Monitoring with `rsyslog` and Splunk

Overview

Monitoring for HPC is a big topic. Many larger HPC groups have whole teams dedicated to monitoring. There are many types of monitoring, many scalability issues and many different kinds of things you may want to monitor for. There are also many different tools available.

We are going to focus on monitoring through system logs. We will setup an ansible play that will aggregate system logs to the master from all of the nodes. We will then setup the (commercial) tool, Splunk, to provide us with analytics based on these system logs.

Step 1: Setting up `rsyslog`

`rsyslog` is a service that takes system log messages and writes them to the log files that we are accustomed to seeing in `/var/log`. `rsyslog` also supports re-transmitting log messages to another host.

In our clusters, we are going to tell `rsyslog` on the nodes to aggregate all of its log messages on the master. This will give us a convenient central location to diagnose and monitor all of our system logs.

Creating the rsyslog role (master)

Let's create our role structure:

```
[lowell@te-cm tasks]$ cd $HOME/ansible/roles
[lowell@te-cm roles]$ mkdir -p rsyslog/tasks
[lowell@te-cm roles]$ cd rsyslog/tasks
```

We will be following a design pattern similar to what we have used in the "Writing Ansible" guide when we created the "ntp" role. Our steps will be:

1. make sure `rsyslog` is installed
2. generate our `rsyslog.conf` file
3. make sure `rsyslog` is enabled and running

Make `tasks/main.yml` with:

```
---
- name: ensure syslog is installed
  package:
    name:
      - rsyslog
    state: present

- name: master rsyslog.conf file
  template:
    src: master-rsyslog.conf.j2
    dest: /etc/rsyslog.conf
    owner: root
    group: root
    mode: 0444
    backup: yes
  notify: restart rsyslog

- name: ensure syslog is running
  systemd:
    name: rsyslog
    state: started
    enabled: yes
```

Notice that we created a `restart rsyslog` handler. Make the file `handlers/main.yml` with:

```
---
- name: restart rsyslog
  systemd:
    name: rsyslog
    state: restarted
```

So far, this should all look very similar to the ntp role.

Now we need our template. Make the file `templates/master-rsyslog.conf.j2` with:

```
$ModLoad imuxsock # provides support for local system logging (e.g. via
logger command)
$ModLoad imjournal # provides access to the systemd journal
$ModLoad imudp
$UDPServerRun 514
$WorkDirectory /var/lib/rsyslog
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat
$IncludeConfig /etc/rsyslog.d/*.conf
$OmitLocalLogging on
$IMJournalStateFile imjournal.state
*.info;mail.none;authpriv.none;cron.none                /var/log/messages
authpriv.*                                              /var/log/secure
mail.*                                                  -/var/log/maillog
cron.*                                                  /var/log/cron
```

```
*.emerg                                                 :omusrmsg:*
uucp,news.crit                                          /var/log/spooler
local7.*                                                /var/log/boot.log
```

Notice that this is setup as a template, but it actually doesn't have any variables in it. We could have done this with the `copy` module. However, if we want to add variables to this file down the road, it will prove convient to already have it as a template. This is a common practice with files that we think may need template variables some day.

Creating the rsyslog role (node)

This covers the configuration for the master, but now we need the configuration for the node. There are several ways we could handle this, but we will handle it be directly manipulating the BOS for warewulf.

Note: this means that rsyslog must be after warewulf in the `site.yml` file.

Let's add the following tasks to the end of the `tasks/main.yml` file:

```
- name: ensure that syslog is installed in BOS
  yum:
    name: rsyslog
    state: present
    installroot: "{{ item.path }}"
  loop: "{{ cluster_bos_images }}"
  notify:
    - rebuild vnfs
    - rebuild image

- name: syslog, rsyslog.conf file
  template:
    src: compute-rsyslog.conf.j2
    dest: '{{ item.path }}/etc/rsyslog.conf'
    owner: root
    group: root
    mode: 0444
    backup: yes
  loop: '{{ cluster_bos_images }}'
  notify:
    - rebuild vnfs
```

These two require a little explanation. First of all, `cluster_bos_images` can be found in your `inventory/group_vars`. It defines information about each image you build for nodes. We actually have support for multiples, but we're only using one.

You'll notice that we used the `yum` module, and not the `package` module. The `yum` module allows us to use `installroot:`, which will work with packages in our BOS.

Finally, notice that this template gets installed inside our BOS. We don't want to add this to the warewulf synced files, because we expect this file is essentially static once setup. Instead, we "bake it in" to the image, i.e. it is

permanently resident in our VNFS.

But, since both of these tasks change the image, we need to make sure the `vnfs` gets rebuilt. The warewulf role adds the handler, `rebuild vnfs`, to make sure that happens.

Now we need the `templates/compute-rsyslog.conf.j2` template:

```
$ModLoad imuxsock # provides support for local system logging (e.g. via
logger command)
$ModLoad imjournal # provides access to the systemd journal
$WorkDirectory /var/lib/rsyslog
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat
$IncludeConfig /etc/rsyslog.d/*.conf
$OmitLocalLogging on
$IMJournalStateFile imjournal.state
*.emerg                                                 :omusrmsg:*
local7.*                                                /var/log/boot.log
{{ syslog_compute_remote_host }}
```

The format for what goes in the `syslog_compute_remote_host` line is:

```
*.* @<host_ip>:<host_port>
```

Let's setup an appropriate default in `defaults/main.yml`:

```
---
syslog_compute_remote_host: "*.* @{{ cluster_sms_ip }}:514"
```

The default port for `rsyslog` is `514`. If you look in `inventories/group_vars/cscnsi`, you'll see that `cluster_sms_ip` is already defined.

Setting dependencies and enabling the role

We noticed above that this role assumes that the `warewulf` role is also run. Ansible gives us a way to declare this dependency. To make the dependency, we need the file `meta/main.yml` in the role with:

```
---
dependencies:
- role: warewulf
```

This will ensure that anytime we enable `rsyslog` we also enable `warewulf`.

Now, let's enable our role in `site.yml`:

```
...
    - { role: warewulf, tags: [ 'warewulf' ] }
    - { role: rsyslog, tags: [ 'rsyslog' ] }
    - { role: ohpc_dev_components, tags: [ 'ohpc_dev_components', 'pe' ] }
...
```

And run it:

```
[lowell@te-cm ansible]$ ansible-playbook -u root -i inventories/hosts
site.yaml -l te-master -t rsyslog
...
PLAY RECAP
*********************************************************************************
***
te-master                  : ok=65   changed=28   unreachable=0
failed=0
```

Notice that we ran with `-t rsyslog`, but it first ran the `warewulf` role as we required by our dependencies.

Let's make sure the right configuration got written into our BOS:

```
[lowell@te-cm ansible]$ ssh root@te-master
Last login: Fri Jun 14 06:33:38 2019 from 172.16.1.252
[root@te-master ~]# cat
/opt/ohpc/admin/images/centos/compute/etc/rsyslog.conf
$ModLoad imuxsock # provides support for local system logging (e.g. via
logger command)
$ModLoad imjournal # provides access to the systemd journal
$WorkDirectory /var/lib/rsyslog
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat
$IncludeConfig /etc/rsyslog.d/*.conf
$OmitLocalLogging on
$IMJournalStateFile imjournal.state
*.emerg                                                 :omusrmsg:*
local7.*                                                /var/log/boot.log
*.* @172.16.0.254:514
```

Ok, that looks right. Now we need to reboot our nodes into the new image.

```
[root@te-master ~]# pdsh -w te[01-10] systemctl reboot
te03: Warning: Permanently added 'te03,172.16.0.3' (ECDSA) to the list of
known hosts.
te07: Warning: Permanently added 'te07,172.16.0.7' (ECDSA) to the list of
known hosts.
te04: Warning: Permanently added 'te04,172.16.0.4' (ECDSA) to the list of
known hosts.
```

```
te05: Warning: Permanently added 'te05,172.16.0.5' (ECDSA) to the list of
known hosts.
...
```

While the nodes are booting, let's tail the `/var/log/messages`. We should start to see syslog events from our nodes coming through:

```
Jun 14 06:52:39 te09 systemd: Removed slice User Slice of root.
Jun 14 06:52:39 te04 systemd: Removed slice User Slice of root.
Jun 14 06:52:44 te09 ntpd[4741]: 0.0.0.0 c628 08 no_sys_peer
Jun 14 06:52:45 te01 ntpd[3491]: 0.0.0.0 0618 08 no_sys_peer
```

We see the format is:

```
<date> <host> <service>: <message>
```

We can also send a test message from one fo the nodes:

```
[root@te01 ~]# logger "Test Message"
```

Should result in:

```
Jun 12 06:55:18 te01 root: Test Message
```

On the master.

Great! Now we an see all syslog events from the nodes on the master.

Step 2: Splunk

We will now setup Splunk to analyze the data from `rsyslog`. Much of this step will be interactive exploration, since splunk provides a web GUI.

We are not going to use ansible to set this up. We will install it directly on the master. This is just a trial license of Splunk, and we're just going to install it on the master to play around with it.

First, download the splunk rpm to your master:

```
[root@te-master ~]# curl -O http://10.0.52.146/repo/splunk.rpm
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left
Speed
```

```
100  366M  100  366M    0     0   130M      0  0:00:02  0:00:02 --:--:--
130M
```

Now install it:

```
[root@te-master ~]# rpm -Uvh splunk.rpm
warning: splunk.rpm: Header V4 RSA/SHA256 Signature, key ID b3cd4420:
NOKEY
Preparing...                          ################################
[100%]
        package splunk-7.3.0-657388c7a488.x86_64 is already installed
```

This will create `/opt/splunk`.

```
[root@te-master ~]# ls /opt/splunk
bin            etc      lib               openssl           share
var
copyright.txt  include  license-eula.txt  README-splunk.txt  splunk-7.3.0-
657388c7a488-linux-2.6-x86_64-manifest
```

We have to manually start splunk the first time. It will require us to accept a license agreement, and set a username/password.

```
[root@te-master bin]# cd /opt/splunk/bin
[root@te-master bin]# ./splunk start

...(accept license, etc)

Checking prerequisites...
        Checking http port [8000]: open
        Checking mgmt port [8089]: open
        Checking appserver port [127.0.0.1:8065]: open
        Checking kvstore port [8191]: open
        Checking configuration... Done.
        Checking critical directories...        Done
        Checking indexes...
                Validated: _audit _internal _introspection _telemetry
_thefishbucket history main summary
        Done
        Checking filesystem compatibility...  Done
        Checking conf files for problems...
        Done
        Checking default conf files for edits...
        Validating installed files against hashes from
'/opt/splunk/splunk-7.3.0-657388c7a488-linux-2.6-x86_64-manifest
        All installed files intact.
        Done
```

```
All preliminary checks passed.

Starting splunk server daemon (splunkd)...
Done
                                                           [  OK  ]

Waiting for web server at http://127.0.0.1:8000 to be available... Done


If you get stuck, we're here to help.
Look for answers here: http://docs.splunk.com

The Splunk web interface is at http://te-master:8000
```

Notice that it starts on port `8000` of the master. We don't actually want to start splunk this way though. Let's stop it, and use their command to enable it through systemd:

```
[root@te-master bin]# ./splunk stop
Stopping splunkd...
Shutting down.  Please wait, as this may take a few minutes.
..                                                         [  OK  ]
Stopping splunk helpers...
                                                           [  OK  ]
Done.
[root@te-master bin]# /opt/splunk/bin/splunk enable boot-start
Init script installed at /etc/init.d/splunk.
Init script is configured to run at boot.
[root@te-master bin]# systemctl start splunk
```

Now splunk is managed by systemd, and will start on reboot.

At this point, you should be able to open a browser and go to:

```
http://<your_ip>:8000/
```

Login with your credentials you set for splunk.

From this point we will explore together on the projector