

# HIGH PERFORMANCE RESEARCH COMPUTING

## HPRC Training

### Introduction to HPRC Computing Resources

February 9, 2023



High Performance  
Research Computing

DIVISION OF RESEARCH

# Outline

- **Introduction**
  - Setup
  - Usage Policies
  - HPRC Cluster Overview
- **Getting Started**
  - Accessing HPRC Clusters
  - HPRC Portal
- **Software Infrastructure**
  - Module System
  - Python Virtual Environments
- **Cluster Computing with Slurm**
- **Need Help?**

# Introduction



# Setup For This Course

To access the resources:

- HPRC Account: [https://hprc.tamu.edu/user\\_services/#accounts](https://hprc.tamu.edu/user_services/#accounts)
- TFA set up: <https://it.tamu.edu/duo/>

To do the exercises:

- Basic Linux/Unix skills
  - Slides from our LINUX short course are at:  
[hprc.tamu.edu/training/intro\\_linux.html](http://hprc.tamu.edu/training/intro_linux.html)
  - Watch the relevant Introduction and Primer videos on our YouTube Channel:  
<https://www.youtube.com/texasamhprc>

Answers to frequently asked questions can be found on our KB:

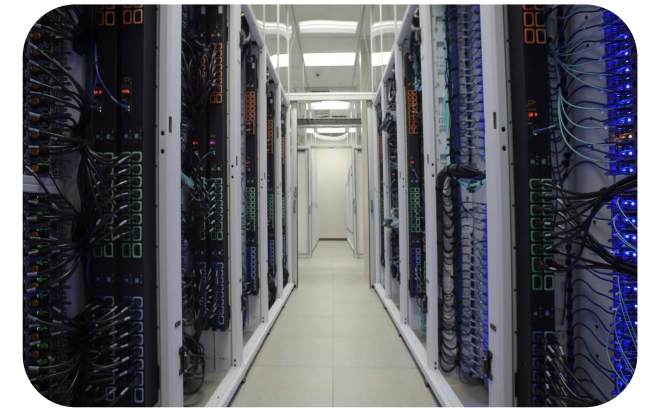
<https://hprc.tamu.edu/kb/FAQ/Accounts/>

# HPRC Clusters

- We operate six clusters
- Today we'll just cover the two "traditional" clusters you're most likely to use:
  - Terra
  - Grace



Terra



Grace

Others:

- Launch
- ViDaL
- **FASTER**
- **ACES**

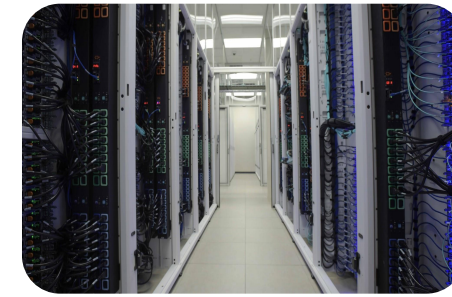
*We have a separate class for these two on Feb 13th!*

See [hprc.tamu.edu/resources](https://hprc.tamu.edu/resources) for more info on all our clusters

# HPRC Clusters: Terra and Grace



Terra

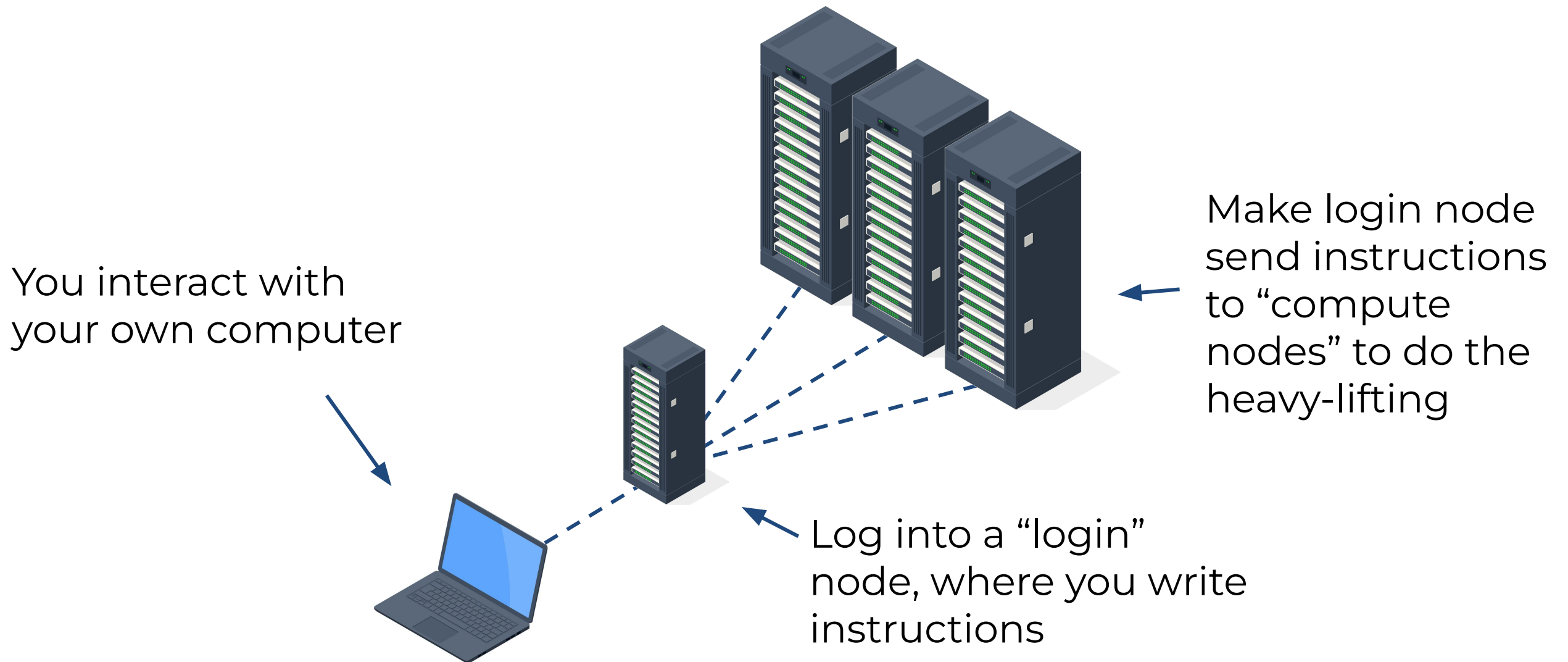


Grace

Total Nodes (Cores)	320 (9,632)	925 (44,656)
General Nodes	28 cores 64GB	48 cores 384GB
Features	GPUs	GPUs - multiprecision Big Memory Nodes
Job Scheduler	Slurm	Slurm
Online Since	2017	2021

[hprc.tamu.edu/resources](https://hprc.tamu.edu/resources)

# Computing on HPRC Clusters



(This image is not to-scale: we have way more machines than this!)

# Clusters Are For You!

What kinds of problems are solved by cluster computing?

- Problems that are too big to fit in a laptop or workstation, due to limitation on memory, core count, or node count
- Problems that scale well with more CPU cores or memory
- Single-threaded problems with many permutations
- Problems that require large high speed storage and/or interconnect



# HPRC Knowledge Base

See our Knowledge Base for announcements, more hardware details, and more about the subjects we cover today.

[hprc.tamu.edu/kb](https://hprc.tamu.edu/kb)  
[hprc.tamu.edu/kb/User-Guides/Grace](https://hprc.tamu.edu/kb/User-Guides/Grace)  
[hprc.tamu.edu/kb/User-Guides/Terra](https://hprc.tamu.edu/kb/User-Guides/Terra)

The screenshot displays the Texas A&M HPRC Knowledge Base website. The top navigation bar includes the HPRC logo, the text "Texas A&M HPRC", a search bar, and a menu with "Home", "Quick Start", "User Guides", "Software", "Helpful Pages", and "FAQ".

The main content area is titled "High Performance Research Computing Home Page". It features an "Announcements" section with a bullet point: "• **FASTER Cluster Status:** Service unit accounting active as of August 1, 2023." Below this is a "Getting an Account" section with two bullet points: "• **Understanding HPRC:** For a brief overview of HPRC and what resources we offer, check out [this page](#) and watch [this video](#) in our getting started series on YouTube. in our getting started series on YouTube." and "• **New to HPRC's resources?** [This page](#) explains the HPRC resources available to the TAMU community. Also".

On the right side, there is a "Table of contents" menu listing: "Announcements", "Getting an Account", "Using The Clusters", "QuickStart Guides", "Batch Jobs", and "HPRC's YouTube Channel".

The "Hardware" section is expanded, showing a list of clusters: "ACES", "FASTER", "Grace", "Hardware", "Key Policies", "Accessing Grace", "Computing Environment", "File Systems", "Compiling and Running Code", "Batch System", "Terra", "OnDemand Portal", "Galaxy", "LMS", and "AMS".

The "Grace: A Dell x86 HPC Cluster" page is shown, featuring a photograph of server racks and a table with the following information:

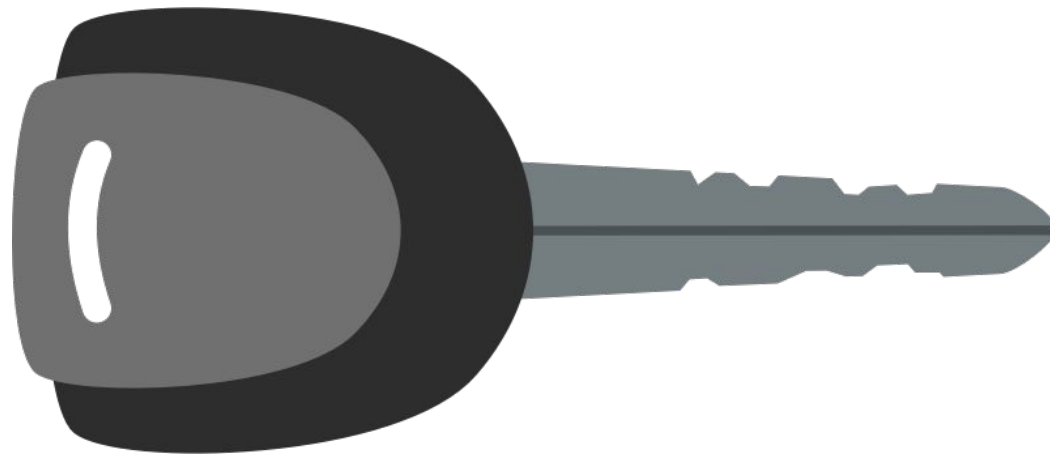
System Name:	Grace
Host Name:	grace.hprc.tamu.edu

The "Terra: A Lenovo x86 HPC Cluster" page is also shown, featuring a table with the following information:

System Name:	Terra
Host Name:	terra.tamu.edu
Operating System:	Linux (CentOS 7)

On the right side of the Terra page, there is a "Table of contents" menu listing: "Usable Memory for Batch Jobs", "Login Nodes", "Mass Storage", "Interconnect", and "Namesake".

# Getting Started



# Usage Policies

## (Be a good compute citizen)

- It is illegal to share computer passwords and accounts by state law and university regulation
- It is prohibited to use HPRC clusters in any manner that violates the United States export control laws and regulations, EAR & ITAR
- Abide by the expressed or implied restrictions in using commercial software

[hprc.tamu.edu/policies](https://hprc.tamu.edu/policies)

# Accessing the HPRC Portal

- (First, if you're off-campus, connect to TAMU VPN: [u.tamu.edu/VPnetwork](https://u.tamu.edu/VPnetwork))
- HPRC webpage: [hprc.tamu.edu](https://hprc.tamu.edu): Portal dropdown menu

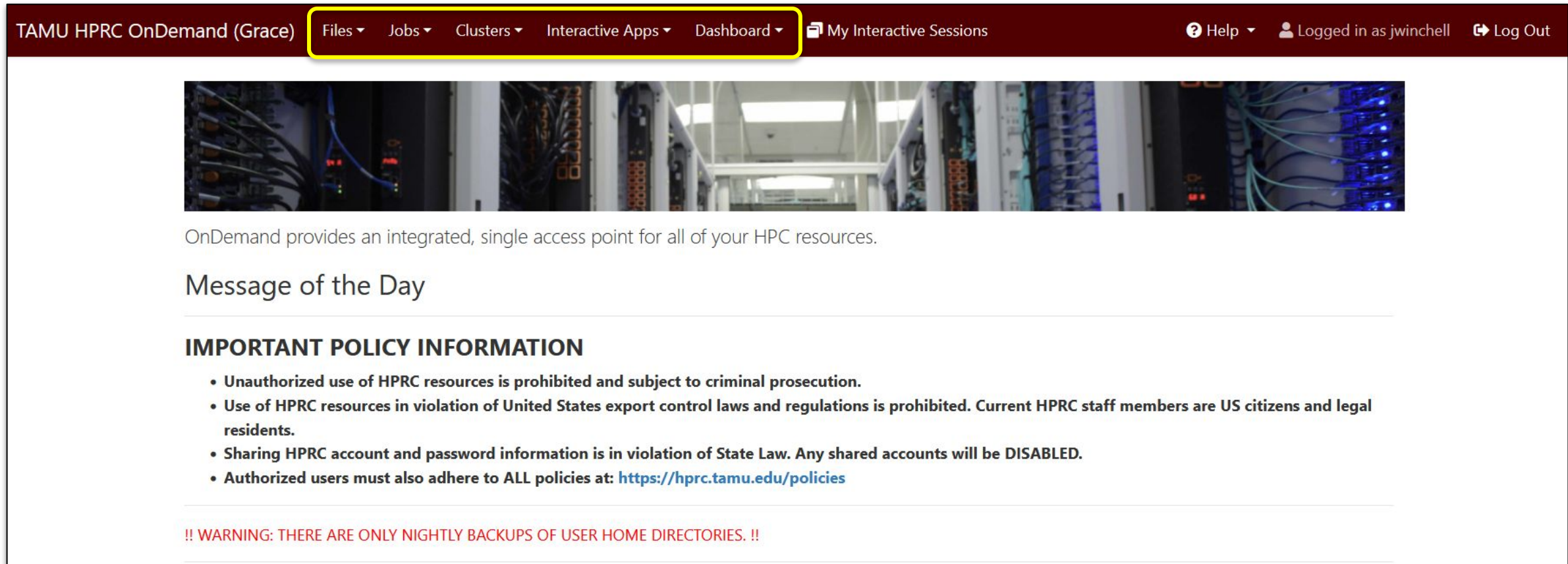
The screenshot displays the HPRC website interface. At the top left is the TAMU logo. The main header reads "TEXAS A&M HIGH PERFORMANCE RESEARCH COMPUTING". On the right side of the header are social media icons for Twitter, YouTube, and WordPress, along with a search icon. The navigation menu includes links for Home, User Services, Resources, Research, Policies, Events, Training, About, and Portal. The Portal link is highlighted with a yellow box. A dropdown menu is open under Portal, listing Terra Portal, Grace Portal, FASTER Portal, FASTER Portal (ACCESS), and ACES Portal (ACCESS). The background of the website features a server room with blue cables and a banner at the bottom that says "TEXAS A&M UNIVERSITY TO ACQUIRE A".

**Quick Links**

- New User Information
- Accounts
  - Apply for Accounts
  - Manage Accounts
- User Consulting
- Training
- Knowledge Base



# HPRC Portal Overview



TAMU HPRC OnDemand (Grace) Files Jobs Clusters Interactive Apps Dashboard My Interactive Sessions Help Logged in as jwinchell Log Out

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

Message of the Day

**IMPORTANT POLICY INFORMATION**

- **Unauthorized use of HPRC resources is prohibited and subject to criminal prosecution.**
- **Use of HPRC resources in violation of United States export control laws and regulations is prohibited. Current HPRC staff members are US citizens and legal residents.**
- **Sharing HPRC account and password information is in violation of State Law. Any shared accounts will be DISABLED.**
- **Authorized users must also adhere to ALL policies at: <https://hprc.tamu.edu/policies>**

**!! WARNING: THERE ARE ONLY NIGHTLY BACKUPS OF USER HOME DIRECTORIES. !!**

**Files:** copy and edit files on the cluster's filesystems

**Jobs:** submit and monitor cluster jobs

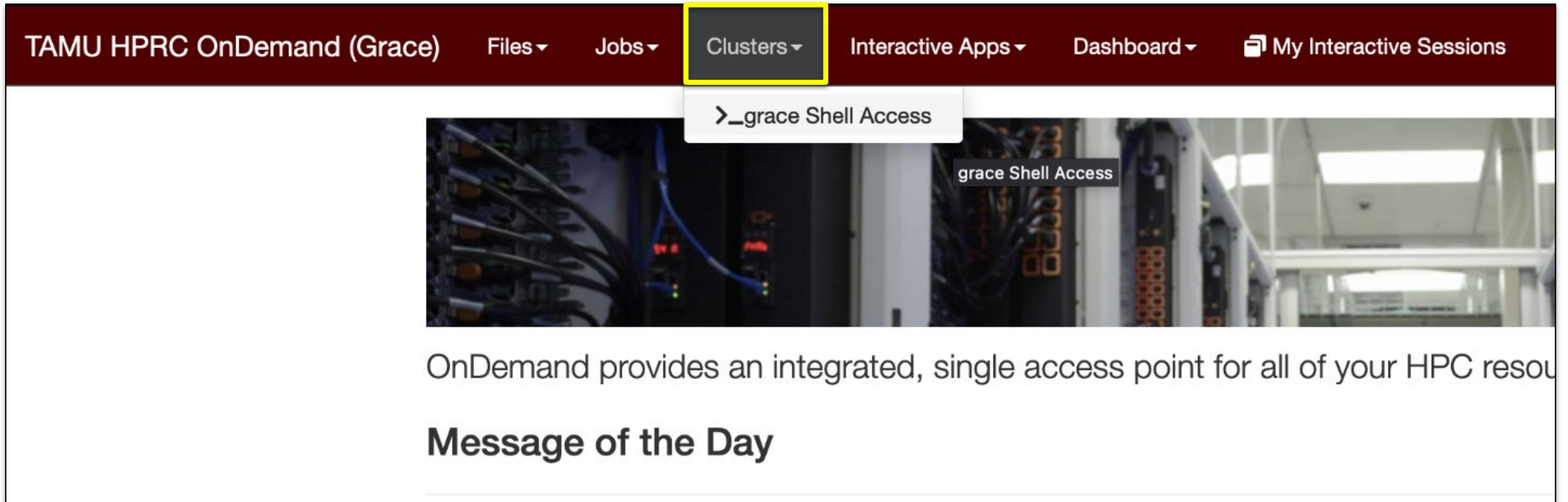
**Clusters:** open a shell terminal (command line) on a login node

**Interactive Apps:** start graphical software on a compute node and connect to it

**Dashboard:** view file quotas and computing account allocations

# Accessing Clusters via the HPRC Portal

- In Portal: “Clusters” dropdown → “Shell Access”



TAMU HPRC OnDemand (Grace) Files ▾ Jobs ▾ Clusters ▾ Interactive Apps ▾ Dashboard ▾ My Interactive Sessions

>\_grace Shell Access

grace Shell Access

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

**Message of the Day**

(Works with most web browsers)

# Accessing Clusters via SSH

- SSH (Secure SHell) allows users to establish a connection between their local machine and the TAMU HPRC clusters.
- SSH Programs:

Operating System	Windows	MacOS	Linux
Programs	<a href="#">MobaXTerm*</a> <a href="#">PowerShell</a> <a href="#">Windows Command Prompt</a> <a href="#">PuTTY SSH</a>	<a href="#">Terminal*</a>	<a href="#">Terminal*</a>
* Recommended			

- If off-campus, connect with VPN first: [u.tamu.edu/VPnetwork](https://u.tamu.edu/VPnetwork)
- If on-campus or connected to VPN, log in with:

```
ssh [NetID]@grace.hprc.tamu.edu
```

```
ssh [NetID]@terra.tamu.edu
```

Green box/highlights =  
text typed into a shell

# Logging In to Clusters

```
Host: grace.hprc.tamu.edu
*****
This computer system and the data herein are available only for authorized
purposes by authorized users. Use for any other purpose is prohibited and may
result in disciplinary actions or criminal prosecution against the user. Usage
may be subject to security testing and monitoring. There is no expectation of
privacy on this system except as otherwise provided by applicable privacy laws.
Refer to University SAP 29.01.03.M0.02 Acceptable Use for more information.
*****

Password: (Type password, not no characters will show up)
Duo two-factor login for [NetID]

Enter a passcode or select one of the following options:

1. Duo Push to XXX-XXX-1234
2. Phone call to XXX-XXX-1234
3. SMS passcodes to XXX-XXX-1234 (next code starts with: 9)

Passcode or option (1-3): 1
Success. Logging you in...
.....
[NetID@grace5 ~]$
```

Once  
connected  
via Portal or  
SSH, sign in  
with TFA



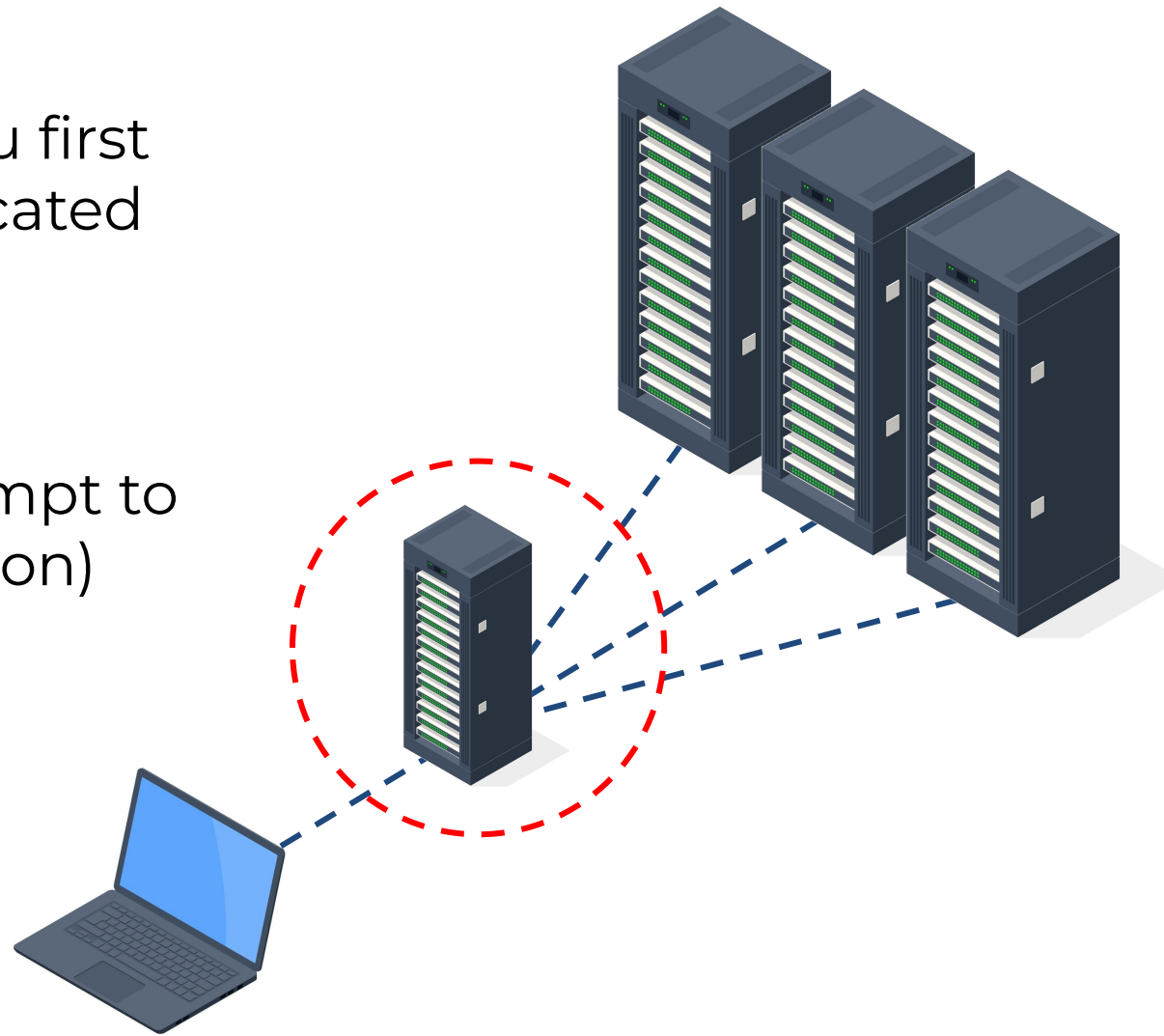
# Hands-on Exercise

1. Log in to the HPRC portal: [hprc.tamu.edu](https://hprc.tamu.edu)
2. Open a terminal on Grace in the portal:  
Clusters → \_Grace Shell Access
3. Use your NetID password and your two-factor Authentication method
4. Do jazz hands if it worked

# Logged-In

Remember, when you first log in, you're on dedicated *login nodes*.

- Grace has 5
  - Terra has 3
- (check your shell prompt to see which one you're on)



These are not for running big processes!

There are rules:

- No processes longer than 1hr
- Sessions idle for 1hr will be killed
- Don't use more than 8 cores
- Don't use sudo

# File Systems and User Directories

Directory	Environment Variable	Space Limit	File Limit	Intended Use
/home/\$USER	\$HOME	10 GB	10,000	Small to modest amounts of processing. Backed-up.
/scratch/user/\$USER	\$SCRATCH	1 TB	250,000	Temporary storage of large files for on-going computations. Not intended to be a long-term storage area; NOT backed up.

**\$SCRATCH** is shared between FASTER (TAMU/ACCESS) and Grace (TAMU) clusters.

View file usage and quota limits on the command line with:

**showquota**

Do **NOT** share your home or scratch directories.

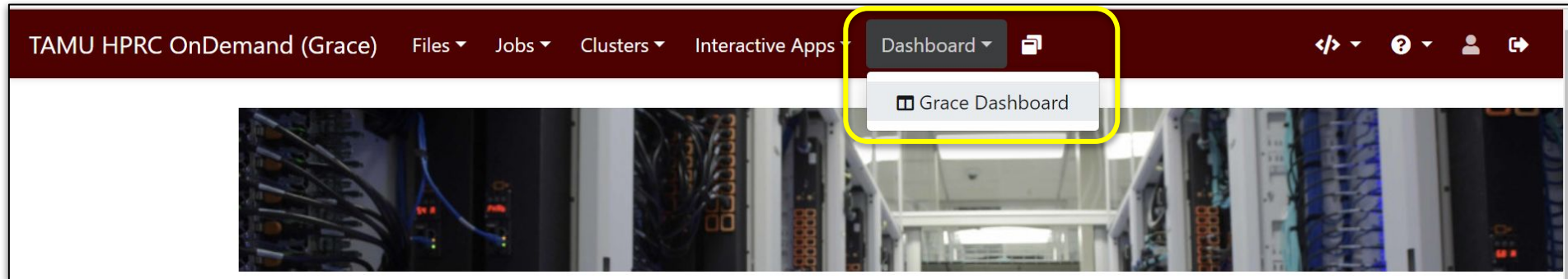
Request a group directory for sharing files.

[https://hprc.tamu.edu/kb/User-Guides/Grace/Filesystems\\_and\\_Files/](https://hprc.tamu.edu/kb/User-Guides/Grace/Filesystems_and_Files/)

[https://hprc.tamu.edu/kb/User-Guides/Terra/Filesystems\\_and\\_Files/](https://hprc.tamu.edu/kb/User-Guides/Terra/Filesystems_and_Files/)

# HPRC Portal Quota Increase Request

Portal Homepage → Grace Dashboard



Request quota increases directly from the dashboard with a guided form

Disk Quotas				
Disk	Disk Usage	Limit	File Usage	Limit
/home	160 KB (0.00 %)	10 GB	35 (0.35 %)	10000
/scratch	837.84 MB (0.08 %)	1 TB	11795 (4.72 %)	250000

[Request Quota Increase](#)

Is this request more than 10TB or for longer than 6 months?  
 Yes  No

Current Scratch Quota  
1 TB

New Scratch Quota  
 TB

Current File Limit  
250000

New File Limit

Justification (Required)  
What data is stored with requested quota?  
What job requires this quota increase?  
What is the input/output size of the job?  
What is your long-term plan for this data?

Comment (Optional)

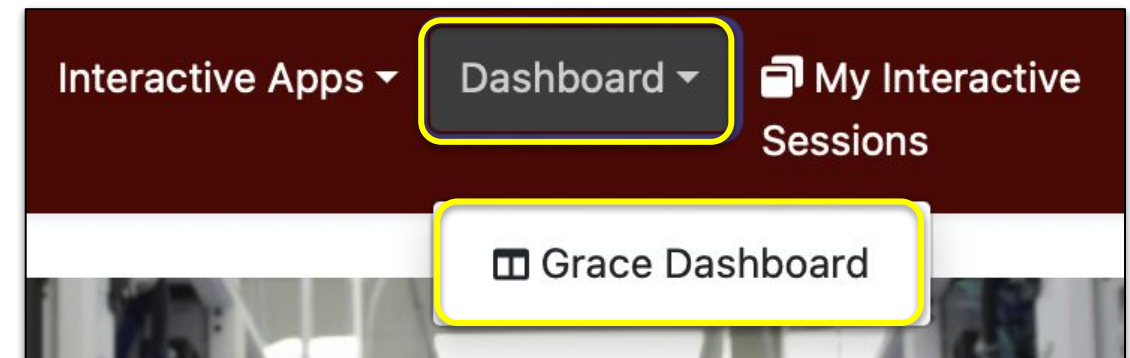
I verify that I will remove any unnecessary data and compress files/folders to save shared resources.

[Submit Request](#)

# Hands-on Exercise

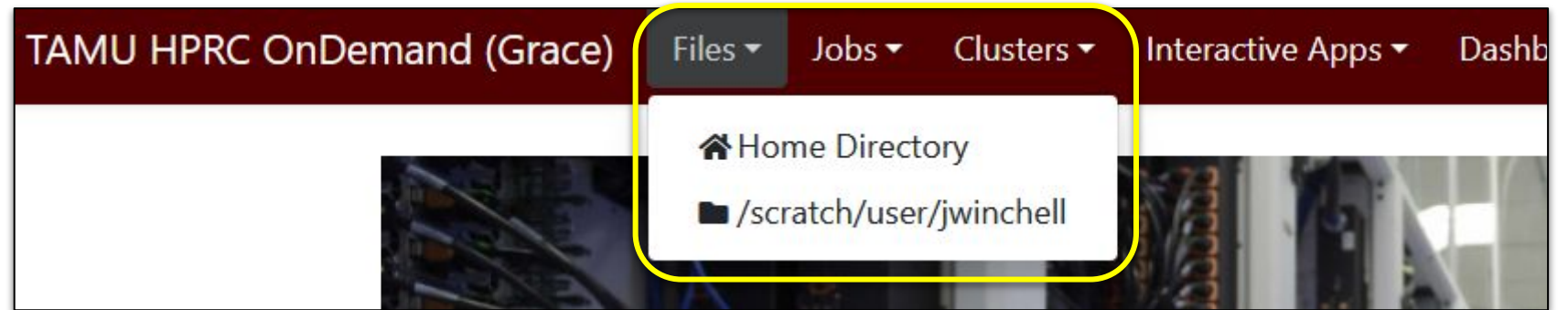
Check your file Quota:

1. In the Portal:  
Dashboard → Grace Dashboard
2. In the shell:  
(use `showquota`)
3. Locate your `/scratch` disk usage stats



# Portal File Browser

The Portal includes a file browser in which you can do many of the things a terminal would do... or just launch a terminal.



Home Directory  
/scratch/user/jwinchell

Open in Terminal | New File | New Directory | Upload | Download | Copy/Move | Delete

/ home / jwinchell / Change directory | Copy path

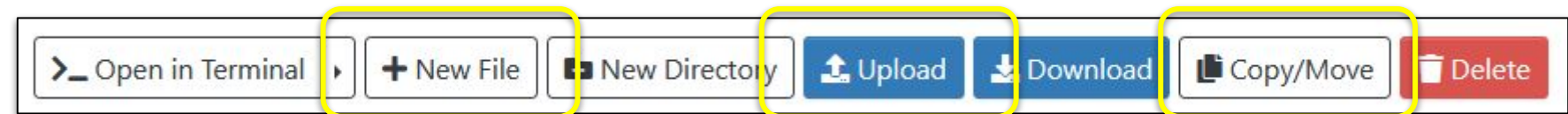
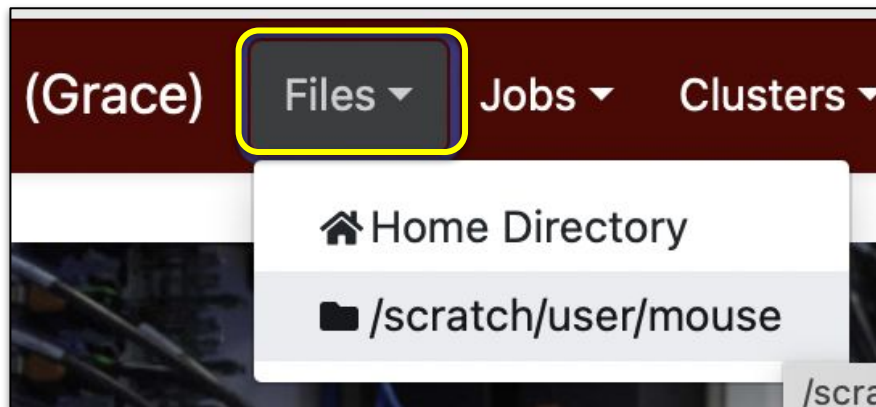
Show Owner/Mode  Show Dotfiles Filter:

Showing 10 of 43 rows - 0 rows selected

Type	Name	Size	Modified at
<input type="checkbox"/> Folder	fortran_tutorial	-	4/8/2022 10:06:50 AM
<input type="checkbox"/> Folder	geant4_workdir	-	3/21/2022 10:11:25 AM

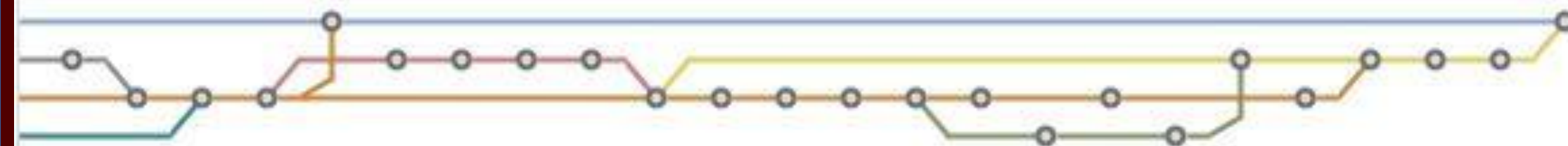
# Hands-on Exercise

1. Enter your Scratch directory from the Portal:  
Files → /scratch/user/<NetID>
2. Create or upload a file
3. Move the file to your home directory





# Software Infrastructure





# Software

- HPRC provides both pre-installed software and installation assistance
- See the Software pages for instructions and examples:
  - [hprc.tamu.edu/kb/Software](https://hprc.tamu.edu/kb/Software)
  - [hprc.tamu.edu/software](https://hprc.tamu.edu/software)
- License-restricted software
  - Check on command line with `license_status` (example later)
  - Contact [help@hprc.tamu.edu](mailto:help@hprc.tamu.edu)
- Contact us for software installation help/request (can use dashboard)
  - User can install software in their home/scratch directories
  - Do **NOT** run the "sudo" command when installing software

# Computing Environment

- **PATH:** the location on disk where an executable or library may be found.
- Paths are saved as **environment variables**, so you can choose which libraries and executables will be used by modifying the variables.
- There is a lot of software, many versions, and many paths to manage.

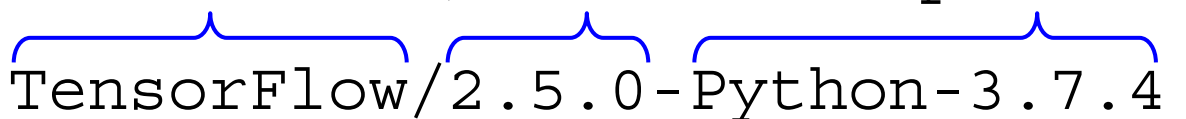
..... How do you manage all these software versions?

<https://hprc.tamu.edu/kb/Software/useful-tools/Modules/>

# Modules

Managing software versions using `lmod`

- Each version of a software, application, library, etc. is available as a [module](#).
  - Module names have the format:

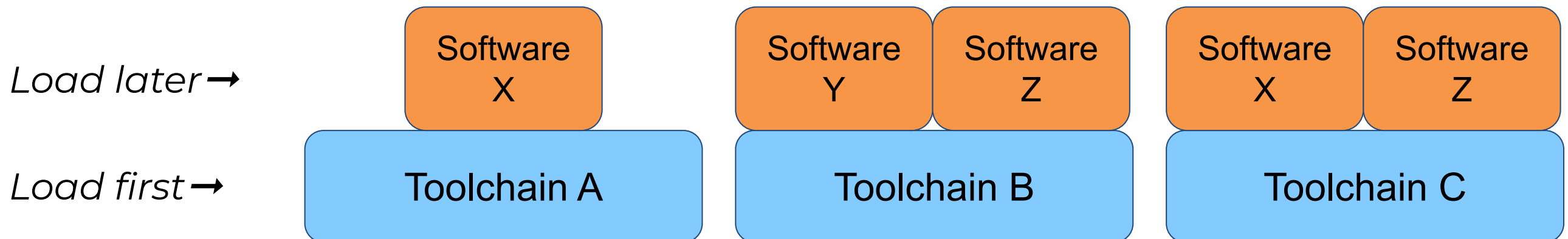
`software-name/version [-dependency-version (optional)]`  
  
`TensorFlow/2.5.0-Python-3.7.4`

*(Note: module names on Terra will have additional information that won't be included on the newer clusters)*

- Loading a module adds its location on disk to your `Path` environment variable.
  - Its dependencies will also be loaded automatically.

# Modules

- Modules are organized *hierarchically*
- The base modules that must be loaded before anything else are called “Toolchains”:
  - The same software may be available from multiple toolchains
  - Don’t mix up software from different toolchains!
- Loading the toolchain makes other modules available



# Modules: Toolchain Details

- Toolchains are combinations of compilers, MPI libraries, and highly optimized math libraries.
- Toolchain components are primarily either Intel or Open Source.

Example toolchains for C++ development:

Components	Open Source	Intel Source	Mixed Source
Compiler only	GCCcore	iccifort	-
Compiler + MPI	gomp	iimpi	iomp
Compiler + MPI + MKL, BLAS, FFTW, LAPACK	foss	intel	iomkl
Compiler + all of the above + CUDA Compiler	fosscuda	intelcuda	iomklc

Example usage: `module load foss/2022a`

<https://hprc.tamu.edu/kb/Software/useful-tools/Toolchains/>

# Module Usage Basics

## `module avail`

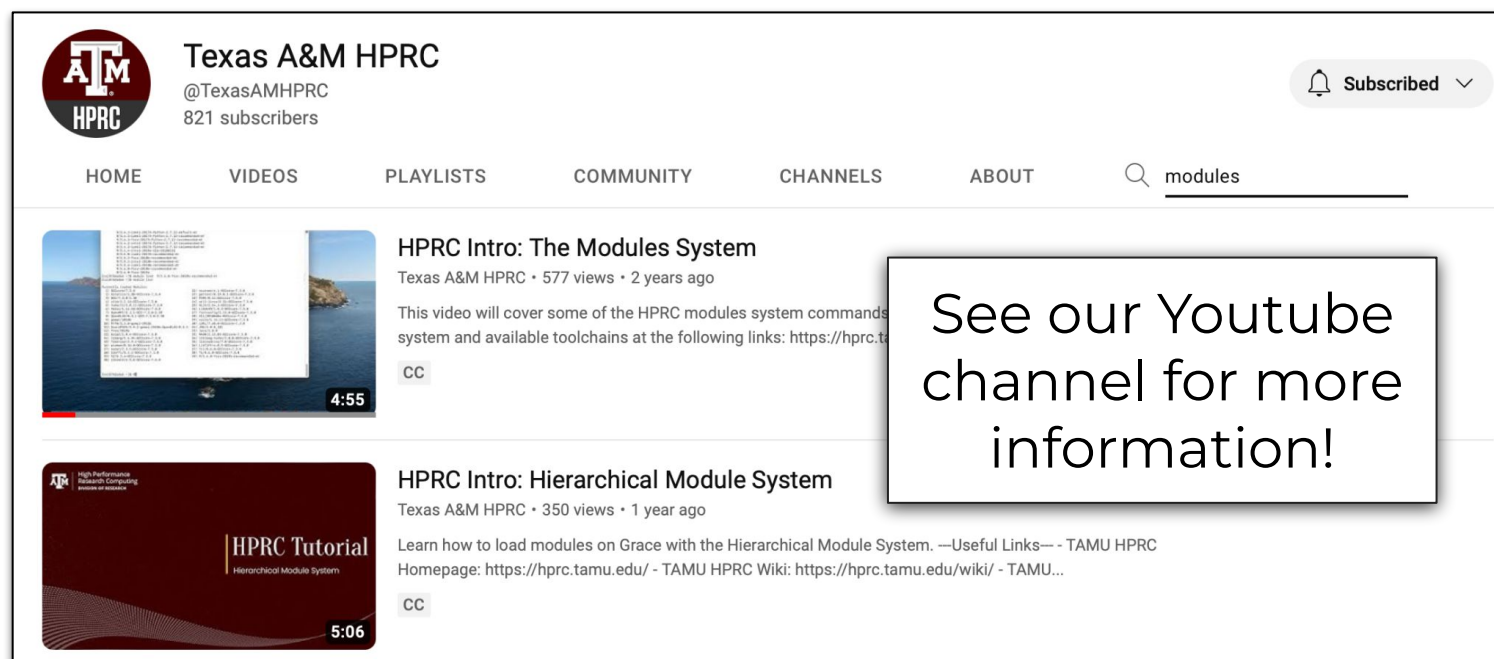
- Lists all available modules (may be slow).
- Navigation:
  - spacebar, arrows, or j and k
  - quit with q
- Case-sensitive search: /
- Use `mla` instead to save results to a file as well

## `module load <module>`

- add `<module>` paths to the current environment variables

## `module spider <word>`

- Case-sensitive search for modules with “word” in name.
- Provide an exact name to see dependencies.



The screenshot shows the YouTube channel for Texas A&M HPRC. The channel name is "Texas A&M HPRC" with the handle "@TexasAMHPRC" and 821 subscribers. The channel is subscribed to. The search bar contains the word "modules". Two videos are displayed:

- HPRC Intro: The Modules System** (4:55): Texas A&M HPRC • 577 views • 2 years ago. Description: "This video will cover some of the HPRC modules system commands system and available toolchains at the following links: https://hprc.t..."
- HPRC Intro: Hierarchical Module System** (5:06): Texas A&M HPRC • 350 views • 1 year ago. Description: "Learn how to load modules on Grace with the Hierarchical Module System. —Useful Links— - TAMU HPRC Homepage: https://hprc.tamu.edu/ - TAMU HPRC Wiki: https://hprc.tamu.edu/wiki/ - TAMU..."

See our Youtube channel for more information!

# Module Commands Reference

- Commonly-used module commands
- Note that on there are some shorthands for you to use

Command	Shorthand	Description
<code>module avail</code>	<code>mla</code> <i>(saves a searchable file as well)</i>	See what modules can be loaded now
<code>module spider</code>		Search for modules and find dependencies
<code>module list</code>	<code>ml</code>	See what modules have been loaded already
<code>module load foo bar</code>	<code>ml foo bar</code>	Load specified module
<code>module unload foo bar</code> <code>module load baz goo</code>	<code>ml -foo -bar baz goo</code>	Load and unload specified modules
<code>module purge</code>	<code>ml purge</code>	Unload all modules

# Module Usage Example

Search for the software “snakemake”;  
saves a file of available modules

```
mla snakemake
```

```
snakemake/5.2.4-Python-3.6.6  
snakemake/5.7.1-Python-3.7.2  
snakemake/5.7.1-Python-3.7.4  
snakemake/5.26.1-Python-3.8.2  
snakemake/6.1.0  
snakemake/6.10.0
```

Choose a specific version and use it  
to search for its dependencies

```
module spider snakemake/6.10.0
```

```
-----  
snakemake: snakemake/6.10.0  
-----
```

**Description:**

The Snakemake workflow management system is a tool to create reproducible and scalable data analyses.

You will need to load all module(s) on any one of the lines below before the "snakemake/6.10.0" module is available to load.

```
GCC/11.2.0 OpenMPI/4.1.1
```

Load the base  
dependency  
module(s) first  
then the  
full module  
name

```
module load GCC/11.2.0 OpenMPI/4.1.1 snakemake/6.10.0
```



# Hands-on Exercise: Module Loading

1.

```
mla blast+
```

-see which versions of BLAST+ are available

2.

```
ml BLAST+/2.13.0
```

-error; you can't do that yet

3.

```
module spider BLAST+/2.13.0
```

-learn how to load this module

4.

```
ml            BLAST+/2.13.0
```

-fill in the blank (with the correct toolchains) to load this module

5.

```
ml list
```

-list all loaded modules

6.

```
ml GCC/10.2.0
```

-change version of a loaded Toolchain module (GCC); notice the message about reloaded modules

7.

```
ml list
```

-list all loaded modules

8.

```
ml purge
```

-remove all loaded modules

# Module Usage Practices

- Software installed as modules are available to all users
  - (except for restricted modules)
- It's a good habit to unload unused modules before loading new modules: `module purge`
- It is recommended to load a specific software version instead of the defaults: `m1 gcc/12.2.0` instead of `m1 gcc`
- Avoid loading modules in your `$HOME/.bashrc`
- Avoid mixing toolchains when loading multiple modules at the same time. This usually leads to one of them not working.

<https://hprc.tamu.edu/kb/Software/useful-tools/Modules/>

# Python and Virtual Environments

Python is a language which supports many external libraries in the form of extensions. (called Python Packages).

Some commonly used packages:

- SciPy & NumPy
- Jupyter notebook
- Scikit-learn

Once you've loaded the appropriate Python module, you can install these additional packages yourself using the *Virtual Environment* feature. Instructions are on our KB:

<https://hprc.tamu.edu/kb/Software/Python/#create-a-virtual-environment>

# Hands-on Exercise: Python Software Install

1.

```
cd $SCRATCH  
mkdir python_envs  
cd python_envs
```

-Move to \$SCRATCH  
Make a place for virtualenvs

2.

```
module purge  
module load GCC/10.2.0 Python/3.8.6
```

-Set up Python module

3.

```
virtualenv my_example_venv  
source my_example_venv/bin/activate
```

-Create and activate virtual environment

4.

```
python -c "import pytime"
```

-Check if python-time is installed (it's not)

5.

```
pip install python-time
```

-Install python-time

6.

```
python -c "import pytime; print(pytime)"
```

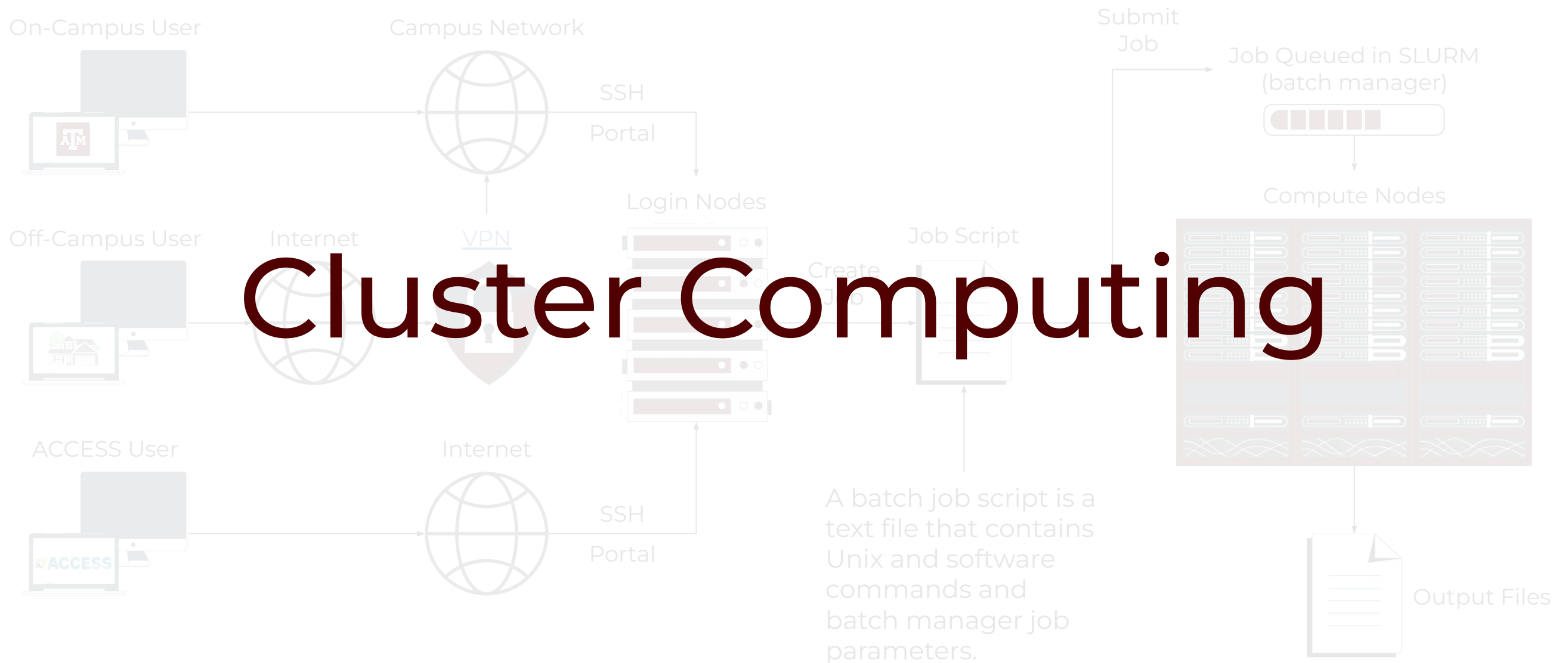
-Where is python-time installed?

7.

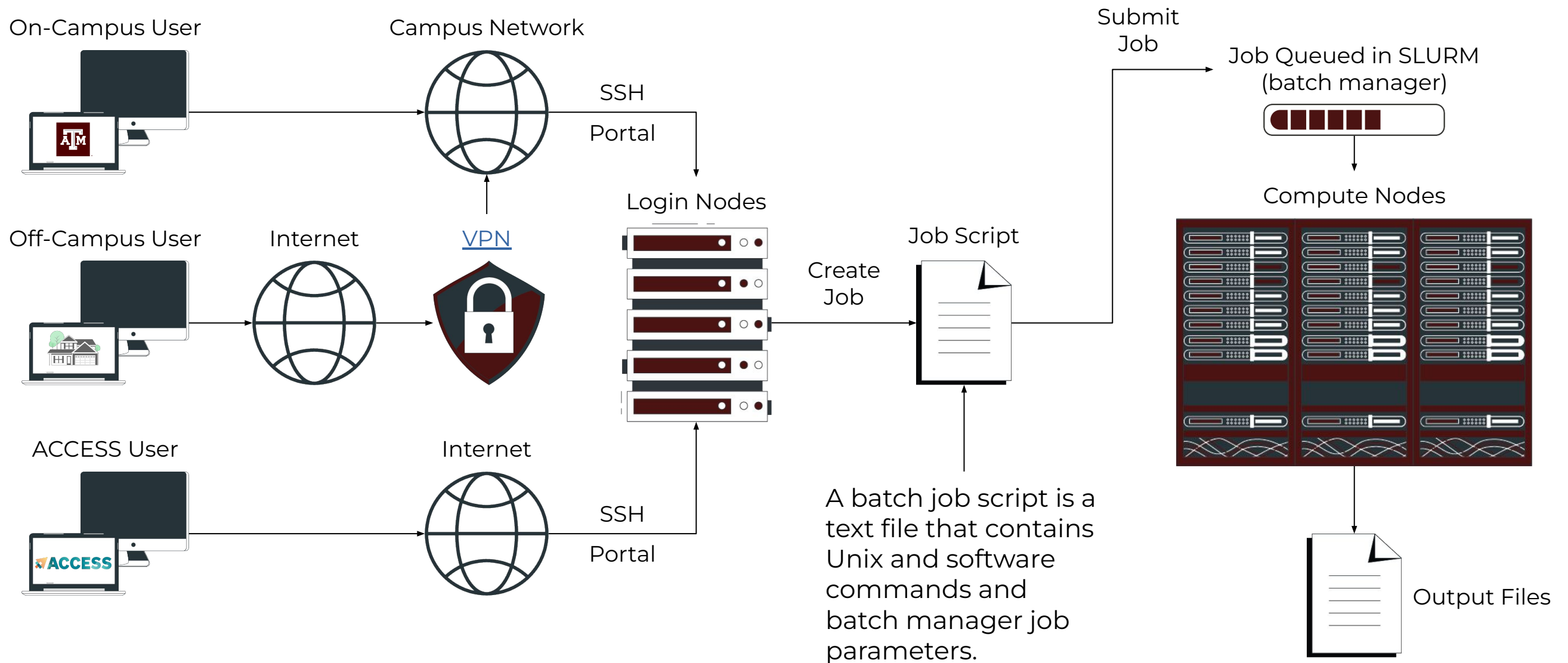
```
deactivate
```

-Close virtual environment

# Cluster Computing



# Batch Computing on HPRC Clusters: Overview



# Consumable Computing Resources

Computing resources are shared!

You won't be able to use an infinite amount!

- Resources external to jobs:
  - Service Unit (SU)
  - Software license/token
- Resources specified in a job file:
  - Processor cores
  - Memory
  - Wall time
  - GPU

# Software Licenses

To check the availability of licensed software on the clusters, use the command **license\_status**.

- List software the tool knows about:
- Help for this command:
  
- Find available license for "ansys":

```
license_status -l
```

```
license_status -h
```

```
license_status -s ansys
```

Exact availability varies by license policy.  
Contact us if you have questions.

```
License status for ANSYS:
```

License Name	# Issued	# In Use	# Available
aa_mcad	50	0	50
aa_r	50	32	18
aim_mp1	50	0	50
.....			

[https://hprc.tamu.edu/kb/Software/useful-tools/License\\_Checker/](https://hprc.tamu.edu/kb/Software/useful-tools/License_Checker/)



# Service Units

- Service Units (SUs) are part of our account management system (AMS).
- With a default allocation, you start with 5000 SUs
- SUs are charged as your jobs spend time on the compute nodes (we'll see how later).  
*(Work on login nodes is not charged, but you can't do big computing there!)*
- You can check your SU balance on both:
  - The command line
  - The HPRC Portal

[https://hprc.tamu.edu/kb/User-Guides/AMS/Service\\_Unit/](https://hprc.tamu.edu/kb/User-Guides/AMS/Service_Unit/)

<https://hprc.tamu.edu/kb/User-Guides/AMS/UI/>

# Check your (SU) Balance: Command Line

- List the SU Balance of your Account(s) with:

```
myproject
```

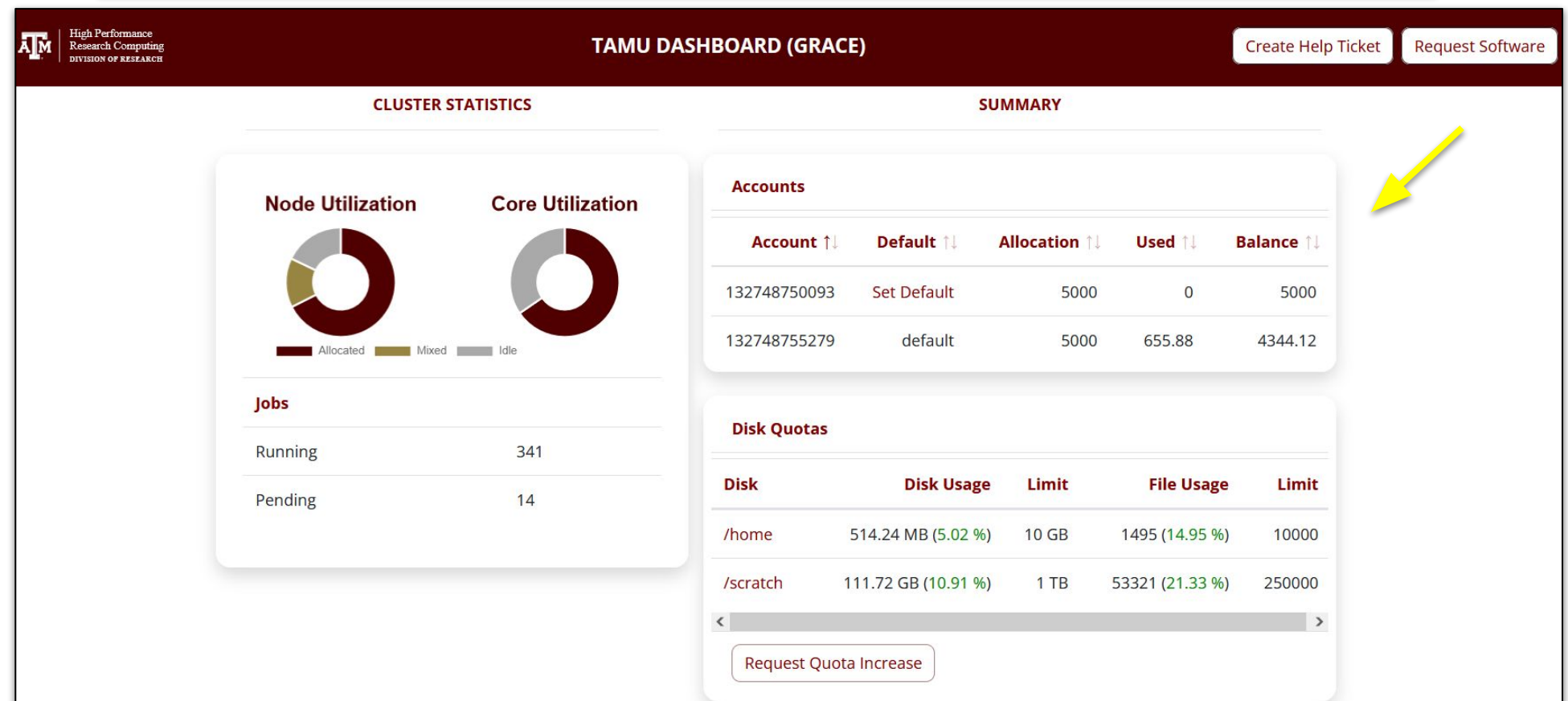
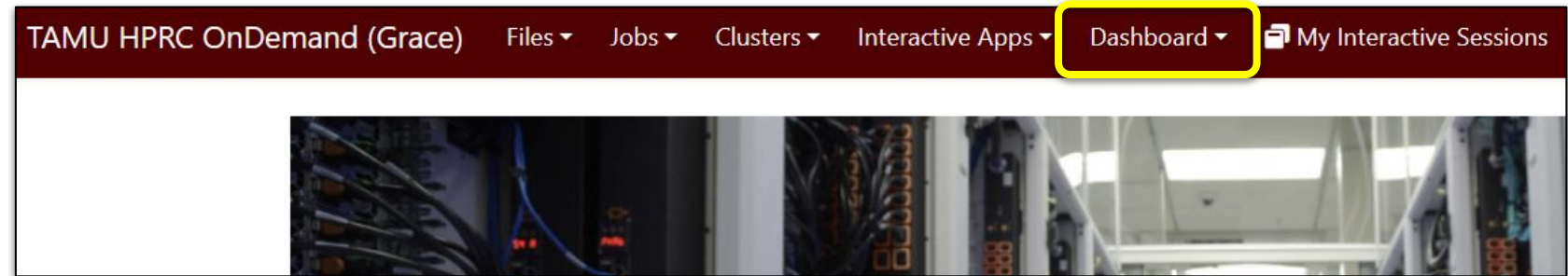
```
=====
List of YourNetID's Project Accounts
-----
| Account | FY | Default | Allocation | Used & Pending SUs | Balance | PI |
-----
| 1228000223136 | 2023 | N | 10000.00 | 0.00 | 10000.00 | Doe, John |
-----
| 1428000243716 | 2023 | Y | 5000.00 | -71.06 | 4928.94 | Doe, Jane |
-----
| 1258000247058 | 2023 | N | 5000.00 | -0.91 | 4999.09 | Doe, Jane |
-----
```

- Run `myproject -d <Account#>` to change default project account  
(replace <Account#> with your number!)
- Run `myproject -h` to see more options

# Check your (SU) Balance: Portal

SUs can also be checked in the Dashboard

(The same place we checked file quotas previously)



**TAMU DASHBOARD (GRACE)** [Create Help Ticket](#) [Request Software](#)

**CLUSTER STATISTICS**

**Node Utilization** **Core Utilization**

Allocated Mixed Idle

**Jobs**

Running	341
Pending	14

**SUMMARY**

**Accounts**

Account ↑↓	Default ↑↓	Allocation ↑↓	Used ↑↓	Balance ↑↓
132748750093	Set Default	5000	0	5000
132748755279	default	5000	655.88	4344.12

**Disk Quotas**

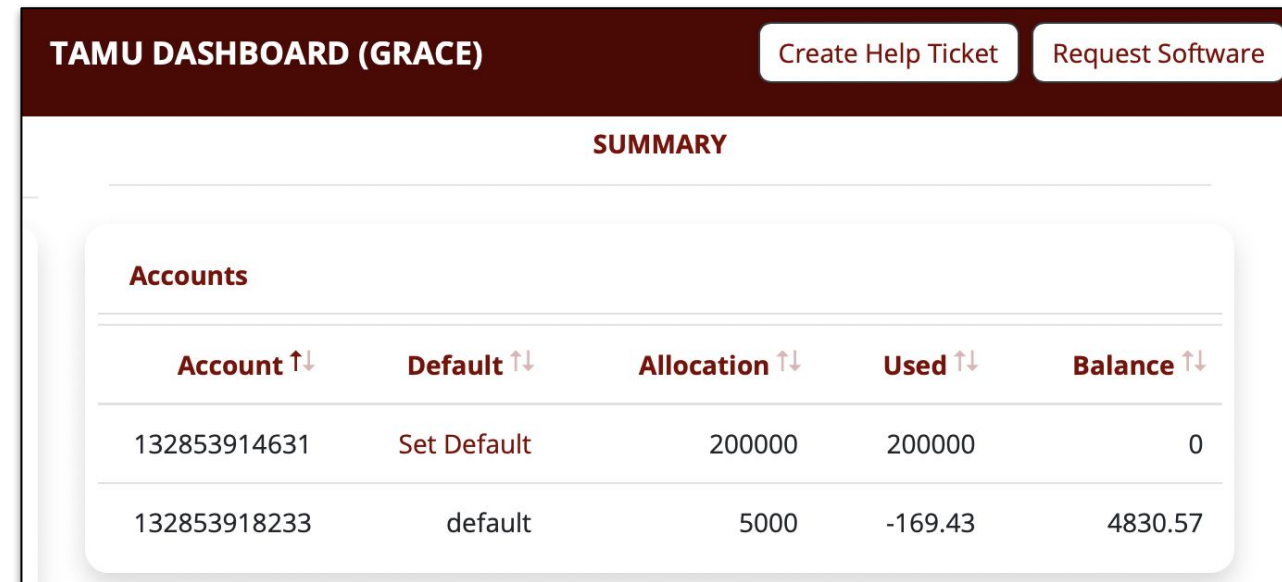
Disk	Disk Usage	Limit	File Usage	Limit
/home	514.24 MB (5.02 %)	10 GB	1495 (14.95 %)	10000
/scratch	111.72 GB (10.91 %)	1 TB	53321 (21.33 %)	250000

[Request Quota Increase](#)

# Hands-on Exercise

1. Check your SUs in both the command line and the Portal
2. Check that you have a default account set

myproject

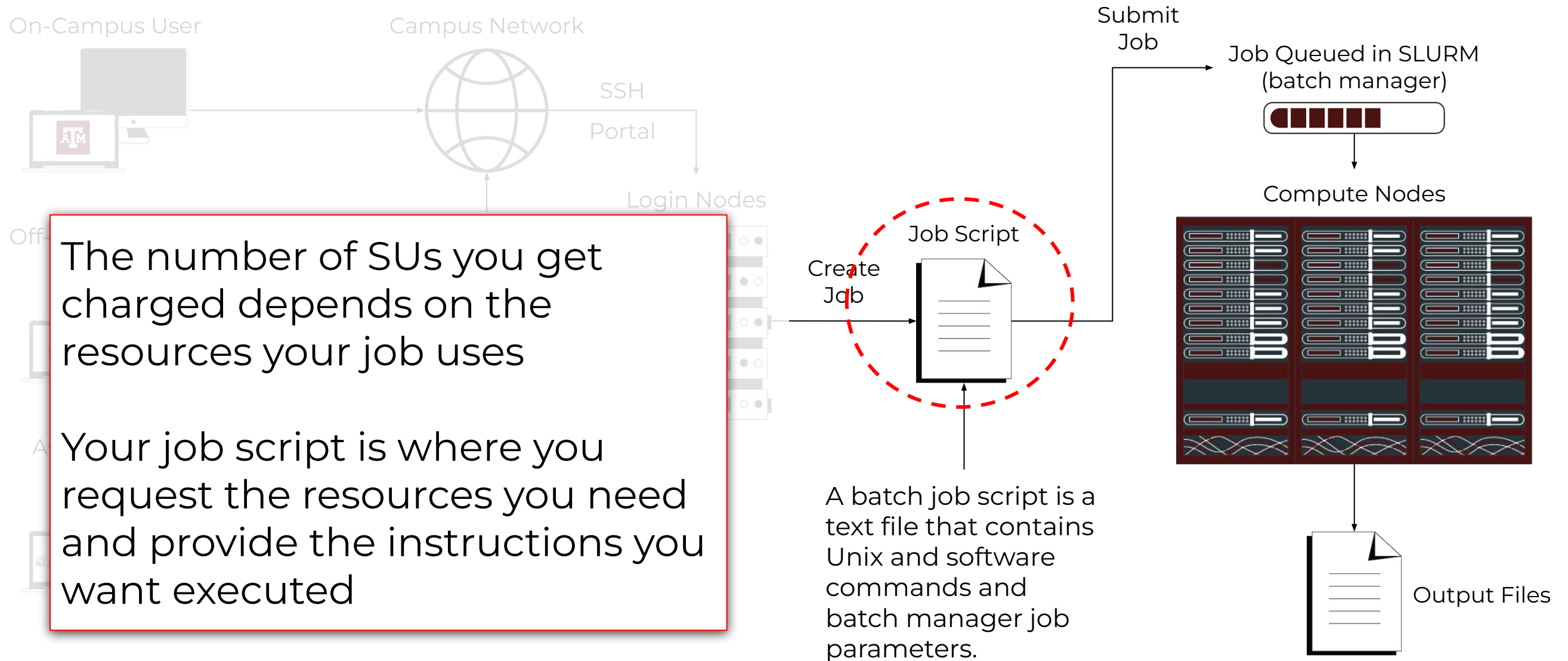


The screenshot shows the TAMU Dashboard (GRACE) interface. At the top, there are two buttons: "Create Help Ticket" and "Request Software". Below the header, the word "SUMMARY" is centered. A table titled "Accounts" is displayed, showing two rows of account information. The first row has an account number 132853914631, is set as the default, has an allocation of 200,000, and a balance of 0. The second row has an account number 132853918233, is the default, has an allocation of 5,000, and a balance of 4,830.57.

Account ↑↓	Default ↑↓	Allocation ↑↓	Used ↑↓	Balance ↑↓
132853914631	Set Default	200000	200000	0
132853918233	default	5000	-169.43	4830.57

We'll return to SUs once we've talked about SLURM and the resources your jobs use up.

# Slurm Job Scripts



# Sample Job Script Structure

```
#!/bin/bash

##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=hello_world
#SBATCH --time=00:15:00
#SBATCH --ntasks=2
#SBATCH --ntasks-per-node=2
#SBATCH --nodes=1
#SBATCH --mem=3G
#SBATCH --output=hello_world_log.%j

# load required module(s)
module purge
module load GCCcore/11.3.0
module load Python/3.10.4
python hello_world.py

# Job Environment variables
echo $SLURM_JOBID
echo $SLURM_SUBMIT_DIR
echo $TMPDIR
echo $SCRATCH
```

← This is a single-line comment and not run as part of the script.

These parameters describe your job to the job scheduler. The lines starting with #SBATCH are NOT comments! See the [Knowledge Base](#) for more info.

Whatever commands or scripts you want to run. Here, we set up the modules we need for our environment, run a python program, and print out some environment variables.

# Submit a Job and Check Job Status

Submit job

```
sbatch example01.job
```

```
Submitted batch job 6853258  
(from job_submit) your job is charged as below  
Project Account: 122792016265  
Account Balance: 1687.066160  
Requested SUs: 3
```

Check status

```
squeue -u $USER
```

or

```
squeue --me
```

JOBID	NAME	USER	PARTITION	NODES	CPUS	STATE	TIME	TIME_LEFT	START_TIME	REASON	NODELIST
6853258	jobname	someuser	xlong	2	96	RUNNING	3-07:36:50	16:23:10	2023-01-23T17:27:3	None	c[180,202]
6853257	jobname	someuser	xlong	2	96	RUNNING	3-07:36:56	16:23:04	2023-01-23T17:27:2	None	c[523-524]

# Hands-on Exercise

Submit a simple job file:

1. Navigate to `/scratch/training/Intro-to-Grace`, either on the command line or in the Portal's file browser
2. Copy the files `hello_world.slrn` and `hello_world.py` to your home directory
3. Return to your home directory yourself
4. Submit the job file with: `sbatch hello_world.slrn`
5. Check the job's status with: `squeue -u $USER` or `squeue --me`
6. Check the output file (will be named like `hello_world_log.#`)



# Job Environment Variables

Each job has access to several self-referential variables:

- `$SLURM_JOBID` = job id
- `$SLURM_SUBMIT_DIR` = directory where job was submitted from
- `$TMPDIR` = `/work/job.$SLURM_JOBID`

You can also still use non-Slurm variables like:

- `$SCRATCH` = `/scratch/user/NetID`

<https://hprc.tamu.edu/kb/Helpful-Pages/Batch-Translation/#environment-variables>

# Important Batch Job Parameters

Slurm	Comment
<code>#SBATCH --time=HH:MM:SS</code>	Specifies the upper time limit for the job.
<code>#SBATCH --ntasks=x</code>	Total number of tasks (cores) for the job.
<code>#SBATCH --ntasks-per-node=xx</code>	Specifies the maximum number of tasks (cores) to allocate per node
<code>#SBATCH --mem=xxxxM</code> or <code>#SBATCH --mem=xG</code>	Sets the maximum amount of memory per <i>node</i> . Can use M for MB or G for GB
<code>#SBATCH --nodes=x</code>	Specifies the number of nodes to use

These usually all go in your job script file

<https://hprc.tamu.edu/kb/Helpful-Pages/Batch-Translation/#job-specifications>

# (Job Template)

```
#!/bin/bash
##ENVIRONMENT SETTINGS; CHANGE WITH CAUTION
#SBATCH --export=NONE          #Do not propagate environment
#SBATCH --get-user-env=L      #Replicate login environment

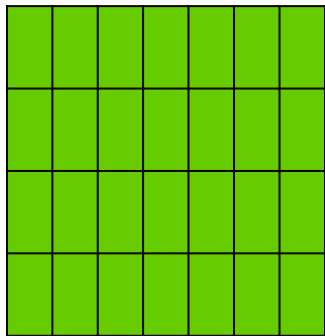
##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=JobExample1 #Set the job name to "JobExample1"
#SBATCH --time=01:30:00       #Set the wall clock limit to 1hr and 30min
#SBATCH --ntasks=1           #Request 1 task
#SBATCH --ntasks-per-node=1  #Request 1 task/core per node
#SBATCH --mem=2560M          #Request 2560MB (2.5GB) per node
#SBATCH --output=Example1Out.%j #Send stdout/err to "Example1Out.[jobID]"

#First Executable Line
```

<https://hprc.tamu.edu/kb/Quick-Start/Grace/#running-your-program-preparing-a-job-file>

# Mapping Jobs to Cores per Node on Terra

A.

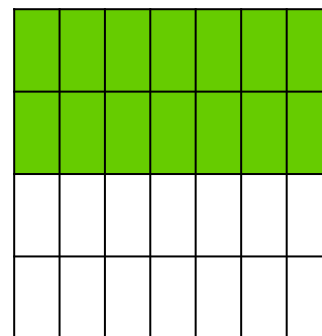
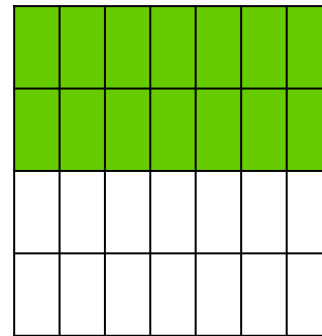


28 cores on  
1 compute node

#SBATCH --ntasks 28  
#SBATCH --tasks-per-node=28

Preferred Mapping  
(if applicable)

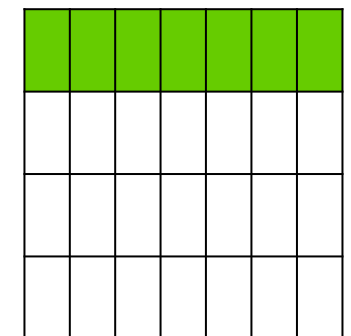
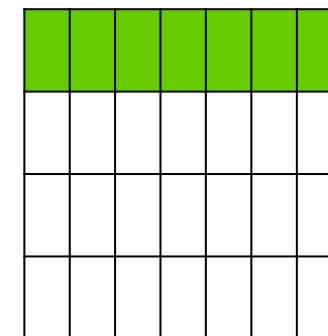
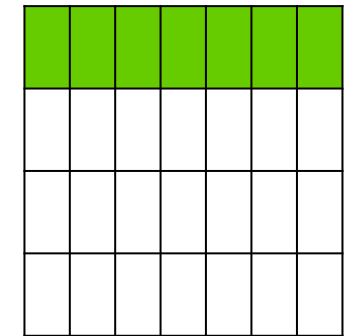
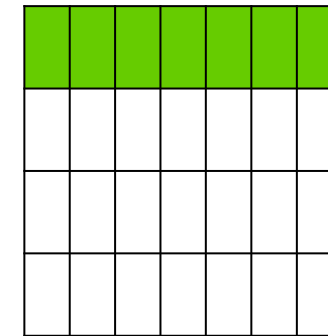
B.



28 cores on  
2 compute nodes

#SBATCH --ntasks 28  
#SBATCH --tasks-per-node=14

C.

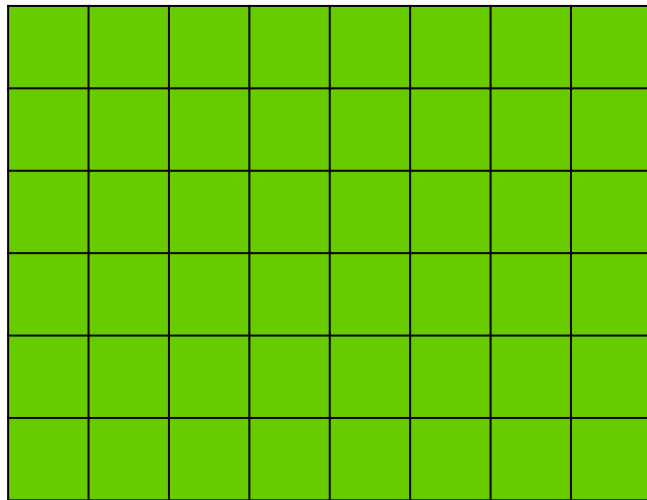


28 cores on  
4 compute nodes

#SBATCH --ntasks 28  
#SBATCH --tasks-per-node=7

# Mapping Jobs to Cores per Node on Grace

A.

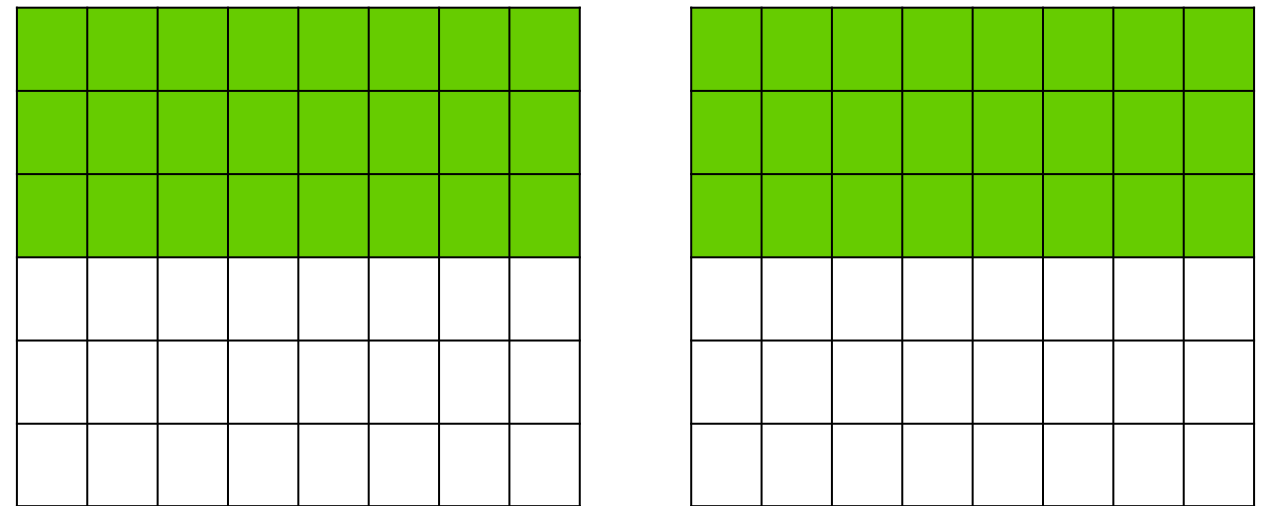


48 cores on  
1 compute node

```
#SBATCH --ntasks=48  
#SBATCH --tasks-per-node=48
```

Preferred Mapping  
(if applicable)

B.



48 cores on  
2 compute nodes

```
#SBATCH --ntasks=48  
#SBATCH --tasks-per-node=24
```

# Slurm Pop Quiz

```
#SBATCH --job-name=stacks_s2
#SBATCH --ntasks=80
#SBATCH --ntasks-per-node=20
#SBATCH --mem=40G
#SBATCH --time=48:00:00
#SBATCH --output stdout.%J
#SBATCH --error stderr.%J
```

How many nodes is this job requesting?

- A. 1600
- B. 80
- C. 20
- D. 4

# Job Memory Requests on Terra

- Specify memory request based on memory per node:  
#SBATCH --mem=xxxxM # memory per node in MB  
or  
#SBATCH --mem=xG # memory per node in GB
- On 64GB nodes, usable memory is at most 56 GB. The per-process memory limit should not exceed 2000 MB for a 28-core job.
- On 128GB nodes, usable memory is at most 112 GB. The per-process memory limit should not exceed 4000 MB for a 28-core job.

# Job Memory Requests on Grace

- Specify memory request based on memory per node:  
#SBATCH --mem=xxxxM # memory per node in MB  
or  
#SBATCH --mem=xG # memory per node in GB
- On 384GB nodes, usable memory is at most 360 GB.  
The per-process memory limit should not exceed ~7500 MB for a 48-core job.
- On 3TB nodes, usable memory is at most 2900 GB.  
The per-process memory limit should not exceed 37120 MB for a 48-core job.



# Slurm: Examples of SUs charged based on Job Cores, Time and Memory Requested

An SU is equivalent to one core hour or a proportional amount of memory on the node for one hour.

On **Grace**: a typical node has 48 cores and 360 GB usable for jobs. (7.5 GB per core)

Number of Cores	GB of memory per core	Total Memory (GB)	Hours	SUs charged
1	7.5	7.5	1	1
24	1	24	1	24
1	360	360	1	48
48	7.5	360	1	48

[https://hprc.tamu.edu/kb/User-Guides/AMS/Service\\_Unit/](https://hprc.tamu.edu/kb/User-Guides/AMS/Service_Unit/)

# Slurm: SU Charges for GPU jobs

GPUs will use more SUs per hour than CPUs

<b>Effective GPU</b>	<b>SU charge per one hour (wall_time)</b>
A100	72
RTX 6000	48
T4	24

# Job Submission and Tracking

Slurm commands	Description
<code>sbatch jobfile1</code>	Submit jobfile1 to batch system
<code>squeue [-u user_name] [-j job_id]</code>	List jobs
<code>scancel job_id</code>	Kill a job
<code>sacct -X -j job_id</code>	Show information for a job (can be when job is running or recently finished)
<code>sacct -X -S YYYY-HH-MM</code>	Show information for all of your jobs since YYYY-HH-MM
<code>pestat -u \$USER</code>	Show resource usage for a running job
<code>seff job_id</code>	Check CPU/memory efficiency for a job

<https://hprc.tamu.edu/kb/Helpful-Pages/Batch-Translation/#job-specifications>

# Batch Queues

- Job submissions are auto-assigned to batch *queues* (also called *partitions*) based on the resources requested:
  - number of cores/nodes and walltime limit
  - specific resources requested
- Some jobs can be directly submitted to a queue:
  - E.g. if gpu nodes are needed, use the gpu partition/queue:  
`#SBATCH --partition=gpu`
- Batch queue policies are used to manage the workload and may be adjusted periodically.

<https://hprc.tamu.edu/kb/User-Guides/Common/BatchProcessing/#batch-queues>

# Checking Resources for Jobs

`sinfo`

`pestat`

`gpuavail`

`maxconfig`

There are several tools available to check what you can use in your jobs...

# Checking Resources for Jobs: Queues

**sinfo**

pestat

gpuavail

maxconfig

```
username@login1:~$ sinfo
PARTITION      AVAIL TIMELIMIT      JOB SIZE      NODES (A/I/O/T)      CPUS (A/I/O/T)
short*         up     2:00:00           1-32          711/0/89/800         33689/439/4272/38400
medium         up     1-00:00:00        1-128         711/0/89/800         33689/439/4272/38400
long           up     7-00:00:00        1-64          711/0/89/800         33689/439/4272/38400
xlong          up     21-00:00:00       1-32          711/0/89/800         33689/439/4272/38400
vnc            up     12:00:00          1-32          100/5/12/117         985/4055/576/5616
gpu            up     4-00:00:00        1-32          100/5/12/117         985/4055/576/5616
bigmem         up     2-00:00:00        1-4           7/0/1/8             560/0/80/640
staff          up     infinite          1-infinite    817/14/101/932       34689/5199/4848/4473
special        up     7-00:00:00        1-infinite    811/5/101/917        34674/4494/4848/4401
gpu-a40        up     4-00:00:00        1-15          6/9/0/15            15/705/0/720
```

Shows job queue status

For the NODES and CPUS columns:  
A = Active (in use by running jobs)  
I = Idle (available for jobs)  
O = Offline (unavailable for jobs)  
T = Total

# Checking Resources for Jobs: Nodes

sinfo

pestat

gpuavail

maxconfig

```
username@login1:~$ pestat -p gpu -G
Print only nodes in partition gpu
GPU GRES (Generic Resource) is printed after each jobid
Hostname      Partition      Node Num_CPU  CPUload  Memsize  Freemem  GRES/node  Joblist
              Partition      State Use/Tot  (15min)  (MB)     (MB)     JobID(JobArrayID) User GRES/job ...
g001          gpu            mix  2  48    1.94    368640  271338  gpu:a100:2 <jobID> <username> <gpus>
g002          gpu            drain* 0  48    0.00*   368640  0       gpu:a100:2
g003          gpu            drain* 0  48    0.01    368640  370427  gpu:a100:2
.....
```

Notable options:

- `-p <partition name>` show only a specific partition
- `-u $USER` show only specified user's jobs
- `-G` show "generic resources" (e.g. the gpus used)

# Checking Resources for Jobs: GPUs

sinfo

pestat

gpuavail

maxconfig

See what what resources in the GPU nodes are available:

```
username@login1:~$ gpuavail

CONFIGURATION
NODE          NODE
TYPE          COUNT
-----
gpu:a100:2    100
gpu:a40:3     15
gpu:rtx:2     9
gpu:t4:4      8

AVAILABILITY
NODE  GPU  GPU  GPUs  CPUs  GB MEM
NAME  TYPE COUNT AVAIL AVAIL AVAIL
-----
g106  a40   3    1    40   260
t001  t4    4    1    45   345
t006  t4    4    2    20   310
```



# Checking Resources for Jobs: SUs

sinfo

pestat

gpuavail

maxconfig

If you ask for too many resources, your job won't run. You can check beforehand:

```
jwinchell@login1:~$ maxconfig
```

```
Grace partitions:  short medium long xlong vnc gpu bigmem special gpu-a40
Grace GPUs in gpu partition:  a100:2  a40:3  rtx:2  t4:4
```

```
Showing max parameters (cores, mem, time) for partition long
```

```
CPU-billing * hours * nodes =  SUs
          48 *   168 *    1 = 8,064
```

```
#!/bin/bash
```

```
#SBATCH --job-name=my_job
```

```
#SBATCH --time=7-00:00:00
```

```
#SBATCH --nodes=1
```

```
#SBATCH --ntasks-per-node=1
```

```
#SBATCH --cpus-per-task=48
```

```
#SBATCH --mem=360G
```

```
#SBATCH --output=stdout.%x.%j
```

```
#SBATCH --error=stderr.%x.%j
```

Check specific partitions with:

```
maxconfig -p <partitionName>
```

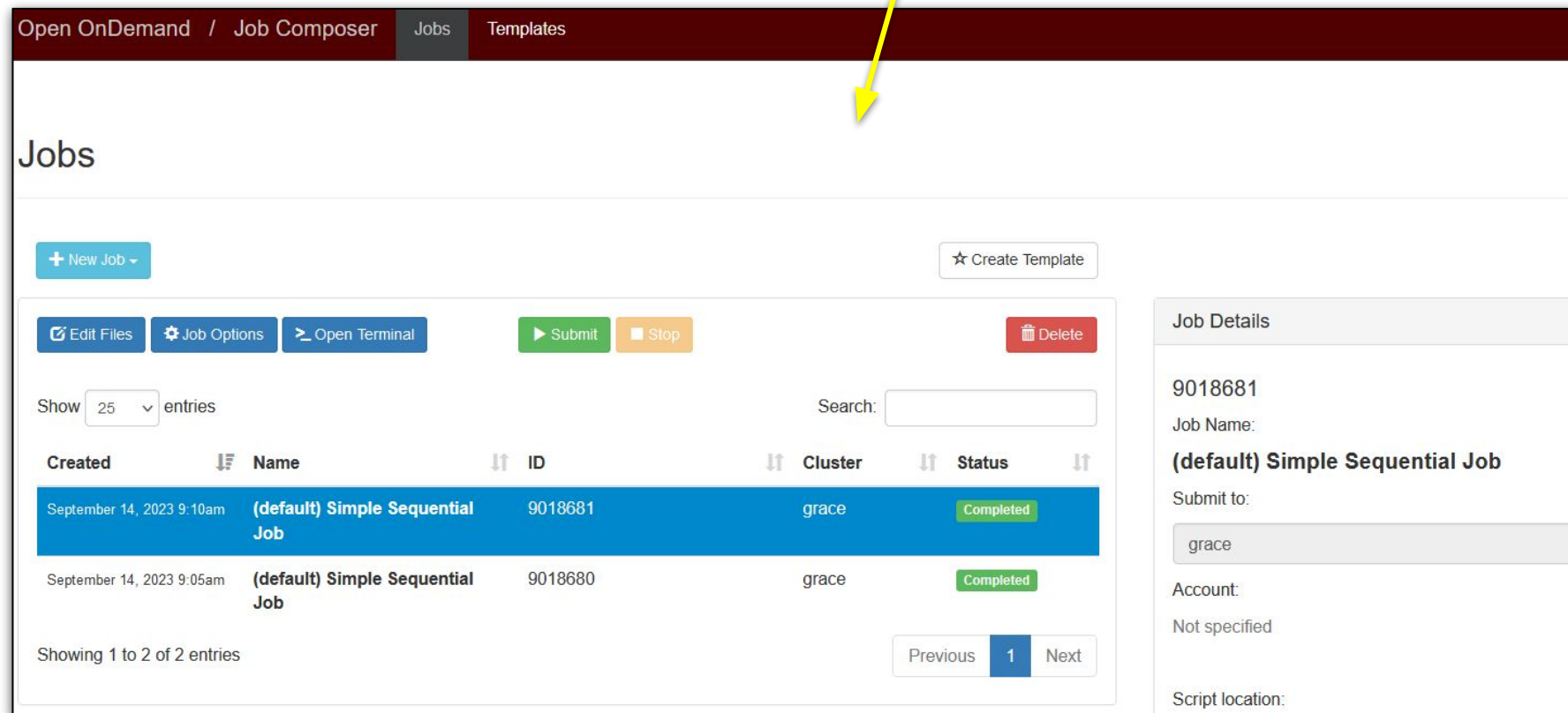
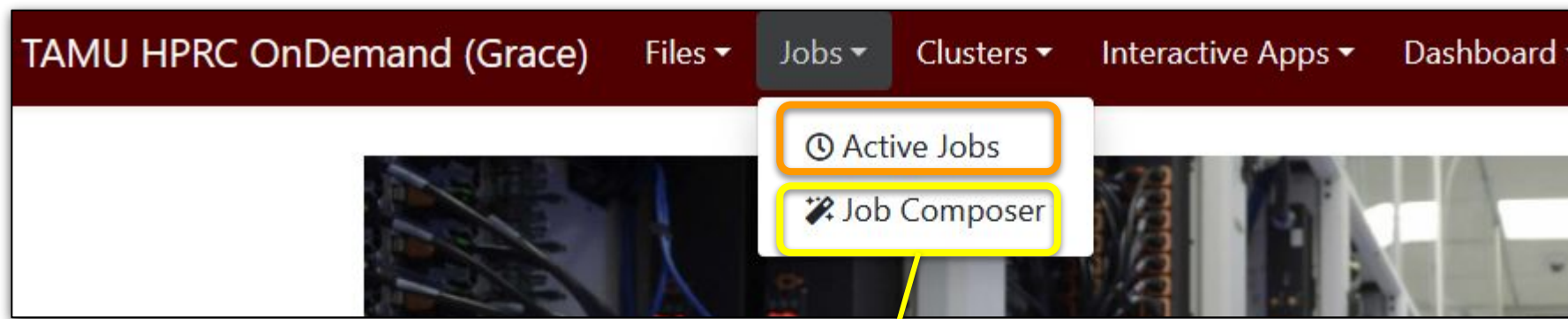
Estimate the SUs a job will require with:

```
maxconfig -f <jobScriptName>
```

# Portal Job Composer

Some job management can also be done from the Portal

- “Active Jobs” is like queue
- “Job Composer” lets you:
  - Create job scripts
  - Save job templates
  - Monitor your own jobs



Jobs

+ New Job - Create Template

Edit Files Job Options Open Terminal Submit Stop Delete

Show 25 entries Search:

Created	Name	ID	Cluster	Status
September 14, 2023 9:10am	(default) Simple Sequential Job	9018681	grace	Completed
September 14, 2023 9:05am	(default) Simple Sequential Job	9018680	grace	Completed

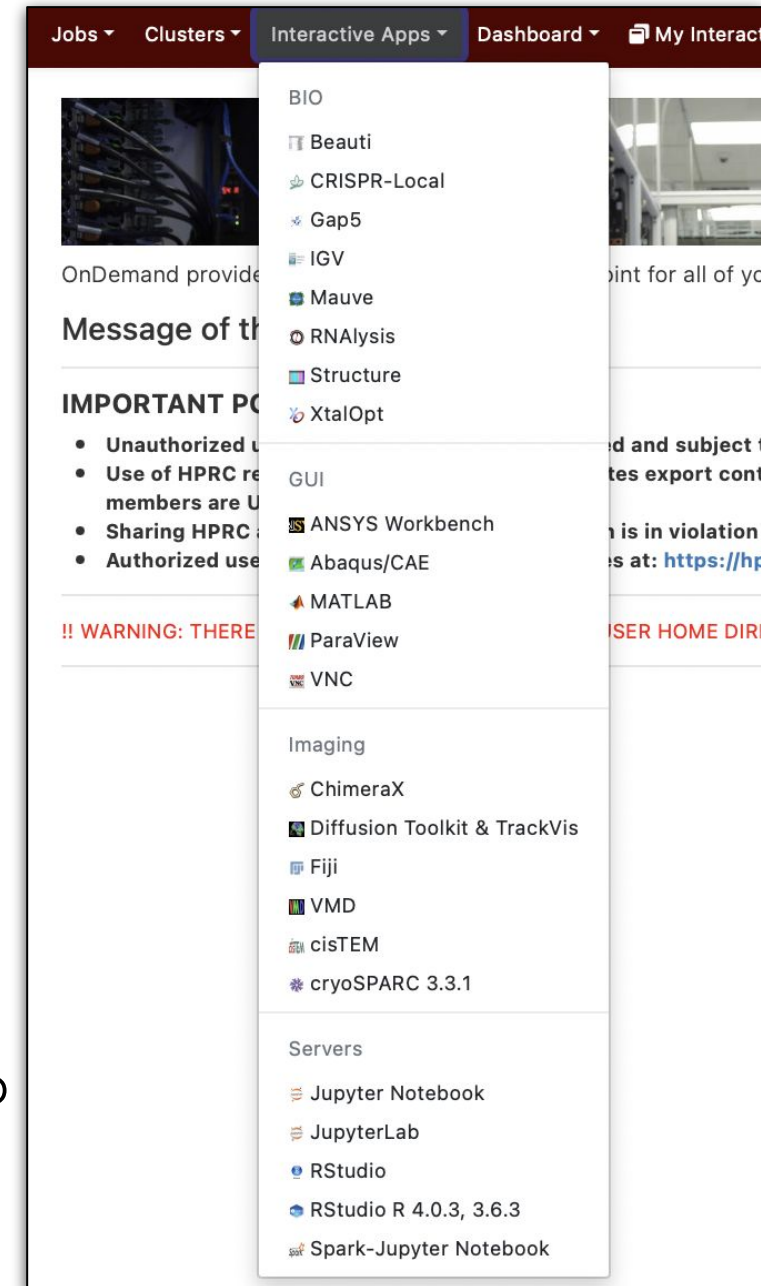
Showing 1 to 2 of 2 entries Previous 1 Next

Job Details

9018681  
Job Name:  
**(default) Simple Sequential Job**  
Submit to:  
grace  
Account:  
Not specified  
Script location:

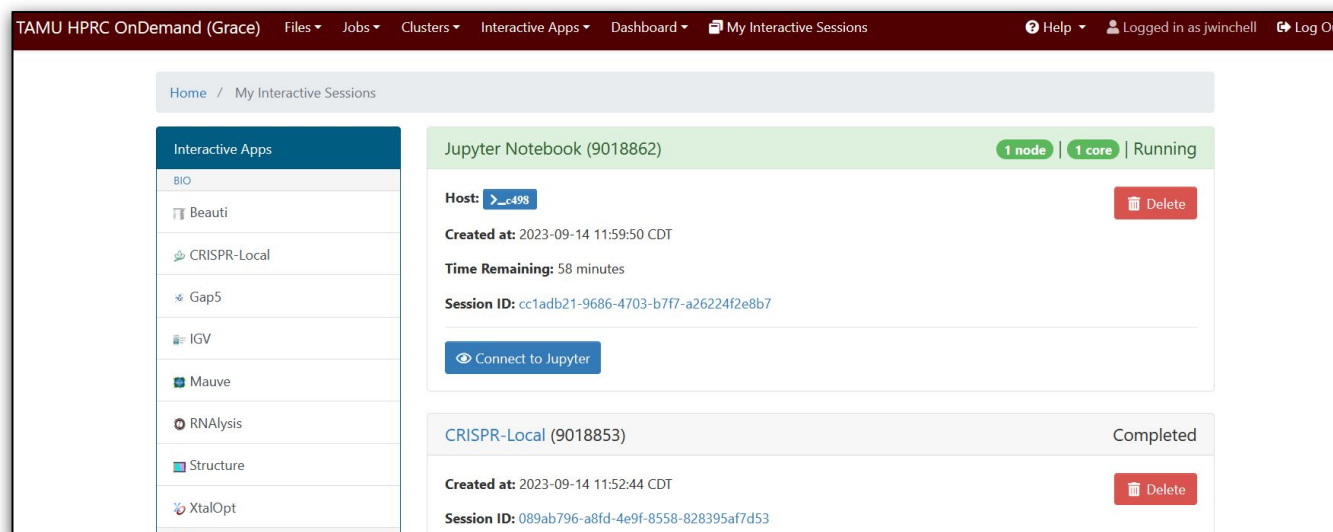
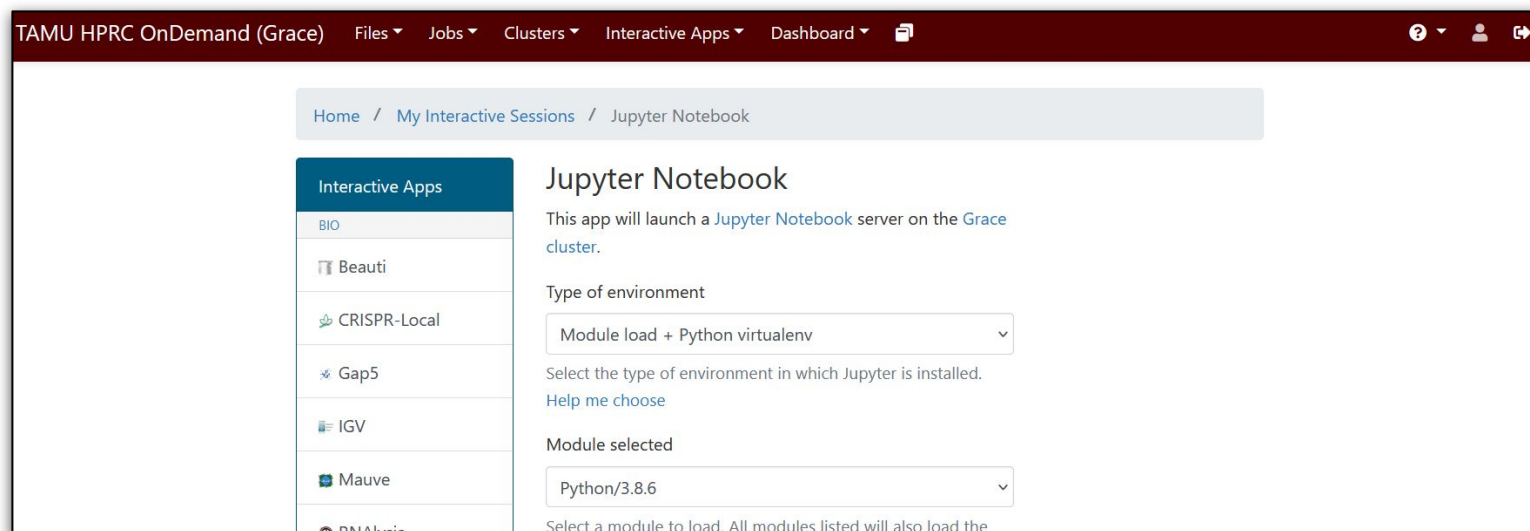
# Other Type of Jobs

- Visualization:
  - [portal.hprc.tamu.edu](http://portal.hprc.tamu.edu)
  - Interactive Apps > choose application
- Large number of concurrent single-core jobs
  - Check out *tamulauncher*
    - Useful for running many single core commands concurrently across multiple nodes within a job
    - Can be used with serial or multi-threaded programs
    - Can only be used in batch jobs
    - If a tamulauncher job gets killed, you can resubmit the same job to complete the unfinished commands in the input file
    - <https://hprc.tamu.edu/kb/Software/tamulauncher/>



# Interactive Apps

1. Select an app from the dropdown menu
2. Specify environment and resources
3. Hit “Launch” at the bottom
4. On the “My Interactive Sessions” screen:
  - a. wait for box to turn green
  - b. connect to app
  - c. quit from the app when you’re done
  - d. box should turn grey



# Need Help?

First check the FAQ <https://hprc.tamu.edu/kb/FAQ/Accounts/>

- Grace User Guide <https://hprc.tamu.edu/kb/User-Guides/Grace/>
- Email your questions to [help@hprc.tamu.edu](mailto:help@hprc.tamu.edu)

Help us help you -- we need more info:

- Which Cluster
- Username
- Job id(s) if any
- Location of your jobfile, input/output files
- Application used if any
- Module(s) loaded if any
- Error messages
- Steps you have taken, so we can reproduce the problem

Remember you can  
start help requests  
from the Dashboard!



High Performance  
Research Computing  
DIVISION OF RESEARCH

Thank you

*Questions?*

# Batch Job Examples (on Grace)



# Slurm Job File (Serial Example)

```
#!/bin/bash

##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=JobExample1           #Set the job name to "JobExample1"
#SBATCH --time=01:30:00                 #Set the wall clock limit to 1hr and 30min
#SBATCH --ntasks=1                       #Request 1 task
#SBATCH --mem=6M                         #Request 6GB per node
#SBATCH --output=Example1Out.%j         #Send stdout/err to "Example1Out.[jobID]"

##OPTIONAL JOB SPECIFICATIONS
##CHANGE ACCOUNT NUMBER AND EMAIL ADDRESS BEFORE USING
##SBATCH --account=123456                #Set billing account to 123456
##SBATCH --mail-type=ALL                 #Send email on all job events
##SBATCH --mail-user=email_address       #Send all emails to email_address

# this intel toolchain is just an example.  recommended toolchain is TBD
module purge
module load intel/2022a

# run your program
./helloworld.omp.C.exe
```

SUs = 1

(rounded)

# Slurm Job File (multi core, single node)

```
#!/bin/bash

##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=JobExample2           # Set the job name to "JobExample2"
#SBATCH --time=6:30:00                   # Set the wall clock limit to 6hr and 30min
#SBATCH --nodes=1                         # Request 1 node
#SBATCH --ntasks-per-node=8              # Request 8 tasks (cores) per node
#SBATCH --mem=8G                          # Request 8GB per node
#SBATCH --output=Example2Out.%j          # Send stdout/err to "Example2Out.[jobID]"

##OPTIONAL JOB SPECIFICATIONS
##CHANGE ACCOUNT NUMBER AND EMAIL ADDRESS BEFORE USING
##SBATCH --account=123456                 # Set billing account to 123456 #find your account with "myproject"
##SBATCH --mail-type=ALL                  # Send email on all job events
##SBATCH --mail-user=email_address        # Send all emails to email_address

# load required module(s)
module purge
module load intel/2022a

# run your program
mpirun ./helloworld.mpi.C.exe
```

SUs = 52

# Slurm Job File (multi core, multi node)

```
#!/bin/bash

##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=JobExample3           # Set the job name to "JobExample3"
#SBATCH --time=1-12:00:00                # Set the wall clock limit to 1 Day and 12hr
#SBATCH --ntasks=8                       # Request 8 tasks (cores)
#SBATCH --ntasks-per-node=2              # Request 2 tasks(cores) per node
#SBATCH --mem=5M                         # Request 5GB per node
#SBATCH --output=Example3Out.%j         # Send stdout and stderr to "stdout.[jobID]"

##OPTIONAL JOB SPECIFICATIONS
##CHANGE ACCOUNT NUMBER AND EMAIL ADDRESS BEFORE USING
##SBATCH --account=123456                # Set billing account to 123456 #find your account with "myproject"
##SBATCH --mail-type=ALL                 # Send email on all job events
##SBATCH --mail-user=email_address       # Send all emails to email_address

# this intel toolchain is just an example.  recommended toolchain is TBD
module purge
module load intel/2022a

# run program with MPI
mpirun ./helloworld.mpi.C.exe
```

SUs = 288

# Slurm Job File (serial GPU)

```
#!/bin/bash

##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=JobExample4           # Set the job name to "JobExample4"
#SBATCH --time=01:00:00                  # Set the wall clock limit to 1hr
#SBATCH --ntasks=1                       # Request 1 task (core)
#SBATCH --mem=5G                          # Request 5GB per node
#SBATCH --output=Example4Out.%j          # Send stdout and stderr to "Example4Out.[jobID]"
#SBATCH --gres=gpu:a100:1                # Request 1 A100 GPUs
#SBATCH --partition=gpu                   # Request the GPU partition/queue

##OPTIONAL JOB SPECIFICATIONS
##CHANGE ACCOUNT NUMBER AND EMAIL ADDRESS BEFORE USING
##SBATCH --account=123456                # Set billing account to 123456 #find your account with "myproject"
##SBATCH --mail-type=ALL                  # Send email on all job events
##SBATCH --mail-user=email_address        # Send all emails to email_address

# load required module(s)
module purge
module load CUDA/11.8.0

# run your program
my_cuda_enabled_program
```

SUs = 73

# Slurm Job File (multi GPU)

```
#!/bin/bash

##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=JobExample5          # Set the job name to "JobExample4"
#SBATCH --time=01:00:00                 # Set the wall clock limit to 1hr
#SBATCH --ntasks=2                      # Request 1 task (core)
#SBATCH --nodes=1                       # Request 250GB per node
#SBATCH --mem=250G                      # Send stdout and stderr to "Example4Out.[jobID]"
#SBATCH --output=Example4Out.%j        # Request 2 A100 GPUs
#SBATCH --gres=gpu:a100:2              # Request the GPU partition/queue

##OPTIONAL JOB SPECIFICATIONS
##CHANGE ACCOUNT NUMBER AND EMAIL ADDRESS BEFORE USING
##SBATCH --account=123456               # Set billing account to 123456 #find your account with "myproject"
##SBATCH --mail-type=ALL                # Send email on all job events
##SBATCH --mail-user=email_address      # Send all emails to email_address

# load required module(s)
module purge
module load CUDA/11.8.0

# run your program
my_cuda_enabled_program
```

SUs = 179