GWDG
GWDG
Timon Vogt

Exercise 1: Building a Slurm cluster

In this exercise, you will use your provided Cloud-VMs to build your own Slurm cluster on them.
Please make sure to read all instructions carefully!

Contents

Task 1: Preparing the Slurm installation (5 min)

1. First, designate one of your nodes as the head node. This node will host the `slurmctld` and the `slurmdbd`.
2. Ensure that you have SSH access to the head node and become `root` on it.
   - You can check your own username using the command `whoami`.
   - If you are not `root` yet, become `root` using the command `sudo -i`.
3. Also ensure that you can access your other nodes via SSH from the head-node.
4. As we will need to download stuff from the Internet, ensure that your head node has a floating IP set, so it can access the Internet.
5. Using your knowledge from the previous lecture, ensure that the head node is hosting a NFS share that is accessible from all other nodes. Ensure that the NFS share is **on all nodes** mounted to the same path.
6. Create the Slurm installation directory structure inside the NFS share:

```
slurm
├── el7
│   ├── dependencies
│   └── slurm
│       └── 22.05
│           ├── build
│           ├── install
│           └── source
└── etc
```

Task 2: Installing Munge (15 min)

Munge is the authentication plugin used by Slurm. It needs to be installed and configured before we can install Slurm.

1. First, enable the `powertools` repository in `yum` to get access to the munge development libraries with this command. Do this **on the head node and inside the warewulf image of your cluster nodes**:

   ```
   yum config-manager --set-enabled powertools
   ```

2. Use the following commands to create a `MUNGE` user and a `munge` group on each node of the cluster:

   ```
   export MUNGEUSER=1001
   groupadd -g $MUNGEUSER munge
   useradd  -m -c "MUNGE Uid 'N' Gid Emporium" -d /var/lib/munge \
   -u $MUNGEUSER -g munge  -s /sbin/nologin munge
   ```

   Repeat these commands on the head node as well as inside the warewulf image of the cluster nodes. With these commands, you ensure that each the `MUNGE` user has the same `uid` and `gid` on each node of the cluster.

3. Next, install Munge and its devel libraries on the head node. Use the CentOS package manager `yum` for that. The packages that need to be installed (on the head node): `munge munge-libs munge-devel`

4. The Munge daemon itself needs to be present on all nodes of the cluster. Install the following binary inside the warewulf image of the nodes as well: `munge`

5. Munge daemons are using a shared key, which needs to be the same across the cluster. On the head node, use the following command to create a new `munge` key:

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

   This key is then located on the head node in the path `/etc/munge/munge.key`. Copy it into the warewulf image of the cluster nodes and move it into the same directory.

6. For security reasons, Munge is very picky about access permissions on its directories. Use the following commands **on the head node as well as inside the warewulf image of the cluster nodes** to ensure the access permissions are set correctly:

   ```
   chown -R munge: /etc/munge/ /var/log/munge/ /var/lib/munge/ /run/munge/
   chmod 0700 /etc/munge/ /var/log/munge/ /var/lib/munge/
   chmod 0711 /run/munge/
   ```

7. After all permissions are set correctly, you can enable the `munge` daemons **on the head node as well as inside the warewulf image of the cluster nodes** with the following command:

   ```
   systemctl enable munge
   ```

8. Next, build the new warewulf images and reboot the worker nodes of the cluster to load the new image.

9. On the head node, start the munge daemon with the following command:

   ```
   systemctl start munge
   ```

10. Ensure that the munge daemons have started correctly, **on all nodes of the cluster**, using the following command:

    ```
    systemctl status munge
    ```

    If `munge` did not start, fix any occuring errors and restart it.

Task 3: Installing MariaDB (5 min)

1. **On the head node**, MariaDB is required as a database for `slurmdbd` to store job states and meta-data. Therefore, before Slurm can be installed, MariaDB needs to be installed and configured **on the head node**. Install the MariaDB server with `yum`. The packages are called: `mariadb-server` and `mariadb-devel`.

2. After the installation completes, use the following commands to start the server and do a "secure" basic MariaDB configuration:

   ```
   systemctl enable mariadb
   systemctl start mariadb
   mysql_secure_installation
   ```

   Choose a random password for the root access to the database. Write it down, as we will need it later. On all other questions, simply choose yes.

3. While probably not necessary for this tutorial, it is "best-practice" to configure MariaDB with larger default storage sizes. Otherwise, in high-demand clusters, Slurm might become very slow, when a lot of jobs are running in parallel. Create the `/etc/my.cnf.d/innodb.cnf` configuration file with the following content:

   ```
   [mysqld]
    innodb_buffer_pool_size=1024M
    innodb_log_file_size=64M
    innodb_lock_wait_timeout=900
   ```

4. Use the following commands to implement the changes:

   ```
   systemctl stop mariadb
   mv /var/lib/mysql/ib_logfile? /tmp/
   systemctl start mariadb
   ```

Task 4: Installing the Slurm dependencies (10 min)

To build Slurm, some dependencies need to be installed first.

1. The first set of dependencies can be installed using the regular CentOS package manager `yum`. Use `yum` to install the following dependencies for Slurm, **on your head node as well as inside the warewulf image of your cluster nodes**:

   - gcc
   - gcc-c++
   - tar
   - make
   - lbzip2
   - python3
   - openssl
   - openssl-devel
   - pam-devel

- numactl
- numactl-devel
- hwloc
- hwloc-devel
- lua
- lua-devel
- readline-devel
- rrdtool-devel
- ncurses-devel
- man2html
- libibmad
- libibumad
- libevent
- libevent-devel

2. Build your warewulf image and reboot your worker nodes.
3. Some other dependencies can not be installed via `yum` since they are either not available in the package registries or the available version is too old. Those dependencies then need to be compiled by hand. For this course, we will do so only for a single dependency: `PMIX`.

   a) Use the `wget` tool to download the source code from `PMIX` for version 3.1.2. Use this url to download the `.tar.gz` file: `https://github.com/openpmix/openpmix/releases/download/v3.1.2/pmix-3.1.2.tar.gz` Then use the `sha1sum` tool to create a checksum of the downloaded file and compare it to the checksum listed on this page: `https://github.com/openpmix/openpmix/releases/tag/v3.1.2`

   b) Create an installation directory structure, similar to the slurm installation directory, in the `dependencies` folder of the slurm directory. The folder structure should look like this:

```
slurm
├── el7
│   ├── dependencies
│   │   └── pmix
│   │       └── 3.1.2
│   │           ├── build
│   │           ├── install
│   │           └── source
│   └── slurm
...
```

   c) Untar the source code into the `source` directory of the `pmix/3.1.2` folder. Use the flag: `--strip-components=1`.

   d) To build `PMIX`, first, enter the build directory and then run the configure command from the source directory like this:

```
../source/configure
↪  --prefix=<path-to-the-nfs-share>/slurm/el7/dependencies/pmix/3.1.2/install
```

e) After `configure` is completed, run the `make -j` and `make install` commands to install `PMIX`

Task 5: Building Slurm (5 min)

Now that we have the dependencies installed, we can begin to build Slurm.

1. First, downloaded the Slurm source code using the `wget` tool. Use the following url: `https://download.schedmd.com/slurm/slurm-22.05.8.tar.bz2`
2. Next, compare the checksum of the downloaded file (again using the `sha1sum` tool) against the provided checksum on the page: `https://www.schedmd.com/downloads.php`
3. Untar the source code into the `source` directory of the `slurm/22.05/sources` folder on the NFS share. Again use the flag `--strip-components=1`.
4. Enter the build directory and run the configure command with the following flags:

```
../source/configure --prefix=<path-to-the-nfs-share>/slurm/el7/slurm/22.05/install
        --sysconfdir=<path-to-the-nfs-share>/slurm/etc
        --with-pmix=<path-to-the-nfs-share>/slurm/el7/dependencies/pmix/3.1.2/install/
```

5. Next, build Slurm by running the `make -j` and afterwards the `make install` command.

Task 6: Configuring Slurm (20 min)

After Slurm has been build onto your cluster, it now needs to be configured.

1. First create the Slurm user **on your head node and in the warewulf image of the nodes of your cluster**, with these commands:

```
export SLURMUSER=1002
groupadd -g $SLURMUSER slurm
useradd  -m -c "SLURM workload manager" -d /var/lib/slurm -u $SLURMUSER -g slurm  -s
↪  /bin/bash slurm
mkdir /var/log/slurm
chown slurm:slurm /var/log/slurm
```

   On your head node, also create the slurm `spool` directory with the following commands:

```
mkdir /var/spool/slurmctld
chown slurm:slurm /var/spool/slurmctld
```

2. **On the head node**, create a configuration file for the `slurmdbd` in the path `<path-to-the-nfs-share>/slurm/etc/slurmdbd.conf` with the following content:

```
AuthType=auth/munge
DbdAddr=192.168.1.1
DbdHost=localhost
SlurmUser=slurm
DebugLevel=4
LogFile=/var/log/slurm/slurmdbd.log
PidFile=/var/run/slurmdbd.pid
StorageType=accounting_storage/mysql
StorageHost=localhost
StoragePass=password
StorageUser=slurm
```

```
StorageLoc=slurm_acct_db
```

Change the IP address at the key `DbdAddr` to the IP adress of your head node. Afterwards, run the following commands:

```
chown slurm:slurm <path-to-your-nfs-share>/slurm/etc/slurmdbd.conf
chmod 600 <path-to-your-nfs-share>/slurm/etc/slurmdbd.conf
```

3. On the head node, create the `slurm_acct_db` database, by running the command `mysql -u root -p` , enter the MariaDB root password you have noted down earlier and then, inside the interactive MariaDB shell, enter the following commands:

```
grant all on slurm_acct_db.* TO 'slurm'@'localhost' identified by 'password' with
↪  grant option;
create database slurm_acct_db;
```

   Use Ctrl+D to exit out of the interactive MariaDB session.

4. Using wget, download the following files from the GWDG owncloud service onto your head node. These files are templates for the configuration files, which need to be edited before they can be used:

   - `wget https://owncloud.gwdg.de/index.php/s/MOObbDMtyAtDEAy/download`
     This file is the `service` file for the `slurmctld`. Change the nfs share paths inside it from `opt` to the path where your nfs share is mounted and move the file to `/etc/systemd/system/slurmctld.service`
   - `wget https://owncloud.gwdg.de/index.php/s/zdmJVrapcC74PAy/download`
     This file is the `service` file for the `slurmdbd`. Change the nfs share paths inside it from `opt` to the path where your nfs share is mounted and move the file to `/etc/systemd/system/slurmdbd.service`
   - `wget https://owncloud.gwdg.de/index.php/s/81TTgfQJuXa8oUu/download`
     This file is the `service` file for the `slurmd`. Change the nfs share paths inside it from `opt` to the path where your nfs share is mounted and move the file to `/etc/systemd/system/slurmd.service` **inside the warewulf image of your cluster nodes**.
   - `wget https://owncloud.gwdg.de/index.php/s/x5w3kM8fDprDcUb/download`
     This file is the `cgroup.conf` configuration file. Move it to the path `<path-to-your-nfs-share/slurm/etc/cgroup.conf`. You dont need to edit it.
   - `wget https://owncloud.gwdg.de/index.php/s/uHg5MdHmtZM1JmF/download`
     This file is a template for the `slurm.conf` configuration file. Move it to the path `<path-to-your-nfs-share>/slurm/etc/slurm.conf`. You dont need to edit it for now, we will edit it in a later step.

5. With the files in place, you can start the `slurmdbd` by running the following commands:

```
systemctl daemon-reload
systemctl enable slurmdbd
systemctl start slurmdbd
```

6. Now, edit the `slurm.conf` file. The only thing we need to add is the **name** of your head node. You can get this name using the `hostname` command on your head node. After getting the name, change the following value in the file `<path-to-your-nfs-share>/slurm/etc/slurm.conf`:

   - `SlurmctldHost`: Change this value to the **name** of your head node.

7. Afterwards, on each of your worker nodes, execute the following command:

```
<path-to-your-nfs-share>/slurm/el7/slurm/22.05/install/sbin/slurmd -C
```

   Copy the first line of the result, containing `NodeName`, `CPUs`, etc keys and add it to the bottom of your

`slurm.conf`. In that line then also add the key `NodeAddr=` with the IP adress of the node.

Task 7: Starting Slurm (5 min)

1. After you have build your `slurm.conf` you can start the `slurmctld` by running the following commands:

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

2. Once the `slurmctld` runs, go into the warewulf image and activate the `slurmd` service with this command:

   ```
   systemctl enable slurmd
   ```

   Then, rebuild your warewulf image and reboot the worker nodes of the cluster.

3. Now that the `slurmctld` runs, you can test your new Slurm cluster by submitting its first job:

   ```
   export PATH=$PATH:<path-to-your-nfs-share>/slurm/el7/slurm/22.05/install/bin/
   sinfo
   srun -N2 hostname
   ```

Congratulations, you now have a functioning Slurm cluster.