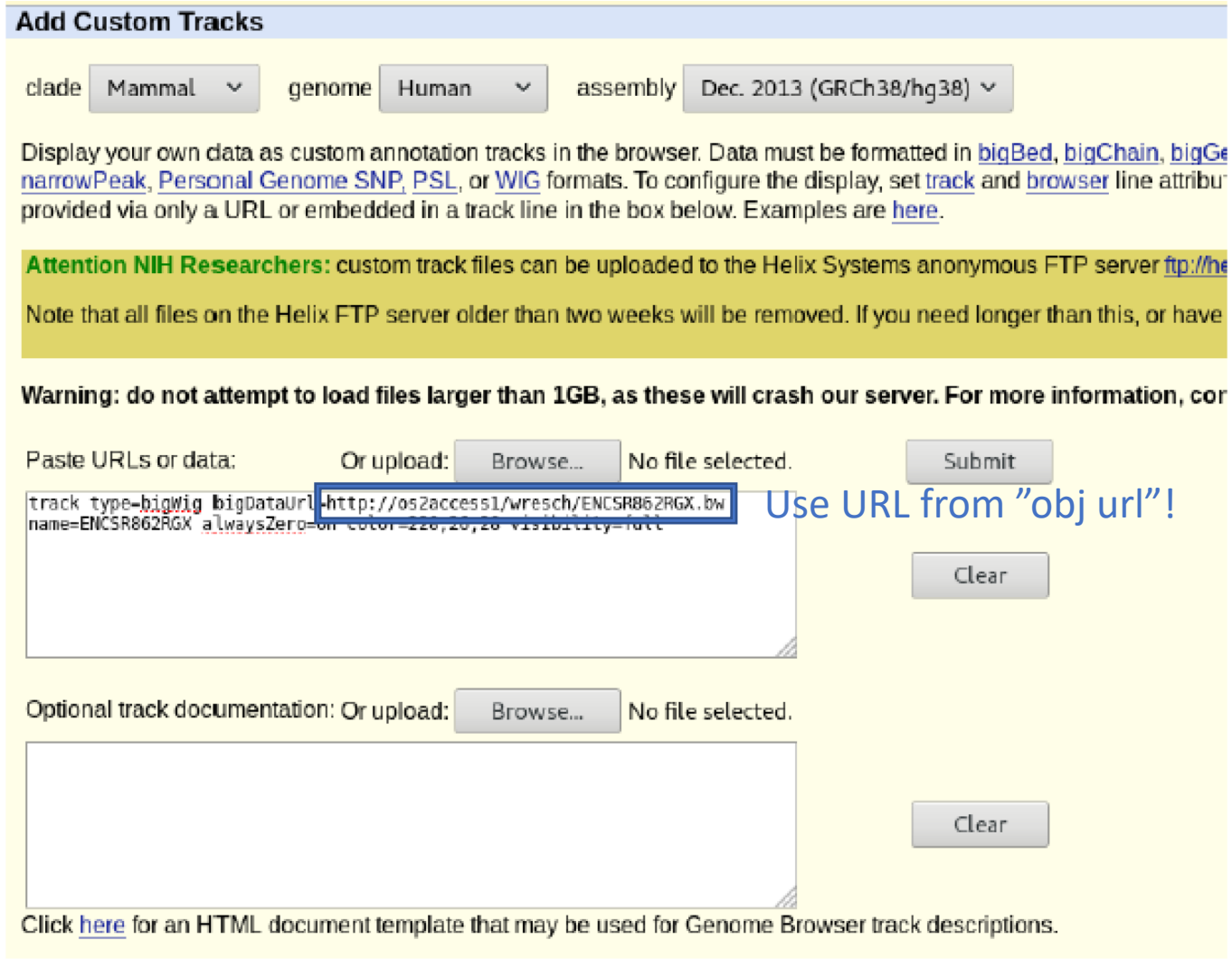
- Uses local output files from STAR
- Puts results back on the object store for later access

Make bigWig available to CIT genome browser

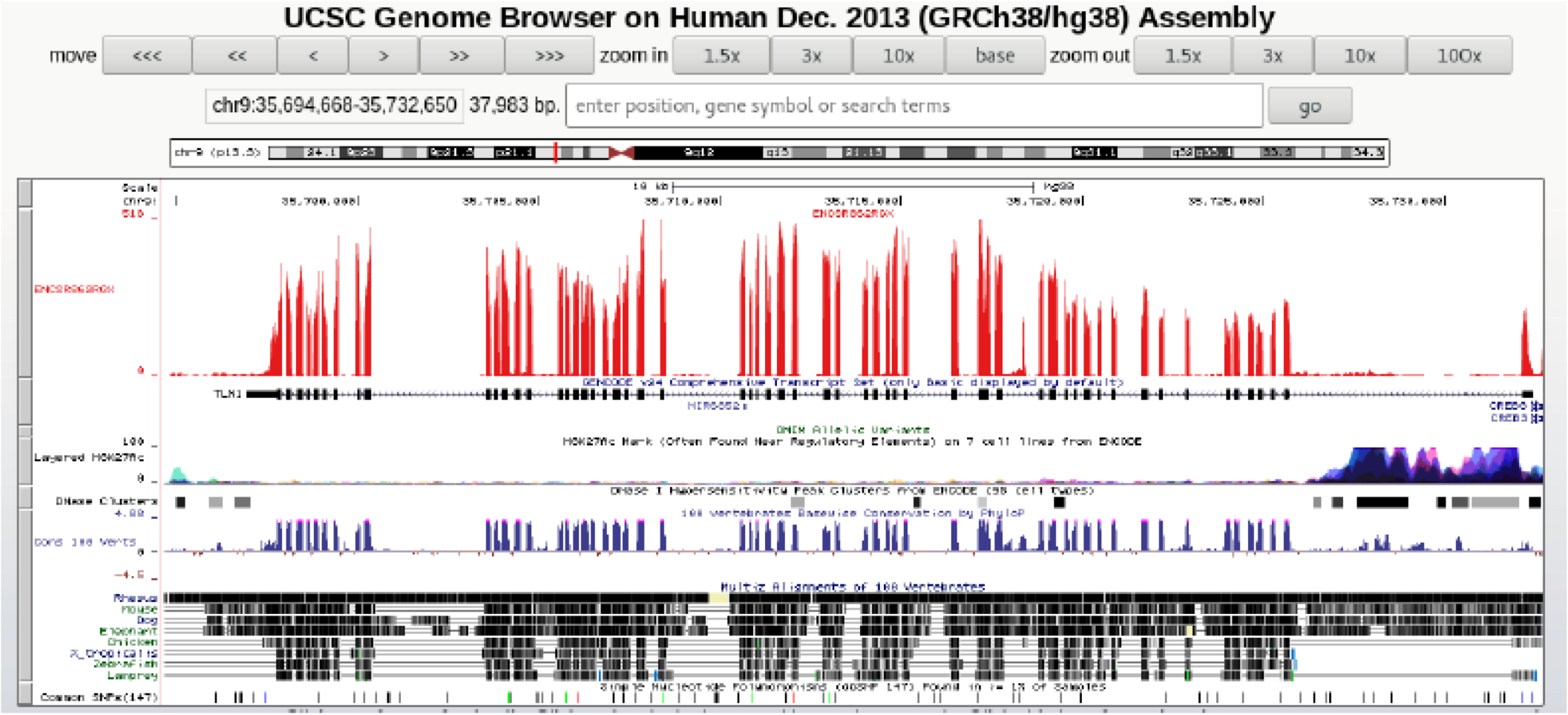
```
node$ obj ls -hp
        Size - Owner ID                                - Modified              Object
   ...
   110.46 MB - e277e9bd-d283-466b-ac0c-b0949ee73baf - 2017-02-22 21:36:43   ENCSR862RGX.bw
                 Wolfgang Resch: FULL_CONTROL
   ...


node$ obj chmod -v $USER public-read ENCSR862RGX.bw
node$ obj ls -hp
         Size - Owner ID                                - Modified              Object
    ...
    110.46 MB - e277e9bd-d283-466b-ac0c-b0949ee73baf - 2017-02-22 21:36:43   ENCSR862RGX.bw
                       Everybody: READ
                 Wolfgang Resch: FULL_CONTROL
    ...

node$ obj url ENCSR862RGX.bw
http://os2access1/wresch/ENCSR862RGX.bw
```

Visualize track

Visualize track



Visualize track



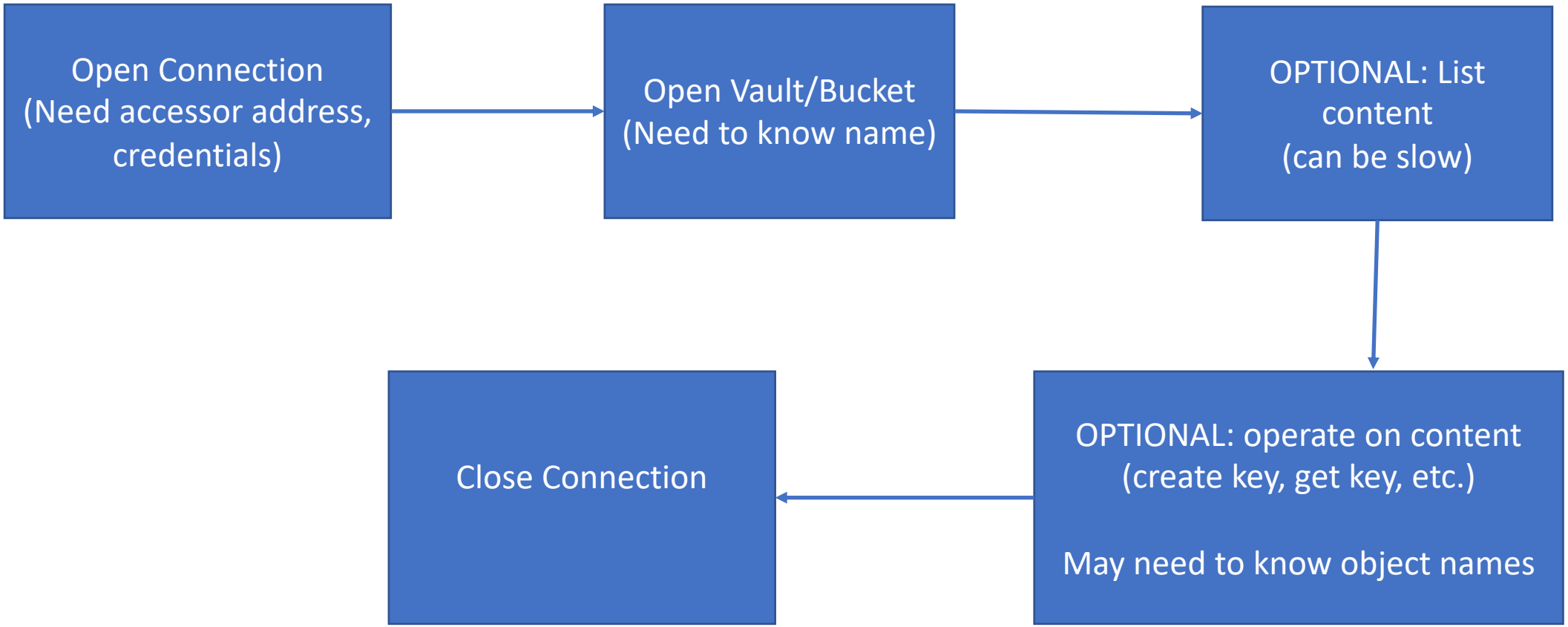
Outline

- Overview of the object storage
- A first practical example
- When would you want to use object storage?
- How do you get access to the object storage?
- Using the NIH HPC object storage
  - HPC staff developed tools
  - Rclone
- Programming your own tools

Programming your own tools

- Why?
  - Need to customize functionality, e.g. reading metadata from objects.
  - Combining reading/writing data to object store with calculations.
  - Complicated access patterns – e.g. reading and writing multiple objects simultaneously.
- Why not?
  - Requires knowledge of Python, Perl, Ruby, C++, etc.
    - NIH HPC developed object tools are all in Python – can use as examples
  - If existing tools satisfy your needs, there's not much point
  - HPC staff fully supports "our" tools (and others like Rclone) – much less support for custom development

Programming workflow



Programming in Python: Boto library

- I've had more luck with boto v2 than boto v3
  - Boto v3 is not compatible with our object store ☹.
- Getting a bucket (vault):

```
conn = boto.connect_s3(host=accessor, \
                       is_secure=False, \
                       calling_format=boto.s3.connection.OrdinaryCallingFormat(),
                       aws_access_key_id=idkey, \
                       aws_secret_access_key=seckey)

try:
    bucket = conn.get_bucket(vname)
except S3ResponseError, e:
    sys.stderr.write('Got response error while getting vault contents.\n')
    sys.stderr.write('Check that your ~/.boto file has the correct API key,\n')
    sys.stderr.write('or contact staff@hpc.nih.gov for help.\n')
    return False
except:
    sys.stderr.write('Unexpected error listing vault.\n')
    sys.stderr.write('Contact staff@hpc.nih.gov for help.\n')
    return False
```

Programming in Python: Boto library

- I've had more luck with boto v2 than boto v3
  - Boto v3 is not compatible with our object store ☹.
- Listing contents:

```
bucket = conn.get_bucket(vname)
keys = bucket.list()
```

```
print('%12s - %-36s - %-21s %-25s' % ('Size','Owner ID','Modified','Object'))
for key in keys:
    if pattern and (not fnmatch.fnmatch(key.name, pattern)):
        continue

    if human:
        print('%12s - %-36s - %-21s %-25s' % (numFmt(key.size,-1),str(key.owner.id),fixTime(key.last_modified),key.name))
    else:
        print('%12s - %-36s - %-21s %-25s' % (key.size,str(key.owner.id),fixTime(key.last_modified),key.name))
```

Programming in Python: Boto library

- I've had more luck with boto v2 than boto v3
  - Boto v3 is not compatible with our object store ☹.
- Deleting an object:

```
try:
    obj = bucket.get_key(objname)
    if not obj:
        raise
except:
    sys.stderr.write('Vault %s has no such object %s - skipping.\n' % (args.vault,objname))
    continue

if not args.dryrun:
    obj.delete()
    if args.verbose:
        print('Object %s removed.' % objname)
```

Programming in Python: Boto library

- I've had more luck with boto v2 than boto v3
  - Boto v3 is not compatible with our object store ☹.
- Putting data:

```
key = bucket.new_key(objname)
for mdkey in md.keys():
    key.set_metadata(mdkey,md[mdkey])
key.set_contents_from_filename(fname)
```

Programming in Python: Boto library

- I've had more luck with boto v2 than boto v3
  - Boto v3 is not compatible with our object store ☹.
- Getting data:

```
try:
    obj = bucket.get_key(objKey)
except:
    sys.stderr.write('Unexpected error fetching object %s from the store - skipping.\n' % objKey)
    if args.verbose:
        traceback.print_exc(file=sys.stderr)
    continue
```

```
if args.stdout:
    obj.get_contents_to_file(sys.stdout)
else:
    obj.get_contents_to_filename(finpath)
```

Wrap-up, summary

- Object store is good for data that is…
  - Read-only in nature
  - Needs to be used regularly for computation
  - Only needs moderate performance
- The object store cannot be used for…
  - Write-intensive data
  - Data that gets updated frequently
  - Archival data (until we make the limited archive available)
- Multiple different tools may be used to read and write data
- You can write your own tools in a variety of programming languages

staff@hpc.nih.gov

Steve Bailey

Steven Fellini, Ph.D.

Susan Chacko, Ph.D.

Gennady Denisov, Ph.D.

Picture unavailable

Afif Elghraoui

David Hoover, Ph.D.

Patsy Jones

Picture unavailable

Charles Lehr

Jean Mao, Ph.D.

Tim Miller

Charlene Osborn

Mark Patkus

Dan Reisman

Wolfgang Resch, Ph.D.

Jerez Te, Ph.D.

Antonio Ulloa, Ph.D.

Jesse Becker

Michael Harris (intern)

Thank you

- Thank you for attending
- Please contact us with questions/feedback
  - Tim Miller – btmiller@helix.nih.gov
  - staff@hpc.nih.gov