# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table of Contents</td>
<td>i</td>
</tr>
<tr>
<td><strong>1 Quickstart Installation Guide</strong></td>
<td></td>
</tr>
<tr>
<td>1.1 Installing The Head Node</td>
<td>1</td>
</tr>
<tr>
<td>1.2 First Boot</td>
<td>4</td>
</tr>
<tr>
<td>1.3 Bothing Regular Nodes</td>
<td>4</td>
</tr>
<tr>
<td>1.4 Quickstart For GPUs</td>
<td>6</td>
</tr>
<tr>
<td>1.5 Optional: Upgrading Python</td>
<td>9</td>
</tr>
<tr>
<td>1.6 Running Bright View</td>
<td>10</td>
</tr>
<tr>
<td><strong>2 Introduction</strong></td>
<td>11</td>
</tr>
<tr>
<td>2.1 What Is Bright Cluster Manager?</td>
<td></td>
</tr>
<tr>
<td>2.1.1 What OS Platforms Is It Available For?</td>
<td>11</td>
</tr>
<tr>
<td>2.1.2 What Architectures Does It Run On?</td>
<td>11</td>
</tr>
<tr>
<td>2.1.3 What OS Platforms Can It Be Managed From?</td>
<td>11</td>
</tr>
<tr>
<td>2.2 Cluster Structure</td>
<td>12</td>
</tr>
<tr>
<td><strong>3 Installing Bright Cluster Manager</strong></td>
<td>15</td>
</tr>
<tr>
<td>3.1 Minimal Hardware Requirements</td>
<td></td>
</tr>
<tr>
<td>3.1.1 Head Node</td>
<td>15</td>
</tr>
<tr>
<td>3.1.2 Compute Nodes</td>
<td>16</td>
</tr>
<tr>
<td>3.2 Supported Hardware</td>
<td></td>
</tr>
<tr>
<td>3.2.1 Compute Nodes</td>
<td>16</td>
</tr>
<tr>
<td>3.2.2 Ethernet Switches</td>
<td>16</td>
</tr>
<tr>
<td>3.2.3 Power Distribution Units</td>
<td>17</td>
</tr>
<tr>
<td>3.2.4 Management Controllers</td>
<td>17</td>
</tr>
<tr>
<td>3.2.5 InfiniBand And Similar</td>
<td>17</td>
</tr>
<tr>
<td>3.2.6 GPUs</td>
<td>17</td>
</tr>
<tr>
<td>3.2.7 MICs</td>
<td>17</td>
</tr>
<tr>
<td>3.2.8 RAID</td>
<td>17</td>
</tr>
<tr>
<td>3.3 Head Node Installation: Bare Metal Method</td>
<td></td>
</tr>
<tr>
<td>3.3.1 ISO Boot Menu</td>
<td>18</td>
</tr>
<tr>
<td>3.3.2 Welcome Screen</td>
<td>19</td>
</tr>
<tr>
<td>3.3.3 Software Licenses</td>
<td>20</td>
</tr>
<tr>
<td>3.3.4 Kernel Modules Configuration</td>
<td>22</td>
</tr>
<tr>
<td>3.3.5 Hardware Info</td>
<td>23</td>
</tr>
<tr>
<td>3.3.6 Installation Source</td>
<td>24</td>
</tr>
<tr>
<td>3.3.7 Cluster Settings</td>
<td>24</td>
</tr>
<tr>
<td>3.3.8 Workload Management Configuration</td>
<td>25</td>
</tr>
<tr>
<td>3.3.9 Network Topology</td>
<td>26</td>
</tr>
<tr>
<td>3.3.10 Head Node Settings</td>
<td>29</td>
</tr>
</tbody>
</table>
## Table of Contents

3.3.11 Compute Nodes Settings .................................................. 30  
3.3.12 BMC Configuration ......................................................... 31  
3.3.13 Networks ......................................................................... 32  
3.3.14 Head node interfaces ....................................................... 34  
3.3.15 Compute node interfaces ................................................... 34  
3.3.16 Disk Layout ...................................................................... 36  
3.3.17 Additional Software ......................................................... 38  
3.3.18 Summary .......................................................................... 38  
3.3.19 Deployment ................................................................. 39  
3.3.20 Licensing And Further Configuration ................................. 40  

3.4 Head Node Installation: Add-On Method .................................. 40  
3.4.1 Prerequisites ................................................................. 41  
3.4.2 Installing The Installer ..................................................... 42  
3.4.3 Running The Installer ....................................................... 42  

3.5 Enabling Remote Browser-Based Installation Via The Text Mode Installer ......................... 48  

4 Licensing Bright Cluster Manager ............................................ 51  
4.1 Displaying License Attributes ............................................... 52  
4.1.1 Displaying License Attributes Within Bright View ............... 52  
4.1.2 Displaying License Attributes Within cmsh ....................... 53  
4.2 Verifying A License—The verify-license Utility ....................... 53  
4.2.1 The verify-license Utility Can Be Used When licenseinfo Cannot Be Used ...................... 53  
4.2.2 Using The verify-license Utility To Troubleshoot License Issues ................................. 53  
4.2.3 Using The versioninfo Command To Verify The Cluster Manager Version .................. 55  
4.3 Requesting And Installing A License Using A Product Key ........ 55  
4.3.1 Is A License Needed?—Verifying License Attributes ............... 56  
4.3.2 The Product Key ................................................................ 56  
4.3.3 Requesting A License With The request-license Script .......... 57  
4.3.4 Installing A License With The install-license Script ............. 59  
4.3.5 Re-Installing A License After Replacing The Hardware ......... 60  
4.3.6 Re-Installing A License After Wiping Or Replacing The Hard Drive ............................ 60  
4.3.7 Re-Installing A License With An Add-On Attribute .............. 61  
4.3.8 Rebooting Nodes After An Install ..................................... 61  
4.3.9 The Customer Portal ....................................................... 62  

5 Linux Distributions That Use Registration .................................. 63  
5.1 Registering A Red Hat Enterprise Linux Based Cluster ............. 63  
5.1.1 Registering A Head Node With RHEL ................................. 63  
5.1.2 Registering A Software Image With RHEL ......................... 64  
5.2 Registering A SUSE Linux Enterprise Server Based Cluster ....... 64  
5.2.1 Registering A Head Node With SUSE ............................... 65  
5.2.2 Registering A Software Image With SUSE ......................... 65  

6 Changing The Network Parameters Of The Head Node .................. 67  
6.1 Introduction ................................................................. 67  
6.2 Method ................................................................. 67  
6.3 Terminology ............................................................. 67
# Third Party Software

## 7.1 Modules Environment

## 7.2 Shorewall

### 7.2.1 The Shorewall Service Paradigm

### 7.2.2 Shorewall Zones, Policies, And Rules

### 7.2.3 Clear And Stop Behavior In service Options, bash Shell Command, And cmsh Shell

### 7.2.4 Further Shorewall Quirks

## 7.3 Compilers

### 7.3.1 GCC

### 7.3.2 Intel Compiler Suite

### 7.3.3 PGI High-Performance Compilers

### 7.3.4 The NVIDIA HPC SDK

## 7.4 CUDA For GPUs

### 7.4.1 Installing CUDA

### 7.4.2 Installing Kernel Development Packages

### 7.4.3 Verifying CUDA

### 7.4.4 Verifying OpenCL

### 7.4.5 Configuring The X Server

### 7.4.6 NVIDIA Validation Suite (Package: cuda-dcgm-nvvs)

### 7.4.7 Further NVIDIA Configuration Via The Cluster Manager

## 7.5 AMD GPU Driver Installation

### 7.5.1 AMD GPU Driver Installation For RHEL/CentOS

### 7.5.2 AMD GPU Driver Installation For Ubuntu

### 7.5.3 AMD GPU Driver Installation For SLED/SLES

## 7.6 OFED Software Stack

### 7.6.1 Choosing A Distribution Version, Or A Vendor Version, Ensuring The Kernel Matches, And Logging The Installation

### 7.6.2 Mellanox OFED Stack Installation Using The Bright Computing Repository

## 7.7 Intel OPA Software Stack

### 7.7.1 Installation

### 7.7.2 Configuration And Deployment

# Burning Nodes

## 8.1 Test Scripts Deployment

## 8.2 Burn Configurations

### 8.2.1 Mail Tag

### 8.2.2 Pre-install And Post-install

### 8.2.3 Post-burn Install Mode

### 8.2.4 Phases

### 8.2.5 Tests

## 8.3 Running A Burn Configuration

### 8.3.1 Burn Configuration And Execution In cmsh

### 8.3.2 Writing A Test Script

### 8.3.3 Burn Failures

## 8.4 Relocating The Burn Logs

### 8.4.1 Configuring The Relocation

### 8.4.2 Testing The Relocation
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Installing And Configuring SELinux</td>
<td>117</td>
</tr>
<tr>
<td>9.1</td>
<td>Introduction</td>
<td>117</td>
</tr>
<tr>
<td>9.2</td>
<td>Enabling SELinux On RHEL7</td>
<td>117</td>
</tr>
<tr>
<td>9.2.1</td>
<td>Case 1: Head And Compute Nodes Are Permissive</td>
<td>117</td>
</tr>
<tr>
<td>9.2.2</td>
<td>Case 2: Head Node Is Permissive And Compute Nodes Are Enforcing</td>
<td>118</td>
</tr>
<tr>
<td>9.2.3</td>
<td>Case 3: Head Node Is Enforcing And Compute Nodes Are Enforcing</td>
<td>119</td>
</tr>
<tr>
<td>9.3</td>
<td>Additional Considerations</td>
<td>119</td>
</tr>
<tr>
<td>9.3.1</td>
<td>Provisioning The .autorelabel File Tag</td>
<td>119</td>
</tr>
<tr>
<td>9.3.2</td>
<td>SELinux Warnings During Regular Node Updates</td>
<td>119</td>
</tr>
<tr>
<td>9.4</td>
<td>Filesystem Security Context Checks</td>
<td>120</td>
</tr>
<tr>
<td>A</td>
<td>Other Licenses, Subscriptions, Or Support Vendors</td>
<td>121</td>
</tr>
<tr>
<td>B</td>
<td>Hardware Recommendations</td>
<td>123</td>
</tr>
<tr>
<td>B.1</td>
<td>Heuristics For Requirements</td>
<td>123</td>
</tr>
<tr>
<td>B.1.1</td>
<td>Heuristics For Requirements For A Regular Node</td>
<td>123</td>
</tr>
<tr>
<td>B.1.2</td>
<td>Heuristics For Requirements For A Head Node</td>
<td>123</td>
</tr>
<tr>
<td>B.2</td>
<td>Observed Head Node Resources Use, And Suggested Specification</td>
<td>124</td>
</tr>
<tr>
<td>B.2.1</td>
<td>Observed Head Node Example CMDaemon And MySQL Resources Use</td>
<td>124</td>
</tr>
<tr>
<td>B.2.2</td>
<td>Suggested Head Node Specification For Clusters Beyond 1000 Nodes</td>
<td>124</td>
</tr>
</tbody>
</table>
Quickstart Installation Guide

This chapter describes a basic and quick installation of Bright Cluster Manager on “bare metal” cluster hardware as a step-by-step process, and gives very little explanation of the steps. Following these steps should allow a moderately experienced cluster administrator to get a cluster up and running in a fairly standard configuration as quickly as possible. This would be without even having to read the introductory Chapter 2 of this manual, let alone the entire manual. References to chapters and sections are provided where appropriate.

Some asides, before getting on with the steps themselves:

- If the cluster has already been installed, tested, and configured, but only needs to be configured now for a new network, then the administrator should only need to look at Chapter 6. Chapter 6 lays out how to carry out the most common configuration changes that usually need to be done to make the cluster work in the new network.

- For administrators that are very unfamiliar with clusters, reading the introduction (Chapter 2) and then the more detailed installation walkthrough for a bare metal installation (Chapter 3, sections 3.1, 3.2, and 3.3) is recommended. Having carried out the head node installation, the administrator can then return to this quickstart chapter (Chapter 1), and continue onward with the quickstart process of regular node installation (section 1.3).

- The configuration and administration of the cluster after it has been installed is covered in the Bright Cluster Manager Administrator Manual. The Administrator Manual should be consulted for further background information as well as guidance on cluster administration tasks, after the introduction (Chapter 2) of the Installation Manual has been read.

- If all else fails, administrator-level support is available via https://support.brightcomputing.com. Section 17.2 of the Administrator Manual has further details on how to brief the support team, so that the issue can be resolved as quickly as possible.

The quickstart steps now follow:

### 1.1 Installing The Head Node

The head node does not need to be connected to the regular nodes at this point, though it helps to have the wiring done beforehand so that how things are connected is known.

1. The BIOS of the head node should have the local time set.
2. The head node should be booted from the Bright Cluster Manager DVD.
3. The option: Install Bright Cluster Manager (Graphical), or Install Bright Cluster Manager (Text mode), should be selected in the text boot menu. The Graphical installation is
recommended, and brings up the GUI installation **Welcome** screen. The **Text mode** installation provides a minimal, Ncurses-based version, of the GUI installation.

Only the GUI installation is discussed in the rest of this quickstart for convenience.

4. At the **Welcome** screen, **Start installation** should be clicked.

5. At the **License** screens:
   - At the **Bright Computing Software License** screen, the acceptance checkbox should be ticked. **Next** should then be ticked.
   - At the **Linux base distribution** screen, the acceptance checkbox should be ticked. **Next** should then be clicked.

6. At the **Kernel Modules** screen, **Next** should be clicked.

7. At the **Hardware Info** screen, the detected hardware should be reviewed. If additional kernel modules are required, then the administrator should go back to the **Kernel Modules** screen. Once all the relevant hardware (Ethernet interfaces, hard drive and DVD drive) is detected, **Next** should be clicked.

8. At the **Installation source** screen, the DVD drive containing the Bright Cluster Manager DVD should be selected, then **Next** clicked.

9. At the **General cluster settings** screen, one or more nameservers and one or more domains can be set, if they have not already been automatically filled. The remaining settings can usually be left as is.

10. At the **Workload management** screen, an HPC workload manager can be selected. The choice can be made later on too, after Bright Cluster Manager has been installed.

11. For the **Network topology** screen, a Type 1 network is the most common.

12. For the **Head node settings** screen, the head is given a name and a password.

13. For the **Compute nodes settings** screen, the head is given a name and a password.
   - The number of racks and regular nodes are specified
   - The base name for the regular nodes is set. Accepting the default of **node** means nodes names are prefixed with **node**, for example: node001, node002...
   - The number of digits to append to the base name is set. For example, accepting the default of 3 means nodes from node001 to node999 are possible names.
   - The correct hardware manufacturer is selected

14. For the **BMC configuration** screen, the use of IPMI/iLO/DRAC/CIMC/Redfish BMCs is carried out. Adding an IPMI/iLO/DRAC/CIMC/Redfish network is needed to configure IPMI/iLO/DRAC/CIMC/Redfish interfaces in a different IP subnet, and is recommended.

15. At the **Networks** screen, the network parameters for the head node should be entered for the interface facing the network named **externalnet**:
   - If using DHCP on that interface, the parameters for **IP Address**, **Netmask** and **Gateway** as suggested by the DHCP server on the external network can be accepted.
   - If not using DHCP on that interface, static values put in instead.

The network parameters for **externalnet** that can be set include the:
1.1 Installing The Head Node

- base address (also called the network address)
- netmask
- domain name

The network externalnet corresponds to the site network that the cluster resides in (for example, a corporate or campus network). The IP address details are therefore the details of the head node for a type 1 externalnet network (figure 3.11). A domain name should be entered to suit the local requirements.

16. For the **Head node interfaces** screen, the head node network interfaces are assigned networks and IP addresses. The assigned values can be reviewed and changed.

17. At the **Compute node interfaces** screen, the compute node interfaces are assigned networks and IP addresses. The assigned values can be reviewed and changed.

18. At the **Disk Partitioning and Layouts** screen, a drive should be selected for the head node. The installation will be done onto this drive, overwriting all its previous content.

The administrator can modify the disk layout for the head node by selecting a pre-defined layout.

For hard drives that have less than about 500GB space, the XML file `master-one-big-partition.xml` is used by default:

<table>
<thead>
<tr>
<th>Partition</th>
<th>Space</th>
<th>Mounted At</th>
<th>Filesystem Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>512M</td>
<td>/boot</td>
<td>ext2</td>
</tr>
<tr>
<td>0</td>
<td>100M</td>
<td>/boot/efi</td>
<td>fat</td>
</tr>
<tr>
<td>2</td>
<td>16G</td>
<td>-</td>
<td>swap</td>
</tr>
<tr>
<td>3</td>
<td>rest</td>
<td>/</td>
<td>xfs</td>
</tr>
</tbody>
</table>

Default layout for up to 500GB: One big partition.

For hard drives that have about 500GB or more of space, the XML file `master-standard.xml` is used by default:

<table>
<thead>
<tr>
<th>Partition</th>
<th>Space</th>
<th>Mounted At</th>
<th>Filesystem Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>512M</td>
<td>/boot</td>
<td>ext2</td>
</tr>
<tr>
<td>0</td>
<td>100M</td>
<td>/boot/efi</td>
<td>fat</td>
</tr>
<tr>
<td>2</td>
<td>16G</td>
<td>-</td>
<td>swap</td>
</tr>
<tr>
<td>3</td>
<td>20G</td>
<td>/tmp</td>
<td>xfs</td>
</tr>
<tr>
<td>4</td>
<td>180G</td>
<td>/var</td>
<td>xfs</td>
</tr>
<tr>
<td>5</td>
<td>rest</td>
<td>/</td>
<td>xfs</td>
</tr>
</tbody>
</table>

Default layout for more than 500GB: Several partitions.

The layouts indicated by these tables may be fine-tuned by editing the XML partitioning definition during this stage. The “max” setting in the XML file corresponds to the “rest” entry in these tables, and means the rest of the drive space is used up for the associated partition, whatever the leftover space is.

There are also other layout templates available from a menu.
19. At the **Additional software** screen, extra software options can be chosen for installation if these were selected for the installation ISO. The extra software options are:

- CUDA
- Ceph
- OFED stack

20. The **Summary** screen should be reviewed. A wrong entry can still be fixed at this point. The **Next** button then starts the installation.

21. The **Deployment** screen should eventually complete. Clicking on **Reboot** reboots the head node.

### 1.2 First Boot

1. The DVD should be removed or the boot-order altered in the BIOS to ensure that the head node boots from the first hard drive.

2. Once the machine is fully booted, a login should be done as **root** with the password that was entered during installation.

3. A check should be done to confirm that the machine is visible on the external network. Also, it should be checked that the second NIC (i.e. `eth1`) is physically connected to the external network.

4. If the parent distribution for Bright Cluster Manager is RHEL and SUSE then registration (Chapter 5) should usually be done.

5. The license parameters should be verified to be correct:

   ```
cmsh -c "main licenseinfo"
   ```

   If the license being used is a temporary license (see **End Time** value), a new license should be requested well before the temporary license expires. The procedure for requesting and installing a new license is described in Chapter 4.

### 1.3 Booting Regular Nodes

1. A check should be done to make sure the first NIC (i.e. `eth0`) on the head node is physically connected to the internal cluster network.

2. The BIOS of regular nodes should be configured to boot from the network. The regular nodes should then be booted. No operating system is expected to be on the regular nodes already. If there is an operating system there already, then by default, it is overwritten by a default image provided by the head node during the next stages.

3. If everything goes well, the node-installer component starts on each regular node and a certificate request is sent to the head node.

   If a regular node does not make it to the node-installer stage, then it is possible that additional kernel modules are needed. Section 5.8 of the *Administrator Manual* contains more information on how to diagnose problems during the regular node booting process.

4. To identify the regular nodes (that is, to assign a host name to each physical node), several options are available. Which option is most convenient depends mostly on the number of regular nodes and whether a (configured) managed Ethernet switch is present.

   Rather than identifying nodes based on their MAC address, it is often beneficial (especially in larger clusters) to identify nodes based on the Ethernet switch port that they are connected to. To allow nodes to be identified based on Ethernet switch ports, section 3.8 of the *Administrator Manual* should be consulted.

© Bright Computing, Inc.
If a node is unidentified, then its node console displays an Ncurses message to indicate it is an unknown node, and the net boot keeps retrying its identification attempts. Any one of the following methods may be used to assign node identities when nodes start up as unidentified nodes:

a. **Identifying each node on the node console**: To manually identify each node, the “Manually select node” option is selected for each node. The node is then identified manually by selecting a node-entry from the list, choosing the Accept option. This option is easiest when there are not many nodes. It requires being able to view the console of each node and keyboard entry to the console.

b. **Identifying nodes using cmsh**: In cmsh the newnodes command in device mode (page 185, section 5.4.2 of the Administrator Manual) can be used to assign identities to nodes from the command line. When called without parameters, the newnodes command can be used to verify that all nodes have booted into the node-installer and are all waiting to be assigned an identity.

c. **Identifying nodes using Bright View**: The node identification resource (page 189, section 5.4.2 of the Administrator Manual) in Bright View automates the process of assigning identities so that manual identification of nodes at the console is not required.

**Example**

To verify that all regular nodes have booted into the node-installer:

```
[root@mycluster ~]# cmsh
[mycluster]# device newnodes

MAC First appeared Detected on switch port
----------------- ------------------------ -----------------------
00:0C:29:D2:68:8D 05 Sep 2011 13:43:13 PDT [no port detected]
00:0C:29:F5:94:8D 05 Sep 2011 13:49:41 PDT [no port detected]

[mycluster]# device newnodes | wc -l
MAC First appeared Detected on switch port
----------------- ------------------------ -----------------------
32
```

**Example**

Once all regular nodes have been booted in the proper order, the order of their appearance on the network can be used to assign node identities. To assign identities node001 through node032 to the first 32 nodes that were booted, the following commands may be used:

```
[root@mycluster ~]# cmsh
[mycluster]# device newnodes -s -n node001..node032

MAC First appeared Hostname
----------------- ----------------------------- ---------
00:0C:29:62:8D:8D Mon, 05 Sep 2011 13:43:13 PDT node001
00:0C:29:54:F5:94 Mon, 05 Sep 2011 13:49:41 PDT node002
```

5. Each regular node is now provisioned and eventually fully boots. In case of problems, section 5.8 of the Administrator Manual should be consulted.
6. Optional: To configure power management, Chapter 4 of the Administrator Manual should be consulted.

1.4 Quickstart For GPUs

This assumes that the head nodes and regular nodes are up and running. Getting GPUs set up right can be done as follows for a typical installation:

1. GPU Hardware And Sanity Check

If the hardware is not detected, then that must first be fixed. Checking for the hardware can be done as follows:

(a) NVIDIA GPU hardware should be detected on the nodes that use it. This is true for NVIDIA GPU units (separate from the nodes) as well as for on-board NVIDIA GPUs. The `lspci` command can be used for detection. For example, for a GPU used by node001:

Example

```
[root@bright91 ~]# ssh node001 lspci | grep NVIDIA
00:07.0 3D controller: NVIDIA Corporation GK110BGL [Tesla K40c] (rev a1)
```

(b) AMD CPUs, which have a GPU integrated with the CPU, the CPU chip, can similarly be identified with `lscpu`:

Example

```
[root@bright91 ~]# ssh node001 lscpu | grep "Model name:"
Model name: AMD Ryzen 5 2500U with Radeon Vega Mobile Gfx
```

The AMD chips can then be checked against the list of AMD chips with AMD GPUs, as listed at https://www.amd.com/en/support/kb/release-notes/rn-prorad-lin-18-20

2. Installing The Software

(a) Details of AMD GPU software installation are given in section 7.5.

(b) For NVIDIA GPUs, assuming the GPU is on the regular node node001, and that the hardware is supported by CUDA 11.1, then software installation is carried out at the head node as follows:

i. The software components are installed for the head node itself with:

```
[root@bright91 ~]# yum install cuda11.1-toolkit cuda11.1-sdk
```

ii. Components are installed into the image used by the nodes that have the GPUs, for example the image default-image, with:

```
[root@bright91 ~]# yum --installroot=/cm/images/default-image install cuda-driver cuda-dcgm
```

iii. The nodes with GPUs can then simply be rebooted to compile the CUDA drivers as the node boots, and to start the CUDA driver up:

```
[root@bright91 ~]# cmsh -c 'device; reboot -n node001..node015'
```

Further details on the basic installation of CUDA for NVIDIA GPUs are given in section 7.4
3. Configuring GPUs For The Cluster Manager

After the GPU software has been installed, the gpusettings submode can be used under device mode for the nodes that have GPUs. This submode allows the cluster administrator to modify GPU values. Keeping the default settings in the submode should be fine, which means configuration of GPU settings can simply be skipped.

However, if needed, then the settings can be modified as described in section 3.13.2 of the Administrator Manual.

(a) on page 130 for NVIDIA GPUs
(b) on page 132 for AMD GPUs.

4. Configuring GPUs For The Workload Manager

A workload manager can be set up from the head node by running:

```
[root@bright91 ~]# cm-wlm-setup
```

This starts up an Ncurses-based configuration. An NVIDIA GPU can be configured via for Slurm using the Setup (Step by Step) option for Slurm (section 7.3.2 of the Administrator Manual).

After configuring the server, submission and client roles for the nodes of the cluster, a screen that asks if GPU resources should be configured is displayed (figure 1.1):

![Figure 1.1: Slurm With cm-wlm-setup: GPU Configuration Entry Screen](image)

Following through brings up a GPU device settings configuration screen (figure 1.2):

![Figure 1.2: Slurm With cm-wlm-setup: GPU Device Settings Configuration Screen](image)

The help text option in the screen gives hints based on the descriptions at https://slurm.schedmd.com/gres.conf.html, and also as seen in Slurm’s man (5) gres.conf.

Figure 1.2 shows 2 physical GPUs on the node being configured. The type is an arbitrary string for the GPU, and each CPU core is allocated an alias GPU device.

The next screen (figure 1.3) allows Nvidia’s CUDA MPS (Multi-Process Service) to be configured:

```
```

© Bright Computing, Inc.
The help text for this screen gives hints on how the fields can be filled in. The number of GPU cores (figure 1.3) for a GPU device can be set.

The rest of the cm-wlm-setup procedure can then be completed.

The regular nodes that had a role change during cm-wlm-setup can then be rebooted to pick up the workload manager (WLM) services. A check via the cmsh command ds should show what nodes need a restart.

**Example**

```
[root@bright91 ~]# cmsh -c 'ds'
node001 ... [ UP ], restart required (cm-wlm-setup: compute role assigned)
node002 ... [ UP ], restart required (cm-wlm-setup: compute role assigned)
...
```

If, for example, the range from node001 to node015 needs to be restarted to get the WLM services running, then it could be carried out with:

**Example**

```
[root@bright91 ~]# cmsh -c 'device; reboot -n node001..node015'
```

More on these attributes can be found in the man pages (man 5 gres.conf).

NVIDIA configuration for Slurm and other workload managers is described in further detail in section 7.5 of the Administrator Manual.

### 5. Running A GPU Slurm Job

A user can be created with, for example, an unimaginative name such as auser:

```
[root@bright91 ~]# cmsh
[bright91]# user add auser
[bright91->user*[auser*]]% set password
enter new password:
retype new password:
[bright91->user*[auser*]]% ..
[bright91->user*]auser% commit
```

The “Hello World” helloworld.cu script from section 8.5.4 of the User Manual can be saved in auser’s directory, and then compiled for a GPU with nvcc:
A `slurmgpuhello.sh` batch script can be built as follows:

```bash
[auser@bright91 ~]$ cat slurmgpuhello.sh
#!/bin/sh
#SBATCH -o my.stdout
#SBATCH --time=30 # time limit to batch job
#SBATCH --ntasks-per-node=1
#SBATCH -p defq # assuming gpu is in defq
#SBATCH --gpus=1
echo -n "hostname is "; hostname
module clear -f
./helloworld
```

It can be submitted as a batch job from the head node:

```bash
[auser@bright91 ~]$ module load slurm
[auser@bright91 ~]$ sbatch slurmgpuhello.sh
Submitted batch job 35
```

The output from submission to a node with a GPU can then be seen:

```bash
[auser@bright91 ~]$ cat my.stdout
hostname is node001
String for encode/decode: Hello World!
String encoded on CPU as: Uryy|-d|yq.
String decoded on GPU as: Hello World!
```

1.5 Optional: Upgrading Python

The version of Python provided by the Linux-based OS distributors typically lags significantly behind the latest upstream version. This is normally a good thing, since the distributors provide integration, and carry out testing to make sure that it works well with the rest of the OS.

Some Bright Cluster Manager tools and packages rely on some of the later features of Python, and by default Bright Cluster Manager installs its own later Python version via a package dependency when needed. So explicit upgrades to Python are not needed for Bright Cluster Manager itself. For example, for CentOS 8.2, the default (distribution) version of Python at the time of writing (January 2021) is 3.6.8, while Bright Cluster Manager by default installs a version 3.7, and deploys version 3.7 when needed.

However, some cluster administrators would also like to have various Python versions available on their cluster for user applications. For example, because users too would like to use later versions for their nicer features. Bright Cluster Manager therefore makes these various versions of Python available for the end user in an integrated manner, as an optional installation.

For example, for Bright Cluster Manager version 9.1, the Python packages that can be installed to provide their associated features are:

- `cm-python36`, for Python 3.6
- `cm-python37`, for Python 3.7

© Bright Computing, Inc.
• *cm-python38*, for Python 3.8

Users can use the `modules` command to switch the environment to the appropriate Python version. For example, to switch to Python 3.8:

```
[root@bright91 ~]# python -V
Python 3.6.8
[root@bright91 ~]# module load python38
[root@bright91 ~]# python -V
Python 3.8.5
```

If the change is carried out correctly, then support is not available for Python-related bugs, but is available for the Bright Cluster Manager-related features.

### 1.6 Running Bright View

To run the Cluster Management GUI (Bright View) on the cluster from a workstation running X11: A recent web browser should be used, and pointed to

https://<head node address>:8081/bright-view/

A suitable web browser is the latest Chrome from Google, but Opera, Firefox, Chromium, and similar should all also just work. The hardware on which the browser runs must be fast enough, and for a reasonable experience, should be roughly equivalent to that of a mid- to high-end desktop of 2016.

The cluster should now be ready for running compute jobs.

For more information:

• This manual, the *Installation Manual*, has more details and background on the installation of the cluster in the next chapters.

• The *Upgrade Manual* describes upgrading from earlier versions of Bright Cluster Manager.

• The *Administrator Manual* describes the general management of the cluster.

• The *User Manual* describes the user environment and how to submit jobs for the end user

• The *Cloudbursting Manual* describes how to deploy the cloud capabilities of the cluster.

• The *Developer Manual* has useful information for developers who would like to program with Bright Cluster Manager.

• The *Machine Learning Manual* describes how to install and configure machine learning capabilities with Bright Cluster Manager.
This chapter introduces some features of Bright Cluster Manager and describes a basic cluster in terms of its hardware.

### 2.1 What Is Bright Cluster Manager?
Bright Cluster Manager 9.1 is a cluster management application built on top of major Linux distributions.

#### 2.1.1 What OS Platforms Is It Available For?
It is available for:

- Red Hat Enterprise Linux Server, CentOS, and Scientific Linux
  - Versions 7.x
  - Versions 8.x (not for Scientific Linux)
- SLES versions:
  - SUSE Enterprise Server 12
  - SUSE Enterprise Server 15
- Ubuntu versions:
  - Bionic Beaver 18.04
  - Focal Fossa 20.04

#### 2.1.2 What Architectures Does It Run On?
Within the distributions listed in section 2.1.1, the Bright Cluster Manager application runs on the following architectures:

- on the x86_64 architecture that is supported by Intel and AMD 64-bit CPUs
- on the arm64 (AArch64) architecture that is supported by ARMv8 CPUs

#### 2.1.3 What OS Platforms Can It Be Managed From?
While Bright Cluster Manager runs directly on the OS platforms listed in section 2.1.1, Bright Cluster Manager can be managed from many more OS platforms, when controlled by these front ends:

- Bright View (section 2.4 of the Administrator Manual): a GUI which conveniently runs on modern desktop web browsers, and therefore on all operating system versions that support a modern browser. This includes Microsoft Windows, MacOS and iOS, and Linux.
- cmsh (section 2.5 of the Administrator Manual): an interactive shell front end that can be accessed from any computing device with a secured SSH terminal access
2.2 Cluster Structure

In its most basic form, a cluster running Bright Cluster Manager contains:

- One machine designated as the head node
- Several machines designated as compute nodes
- One or more (possibly managed) Ethernet switches
- One or more power distribution units (Optional)

The head node is the most important machine within a cluster because it controls all other devices, such as compute nodes, switches and power distribution units. Furthermore, the head node is also the host that all users (including the administrator) log in to in a default cluster. The head node is typically the only machine that is connected directly to the external network and is usually the only machine in a cluster that is equipped with a monitor and keyboard. The head node provides several vital services to the rest of the cluster, such as central data storage, workload management, user management, DNS and DHCP service. The head node in a cluster is also frequently referred to as the master node.

Often, the head node is replicated to a second head node, frequently called a passive head node. If the active head node fails, the passive head node can become active and take over. This is known as a high availability setup, and is a typical configuration (Chapter 18 of the Administrator Manual) in Bright Cluster Manager.

A cluster normally contains a considerable number of non-head, or regular nodes, also referred to as simply nodes. The head node, not surprisingly, manages these regular nodes over the network.

Most of the regular nodes are compute nodes. Compute nodes are the machines that will do the heavy work when a cluster is being used for large computations. In addition to compute nodes, larger clusters may have other types of nodes as well (e.g. storage nodes and login nodes). Nodes typically install automatically through the (network bootable) node provisioning system that is included with Bright Cluster Manager. Every time a compute node is started, the software installed on its local hard drive is synchronized automatically against a software image which resides on the head node. This ensures that a node can always be brought back to a “known state”. The node provisioning system greatly eases compute node administration and makes it trivial to replace an entire node in the event of hardware failure. Software changes need to be carried out only once (in the software image), and can easily be undone. In general, there will rarely be a need to log on to a compute node directly.

In most cases, a cluster has a private internal network, which is usually built from one or multiple managed Gigabit Ethernet switches, or made up of an InfiniBand or Omni-Path fabric. The internal network connects all nodes to the head node and to each other. Compute nodes use the internal network for booting, data storage and interprocess communication. In more advanced cluster setups, there may be several dedicated networks. It should be noted that the external network—which could be a university campus network, company network or the Internet—is not normally directly connected to the internal network. Instead, only the head node is connected to the external network.

Figure 2.1 illustrates a typical cluster network setup.
Most clusters are equipped with one or more power distribution units. These units supply power to all compute nodes and are also connected to the internal cluster network. The head node in a cluster can use the power control units to switch compute nodes on or off. From the head node, it is straightforward to power on/off a large number of compute nodes with a single command.
3

Installing Bright Cluster Manager

This chapter describes in detail the installation of Bright Cluster Manager onto the head node of a cluster. Sections 3.1 and 3.2 list hardware requirements and supported hardware. Section 3.3 gives step-by-step instructions on installing Bright Cluster Manager from a DVD or USB drive onto a head node that has no operating system running on it initially, while section 3.4 gives instructions on installing onto a head node that already has an operating system running on it.

Once the head node is installed, the other, regular, nodes can (network) boot off the head node and provision themselves from it with a default image, without requiring a Linux distribution DVD or USB drive themselves. Regular nodes normally have any existing data wiped during the process of provisioning from the head node, which means that a faulty drive can normally simply be replaced by taking the regular node offline, replacing its drive, and then bringing the node back online, without special reconfiguration. The details of the network boot and provisioning process for the regular nodes are described in Chapter 5 of the Administrator Manual.

The installation of software on an already-configured cluster running Bright Cluster Manager is described in Chapter 12 of the Administrator Manual.

3.1 Minimal Hardware Requirements

The following are minimal hardware requirements, suitable for a cluster of one head node and two regular compute nodes:

3.1.1 Head Node
- x86-64, or ARMv8 CPU
- For running the application software:
  - 4GB RAM for the x86 architecture
  - 16GB RAM for the ARMv8 architecture,
- 80GB diskspace
- 2 Gigabit Ethernet NICs (for the most common Type 1 topology (section 3.3.9))
- DVD drive or USB drive

The value of 4GB RAM for the x86 head node, while technically correct as a minimum for running the application, is generally not a practical minimum when carrying out a standard installation. This is because the standard bare-metal installation (section 3.3) runs best with 8GB RAM.

© Bright Computing, Inc.
3.1.2 Compute Nodes

- x86-64, ARMv8 CPU
- 1GB RAM (at least 4GB is recommended for diskless nodes)
- 1 Gigabit Ethernet NIC

Recommended hardware requirements for larger clusters are discussed in detail in Appendix B.

3.2 Supported Hardware

The following hardware is supported:

3.2.1 Compute Nodes

- Asus
- Atos
- Cavium
- Cisco
- Cray
- Dell EMC
- Fujitsu
- HPE (Hewlett Packard Enterprise)
- Huawei
- IBM
- Lenovo
- NVIDIA DGX
- Oracle
- SGI (ICE X)
- SuperMicro

Other brands are also expected to work, even if not explicitly supported.

3.2.2 Ethernet Switches

- Cisco
- Dell
- HP ProCurve and HP Networking Ethernet Switches
- Huawei
- Netgear
- Nortel
- SuperMicro

Other brands are also expected to work, although not explicitly supported.
3.2.3 Power Distribution Units
- APC (American Power Conversion) Switched Rack PDU

Other brands with the same SNMP MIB mappings are also expected to work, although not explicitly supported.

3.2.4 Management Controllers
- HP iLO 1/2/3
- iDRAC
- IPMI 1.5/2.0
- CIMC
- Redfish v1

3.2.5 InfiniBand And Similar
- Intel Omni-Path
- Mellanox HCAs, and most other InfiniBand HCAs
- Mellanox, Voltaire, Flextronics InfiniBand switches
- Most other InfiniBand switches

3.2.6 GPUs
- AMD Radeon GPUs, as listed at https://support.amd.com/en-us_kb-articles/Pages/Radeon-Software-for-Linux-Release-Notes.aspx
- NVIDIA Tesla with latest recommended drivers
- NVIDIA GeForce and other older generations are mostly supported. Bright Computing can be consulted for details.
- NVIDIA DGX servers and workstations are supported.

3.2.7 MICs
- Xeon Phi: All Xeon Phi processor versions from Knights Landing onward. PCI-e coprocessor versions of Xeon Phi do not have direct integration with Bright Cluster Manager from Bright Cluster Manager version 8.2 onwards.

3.2.8 RAID
Software or hardware RAID are supported. Fake RAID is not regarded as a serious production option and is supported accordingly.

3.3 Head Node Installation: Bare Metal Method

A bare metal installation, that is, installing the head node onto a machine with no operating system on it already, is the recommended option. This is because it cannot run into issues from an existing configuration. An operating system from one of the ones listed in section 2.1 is installed during a bare metal installation. The alternative to a bare metal installation is the add-on installation of section 3.4.

Just to be clear, a bare metal installation can be a physical machine with nothing running on it, but it can also a be virtual machine—such as a VMware, VirtualBox, or KVM instance—with nothing running on it. Virtual instances may need some additional adjustment to ensure virtualization-related settings

© Bright Computing, Inc.
are dealt with correctly. Details on installing Bright Cluster Manager onto virtual instances can be found in the Bright Cluster Manager Knowledge Base at http://kb.brightcomputing.com.

To start a physical bare metal installation, the time in the BIOS of the head node is set to local time. The head node is then made to boot from DVD or USB, which can typically be done by appropriate keystrokes when the head node boots, or via a BIOS configuration.

**Special steps for installation from a bootable USB device:** If a bootable USB device is to be used, then the instructions within the Bright ISO, in the file README.BRIGHTUSB should be followed to copy the ISO image over to the USB device. After copying the ISO image, the MD5 checksum should be validated to verify that the copied ISO is not corrupt. This is important, because corruption is possible in subtle ways that may affect operations later on, and in ways that are difficult to uncover.

### 3.3.1 ISO Boot Menu

If booting from a DVD or USB drive, then a pre-installer menu called the *ISO boot menu* first loads up. The menu is automatically presented as either a BIOS (or UEFI legacy BIOS emulation) version, or as a UEFI version (figures 3.1 and 3.2) and effectively function in the same way.

```bash
> /boot/kernel xlogin dwdinstall nokeymap initrd=/boot/rootfs.cfg root=/dev/ram0 rw uga=normal blacklist=nouveau metconf=eth0:static,10.141.161.253,10.141.254.255,255.255.0.0:secretpass
```

**Figure 3.1: Installation: ISO Boot Menu For Legacy BIOS**

```bash
> Bright Cluster Manager Master Installation Environment

Boot from harddrive
Install Bright Cluster Manager (Graphical)
Install Bright Cluster Manager (Text mode)
Memtest
```

**Figure 3.2: Installation: ISO Boot Menu For UEFI**

The ISO Boot menu offers a default option of booting from the hard drive, with a countdown to
starting the hard drive boot. To install Bright Cluster Manager, the countdown should be interrupted by selecting the option of “Install Bright Cluster Manager (Graphical)” instead.

Selecting the option allows kernel parameter options to be provided to the installer.

Default kernel parameter options are provided so that the administrator can simply press the enter key to go straight on to start the installer, and bring up the welcome screen (section 3.3.2).

**Optional: The netconf Custom Kernel Parameter**

A non-default kernel parameter option is netconf. Setting configuration values for this option configures login and network settings for the cluster manager, and also means that SSH and VNC servers are launched as the welcome screen (section 3.3.2) of the Bright Cluster Manager installer starts up. This then allows the cluster administrator to carry out a remote installation, instead of having to remain at the console.

The netconf option can be set for both the GUI and text installer installation.

The netconf custom kernel parameter requires 3 settings:

1. a setting for the external network interface that is to be used. For example: eth0 or eth1.

2. a setting for the network configuration of the external network, to be explained soon. The network configuration option can be built either using static IP addressing or with DHCP.

   - For static IP addressing, the network configuration format is comma-separated:
     \[ \text{static:<IP address>,<gateway address>,<netmask>} \]
   - For DHCP addressing, the format is simply specified using the string dhcp.

3. a setting for the password, for example secretpass, for the login to the cluster manager that is about to be installed.

**Example**

With static IP addressing:

\[ \text{netconf=eth0:static:10.141.161.253,10.141.255.254,255.255.0.0:secretpass} \]

**Example**

With DHCP addressing:

\[ \text{netconf=eth0:dhcp:secretpass} \]

A remote installation can alternatively be carried out later on without setting netconf, by using the text mode installer to set up networking (section 3.5), or by using GUI mode installer Continue remotely option (figure 3.4).

**3.3.2 Welcome Screen**

The welcome screen is shown in figure 3.3.

© Bright Computing, Inc.
An administrator who would like to simply start installation can click on the Start installation button at the left side of the screen.

### 3.3.3 Software Licenses

The next screen that is displayed asks the user to agree to the Bright Cluster Manager license (figure 3.4):
3.3 Head Node Installation: Bare Metal Method

A similar screen after that asks the user to agree to the Base Distribution EULA. This is the end user license agreement for the distribution that is to be used as the base upon which Bright Cluster Manager is to run.

Options During The Installation Screens
As is seen in figure 3.4, the main screens during installation have the following clickable options along the bottom of the screen:

- **View Licence**: shows a summary of the licence attributes.
- **Load config**: allows an existing configuration file to be loaded and used by the installation. This option is available only during the first few screens.
- **Show config**: allows any already loaded configuration file to be displayed. There is a default configuration loaded by default, with values that may suit the cluster already. However, the defaults are not expected to be optimal, and may not even work for the actual physical configuration.
- **Continue remotely**: allows the administrator to leave the console and access the cluster from a remote location. This can be useful for administrators who prefer to avoid working inside a noisy cold data center.
- **Back**: if not grayed out, allows the administrator to go back a step in the installation.
- **Next**: allows the administrator to proceed to the next screen.

Figure 3.4: Bright Cluster Manager Software License
3.3.4 Kernel Modules Configuration

The Kernel Modules screen (figure 3.5) shows the kernel modules recommended for loading based on a hardware probe:

![Kernel Modules](image)

**Figure 3.5: Kernel Modules Recommended For Loading After Probing**

Changes to the modules to be loaded can be entered by reordering the loading order of modules, by removing modules, and adding new modules. Clicking the + button opens an input box for adding a module name and optional module parameters (figure 3.6). The module can be selected from a built-in; it can be automatically extracted from a .deb or .rpm package; or it can simply be selected by selecting an available .ko kernel module file from the filesystem.

A module can also be blacklisted, which means it is prevented from being used, by clicking on the \( \bigcirc \) button. This can be useful when replacing one module with another.
Clicking Next then leads to the “Hardware info” overview screen, described next.

3.3.5 Hardware Info
The Hardware Info screen (figure 3.7) provides an overview of detected hardware based on the hardware probe used to load kernel modules. If any hardware is not detected at this stage, then the Back button can be used to go back to the Kernel Modules screen (figure 3.5) to add the appropriate modules, and then the Hardware Information screen can be returned to, to see if the hardware has been detected.

Figure 3.7: Hardware Overview Based On Hardware Detection Used For Loading Kernel Modules

© Bright Computing, Inc.
Clicking Next in the Hardware Info screen leads to the Installation source configuration screen, described next.

### 3.3.6 Installation Source

The installation source screen (figure 3.8) detects the available DVD-ROM devices.

![DVD Selection](image)

The administrator must select the correct device to continue the installation. Optionally, a media integrity check can be set. Clicking on the Next button starts the media integrity check, if it was set. The media integrity check can take about a minute to run. If all is well, then the “Cluster settings” setup screen is displayed, as described next.

### 3.3.7 Cluster Settings

The General cluster settings screen (figure 3.9) provides an overview of some mandatory and optional settings.
Figure 3.9: General cluster settings

Settings that can be provided in this screen are:

- **Cluster name**
- **Administrator email**: Where mail to the administrator goes. This need not be local.
- **Time zone**
- **Time servers**: The defaults are pool.ntp.org servers. A time server is recommended to avoid problems due to time discrepancies between nodes.
- **Nameservers**: One or more DNS servers can be set.
- **Search domains**: One or more domains can be set.
- **Environment modules**: Traditional Tcl modules are set by default. Lmod is an alternative.

### 3.3.8 Workload Management Configuration

The “Workload manager” configuration screen (figure 3.10) allows a supported workload manager to be selected. A workload management system is highly recommended to run multiple compute jobs on a cluster.
26 Installing Bright Cluster Manager

Figure 3.10: Workload Management Setup

If no workload manager is selected here, then it can be installed later on, after the cluster installation without the workload manager has been done. Details on installing a workload manager later on are given in Chapter 7 on workload management of the *Administrator Manual*.

If a workload management system is selected, then the number of slots per node is set by default to 8.

The head node can also be selected for use as a compute node, which can be a sensible choice on small clusters.

Clicking Next on this screen leads to the Network topology screen.

### 3.3.9 Network Topology

Regular nodes are always located on an internal network, by default called *Internalnet*.

The Network Topology screen allows selection of one of three different network topologies (figures 3.11, 3.12, 3.13):
A type 1 network: has its nodes connected on a private internal network. This is the default network setup. In this topology, a network packet from a head or regular node destined for any external network that the cluster is attached to, by default called Externalnet, can only reach the external network by being routed and forwarded at the head node itself. The packet routing for Externalnet is configured at the head node.
A **type 2** network: has its nodes connected via a router to a public network. In this topology, a network packet from a regular node destined for outside the cluster does not go via the head node, but uses the router to reach a public network. Packets destined for the head node however still go directly to the head node. Any routing for beyond the router is configured on the router, and not on the cluster or its parts. Care should be taken to avoid DHCP conflicts between the DHCP server on the head node and any existing DHCP server on the internal network if the cluster is being placed within an existing corporate network that is also part of Internalnet (there is no Externalnet in this topology). Typically, in the case where the cluster becomes part of an existing network, there is another router configured and placed between the regular corporate machines and the cluster nodes to shield them from effects on each other.
A type 3 network: has its nodes connected on a routed public network. In this topology, a network packet from a regular node, destined for another network, uses a router to get to it. The head node, being on another network, can only be reached via a router too. The network the regular nodes are on is called Internalnet by default, and the network the head node is on is called Managementnet by default. Any routing configuration for beyond the routers that are attached to the Internalnet and Managementnet networks is configured on the routers, and not on the clusters or its parts.

Selecting the network topology helps decide the predefined networks on the Networks settings screen later (figure 3.17). Clicking Next here leads to the Head node settings screen, described next.

### 3.3.10 Head Node Settings

The head node settings screen (figure 3.14) can be used to set, for the head node:

- the hostname
- the password
- the hardware manufacturer
Clicking Next leads to the Compute node settings screen, described next.

### 3.3.11 Compute Nodes Settings

The compute nodes settings screen (figure 3.15) can be used to set, for the compute nodes:

- the number of racks used to host the nodes
- the number of nodes
- the naming format for the nodes. This consists of:
  - the node base name (default: node)
  - the node start number (default: 1)
  - the node digits (default: 3)

By default therefore, the first compute node takes the name node001, the second compute node takes the name node002, and so on.

- the hardware manufacturer
3.3 Head Node Installation: Bare Metal Method

3.3.12 BMC Configuration

The BMC (Baseboard Management Controller) screen (figure 3.16) configures BMCs compatible with IPMI, iDRAC, iLO, or CIMC.

The BMCs can be configured for the head or compute nodes.

Figure 3.15: Head node settings

Clicking Next leads to the BMC configuration screen, described next.
If the administrator confirms that the nodes are to use BMCs (Baseboard Management Controllers) that are compatible with IPMI, iLO, CIMC, iDRAC, or Redfish, then the BMC network options appear. By default, for the compute nodes, the BMC is automatically configured.

For a Type 1 network, the head node BMC is often connected to an ethernet segment that has the external network running on it, while the BMCs on the compute nodes are normally connected to an ethernet segment that has the internal network on it.

Once a network associated with the ethernet segment is chosen, it means that further BMC-related networking values can be set for the BMCs.

A new Layer 3 IP subnet can be created for BMC interfaces.

The BMC interface can be configured as a shared physical interface with an already existing network interface. However this can in some cases cause problems during early system BIOS checks. A dedicated physical BMC interface is therefore recommended.

If a BMC is configured, then the BMC password is set to a random value. Retrieving and changing a BMC password is covered in section 3.7.2 of the Administrator Manual. BMC configuration is discussed further in section 3.7 of the Administrator Manual.

Clicking Next leads to the Networks screen, described next.

### 3.3.13 Networks

The Networks configuration screen (figure 3.17) displays the predefined list of networks, based on the selection of network topology and BMC networks made in the earlier screens.
3.3 Head Node Installation: Bare Metal Method

The Networks configuration screen allows the parameters of the network interfaces to be configured via tabs for each network. In addition to any BMC networks:

For a type 1 setup, an external network and an internal network are always defined.

For a type 2 setup, an internal network is defined but no external network is defined.

For a type 3 setup, an internal network and a management network are defined.

Thus, for a type 1 network, for example, the networking details:
- for externalnet correspond to the details of the head node external network interface.
- for internalnet correspond to the details of how the compute nodes are configured.
- for a BMC network correspond to the details of how the BMC is connected

Additional custom networks can be added in the Networks configuration screen by clicking on the button.

Clicking Next in this screen validates all network settings. Invalid settings for any of the defined networks cause an alert to be displayed, explaining the error. A correction is then needed to proceed further. Settings may of course be valid, but incorrect—the validation is merely a sanity check. It may be wise for the cluster administrator to check with the network specialist that the networks that have been configured are set up as really intended.

If all settings are valid, then the Next button brings the installation on to the Head node interfaces screen, described next.

© Bright Computing, Inc.
3.3.14 Head node interfaces

The Head node interfaces screen (figure 3.18) allows a review of the head node network interfaces and their proposed settings.

If a BMC network is to be shared with a regular network, then an alias interface is shown too. In figure 3.18 an alias interface, ens3:ipmi, is shown.

Interfaces can be created or removed.

Dropdown selection allows the proposed values to be changed. It is possible to swap network interfaces with dropdown selection.

Clicking the Next button brings the installation on to the Compute node interfaces screen, described next.

3.3.15 Compute node interfaces

The Compute node interfaces screen (figure 3.19) allows a review of the compute node network interfaces and their proposed settings.
IP addresses for each of the interfaces on all nodes will be assigned automatically from a consecutive range of IP addresses. The starting address in the range is determined by adding the specified offset to the base address of the network. For example, a selected network base address 10.141.0.0 with an IP offset of 0.0.0.0 will cause the first node to be assigned 10.141.0.1, and the second node 10.141.0.2.

Example

Taking the case where BOOTIF and IPMI interfaces would have IP addresses on the same network with the default IP offset:

Then, on a cluster of 10 nodes, a modified IPMI IP offset of 0.0.0.20 means:

- the BOOTIF interfaces stay on 10.141.0.1,\ldots,10.141.0.10 while

© Bright Computing, Inc.
• the IPMI interfaces range from 10.141.0.21,..,10.141.0.30

Clicking the Next button brings the installation on to the Disk layout screen, described next.

3.3.16 Disk Layout

The partitioning layout XML schema is described in detail in Appendix D of the Administrator Manual. Within the Disk layout configuration screen (figure 3.20):

- the administrator must select the drive on the head node where the cluster manager is to be installed.
- the administrator must set the disk partitioning layout for the head node and regular nodes with the two options: Head node disk layout and Compute nodes disk layout.
  - The head node by default uses
    - one big partition if it has a drive size smaller than about 500GB
    - several partitions if it has a drive size greater than or equal to about 500GB.
  - The regular node by default uses several partitions.
  
A partitioning layout other than the default can be selected for installation from the drop-down boxes for the head node and regular nodes. Possible partitioning options include RAID, failover, and STIG-compliant schemes.
  
- A custom partitioning layout specification file can be added with the icon.
The partitioning layout can be edited with the icon. This brings up a screen (figure 3.21) that allows the administrator to view and change layout values within the layout’s configuration XML file:

![Image](image_url)

**Figure 3.21: Edit Disk Partitioning**

The XML schema is described in Appendix D.1 of the *Administrator Manual*.

The head node partitioning layout is the only installation setting that cannot easily be changed after the completion (section 3.3.19) of installation. It should therefore be decided upon with care.

By default, Bright Cluster Manager mounts filesystems on the head node with ACLs set and extended attributes set.

The XML schema allows the definition of a great variety of layouts in the layout’s configuration XML file:

**Example**

1. for a large cluster or for a cluster that is generating a lot of monitoring or burn data, the default partition layout partition size for /var may fill up with log messages because log messages are usually stored under /var/log/. If /var is in a partition of its own, as in the default head node partitioning layout presented when the hard drive is about 500GB or more, then providing a larger size of partition than the default for /var allows more logging to take place before /var is full. Modifying the value found within the `<size>` tags associated with that partition in the XML file modifies the size of the partition that is to be installed. This can be conveniently done from the front end shown in figure 3.21.

2. the administrator could specify the layout for multiple non-RAID drives on the head node using one `<blockdev>` tag pair within an enclosing `<device>` tag pair for each drive.

Clicking **Next** on the Disk layout screen leads to the Additional software screen, described next.
3.3.17 Additional Software

The Additional software screen (figure 3.22) displays software that can be added to the cluster if it was provided with the installer. The software is only provided to the installer if selected when generating the installer ISO.

![Additional software screen](image)

Clicking Next on the Additional software screen leads to the Summary screen, described next.

3.3.18 Summary

The Summary screen (figure 3.23), summarizes some of the installation settings and parameters configured during the previous stages.

© Bright Computing, Inc.
3.3 Head Node Installation: Bare Metal Method

3.3.19 Deployment

The Deployment screen (figure 3.24) shows the progress of the deployment. It is not possible to navigate back to previous screens once the installation has begun. The installation log can be viewed in detail by clicking on Install log.

The Reboot button restarts the machine. Alternatively, the head node can be set to automatically reboot when deployment is complete.

During the reboot, the BIOS boot order may need changing, or the DVD may need to be removed, in order to boot from the hard drive on which Bright Cluster Manager has been installed.
After rebooting, the system starts and presents a login prompt. After logging in as root using the password that was set during the installation procedure, the system is ready to be configured.

3.3.20 Licensing And Further Configuration

The administrator with no interest in the add-on method of installation can skip on to installing the license (Chapter 4). After that, the administrator can look through the Administrator Manual, where tools and concepts used with Bright Cluster Manager are introduced, so that further cluster configuration can be carried out.

3.4 Head Node Installation: Add-On Method

An add-on installation, in contrast to the bare metal installation (section 3.3), is an installation that is done onto a machine that is already running one of the supported distributions of section 2.1. The installation of the distribution can therefore be skipped for this case. However, unlike the bare metal installation, the add-on is not recommended for inexperienced cluster administrators. This is because of the following reasons:

- The installation configuration may conflict with what has already been installed. The problems that arise can always be resolved, but an administrator that is not familiar with Bright Cluster Manager should be prepared for troubleshooting.
- Additional repositories typically need to be added (section 3.4.1).
- Dependency and deprecated issues may need workarounds.
3.4 Head Node Installation: Add-On Method

- After the add-on installation has been carried out on the head node, two software images for the regular nodes must still be installed into directories on the head node. These are the node-installer image and the software image, which are images used by regular nodes after they are powered up.
  - The node-installer image is an image that is run by a node after the Linux kernel and modules have been loaded onto the node near the end of the standard pre-init state. The node is in the node-installer stage when this image is being run. The node-installer has tools that are used to deploy the node. The deployment is done with the help of Bright Cluster Manager to help work out how the node can be provisioned, and Bright Cluster Manager also helps carry out further configuration so that the node can be managed. The node-installer ends by starting the init system for the node.
  - The software image is provisioned to regular nodes as part of the node-installer stage, and is most of the software that is to be installed on the regular node.

The creation and installation of a software image requires some understanding of the Linux operating system as well as Bright Cluster Manager. Software image management is described more detail in section 12.6 of the Administrator Manual.

3.4.1 Prerequisites

For the add-on method

- The operating systems validated and supported for add-on installation in Bright Cluster Manager 9.1 are currently RHEL7 and CentOS8/RHEL8.
- The operating system must obviously follow system administration best practices so that it works properly with the official distribution, when Bright Cluster Manager is added on.
- The items of software that Bright Cluster Manager adds must be allowed to overrule in any conflict with what is already installed, or the end result of the installation cannot be supported.
- MariaDB must be installed. If MariaDB is already deployed, then care must be taken to ensure that any configuration conflict with the Bright Cluster Manager configuration is resolved.
- It is highly recommended to use a freshly-installed distribution rather than one which is already in use.
- SELinux must be disabled. Typically this is done by setting SELINUX=disabled in the file /etc/selinux/config.
- A product key is needed.
- There must be repository access to the supported distribution.
  - Internet access makes up-to-date repository access possible. RHEL repository access requires a subscription from Red Hat (Chapter 5).
  - For high-security environments without internet access, an alternative is to mount a DVD device or ISO image to the head node containing a local repository snapshot of the parent distribution, and specify the repository location when running the installation command. Configuring a local repository is described in section 12.6.3 of the Administrator Manual.
  - The command `yum repolist` shows the list of enabled repositories.
    For RHEL7, repository access is required to:
    * `rhel-7-server-rpms`
    * `rhel-7-server-optional-rpms`
The repositories can be enabled with the RHEL7 subscription manager using a command of the form:

```
subscription-manager repos --enable=rhel-7-server-rpms --enable=rhel-7-server-optional...
```

EPEL repository access must also be enabled:

**Example**

```
yum install epel-release
```

For CentOS8, repository access (seen with `yum repolist`) may be needed to:

- appstream (CentOS Linux 8 - AppStream)
- baseos (CentOS Linux 8 - BaseOS)
- cm_ml_rhel8-9.1-updates (Cluster Manager 9.1 Machine Learning for Red Hat Enterprise Linux 8 - Updates)
- cm_rhel8-9.1-updates (Cluster Manager 9.1 for Red Hat Enterprise Linux 8 - Updates)
- epel (yum install epel-release, installs modular repo too)
- epel_modular (Extra Packages for Enterprise Linux Modular 8 - x86_64)
- extras (CentOS Linux 8 - Extras)
- powertools (CentOS Linux 8 - PowerTools)

### 3.4.2 Installing The Installer

To carry out an add-on installation, the `bright-installer` package and its dependencies must be installed with a package installer.

The `bright-installer` package and its dependencies can be obtained from a Bright Cluster Manager installation DVD, in the directory `/addon/`. Additionally, there must be access to the Linux distribution’s repositories as well the EPEL repository in the case of RHEL-based distributions. All packages under the `addon/` directory can be installed as follows:

```
[root@rhel7 ~]# cd <path to mounted Bright DVD addon directory>
[root@rhel7 addon]# rpm -i cm-config-* #install cm-config-cm cm-config-yum
[root@rhel7 addon]# yum install bright-installer* #installs bright-installer + dependencies
```

If installing to Ubuntu, then instead of the `yum` command, the `dpkg -i` command is used to install the `.deb` packages that are in the `addon` directory.

Some additional dependencies may be installed by the package manager. Installation progress is logged in `/var/log/install-bright.log`.

### 3.4.3 Running The Installer

**The Help Text For install-bright**

The installer is run with the command `install-bright`. Running it with the `-h` option displays the following help text:

```
[root@bright91 ~]# install-bright -h
usage: install-bright [-h] (-d <path to dvd> | -n | -l) [-c <path to config>]
  [-x <pkg1-name,pkg2-name,..,pkgN-name> | @excludefile]
  [--answers-file ANSWERS_FILE] [--no-progress]
```

© Bright Computing, Inc.
3.4 Head Node Installation: Add-On Method

optional arguments:
-h, --help    show this help message and exit
-d <path to dvd>, --fromdvd <path to dvd>    Path to a Bright DVD/USB device, or mounted directory or path to a Bright ISO file
-n, --network    Install over network
-l, --localrepo    Do not update any repo configuration, just use existing repository settings
-c <path to config>, --useconfig <path to config>    Use predefined config file
-x <pkg1-name,pkg2-name,...,pkgN-name> | @excludefile, --excludepackages <pkg1-name,pkg2-name,...,pkgN-name> | @excludefile    Comma-separated list of packages to exclude from being installed or a path to a file with the exclude list
-v, --verbose    Turn on verbose mode
-m, --minimalinstall    Only install Bright packages and dependencies
-f, --ignoreconflicts    Ignore package conflicts check
-s, --skippackagesetup    Skip repository configuration, validation and package installation
--no-hpc    Disable installing HPC packages.
--dry-run
--answers-file ANSWERS_FILE
--no-progress

1. install-bright -n
2. install-bright -l
3. install-bright -d /dev/sr0
4. install-bright -d /media/Bright-DVD
5. install-bright -d /tmp/bright9.1-rhel8u2.iso
6. install-bright -d /tmp/bright9.1-rhel8u2.iso -x libXTrap,xorg-x11-resutils

Usage Examples For install-bright
- Install Bright Cluster Manager directly from the Bright Computing repositories over the internet:
  
  install-bright -n

- Install Bright Cluster Manager using a Bright Computing DVD as the package repository:

  install-bright -d /dev/sr0

- Install Bright Cluster Manager using a Bright Computing ISO as the package repository:

  install-bright -d /tmp/bright-centos8.iso

- Install Bright Cluster Manager from a local repository which has already been configured. This also assumes that the repository configuration files for zypper/YUM/APT use are already in place:

  install-bright -l

An Installation Run For install-bright
The most common installation option is with an internet connection. Any required software packages are asked for at the start:

Example
Installing Bright Cluster Manager

[root@rhel7 ~]# install-bright -n

Please install the follow pre-requisites
----------------------------------------
createrepo
[root@rhel7 ~]# yum install createrepo
...

After all the packages are installed on the head node, the installer can be run again. It checks for some software conflicts, and warns about the ones it runs into:

Example

[root@rhel7 ~]# install-bright -n

INFO/ERROR/WARNING:
--------------------
WARNING:
A DHCP daemon is already running. Bright Cluster Manager provides a customized DHCP server, and will update the 'dhcpp' configuration files. It is highly recommended that you stop your existing DHCP server, and let Bright Cluster Manager configure your dhcp server.

You can also choose to ignore this message, and proceed with the existing DHCP server, which may or may not work.
--------------------
Continue(c)/Exit(e)? e
[root@rhel7 ~]# service dhcp stop
Shutting down dhcpd: [ OK ]

Having resolved potential software conflicts, the product key (supplied by Bright Computing or its vendor) is supplied:

Example

[root@rhel7 ~]# install-bright -n

Bright Cluster Manager Product Key Activation
---------------------------------------------
Product key [XXXXX-XXXXX-XXXXX-XXXXX-XXXXX]: 001323-134122-134134-314384-987986
...
License Parameters
-------------------
Country Name (2 letter code) []: US
State or Province Name (full name) []: CA
Locality (city) []: San Francisco
Organization Name (e.g. company) []: Bright
Organization Unit (e.g. department) []: Development
Cluster Name []: bright91
MAC address [??:??:??:??:??:??]: 08:B8:BD:7F:59:4B

Submit certificate request to Bright Computing? [y(yes)/n(no)]: y

Contacting license server ... License granted.
License has been installed in /cm/local/apps/cmd/etc/

© Bright Computing, Inc.
The software license is displayed, and can be clicked through. Some warning is given about the configuration changes about to take place:

Please be aware that the Bright Cluster Manager will re-write the following configuration on your system:
- Update network configuration files.
- Start a DHCP server on the management network.
- Update syslog configuration

The software configuration sections is reached. Default Bright Cluster Manager values are provided, but should normally be changed to appropriate values for the cluster. Questions asked are:

Management network parameters
--------------------------------
Network Name [internalnet]:
Base Address [10.141.0.0]:
    Netmask Bits [16]:
Domain Name [eth.cluster]:

Management interface parameters
--------------------------------
Interface Name [eth0]:
IP Address [10.141.255.254]:

External network parameters
----------------------------
Network Name [externalnet]:
    Base Address [DHCP]:
    Netmask Bits [24]:
    Domain Name []: cm.cluster

External interface parameters
----------------------------
Interface Name [eth1]:
IP Address [DHCP]:

External name servers list (space separated)
--------------------------------------------
List [10.150.255.254]:

Root password
--------------
Please enter the cluster root password:

MySQL root password
---------------------
Please enter the MySQL root password:

The Bright Cluster Manager packages are then installed and configured. The stages include, towards the end:

Example

Using build config /root/.brightcm/build-config.xml

© Bright Computing, Inc.
Installation progress is logged in /var/log/install-bright.log

***

- Operating system general configuration [OK]
- Updating database authentication [OK]
- Setting up MySQL database [OK]
- Starting syslog [OK]
- Initializing cluster management daemon [OK]
- Generating admin certificates [OK]
- Initializing HPC components [OK]
- Starting cluster management daemon [OK]
- Setting up httpd daemon [OK]
- Setting up Ldap services [OK]

If all is well, a congratulatory message then shows up, informing the administrator that Bright Cluster Manager has been installed successfully, that the host is now a head node.

Installing The Software Image For Regular Nodes After The `install-bright` Installation Run

The `install-bright` command does not create the node-installer or the node software image. A functional cluster needs regular nodes to work with the head node. The regular nodes at this point of the installation still need to be set up. To do that, a software image (section 2.1.2 of the Administrator Manual) and a node-installer image must now be created for the regular nodes on the head node. The regular nodes, when booting, use these images when they boot up to become a part of the cluster. How to create an image from a base tar image with `cm-create-image` is described in section 12.6 of the Administrator Manual.

- A default node software image can be created with:

  ```
  cm-create-image --imagename default-image --image-type node-installer --fromarchive
  <path to base archive> ...
  ```

- A default node-installer image can be created with:

  ```
  cm-create-image --imagename node-installer --image-type node-installer --fromarchive
  <path to base archive> ...
  ```

Once the head node, the software image, and the node-installer image have been built, the head node installation is complete. The cluster is then essentially at the same stage as that at the end of section 3.3.19 of the bare metal installation, except for that the software image is possibly a more customized image than the default image provided with the bare-metal installation.

The Build Configuration Files

This section is mainly intended for deploying installations that have been pre-configured by the administrator. It can therefore be skipped in a first reading.

The build configuration file of a cluster contains the configuration scheme for a cluster. The bare metal and add-on installations both generate their own, separate build configuration files, stored in separate locations.

Most administrators do not deal with a build configuration file directly, partly because a need to do this arises only in rare and special cases, and partly because it is easy to make mistakes. An overview, omitting details, is given here to indicate how the build configuration file relates to the installations carried out in this chapter and how it may be used.

The bare metal build configuration file: The file at:

```
/root/cm/build-config.xml
```
on the head node contains cluster configuration settings and the list of distribution packages that are installed during the bare metal installation. Once the installation has completed, this file is static, and does not change as the cluster configuration changes.

**The add-on installation build configuration file:** Similarly, the file at:
\[
/root/.brightcm/build-config.xml
\]
contains configuration settings. However, it does not contain a list of distribution packages. The file is created during the add-on installation, and if the installation is interrupted, the installation can be resumed at the point of the last confirmation prior to the interruption. This is done by using the `-c` option to `install-bright` as follows:

**Example**

```
install-bright -c /root/.brightcm/build-config.xml
```

**Both original “build” configuration XML files can be copied and installed via the `-i` initialize option:** For example:

```
service cmd stop
cmd -i build-config-copy.xml  #reinitializes CMDaemon from scratch
service cmd start
```

overwrites the old configuration. It means that the new cluster presents the same cluster manager configuration as the old one did initially. This can only be expected to work with identical hardware because of hardware dependency issues.

**An XML configuration file can be exported via the `-x` option to `cmd`:** For example:

```
service cmd stop
cmd -x myconfig.xml
service cmd start
```

Exporting the configuration is sometimes helpful in examining the XML configuration of an existing cluster after configuration changes have been made to the original installation. This “snapshot” can then, for example, be used to customize a `build-config.xml` file in order to deploy a custom version of Bright Cluster Manager.

An exported configuration cannot replace the original bare-metal `build-config.xml` during the installation procedure. For example, if the original bare-metal file is replaced by the exported version by opening up another console with `alt-f2`, before the point where the “Start” button is clicked (figure 3.23), then the installation will fail. This is because the replacement does not contain the list of packages to be installed.

The exported configuration can however be used after a distribution is already installed. This is true for a head node that has been installed from bare-metal, and is also true for a head node that has undergone or is about to undergo an add-on installation. This is because a head node does not rely on a packages list in the XML file in the case of

- after a bare-metal installation, and
- before or after an add-on installation.

These possibilities for an export file are indicated by the following table:
### 3.5 Enabling Remote Browser-Based Installation Via The Text Mode Installer

When carrying out an installation as in section 3.3, the installer is normally run on the machine that is to be the head node of the cluster. For RHEL7 and derivatives, for Ubuntu, and for SLES, a text mode installer is presented as an alternative to the GUI installer (figures 3.1 and 3.2).

The text mode installer is a very minimal installer compared with the GUI installer. The GUI installation is therefore usually preferred.

However, in some cases the GUI installation can fail to start. For example, if X is not working correctly for some reason on the head node.

A way to still run a GUI installation is then to first run the text mode installer, and use it to run the Remote Install option from its main menu (figure 3.25):

![Figure 3.25: Ncurses Remote Installation Option](image)

This then sets up network connectivity, and provides the cluster administrator with a remote URL (figure 3.26):

<table>
<thead>
<tr>
<th>Install Type</th>
<th>Before or After Installation</th>
<th>Cluster Installs from Export?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bare Metal</td>
<td>Before</td>
<td>No</td>
</tr>
<tr>
<td>Bare Metal</td>
<td>After</td>
<td>Yes</td>
</tr>
<tr>
<td>Add-on Install</td>
<td>Before</td>
<td>Yes</td>
</tr>
<tr>
<td>Add-on Install</td>
<td>After</td>
<td>Yes</td>
</tr>
</tbody>
</table>
A browser that is on a machine with connectivity to the head node can then use the provided remote URL. This then brings up the GUI installer within the browser.

An alternative to running the text mode installer to obtain the remote URL is to use the `netconf` kernel parameter instead. Details on configuring this are given in section 3.3.1.
Licensing Bright Cluster Manager

This chapter explains how a Bright Cluster Manager license is viewed, verified, requested, and installed.

Typically, for a new cluster that is purchased from a reseller, the cluster may have Bright Cluster Manager already set up on it.

Bright Cluster Manager can be run with a temporary, or evaluation license, which allows the administrator to try it out. This typically has some restrictions on the period of validity for the license, or the number of nodes in the cluster. The evaluation license also comes with the online ISO download for Bright Cluster Manager, which is available for product key owners via http://customer.brightcomputing.com/Download

The other type of license is the full license, which is almost always a subscription license. Installing a full license allows the cluster to function without the restrictions of the evaluation license. The administrator therefore usually requests a full license, and installs it. This normally only requires the administrator to:

- Have the product key at hand
- Run the request-license script on the head node

The preceding takes care of the licensing needs for most administrators, and the rest of this chapter can then usually conveniently be skipped.

Administrators who would like a better background understanding on how licensing is installed and used in Bright Cluster Manager can go on to read the rest of this chapter.

CMDaemon can run only with an unexpired evaluation or unexpired full license. CMDaemon is the engine that runs Bright Cluster Manager, and is what is normally recommended for further configuration of the cluster. Basic CMDaemon-based cluster configuration is covered in Chapter 3 of the Administrator Manual.

Any Bright Cluster Manager installation requires a license file to be present on the head node. The license file details the attributes under which a particular Bright Cluster Manager installation has been licensed.

Example

- the “Licensee” details, which include the name of the organization, is an attribute of the license file that specifies the condition that only the specified organization may use the software
- the “Licensed nodes” attribute specifies the maximum number of nodes that the cluster manager may manage. Head nodes are also regarded as nodes for this attribute.
• the “Expiration date” of the license is an attribute that sets when the license expires. It is sometimes set to a date in the near future so that the cluster owner effectively has a trial period. A new license with a longer period can be requested (section 4.3) after the owner decides to continue using the cluster with Bright Cluster Manager.

A license file can only be used on the machine for which it has been generated and cannot be changed once it has been issued. This means that to change licensing conditions, a new license file must be issued.

The license file is sometimes referred to as the cluster certificate, or head node certificate, because it is the X509v3 certificate of the head node, and is used throughout cluster operations. Its components are located under /cm/local/apps/cmd/etc/. Section 2.3 of the Administrator Manual has more information on certificate-based authentication.

4.1 Displaying License Attributes

Before starting the configuration of a cluster, it is important to verify that the attributes included in the license file have been assigned the correct values. The license file is installed in the following location:

/cm/local/apps/cmd/etc/cluster.pem

and the associated private key file is in:

/cm/local/apps/cmd/etc/cluster.key

4.1.1 Displaying License Attributes Within Bright View

If using Bright View\(^1\), then to verify that the attributes of the license have been assigned the correct values, the license details can be displayed by selecting the Cluster resource, then in the Partition base window that opens up, selecting the License info menu option (figure 4.1):

![Figure 4.1: License Information](image)

\(^1\) Bright View is typically accessed via a “home” URL in the form of https://<head node address>:8081/bright-view/
4.1.2 Displaying License Attributes Within cmsh

Alternatively the licenseinfo command within the main mode of cmsh may be used:

Example

[root@bright91 ~]# cmsh
[bright91]# main licenseinfo
License Information

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Licensee</td>
<td>/C=US/ST=California/L=San Jose/O=Bright Computing/OU=Temporary Licensing/CN=003040</td>
</tr>
<tr>
<td>Serial number</td>
<td>1210883</td>
</tr>
<tr>
<td>Start time</td>
<td>Mon Oct 16 01:00:00 2017</td>
</tr>
<tr>
<td>End time</td>
<td>Fri Dec 31 23:59:00 2038</td>
</tr>
<tr>
<td>Version</td>
<td>7.0 and above</td>
</tr>
<tr>
<td>Edition</td>
<td>Advanced</td>
</tr>
<tr>
<td>Type</td>
<td>Commercial</td>
</tr>
<tr>
<td>Pre-paid nodes</td>
<td>100</td>
</tr>
<tr>
<td>Pay-per-use nodes</td>
<td>Yes</td>
</tr>
<tr>
<td>Data Science nodes</td>
<td>80</td>
</tr>
<tr>
<td>Node count</td>
<td>3</td>
</tr>
<tr>
<td>Allow edge sites</td>
<td>Yes</td>
</tr>
<tr>
<td>Accelerator node count</td>
<td>0</td>
</tr>
<tr>
<td>MAC address / Cloud ID</td>
<td>FA:16:3E:DE:A4:35</td>
</tr>
</tbody>
</table>

The license shown in the preceding example allows 100 nodes to be used. In addition, an unlimited number of pay-per-use nodes can be used when cloudbursting. Pay-per-use nodes are covered in section 8.5 of the Cloudbursting Manual.

The license is tied to a specific MAC address, so it cannot simply be used elsewhere. For convenience, the Node Count field in the output of licenseinfo shows the current number of nodes used.

4.2 Verifying A License—The verify-license Utility

4.2.1 The verify-license Utility Can Be Used When licenseinfo Cannot Be Used

Unlike the licenseinfo command in cmsh (section 4.1), the verify-license utility can check licenses even if the cluster management daemon is not running.

When an invalid license is used, the cluster management daemon cannot start. The license problem is logged in the cluster management daemon log file:

Example

[root@bright91 ~]# service cmd start
Waiting for CMDaemon to start...
CMDaemon failed to start please see log file.
[root@bright91 ~]# tail -1 /var/log/cmdaemon
Dec 30 15:57:02 bright91 CMDaemon: Fatal: License has expired

but further information cannot be obtained using Bright View or cmsh, because these clients themselves obtain their information from the cluster management daemon.

In such a case, the verify-license utility allows the troubleshooting of license issues.

4.2.2 Using The verify-license Utility To Troubleshoot License Issues

There are four ways in which the verify-license utility can be used:
1. Using **verify-license with no options**: simply displays a usage text:

**Example**

```
[root@bright91 ~]# verify-license
Usage: verify-license <path to certificate> <path to keyfile> <verify|info|monthsleft[=12]>
   verify-license <verify|info|monthsleft[=12]> (uses /cm/local/apps/cmd/etc/cluster.pem,key)
```

2. Using **verify-license with the info option**: prints license details:

**Example**

```
[root@bright91 ~]# verify-license info
============ Certificate Information ================
Version: 7.0 and above
Edition: Advanced
Common name: bright91
Organization: Bright Computing
Organizational unit: Development
Locality: San Jose
State: California
Country: US
Serial: 1121317
Starting date: 16 Oct 2017
Expiration date: 31 Dec 2038
MAC address: FA:16:3E:91:14:86
Pre-paid nodes: 100
Pay-per-use Nodes: Yes
Data Science Nodes: 80
Max Openstack Nodes: 70
Accounting & reporting: Yes
Allow edge sites: Yes
License type: Commercial
=====================================================
```

3. Using **verify-license with the verify option**: checks the validity of the license:

- If the license is valid, then no output is produced and the utility exits with exit-code 0.
- If the license is invalid, then output is produced indicating what is wrong. Messages such as these are then displayed:

  - If the license is old:

    **Example**

    ```
    [root@bright91 ~]# verify-license verify
    License has expired
    License verification failed.
    ```

  - If the certificate is not from Bright Computing:

    **Example**

    ```
    [root@bright91 ~]# verify-license verify
    Invalid license: This certificate was not signed by Bright Computing
    License verification failed.
    ```
4.3 Requesting And Installing A License Using A Product Key

4. Using **verify-license with the monthsleft[=value] option:**

- If a number value is set for monthsleft, then
  - if the license is due to expire in more than that number of months, then the verify-license command returns nothing.
  - if the license is due to expire in less than that number of months, then the verify-license command returns the date of expiry.

- If a number value is not set for monthsleft, then the value is set to 12 by default. In other words, the default value means that if the license is due to expire in less than 12 months, then the date of expiry of the license is displayed.

**Example**

[root@bright91 etc]# date
Wed Sep 19 14:55:16 CET 2018
[root@bright91 etc]# verify-license monthsleft
Bright Cluster Manager License expiration date: 31 Dec 2018
[root@bright91 etc]# verify-license monthsleft=3
[root@bright91 etc]# verify-license monthsleft=4
Bright Cluster Manager License expiration date: 31 Dec 2018

4.2.3 Using The **versioninfo** Command To Verify The Cluster Manager Version

The license version should not be confused with the cluster version. The license version is a license format version that rarely changes between cluster manager version releases. Thus a cluster can have a license with version 7.0, which was the license format introduced during Bright Cluster Manager 7.0, and have a cluster manager version 8.1.

The version of a cluster can be viewed with using the versioninfo command, which can be run from the main mode of cmsh as follows:

**Example**

root@bright91 ~]# cmsh
[bright91]% main
[bright91->main]% versioninfo
Version Information

<table>
<thead>
<tr>
<th>Cluster Manager</th>
<th>9.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMDaemon</td>
<td>2.1</td>
</tr>
<tr>
<td>CMDaemon Build Index</td>
<td>147959</td>
</tr>
<tr>
<td>CMDaemon Build Hash</td>
<td>1f03bf9a1e</td>
</tr>
<tr>
<td>Database Version</td>
<td>36241</td>
</tr>
</tbody>
</table>

In the preceding example, the version of the cluster manager is 9.1.

4.3 Requesting And Installing A License Using A Product Key

The license file is introduced at the start of this chapter (Chapter 4). As stated there, most administrators that have installed a new cluster, and who need to install a license on the cluster in order to make their Bright Cluster Manager fully functional, only need to do the following:

- Have their product key at hand

- Run the install-license script on the head node

The details of this section are therefore usually only of interest if a more explicit understanding of the process is required for some reason.
4.3.1 Is A License Needed?—Verifying License Attributes

Before installing a license, the license attributes should be verified (section 4.2) to check if installation is actually needed. If the attributes of the license are correct, the remaining parts of this section (4.3) may safely be skipped. Otherwise the product key (page 56) is used to install a license.

Incorrect license attributes will cause cluster configuration to fail or may lead to a misconfigured cluster. A misconfigured cluster may, for example, not have the ability to handle the full number of nodes. In particular, the license date should be checked to make sure that the license has not expired. If the license is invalid, and it should be valid according to the administrator, then the Bright Computing reseller that provided the software should be contacted with details to correct matters.

If Bright Cluster Manager is already running with a regular license, and if the license is due to expire, then reminders are sent to the administrator e-mail address (page 63 of the Administrator Manual).

4.3.2 The Product Key

A product key is issued by an account manager for Bright Cluster Manager. The product key allows a license to be obtained to run Bright Cluster Manager.

An account manager is the person at Bright Computing who checks that the product key user has the right entitlements to use the key before it is issued. The customer is informed who the account manager is when Bright Cluster Manager is purchased. Purchasing and licensing period queries are normally dealt with by the account manager, while other technical queries that cannot be answered by existing documentation can be dealt with by Bright Cluster Manager technical support (section 17.2 of the Administrator Manual).

The following product key types are possible:

- **Evaluation product key**: An evaluation license is a temporary license that can be installed via an evaluation product key. The evaluation product key is valid for a maximum of 3 months from a specified date, unless the account manager approves a further extension.

  If a cluster has Bright Cluster Manager installed on it, then a temporary license to run the cluster can be installed with an evaluation product key. Such a key allows the cluster to run with defined attributes, such as a certain number of nodes and features enabled, depending on what was agreed upon with the account manager. The temporary license is valid until the product key expires, unless the account manager has approved further extension of the product key, and the license has been re-installed.

  DVD downloads of Bright Cluster Manager from the Bright Computing website come with a built-in license that overrides any product key attributes. The license is valid for a maximum of 3 months from the download date. An evaluation product key allows the user to download such a DVD, and the built-in license then allows 2-node clusters to be tried out. Such a cluster can comprise 1 head node and 1 compute node, or comprise 2 head nodes.

- **Subscription product key**: A subscription license is a license can be installed with a subscription product key. The subscription product key has some attributes that decide the subscription length and other settings for the license. At the time of writing (September 2017), the subscription duration is a maximum of 5 years from a specified date.

  If a cluster has Bright Cluster Manager installed on it, then a subscription license to run the cluster can be installed with a subscription product key. Such a key allows the cluster to run with defined attributes, such as a certain number of nodes and features enabled, depending on what was agreed upon with the account manager. The subscription license is valid until the subscription product key expires.

- **Hardware lifetime product key**: This is a legacy product key that is supported for the hardware lifetime. It is no longer issued.
The product key looks like: the following pattern of digits:

000354-515786-112224-207441-186713

If the product key has been used on the cluster already: then it can be retrieved from the CSR file (page 58) with the command:

```
cm-get-product-key
```

The product key allows: the administrator:

- to obtain and activate a license, which allows the cluster to function
- to register the key using the Bright Computing customer portal (section 4.3.9) account.

The following terminology is used: when talking about product keys, locking, licenses, installation, and registration:

- activating a license: A product key is obtained from any Bright Cluster Manager (re)seller. It is used to obtain and activate a license file. Activation means that Bright Computing records that the product key has been used to obtain a license file. The license obtained by product key activation permits the cluster to work with particular settings. For example, the subscription period, and the number of nodes. The subscription start and end date cannot be altered for the license file associated with the key, so an administrator normally activates the license file as soon as possible after the starting date in order to not waste the subscription period.

- locking a product key: The administrator is normally allowed to use a product key to activate a license only once. This is because a product key is locked on activation of the license. A locked state means that product key cannot activate a new license—it is “used up”.

  An activated license only works on the hardware that the product key was used with. This could obviously be a problem if the administrator wants to move Bright Cluster Manager to new hardware. In such a case, the product key must be unlocked. Unlocking is possible for a subscription license via the customer portal (section 4.3.9). Unlocking an evaluation license, or a hardware lifetime license, is possible by sending a request to the account manager at Bright Computing to unlock the product key. Once the product key is unlocked, then it can be used once again to activate a new license.

- license installation: License installation occurs on the cluster after the license is activated and issued. The installation is done automatically if possible. Sometimes installation needs to be done manually, as explained in the section on the request-license script (page 57). The license can only work on the hardware it was specified for. After installation is complete, the cluster runs with the activated license.

- product key registration: Product key registration occurs on the customer portal (section 4.3.9) account when the product key is associated with the account.

### 4.3.3 Requesting A License With The request-license Script

If the license has expired, or if the license attributes are otherwise not correct, a new license file must be requested.

The request for a new license file is made using a product key (page 56) with the request-license command.

The request-license command is used to request and activate a license, and works most conveniently with a cluster that is able to access the internet. The request can also be made regardless of cluster connectivity to outside networks.
There are three options to use the product key to get the license:

1. **Direct WWW access:** If the cluster has access to the WWW port, then a successful completion of the `request-license` command obtains and activates the license. It also locks the product key.

   - **Proxy WWW access:** If the cluster uses a web-proxy, then the environment variable `http_proxy` must be set before the `request-license` command is run. From a bash prompt this is set with:
     
     ```sh
     export http_proxy=<proxy>
     ```
     
     where `<proxy>` is the hostname or IP address of the proxy. An equivalent alternative is that the `ScriptEnvironment` directive (page 793 of the Administrator Manual), which is a CMDae-mon directive, can be set and activated (page 775 of the Administrator Manual).

2. **Off-cluster WWW access:** If the cluster does not have access to the WWW port, but the administrator does have off-cluster web-browser access, then the point at which the `request-license` command prompts “Submit certificate request to http://licensing.brightcomputing.com/licensing/index.cgi ?” should be answered negatively. CSR (Certificate Sign Request) data generated is then conveniently displayed on the screen as well as saved in the file `/cm/local/apps/cmd/etc/cluster.csr.new`. The `cluster.csr.new` file may be taken off-cluster and processed with an off-cluster browser.

   The CSR file should not be confused with the private key file, `cluster.key.new`, created shortly beforehand by the `request-license` command. In order to maintain cluster security, the private key file must, in principle, never leave the cluster.

   At the off-cluster web-browser, the administrator may enter the `cluster.csr.new` content in a web form at:

   http://licensing.brightcomputing.com/licensing

   A signed license text is returned. At Bright Computing the license is noted as having been activated, and the product key is locked.

   The signed license text received by the administrator is in the form of a plain text certificate. As the web form response explains, it can be saved directly from most browsers. Cutting and pasting the text into an editor and then saving it is possible too, since the response is plain text. The saved signed license file, `<signedlicense>`, should then be put on the head node. If there is a copy of the file on the off-cluster machine, the administrator should consider wiping that copy in order to reduce information leakage.

   The command:

   ```sh
   install-license <signedlicense>
   ```

   installs the signed license on the head node, and is described further on page 59. Installation means the cluster now runs with the activated certificate.

3. **Fax or physical delivery:** If no internet access is available at all to the administrator, the CSR data may be faxed or sent as a physical delivery (postal mail print out, USB flash drive/floppy disk) to any Bright Cluster Manager reseller. A certificate will be faxed or sent back in response, the license will be noted by Bright Computing as having been activated, and the associated product key will be noted as being locked. The certificate can then be handled further as described in option 2.

**Example**

```
[root@bright91 ~]# request-license
Product Key (XXXXX-XXXXX-XXXXX-XXXXX-XXXXX): 000354-515786-112224-207440-186713
```
Country Name (2 letter code): US
State or Province Name (full name): California
Locality Name (e.g. city): San Jose
Organization Name (e.g. company): Bright Computing, Inc.
Organizational Unit Name (e.g. department): Development
Cluster Name: bright91
Private key data saved to /cm/local/apps/cmd/etc/cluster.key.new
MAC Address of primary head node (bright91) for eth0 [00:0C:29:87:B8:B3]:
Will this cluster use a high-availability setup with 2 head nodes? [y/N] n
Certificate request data saved to /cm/local/apps/cmd/etc/cluster.csr.new
Contacting http://licensing.brightcomputing.com/licensing/...
License granted.
License data was saved to /cm/local/apps/cmd/etc/cluster.pem.new
Install license ? [Y/n] n
Use "install-license /cm/local/apps/cmd/etc/cluster.pem.new" to install the license.

4.3.4 Installing A License With The install-license Script

Referring to the preceding request-license example output:

The administrator is prompted to enter the MAC address for eth0.
After the certificate request is sent to Bright Computing and approved, the license is granted.
If the prompt “Install license?” is answered with a “Y” (the default), the install-license script is run automatically by the request-license script.
If the prompt is answered with an “n” then the install-license script must be run explicitly later on by the administrator in order to complete installation of the license. This is typically needed for clusters that have no direct or proxy web access (page 58).
The decision that is made at the prompt also has consequences for the reboot requirements of nodes. These consequences are explained in detail in section 4.3.8.
The install-license script takes the temporary location of the new license file generated by request-license as its argument, and installs related files on the head node. Running it completes the license installation on the head node.

Example

Assuming the new certificate is saved as cluster.pem.new:

[root@bright91 ~]# install-license /cm/local/apps/cmd/etc/cluster.pem.new
======== Certificate Information ========
Version: 9.1
Edition: Advanced
Common name: bright91
Organization: Bright Computing, Inc.
Organizational unit: Development
Locality: San Jose
State: California
Country: US
Serial: 9463
Starting date: 23 Dec 2012
Expiration date: 31 Dec 2013
MAC address: 08:0A:27:BA:B9:43
Pre-paid nodes: 10
Max Pay-per-use Nodes: 1000

© Bright Computing, Inc.
4.3.5 Re-Installing A License After Replacing The Hardware

If a new head node is to be run on new hardware then:

- If the old head node is not able to run normally, then the new head node can have the head node data placed on it from the old head node data backup.

- If the old head node is still running normally, then the new head node can have data placed on it by a cloning action run from the old head node (section 18.4.8 of the Administrator Manual).

If the head node hardware has changed, then:

- a user with a subscription license can unlock the product key directly via the customer portal (section 4.3.9).

- a user with a hardware license almost always has the license under the condition that it expires when the hardware expires. Therefore, a user with a hardware license who is replacing the hardware is almost always restricted from a license reinstallation. Users without this restriction may request the account manager at Bright Computing to unlock the product key.

Using the product key with the request-license script then allows a new license to be requested, which can then be installed by running the install-license script. The install-license script may not actually be needed, but it does no harm to run it just in case afterwards.

4.3.6 Re-Installing A License After Wiping Or Replacing The Hard Drive

If the head node hardware has otherwise not changed:

- The full drive image can be copied on to a blank drive and the system will work as before.

- Alternatively, if a new installation from scratch is done

  - then after the installation is done, a license can be requested and installed once more using the same product key, using the request-license command. Because the product key is normally locked when the previous license request was done, a request to unlock the product key usually needs to be sent to the account manager at Bright Computing before the license request can be executed.

  - If the administrator wants to avoid using the request-license command and having to type in a product key, then some certificate key pairs must be placed on the new drive from the old drive, in the same locations. The procedure that can be followed is:

    1. in the directory /cm/local/apps/cmd/etc/, the following key pair is copied over:
       
       * cluster.key
       
       * cluster.pem
       
       Copying these across means that request-license does not need to be used.

    2. The admin.{pem|key} key pair files can then be placed in the directory /root/.cm/cmsh/.

      Two options are:

      - the following key pair can be copied over:
        
        * admin.key
4.3 Requesting And Installing A License Using A Product Key

- admin.pem

or

* a fresh admin.{pem|key} key pair can be generated instead via a cmd -b option:

**Example**

```
[root@bright91 ~]# service cmd stop
[root@bright91 ~]# cmd -b
[root@bright91 ~]# [...]
Tue Jan 21 11:47:54 [ CMD ] Info: Created certificate in admin.pem
Tue Jan 21 11:47:54 [ CMD ] Info: Created certificate in admin.key
[root@bright91 ~]# [...]
[root@bright91 ~]# chmod 600 admin.*
[root@bright91 ~]# mv admin.* /root/.cm/cmsh/
[root@bright91 ~]# service cmd start
```

It is recommended for security reasons that the administrator ensure that unnecessary extra certificate key pair copies no longer exist after installation on the new drive.

**4.3.7 Re-Installing A License With An Add-On Attribute**

An add-on to a product key, such as the Data Science add-on, can be enabled on a license that does not originally have that attribute. For the Data Science add-on, the URL `http://licensing.brightcomputing.com/licensing/activate-data-science` provides a wizard to activate the Bright Data Science functionalities. For other add-ons, Bright support can be contacted via `http://support.brightcomputing.com` for further instructions.

**4.3.8 Rebooting Nodes After An Install**

The first time a product key is used: After using a product key with the command `request-license` during a cluster installation, and then running `install-license`, a reboot is required of all nodes in order for them to pick up and install their new certificates (section 5.4.1 of the Administrator Manual). The `install-license` script has at this point already renewed the administrator certificates on the head node that are for use with `cmsh` and Bright View. The parallel execution command `pdsh -g computenode reboot` suggested towards the end of the `install-license` script output is what can be used to reboot all other nodes. Since such a command is best done by an administrator manually, `pdsh -g computenode reboot` is not scripted.

The subsequent times that the same product key is used: If a license has become invalid, a new license may be requested. On running the command `request-license` for the cluster, the administrator is prompted on whether to re-use the existing keys and settings from the existing license:

**Example**

```
[root@bright91 ~]# request-license
Product Key (XXXXXX-XXXXXX-XXXXXX-XXXXXX-XXXXXX): 061792-900914-800220-270420-077926

Existing license was found:
  Country: US
  State: None
  Locality: None
  Organization: None
  Organizational Unit: None
  Cluster Name: bright91

You can choose whether to re-use private key and settings from existing license.
  If you answer NO, existing certificates will be invalidated and nodes will have to be rebooted.
```

© Bright Computing, Inc.
If the existing keys are kept, a `pdsh -g computenode reboot` is not required. This is because these keys are X509v3 certificates issued from the head node. For these:

- Any node certificates (section 5.4.1 of the *Administrator Manual*) that were generated using the old certificate are therefore still valid and so regenerating them for nodes via a reboot is not required, allowing users to continue working uninterrupted. On reboot new node certificates are generated and used if needed.

- User certificates (section 6.4 of the *Administrator Manual*) become invalid during certificate regeneration when CMDaemon restarts itself. It is therefore advised to install a permanent license as soon as possible, or alternatively, to not bother creating user certificates until a permanent license has been set up for the cluster.

If the existing keys are not re-used, then node communication ceases until the nodes are rebooted. If there are jobs running on the Bright Cluster Manager nodes, they cannot then complete.

After the license is installed, verifying the license attribute values is a good idea. This can be done using the `licenseinfo` command in `cmsh`, or by selecting the License info menu option from within the Partition base window in Bright View’s Cluster resource (section 4.1).

**The License Log File**

License installation and changes are logged in

```
/var/spool/cmd/license.log
```

to help debug issues.

**4.3.9 The Customer Portal**

Bright Cluster Manager owners with a subscription license can use the customer portal at `https://customer.brightcomputing.com/Customer-Login` to:

- Register a subscription product key
- Unlock a subscription product key
- Request support, including with a non-activated key
- Opt-in to receive release notes e-mails
- See usage statistics
- See AWS-related prices and account balance
This chapter describes setting up registered access for the Bright Cluster Manager with the Red Hat and SUSE distributions.

The head node and regular node images can be set up with registered access to the enterprise Linux distributions of Red Hat and SUSE so that updates from their repositories can take place on the cluster correctly. This allows the distributions to continue to provide support and security updates. Registered access can also be set up in order to create an up-to-date custom software image (section 12.6 of the Administrator Manual) if using Red Hat or SUSE as the base distribution.

Registered access can be avoided for the head node and regular node images by moving the registration requirement to outside the cluster. This can be done by configuring registration to run from a local mirror to the enterprise Linux distributions. The head node and regular node images are then configured to access the local mirror repository. This configuration has the benefit of reducing the traffic between the local network and the internet. However it should be noted that the traffic from node updates scales according to the number of regular node images, rather than according to the number of nodes in the cluster. In most cases, therefore, the added complication of running a separate repository mirror, is unlikely to be worth implementing.

5.1 Registering A Red Hat Enterprise Linux Based Cluster

To register a Red Hat Enterprise Linux (RHEL) system, Red Hat subscriptions are needed as described at https://www.redhat.com/. Registration with the Red Hat Network is needed to install new RHEL packages or receive RHEL package updates, as well as carry out some other tasks.

5.1.1 Registering A Head Node With RHEL

An RHEL head node can be registered from the command line with the subscription-manager command. This uses the Red Hat subscription service username and password as shown:

[root@bright91 ~]# subscription-manager register --username <username> --password <password> \ 
--auto-attach

The --auto-attach option allows a system to update its subscription automatically, so that the system ends up with a valid subscription state.

If the head node has no direct connection to the internet, then an HTTP proxy can be configured as a command line option. The subscription-manager man pages give details on configuring the proxy from the command line.

A valid subscription means that, if all is well, then the RHEL server RPMs repository (rhel-6-server-rpms or rhel-7-server-rpms) is enabled, and means that RPMs can be picked up from that repository.
The optional RPMs repository (rhel-6-server-optional-rpms, rhel-7-server-optional-rpms) must still be enabled using, for example:

```
[root@bright91 ~]# subscription-manager repos --enable rhel-7-server-optional-rpms
Repository 'rhel-7-server-optional-rpms' is enabled for this system.
```

For some RHEL7 packages, the RHEL7 extras repository has to be enabled in a similar manner. The option used is then `--enable rhel-7-server-extras-rpms`.

A list of the available repositories for a subscription can be retrieved using:

```
[root@bright91 ~]# subscription-manager repos --list
```

After registration, the yum subscription-manager plugin is enabled. This means that yum can now be used to install and update from the Red Hat Network repositories.

### 5.1.2 Registering A Software Image With RHEL

The subscription-manager command can be used to register an RHEL software image. If the head node, on which the software image resides, has no direct connection to the internet, then an HTTP proxy can be configured as a command line option. The subscription-manager man pages give details on configuring the proxy from the command line.

The default software image, `default-image`, can be registered by mounting some parts of the filesystem image, and then carrying out the registration within the image by using the Red Hat subscription service username and password. This can be carried out on the head node, with the help of the Bright Cluster Manager cm-chroot-sw-img tool (page 504 of the Administrator Manual) as follows:

```
[root@bright91 ~]# cm-chroot-sw-img /cm/images/default-image subscription-manager register --username <username> --password <password> --auto-attach
```

After the software image is registered, the optional and extras RPMs repository must be enabled using, for RHEL7 systems:

```
[root@bright91 ~]# cm-chroot-sw-img /cm/images/default-image subscription-manager repos --enable rhel-7-server-optional-rpms
[root@bright91 ~]# cm-chroot-sw-img /cm/images/default-image subscription-manager repos --enable rhel-7-server-extras-rpms
```

After registration, the yum subscription-manager plugin is enabled within the software image. This means that yum can now be used to install and update the software image from the Red Hat Network repositories.

### 5.2 Registering A SUSE Linux Enterprise Server Based Cluster

To register a SUSE Linux Enterprise Server system, SUSE Linux Enterprise Server subscriptions are needed as described at http://www.suse.com/. Registration with Novell helps with installing new SLES packages or receiving SLES package updates, as well as to carry out some other tasks.
5.2 Registering A SUSE Linux Enterprise Server Based Cluster

5.2.1 Registering A Head Node With SUSE

The SUSEConnect command can be used to register a SUSE 12 head node. If the head node has no direct connection to the internet, then the HTTP_PROXY and HTTPS_PROXY environment variables can be set to access the internet via a proxy. Running the registration command with the help option, “--help”, provides further information about the command and its options.

The head node can be registered as follows:

```
[root@bright91 ~]# SUSEConnect -e <e-mail address> -r regcode-\n sles=<activation code> -u https://scc.suse.com #for SLES12
```

The e-mail address used is the address that was used to register the subscription with Novell. When logged in on the Novell site, the activation code or registration code can be found at the products overview page after selecting “SUSE Linux Enterprise Server”.

After registering, the SLES and SLE SDK repositories are added to the repository list and enabled.

The defined repositories can be listed with:

```
[root@bright91 ~]# zypper lr
```

and the head node can be updated with:

```
[root@bright91 ~]# zypper refresh
[root@bright91 ~]# zypper update
```

5.2.2 Registering A Software Image With SUSE

The SUSEConnect command can be used to register a SUSE12 software image. If the head node on which the software image resides has no direct connection to the internet, then the HTTP_PROXY and HTTPS_PROXY environment variables can be set to access the internet via a proxy. Running the command with the help option, “--help”, provides further information about the command and its options.

The default software image default-image can be registered by running the following on the head node:

```
[root@bright91 ~]# chroot /cm/images/default-image \n SUSEConnect -e <e-mail address> -r regcode-sles= \n <activation code> -u https://scc.suse.com #for SLES12
```

The e-mail address is the address used to register the subscription with Novell. When logged in on the Novell site, the activation code or registration code can be found at the products overview page after selecting “SUSE Linux Enterprise Server”.

When running the registration command, warnings about the /sys or /proc filesystems can be ignored. The command tries to query hardware information via these filesystems, but these are empty filesystems in a software image, and only fill up on the node itself after the image is provisioned to the node.

Instead of registering the software image, the SLES repositories can be enabled for the default-image software image with:

```
[root@bright91 ~]# cp /etc/zypp/repos.d/* /cm/images/default-image/etc/zypp/repos.d/
[root@bright91 ~]# cp /etc/zypp/credentials.d/* /cm/images/default-image/etc/zypp/\n credentials.d/
[root@bright91 ~]# cp /etc/zypp/service.d/* /cm/images/default-image/etc/zypp/\n service.d/
```

© Bright Computing, Inc.
The copied files should be reviewed. Any unwanted repositories, unwanted service files, and unwanted credential files, must be removed.

The repository list of the default-image software image can be viewed with the chroot option, -R, as follows:

[root@bright91 ~]# zypper -R /cm/images/default-image lr

and the software image can be updated with:

[root@bright91 ~]# export PBL_SKIP_BOOT_TEST=1
[root@bright91 ~]# zypper -R /cm/images/default-image refresh
[root@bright91 ~]# zypper -R /cm/images/default-image update
[root@bright91 ~]# zypper -R /cm/images/default-image clean --all
Changing The Network Parameters Of The Head Node

6.1 Introduction

After a cluster physically arrives at its site, the administrator often has to change the network settings to suit the site. Details on this are given in section 3.2.1 of the Administrator Manual. However, it relies on understanding the material leading up to that section.

This chapter is therefore a quickstart document—conveniently a mere 3 pages—explaining how to change basic IPv4 network settings while assuming no prior knowledge of Bright Cluster Manager and its network configuration interface.

6.2 Method

A cluster consists of a head node, say bright91 and one or more regular nodes. The head node of the cluster is assumed to face the internal network (the network of regular nodes) on one interface, say eth0. The external network leading to the internet is then on another interface, say eth1. This is referred to as a type 1 configuration in this manual (section 3.3.9).

Typically, an administrator gives the head node a static external IP address before actually connecting it up to the external network. This requires logging into the physical head node with the vendor-supplied root password. The original network parameters of the head node can then be viewed and set. For example for eth1:

# cmsh -c "device interfaces bright91; get eth1 dhcp"
yes

Here, yes means the interface accepts DHCP server-supplied values.

Disabling DHCP acceptance allows a static IP address, for example 192.168.1.176, to be set:

# cmsh -c "device interfaces bright91; set eth1 dhcp no"
# cmsh -c "device interfaces bright91; set eth1 ip 192.168.1.176; commit"
# cmsh -c "device interfaces bright91; get eth1 ip"
192.168.1.176

Other external network parameters can be viewed and set in a similar way, as shown in table 6.1. A reboot implements the networking changes.

6.3 Terminology

A reminder about the less well-known terminology in the table:

- netmaskbits is the netmask size, or prefix-length, in bits. In IPv4’s 32-bit addressing, this can be up to 31 bits, so it is a number between 1 and 31. For example: networks with 256 ($2^8$) addresses (i.e.
## Table 6.1: External Network Parameters And How To Change Them On The Head Node

<table>
<thead>
<tr>
<th>Network Parameter</th>
<th>Description</th>
<th>Operation</th>
<th>Command Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP</td>
<td>IP address of head node</td>
<td>set</td>
<td><code>cmsh -c &quot;device interfaces bright91; get eth1 ip&quot;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>cmsh -c &quot;device interfaces bright91; set eth1 ip [address]; commit&quot;</code></td>
</tr>
<tr>
<td>baseaddress</td>
<td>base IP address (network view)</td>
<td>set</td>
<td><code>cmsh -c &quot;network get externalnet baseaddress&quot;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>cmsh -c &quot;network; set externalnet baseaddress [address]; commit&quot;</code></td>
</tr>
<tr>
<td>broadcastaddress</td>
<td>broadcast IP address of network view</td>
<td>set</td>
<td><code>cmsh -c &quot;network get externalnet broadcastaddress&quot;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>cmsh -c &quot;network; set externalnet broadcastaddress [address]; commit&quot;</code></td>
</tr>
<tr>
<td>netmaskbits</td>
<td>netmask in CIDR notation</td>
<td>set</td>
<td><code>cmsh -c &quot;network get externalnet netmaskbits&quot;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>cmsh -c &quot;network; set externalnet netmaskbits [netmask in CIDR notation]; commit&quot;</code></td>
</tr>
<tr>
<td>gateway</td>
<td>gateway (default route)</td>
<td>set</td>
<td><code>cmsh -c &quot;network get externalnet gateway&quot;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>cmsh -c &quot;network; set externalnet gateway [address]; commit&quot;</code></td>
</tr>
<tr>
<td>nameservers</td>
<td>nameserver IP addresses</td>
<td>set</td>
<td><code>cmsh -c &quot;partition get base nameservers&quot;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>cmsh -c &quot;partition; set base nameservers [address]; commit&quot;</code></td>
</tr>
<tr>
<td>searchdomains</td>
<td>name of search domains</td>
<td>set</td>
<td><code>cmsh -c &quot;partition get base searchdomains&quot;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>cmsh -c &quot;partition; set base searchdomains [hostname]; commit&quot;</code></td>
</tr>
<tr>
<td>timeservers</td>
<td>name of timeservers</td>
<td>set</td>
<td><code>cmsh -c &quot;partition get base timeservers&quot;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>cmsh -c &quot;partition; set base timeservers [timeservers]; commit&quot;</code></td>
</tr>
</tbody>
</table>

- Address is set to 0.0.0.0 then the value offered by the DHCP server on the external network is accepted.
- Space-separated multiple values are also accepted for these parameters when setting the value for address or hostname.

© Bright Computing, Inc.
with host addresses specified with the last 8 bits) have a netmask size of 24 bits. They are written in CIDR notation with a trailing “/24”, and are commonly spoken of as “slash 24” networks.

- **baseaddress** is the IP address of the network the head node is on, rather than the IP address of the head node itself. The baseaddress is specified by taking netmaskbits number of bits from the IP address of the head node. Examples:

  - A network with 256 (2^8) host addresses: This implies the first 24 bits of the head node’s IP address are the network address, and the remaining 8 bits are zeroed. This is specified by using “0” as the last value in the dotted-quad notation (i.e. zeroing the last 8 bits). For example: 192.168.3.0
  - A network with 128 (2^7) host addresses: Here netmaskbits is 25 bits in size, and only the last 7 bits are zeroed. In dotted-quad notation this implies “128” as the last quad value (i.e. zeroing the last 7 bits). For example: 192.168.3.128.

When in doubt, or if the preceding terminology is not understood, then the values to use can be calculated using the head node’s sptcalc utility. To use it, the IP address in CIDR format for the head node must be known.

When run using a CIDR address value of 192.168.3.130/25, the output is (some output removed for clarity):

```
# sptcalc 192.168.3.130/25
Host address - 192.168.3.130  
Network address - 192.168.3.128  
Network mask - 255.255.255.128  
Network mask (bits) - 25  
Broadcast address - 192.168.3.255  
Addresses in network - 128  
Network range - 192.168.3.128 - 192.168.3.255
```

Running it with the `-b` (binary) option may aid comprehension:

```
# sptcalc -b 192.168.3.130/25
Host address - 11000000.10101000.00000011.10000010  
Network address - 11000000.10101000.00000011.10000000  
Network mask - 11111111.11111111.11111111.10000000  
Broadcast address - 11000000.10101000.00000011.11111111  
Network range - 11000000.10101000.00000011.10000000 - 11000000.10101000.00000011.11111111
```

© Bright Computing, Inc.
In this chapter, several third party software packages included in the Bright Cluster Manager repository are described briefly. For all packages, references to the complete documentation are provided.

### 7.1 Modules Environment

RHEL and derivatives, and SLES Bright Cluster Manager package name: env-modules

Ubuntu package name: cm-modules

The *modules environment* package is installed by default on the head node. The home page for the software is at [http://modules.sourceforge.net/](http://modules.sourceforge.net/). The software allows a user of a cluster to modify the shell environment for a particular application, or even for a particular version of an application. Typically, a module file defines additions to environment variables such as `PATH`, `LD_LIBRARY_PATH`, and `MANPATH`.

Cluster users use the `module` command to load or remove modules from their environment. The `module(1)` man page has more details about the command, and aspects of the modules environment that are relevant for administrators are discussed in section 2.2 of the *Administrator Manual*. Also discussed there is Lmod, the Lua-based alternative to the Tcl-based traditional modules environment package.

The modules environment from a user’s perspective is covered in section 2.3 of the *User Manual*.

### 7.2 Shorewall

Package name: shorewall

#### 7.2.1 The Shorewall Service Paradigm

Bright Cluster Manager provides the Shoreline Firewall (more commonly known as “Shorewall”) package from the Bright repository. The package provides firewall and gateway functionality on the head node of a cluster.

Shorewall is a flexible and powerful high-level interface for the netfilter packet filtering framework. Netfilter is a standard part of Linux kernels. As its building blocks, Shorewall uses `iptables` and `iptables6` commands to configure netfilter. All aspects of firewall and gateway configuration are handled through the configuration files located in `/etc/shorewall`.

Shorewall IPv4 configuration is managed with the `shorewall` command, while IPv6 configuration is managed via the `shorewall6` command. IPv4 filtering and IPv6 filtering are treated as separate services in Shorewall. For convenience, only IPv4 Shorewall is described from here onward, because IPv6 management is largely similar.

After modifying Shorewall configuration files, Shorewall must be restarted to have the new configuration take effect. From the shell prompt, this can be carried out with:

```
     service shorewall restart
```
In Bright Cluster Manager 9.1, Shorewall is managed by CMDaemon, in order to handle the automation of cloud node access. Restarting Shorewall can thus also be carried out within the services sub-mode (section 3.11 of the Administrator Manual), on the head node. For example a head node bright91 the cmsh session to carry out a restart of shorewall might be:

```
[bright91->device[bright91]->services[shorewall]]% restart
restart Successfully restarted service shorewall on: bright91
```

System administrators who need a deeper understanding of how Shorewall is implemented should be aware that Shorewall does not really run as a daemon process. The command to restart the service therefore does not stop and start a shorewall daemon. Instead it carries out the configuration of netfilter through implementing the iptables configuration settings, and then exits. It exits without leaving a shorewall process up and running, even though service shorewall status shows it is running.

### 7.2.2 Shorewall Zones, Policies, And Rules

In the default setup, Shorewall provides gateway functionality to the internal cluster network on the first network interface (eth0). This network is known as the nat zone to Shorewall. The external network (i.e. the connection to the outside world) is assumed to be on the second network interface (eth1). This network is known as the net zone in Shorewall.

Letting Bright Cluster Manager take care of the network interfaces settings is recommended for all interfaces on the head node (section 3.2 of the Administrator Manual). The file /etc/shorewall/interfaces is generated by the cluster management daemon, and any extra instructions that cannot be added via Bright View or cmsh can be added outside of the file section clearly demarcated as being maintained by CMDaemon.

Shorewall is configured by default (through /etc/shorewall/policy) to deny all incoming traffic from the net zone, except for the traffic that has been explicitly allowed in /etc/shorewall/rules. Providing (a subset of) the outside world with access to a service running on a cluster, can be accomplished by creating appropriate rules in /etc/shorewall/rules. By default, the cluster responds to ICMP ping packets. Also, during cluster installation, the following ports are open by default, but can be set to be blocked by the administrator:

- SSH
- HTTP
- HTTPS
- port 8081, which allows access to the cluster management daemon.

### 7.2.3 Clear And Stop Behavior In service Options, bash Shell Command, And cmsh Shell

To remove all rules, for example for testing purposes, the clear option should be used from the Unix shell. This then allows all network traffic through:

```
shorewall clear
```

Administrators should be aware that in the Linux distributions supported by Bright Cluster Manager, the service shorewall stop command corresponds to the unix shell shorewall stop command, and not to the unix shell shorewall clear command. The stop option for the service and shell blocks network traffic but allows a pre-defined minimal safe set of connections, and is not the same as completely removing Shorewall from consideration. The stop options discussed so far should not be confused with the equivalent stop option in the cmsh shell.

This situation is indicated in the following table:
### Correspondence Of Stop And Clear Options In Shorewall Vs cmsh

<table>
<thead>
<tr>
<th>iptables rules</th>
<th>Service</th>
<th>Unix Shell</th>
<th>cmsh shell</th>
</tr>
</thead>
<tbody>
<tr>
<td>keep a safe set</td>
<td>service shorewall stop</td>
<td>shorewall stop</td>
<td>no equivalent</td>
</tr>
<tr>
<td>clear all rules</td>
<td>no equivalent</td>
<td>shorewall clear</td>
<td>stop shorewall</td>
</tr>
</tbody>
</table>

### 7.2.4 Further Shorewall Quirks

#### Standard Distribution Firewall Should Be Disabled

Administrators should also be aware that RHEL and its derivatives run their own set of high-level iptables setup scripts if the standard distribution firewall is enabled. To avoid conflict, the standard distribution firewall in RHEL7 and its derivatives, Firewalld, must stay disabled, because Bright Cluster Manager requires Shorewall for regular functioning. Shorewall can be configured to set up whatever iptables rules are installed by the standard distribution script instead.

#### Shorewall Stopped Outside Of Bright Cluster Manager Considered Harmful

System administrators wishing to stop Shorewall should note that Bright Cluster Manager by default has the `autostart` setting (section 3.11 of the Administrator Manual) set to on. With such a value, CMDaemon attempts to restart a stopped Shorewall if the service has been stopped from outside of cmsh or Bright View.

Stopping Shorewall outside of cmsh or Bright View is considered harmful, because it can trigger a failover. This is because stopping Shorewall blocks the failover prevention monitoring tests. These tests are the status ping and backup ping (both based on ICMP packet connections), and the CMDaemon status (based on REST calls) (section 18.4.2 of the Administrator Manual). In most cases, with default settings, Shorewall is not restarted in time, even when autostart is on, so that a failover then takes place.

A failover procedure is quite a sensible option when Shorewall is stopped from outside of cmsh or Bright View, because besides the failover monitoring tests failing, other failures also make the head node pretty useless. The blocking of ports means that, amongst others, workload managers and NFS shares are also unable to connect. Ideally, therefore, Shorewall should not be stopped outside cmsh or Bright View in the first place.

Full documentation on the specifics of Shorewall is available at [http://www.shorewall.net](http://www.shorewall.net).

### 7.3 Compilers

Bright Computing provides convenient RPM and .deb packages for several compilers that are popular in the HPC community. All of those may be installed through `yum`, `zypper`, or `apt` (section 12.2 of the Administrator Manual) but (with the exception of GCC) require an installed license file to be used.

#### 7.3.1 GCC

Package name: `gcc-recent` for RHEL and derivatives, and SLES. `cm-gcc` for Ubuntu

The GCC suite that the distribution provides is also present by default.

#### 7.3.2 Intel Compiler Suite

Package names:

| Packages In The Intel Compiler Suite Versions For RHEL And Derivatives, SLES, And Ubuntu |
|---------------------------------------------|-------------------------------------|-------------------------------------|
| 2018                                       | 2019                               | 2020                               |
| intel-cc-2018                              | intel-cc-2019                      | intel-cc-2020                      |

© Bright Computing, Inc.
The Intel compiler packages are provided as part of a suite. So:

- Intel® Parallel Studio XE 2018 provides a 2018 version of the suite
- Intel® Parallel Studio XE 2019 provides a 2019 version of the suite
- Intel® Parallel Studio XE 2020 provides a 2020 version of the suite

Bright Cluster Manager 9.1 provides x86_64 packages for the 2018, 2019, and 2020 versions of the Intel compiler suites. These are for RHEL and derivatives, for SLES, and for Ubuntu, except for the following packages and distributions:

- The 2018 version of the Intel compiler suite is not supported for RHEL8 and derivatives, and is also not supported for Ubuntu 20.04. Therefore, for the 2018 suite, packages for these distributions are not available.

- The 2019 version of the Intel compiler suite is not supported for Ubuntu 20.04. Therefore, for the 2019 suite, a package for this distribution is not available.

Typically the compiler suite includes the Intel Fortran (indicated by `fc`) and Intel C++ compilers (part of the C compiler package, indicated by `cc`). 32-bit compilers are included in the `intelmkl/<year>` and `intel-cc/<year>` packages.

For the other packages, a 32-bit version is sometimes available separately. The 32-bit packages have package names ending in “-32”.

Both the 32-bit and 64-bit versions can be invoked through the same set of commands. The modules environment (section 2.2 of the Administrator Manual) provided when installing the packages can be loaded accordingly, to select one of the two versions. For the C++ and Fortran compilers the 64-bit and 32-bit modules are called as modules beginning with `intelmkl/64` and `intelmkl/32` respectively.

The Intel compiler can be accessed by loading the compiler modules under `intelmkl/64` or `intelmkl/32`. The following commands can be used to run the Intel compilers:

- `icc`: Intel C/C++ compiler

<table>
<thead>
<tr>
<th>Package</th>
<th>Intel-DAAL-2018</th>
<th>Intel-DAAL-2019</th>
<th>Intel-DAAL-2020</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Intel-FC-2018</td>
<td>Intel-FC-2019</td>
<td>Intel-FC-2020</td>
</tr>
<tr>
<td></td>
<td>Intel-GDB-2018</td>
<td>Intel-GDB-2019</td>
<td>Intel-GDB-2020</td>
</tr>
<tr>
<td></td>
<td>Intel-Itac-2018</td>
<td>Intel-Itac-2019</td>
<td>Intel-Itac-2020</td>
</tr>
<tr>
<td></td>
<td>Intel-Mkl-2018</td>
<td>Intel-Mkl-2019</td>
<td>Intel-Mkl-2020</td>
</tr>
<tr>
<td></td>
<td>Intel-Mpi-2018</td>
<td>Intel-Mpi-2019</td>
<td>Intel-Mpi-2020</td>
</tr>
<tr>
<td></td>
<td>Intel-Openmp-2018</td>
<td>Intel-Openmp-2019</td>
<td>Intel-Openmp-2020</td>
</tr>
<tr>
<td></td>
<td>Intel-Tbb-2018</td>
<td>Intel-Tbb-2019</td>
<td>Intel-Tbb-2020</td>
</tr>
</tbody>
</table>
• ifort: Intel Fortran 90/95 compiler

Optional packages are:

• intel-ipp: Integrated Performance Primitives
• intel-daal: Data Analytics Acceleration Library
• intel-gdb: The GNU Project Debugger For Intel Architecture
• intel-icx: C++ Compiler with LLVM Technology
• intel-mpi: Multifabric MPI Library
• intel-mkl: Math Kernel Library
• intel-itac: Trace Analyzer And Collector
• intel-tbb: Threading Building Blocks

A short summary of a package can be shown using, for example: “yum info intel-fc-<year>“.

The compiler packages require a license, obtainable from Intel, and placed in /cm/shared/licenses/intel.


In the following example the license file is copied into the appropriate location, the C/C++ compiler is installed, and a modules environment (section 2.2 of the Administrator Manual) is loaded for use in this session by the root user. Furthermore, the modules environment is added for regular root user use with “module initadd”:

Example

[root@bright91~]# cp license file /cm/shared/licenses/intel/
[root@bright91~]# yum install intel-cc-2019
(installation text output skipped)
[root@bright91~]# module load intel/compiler/64/2019/19.0.2
[root@bright91~]# module initadd intel/compiler/64/2019/19.0.2

How to load modules for use and regular use by non-root users is explained in section 2.2.3 of the Administrator Manual.

7.3.3 PGI High-Performance Compilers

Package name: pgi

The PGI compiler package contains the PGI C++ and Fortran 77/90/95 compilers.

• pgcc: PGI C compiler
• pgc++: PGI C++ compiler
• pgf77: PGI Fortran 77 compiler
• pgf90: PGI Fortran 90 compiler
• pgf95: PGI Fortran 95 compiler
• pgdbg: PGI debugger

© Bright Computing, Inc.
The package can be installed with, for example:

```bash
yum install pgi
```

The license file for PGI can be placed at:

```plaintext
/cm/shared/licenses/pgi/license.dat
```

The PGI module environment can be loaded with:

```bash
module load shared pgi
```

Further documentation for the PGI High-Performance Compilers is available at:

```plaintext
http://www.pgroup.com/resources/docs.htm
```

### 7.3.4 The NVIDIA HPC SDK

The NVIDIA HPC software development kit (https://developer.nvidia.com/hpc-sdk) is a suite of compilers, libraries, and other tools for HPC.

**Features include:**

- The NVIDIA HPC SDK C, C++, and Fortran compilers that support GPU acceleration of HPC modeling and simulation applications with standard C++ and Fortran, OpenACC directives, and CUDA.

- GPU-accelerated math libraries that maximize performance on common HPC algorithms, and optimized communications libraries enable standards-based multi-GPU and scalable systems programming.

- Performance profiling and debugging tools that simplify porting and optimization of HPC applications

- Containerization tools that enable easy deployment on-premises or in the cloud.

- Support for ARM, OpenPOWER, x86-64 CPUs, as well as NVIDIA GPUs, running Linux

### Packages And Versions

The main NVIDIA HPC SDK package is available as a package `cm-nvhpc`. The `cm-nvhpc` package, in turn, depends on a package with the naming format `cm-nvhpc-cuda<number>`. Here, `<number>` is a CUDA version number.

The latest CUDA version number that is compatible with the `cm-nvhpc` package version at the time of release is set as the package dependency. Older CUDA packages are optional but can also be installed.

A command line way to check the release versions and availability is as indicated by the following outputs for a Rocky Linux 8.5 cluster:

**Example**

```bash
[root@bright91 ~]# yum info cm-nvhpc | grep -Version
Version : 21.11

[root@bright91 ~]# yum install cm-nvhpc
... Installing dependencies:  
  cm-nvhpc-cuda11.5  
... [root@bright91 ~]# yum search cm-nvhpc-cuda*
... cm-nvhpc-cuda10.2.x86_64 : NVIDIA HPC SDK  
  cm-nvhpc-cuda11.0.x86_64 : NVIDIA HPC SDK  
  cm-nvhpc-cuda11.5.x86_64 : NVIDIA HPC SDK
```

© Bright Computing, Inc.
The preceding output was what was available at the time of writing (May 2022). The output can be expected to change.

A browser-based way to check the cm-nvhpc versions and CUDA availability situation for Bright versions, distributions and architecture is to use cm-nvhpc as a string in the Bright Computing distributed packages list at https://support.brightcomputing.com/packages-dashboard

**Compiler Modules**

The cm-nvhpc package makes several modules available:

```
[root@bright91 ~]# module avail | grep -o -P 'nvhpc.*?/
```

nvhpc-byo-compiler/
nvhpc-nompi/
nvhpc/

The byo tag is an abbreviation for 'bring-your-own', and means that the general compiler environment for C, C++ and Fortran are not set. The nompi tag implies that paths to the MPI binaries and MPI libraries that come with cm-nvhpc are not set.

**Viewing Installed Available CUDA Versions, And The Running CUDA Version**

The installed available CUDA versions for nvhpc can be viewed with:

**Example**

```
[root@bright91 ~] # module load shared nvhpc
[root@bright91 ~] # basename -a $(ls -1d $NVHPC_ROOT/cuda/[0-9]*)
11.0
11.5
```

The running CUDA version for nvhpc can be viewed with:

**Example**

```
[root@bright91 ~] # nvcc --version
```

nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2021 NVIDIA Corporation
Built on Thu_Nov_18_09:45:30_PST_2021
Cuda compilation tools, release 11.5, V11.5.119
Build cuda_11.5.r11.5/compiler.30672275_0

**Changing The Running CUDA Version**

In the preceding example, CUDA version 11.5 is the default version, while CUDA version 11.0 is also seen to be available.

The CUDA version that is used can be changed on the nodes where nvhpc and the CUDA versions have been installed.

For example, the version can be changed to version 11.0, as follows, for:

- nvhpc cluster-wide:

  ```
  [root bright91] # makelocalrc $(NVHPC_ROOT)/compilers/bin -cuda 11.0 -x
  ```

- nvhpc on a specific head or compute node, as specified by `hostname` -s:

  ```
  [root bright91] # makelocalrc $(NVHPC_ROOT)/compilers/bin -cuda 11.0 -o > \\
  $(NVHPC_ROOT)/compilers/bin/localrc.$(hostname -s)
  ```

- nvhpc for a specific user on a specific head or compute node, as specified by `hostname` -s:

© Bright Computing, Inc.
Third Party Software

```
[root bright91] # make localrc ${NVHPC_ROOT}/compilers/bin -cuda 11.0 -o > 
${HOME}/localrc.$(hostname -s)
```

If the `nvhpc` compiler is run then:

1. the `{NVHPC_ROOT}/compilers/bin/localrc` configuration file is read,
2. followed by a second configuration file, which—if they exist—is either:
   - the `{NVHPC_ROOT}/compilers/bin/localrc.$(hostname -s)` configuration file
   - or the `{HOME}/localrc.$(hostname -s)` configuration file

The second configuration file overwrites any settings set with `{NVHPC_ROOT}/compilers/bin/localrc`.

If the `{NVHPC_ROOT}/compilers/bin/localrc.$(hostname -s)` configuration file exists, then a `{HOME}/localrc.$(hostname -s)` is ignored.

### 7.4 CUDA For GPUs

The optional CUDA packages should be deployed in order to take advantage of the computational capabilities of NVIDIA GPUs. The packages may already be in place, and ready for deployment on the cluster, depending on the particular Bright Cluster Manager software that was obtained. If the CUDA packages are not in place, then they can be picked up from the Bright Computing repositories, or a local mirror.

#### 7.4.1 Installing CUDA

**CUDA Packages Available**

At the time of writing of this section (May 2022) CUDA 10.1, 10.2, 11.0, 11.1, 11.2, 11.3, 11.4, 11.5 and 11.6 packages exist in the YUM, zypper, and APT repositories of Bright Computing. The available versions are updated typically in the next subversion release of Bright Cluster Manager after the upstream changes are made available. The latest packages available can be viewed at [https://support.brightcomputing.com/packages-dashboard/](https://support.brightcomputing.com/packages-dashboard/).

**CUDA packages that the cluster administrator manages:** At the time of writing, the packages that the cluster administrator can install or remove for Bright Cluster Manager are:
<table>
<thead>
<tr>
<th>Package</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cuda10.1-toolkit*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cuda10.2-toolkit*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cuda11.0-toolkit†</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cuda11.1-toolkit†</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cuda11.2-toolkit†</td>
<td>shared</td>
<td>CUDA math libraries and utilities</td>
</tr>
<tr>
<td>cuda11.3-toolkit†</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cuda11.4-toolkit†</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cuda11.5-toolkit†</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cuda11.6-toolkit†</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cuda10.2-visual-tools*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cuda11.0-visual-tools†</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cuda11.1-visual-tools†</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cuda11.2-visual-tools†</td>
<td>shared</td>
<td>CUDA visual toolkit</td>
</tr>
<tr>
<td>cuda11.3-visual-tools†</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cuda11.4-visual-tools†</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cuda11.5-visual-tools†</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cuda11.6-visual-tools†</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cuda10.1-sdk*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cuda10.2-sdk*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cuda11.0-sdk†</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cuda11.1-sdk†</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cuda11.2-sdk†</td>
<td>shared</td>
<td>CUDA software development kit</td>
</tr>
<tr>
<td>cuda11.3-sdk†</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cuda11.4-sdk†</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cuda11.5-sdk†</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cuda11.6-sdk†</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cuda-driver</td>
<td>local</td>
<td>CUDA Tesla GPU driver and libraries.</td>
</tr>
<tr>
<td>cuda-dcgm</td>
<td>local</td>
<td>CUDA Data Center GPU Manager (DCGM). This includes the dcgm1 CLI tool.</td>
</tr>
<tr>
<td>cuda-fabric-manager‡</td>
<td>local</td>
<td>Fabric Manager tools</td>
</tr>
<tr>
<td>cuda-fabric-manager-development‡</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cuda-xorg**</td>
<td>local</td>
<td>CUDA X.org driver and libraries</td>
</tr>
</tbody>
</table>

* not available in Ubuntu2004
† not available in SLES12
‡ optional, intended only for hardware containing an NvSwitch, such as a DGX system
** optional, not used in CentOS8, RHEL8, Ubuntu
The packages of type shared in the preceding table should be installed on the head nodes of a cluster using CUDA-compatible GPUs. The packages of type local should be installed to all nodes that access the GPUs. In most cases this means that the cuda-driver and cuda-dcgm packages should be installed in a software image (section 2.1.2 of the Administrator Manual).

If a head node also accesses GPUs, then the cuda-driver and cuda-dcgm packages should be installed on it, too.

For packages of type shared, the particular CUDA version that is run on the node can be selected via a modules environment command:

Example

```
module add shared cuda11.1/toolkit
```

CUDA packages that the cluster administrator normally does not manage: As an aside, there are also the CUDA DCGM packages:

<table>
<thead>
<tr>
<th>Package</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cuda-dcgm-libs</td>
<td>local</td>
<td>NVIDIA DCGM libraries, installed by default</td>
</tr>
<tr>
<td>cuda-dcgm-devel</td>
<td>local</td>
<td>NVIDIA DCGM development files, not installed by default</td>
</tr>
<tr>
<td>cuda-dcgm-nvvs</td>
<td>local</td>
<td>NVIDIA DCGM validation suite, not installed by default</td>
</tr>
</tbody>
</table>

The preceding DCGM packages are installed in Bright Cluster Manager, because CMDaemon uses them to manage NVidia Tesla GPUs. Tesla drivers normally work for the latest CUDA version, and may not therefore not (yet) support the latest GeForce GPUs.

CUDA package that the cluster administrator may wish to install for CUDA programming: CUB is a CUDA programming library that developers may wish to access. It is provided by the package cm-cub-cuda, from the Machine Learning (cm-ml) repository.

CUDA package installation basic sanity check:

- The NVIDIA GPU hardware should be detectable by the kernel, otherwise the GPUs cannot be used by the drivers. Running the `lspci` command on the device with the GPU before the CUDA package driver installation is a quick check that should make it clear if the NVIDIA hardware is detected in the first place:

  Example

  
  ```
  running `lspci` on node001 which is where the GPU should be:
  ```

  ```
  [root@node001]# lspci | grep NVIDIA
  00:08.0 3D controller: NVIDIA Corporation GV100GL [Tesla V100 PCIe 32GB] (rev a1)
  ```

  If the hardware is not detected by the kernel already, then the administrator should reassess the situation.

- Only after CUDA package installation has taken place, and after rebooting the node with the GPU, are GPU details visible using the `sysinfo` command:

  Example

  
  ```
  running `sysinfo` on node001, which is where the GPU is, via `cmsh` on the head node, while cuda-dcgm is not yet ready:
  ```
Example

running `sysinfo` on node001, which is where the GPU is, via cmsh on the head node, after cuda-dcgm is ready:

```
[bright91->device[node001]]% sysinfo | grep GPU
Number of GPUs 1
GPU Driver Version 460.32.03
GPU0 Name NVIDIA Tesla V100-PCIE-32GB
GPU0 Power Limit 250 W
GPU0 Serial 0420419010206
GPU0 BIOS 88.00.48.00.02
```

CUDA package installation guidelines for compilation with login nodes:

- CUDA compilation should take place on a node that uses NVidia GPUs during compilation.
  - Using a workload manager to allocate this task to GPU nodes is recommended.
- Cross compilation of CUDA software is generally not a best practice due to resource consumption, which can even lead to crashes.
  - If, despite this, cross compilation with a CPU is done, then the cuda-driver package should be installed on the node on which the compilation is done, and the GPU-related services on the node, such as:
    - `cuda-driver.service`
    - `nvidia-persistenced.service`
    - `cuda-dcgm.service`
    should be disabled.

CUDA Package Dependencies

The CUDA packages require access to the Bright Cluster Manager repositories and to the distribution repositories.

In particular, the `freeglut`, `freeglut-devel`, and `xorg-x11-util-macros` packages are required. The installation ISO/DVD that Red Hat provides contains packages from the main repository, and does not contain these packages. For convenience, these packages are provided with a Bright Cluster Manager installation ISO/DVD for Red Hat. Updates must however come from a subscription to the Red Hat supplementary/optinal channels. Packages that are needed for a working Bright cluster, and which are provided by the Bright Cluster Manager installation ISO/DVD, but which are not provided in the Red Hat installation DVD, are discussed in general in section 12.6.2 of the *Administrator Manual*, where the problems that such package dependencies can cause when creating a software image for a cluster with cm-create-image are discussed.

As a separate issue:
yum -q deplist cuda-driver | grep freeglut-devel

dependency: freeglut-devel

provider: freeglut-devel-3.0.0-8.el8.i686

provider: freeglut-devel-3.0.0-8.el8.x86_64

shows that one of the dependencies of the cuda-driver package is the freeglut-devel package, so it should be installed on a node that accesses a GPU. If the CUDA SDK source is to be compiled on the head node (with the head node not accessing a GPU, and with the cuda-driver package not installed) then the freeglut, freeglut-devel, and libXi-devel packages should be installed on the head node.

The cuda-driver package is used to compile the kernel drivers which manage the GPU. Therefore, when installing cuda-driver with yum, several other X11-related packages are installed too, due to package dependencies.

The cuda*-sdk packages can be used to compile libraries and tools that are not part of the CUDA toolkit, but used by CUDA software developers, such as the deviceQuery binary (section 7.4.3).

The cuda-xorg package is optional, and contains the driver and libraries for an X server

Example

For example, on a cluster where (some of) the nodes access GPUs, but the head node does not access a GPU, the following commands can be issued on the head node to install the CUDA 11.2 packages using YUM:

[root@mycluster ~]# yum install cuda11.2-toolkit cuda11.1-sdk
[root@mycluster ~]# yum --installroot=/cm/images/default-image install cuda-driver cuda-dcgm

The --installroot command installs to the image used by the nodes. Here the image used by the nodes is assumed to be default-image. To ensure the software is installed from the image to the nodes, the imageupdate command can be run from within cmsh for the appropriate nodes.

[root@mycluster ~]# cmsh
[mycluster]# device
[mycluster->device]# imageupdate -n node0[01-10] -w

Compiling And Loading CUDA Drivers On The Fly

The cuda-driver package provides an init script which is executed at boot-time to load the CUDA driver. Because the CUDA driver depends on the running kernel, the script compiles the CUDA driver on the fly, and subsequently loads the module into the running kernel, and creates the nvidia* devices (/dev/nvidia0, /dev/nvidia1...).

The actions of compiling the cuda-driver, loading the module, and creating the nvidia* devices, can also be carried out during normal running by manually running the init script.

Loading the CUDA driver causes a number of diagnostic kernel messages to be logged:

Example

[root@mycluster ~]# /etc/init.d/cuda-driver start
Compiling nvidia driver... loading... create device(s)... [ OK ]
[root@mycluster ~]# dmesg
...nvidia-nvlink: Nvlink Core is being initialized, major device number 241
[drm] Initialized nvidia-drm 0.0.0 20150116 for 0000:82:00.0 on minor 1
NVRM: loading NVIDIA UNIX x86_64 Kernel Module 361.93.03 Tue Sep 27 22:40:25 PDT 2016
nvidia-uvm: Loaded the UVM driver in 8 mode, major device number 240
nvidia 0000:82:00.0: irq 200 for MSI/MSI-X

Versions of Red Hat 7 and beyond, and derived versions, as well as versions of SLES version 12 and beyond, use systemd instead of an init-based system. For these the equivalent starting command command is:
Example

[root@mycluster ~]# systemctl start cuda-driver

If there is a failure in compiling the CUDA module, it is usually indicated by a message saying “Could not make module”, “NVRM: API mismatch:”, or “Cannot determine kernel version”. Such a failure typically occurs because compilation is not possible due to missing the correct kernel development package from the distribution. Section 7.4.2 explains how to check for, and install, the appropriate missing package.

7.4.2 Installing Kernel Development Packages

This section can be skipped if there is no CUDA compilation problem.

Typically, a CUDA compilation problem (section 7.4.1) is due to a missing or mismatched kernel package and kernel-devel package.

To check the head node and software images for the installation status of the kernel-devel package, the Bright Cluster Manager utility kernel-devel-check is used (section 12.3.5 of the Administrator Manual).

Alternatively, if a standard kernel is in use by the image, then simply upgrading CUDA, the standard kernel, and kernel-devel, to their latest versions may be a good tactic to fix a CUDA compilation problem, because the kernel and kernel-devel package versions become synchronized during such an upgrade.

7.4.3 Verifying CUDA

An extensive method to verify that CUDA is working is to run the verify_cudaX.sh script, located in the CUDA SDK directory.

This script first copies the CUDA SDK source to a local directory under /tmp or /local. It then builds CUDA test binaries and runs them. It is possible to select which of the CUDA test binaries are run. These binaries clutter up the disk and are not intended for use as regular tools, so the administrator is urged to remove them after they are built and run.

A help text showing available script options is displayed when “verify_cudaX.sh -h” is run.

The script can be run as follows on the head or regular node (some output elided):

Example

[root@node001 ~]# module load shared cuda11.2/toolkit
[root@node001 ~]# cd $CUDA_SDK
[root@node001 11.2.0]# ./verify_cuda11.2.sh
Copy cuda11.2 sdk files to "/tmp/cuda11.2" directory.
make clean
make (may take a while)

Run all tests? (y/N)? y

Executing: /tmp/cuda11.2/bin/x86_64/linux/release/alignedTypes

[/tmp/cuda10.2/bin/x86_64/linux/release/alignedTypes] - Starting...
GPU Device 0: "Volta" with compute capability 7.0

[Tesla V100-PCIe-32GB] has 80 MP(s) x 64 (Cores/MP) = 5120 (Cores)
> Compute scaling value = 1.00
> Memory Size = 49999872
Allocating memory...
Generating host input data array...
Uploading input data to GPU memory...
Testing misaligned types...
uint8...
Avg. time: 1.234375 ms / Copy throughput: 37.724362 GB/s.
TEST OK
uint16...
...
...

All cuda11.2 just compiled test programs can be found in the 
"/tmp/cuda11.2/bin/x86_64/linux/release/" directory
They can be executed from the "/tmp/cuda11.2" directory.

The "/tmp/cuda11.2" directory may take up a lot of disk space.
Use "rm -rf /tmp/cuda11.2" to remove the data.

Another method to verify that CUDA is working, is to build and use the deviceQuery command on
a node accessing one or more GPUs. The deviceQuery command lists all CUDA-capable GPUs that a
device can access, along with several of their properties (some output elided):

Example

[root@node001 ~]# module load shared cuda11.2/toolkit
[root@node001 ~]# cd $CUDA_SDK
[root@node001 11.2.0]# cd 1Utilities/deviceQuery
[root@node001 deviceQuery ]# make
...
mkdir -p ../../bin/x86_64/linux/release
cp deviceQuery ../../bin/x86_64/linux/release
[root@node001 deviceQuery ]# ./deviceQuery
./deviceQuery Starting...

CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "Tesla V100-PCIE-32GB"
CUDA Driver Version / Runtime Version 11.2 / 11.2
CUDA Capability Major/Minor version number: 7.0
Total amount of global memory: 32510 MBytes (34089730048 bytes)
(80) Multiprocessors, ( 64) CUDA Cores/MP: 5120 CUDA Cores
GPU Max Clock rate: 1380 MHz (1.38 GHz)
Memory Clock rate: 877 Mhz
...

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 11.2, CUDA Runtime Version = 11.2, NumDevs
= 1
Result = PASS

The CUDA user manual has further information on how to run compute jobs using CUDA.

Further information on CUDA verification: More on verification can be found in the NVIDIA CUDA
Linux.pdf.
7.4 CUDA For GPUs

7.4.4 Verifying OpenCL
CUDA also contains an OpenCL compatible interface. To verify that OpenCL is working, or failing, the verify_opencl.sh script can be run (some output elided):

**Example**

```
[root@cuda-test ~]# module load shared cuda10.2/toolkit
[root@cuda-test ~]# cd $CUDA_SDK
[root@cuda-test 11.2.0]# ./verify_opencl.sh

Copy opencl files to "/tmp/opencl" directory.
make clean
make (may take a while)
Run all tests? (y/N)? y
Executing: /tmp/opencl/OpenCL/bin/linux/release/oclBandwidthTest

[oclBandwidthTest] starting...
/tmp/opencl/OpenCL/bin/linux/release/oclBandwidthTest Starting...

...

All opencl just compiled test programs can be found in the "/tmp/opencl/OpenCL/bin/linux/release/" directory
They can be executed from the "/tmp/opencl/OpenCL" directory.

The "/tmp/opencl" directory may take up a lot of disk space.
Use "rm -rf /tmp/opencl" to remove the data.
```

7.4.5 Configuring The X Server

The X server can be configured to use a CUDA GPU. To support the X server, X.org, the cuda-driver, and cuda-xorg packages need to be installed.

The default configuration file for X.org is "/etc/X11/xorg.conf". If there is no xorg.conf file, then a template can be generated with the help of Xorg -configure.

The following file pathname lines may need to be added to the Files section of the X configuration file:

```
ModulePath "/usr/lib64/xorg/modules/extensions/nvidia"
ModulePath "/usr/lib64/xorg/modules/extensions"
ModulePath "/usr/lib64/xorg/modules"
```

The following dynamic module loading line may need to be added to the Module section of the X configuration:

```
Load "glx"
```

The following graphics device description lines need to be replaced in the Device section of the X configuration:

```
Driver "nvidia"
```

The BusID line may need to be replaced with the ID shown for the GPU by the lspci command.

**Example**

```
Section "ServerLayout"
  Identifier "Default Layout"
  Screen 0 "Screen0" 0 0
```

© Bright Computing, Inc.
7.4.6 NVIDIA Validation Suite (Package: cuda-dcgm-nvvs)

The NVIDIA Validation Suite (NVVS) runs diagnostics to validate the NVIDIA software components. Running the `nvvs` binary directly is now deprecated. The diagnostics are now run instead as part of the `dcgm` `diag` command.

The package for NVVS is not installed by default. If the package is not installed, then an attempt to run the `dcgm` utility fails:

**Example**

```
[root@bright91 ~] module load cuda-dcgm
[root@bright91 ~]# dcgm diag -r 1
Error: Diagnostic could not be run because the Tesla recommended driver is not being used.
```

The package can be installed with:

**Example**

```
[root@bright91 ~] module load cuda-dcgm
[root@bright91 ~] yum install cuda-dcgm-nvvs
```
After it is installed, the node on which the installation is done must be rebooted. Running the diagnostic after the reboot should display output similar to:

Example

```
[root@bright91 ~] module load cuda-dcgm
[root@bright91 ~]# dcgm-diag -r 3
Successfully ran diagnostic for group.
+---------------------------+------------------------------------------------+
| Diagnostic                | Result |
+===========================+================================================+
|---- Deployment ----------+------------------------------------------------|
| Blacklist                 | Pass   |
| NVML Library              | Pass   |
| CUDA Main Library         | Pass   |
| Permissions and OS Blocks | Pass   |
| Persistence Mode          | Pass   |
| Environment Variables     | Pass   |
| Page Retirement           | Pass   |
| Graphics Processes        | Pass   |
| Inforom                   | Pass   |
|---- Integration ----------+------------------------------------------------|
| PCIe                      | Pass - All |
|---- Hardware -------------+------------------------------------------------|
| GPU Memory                | Pass - All |
| Diagnostic                | Pass - All |
|---- Stress                +------------------------------------------------|
| SM Stress                 | Pass - All |
| Targeted Stress           | Pass - All |
| Targeted Power            | Pass - All |
| Memory Bandwidth          | Pass - All |
+---------------------------+------------------------------------------------+
```

7.4.7 Further NVIDIA Configuration Via The Cluster Manager

After the installation and verification has been carried out for CUDA drivers, further configuration as part of CMDaemon management can be carried out, as explained in section 3.13.2 of the Administrator Manual.

7.5 AMD GPU Driver Installation

AMD GPUs require drivers to be installed in order to work. The DKMS system, which is used to compile kernel modules that are not part of the mainline kernel, is used for creating AMD GPU kernel modules.

7.5.1 AMD GPU Driver Installation For RHEL/CentOS

For RHEL/CentOS 7.8/8.2 the following procedure can be followed:

The `default-image` is first cloned to an image that is to be the AMD GPU image, for example `am`:

Example

```
root@bright91:~# cmsh
[bright91] % softwareimage
[bright91->softwareimage] % clone default-image am
[bright91->softwareimage*am*] % commit
[bright91->softwareimage*am*] % quit
```

© Bright Computing, Inc.
To install the packages, the description in https://rocmdocs.amd.com/en/latest/Installation-Guide/Installation-Guide.html#centos-rhel at the time of writing (January 2021) is followed. The installation must be done in the image, which for a RHEL/CentOS image uses a chroot into the image, and uses a bind mount to have some special filesystem directories (/proc, /sys, and similar) be available during the package installation. This is needed for the DKMS installation.

Bind mounting the filesystems and then chrooting is a little tedious, so the cm-chroot-sw-img utility (page 504 of the Administrator Manual) is used to automate the job.

The following session output illustrates the procedure for CentOS, with much text elided:

```
root@bright91 ~# cm-chroot-sw-img /cm/images/am/
mounted /cm/images/default-image/dev
mounted /cm/images/default-image/dev/pts
mounted /cm/images/default-image/proc
mounted /cm/images/default-image/sys
mounted /cm/images/default-image/run
...
```

The following repositories are enabled for RHEL v7:

```
root@bright91 ~# subscription-manager repos --enable rhel-server-rhscl-7-rpms
root@bright91 ~# subscription-manager repos --enable rhel-7-server-optional-rpms
root@bright91 ~# subscription-manager repos --enable rhel-7-server-extras-rpms
```

For DKMS to work with RHEL/Centos 7, the developer toolset 7 (devtoolset-7) must be installed. For CentOS/RHEL v8 the developer toolset is not needed.

The instructions at https://www.softwarecollections.org/en/scls/rhscl/devtoolset-7/ should be followed to set up the devtoolset-7 environment.

In the image, an rocm.repo configuration file should be created with the following values:

Example

```
[root@am:/]# cat /etc/yum.repos.d/rocm.repo
[ROCM]
name=ROCm
baseurl=https://repo.radeon.com/rocm/centos8/rpm
enabled=1
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
```

The rocm-dkms package can then be installed, the chroot exited, and the bind mounts then automatically unmount:

```
[root@am:/]# yum install rocm-dkms
...
[root@am:/]# exit
```

```
unmounted /cm/images/am/dev/pts
unmounted /cm/images/am/dev
unmounted /cm/images/am/proc
unmounted /cm/images/am/sys
unmounted /cm/images/am/run
```

The nodes that are to use the driver should then be set to use the new image, and should be rebooted:

Example
7.5 AMD GPU Driver Installation

Normal nodes without the AMD GPU also boot up without crashing if they are set to use this image, but will not be able to run OpenCL programs.

7.5.2 AMD GPU Driver Installation For Ubuntu

For Ubuntu 18.04 and later (LTS releases) the following procedure can be followed.

As for the Centos/RHEL case, the procedure begins with cloning the default-image to an image that is to be the AMD GPU image, such as, for example, am.

To install the packages, the description in https://rocmdocs.amd.com/en/latest/Installation_Guide/Installation-Guide.html#ubuntu at the time of writing (January 2021) is followed. The installation must be done in the image, which for an Ubuntu image uses a chroot into the image, and uses a bind mount to have the /proc, /sys, and other special directories be available during the package installation (section 12.4 of the Administrator Manual). The following session output illustrates the procedure, with much text elided, with the help of the cm-chroot-sw-img utility (page 504 of the Administrator Manual):

```
root@bright91:~# cm-chroot-sw-img /cm/images/am/
... 
root@bright91:~/# wget -qO - https://repo.radeon.com/rocm/rocm.gpg.key | apt-key add -
OK 
root@bright91:~/# echo 'deb [arch=amd64] https://repo.radeon.com/rocm/apt/debian/ xenial main' \ 
| sudo tee /etc/apt/sources.list.d/rocm.list 
root@bright91:~/# cat /etc/apt/sources.list.d/rocm.list 
... 
root@bright91:~/# apt install rocm-dkms 
Reading package lists... Done 
Building dependency tree 
Reading state information... Done 
... 
Setting up rock-dkms (1:4.0-23) ... 
Loading new amdgpu-4.0-23 DKMS files... 
Building for 4.18.0-18-generic 
Building for architecture x86_64 
Building initial module for 4.18.0-18-generic 
Done. 
Forcing installation of amdgpu 

amdgpu: 
Running module version sanity check. 
- Original module 
```
- Installation
- Installing to /lib/modules/4.18.0-18-generic/updates/dkms/

**amdttm.ko:**
Running module version sanity check.
- Original module
- Installation
- Installing to /lib/modules/4.18.0-18-generic/updates/dkms/

**amdcl.ko:**
Running module version sanity check.
- Original module
- Installation
- Installing to /lib/modules/4.18.0-18-generic/updates/dkms/

**amd-sched.ko:**
Running module version sanity check.
- Original module
- Installation
- Installing to /lib/modules/4.18.0-18-generic/updates/dkms/

depmod......

**DKMS: install completed.**
update-initramfs: Generating /boot/initrd.img-4.18.0-18-generic

*cp: cannot stat '/etc/iscsi/initiatorname.iscsi': No such file or directory*

```
root@bright91:/# exit
```

The nodes that are to use the driver should then be set to use the new image, and should be rebooted.

```
[bright91->device[node001]]% set softwareimage am
[bright91->device*[node001*]]% commit
[bright91->device[node001]]% reboot node001
```

Normal nodes without the AMD GPU also boot up without crashing if they are set to use this image, but are not able to run OpenCL programs.

### 7.5.3 AMD GPU Driver Installation For SLED/SLES

The driver can be installed in SLES15SP2, in a similar way to the CentOS/RHEL case, as follows:

1. The procedure begins with cloning the default-image to an image that is to be the AMD GPU image, such as, for example, am.

   ```
   root@bright91:~# cmsh
   [bright91] % softwareimage
   [bright91->softwareimage] % clone default-image am
   [bright91->softwareimage*[am*]] % commit
   [bright91->softwareimage[am]] %
   [notice] bright91: Started to copy: /cm/images/default-image -> /cm/images/am (117)
   [bright91->softwareimage[am]] % quit
   
   2. The new software image should then be chrooted into:
   ```
   bright91:~ # chroot /cm/images/am
   bright91 am:/ #
The following commands are then used to add the ROCm repository, and to install the ROCm driver and software:

zypper install dkms
zypper clean -a
zypper addrepo https://repo.radeon.com/rocm/zyp/zypper/ rocm
rpm --import https://repo.radeon.com/rocm/rocm.gpg.key
zypper --gpg-auto-import-keys install rocm-dkms

3. A sample output session is:

Example

am:/ # zypper install dkms
New repository or package signing key received:
Repository: Cluster Manager Machine Learning Trunk - Updates
Key Name: Bright Computing Development Team <dev@brightcomputing.com>
Key Fingerprint: E5363DE9 0EB02EFG CS8195F3 EF692F66 FA4DE6D0
Key Created: Sun 13 Sep 2009 11:19:36 PM CEST
Key Expires: (does not expire)
Subkey: 80526A8C005E7CBC 2009-09-13 [does not expire]
Rpm Name: gpg-pubkey-fa4de6d0-4aad61e8

Do you want to reject the key, trust temporarily, or trust always? [r/t/a/?] (r): a
Building repository 'Cluster Manager Machine Learning Trunk - Updates' cache ............[done]
Building repository 'Cluster Manager Trunk - Updates' cache .............................[done]
Building repository 'bright-dist' cache ....................................................[done]
Building repository 'sle-15-backports' cache ...........................................[done]
Building repository 'sle-15-backports-internal' cache .................................[done]
Loading repository data...
Reading installed packages...
Resolving package dependencies...
The following 6 NEW packages are going to be installed:
dkms kernel-preempt-devel kernel-syms mozilla-nss-tools pesign pesign-obs-integration
The following 6 packages have no support information from their vendor:
dkms kernel-preempt-devel kernel-syms mozilla-nss-tools pesign pesign-obs-integration

6 new packages to install.
Overall download size: 6.4 MiB. Already cached: 0 B. After the operation, additional 6.5 MiB will be used.
Continue? [y/n/v/...?] shows all options] (y): y

... Additional rpm output:
warning: /var/cache/zypp/packages/sle-15-backports/noarch/dkms-2.3-bp150.2.4.noarch.rpm: Header V3 RSA/SHA256 Signature, key ID 65176565: NOKEY Failed to connect to bus: No such file or directory

am:/ # zypper clean -a
All repositories have been cleaned up.
am:/ # zypper addrepo https://repo.radeon.com/rocm/zyp/zypper/ rocm
Adding repository 'rocm' ....................................................................[done]
Repository 'rocm' successfully added
URI : https://repo.radeon.com/rocm/zypper/
Enabled : Yes
GPG Check : Yes
Autorefresh : No
Priority : 99 (default priority)

Repository priorities in effect: (See 'zypper lr -P' for details)
11 (raised priority) : 2 repositories
99 (default priority) : 4 repositories

am:/ # zypper ref
Repository 'Cluster Manager Machine Learning Trunk - Updates' is up to date.
Repository 'Cluster Manager Trunk - Updates' is up to date.
Repository 'bright-dist' is up to date.
Retrieving repository 'rocm' metadata -----------------------------------------------------
Looking for gpg key ID 1A693C5C in cache /var/cache/zypp/pubkeys.
Repository rocm does not define additional 'gpgkey=' URLs.
Warning: File 'repomd.xml' from repository 'rocm' is signed with an unknown key '9386B48A1A693C5C'.

Note: Signing data enables the recipient to verify that no modifications occurred after the data were signed. Accepting data with no, wrong or unknown signature can lead to a corrupted system and in extreme cases even to a system compromise.

Note: File 'repomd.xml' is the repositories master index file. It ensures the integrity of the whole repo.

Warning: We can't verify that no one meddled with this file, so it might not be trustworthy anymore! You should not continue unless you know it's safe.

File 'repomd.xml' from repository 'rocm' is signed with an unknown key '9386B48A1A693C5C'.
Continue? [yes/no] (no): yes
Retrieving repository 'rocm' metadata ...................................................
done
Building repository 'rocm' cache .....................................................
done
Repository 'sle-15-backports' is up to date.
Repository 'sle-15-backports-internal' is up to date.
All repositories have been refreshed.

am:/ # rpm --import https://repo.radeon.com/rocm/rocm.gpg.key
am:/ # zypper --gpg-auto-import-keys install rocm-dkms
Loading repository data...
Reading installed packages...
Resolving package dependencies...

The following 30 NEW packages are going to be installed:
 babeltrace comgr hip-base hip-doc hip-roccl hip-samples hsa-amdgpuвлечен
 hsaamdgpu gboolean openssl-extras perl-File-Which rock-dkms
 rock-dkms-firmware roccl-ocl roccl-cmake roccl-dbapi roccl-dev roccl-device-libs
 roccl-dkms roccl-gdb rocminfo roccl-rocmdev roccl-smi lib64 rocm-utils
 rocprofiler-dev roctracer-dev

The following 30 packages have no support information from their vendor:
 babeltrace comgr hip-base hip-doc hip-roccl hip-samples hsa-amdgpuвлечен
 hsaamdgpu gboolean openssl-extras perl-File-Which rock-dkms
 rock-dkms-firmware roccl-ocl roccl-cmake roccl-dbapi roccl-dev roccl-device-libs
 roccl-dkms roccl-gdb rocminfo roccl-rocmdev roccl-smi lib64 rocm-utils
 rocprofiler-dev roctracer-dev

© Bright Computing, Inc.
30 new packages to install.
Overall download size: 560.6 MiB. Already cached: 0 B. After the operation, additional 2.9 GiB will be used.
Continue? [y/n/v/...? shows all options] (y): y
Retrieving package perl-File-Which-1.22-1.22.noarch 29.2 KiB (29.2 KiB unpacked)

(15/30) Installing: rock-dkms-1:4.0-23.el7.noarch ...................................... [done]
Additional rpm output:
Loading new amdgpu-4.0-23.el7 DKMS files...
Building for 5.3.18-22-default
Building initial module for 5.3.18-22-default
Done.
Forcing installation of amdgpu

amdgpu:
Running module version sanity check.
- Original module
  - Found /lib/modules/5.3.18-22-default/kernel/drivers/gpu/drm/amd/amdgpu/amdgpu.ko
  - Storing in /var/lib/dkms/amdgpu/original_module/5.3.18-22-default/x86_64/
  - Archiving for uninstallation purposes
- Installation
  - Installing to /lib/modules/5.3.18-22-default/updates/

amdttm.ko:
Running module version sanity check.
- Original module
- Installation
  - Installing to /lib/modules/5.3.18-22-default/updates/

amdkcl.ko:
Running module version sanity check.
- Original module
- Installation
  - Installing to /lib/modules/5.3.18-22-default/updates/

amd-sched.ko:
Running module version sanity check.
- Original module
- Installation
  - Installing to /lib/modules/5.3.18-22-default/updates/

depmod....

DKMS: install completed.

(16/30) Installing: hip-doc-4.0.20496.5685.40000-sles151.23.x86_64 .................... [done]
(17/30) Installing: rocm-gdb-10.1_rocm_rel_4.0_23-1.x86_64 ............................ [done]
(18/30) Installing: roctracer-dev-1.0.0-1.x86_64 ........................................ [done]
(19/30) Installing: rocprofiler-dev-1.0.0-1.x86_64 ..................................... [done]
(20/30) Installing: rocinfo-1.40000.0-1.x86_64 .......................................... [done]
(21/30) Installing: rocm-opencl-3.6Beta_17_g875c1f8_rocm_rel_4.0_23-1.x86_64 ........ [done]
(22/30) Installing: openmp-extras-12.10-0.x86_64 ....................................... [done]
(23/30) Installing: hsa-amd-aqlprofile-1.0.0-1.x86_64 .................................. [done]
(24/30) Installing: hip-roccrlr-4.0.20496.5685.40000-sles151.23.x86_64 ............... [done]
(25/30) Installing: rocm-opencl-devel-3.6Beta_17.g875c1f8_rocm_rel_4.0.23-1.x86_64 ...... [done]
(26/30) Installing: hip-samples-4.0.20496.5685.40000-sles151.23.x86_64 ..................... [done]
(27/30) Installing: rocm-clang-ocl-0.5.0.64_rocm_rel_4.0.23_50fb51a-1.x86_64 ............ [done]
(28/30) Installing: rocm-utils-4.0.0.40000-sles151.23.x86_64 ................................ [done]
(29/30) Installing: rocm-dev-4.0.0.40000-sles151.23.x86_64 .................................. [done]
(30/30) Installing: rocm-dkms-4.0.0.40000-sles151.23.x86_64 ................................. [done]

am:/ # exit

umounted /cm/images/am/dev/pts
umounted /cm/images/am/dev
umounted /cm/images/am/proc
umounted /cm/images/am/sys
umounted /cm/images/am/run

4. The nodes that are to use the driver should then be set to use the new image, and should be rebooted:

Example

root@bright91 ~> cmsh
[bright91]% device use node001
[bright91->device[node001]]% set softwareimage am
[bright91->device[node001]]% commit
[bright91->device[node001]]% reboot node001

Normal nodes without the AMD GPU also boot up without crashing if they are set to use this image, but will not be able to run OpenCL programs.

7.6 OFED Software Stack

This section explains how OFED packages are installed so that Bright Cluster Manager can use InfiniBand for data transfer during regular use—that is for computation runs, rather than for booting up or provisioning. The configuration of PXE booting over InfiniBand is described in section 5.1.3 of the Administrator Manual. The configuration of node provisioning over InfiniBand is described in section 5.3.3 of the Administrator Manual.

7.6.1 Choosing A Distribution Version, Or A Vendor Version, Ensuring The Kernel Matches, And Logging The Installation

By default, the Linux distribution OFED packages are matched to the distribution kernel version and installed on the cluster. This is the safest option in most cases, and also allows NFS over RDMA.

Bright Cluster Manager also packages the OFED software developed by Mellanox. These vendor OFED packages can be more recent than the distribution packages, which means that they can provide support for more recent hardware and firmware, as well as more features.

For the vendor OFED packages to work, the OFED firmware as provided by the manufacturer should in general be recent, to ensure software driver compatibility.

The Bright Cluster Manager vendor OFED packages can be selected and installed during the initial cluster installation (figure 3.22), replacing the default distribution OFED stack. The stack can also be installed later on, after the cluster is set up.

If there is no OFED kernel modules package available for the kernel in use, then the Bright Computing OFED install script tries to build the package from the source package provided by the vendor Mellanox. However, very recent kernels may not yet be supported by the source package. If a build fails for such a kernel, then the OFED software stack will fail to install, and nothing is changed on the head
node or the software image. OFED hardware manufacturers resolve build problems with their software shortly after they become aware of them, but in the meantime a supported kernel must be used.

When updating kernels on the head or the regular nodes, the updated Bright Cluster Manager OFED software stack must be reinstalled (if there are packages already available for the kernel) or rebuilt (if there are no such packages provided).

If the Bright Cluster Manager OFED software stack is installed during the cluster installation procedure itself (section 3.3.17), then some basic information is logged to /var/log/cmfirstboot.log, which is the general first boot log.

If the Bright Cluster Manager OFED software stack is not installed during the cluster installation procedure itself, then it can be installed later when the cluster is up and running. A successful installation of the Bright Cluster Manager OFED software stack (section 7.6.2) onto a running cluster includes the running of an installation script after the Bright Cluster Manager OFED package installation. The vendor and version number installed can then be found in /etc/cm-ofed. Further installation details can be found in /var/log/cm-ofed.log.

7.6.2 Mellanox OFED Stack Installation Using The Bright Computing Repository

Package names: mlnx-ofed47, mlnx-ofed49, mlnx-ofed50, mlnx-ofed51

The Mellanox stacks are installed and configured by Bright Cluster Manager in an identical way as far as the administrator is concerned. In this section (section 7.6.2):

<vendor-ofedVersion>

is used to indicate where the administrator must carry out a substitution. For Mellanox, the substitution is one of the following:

• mlnx-ofed47 for the Mellanox version 4.7 stack
• mlnx-ofed49 for the Mellanox version 4.9 stack
• mlnx-ofed50 for the Mellanox version 5.0 stack
• mlnx-ofed51 for the Mellanox version 5.1 stack

These stacks are currently supported by the Bright Cluster Manager 9.1-supported distributions (RHEL and derivatives, and SLES) according to the compatibility matrix at https://www.mellanox.com/support/mlnx-ofed-matrix.

For convenience, an abridged table derived from that URL, showing the relevant content on 13th October 2020 for mlnx-ofed47, mlnx-ofed49, mlnx-ofed50, and mlnx-ofed51, is reproduced here:
The `mlnx-ofed49` stack is a LTS release, aimed mainly at supporting older hardware. The stack may be useful for one of the following:

- ConnectX-3 Pro cards
- ConnectX-3 cards
- Connect-IB cards
- RDMA experimental verbs library (`mlnx_lib`)

For other use cases it usually makes sense to get the most recent supported stack. Each stack version needs to be matched to a firmware version associated with the OFED device used. OFED devices used, ConnectX, BlueField, and others, are also matched along with their firmware version at the URL [https://www.mellanox.com/support/mlnx-ofed-matrix](https://www.mellanox.com/support/mlnx-ofed-matrix). Deviating from compatible versions is not supported.

Details on the compatibility of older stack versions can also be found via that URL.

Returning back to the subject of OFED package installation via the package manager: For example, a `yum install` command indicated by:

```
yum install <vendor-ofedVersion>
```

means that the installation of the Bright Computing OFED package is executed with one of these corresponding `yum install` commands:

- `yum install mlnx-ofed47`
- `yum install mlnx-ofed49`
- `yum install mlnx-ofed50`
- `yum install mlnx-ofed51`
Installing The OFED Stack Provided By The Bright Computing Repository Vendor Package

Running the package manager command associated with the distribution (yum install, zypper up, apt install), unpacks and installs or updates several packages and scripts. For example:

```
$ yum install <vendor-ofedVersion>
```

However, it does not carry out the installation and configuration of the driver itself due to the fundamental nature of the changes it would carry out. The script:

```
<vendor-ofedVersion>-install.sh
```

can be used after the package manager installation to carry out the installation and configuration of the driver itself. The script can be run on the nodes as follows:

- **On the head node,** the default distribution OFED software stack can be replaced with the vendor OFED software stack made available from the Bright Computing repository, by using the script’s head option, `-h`:

  ```
  [root@bright91~]# /cm/local/apps/<vendor-ofedVersion>/current/bin/<vendor-ofedVersion>-install.sh -h
  
  A reboot is recommended after the script completes the install.
  ```

- **For a software image,** for example `default-image`, used by the regular nodes, the default distribution OFED software stack can be replaced with the vendor OFED software stack made available from the Bright Computing repository, by using the script’s software image option, `-s`:

  ```
  [root@bright91~]# /cm/local/apps/<vendor-ofedVersion>/current/bin/<vendor-ofedVersion>-install.sh -s default-image
  
  A reboot updates the software image on the regular node.
  ```

If the distribution kernel is updated on any of these head or regular nodes after the vendor OFED stack has been installed, then the vendor OFED kernel modules made available from the Bright Computing repository must be recompiled and reinstalled. This can be done by running the installation scripts again. This replaces the kernel modules, along with all the other OFED packages again.

The OFED Stack provided by Bright Computing can be removed by appending the `-r` option to the appropriate `-h` or `-s` option. Removing the packages from a head node or software image can lead to package dependency breakage, and software not working any more. So using the “-r” option should be done with caution.

Upgrading Kernels When The OFED Stack Has Been Provided By The Bright Computing Repository Vendor Package—Reinstallation Of The OFED Stack

For all distributions, as explained in the preceding text, a vendor OFED stack is installed and configured via the script `<vendor-ofedVersion>-install.sh`. OFED reinstallation may be needed if the kernel is upgraded.

**In Ubuntu:** if the OFED stack is installed from the distribution or vendor OFED .deb packages, then the DKMS (Dynamic Kernel Module System) framework makes upgraded vendor OFED kernel modules available at a higher preference than the distribution OFED kernel modules for a standard distribution kernel. If there is a kernel upgrade that causes unexpected behavior from the vendor OFED package, then the cluster administrator can still configure the distribution OFED for use by setting the distribution OFED kernel module as the preferred kernel module. So no kernel-related packages need to be excluded from vendor OFED upgrades or kernel upgrades. Typically, Ubuntu clusters can have a package update (apt upgrade) carried out, with no explicit changes needed to take care of the OFED stack.

© Bright Computing, Inc.
For RHEL and derivatives, and SLES: if the OFED stack is installed from the vendor OFED RPM packages, then the script customizes the vendor OFED stack for the existing kernel, and replaces the distribution stack. However, updating the kernel afterwards, without updating the stack along with it, could lead to unexpected behavior due to the customization. Kernel and kernel development updates are therefore prevented from taking place by a package management system block. Updating the kernel, kernel development, and OFED stack in such a configuration therefore requires that the administrator manually overrides the block so that the OFED stack can be handled with consideration.

The following procedure can thus be followed to update and install the kernel packages and OFED stack:

1. Overriding the block (not needed for Ubuntu):
   - In Red Hat-based systems, the `/etc/yum.conf` file must be edited. In that file, in the line that starts with `exclude`, the `kernel` and `kernel-devel` packages need to be removed, so that they are no longer excluded from updates.
   - In SUSE, the `kernel-default` and `kernel-default-devel` packages must be unlocked. The command:
     ```
     zypper removelock kernel-default kernel-default-devel
     ```
     unlocks them so that they can take part in updates again.

2. Updating the kernel and kernel development packages:
   - `yum update`—or for SUSE `zypper up`—updates the packages on the head node.
   - To update the packages on the regular nodes the procedure outlined in section 12.3.3 of the `Administrator Manual` is followed:
     - The packages on the regular node image (for example, `default-image`) are updated according to distribution:
       * in Red Hat-based systems as follows:
         ```
         yum --installroot=/cm/images/default-image update
         ```
       * or in SLES as follows:
         ```
         zypper --root=/cm/images/default-image up
         ```
       * or in Ubuntu as follows, using the `cm-chroot-sw-img` tool (page 504 of the `Administrator Manual`):
         ```
         root@bright91:~# cm-chroot-sw-img /cm/images/default-image
         root@bright91:~# apt update; apt upgrade  #upgrade takes place in image
         ...
         root@bright91:~# exit  #get out of chroot
         ```
     - The `kernelversion` setting for the regular node image, which in this example is the default `default-image`, can be updated as follows:

       **Example**
       ```
       [root@bright91 ~]# cmsh
       [bright91]# softwareimage
       [bright91->softwareimage]# use default-image
       [bright91->softwareimage[default-image]]# set kernelversion 3.10.0-327.3.1.el7.x86_64
       [bright91->softwareimage[default-image*]]# commit
       ```

     This ensures that the updated kernel is used after reboot. Tab-completion in the `set kernelversion` line prompts for the right kernel from available options.
3. A reboot of the head and regular nodes installs the new kernel.

4. Configuring and installing the vendor OFED stack driver for the new kernel is done by running the script `<vendor-ofedVersion>-install.sh` as before, as follows:

   - For a stack that is on the head node, the compilation should be done together with the `-h` option:
     ```bash
     [root@bright91 ~]# /cm/local/apps/<vendor-ofedVersion>/current/bin/<vendor-ofedVersion>-install.sh -h
     ```

   - For a software image used by the regular nodes, for example default-image, the compilation should be done together with the `-s` option:
     ```bash
     [root@bright91 ~]# /cm/local/apps/<vendor-ofedVersion>/current/bin/<vendor-ofedVersion>-install.sh -s default-image
     ```

These configuration and installation steps for the vendor OFED driver are typically not needed for Ubuntu.

### 7.7 Intel OPA Software Stack

The Intel Omni-Path Architecture (OPA) Software Stack is available for RHEL7 and derivatives, and SLES 12.

#### 7.7.1 Installation

After the head node has been installed, the Intel OPA Software Stack can be installed by executing the following commands on the head node:

```bash
[root@bright91 ~]# yum install intel-opa
```

The `yum` command installs the package containing the OPA stack itself, as well as the installation scripts required for installing and configuring the kernel drivers. These are automatically placed under a subdirectory named after the OPA stack version.

#### 7.7.2 Configuration And Deployment

The drivers must then be configured for the running kernel, and the OPA stack deployed on the head node. If the subdirectory for the OPA stack version is called `<version>`, then the following commands can be executed to carry out the configuration and deployment:

For the head node:

```bash
/cm/local/apps/intel-opa/<version>/bin/intel-opa-install.sh -h
```

For the software images, the OPA stack can be configured and deployed for each software image as follows:

```bash
/cm/local/apps/intel-opa/<version>/bin/intel-opa-install.sh -s <name of software image>
```

The OPA MTU size is not changed by Bright Cluster Manager during installation. An Intel recommendation at the time of writing (October 2017) in https://www.intel.com/content/dam/support/us/en/documents/network-and-i-o/fabric-products/Intel_OP_Performance_Tuning_UG_H93143_v3_0.pdf, in section 6.2 is:

**OPA on the other hand can support MTU sizes from 2048B (2K) up to 8192B (8KB) for verbs or PSM 2 traffic. Intel recommends you use the 8KB MTU default for RDMA requests of 8KB or more.**
Burning Nodes

The burn framework is a component of Bright Cluster Manager 9.1 that can automatically run test scripts on specified nodes within a cluster. The framework is designed to stress test newly built machines and to detect components that may fail under load. Nodes undergoing a burn session with the default burn configuration, lose their filesystem and partition data for all attached drives, and revert to their software image on provisioning after a reboot.

8.1 Test Scripts Deployment

The framework requires power management to be running and working properly so that the node can be power cycled by the scripts used. In modern clusters power management is typically achieved by enabling a baseboard management controller such as IPMI, iLO, or Redfish. Details on power management are given in Chapter 4 of the Administrator Manual.

The framework can run any executable script. The default test scripts are mostly bash shell scripts and Perl scripts. Each test script has a directory in /cm/shared/apps/cmburn containing the script. The directory and test script must have the same name. For example: /cm/shared/apps/cmburn/disktest/disktest is the default script used for testing a disk. More on the contents of a test script is given in section 8.3.2.

8.2 Burn Configurations

A burn configuration is an XML file stored in the CMDaemon database that specifies the burn tests and the order in which they run. Within the burn configuration the tests are normally grouped into sequences, and several sequences typically make up a phase. Phases in turn are grouped in either a pre-install section or post-install section. A simple example of such a burn configuration could therefore look like:

Example

```xml
<?xml version="1.0"?>
<burnconfig>

<mail>
  <address>root@master</address>
  <address>some@other.address</address>
</mail>

<pre-install>

  <phase name="01-hwinfo">
    <test name="hwinfo"/>
    <test name="hwdiff"/>

© Bright Computing, Inc.
8.2.1 Mail Tag
The optional <mail> tag pair can add a sequence of e-mail addresses, with each address enclosed in an <address> tag pair. These addresses receive burn failure and warning messages, as well as a notice when the burn run has completed.

8.2.2 Pre-install And Post-install
The pre-install part of a burn configuration is configured with the <pre-install> tag pair, and run from inside a node-installer environment. This environment is a limited Linux environment and allows some simpler tests to run before loading up the full Linux node environment.

Similarly, the post-install part of a burn configuration uses the <post-install> tag pair to run from inside the full Linux node environment. This environment allows more complex tests to run.

8.2.3 Post-burn Install Mode
The optional <post-burn-install> tag pair allows the administrator to specify the install mode (section 5.4.4 of the Administrator Manual) after burn. The tag pair can enclose a setting of AUTO, FULL, MAIN, or NOSYNC. The default setting is the install mode that was set before burn started.

8.2.4 Phases
The phases sections must exist. If there is no content for the phases, the phases tags must still be in place (“must exist”). Each phase must have a unique name and must be written in the burn configuration file in alphanumerical order. By default, numbers are used as prefixes. The phases are executed in sequence.

8.2.5 Tests
Each phase consists of one or more test tags. The tests can optionally be passed arguments using the args property of the burn configuration file (section 8.2). If multiple arguments are required, they should be a space separated list, with the (single) list being the args property.

Tests in the same phase are run simultaneously.
Most tests test something and then end. For example, the disk test tests the performance of all drives and then quits.

Tests which are designed to end automatically are known as non-endless tests.

Tests designed to monitor continuously are known as endless tests. Endless tests are not really endless. They end once all the non-endless tests in the same phase are ended, thus bringing an end to the phase. Endless tests typically test for errors caused by the load induced by the non-endless tests. For example the mce_check test continuously keeps an eye out for Machine Check Exceptions while the non-endless tests in the same phase are run.

A special test is the final test, memtest86, which is part of the default burn run, as configured in the XML configuration default-destructive. It does run endlessly if left to run. To end it, the administrator can deal with its output at the node console or can power reset the node. It is usually convenient to remove memtest86 from the default XML configuration in larger clusters, and to rely on the HPL and memtester tests instead, for uncovering memory hardware errors.

8.3 Running A Burn Configuration

Burn configurations can be viewed and executed from cmsh.

8.3.1 Burn Configuration And Execution In cmsh

Burn Configuration File Settings

From cmsh, the burn configurations can be accessed from partition mode as follows:

Example

```
[bright91] partition use base
[bright91->partition[base]]% burnconfigs
[bright91->partition[base]->burnconfigs]% list

Name (key) Description XML
------------------- ------------------------ -------------
default-destructive Standard burn test. <2614 bytes> 
long-hpl Run HPL test for a long+ <829 bytes>
```

The values of a particular burn configuration (default-destructive in the following example) can be viewed as follows:

Example

```
[bright91->partition[base]->burnconfigs]% use default-destructive
[bright91->partition[base]->burnconfigs[default-destructive]]% show

Parameter Value
---------------------- --------------------------------------------------------
Description Standard destructive burn test. Beware, wipes the disks! 
Name default-destructive 
Revision 
XML <2614 bytes>
```

The set command can be used to modify existing values of the burn configuration, that is: Description, Name, and XML. XML is the burn configuration file itself. The get xml command can be used to view the file, while using set xml opens up the default text editor, thus allowing the burn configuration to be modified.

A new burn configuration can also be added with the add command. The new burn configuration can be created from scratch with the set command. However, an XML file can also be imported to the new burn configuration by specifying the full path of the XML file to be imported:

Example
The burn configuration can also be edited when carrying out burn execution with the burn command.

**Executing A Burn**

A burn as specified by the burn configuration file can be executed in cmsh using the burn command of device mode.

**Burn commands:** The burn commands can modify these properties, as well as execute other burn-related operations.

The burn commands are executed within device mode, and are:

- burn start
- burn stop
- burn status
- burn log

The burn help text that follows lists the detailed options. Next, operations with the burn commands illustrate how the options may be used along with some features.

```
[bright91->partition[base]->burnconfigs]% add boxburn
[bright91->partition[base]->burnconfigs*[boxburn*]]% set xml /tmp/im.xml
```

© Bright Computing, Inc.
Include all nodes that have the given image, e.g default-image or default-image,gpu-image

-t, --type <type>
Type of devices, e.g node or virtualnode,cloudnode

-i, --intersection
Calculate the intersection of the above selections

-u, --union
Calculate the union of the above selections

-l, --role role
Filter all nodes that have the given role

-s, --status <status>
Only run command on nodes with specified status, e.g. UP, "CLOSED|DOWN", "INST.*"

--config <name>
Burn with the specified burn configuration. See in partition burn configurations for a list of valid names

--file <path>
Burn with the specified file instead of burn configuration

--later
Do not reboot nodes now, wait until manual reboot

--edit
Open editor for last minute changes

--no-drain
Do not drain the node from WLM before starting to burn

--no-undrain
Do not undrain the node from WLM after burn is complete

-p, --path
Show path to the burn log files. Of the form: /var/spool/burn/<mac>.

-v, --verbose
Show verbose output (only for burn status)

--sort <field1>[,<field2>,...]
Override default sort order (only for burn status)

-d, --delimiter <string>
Set default row separator (only for burn status)

-d, --delay <seconds>
Wait <seconds> between executing two sequential power commands. This option is ignored for the status command
Examples:

```
burn --config default-destructive start -n node001
```

**Burn command operations:** Burn commands allow the following operations, and have the following features:

- **start, stop, status, log:** The basic burn operations allow a burn to be started or stopped, and the status of a burn to be viewed and logged.
  - The “`burn start`” command always needs a configuration file name. In the following it is boxburn. The command also always needs to be given the nodes it operates on:
    ```
    [bright91->device] burn --config boxburn -n node007 start
    Power reset nodes
    [bright91->device] ipmi0 .................... [ RESET ] node007
    Fri Nov 3 ... [notice] bright91: node007 [ DOWN ]
    [bright91->device] Fri Nov 3 ... [notice] bright91: node007 [ INSTALLING ] (node installer started)
    [bright91->device] Fri Nov 3 ... [notice] bright91: node007 [ INSTALLING ] (running burn in tests)
    ...
    ```
  - The “`burn stop`” command only needs to be given the nodes it operates on, for example:
    ```
    [bright91->device] burn -n node007 stop
    ```
  - The “`burn status`” command:
    ```
    * may be given the nodes for which the status is to be found, for example:
    [bright91->device] burn status -n node005..node007
    Hostname Burn name Status New burn on PXE Phase ...
    -------- --------- ------------------------- --------------- -------- ...
    node005 no burn results available no ...
    node006 currently not burning no ...
    node007 boxburn Burning yes 02-disks ...
    ```
    * Each line of output is quite long, so each line has been rendered truncated and ellipsized.
    * The ellipsis marks in the 5 preceding output lines align with the lines that follow.
    * That is, the lines that follow are the endings of the preceding 5 lines:
    ```
    ...Warnings Tests
    ...
    ...
    ...
    /var/spool/burn/c8-1f-66-f2-61-c0/02-disks/disktest (S,171),
    /var/spool/burn/c8-1f-66-f2-61-c0/02-disks/kmon (S),
    /var/spool/burn...
    ```

- The “`burn log`” command displays the burn log for specified node groupings. Each node with a boot MAC address of `<mac>` has an associated burn log file.
  - Advanced options include the following:
- `-n|--nodes, -g|--group, -c|--category, -r|--rack, -h|--chassis, -e|--overlay, -t|--type, -i|--intersection, -u|--union, -r|--role, -s|--status`: Burn commands can be executed over various node groupings.

- `--config <burn configuration>, --file <path>`: The burn configuration file can be set as one of the XML burn configurations listed in partition mode, or it can be set as the path of the XML file from the file system.

- `-l|--later`: This option disables the immediate power reset that occurs on running the “burn start” or “burn stop” command on a node. This allows the administrator to power down manually, when preferred.

- `-e|--edit`: The burn configuration file can be edited with the `-e` option for the “burn start” command. This is an alternative to editing the burn configuration file in partition mode.

- `--no-drain, --no-undrain`: The node is not drained before burning, or not undrained after burning.

- `-l|--later`: This option disables the immediate power reset that occurs on running the “burn start” or “burn stop” command on a node. This allows the administrator to power down manually, when preferred.

- `-p|--path`: This shows the burn log path, by default (section 8.4) on the head node under a directory named after the mac address `/var/spool/burn/<mac>`.

**Example**

```bash
[bright91->device]# burn -p -n node001..node003 log
node001: /var/spool/burn/fa-16-3e-60-1d-c7
node002: /var/spool/burn/fa-16-3e-8a-d8-0e
node003: /var/spool/burn/fa-16-3e-af-38-cd
```

**Burn command output examples:** The burn status command has a compact one-line output per node:

**Example**

```bash
[bright91->device]# burn -n node001 status
node001 (00000000a000) - W(0) phase 02-disks 00:02:58 (D:H:M) FAILED, mce_check (SP),
disktest (SF,61), kmon (SP)
```

The fields in the preceding output example are:

© Bright Computing, Inc.
<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
<th>Meaning Here</th>
</tr>
</thead>
<tbody>
<tr>
<td>The node name</td>
<td>node001</td>
<td></td>
</tr>
<tr>
<td>The node tag</td>
<td>(00000000a000)</td>
<td></td>
</tr>
<tr>
<td>Warnings since start of burn</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>The current phase name</td>
<td>02-disks</td>
<td>Burn configuration phase being run is 02-disks</td>
</tr>
<tr>
<td>Time since phase started</td>
<td>00:02:58 (D:H:M)</td>
<td>2 hours 58 minutes</td>
</tr>
<tr>
<td>State of current phase</td>
<td>FAILED</td>
<td>Failed in 02-disks</td>
</tr>
<tr>
<td>burn test for MCE</td>
<td>mce_check (SP)</td>
<td>Started and Passed</td>
</tr>
<tr>
<td>burn test for disks</td>
<td>disktest (SF,61)</td>
<td>Started and Failed 61 is the speed, and is custom information</td>
</tr>
<tr>
<td>burn test kernel log monitor</td>
<td>kmon (SP)</td>
<td>Started and Passed</td>
</tr>
</tbody>
</table>

Each test in a phase uses these letters to display its status:

<table>
<thead>
<tr>
<th>Letter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>started</td>
</tr>
<tr>
<td>W</td>
<td>warning</td>
</tr>
<tr>
<td>F</td>
<td>failed</td>
</tr>
<tr>
<td>P</td>
<td>passed</td>
</tr>
</tbody>
</table>

The “burn log” command output looks like the following (some output elided):

```
bright91->device]$ burn -n node001 log
Thu ... 2012: node001 - burn-control: burn framework initializing
Thu ... 2012: node001 - burn-control: e-mail will be sent to: root@master
Thu ... 2012: node001 - burn-control: finding next pre-install phase
Thu ... 2012: node001 - burn-control: starting phase 01-hwinfo
Thu ... 2012: node001 - burn-control: starting test /cm/shared/apps/cmburn/hwinfo
Thu ... 2012: node001 - burn-control: starting test /cm/shared/apps/cmburn/sleep
Thu ... 2012: node001 - sleep: sleeping for 10 seconds
Thu ... 2012: node001 - hwinfo: hardware information
Thu ... 2012: node001 - hwinfo: CPU1: vendor_id = AuthenticAMD
...
Thu ... 2012: node001 - burn-control: test hwinfo has ended, test passed
Thu ... 2012: node001 - burn-control: test sleep has ended, test passed
Thu ... 2012: node001 - burn-control: all non-endless test are done, terminating endless tests
Thu ... 2012: node001 - burn-control: phase 01-hwinfo passed
Thu ... 2012: node001 - burn-control: finding next pre-install phase
Thu ... 2012: node001 - burn-control: starting phase 02-disks
Thu ... 2012: node001 - burn-control: starting test /cm/shared/apps/cmburn/disktest
```
8.3 Running A Burn Configuration

The output of the `burn log` command is actually the `messages` file in the burn directory, for the node associated with a MAC-address directory `<mac>`. The burn directory is at `/var/spool/burn/` and the `messages` file is thus located at:

```
/var/spool/burn/<mac>/messages
```

The tests have their log files in their own directories under the MAC-address directory `<mac>`, using their phase name. For example, the pre-install section has a phase named `01-hwinfo`. The output logs of this test are then stored under:

```
/var/spool/burn/<mac>/01-hwinfo/
```

### 8.3.2 Writing A Test Script

This section describes a sample test script for use within the burn framework. The script is typically a shell or Perl script. The sample that follows is a Bash script, while the `hpl` script is an example in Perl. Section 8.1 describes how to deploy the script.

#### Non-endless Tests

The following example test script is not a working test script, but can be used as a template for a non-endless test:

**Example**

```
#!/bin/bash

# We need to know our own test name, amongst other things for logging.
me=`basename $0`

# This first argument passed to a test script by the burn framework is a
# path to a spool directory. The directory is created by the framework.
```
# Inside the spool directory a sub-directory with the same name as the
# test is also created. This directory ($spooldir/$me) should be used
# for any output files etc. Note that the script should possibly remove
# any previous output files before starting.
spooldir=$1

# In case of success, the script should touch $passedfile before exiting.
passedfile=$spooldir/$me.passed

# In case of failure, the script should touch $failedfile before exiting.
# Note that the framework will create this file if a script exits without
# creating $passedfile. The file should contain a summary of the failure.
failedfile=$spooldir/$me.failed

# In case a test detects trouble but does not want the entire burn to be
# halted $warningfile and $passedfile should be created. Any warnings
# should be written to this file.
warningfile=$spooldir/$me.warning

# Some short status info can be written to this file. For instance, the
# stresscpu test outputs something like 13/60 to this file to indicate
# time remaining.
# Keep the content on one line and as short as possible!
statusfile=$spooldir/$me.status

# A test script can be passed arguments from the burn configuration. It
# is recommended to supply default values and test if any values have
# been overridden from the config file. Set some defaults:
option1=40
option2=some_other_value

# Test if option1 and/or option2 was specified (note that $1 was to
# spooldir parameter):
if [ ! x$2 = "x" ]; then
   option1=$2
fi
if [ ! x$3 = "x" ]; then
   option2=$3
fi

# Some scripts may require some cleanup. For instance a test might fail
# and be
# restarted after hardware fixes.
rm -f $spooldir/$me/*.out &>/dev/null

# Send a message to the burn log file, syslog and the screen.
# Always prefix with $me!
blog "$me: starting, option1 = $option1 option2 = $option2"

# Run your test here:
run-my-test
if [ ! its_all_good ]; then
   blog "$me: w00t, it's all good! my-test passed."
   touch $passedfile
   exit 0

elif [ was_a_problem ]; then
  blog "$me: WARNING, it did not make sense to run this test. You don't have special device X."
  echo "some warning" >> $warningfile  # note the append!
  touch $passedfile
  exit 0
else
  blog "$me: Aiii, we're all gonna die! my-test FAILED!"
  echo "Failure message." > $failedfile
  exit 0
fi

Endless Tests

The following example test script is not a working test, but can be used as a template for an endless test.

Example

#!/bin/bash

# We need to know our own test name, amongst other things for logging.
me=`basename $0`

# This first argument passed to a test script by the burn framework is a
# path to a spool directory. The directory is created by the framework.
# Inside the spool directory a sub-directory with the same name as the
# test is also created. This directory ($spooldir/$me) should be used
# for any output files etc. Note that the script should possibly remove
# any previous output files before starting.
spooldir=$1

# In case of success, the script should touch $passedfile before exiting.
passedfile=$spooldir/$me.passed

# In case of failure, the script should touch $failedfile before exiting.
# Note that the framework will create this file if a script exits without
# creating $passedfile. The file should contain a summary of the failure.
failedfile=$spooldir/$me.failed

# In case a test detects trouble but does not want the entire burn to be
# halted $warningfile _and_ $passedfile should be created. Any warnings
# should be written to this file.
warningfile=$spooldir/$me.warning

# Some short status info can be written to this file. For instance, the
# stresscpu test outputs something like 13/60 to this file to indicate
# time remaining.
# Keep the content on one line and as short as possible!
statusfile=$spooldir/$me.status

# Since this in an endless test the framework needs a way of stopping it
# once all non-endless test in the same phase are done. It does this by
# calling the script once more and passing a "-terminate" argument.
if [ "$2" == "-terminate" ]; then
  blog "$me: terminating"

  # remove the lock file the main loop is checking for
  rm $spooldir/$me/running
"
```bash
blog "$me: waiting for $me to stop"
# wait for the main loop to die
while [ -d /proc/*/grave.ts1 ]
  do
    sleep 1
  done
blog "$me: terminated"
else
  blog "$me: starting test, checking every minute"

# Some scripts may require some cleanup. For instance a test might fail
# and be restarted after hardware fixes.
rm -f $spooldir/$me/*.out &>/dev/null

# create internal lock file, the script will remove this if it is
# requested to end
touch $spooldir/$me/running

# save our process id
echo $$ > "$spooldir/$me/pid"

while [ -e "$spooldir/$me/running" ]
  do
    run-some-check
    if [ was_a_problem ]; then
      blog "$me: WARNING, something unexpected happened."
      echo "some warning" >> $warningfile # note the append!
    elif [ failure ]; then
      blog "$me: Aiii, we're all gonna die! my-test FAILED!"
      echo "Failure message." > $failedfile
    fi
    sleep 60
  done

# This part is only reached when the test is terminating.
if [ ! -e "$failedfile" ]; then
  blog "$me: no problem detected"
touch $passedfile
else
  blog "$me: test ended with a failure"
fi

8.3.3 Burn Failures
Whenever the burn process fails, the output of the burn log command shows the phase that has failed and that the burn terminates.

Example
Thu ... 2012: node001 - burn-control: phase 02-disks FAILED
Thu ... 2012: node001 - burn-control: burn will terminate
```
8.4 Relocating The Burn Logs

Here, burn-control, which is the parent of the disk testing process, keeps track of the tests that pass and fail. On failure of a test, burn-control terminates all tests.

The node that has failed then requires intervention from the administrator in order to change state. The node does not restart by default. The administrator should be aware that the state reported by the node to CMDaemon remains burning at this point, even though it is not actually doing anything.

To change the state, the burn must be stopped with the `burn stop` command in cmsh. If the node is restarted without explicitly stopping the burn, then it simply retries the phase at which it failed.

Under the burn log directory, the log of the particular test that failed for a particular node can sometimes suggest a reason for the failure. For retries, old logs are not overwritten, but moved to a directory with the same name, and a number appended indicating the try number. Thus:

Example

First try, and failing at 02-disks tests:

```
ls -ld 02-disks*/
drwxr-xr-x 6 root root 4096 Jan 10 16:26 02-disks
```

2nd try, after failing again:

```
ls -ld 02-disks*/
drwxr-xr-x 6 root root 4096 Jan 10 16:49 02-disks
```

3rd try, after failing again:

```
ls -ld 02-disks*/
drwxr-xr-x 6 root root 4096 Jan 10 16:59 02-disks
```

8.4 Relocating The Burn Logs

A burn run can append substantial amounts of log data to the default burn spool at `/var/spool/burn`. To avoid filling up the head node with such logs, they can be appended elsewhere.

8.4.1 Configuring The Relocation

The 3-part procedure that can be followed is:

1. The `BurnSpoolDir` setting can be set in the CMDaemon configuration file on the head node, at `/cm/local/apps/cmd/etc/cmd.conf`. The `BurnSpoolDir` setting tells CMDaemon where to look for burn data when the burn status is requested through cmsh.
   - `BurnSpoolDir="/var/spool/burn"`
   
   CMDaemon should be restarted after the configuration has been set. This can be done with:
   
   ```
   service cmd restart
   ```

2. The `burnSpoolHost` setting, which matches the host, and `burnSpoolPath` setting, which matches the location, can be changed in the node-installer configuration file on the head node, at `/cm/node-installer/scripts/node-installer.conf` (for multi-arch/multidistro configurations the path takes the form: `/cm/node-installer-<distribution>-<architecture>/scripts/node-installer.conf`). These have the following values by default:
   - `burnSpoolHost = master`

© Bright Computing, Inc.
• burnSpoolPath = /var/spool/burn

These values define the NFS-mounted spool directory.
The burnSpoolHost value should be set to the new DNS host name, or to an IP address. The burnSpoolPath value should be set to the new path for the data.

3. Part 3 of the procedure adds a new location to export the burn log. This is only relevant if the spool directory is being relocated within the head node. If the spool is on an external fileserver, the existing burn log export may as well be removed.

The new location can be added to the head node as a path value, from a writable filesystem export name. The writable filesystem export name can most easily be added using Bright View, via the clickpath:

Devices→Head Nodes→Edit→Settings→Filesystem exports→Add

Adding a new name like this is recommended, instead of just modifying the path value in an existing Filesystem exports name. This is because changing things back if the configuration is done incorrectly is then easy. By default, the existing Filesystem exports for the burn directory has the name:

• /var/spool/burn@internalnet

and has a path associated with it with a default value of:

• /var/spool/burn

When the new name is set in Filesystem exports, the associated path value can be set in agreement with the values set earlier in parts 1 and 2.

If using cmsh instead of Bright View, then the change can be carried out from within the fsexports submode. Section 3.10.1 of the Administrator Manual gives more detail on similar examples of how to add such filesystem exports.

8.4.2 Testing The Relocation

To test the changes, it is wise to first try a single node with a short burn configuration. This allows the administrator to check that install and post-install tests can access the spool directories. Otherwise there is a risk of waiting hours for the pre-install tests to complete, only to have the burn abort on the post-install tests. The following short burn configuration can be used:

Example

<burnconfig>
<pre-install>
<phase name="01-hwinfo">
<test name="hwinfo"/>
<test name="sleep" args="10"/>
</phase>
</pre-install>
<post-install>
<phase name="02-mprime">
<test name="mprime" args="2"/>
<test name="mce_check" endless="1"/>
<test name="kmon" endless="1"/>
</phase>
</post-install>
</burnconfig>
To burn a single node with this configuration, the following could be run from the device mode of \texttt{cmsh}:

\textbf{Example}

\begin{verbatim}
[bright91->device]% burn start --config default-destructive --edit -n node001
\end{verbatim}

This makes an editor pop up containing the default burn configuration. The content can be replaced with the short burn configuration. Saving and quitting the editor causes the node to power cycle and start its burn.

The example burn configuration typically completes in less than 10 minutes or so, depending mostly on how fast the node can be provisioned. It runs the \texttt{mprime} test for about two minutes.
9

Installing And Configuring SELinux

9.1 Introduction

Security-Enhanced Linux (SELinux) can be enabled on selected nodes. On a standard Linux operating system where SELinux is enabled, it is typically initialized in the kernel during the execution of the init script inside the initrd when booting from a hard drive. However, in the case of nodes provisioned by Bright Cluster Manager, via PXE boot, the SELinux initialization occurs at the very end of the node installer phase.

SELinux is disabled by default because its security policies are typically customized to the needs of the organization using it. The administrator is therefore the one who must decide on appropriate access control security policies. When creating such custom policies special care should be taken that the cmd process is executed in, ideally, an unconfined context.

Before enabling SELinux on a cluster, the administrator is advised to first check that the Linux distribution used offers enterprise support for SELinux-enabled systems. This is because support for SELinux should be provided by the distribution in case of issues.

Enabling SELinux is only advised by Bright Cluster Manager if the internal security policies of the organization absolutely require it. This is because it requires custom changes from the administrator. If something is not working right, then the effect of these custom changes on the installation must also be taken into consideration, which can sometimes be difficult.

9.2 Enabling SELinux On RHEL7

9.2.1 Case 1: Head And Compute Nodes Are Permissive

Head Node Configuration

SELINUX is set to permissive in /etc/selinux/config:

```
[root@bright91 ~]# grep SELINUX /etc/selinux/config
# SELINUX= can take one of these three values:
SELINUX=permissive
```

The head node is then rebooted to implement the setting, and the setting can be checked with the getenforce command:

```
[root@bright91 ~]# getenforce
Permissive
```

Regular Node Configuration

- SELINUX is set to Permissive in /etc/selinux/config in the relevant software images.
- The SELinux configuration parameters in node-installer.conf are set as follows:
Installing And Configuring SELinux

For multiarch/multidistro configurations the node-installer path in the preceding session takes the form: `/cm/node-installer-<distribution>-<architecture>/scripts/node-installer.conf`. The compute nodes are then rebooted to implement the settings. A node can be checked with the `getenforce` command:

```
[root@node001 ~]# getenforce
Permissive
```

### 9.2.2 Case 2: Head Node Is Permissive And Compute Nodes Are Enforcing

**Head Node Configuration**

SELINUX is set to permissive in `/etc/selinux/config`:

```
[root@bright91 ~]# grep SELINUX /etc/selinux/config
# SELINUX= can take one of these three values:
SELINUX=permissive
```

The head node is then rebooted to implement the setting, and the setting can be checked with the `getenforce` command:

```
[root@bright91 ~]# getenforce
Permissive
```

**Regular Node Configuration**

- SELINUX is set to Enforcing in `/etc/selinux/config` in the relevant software images:

- The SELinux configuration parameters in `node-installer.conf` are set as follows:

```
[root@bright91 ~]# grep SEL /cm/node-installer/scripts/node-installer.conf
SELinuxInitialize = true
SELinuxFileContextActionInNoSyncInstallMode = 2
SELinuxFileContextActionInAutoInstallMode = 2
SELinuxFileContextActionInFullInstallMode = 2
SELinuxRebootAfterCompleteFilesystemContextRelabeling = false
SELinuxUseNFSHomeDirs = true
```

The compute nodes are then rebooted to implement the settings. A node can be checked with the `getenforce` command:

```
[root@node001 ~]# getenforce
Enforcing
```
9.2.3 Case 3: Head Node Is Enforcing And Compute Nodes Are Enforcing

Head Node Configuration

SELinux is set to enforcing in `/etc/selinux/config`:

```bash
[root@bright91 ~]# grep SELINUX /etc/selinux/config
# SELINUX= can take one of these three values:
SELINUX=enforcing
```

The head node is then rebooted to implement the setting, and the setting can be checked with the `getenforce` command:

```bash
[root@bright91 ~]# getenforce
Enforcing
```

Regular Node Configuration

- SELinux is set to enforcing in `/etc/selinux/config` in the relevant software images:

- The SELinux configuration parameters in `node-installer.conf` are set as follows:

  ```bash
  Example
  [root@bright91 ~]# grep SEL /cm/node-installer/scripts/node-installer.conf
  SELinuxInitialize = true
  SELinuxFileContextActionInNoSyncInstallMode = 2
  SELinuxFileContextActionInAutoInstallMode = 2
  SELinuxFileContextActionInFullInstallMode = 2
  SELinuxRebootAfterCompleteFilesystemContextRelabeling = false
  SELinuxUseNFSHomeDirs = true
  ```

The compute nodes are then rebooted to implement the settings. A node can be checked with the `getenforce` command:

```bash
[root@node001 ~]# getenforce
Enforcing
```

9.3 Additional Considerations

9.3.1 Provisioning The .autorelabel File Tag

It is advised that the `/cm/images/<image>/autorelabel` file only get transferred to the regular nodes during a FULL node installation. I.e., it should not be transferred during the AUTO node installation. This can be achieved by appending the following entry to the `excludelistsyncinstall` property of the node category:

```
no-new-files: - /.autorelabel
```

Why this is done is explained in section 9.4.

9.3.2 SELinux Warnings During Regular Node Updates

When software images are updated (section 12.4 of the *Administrator Manual*), messages such as the following may be displayed:

```
SELinux: Could not downgrade policy file /etc/selinux/targeted/policy/
        policy.24, searching for an older version.
SELinux: Could not open policy file <= /etc/selinux/targeted/policy/po
        licy.24: No such file or directory
```

These messages are displayed if the SELinux status cannot be retrieved. For a default image, the SELinux status is disabled by default, in which case the messages can safely be ignored.
9.4 Filesystem Security Context Checks

Ensuring that the files present on the node have correct security contexts applied to them is an important part of enforcing a security policy.

- In the case of the head nodes, if SELinux is enabled or permissive in /etc/selinux/config, then the filesystem security context check is performed by the default system startup scripts. This is, by default, performed only if the presence of the /.autorelabel file is detected on the root filesystem.

- In the case of a regular node, SELinux can be set to enabled or permissive for the node image, by default in /cm/images/default-image/etc/selinux/config. So far, this is reasonably similar to how it is done on the head node.

However, the filesystem security context check process is significantly different due to additional SELinux configuration processing for a regular node in the node-installer, which is responsible for the correctness of security contexts on the filesystem of the node. The SELinux filesystem security context processing is determined by SELinux directives in the /cm/node-installer/scripts/node-installer.conf configuration file (for multiarch/multidistro configurations the path takes the form: /cm/node-installer-*distribution>*architecture>/scripts/node-installer.conf).

The directives in the file decide the behavior of how SELinux filesystem security contexts are set up.

- If the regular node has undergone full provisioning, and the /.autorelabel file exists on the node’s local filesystem, then by default the security contexts of all eligible filesystems of that node are restored by the node installer. This behavior is analogous to what the startup subsystem scripts would normally do. However, since the node installer removes the /.autorelabel file after performing the context restore, the operating system startup script does not detect it once the system boot continues. This behavior occurs for the default setting of SELinuxFileContextActionInFullInstallMode 0 for an enabled or permissive SELinux for that node.

- If the regular node has undergone a sync provisioning (e.g. installed in AUTO mode), then by default the node installer will only restore the security context on the files which were modified during provisioning and on files which were generated by the node installer itself. This is typically significantly faster than performing a full filesystem security context restore on all eligible filesystems. This behavior occurs for the default setting of SELinuxFileContextActionInAutoInstallMode 0 for an enabled or permissive SELinux for that node.

- If the regular node has undergone a nosync installation, then by default there is no SELinux file security context check in the node-installer. Any /.autorelabel file is removed, and there is subsequently no SELinux file security context check in the system startup scripts. This behavior occurs for the default setting of SELinuxFileContextActionInNoSyncInstallMode 2 for an enabled or permissive SELinux for that node.

The SELinux filesystem security context configuration behavior described in the preceding can be altered in the node-installer.conf file. For example, it is always possible to force a full filesystem security context restore in the AUTO install mode, or to leave the context checking to the operating system’s startup scripts. Details on how the SELinuxFileContextActionsIn*InstallMode directives can be modified and what the changes mean are given in that file.
Other Licenses, Subscriptions, Or Support Vendors

Bright Cluster Manager comes with enough software to allow it to work with no additional commercial requirements other than its own. However, Bright Cluster Manager integrates with some other products that have their own separate commercial requirements. The following table lists commercial software that requires a separate license, subscription, or support vendor, and an associated URL where more information can be found.

<table>
<thead>
<tr>
<th>Software</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Workload managers</strong></td>
<td></td>
</tr>
<tr>
<td>PBS Pro</td>
<td><a href="http://www.pbsworks.com">http://www.pbsworks.com</a></td>
</tr>
<tr>
<td>MOAB</td>
<td><a href="http://www.adaptivecomputing.com">http://www.adaptivecomputing.com</a></td>
</tr>
<tr>
<td>UGE</td>
<td><a href="http://www.univa.com">http://www.univa.com</a></td>
</tr>
<tr>
<td><strong>Distributions</strong></td>
<td></td>
</tr>
<tr>
<td>Suse</td>
<td><a href="http://www.suse.com">http://www.suse.com</a></td>
</tr>
<tr>
<td>Red Hat</td>
<td><a href="http://www.redhat.com">http://www.redhat.com</a></td>
</tr>
<tr>
<td><strong>Compilers</strong></td>
<td></td>
</tr>
<tr>
<td>PGI High-performance</td>
<td><a href="http://www.pgroup.com/">http://www.pgroup.com/</a></td>
</tr>
<tr>
<td><strong>Miscellaneous</strong></td>
<td></td>
</tr>
</tbody>
</table>

...continues
...continued

<table>
<thead>
<tr>
<th>Software</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon AWS</td>
<td><a href="http://aws.amazon.com">http://aws.amazon.com</a></td>
</tr>
</tbody>
</table>
Hardware Recommendations

The hardware suggestions in section 3.1 are for a minimal cluster, and are inadequate for larger clusters.

For larger clusters, hardware suggestions and examples are given in this section.

The memory used depends significantly on CMDaemon, which is the main Bright Cluster Manager service component, and on the number of processes running on the head node or regular node. The number of processes mostly depends on the number of metrics and health checks that are run.

Hard drive storage mostly depends on the number of metrics and health checks that are managed by CMDaemon.

B.1 Heuristics For Requirements

Normal system processes run on the head and regular node if the cluster manager is not running, and take up their own RAM and drive space.

B.1.1 Heuristics For Requirements For A Regular Node

A calculation of typical regular node requirements can be made as follows:

Regular Node Disk Size
For disked nodes, a disk size of around 16 GB is the minimum needed. 128GB should always be fine at the time of writing (May 2018). The disk size should be large enough to hold the entire regular node image that the head node supplies to it, which typically is around 5GB, along with swap, log files and other local overhead for the jobs that will run on the regular node.

Regular Node Memory Size
The total RAM required is roughly the sum of:
RAM used for non-Bright system processes + 50MB + (number of nodes × 10kB).

B.1.2 Heuristics For Requirements For A Head Node

A calculation of typical head node requirements can be made as follows:

Head Node Disk Size
The disk size required is roughly the sum of:
space needed by operating system without cluster manager + 5GB per regular node image + (100kB × number of metrics and health checks × number of devices)

A device means any item seen as a device by CMDaemon. A list of devices can be seen by cmsh under its device node. Examples of devices are: regular nodes, cloud nodes, switches, head nodes, GPU units, and PDUs.
Head Node Memory Size
The total RAM required is roughly the sum of:
\[ \text{RAM used for normal system process} + 100\text{MB} + (\text{number of nodes} \times 1.8\text{MB}) \]

This assumes less than 100 metrics and health checks are being measured, which is a default for systems that are just head nodes and regular nodes. Beyond the first 100 metrics and health checks, each further 100 extra take about 1MB extra per device.

B.2 Observed Head Node Resources Use, And Suggested Specification

B.2.1 Observed Head Node Example CMDaemon And MySQL Resources Use
CMDaemon and MySQL have the following approximate default resource usage on the head node as the number of nodes increases:

<table>
<thead>
<tr>
<th>Number of nodes</th>
<th>CMDaemon + MySQL RAM/GB</th>
<th>CMDaemon RAM/GB</th>
<th>Disk Use/GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>16</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>2000</td>
<td>32</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>5000</td>
<td>64</td>
<td>10</td>
<td>50</td>
</tr>
</tbody>
</table>

B.2.2 Suggested Head Node Specification For Clusters Beyond 1000 Nodes
For clusters with more than 1000 nodes, a head node is recommended with at least the following specifications:

- 24 cores
- 128 GB RAM
- 512 GB SSD

The extra RAM is useful for caching the filesystem, so scrimping on it makes little sense. Handy for speedy retrievals is to place the monitoring data files, which are by default located under `/var/spool/cmd/monitoring/`, on an SSD.

A dedicated `/var` or `/var/lib/mysql` partition for clusters with greater than 2500 nodes is also a good idea.