Slurm installation and upgrading

Contents

- Slurm installation and upgrading
  - Hardware optimization for the slurmctld master server
  - Create global user accounts
  - Slurm authentication plugin
  - Munge authentication service
    - Install the latest Munge version
    - Munge 0.5.13 only: Increase number of threads
    - Munge configuration and testing
  - Build Slurm RPMs
    - Install prerequisites
      - Optional prerequisites
    - Install MariaDB database
    - Build Slurm packages
      - Build Slurm with optional features
  - Installing RPMs
    - Configure Slurm logging
  - Upgrading Slurm
    - Upgrade of MySQL/MariaDB
    - Make a dry run database upgrade
    - Upgrading on EL8 and EL9
      - Upgrade slurmdbd
      - Upgrade slurmctld
      - Install slurm-libpmi
      - Upgrade MPI applications
      - Upgrade slurmd on nodes
  - Migrate the slurmctld service to another server
    - Migrate slurmctld version <= 23.02
  - Log file rotation

Jump to our top-level Slurm page: Slurm batch queueing system

To get started with Slurm see the Slurm_Quick_Start Administrator Guide. See also CECI Slurm Quick Start Tutorial.

Hardware optimization for the slurmctld master server

SchedMD recommends that the slurmctld server should have only a few, but **very fast CPU cores**, in order to ensure the best responsiveness.

The file system for `/var/spool/slurmctld/` should be mounted on the fastest possible disks (SSD or NVMe if possible).

Create global user accounts

There must be a uniform user and group name space (including UIDs and GIDs) across the cluster, see the Slurm_Quick_Start Administrator Guide.

It is not necessary to permit user logins to the control hosts (*ControlMachine* or *BackupController*), but the users and groups must be configured on those hosts. To restrict user login by SSH, use the `AllowUsers` parameter in sshd_config.

Slurm and Munge require consistent UID and GID across **all servers and nodes in the cluster**, including the `slurm` and `munge` users.

Very important: Avoid UID and GID values below 1000, as defined in the standard configuration file `/etc/login.defs` by the parameters `UID_MIN, UID_MAX, GID_MIN, GID_MAX`, see also the User_identifier page.

Create the users/groups for `slurm` and `munge`, for example:

```
export MUNGEUSER=1005
groupadd -g $MUNGEUSER munge
useradd  -m -c "MUNGE Uid 'N' Gid Emporium" -d /var/lib/munge -u $MUNGEUSER -g munge  -s
/sbin/nologin munge
export SlurmUSER=1001
groupadd -g $SlurmUSER slurm
useradd  -m -c "Slurm workload manager" -d /var/lib/slurm -u $SlurmUSER -g slurm  -s /bin/bash
slurm
```

Important: Make sure that these same users are **created identically on all nodes**. User/group creation must be done **prior to installing RPMs** (which would create random UID/GID pairs if these users don't exist).

Please note that UIDs and GIDs up to 1000 are currently reserved for EL Linux system users, see this article and the file `/etc/login.defs`.

Slurm authentication plugin

For an overview of authentication see the Authentication_Plugins page. Beginning with version 23.11, Slurm has its own plugin that can create and validate credentials. It validates that the requests come from legitimate UIDs and GIDs on other hosts with matching users and groups.

Munge authentication service

For an overview of authentication see the Authentication_Plugins page. The Munge authentication plugins identifies the user originating a message. You should read the Munge_installation guide and the Munge_wiki.

The EL8 and EL9 distributions contain Munge RPM packages version 0.5.13, but it is preferred to install a later version as discussed below. The in-distro packages may be installed by:

```
dnf install munge munge-libs munge-devel
```

On busy servers such as the slurmctld server, the munged daemon could become a bottleneck, see the presentation *Field Notes 5: From The Frontlines of Slurm Support* in the Slurm_publications page. On busy servers it is recommended to increase the number of threads, see the munged manual page, however, this is the default in the latest Munge_release. The issue is discussed in excessive logging of: "Suspended new connections while processing backlog".

Install the latest Munge version

We recommend to install the latest Munge_release RPMs (currently 0.5.16) due to new features and bug fixes. Build RPM packages by:

```
wget https://github.com/dun/munge/releases/download/munge-0.5.16/munge-0.5.16.tar.xz
rpmbuild -ta munge-0.5.16.tar.xz
```

and install them from the directory `~/rpmbuild/RPMS/x86_64/`.

With Munge 0.5.16 a configuration file `/etc/sysconfig/munge` is now used by the *munge* service, and you may for example add this configuration to increase the number of threads to 10:

```
OPTIONS="--key-file=/etc/munge/munge.key --num-threads=10"
```

Munge prior to version 0.5.15 has an issue_94 *excessive logging of: "Suspended new connections while processing backlog"* which might cause the *munged.log* file to **fill up the system disk**.

See also the page Configure maximum number of open files where it is **highly recommended** to increase the `fs.file-max` limit in `/etc/sysctl.conf` significantly on **all Slurm compute nodes**.

Munge 0.5.13 only: Increase number of threads

Only in case you have decided to use the **default** EL8/EL9 Munge version 0.5.13, this version does not honor an options file, see Let systemd unit file use /etc/sysconfig/munge for munge options.

You can increase the number of threads in *munged* as follows. Copy the Systemd unit file:

```
cp /usr/lib/systemd/system/munge.service /etc/systemd/system/munge.service
```

See Modify systemd unit file without altering upstream unit file. Then edit this line in the copied unit file:

```
ExecStart=/usr/sbin/munged --num-threads 10
```

and restart the *munge* service:

```
systemctl daemon-reload
systemctl restart munge
```

Munge configuration and testing

You may check the *munged* log file `/var/log/munge/munged.log` for any warnings or errors.

By default Munge uses an AES AES-128 cipher and SHA-256 HMAC (*Hash-based Message Authentication Code*). Display these encryption options by:

```
munge -C
munge -M
```

On the **Head node (only)** create a secret key to be used globally on every node (see the Munge_installation guide):

```
dd if=/dev/urandom bs=1 count=1024 > /etc/munge/munge.key
chown munge: /etc/munge/munge.key
chmod 400 /etc/munge/munge.key
```

Alternatively use this command (slow):

```
/usr/sbin/create-munge-key -r
```

NOTE: For a discussion of using `/dev/random` in stead of `/dev/urandom` (pseudo-random) as recommended in the Munge_installation guide, see Myths about /dev/urandom.

Securely propagate `/etc/munge/munge.key` (e.g., via SSH) to all other hosts within the same security realm:

```
scp -p /etc/munge/munge.key hostXXX:/etc/munge/munge.key
```

Make sure to set the correct ownership and mode on all nodes:

```
chown -R munge: /etc/munge/ /var/log/munge/
chmod 0700 /etc/munge/ /var/log/munge/
```

Then enable and start the Munge service on all nodes:

```
systemctl enable munge
systemctl start  munge
```

Run some **tests** as described in the Munge_installation guide:

```
munge -n
munge -n | unmunge          # Displays information about the Munge key
munge -n | ssh somehost unmunge
remunge
```

Build Slurm RPMs

Read the Slurm_Quick_Start Administrator Guide, especially the section below this text:

```
Optional Slurm plugins will be built automatically when the configure script detects that the
required build requirements are present.
Build dependencies for various plugins and commands are denoted below:
```

You must decide which Slurm plugins to activate in the RPM packages which you build, especially items such as:

- cgroup Task Affinity
- cgroup_v2 plugin (requires *dbus-devel* and *libbpf* libraries)
- Munge support
- Lua Support
- PAM support
- NUMA Affinity

Install prerequisites

You will need to enable the repositories `PowerTools` (EL8) or `CRB` (EL9), see the discussion of Rocky_Linux_Repositories, and then enable also the EPEL repository:

```
dnf config-manager --set-enabled powertools # EL8
dnf config-manager --set-enabled crb        # EL9
dnf install epel-release
dnf clean all
```

Slurm can be built with a number of **optional plugins**, each of which has some prerequisite library. The Slurm_Quick_Start guide lists these in the section Building and Installing Slurm.

Install required Slurm prerequisite packages, as well as several optional packages that enable the desired Slurm plugins:

```
dnf install mariadb-server mariadb-devel
dnf install rpm-build gcc python3 openssl openssl-devel pam-devel numactl numactl-devel hwloc
hwloc-devel lua lua-devel readline-devel rrdtool-devel ncurses-devel gtk2-devel libibmad
libibumad perl-Switch perl-ExtUtils-MakeMaker xorg-x11-xauth dbus-devel libbpf bash-completion
```

We recommend to install_the_latest_munge_version (currently 0.5.16) because of bug fixes improving the EL8/EL9 version:

```
dnf install munge munge-libs munge-devel
```

Install the following packages from EPEL:

```
dnf install libssh2-devel man2html
```

Optional prerequisites

Certain Slurm tools and plugins require additional prerequisites **before** building Slurm RPM packages:

1. IPMI library: If you want to implement power saving as described in the Power_Saving_Guide then you must install the FreeIPMI development library prerequisite:

   ```
   dnf install freeipmi-devel
   ```

   See the presentation *Saving Power with Slurm by Ole Nielsen* in the Slurm_publications page, and the section on Building IPMI power monitoring into Slurm.

2. If you want to build the **Slurm REST API** daemon named slurmrestd, then you must install these prerequisites also:

   ```
   dnf install http-parser-devel json-c-devel libjwt-devel
   ```

   Notice: The minimum version requirements are listed in the rest_quickstart guide:

   - HTTP Parser (>= v2.6.0),
   - LibYAML (optional, >= v0.2.5),
   - JSON-C (>= v1.12.0).

   See the presentation *Slurm's REST API by Nathan Rini, SchedMD* in the Slurm_publications page. You may like to install the jq - Command-line JSON processor also:

   ```
   dnf install jq
   ```

3. For EL9 only: Enable YAML command output (for example, `sinfo --yaml`) by installing the `libyaml-devel` library:

   - **Important**: The *libyaml* **must** be version >= 0.2.5, see bug_17673, and EL9 provides this version. The *libyaml* provided by EL8 is version 0.1.X and **should not be used**!

Install MariaDB database

First install the MariaDB database version 10.3:

```
dnf install mariadb-server mariadb-devel
```

NOTICE: Do not forget to configure the database as described in the Slurm database page!

If you plan to use Ansible to manage the database, it will require this Python package:

```
dnf install python3-mysql (EL8)
dnf install python3-PyMySQL (EL9)
```

Build Slurm packages

Get the Slurm source code from the Slurm_download page.

Set the version (for example, 24.05.4 and build Slurm RPM packages by:

```
export VER=24.05.4
rpmbuild -ta slurm-$VER.tar.bz2 --with mysql
```

Notes about the `--with mysql` option:

- The `--with mysql` option is not strictly necessary because the `slurm-slurmdbd` package will be built by default, but using this option will catch the scenario where your forgot to install the `mariadb-devel` packages as described above, see also bug_8882 and this mailing list posting.
- From Slurm 23.11 the `--with mysql` option has been removed, see the NEWS file. The default behavior now is to always require one of the sql development libraries.

The RPM packages will typically be found in `$HOME/rpmbuild/RPMS/x86_64/` and should be installed on all relevant nodes.

Build Slurm with optional features

You may build Slurm packages including optional features:

- If you want to implement power saving as described in the Power_Saving_Guide then you can ensure that FreeIPMI gets built in by adding:

```
rpmbuild <...> --with freeipmi
```

This will be available from Slurm 23.11 where the presense of the `freeipmi-devel` package gets verified, see bug_17900.

- If you want to build the **Slurm REST API** daemon named slurmrestd you must add:

```
rpmbuild <...> --with slurmrestd
```

- For EL9 only: Enable YAML command output (for example, `sinfo --yaml`):

```
rpmbuild <...> --with yaml
```

Notice that *libyaml* version 0.2.5 or later is required (see above), and this is only available starting with EL9, so the `--with yaml` option should **not** be used on EL8 and older releases!

Installing RPMs

Study the configuration information in the Quick Start Administrator_Guide. The RPMs to be installed on the head node, compute nodes, and slurmdbd node can vary by configuration, but here is a suggested starting point:

- **Head** node where the slurmctld daemon runs:

```
export VER=24.05.4
dnf install slurm-$VER*rpm slurm-devel-$VER*rpm slurm-perlapi-$VER*rpm slurm-
torque-$VER*rpm slurm-example-configs-$VER*rpm
systemctl enable slurmctld
```

The following must be done on the Head node because the RPM installation does not include this. Create the spool and log directories and make them owned by the slurm user:

```
mkdir /var/spool/slurmctld /var/log/slurm
chown slurm: /var/spool/slurmctld /var/log/slurm
chmod 755 /var/spool/slurmctld /var/log/slurm
```

Create log files:

```
touch /var/log/slurm/slurmctld.log
chown slurm: /var/log/slurm/slurmctld.log
```

Servers which should offer slurmrestd should install also this package:

```
dnf install slurm-slurmrestd-$VER*rpm
```

The *slurm-torque* package could perhaps be omitted, but it does contain a useful `/usr/bin/mpiexec` wrapper script.

- On **Compute nodes** install slurmd and possibly also the *slurm-pam_slurm* RPM package to prevent rogue users from logging in:

```
export VER=24.05.4
dnf install slurm-slurmd-$VER*rpm slurm-pam_slurm-$VER*rpm
systemctl enable slurmd
```

The following must be done on each compute node because the RPM installation does not include this. Create the slurmd spool and log directories and make the correct ownership:

```
mkdir /var/spool/slurmd /var/log/slurm
chown slurm: /var/spool/slurmd  /var/log/slurm
chmod 755 /var/spool/slurmd  /var/log/slurm
```

Create log files:

```
touch /var/log/slurm/slurmd.log
chown slurm: /var/log/slurm/slurmd.log
```

You may consider this RPM as well with special PMIx libraries:

```
dnf install slurm-libpmi-$VER*rpm
```

- **Database** (slurmdbd service) node:

```
export VER=24.05.4
dnf install slurm-$VER*rpm slurm-devel-$VER*rpm slurm-slurmdbd-$VER*rpm
```

Create the slurmdbd log directory and log file, and make the correct ownership and permissions:

```
mkdir /var/log/slurm
touch /var/log/slurm/slurmdbd.log
chown slurm: /var/log/slurm /var/log/slurm/slurmdbd.log
chmod 750 /var/log/slurm
chmod 640 /var/log/slurm/slurmdbd.log
```

Explicitly enable the slurmdbd service:

```
systemctl enable slurmdbd
```

- On **Login nodes** install these packages:

```
export VER=24.05.4
dnf install slurm-$VER*rpm slurm-devel-$VER*rpm slurm-contribs-$VER*rpm slurm-
perlapi-$VER*rpm
```

Configure Slurm logging

The Slurm logfile directory is undefined in the RPMs since you have to define it in slurm.conf. See *SlurmdLogFile* and *SlurmctldLogFile* in the slurm.conf page, and *LogFile* in the slurmdbd.conf page.

Check your logging configuration with:

```
# grep -i logfile /etc/slurm/slurm.conf
SlurmctldLogFile=/var/log/slurm/slurmctld.log
SlurmdLogFile=/var/log/slurm/slurmd.log
# scontrol show config | grep -i logfile
SlurmctldLogFile        = /var/log/slurm/slurmctld.log
SlurmdLogFile           = /var/log/slurm/slurmd.log
SlurmSchedLogFile       = /var/log/slurm/slurmsched.log
```

If log files are configured, you have to create the log file directory manually:

```
mkdir /var/log/slurm
chown slurm.slurm /var/log/slurm
```

See the more general description in Bug_8272.

Upgrading Slurm

New Slurm updates are released about every 6 months (the interval was 9 months prior to Slurm 24.05). Follow the Upgrades instructions in the Slurm_Quick_Start page, see also presentations by Tim Wickberg in the Slurm_publications page. Pay attention to these statements:

- You may upgrade at most by 2 major versions (3 versions starting from 24.11), see the Upgrades page.
- When changing the version to a higher release number (e.g from 22.05.x to 23.02.x) **always** upgrade the slurmdbd daemon first.
- Be mindful of your configured `SlurmdTimeout` and `SlurmctldTimeout` values: Increase/decrease them as needed.
- The recommended upgrade order is that versions may be mixed as follows:

  ```
  slurmdbd >= slurmctld >= slurmd >= commands
  ```

  Actually, the term "commands" here primarily refers to the **login nodes**, because all Slurm commands (sinfo , squeue etc.) are **not interoperable** with an older slurmctld version, as explained in bug_17418, due to RPC changes! It is OK to upgrade Slurm on login nodes **after** slurmctld has been upgraded. The slurmd on compute nodes can be upgraded over a period of time, and older slurmd versions will continue to work with an upgraded slurmctld, although it is recommended to upgrade as soon as possible.

- The following command can report current jobs that have been orphaned on the local cluster and are now runaway:

  ```
  sacctmgr show runawayjobs
  ```

Regarding the Slurm database, also make sure to:

- Make a database dump (see Slurm database) prior to the slurmdbd upgrade.
- Start the slurmdbd service manually after the upgrade in order to avoid timeouts (see bug_4450). In stead of starting the slurmdbd Systemd service, it **strongly recommended to start the slurmdbd daemon manually**. If you use the `systemctl` command, it is very likely to **exceed a system time limit** and kill slurmdbd before the database conversion has been completed!

  The recommended way to perform the slurmdbd database upgrade is therefore:

  ```
  time slurmdbd -D -vvv
  ```

See further info below.

Upgrade of MySQL/MariaDB

If you restore a database dump (see Slurm database) onto a different server running a **newer MySQL/MariaDB version**, there are some extra steps.

See Upgrading from MySQL to MariaDB about running the mysql_upgrade command:

```
mysql_upgrade
```

whenever major (or even minor) version upgrades are made, or when migrating from MySQL to MariaDB.

It may be necessary to restart the `mysqld` service or reboot the server after this upgrade (??).

Make a dry run database upgrade

Optional but strongly recommended: You can test the database upgrade procedure before doing the real upgrade.

In order to verify and time the slurmdbd database upgrade you may make a dry_run upgrade for testing before actual deployment.

Here is a suggested procedure:

1. Drain a compute node running the **current** Slurm version and use it for testing the database.
2. Install the database RPM packages and configure the database **EXACTLY** as described in the Slurm database page:

   ```
   dnf install mariadb-server mariadb-devel
   ```

   Configure the MySQL/MariaDB database as described in the Slurm database page.
3. Copy the latest database dump file (`/root/mysql_dump`, see Slurm database) from the main server to the compute node. Load the dump file into the testing database:

   ```
   time mysql -u root -p < /root/mysql_dump
   ```

   If the dump file is in some compressed format:

```
time zcat mysql_dump.gz | mysql -u root -p
time bzcat mysql_dump.bz2 | mysql -u root -p
```

The MariaDB/MySQL *password* will be asked for. Reading in the database dump may take **many minutes** depending on the size of the dump file, the storage system speed, and the CPU performance. The `time` command will report the time usage.

Verify the database contents on the compute node by making a new database dump and compare it to the original dump.

4. Select a suitable *slurm* user's **database password**. Now follow the Slurm accounting page instructions (using -p to enter the database password):

```
# mysql -p
grant all on slurm_acct_db.* TO 'slurm'@'localhost' identified by 'some_pass' with grant
option;  ### WARNING: change the some_pass
SHOW GRANTS;
SHOW VARIABLES LIKE 'have_innodb';
create database slurm_acct_db;
quit;
```

WARNING: Use the *slurm* database user's password **in stead of** `some_pass`.

5. The following actions must be performed on the drained compute node.

   First stop the regular slurmd daemons on the compute node:

```
systemctl stop slurmd
```

   Install the **OLD** (the cluster's current version, say, NN.NN) additional slurmdbd database RPMs as described above:

```
VER=NN.NN
dnf install slurm-slurmdbd-$VER*rpm
```

   Information about building RPMs is in the Slurm installation and upgrading page.

6. Make sure that the `/etc/slurm` directory exists (it is not needed in configless Slurm clusters):

```
$ ls -lad /etc/slurm
drwxr-xr-x. 5 root root 4096 Feb 22 10:12 /etc/slurm
```

Copy the configuration files from the main server to the compute node:

```
/etc/slurm/slurmdbd.conf
/etc/slurm/slurm.conf
```

Important: Edit these files to replace the database server name by `localhost` so that all further actions take place on the compute node, **not** the *real* database server.

Configure this in `slurmdbd.conf`:

```
DbdHost=localhost
StorageHost=localhost
StoragePass=<slurm database user password>  # See above
```

and configure this in `slurm.conf`:

```
AccountingStorageHost=localhost
```

Set up files and permissions:

```
chown slurm: /etc/slurm/slurmdbd.conf
chmod 600 /etc/slurm/slurmdbd.conf
touch /var/log/slurm/slurmdbd.log
chown slurm: /var/log/slurm/slurmdbd.log
```

7. Make sure that slurmdbd is running, and start it if necessary:

```
systemctl status slurmdbd
systemctl start slurmdbd
```

Make some query to test slurmdbd:

```
sacctmgr show user -s
```

If all is well, stop the slurmdbd before the upgrade below:

```
systemctl stop slurmdbd
```

8. At this point you have a Slurm database server running an exact copy of your main Slurm database!

   Now it is time to do some testing. Update all Slurm RPMs to the new version (say, 24.05.4 built as shown above):

   ```
   export VER=24.05.4
   dnf update slurm*$VER*.rpm
   ```

   Optional: In case you use the auto_tmpdir RPM package, you have to remove it first because it will block the Slurm upgrade, see also Temporary job directories.

9. Perform and time the actual database upgrade:

   ```
   time slurmdbd -D -vvv
   ```

   and wait for the output:

   ```
   slurmdbd: debug2: accounting_storage/as_mysql: as_mysql_roll_usage: Everything rolled up
   ```

   and do a Control-C. Please note that the database table conversions may take **several minutes** or longer, depending on the size of the tables.

   Write down the timing information from the `time` command, since this will be the expected approximate time when you later perform the *real* upgrade. However, the storage system performance is important for all database operations, so timings may vary substantially between servers.

   Now start the service as usual:

   ```
   systemctl start slurmdbd
   ```

10. Make some query to test slurmdbd:

    ```
    sacctmgr show user -s
    ```

    and make some other tests to verify that slurmdbd is responding correctly.

11. When all tests have been completed successfully, reinstall the compute node to its default installation.

Upgrading on EL8 and EL9

Let's assume that you have built the updated RPM packages for EL8 or EL9 and copied them to the current directory so you can use `dnf` commands on the files directly.

Upgrade slurmdbd

The upgrading steps for the slurmdbd host are:

1. Stop the slurmdbd service:

```
systemctl stop slurmdbd
```

2. Make a dump of the MySQL/Mariadb database (see Slurm database).
3. Update all RPMs:

```
export VER=24.05.4
dnf update slurm*$VER*.rpm
```

4. Start the slurmdbd service **manually** after the upgrade in order to avoid Systemd timeouts (see bug_4450). In stead of starting the slurmdbd service with `systemctl`, it is most likely necessary to **start the daemon manually**. If you were to use the `systemctl` command, it is very likely to **exceed a system time limit** and kill slurmdbd before the database conversion has been completed.

   Perform and time the actual database upgrade:

```
time slurmdbd -D -vvv
```

   The completion of the database conversion may be printed with text like:

```
slurmdbd: debug2: accounting_storage/as_mysql: as_mysql_roll_usage: Everything rolled up
```

   Then stop slurmdbd with a Control-C. Please note that the database table conversions may take **a number of minutes** or longer, depending on the size of the database tables.

5. Now start the slurmdbd service normally:

```
systemctl start slurmdbd
```

6. Make some database query to test slurmdbd:

```
sacctmgr show user -s
```

WARNING: Newer versions of user commands like `sinfo`, `squeue` etc. are **not interoperable** with an older slurmctld version, as explained in bug_17418, due to RPC changes!

Upgrade slurmctld

The upgrading steps for the slurmctld host are:

1. Change the timeout values in slurm.conf to:

   ```
   SlurmctldTimeout=3600
   SlurmdTimeout=3600
   ```

   and copy `/etc/slurm/slurm.conf` to all nodes (not needed in configless Slurm clusters). Then reconfigure the running daemons and test the timeout and `StateSaveLocation` values:

   ```
   scontrol reconfigure
   scontrol show config | grep Timeout
   scontrol show config | grep StateSaveLocation
   ```

2. Stop the slurmctld service:

   ```
   systemctl stop slurmctld
   ```

3. Make a backup copy of the `StateSaveLocation` (check your configuration first) `/var/spool/slurmctld` directory:

- Check the size of the `StateSaveLocation` and the backup destination to ensure there is sufficient disk space:

```
du -sm /var/spool/slurmctld/
df -h $HOME
```

- Then make a tar-ball backup file:

```
tar cf $HOME/var.spool.slurmctld.tar /var/spool/slurmctld/*
```

- Make sure the contents of the tar-ball file look correct:

```
less $HOME/var.spool.slurmctld.tar
```

4. Upgrade the RPMs, for example:

```
export VER=24.05.4
dnf update slurm*$VER-*.rpm
```

5. Enable and restart the slurmctld service:

```
systemctl enable slurmctld
systemctl restart slurmctld
```

6. Check the cluster nodes' health using `sinfo` and check for any `Nodes ... not responding` errors in `slurmctld.log`. It may be necessary to restart all the `slurmd` on all nodes, for example, using the clush command (see the Slurm batch queueing system page about ClusterShell):

```
clush -ba systemctl restart slurmd
```

7. Restore the previous timeout values in slurm.conf (item 1.).

Note: The compute nodes should be upgraded at your earliest convenience.

Install slurm-libpmi

On the compute nodes, only, you may consider this RPM as well with special PMIx libraries:

```
dnf install slurm-libpmi-$VER*rpm
```

Upgrade MPI applications

MPI applications such as **OpenMPI** may be linked against the `/usr/lib64/libslurm.so` library. In this context you must understand the remark in the Upgrades page:

```
The libslurm.so version is increased every major release.
So things like MPI libraries with Slurm integration should be recompiled.
Sometimes it works to just symlink the old .so name(s) to the new one, but this has no
guarantee of working.
```

In the thread Need for recompiling openmpi built with –with-pmi? it has been found that:

```
It looks like it is the presence of lib64/libpmi2.la and lib64/libpmi.la that is the
"culprit". They are installed by the slurm-devel RPM.
Openmpi uses GNU libtool for linking, which finds these files, and follow their
"dependency_libs" specification, thus linking directly to libslurm.so.
```

Slurm version 16.05 and later no longer installs the libpmi*.la files. This should mean that if your OpenMPI was built against Slurm 16.05 or later, there should be no problem (we think), but otherwise you probably must rebuild your MPI applications and install them again at the same time that you upgrade the slurmd on the compute nodes.

To check for the presence of the “bad” files, go to your software build host and search:

```
locate libpmi2.la
locate libpmi.la
```

TODO: Find a way to read relevant MPI libraries like this example:

```
readelf -d libmca_common_pmi.so
```

Upgrade slurmd on nodes

First determine which Slurm version the nodes are running, for example, using the clush command (see the Slurm batch queueing system page about ClusterShell):

```
clush -bg <partition> slurmd -V
```

The **quick and usually OK procedure** would be to simply update the RPMs (here: version 24.05.4 on all nodes:

```
clush -bw <nodelist> 'dnf -y update /some/path/slurm*24.05.4*.rpm'
```

This would automatically restart and enable slurmd on the nodes without any loss of running batch jobs.

For the compute nodes running slurmd the **safe procedure** could be:

1. Drain all desired compute nodes in a <nodelist>:

   ```
   scontrol update NodeName=<nodelist> State=draining Reason="Upgrading slurmd"
   ```

   Nodes will change from the *DRAINING* to the *DRAINED* state as the jobs are completed. Check which nodes have become *DRAINED*:

   ```
   sinfo -t drained
   ```

2. Stop the slurmd daemons on compute nodes:

   ```
   clush -bw <nodelist> systemctl stop slurmd
   ```

3. Update the RPMs (here: version 24.05.4 on nodes:

   ```
   clush -bw <nodelist> 'dnf -y update /some/path/slurm*24.05.4*.rpm'
   ```

   and make sure to install also the new `slurm-slurmd` and `slurm-contribs` packages.

   Now enable the slurmd service:

   ```
   clush -bw <nodelist> systemctl enable slurmd
   ```

4. For restarting slurmd there are two alternatives:

   a. Restart slurmd or simply reboot the nodes in the *DRAINED* state:

```
clush -bw <nodelist> systemctl daemon-reload
clush -bw <nodelist> systemctl restart slurmd
  or simply reboot:
clush -bw <nodelist> shutdown -r now
```

b. Reboot the nodes automatically as they become idle using the **RebootProgram** as configured in slurm.conf, see the scontrol **reboot** option and explanation in the man-page:

```
scontrol reboot [ASAP] [NodeList]
```

5. Return upgraded nodes to the IDLE state:

```
scontrol update NodeName=<nodelist> State=resume
```

Finally, restore the timeout values in slurm.conf to their defaults, for example:

```
SlurmctldTimeout=600
SlurmdTimeout=300
```

and copy `/etc/slurm/slurm.conf` to all nodes (not needed in configless Slurm clusters). Then reconfigure the running daemons:

```
scontrol reconfigure
```

Again, consult the Upgrades page before you start!

Migrate the slurmctld service to another server

It may be required to migrate the slurmctld service to another server, for example, when a major OS version update is needed, or when the server must be migrated to another hardware.

From Slurm 23.11 and later, migrating the slurmctld service is quite easy, and **does not** require to stop all running jobs, since a major improvement is stated in the Release notes:

- *Update slurmstepd processes with current SlurmctldHost settings, allowing for controller changes without draining all compute jobs.*

This change allows slurmstepd to receive an updated `SlurmctldHost` setting so that running jobs will report back to the new controller when they finish. See the Slurm_publications presentation `Slurm 23.02, 23.11, and Beyond` by Tim Wickberg, SchedMD. Notice, however, that slurmd ignores any changes in slurm.conf or the DNS SRV_record (i.e., when running a Configless Slurm setup):

- When slurmd is started, it caches its configuration files as is discussed in bug_20462.
- Therefore it is required to restart slurmd on all compute notes after modifying slurm.conf and the DNS SRV_record (if applicable).

The slurmctld migration process for Slurm 23.11 and later does **not** require to stop all running jobs, and the details are discussed in bug_20070 .

We have successfully performed a slurmctld migration following this procedure:

1. On the old `SlurmctldHost` server change the timeout values in slurm.conf to a high value:

   ```
   SlurmctldTimeout=3600
   SlurmdTimeout=3600
   ```

   and make an `scontrol reconfigure`.

2. Stop and disable the slurmctld service on the old `SlurmctldHost` server:

   ```
   <old-server>: systemctl stop slurmctld
   <old-server>: systemctl disable slurmctld
   ```

3. Copy all Slurm configuration files `/etc/slurm/*.conf` from the old server to the **new** `SlurmctldHost` server. Also make sure the Slurm logfile directory exists and has correct ownership (see Configure Slurm logging):

   ```
   mkdir -pv /var/log/slurm
   touch /var/log/slurm/slurmctld.log
   chown -R slurm.slurm /var/log/slurm
   ```

4. In configless Slurm clusters update the DNS SRV_record, see Configless Slurm setup
5. Migrate slurmctld to new machine: Make a tar-ball copy or rsync the `StateSaveLocation` directory (typically `/var/spool/slurmctld`) to the new server, for example:

   ```
   <old-server>: $ rsync -aq /var/spool/slurmctld/ <new-server>:/var/spool/slurmctld/
   ```

Make sure the permissions allow the *SlurmUser* to read and write the folder!

6. **Remember** to update slurm.conf with the new `SlurmctldHost` name, and remember to update the login nodes as well!
7. Start and enable the slurmctld service on the new server:

```
systemctl start slurmctld
systemctl enable slurmctld
```

8. As discussed in bug_20462 it is necessary to restart slurmd on **all compute nodes** so they can pick up the new `SlurmctldHost` value in slurm.conf. For example, use the clush command (see the Slurm batch queueing system page about ClusterShell):

```
clush -ba systemctl restart slurmd
```

9. When everything is working correctly, restore the timeout values in slurm.conf to their defaults, for example:

```
SlurmctldTimeout=600
SlurmdTimeout=300
```

   and make a `scontrol reconfigure`.

If **not** using Configless Slurm setup you must distribute slurm.conf manually to all nodes in step 4.

Migrate slurmctld version <= 23.02

In Slurm 23.02 and older, changes to `SlurmctldHost` are not possible while jobs are running on the system. Therefore you have to **stop all running jobs**, for example by making a Resource Reservation. Read the FAQ How should I relocate the primary or backup controller? with the procedure:

- Stop all Slurm daemons.
- Modify the `SlurmctldHost` values in the slurm.conf file.
- Distribute the updated slurm.conf file to all nodes. When using Configless Slurm setup see the section above.
- Copy the `StateSaveLocation` directory to the new host and make sure the permissions allow the SlurmUser to read and write it.
- Restart all Slurm daemons.

Log file rotation

The Slurm log files may be stored in the `/var/log/slurm` directory, and they may grow rapidly on a busy system. Especially the `slurmctld.log` file on the controller machine may grow very large.

Therefore you probably want to configure logrotate to administer your log files. On RHEL Linux and clones the logrotate configuration files are in the `/etc/logrotate.d/` directory.

Manual configuration of logging is required because the SchedMD RPM files do not contain the logrotate setup, see bug_3904, bug_2215, and bug_4393. See also the section *LOGGING* at the end of the slurm.conf page with an example logrotate script.

First install the relevant RPM:

```
dnf install logrotate
```

Create the following script `/etc/logrotate.d/slurm` which will rotate and compress the slurmctld log file on a weekly basis:

```
/var/log/slurm/*.log {
      compress
      missingok
      nocopytruncate
      nodelaycompress
      nomail
      notifempty
      noolddir
      rotate 5
      sharedscripts
      size=5M
      create 640 slurm root
      postrotate
              pkill -x --signal SIGUSR2 slurmctld
              pkill -x --signal SIGUSR2 slurmd
              pkill -x --signal SIGUSR2 slurmdbd
              exit 0
      endscript
}
```

Warning: Do **not** run `scontrol reconfig` or restart slurmctld to rotate the log files, since this will incur a huge overhead.

See the NEWS file for changes related to SIGUSR2:

```
Modify all daemons to re-open log files on receipt of SIGUSR2 signal. This is much than using
SIGHUP to re-read the configuration file and rebuild various tables.
```