Interactive Demo:

Introduction to High Performance Computing (HPC)

Michelle Franc Ragsac
3rd Year BISB | PI: Emma Farley, PhD
mragsac@eng.ucsd.edu

What's the point of this module?

1. **Briefly learn about HPC systems and their utility in bioinformatics**
   We'll cover the basics of what "High-Performance Computing" is and why having it is useful for big data and bioinformatics research

2. **Get some brief hands-on experience with HPC systems using TSCC**
   Using training accounts, we'll log into the Triton Shared Compute Cluster (TSCC) and learn about the basics about submitting compute jobs

3. **Learn about alternatives to TSCC that people on campus use**
   At the end, we'll briefly cover some alternatives to TSCC that labs on campus use, such as XSEDE's COMET and Amazon AWS

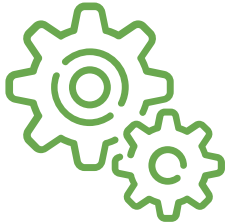
How is this module organized?

BACKGROUND

We'll start off by going through some background information on HPC and TSCC

DEMO

After learning a little bit about what HPC systems are, we'll get you logged into TSCC for a demo



FINAL TIDBITS

Finally, we'll finish off by going over some alternatives to TSCC, such as XSEDE COMET and Amazon AWS

Section I:
Background

What is HPC?
What is TSCC?

Motivations for Using a Compute Cluster

- Computational research problems can often outgrow the capabilities of one's local system
- To solve this, access to more computers is needed!

- **Servers** contain much ***more storage, memory, and compute capacity*** than a local device
- **Clusters** are multiple servers that are linked together

https://www.seattletimes.com/explore/at-home/drowning-in-paperwork-how-to-get-it-organized/

What are terms I should know?

These terms are used a lot in different contexts, so it's important to know what they each mean and how they're related to each other!

cloud	generic term for computing resources that are provisioned to users on demand or as needed
HPC system	"High-Performance Computing" systems are resources for computationally intensive operations that are comprised of integrated processing and storage elements
cluster	small- to moderate-scale HPC resources (e.g., TSCC)

What is the Triton Shared Compute Cluster (TSCC)?

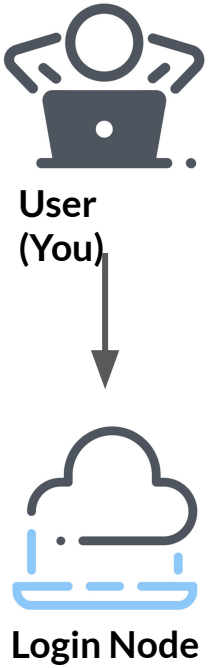
https://www.elementaconsulting.com/projects/uc-san-diego-supercomputer-center/

- Research cluster housed on-campus at the **San Diego Supercomputer Center (SDSC)**
  - North Campus; Next door to RIMAC Gym
- **"Condo Cluster" → Built by researchers**
  - Each **node** that a lab purchases for TSCC contributes to the general compute power of the cluster for researchers to use
- **"Hotel Services" → Pay-As-You-Go HPC**

Additional Resources:

🔗 TSCC User Guide: https://www.sdsc.edu/support/user_guides/tscc.html

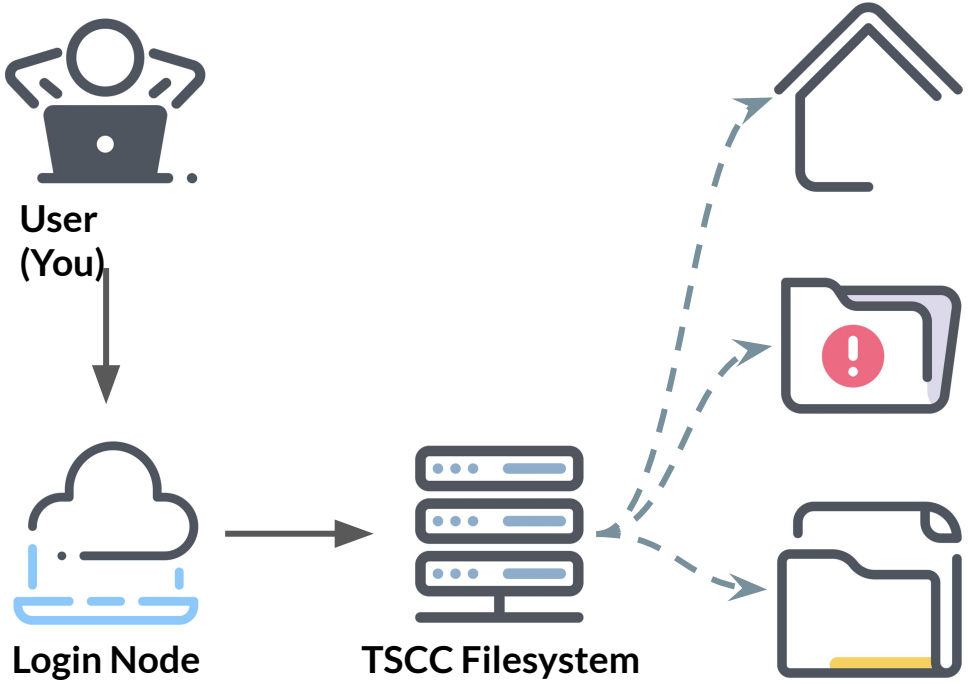
How do I access files on TSCC?



There are 4 **login nodes** on TSCC:

```
ssh $USER_NAME@tscc-login1.sdsc.edu
ssh $USER_NAME@tscc-login2.sdsc.edu
ssh $USER_NAME@tscc-login11.sdsc.edu
ssh $USER_NAME@tscc-login12.sdsc.edu
```

How do I access files on TSCC?

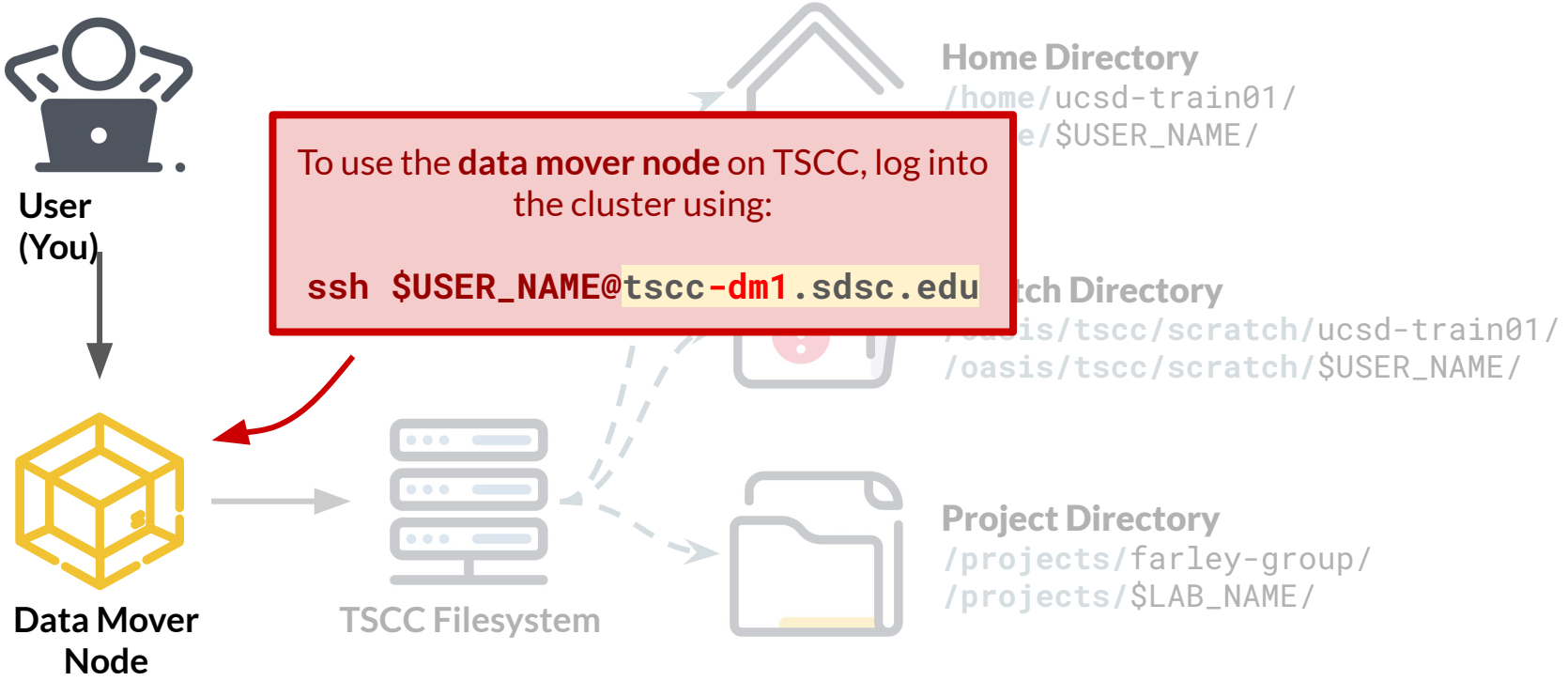


Home Directory
`/home/ucsd-train01/`
`/home/$USER_NAME/`

Scratch Directory
`/oasis/tscc/scratch/ucsd-train01/`
`/oasis/tscc/scratch/$USER_NAME/`

Project Directory
`/projects/farley-group/`
`/projects/$LAB_NAME/`

How do I move files on and off TSCC?



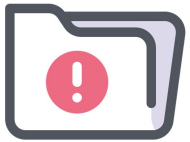
To use the **data mover node** on TSCC, log into the cluster using:

```
ssh $USER_NAME@tscc-dm1.sdsc.edu
```

TSCC Filesystem: Home vs. Scratch vs. Project

Home Directory
`/home/ucsd-train01/`
`/home/$USER_NAME/`

- Each user has their own permanent home folder
- Very minimal space (**100 GB**)



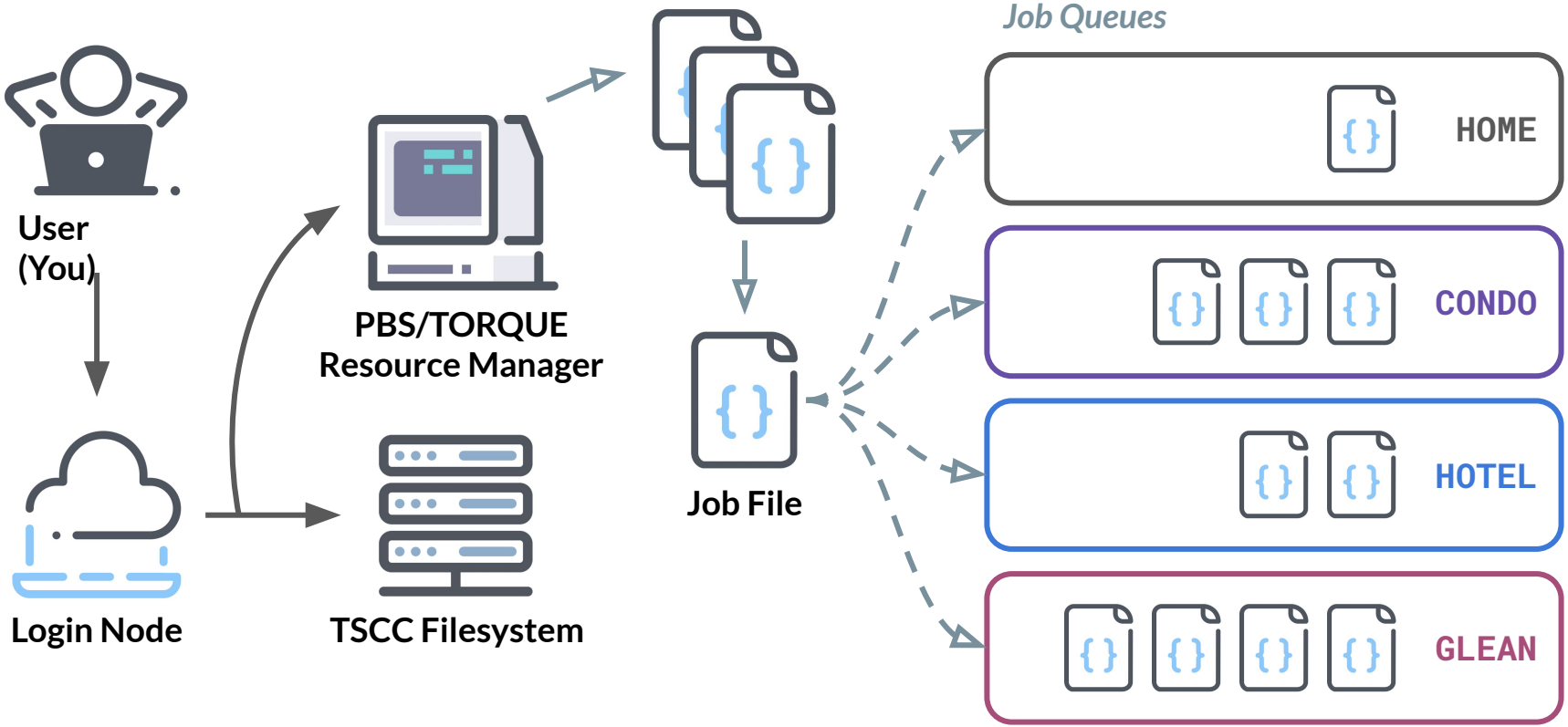
Scratch Directory
`/oasis/tscc/scratch/ucsd-train01/`
`/oasis/tscc/scratch/$USER_NAME/`

- Each user has their own temporary folder
- Lots of space (**25 TB**), *but* untouched files get purged **every 3 months**

Project Directory
`/projects/farley-group/`
`/projects/$LAB_NAME/`

- Long-term archival storage space purchased by labs (e.g., **`ps-farleylab`**)
- Shared amongst members of a lab

How do I use TSCC for computing?



TSCC Job Queues: Home vs. Condo vs. Hotel vs. Glean

HOME

- Purchased nodes reserved for members of a particular user group (e.g., members of the Farley Lab)
- **Unlimited time limit**

CONDO

- Allows contributors to TSCC to run their compute jobs on nodes greater than those they have purchased
- **8-hour time limit**

HOTEL

- Supports all non-contributors to TSCC
- **168-hour time limit**

***This is the only node we're allowed to use our training accounts for*

GLEAN

- Allows users to run jobs free-of-charge on available idle nodes within the **CONDO** queue
- **1-hour time limit**

Section II:
Interactive Demo

How do I log into TSCC?
How do I navigate TSCC?
How do I access software on TSCC?
How do I submit jobs on TSCC?

Before we get started...

Have you sent your Public SSH Key to Michelle to add to your training account?

If you don't know how to do this, please refer to the Module 1a: Bench to Terminal slides!

DEMO

What are the goals of this interactive module?

By the end of this module, we hope that you'll become **familiar** with:

- ▢ Logging onto TSCC using the `ssh` command
- ▢ Interacting with TSCC using UNIX commands
- ▢ Transferring files to TSCC with the UNIX command `wget`
- ▢ Running UNIX, Python, and R scripts on the Command Line
- ▢ Running Interactive and Non-Interactive Compute Jobs with the `qsub` command
- ▢ Launching Jupyter Notebooks with a Python or R kernel

DEMO

Logging onto TSCC with the ssh Command

To start off, we'll be using our **local computers** to access the **remote system**, TSCC!

1. Open your favorite Terminal application
2. Open the account spreadsheet and figure out what your assigned username is
   a. **BISB Bootcamp 2020 TSCC Training Account Assignments:**
      https://docs.google.com/spreadsheets/d/1GRNTD9zDvqXrUI63F2KNjONN9Rgd_Inu3wWOm79HNWU/edit#gid=0
   b. From here on out, I'll be generalizing all of the instructions!
      Wherever you see **$USER_NAME**, replace it with *your* assigned TSCC training account
3. Log into TSCC with the following command:

```
ssh $USER_NAME@tscc-login.sdsc.edu
```

The `tscc-login.sdsc.edu` hostname will automatically redirect you to a login node in order to balance the number of users on a login node at a single time

DEMO

Navigating TSCC: Command-Line Interfaces

- **Commands** are directives given to a computer to perform specific tasks
- TSCC lacks a Graphical User Interface (GUI) so we will instead navigate it using a **command-line interface** with UNIX commands!

In this next portion, we'll go over some basic UNIX commands!

***We also covered these commands in the Module 1a: Bench to Terminal presentation*

Additional Resources:

- *Introduction to the Command-Line Interface*
  https://www.codecademy.com/learn/learn-the-command-line
  https://carlalexander.ca/introduction-command-line-interface/
- *UNIX Command Line Cheat Sheets*
  https://files.fosswire.com/2007/08/fwunixref.pdf
  http://cheatsheetworld.com/programming/unix-linux-cheat-sheet/

DEMO

(basic)

Navigating TSCC: Some UNIX Commands, Pt. 1

`pwd` "Print Working Directory" → Show your current location

`ls` "LiSt" → Show the contents in your current location

`cd $LOCATION_OF_INTEREST` "Change Directory" → Change current location

`$HOME`	home directory environment variable	(e.g., `cd $HOME`)
`~`	home directory shortcut	(e.g., `cd ~`)
`..`	move back one directory in the hierarchy	(e.g., `cd ..` or `cd ../other_folder/`)
`-`	previous directory you were located in	(e.g., `cd -`)

`man $COMMAND_NAME` "MANual" → Show a detailed manual of the command of interest

ExplainShell (https://www.explainshell.com/) is a helpful website that matches command-line argument flags to what they mean!

`mkdir $DIRECTORY_NAME`

"MaKe DIRectory" →
Create a folder/directory with the provided name

DEMO

(basic)

Navigating TSCC: Some UNIX Commands, Pt. 2

Viewing the Contents of a File

`head $FILE_NAME` Preview the **beginning** of a file's contents

`tail $FILE_NAME` Preview the **ending** of a file's contents

`cat $FILE_NAME` "Con**CAT**enate" → Preview **all** of a file's contents

---

(*Beginning of*) Manipulating Files

`cat $FILE_1 $FILE_2` Con**CAT**enate **$FILE_1** and **$FILE_2** then output result **to the command line**

You can save the output into a file with the **>** *symbol* (e.g., **`cat $FILE_1 $FILE_2 > $FILE_3`**)

`cp $FILE_1 $FILE_2` "**C**o**P**y" → Copy **$FILE_1** to a new file called **$FILE_2**

`mv $FILE_1 $FILE_2` "**M**o**V**e" → Rename **$FILE_1** to have the name **$FILE_2**

You can move the file to a new location (e.g., **`mv $FILE_1 /new/location/`**)

`rm $FILE_NAME` "**R**e**M**ove" → Delete **$FILE_NAME**

CAUTION: Be careful with this command! Once a file is removed, it can rarely be recovered!

DEMO

Navigating TSCC: Generating Shortcuts

When you login to TSCC, you usually start off at your **$HOME** directory!

We'll start off by generating a "soft-link" or shortcut to your scratch directory before we get into more UNIX commands! → **ln -s**

User (You)

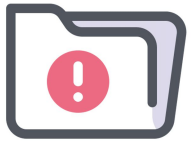
```
ssh $USER_NAME@tscc-login.sdsc.edu
```

Login Node

TSCC Filesystem

Home Directory
/home/$USER_NAME/

```
ln -s /oasis/tscc/scratch/$USER_NAME scratch
```



Scratch Directory
/oasis/tscc/scratch/$USER_NAME/

DEMO

CHECKPOINT! on Module Learning Goals

By the end of this module, we hope that you'll become **familiar** with:

- ✓ ~~Logging onto TSCC using the `ssh` command~~
- ✓ ~~Interacting with TSCC using UNIX commands~~
- ▢ Transferring files to TSCC with the UNIX command `wget`
- ▢ Running UNIX, Python, and R scripts on the Command Line
- ▢ Running Interactive and Non-Interactive Compute Jobs with the `qsub` command
- ▢ Launching Jupyter Notebooks with a Python or R kernel

DEMO

Transferring files to TSCC with `wget`, Pt. 1

If you're transferring large files to TSCC, it's **highly recommended** to use the data mover node located at **`tscc-dm1.sdsc.edu`** instead of the login node!

Proper Cluster Etiquette: DO NOT use the login node for computationally intensive operations! It slows down the cluster for everybody and people *will* get mad ;-(

In this portion of the module, we'll be using **`wget`** *to download some things we need for later!*

Generally, ***`wget`*** *works with the syntax:* `wget $URL_OF_INTEREST`

Transferring files to TSCC with `wget`, Pt. 2

For this module, let's start off by downloading some simple demo scripts in Python and R from the GitHub website:

1. Go to the GitHub site for today's module:
   https://github.com/mragsac/BISB-Bootcamp-2020/tree/master/day3/module4_hpc-crash-course

2. Download both of the files in the **`demo-scripts/`** folder using **`wget`** to your TSCC home directory (**`/home/$USER_NAME/`**)
   a. First, select the file you wish to download.
      This should open a new window with the contents of the file!
   b. Next, right click the "Download" button that appears, then copy the link address
   c. Finally, download the file to your TSCC account using the command:

```
wget http://github.com/link/to/file
```

DEMO

Environment Modules on TSCC

- TSCC has many pre-installed software packages that are stored under **modules** for users to access for their analyses *(including bioinformatics software!)*
  - **TSCC Environment Modules Section in the User Guide:** https://www.sdsc.edu/support/user_guides/tscc.html#env-modules

Command	Description
`module list`	List the modules that are currently loaded
`module avail`	List the modules that are available
`module load $MODULE_NAME`	Load **`$MODULE_NAME`** into the current environment
`module unload $MODULE_NAME`	Unload **`$MODULE_NAME`** from the current environment

Using TSCC's Pre-Installed Modules to Run Python and R Scripts

We can use some of TSCC's pre-installed modules to run the Python and R scripts that we downloaded in the previous section on `wget`!

1. Load the Python module that's available on TSCC with the command:
   ```
   module load python
   ```
2. Run the Python script with the command:
   ```
   python Python-Demo-Script.py
   ```
3. Load the R module that's available on TSCC with the command:
   ```
   module load R
   ```
4. Run the R script with the command:
   ```
   Rscript R-Demo-Script.R
   ```

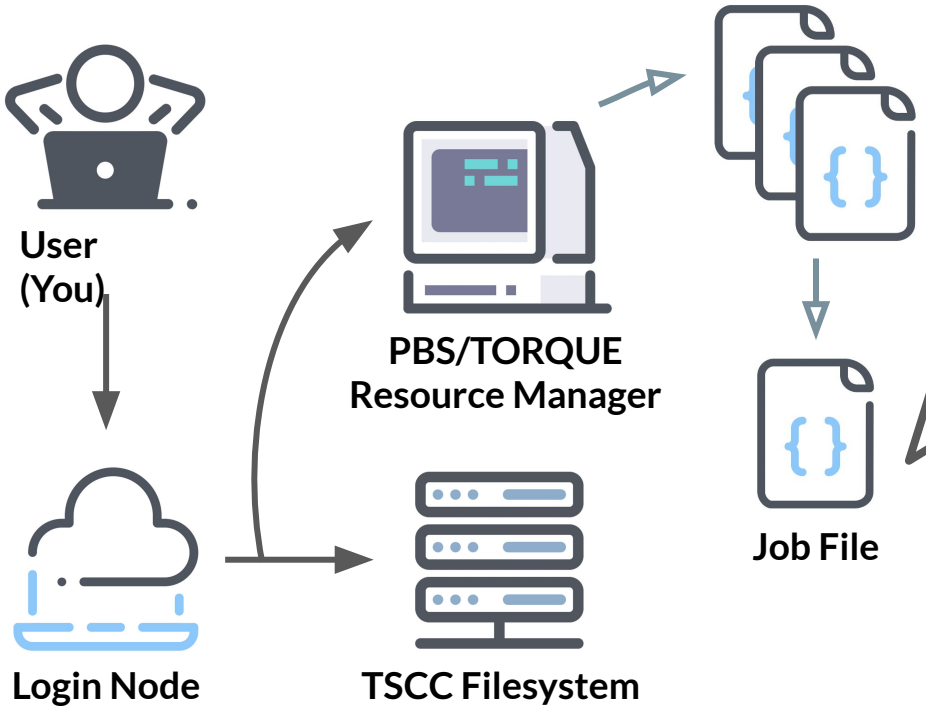
DEMO

CHECKPOINT! on Module Learning Goals

By the end of this module, we hope that you'll become **familiar** with:

- ✓ ~~Logging onto TSCC using the `ssh` command~~
- ✓ ~~Interacting with TSCC using UNIX commands~~
- ✓ ~~Transferring files to TSCC with the UNIX command `wget`~~
- ✓ ~~Running UNIX, Python, and R scripts on the Command Line~~
- ▢ Running Interactive and Non-Interactive Compute Jobs with the `qsub` command
- ▢ Launching Jupyter Notebooks with a Python or R kernel

What do cluster "Job Files" contain?



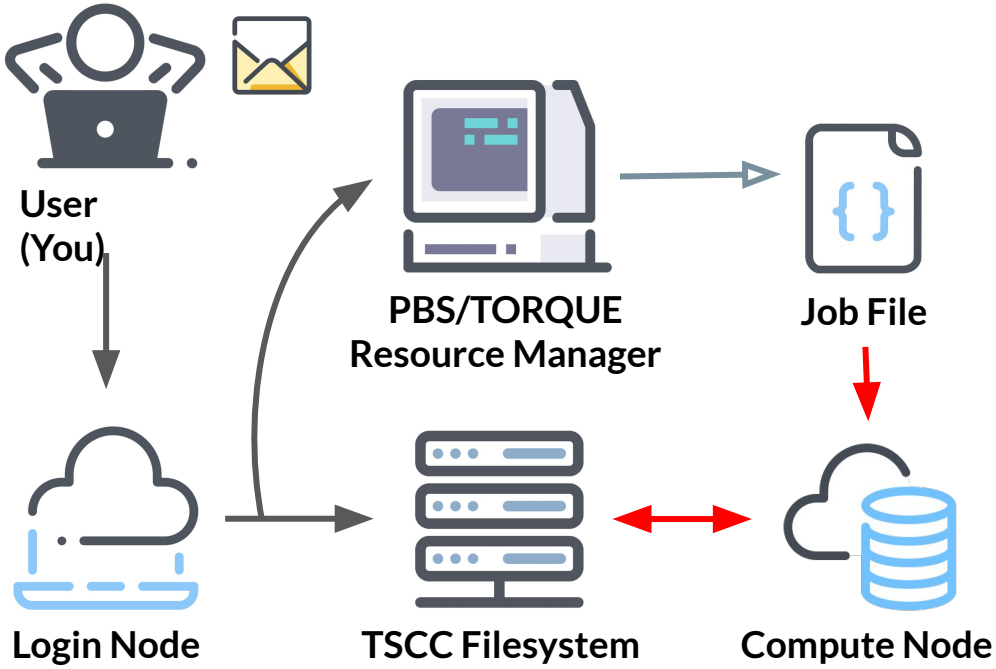
Contents of a Typical Job File:

- Name of the Job
- Queue for the Job
- Amount of Walltime Required
- Requested Amount of Computational Resources
- Commands for the Job

- Name of the Job's Log/Error File
- *Who is paying for the Job?*
- *Where should TSCC send status updates about the Job?*

DEMO

Submitting a Non-Interactive Compute Job

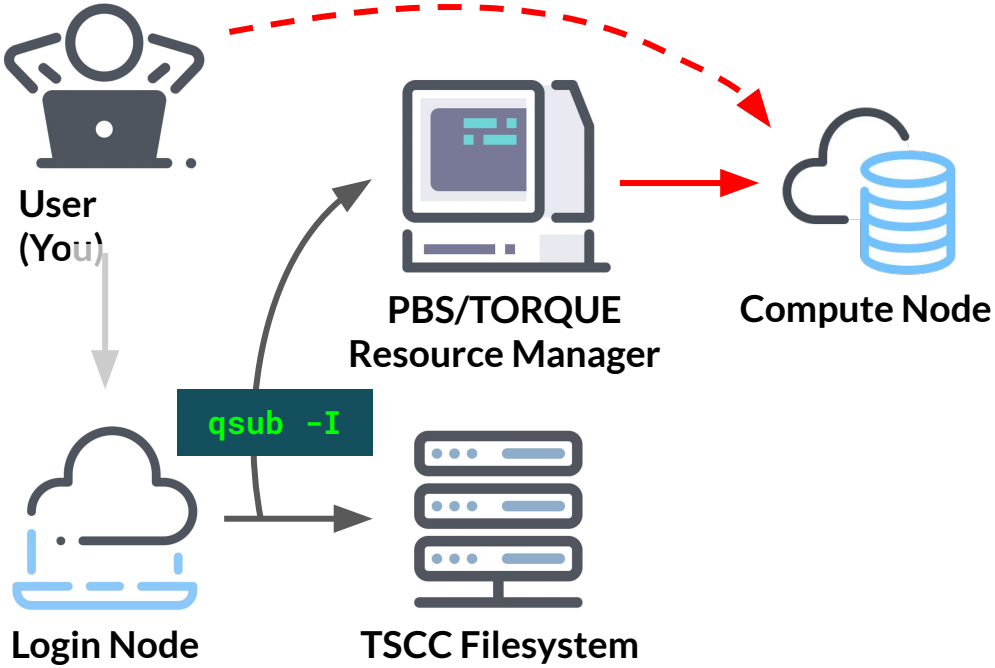


Non-Interactive Compute Jobs

1. **Submit a Job File with required parameters** for the computational task
2. After the job gets off the queue, the job is allocated a node with the required resources and **runs the instructions in the Job File**
3. When the job is completed, the user is notified via Email *(if it was set)*

DEMO

Submitting an Interactive Compute Job



Interactive Compute Jobs

1. **Submit a command on the CLI with the parameters** for the job
2. After the job gets off the queue, the job is allocated a node with the required resources and **the user is moved from the login node to the assigned node to run commands**
3. When the job is completed, the user can exit the node

It can take a while to do things with **conda** on TSCC, so we pre-installed things for you!

:-)

These hidden slides contain:

1. How we installed **conda** on your accounts
2. How we installed packages using **conda** on your accounts

Installing the conda Package Manager

We downloaded the **conda** package manager (through the smaller-scale **miniconda** install) to your account with the command:

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
```

****If you don't know what a package manager is, feel free to reference the* Module 1a: Bench to Terminal slides !
****If you want to learn more about miniconda, you can visit the documentation here: https://docs.conda.io/en/latest/miniconda.html*

After downloading the file, we installed **conda** on your account with the command:

```
bash Miniconda3-latest-Linux-x86_64.sh
```

****The installation file is just a* **bash** *script (denoted by the file ending of* **.sh***), so we can run it with the* **bash** *command*

DEMO

Creating & Changing `conda` Environments

- Environments are a handy way of compartmentalizing what exact tools (i.e., version information) you were using for an analysis when you did it
  - Generally, I have a separate environment for each type of analysis that I'm doing so that it's easy to debug any dependency issues I may run into later, or debug issues with beta software

We created a new environment for this module called **`module4_python`** with:

```
$ conda create --name module4_python
```

After creating the environment, we can then change into it from our **`base`** environment (the default, home environment) using the command:

```
$ conda activate module4_python
```

DEMO

Installing Python Packages with conda

After activating the environment **module4_python**, we installed several packages:

```
(module4_python) $ conda install -c anaconda numpy pandas matplotlib scikit-learn
```

- NumPy adds support for large, multi-dimensional arrays and optimized linear algebra in Python
- pandas adds support for Excel-like table operations in Python (i.e., R Data Frames)
- Matplotlib is a Python plotting library
- scikit-learn is a Python library used for machine learning

```
(module4_python) $ conda install -c conda-forge jupyterlab
```

- JupyterLab is a web-based user interface for Jupyter Notebooks

DEMO

Basics on Configuring conda Environments

How do I create a conda environment?

```
conda create --name $ENVIRONMENT_NAME_GOES_HERE
```

How do I change into the new environment I've created?

```
conda activate $ENVIRONMENT_NAME_GOES_HERE
```

And how do I get out of the environment?

```
conda deactivate
```

How do I see what packages are in my current environment?

```
conda list
```

How do I install packages with conda into my environment?

```
conda install $PACKAGE_NAME_GOES_HERE
```

How do I delete an environment?

```
conda env remove --name $ENVIRONMENT_NAME_GOES_HERE
```

DEMO

Installing Bioinformatics Tools with conda

- **Bioconda** is a channel for **conda** that specializes in bioinformatics software!
  - **Bioconda Home Page:** https://anaconda.org/bioconda/
- We can configure **conda** to access this channel with the commands:

```
conda config --add channels defaults
conda config --add channels bioconda
conda config --add channels conda-forge
```

- Now we can install bioinformatics software using **conda** using the command:
  `conda install $NAME_OF_BIOINFORMATICS_TOOL` (e.g., `conda install bwa`)

DEMO

Installing R Tools with conda

- In addition to installing Python packages or bioinformatics software through **conda**, it's also possible to download R packages through **conda**!

We created a separate conda environment for R development with the commands:

```
$ conda create --name module4_R
$ conda activate module4_R
(module4_R) $ conda install -c conda-forge jupyterlab
(module4_R) $ conda install r-essentials r-base
```

****Generally, R packages on* **conda** *will have the prefix "r-" before the package name*

****If there isn't a certain R package you're interested available on conda, you can also install it through the R installation command* `install.package("$PACKAGE_NAME")` *to your environment, and it will stay within your R environment*

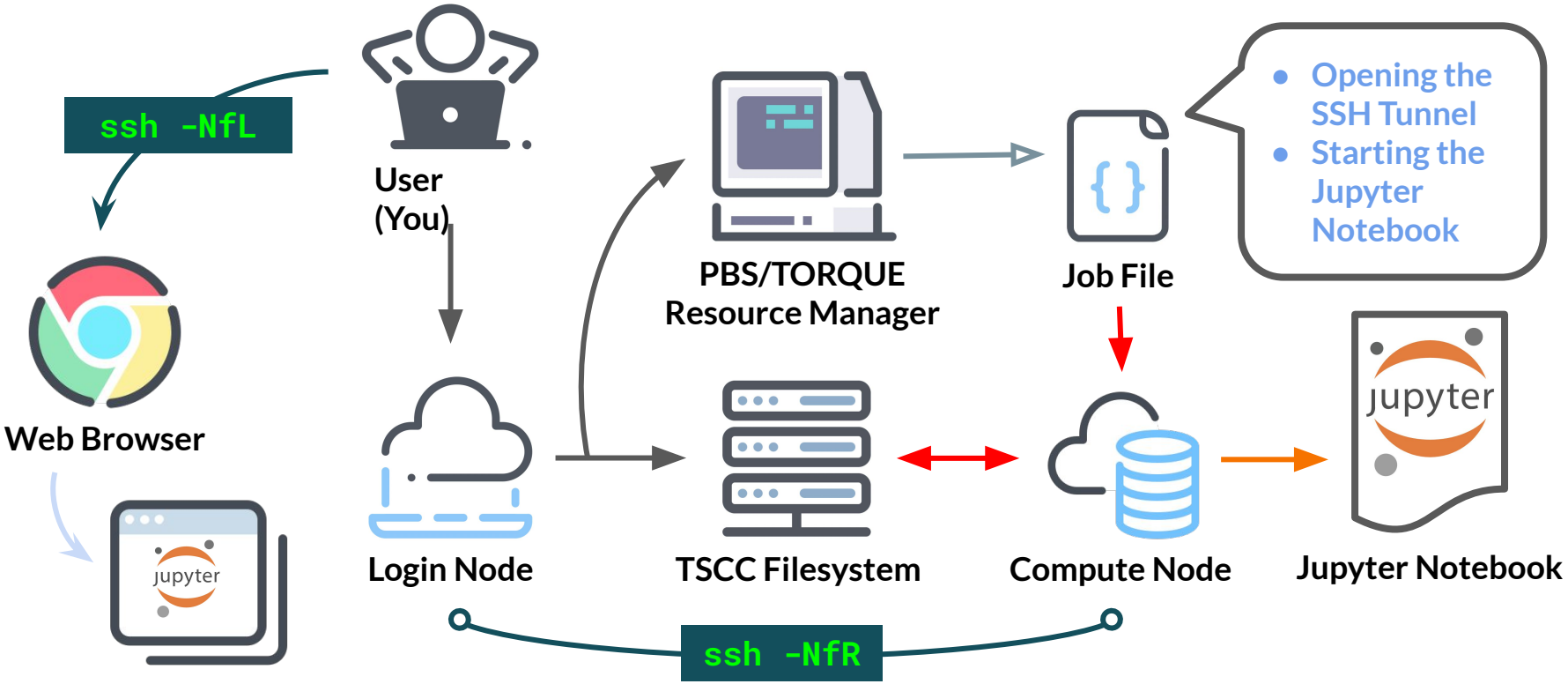
End of **conda** information section

What are Jupyter Notebooks?

- **Jupyter Notebooks** allow you to create and share documents that contain live code, equations, visualizations, and accompanying text in Markdown
  - **Project Jupyter Web Page:** https://jupyter.org/

- **JupyterLab** is a flexible web-based interactive development environment for Jupyter Notebooks, code, and data
  - **JupyterLab Documentation Page:** https://jupyterlab.readthedocs.io/en/stable/

Jupyter Notebooks on TSCC: Flowchart



DEMO

Starting a Jupyter Notebook on TSCC

Let's get some experience with submitting a non-interactive job to initialize a remote Jupyter Notebook session on TSCC that you'll be able to access on your local computer!

1. If your interactive job hasn't finished yet, let's return to the login node with:
   ```
   exit
   ```
2. Copy the Jupyter Notebook job script from the instructor account with:
   ```
   cp /home/ucsd-train111/src/TSCC_jupyter.sh /home/$USER_NAME/
   ```
3. Modify the contents Jupyter Notebook job file with the command:
   ```
   nano TSCC_jupyter.sh
   ```
   a. Modify the email that associated with job status updates so it's not mragsac@eng.ucsd.edu !!
   b. Modify the **$JUPYTER_PORT** variable to any number ***larger*** than 1048
4. Submit the Jupyter Notebook job to the job scheduler with the command:
   ```
   qsub TSCC_jupyter.sh
   ```

DEMO

What's in the TSCC_jupyter.sh job file?

```
##############################################
# !!!!! JOB COMMANDS BELOW THIS HEADER !!!!! #
##############################################

# Prints some diagnostic information for debugging later if necessary
echo ""
echo "Current Date: " date
echo "Current Working Directory: " pwd
echo ""

# Change number to be larger than 1024
JUPYTER_PORT=8448

echo "This job starts a remote Jupyter Notebook instance on TSCC!"
echo "Assigned Jupyter Port: " $JUPYTER_PORT
echo ""

# !!! If this script doesn't work, one of the reasons could be that
#     your desired port is already used. In that case, change the number!

##############################################

# Define environments to use with this notebook here;
# Unfortuantely the conda activate convention does not work here,
# so we have to use a different syntax to activate the environment
# (e.g. source activate [ENVIRONMENT_NAME])

source activate module4_python

##############################################

# Set up the SSH tunnel betweeing the COMPUTING and LOGIN node
# and then launch the notebook without a browser to travel through the tunnel

ssh -N -f -R $JUPYTER_PORT:localhost:$JUPYTER_PORT $PBS_O_HOST
jupyter lab --port=$JUPYTER_PORT --no-browser

# Uncomment the line below if you would like to use jupyter notebooks instead of the jupyter l
# jupyter notebook --port=$JUPYTER_PORT --no-browser
```

Starts off with printing some diagnostic information to include in the job output file...

Designates the $JUPYTER_PORT variable to use when tunneling the notebook, and prints that information to the job output file...

Activates the module4_python conda environment since it contains the jupyterlab package

Initializes the ssh tunnel to our $PBS_O_HOST using our $JUPYTER_PORT

Starts the Jupyter Lab session *without a browser* using our $JUPYTER_PORT

DEMO

(a brief introduction)

TORQUE/PBS Job Management & Monitoring

`qsub $JOB_FILE_NAME` "**Q**ueue **SUB**mit" → Submits the job file, **$JOB_FILE_NAME**, to the scheduler

`qsub -I` Submits an interactive (**-I**) job to the scheduler

`qstat` "**Q**ueue **STAT**us" → Displays the status of all jobs in the system

`qstat $JOB_ID` Displays the status of the job with ID **$JOB_ID**

`qstat -u $USER_NAME` Displays the status of all jobs associated with the user (**-u**) called **$USER_NAME**

`qpeek $JOB_ID` "**Q**ueue **PEEK**" → Previews the current output log of the running job, **$JOB_ID**

`qpeek -e $JOB_ID` Previews the current error (**-e**) log of the running job, **$JOB_ID**

`qdel $JOB_ID` "**Q**ueue **DEL**ete" → Deletes/cancels the job with ID **$JOB_ID**

DEMO

Connecting to a Remote Jupyter Notebook

Now that our Jupyter Notebook is running on one of the TSCC nodes,
we need to connect to it with our local computer…

1. Activate the port for your Jupyter Notebook on TSCC with the command:
   ```
   wget localhost:$JUPYTER_PORT
   ```
2. Copy the "Local" Jupyter Notebook script to your TSCC account with:
   ```
   cp /home/ucsd-train111/src/LOCAL_jupyter.sh /home/$USER_NAME/
   ```
3. **On your local computer,** make a copy the "Local" Jupyter Notebook from your TSCC account to your local computer with the command:
   ```
   scp $USER_NAME@tscc-dm1.sdsc.edu:/home/$USER_NAME/LOCAL_jupyter.sh .
   ```
4. With the file transferred, run it **on your local computer** with the command:
   ```
   bash LOCAL_jupyter.sh -u $USER_NAME -p $JUPYTER_PORT -s $LOGIN_NODE
   ```
5. Determine the Jupyter Notebook URL to visit with the command:
   ```
   qpeek -e $JOB_ID
   ```

What's in the LOCAL_jupyter.sh job file?

Command we submitted via the **LOCAL_jupyter.sh** script to connect the SSH tunnel:

```
bash LOCAL_jupyter.sh -p $JUPYTER_PORT -u $USER_NAME -s $LOGIN_NODE
```

The single command within the **LOCAL_jupyter.sh** script that connects the inputs:

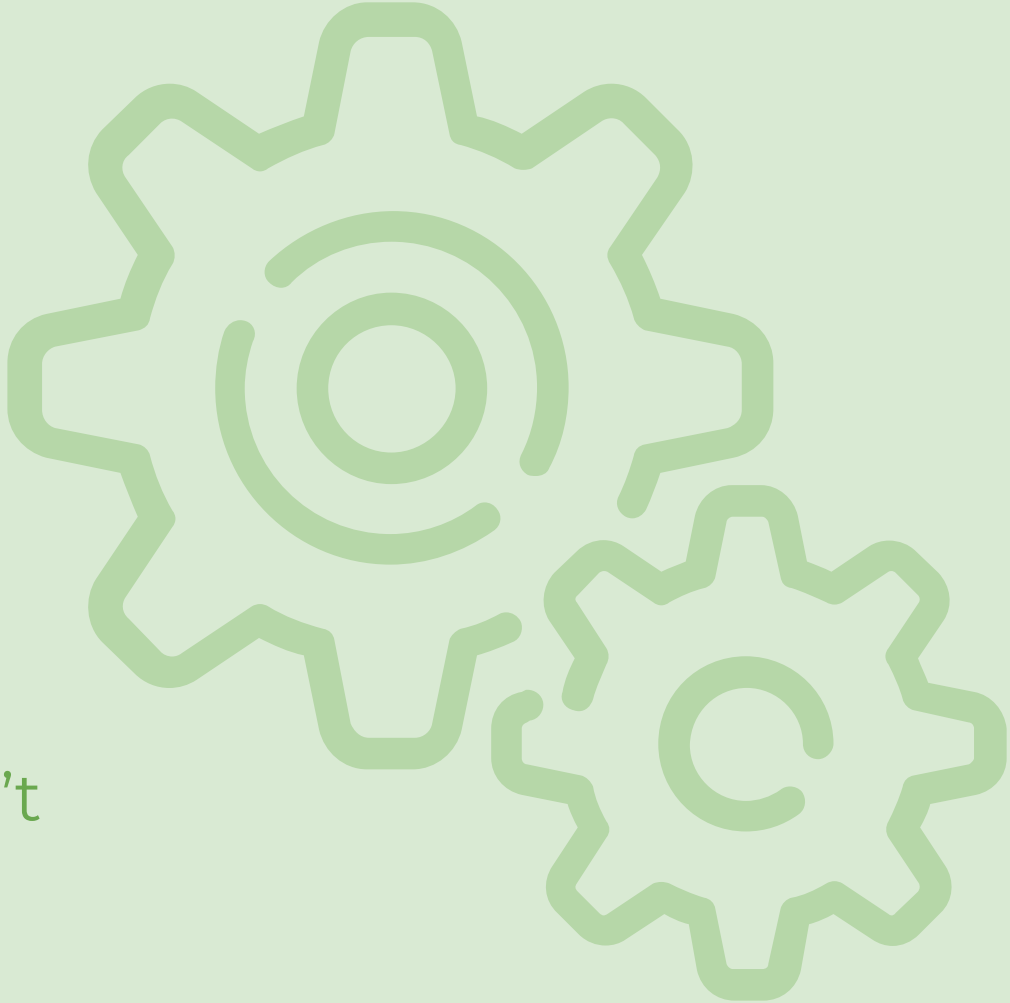
```
ssh -f -N -L $JUPYTER_PORT:localhost:$JUPYTER_PORT $USER_NAME@$LOGIN_NODE
```

establishes a connection through the **$JUPYTER_PORT** located at the **$USER_NAME@$LOGIN_NODE** address so that we can access it on our own computer

Section II:
Interactive Demo

Now, we'll be going through some Jupyter Notebooks we've prepared for you on TSCC!

Section III:
Additional Tidbits



What other resources do labs use if they don’t use TSCC?

What are some limitations to TSCC?

- To access TSCC, your PI either needs to own *physical* nodes that are a part of the cluster, or purchase service units (SUs) for you to use
  - Physical node purchases can be expensive, and they don't include project storage by default
- There are *very* limited numbers of GPU and **`pdafm`** (large memory) nodes
- TSCC can be **notoriously slow** → Especially when there are a lot of people using the login node for computationally-intensive operations!
- TSCC is a relatively small cluster so resources can be more limited compared to other options (e.g., COMET)

The XSEDE Program & The COMET Cluster

- *"The **E**xtreme **S**cience and **E**ngineering **D**iscovery **E**nvironment (**XSEDE**) is a single virtual system that scientists can use to share computing resources, data, and expertise"* — https://www.xsede.org/
  - NSF-funded resource meant to provide scientists with computational resources and education
- Researchers can request a **Resource Allocation** on any of the HPC or Data Storage systems affiliated with the XSEDE program

- **COMET** is a dedicated XSEDE cluster that is housed at SDSC!
  - Very similar to TSCC in structure, but uses **`SLURM`** instead of **`TORQUE/PBS`**
- **EXPANSE** is a new, upcoming XSEDE cluster that will also be housed at SDSC
  - NSF awarded $10M to SDSC to deploy this new supercomputer

***Frequently used in industry*

Amazon Web Services (AWS)

ssh

Pay for what you use

vCPU	MEM.	AWS
2	8GB	$0.0928
4	16GB	$0.1856
8	32GB	$0.3712

ec2 instance

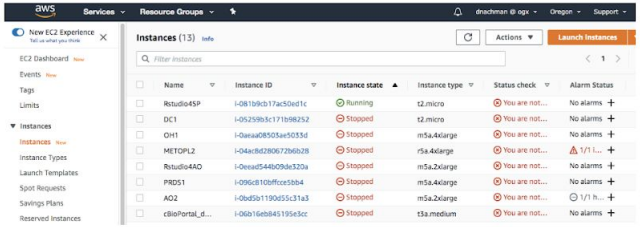
- Cloud computing
- On demand
- Flexible specs
- Pay per hour

s3 bucket

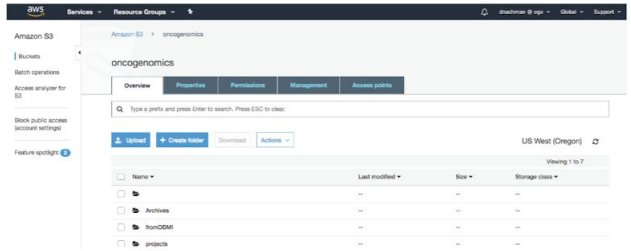
- Data storage service
- Fast accessibility

Web interfaces

ec2 management



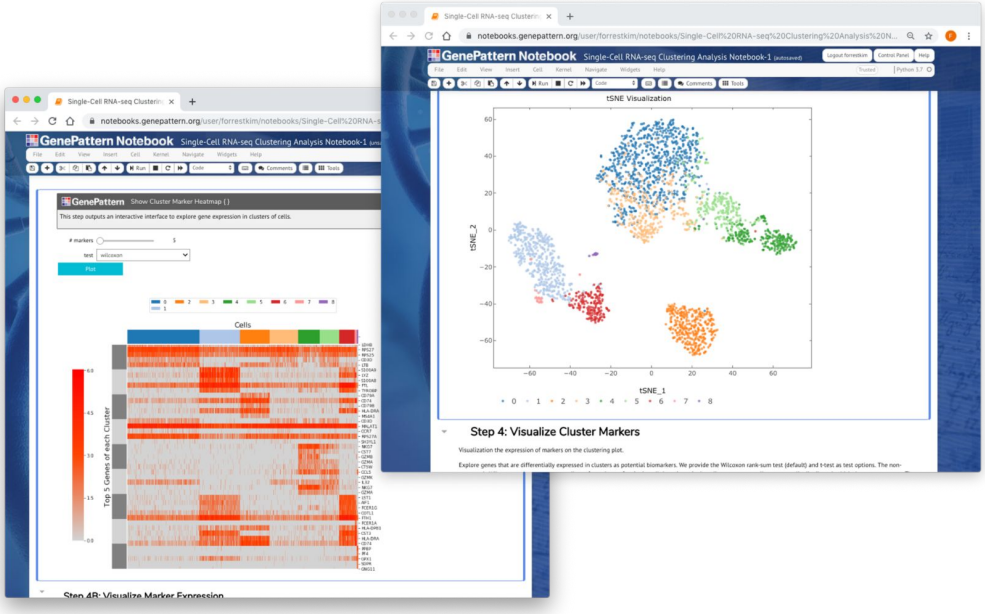
s3 data access



Thanks to Dana Nachmanson (4th Year BISB) for this slide!

GenePattern Notebook Platform

- *Developed here at UCSD in the Mesirov Lab! Go Tritons!*
- https://notebook.genepattern.org/
- Free, open-source online notebook analysis workspace
- Notebooks can be shared with collaborators or included in publications
- Common genomic and machine learning workflows are available in their "Public Notebook Library"



https://notebook.genepattern.org/#gsc.tab=0

Not intended for Protected Health Information (PHI)

Considerations for Handling Biological Data

- While there are many resources with interesting data for bioinformatics analyses, it's important to consider **if that data is sensitive/protected!**
- There may be limitations in how you can analyze or store biomedical data
  - e.g., Has personal identifying information been removed? Is the data encrypted?
    Are you working on a HIPAA-compliant server for your analyses?
    Are you following HIPAA guidelines for storing data?
- Generally, your PI or the collaborator you are working with should know or have an idea of what guidelines to follow
  - e.g., Some labs have compute clusters that operate behind a firewall to protect sensitive data

Additional Resources:

🔗 UCSD Research Affairs IT Guidelines for Handling Sensitive Data:
https://blink.ucsd.edu/research/about-us/RAIT/sensitive-data-guidelines.html

Additional Background Slides

Some notes about `conda` on TSCC...

- **`conda`** is really convenient to have on the cluster to install packages and manage different software environments
- However, running **`conda`** operations on the login node can make the login node slow → *Other users will have issues logging in :-(*

- It's recommended by the TSCC Admins that you create a **hidden file** in **`/home/$USER_NAME/`** called **`.condainit`** with the **`conda`** startup information that can be found in your **`.bashrc`**
- Then, if you ever need to run **`conda`**, you would then need to first load it into your environment using the command **`source .condainit`** before doing anything else

Alternative File Transfer Commands: `scp`

There are other ways to transfer files other than the **`wget`** *command...*
These next few slides contain a few more that are useful!

`scp` ("Secure CoPy") allows secure transferring of files between a local host and a remote host or two remote hosts

- **Example 1**: *Copy file from a remote host to local host*
  ```
  scp username@remote_host:/path/to/file.txt /local/path/to/destination/
  ```
- **Example 2**: *Copy directory from a remote host to a local host*
  ```
  scp -r username@remote_host:/path/to/directory/ /local/path/to/destination/
  ```
- **Example 3**: *Copy file from local host to a remote host*
  ```
  scp /local/path/to/file.txt username@remote_host:/path/to/destination/
  ```
- **Example 4**: *Copy file from a remote host to a remote host*
  ```
  scp username@from_remote_host:/path/to/file.txt username@to_remote_host:/path/to/destination
  ```

Alternative File Transfer Commands: `rsync`

Like **scp**, **rsync** ("Remote SYNC") is another remote and local file transfer tool; however, it employs special algorithms to make the data transfer process faster

- **Example 1**: *Copy file between two local locations*
  ```
  rsync -zvh /path/to/file.txt /new/path/to/destination/
  ```
- **Example 2**: *Copy directory from a local location to a new location*
  ```
  rsync -avzh /path/to/originalDirectory/ /new/path/to/destination/
  ```
- **Example 3**: *Copy directory from a local location to a remote host*
  ```
  rsync -avz /path/to/originalDirectory/ username@remote_host:/path/to/destination/
  ```
- **Example 4**: *Copy directory from a remote host to a local machine*
  ```
  rsync -avzh username@remote_host:/path/to/originalDirectory /new/path/to/destination/
  ```
- **Example 5**: *Copy file from a remote host to a local machine using ssh*
  ```
  rsync -avzhe ssh username@remote_host:/path/to/file.txt /new/path/to/destination/
  ```
- **Example 6**: *Copy file from local machine to a remote server using ssh*
  ```
  rsync -avzhe ssh /path/to/file.txt username@remote_host:/path/to/destination/
  ```

Alternative File Transfer Commands: `curl`

`curl` is a tool to transfer data from or to a server, similar to **`wget`**

- **Example 1**: *Obtain a file from a specific FTP server*
  ```
  curl ftp://cool.ftp.website.com/file.txt
  ```
- **Example 2**: *Get the directory listing of a FTP site*
  ```
  curl ftp://cool.ftp.website.com/
  ```
- **Example 3a:** *Obtain a file from a specific FTP server using a username and password combination*
  ```
  curl ftp://username:password@cool.ftp.website.com/file.txt
  ```
- **Example 3b:** *Obtain a file from a specific FTP server using a username and password combination*
  ```
  curl -u username:password ftp://cool.ftp.website.com/file.txt
  ```

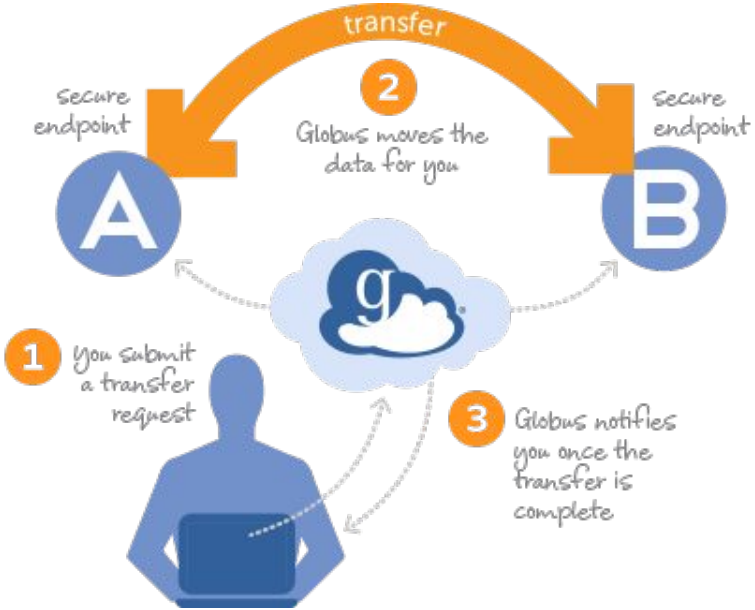
<u>Additional Resources:</u>

🔗 *Documentation for* **`curl`** https://curl.haxx.se/docs/

🔗 *Tutorials for* **`curl`** https://curl.haxx.se/docs/manual.html
https://gist.github.com/joyrexus/85bf6b02979d8a7b0308
https://gist.github.com/joyrexus/85bf6b02979d8a7b0308

File Transfer with Globus on TSCC

- **Globus** is a platform for secure research data transfer between systems and across different organizations
  - **Globus Platform Website:** https://www.globus.org/
- Users with access to SDSC XSEDE's COMET or GORDON resources have access to the Globus platform and can use it for data transfer
  - **Using Globus Online Sharing at SDSC:** https://www.sdsc.edu/education_and_training/tutorials1/globus.html



https://www.globus.org/data-transfer