# Intro To SLURM
# for existing users of
# Joint High Performance Computing Exchange (JHPCE)

JOHNS HOPKINS
BLOOMBERG SCHOOL
*of* PUBLIC HEALTH

http://www.jhpce.jhu.edu/

Version: 20230815

# Contents

- Motivation: JHPCE 3.0
- Terminology
- Logging in
- Basics of running programs on the cluster
- Details – limits and resources
- Examples

# Motivation: JHPCE 3.0

We are implementing an upgrade of the cluster

- from CentOS 7.9 (based on RHEL 7.9)
- to Rocky 9.2 (based on RHEL 9.2).

Internally we refer to these as JHPCE 2.0 and JHPCE 3.0, or in shorthand, J2 and J3.

Part of this upgrade is a switch in the choice of **job scheduler.**

- J2 uses SGE (the Sun Grid Engine).
- J3 will use “SLURM” (Simple Linux Utility for Resource Management)

# SLURM ~= SGE

SLURM and SGE perform the same functions, and so use the same or similar terms.

Some differences:

- **Queues:** SLURM calls them **partitions**.
- **Memory:** It does not use as many memory variables as SGE. No mem_free, h_vmem, h_stack, ...
- **Memory:** It does allow you to request memory in two ways:
  - --mem: memory per node (for all cores used)
  - --mem-per-cpu: memory per core (perhaps harder to accurately estimate, but similar to SGE's method of (cores requested)x(memory requested)=total memory for job)
  - qrsh -pe local 2 -l mem_free=16G,h_vmem=16G  # total memory per node: 32GB

# SLURM ~= SGE (cont'd)

More differences:

- **CPU** vs **core**: SLURM uses these terms interchangeably
- **Current working directory:** SGE uses –cwd. **`sbatch`** & **`srun`** by default run in the same directory in which you issued those commands, unless the --chdir argument is used.
- **Batch files:** SLURM requires an interpreter to be defined on the file's first line (e.g. #!/bin/bash) or the file to be marked executable.
- **Reservations:** SLURM has many features meant to make creating complex job-processing scenarios possible. Examples include creating batch jobs which then call **`srun`**. We don't yet have a lot of experience with the **`salloc`** command.

# SLURM on JHPCE Is A New Thing

We are excited about the cluster upgrade! It is a chance to install newer software versions with all of their optimizations, bug fixes and new features.

While SLURM is similar to SGE, it brings with it changes that we all need to master.

This document is aimed at current JHPCE users, who are familiar with SGE.

We can use your help building J3!

JHPCE web site documentation has not yet been updated for SLURM. When contacting the bitsupport & bithelp mailing lists, please be sure to mention that you are writing about a J3 issue.

# Transition to JHPCE 3.0

We are creating J3 by standing up some new servers which are managing several re-purposed J2 nodes. Over time we will rebuild more J2 nodes into J3 nodes and enable more queues.

## New J3 login servers

- login31 (will become replacement jhpce01) – access as jhpce03.jhsph.edu
- login42 (will become replacement jhpce02) – being built

## Initial J3 compute nodes:

- compute-094 – reserved for system development
- compute-111
- compute-151 – reserved for system development
- compute-152
- compute-153

## Initial J3 partitions (queues) (1st three share all available nodes):

- shared – the default
- interactive
- debug
- sysadmin – reserved for system development

# SGE vs SLURM

| SGE | SLURM | Notes |
|---|---|---|
| qsub | sbatch | SLURM requires an interpreter first line |
| qrsh | srun --pty --x11 bash | Interactive login |
| qhost –q | sinfo | Queue status, by queue |
| qhost | sinfo –N | Queue status, by node. |
| qstat –u * | squeue | See everyone's jobs |
| qdel | scancel | Cancel job |
| qhold | scontrol hold | Pause job |
| qrls | scontrol release | Release job |

Table-based command comparisons:

- https://hpcsupport.utsa.edu/foswiki/pub/Main/SampleSlurmSubmitScripts/SGEtoSLURMconversion.pdf
- https://docs.mpcdf.mpg.de/doc/computing/clusters/aux/migration-from-sge-to-slurm

# Primary Commands – SLURM (Simple Linux Utility for Resource Management)

- **`sbatch`** – submit a batch job to the cluster
- **`srun --pty --x11 bash`** – establish an interactive session

- **`scancel`** – cancel or pause a job
- **`squeue`** – see the status of running & pending jobs
- **`sacct`** – see the status of past (& running) jobs
- **`sinfo`** – see status of the compute nodes
- **`sstat`** – see statistics from running jobs
- **`sview`** – graphic view of cluster

slurm®
workload manager

## Job Submission

**salloc** - Obtain a job allocation.
**sbatch** - Submit a batch script for later execution.
**srun** - Obtain a job allocation (as needed) and execute an application.

| | |
|---|---|
| –array=<indexes> (e.g. "--array=1-10") | Job array specification. (sbatch command only) |
| –account=<name> | Account to be charged for resources used. |
| –begin=<time> (e.g. "--begin=18:00:00") | Initiate job after specified time. |
| –clusters=<name> | Cluster(s) to run the job. (sbatch command only) |
| –constraint=<features> | Required node features. |
| --cpu-per-task=<count> | Number of CPUs required per task. |
| –dependency=<state:jobid> | Defer job until specified jobs reach specified state. |
| –error=<filename> | File in which to store job error messages. |
| –exclude=<names> | Specific host names to exclude from job allocation. |
| –exclusive[=user] | Allocated nodes can not be shared with other jobs/users. |
| –export=<name[=value]> | Export identified environment variables. |
| –gres=<name[:count]> | Generic resources required per node. |
| –input=<name> | File from which to read job input data. |
| –job-name=<name> | Job name. |
| –label | Prepend task ID to output. (srun command only) |
| –licenses=<name[:count]> | License resources required for entire job. |
| --mem=<MB> | Memory required per node. |
| --mem-per-cpu=<MB> | Memory required per allocated CPU. |
| -N<minnodes[-maxnodes]> | Node count required for the job. |
| -n<count> | Number of tasks to be launched. |
| --nodelist=<names> | Specific host names to include in job allocation. |
| --output=<name> | File in which to store job output. |
| --partition=<names> | Partition/queue in which to run the job. |
| --qos=<name> | Quality Of Service. |
| --signal=[B:]<num>[@time] | Signal job when approaching time limit. |
| --time=<time> | Wall clock time limit. |
| --wrap=<command_string> | Wrap specified command in a simple "sh" shell. (sbatch command only) |

## Accounting

**sacct** - Display accounting data.

| | |
|---|---|
| --allusers | Displays all users jobs. |
| --accounts=<name> | Displays jobs with specified accounts. |
| --endtime=<time> | End of reporting period. |
| --format=<spec> | Format output. |
| --name=<jobname> | Display jobs that have any of these name(s). |
| --partition=<names> | Comma separated list of partitions to select jobs and job steps from. |
| --state=<state_list> | Display jobs with specified states. |
| --starttime=<time> | Start of reporting period. |

SchedMD
Slurm Support and Development

**sacctmgr** - View and modify account information.
Options:

| | |
|---|---|
| --immediate | Commit changes immediately. |
| --parseable | Output delimited by '\|' |

Commands:

| | |
|---|---|
| add <ENTITY> <SPECS> create <ENTITY> <SPECS> | Add an entity. Identical to the **create** command. |
| delete <*ENTITY*> where <*SPECS*> | Delete the specified entities. |
| list <ENTITY> [<SPECS>] | Display information about the specific entity. |
| modify <*ENTITY*> where <*SPECS*> set <*SPECS*> | Modify an entity. |

Entities:

| | |
|---|---|
| account | Account associated with job. |
| cluster | *ClusterName* parameter in the *slurm.conf*. |
| qos | Quality of Service. |
| user | User name in system. |

## Job Management

**sbcast** - Transfer file to a job's compute nodes.

*sbcast [options] SOURCE DESTINATION*

| | |
|---|---|
| --force | Replace previously existing file. |
| --preserve | Preserve modification times, access times, and access permissions. |

**scancel** - Signal jobs, job arrays, and/or job steps.

| | |
|---|---|
| --account=<name> | Operate only on jobs charging the specified account. |
| --name=<name> | Operate only on jobs with specified name. |
| --partition=<names> | Operate only on jobs in the specified partition/queue. |
| --qos=<name> | Operate only on jobs using the specified quality of service. |

| | |
|---|---|
| --reservation=<name> | Operate only on jobs using the specified reservation. |
| --state=<names> | Operate only on jobs in the specified state. |
| --user=<name> | Operate only on jobs from the specified user. |
| --nodelist=<names> | Operate only on jobs using the specified compute nodes. |

**squeue** - View information about jobs.

| | |
|---|---|
| --account=<name> | View only jobs with specified accounts. |
| --clusters=<name> | View jobs on specified clusters. |
| --format=<spec> (e.g. "--format=%i %j") | Output format to display. Specify fields, size, order, etc. |
| --jobs<job_id_list> | Comma separated list of job IDs to display. |
| --name=<name> | View only jobs with specified names. |
| --partition=<names> | View only jobs in specified partitions. |
| --priority | Sort jobs by priority. |
| --qos=<name> | View only jobs with specified Qualities Of Service. |
| --start | Report the expected start time and resources to be allocated for pending jobs in order of increasing start time. |
| --state=<names> | View only jobs with specified states. |
| --users=<names> | View only jobs for specified users. |

**sinfo** - View information about nodes and partitions.

| | |
|---|---|
| --all | Display information about all partitions. |
| --dead | If set, only report state information for non-responding (dead) nodes. |
| --format=<spec> | Output format to display. |
| --iterate=<seconds> | Print the state at specified interval. |
| --long | Print more detailed information. |
| --Node | Print information in a node-oriented format. |
| --partition=<names> | View only specified partitions. |
| --reservation | Display information about advanced reservations. |
| -R | Display reasons nodes are in the down, drained, fail or failing state. |
| --state=<names> | View only nodes specified states. |

**scontrol** - Used view and modify configuration and state. Also see the **sview** graphical user interface version.

| | |
|---|---|
| --details | Make show command print more details. |
| --oneliner | Print information on one line. |

Commands:

| | |
|---|---|
| create *SPECIFICATION* | Create a new partition or . |
| delete *SPECIFICATION* | Delete the entry with the specified SPECIFICATION |
| reconfigure | All Slurm daemons will re-read the configuration file. |
| requeue JOB_LIST | Requeue a running, suspended or completed batch job. |
| show ENTITY ID | Display the state of the specified entity with the specified identification |
| update *SPECIFICATION* | Update job, step, node, partition, or reservation configuration per the supplied specification. |

## Environment Variables

| | |
|---|---|
| SLURM_ARRAY_JOB_ID | Set to the job ID if part of a job array. |
| SLURM_ARRAY_TASK_ID | Set to the task ID if part of a job array. |
| SLURM_CLUSTER_NAME | Name of the cluster executing the job. |
| SLURM_CPUS_PER_TASK | Number of CPUs requested per task. |
| SLURM_JOB_ACCOUNT | Account name. |
| SLURM_JOB_ID | Job ID. |
| SLURM_JOB_NAME | Job Name. |
| SLURM_JOB_NODELIST | Names of nodes allocated to job. |
| SLURM_JOB_NUM_NODES | Number of nodes allocated to job. |
| SLURM_JOB_PARTITION | Partition/queue running the job. |
| SLURM_JOB_UID | User ID of the job's owner. |
| SLURM_JOB_USER | User name of the job's owner. |
| SLURM_RESTART_COUNT | Number of times job has restarted. |
| SLURM_PROCID | Task ID (MPI rank). |
| SLURM_STEP_ID | Job step ID. |
| SLURM_STEP_NUM_TASKS | Task count (number of MPI ranks). |

## Daemons

| | |
|---|---|
| slurmctld | Executes on cluster's "head" node to manage workload. |
| slurmd | Executes on each compute node to locally manage resources. |
| slurmdbd | Manages database of resources limits, licenses, and archives accounting records. |

SchedMD Slurm Support and Development

slurm® workload manager



http://www.schedmd.com

# SLURM Environment Variables

Environment variables can be used to:

- pass parameters to your shell scripts
  - See --export option to sbatch (read the manual page with: `man sbatch`)
  - Scripts need to be written to look for them
- define SLURM directives
  - Remember their order of precedence (after command line arguments and before #SBATCH lines)
  - Some examples (which point out that there are more variables than just SLURM_ ones, e.g. SALLOC_ ones) can be found here: https://uwaterloo.ca/math-faculty-computing-facility/services/service-catalogue-teaching-linux/job-submit-commands-examples#slurm-options

# SLURM Documentation

Key links:

- https://slurm.schedmd.com/documentation.html
- https://slurm.schedmd.com/quickstart.html
- https://slurm.schedmd.com/man_index.html

The latest SLURM is 23.02.

We have installed SLURM version 22.05.09.

In a few cases, you might want to consult our specific version's documentation:
https://slurm.schedmd.com/archive/slurm-22.05.9/

# Contents

- Motivation: JHPCE 3.0
- Terminology
- Logging in
- Basics of running programs on the cluster
- Details – limits and resources
- Examples

# Lab 1 - Accessing the SLURM nodes

Log into a new J3 login node by ssh'ing to jhpce03.jhsph.edu

New J3 login servers

- login31 (will become replacement jhpce01) – access as jhpce03.jhsph.edu
- login42 (will become replacement jhpce02) –being built

After logging into one of the SLURM nodes, you will find that:

- you can access SLURM commands and manual pages
- /jhpce/shared/ has different contents than on the J2 cluster
- your home directory & /fastscratch/myscratch are the same on J2 and J3

# Lab 1 - Accessing the SLURM nodes (cont'd)

WARNING

Because your home directory is the same on both J2 and J3, your configuration and other files may not be appropriate for one cluster or the other!!!

For example, your R packages in ~/R/4.3 may have been compiled in the old cluster but are still available in the new. They may not work correctly, and the errors may be silent or hard to track down to the original cause.

You might want to consider:

- renaming or moving aside files or directories like ~/R when switching between the two clusters
- backing up files or directories before trying out the new cluster, e.g.
    - cd; tar czf sge-versions.tgz .[a-zA-Z]* R  # and maybe others

# Lab 2 - Using the SLURM cluster

You can copy a SLURM version of class scripts to your home directory for experimentation.

cd
rsync –a /jhpce/shared/jhpce/slurm/class-scripts class-scripts-slurm

Example 2a – using an **interactive** session

```
srun --pty --x11 bash
cd class-scripts-slurm
./script1 # script is executable and first line is #!/bin/bash
exit
```

Example 2b – submitting a **batch** job

```
cd class-scripts-slurm
sbatch script2  # note script2 doesn't need to be executable
squeue
sstat –j JOBID       # use the number of your job
```

examine results files with the **`cat`** or **`less`** commands

```
slurm-JOBID.out slurm-JOBID.err
```

Note: Your script or interactive shell run in the same directory in which you ran sbatch or srun, unless the --chdir argument is used.

# Never run a job on the login node!

This has not changed in the new cluster. Login nodes have many fewer resources than the compute nodes. They are a shared resource.

- Always use "**`sbatch`**" or "**`srun`**" to make use of the compute nodes
- Jobs that are found running on the login node may be killed at will
- If you are going to be compiling programs, do so on a compute node via **`srun`**.
- Even something as simple as copying large files should be done via **`srun`** `or` **`sbatch`**

# Useful Slurm commands

## **squeue** – shows information about running & pending jobs

```
squeue  # defaults to all jobs for all users
squeue --me -t r,pd  # just my running & pending jobs
```

## **sacct** – shows information about completed jobs

```
sacct -aj JOBID
sacct -S=2023-06-1514:30  # started after 2:30pm
sacct --helpformat # lists avail info fields
sacct --units=M –j 130.batch -o
JobID,MaxVMSizeNode,MaxVMSize,AveVMSize,MaxRSS,AveRSS,MaxDiskRead,MaxDiskWrit
e,AveCPUFreq,TRESUsageInMax%-20  # that is all on one line; capitalization does not matter
```

## **scancel** – deletes your job (also pause)

```
scancel JOBID
scancel –u <username>  # cancels all of that user's jobs
```

# Useful Slurm commands (cont'd)

## `scontrol` – shows information about many types of things

`scontrol show node [nodename]` # defaults to all nodes

```
NodeName=compute-153 Arch=x86_64 CoresPerSocket=32
   CPUAlloc=56 CPUEfctv=128 CPUTot=128 CPULoad=7.00

   NodeAddr=compute-153 NodeHostName=compute-153 Version=22.05.9
   OS=Linux 5.14.0-284.11.1.el9_2.x86_64 #1 SMP PREEMPT_DYNAMIC Tue May 9 17:09:15 UTC 2023

   RealMemory=515415 AllocMem=458752 FreeMem=281975 Sockets=2 Boards=1
   State=MIXED ThreadsPerCore=2 TmpDisk=0 Weight=1 Owner=N/A MCS_label=N/A
   Partitions=debug,interactive,shared

   CfgTRES=cpu=128,mem=515415M,billing=128
   AllocTRES=cpu=56,mem=448G
```

TRES means Trackable RESources

# Useful Slurm commands (cont'd)

**scontrol** – shows information about many types of things

```
scontrol show job <jobid>
```

```
login31:~% scontrol show job 22800

JobId=22800 JobName=scr120
   UserId=tunison(42629) GroupId=users(100)
   Priority=1 Nice=0 Account=(null) QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)

   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:00:49 TimeLimit=90-00:00:00 TimeMin=N/A
   SubmitTime=2023-08-15T21:09:02 EligibleTime=2023-08-15T21:09:02
   AccrueTime=2023-08-15T21:09:02
   StartTime=2023-08-15T21:09:03 EndTime=2023-11-13T20:09:03 Deadline=N/A
   Scheduler=Backfill

   Partition=shared AllocNode:Sid=login31:3264028
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=compute-094
   BatchHost=compute-094

   NumNodes=1 NumCPUs=2 NumTasks=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=2,mem=2G,node=1,billing=2
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
   MinCPUsNode=1 MinMemoryNode=2G MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
   Command=/users/tunison/scr120
   WorkDir=/users/tunison
   StdErr=/users/tunison/slurm-22800.out
   StdIn=/dev/null
   StdOut=/users/tunison/slurm-22800.out
```

# Contents

- Motivation: JHPCE 3.0
- Terminology
- Logging in
- Basics of running programs on the cluster
- Details – limits and resources
- Examples

# Requesting additional RAM & cores

- By default, when you submit a job with sbatch, or run srun, you are allotted 5GB of RAM and 1 core for your job.

- You can request more RAM by setting the "--mem" or “--mem-per-cpu” options
  - --mem: memory per node (for all cores used)
  - --mem-per-cpu: memory per core (harder to accurately estimate)

- Examples:

```
sbatch --mem=10G job1.sh
```

or

```
srun --mem-per-cpu=5G --cpus-per-task=2 --pty --x11 bash
```

# Estimating RAM usage

- No easy formula. Running an example job best.
- A good place to start is the size of the files you will be reading in. Add a bit extra, as a starting point.
- You can add sstat commands to your sbatch scripts (more than once, if desired) to gather info from your running job:

```
sstat –a –o
JobID,MaxVMSizeNode,MaxVMSize,AveVMSize,MaxRSS,AveRS
S,MaxDiskRead,MaxDiskWrite,AveCPUFreq,TRESUsageInMax
–j ${SLURM_JOB_ID} # this is all entered on a single line
```

- Display the fields available for use with –o

```
sstat –e # same as ––helpformat
sacct –e
```

(sstat only works for currently running jobs. Use sacct to see completed jobs.)

# Supplying options to your sbatch job

- You can supply SLURM directives to sbatch in 4 ways:
- Order of precedence:

1. On the command line

   ```
   $ sbatch --mail-type=FAIL,END --mail-user=john@jhu.edu script2
   ```

   Email notification is a great option for a **handful** of long running jobs. This is a **horrible** option for 1000s of jobs, and has caused users to have their email accounts suspended.

2. Environment variables

3. Embedding them in your batch job script

   Lines which start with "#SBATCH" are interpreted as options to **`sbatch.`**

   Such lines must:

   - start at the very beginning of a line
   - come after the interpreter line #!/bin/bash
   - come before any commands

   **`$ less /jhpce/shared/jhpce/slurm/class-scripts/script3.annotated`**

   This file contains many examples!!

# Supplying options to your sbatch job (cont'd)

I believe that this is last in order of precedence:

4. In your ~/.slurm/defaults file

```
Syntax is: [<command>:][<cluster>:] <option> = <value>
Where [ ] indicates an optional argument
Command can be one of (at least): srun, sbatch, salloc
(But perhaps other commands also refer to the file.)
You need to specify an asterisk in between colons
We have not tested blank or commented lines.
```

Example contents:

```
mem=2GB
mail-user=franksmith@jh.edu
srun:*:partition=debug
sbatch:*:mail-type=FAIL,END
```

# Contents

- Motivation: JHPCE 3.0
- Terminology
- Logging in
- Basics of running programs on the cluster
- Details – limits and resources
- Examples

# Modules

Modules are sets of configuration information which change your environment to suite a particular software package.

We have modules for R, SAS, Mathematica, python, . . . but others need to be built under the new operating system.

- module list
- module avail
- module avail stata
- module load
- module unload
- module describe

# Code Directories under /jhpce/shared

Modules and their definition files will be created here

| Contributing users | Is Stored Under /jhpce/shared/... | Example modules |
|---|---|---|
| Certain volunteers | community/ | R (now an alias for conda_R) |
| Lieber | libd/ | nextflow, spaceranger |
| JHPCE Staff | jhpce/ | rstudio, matlab |

# Lab 3

Running R on the cluster:
- In `$HOME/class-scripts/R-demo`, note 2 files – Script file and R file

- Submit Script file
  - `sbatch plot1.sh`
- Run R commands interactively
  - `srun --pty --x11 bash`
  - `module load R`
  - `R`
  - `Open and run plot1.r`

```
[compute-151 slurm/class-scripts-slurm/R-demo]$ cat plot1.sh
#!/bin/bash
# Run the "R" program to read in the "plot1.r" script.

echo "`date`: Loading R Module"
module load R

echo "`date`: Running R Job"
R CMD BATCH plot1.r

echo "`date`: R job complete"
exit 0
[compute-151 slurm/class-scripts-slurm/R-demo]$
```

plot1.r creates plot1-R-results.pdf which you can view with xpdf or a web browser (firefox or chromium-browser)

# Lab 4

## Running RStudio

- X Windows Setup
  - For Windows, MobaXterm has an X server built into it
  - For Mac, you need to have the Xquartz program installed (which requires a reboot), and you need to add the "-X" option to ssh:

```
$ ssh -X yourusername#@jhpce03.jhsph.edu
```

- Start up Rstudio

```
$ srun --pty --x11 --mem=10G bash
$ module load R
$ module load rstudio
$ rstudio
$ exit                          # log out of your srun session
```

# Lab 5a – Running Stata

**Batch:**

```
$ cd $HOME/class-scripts/stata-demo
$ ls
$ less stata-demo1.sh   # see contents of the batch script
$ cat stata-demo1.do    # see contents of the stata program
$ sbatch stata-demo1.sh
$ cat stata-demo1.log   # see the output
```

**Interactive:**

```
$ srun --pty --x11 --cpus-per-task=4 bash
$ module load stata
$ stata-mp
```

or

```
$ xstata-mp # starts the GUI interface
```

Notes:

- The program and script do not need to be named the same, but it is good practice to keep them the same when possible.
- File extensions are sometimes meaningful in Linux. SAS doesn't care, but Stata programs need to have ".do" as the extension. It is good practice for human readability.
- By default "stata" runs a single thread. For faster results when running on real data, request 2 or more cores and use the command "stata-mp" instead of "stata"

# Lab 5b – Running SAS

- SAS example:

**Batch:**

```
$ cd $HOME/class-scripts/SAS-demo
$ ls
$ cat sas-demo1.sh
$ cat class-info.sas
$ sbatch sas-demo1.sh
```

**Interactive:**

```
$ srun --pty --x11 bash      # or use handy bash routine named: jsrun
$ module load sas
$ sas
$ exit                       # log out of your srun session
```

To display a plot, sas needs to send it to a web browser. You can add one or both of the following bash routines to your .bashrc file. They start sas configured to launch Firefox (fsas) or Chrome (csas) if plotting is done.

To see bash routines, run `declare –f routine_name` Example definitions:

```
jsrun ()
{
    /usr/bin/srun --pty --x11 "$@" bash
}

csas ()
{
    sas -helpbrowser SAS -xrm "SAS.webBrowser:'/usr/bin/chromium-browser'" -xrm
"SAS.helpBrowser:'/usr/bin/chromium-browser'" "$@" > /dev/null 2>&1
}
```

# Summary

- Review
  - Use **`ssh jhpce03.jhsph.edu`** to connect to new JHPCE cluster
  - Use sbatch and srun to submit jobs
  - Never run jobs on the login node
  - Helpful resources
    - https://slurm.schedmd.com/documentation.html
    - http://www.jhpce.jhu.edu/
    - bitsupport@lists.johnshopkins.edu - System issues
    - bithelp@lists.johnshopkins.edu - Application issues