Day-to-day Administration: Can I mix Linux distributions on my cluster?

Can the Bright Cluster Manager (BCM) be used to manage a mix of Linux distributions within the same cluster?

Yes.

Introduction

With BCM, multiple, different images can be used in different ways. You can deploy those images across the entire cluster, or deploy on specified groups of nodes and run multiple images simultaneously in the cluster.

Sometimes the user requires a mix of different Linux distributions, to be run at the same time for various reasons.

E.g.: a software that is licensed/certified for RHEL 5.5 has to run on a sub-set of the nodes of a cluster, for nodes that are otherwise running a later RHEL 6. version.

In general there is no issue with doing that except for one detail. The /cm/shared directory is exported by the head node and mounted by all the nodes. This directory provides certain pre-compiled binaries needed by the nodes, such as for example the MPI compilers and the BLAS libraries. These libraries have dynamic dependencies upon other libraries and such dependencies might not be resolved properly if the software image image distributions and the head node distribution are different.

The solution to this problem is to add an extra head node running the secondary distribution. This node -- unlike the principal and fail-over head node of the cluster -- will not be used for management and provisioning purposes, but will instead only be used for providing the /cm/shared directory filesystem export. The benefit of adding this extra node is that software in /cm/shared will then be able to receive regular updates. Installing those packages inside the software image is not possible, as the software packages provided with the BCM are not relocatable.

In this Knowledge Base article we will discuss the steps required to create a cluster with a mix of distributions.

Step 1: Installing a new head node

A new head node that will provide the /cm/shared directory needs to be installed. If, for example, the original installation of the cluster used RHEL 6.x and the new distribution is SLES 11 SP2, then the required DVD ISO for SLES 11 SP2 needs to be downloaded from the Customer Portal. This ISO will be used to install the new head node.

The new head node can be either a physical node or a virtual machine. It is important to ensure that the network interface of the new head node is able to communicate on the network on which the compute nodes are attached to. The user must also ensure that the minimal hardware requirements for a head node according to Section 2.1.1 of the Administrators manual (ver. 6.1) are met. In most cases a workload manager is node needed to run on this head node. Finally, when the networks are configured (see section 2.3.8 of the manual) the user needs to ensure that the base IP address and subnet mask settings match those that are used by the cluster. It is also important to ensure that the IP address of the interface of the head node that is assigned to the node provisioning network is obtained via DHCP. This option can be changed later at the extra cost of a head node reboot.

Step 2: on the old head node

After the new head node installation is complete, the user needs to connect to the principal head node of the cluster and carry out a some tasks before the new nodes can use the new Linux distribution's software images.

- Add the new head node as a generic device so that the DHCP daemon on the principal head node can provide an IP lease, the DNS records are set and the status of the new head node can be monitored. This can be done via cmsh as in the example below:

```
[panos-trunk-fb->device]% device add genericdevice panos-sles
[panos-trunk-fb->device*[panos-sles*]]% set mac 52:54:00:f8:b2:72
[panos-trunk-fb->device*[panos-sles*]]% commit
[panos-trunk-fb->device[panos-sles]]% show
Parameter                      Value
------------------------------ ------------------------------
Activation                     Tue, 09 Jul 2013 17:44:33 CEST
Additional Hostnames
Container index                0
Custom ping script
Custom ping script argument
Custom power script
Custom power script argument
Device height                  1
Device position                0
Ethernet switch
Hostname                       panos-sles
Ip                             0.0.0.0
Mac                            52:54:00:f8:b2:72
Model
Network


Notes                          <0 bytes>
Partition                      base
Power control                  apc
```

```
PowerDistributionUnits
Rack
Revision
Tag                            00000000a000
Type                           GenericDevice
```

At this point a pre-assigned IP address for the new node can also be set via:

```
[panos-trunk-fb->device*[panos-sles*]]% set ip 10.141.1.100
[panos-trunk-fb->device*[panos-sles*]]% commit
```

- After this the user will have to log-in into the new head node and change the IP address of the relevant interface (eth1 in this example) to DHCP:

```
panos-sles:~ # cmsh
[panos-sles]% device  use panos-sles
[panos-sles->device[panos-sles]]% interfaces
[panos-sles->device[panos-sles]->interfaces]% use eth1
[panos-sles->device[panos-sles]->interfaces[eth1]]% set dhcp on
[panos-sles->device*[panos-sles*]->interfaces*[eth1*]]% commit
[panos-sles->device[panos-sles]->interfaces[eth1]]%
```

- It might also be desirable to prevent the new head node from responding to DHCP requests from new nodes:

```
[panos-sles->network[internalnet]]% set lockdowndhcpd  on
[panos-sles->network*[internalnet*]]% commit
[panos-sles->network[internalnet]]%
Tue Jul  9 17:52:24 2013 [notice] panos-sles: Service dhcpd was restarted
```

Finally, an update can be run on the new head node (yum update or zypper up) to ensure that the latest versions of the packages are installed. At this point the new head node is ready to serve the /cm/shared directory to the nodes that will run the new Linux distribution.

Step 3:

The next step is to create a software image with the new Linux distribution. For that, a base distribution tarball is needed which can be copied directly from the ISO file that was downloaded in step 1.

First, mount the ISO:

```
[root@panos-trunk-f ~]# mount -o loop -t iso9660
bright6.1-sles11sp2.iso /mnt/cd0/
```

The base distribution tarball is located in the data subdirectory of the ISO image:

```
cp /mnt/cd0/data/SLES11sp2.tar.gz ~/tmp
```

A new software image can be created from the tarball by running cm-create-image command:

```
[root@panos-trunk-f ~]# cm-create-image  -a SLES11sp2.tar.gz -n
sles-image
```

The option -n specifies the name of the new image.

```
[root@panos-trunk-f ~]# cm-create-image  -a SLES11sp2.tar.gz -n
sles-image2
                Validating base tar file  .....   [  OK  ]
                   Running sanity checks  .....   [  OK  ]
                 Unpacking base tar file  .....   [  OK  ]

   ******************** IMPORTANT ****************************
   Please confirm that the base distribution repositories for
   the software image are enabled. For instructions on how to
   enable repositories for your software image, please refer
   the administrator's manual.

   Image creation can be resumed in one of the following ways:
   -----------------------------------------------------------
   1. Enter 'e' to exit, and configure repositories.
      Then, restart program with the -d (--fromdir) option.
      cm-create-image -d /cm/images/sles-image2 -n sles-image2

   2. Open a new console, and configure repositories.
      Then enter 'c' on this console, to continue software
      image creation.

   ***********************************************************

Continue(c)/Exit(e)? c
                   Copying cm repo files  .....   [  OK  ]
           Validating repo configuration  .....   [FAILED]

The following base distribution packages were not found:
--------------------------------------------------------

| gcc43-fortran                | libboost_date_time1_36_0     |
libboost_filesystem1_36_0    |
| libboost_graph1_36_0         | libboost_iostreams1_36_0     |
libboost_math1_36_0          |
| libboost_python1_36_0        | libboost_serialization1_36_0 |
libboost_system1_36_0        |
| libboost_test1_36_0          | libboost_thread1_36_0        |
libboost_wave1_36_0          |
| libexpat-devel               | libnuma-devel                |
libpciaccess0-devel          |
| libquadmath46                | libstdc++46-devel            |
libstdc++46-devel-32bit      |
| perl-libxml-perl             | libgfortran46                |
sle-sdk-release              |
| sle-sdk-release-SDK          | fuse-devel                   |
libxslt-devel                |
| libxslt-python               | libxml2-devel                |
readline-devel               |
| libgcrypt-devel              | libgpg-error-devel           |
libibcm-devel                |
| libibcommon-devel            | libibmad-devel               |
libibumad-devel              |
| libibverbs-devel             | opensm-devel                 |
librdmacm-devel              |
| dapl-devel                   | libmthca-rdmav2-devel        |
libmlx4-rdmav2-devel         |
| libnes-rdmav2-devel          | libcxgb3-rdmav2-devel        |
libibcm-devel-32bit          |
| libibmad-devel-32bit         | libibumad-devel-32bit        |
libibverbs-devel-32bit       |
| libibcommon-devel-32bit      | opensm-devel-32bit           |
perftest                     |
| fontconfig-devel             | freetype2-devel              |
db43                         |
| libjpeg-devel                | libpixman-1-0-devel          |
libpng-devel                 |
| libuuid-devel                | MesaGLw                      |
openmotif-devel              |
| openmotif-libs               | xorg-x11-devel               |
xorg-x11-fonts-devel         |
| xorg-x11-libfontenc-devel    | xorg-x11-libICE-devel        |
xorg-x11-libSM-devel         |
| xorg-x11-libX11-devel        | xorg-x11-libXau-devel        |
xorg-x11-libxcb-devel        |
| xorg-x11-libXdmcp-devel      | xorg-x11-libXext-devel       |
xorg-x11-libXfixes-devel     |
| xorg-x11-libxkbfile-devel    | xorg-x11-libXmu-devel        |
xorg-x11-libXp-devel         |
| xorg-x11-libXpm-devel        | xorg-x11-libXprintUtil-devel |
xorg-x11-libXrender-devel    |
| xorg-x11-libXt-devel         | xorg-x11-libXv-devel         |
xorg-x11-proto-devel         |
| xorg-x11-util-devel          | xorg-x11-xtrans-devel        |
zlib-devel                   |
```

This usually means that access to the already configured repositories in the software image failed,

or that the repositories that are currently configured, do not provide all the required packages.

```
----------------------------------------------------------------------
-----------------------------------------------------
The following repositories are defined in the image:
----------------------------------------------------------------------
-----------------------------------------------------
# | Alias                                            |
Name                                             | Enabled | Refresh
--+--------------------------------------------------+----------------
----------------------------------+---------+--------
1 | Cluster_Manager_Base                             | Cluster Manager
Trunk - Base                     | Yes     | No
2 | Cluster_Manager_Base_Extra                       | Extra rpms -
Base                                | Yes     | No
3 | Cluster_Manager_Updates                          | Cluster Manager
Trunk - Updates                  | Yes     | No
4 | Cluster_Manager_Updates_Extra                    | Extra rpms -
Updates                             | Yes     | No
5 | SUSE-Linux-Enterprise-Server-11-SP2 11.2.2-1.234 |
SUSE-Linux-Enterprise-Server-11-SP2 11.2.2-1.234 | Yes     | No

----------------------------------------------------------------------
-----------------------------------------------------
```

Please verify that you have access to the above repositories from inside the software image, or whether it is required to configure additional repositories in the software image.

Please refer your SLES Enterprise subscription and manuals for more detailed information on how to add/enable additional software repositories.

After the required repositories have been added/enabled, please verify as follows:

```
chroot /cm/images/sles-image2
yum list available <packagename>
```

Then re-run the image creation:

```
cm-create-image -d /cm/images/sles-image2 -n sles-image2
```

ERROR:

Note that the above step failed because some of the RPMs could not be retrieved from the repositories of the new Linux distribution. In this case, this happened because an entitlement was not available. To fix this the chroot to the image and use the relevant to your distribution utilities to enable access to downloads:

```
chroot /cm/images/sles-image2
```

After that re-run the image creation, but using the directory where the archive was extracted to instead:

```
cm-create-image -d /cm/images/sles-image -n sles-image
```

After that a new ramdisk needs to be created. In CMSH use the createramdisk command of the softwareimage sub-mode:

```
[panos-trunk-fb]% softwareimage
[panos-trunk-fb->softwareimage]% use sles-image
[panos-trunk-fb->softwareimage[sles-image]]% set kernelversion
3.0.13-0.27-default
[panos-trunk-fb->softwareimage[sles-image]]% createramdisk
[panos-trunk-fb->softwareimage[sles-image]]% commit
```

Step 4:

The next step is to create a special category for the nodes that will be provisioned with the new Linux distribution.

```
[panos-trunk-fb]% category
[panos-trunk-fb->category]% add sles-nodes
```

Then set the newly created image as the default image for this category:

```
[panos-trunk-fb->category*[sles-nodes*]]% set softwareimage sles-image
```

And change the file system mount for this category so that it points to the NFS share provided by the new head node:

```
[panos-trunk-fb->category*[sles-nodes*]]% fsmounts
[panos-trunk-fb->category*[sles-nodes*]->fsmounts]% use /cm/shared
[panos-trunk-fb->category*[sles-nodes*]->fsmounts[/cm/shared]]% set
device panos-sles:/cm/shared
[panos-trunk-fb->category*[sles-nodes*]->fsmounts*[/cm/shared*]]%
```

```
commit
```

Step 5:

The last step is to set nodes to the new category:

For a single node:

```
[panos-trunk-fb->device]% use node002
[panos-trunk-fb->device[node002]]% set category sles-nodes
[panos-trunk-fb->device*[node002*]]% commit
```

For a range of nodes e.g.

```
foreach -n  node128..node160 (set category sles-nodes; commit)
```

and reboot the compute nodes.

After the reboot the new nodes will use SLES 11 SP2 as their Linux distribution. Switching the Linux distribution of a node merely requires switching its node category after this step. More Linux distributions can be configured by following steps 1 through 5 for the extra distributions. Finally, note that software updates in /cm/shared will take place by updating the packages of the newly added head node.

Addendum: Configuring a Workload Manager in a multi distribution setup

A multi-distribution Bright cluster needs special care when installing and configuring a Workload Manager (WLM). The details can vary from one Workload Manager to another. This section tries to cover the most important points, and the Workload Manager’s documentation can then be referred to for further details.

Conventions

The following abbreviations will be used to refer to the components of the multi-distribution setup.

- H1 - The head node of the cluster
- D1 - The Linux distribution being run on H1

- H2 - The head node which was configured to provide /cm/shared for a different distribution, as described in this KB article
- D2 - The Linux distribution run on H2

Is it necessary to configure the Workload Manager for all nodes?

If no jobs are to run on the nodes running D2, then there is no special care required.

First step: Deploying the Workload Manager

The Administrator Manual explains how each WLM is installed and configured. The installation is usually done by installing packages, and then running wlm-setup on H1. For a mixed-distribution setup, the installation steps must also be run in H2, so that the WLM binaries for D2 are available in /cm/shared.

Second step: referencing the correct configuration files

After setting up the cluster as described in this article, and after installing the WLM, the D2 compute nodes will be mounting the /cm/shared directory from H2. This will cause problems in Workload Managers where the compute nodes require access to configuration files which are stored in /cm/shared. A discussion of some problematic cases follows:

Slurm

When the slurm service is started on a compute node, it requires access to the configuration files in /cm/shared/apps/slurm/var/etc. Those files are only generated in H1. Trying to start the slurm service in a D2 compute node therefore fails, because the files cannot be found.

To solve this, D2 nodes must mount the /cm/shared/apps/slurm/var/etc directory from H1. This can be done by adding a new fsmount from the category:

```
% fsmounts

% clone /cm/shared /cm/shared/apps/slurm/var/etc/

% set device $localnfsserver:/cm/shared/apps/slurm/var/etc/

% commit
```

LSF

The same type of problem occurs for LSF as for Slurm. The directory that has to be mounted from H1 in this case is /cm/shared/apps/lsf/var/conf

OpenLava

The same type of problem occurs for OpenLava as for Slurm. The directory that has to be mounted from H1 in this case is /cm/shared/apps/openlava/var/etc

Other Workload Managers

No other Workload Manager seems to require that the compute nodes read the configuration files from /cm/shared. In case reading from /cm/shared is needed, then the administrator should identify the configuration files needed, and create the appropriate fsmount as described in the preceding text.

Possible problems

Distributions that use systemd may result in WLM services that are configured differently. At the time of writing of this article (March 2017), this behavior was detected in UGE 8.4.4. When wlm-setup is run for such a WLM, then an init.d script or a systemd unit is created inside the software image, depending on the Linux distribution used by the cluster. This could cause problems, as illustrated by the following scenarios:

1 - **D1 doesn’t use systemd and D2 does use systemd.** In this case, wlm-setup creates an init.d script inside the D2 software image. When CMDaemon then tries to start the services for WLM in a D2 compute node, it should be able to run the correct service anyway.

2 - **D1 uses systemd and D2 doesn’t.** In this case, wlm-setup creates a systemd unit file inside the D2 software image. When CMDaemon tries to start the services for WLM in a D2 compute node, it will not be able to find the service and cannot start it.

To debug this kind of problem, the administrator can check the installation log generated by wlm-setup in order to see what files were installed in the software image. If CMDaemon is not able to start a service due to such a scenario, then the administrator can fix it by copying the required files into place manually.