



# Alpine Job Submission



Research Computing  
UNIVERSITY OF COLORADO **BOULDER**

**Be Boulder.**

# HPC Job Submission

*Instructor:* Trevor Hall

*RC Homepage:* <https://www.colorado.edu/rc/>

*RC Docs:* <https://curc.readthedocs.io/en/latest/>

*RC Helpdesk:* [rc-help@colorado.edu](mailto:rc-help@colorado.edu)

**Course Materials:**

[https://github.com/ResearchComputing/Summer\\_Camp\\_2023](https://github.com/ResearchComputing/Summer_Camp_2023)

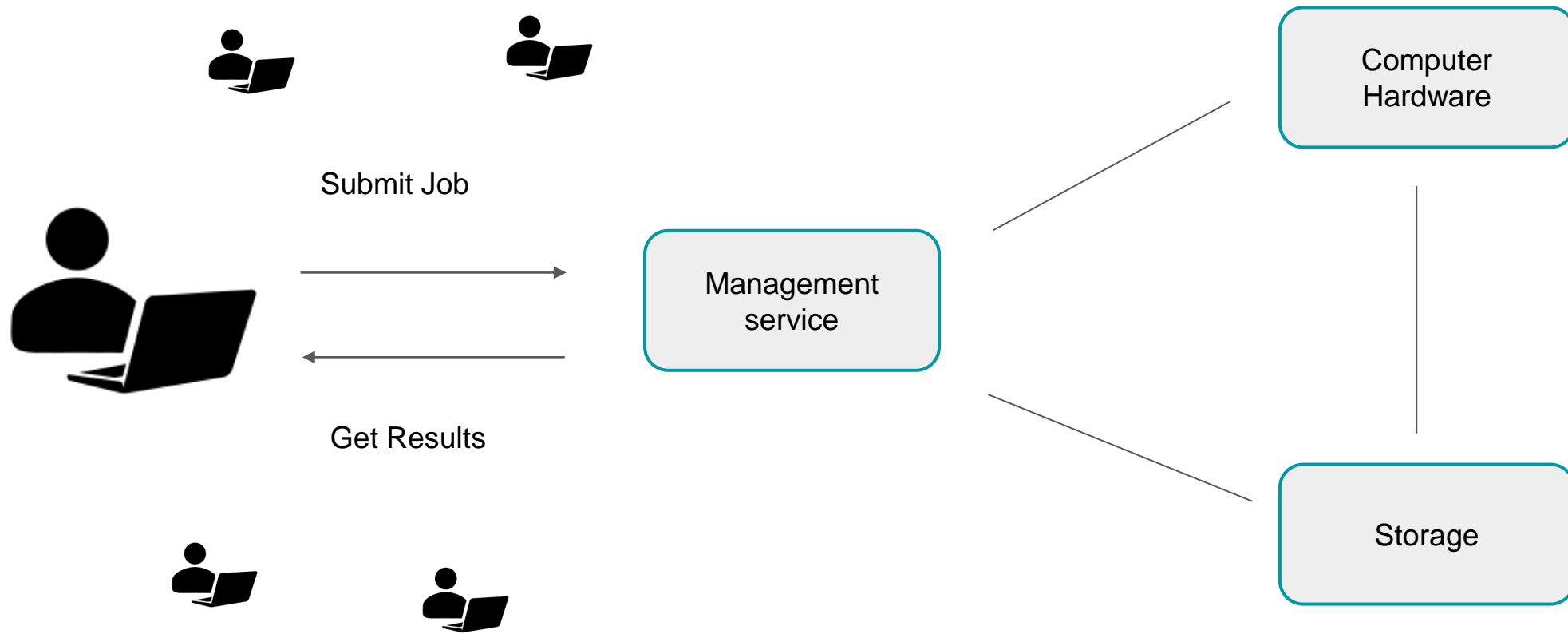
**Survey:** <http://tinyurl.com/curc-survey18>

*Adapted from presentations by RC members Andrew Monaghan, Aaron Holt, John Blaas, and Mea Trehan: [1](#), [2](#), [3](#), [4](#).*

# Outline

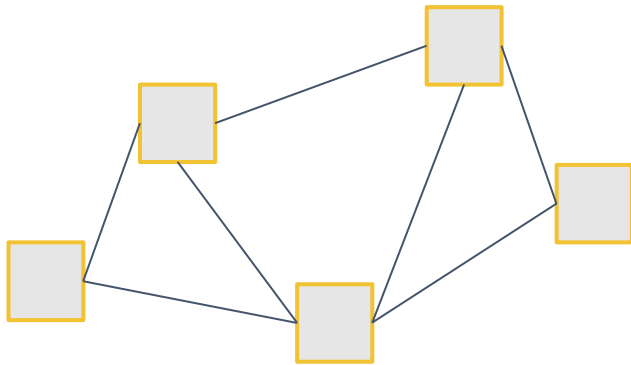
- General Information
  - Alpine resources
- Examples of submitting jobs to the supercomputer!
  - Traditional job submission (terminal)
    - Simple batch jobs: hello world, running programs
    - GPU Jobs
    - Advanced batch jobs: mpi, serial-parallel
    - Interactive jobs

# HPC - High Performance Computing



# HPC Cluster: Alpine

**Alpine**

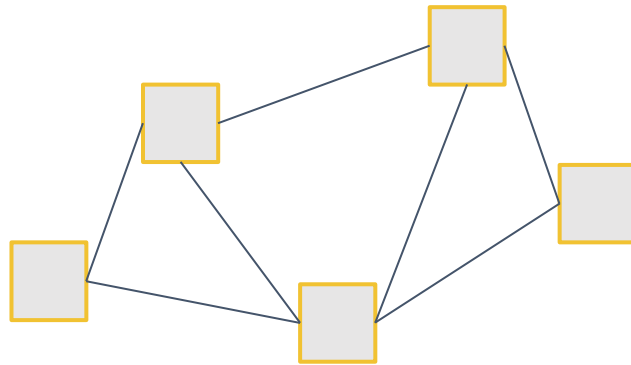


- Alpine is the 3rd-generation HPC cluster at CURC, following:
  - Janus
  - RMACC Summit
- Alpine is a heterogeneous cluster with hardware currently provided by CU Boulder, CSU, and Anschutz Medical Campus
- Access is available to CU Boulder, CSU, AMC, and RMACC users



# HPC Cluster: Alpine

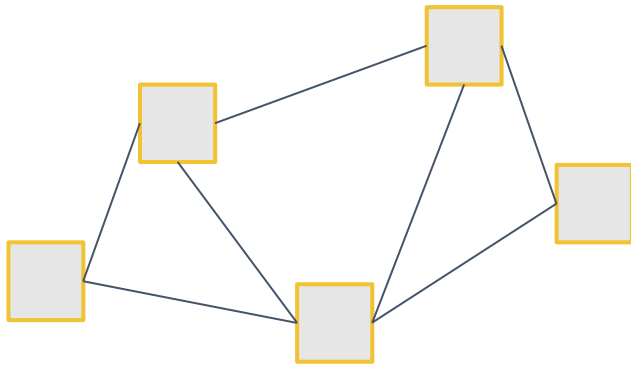
Alpine



- Hardware on Alpine will continue to be purchased and released in stages:
- Alpine (stage 3):
  - 184 General CPU Nodes
    - *AMD Milan, 64 Core, 3.74G RAM/Core*
  - 8 NVIDIA GPU Nodes
    - *3x NVIDIA A100 (atop General CPU node)*
  - 8 AMD GPU Nodes
    - *3x AMD MI100 (atop General CPU node)*
  - 12 AMD High-Memory Nodes
    - *AMD Milan, 48 Core, 21.5G RAM/Core*
  - Additional Hardware contributed by CSU, AMC
    - *Nodes which boost priority for CSU/AMC users*

# HPC Cluster: Alpine

**Alpine**



- **Interconnect**

- **CPU nodes:** HDR-100 InfiniBand (200Gb inter-node fabric)
- **GPU nodes:** 2x25 Gb Ethernet +RoCE
  - nvlink compatibility in progress
- **Scratch Storage:** 25Gb Ethernet +RoCE

- **Operating System**

- RedHat Enterprise Linux version 8 operating system

# Submitting Jobs via Terminal



# RC Access: Logging in

- To login to an RC login node:

```
$ ssh <username>@login.rc.colorado.edu
```

Supply your IdentiKey password and your Duo app will alert you to confirm the login

\*CU and CSU exclusive

---

# RC Access: Logging in

CURC Open OnDemand is a browser based, integrated, single access point for all of your HPC resources at CU Research Computing.

- CU Boulder: Visit <https://ondemand.rc.colorado.edu>.
- Other RMACC Institutions: Visit <https://ondemand-rmacc.rc.colorado.edu/>

# Working on RC Resources

- When you first log in, you will be on a login node. Your prompt:

```
[user@loginNN ~]$
```

- The login nodes are lightweight virtual machines primarily intended to serve as 'gateways' to RC resources. In order to get a better view of the software available on Alpine start a compile job.

```
[user@loginNN ~]$ acompile
```

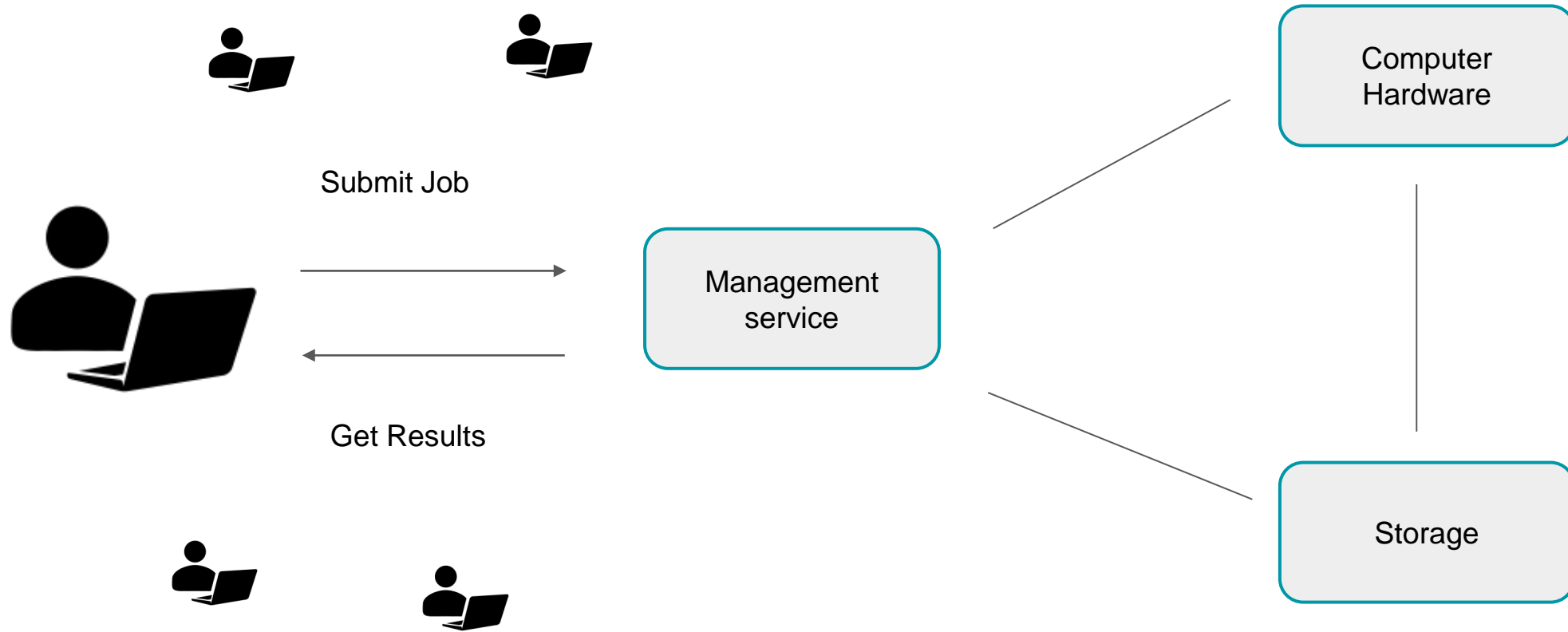
- Navigate to a workspace of your choice (e.g. scratch) and download the material for this workshop:

```
[user@c3cpu-a5-u32-4 ~]$ git clone  
https://github.com/ResearchComputing/Summer_Camp_2023.git  
[user@c3cpu-a5-u32-4 ~]$ cd /Day_Two/Submitting_Jobs/  
[user@c3cpu-a5-u32-4 ~]$ export SPINUP_ROOT=$(pwd)
```

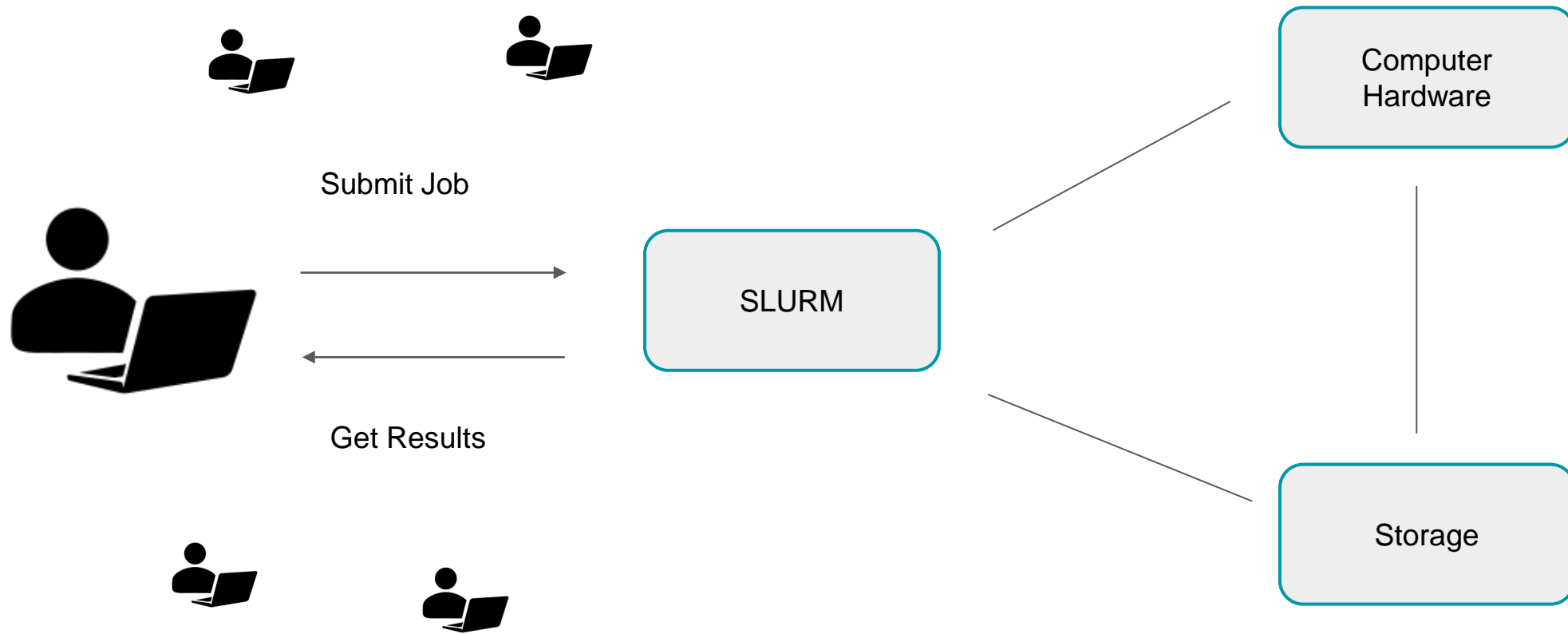
# Jobs

- Because our clusters are shared resources with many users trying to utilize available compute with their applications, we need a system to divide compute in a simple and fair system.
- SLURM
  - Simple **L**inux **U**tility for **R**esource **M**anagement
- Through SLURM, users can grab allotments of compute resources called Jobs
- 2 Types of Jobs
  - **Batch Jobs**
  - **Interactive Jobs**

# HPC - High Performance Computing



# HPC - High Performance Computing



# Batch Jobs

- **Batch Jobs** are jobs you submit to the scheduler where they are run later without supervision.
  - By far the most common job on Alpine
  - Requires a job script
- A job script is simply a script that includes **SLURM directives** (resource specifics) ahead of any commands.



# Submit your first batch job

- First, load up the **slurm Alpine** module

```
$ module load slurm/alpine
```

- **sbatch**: command to submit a batch job
- Submit your first job! :

```
$ cd $SPINUP_ROOT/alpine_scripts  
$ sbatch submit_test.sh
```

- The SLURM Script contains the parameters needed to define a job but additional flags can be used to temporarily replace any set parameters.

```
$ sbatch --partition=atesting --qos=testing submit_test.sh
```

<http://slurm.schedmd.com/sbatch.htm>

# Anatomy of a job script

```
#!/bin/bash

## Directives (HPC Resources)
#SBATCH --<resource>=<amount>

## Software
module load <software>

## User scripting
<command>
```

# Anatomy of a job script

- open `alpine_scripts/submit_test.sh` (nano or vim)

```
#!/bin/bash

## Directives
#SBATCH --ntasks=1           # Number of requested tasks/cores
#SBATCH --time=0:01:00       # Max run time
#SBATCH --partition=amilan    # Specify Alpine CPU node
#SBATCH --output=test_%j.out  # Rename standard output file

## Software
module purge                  # Purge all existing modules

## User commands
echo "This is a test of user $USER"
```

# Job Options

Specified at command line or in job script as...

**#SBATCH** <options> ...where options include:

- **Partition:** `--partition=<partition_name>`
- Sending emails: `--mail-type=<type>`
- Email address: `--mail-user=<user>`
- **Number of nodes:** `--nodes=<nodes>`
- **Number of cores:** `--ntasks=<number-of-tasks>`
- Quality of service: `--qos=<qos>`
- Allocation: `--account=<account_name>`
- Wall time: `--time=<wall time>`
- Job Name: `--job-name=<jobname>`
- **Output:** `--output=<name>`

*More on slurm commands: <https://slurm.schedmd.com/quickstart.html>*

---

*FYI: You do NOT actually type <> above – this designates something specific you as a user must enter about your job*

# Alpine Partitions

Partition	Description	# of nodes	RAM/core (GB)	cores/node	GPUs/node
amilan	General Compute Node: AMD Milan	64	3.74	64	0
ami100	GPU Node: 3x AMD MI100	8	3.74	64	3
aa100	GPU Node: 3x Nvidia A100	8	3.74	64	3
amem	High-memory node	4	21.5	48	0

# Quality of Service

- Quality of Service specifies additional constraints Job
  - On Alpine, this means if your job needs to run longer than 1 day
  - Specify with the `--qos` flag
  - Doesn't need to be set otherwise

```
#SBATCH --qos=long
```

QoS	Description	Max wall time	Max jobs/user	Max nodes/user
normal	Default QoS	Derived from partition	n/a	256
long	For jobs needing longer wall times	7 D	n/a	20
mem	High-memory jobs	7 D	n/a	12

# Writing your first job script



# Your turn!

- Create a job script and submit it as a batch job with the following instructions:
  1. Navigate back to the `job_submission_spinup` directory
  2. Create file `alpine_scripts/sleep.sh`
  3. The job should contain the following commands:

```
echo "Running on host" `hostname`  
echo "Starting Sleep"  
sleep 30  
echo "Ending Sleep. Exiting Job!"
```

*Details on job script parameters are in the next slide*

# Job details of `sleep.sh`

1. The job will run on **1 core on 1 node**
2. We will request a **1 minute wall time**
3. Run on the **amilan partition**
4. Set the output file to be named **“sleep\_%j.out”**
5. Contains the following **commands** ->

```
echo "Running on host" `hostname`  
echo "Starting Sleep"  
sleep 30  
echo "Ending Sleep. Exiting Job!"
```

\* Bonus: Email yourself when the job ends

```
$ sbatch alpine_scripts/sleep.sh
```

# Job Output

- Once a job completes its execution, the standard output of the script will be redirected to an output file.
  - Great for debugging!
  - Could be different from output generated by your application
  - File is created in directory job was run unless specified in your `--output` directive.
  - If the directive `--output` is not provided then a generic file name will be used (slurm\_XXXXXX.out).

```
$ cat output/sleep.XXXXXX.out # where XXXXXX is your Job Id
```

*Solution can be found in “./solutions” subdirectory*

# Checking your jobs (1)

- **squeue**: Monitor your jobs status **in queue and while running**:
  - By Default shows all jobs in queue
  - Narrow this down with:

```
$ squeue -u <username>  
$ squeue -p <partition>
```

- **sacct**: Check back on usage statistics of **previous Jobs**
  - By default only checks all jobs from the start of the current day.
  - Narrow this down with:

```
$ sacct -u <username>  
$ sacct --start=MM/DD/YY -u <username>  
$ sacct -j <job-id>
```

# Checking your jobs (2)

- Another method of checking details of your job while running is with `scontrol`
- Advanced command usually used by system administrators, but you can use it too!

```
$ scontrol show job <job number>
```

- `seff`: Utility to **check efficiency post-job**

```
$ module load slurmttools  
$ seff <job number>
```

# Software and Jobs

- Okay so running a job is easy, but how do I run a job with my software?
- LMOD
  - Module system on CURC systems
  - Modifies your environment to make your desired software visible to your terminal.

```
$ module load matlab  
$ ml matlab #shorthand version!
```

# Software and Jobs (2)

- More LMOD commands:

```
$ module purge           #Unloads all current modules
$ module unload matlab   #Unloads matlab
$ module spider matlab    #Searches for matlab in module tree
```

- What if my software isn't available through LMOD?
  - Software must be installed locally if not available through LMOD
  - RC User support is happy to assist, *installs are best effort*
  - For more assistance contact [rc-help@colorado.edu](mailto:rc-help@colorado.edu)



# Example 1: Serial R Code

# Running an external program

- Let's run R on an R script
- Batch script calls and runs `programs/R_program.R`
  - Let's take a look at the R program
- Let's examine the batch script `scripts/submit_R.sh`
  - Note how R is loaded
  - R program can be run with “Rscript <script>”
- Go ahead and submit the batch script:

```
$ sbatch scripts/submit_R.sh
```

# GPU Jobs

# GPU Jobs

- On Alpine the `--gres` slurm directive is ***required*** to use GPU accelerators on a GPU node.
- At a minimum, one would specify:
  - A GPU partition (e.g. `--partition=aa100` for an nvidia GPU node)
  - `--gres=gpu` in a job to specify that they would like to use a single gpu on their specified partition
    - You can request up to 3 accelerators on Alpine (e.g. `--gres=gpu:3` )

# GPU Job Script Example

```
#!/bin/bash

## Directives
#SBATCH --ntasks=1           # Number of requested tasks/cores
#SBATCH --time=0:01:00      # Max run time
#SBATCH --partition=aa100    # Specify Alpine NVIDIA A100 node
#SBATCH --gres=gpu:2         # Request 2 GPUs
from the node
```

# Advanced Job Scripts

# Running an mpi job

- For cases where you have a code that is parallelized, meaning it can run across multiple cores.
- Number of tasks always  $> 1$ . E.g.,

```
#SBATCH --ntasks=4
```

- Will always need to load a compiler and mpi. E.g.,

```
module load intel impi
```

- Executable preceded with mpirun, srun, or mpiexec. E.g.,

```
mpirun -np 4 python yourscrip.py
```

- Examine and run the example `'submit_python_mpi.sh'`

```
$ sbatch scripts/submit_python_mpi.sh
```



# Running serial jobs in parallel

- Not all code is designed to run with MPI (nor always makes sense to do so)
- RC has a couple different tools that lets users run serial programs in parallel
  - [RC LoadBalancer](#)
  - [GNU Parallel](#)
- Example in: [scripts/python\\_loadbalance.sh](#)

# Interactive Jobs

# Interactive jobs

- Sometimes we want our job to run in the background
- Sometimes we want to work on program in real time
  - Great for testing, debugging
- We can get access to a compute node interactively with `sinteractive`
- For example, let's run the R job we previously ran as a batch job, but this time let's do it interactively

# Running an interactive job

- To work with R interactively, we request time from Alpine
- When the resources become available the job starts
- Commands to run:

```
$ sinteractive --time=00:10:00
```

- Once we receive a prompt, then:

```
$ module load R  
$ cd programs  
$ Rscript R_program.R
```

- Once we finish we must exit! (job will time out eventually)

```
$ exit
```

# Thank you!

- Survey: <http://tinyurl.com/curc-survey18>
- Contact information: [rc-help@Colorado.edu](mailto:rc-help@Colorado.edu)
- Slurm Commands: <https://slurm.schedmd.com/quickstart.html>