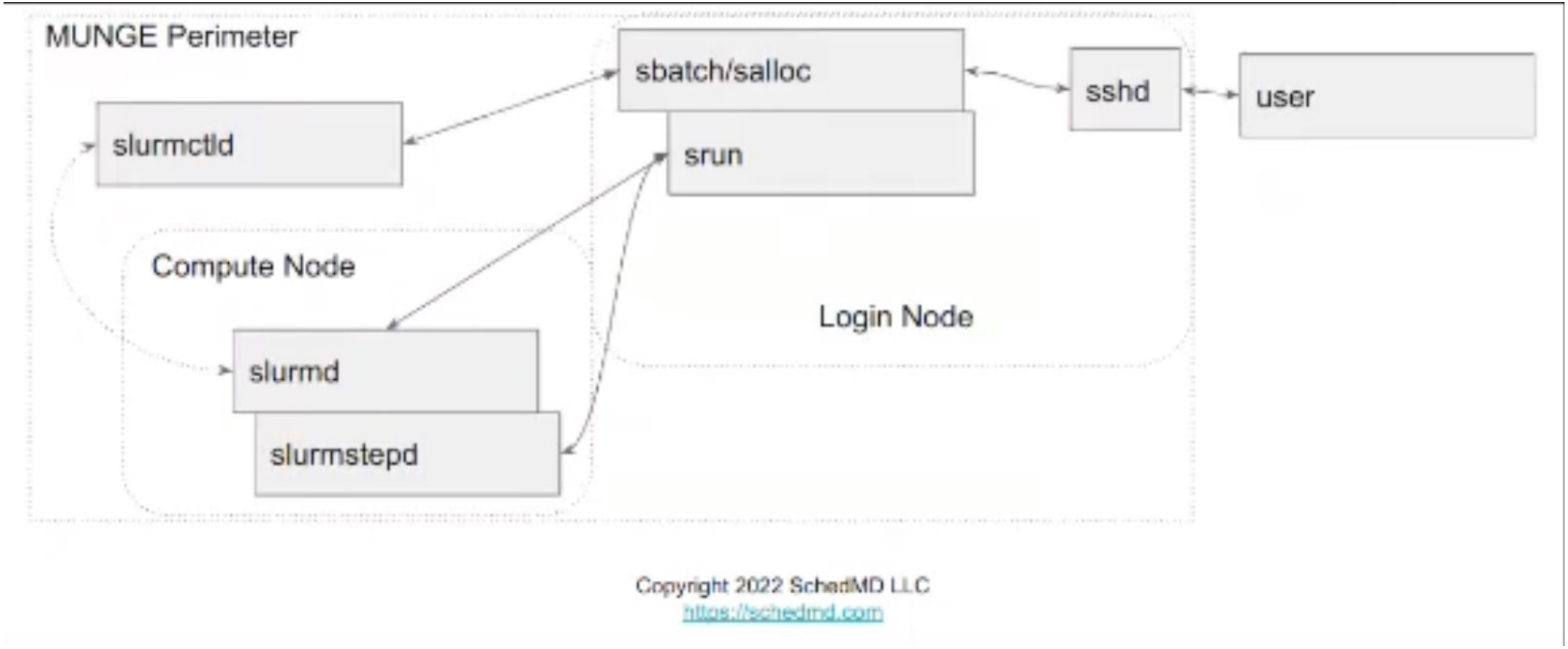
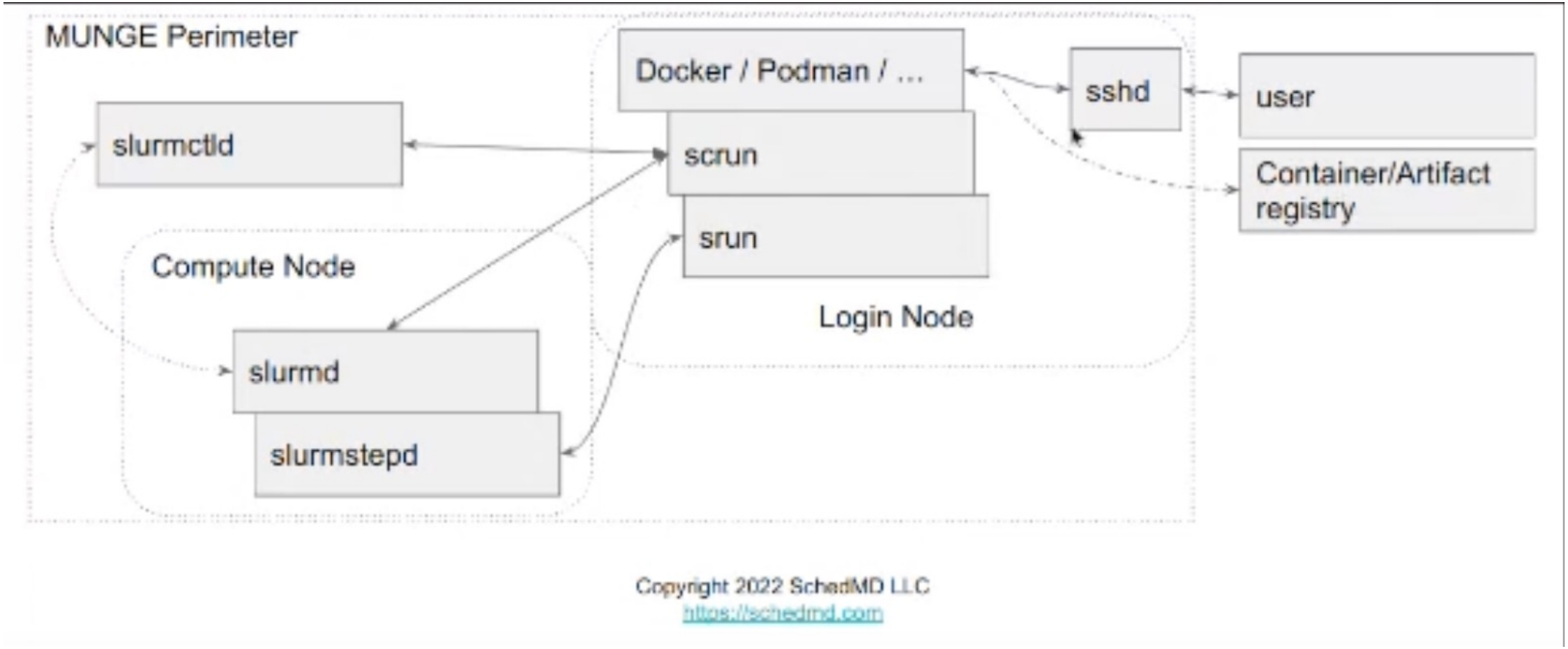
Lecture 16: Slurm

Installation and Configuration

How Slurm Works (standard model)



How Slurm Works (new model)



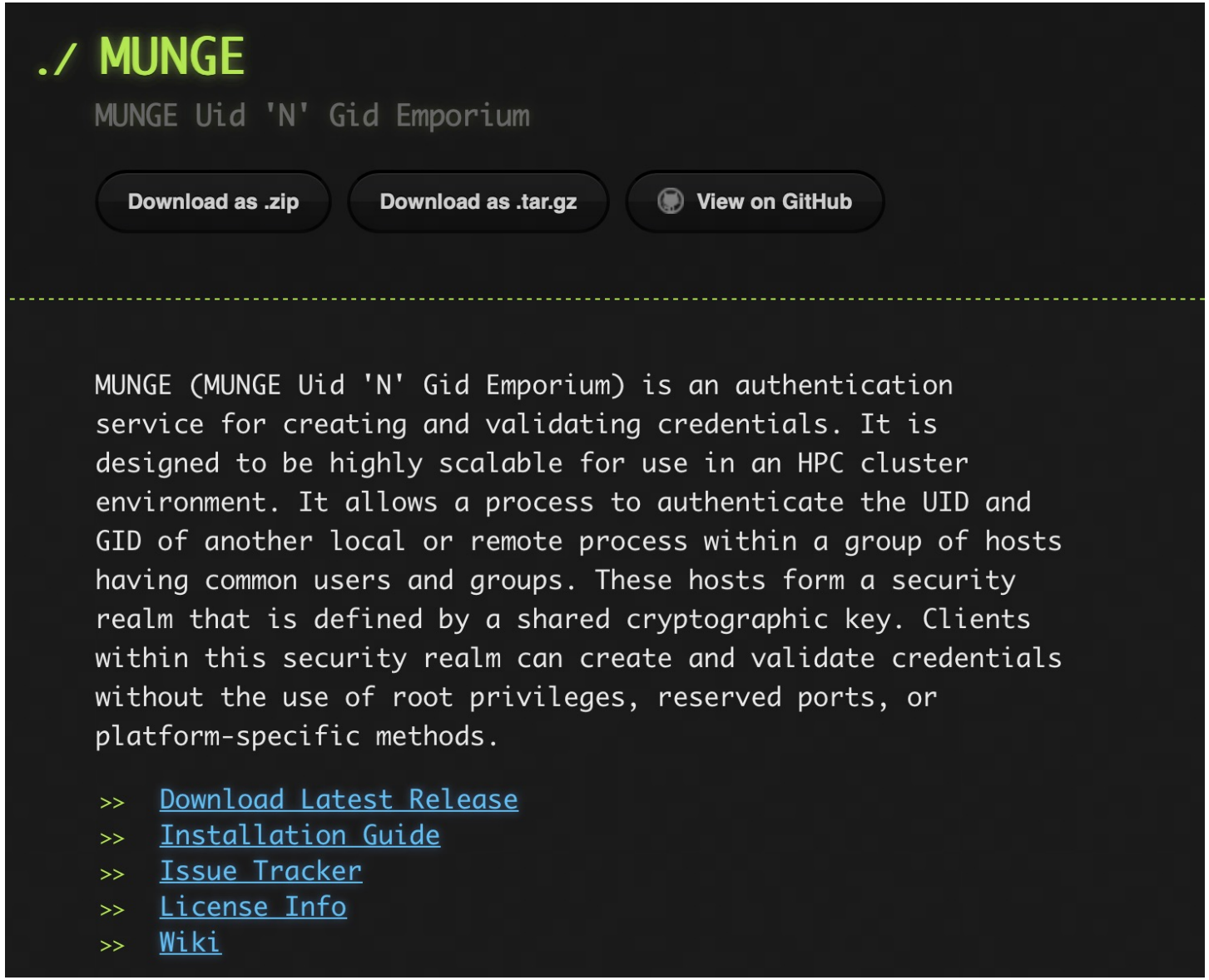
Slurm Control Daemon (slurmctld)

- This daemon runs the scheduler and decides what jobs run where. It talks to the slurm daemons on the compute nodes – telling them when the run and when to kill processes running there.

MUNGE

- MUNGE Ui 'N' Gid Emporium
- https://dun.github.io/munge/
- Purpose: Authentication across a cluster.
- Maps Group IDs (GIDs) and User IDs (UIDs).

Enter `id` to see your UID and GIDs.



Slurm Daemon (slurmd)

- Runs on the compute nodes and communicates with the slurmctld controller daemon.
- Runs under the root account because it needs complete control over processes running on the compute node.

Slurm Setup Tasks

- https://slurm.schedmd.com/quickstart_admin.html

1. Install MUNGE to synchronize authentication across the cluster
2. Check that users and groups are resolvable on the compute nodes
3. Create a “slurm” user under which the daemons will run
4. Make sure time is synchronized across the cluster
5. Download the latest Slurm code as Red Hat Packages (RPM)
6. Build RPMs and install on the Head Node
   - slurm, slurm-perlapi, slurm-slurmctld, slurm-slurmdbd
7. Build RPMs and install on the Compute Node
   - slurm-slurmd
8. Restricting access to compute nodes with PAM
9. Start the slurm daemons.
10. Submit a test job
11. Run scontrol to get slurm’s jobs status

Security
Software Installation
Testing

Security (Authentication and Authorisation)

- Authentication: Checking that a person or process is who they say they are.
- Authorisation: Making sure that people and processes only access things they are allowed to access.

- Tools: passwords, authentication keys, Privileged Access Management (PAM modules), MUNGE.

Security (Authentication and Authorisation)

- Authentication: Checking that a person or process is who they say they are.
- Authorisation: Making sure that people and processes only access things they are allowed to access.

- Tools: passwords, authentication keys, Privileged Access Management (PAM modules), MUNGE.

Install MUNGE on the Head Node

```
[matthew@moonshine ~]$ yum search munge
Last metadata expiration check: 0:12:41 ago on Sun 31 Mar 2024 06:58:22 PM CDT.
============== Name Exactly Matched: munge ==================
munge.x86_64 : Enables uid & gid authentication across a host cluster
=========================Name Matched: munge ======================
munge-devel.i686 : Development files for uid * gid authentication across a host cluster
munge-devel.x86_64 : Development files for uid * gid authentication across a host cluster
munge-libs.x86_64 : Runtime libs for uid * gid authentication across a host cluster
munge-libs.i686 : Runtime libs for uid * gid authentication across a host cluster

[matthew@moonshine ~]$ sudo yum install munge munge-devel

[matthew@moonshine ~]$ getent passwd munge
munge:x:987:987:Runs Uid N Gid Emporium:/run/munge:/sbin/nologin
```

Generate MUNGE Authentication Key

```
[matthew@moonshine ~]$ sudo create-munge-key
Generating a pseudo-random key using /dev/urandom completed.

[matthew@moonshine ~]$ sudo ls /etc/munge
munge.key
```

Enable Munge systemd unit

```
[matthew@moonshine ~]$ sudo systemctl enable munge
Created symlink /etc/systemd/system/multi-user.target.wants/munge.service →
/usr/lib/systemd/system/munge.service.


○ munge.service - MUNGE authentication service
     Loaded: loaded (/usr/lib/systemd/system/munge.service; enabled; preset: disabled)
     Active: inactive (dead)
       Docs: man:munged(8)


Mar 31 19:29:55 moonshine systemd[1]: /usr/lib/systemd/system/munge.service:10: PIDFile=
references a path… /run/munge/munge.pid
```

Enable Munge systemd unit

```
[matthew@moonshine ~]$ sudo systemctl enable munge
Created symlink /etc/systemd/system/multi-user.target.wants/munge.service →
/usr/lib/systemd/system/munge.service.


○ munge.service - MUNGE authentication service
     Loaded: loaded (/usr/lib/systemd/system/munge.service; enabled; preset: disabled)
     Active: inactive (dead)
       Docs: man:munged(8)


Mar 31 19:29:55 moonshine systemd[1]: /usr/lib/systemd/system/munge.service:10: PIDFile=
references a path… /var/run/munge/munge.pid

[matthew@moonshine ~]$ sudo systemctl edit munge
```

Correct the Process ID Path for new Linux

matthew — matthew@moonshine:~ — ssh moonshine — 108×27

```
### Editing /etc/systemd/system/munge.service.d/override.conf
### Anything between here and the comment below will become the new contents of the file



### Lines below this comment will be discarded

### /usr/lib/systemd/system/munge.service
# [Unit]
# Description=MUNGE authentication service
# Documentation=man:munged(8)
# After=network.target
# After=time-sync.target
#
# [Service]
# Type=forking
# ExecStart=/usr/sbin/munged
# PIDFile=/var/run/munge/munged.pid
# User=munge
# Group=munge
# Restart=on-abort
#
# [Install]
# WantedBy=multi-user.target
~
~
<temd/system/munge.service.d/.#override.conf87d0f78772d8f4a5" [noeol] 24L, 558B          1,1           All
```

Correct the Process ID Path for new Linux

matthew — matthew@moonshine:~ — ssh moonshine — 108×27

```
### Editing /etc/systemd/system/munge.service.d/override.conf
### Anything between here and the comment below will become the new contents of the file

PIDFile=/var/run/munged.pid

### Lines below this comment will be discarded

### /usr/lib/systemd/system/munge.service
# [Unit]
# Description=MUNGE authentication service
# Documentation=man:munged(8)
# After=network.target
# After=time-sync.target
#
# [Service]
# Type=forking
# ExecStart=/usr/sbin/munged
# PIDFile=/var/run/munge/munged.pid
# User=munge
# Group=munge
# Restart=on-abort
#
# [Install]
# WantedBy=multi-user.target
~
~
-- INSERT --                                                    4,28          All
```

Correct the Process ID Path for new Linux

matthew — matthew@moonshine:~ — ssh moonshine — 108×27

```
### Editing /etc/systemd/system/munge.service.d/override.conf
### Anything between here and the comment below will become the new contents of the file

PIDFile=/var/run/munged.pid

### Lines below this comment will be discarded

### /usr/lib/systemd/system/munge.service
# [Unit]
# Description=MUNGE authentication service
# Documentation=man:munged(8)
# After=network.target
# After=time-sync.target
#
# [Service]
# Type=forking
# ExecStart=/usr/sbin/munged
# PIDFile=/var/run/munge/munged.pid
# User=munge
# Group=munge
# Restart=on-abort
#
# [Install]
# WantedBy=multi-user.target
~
~
:wq
```

Correct the Process ID Path for new Linux

```
[matthew@moonshine ~]$ sudo systemctl edit munge
[matthew@moonshine ~]$ sudo systemctl restart munge
[matthew@moonshine ~]$ sudo systemctl status munge
● munge.service - MUNGE authentication service
     Loaded: loaded (/usr/lib/systemd/system/munge.service; enabled; preset: disabled)
    Drop-In: /etc/systemd/system/munge.service.d
             └─override.conf
     Active: active (running) since Sun 2024-03-31 19:46:34 CDT; 5s ago
       Docs: man:munged(8)
    Process: 142902 ExecStart=/usr/sbin/munged (code=exited, status=0/SUCCESS)
   Main PID: 142904 (munged)
      Tasks: 4 (limit: 407887)
     Memory: 1.7M
        CPU: 10ms
     CGroup: /system.slice/munge.service
             └─142904 /usr/sbin/munged


Mar 31 19:46:34 moonshine systemd[1]: Starting MUNGE authentication service...
Mar 31 19:46:34 moonshine systemd[1]: Started MUNGE authentication service.
Mar 31 19:46:39 moonshine systemd[1]:
/etc/systemd/system/munge.service.d/override.conf:1: Assignment outsi
```

Install MUNGE into the Compute Node Image

```
[matthew@moonshine ~]$ sudo -i
[root@moonshine ~]# wwctl container exec rocky-9 /bin/bash
[rocky-9] Warewulf> yum install munge

<snip>

Installed:
  munge-0.5.13-13.el9.x86_64                    munge-libs-0.5.13-13.el9.x86_64


[rocky-9] Warewulf> systemctl enable munge
[rocky-9] Warewulf> exit

+ dnf clean all
25 files removed
Rebuilding container...
Created image for VNFS container rocky-9:
/var/lib/warewulf/provision/container/rocky-9.img
Compressed image for VNFS container rocky-9:
/var/lib/warewulf/provision/container/rocky-9.img.gz
```

Create munge.key overlay

```
[root@moonshine ~]# wwctl overlay import --parents wwinit /etc/munge/munge.key
Building overlay for moonshine01: [wwinit]
Created image for overlay moonshine01/[wwinit]:
/var/lib/warewulf/provision/overlays/moonshine01/wwinit.img
Compressed image for overlay moonshine01/[wwinit]:
/var/lib/warewulf/provision/overlays/moonshine01/wwinit.img.gz
```

Add the corrected munge service file to the overlay

```
[root@moonshine ~]# wwctl overlay import --parents wwinit
/usr/lib/systemd/system/munge.service
Building overlay for moonshine01: [wwinit]
Created image for overlay moonshine01/[wwinit]:
/var/lib/warewulf/provision/overlays/moonshine01/wwinit.img
Compressed image for overlay moonshine01/[wwinit]:
/var/lib/warewulf/provision/overlays/moonshine01/wwinit.img.gz

[root@moonshine ~]# file /var/lib/warewulf/overlays/wwinit/rootfs/usr/lib/systemd/system/
/var/lib/warewulf/overlays/wwinit/rootfs/usr/lib/systemd/system/: directory
```

Set permissions for munge directories in compute node container

```
[root@moonshine ~]# wwctl overlay chown wwinit /etc/munge/munge.key $(id -u munge) $(id -g munge)
[root@moonshine ~]# wwctl overlay chmod wwinit /etc/munge/munge.key 0400
[root@moonshine ~]# wwctl overlay chown wwinit /etc/munge $(id -u munge) $(id -g munge)
[root@moonshine ~]# wwctl overlay chmod wwinit /etc/munge 0700
```

Add the corrected munge service file to the overlay

```
[root@moonshine ~]# wwctl overlay build
Building system overlays for moonshine01: [wwinit]
Created image for overlay moonshine01/[wwinit]:
/var/lib/warewulf/provision/overlays/moonshine01/__SYSTEM__.img
Compressed image for overlay moonshine01/[wwinit]:
/var/lib/warewulf/provision/overlays/moonshine01/__SYSTEM__.img.gz
Building runtime overlays for moonshine01: [generic]
Created image for overlay moonshine01/[generic]:
/var/lib/warewulf/provision/overlays/moonshine01/__RUNTIME__.img
Compressed image for overlay moonshine01/[generic]:
/var/lib/warewulf/provision/overlays/moonshine01/__RUNTIME__.img.gz
```

Set the permissions for MUNGE on the Head Node

```
[root@moonshine ~]# chown -R munge: /etc/munge/ /var/log/munge/ /var/lib/munge/ /run/munge/
[root@moonshine ~]# chmod 0700 /etc/munge/ /var/log/munge/ /var/lib/munge/
[root@moonshine ~]# chmod 0711 /var/run/munge/
```

Boot from the new image to test changes later

```
[root@moonshine ~]# ssh moonshine01
[root@moonshine01 ~]# reboot
```

Enable the MUNGE unit on the head node and in the compute node image

```
[root@moonshine ~]# systemctl enable munge
Created symlink /etc/systemd/system/multi-user.target.wants/munge.service →
/usr/lib/systemd/system/munge.service.
```

```
[root@moonshine ~]# munge -n
MUNGE:AwQFAAAJfgI5RDRCyuX0bWqDHXIc2ZLL0JNcrI5dAkvu3ivw7I3knZasEjnYSuzh2z1EAKKcXQt5OKwDQAgAfKvqiTyJSqVwnGwnWniBd2+AG1FgfNEZDlCrRoNugSIDiJm4KH0=:
[root@moonshine ~]# munge -n | unmunge
STATUS:          Success (0)
ENCODE_HOST:     moonshine (10.0.0.1)
ENCODE_TIME:     2024-03-31 22:33:08 -0500 (1711942388)
DECODE_TIME:     2024-03-31 22:33:08 -0500 (1711942388)
TTL:             300
CIPHER:          aes128 (4)
MAC:             sha256 (5)
ZIP:             none (0)
UID:             root (0)
GID:             root (0)
LENGTH:          0


[root@moonshine ~]# munge -n | ssh moonshine01 unmunge
STATUS:          Success (0)
ENCODE_HOST:     moonshine (10.0.0.1)
ENCODE_TIME:     2024-04-01 03:33:27 +0000 (1711942407)
DECODE_TIME:     2024-04-01 03:33:35 +0000 (1711942415)
TTL:             300
CIPHER:          aes128 (4)
MAC:             sha256 (5)
ZIP:             none (0)
UID:             root (0)
GID:             root (0)
LENGTH:          0


[root@moonshine ~]# remunge
2024-03-31 22:33:45 Spawning 1 thread for encoding
2024-03-31 22:33:45 Processing credentials for 1 second
2024-03-31 22:33:46 Processed 14748 credentials in 1.000s (14745 creds/sec)
```

Test MUNGE

Time Sychronisation

```
[root@moonshine ~]# sudo timedatectl set-timezone America/Denver


[root@moonshine ~]# wwctl overlay import wwinit /etc/localtime

[rocky-9] Warewulf> yum install chrony

[root@moonshine ~]# wwctl overlay build

[rocky-9] Warewulf> exit
exit
+ dnf clean all
25 files removed
Rebuilding container...
Created image for VNFS container rocky-9:
/var/lib/warewulf/provision/container/rocky-9.img
Compressed image for VNFS container rocky-9:
/var/lib/warewulf/provision/container/rocky-9.img.gz
[root@moonshine ~]#
```

Check time update on compute node

```
[root@moonshine01 ~]# systemctl status chronyd
● chronyd.service - NTP client/server
     Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; preset: enabled)
     Active: active (running) since Sun 2024-03-31 23:14:25 MDT; 1min 17s ago
       Docs: man:chronyd(8)
             man:chrony.conf(5)
    Process: 718 ExecStart=/usr/sbin/chronyd $OPTIONS (code=exited, status=0/SUCCESS)
   Main PID: 730 (chronyd)
      Tasks: 1 (limit: 406326)
     Memory: 1016.0K
        CPU: 47ms
     CGroup: /system.slice/chronyd.service
             └─730 /usr/sbin/chronyd -F 2


Mar 31 23:14:25 moonshine01 systemd[1]: Starting NTP client/server...
Mar 31 23:14:25 moonshine01 chronyd[730]: chronyd version 4.3 starting (+CMDMON +NTP +REFCLOCK
+RTC +PRIVDR
Mar 31 23:14:25 moonshine01 chronyd[730]: Using right/UTC timezone to obtain leap second data
Mar 31 23:14:25 moonshine01 chronyd[730]: Loaded seccomp filter (level 2)
Mar 31 23:14:25 moonshine01 systemd[1]: Started NTP client/server.
Mar 31 23:15:04 moonshine01 chronyd[730]: Selected source 152.70.159.102 (2.rocky.pool.ntp.org)
Mar 31 23:15:04 moonshine01 chronyd[730]: System clock wrong by -8.276273 seconds
Mar 31 23:14:56 moonshine01 chronyd[730]: System clock was stepped by -8.276273 seconds
Mar 31 23:14:56 moonshine01 chronyd[730]: System clock TAI offset set to 37 seconds
```

Create the Slurm user and group

```
[root@moonshine ~]# export SLURMUSER=900

[root@moonshine ~]# groupadd -g $SLURMUSER slurm

[root@moonshine ~]# useradd  -m -c "SLURM workload manager" -d
/var/lib/slurm -u $SLURMUSER -g slurm  -s /bin/bash slurm
```

Create the Slurm user and group

```
[matthew@moonshine ~]$  getent passwd 900
slurm:x:900:900:SLURM workload manager:/var/lib/slurm:/bin/bash
[matthew@moonshine ~]$  getent group 900
slurm:x:900:
```

Always check that the commands you run did what you expected.

Create the Slurm user and group – sync the Slurm user and group with the container

```
[matthew@moonshine ~]$ wwctl container syncuser --write rocky-9 --build
uid/gid synced for container rocky-9
Created image for VNFS container rocky-9:
/var/lib/warewulf/provision/container/rocky-9.img
Compressed image for VNFS container rocky-9:
/var/lib/warewulf/provision/container/rocky-9.img.gz
```

Setup Database for Slurm to Use for Accounting

```
[root@moonshine ~]# yum install mariadb-server mariadb-devel

Installed:
  checkpolicy-3.5-1.el9.x86_64                 mariadb-3:10.5.22-1.el9_2.x86_64
  mariadb-backup-3:10.5.22-1.el9_2.x86_64      mariadb-common-3:10.5.22-1.el9_2.x86_64
  mariadb-errmsg-3:10.5.22-1.el9_2.x86_64      mariadb-gssapi-server-3:10.5.22-1.el9_2.x86_64
  mariadb-server-3:10.5.22-1.el9_2.x86_64      mariadb-server-utils-3:10.5.22-1.el9_2.x86_64
  mysql-selinux-1.0.5-1.el9_0.noarch           perl-DBD-MariaDB-1.21-16.el9_0.x86_64
  perl-DBI-1.643-9.el9.x86_64                  policycoreutils-python-utils-3.5-3.el9_3.noarch
  python3-audit-3.0.7-104.el9.x86_64           python3-distro-1.5.0-7.el9.noarch
  python3-libsemanage-3.5-2.el9.x86_64         python3-policycoreutils-3.5-3.el9_3.noarch
  python3-setools-4.4.3-1.el9.x86_64           python3-setuptools-53.0.0-12.el9.noarch

[root@moonshine ~]# systemctl enable mariadb
[root@moonshine ~]# systemctl start mariadb
[root@moonshine ~]# mysql_secure_installation
```

Setup Database for Slurm to Use

Set the password to CS491PW!# and then respond yes to all the prompts except the one that offers to reset the root password.

Slurm Installation Prerequisites

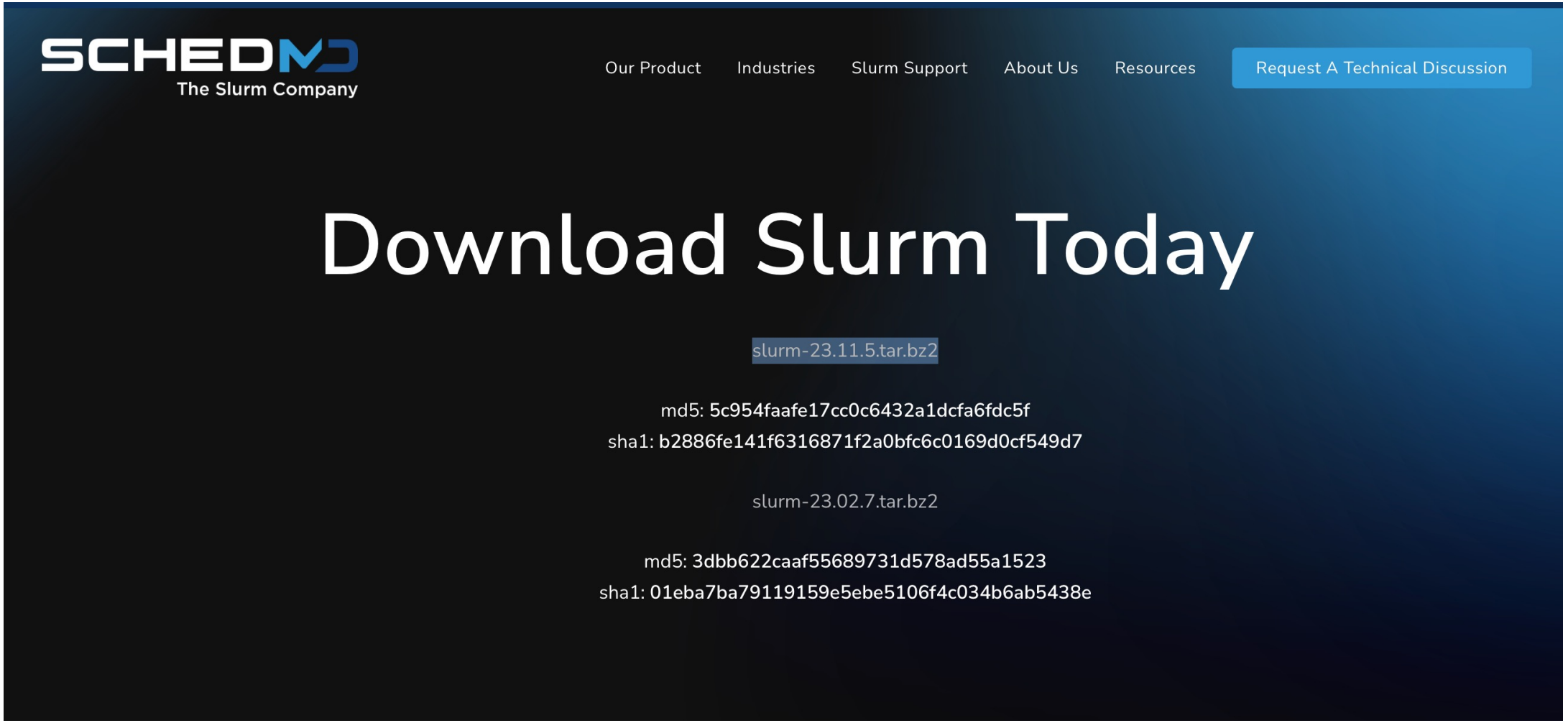
```
[root@moonshine01 ~]# sudo yum install mariadb-devel munge-
devel pam-devel readline-devel perl
```

The head node has many of the following installed already – but the compute node won't

```
[root@moonshine ~]# wwctl container exec rocky-9 /bin/bash
[rocky-9] Warewulf> [rocky-9] Warewulf> yum install dnf-plugins-core
[rocky-9] Warewulf> yum config-manager --set-enabled crb

[rocky-9] Warewulf> yum install gcc gcc-c++ tar make python3 openssl
openssl-devel pam-devel numactl-devel hwloc hwloc-devel lua lua-
devel readline-devel rrdtool-devel ncurses-devel libibmad libibumad
libevent libevent-devel dbus-devel
```

Needed for cgroup v2 compatability

Slurm Installation




```
wget https://download.schedmd.com/slurm/slurm-23.11.5.tar.bz2
```

Build the Code with RPM

```
[matthew@moonshine ~]$ wget https://download.schedmd.com/slurm/slurm-23.11.5.tar.bz2
--2024-03-31 22:48:45--  https://download.schedmd.com/slurm/slurm-23.11.5.tar.bz2
Resolving download.schedmd.com (download.schedmd.com)... 71.19.154.210
Connecting to download.schedmd.com (download.schedmd.com)|71.19.154.210|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 7365175 (7.0M) [application/x-bzip2]
Saving to: ‘slurm-23.11.5.tar.bz2’


slurm-
23.11.5.tar.bz2      100%[=====================================>]   7.02M  6.62MB/s
  in 1.1s


2024-03-31 22:48:46 (6.62 MB/s) - ‘slurm-23.11.5.tar.bz2’ saved [7365175/7365175]

[matthew@moonshine ~]$ rpmbuild -ta slurm-23.11.5.tar.bz2
```

Install the packages with yum

```
[matthew@moonshine ~]$ rpmbuild -ta slurm-23.11.5.tar.bz2
[matthew@moonshine ~]$ cd rpmbuild/RPMS/x86_64
[matthew@moonshine x86_64]$ sudo yum localinstall slurm-23.11.5-1.el9.x86_64.rpm
[matthew@moonshine x86_64]$ sudo yum localinstall slurm-slurmctld-23.11.5-1.el9.x86_64.rpm
[matthew@moonshine x86_64]$ sudo yum localinstall slurm-perlapi-23.11.5-1.el9.x86_64.rpm
[matthew@moonshine x86_64]$ sudo yum localinstall slurm-slurmdbd-23.11.5-1.el9.x86_64.rpm
[matthew@moonshine x86_64]$ sudo yum localinstall slurm-pam_slurm-23.11.5-1.el9.x86_64.rpm
[matthew@moonshine x86_64]$ sudo yum localinstall slurm-example-configs-23.11.5-1.el9.x86_64.rpm
```

Install slurm into the compute node boot image

```
[rocky-9] Warewulf> wget https://download.schedmd.com/slurm/slurm-23.11.5.tar.bz2
[rocky-9] Warewulf> yum install rpm-build
[rocky-9] Warewulf> rpmbuild -ta slurm-23.11.5.tar.bz2
[rocky-9] Warewulf> cd /root/rpmbuild/RPMS/x86_64/
[rocky-9] Warewulf> ls
slurm-23.11.5-1.el9.x86_64.rpm                  slurm-perlapi-23.11.5-1.el9.x86_64.rpm
slurm-contribs-23.11.5-1.el9.x86_64.rpm         slurm-sackd-23.11.5-1.el9.x86_64.rpm
slurm-devel-23.11.5-1.el9.x86_64.rpm            slurm-slurmctld-23.11.5-
1.el9.x86_64.rpm
slurm-example-configs-23.11.5-1.el9.x86_64.rpm  slurm-slurmd-23.11.5-1.el9.x86_64.rpm
slurm-libpmi-23.11.5-1.el9.x86_64.rpm           slurm-slurmdbd-23.11.5-
1.el9.x86_64.rpm
slurm-openlava-23.11.5-1.el9.x86_64.rpm         slurm-torque-23.11.5-1.el9.x86_64.rpm
slurm-pam_slurm-23.11.5-1.el9.x86_64.rpm
[rocky-9] Warewulf>
```

Install the packages with yum

```
[rocky-9] Warewulf> yum localinstall slurm-23.11.5-1.el9.x86_64.rpm slurm-slurmd-
23.11.5-1.el9.x86_64.rpm
Last metadata expiration check: 0:23:34 ago on Mon Apr  1 06:05:16 2024.
Dependencies resolved.

Installed:
  slurm-23.11.5-1.el9.x86_64    slurm-slurmd-23.11.5-1.el9.x86_64


Complete!
```

Install the packages with yum

```
[rocky-9] Warewulf> ls
anaconda-ks.cfg  anaconda-post.log  original-ks.cfg  rpmbuild
[rocky-9] Warewulf> rm -rf rpmbuild
[rocky-9] Warewulf> exit
exit

+ dnf clean all
35 files removed
Rebuilding container...
Created image for VNFS container rocky-9: /var/lib/warewulf/provision/container/rocky-
9.img
Compressed image for VNFS container rocky-9:
/var/lib/warewulf/provision/container/rocky-9.img.gz
```

Create spool directory on the Head Node

```
[root@moonshine ~]# mkdir /var/spool/slurmctld
[root@moonshine ~]# chown slurm:slurm /var/spool/slurmctld
```

A spool is a place to write data that will be processed later and maybe by a different program.

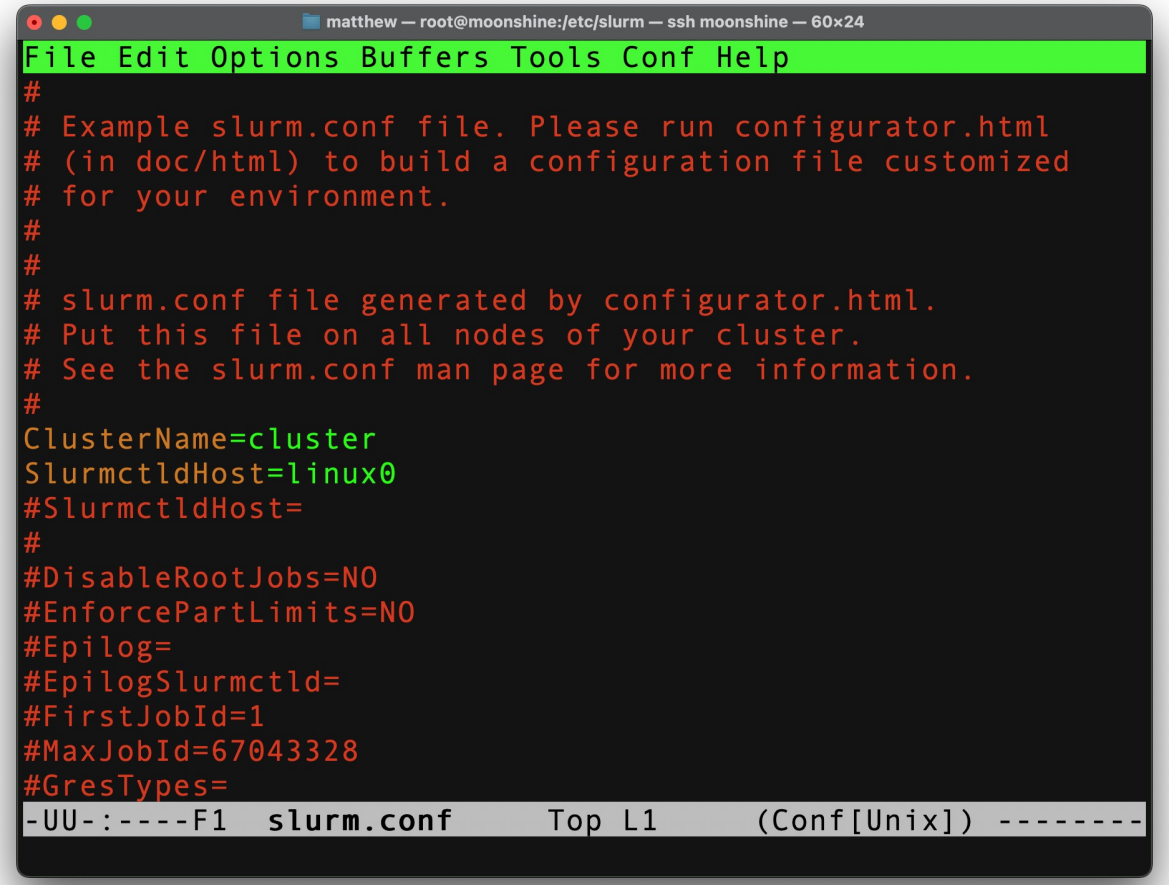
For example, an email spool might be an outgoing email queue, or a printer spool might be where you put documents to be printed.

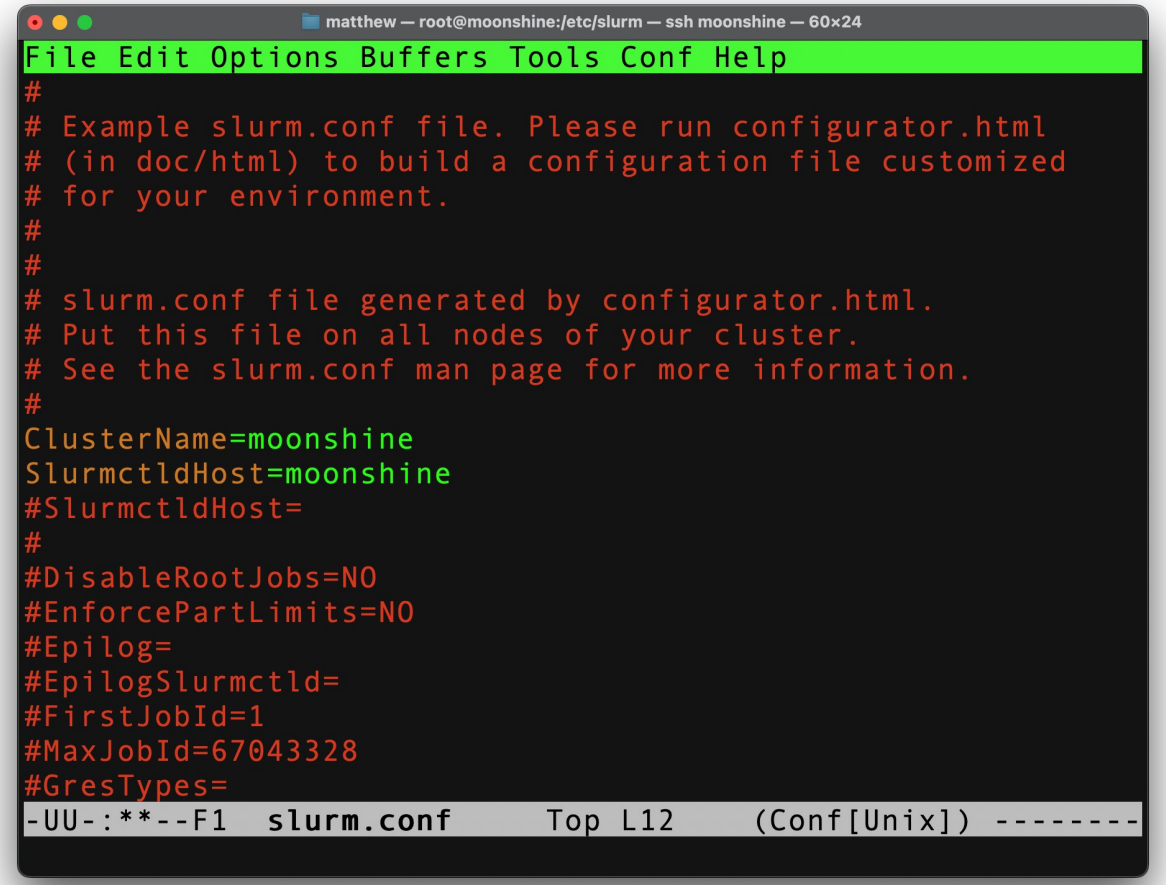
Configure Slurm

```
[root@moonshine ~]# cd /etc/slurm/
[root@moonshine slurm]# ls
cgroup.conf.example job_submit.lua.example slurm.conf.example
cli_filter.lua.example prolog.example slurmdbd.conf.example
[root@moonshine slurm]# cp slurm.conf.example slurm.conf
[root@moonshine slurm]# cp slurmdbd.conf.example slurmdbd.conf
```

Configure Slurm

```
[root@moonshine slurm]# emacs slurm.conf
```





Configure Slurm

Allow Slurm ports through Firewalld

```
[root@moonshine slurm]# firewall-cmd --permanent --zone=internal --add-port=6817/tcp
success
[root@moonshine slurm]# firewall-cmd --permanent --zone=internal --add-port=6819/tcp
success
[root@moonshine slurm]# firewall-cmd --reload
success
```

We may end up opening all ports on the internal zone to handle interactive jobs later

Enable and Start the System Control Daemon

```
[root@moonshine slurm]# systemctl enable --now slurmctld
Created symlink /etc/systemd/system/multi-
user.target.wants/slurmctld.service →
/usr/lib/systemd/system/slurmctld.service.
[root@moonshine slurm]#
```

Start the System Control Daemon

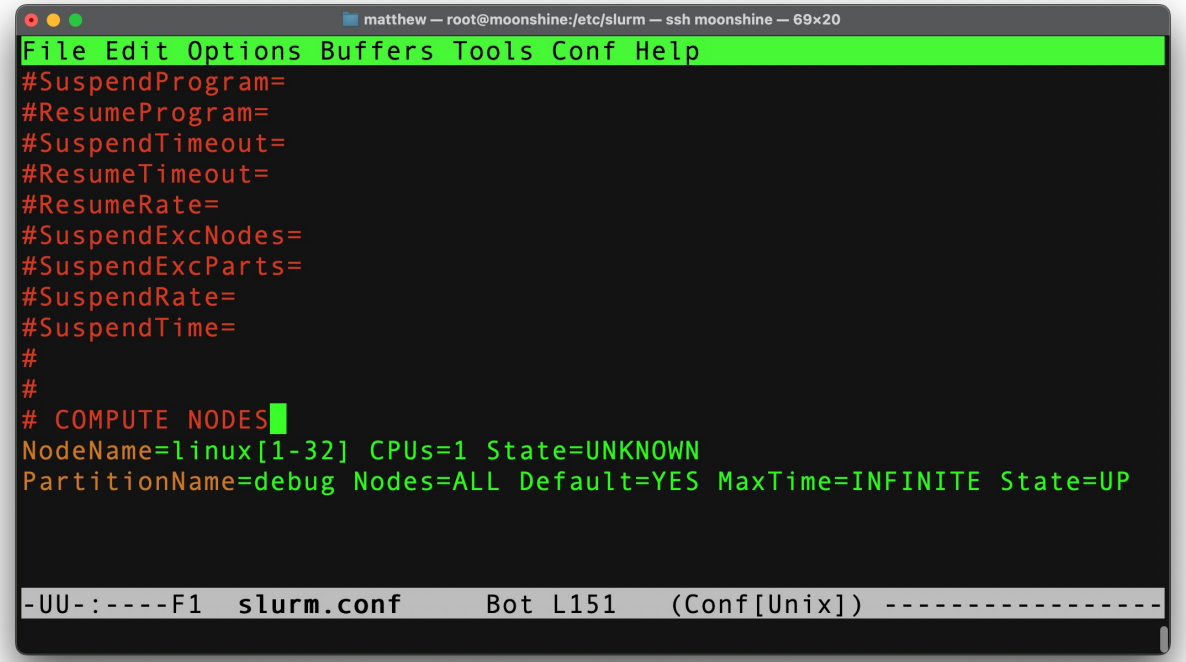
```
[root@moonshine slurm]# systemctl status slurmctld
● slurmctld.service - Slurm controller daemon
     Loaded: loaded (/usr/lib/systemd/system/slurmctld.service; enabled;
preset: disabled)
     Active: active (running) since Mon 2024-04-01 01:36:38 MDT; 37s ago
   Main PID: 254614 (slurmctld)
      Tasks: 14
     Memory: 3.3M
        CPU: 56ms
     CGroup: /system.slice/slurmctld.service
             ├─254614 /usr/sbin/slurmctld --systemd
             └─254615 "slurmctld: slurmscriptd"


Apr 01 01:36:38 moonshine slurmctld[254614]: slurmctld: error:
slurm_get_port: Address family '0' not supported
Apr 01 01:36:38 moonshine slurmctld[254614]: slurmctld: error: Apr 01
01:36:38 moonshine slurmctld[254614]: slurmctld: Running as primary
controller
```

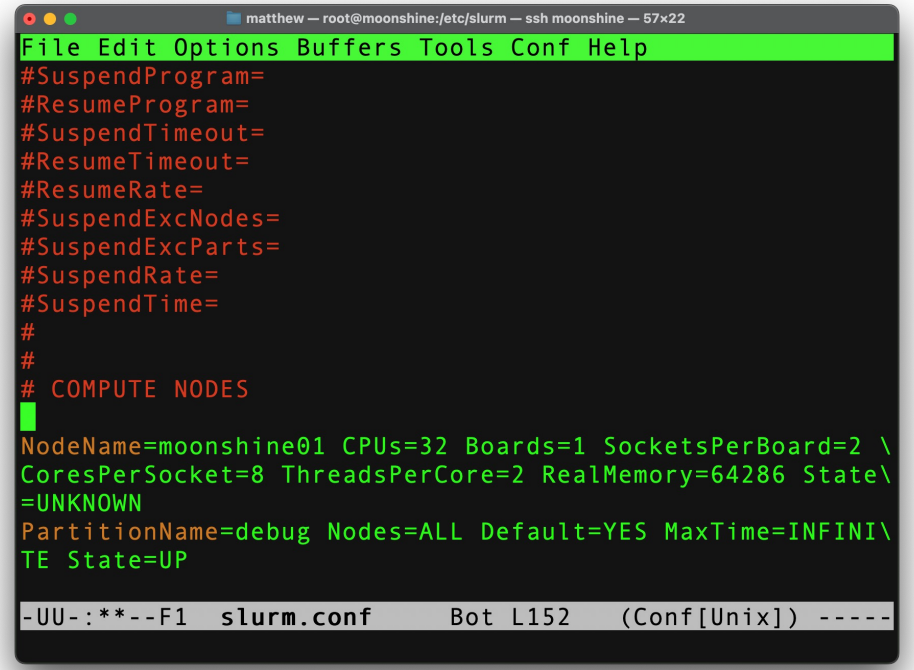
Import slurm.conf into Compute Node Image

```
[root@moonshine slurm]# wwctl overlay import --parents wwinit /etc/slurm/slurm.conf
Building overlay for moonshine01: [wwinit]
Created image for overlay moonshine01/[wwinit]:
/var/lib/warewulf/provision/overlays/moonshine01/wwinit.img
Compressed image for overlay moonshine01/[wwinit]:
/var/lib/warewulf/provision/overlays/moonshine01/wwinit.img.gz
[root@moonshine slurm]# cat /var/lib/warewulf/overlays/wwinit/rootfs/etc/slurm/slurm.conf
wwctl overlay build
Building system overlays for moonshine01: [wwinit]
Created image for overlay moonshine01/[wwinit]:
/var/lib/warewulf/provision/overlays/moonshine01/__SYSTEM__.img
Compressed image for overlay moonshine01/[wwinit]:
/var/lib/warewulf/provision/overlays/moonshine01/__SYSTEM__.img.gz
Building runtime overlays for moonshine01: [generic]
Created image for overlay moonshine01/[generic]:
/var/lib/warewulf/provision/overlays/moonshine01/__RUNTIME__.img
Compressed image for overlay moonshine01/[generic]:
/var/lib/warewulf/provision/overlays/moonshine01/__RUNTIME__.img.gz
```

Get Compute Node Hardware Info

```
[root@moonshine01 ~]# slurmd -C
NodeName=moonshine01 CPUs=32 Boards=1
SocketsPerBoard=2 CoresPerSocket=8
ThreadsPerCore=2 RealMemory=64286
UpTime=0-00:08:32
```





Add Node Info to the Slurm Config

Update the slurm.conf in Warewulf

```
[root@moonshine slurm]# wwctl overlay del wwinit /etc/slurm/slurm.conf
[root@moonshine slurm]# wwctl overlay import wwinit /etc/slurm/slurm.conf
Building overlay for moonshine01: [wwinit]
Created image for overlay moonshine01/[wwinit]:
/var/lib/warewulf/provision/overlays/moonshine01/wwinit.img
Compressed image for overlay moonshine01/[wwinit]:
/var/lib/warewulf/provision/overlays/moonshine01/wwinit.img.gz
[root@moonshine slurm]# wwctl overlay build
```

Create Cgroup slurm config

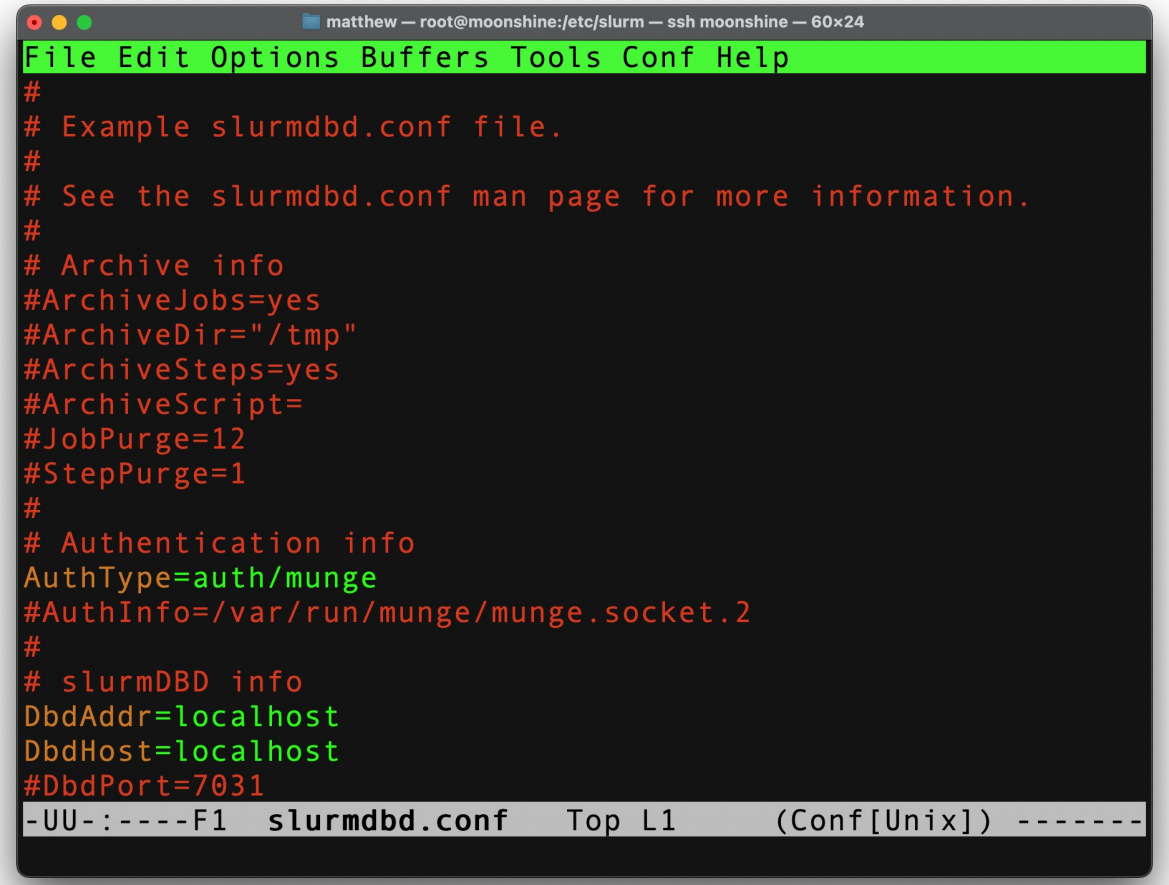
```
[root@moonshine slurm]# cp cgroup.conf.example cgroup.conf
[root@moonshine slurm]# cat cgroup.conf
###
#
# Slurm cgroup support configuration file
#
# See man slurm.conf and man cgroup.conf for further
# information on cgroup configuration parameters
#--
ConstrainCores=yes
ConstrainDevices=yes
ConstrainRAMSpace=yes
ConstrainSwapSpace=yes
```

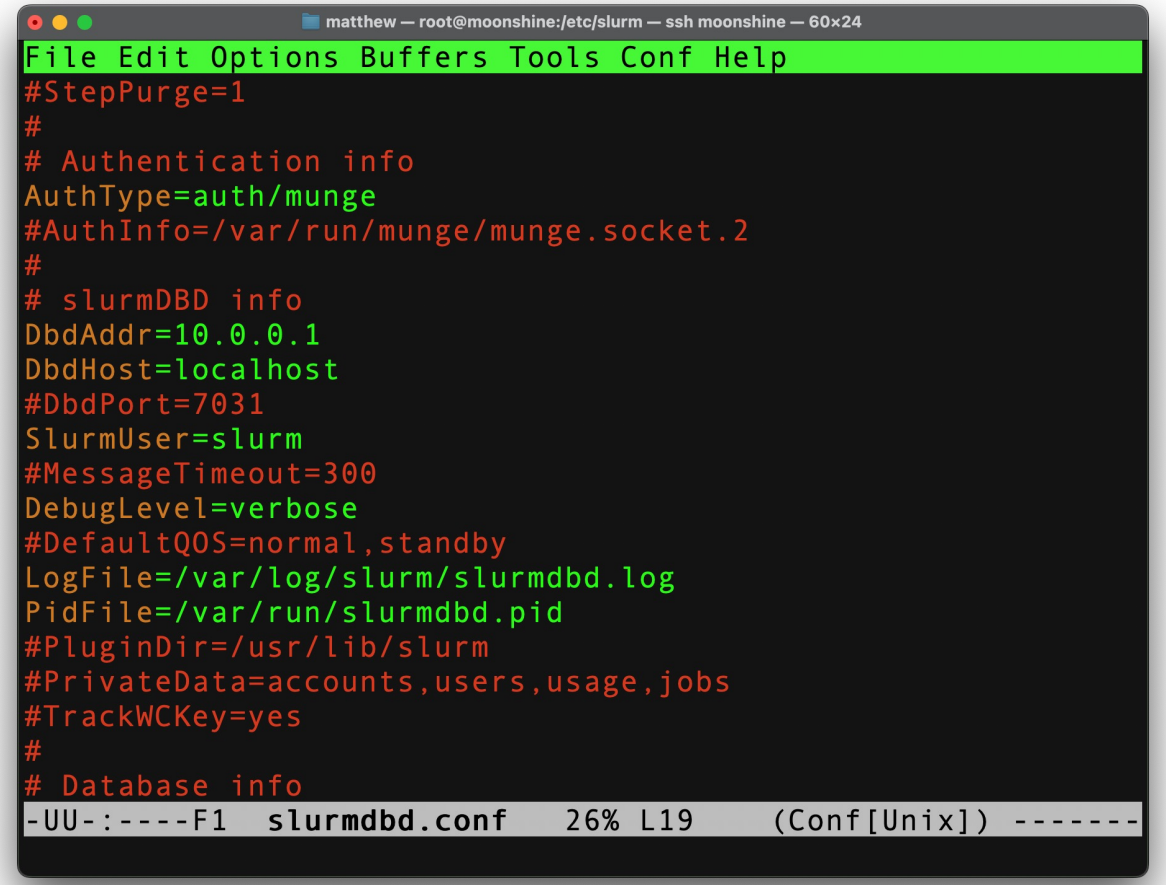
Create Cgroup slurm config add to overlay

```
[root@moonshine slurm]# wwctl overlay import wwinit /etc/slurm/cgroup.conf
Building overlay for moonshine01: [wwinit]
Created image for overlay moonshine01/[wwinit]:
/var/lib/warewulf/provision/overlays/moonshine01/wwinit.img
Compressed image for overlay moonshine01/[wwinit]:
/var/lib/warewulf/provision/overlays/moonshine01/wwinit.img.gz
[root@moonshine slurm]# wwctl overlay build
Building system overlays for moonshine01: [wwinit]
Created image for overlay moonshine01/[wwinit]:
/var/lib/warewulf/provision/overlays/moonshine01/__SYSTEM__.img
Compressed image for overlay moonshine01/[wwinit]:
/var/lib/warewulf/provision/overlays/moonshine01/__SYSTEM__.img.gz
Building runtime overlays for moonshine01: [generic]
Created image for overlay moonshine01/[generic]:
/var/lib/warewulf/provision/overlays/moonshine01/__RUNTIME__.img
Compressed image for overlay moonshine01/[generic]:
/var/lib/warewulf/provision/overlays/moonshine01/__RUNTIME__.img.gz
```

Setup Database for Slurm to Use

```
[root@moonshine slurm]# emacs /etc/slurm/slurmdbd.conf
```





Configure Slurm DB

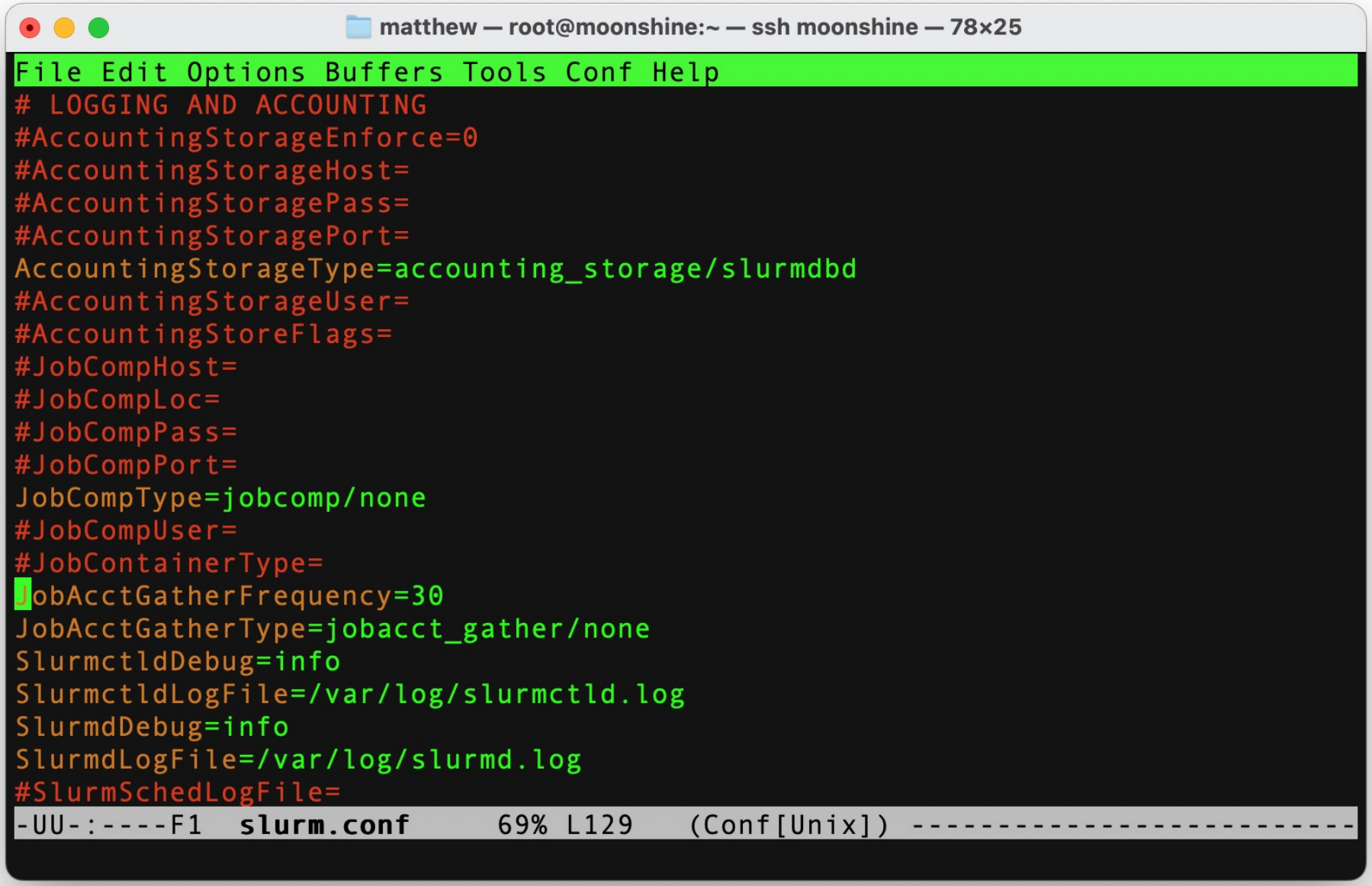
Configure Slurm Database

```
[root@moonshine slurm]# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 13
Server version: 10.5.22-MariaDB MariaDB Server


Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.


Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.


MariaDB [(none)]>
```

Configure Slurm Database

```
MariaDB [(none)]> grant all on slurm_acct_db.* TO 'slurm'@'localhost'
identified by 'password' with grant option;
Query OK, 0 rows affected (0.016 sec)


MariaDB [(none)]> create database slurm_acct_db;
Query OK, 1 row affected (0.000 sec)


MariaDB [(none)]> ^DBye
```

Enable Accounting in slurm.conf



Now we can bring up our compute node

```
[root@moonshine ~]# sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug*       up   infinite      1  down* moonshine01

[root@moonshine ~]# scontrol update nodename=moonshine01 state=idle

[root@moonshine ~]# sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug*       up   infinite      1  idle* moonshine01
```

Testing a Slurm Batch Script

- Now we are ready to test Slurm.

Create a simple Slurm script that prints the name of the computer it ran on and runs the “uptime” function. Let’s call the script test.slurm.

```
#!/bin/bash
#SBATCH --job-name test
hostname
uptime
```

Add Compute Node Addresses to the Trusted Zone

```
[matthew@moonshine ~]$ cat test.slurm
#!/bin/bash
#SBATCH --job-name test
hostname
uptime

[matthew@moonshine ~]$ sbatch test.slurm
Submitted batch job 20
[matthew@moonshine ~]$ cat slurm-20.out
moonshine01
 12:01:15 up 1 day,  8:30,  0 users,  load average: 0.00, 0.00, 0.00
```

Enabling Interactive Sessions

- Srun executes programs interactively.
- This requires that the compute nodes be able to send information back to the head node so the user can see the program output in real time.
- The ports used in the socket connection (recall that a socket is the combination of port and IP address) are randomly generated.
- There might be lots of these socket connections.
- For expediency (and this is the standard method) we will add all compute nodes IP addresses to the “trusted” firewall daemon zone on the Head Node.
- The trusted zone allows connections on all ports.

Add Compute Node Addresses to the Trusted Zone

```
[matthew@moonshine ~]$ sudo firewall-cmd --zone=trusted --add-source=10.0.0.0/24

[matthew@moonshine ~]$ sudo firewall-cmd --zone=trusted --list-all
trusted (active)
  target: ACCEPT
  icmp-block-inversion: no
  interfaces:
  sources: 10.0.0.0/24
  services:
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

Now we can use srun

```
[matthew@moonshine ~]$ srun --pty bash
[matthew@moonshine01 ~]$ hostname
moonshine01
[matthew@moonshine01 ~]$ exit
exit
```

Congratulations

- Now you have a basic functioning Slurm system.