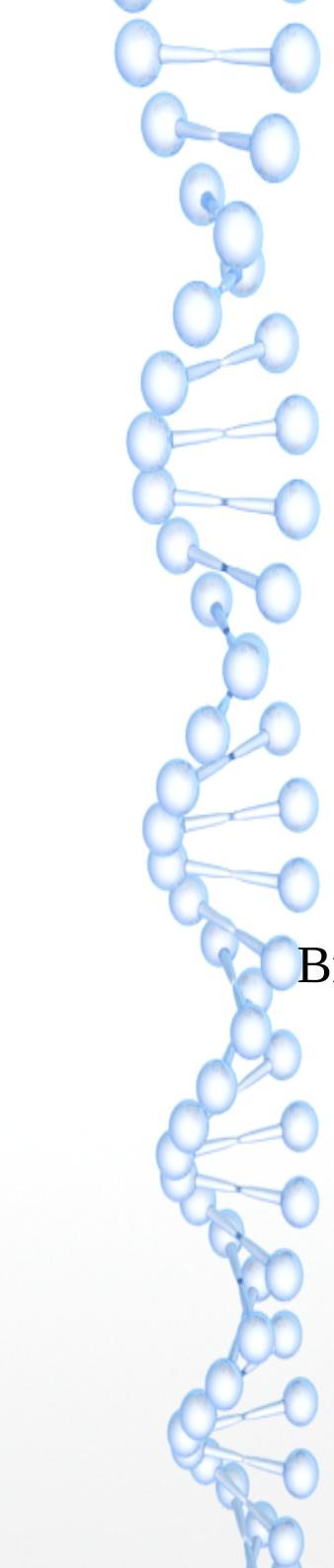


Introduction to Rosalind and Wilkins HPC

an introduction to Linux and high performance computing with SLURM

Brought to you by the TICR, sub-division of the Colorado Center for Personalized Medicine (CCPM)

Outline

- TICR, Rosalind, Wilkins
- Introduction to Basic Linux
- Introduction to Basic SLURM
- Utilizing HPC (Bioinformatics Example)

ITEMS/HANDOUTS

- Linux Cheat Sheet PDF
- Rosalind Map PDF
- Permissions PDF

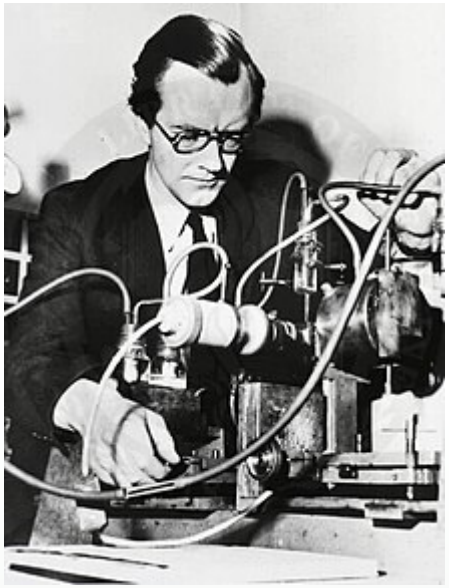
TICR, Rosalind, Wilkins

TICR
TRANSATIONAL INFORMATICS AND COMPUTATIONAL RESOURCE

OIT
DENVER | ANSCHUTZ

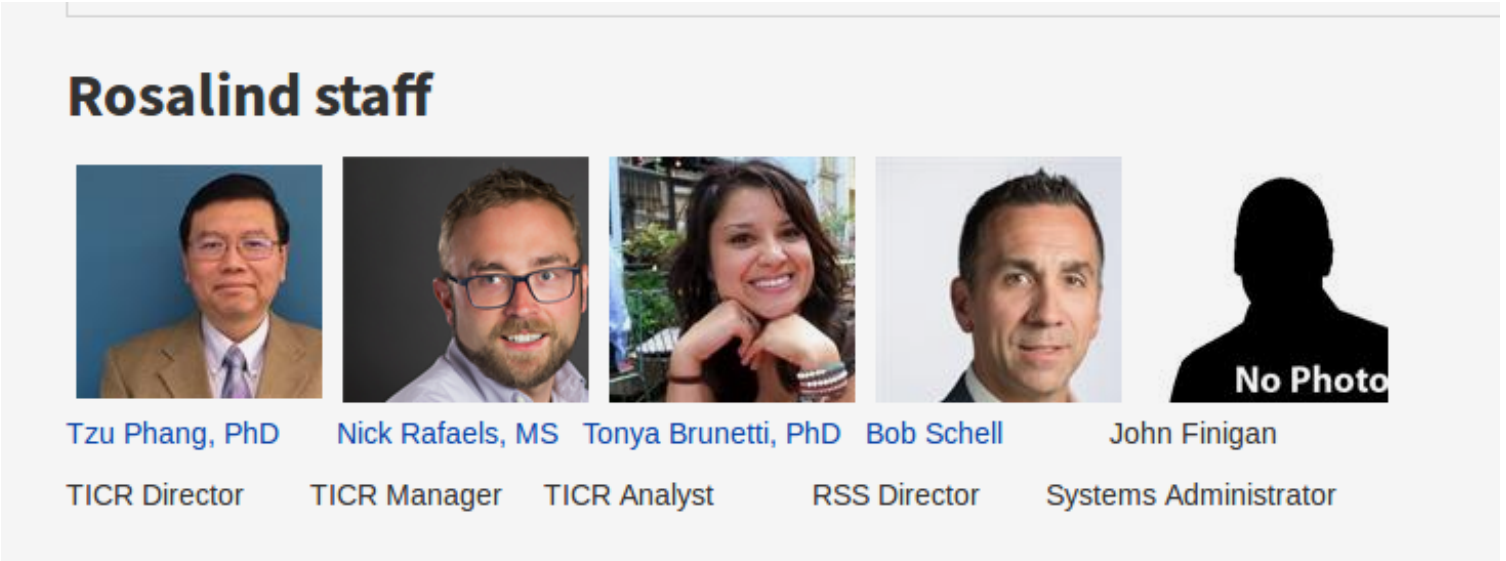
Research and Shared Services

COLORADO CENTER FOR PERSONALIZED MEDICINE

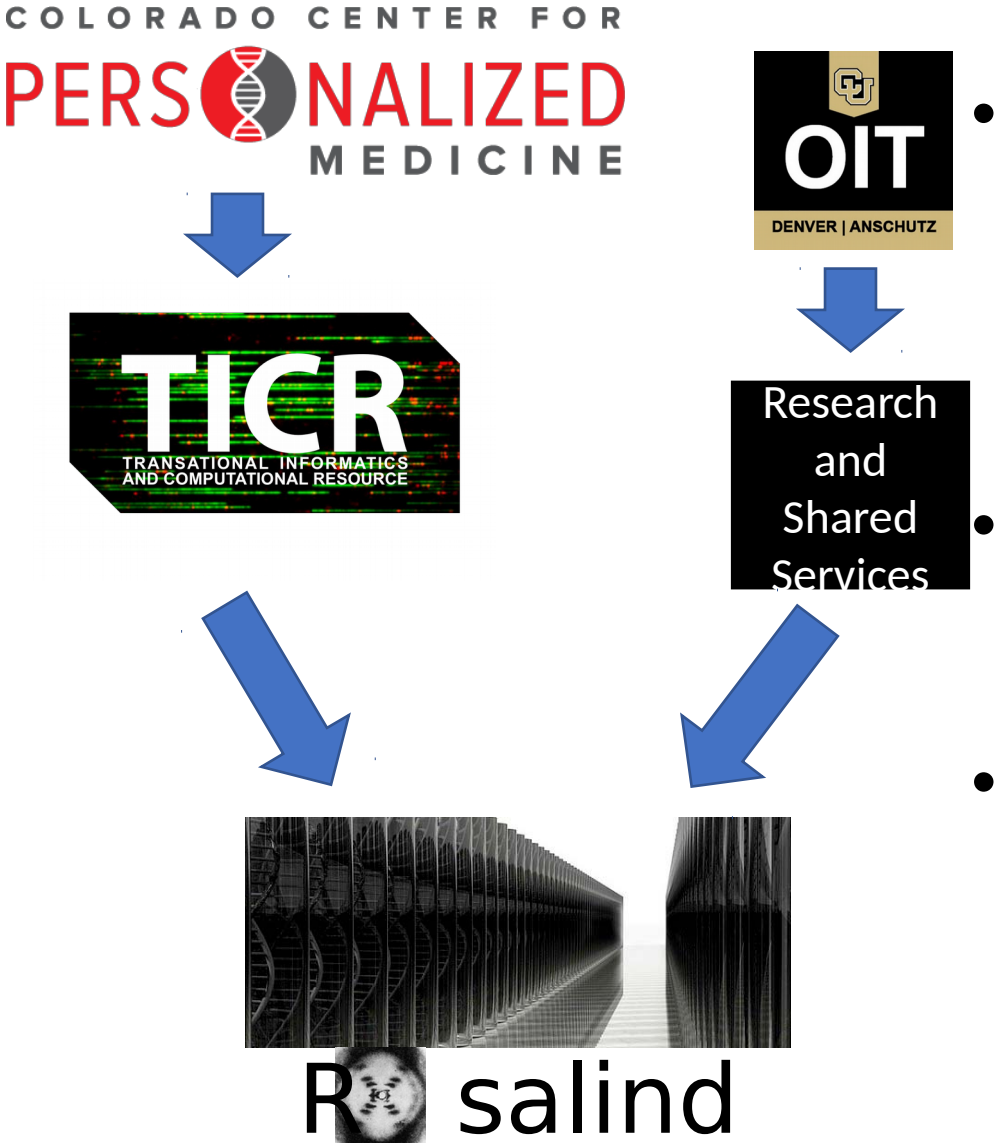


Who is TICR?

- **T**ranslational **I**nformatics and **C**omputational **R**esources
- **Roles and Responsibilities:**
  - Rosalind support and outreach
  - CCPM Biobank pipelines/analysis/validation/architecture development
  - computational informatics and programming support for CCPM
  - work closely with OIT and Health Data Compass to address CCPM and Biobank computational needs
- Vist our website at **www.ucdenver.edu/Rosalind**



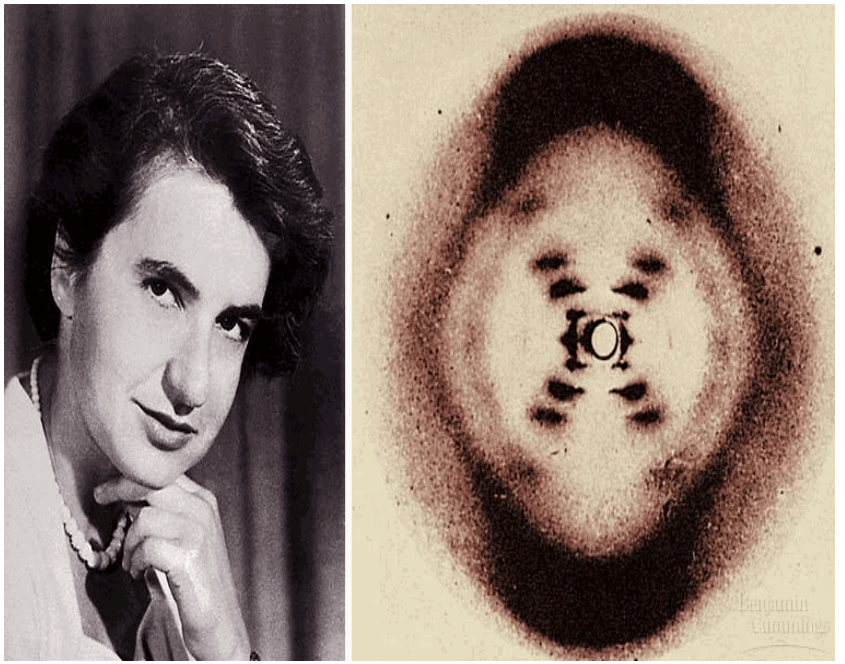
Introduction and Background

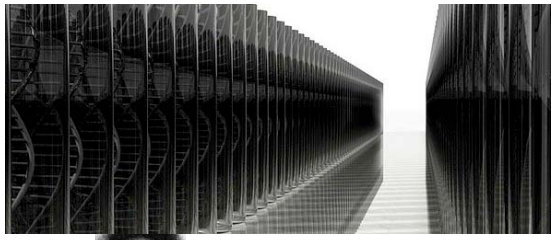


- The Translational Informatics and Computational Resource (TICR) within the Colorado Center for Personalized Medicine (CCPM) has partnered with
- Research and Shared Services (RSS) within the Office of Information Technology (OIT)
- To build a high performance computing system (HPC), Rosalind.

Who is Rosalind?

- Named after Rosalind Franklin, a chemist and X-ray crystallographer whose research was central to the discovery of the double-helix structure of DNA.
- Her colleague, Maurice Wilkins, passed along the X-ray image of DNA (on the right) to Watson and Crick in 1953, which confirmed the 3-D structure of DNA they had hypothesized
- Rosalind Franklin passed away from cancer in 1958 before Watson, Crick and Wilkins were awarded the Nobel Prize in 1962.





Rosalind

Swift
Secure
Solution to
Support
Science!

- Swift - Rosalind provides the capacity and the power to quickly process big data driven hypotheses
- Secure - Rosalind utilizes resources from OIT to store and backup your data safely, and provides a solution for analyzing highly sensitive data
- Solution - Big data queries drive innovations in science and health care
- Support - OIT and TICR provide support to facilitate your computational research needs
- Science - Choose Rosalind to discover new ways to look at science!

Rosalind HPC metrics and cost

- Compute and Storage
  - 768 cores at 128 GB RAM per node (4TB and 32 nodes total)
  - 72 cores at 1.5 TB RAM per node (3TB and 2 nodes total)
  - 3.7 PB of usable storage
- Cost
  - Compute Costs Standard Rate: $0.121 per core-hour
  - Storage Costs Standard Rate: $0.02 per GB/month
- Billing
  - Occurs on a monthly basis by speed type linked to project
  - Billing within project is sorted by individual based on compute and storage.

redhat.

slurm
workload manager

Rosalind HPC cost example

- Raw data and result storage: 500 GB
- Temporary data storage: 200 GB
- Computational needs: 40 core-hours / sample = 40 * 30 = 1,200 core-hours
- Estimated prototyping and statistical analysis computational needs: 20% of above: 240 core-hours

Sample(s)	30

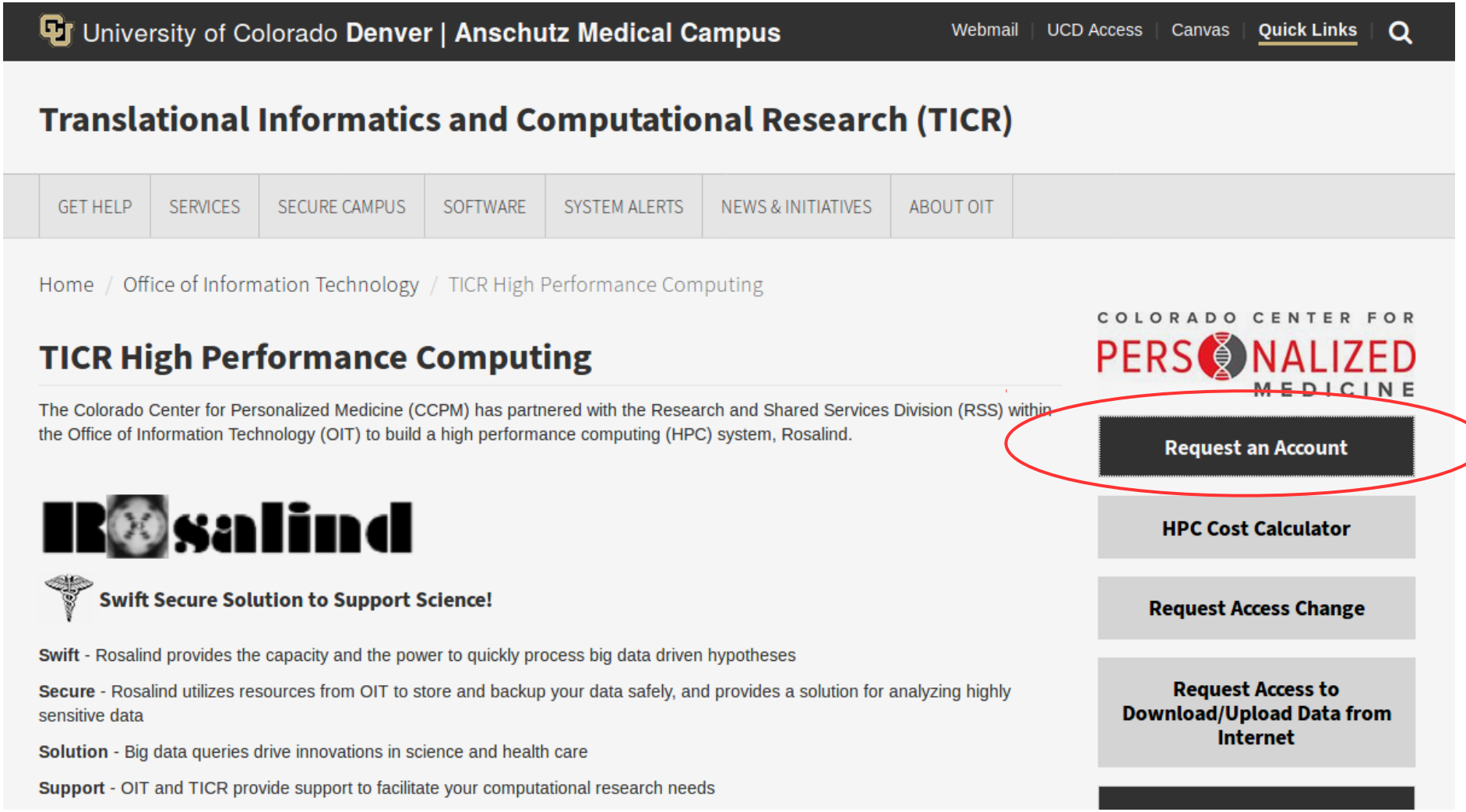
Usages	Cost	Your Needs	Time Needed	Total
Raw Data Storage	$0.02/GB/month	500	6	$60.00
Intermediate Files Storage	$0.02/GB/month	200	3	$12.00
Computation	$0.121/core-hour	1200		$145.20
Prototyping	$0.121/core-hour	240		$29.04
Estimated Cost				$246.24
Per Sample Cost				$8.21

You can price out your own project needs:

[HPC cost calculator]

How to apply for an account on Rosalind?

www.ucdenver.edu/Rosalind
CCPM-Rosalind@ucdenver.edu



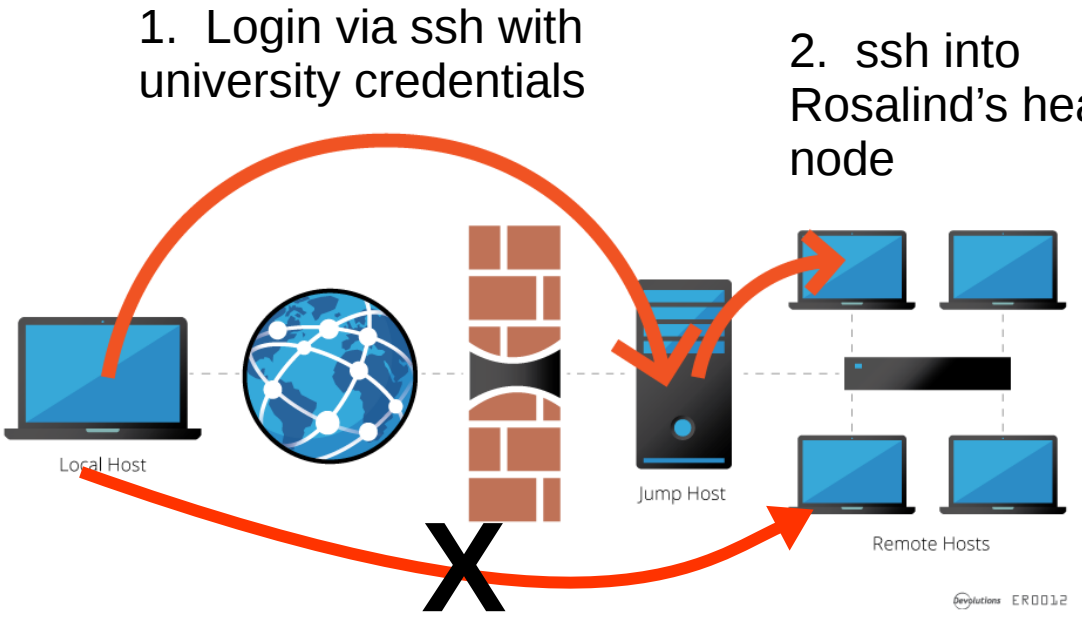
Why learn on the Wilkins demo environment when we could learn on Rosalind?

- Rosalind is HIPAA compliant
  - Secure!
  - You can put PHI on here, where most HPCs do not support this (Biobank with Compass?)
  - You can put sensitive information on here, where most HPCs do not support this
  - We are officially supported by AMC and OIT
  - We offer a lot of one-on-one, case-by-case customer service to help make your experience on the HPC better

In order to keep Rosalind HIPAA compliant we require certifications and training and Rosalind lives behind a jump host, so we mimic how Rosalind works on a non-HIPAA compliant demo system we named Wilkins

What is a jump host?

A jump host, also known as a jump server, jumpbox, or secure administrative host, is a server that allows you to connect or “jump” to other remote servers



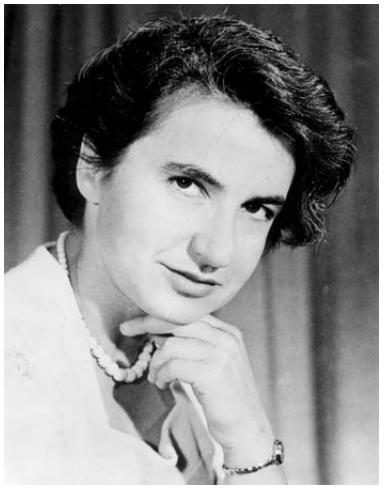
Why can’t I just connect to Rosalind directly?

- Rosalind is HIPAA-compliant meaning many users may have highly confidential and sensitive information about people such as electronic health records (EMR), protected health information (PHI), genomic data, transcripts and grades of students, etc… so we must take every precaution to protect this
- It minimizes risk to Rosalind by making malware and viruses harder to penetrate and compromise the system by forming a firewall from the Internet and outside world

Wilkins Demo vs Rosalind Environment

Goal: Mimic how Rosalind works as closely as possible on the Wilkins environment

- Not HIPAA compliant
- No jump host
- Very limited amount of RAM and space



- HIPAA compliant
- Has jump host
- Lots of RAM and space

Rosalind Support and Resources

- Our Website: www.ucdenver.edu/Rosalind

- Github help page/getting started
github.com/tbrunetti/Rosalind_HPC

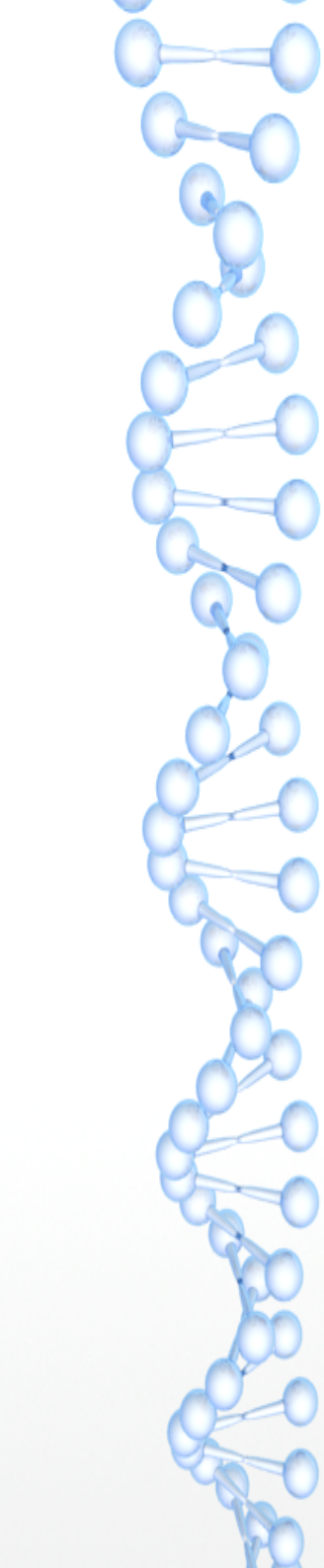
- Email: CCPM-Rosalind@ucdenver.edu
hpcsupport@ucdenver.edu

- Bootcamps

- Write HPC into your Grant

- Biomedical Data Science
(ucdenver.edu/biodatascience)

- Lynda.com; codecademy.com;

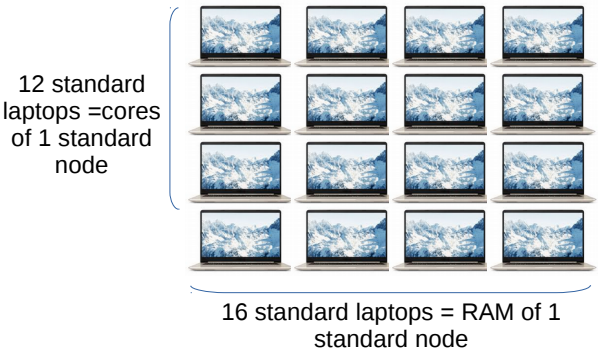


Why the HPC and why Linux?

Benefits of HPC

Standard laptop
~8GB RAM,
500GB hard drive,
dual-core
processor

Rosalind Standard Compute Node
128 GB RAM, 24 cores

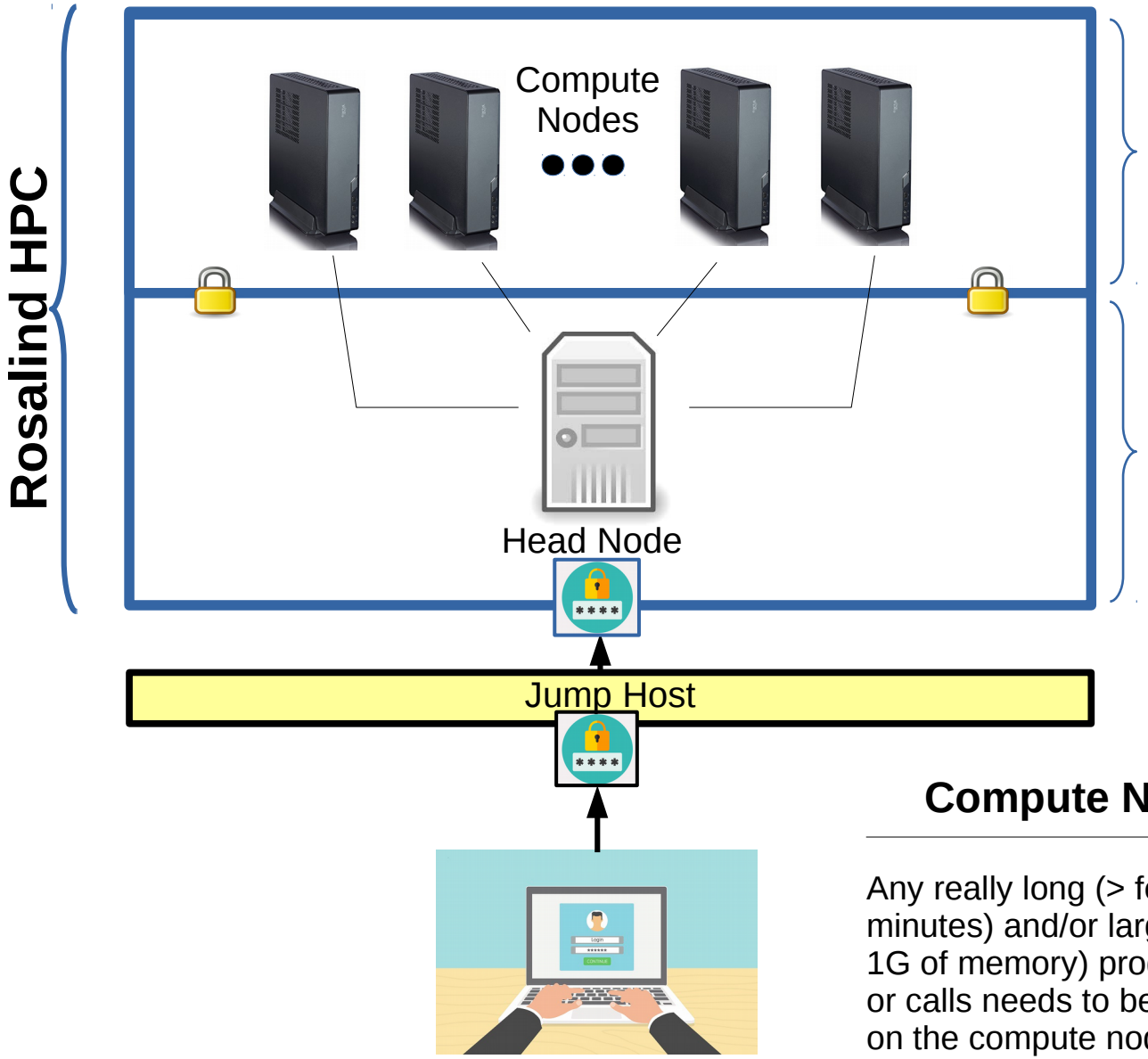


Rosalind Standard Compute Node Cluster
4096 GB RAM, 768 cores

The two high memory nodes add an additional 3TB of RAM and 72 cores!

GRAND TOTAL: 7,096 GB RAM (~7TB) and 840 cores with 3.7 PB storage
OR...

887 laptops to achieve the same RAM and 420 laptops to achieve the same number of cores!



The **compute nodes** are where all the power of the HPC is harnessed.

The only node the user directly interacts with on Rosalind is the **head node**.

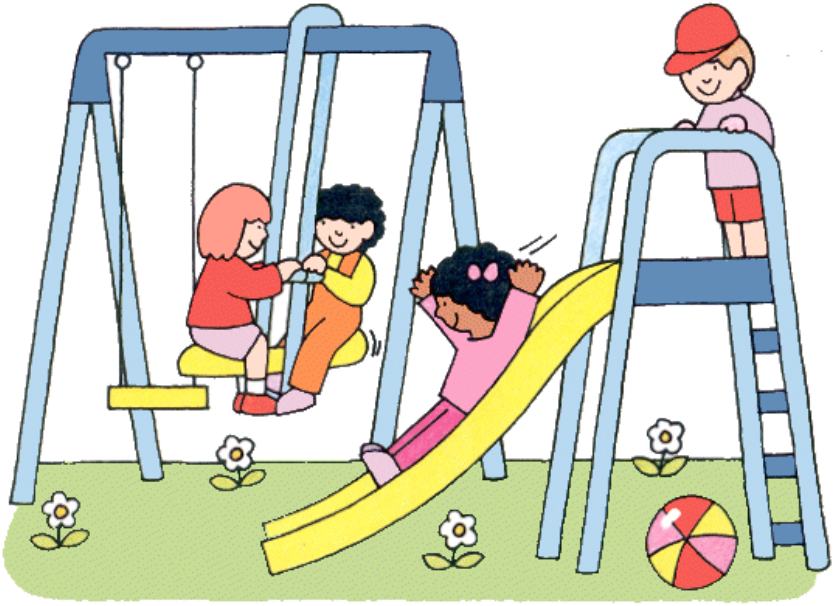
This node is the master or brain behind HPC.

Compute Node	Head Node
Any really long (> few minutes) and/or large (> 1G of memory) programs or calls needs to be used on the compute nodes. The compute nodes wait for the head node to assign them to a job or task.	Anytime a user makes calls through the command line on Rosalind, it is being performed on the head node. The head node is very small and limited in ability. Its main function is to work with the scheduler and delegate tasks and jobs to the compute nodes that users want to utilize.

Rosalind is a shared resource

What does that mean?

- This means that there are several different users and projects on Rosalind all sharing the same compute resources
- This means we have to be mindful of a few things:
    - we don’t want to install software for everyone since different groups will want their own versions
    - We want to make sure all your data is secure and only seen by you and your group members
    - We also want the resources to be shared among our users in the fairest way possible



How does that work?

Linux securities in place
Job scheduler (SLURM)

What is the difference between your personal computer and an HPC

- CPU, RAM and storage are very limited
- You have administrative master access
  - You can install anything you want for all users
- You can run anything you want at any time you want

- Large RAM/storage
- No sudo privileges
  - You can only install software for yourself in your own space, not across the whole HPC
- Fairly share resources using a resource management system

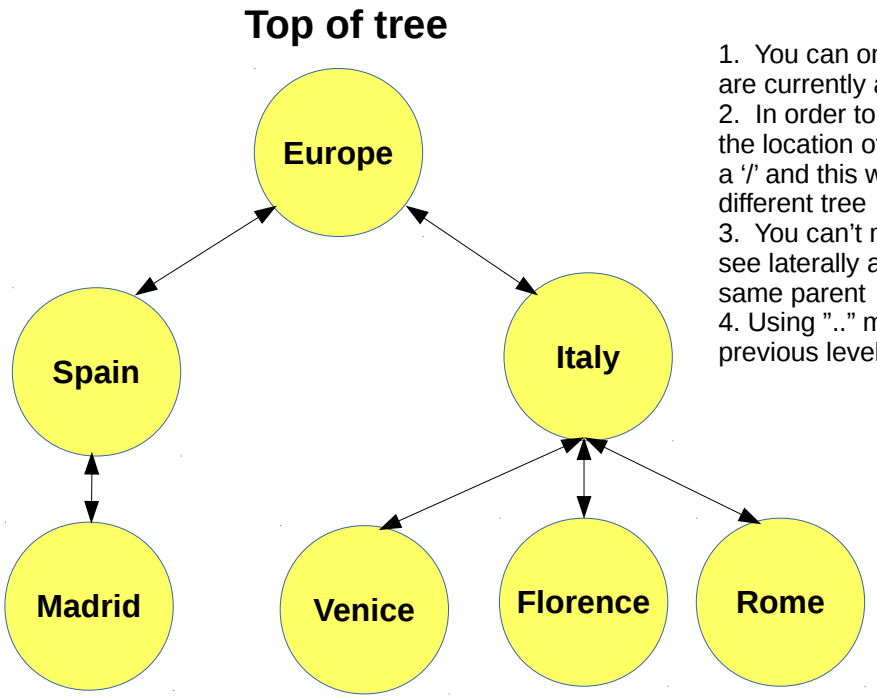
What is Linux and Why?

- Scalability
- Open Source OS
- Flexible
  - Ability to communicate with a variety of networks and OS
- Linux has has been around since 1991 with Unix making its first appearance 1969 therefore making is a fairly developed and well-understood OS

Mac OSX (Unix back end) 2001; all older version Macintosh OS

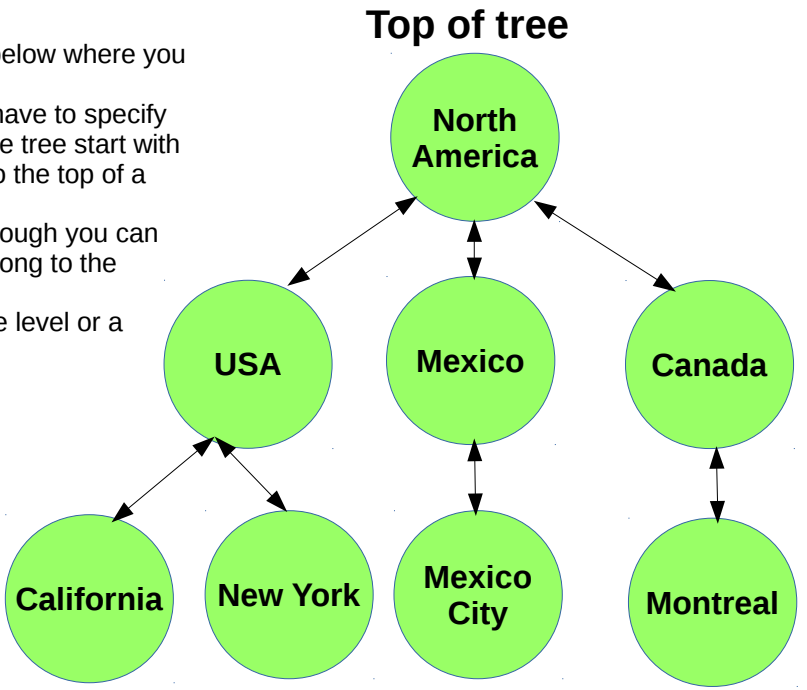
Since 1985; based on Microsoft Disk Operating System (MS-DOS) - single-user single-task operating system

Example of Linux Path Architecture



Rules:

1. You can only see one level below where you are currently at
2. In order to cross trees, you have to specify the location of the very top of the tree start with a '/' and this will "teleport" you to the top of a different tree
3. You can't move laterally, although you can see laterally as long as they belong to the same parent
4. Using ".." means to go up one level or a previous level



If I was in North America, I could get to New York by taking the following path:
USA/New York. If I then decided I wanted to go to California from New York I could do a few things:

- /North America/USA/California
- ../California ← Why does this work? Remember .. means to move up one level from current!

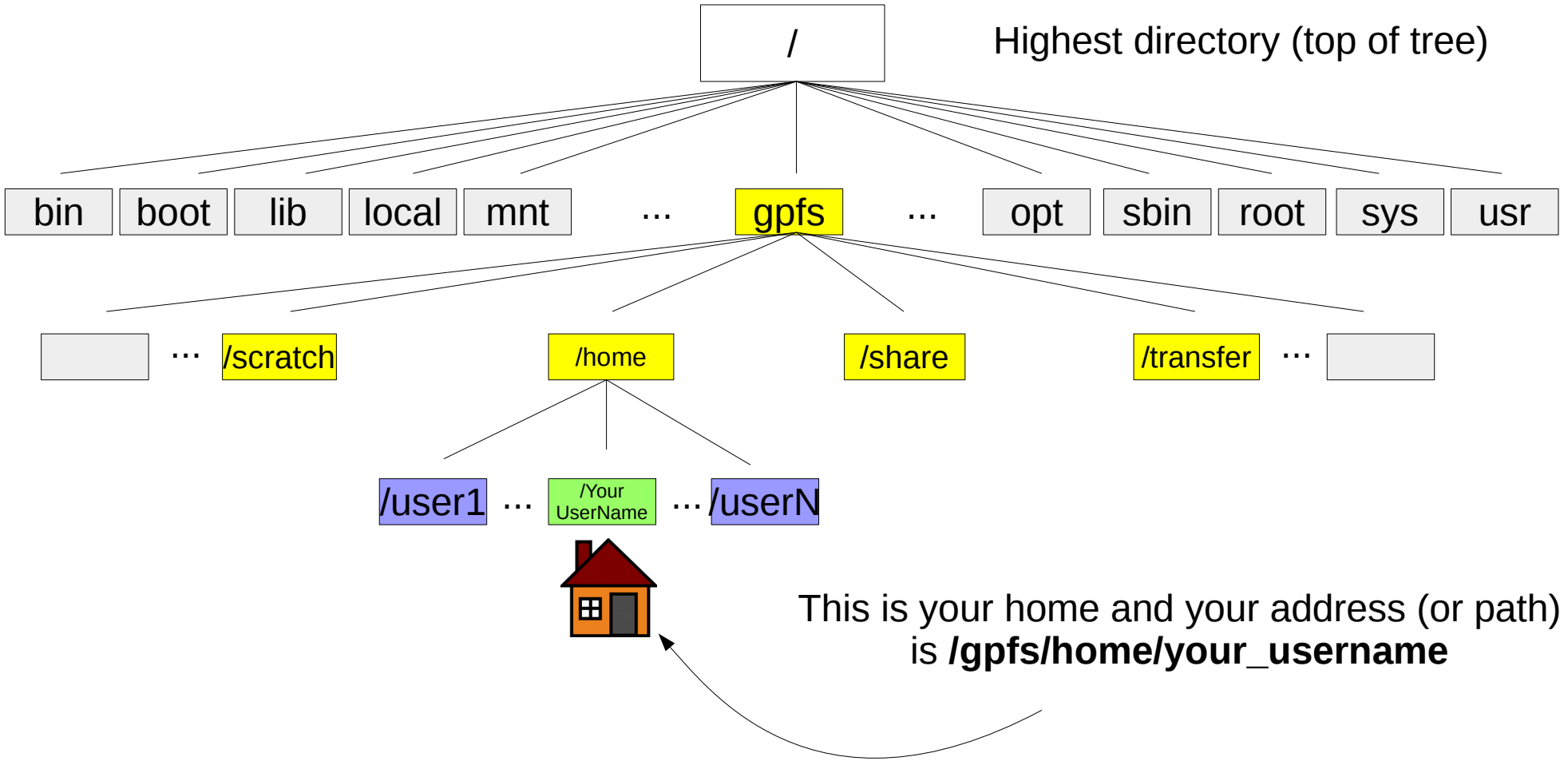
Starting at Europe, how do you get to Rome? From Rome what is the path to Venice?

/Italy/Rome ../Venice **or** /Europe/Italy/Venice

If you are in Rome, how do you get to Mexico? From Mexico what cities can you see?

/North America/Mexico Mexico City

Rosalind Road Map



= sudo required

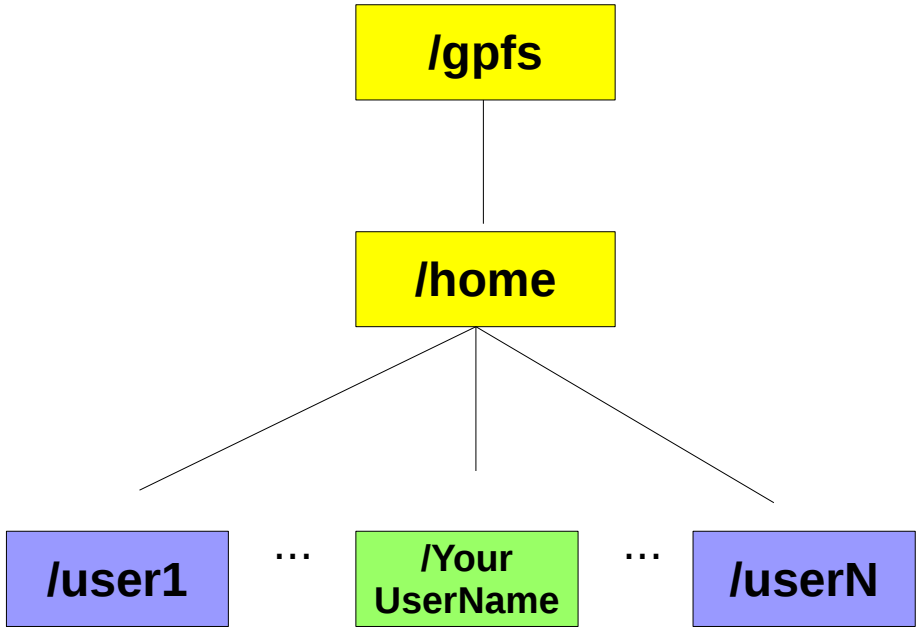
= limited access

= full access*

= no access w/o permission

But what person doesn't have a shortcut to get home?!...

Shortcuts to home



$HOME is a variable that will always take you back to /gpfs/home/your_usernname

$HOME = /gpfs/home/your_username

= sudo required
= limited access
= full access*
= no access w/o permission

Sometimes you will see /homelink/your_username and this is because we have created a link to called /homelink that links to /gpfs/home

Let’s login!

Was everyone able to install an ssh client or open a terminal that is not MS-DOS?

Open your favorite ssh client or open a terminal if you plan on using the command line to log in

Windows Users

PuTTY
MobaXterm
ZOC
Private Shell
Tera Term
LogMeTT
Cygwin w/openSSH

Mac Users

terminal.app
iTerm2
Termius
ZOC
PuTTY

Linux Users

Command prompt w/openSSH (CTRL+ALT+T) for debian/ubuntu (CTRL+SHIFT+T) for Redhat/Fedora/CentOS

```
[brunettt@myLocalComputer ~]$ ssh brunettt@cubipmtest02.ucdenver.pvt
```

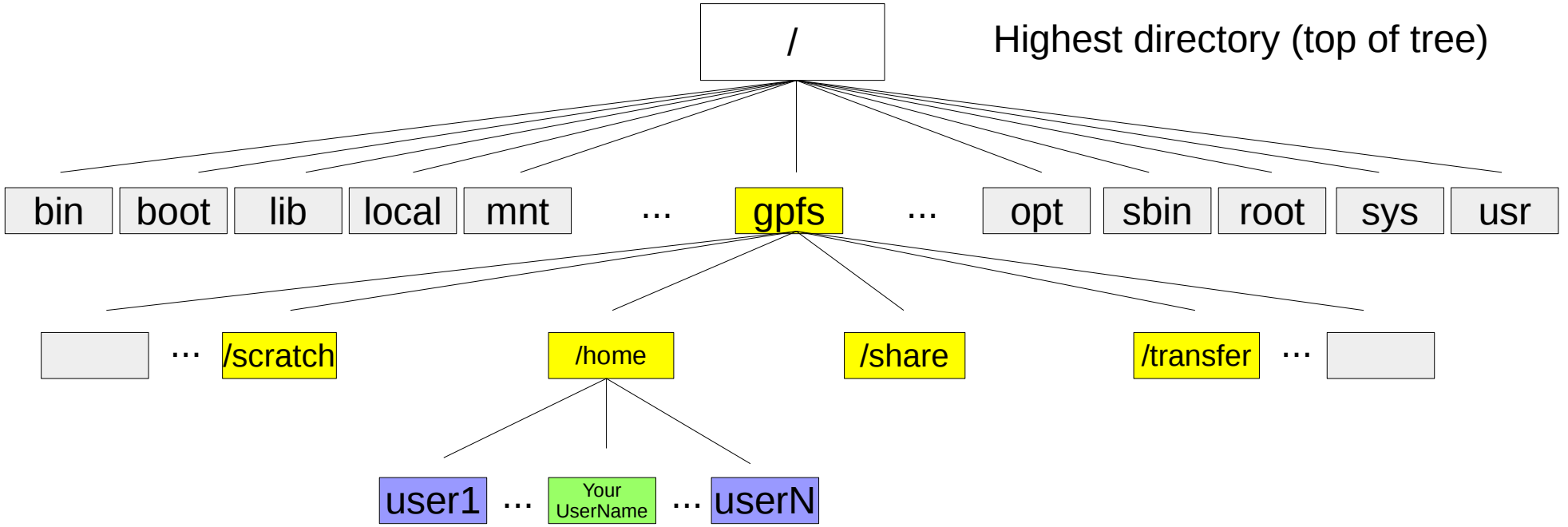
Where are you?

- **p**rint **w**orking **d**irectory

```
[brunettt@cubipmtest02 ~]$ pwd
```

```
/homelink/brunettt
```

- This shows you that the *folder (aka directory)* labeled as your username (brunettt) lives inside the directory homelink
- Each time you ssh into the head node, it will always take you to your home directory located on the head node



What should be stored in /homelink/your_username?

- small reference files
- scripts
- locally installed programs

Space is very limited in home!

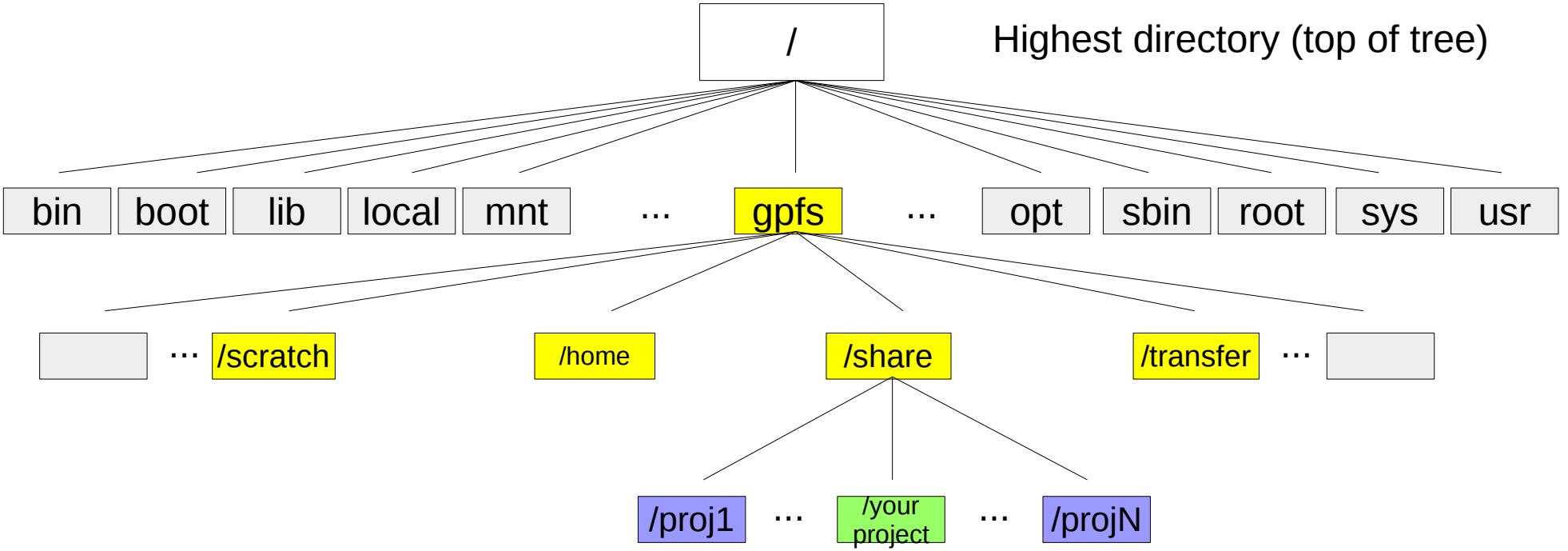
But I have a LOT of data!

= sudo required

= limited access

= full access*

= no access w/o permission



= sudo required

= limited access

= full access*

= no access w/o permission

What should be stored in /gpfs/share/your_project?

- programs, scripts, files that are shared between member of your project
- Large data sets and text files
- Output generated from running programs and scripts

LOTS OF SPACE! Most of your files and output should be stored here!

How do we travel from home to our project share?

Linux Shortcuts

- **Ctrl+c** = cancels/exits current prompt or program
- **up and down arrows** = cycles through previously called commands
- **tab + tab (double tab)** = auto fills name of file or directory if it exists in the present working directory

Linux Tips and Best Practices

- Linux is case sensitive! HelloWorld is not the same as helloworld which is not the same as HelloWORLD, etc…
- Avoid using spaces in names of files and directories Linux uses spaces as a delimiter for arguments in a program or command
- Avoid using special characters in the names of files and directories. Many of these characters have special meaning in the Linux world. @#$!%^&)”\’.*()?/~+

Navigation through Rosalind

- **c**hange **d**irectory

  cd is used to change directories. You can follow the cd command with any full length path or any relative path

- The general format for using it is the following: **`cd <your destination address>`**

```
[brunettt@cubipmtest02 ~]$ cd   /gpfs/share/training
```

```
[brunettt@cubipmtest02 training]$ pwd
/gpfs/share/training/
```

1 Using `cd`, try to navigate to your project share directory. 2 After you change directories, how can you confirm your location?

1

```
cd ../../gpfs/share/myProject
cd /gpfs/share/myProject
```

2

```
pwd
```

What items are in your project share?

- **l**ist **s**egments

```
[brunettt@cubipmtest02 training]$ ls
```

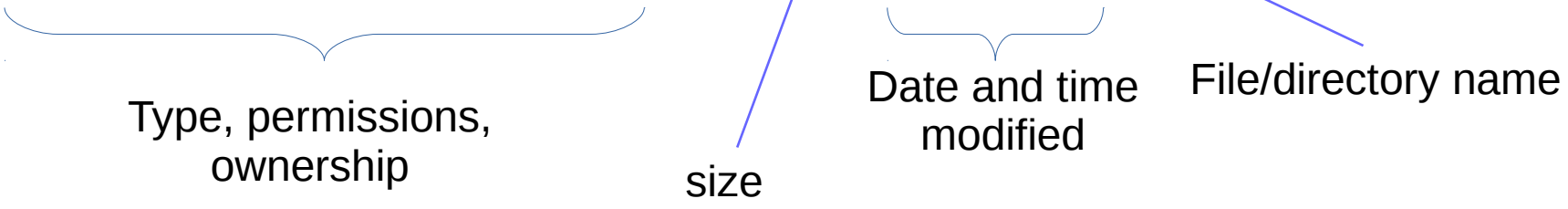
```
drosophila_fastq  drosophila_ref_files README.txt
```

- This will list all the files and folders in your present working directory
- You may be interested in knowing more about these files so we can give the `ls` command 2 optional arguments

```
[brunettt@cubipmtest02 training] ls -lh
```

```
[brunettt@cubipmtest02 training] ls -lh
```

```
[brunettt@cubipmtest02 training]
drwx--S---. 2 brunettt ticr_wilkins_user  156 Jun  8 12:30 drosophila_fastq
drwx--S---. 2 brunettt ticr_wilkins_user 4.0K Jun  8 11:20 drosophila_ref_files
-rw---S---. 1 brunettt ticr_wilkins_user  809 Jun  8 12:41 README.txt
```



Permissions and ownership are very important and allow you to control who can access your data files. There is a supplemental PDF that explains this but for the sake of we will not cover this in the introduction bootcamp

Create your first directory

- **m**a**k**e **dir**ectory

General format:

`mkdir myNewDir`

```
[brunettt@cubipmtest02 training] mkdir Tonya
```

Let's make a new directory in **`/gpfs/share/training`** labeled with your name (no spaces!)

Now confirm this directory was made by using the **`ls`** command. Do you see it?

Let’s make a copy

- **c**o**p**y

  General format:

  `cp myFile /path/to/destination`

```
[brunettt@cubipmtest02 training] cp README.txt Tonya/
```

Let’s copy the README.txt into your newly created directory

Navigate into your new directory. Confirm using **`ls`** that you made a copy of the README.txt file

How can we look at what is inside a file?

- **less**

General format:

`less myFile.txt`

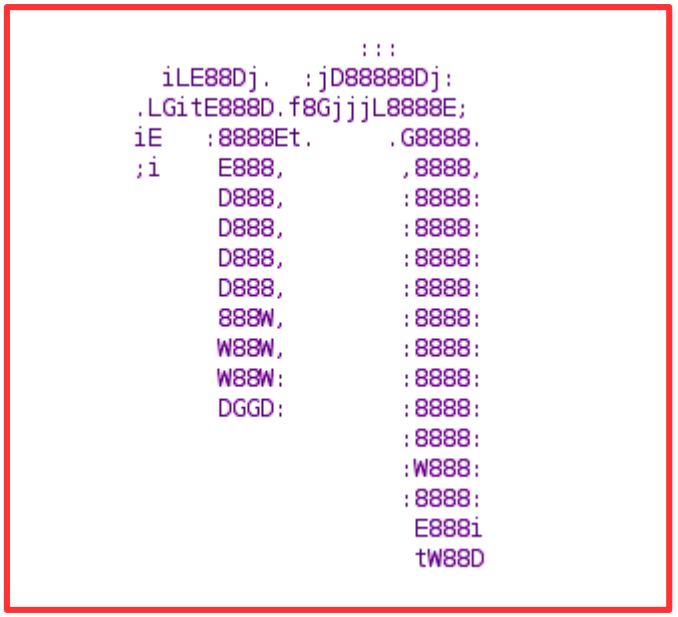
```
[brunettt@cubipmtest02 Tonya] less README.txt
```

Open, read, and scroll through a file.

You cannot modify or write in the file!

What if I want to type something in the README.txt file?

- You may have noticed that **`less`** does not allow you to modify any files, just read them. In order to modify a file you need to use a **text editor**
- There are many flavors of text editors and many are installed on most HPCs: nano, vi/vim, emacs are some of the more commonly used editors



Most user friendly → Steeper learning curve

- Navigate to your share folder and let's open README.txt using nano

```
[brunettt@cubipmtest02 training] nano README.txt
```

```
 GNU nano 2.3.1                          File: README.txt

Welcome to the Introduction to Rosalind HPC Tutorial!  We are happy to have you here!

Here is an explanation of all the contents that are in your project share

Directory: drosophila_fastq
        -This contains sequencing data from the model organism Drosophila melanogaster in fastq format.  Fastq (.fq, .fastq) is the raw data f$


Directory: drosophila_ref_files
        -This contains all the reference files and reference file indicies (required for alignmnet) for Drosophila melanogaster.  This is what$
```

- nano is great because it gives you a cheat sheet at the bottom of how to perform everything! **"^" means to use the control button**

```
                                [ Read 10 lines ]
^G Get Help    ^O WriteOut    ^R Read File    ^Y Prev Page    ^K Cut Text    ^C Cur Pos
^X Exit        ^J Justify     ^W Where Is     ^V Next Page    ^U UnCut Text  ^T To Spell
```

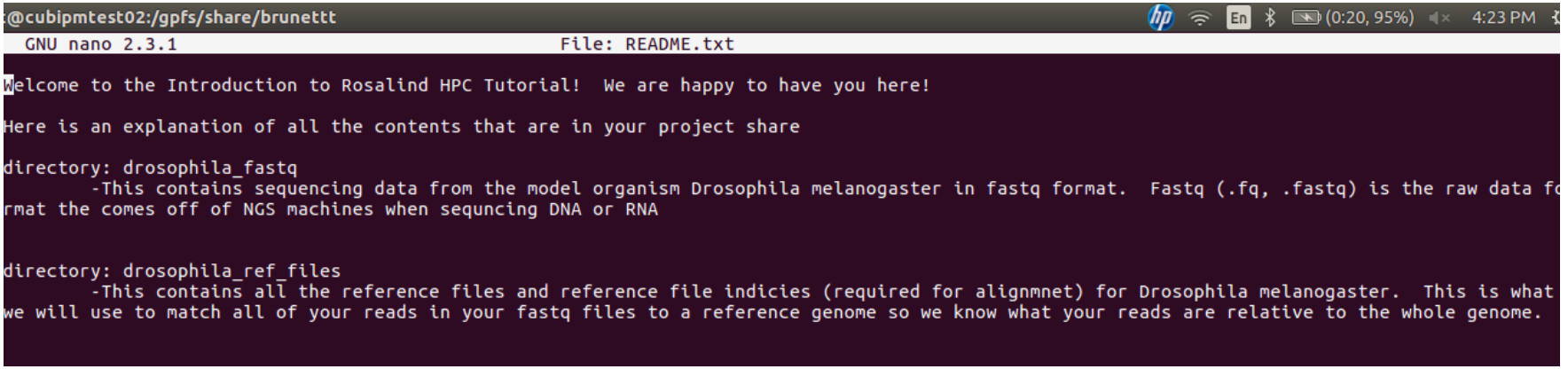
- First call the "Get Help" command (^G or ctrl+G). You will see that at the bottom it tells you how to wrap the text so that the line doesn't run over to the next page

```
(M-B)                   Backup files enable/disable
(M-F)                   Multiple file buffers enable/disable
(M-M)                   Mouse support enable/disable
(M-N)                   No conversion from DOS/Mac format enable/disable
(M-Z)                   Suspension enable/disable
(M-$)                   Soft line wrapping enable/disable

^Y Prev Page            ^P Prev Line            ^X Exit
^V Next Page            ^N Next Line
```

means after exiting help (ctrl+x) press "esc" and then press "shift+4" since that is the dollar symbol

- Now you can see the text in the README.txt files has been wrapped so all the words do not overflow off the screen.



- Now scroll to the bottom of the file and add the following lines:

```
directory:  yourName
        -This will contain output files from some bioinformatics software
```

- Then go ahead and save your file. Give it a new file name by using ctrl+O and typing in a new file name and pressing enter. Please put your name or initials Somewhere in the file name. Then exit nano.

Running your first shell script

- Navigate to your $HOME and into your scripts directory.

```
[brunettt@cubipmtest02 scripts] ./myJob_onHeadNode.sh
```

This will tell Linux to run a shell script or program

- What does the output look like?

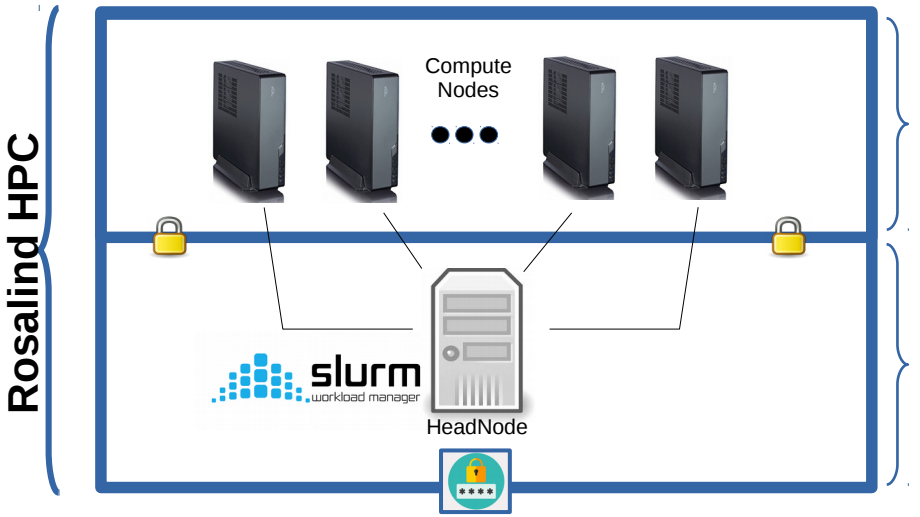
```
[brunettt@cubipmtest02 scripts] ./myJob_onHeadNode.sh
/homelink/brunettt/whole_kit/scripts
“Hello World!”
```

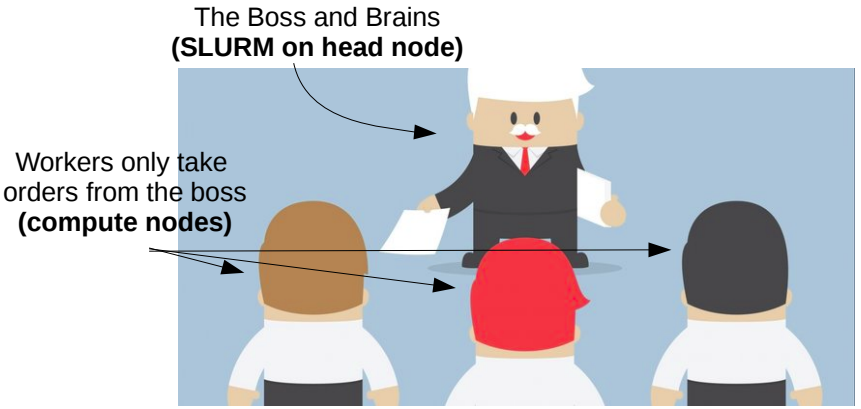
Great, we were able to run our first script on the head node! But remember, the head node is not built to run programs. How do we modify this so that it uses the compute nodes?

Scheduling and Submitting a Job...or Jobs!

Rosalind Utilizes SLURM

- **S**imple **L**inux **U**tility for **R**esource **M**anagement
- SLURM Workload Manager is just one of many flavors of job scheduling softwares available and happens to be the one that we chose for Rosalind
- SLURM lives on the head node
- In order to use any of the the compute nodes, you must first communicate your request with SLURM on the head node.
- SLURM will process your request and assign your request to an available compute node.
- You never have direct access to the compute nodes!





HPCs use job schedulers

- What is a job scheduler?
  - responsible for executing multiple job requests (i.e. your programs) whether they come from the same or different user(s) and determining what resources are available, and whose jobs are next in line to be run
- Why do we need a job scheduler?
  - "Fair-Share"
    - Sophisticated algorithm to schedule jobs fairly on a shared resource
    - The scheduler knows when each user made submitted requests so it can plan jobs effectively and know how long a job has been waiting in the queue or line
    - In the event all resources are being utilized, the "fair-share" policy is implemented meaning those users who have used the least amount of computer hours/resources within a window of time, get pushed to the front of the queue.
      - Prevents one individual from constantly using all the resources as they become available

HPCs use job schedulers

- Efficiency
  - Jobs that are shorter or require less resources typically are in the queue shorter since resources become available more quickly and are more likely to fit into an untapped resource
  - Scheduler knows how many resources every user has requested and the time and memory constraints of those resources.

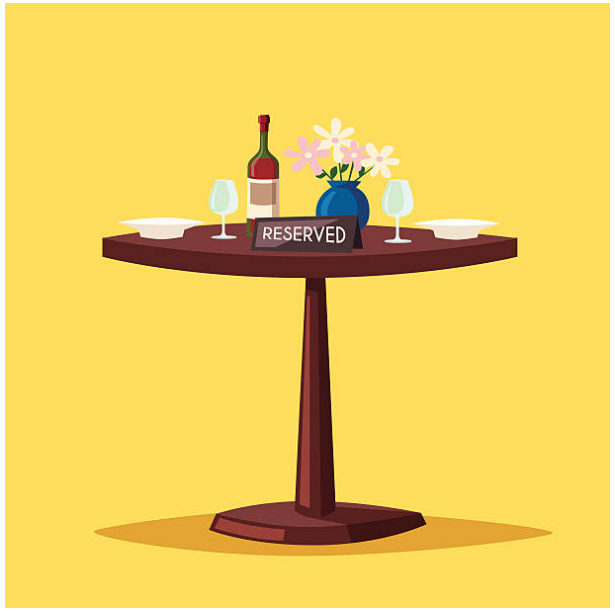
How do you communicate with SLURM?

- SLURM has its own special language that are specific to SLURM.
- Almost all SLURM commands will begin with a lowercase ‘s’
- **sbatch** tells SLURM that you are about to submit an order using a batch script to use the compute nodes to run one of your jobs.


```
[brunettt@demo.system01 ~]$ sbatch myJob_ComputeNodes.sh
```

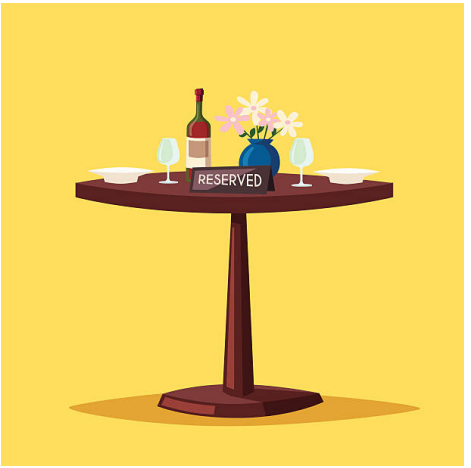
What is a BATCH script?

SLURM requires a specially formatted shell script so it knows how to allocate resources, how to utilize those resources, and how to run your program



1. Request and Allocate Resources

- We have created a short example batch script in your $HOME/whole_kit/scripts called **`myJob_ComputeNodes.sh`**. Copy this script and rename the file **`test_yourUsername.sh`** Open this script.
- All allocation requests are made at the top of your file after the line: #!/bin/bash
- All reservation information all must start with #SBATCH



"Make a reservation at our popular restaurant!"

```
#!/bin/bash

#SBATCH --time=5
#SBATCH --mem=1000
#SBATCH --job-name=testing
#SBATCH --output=testing.out
#SBATCH --error=testing.err
#SBATCH --node=1
#SBATCH --ntasks=1
#SBATCH -p defq
```

Max time in minutes, also take days-hours:minutes format

Max memory (RAM) in MB

Name of job

Name of file to write standard output

Name of file to write standard error

Number of Nodes needed

Number of tasks per node

The node type or "queue" to run job

2. Give the script instructions and submit the request

- This is where you will place your order, i.e. tell SLURM the exact order of the exact commands you want to run on the compute nodes

```
#!/bin/bash

#SBATCH --time=3
#SBATCH --mem=1000
#SBATCH --job-name=testing
#SBATCH --output=testing.out
#SBATCH --error=testing.err
#SBATCH --node=1
#SBATCH --ntasks=1
#SBATCH -p defq

pwd
echo "Hello World!"
sleep 120
```

Write down your "order" for the waiter

Rename the output to be **`test_yourUsername.log`** and rename the error log to be **`test_yourUsername.err`**

```
[brunettt@demo.system01 ~]$ sbatch test_brunettt.sh
Submitted batch job 233830
```

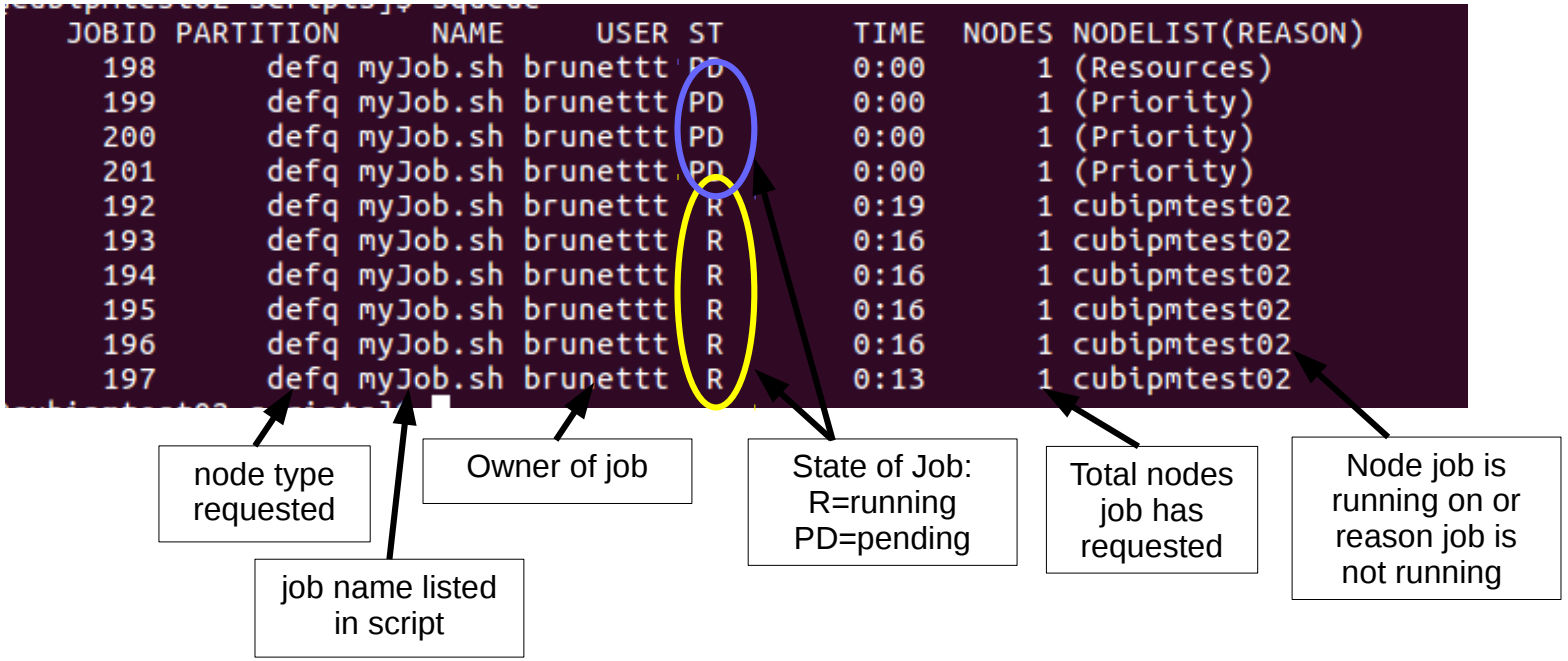
Waiter submits your "order" in the form of a shell script to the boss who will assign it to a chef, in this case are the compute nodes

Checking your place in line

```
[brunettt@cubipmtest02 ~]$ squeue
```

```
[brunettt@cubipmtest02 ~]$ squeue -u brunettt
```

```
[brunettt@cubipmtest02 ~]$ squeue --job 233839
```



Oops I made a mistake, how do I cancel a job>

```
[brunettt@cubipmtest02 ~]$ scancel 210
```

Job id to cancel

Want to cancel everything you are running?

```
[brunettt@cubipmtest02 ~]$ scancel -u brunettt
```

your username

You can only cancel jobs that you submitted!

- What happens if I go over the time and/or memory reserved?

  Your job will be killed! SLURM cannot “add on” or extend additional resources!!!

- Can’t I just request the max time and memory all the time?

  You can, but remember, “fair-share” policy and efficiency will potentially prevent your jobs from running in a timely manner due to lack of available requested resources. Your job will only start running after those resources are available and you are next in line to receive those resources

3. Collect output and pay the bill

- Navigate to where your test_brunettt.sh was run
- You should 2 outputs: testing_brunettt.log and testing_brunettt.err. Open this. What do you see?

- SLURM calculates how much storage and the number of core-hours you have used during the month and will bill your speedtype at the beginning of each month

Running Bioinformatics Software

Bioinformatic Sofware: *fastqc* example

- We are now going to look at the quality of some of the reads in the **`drosophila_fastq`** directory
- Navigate to your scripts directory inside **`$HOME/whole_kit`** and open **`check_seq_qual.sh`** with the nano text editor

```
 GNU nano 2.3.1                    File: check_seq_qual.sh

#!/bin/bash

#SBATCH -p defq
#SBATCH --time=10
#SBATCH --mem=1000
#SBATCH --ntasks=1
#SBATCH --job-name=fastqc
#SBATCH --error=check_seq_qual.err
#SBATCH --output=check_seq_qual.log

#TO DO
#set job-name to be <yourInitials>_fastqc
#change the error and output paths so that SLURM puts your error and log files into the directory you created in the /gpfs/share/training
#change the path following the -o arguments of both fastqc calls to output to your ATAC-seq directories by updating the outputDir variable to t
he directory you created in /gpfs/share/training

read1='/gpfs/share/training/drosophila_fastq/SRR4044399_PE_chr2L_subsample_read1.fastq'
outputDir=''

#command call to fastqc sotware
time /gpfs/share/training/software/FastQC/fastqc $read1 -o $outputDir
```

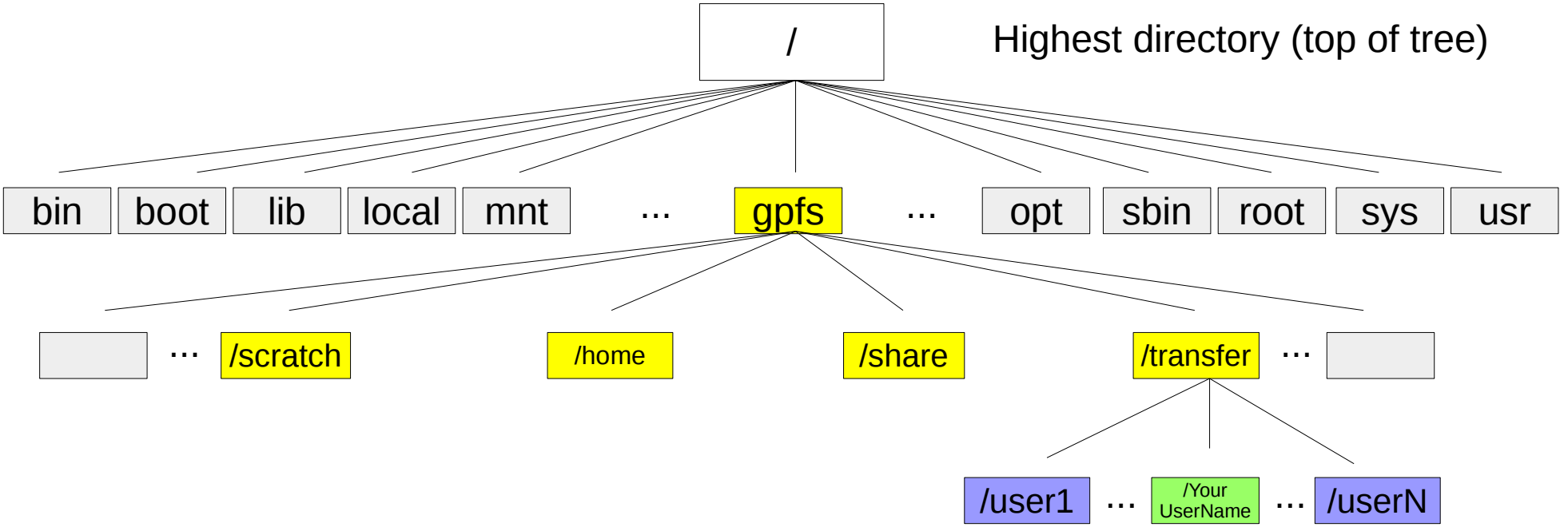
- Make the changes listed in the #TO DO section
- Save the script and submit it to SLURM
    - What is the status of your job?
    - When it finishes, check the error and log files. What do they contain?

```
Option outdir requires an argument
Started analysis of SRR4044399_PE_chr2L_subsample_read1.fastq
Approx 5% complete for SRR4044399_PE_chr2L_subsample_read1.fastq
Approx 10% complete for SRR4044399_PE_chr2L_subsample_read1.fastq
Approx 15% complete for SRR4044399_PE_chr2L_subsample_read1.fastq
Approx 20% complete for SRR4044399_PE_chr2L_subsample_read1.fastq
Approx 25% complete for SRR4044399_PE_chr2L_subsample_read1.fastq
Approx 30% complete for SRR4044399_PE_chr2L_subsample_read1.fastq
Approx 35% complete for SRR4044399_PE_chr2L_subsample_read1.fastq
Approx 40% complete for SRR4044399_PE_chr2L_subsample_read1.fastq
Approx 45% complete for SRR4044399_PE_chr2L_subsample_read1.fastq
Approx 50% complete for SRR4044399_PE_chr2L_subsample_read1.fastq
Approx 55% complete for SRR4044399_PE_chr2L_subsample_read1.fastq
Approx 60% complete for SRR4044399_PE_chr2L_subsample_read1.fastq
Approx 65% complete for SRR4044399_PE_chr2L_subsample_read1.fastq
Approx 70% complete for SRR4044399_PE_chr2L_subsample_read1.fastq
Approx 75% complete for SRR4044399_PE_chr2L_subsample_read1.fastq
Approx 80% complete for SRR4044399_PE_chr2L_subsample_read1.fastq
Approx 85% complete for SRR4044399_PE_chr2L_subsample_read1.fastq
Approx 90% complete for SRR4044399_PE_chr2L_subsample_read1.fastq
Approx 95% complete for SRR4044399_PE_chr2L_subsample_read1.fastq

real    0m4.342s
user    0m6.566s
sys     0m0.186s
check_seq_qual.err (END)
```

```
Analysis complete for SRR4044399_PE_chr2L_subsample_read1.fastq
check_seq_qual.log (END)
```

- Where are the expected results from fastqc located?
- How do you view these results?



Rosalind and Wilkins have a separate location so that files can be uploaded to Rosalind/Wilkins and download from Rosalind/Wilkins using **SFTP**

What should be stored in /gpfs/transfer/your_username?

- Files that need to be transferred into/out of Rosalind onto a HIPAA-compliant device

= sudo required

= limited access

= full access*

= no access w/o permission

Steps for getting data out of Wilkins

1. You can **`cp`** files into the staging area

```
[brunettt@cubipmtest02 training] cp myResults /gpfs/transfer/myUsername
```

2. Go to your local computer and open an sftp client
3. Put in staging area address sftp

```
[brunettt@local.host Downloads] sftp brunettt@cubipmtest02.ucdenver.pvt
```

4. **`get`** /gpfs/tranfer/myUsername/myFile.txt

```
sftp> get /gpfs/transfer/brunettt/myFile.txt
```

5. Exit **`sftp`**

```
sftp> quit
```

6. Now open on your local machine!

We hope you enjoyed this mini-workshop and we hope it will entice you to join the Rosalind community!

Please feel free to contact our team if you have any questions!

CCPM-Rosalind@ucdenver.edu

www.ucdenver.edu/Rosalind