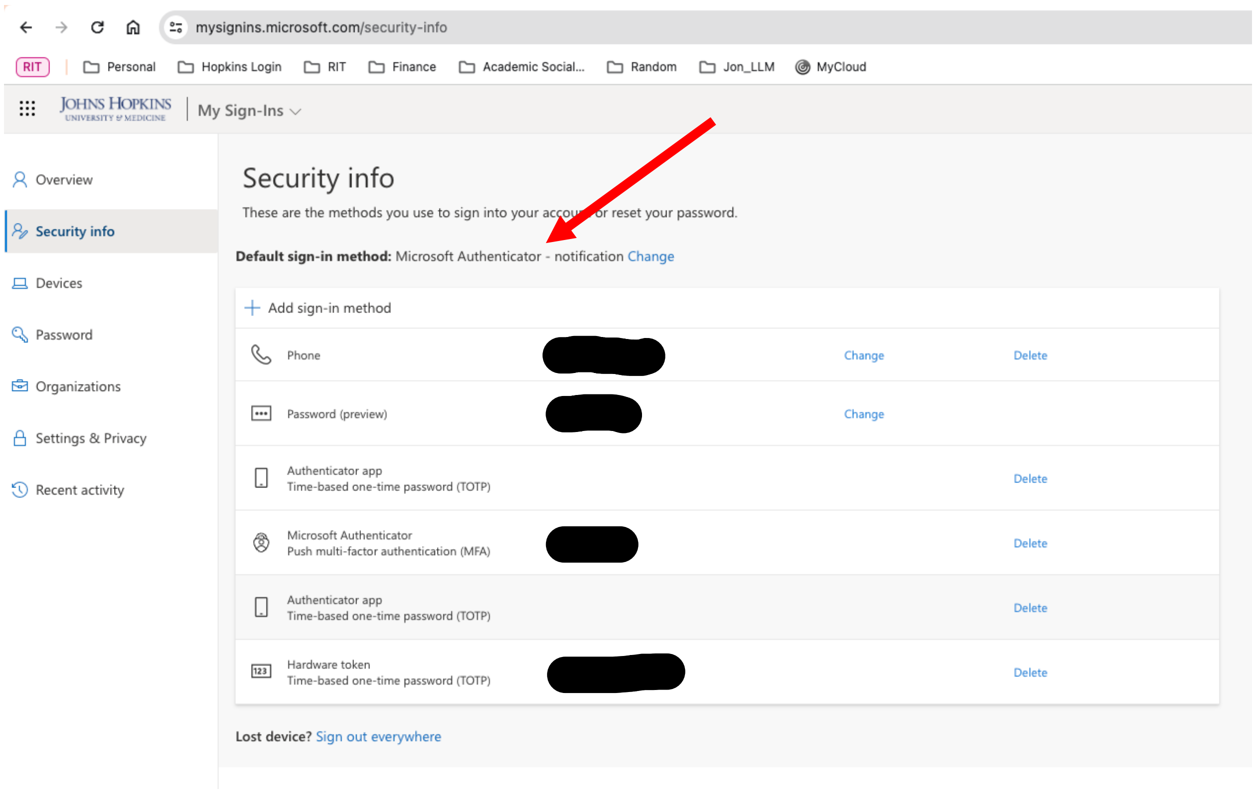
DISCOVERY – The RIT-HPC

DISCOVERY, the Research IT - High-Performance Computing cluster [RIT-HPC], is an excellent resource to help researchers efficiently and expertly complete computational studies and data analysis in a secure, maintained environment. The Research IT team is dedicated to supplying IT solutions to the research community. Support through ServiceNow can be used for reporting an issue, requesting a new tool or application, or requesting a new user account and the rithhpc-help@jh.edu can be used to initiate a ticket with the RIT-HPC admin staff. More documentation will follow, and you will be updated through the rithpc-users@lists.jh.edu which you are subscribed to at time of user provisioning. Use the guide below to get started.

---

1. *Confirm user has Multi Factor Authentication (MFA) that is compatible with HPC login*

Microsoft Authenticator App is the only method of MFA compatible with the RIT-HPC and by going to https://mysignins.microsoft.com/security-info you can confirm your default sign in method. If it is anything other than Microsoft Authenticator, continue to MFA Resource Center to enroll in Microsoft Azure MFA.



Security info

These are the methods you use to sign into your account or reset your password.

Default sign-in method: Microsoft Authenticator - notification Change

+ Add sign-in method

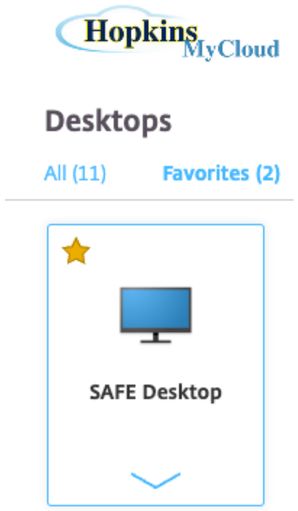
Phone	[redacted]	Change	Delete
Password (preview)	[redacted]	Change	
Authenticator app Time-based one-time password (TOTP)			Delete
Microsoft Authenticator Push multi-factor authentication (MFA)	[redacted]		Delete
Authenticator app Time-based one-time password (TOTP)			Delete
Hardware token Time-based one-time password (TOTP)	[redacted]		Delete

Lost device? Sign out everywhere

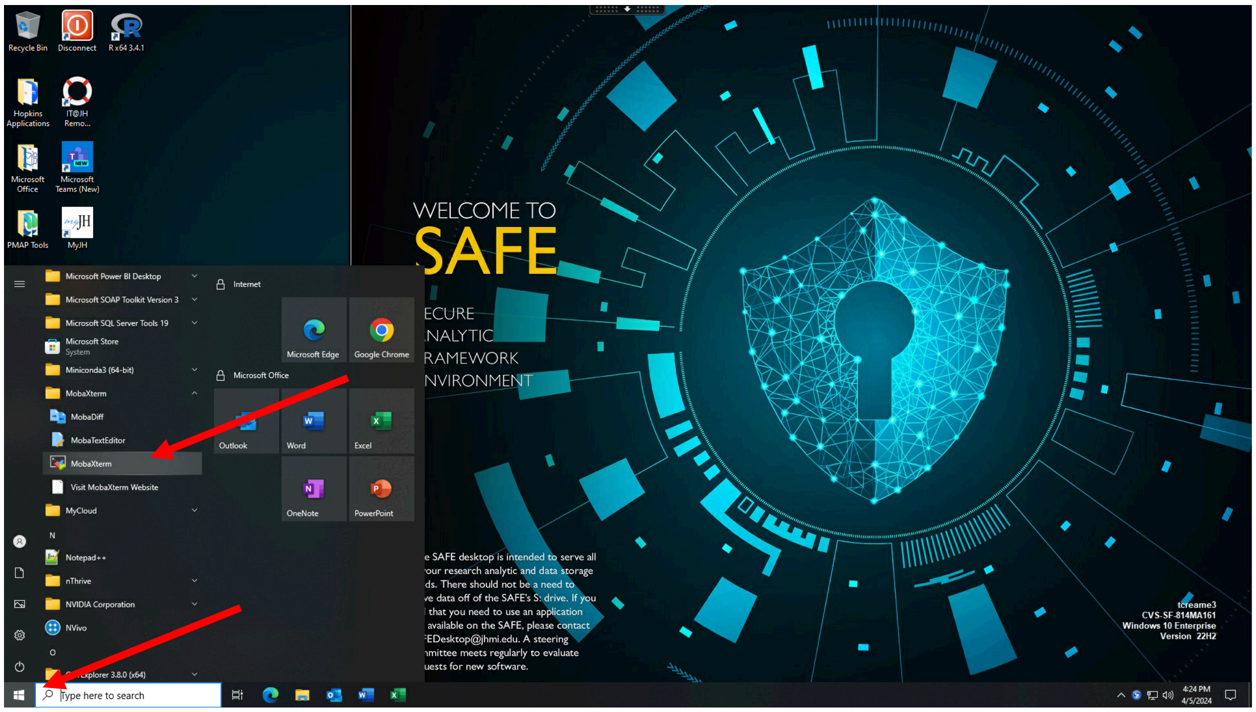
2. *RIT-HPC access through SSH client on SAFE Desktop*

The RIT-HPC is **only** accessible through a SAFE Desktop image and the OnDemand web interface [see Section 3.] SSH clients such as BitVise and MobaXterm are the access software that can be found on the SAFE Desktop. As an RIT-HPC user, you are automatically provisioned a SAFE Desktop. Here we will use MobaXterm as the example for this tutorial.

To access your SAFE Desktop, you need Citrix Workspace installed [Windows or Mac download links] and you can go to https://mycloudgwint.jh.edu/Citrix/MyCloudWeb/ . Select the SAFE Desktop icon and .ica file will be downloaded to your computer. Once you open that configuration file, Citrix will load the SAFE Desktop environment in a new window. This may take a few minutes the first time use SAFE or if you haven't opened SAFE in a while.

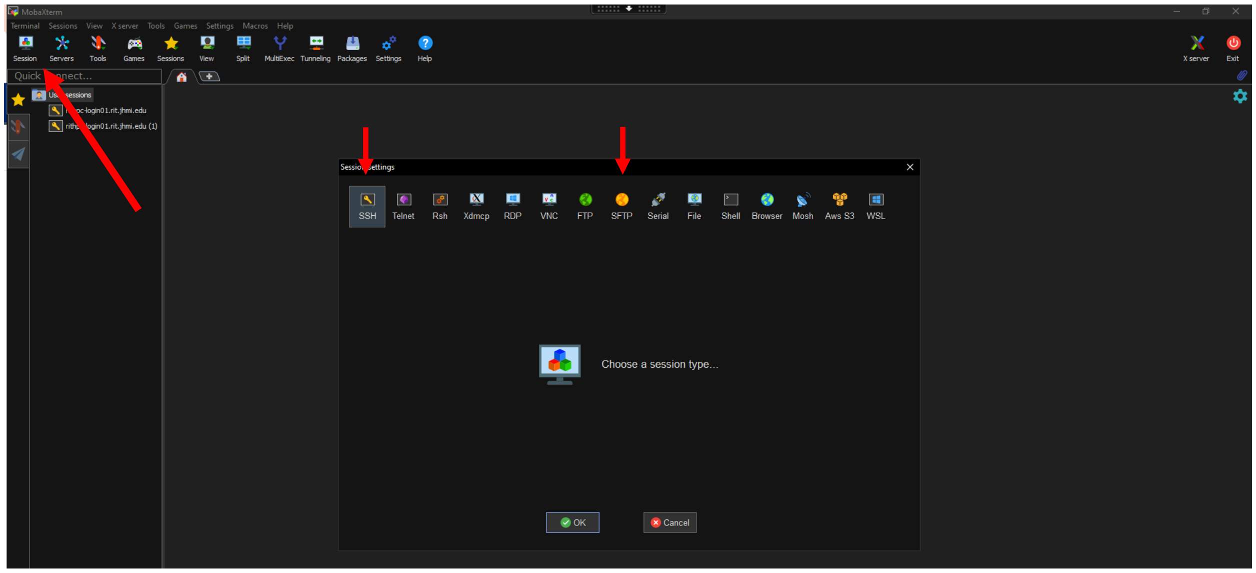


Once in the SAFE Desktop you can navigate to the SSH clients from the Windows Start icon in the lower left corner. The application MobaXterm is listed in a folder with the same name.

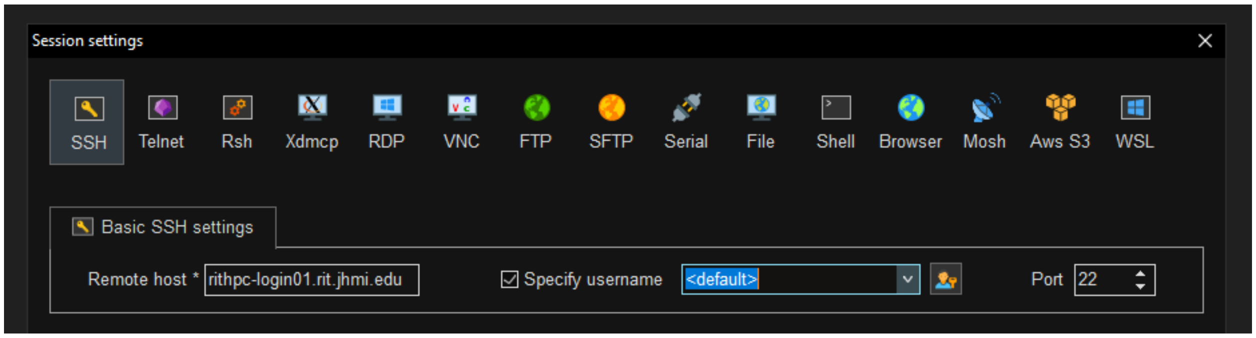


e SAFE desktop is intended to serve all
our research analytic and data storage
ds. There should not be a need to
ve data off of the SAFE's S: drive. If you
that you need to use an application
available on the SAFE, please contact
EDesktop@jhmi.edu. A steering
nmittee meets regularly to evaluate
uests for new software.

Once within the SAFE Desktop environment and with MobaXterm open, you can initiate a new session to connect to the HPC. Here you can select SSH for creating a terminal prompt to access the infrastructure of the HPC and run your commands or SFTP for a GUI for transferring files to and from the SAFE environment.

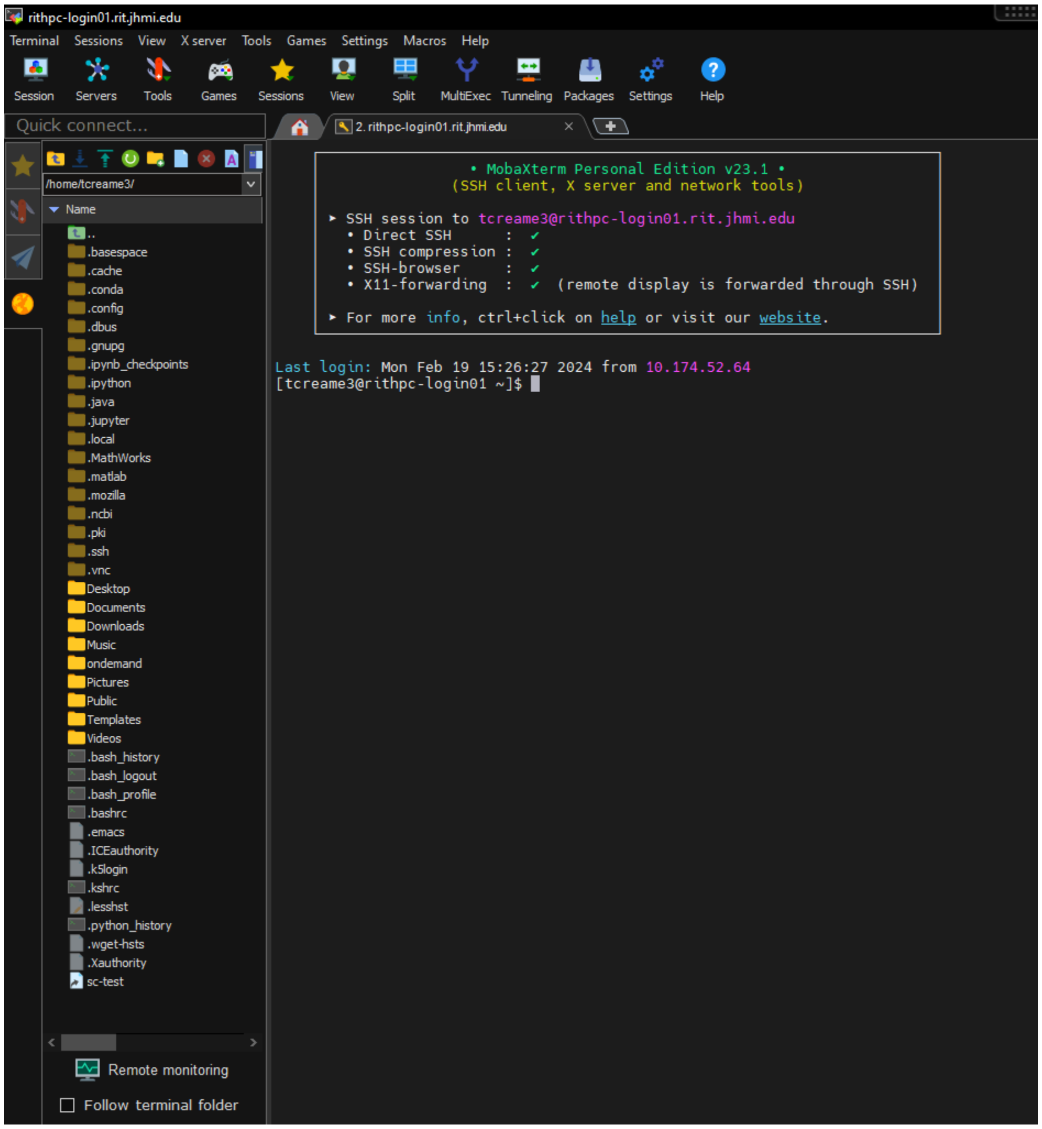

MobaXterm
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
Quick connect...
User sessions
rithpc-login01.rit.jhmi.edu
rithpc-login01.rit.jhmi.edu (1)
Session settings
SSH Telnet Rsh Xdmcp RDP VNC FTP SFTP Serial File Shell Browser Mosh Aws S3 WSL
Choose a session type...
OK
Cancel

Select SSH and the login setting window will appear with fields for the connection settings. The remote host is either `rithpc-login.rit.jhmi.edu`. Specify username will use the login credentials from the SAFE environment. You can also unselect this and use your JHED ID [i.e. tcreame3]. The default port 22 is correct for this connection. You will be asked for your JHED password and then you will need to approve the login on you Microsoft Authenticator App.


Session settings
SSH Telnet Rsh Xdmcp RDP VNC FTP SFTP Serial File Shell Browser Mosh Aws S3 WSL
Basic SSH settings

Remote host * rithpc-login01.rit.jhmi.edu ☑ Specify username <default> Port 22

Once you have successfully logged into the login node, you will be in your home directory that has a 100GB allocation. **<u>Please do not store large or many files here</u>**. It is a staging directory that will house temporary files for the OnDemand services, custom python environment or packages, and/or a small private storage allocation for your use that you don't want shared in the project directory.



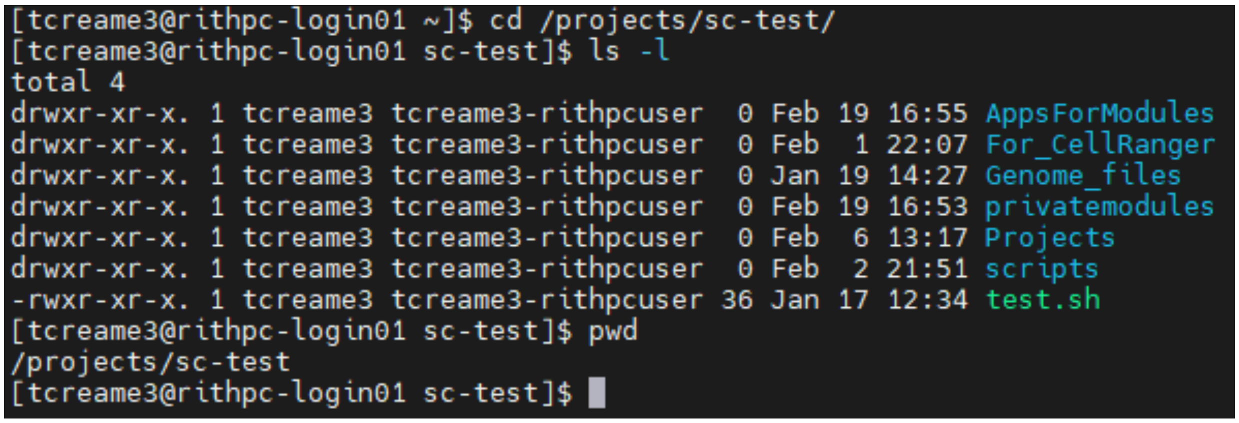
```
                    • MobaXterm Personal Edition v23.1 •
                  (SSH client, X server and network tools)

 ➤ SSH session to tcreame3@rithpc-login01.rit.jhmi.edu
   • Direct SSH      :  ✓
   • SSH compression :  ✓
   • SSH-browser     :  ✓
   • X11-forwarding  :  ✓  (remote display is forwarded through SSH)

 ➤ For more info, ctrl+click on help or visit our website.

Last login: Mon Feb 19 15:26:27 2024 from 10.174.52.64
[tcreame3@rithpc-login01 ~]$
```

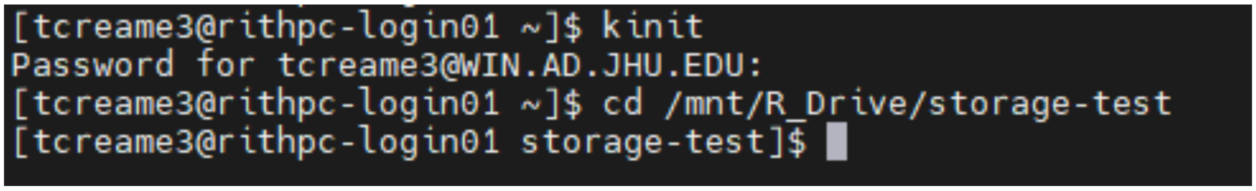
The project directory [Project_PATH = /projects/<Provisioned Directory>] was created for the collective group associated with the lab or project of the provisioned the allocation. Everyone under this group has read/write access to this project directory. This shared space has an allocation size that is determined at the beginning provisioning step but can be increased by submitting a request through email to RITServices@jh.edu or through a future support portal that will be found in ServiceNow.



```
[tcreame3@rithpc-login01 ~]$ cd /projects/sc-test/
[tcreame3@rithpc-login01 sc-test]$ ls -l
total 4
drwxr-xr-x. 1 tcreame3 tcreame3-rithpcuser  0 Feb 19 16:55 AppsForModules
drwxr-xr-x. 1 tcreame3 tcreame3-rithpcuser  0 Feb  1 22:07 For_CellRanger
drwxr-xr-x. 1 tcreame3 tcreame3-rithpcuser  0 Jan 19 14:27 Genome_files
drwxr-xr-x. 1 tcreame3 tcreame3-rithpcuser  0 Feb 19 16:53 privatemodules
drwxr-xr-x. 1 tcreame3 tcreame3-rithpcuser  0 Feb  6 13:17 Projects
drwxr-xr-x. 1 tcreame3 tcreame3-rithpcuser  0 Feb  2 21:51 scripts
-rwxr-xr-x. 1 tcreame3 tcreame3-rithpcuser 36 Jan 17 12:34 test.sh
[tcreame3@rithpc-login01 sc-test]$ pwd
/projects/sc-test
[tcreame3@rithpc-login01 sc-test]$
```

3. *Mount a Research NAS share on the RIT-HPC*

To mount Research NAS [R: drive] allocations to the RIT-HPC, you need to run a code snippet to acquire the proper authentication and then change directories into the allocation to access your share. You are only able to access shares that you have been given access to and Research NAS shares or access control modifications can be requested through the RIT website intake form. While on the RIT-HPC command line, execute the `kinit` command and enter your JHED password to authenticate your NAS share. Then you can change directories to the NAS allocation by using the `cd /mnt/R_Drive/<NAS_Share_Name>` where your NAS share name replaces the final directory destination [see example below; NAS share name is case sensitive]. The `kinit` command **needs** to be run to authenticate the NAS share each session you want to mount the share.
[Note: The Kerberos ticket expires every few days so if you can't connect, try rerunning the kinit command.]



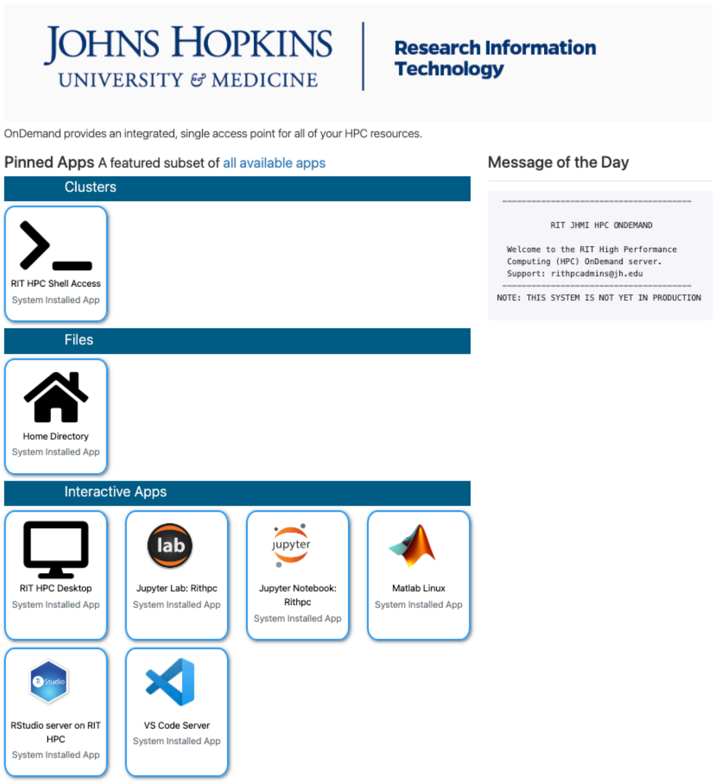
```
[tcreame3@rithpc-login01 ~]$ kinit
Password for tcreame3@WIN.AD.JHU.EDU:
[tcreame3@rithpc-login01 ~]$ cd /mnt/R_Drive/storage-test
[tcreame3@rithpc-login01 storage-test]$
```

* The NAS share can be seen by the login nodes, but the compute nodes cannot see the files. The Research NAS share cannot be used for computational storage space. In other words, analyses cannot happen within the NAS share directory, data must first be copied to an RIT-HPC storage location before compute nodes are able to see it.

4. *OnDemand interactive web portal*

The OnDemand web interface [https://rithpc-ondemand.rit.jhmi.edu/pun/sys/dashboard/] is an accessible graphic user interface (GUI) for a select group of applications and a virtual desktop interface (VDI) to access the Linux RIT-HPC OS.

The applications accessible through OnDemand are Jupyter Lab, Jupyter Notebook, MatLab, RStudio, and VS Code. These applications as well as the RIT-HPC Desktop, which will launch a Linux VDI with the HPC file system, can be reserved with specified resources for a time designated by the user in an easy to use web interface.


JOHNS HOPKINS UNIVERSITY & MEDICINE
Research Information Technology

OnDemand provides an integrated, single access point for all of your HPC resources.

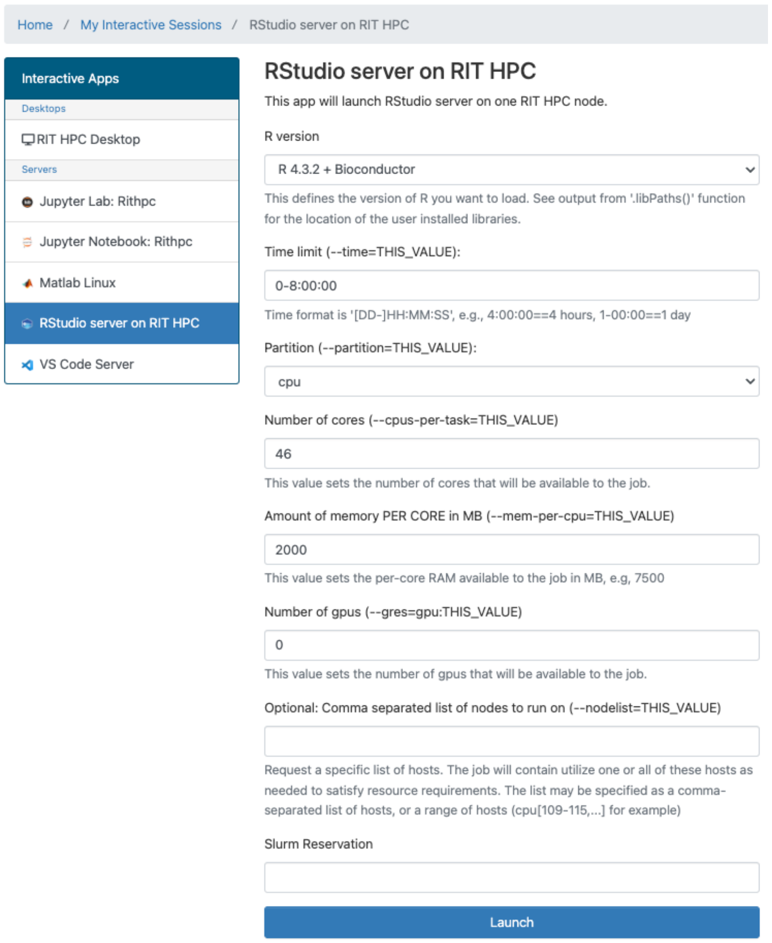
Pinned Apps A featured subset of all available apps

Message of the Day

```
RIT JHMI HPC ONDEMAND

Welcome to the RIT High Performance
Computing (HPC) OnDemand server.
Support: rithpcadmins@jh.edu

NOTE: THIS SYSTEM IS NOT YET IN PRODUCTION
```



RStudio server on RIT HPC

This app will launch RStudio server on one RIT HPC node.

R version: R 4.3.2 + Bioconductor

This defines the version of R you want to load. See output from '.libPaths()' function for the location of the user installed libraries.

Time limit (--time=THIS_VALUE): 0-8:00:00

Time format is '[DD-]HH:MM:SS', e.g., 4:00:00==4 hours, 1-00:00==1 day

Partition (--partition=THIS_VALUE): cpu

Number of cores (--cpus-per-task=THIS_VALUE): 46

This value sets the number of cores that will be available to the job.

Amount of memory PER CORE in MB (--mem-per-cpu=THIS_VALUE): 2000

This value sets the per-core RAM available to the job in MB, e.g, 7500

Number of gpus (--gres=gpu:THIS_VALUE): 0

This value sets the number of gpus that will be available to the job.

Optional: Comma separated list of nodes to run on (--nodelist=THIS_VALUE):

Request a specific list of hosts. The job will contain utilize one or all of these hosts as needed to satisfy resource requirements. The list may be specified as a comma-separated list of hosts, or a range of hosts (cpu[109-115,...] for example)

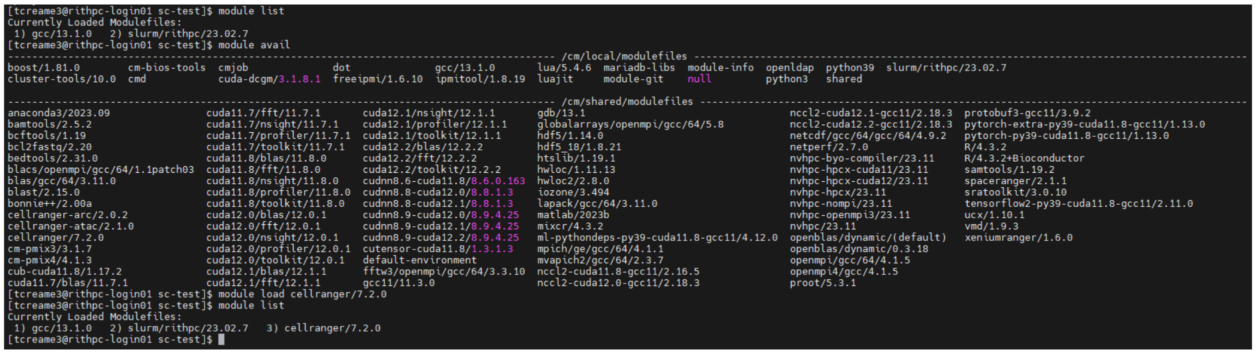
Slurm Reservation:

The reservation has pulldown menus that describe the instance or version of software being selected, the time limit for the reservation up to 10 days, which partition of compute nodes to use, the number of cores needed for the reservation, the MB of RAM **per core**, and the number of GPUs. The partitions are `cpu`, `gpu`, and `interactive` where the `cpu` partition contains a large bank of CPU cores per node, the `gpu` partition has GPU and CPU containing compute nodes, and `interactive` which has a max time limit of 4 hours to be used for code/script testing or optimization. Once launched the page will refresh with a status window that will spin up the interactive session and create a link to a window with the software of interest.

5. *SLURM Scheduler for job submission or interactive session with modular software packages*

The RIT-HPC uses SLURM Workload Manager for server management including, but not limited to, software management though modules, queueing of job submissions, and accounting of usage per user and group. Here is a link to some SLURM command shortcuts which are also pasted on the following pages.

The modular software management can be accessed with the SLURM command `module` and can be used to list loaded modules, see available modules, and load and unload modules for use.



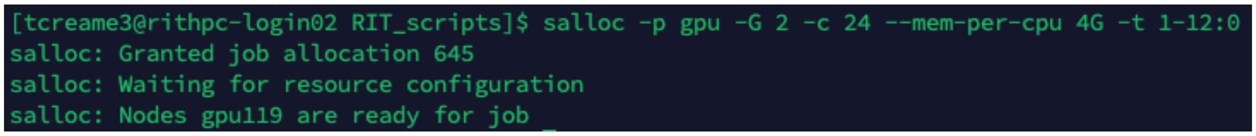
```
[tcreame3@rithpc-login01 sc-test]$ module list
Currently Loaded Modulefiles:
 1) gcc/13.1.0   2) slurm/rithpc/23.02.7
[tcreame3@rithpc-login01 sc-test]$ module avail
------------------------------------------------------------------------------------------- /cm/local/modulefiles -------------------------------------------------------------------------------------------
boost/1.81.0        cm-bios-tools  cmjob              dot              gcc/13.1.0       lua/5.4.6  mariadb-libs  module-info  openldap  python39  slurm/rithpc/23.02.7
cluster-tools/10.0  cmd            cuda-dcgm/3.1.8.1  freeipmi/1.6.10  ipmitool/1.8.19  luajit     module-git    null         python3   shared

------------------------------------------------------------------------------------------ /cm/shared/modulefiles -------------------------------------------------------------------------------------------
anaconda3/2023.09               cuda11.7/fft/11.7.1       cudnn12.1/nsight/12.1.1     gdb/13.1                                 nccl2-cuda12.1-gcc11/2.18.3  protobuf3-gcc11/3.9.2
bamtools/2.5.2                  cuda11.7/nsight/11.7.1    cuda12.1/profiler/12.1.1    globalarrays/openmpi/gcc/64/5.8          nccl2-cuda12.2-gcc11/2.18.3  pytorch-extra-py39-cuda11.8-gcc11/1.13.0
bcftools/1.19                   cuda11.7/profiler/11.7.1  cuda12.1/toolkit/12.1.1     hdf5/1.14.0                              netcdf/gcc/64/gcc/64/4.9.2   pytorch-py39-cuda11.8-gcc11/1.13.0
bcl2fastq/2.20                  cuda11.7/toolkit/11.7.1   cuda12.2/blas/12.2.2        hdf5_18/1.8.21                           netperf/2.7.0                R/4.3.2
bedtools/2.31.0                 cuda11.8/blas/11.8.0      cuda12.2/fft/12.2.2         htslib/1.19.1                            nvhpc-byo-compiler/23.11     R/4.3.2+Bioconductor
blacs/openmpi/gcc/64/1.1patch03 cuda11.8/fft/11.8.0       cuda12.2/toolkit/12.2.2     hwloc/1.11.13                            nvhpc-hpcx-cuda11/23.11      samtools/1.19.2
blas/gcc/64/3.11.0              cuda11.8/nsight/11.8.0    cudnn8.6-cuda11.8/8.6.0.163 hwloc2/2.8.0                             nvhpc-hpcx-cuda12/23.11      spaceranger/2.1.1
blast/2.15.0                    cuda11.8/profiler/11.8.0  cudnn8.8-cuda12.0/8.8.1.3   iozone/3.494                             nvhpc-hpcx/23.11             sratoolkit/3.0.10
bonnie++/2.00a                  cuda11.8/toolkit/11.8.0   cudnn8.8-cuda12.1/8.8.1.3   lapack/gcc/64/3.11.0                     nvhpc-nompi/23.11            tensorflow2-py39-cuda11.8-gcc11/2.11.0
cellranger-arc/2.0.2            cuda12.0/blas/12.0.1      cudnn8.9-cuda12.0/8.9.4.25  matlab/2023b                             nvhpc-openmpi3/23.11         ucx/1.10.1
cellranger-atac/2.1.0           cuda12.0/fft/12.0.1       cudnn8.9-cuda12.1/8.9.4.25  mixcr/4.3.2                              nvhpc/23.11                  vmd/1.9.3
cellranger/7.2.0                cuda12.0/nsight/12.0.1    cudnn8.9-cuda12.2/8.9.4.25  ml-pythondeps-py39-cuda11.8-gcc11/4.12.0 openblas/dynamic/(default)   xeniumranger/1.6.0
cm-pmix3/3.1.7                  cuda12.0/profiler/12.0.1  cutensor-cuda11.8/1.3.1.3   mpich/ge/gcc/64/4.1.1                    openblas/dynamic/0.3.18
cm-pmix4/4.1.3                  cuda12.0/toolkit/12.0.1   default-environment         mvapich2/gcc/64/2.3.7                    openmpi/gcc/64/4.1.5
cub-cuda11.8/1.17.2             cuda12.1/blas/12.1.1      fftw3/openmpi/gcc/64/3.3.10 nccl2-cuda11.8-gcc11/2.16.5              openmpi4/gcc/4.1.5
cuda11.7/blas/11.7.1            cuda12.1/fft/12.1.1       gcc11/11.3.0                nccl2-cuda12.0-gcc11/2.18.3              proot/5.3.1
[tcreame3@rithpc-login01 sc-test]$ module load cellranger/7.2.0
[tcreame3@rithpc-login01 sc-test]$ module list
Currently Loaded Modulefiles:
 1) gcc/13.1.0   2) slurm/rithpc/23.02.7   3) cellranger/7.2.0
[tcreame3@rithpc-login01 sc-test]$ 
```

As described above in the OnDemand section, the partitions are `cpu`, `gpu`, and `interactive` where the `cpu` partition contains a large bank of CPU cores per node, the `gpu` partition has GPU and CPU containing compute nodes, and `interactive` which has a max time limit of 4 hours to be used for code/script testing or optimization. To see the resources per node, use the command ` sinfo -N -o "%20N  %10c  %10m  %25f  %10G "` to display CPU core, RAM, and GPU information per node.

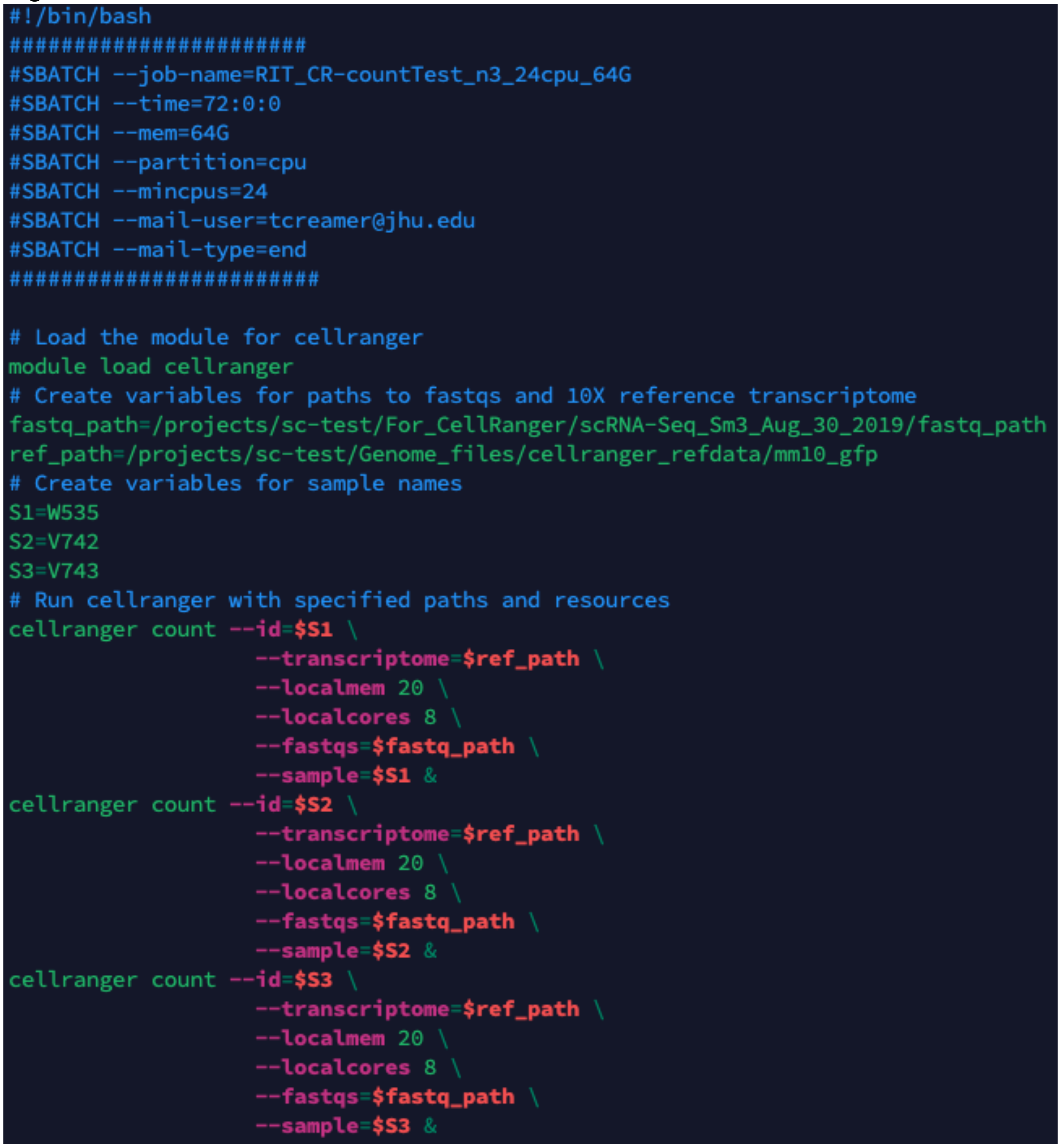
To request an interactive session on a compute node, one can invoke `salloc` which will create an interactive allocation. In the terminal window, you can use the manual pages to see the parameters for the slurm commands (e.g. `man salloc` for the salloc command).

Below `salloc -p gpu -G 2 -c 24 --mem-per-cpu 4G -t 1-12:0` is used to create a session on the gpu partition with 2 GPUs, 24 CPUs, and 96GB of RAM for 1day and 12 hours.



```
[tcreame3@rithpc-login02 RIT_scripts]$ salloc -p gpu -G 2 -c 24 --mem-per-cpu 4G -t 1-12:0
salloc: Granted job allocation 645
salloc: Waiting for resource configuration
salloc: Nodes gpu119 are ready for job
```

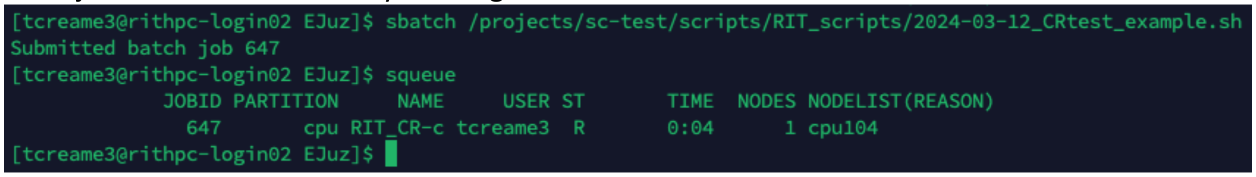
An example SLURM bash script for running 3 samples through CellRanger for single cell RNA seq alignments:



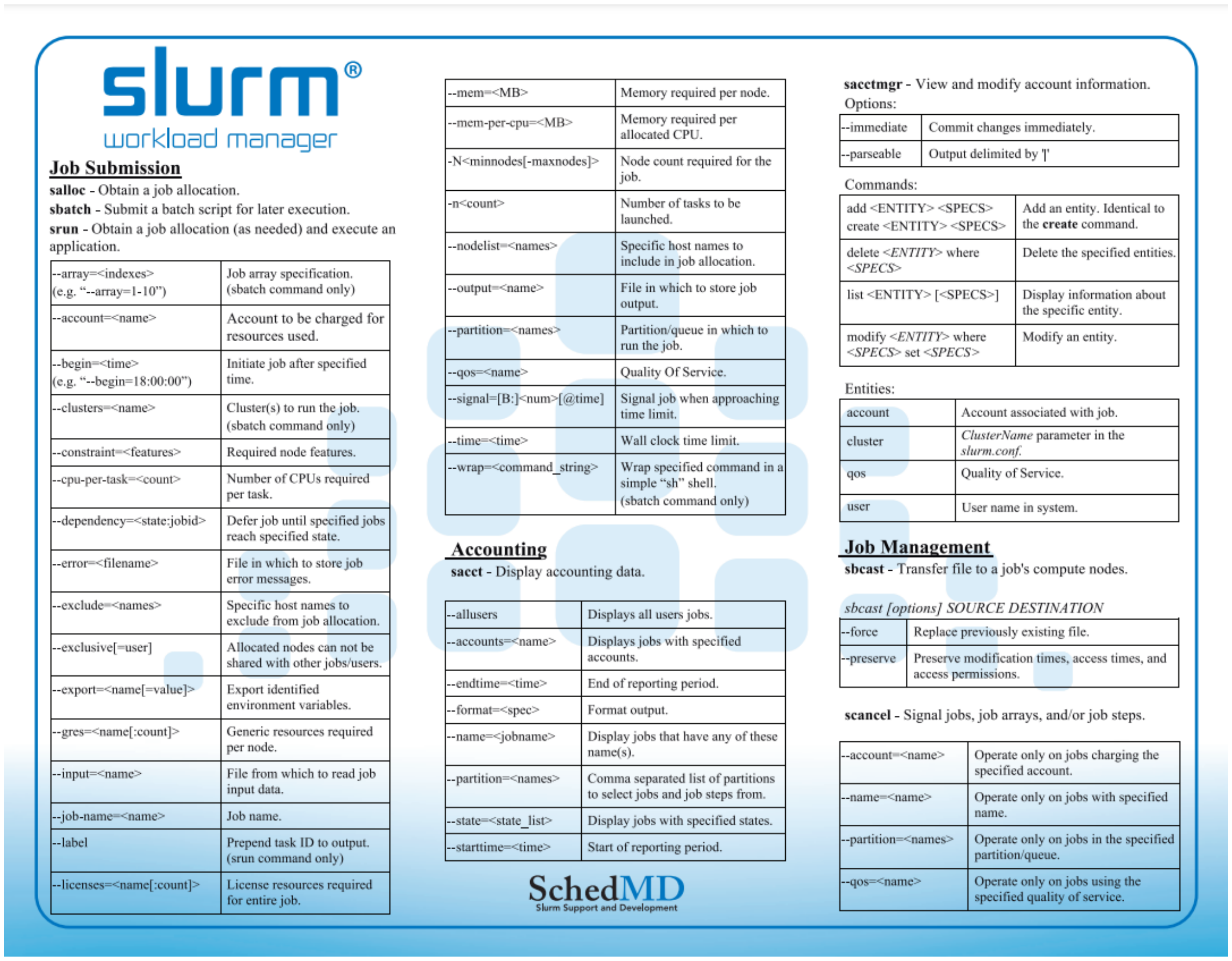
```
#!/bin/bash
#######################
#SBATCH --job-name=RIT_CR-countTest_n3_24cpu_64G
#SBATCH --time=72:0:0
#SBATCH --mem=64G
#SBATCH --partition=cpu
#SBATCH --mincpus=24
#SBATCH --mail-user=tcreamer@jhu.edu
#SBATCH --mail-type=end
########################

# Load the module for cellranger
module load cellranger
# Create variables for paths to fastqs and 10X reference transcriptome
fastq_path=/projects/sc-test/For_CellRanger/scRNA-Seq_Sm3_Aug_30_2019/fastq_path
ref_path=/projects/sc-test/Genome_files/cellranger_refdata/mm10_gfp
# Create variables for sample names
S1=W535
S2=V742
S3=V743
# Run cellranger with specified paths and resources
cellranger count --id=$S1 \
                   --transcriptome=$ref_path \
                   --localmem 20 \
                   --localcores 8 \
                   --fastqs=$fastq_path \
                   --sample=$S1 &
cellranger count --id=$S2 \
                   --transcriptome=$ref_path \
                   --localmem 20 \
                   --localcores 8 \
                   --fastqs=$fastq_path \
                   --sample=$S2 &
cellranger count --id=$S3 \
                   --transcriptome=$ref_path \
                   --localmem 20 \
                   --localcores 8 \
                   --fastqs=$fastq_path \
                   --sample=$S3 &
```

The `sbatch` command will submit the script and the `squeue` command will allow you to see what jobs the user is currently running.



```
[tcreame3@rithpc-login02 EJuz]$ sbatch /projects/sc-test/scripts/RIT_scripts/2024-03-12_CRtest_example.sh
Submitted batch job 647
[tcreame3@rithpc-login02 EJuz]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
               647       cpu RIT_CR-c tcreame3  R       0:04      1 cpu104
[tcreame3@rithpc-login02 EJuz]$ 
```



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

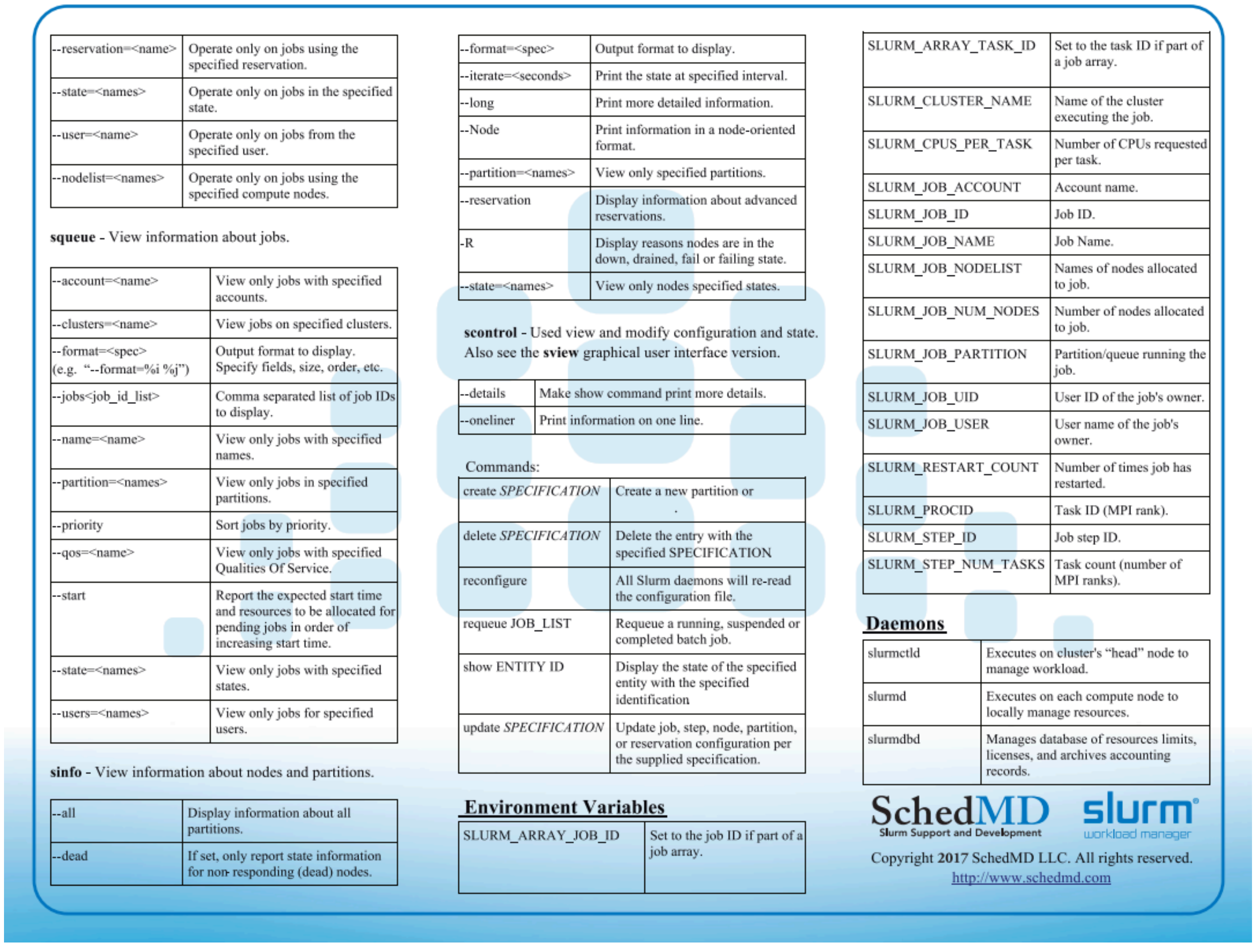
sbcast - Transfer file to a job's compute nodes.

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