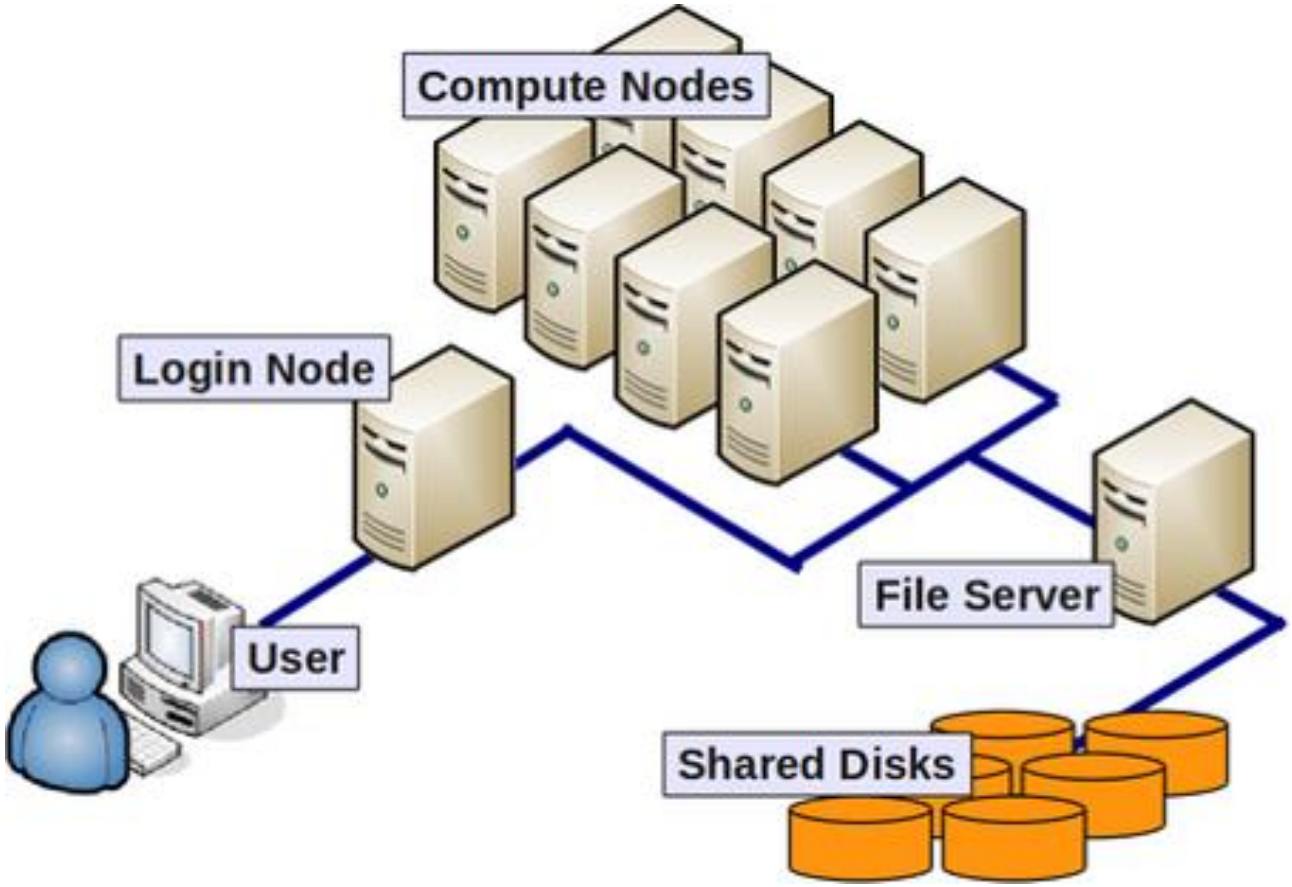
O2 for New Users

HMS Research Computing

O2 Tech Specs

- 7000 cores
- 32 or 28 cores per node
- 256 GB RAM per node (8GB/core)
- 24 GPUs (8 M40 / 16 K80)
- Login/load balancer 5 VM (8 cores/16GB memory)
- InfiniBand connectivity between nodes available
- CentOS 7

Generic Cluster Architecture

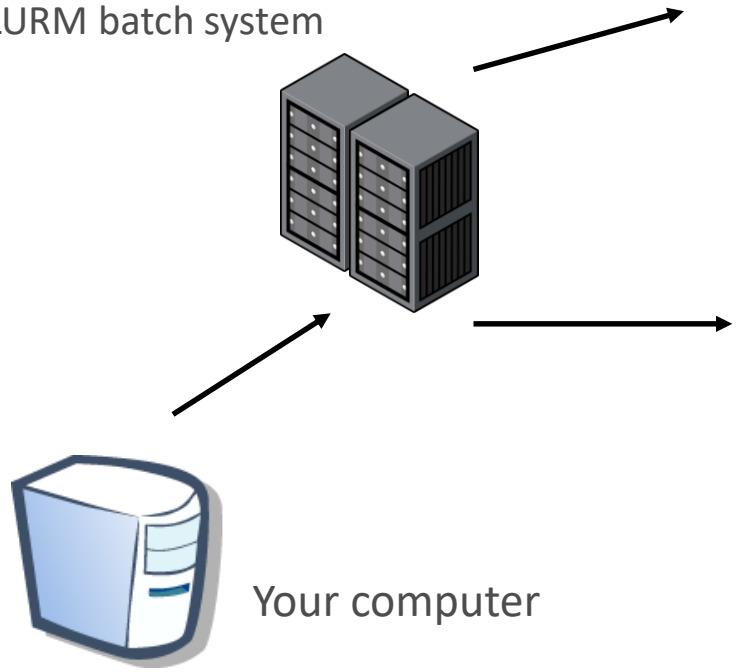


Storage on O2

O2 Primary Storage

O2 Cluster

- 7000+ cores
- SLURM batch system



/home

- /home/user_id
- quota: 100GB per user
- Backup: extra copy & snapshots:
- daily to 14 days, weekly up to 60 days

/n/data1, /n/data2, /n/groups

- /n/data1/institution/dept/lab/your_dir
- quota: expandable
- Backup: extra copy & snapshots:
- daily to 14 days, weekly up to 60 days

Temporary “Scratch” storage

- **/n/scratch2**
- **For data only needed temporarily during analyses.**
- **Each account can use up to 10 TB and 1 million files/directories**

  - **Lustre** --> a high-performance parallel file system running on DDN Storage.
  - More than 1 PB of total shared disk space.
  - No backups! Files are automatically deleted after unaccessed for 30 days, to save space.
  - More info at: http://hmsrc.me/O2docs

Checking Storage Usage

- To check your storage available:

  mfk8@login01:~$ quota

  Home directory: you get 100 GB, total.

  Group directories: space varies, can increase.

  /n/groups/groupname
  /n/data1
  /n/data2

Checking Storage Usage: scratch2

- mfk8@login01:~$ lfs quota -h /n/scratch2
- Quota is on user basis, not group basis
- Users are entitled to 10TB and up to 1 million files/directories

Storage Policies

- /home: 14 day snapshots + 60 day full backup
- /n/groups, /n/data1, /n/data2: 14 day snapshots + 60 day full backup
- /n/scratch2: 30 day retention, no backups
- Long Term Storage options

Snapshots - Isilon

- Snapshots (static) are retained for up to 60 days: recover data
- mfk8@compute-a:~$ cd .snapshot
- mfk8@compute-a:~$ ls

  Orchestra_home_daily_2015-10-02-02-00

  Orchestra_home_daily_2015-10-01-02-00
- mfk8@compute-a:~$ cd Orchestra_home_daily_2015-10-02-02-00
- mfk8@compute-a:~$ cp MyRetreivedFile ~

Logging into O2

Create a New O2 Account

- http://rc.hms.harvard.edu/#cluster

  Click the red button and fill out the form!

- Your username will be your eCommons ID, with your eCommons password.

Logging Into Orchestra: Mac

- Open a terminal (search “terminal”)

  ```
  ssh YourECommons@o2.hms.harvard.edu
  ```

- To display graphics back to your desktop (X11 forwarding)

  Install XQuartz (google it) and have it running

  ```
  ssh -XY YourECommons@o2.hms.harvard.edu
  ```

Logging Into Orchestra: Windows

- Install MobaXterm (google it)

```
ssh YourECommons@o2.hms.harvard.edu
```

- To display graphics back to your desktop (X11 forwarding); MobaXterm already has an X11 client built-in

```
ssh -XY YourEcommons@o2.hms.harvard.edu
```

Logging Into Orchestra: Linux

- Open a terminal (search: “terminal”)

```
ssh YourEcommons@o2.hms.harvard.edu
```

For graphics (X11 Forwarding)

```
ssh -XY YourEcommons@o2.hms.harvard.edu
```

Welcome to O2!

- Where are you in O2?

  See your terminal!

  ```
  mfk8@login01: ~$
  ```

- You are logged into a “shell login server”, login01-05.These are not meant for heavy lifting!
- You are in your home directory. This is symbolized by the “tilde ” (~). This is shorthand for: /home/eCommons
- You are in a bash environment. “$” means “ready to accept your commands!”

Interactive Sessions

- The login servers are not designed to handle intensive processes, and CPU usage is throttled. Start by entering your first job! This will (usually) log you into a "compute node!"

```
mfk8@login01:~$ srun --pty -p interactive -t 0-12:00 --mem 8G
bash
```

"`srun --pty`" is how interactives are started

"`-p interactive`" is the partition

"`-t 0-12:00`" is the time limit (12 hours)

"`--mem 8G`" is the memory requested

```
mfk8@compute-a:~/$
```

Linux Basics

Listing a Folder's Contents

- To see the contents of the current folder you are in (~ means "/home/username/"), type **lis**t (ls):

```
mfk8@compute-a:~$ ls
```

- To get the details of a folder's contents, add "-l"

```
mfk8@compute-a:~$ ls -l
```

- You don't have to be in a directory to see its contents

```
mfk8@compute-a:~$ ls /n/groups/rc-training/introtohpc
```

Viewing File Contents

- “less” to view file contents
- Navigate up/down, search
- “q” to quit

```
mfk8@compute-a:~$ less ~/.bashrc
```

Making a Folder (Directory)

- "mkdir" stands for "**mk**e **dir**ectory."
- Create a new directory for this exercise
- Spaces are discouraged. (Underscores are fine!) Case counts in Linux.

```
mfk8@compute-a:~$ mkdir MyTestDir
```

Moving Around: Change Directory

- "cd" stands for "**c**hange **d**irectory"
- 1 period "."means "current directory"
- 2 periods ".." means "the directory above"

```
mfk8@compute-a:~$ cd MyTestDir
```

Notice how the prompt tells you where you are!

```
mfk8@compute-a:~/MyTestDir$ cd ..
mfk8@compute-a:~$
```

Creating a Simple Text File

- "Nano," "vi", "emacs" are simple command-line editors available.
- To create a new file, type the editor you want, then the name of the new file. To edit an existing file, do the same.

```
mfk8@compute-a:~$ nano myfile.txt
    This is my new file text.
    (Control-X to save (yes) and exit.)
mfk8@compute-a:~$
mfk8@compute-a:~$ ls
    myfile.txt
```

Copying Files

- "cp" to **co**p**y** a file from a destination to a new destination. "cp" "from" "to"
- `cp -r` to copy folders (recursively)

```
mfk8@compute-a:~$ cp myfile.txt MyTestDir/
```

- You can copy a file to the current folder or to a new folder with a different name by specifying a different name (rename)

```
mfk8@compute-a:~$ cp myfile.txt mycopy2.txt
#copying and renaming
```

Moving Data

- "**m**ove" "from" "to"

```
mfk8@compute-a~:$ mv MyTestDir/myfile.txt ~
#this rewrites myfile.txt, since it already exists!
mfk8@compute-a~:$ mv MyTestDir/ MyTestDir2/
#in-place move and rename
```

Removing Files/Folders

- "rm" to **rem**ove a file

```
mfk8@compute-a:~$ rm myfile.txt
```

- "rm –r" to remove a folder **r**ecursively

```
mfk8@compute-a:~$ rm -r MyTestDir2
```

Wildcard * Pattern Matching

- Useful for copying/removing/etc all files matching a certain pattern
- Example Case:

  To copy “all” files ending in “.fastq”:

  ```
  $ cp *.fastq NewFastqFolder
  ```

Getting Data Onto Orchestra

- Use an FTP client of your choice
- Mac/Windows/Linux: Filezilla (google it)
- Connect to:

  transfer.rc.hms.harvard.edu

  your username and password (lowercase username)

  port 22

Software on O2

LMOD: Software Modules

- LMOD system adds directory paths of software into $PATH variable, and resolves software dependencies and conflicts
- Most software compiled against gcc-6.2.0: load first
- `$ module load gcc/6.2.0`
- `#module avail` #to see software now available
- `$ module spider` #verbose software currently available
- `$ module load software/version` #load software
- `$ module unload software/version` #unload
- `$ module purge` #dump all modules
- `$ module help <software>` #only for loaded software

Loading/Unloading Modules

- Loading modules

```
$ module load gcc/6.2.0 bowtie2/2.2.9
```

- Which module version is loaded (if at all)?

```
$ which bowtie2
```

- See all modules loaded

```
$ module list
```

- Unloading modules

```
$ module unload bowtie/2.2.9
```

- Dump all modules

```
$ module purge
```

Compiling your own software

- Users can compile software in their /home or /n/groups directories, where they have permission
- Binaries just require “unzipping” (ie tar –zxvf .tgz)

Installing Software: Binary Example

- `mfk8@login01:~$ srun --pty -p interactive -t 0-12:00 --mem 8G bash`
- `mfk8@compute-a:~$ wget http://path/to/binary/mysoftware.tar.gz`
- `mfk8@compute-a:~$ tar -zxvf mysoftware.tar.gz`
- `mfk8@compuate-a:~$ ls mysoftware/bin`

Programming Languages

- Python: load module (2.7.12, conda2, 3.6.0)
  - use virtualenv to maintain packages (pip/easy install)
- R: load module (3.2.5, 3.3.3, 3.4.1)
  - Setup O2-specific personal R library, .Renviron (install.packages/biocLite)
- Perl: load module (5.24.0)
  - Setup O2-specific local::lib (cpan/cpanm) in .bashrc
- MATLAB: load module (2016b, 2017a)
  - Setup cluster profile, only use on 1 HPC at a time

MPI on O2

- Message Passing Interface
- Distribute work over multiple nodes, allowing for the utilization of more cores
- openMPI-2.0.1 compiled against GCC 6.2.0
- MATLAB, Python, R, Perl, Java, C++, Fortran implementations
- Needs wrapper function “mpirun” to dispatch to compute nodes with SLURM
- Run in “mpi” partition `-p mpi` after being added to partition
- Core cap: 640 processors, 5 day runtime

Constructing Jobs

Submitting Jobs

- In an “interactive session”, programs can be called directly.

```
mfk8@compute-a:~$ bowtie –n 4 hg19 file1_1.fq file1_2.fq
```

- From the login shell (and also interactive or any compute nodes), a program is submitted to O2 via a job (sbatch)

```
mfk8@compute-a:~$ sbatch mybowtiejob.sh
```

Jobs: sbatch

- All in one line: --wrap="command here" #not recommended
- sbatch –p partition –t 0-1:00 --wrap="sh script.sh"

```
$ sbatch –p short –t 0-1:00 --wrap="cp file.txt .."
```

- Complete shell script #recommended
- `$ sbatch completeSlurmJob.run`

```
#!/bin/bash
#SBATCH –p short
#SBATCH –t 0-1:00
cp file.txt ..
```

Partitions -p

Partition	Priority	Max Runtime	Max Cores	Limits
short	12	12 hours	20	
medium	6	5 days	20	
long	4	30 days	20	
interactive	14	12 hours	20	2 job limit
priority	14	30 days	20	2 job limit
mpi	12	5 days	640	20 core min
highmem	12	5 days	28	750G
gpu		72 GPU hours	20cpu	
transfer		5 days	4	

Runtime: -t

- -t days-hours:minutes
- -t hours:minutes:seconds
- Need to specify how long you estimate your job will run for
- Aim for 125% (over)
- Subject to maximum per partition
- Excessive runlimits (like partition max) take longer to dispatch, and affect fairshare

CPU: -c

- -c X to designate CPU: max 20
- -N X to constrain all cores to X nodes
- CPU time: wall time (-t) * (-c) CPUs used
- Unable to use CPU not requested (no overefficient jobs): cgroups constraint
- Adding more cores does not mean jobs will scale linearly with time, and causes longer pend times

Memory: --mem

- Only 1G is allocated by default
- `--mem XG` #total memory over all cores
- `--mem-per-cpu XG` #total memory per CPU requested, use for MPI
- No unit request (G) defaults to Megabytes
  - 8G ~= 8000

Job Construction

```
#!/bin/bash
#SBATCH -p short #partition
#SBATCH -t 0-01:00 #time days-hr:min
#SBATCH -c X #number of cores
#SBATCH -N 1 #confine cores to 1 node, default
#SBATCH --mem=XG #memory per job (all cores), GB
#SBATCH -o %j.out #out file
#SBATCH -e %j.err #error file
#SBATCH --mail-type=BEGIN/END/FAIL/ALL
#SBATCH --mail-user=mfk8@med.harvard.edu
```

Output/Error Files

- Can add jobid to filename with %j
- Sample:
  -e %j.err
  -o %j.out
- SLURM by default creates this outfile:
  slurm-jobid.out
- Additional Flags
- %a job array id
- %A master array job id
- %N node name
- %u user id

Mail

- Mail is not auto-generated upon completion/failure
- `#SBATCH --mail-type= NONE, BEGIN, END, FAIL, REQUEUE, ALL`
- `#SBATCH --mail-user=mfk8@med.harvard.edu`
- Not recommended, not a verbose output like LSF
- Use sacct instead

Job Master Allocations/Job Steps

- "sbatch" or "salloc" will create a master allocation of shared resources
- To run job steps or multiple commands in parallel with resources from the master allocation, use srun

```
#/bin/bash
#SBATCH -c 8
#SBATCH --mem 32000
srun -c 2 --mem=8000 COMMAND1 &
srun -c 4 --mem=8000 COMMAND2 &
srun -c 1 --mem=4000 COMMAND3 &
srun -c 1 --mem 12000 COMMAND4 &
wait #necessary
```

starts 4 independent parallel tasks with memory constraints

Job Arrays

- `sbatch --array=1-30 submit.sh`
- `#SBATCH --array=1-30`
- slurm creates:
- `$SLURM_JOB_ID` #jobid of each job in array %j
- `$SLURM_ARRAY_JOB_ID` #jobid of entire array %A
- `${SLURM_ARRAY_TASK_ID}` #index of job %a

Example:

Files named: File1.txt File2.txt File3.txt ...File30.txt

Execution is:

```
myscript.sh File${SLURM_ARRAY_TASKID}.txt
```

Job Dependencies

- `sbatch --dependency=`
- `after:jobid` #(asynchronous)
- `afterany:jobid` #after exit or done
- `afterok:jobid` #success, exit code 0
- `afternotok:jobid` #failure
- `singleton` #after jobs with same name have terminated
- `--kill-on-invalid-dep=<yes|no>` #kill on unmet dependency (on by default)

Command Line Arguments

- slurm scripts can take command line arguments Reference as $1, $2 etc
- `sbatch submit.run 25 output.txt`

```
#!/bin/bash
#SBATCH -p short
#SBATCH -t 0-1:00
python myscript.py $1 $2

#runs as
python myscript.py 25 output.txt
```

Job Priority

- Dynamically assigned
- Factors contributing:
- Age, Fairshare, Partition, QOS, Nice
- Fairshare: 0-1 scale

Job Management

Job Monitoring

- `$ squeue –u eCommons –t RUNNING/ PENDING`
- `$ squeue –u eCommons –p partition`
- `$ squeue –u eCommons --start`
- Detailed job info:
  `$ scontrol show jobid <jobid>`
- Completed job statistics:
  `$ sacct -j <jobid> --format=JobID,JobName,MaxRSS,Elapsed`

Job information: sacct

- Advanced job accounting options
- `$ sacct -j jobid`

Options:

`--name`

`-r/--partition`

`-s/--state`

`-o/--format`

- `$ sacct –u eCommons`
- `$ sacct --helpformat` #get available accounting features

sacct: basic information

- `sacct --helpformat` displays fields
- RC Recommends:

```
sacct –u eCommons --
format=jobid,Partition,state%22,MaxRSS,MaxVMSize,ReqTRES%20,Star
t%20,End%20,Timelimit%14,exitcode --units=G
```

- Gives jobid, partition, end state, Max Memory, Max Virtual Memory, Requested Cores/Mem, Start, Stop, Timelimit, Exit Code in GB format

sacct: state

- BF BOOT_FAIL
- CA CANCELLED
- CD COMPLETED
- CF CONFIGURING
- CG COMPLETING
- DL DEADLINE
- F FAILED
- NF NODE_FAIL
- PD PENDING
- PR PREEMPTED
- R RUNNING
- RS RESIZING
- S SUSPENDED
- TO TIMEOUT

Cancelling/Pausing Jobs

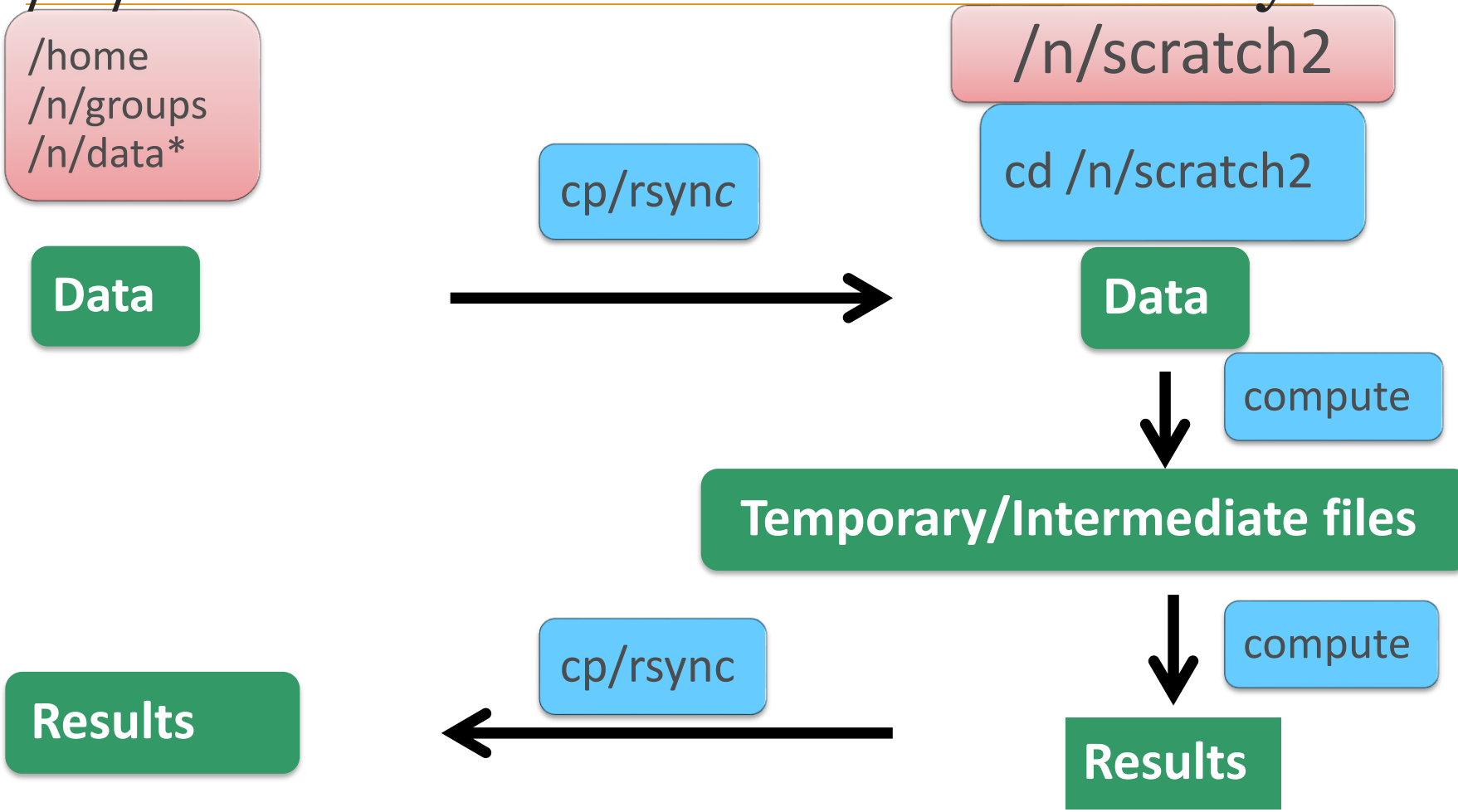
- `$ scancel <jobid>`
- `$ scancel -t PENDING`
- `$ scancel --name JOBNAME`
- `$ scancel jobid_[indices]` #array indices
- `$ scontrol hold <jobid>` #pause pending jobs
- `$ scontrol release <jobid>` #resume
- `$ scontrol requeue <jobid>` #cancel and rerun

Utilizing /n/scratch2

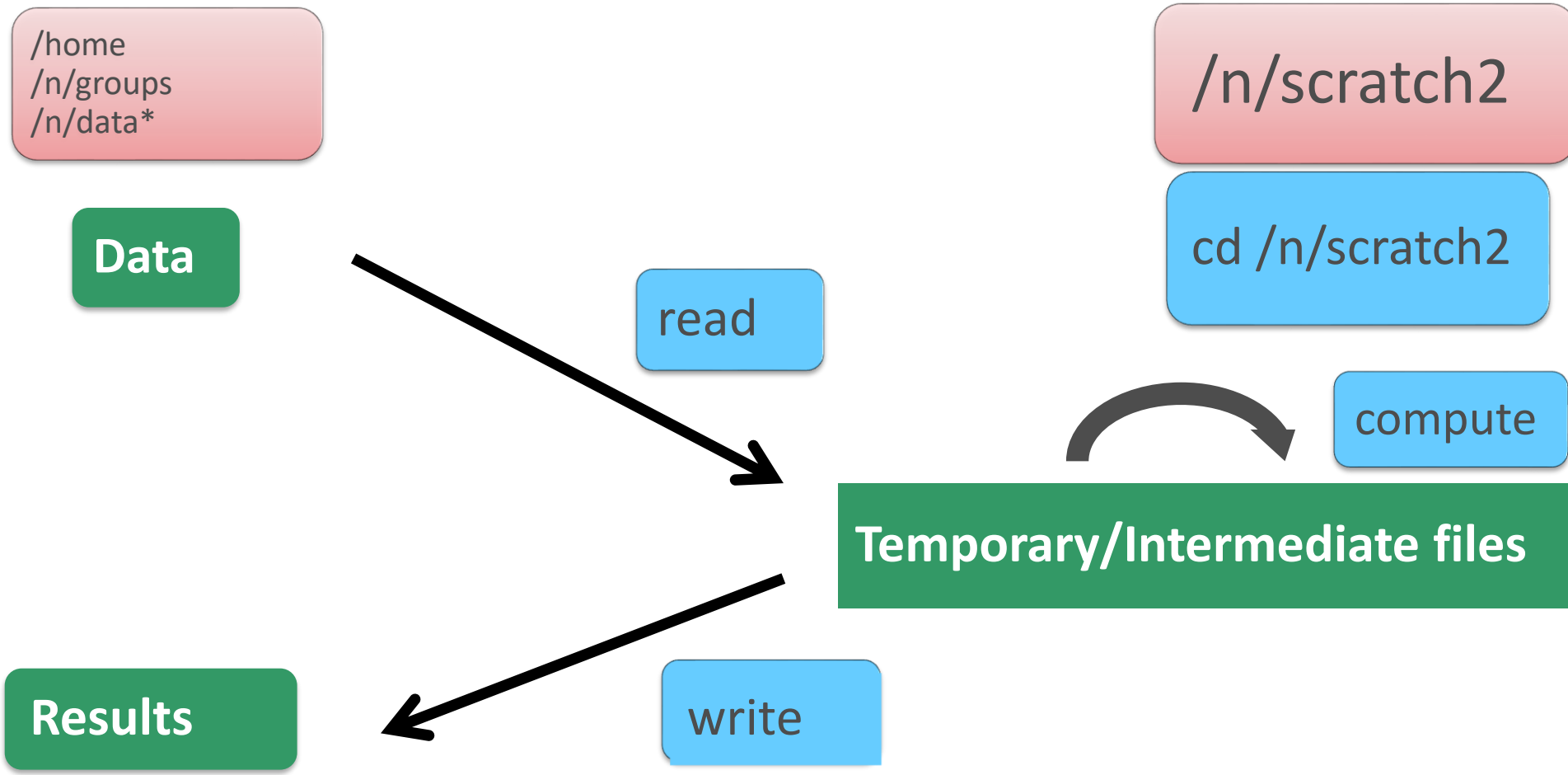
Utilizing /n/scratch2

- Designed for writing large, temporary files
- Use cases:
- Keep original files in /n/groups (/n/data*) or /home, write intermediate files to /n/scratch2, write final files to /n/groups (/n/data*) or /home
- Change working directory to /n/scratch2, read files from /n/groups (/n/data*) or /home, write temp files to working directory, write or copy output back to /n/groups (/n/data*) or /home
- Copy input files to /n/scratch2, compute against, copy output files to /n/groups (/n/data*) or /home

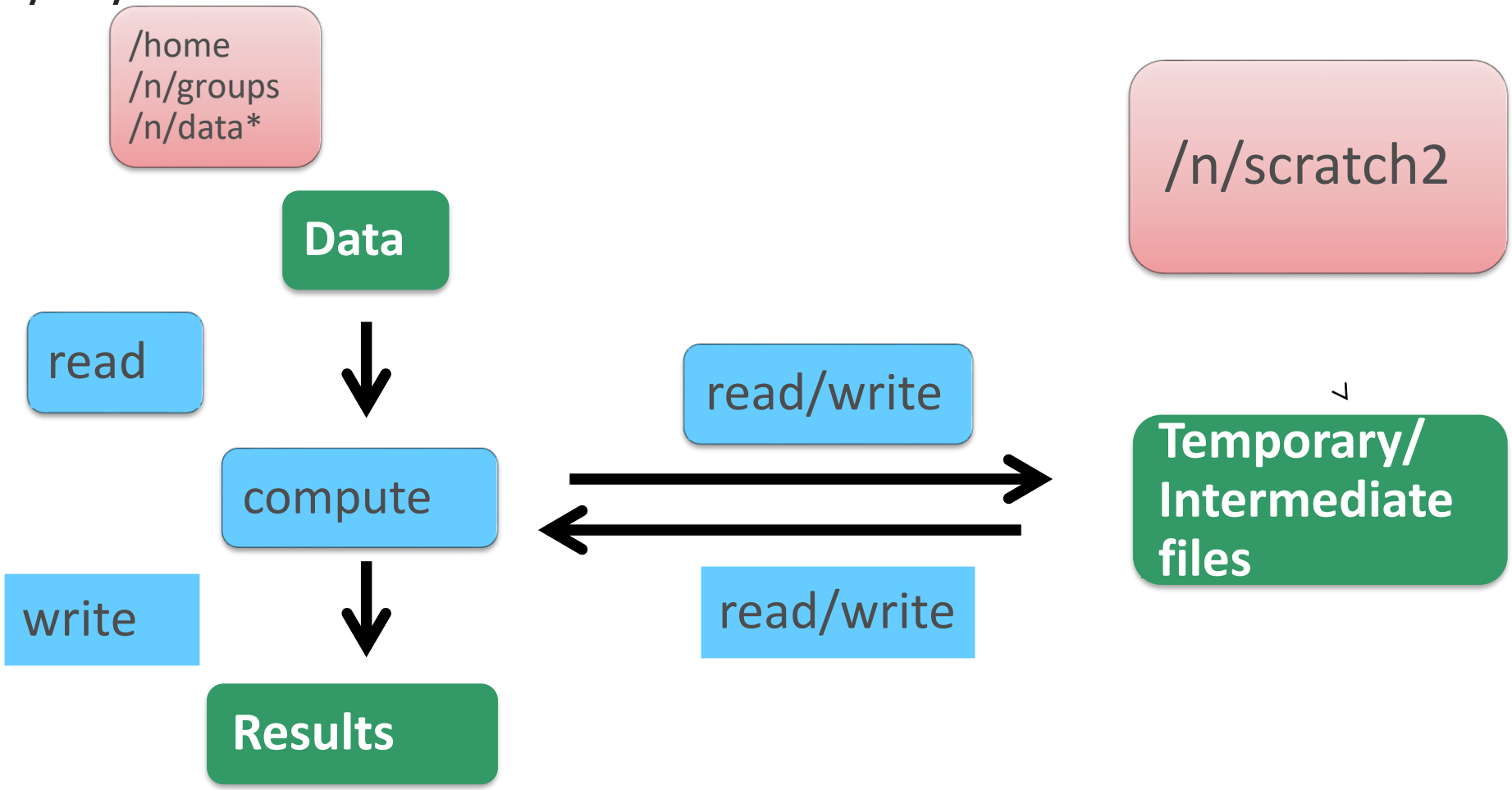
/n/scratch2 Workflow: Redundancy



/n/scratch2 Workflow: Medium Flexibility



/n/scratch2 Workflow: Best Practice



File Properties

- “chmod” to change who can read/write/execute files/directories

  chmod options file/directory

  Who? **u**ser **g**roup **o**thers **a**ll (u/g/o/a)

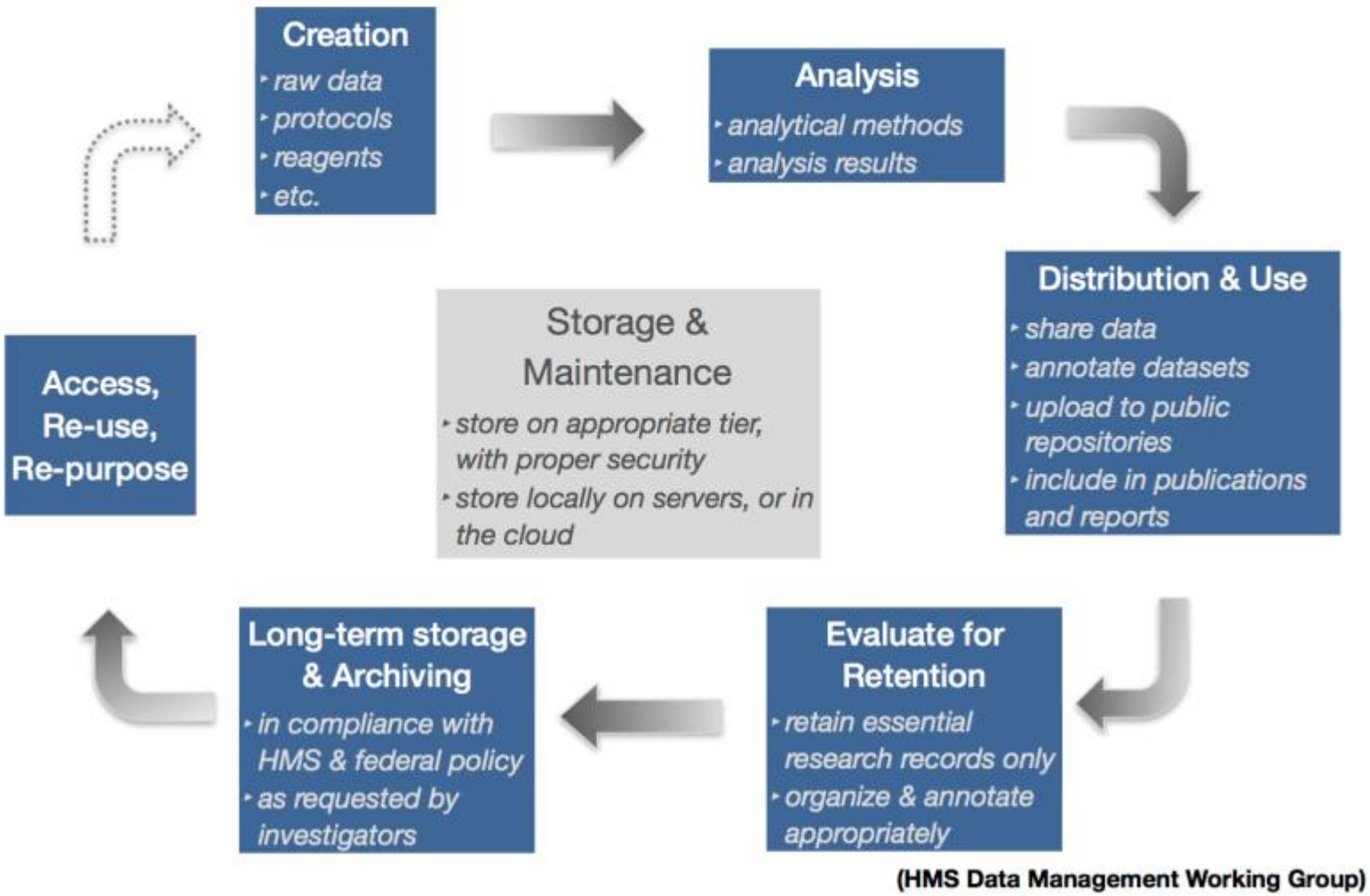
  What? **r**ead **w**rite e**x**ecute (r/w/x)

  Do? +/-

- chmod u+x myfile #makes a file executable to owner
- chmod o-rwx myfile #takes away permission from others to read/write execute

Data and Script Management

Data lifecycle for biomedical research



(HMS Data Management Working Group)

Data Management Planning

- When starting new projects, spend time thinking about how you will organize your data and metadata.
- What should the directory structure look like?
- Consider data type and file sizes, and total amount of data collected.
- Raw data:
  - Raw data should not be modified after data are collected.
  - Determine where raw data should be stored.
  - As data are annotated and analyzed, the resulting derived data files should be saved separately.
  - Consider identifying and using a read-only repository for raw data.

Thinking About Metadata

- Ask yourself:
  - What does metadata for this project look like?
  - What information should I keep track of?
  - Would a new project member be able to follow how data are created, stored, and documented?
- Create a readme.txt and store with each distinct dataset
  - Explain file naming conventions, abbreviations, codes, etc
  - Save as a plain text file
  - Avoid proprietary formats (e.g., Microsoft Word)

Example Readme File

Dataset title: Raw Images for Experiment A, Smith Lab

Principal Investigator: John Smith, PI, 555-555-5555, jsmith@hms.harvard.edu

Filename structure:
Structure:

ExperimentName_InstrumentID_CaptureDateTime_ImageID.tif

The base filename is composed of the name of the experiment, the ID number of the

instrument used, the date and time that the image was captured, and the unique identifier of the image.
Attributes:

ExperimentName = Name of the experiment.

Instrument ID = Five-digit code assigned to the lab instrument. See the Codes section for a list of instruments and their ID numbers.

CaptureDateTime = Date and time at which the image was captured, in YYYYMMDDThhmm format.

Image ID = Three-digit unique identifier for image, such as 001, 002, 003.
Codes:

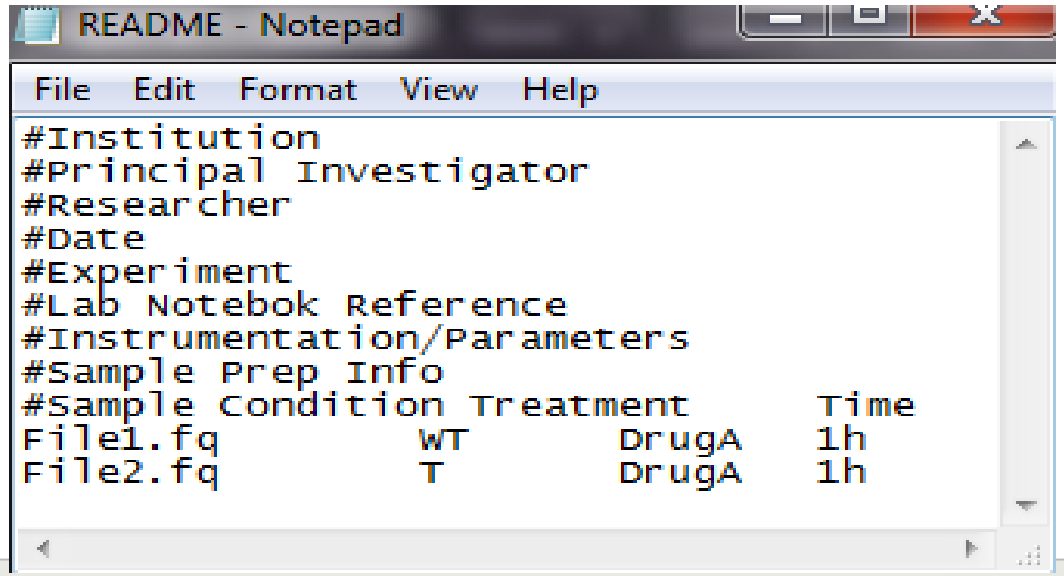
[List of instruments and IDs]
Examples:

daf2-age1_14052_20150412T0515_005.tif
File formats: tif

Versioning: All changes to this dataset will be documented in a changelog in this readme document.

Example Readme File

- Create in text editor (Notepad, TextEdit, vi, emacs, nano)
- Start metadata information with comment (hashtag)
- Save as tab-delimited .txt file (usually README.txt)



Naming Convention Best Practices Files (1 of 2)

- Use naming conventions consistently.
- Should be descriptive and provide contextual information.
- File Name Length:
  - Don’t make the name too long
  - Aim for 40-50 characters
  - Operating systems have different limits to the number of characters

Naming Convention Best Practices Files (2 of 2)

- Consider including a combination of the following:
  - Project or experiment name or acronym
  - Lab name/location
  - Researcher name/initials
  - Date or date range of experiment
  - Reference to lab notebook record
  - Type of data
  - Experiment conditions
  - Version number of file

Naming Convention Best Practices DOs

- DOs
  - Dates:
    - YYYYMMDD (e.g., 20160907)
  - Times:
    - use 24-hour military time to avoid confusion over a.m./p.m. (e.g., 1623 for 4:23 pm)
  - Sequential numbering:
    - use leading zeros (e.g., 001, 002, … 010, 011, … 100, 101, etc.)
  - Names:
    - surname then given (e.g., Smith_Bob)
  - Versioning:
    - use numbers to indicate updated versions (v1, v2)

Naming Convention Practice DON'Ts

- DON'Ts
  - Avoid special characters, such as
- ~ ! @ # $ % ^ & * ( ) ` ; : < > ? . , [ ] { } ' " |
  - Do not use carriage returns.
- Future-proof files: may need to be ported to a Linux/HPC environment later
- Special characters are often used for specific tasks in different operation systems.
- Commas are problematic when using comma separated values (csv) file format.

Naming Convention Practice DON'Ts Part II

- DON'Ts
  - Do not use spaces. Instead, try:
    - Underscores (e.g., file_name.xxx)
    - Dashes (e.g., file-name.xxx)
    - No separation (e.g., filename.xxx)
    - Camel case* (e.g., FileName.xxx)
- Some operating systems are case sensitive
- Some software will not recognize file names with spaces.
- File names with spaces must be enclosed in quotes when using the command line.

Naming Convention Best Practices Examples

- Example files with no naming conventions:
  - Test data 2016.xlsx
  - Final FINAL! last version.docx
- Example files with naming conventions:
  - 20160104_ProjectA_Ex1Test1_SmithE_v1.xlsx
  - 20160104_ProjectA_MeetingNotes_SmithE_v.1.docx

General best practices

- Be careful with Excel!
  - It introduces characters that are not recognized by UNIX
  - It modifies gene names into dates, e.g. SEPT2 ~ September 2
  - Be careful when sorting, make sure all columns are included
  - Be careful with numeric data, and any unexpected increments from one row to the next
- Don’t modify raw data
  - Raw data is sacrosanct!
  - Raw data can be anything you start your analysis with, e.g. fastq files, files from collaborators, data
  - Keep it as it is, if you want to change the names, make a symbolic link with the new name.

Long Term Storage

- HMS IT offers a Long-Term Storage service
- For storing large quantities of **infrequently** accessed data that do not need to be retrieved immediately
- Ideal for:
  - Completed or published projects
- For more information about the service, please visit:

http://rits.hms.harvard.edu/dm/lts

- Send questions to the Research Data Manager:

rdm@hms.harvard.edu

Publishing and Data Retention (1 of 2)

- Adhere to your lab's standard practices for data management and organization.
  - If you do not have standards, make them, write them down and follow them.
- Keep your data for **at least** seven (7) years.
  - IP (Intellectual Property), human subjects information and other factors can influence (extend) retention timelines, so before you delete data, check with your lab and sponsor guidelines first. If you have IP, talk to the Office of Technology Department (OTD).
- Store your data on University premises and/or systems.
  - If you are not sure what options are available to you, contact HUIT.

Publishing and Data Retention (2 of 2)

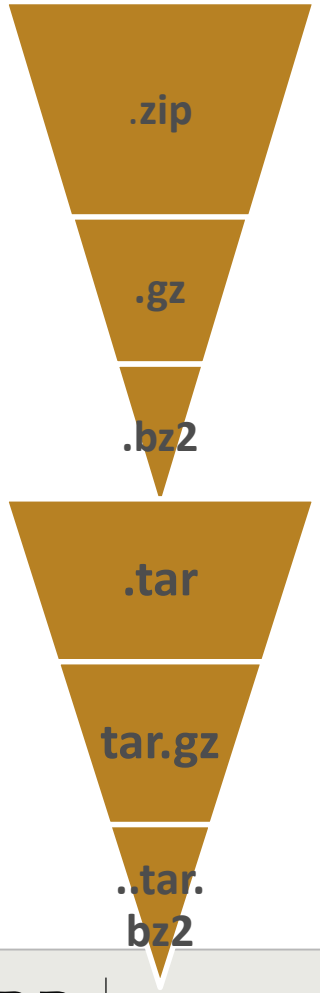
- If you leave HMS, and would like a copy of your data, please discuss this with your PI. HMS must retain the original data in order to meet its obligations to sponsors and the federal government.
- Remember: Harvard administration is here to help you organize and manage your materials. Stay in compliance and reduce administrative burden by calling HUIT or talking to your department administrator.
- HU's Retention and Maintenance of Research Records and Data: Principles and FAQs: http://osp.finance.harvard.edu/files/osp/files/research_records_and_data_retention_and_maintenance_guidance_2015.pdf

Data Deposition

- Certain NIH-funded research requires deposition of data into public repositories
- NIMH: dbGAP, GEO, NDAR
- Harvard DataVerse: Harvard Libraries, HUIT, IQSS
- Structural Biology Data Grid

Data Compression Methods

- zip: DEFLATE coding
- gzip: Lempel-Ziv coding (LZ77)
- bzip2: Burrows-Wheeler block sorting text and Huffman coding
- tar: archival utility preserving hierarchy and permissions, often used with gzip and bzip2



Scripting: Version Tracking

- Record changes (additions/deletions/replacements) to scripts
- Collaboration: many people can work on a file at once
- Helps with reverting to previous (working) versions
- Public or private repository options
- GitHub
- Bitbucket
- SVN
- Open Science Framework: integrate: Github/DropBox/Google Drive/AWS and more

OMERO

- Microscopy image and metadata management service of the Image Management Core
- Java Application or web interface
- Browse and filter through dimensions, z-sections and timepoints
- Analyze through Java, Python, C++ or MATLAB, Fiji/ImageJ using API/plugins to interface with OMERO server
- Orchestra: CLI environment module, Java desktop client, or web interface
- Upload data from research.files, /home, /n/groups, /n/data1, /n/data2
- imc-support@hms.harvard.edu
- http://imc.hms.harvard.edu

For more direction

- http://hmsrc.me/O2docs
- http://rc.hms.harvard.edu
- RC Office Hours: Wed 1-3p Gordon Hall 500
- rchelp@hms.harvard.edu