Date Published: 07/08/2021

Student Cluster Competition - Tutorial 3

Table of Contents

Overview

This tutorial demonstrates *cluster monitoring* and *workload scheduling*. These two components are critical to a typical HPC environment. Monitoring is a widely used component in system administration (including enterprise datacentres and corporate networks). Monitoring allows administrators to be aware of what is happening on any system that is being monitored and is useful to proactively identify where any potential issues may be. A workload scheduler ensures that users' jobs are handled properly to fairly balance all scheduled jobs with the resources available at any time.

In this tutorial you will:

- [ ] Install the Zabbix monitoring server on your head node.
- [ ] Install Zabbix monitoring clients on your compute node(s).
- [ ] Configure Zabbix in order to monitor your virtual HPC cluster.
- [ ] Install the Slurm workload manager across your cluster.
- [ ] Submit a test job to run on your cluster through the newly-configured workload manager.

```
Date Published: 07/08/2021
```

Part 1 - Monitoring With Zabbix

Zabbix is an open source solution for network and hardware monitoring. Zabbix includes various graphing and visualisation options, which allow you to quickly assess the utilisation and efficiency of a fleet of computers. This can also allow you to spot potential issues that may emerge, such as a node in your cluster no longer being operational for whatever reason.

Install And Configure The Zabbix Server On The Head Node

Zabbix Server Setup

The Zabbix server runs on the head node of your cluster and will manage the collection of information from other computers. It also takes care of organising the incoming data and can display it via a website.

1. Install the Zabbix repository from the Zabbix website directly by invoking the following command:

   ```
   [...@headnode ~]$ sudo rpm -Uvh \
                       https://repo.zabbix.com/zabbix/5.0/rhel/8/x86_64/zabbix-release-5.0-1.el8.noarch.rpm
   [...@headnode ~]$ sudo dnf clean all
   ```

2. Once you have the repository installed you can proceed to install the Zabbix server, agent and web support:

   ```
   [...@headnode ~]$ sudo dnf -y install zabbix-server-mysql zabbix-web-mysql zabbix-apache-conf zabbix-agent
   ```

3. Zabbix utilises a SQL database for its configuration. We can install a free and commonly used SQL database called MariaDB using the following command:

   ```
   [...@headnode ~]$ sudo dnf install mariadb-server
   ```

4. **Start** and **enable** the service `mariadb` server

5. Configure the MariaDB database.

   ```
   [...@headnode ~]$ sudo mysql_secure_installation
   ```

   You will be presented with a series of prompts:

Prompt	Your Response
Enter current password for root (enter for none):	Press enter key
Set root password? [Y/n]	Y
New password:	Specify <rootDBpass\>
Re-enter new password:	Repeat <rootDBpass\>
Remove anonymous users? [Y/n]	Y
Disallow root login remotely? [Y/n]	Y

Prompt	Your Response
Remove test database and access to it? [Y/n]	Y
Reload privilege tables now? [Y/n]	Y

- `<rootDBpass>` is a password that you create and is used to enter MariaDB with administrative permissions.

6. Zabbix needs to know which database to speak to inside of MariaDB, and currently none exist. Create the appropriate database entries that Zabbix expects to use:

```
[...@headnode ~]$ mysql -uroot -p'<rootDBpass>' -e "create database zabbix character set utf8 collate utf8_bin;"
[...@headnode ~]$ mysql -uroot -p'<rootDBpass>' -e \
                    "grant all privileges on zabbix.* to zabbix@localhost identified by '<zabbixDBpass>';"
```

- `<zabbixDBpass>` is a password that you create and is used by Zabbix to connect to the specific database that you created for it.

7. For the database that you have created for the Zabbix server, there must be certain database schemas defined in the SQL database that you created for it. A *database schema* indicates which fields are included on each table in the database, as well as stipulating which tables or relations make up the database. Once these schemata are defined, the Zabbix server is able to read and write the appropriate data entries into the database according to the format that it expects. These schema definitions are not inside of your database installation by default, but are provided by Zabbix. To load them into the database, run the following command:

```
[...@headnode ~]$ zcat /usr/share/doc/zabbix-server-mysql/create.sql.gz | \
                    mysql -uzabbix -p'<zabbixDBpass>' zabbix
```

> **! >>> This operation may take some time to finish. Please be patient.**

8. Configure Zabbix to use the correct password that you specified to `<zabbixDBpassword>` in your database. Edit the file `/etc/zabbix/zabbix_server.conf` and change the following line:

```
DBPassword=<zabbixDBpassword>
```

You may need to uncomment (remove the leading "#" and space) the `DBPassword` entry.

9. Zabbix needs to have some *SELinux* policies enabled in order for it to run correctly. Install the following packages:

```
[...@headnode ~]$ sudo dnf install policycoreutils checkpolicy setroubleshoot-server
```

Next create a new directory within your home directory and then create and edit a file called `zabbix_server_add.te` (in the example below, `nano` can be replaced with your preferred text editor):

```
[...@headnode ~]$ mkdir -p ~/zabbix-selinux
[...@headnode ~]$ cd ~/zabbix-selinux/
[...@headnode ~]$ nano zabbix_server_add.te
```

In this file, paste the following text:

```
module zabbix_server_add 1.1;

require {
        type zabbix_var_run_t;
        type tmp_t;
        type zabbix_t;
        class sock_file { create unlink write };
        class unix_stream_socket connectto;
        class process setrlimit;
        class capability dac_override;
}

#============= zabbix_t ==============

allow zabbix_t self:process setrlimit;
allow zabbix_t self:unix_stream_socket connectto;
allow zabbix_t tmp_t:sock_file { create unlink write };
allow zabbix_t zabbix_var_run_t:sock_file { create unlink write };
allow zabbix_t self:capability dac_override;
```

This is a policy that will allow Zabbix to work correctly. It allows the Zabbix server process to do things like create network connections and create files in privileged system locations.

Please read more about SELinux here if you are interested.

We now need to load it into SELinux:

```
[...@headnode ~]$ checkmodule -M -m -o zabbix_server_add.mod zabbix_server_add.te
[...@headnode ~]$ semodule_package -m zabbix_server_add.mod -o zabbix_server_add.pp
[...@headnode ~]$ sudo semodule -i zabbix_server_add.pp
```

And lastly, we need to specify some other SELinux rules to be enabled.

These rules are defined using SELinux Booleans, and in particular we want to:

- allow HTTPD scripts and modules to connect to the network
- allow HTTP daemon to connect to Zabbix
- allow Zabbix to connect to unreserved ports

```
[...@headnode ~]$ sudo setsebool -P httpd_can_network_connect 1
[...@headnode ~]$ sudo setsebool -P httpd_can_connect_zabbix 1
[...@headnode ~]$ sudo setsebool -P zabbix_can_network on
```

10. **Start** and **enable** the `zabbix-server` and `zabbix-agent` services on the head node.

11. Since we've already allowed HTTP and HTTPS traffic through the firewall in a pervious tutorial, we don't need to do it again.

Zabbix Web Interface Setup

System Configuration

1. Uncomment the following line in `/etc/php-fpm.d/zabbix.conf` :

```
; php_value[date.timezone] = Europe/Riga
```

and change it to:

```
php_value[date.timezone] = Africa/Johannesburg
```

2. Restart the Apache web server (`httpd`) and start `php-fpm` as well. Make sure they're both set to start at boot as well (**enable** them.).

3. You should now be able to access the web frontend of the Zabbix server. To do so, create an SSH tunnel as shown in the previous tutorial and visit the appropriate web address with `/zabbix` appended to it. For example, with a dynamic tunnel:

```
https://headnode.cluster.scc/zabbix
```

If everything is configured properly, you will be prompted with the setup screen for Zabbix. Perform the following actions:

Web Interface Configuration

1. On the "Welcome" screen, press `Next step`.

2. On the "Check of pre-requisites" screen, press `Next step`.

3. On the "Configure DB connection" screen, enter the following:

Setting	Answer
Database type	MySQL
Database host	localhost
Database port	0
Database name	zabbix
User	zabbix
Password	<zabbixDBpass>

4. On the "Zabbix server details" screen, change "Host" to your **full head node name** and press `Next step`.

5. On the "Install" screen, press `Next step`.

6. You will be presented with a screen that says "**Congratulations! You have successfully installed Zabbix frontend.**" Here you can press `Finish`.

7. The default Zabbix web UI login details are as follows: `username: Admin`, `password: zabbix`. Log in and change the password.

Install And Configure The Zabbix Agent On The Compute Nodes

Zabbix can monitor systems with or without software installed on the monitored host machine, however, using a Zabbix agent provides a much richer monitoring service (such as CPU load, network utilization, disk space, etc.). For this reason, we will be installing a Zabbix agent on all host machines we wish to monitor.

1. On each of your nodes, add the Zabbix repository (you've already done this on the head node.).

2. Install the Zabbix agent on each of the nodes:

```
[...@computenode ~]$ sudo dnf -y install zabbix-agent
```

3. Edit the file `/etc/zabbix/zabbix_agentd.conf` on all nodes and change the following lines:

```
Server=<head_node_hostname>
ServerActive=<head_node_hostname>>
Hostname=<compute_node_hostname>
```

4. **Start** and **enable** the `zabbix-agent` service

5. Create the internal firewall zone for the compute node and set the firewall to allow all internal traffic. This allows the Zabbix server to get information from the agent on the compute node. We are only allowing all internal traffic, because we did it in the previous tutorial for the head node. **Again, this is not recommended in real, production environments! We're just doing this here for simplicity.**

```
[...@computenode ~]$ sudo nmcli c mod <internal_interface> connection.zone internal
[...@computenode ~]$ sudo firewall-cmd --permanent --zone=internal --set-target=ACCEPT
[...@computenode ~]$ sudo firewall-cmd --reload
```

Monitoring Your Infrastructure

You will need to tunnel into the cluster to get HTTP access as you did in the beginning of the second tutorial to continue.

1. Visit http://10.128.24.xx/zabbix with your browser (remember you are using a tunnel) and log in.

2. Change the default password by clicking "User settings" at the bottom left and selecting `Change password`.

3. Navigate to `Configuration > Hosts` and enable your headnode for monitoring (this may take a minute to update).

4. To check that your monitoring is working, navigate to `Monitoring > Hosts > Zabbix server (head node) > Graphs`. If the graphs have data then Zabbix is configured to monitor the node.

5. Add your compute node by going to `Configuration > Hosts > Create host`.

   i. On the "Host" page, fill in the appropriate details. Enter the host name (DNS), assign it to the "Linux Servers" group and specify its IP address and full host name under Interfaces.
   ii. On the "Templates" page, click `Select` and in the "Host group" search bar enter `Templates/Operating Systems` and select "Template OS Linux by Zabbix agent".
   iii. Under "Inventory", select `Automatic`.
   iv. You can click `Add` to add the host.

6. You can create your own custom graph to display multiple properties from both of your nodes.

!!! Take a Screenshot of this, save it in `png` format and upload it into your teams private Gitlab repository under the `Zabbix` folder.

Date Published: 07/08/2021

Part 2 - Slurm Workload Manager

The Slurm Workload Manager (formerly known as Simple Linux Utility for Resource Management), is a free and open-source job scheduler for Linux, used by many of the world's supercomputers/computer clusters. It allows you to manage the resources of a cluster by deciding how users get access for some duration of time so they can perform work. To find out more, please visit the Slurm Website.

Prerequisites

1. Make sure the clocks, i.e. chrony daemons, are synchronized across the cluster.

2. Generate a SLURM and MUNGE user on all of your nodes:

   - **If you have FreeIPA authentication working**
     - Create the users using the FreeIPA web interface. **Do NOT add them to the sysadmin group**.
   - **If you do NOT have FreeIPA authentication working**
     - `useradd slurm`
     - Ensure that users and groups (UIDs and GIDs) are synchronized across the cluster. Read up on the appropriate /etc/shadow and /etc/password files.

Server Setup

1. Install the MUNGE package. MUNGE is an authentication service that makes sure user credentials are valid and is specifically designed for HPC use.

   First, we will enable the **EPEL** *(Extra Packages for Enterprise Linux)* repository for `dnf`, which contains extra software that we require for MUNGE and Slurm:

   ```
   [...@headnode ~]$ sudo dnf install epel-release
   ```

   Then we can install MUNGE, pulling the development source code from the `powertools` repository:

   ```
   [...@headnode ~]$ sudo dnf --enablerepo=powertools install munge munge-libs munge-devel
   ```

2. Generate a MUNGE key for client authentication:

   ```
   [...@headnode ~]$ sudo /usr/sbin/create-munge-key -r
   [...@headnode ~]$ sudo chown munge:munge /etc/munge/munge.key
   [...@headnode ~]$ sudo chmod 600 /etc/munge/munge.key
   ```

3. Using `scp`, copy the MUNGE key to your compute node to allow it to authenticate:

   i. SSH into your compute node and create the directory `/etc/munge`. Then exit back to the head node.

   ii. `scp /etc/munge/munge.key <compute_node_name_or_ip>:/etc/munge/munge.key`

4. **Start** and **enable** the `munge` service

5. Install dependency packages:

Date Published: 07/08/2021

```
[...@headnode ~]$ sudo dnf --enablerepo=powertools install python3 gcc openssl openssl-devel pam-devel numactl \
                    numactl-devel hwloc lua readline-devel ncurses-devel man2html libibmad libibumad \
                    rpm-build perl-ExtUtils-MakeMaker rrdtool-devel lua-devel hwloc-devel \
                    perl-Switch libssh2-devel mariadb-devel
[...@headnode ~]$ sudo dnf groupinstall "Development Tools"
```

6. Download the 20.11.9 version of the Slurm source code tarball (.tar.bz2) from https://download.schedmd.com/slurm/. Copy the URL for `slurm-20.11.9.tar.bz2` from your browser and use the `wget` command to easily download files directly to your VM.

7. Environment variables are a convenient way to store a name and value for easier recovery when they're needed. Export the version of the tarball you downloaded to the environment variable VERSION. This will make installation easier as you will see how we reference the environment variable instead of typing out the version number at every instance.

```
[...@headnode ~]$ export VERSION=20.11.9
```

8. Build RPM packages for Slurm for installation

```
[...@headnode ~]$ sudo rpmbuild -ta slurm-$VERSION.tar.bz2
```

   This should successfully generate Slurm RPMs in the directory that you invoked the `rpmbuild` command from.

9. Copy these RPMs to your compute node to install later, using `scp`.

10. Install Slurm server

```
[...@headnode ~]$ sudo mkdir -p /var/log/slurm /var/spool/slurm/ctld /var/spool/slurm/d
[...@headnode ~]$ sudo chown -R slurm:slurm /var/log/slurm /var/spool/slurm/ctld /var/spool/slurm/d
[root@headnode ~]$ cd /root/rpmbuild/RPMS/x86_64/
[root@headnode ~]$ dnf localinstall slurm-$VERSION*.rpm slurm-devel-$VERSION*.rpm \
                     slurm-example-configs-$VERSION*.rpm slurm-perlapi-$VERSION*.rpm \
                     slurm-torque-$VERSION*.rpm slurm-slurmctld-$VERSION*.rpm
```

11. Setup Slurm server

```
[...@headnode ~]$ sudo cp /etc/slurm/slurm.conf.example /etc/slurm/slurm.conf
```

   Edit this file (`/etc/slurm/slurm.conf`) and set appropriate values for:

```
ClusterName=      #Name of your cluster (whatever you want)
ControlMachine=   #DNS name of the head node
```

   Populate the nodes and partitions at the bottom:

```
NodeName=<computenode> Sockets=<num_sockets> CoresPerSocket=<num_cpu_cores> ThreadsPerCore=<num_threads_per_core>
State=UNKNOWN PartitionName=debug Nodes=ALL Default=YES MaxTime=INFINITE State=UP
```

   To check how many cores your compute node has, run `lscpu` on the compute node. You will get output including `CPU(s)`, `Thread(s) per core`, `Core(s) per socket` and more that will help you determine what to use for the Slurm configuration.

Hint: if you overspec your compute resources in the definition file then Slurm will not be able to use the nodes.

12. **Start** and **enable** the `slurmctld` service on the head node.

Client Setup

1. Setup MUNGE:

```
[...@computenode ~]$ sudo dnf install munge munge-libs
[...@computenode ~]$ sudo chown -R munge:munge /etc/munge/munge.key
```

2. **Start** and **enable** the `munge` service.

3. Test MUNGE to the head node:

```
[...@computenode ~]$ munge -n | ssh headnode.cluster.scc unmunge
[...@computenode ~]$ remunge
```

4. The `.rpm` files that were previously created on the head node must be made available to the compute node. Either copy the `.rpm` files to the compute node or move them to an NFS shared directory on your head node so that they are accessible on the compute node. Next, install the Slurm client on the compute node. Once again export the version of your Slurm instance to the environment variable `$VERSION`:

```
[...@computenode ~]$ sudo dnf --enablerepo=powertools install perl-Switch perl-ExtUtils-MakeMaker
[...@computenode ~]$ sudo dnf localinstall slurm-$VERSION*.rpm slurm-devel-$VERSION*.rpm  \
                      slurm-perlapi-$VERSION*.rpm slurm-torque-$VERSION*.rpm \
                      slurm-example-configs-$VERSION*.rpm slurm-slurmd-$VERSION*.rpm \
                      slurm-pam_slurm-$VERSION*.rpm
```

5. Create the `/var/spool/slurm/d` directory and make it owned by user and group `slurm`.

6. Copy the `slurm.conf` file you editted on your head node to the same location on your compute node.

7. **Start** and **enable** the `slurmd` service.

Return to your head node. To demonstrate that your scheduler is working you can run the following command as your normal user:

```
[...@headnode ~]$ sinfo
```

You should see your compute node in an idle state.

Slurm allows for jobs to be submitted in *batch* (set-and-forget) or *interactive* (real-time response to the user) modes. Start an interactive session on your compute node via the scheduler with

```
[...@headnode ~]$ srun -N 1 --pty bash
```

You should automatically be logged into your compute node. This is done via Slurm. Re-run `sinfo` now and also run the command `squeue`. Here you will see that your compute node is now allocated to this job.

Date Published: 07/08/2021

To finish, type `exit` and you'll be placed back on your head node. If you run `squeue` again, you will now see that the list is empty.

Date Published: 07/08/2021

To confirm that your node configuration is correct, you can run the following command on the head node:

```
[...@headnode ~]$ sinfo -alN
```

The `S:C:T` column means "sockets, cores, threads" and your numbers for your compute node should match the settings that you made in the `slurm.conf` file.