NVIDIA Base Command Manager 11

Installation Manual

Revision: d7928f6

Date: Wed Oct 21 2026

©2026 NVIDIA Corporation & affiliates. All Rights Reserved. This manual or parts thereof may not be reproduced in any form unless permitted by contract or by written permission of NVIDIA Corporation.

Trademarks

Linux is a registered trademark of Linus Torvalds. PathScale is a registered trademark of Cray, Inc. Red Hat and all Red Hat-based trademarks are trademarks or registered trademarks of Red Hat, Inc. SUSE is a registered trademark of SUSE LLC. NVIDIA, CUDA, GPUDirect, HPC SDK, NVIDIA DGX, NVIDIA Nsight, and NVLink are registered trademarks of NVIDIA Corporation. FLEXlm is a registered trademark of Flexera Software, Inc. Green Provisioning is a trademark of Altair Engineering, Inc. All other trademarks are the property of their respective owners.

Rights and Restrictions

All statements, specifications, recommendations, and technical information contained herein are current or planned as of the date of publication of this document. They are reliable as of the time of this writing and are presented without warranty of any kind, expressed or implied. NVIDIA Corporation shall not be liable for technical or editorial errors or omissions which may occur in this document. NVIDIA Corporation shall not be liable for any damages resulting from the use of this document.

Limitation of Liability and Damages Pertaining to NVIDIA Corporation

The NVIDIA Base Command Manager product principally consists of free software that is licensed by the Linux authors free of charge. NVIDIA Corporation shall have no liability nor will NVIDIA Corporation provide any warranty for the NVIDIA Base Command Manager to the extent that is permitted by law. Unless confirmed in writing, the Linux authors and/or third parties provide the program as is without any warranty, either expressed or implied, including, but not limited to, marketability or suitability for a specific purpose. The user of the NVIDIA Base Command Manager product shall accept the full risk for the quality or performance of the product. Should the product malfunction, the costs for repair, service, or correction will be borne by the user of the NVIDIA Base Command Manager product. No copyright owner or third party who has modified or distributed the program as permitted in this license shall be held liable for damages, including general or specific damages, damages caused by side effects or consequential damages, resulting from the use of the program or the un-usability of the program (including, but not limited to, loss of data, incorrect processing of data, losses that must be borne by you or others, or the inability of the program to work together with any other program), even if a copyright owner or third party had been advised about the possibility of such damages unless such copyright owner or third party has signed a writing to the contrary.

Table of Contents

Table of Contents . . . 3

1 Quickstart Installation Guide **7**
1.1 Installing The Head Node . . . 7
1.2 First Boot . . . 10
1.3 Booting Regular Nodes . . . 10
1.4 Quickstart For GPUs . . . 12
1.5 Optional: Upgrading Python . . . 15
1.6 Running Base View . . . 16

2 Introduction **19**
2.1 What Is NVIDIA Base Command Manager? . . . 19
2.1.1 What OS Platforms Is It Available For? . . . 19
2.1.2 What Architectures Does It Run On? . . . 20
2.1.3 What Features Are Supported Per OS And Architecture? . . . 20
2.1.4 What OS Platforms Can It Be Managed From? . . . 20
2.2 Cluster Structure . . . 20

3 Installing NVIDIA Base Command Manager **23**
3.1 Minimal Hardware Requirements . . . 23
3.1.1 Head Node . . . 23
3.1.2 Compute Nodes . . . 23
3.2 Supported Hardware . . . 24
3.2.1 Compute Nodes . . . 24
3.2.2 Ethernet Switches . . . 24
3.2.3 Power Distribution Units . . . 24
3.2.4 Management Controllers . . . 25
3.2.5 InfiniBand And Similar . . . 25
3.2.6 GPUs . . . 25
3.2.7 RAID . . . 25
3.3 Head Node Installation: Bare Metal Method . . . 25
3.3.1 ISO Boot Menu . . . 26
3.3.2 Welcome Screen . . . 27
3.3.3 Software Licenses . . . 27
3.3.4 Kernel Modules Configuration . . . 29
3.3.5 Hardware Info . . . 30
3.3.6 Installation Source . . . 31
3.3.7 Cluster Settings . . . 32
3.3.8 Workload Management Configuration . . . 33
3.3.9 Network Topology . . . 34
3.3.10 Head Node Settings . . . 37

3.3.11 Compute Nodes Settings . . . 38
3.3.12 BMC Configuration . . . 39
3.3.13 Networks . . . 40
3.3.14 Head node interfaces . . . 42
3.3.15 Compute node interfaces . . . 42
3.3.16 Disk Layout . . . 44
3.3.17 Disk Layout Settings . . . 44
3.3.18 Additional Software . . . 47
3.3.19 Summary . . . 48
3.3.20 Deployment . . . 49
3.3.21 Licensing And Further Configuration . . . 50
3.4 Head Node Installation: Ansible Add-On Method . . . 50
3.4.1 An Overview Of Ansible . . . 51
3.4.2 The Head Node Add-on Installer And Ansible . . . 51
3.4.3 Deploying A RHEL Head Node From A Rocky BCM ISO . . . 52
3.5 Enabling Remote Browser-Based Installation Via The Text Mode Installer . . . 54

4 Licensing NVIDIA Base Command Manager 57
4.1 Displaying License Attributes . . . 58
4.1.1 Displaying License Attributes Within Base View . . . 58
4.1.2 Displaying License Attributes Within `cmsh` . . . 59
4.2 Verifying A License—The `verify-license` Utility . . . 59
4.2.1 The `verify-license` Utility Can Be Used When `licenseinfo` Cannot Be Used . . . 60
4.2.2 Using The `verify-license` Utility To Troubleshoot License Issues . . . 60
4.2.3 Using The `versioninfo` Command To Verify The BCM Software Version . . . 61
4.3 Requesting And Installing A License Using A Product Key . . . 62
4.3.1 Is A License Needed?—Verifying License Attributes . . . 62
4.3.2 The Product Key . . . 62
4.3.3 Requesting A License With The `request-license` Script . . . 64
4.3.4 Installing A License With The `install-license` Script . . . 65
4.3.5 Re-Installing A License After Replacing The Hardware . . . 66
4.3.6 Re-Installing A License After Wiping Or Replacing The Hard Drive . . . 67
4.3.7 Rebooting Nodes After An Install . . . 67
4.3.8 Getting Help With Licensing Issues . . . 68

5 Linux Distributions That Use Registration 69
5.1 Registering A Red Hat Enterprise Linux Based Cluster . . . 69
5.1.1 Registering A Head Node With RHEL . . . 69
5.1.2 Registering A Software Image With RHEL . . . 70
5.2 Registering A SUSE Linux Enterprise Server Based Cluster . . . 70
5.2.1 Registering A Head Node With SUSE . . . 70
5.2.2 Registering A Software Image With SUSE . . . 71

6 Changing The Network Parameters Of The Head Node 73
6.1 Introduction . . . 73
6.2 Method . . . 73
6.3 Terminology . . . 73

7 Third Party Software **77**
7.1 Modules Environment . . . 77
7.2 Shorewall . . . 77
7.2.1 The Shorewall Service Paradigm . . . 77
7.2.2 Shorewall Zones, Policies, And Rules . . . 78
7.2.3 Clear And Stop Behavior In `service` Options, `bash` Shell Command, And `cmsh` Shell 79
7.2.4 Adding To Shorewall Configuration Via A Role In `cmsh` . . . 79
7.2.5 Further Shorewall Quirks . . . 82
7.3 The GCC Compiler . . . 83
7.4 AMD GPU Driver Installation . . . 83
7.4.1 AMD GPU Hardware Check . . . 83
7.4.2 AMD GPU Driver Installation Per Supported Distribution . . . 84

8 The NVIDIA HPC SDK **87**
8.1 Packages And Versions . . . 87
8.2 Compiler Modules . . . 88
8.3 Viewing Installed Available CUDA Versions, And The Running CUDA Version . . . 88
8.4 Changing The Running CUDA Version . . . 89

9 CUDA For GPUs **91**
9.1 Installing CUDA . . . 91
9.1.1 CUDA Packages Available . . . 91
9.1.2 CUDA Package And Modules . . . 95
9.2 Verifying CUDA . . . 96
9.3 Verifying OpenCL . . . 98
9.4 NVIDIA Validation Suite (Package: `cuda-dcgm-nvvs`) . . . 99
9.5 Further NVIDIA Configuration Via The Cluster Manager . . . 101

10 OFED Software Stack **103**
10.1 Choosing A Distribution Version, Or A Non-distribution Version, Ensuring The Kernel Matches, And Logging The Installation . . . 103
10.2 DOCA OFED Stack Installation Using The BCM DOCA OFED Installation Script . . . 103

11 Burning Nodes **105**
11.1 Test Scripts Deployment . . . 105
11.2 Burn Configurations . . . 105
11.2.1 Burn Configuration Simple XML Example . . . 105
11.2.2 Burn Configuration XML Schema Definition . . . 107
11.3 Running A Burn Configuration . . . 107
11.3.1 Burn Configuration And Execution In `cmsh` . . . 107
11.3.2 Writing A Test Script . . . 113
11.3.3 Burn Failures . . . 116
11.4 Relocating The Burn Logs . . . 117
11.4.1 Configuring The Relocation . . . 117
11.4.2 Testing The Relocation . . . 118

12 Installing And Configuring SELinux **121**
12.1 Introduction . . . 121
12.2 Enabling SELinux On RHEL8, Rocky 9 . . . 121
12.2.1 Setting SELinux Parameters . . . 121
12.2.2 Setting Up On The Head Node . . . 122
12.2.3 Setting Up On The Regular Nodes . . . 122

A Other Licenses, Subscriptions, Or Support Vendors **125**

B Hardware Recommendations **127**
B.1 Heuristics For Requirements . . . 127
B.1.1 Heuristics For Requirements For A Regular Node . . . 127
B.1.2 Heuristics For Requirements For A Head Node . . . 127
B.2 Observed Head Node Resources Use, And Suggested Specification . . . 128
B.2.1 Observed Head Node Example CMDaemon And MySQL Resources Use . . . 128
B.2.2 Suggested Head Node Specification For Clusters Beyond 1000 Nodes . . . 128

C NVIDIA BCM Scope Of Support **129**
C.1 Included In BCM Support . . . 129
C.1.1 Support For Cluster Installation . . . 129
C.1.2 Support For Cluster Configuration . . . 129
C.1.3 Support For Integration Of Cluster In Site Infrastructure . . . 129
C.1.4 Support For Integration With Microsoft Active Directory . . . 129
C.1.5 Support For First-Party Hardware Only . . . 130
C.1.6 Support For Software Upgrades . . . 130
C.2 Excluded From BCM Support . . . 130
C.2.1 Help With Third-Party Software . . . 130
C.2.2 Help With User Applications . . . 130
C.2.3 Help With Workload Management . . . 130
C.2.4 Help With Performance Issues . . . 130
C.2.5 Root-Cause Analysis . . . 130
C.2.6 Phone Support . . . 130
C.3 Support Outside Of The Standard Scope—Getting Professional Services . . . 131

1

Quickstart Installation Guide

This chapter describes a basic and quick installation of NVIDIA Base Command Manager (BCM) 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 BCM *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://www.nvidia.com/en-us/data-center/bright-cluster-manager/support/`. Section 14.2 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 BCM installation DVD or from a USB flash drive with the DVD ISO on it.

3. The option: `Start Base Command Manager Graphical Installer`, or `Start Base Command ManagerText Installer`, should be selected in the text boot menu. The graphical installation is recommended, and brings up the GUI installation welcome screen. The text installer 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 GUI welcome screen, the `Start installation` button should be clicked.

5. At the EULA screens:

   - At the NVIDIA EULA screen, the acceptance checkbox should be ticked. `Next` should then be ticked.
   - At the Linux base distribution EULA 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 BCM DVD, or the USB flash drive containing the DVD ISO, 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 BCM 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:

- 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 3.10). 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 layout` screen, a drive should be selected for the head node. The installation will be done onto this drive, overwriting all its previous content.

19. At the `Disk layout Settings` screen, 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` in the installer file system is used by default:

Partition	Space	Mounted At	Filesystem Type
1	1G	/boot	ext2
0	100M	/boot/efi	fat
2	16G	-	swap
3	rest	/	xfs

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:

Partition	Space	Mounted At	Filesystem Type
1	1G	/boot	ext2
0	100M	/boot/efi	fat
2	16G	-	swap
3	20G	/tmp	xfs
4	180G	/var	xfs
5	rest	/	xfs

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 layout templates available from a menu, other than the preceding default layouts.

20. 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
    - OFED stack

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

22. 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 BCM 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.

6. The head node software should be updated via its package manager (yum, dnf, apt, zypper) so that it has the latest packages (sections 9.2 -9.3 of the *Administrator Manual*).

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.

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 269, 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 Base View**: The node identification resource (page 273, section 5.4.2 of the *Administrator Manual*) in Base 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:29:54:F5:94  05 Sep 2011 13:49:41 PDT [no port detected]
...
[mycluster]% device newnodes | wc -l
MAC                First appeared      Detected on switch port
-----------------  ------------------- -----------------------
32
[mycluster]% exit
[root@mycluster ~]#
```

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:D2:68:8D  Mon, 05 Sep 2011 13:43:13 PDT node001
00:0C:29:54:F5:94  Mon, 05 Sep 2011 13:49:41 PDT node002
...
[mycluster]% exit
[root@mycluster ~]#
```

5. Each regular node is now provisioned and eventually fully boots. For troubleshooting node booting issues, section 5.8 of the *Administrator Manual* should be consulted.

6. *Optional:* To configure node-level power management, Chapter 4 of the *Administrator Manual* should be consulted.

7. To update the software on the nodes, a package manager is used to install to the node image filesystem that is on the head node.

   The node image filesystem should be updated via its package manager (yum, dnf, apt, zypper) so that it has the latest packages (sections 9.4 -9.5 of the *Administrator Manual*).

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, the presence of GPU hardware on `node001` can be detected by the kernel, even if its associated kernel module software drivers are not loaded:

   Example

   ```
   [root@basecm11 ~]# ssh node001 lspci | grep NVIDIA
   00:07.0 3D controller: NVIDIA Corporation GK110BGL [Tesla K40c] (rev a1)
   ```

   If there is no `3D controller` string, then it typically means that the GPU is too old to be supported, or that there is no GPU actually there. Typically, each GPU displays a line, corresponding to its location on the PCI bus. So 4 GPUs on a node would show 4 lines.

   Supported GPU architectures at the time of writing (May 2026) are Turing, Ampere, Ada Lovelace, Hopper, and Blackwell.

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

   Example

   ```
   [root@basecm11 ~]# 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 7.4.

   (b) For NVIDIA GPUs, assuming the GPU is on the compute node `node001`, and that the hardware is supported by CUDA 13.0, then software installation is carried out at the head node as follows:

i. CUDA toolkit, software development kit, and visual tools are installed on the head node itself with:

Example

```
root@basecm11:~# apt install cuda13.0-toolkit cuda13.0-sdk cuda13.0-visual-tools
```

ii. By default, the image (`/cm/images/default-image`) used by nodes already has the required NVIDIA driver software packages installed. These therefore do not need to be installed explicitly.
Further details on the basic installation of CUDA for NVIDIA GPUs are given in Chapter 9.

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 3.17.2 of the *Administrator Manual*,

(a) on page 198 for NVIDIA GPUs

(b) on page 200 for AMD GPUs.

4. **Configuring GPUs For The Workload Manager**

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

```
[root@basecm11 ~]# cm-wlm-setup
```

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

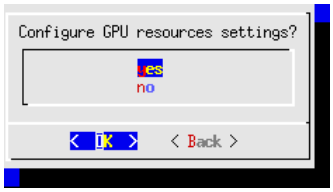

Figure 1.1: Slurm With `cm-wlm-setup`: GPU Configuration Entry Screen

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

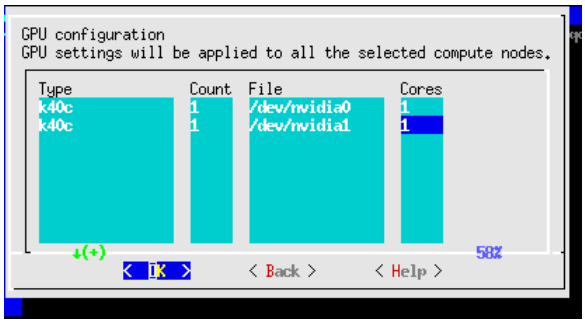

Figure 1.2: Slurm With `cm-wlm-setup`: GPU Device Settings Configuration Screen

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 gres.conf.5`.

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 the NVIDIA CUDA MPS (Multi-Process Service) to be configured:

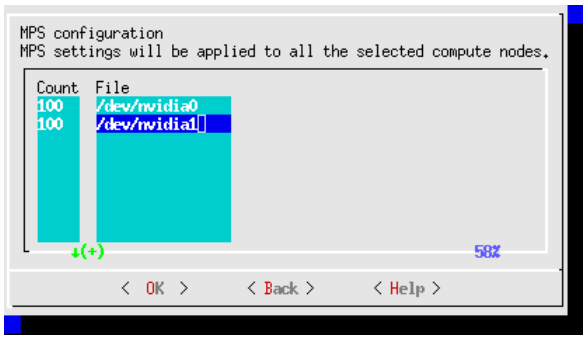

Figure 1.3: Configuring An NVIDIA GPU For Slurm With `cm-wlm-setup`: MPS Settings Configuration Screen

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@basecm11 ~]# 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@basecm11 ~]# cmsh -c 'device; reboot -n node001..node015'
```

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@basecm11 ~]# cmsh
[basecm11]% user add auser
[basecm11->user*[auser*]]% set password
enter new password:
retype new password:
[basecm11->user*[auser*]]% ..
[basecm11->user*]% commit
```

The "Hello World" `helloworld.cu` script from section 6.5.4 of the *User Manual* can be saved in `auser`'s directory, and then compiled for a GPU with `nvcc`:

```
[auser@basecm11 ~]$ module load shared cuda12.3/toolkit
[auser@basecm11 ~]$ nvcc helloworld.cu -o helloworld
```

A `slurmgpuhello.sh` batch script can be built as follows:

```
[auser@basecm11 ~]$ 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:

```
[auser@basecm11 ~]$ module load slurm
[auser@basecm11 ~]$ sbatch slurmgpuhello.sh
Submitted batch job 35
```

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

```
[auser@basecm11 ~]$ 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!
[auser@basecm11 ~]$
```

More about Slurm batch scripts and GPU compilation can be found in Chapter 6 of the *User Manual*.

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 BCM tools and packages rely on some of the later features of Python, and by default BCM installs its own later Python version via a package dependency when needed. So explicit upgrades to Python are not needed for BCM itself. For example, for Rocky Linux 9.5, the default (distribution) version of Python at the time of writing (June 2025) is Python version 3.9, while BCM version 11 installs and deploys a Python version 3.12 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. BCM therefore makes these various versions of Python available for the end user in an integrated manner, as an optional installation.

For example, for NVIDIA Base Command Manager version 11.0, the Python packages that can be installed from the cluster manager repository are:

- cm-python39, for Python 3.9
- cm-python311, for Python 3.11
- cm-python312, for Python 3.12

If the packages are installed, then users can use the modules command to switch the environment to the appropriate Python version. For example, a user fred can switch to Python 3.12 with:

```
[fred@basecm11 ~]$ python -V
Python 3.9.18
[fred@basecm11 ~]$ module load python312
[fred@basecm11 ~]$ python -V
Python 3.12.11
```

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

The Jupyter module is an example of a module that automatically loads its own version of Python. This may cause confusion if the system administrator is not paying attention to which version of Python is active. For example, in BCM 11.0, with Rocky 9.5, if the user is using the distribution version of Python 3.9, and then Jupyter is loaded, the Python version used changes to 3.12 as it automatically loads the python312 module:

Example

```
[root@basecm11 ~]# module list
Currently Loaded Modulefiles:
 1) shared   2) cmsh   3) cmd   4) cluster-tools/master   5) cm-setup/master   6) cm-image/master
[root@basecm11 ~]# python -V
Python 3.9.19
[root@basecm11 ~]# module load jupyter
Loading jupyter/16.0.6
  Loading requirement: python312
[root@basecm11 ~]# python -V
Python 3.12.11
[root@basecm11 ~]# module remove python312
Unloading python312
  ERROR: python312 cannot be unloaded due to a prereq.
    HINT: Might try "module unload jupyter/16.0.6" first.
```

1.6 Running Base View

To run Base View on the cluster from a workstation running X11: A recent web browser should be used, and pointed to

https://<*head node address*>:8081/base-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 should be reasonably modern.
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 NVIDIA Base Command 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 BCM.**
- **The *Containerization Manual* describes how to manage containers with BCM.**
- **The *NVIDIA Mission Control Manual* describes NVIDIA Mission Control capabilities and integration with BCM.**

2

Introduction

This chapter introduces some features of NVIDIA Base Command Manager and describes a basic cluster in terms of its hardware.

2.1 What Is NVIDIA Base Command Manager?

NVIDIA Base Command Manager 11 is a cluster management application built on top of major Linux distributions.

2.1.1 What OS Platforms Is It Available For?

BCM is available for the following OS platforms:

- Red Hat Enterprise Linux Server and some derivatives, such as Rocky Linux
  - Versions 8.*x*
  - Versions 9.*x*.
- SLES versions:
  - SUSE Enterprise Server 15
- Ubuntu versions:
  - Jammy Jellyfish 22.04
  - Noble Numbat 24.04
  - Resolute Raccoon 26.04
  - DGX OS 7 (described at `https://docs.nvidia.com/dgx/dgx-os-7-user-guide/introduction.html`)
  - DGX OS 8 (as of 8.0, x86_64 systems only; described at `https://docs.nvidia.com/dgx/dgx-os-8-user-guide/introduction.html`)

Typically BCM is installed either on a head node, or on a pair of head nodes in a high availability configuration. Images are provisioned to the non-head nodes from the head node(s). The OS for the images is by default the same as that of the head node(s), but the OS used for the images can be changed later on (section 9.6 of the *Administrator Manual*).

At the time of writing of this section in September 2023, the OS platforms listed all support NVIDIA AI Enterprise (`https://docs.nvidia.com/ai-enterprise/index.html`), with the exception for now of RHEL9 and derivatives.

2.1.2 What Architectures Does It Run On?

Within the distributions listed in section 2.1.1, the BCM 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 Features Are Supported Per OS And Architecture?

BCM aims to run so that it is independent of the chosen OS and architecture. That is, it aims to work the same way for each OS or architecture. There are necessarily some limitations to this, and these are documented in a feature matrix at `https://service.bcm.nvidia.com/feature-matrix/`.

2.1.4 What OS Platforms Can It Be Managed From?

While BCM runs directly on the OS platforms listed in section 2.1.1, BCM can be managed from many more OS platforms, when controlled by these front ends:

- Base 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 BCM 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 15 of the *Administrator Manual*) in BCM.

A cluster normally contains a considerable number of non-head, or *regular nodes*, also referred to simply as 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 BCM. 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.

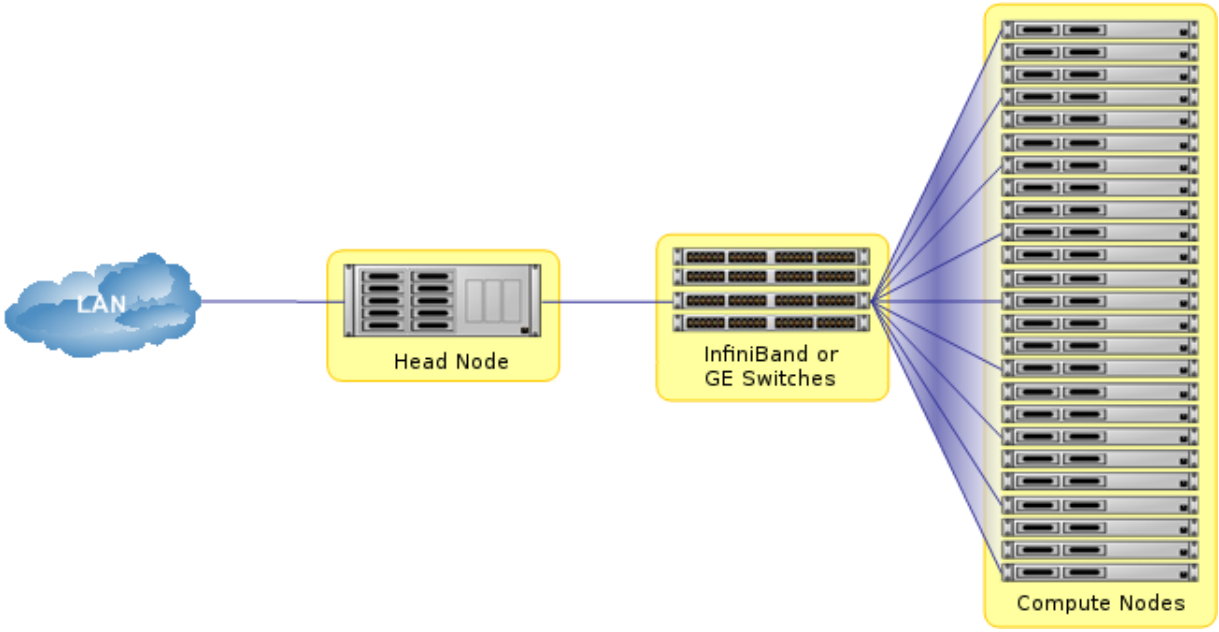

Figure 2.1: Cluster network

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 NVIDIA Base Command Manager

This chapter describes in detail the installation of NVIDIA Base Command 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 BCM 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 BCM is described in Chapter 9 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. However, standard bare-metal installation (section 3.3) runs best with at least 8GB RAM.
  - 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

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

An NVIDIA-certified for AI enterprise system is a system with NVIDIA GPUs and networking that has passed a set of certification tests to validate its performance, reliability, and scale for data centers. Many system builders that have partnered with NVIDIA can provide such certified hardware. These system builders are listed in `https://www.nvidia.com/en-us/data-center/data-center-gpus/qualified-system-catalog`.

The following hardware is also supported, even if it may not be NVIDIA-certified for AI Enterprise:

3.2.1 Compute Nodes

- ASUS
- Atos
- Cisco
- Dell EMC
- Fujitsu
- HPE (Hewlett Packard Enterprise)
- IBM
- Lenovo
- NVIDIA DGX
- Oracle
- Supermicro

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

3.2.2 Ethernet Switches

- Cisco
- Dell
- HP Networking Switches
- Netgear
- 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
- iDRAC
- IPMI 1.5/2.0
- CIMC
- Redfish v1

3.2.5 InfiniBand And Similar

- NVIDIA (formerly Mellanox) HCAs, and most other InfiniBand HCAs
- NVIDIA (formerly Mellanox), and most other InfiniBand switches

3.2.6 GPUs

- NVIDIA GPUs later than Volta, with latest recommended drivers
- NVIDIA DGX OS 7
- NVIDIA DGX OS 8, x86_64 systems only (`B300`, `B200`, `H200`, `H100`, `H800`). As of DGX OS 8.0, ARM64 systems, including `GB200` and `GB300`, are not supported
- AMD Radeon GPUs, as listed at `https://support.amd.com/en-us/kb-articles/Pages/Radeon-Software-for-Linux-Release-Notes.aspx`

The GPUs that are NVIDIA-certified for AI Enterprise are listed in the support matrix at `https://docs.nvidia.com/ai-enterprise/latest/product-support-matrix/index.html#support-matrix`.

3.2.7 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 be a 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 are dealt with correctly. Details on installing BCM onto virtual instances can be found in the Knowledge Articles of the NVIDIA Enterprise Support Portal at `https://enterprise-support.nvidia.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 BCM 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 (figure 3.1).

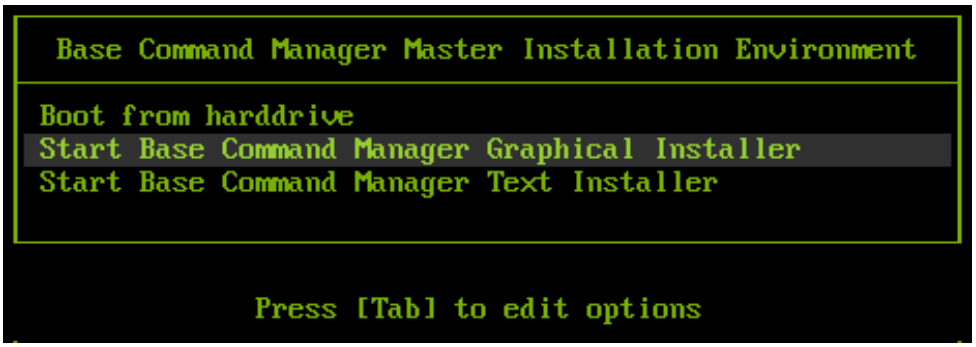

Figure 3.1: Installation: ISO Boot Menu

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 BCM, the countdown should be interrupted by selecting the option of "`Start Base Command Manager Graphical Installer`" 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 BCM 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:
     `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:

```
netconf=eth0:static:10.141.161.253,10.141.255.254,255.255.0.0:secretpass
```

Example

With DHCP addressing:

```
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 3.3).

3.3.2 Welcome Screen

The welcome screen is shown in figure 3.2.

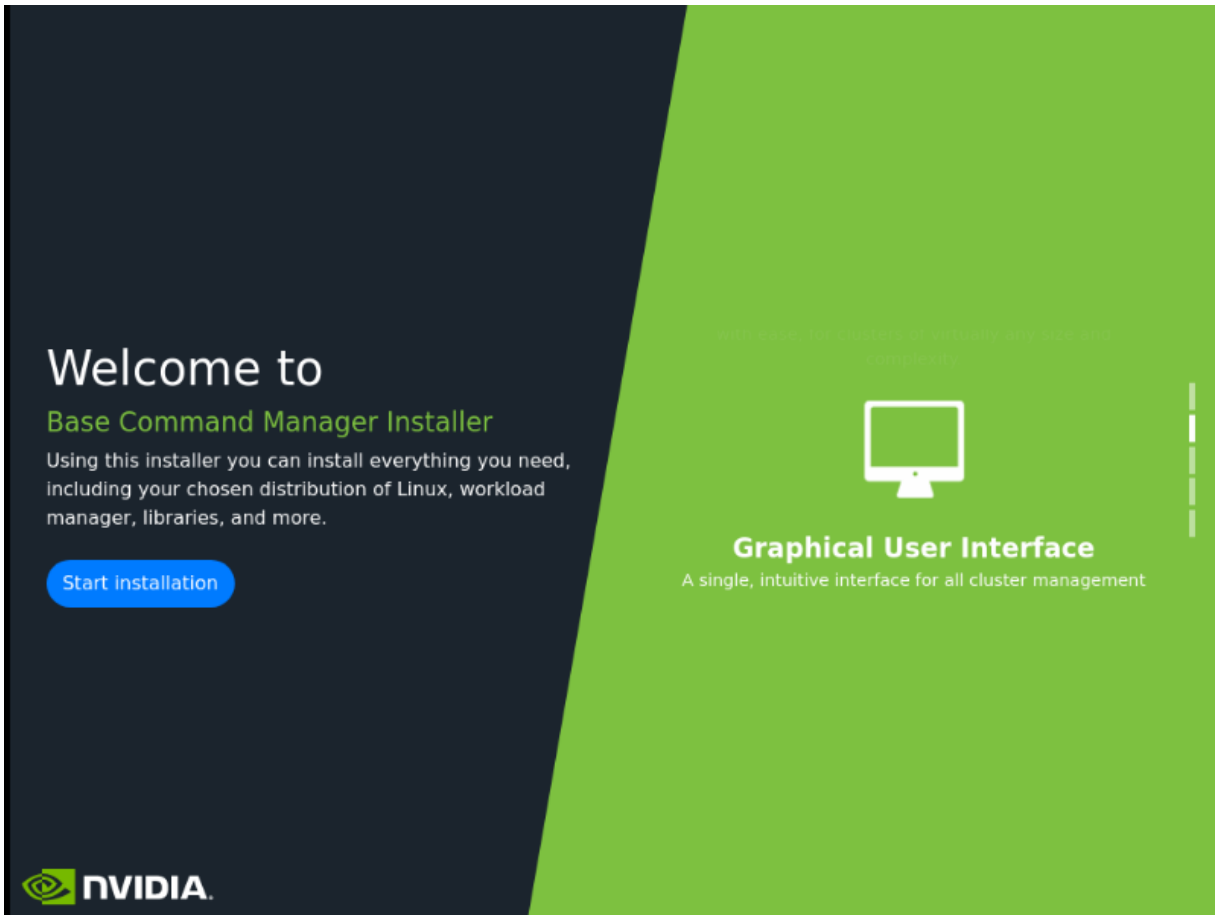

Figure 3.2: Installation welcome screen for BCM

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 read and agree to the NVIDIA Software Licence Agreement (figure 3.3):

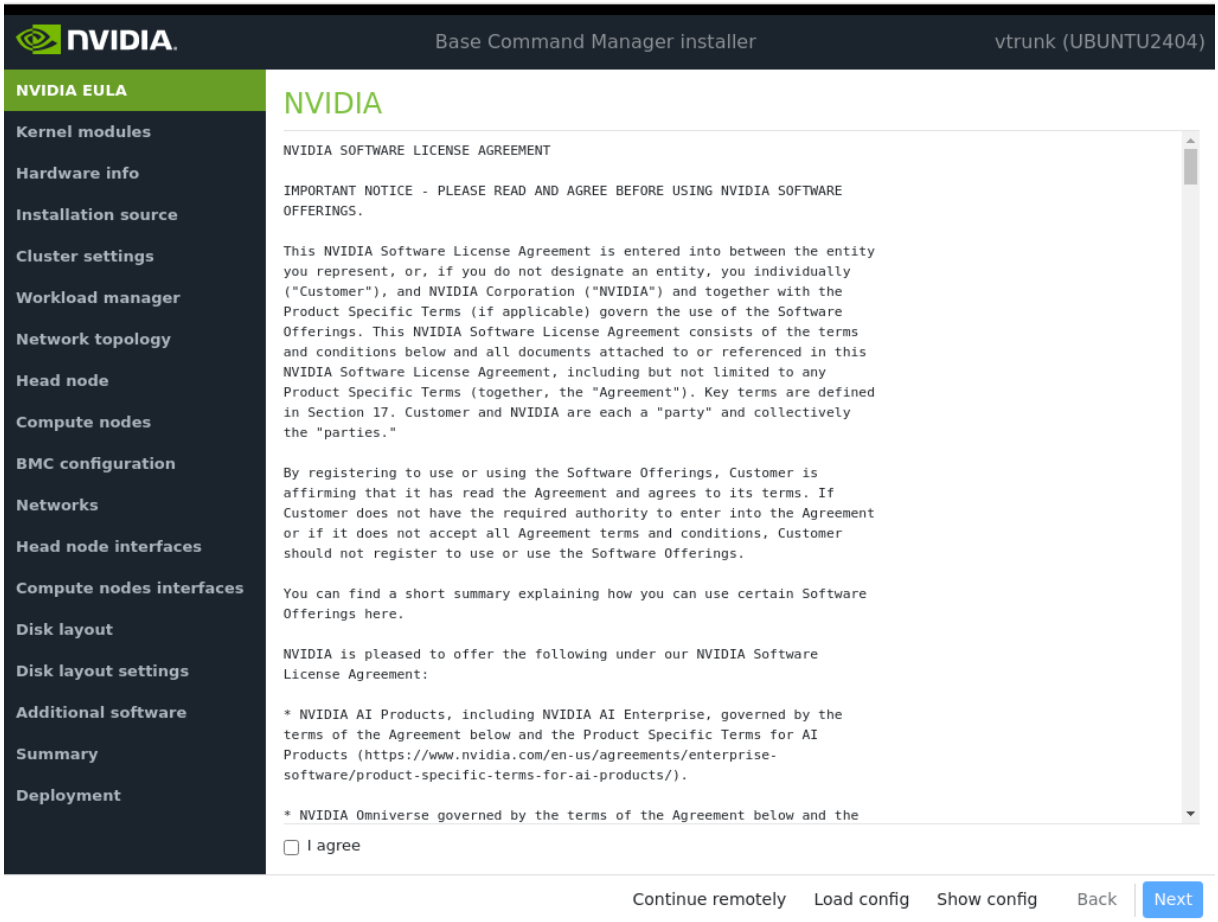

Figure 3.3: NVIDIA Base Command Manager Software License Agreement

The agreement is carried out by ticking the checkbox and clicking on the `Next` button.

Another license agreement screen after that asks the user to agree to the Base Distribution license. This is the end user license agreement for the distribution (RHEL, Rocky, Ubuntu and so on) that is to be used as the base upon which BCM is to run.

Options In GUI During The Installation Screens

As is seen in figure 3.3, the main screens during installation have the following clickable options along the bottom of the screen:

- `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, for example, prefer to avoid working inside a noisy cold data center. If `Continue remotely` is selected, then addresses are displayed on the console screen, for use with a web browser or SSH, and the console installation screen is locked.
- `Back`: if not grayed out, the `Back` option allows the administrator to go back a step in the installation.

- `Next`: allows the administrator to proceed to the next screen.

Console And Log Access During Installation

The cluster administrator can get a root console and view logging by going through the virtual terminals.

For Ubuntu and Rocky the terminals should be allocated as follows:

- Ctrl+Alt+F1: Root shell
- Ctrl+Alt+F2: Installer (GUI/TUI)
- Ctrl+Alt+F3: Logging (at least on Rocky/RHEL)
- Ctrl+Alt+F5+: Additional shells

The allocation may change.

3.3.4 Kernel Modules Configuration

The `Kernel Modules` screen (figure 3.4) shows the kernel modules recommended for loading based on a hardware probe:

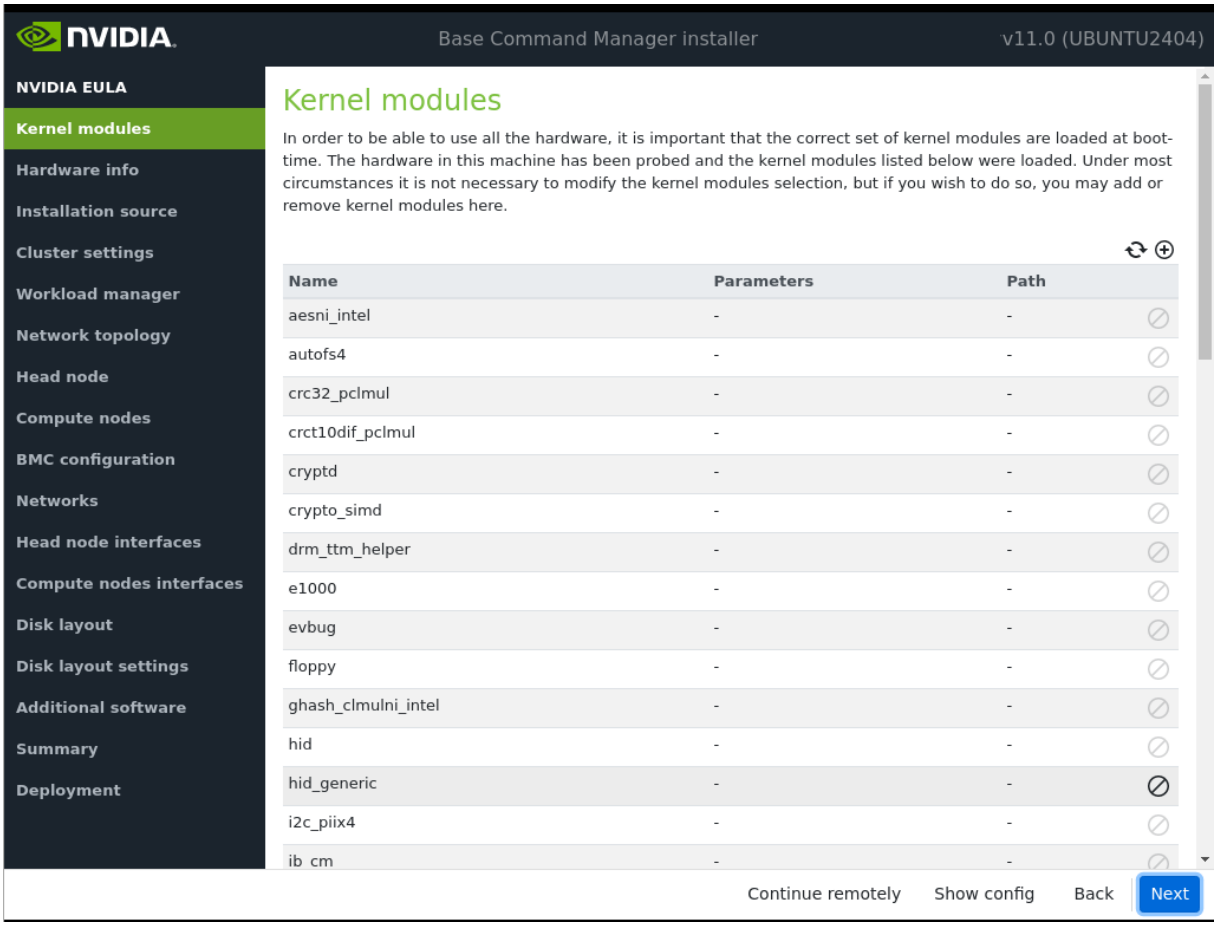

Figure 3.4: Kernel Modules Recommended For Loading After Probing

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 3.5). 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 ⊘ button. This can be useful when replacing one module with another.

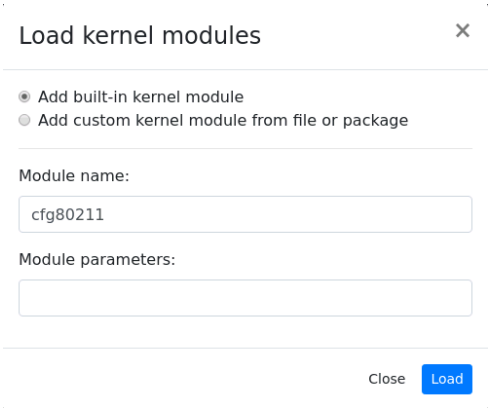

Figure 3.5: Adding Kernel Modules

Clicking `Next` then leads to the "`Hardware info`" overview screen, described next.

3.3.5 Hardware Info

The `Hardware Info` screen (figure 3.6) 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 3.4) to add the appropriate modules, and then the `Hardware Information` screen can be returned to, to see if the hardware has been detected.

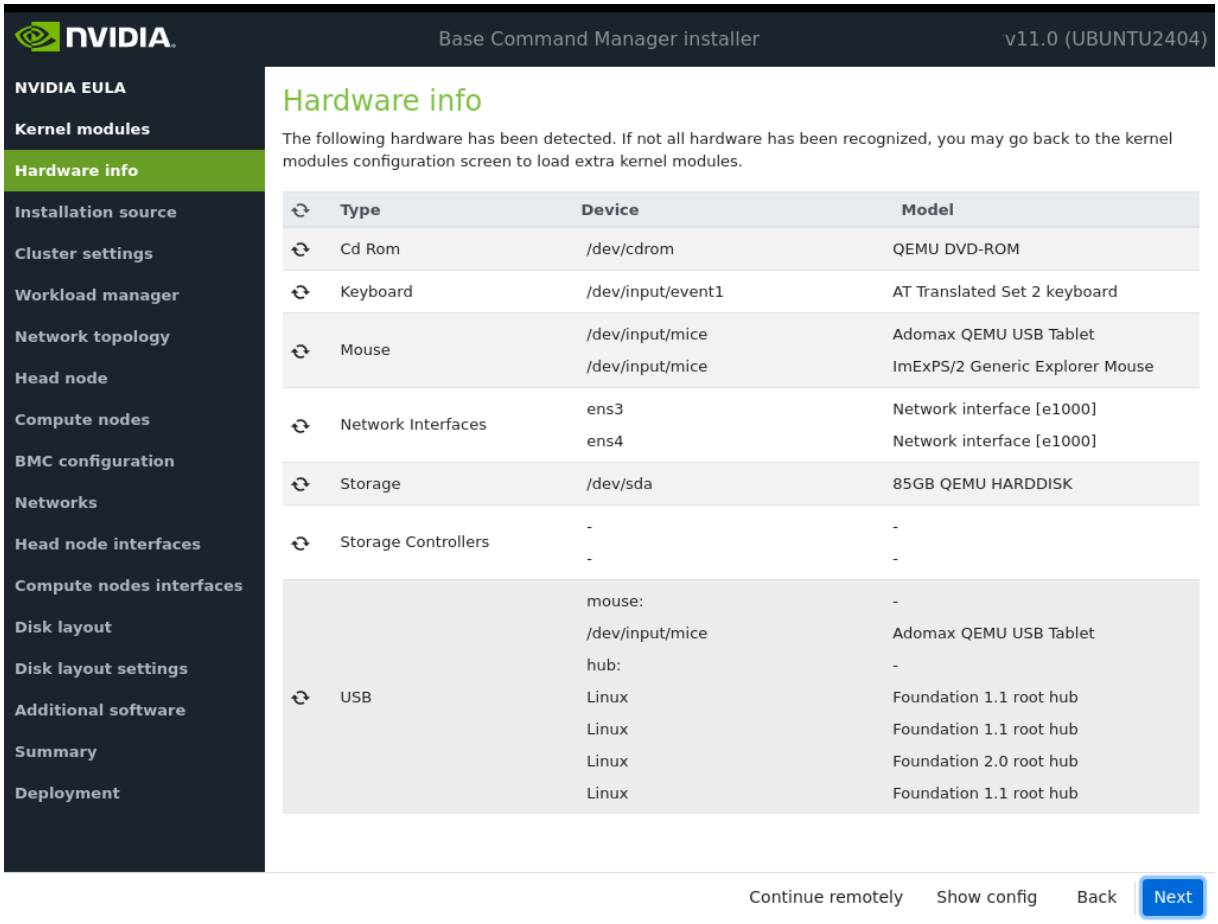

Figure 3.6: Hardware Overview Based On Hardware Detection Used For Loading Kernel Modules

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 3.7) detects the available DVD-ROM devices.

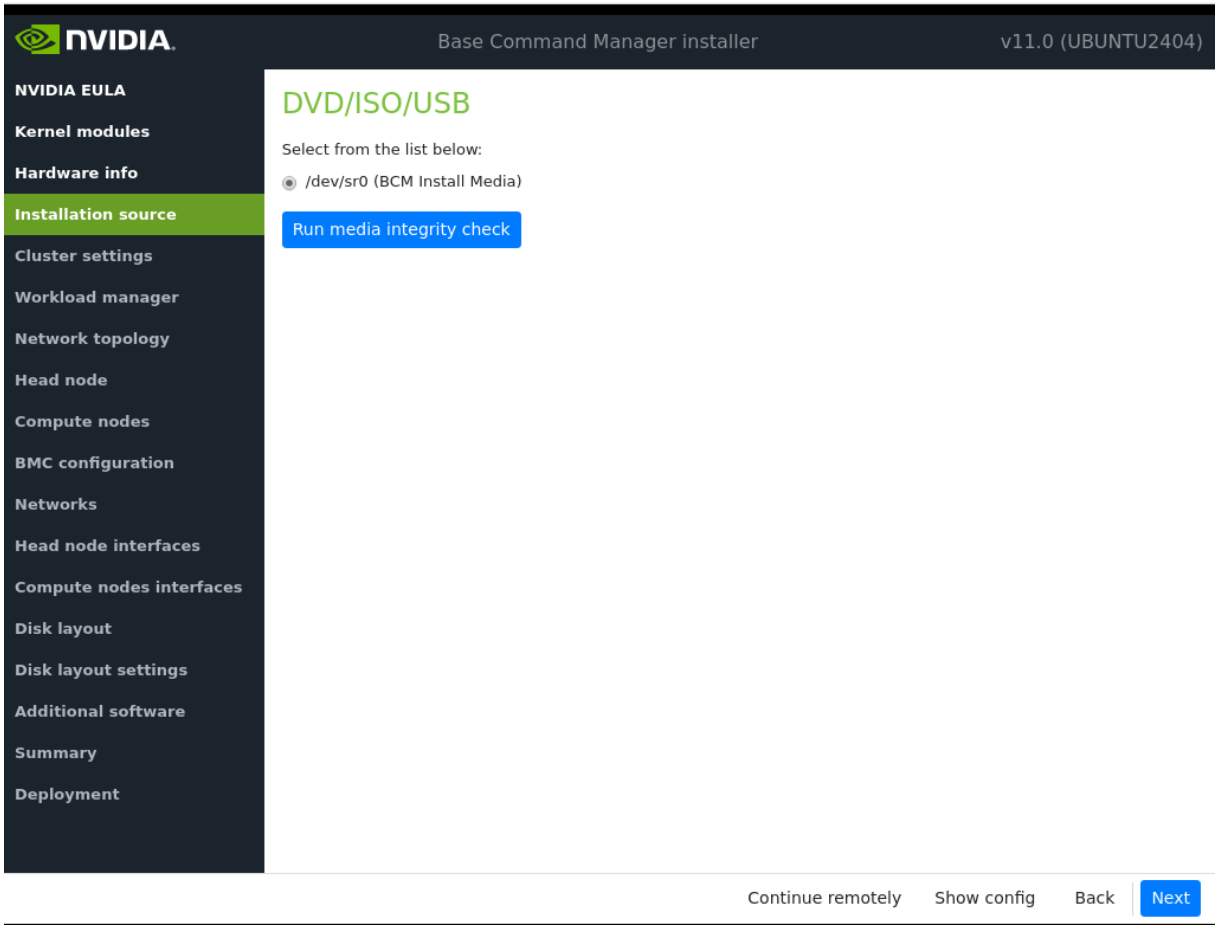

Figure 3.7: DVD Selection

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 3.8) provides an overview of some mandatory and optional settings.

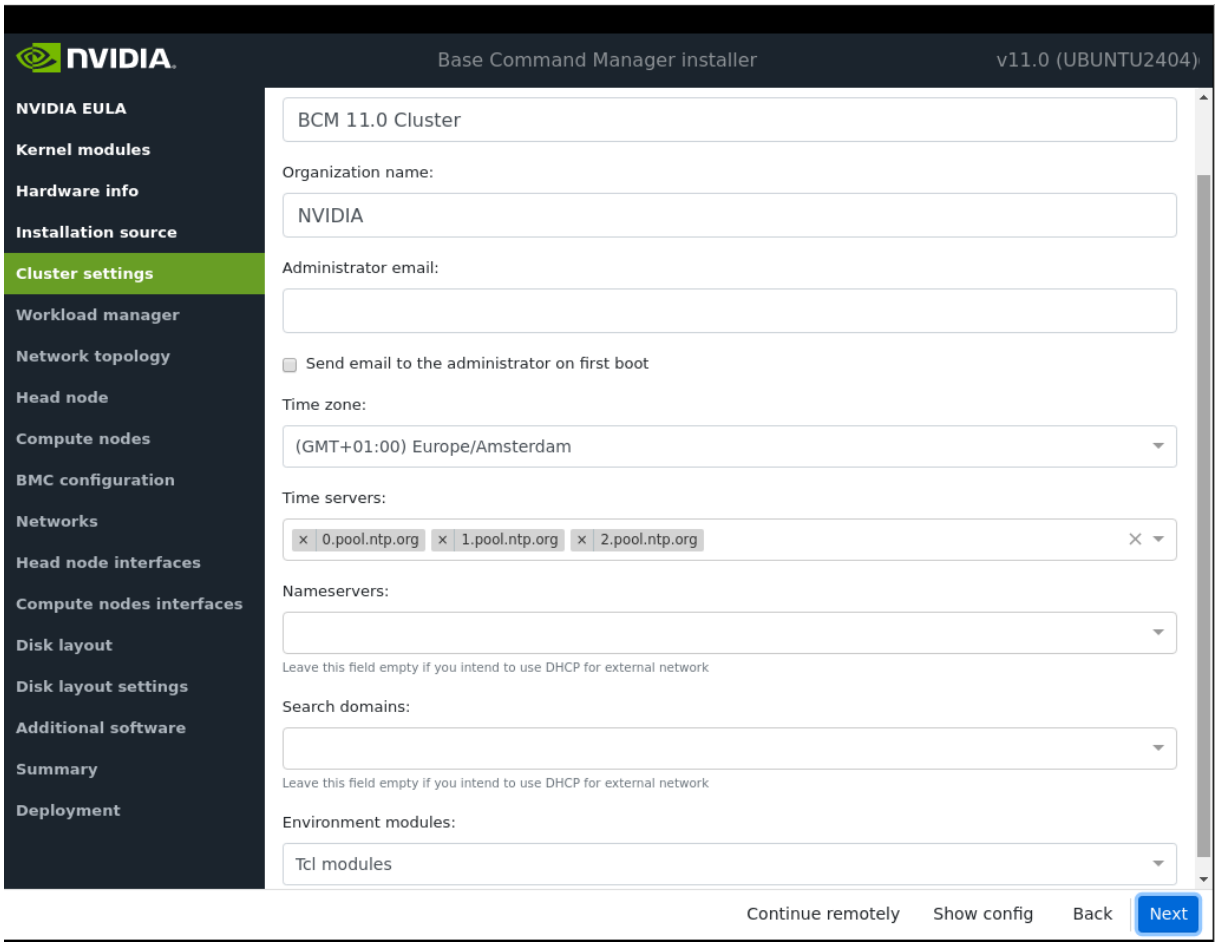

Figure 3.8: General cluster settings

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 3.9) allows a supported workload manager to be selected. A workload management system is highly recommended to run multiple compute jobs on a cluster.

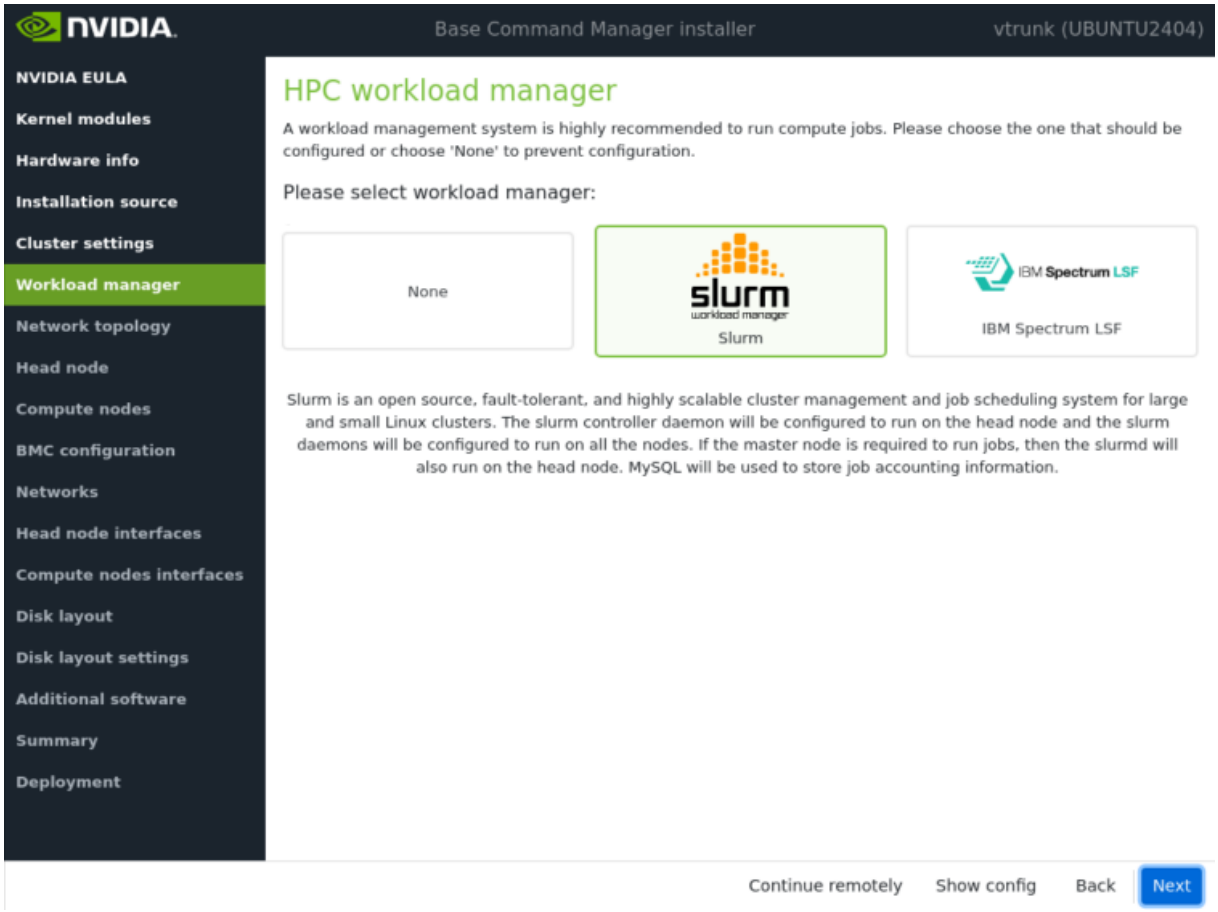

Figure 3.9: Workload Management Setup

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*.

The default client slot number that is set depends on the workload manager chosen.

- For all workload management systems, the number of client slots is determined automatically.

The head node can also be selected for running jobs, thereby acting as an additional compute node. This can be a sensible choice on small clusters if the head node can spare such resources.

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 3.10, 3.11, 3.12):

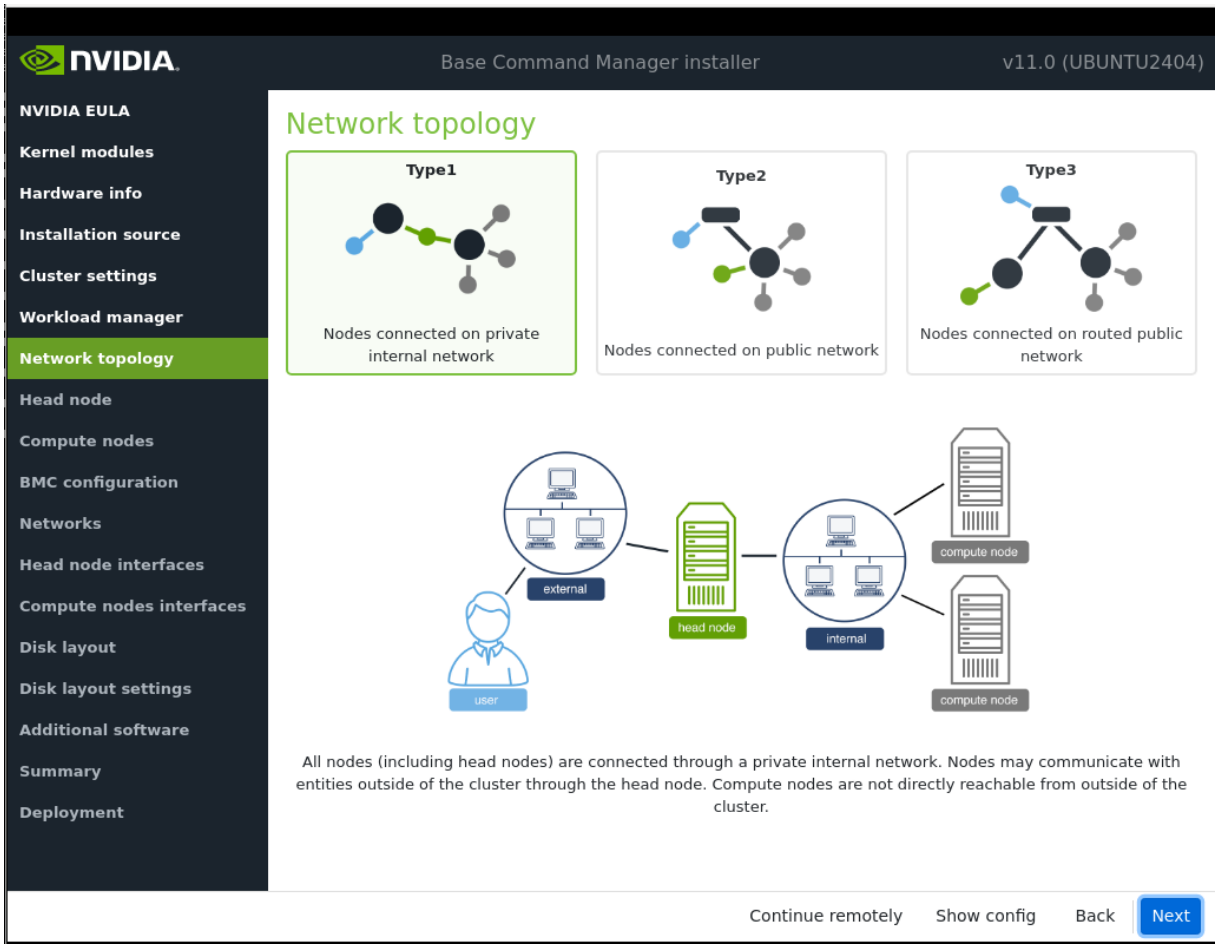

Figure 3.10: Networks Topology: Type 1 network selection

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 1 network is the most common and simple way to run the cluster. It means that the head node provides DHCP and PXE services (during pre-init stage node booting only) to a secondary, isolated network for the worker nodes, segregating the cluster traffic. The external (typically a corporate) network is then only used to provide access to the head node(s) for management.

One limitation is that broader network access must be provided through routing or via a proxy, should anyone outside of the cluster network need to access a node.

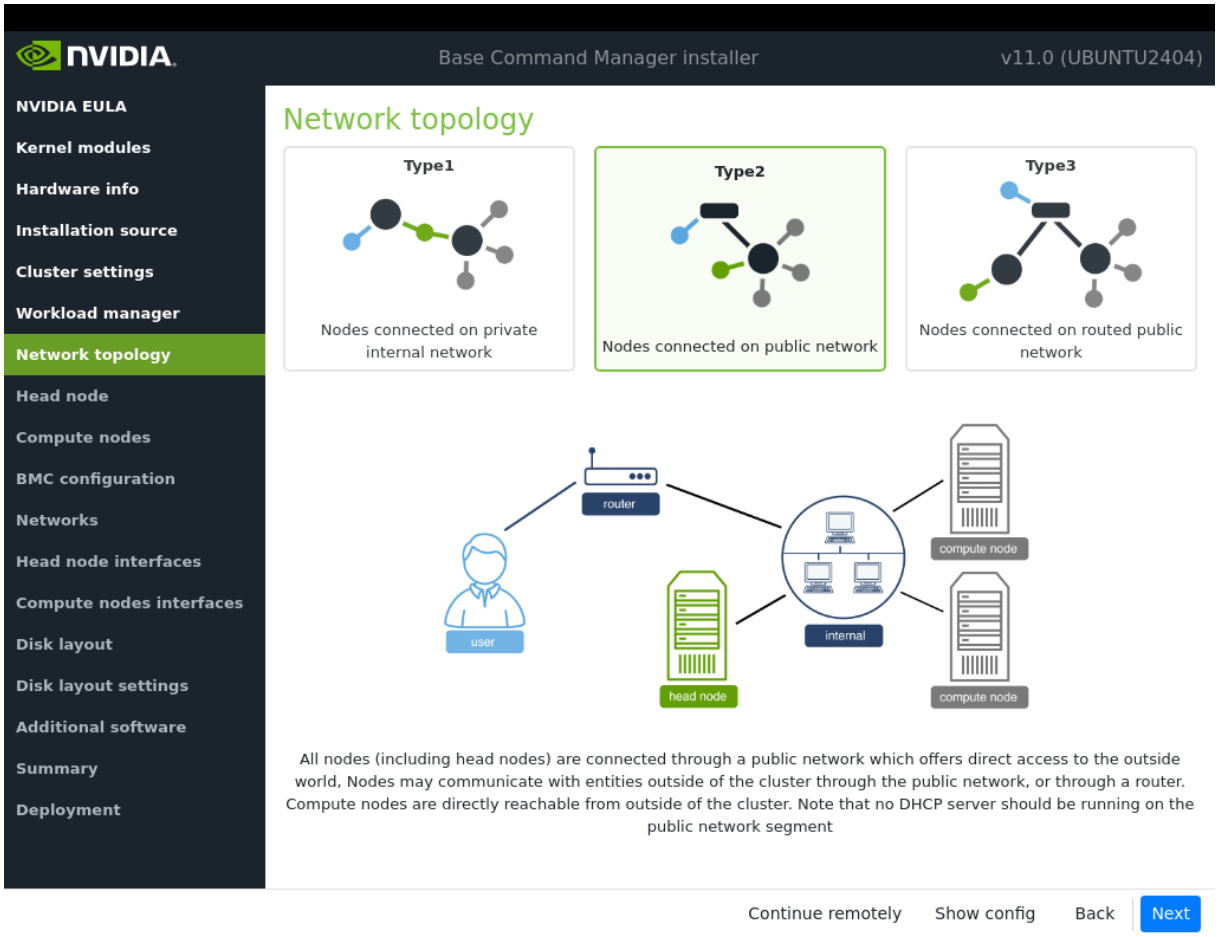

Figure 3.11: Networks Topology: Type 2 network selection

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 between the regular nodes and the head node however still normally go directly to each other, including the DHCP/PXE-related packets during pre-init stage node booting, since in a normal configuration the regular nodes and head node are on the same network.

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 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 2 network does not isolate the worker nodes from the network above it. Instead, each node remains reachable through the main data plane. This is useful for clusters hosting services, such as a web portal, avoiding the use of proxies.

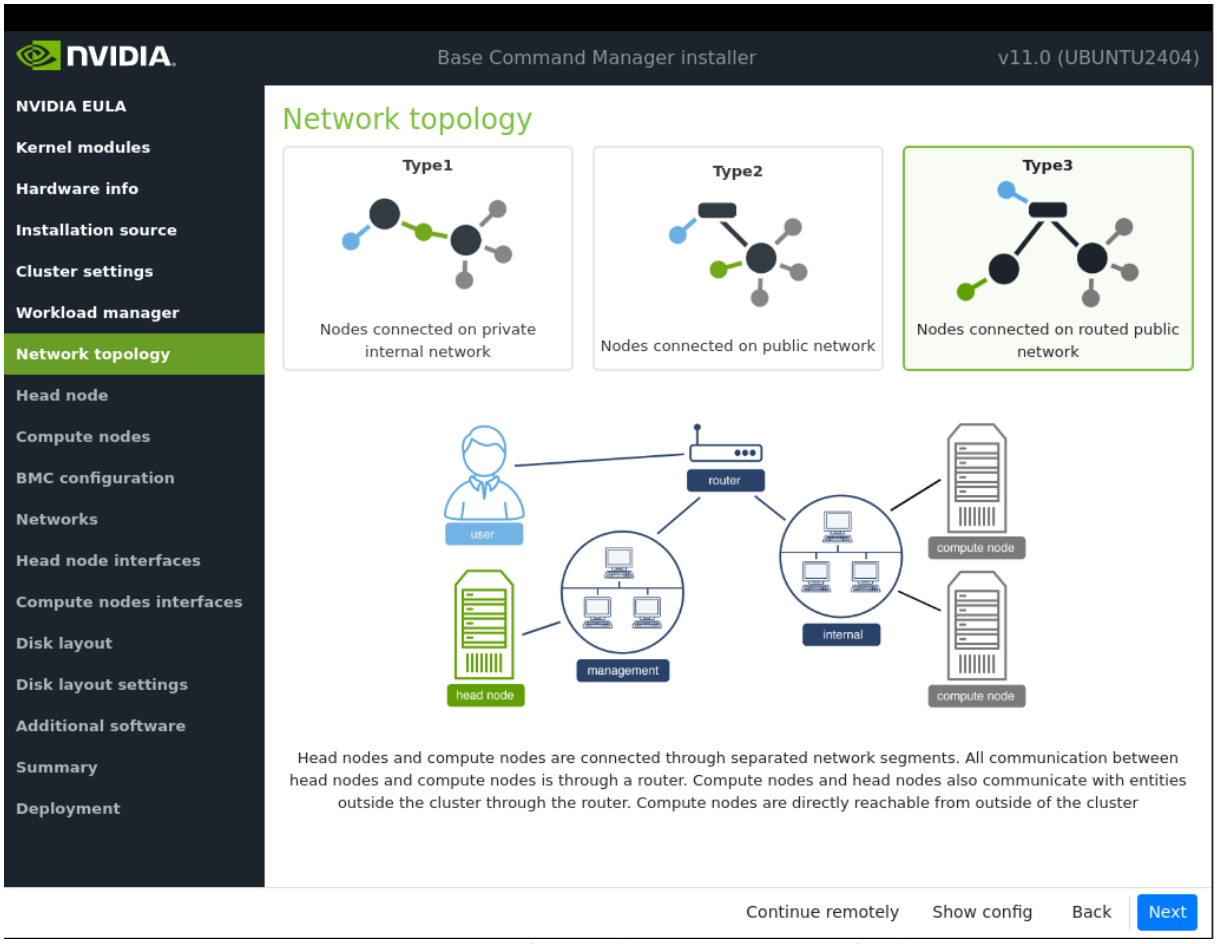

Figure 3.12: Networks Topology: Type 3 network selection

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.

A consequence of using a router in the type 3 configuration is that the communication between the head node and the regular nodes is via OSI layer 3. So, OSI layer 2 used by DHCP is not directly supported. However, DHCP/PXE packets still need to be exchanged between the head and regular nodes during pre-init node boot. The usual way to relay the packets is using a *DHCP relay agent*. Configuration of a DHCP relay agent is outside the scope of BCM configuration, and is typically done by the network administrator or the router vendor.

For a type 2 network, a DHCP relay agent may also be needed if the regular nodes are spread across several subnets.

Selecting the network topology helps decide the predefined networks on the `Networks` settings screen later (figure 3.16). Clicking `Next` here leads to the `Head node settings` screen, described next.

3.3.10 Head Node Settings

The head node settings screen (figure 3.13) can be used to set, for the head node:

- the hostname

- the password
- the hardware manufacturer

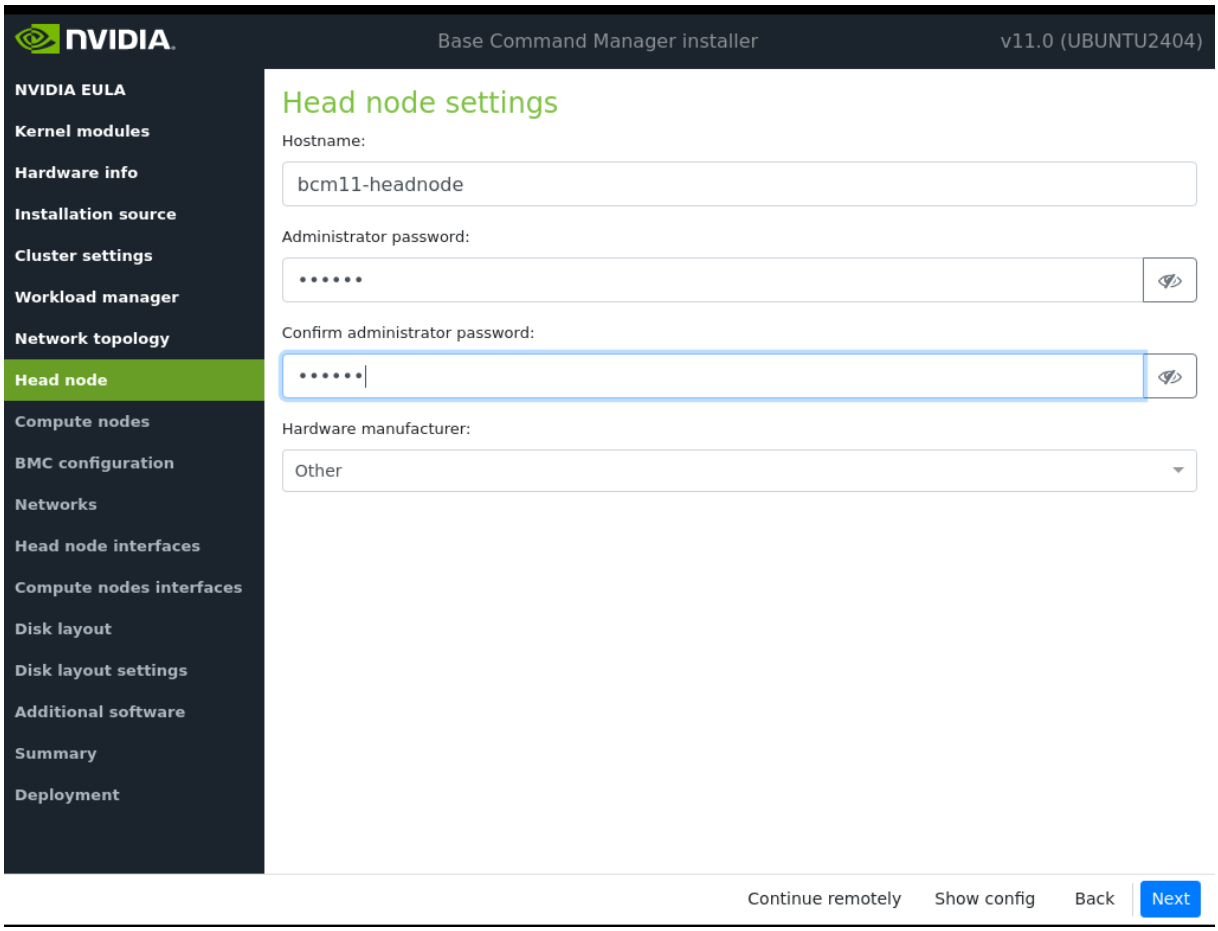

Figure 3.13: Head node settings

Clicking `Next` leads to the `Compute node settings` screen, described next.

3.3.11 Compute Nodes Settings

The compute nodes settings screen (figure 3.14) 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

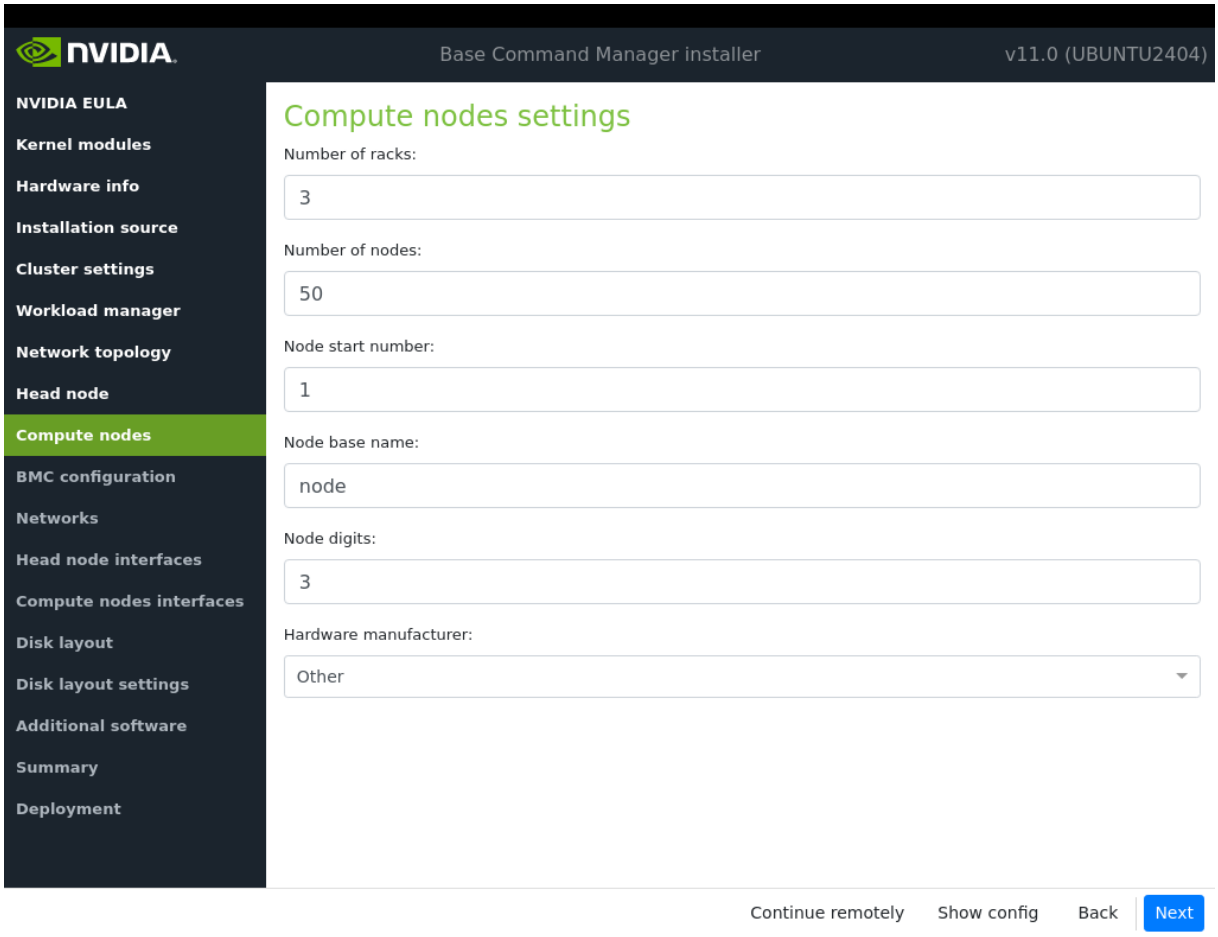

Figure 3.14: Compute node settings

Clicking `Next` leads to the `BMC configuration` screen, described next.

3.3.12 BMC Configuration

The BMC (Baseboard Management Controller) screen (figure 3.15) configures BMCs compatible with IPMI, iDRAC, iLO, or CIMC.

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

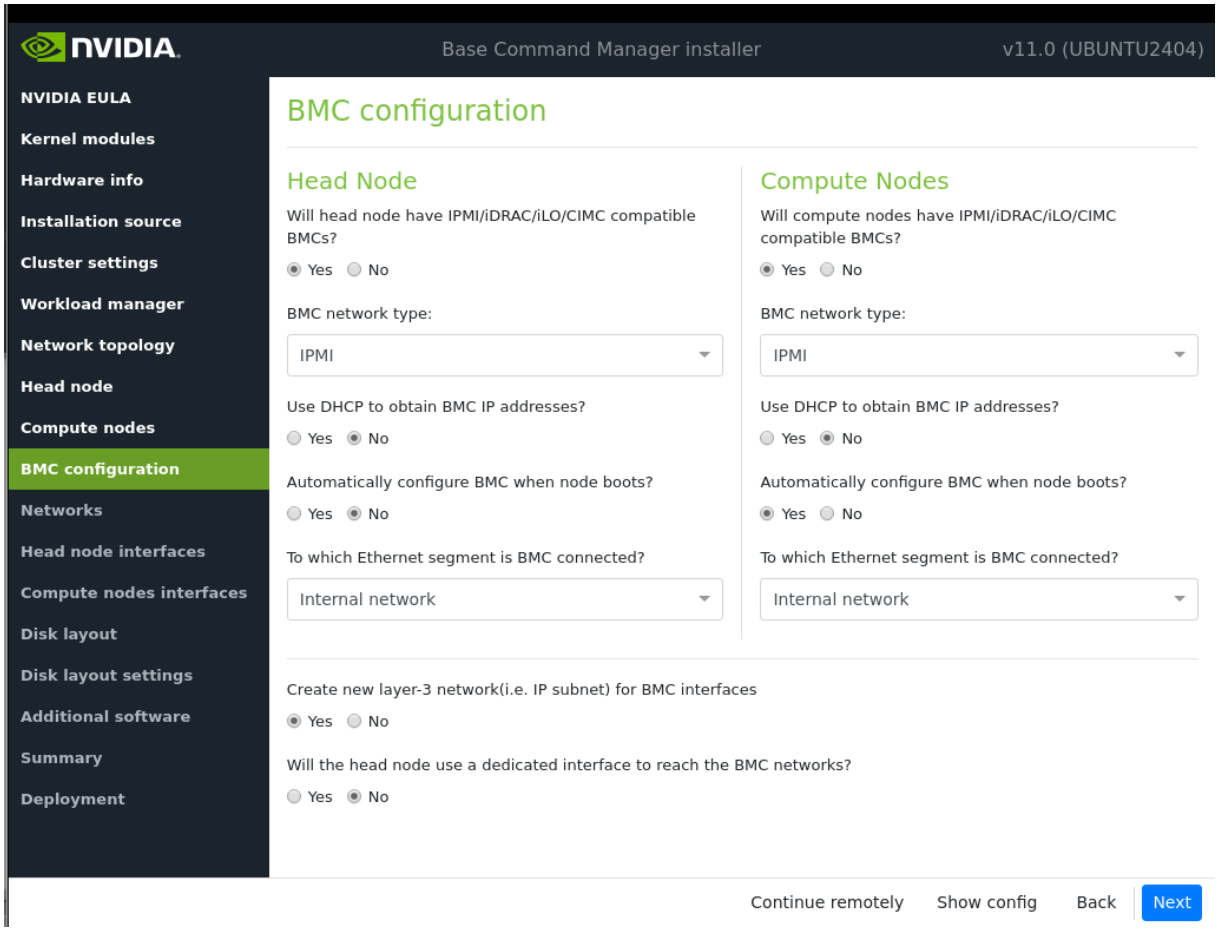

Figure 3.15: BMC configuration

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 3.16) displays the predefined list of networks, based on the selection of network topology and BMC networks made in the earlier screens.

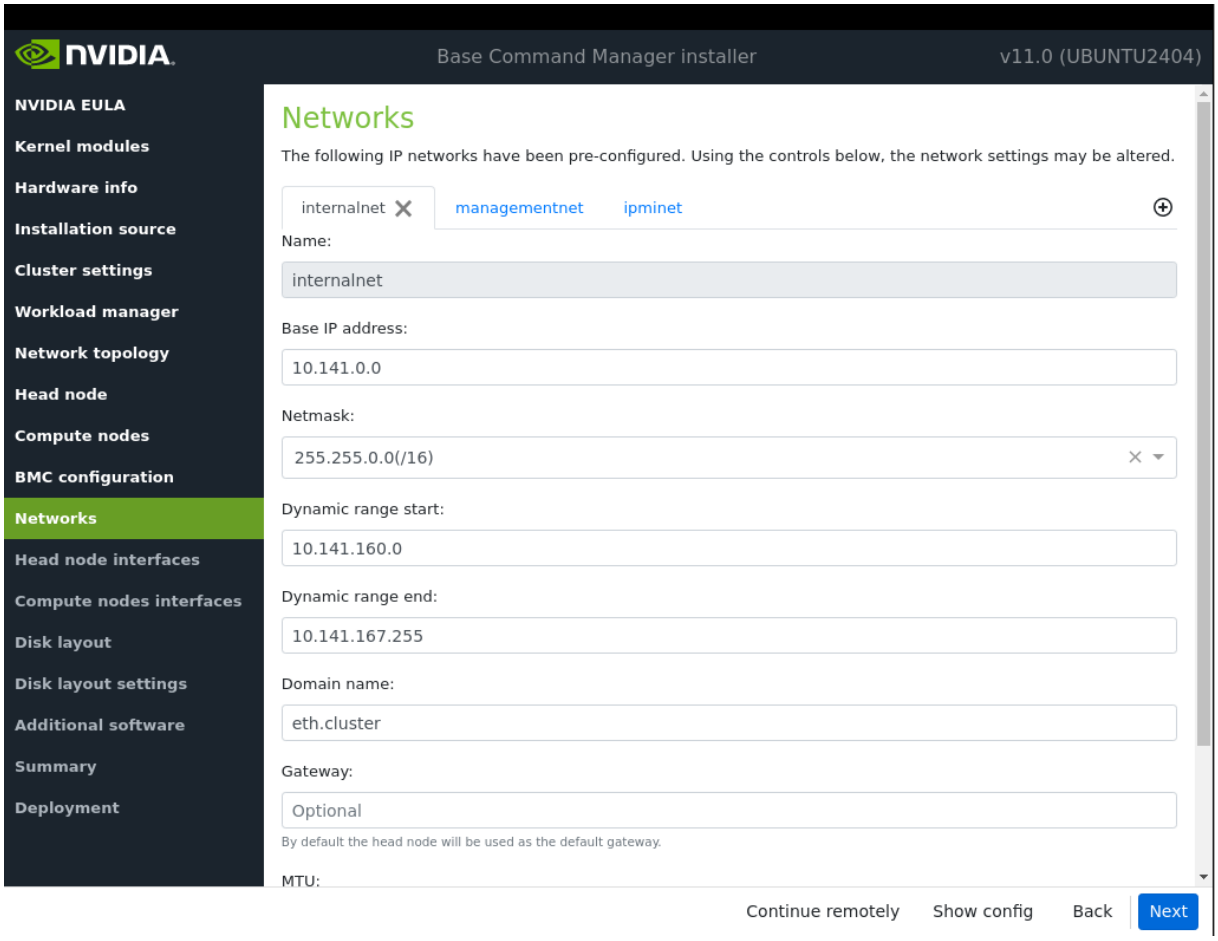

Figure 3.16: Networks configuration, `internalnet` tab

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.

3.3.14 Head node interfaces

The `Head node interfaces` screen (figure 3.17) allows a review of the head node network interfaces and their proposed settings.

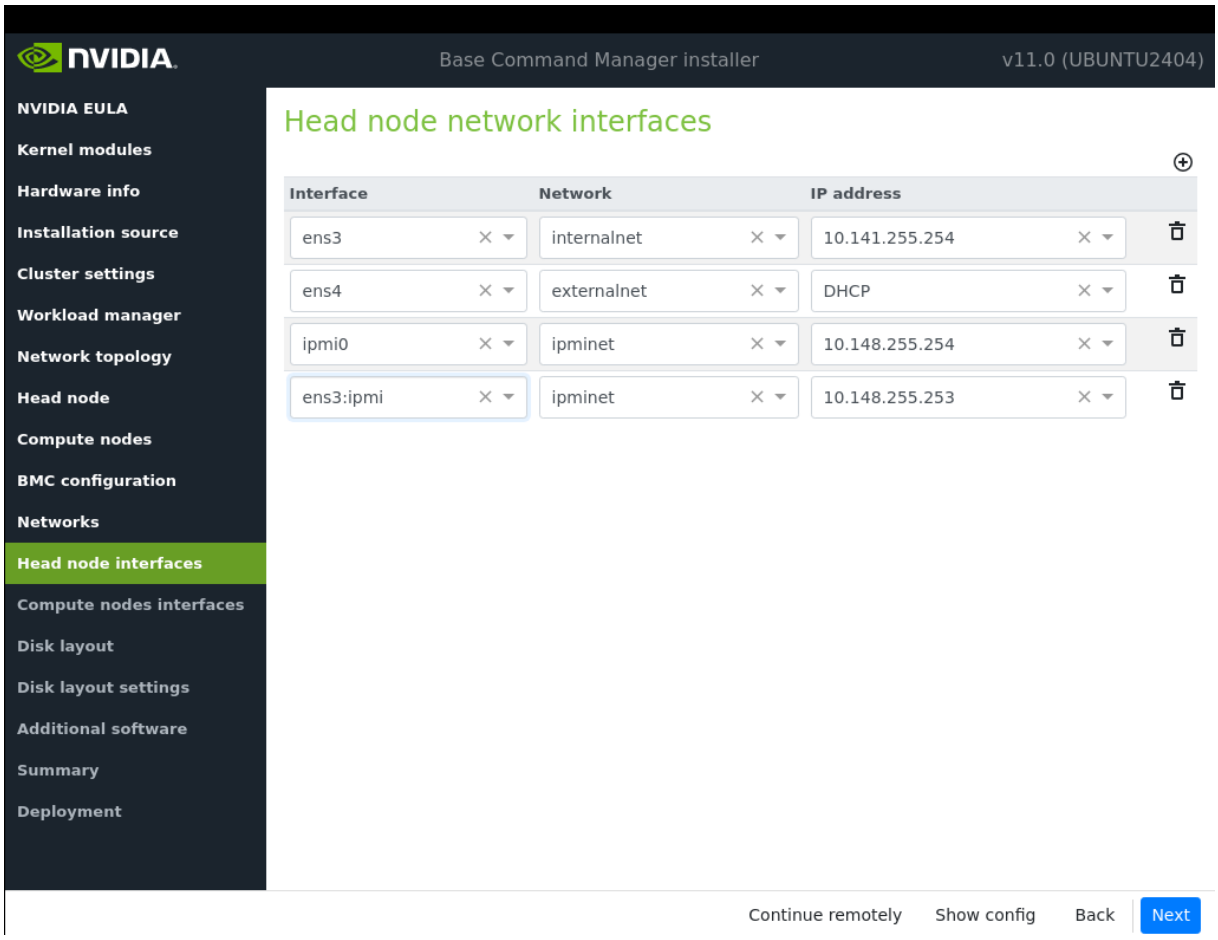

Figure 3.17: Head node interfaces configuration

If a BMC network is to be shared with a regular network, then an alias interface is shown too. In figure 3.17 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 3.18) allows a review of the compute node network interfaces and their proposed settings.

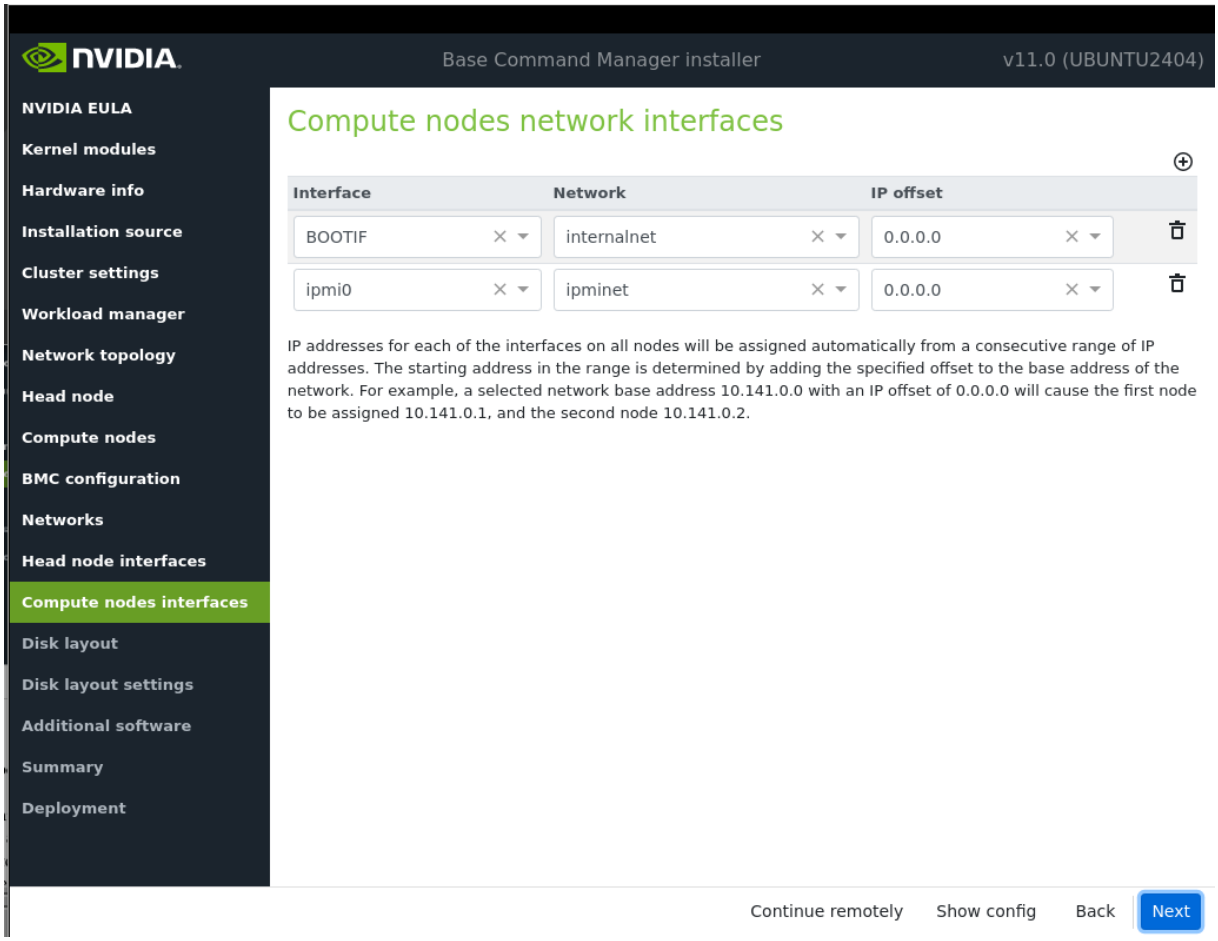

Figure 3.18: Compute node interfaces configuration

The boot interface `BOOTIF` is used to pick up the image for the node via node provisioning.

The IP offset is used to calculate the IP address assigned to a regular node interface. The nodes are conveniently numbered in a sequence, so their interfaces are typically also given a network IP address that is in a sequence on a selected network. In BCM, interfaces by default have their IP addresses assigned to them sequentially, in steps of 1, starting after the network base address.

The default IP offset is 0.0.0.0, which means that the node interfaces by default start their range at the usual default values in their network.

With a modified IP offset, the point at which addressing starts is altered. For example, a different offset might be desirable when no IPMI network has been defined, but the nodes of the cluster do have IPMI interfaces in addition to the regular network interfaces. If a modified IP offset is not set for one of the interfaces, then the `BOOTIF` and `ipmi0` interfaces get IP addresses assigned on the same network by default, which could be confusing.

However, if an offset is entered for the `ipmi0` interface, then the assigned IPMI IP addresses start from the IP address specified by the offset. That is, each modified IPMI address takes the value:

$address\ that\ would\ be\ assigned\ by\ default + IP\ offset$

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,...,10.141.0.10 while

- 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

In the `Disk layout` configuration screen (figure 3.19) the administrator must select the drive on the head node where the BCM software is to be installed.

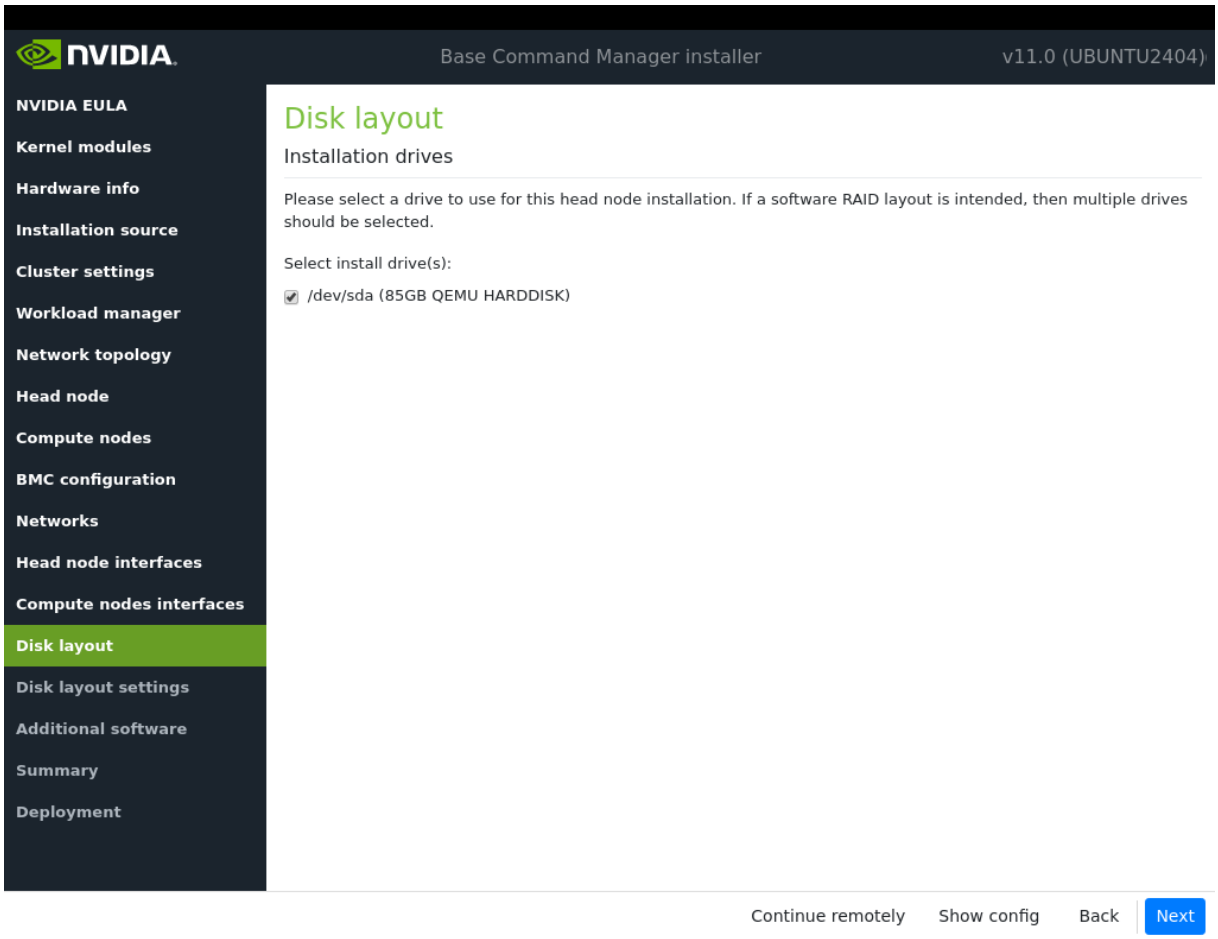

Figure 3.19: Disk layout screen

3.3.17 Disk Layout Settings

The next screen is the `Disk layout settings` configuration screen (figure 3.20).

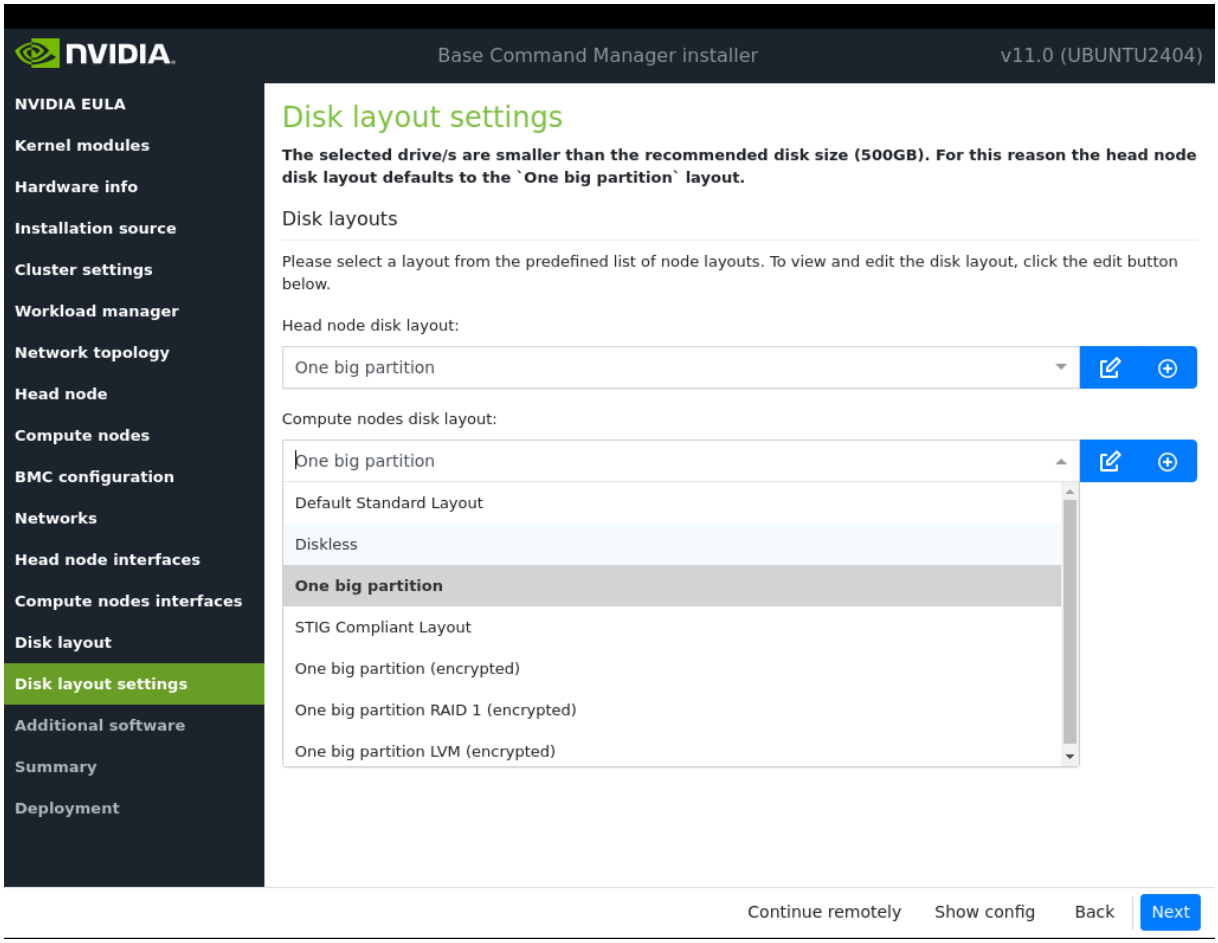

Figure 3.20: Disk layout settings screen

It is used to set the partitioning layouts. The partitioning layout XML schema is described in detail in Appendix D of the *Administrator Manual*.

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 compute node by default uses several partitions, using the `Default Standard Layout`.

  A partitioning layout other than the default can be selected for installation from the drop-down options for the head node and regular nodes. Possible partitioning options include RAID, failover, STIG-compliant, and LUKS schemes.

- The ⊕ icon can be clicked to allow a custom partitioning layout specification to be added:
  - – as a file
  - – from the default template, for use as a starting point for a specification

- Partitioning layouts 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:

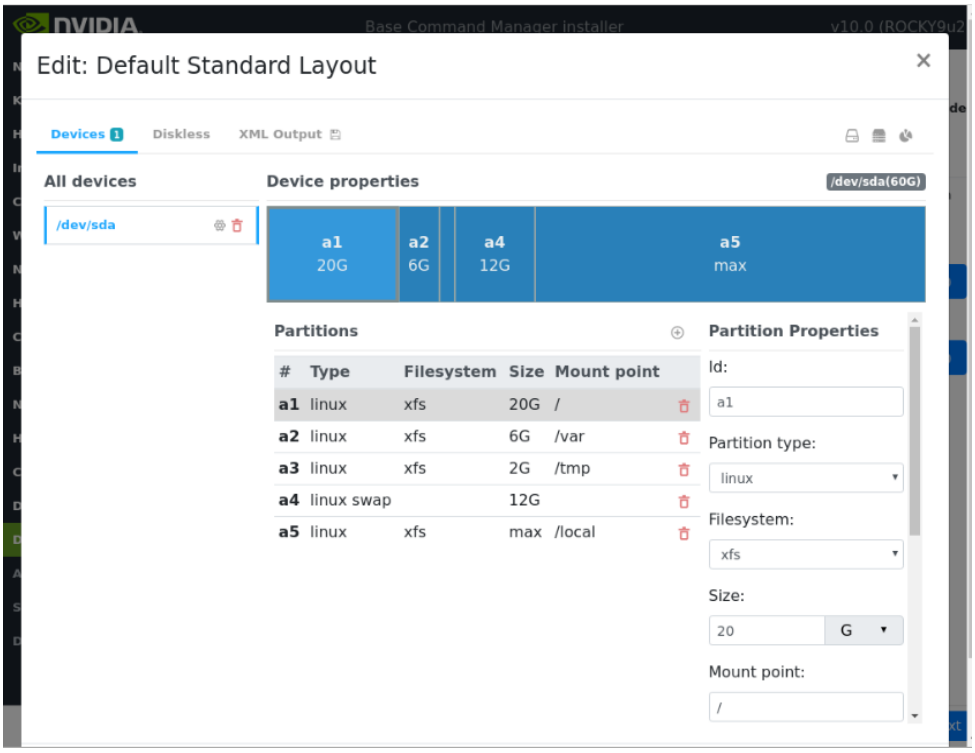

Figure 3.21: Edit Disk Partitioning

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 3.3.20) of installation. It should therefore be decided upon with care.
- By default, BCM 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></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></blockdev>` tag pair within an enclosing `<device></device>` tag pair for each drive.
3. For non-boot partitions, it is possible to set up LUKS encrypted disk partitions on head and regular nodes. Scrolling through the `Partition Properties` column of figure 3.21, and ticking the `Enable encryption` checkbox, makes the LUKS configuration parameters available (figure 3.22):

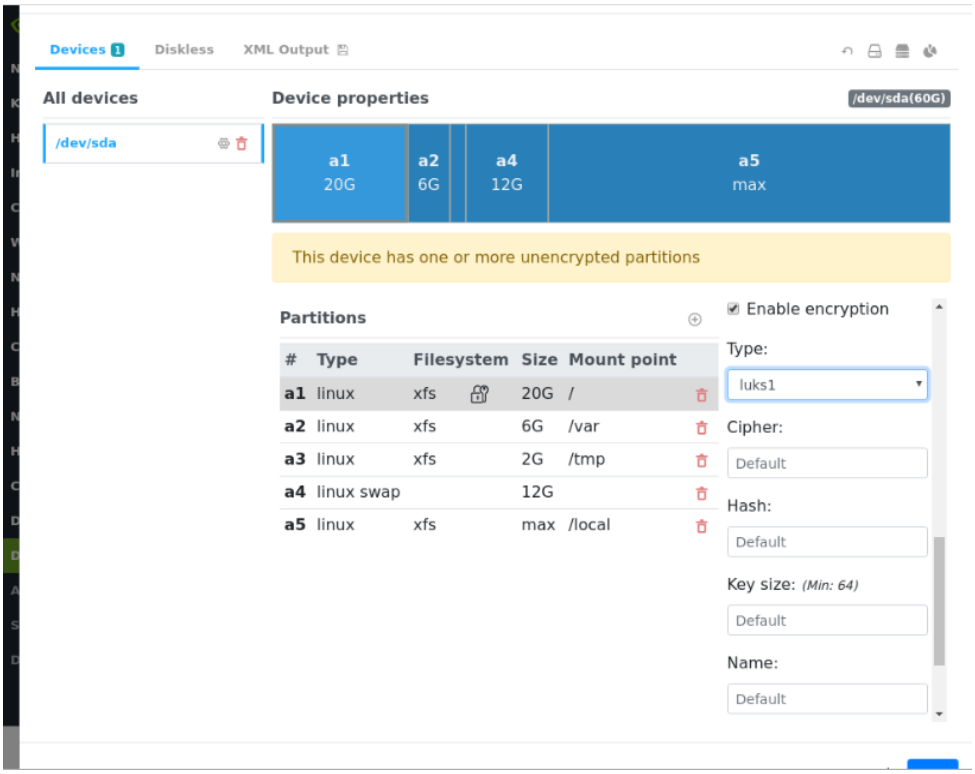

Figure 3.22: Disk Encryption Configuration

The parameters can be left at their default values to set up an encrypted partition.

If setting parameters, then there are some existing fields to set the more common parameters. Settings for less-common parameters that have no existing fields can be specified and appended to the field with the `Additional Parameters:` setting.

The settings are automatically stored in the XML specification for the disk layout and can be viewed there by selecting the `XML Output` tab.

How a cluster administrator applies this configured disk encryption to a node that is booting up is covered in Appendix D.17 of the *Administrator Manual*.

Clicking `Next` on the `Disk layout` screen leads to the `Additional software` screen, described next.

3.3.18 Additional Software

The `Additional software` screen (figure 3.23) 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. CUDA is a possible additional software.

If NVIDIA AI Enterprise is to be used, then these options must not be selected.

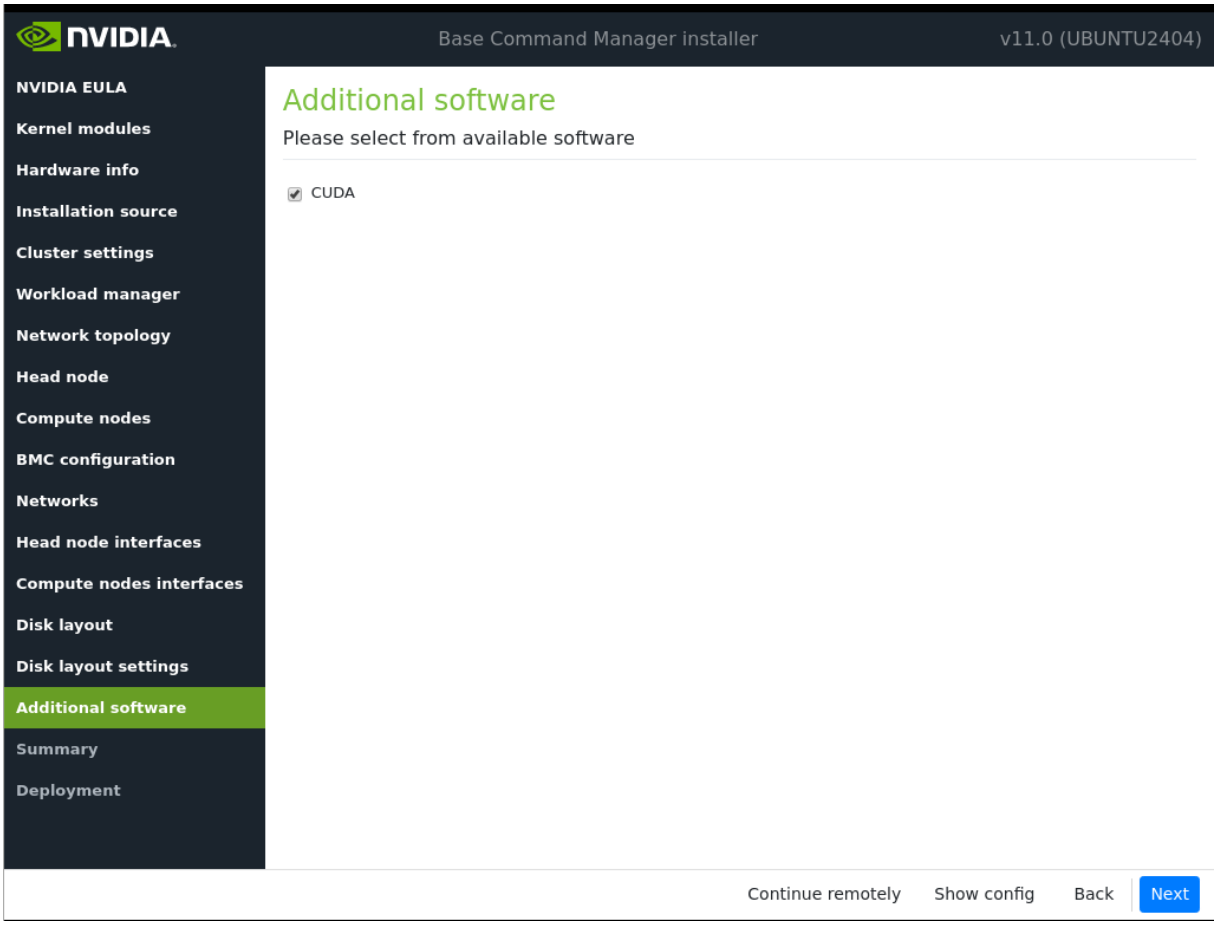

Figure 3.23: Additional software screen

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

3.3.19 Summary

The `Summary` screen (figure 3.24), summarizes some of the installation settings and parameters configured during the previous stages.

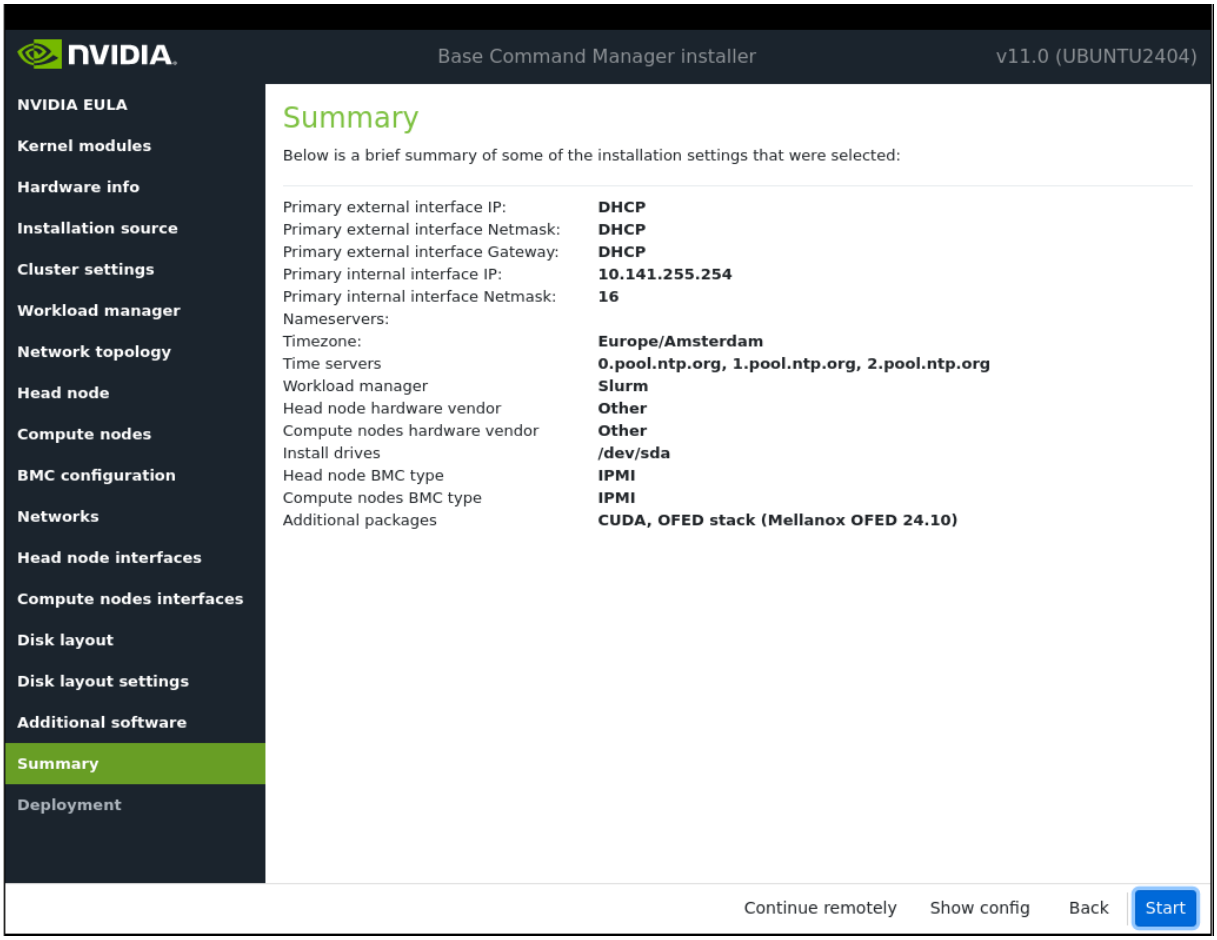

Figure 3.24: Summary of Installation Settings

Going back to correct values is still possible at this stage.

Clicking on the `Next` button leads to the "`Deployment`" screen, described next.

3.3.20 Deployment

The `Deployment` screen (figure 3.25) 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 BCM has been installed.

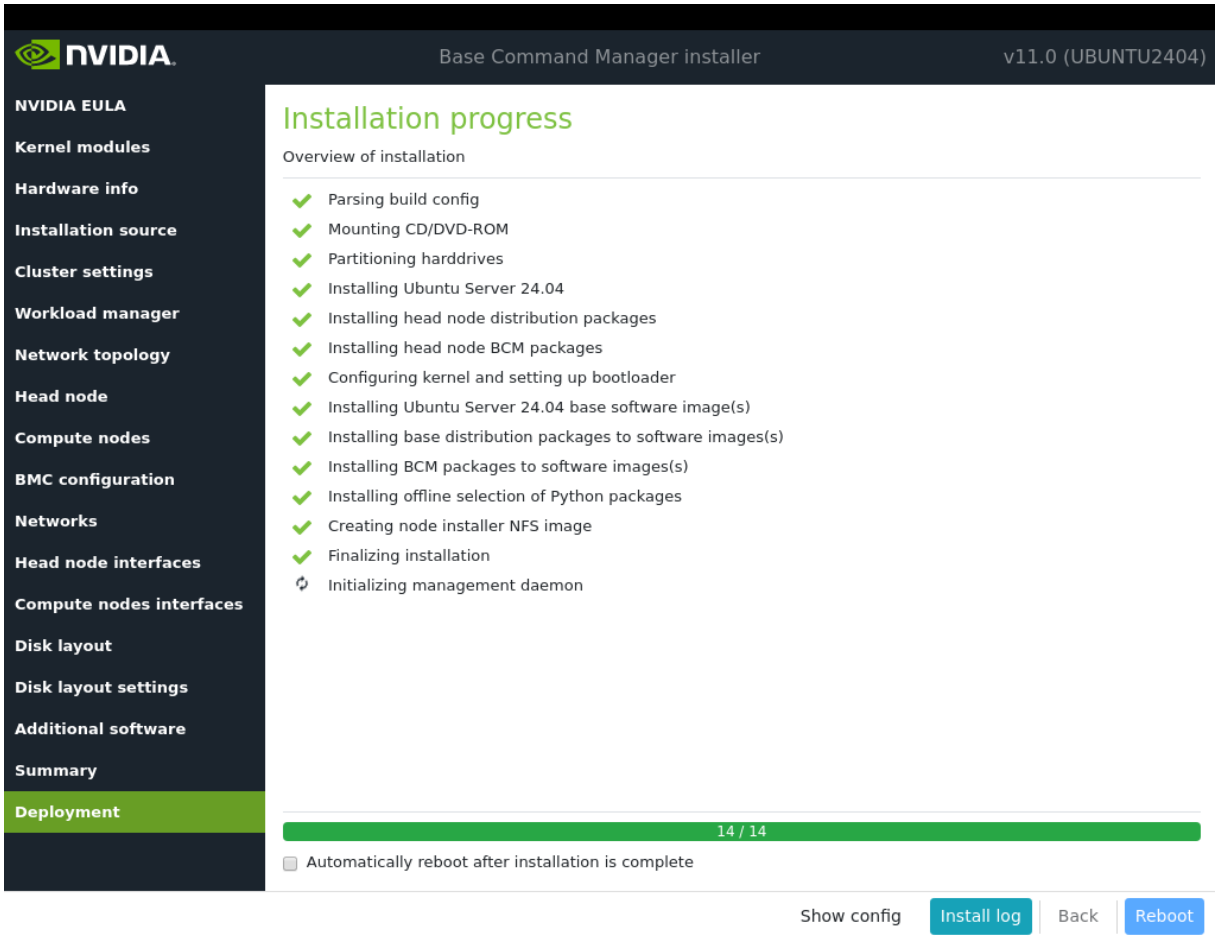

Figure 3.25: Deployment completed

After rebooting, the system starts and presents a login prompt. The cluster administrator can log in as `root` using the password that was set during the installation procedure.

The cluster should then be updated with the latest packages (Chapter 9 of the *Administrator Manual*).

After the latest updates have been installed, the system is ready to be configured.

3.3.21 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 BCM are introduced, so that further cluster configuration can be carried out.

3.4 Head Node Installation: Ansible 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. 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 BCM should be prepared for troubleshooting.
- Additional repositories typically need to be added

- Dependency and deprecated issues may need workarounds.
- Resolving an issue due to a custom configuration in the OS distribution image is outside the scope of standard support.

Using the head node installer Ansible collection is the recommended method for carrying out add-on installations.

Aside: Ansible can also be used with BCM once NVIDIA Base Command Manager is installed. This integration is described in section 14.10 of the *Administrator Manual*.

3.4.1 An Overview Of Ansible

Ansible is a popular automated configuration management software.

The BCM administrator is expected to have some experience already with Ansible. The basic concepts are covered in the official Ansible documentation at `https://docs.ansible.com/ansible/latest/user_guide/basic_concepts.html`, and further details are accessible from that site too.

As a reminder:

- Ansible is designed to administer groups of machines from an *inventory* of machines.
- An Ansible *module* is code, usually in Python, that is executed by Ansible to carry out Ansible *tasks*, usually on a remote node. The module returns values.
- An Ansible *playbook* is a YAML file. The file declares a configuration that is to be executed ("the playbook is followed") on selected machines. The execution is usually carried out over SSH, by placing modules on the remote machine.
- Traditionally, official Ansible content was obtained as a part of milestone releases of Ansible Engine, (the Red Hat version of Ansible for the enterprise).
- Since Ansible version 2.10, the official way to distribute content is via Ansible content *collections*. Collections are composed of Ansible playbooks, modules, module utilities and plugins. The collection is a formatted set of tools used to achieve automation with Ansible.
- The official Ansible list of collections is at `https://docs.ansible.com/ansible/latest/collections/index.html#list-of-collections`. At the time of writing of this section (March 2022) there were 100 collections.
- Community-supported collections are also available, at `galaxy.ansible.com`.

3.4.2 The Head Node Add-on Installer And Ansible

The head node installer is shipped as an Ansible collection. If the correct parameters are defined in the user's playbooks and roles, then the collection defines `brightcomputing.installer.head_node`. The `brightcomputing.installer.head_node` deploys the BCM head node onto the supported distribution,

The head node installation also includes deployment of the default software image and node-installer image components, which is required for provisioning compute nodes. Using the head node installer collection requires practical knowledge of Ansible and BCM. Add-on deployment is supported on both bare-metal and public cloud (AWS and Azure).

Locations For The Head Node Add-on Installer

- `https://galaxy.ansible.com/ui/repo/published/brightcomputing/installer110/` is the official Ansible Galaxy location for detailed documentation about the head node installer collection and its usage.
- `https://github.com/Bright-Computing/bright-installer-ansible/tree/main/playbooks` contains additional documentation and example playbooks.

3.4.3 Deploying A RHEL Head Node From A Rocky BCM ISO

NVIDIA Base Command Manager ships a BCM ISO for each supported distribution, including Rocky, RHEL, AlmaLinux, Ubuntu, and SLES. The RHEL ISO cannot be redistributed to customers under Red Hat's licensing terms, so a head node that must run RHEL is instead deployed from the NVIDIA Base Command Manager Rocky ISO. Rocky is binary-compatible with RHEL and freely redistributable, which makes it a suitable BCM package source for a RHEL host.

A complication is that a BCM ISO contains two distinct sets of packages:

- BCM's own packages, which install on any EL-family distribution, including RHEL.
- A bundled copy of the OS packages the ISO was built against—on the Rocky ISO, these are Rocky's, not RHEL's.

Serving that second set to a RHEL host causes version conflicts (for example, a Rocky `glibc` against the host's RHEL `glibc`). The deployment therefore uses the ISO only for BCM's packages, disables the ISO's OS repositories, and obtains the RHEL distribution packages from a separate RHEL source, such as an internal mirror or the host's own subscription.

In practice this is four steps:

1. The head node installer Ansible role is pointed at the Rocky BCM ISO (`install_medium: dvd`).
2. The ISO's OS repositories are disabled so Rocky RPMs are not layered onto RHEL.
3. The head node's RHEL packages are provided from a separate source—an internal mirror or the host's subscription.
4. The base distribution for the compute-node software image is chosen—the Rocky image from the ISO (the default), or a custom RHEL image built from a separate tarball.

Compute nodes boot from a software image, so a normal cluster deployment builds one; the decision here is which distribution that image is based on. The two recipes that follow cover both options, and the per-setting notes after them explain each line.

Recipe 1—The Rocky Software Image Is Kept (Simplest)

The head node runs RHEL; compute nodes get the Rocky image that ships on the ISO.

Example

```
# host_vars/hn-rhel9u6
install_medium: dvd
install_medium_dvd_path: /root/bcm-11-rocky9.6.iso

# Don't let the Rocky ISO enable its OS repos on top of RHEL:
bright_iso_enable_dist_repos: false
bright_iso_enable_installer_repos: false

# Give the head node its RHEL packages from a separate mirror:
pre_install_repo_files:
  - repos/RedHat_9.6-dist.repo

# post_install_default_image_archive left UNSET -> the role uses the
# Rocky archive shipped on the ISO for the compute-node image.
```

Recipe 2—A Custom RHEL Software Image Is Built

Same as Recipe 1, but with a pre-built RHEL root-filesystem tarball supplied, and `cm-create-image` told to build against the RHEL repository.

Example

```
# host_vars/hn-rhel9u6
install_medium: dvd
install_medium_dvd_path: /root/bcm-11-rocky9.6.iso

bright_iso_enable_dist_repos: false
bright_iso_enable_installer_repos: false

pre_install_repo_files:
  - repos/RedHat_9.6-dist.repo

# 1) fetch a pre-built RHEL rootfs tarball onto the head node...
pre_install_files:
  - url:  https://mirror.internal/images/RHEL9u6.tar.gz
    dest: /root/RHEL9u6.tar.gz

# 2) ...and use it as the default-image / node-installer source:
post_install_default_image_archive: /root/RHEL9u6.tar.gz

# 3) build the image against the RHEL repo, and skip the Rocky DVD repos:
cm_create_image_extra_args: >-
  --basedistrepo /etc/yum.repos.d/RedHat_9.6-dist.repo
  --disable-dvd-repos bright-dvd-distextra,bright-dvd-repo-cuda,bright-dvd-repo-bcm-extra

exclude_software_images_distro_packages:
  - redhat-indexhtml
  - hponcfg
```

What Each Setting Does

`bright_iso_enable_dist_repos` **and** `bright_iso_enable_installer_repos`**:** the ISO writes repository files for its own (Rocky) packages. These flags keep them disabled so `dnf` does not pull Rocky RPMs onto the RHEL host. Defaults are `true` and `false` respectively. On a RHEL host, both should be set to `false`.

`pre_install_repo_files`**:** a convenience for putting a RHEL package source in place using `.repo` files copied from the Ansible controller. The files go into the head node's `/etc/yum.repos.d/` before BCM repositories are set up. The head node needs RHEL packages reachable by that point. How they get there is up to the administrator—a subscription already registered on the host, repositories configured by other means, or this setting. When used, the entries are paths to `.repo` files, resolved from the playbook's `files/` directory.

`pre_install_files`**:** files staged on the head node before installation. `url:` is used to download a file on the target, or `src:` to upload one from the controller. In Recipe 2 this fetches the RHEL image tarball.

`post_install_default_image_archive`**:** the root filesystem tarball used to build the compute-node default image and the node-installer image. If it is left unset, the Rocky archive from the ISO is used. If it is set to a RHEL tarball, a RHEL compute-node image is built instead.

`cm_create_image_extra_args`: extra flags appended to `cm-create-image`. `--basedistrepo` points it at the RHEL repository. `--disable-dvd-repos` drops the Rocky DVD repositories from the image build.

Airgapped Versus Online—The Only Difference

Deploying airgapped only changes where the RHEL packages come from:

- **Airgapped**—an internal-mirror `.repo` file is copied in via `pre_install_repo_files`, and `--basedistrepo` points at it.
- **Online (subscribed host)**—`pre_install_repo_files` is omitted, and `--basedistrepo` points at the host's own subscription repository (`/etc/yum.repos.d/redhat.repo`) instead.

Known Limitations

- The ISO's distribution repositories should be kept disabled on a RHEL host. The ISO serves Rocky RPMs. Enabling them alongside RHEL causes hard package conflicts (for example, a Rocky 9.6 `glibc` against a RHEL 9.8 `glibc`). `bright_iso_enable_dist_repos: false` should be set, and RHEL packages provided from a RHEL mirror or subscription instead.
- The Rocky ISO carries a Rocky software-image archive, so a RHEL software image requires a separately supplied, pre-built RHEL tarball.
- The EL minor version of the Rocky ISO should be matched to the RHEL host—for example, a RHEL 9.6 host needs a `rocky9.6` BCM ISO.
- The `--disable-dvd-repos` argument to `cm-create-image` requires a patched `cm-create-image` that has not yet been released. Until that change ships, it should be omitted from `cm_create_image_extra_args`, relying on `bright_iso_enable_dist_repos: false` alone.

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. A text mode installer is presented as an alternative to the GUI installer (figure 3.1).

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 3.26):

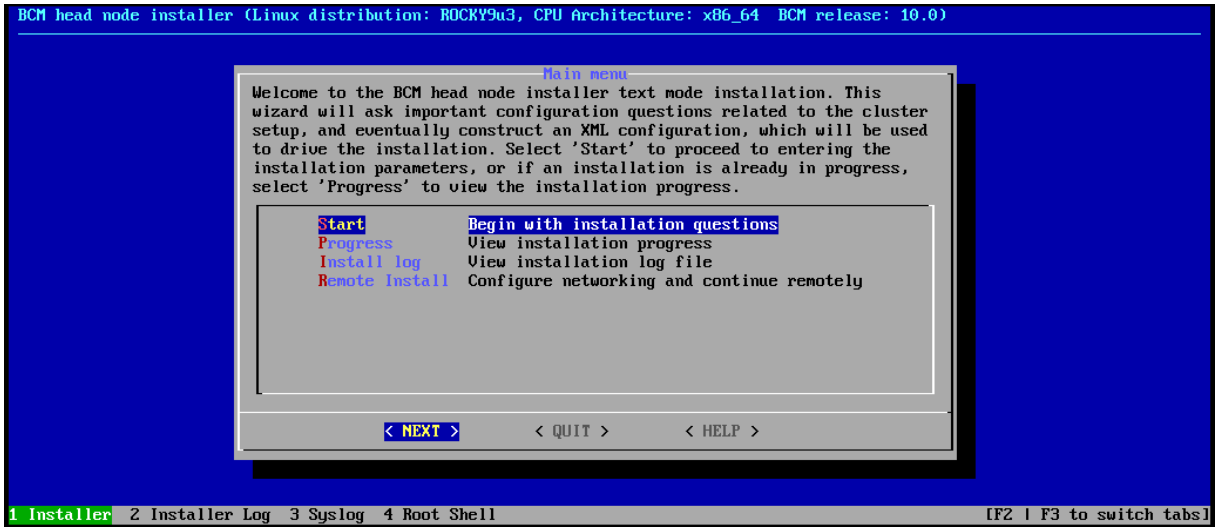

Figure 3.26: ncurses Remote Installation Option

This then sets up network connectivity, and provides the cluster administrator with a remote URL (figure 3.27):

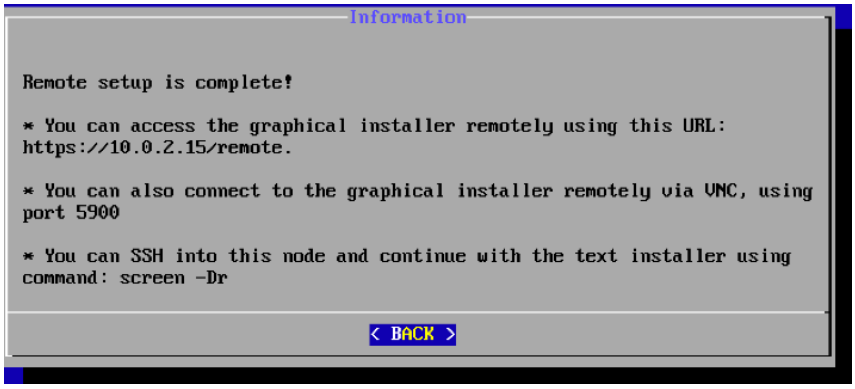

Figure 3.27: ncurses Remote Installation Remote URL Displayed

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.

4

Licensing NVIDIA Base Command Manager

This chapter explains how an NVIDIA Base Command Manager license is viewed, verified, requested, and installed. The use of a product key to activate the license is also explained.

Typically, for a new cluster that is purchased from a reseller, the cluster has BCM already set up on it.

- BCM 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 or GPUs in the cluster.
  - The default evaluation license comes with the online ISO download. The ISO is available for product key owners via `http://customer.brightcomputing.com/Download`. The license that is shipped with the ISO allows up to two nodes to be used. It can be used for a head node and up to one compute node, or it can be used with two head nodes. Clusters that run with this license do not qualify for support beyond the basic installation of Chapter 3. The idea behind this license is that the installed BCM software then allows installation of an evaluation license with greater privileges, or another full license.
  - A custom evaluation license can be set up by the NVIDIA sales team and configured for an agreed-upon number of GPUs and an agreed-upon validity period, typically from 1 to 3 months.
- In contrast to an evaluation license, there is the *full license*. A full license is almost always a subscription license. Installing a full license allows BCM to function without the restrictions of an 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 BCM 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 BCM, 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 BCM installation requires a *license file* to be present on the head node. The license file details the attributes under which a particular BCM 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 tokens`" attribute specifies the maximum number of GPUs that the cluster manager may manage. This is also the maximum number of nodes that the license covers. 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 BCM.

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 Base View

Base View is typically accessed using a "home" URL in the form of:
`https://<`*head node address*`>:8081/base-view/`

To verify that the attributes of the license have been assigned the correct values, the license details can be displayed with the navigation path `Cluster` > `License Information` (figure 4.1).

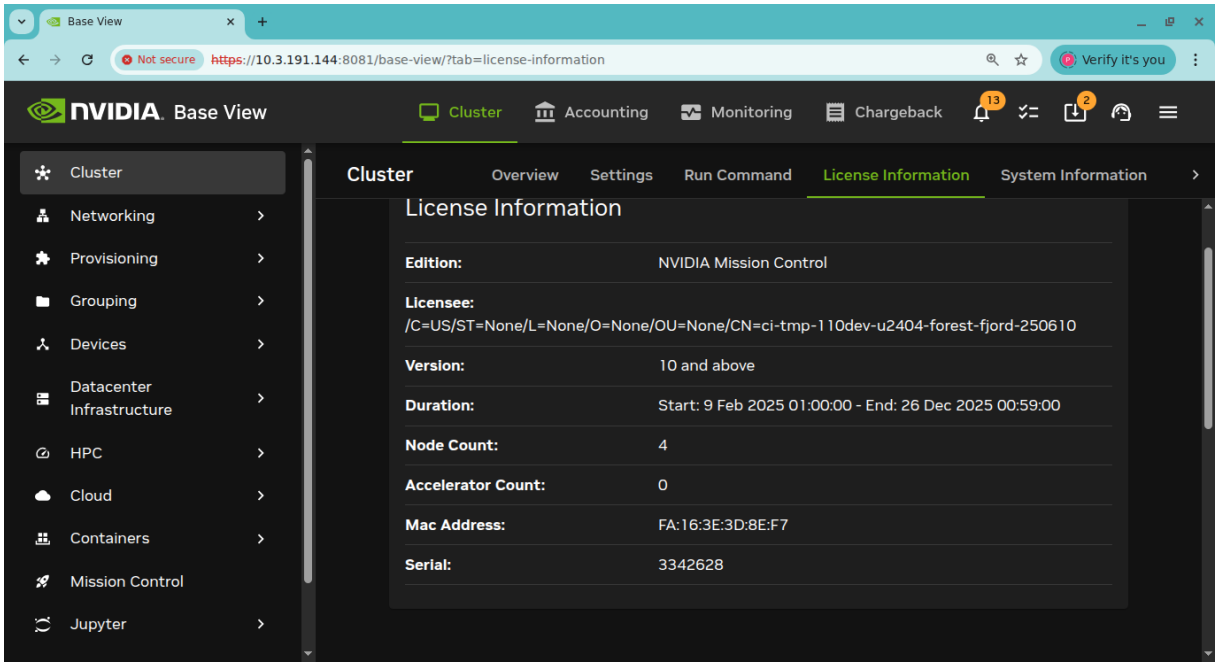

Figure 4.1: License information

4.1.2 Displaying License Attributes Within cmsh

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

Example

```
[root@basecm11 ~]# cmsh
[basecm11]% main licenseinfo

License Information
---------------------------------- -----------------------------------------------------
Licensee                           /C=US/ST=California/L=San Jose/O=NVIDIA Corporation...
Serial number                      2913820
Start time                         Sat Feb  8 16:00:00 2025
End time                           Sun May 25 16:59:00 2025
Version                            10 and above
Edition                            NVIDIA Mission Control
Type                               Free
Licensed nodes                     56 / 8192
Licensed accelerators              144 / 8192
Accelerators details               36 * 4 gpu
Allow edge sites                   Yes
MAC address / Cloud ID             C2:5F:B9:48:A2:3C|7C:C2:A8:6B:9B:EA
[basecm11]%
```

The license shown in the preceding example allows 8192 nodes and 8192 GPUs to be used.

The license is tied to a specific MAC address, or to 2 specific MAC addresses in the case of high-availability, so that it cannot simply be used elsewhere. For convenience, the Used tokens 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 cluster administrator can run the `verify-license` utility to 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@basecm11 ~]# service cmd start
Waiting for CMDaemon to start...
CMDaemon failed to start please see log file.
[root@basecm11 ~]# tail -1 /var/log/cmdaemon
Dec 30 15:57:02 basecm11 CMDaemon: Fatal: License has expired
```

but further information cannot be obtained using Base 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@basecm11 ~]# 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@basecm11 ~]# verify-license info
========= Certificate Information ========
Version:                    10
Edition:                    NVIDIA Mission Control
OEM:                        NVIDIA
Common name:                basecm11
Organization:               My organization
Organizational unit:        None
Locality:                   None
State:                      None
Country:                    US
Serial:                     3174979
Starting date:              09/Feb/2025
Expiration date:            25/Sep/2025
MAC address / Cloud ID:     FA:16:3E:73:6E:5E
Licensed tokens:            8192
Accounting & Reporting:     Yes
Allow edge sites:           Yes
License type:               Free
==========================================
```

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@basecm11 ~]# verify-license verify
    License has expired
    License verification failed.
    ```

  - If the certificate is not from Bright Computing:

    Example

    ```
    [root@basecm11 ~]# verify-license verify
    Invalid license: This certificate was not signed by a correct signing authority
    License verification failed.
    ```

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.

A license, with just over 3 months validity left, is examined in the following example:

Example

```
[root@basecm11 etc]# date
Tue Jun 24 12:49:41 PM CEST 2025
[root@basecm11 etc]# verify-license monthsleft=3
[root@basecm11 etc]# verify-license monthsleft=4
Cluster Manager license expiration date: 25/Sep/2025, time remaining: 13w 1d
[root@basecm11 etc]# verify-license monthsleft
Cluster Manager license expiration date: 25/Sep/2025, time remaining: 13w 1d
```

4.2.3 Using The `versioninfo` Command To Verify The BCM Software Version

The license version should not be confused with the BCM software version. The license version is a license format version that rarely changes between cluster manager version releases. Thus a cluster can have a license version 10.0, which was the license format introduced during BCM 10.0, and BCM can be at software version 11.0.

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

Example

```
[root@basecm11 ~]# cmsh
[basecm11]% main
[basecm11->main]% licenseinfo | grep Version
Version                              10 and above
[basecm11->main]% versioninfo
versioninfo
Version Information
------------------------ ----------------------------------------------------
Cluster Manager          11.0
CMDaemon                 3.1
CMDaemon Build Index     163199
CMDaemon Build Hash      70ba1c343e
Database Version         36375
```

In the preceding example, the version of the cluster manager is `11.0`, while the license is for BCM 10 and above.

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 BCM 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 62) 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 GPUs or 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 BCM reseller that provided the software should be contacted with details to correct matters.

If BCM 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 98 of the *Administrator Manual*).

4.3.2 The Product Key

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

An account manager is the person from BCM 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 BCM 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 BCM technical support (section 14.2 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 BCM installed on it, then a temporary license to run the cluster can be installed with an evaluation product key. Such a key allows to 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.

Evaluation ISO downloads of BCM from the BCM website come with a built-in license that overrides any product key attributes. The license is temporary. An evaluation ISO product key allows the user to download the ISO with a temporary built-in license. The temporary 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 it can comprise 2 head nodes. A 2-node cluster is a very limited cluster, and support for this is given accordingly.

- **Subscription product key**: A subscription license is a license that 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. The length can be set to any value.

  If a cluster has BCM 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 64) with the command:

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

The product key allows: the administrator to obtain and activate a license, which allows the cluster manager to function.

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 BCM (re)seller. It is used to obtain and *activate* a license file. Activation means that NVIDIA 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 BCM to new hardware. In such a case,

the product key must be unlocked. Unlocking is possible for a subscription license via `https://customer.brightcomputing.com/unlock`.

Unlocking an evaluation license, or a hardware lifetime license, is possible by sending a request to the account manager at NVIDIA 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 64). The license can only work on the hardware it was specified for. After installation is complete, the cluster runs with the activated license.

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 62) 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:

     `export http_proxy=<proxy>`

     where <*proxy*> is the hostname or IP address of the proxy. An equivalent alternative is that the `ScriptEnvironment` directive (page 926 of the *Administrator Manual*), which is a CMDaemon directive, can be set and activated (page 905 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 NVIDIA 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:

`install-license <signedlicense>`

installs the signed license on the head node, and is described further on page 65. 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 BCM reseller. A certificate will be faxed or sent back in response, the license will be noted by NVIDIA 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@basecm11 ~]# request-license
Product Key (XXXXXX-XXXXXX-XXXXXX-XXXXXX-XXXXXX):
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): ACME Inc.
Organizational Unit Name (e.g. department): Development
Cluster Name: basecm11
Private key data saved to /cm/local/apps/cmd/etc/cluster.key.new

MAC Address of primary head node (basecm11) 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
Submit certificate request to http://licensing.brightcomputing.com/licensing/ ? [Y/n] y

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 NVIDIA 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 64).

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 4.3.7.

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@basecm11 ~]# install-license /cm/local/apps/cmd/etc/cluster.pem.new
========= Certificate Information ========
Version:             HEAD
Edition:             Advanced
Common name:         basecm11
Organization:        ACME 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
==========================================

Is the license information correct ? [Y/n] y

Installed new license

Restarting Cluster Manager Daemon to use new license: OK
```

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 15.4.8 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 `https://customer.brightcomputing.com/unlock`.
- 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 NVIDIA 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 NVIDIA 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`
         · `admin.pem`

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

         Example

         ```
         [root@basecm11 ~]# service cmd stop
         [root@basecm11 ~]# cmd -b
         [root@basecm11 ~]# [...]
          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@basecm11 ~]# [...]
         [root@basecm11 ~]# chmod 600 admin.*
         [root@basecm11 ~]# mv admin.* /root/.cm/cmsh/
         [root@basecm11 ~]# 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 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 for the first time, 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 Base 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, or another 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, with the same product key, or another product key, the administrator is prompted on whether to re-use the existing keys and settings from the existing license:

Example

```
[root@basecm11 ~]# 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: basecm11
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.
...
```

- 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 6.5 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 BCM 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 Base 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.

4.3.8 Getting Help With Licensing Issues

The customer portal at `https://enterprise-support.nvidia.com/s/create-case` allows support tickets to be submitted for licensing issues. Further details on this are in section 14.2.1 of the *Administrator Manual*.

5

Linux Distributions That Use Registration

This chapter describes setting up registered access for the NVIDIA Base Command 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 9.6 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@basecm11 ~]# 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.

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.

The list of enabled and disabled repositories can be seen with:

```
[root@basecm11 ~]#  yum repolist all
```

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 BCM `cm-chroot-sw-img` tool (page 536 of the *Administrator Manual*) as follows:

```
[root@basecm11 ~]# cm-chroot-sw-img /cm/images/default-image subscription-manager register --username \
<username> --password <password> --auto-attach
```

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.

The list of enabled repositories in the image can be seen with:

```
[root@basecm11 ~]# cm-chroot-sw-img /cm/images/default-image yum repolist
repo id                          repo name
AppStream                        AppStream Packages Red Hat Enterprise Linux 9
BaseOS                           BaseOS Packages Red Hat Enterprise Linux 9
...
```

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 SUSE helps with installing new SLES packages or receiving SLES package updates, as well as to carry out some other tasks.

5.2.1 Registering A Head Node With SUSE

The `suseconnect` command can be used to register a SUSE 15 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@basecm11 ~]# suseconnect -e <e-mail address> -r <registration code> -u https://scc.suse.com
```

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

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

The defined repositories can be listed with:

```
[root@basecm11 ~]# zypper lr
```

and the head node can be updated with:

```
[root@basecm11 ~]# zypper refresh
[root@basecm11 ~]# zypper update
```

5.2.2 Registering A Software Image With SUSE

The `SUSEConnect` command can be used to register a SUSE 15 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@basecm11 ~]# cm-chroot-sw-img /cm/images/default-image \
suseconnect -e <e-mail address> -r <registration code> \
-u https://scc.suse.com
```

The e-mail address is the address used to register the subscription with SUSE. When logged in on the SUSE 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@basecm11 ~]# cp /etc/zypp/repos.d/* /cm/images/default-image/etc/zypp/repos.d/
[root@basecm11 ~]# cp /etc/zypp/credentials.d/* /cm/images/default-image/etc/zypp/credentials.d/
[root@basecm11 ~]# cp /etc/zypp/service.d/* /cm/images/default-image/etc/zypp/service.d/
```

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@basecm11 ~]# zypper -R /cm/images/default-image lr
```

and the software image can be updated with:

```
[root@basecm11 ~]# export PBL_SKIP_BOOT_TEST=1
[root@basecm11 ~]# zypper -R /cm/images/default-image refresh
[root@basecm11 ~]# zypper -R /cm/images/default-image update
[root@basecm11 ~]# zypper -R /cm/images/default-image clean --all
```

6

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 NVIDIA Base Command Manager and its network configuration interface.

6.2 Method

A cluster consists of a head node, say `basecm11` 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 basecm11; 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 basecm11; set eth1 dhcp no"
# cmsh -c "device interfaces basecm11; set eth1 ip 192.168.1.176; commit"
# cmsh -c "device interfaces basecm11; 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*

Network Parameter	Description	Operation	Command Used
IP*	IP address of head node on eth1 interface	view	`cmsh -c "device interfaces basecm11; get eth1 ip"`
		set	`cmsh -c "device interfaces basecm11; set eth1 ip` *address*`; commit"`
baseaddress*	base IP address (network address) of network	view	`cmsh -c "network get externalnet baseaddress"`
		set	`cmsh -c "network; set externalnet baseaddress` *address*`; commit"`
broadcastaddress*	broadcast IP address of network	view	`cmsh -c "network get externalnet broadcastaddress"`
		set	`cmsh -c "network; set externalnet broadcastaddress` *address*`; commit"`
netmaskbits	netmask in CIDR notation (number after "/", or prefix length)	view	`cmsh -c "network get externalnet netmaskbits"`
		set	`cmsh -c "network; set externalnet netmaskbits` *bitsize*`; commit"`
gateway*	gateway IP address (default route)	view	`cmsh -c "network get externalnet gateway"`
		set	`cmsh -c "network; set externalnet gateway` *address*`; commit"`
nameservers*, **	nameserver IP addresses	view	`cmsh -c "partition get base nameservers"`
		set	`cmsh -c "partition; set base nameservers` *address*`; commit"`
searchdomains**	name of search domains	view	`cmsh -c "partition get base searchdomains"`
		set	`cmsh -c "partition; set base searchdomains` *hostname*`; commit"`
timeservers**	name of timeservers	view	`cmsh -c "partition get base timeservers"`
		set	`cmsh -c "partition; set base timeservers` *address*`; commit"`

* If *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*.

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 `sipcalc` 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):

```
# sipcalc 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:

```
# sipcalc -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
```

7

Third Party Software

In this chapter, several third party software packages included in the BCM repository are described briefly. For all packages, references to the complete documentation are provided.

The packages dashboard at `https://service.bcm.nvidia.com/packages-dashboard/` lists the available supported package versions per BCM version, architecture, and distribution.

7.1 Modules Environment

RHEL and derivatives, and SLES BCM 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/`. 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

BCM provides the Shoreline Firewall (more commonly known as "Shorewall") package from the BCM 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 under `/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 NVIDIA Base Command Manager 11, 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` submode (section 3.15 of the *Administrator Manual*) of `cmsh` on the head node. For example, on a head node `basecm11` a restart of `shorewall` can be carried out with:

```
[basecm11->device[basecm11]->services[shorewall]]% restart
restart Successfully restarted service shorewall on: basecm11
```

System administrators may sometimes need to be aware that Shorewall does not really run as a daemon process. That is, the command to restart the service does not stop and start a `shorewall` daemon. Instead the command configures netfilter using the iptables settings specified in the shorewall configuration files, and then exits. It exits without leaving a `shorewall` process up and running, even though `service shorewall status` implies that such a service 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 BCM 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 Base 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`.

A rule modification is typically needed if Shorewall denies access to a port, on the head node, that needs to be accessed from an external network. The access might be, for example, for carrying out port forwarding from the head node to a compute node when using the `portforward` command (page 120 of the *Administrator Manual*).

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.

The `cm-cmd-ports` Utility

Port 8081 is the default port that CMDaemon listens to when using the HTTPS protocol to manage the nodes. If, for example, a new software needs that port, then `cm-cmd-ports` utility can be used to set the HTTPS protocol port that CMDaemon listens on to another port, such as 8082.

Example

```
[root@basecm11 ~]# cm-cmd-ports --https 8082
```

A restart of CMDaemon has the change take effect, and takes care of opening the firewall on port 8082 for CMDaemon, by adding a line to the `rules` file of Shorewall. The original port 8081 remains open, but CMDaemon no longer listens to it.

The status of ports used by the cluster manager can be listed with:

```
[root@basecm11 ~]# cm-cmd-ports -l
type            http      https  firewall rule    path
--------------  ------  -------  ---------------  ----------------------------------------------------
image           8080       8082                   /cm/images/default-image/cm/local/apps/cmd/etc/cmd.conf
image           8080       8082  True             /cm/local/apps/cmd/etc/cmd.conf
node-installer             8082                   /cm/node-installer/scripts/node-installer.conf
```

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 BCM, 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 under services mode under `device` mode.

This situation is indicated in the following table:

Correspondence of `stop`, `clear`, *and* `restart` *in Shorewall vs* `cmsh`

iptables rules	Service	Unix Shell	`cmsh` shell
keep a safe set:	service shorewall stop	shorewall stop	*no equivalent*
clear all rules:	*no equivalent*	shorewall clear	stop shorewall
start with stored rules:	service shorewall restart	shorewall restart	restart shorewall

7.2.4 Adding To Shorewall Configuration Via A Role In `cmsh`

This section, about adding to Shorewall configuration via a role in `cmsh`, requires some familiarity with BCM and Shorewall, and can be skipped at a first reading of the manuals. For convenience, cross-references are provided within this section to the material that the administrator should have some familiarity with before configuring Shorewall via a role.

The command line front end manager to BCM is `cmsh` (section 2.5 of the *Administrator Manual*). Within `cmsh`, services are often implemented as roles (section 2.1.5 of the *Administrator Manual*).

Administrators that have become familiar with using roles within `cmsh` can carry out some Shorewall configuration within the `firewall` role of the head node, via modes (section 2.5.2 of the *Administrator Manual*) under the `firewall` role. Firewall modes were introduced in BCM version 9.1.

Shorewall configuration can go far beyond just what can be done within the modes of `firewall`. These modes are just the BCM way to add changes to some Shorewall firewall configurations, and are aimed at making firewall management for the cluster easier.

BCM provides modes for modifying Shorewall files as indicated by the following table:

mode	used for modifying:	Shorewall file man pages
policies	policies in /etc/shorewall/policy	https://shorewall.org/manpages/shorewall-policy.html
routes	routes /etc/shorewall/routes	https://shorewall.org/manpages/shorewall-routes.html (not present in Ubuntu clusters by default)
interfaces	interfaces in /etc/shorewall/interfaces	https://shorewall.org/manpages/shorewall-interfaces.html
zones	zones in /etc/shorewall/zones	https://shorewall.org/manpages/shorewall-zones.html
openports	ports in /etc/shorewall/rules	https://shorewall.org/manpages/shorewall-rules.html

For example, to add a policy to the BCM-managed section of the /etc/shorewall/policy file on the head node, a cmsh session can be run as follows:

Example

```
[root@head ~]# cmsh
[head]% device use head; roles
[head->device[head]->roles]% use firewall
[head->device[head]->roles[firewall]]% show
Parameter                        Value
-------------------------------- ------------------------------------------------
Name                             firewall
Revision
Type                             FirewallRole
Add services                     yes
Shorewall                        no
Open ports                       <0 in submode>
Zones                            <0 in submode>
Interfaces                       <0 in submode>
Policies                         <0 in submode>
Routes                           <0 in submode>
[head->device[head]->roles[firewall]]% policies
[head->device[head]->roles[firewall]->policies]% list
Index  Source Dest   Policy       Log       Options
------ ------ ------ ------------ --------- ------------
```

The preceding output shows no additional policies are currently managed by BCM in that mode. To add some BCM-managed policies, the administrator can check how to do it by typing in the add command (section 2.5.3 of the *Administrator Manual*) without any arguments:

```
[head->device[head]->roles[firewall]->policies]% add
Name:
          Create a new firewallpolicy with specified policy

Usage:
          add <policy>
          add <source>
          add <source> <dest>
          add <source> <dest> <policy>
          add <source> <dest> <policy> <log>
          add <source> <dest> <policy> <log> <options>
```

```
Examples:
          add loc net ACCEPT
          add loc net ACCEPT info
```

Based on the lookup and some familiarity with Shorewall's policy file, the administrator can now compose suitable arguments for the `add` command, and `commit` (section 2.5.3 of the *Administrator Manual*) the changes:

Example

```
[head->device[head]->roles[firewall]->policies]% add net fw ACCEPT info
[head->device*[head*]->roles*[firewall*]->policies[0]]% #preceding means accept all traffic from other subnets
[head->device*[head*]->roles*[firewall*]->policies[0]]% list
Index  Source Dest   Policy       Log       Options
------ ------ ------ ------------ --------- ------------
0      net    fw     ACCEPT       info
[head->device*[head*]->roles*[firewall*]->policies[0]]% #next lines show how policy section changes on commit
[head->device*[head*]->roles*[firewall*]->policies[0]]% !head -4 /etc/shorewall/policy
# This section of this file was automatically generated by cmd. Do not edit manually!
# BEGIN AUTOGENERATED SECTION -- DO NOT REMOVE
# END AUTOGENERATED SECTION   -- DO NOT REMOVE

[head->device*[head*]->roles*[firewall*]->policies[0]]% commit
[head->device[head]->roles[firewall]->policies]% !head -4 /etc/shorewall/policy
# This section of this file was automatically generated by cmd. Do not edit manually!
# BEGIN AUTOGENERATED SECTION -- DO NOT REMOVE
net     fw      ACCEPT  info
# END AUTOGENERATED SECTION   -- DO NOT REMOVE
```

After the `commit` command is run, the additional policy is placed and becomes active in the `policy` file.

Similarly, to open a port on the head node via the `openports` mode, a `cmsh` session can be run as follows:

Example

```
[head->device[head]->roles[firewall]]% openports
[head->device[head]->roles[firewall]->openports]% add
Name:
          Create a new firewallopenport with specified port

Usage:
          add <port>
          add <description> <port>
          add <action> <network> <dest|port> <protocol> <dest>
          add <action> <network> <dest|port> <protocol> <dest> <cidr>
          add <action> <network> <dest|port> <protocol> <dest> <cidr> <description>

Examples:
          add 8080
          add grafana 3000
          add ACCEPT net 8080
          add ACCEPT net 8080 tcp
          add ACCEPT net 8080 tcp fw
          add ACCEPT net 8080 tcp fw 10.3.9.0/24
          add ACCEPT net 8080 tcp fw 10.3.9.0/24 "nginx proxy"
```

Based on the lookup, the administrator can compose suitable arguments for the `add` command, and `commit` (section 2.5.3 of the *Administrator Manual*) the changes:

Example

```
[head->device[head]->roles[firewall]->openports]% add ACCEPT net 443 tcp fw
[head->device*[head*]->roles*[firewall*]->openports*[0]]% commit
[head->device[head]->roles[firewall]->openports]% !tail -6 /etc/shorewall/rules
# BEGIN AUTOGENERATED SECTION -- DO NOT REMOVE
ACCEPT   net         fw              TCP     443
# END AUTOGENERATED SECTION   -- DO NOT REMOVE
```

After the `commit` command is run, the new port rule is placed in the `rules` file, and the `shorewall` service is restarted to apply it.

7.2.5 Further Shorewall Quirks

Shorewall Is Normally Only On The Head Node

Shorewall normally runs on the head node.

Shorewall can also run on the regular nodes, but does not run on them by default.

In a type 1 network topology (section 3.3.9), running Shorewall on the regular nodes makes no real sense from a security point of view. This is because the head node controls all the regular nodes, and access to the regular nodes is through the head node firewall.

For type 2 and type 3 topologies, turning Shorewall on for the regular nodes must be considered. It can be turned on by setting the `shorewall` attribute in the `firewall` role for the node from `off` to `on`.

Example

```
[basecm11->device*[node001*]->roles*[firewall*]]% get shorewall
no
[basecm11->device*[node001*]->roles*[firewall*]]% !ssh node001 systemctl status shorewall.service
...
Active: inactive (dead)
...
[basecm11->device*[node001*]->roles*[firewall*]]% set shorewall on
[basecm11->device*[node001*]->roles*[firewall*]]% commit ; show
Parameter                        Value
-------------------------------- ------------------------------------------------
Name                             firewall
Revision
Type                             FirewallRole
Add services                     yes
Shorewall                        yes
...

[basecm11->device[node001]->roles[firewall]]% !systemctl status shorewall.service
ssh: connect to host node001 port 22: Connection refused
```

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, Firewalld, must stay disabled in RHEL8 and 9 and their derivatives. This is because BCM 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 BCM Considered Harmful

System administrators wishing to stop Shorewall should note that BCM by default has the `autostart` setting (section 3.15 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 Base View.

Stopping Shorewall outside of `cmsh` or Base 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 `ping` and `failoverping` checks (both based on SYN connections), and the CMDaemon status (based on REST calls) (section 15.4.2 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 Base 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 Base View in the first place.

Full documentation on the specifics of Shorewall is available at `http://www.shorewall.net`.

7.3 The GCC Compiler

The GCC compiler may be installed through `yum`, `zypper`, or `apt` (section 9.2 of the *Administrator Manual*).

The package names are

- `gcc-recent` for RHEL and derivatives, and SLES
- `cm-gcc` for Ubuntu

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

An alternative to the GCC compiler is the Intel compiler, which is part of the oneAPI toolkit, available from `https://www.intel.com/content/www/us/en/developer/tools/oneapi/toolkits.html`. Earlier versions of BCM had the Intel compiler components available from the BCM repositories; this is no longer the case for BCM 10.0 and onward.

7.4 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 on the image to be used by the nodes that have GPUs.

7.4.1 AMD GPU Hardware Check

A hardware check to see if the node has an AMD GPU is:

Example

```
[root@node001 ~]# lspci | grep AMD
00:06.0 VGA compatible controller: Advanced Micro Devices, Inc. [AMD/ATI] Vega 10\
 [Instinct MI25/MI25x2/V340/V320]
```

If the kernel has detected the AMD GPU, then a `grep amdgpu` of the `dmesg` output shows all sorts of messages that indicate that:

Example

```
[root@node001 ~]# dmesg | grep amdgpu
[  157.942430] [drm] amdgpu kernel modesetting enabled.
```

```
[  157.945225] amdgpu: CRAT table not found
[  157.945241] amdgpu: Virtual CRAT table created for CPU
[  157.945265] amdgpu: Topology: Add CPU node
[  158.211025] amdgpu 0000:00:06.0: amdgpu: Fetched VBIOS from ROM
[  158.211065] amdgpu: ATOM BIOS: 113-D0513600-109
[  158.226062] amdgpu 0000:00:06.0: amdgpu: Trusted Memory Zone (TMZ) feature not supported
[  159.253402] amdgpu 0000:00:06.0: amdgpu: MEM ECC is active.
[  159.253418] amdgpu 0000:00:06.0: amdgpu: SRAM ECC is not presented.
[  159.253465] amdgpu 0000:00:06.0: amdgpu: VRAM: 16368M 0x000000F400000000 - 0x000000F7FEFFFFFF (16368M used)
...
```

7.4.2 AMD GPU Driver Installation Per Supported Distribution

Across the distributions—RHEL and derivatives, Ubuntu, and SLES—the procedure to install the GPU drivers is very similar.

Cloning An Image For AMD GPU Nodes

In all distributions, the `default-image` can first be cloned to an image that is to be the AMD GPU image, for example `am`:

Example

```
[root@basecm11:~]# cmsh
[basecm11]% softwareimage
[basecm11->softwareimage]% clone default-image am
[basecm11->softwareimage*[am*]]% commit
[basecm11->softwareimage[am]]%
[notice] basecm11: Started to copy: /cm/images/default-image -> /cm/images/am (4117)
...
[notice] basecm11: Initial ramdisk for image am was generated successfully
[basecm11->softwareimage[am]]% quit
```

AMD Instructions For Installing The Driver Packages On AMD GPU Nodes

Instructions from AMD describe how to install the driver packages. At the time of writing (February 2024) the instructions are at:

`https://rocm.docs.amd.com/projects/install-on-linux/en/latest/tutorial/quick-start.html`

The instructions describe two ways of installing the driver:

1. configuring the AMD driver repository for the package manager, and then installing the driver via standard package manager commands and OS changes
2. installing an AMDGPU installer that takes care of the installation

In either case, the instructions that are followed must be the ones for the OS release version of the software image that is on the AMD GPU nodes. The release version can be found with:

Example

```
root@basecm11:~# grep PRETTY /etc/os-release
PRETTY_NAME="Rocky Linux 9.2 (Blue Onyx)"
```

Installing The Driver Packages On AMD GPU Nodes On The Node Images Via Chroot

The installation must be done in the image. This is done using a chroot into the image, and using 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 536 of the *Administrator Manual*) is used to automate the job.

The following session output illustrates (with much text elided) the procedure for the second of the two ways of installing the driver. That is, installing and running the AMDGPU installer on a software image:

The chrooted image created earlier (`am`) is first entered:

Example

```
[root@basecm11 ~]# cm-chroot-sw-img /cm/images/am/
mounted /cm/images/default-image/dev
mounted /cm/images/default-image/dev/pts
...
```

The `amdgpu-install` package can then be installed, as per the AMD instructions. The package provides the AMDGPU installer (`amdgpu-install`), which is run in a chrooted image to install the ROCm stack. Installation takes a while, with much of the time taken up by the compilation steps that are carried out for the DKMS parts.

- For RHEL 9.2 and derivatives the procedure looks like:

  Example

  ```
  [root@am /]# urldomain=https://repo.radeon.com
  [root@am /]# urlpath=/amdgpu-install/6.0.2/rhel/9.2/amdgpu-install-6.0.60002-1.el9.noarch.rpm
  [root@am /]# yum install $urldomain$urlpath
  ...
  [root@am /]# amdgpu-install --usecase=graphics,rocm
  ...
  ```

  The exact path depends on which version is used.

- For Ubuntu 22.04 (Jammy Jellyfish) the procedure looks like:

  Example

  ```
  root@am:/# urldomain=https://repo.radeon.com
  root@am:/# urlpath=/amdgpu-install/6.0.2/ubuntu/jammy/amdgpu-install_6.0.60002-1_all.deb
  root@am:/# wget $urldomain$urlpath
  ...
  root@am:/# apt install ./amdgpu-install_6.0.60002-1_all.deb
  ...
  root@am:/# amdgpu-install --usecase=rocm
  ...
  ```

- For SLES15 the procedure for SLES15sp5 looks like:

  Example

  ```
  am:/ # urldomain=https://repo.radeon.com
  am:/ # urlpath=/amdgpu-install/latest/sle/15.5/amdgpu-install-6.0.60000-1.noarch.rpm
  am:/ # zypper --no-gpg-checks install $urldomain$urlpath
  ...
  am:/ # amdgpu-install --usecase=graphics,rocm
  ...
  ```

  The exact path depends on which service pack (sp) version is in use on the node.

Replacing The `rocm-smi` Package With A Package From BCM On The AMD GPU Nodes Image

Running the AMDGPU installer installs the `rocm-smi` command, which is used for some monitoring tasks related to AMD GPUs. BCM requires its own version of this command to collect AMD GPU statistics. If such statistics are needed, then the BCM `cm-rocm-smi` package should be installed in the image too, to provide the BCM version:

```
[root@am /]# yum install cm-rocm-smi
```
for RHEL and derivatives

For Ubuntu, the equivalent installation command is `apt install cm-rocm-smi`, while for SLES it is `zypper install cm-rocm-smi`.

The chroot for the prepared image can now be exited, which automatically unmounts the bind mounts:

Example

```
[root@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
```

Using The Image For The AMD GPU Nodes

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

Example

```
[root@basecm11 ~]# cmsh
[basecm11]% device use node001
[basecm11->device[node001]]% set softwareimage am
[basecm11->device*[node001*]]% commit
[basecm11->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.

After the configuration file has been placed, the `ldconfig` command is run, still within chroot, to link the library in the image(s).

8

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

8.1 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 9.3 cluster:

Example

```
[root@basecm11 ~]# yum info cm-nvhpc | grep ^Version
Version      : 25.3

[root@basecm11 ~]# yum install cm-nvhpc
...
Installing dependencies:
 cm-nvhpc-cuda12.8
..
[root@basecm11 ~]# yum search cm-nvhpc-cuda*
...
cm-nvhpc-cuda11.8.x86_64 : NVIDIA HPC SDK
cm-nvhpc-cuda12.6.x86_64 : NVIDIA HPC SDK
cm-nvhpc-cuda12.8.x86_64 : NVIDIA HPC SDK
```

The preceding output is what was available at the time of writing (June 2025). The output can be expected to change.

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

8.2 Compiler Modules

The cm-nvhpc package makes several environment modules available for compiling:

```
[root@basecm11 ~]# module avail | grep -o -P 'nvhpc.*?/'
nvhpc-hpcx-cuda11/
nvhpc/
nvhpc-hpcx-cuda12/
nvhpc-byo-compiler/
nvhpc-hpcx/
nvhpc-hpcx-2.20-cuda12/
nvhpc-nompi/
```

The nvhpc-hpcx-cuda11 or nvhpc-hpcx-cuda12 environment module sets up the HPC-X library environment with the selected CUDA version.

The nvhpc-hpcx environment module sets up the HPC-X library environment. This is an alternative to the OpenMPI 3.x library.

The nompi tag implies that paths to the MPI binaries and MPI libraries that come with cm-nvhpc are not set, so that no MPI library is used from the package. An external MPI library can then be used with the nvhpc-nompi compiler.

The nvhpc environment module is the standard HPC SDK.

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.

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

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

Example

```
[root@basecm11] ~# module load shared nvhpc
[root@basecm11] ~# basename -a $(ls -1d $NVHPC_ROOT/cuda/[0-9]*)
12.6
12.8
```

The running CUDA version for nvhpc can be viewed with:

Example

```
[root@basecm11 ~]# nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2025 NVIDIA Corporation
Built on Fri_Feb_21_20:23:50_PST_2025
Cuda compilation tools, release 12.8, V12.8.93
Build cuda_12.8.r12.8/compiler.35583870_0
```

8.4 Changing The Running CUDA Version

In the preceding example, CUDA version 12.8 is the default version, while CUDA version 12.6 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 12.6, as follows, for:

- `nvhpc` cluster-wide:

```
[root basecm11] # makelocalrc ${NVHPC_ROOT}/compilers/bin -cuda 12.6 -x
```

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

```
[root basecm11] # makelocalrc ${NVHPC_ROOT}/compilers/bin -cuda 12.6 -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`:

```
[root basecm11] # makelocalrc ${NVHPC_ROOT}/compilers/bin -cuda 12.6 -o > \
${HOME}/localrc.$(hostname -s)
```

If the `nvhpc` compiler is run then:

1. the `${NVHPC_ROOT}/compilers/bin/localrc` configuration file is read,
2. followed by a second configuration file, which—if it exists—is either:
   - the `${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.

9

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 BCM software that was obtained. If the CUDA packages are not in place, then they can be picked up from the BCM repositories, or a local mirror.

9.1 Installing CUDA

9.1.1 CUDA Packages Available

At the time of writing of this section (April 2026), the CUDA 13.1 package is the most recent version available in the BCM YUM, zypper, and APT repositories. The available versions are updated typically in the next point release version of NVIDIA Base Command Manager after the upstream changes are made available. An up-to-date list of packages available for a particular distribution and architecture can be viewed at `https://service.bcm.nvidia.com/packages-dashboard/`.

Thus, for example, at the time of writing, Rocky Linux 9.6 has many CUDA versions available, in the range from CUDA 11.7 to CUDA 13.1 for the `x86_64` architecture, while Ubuntu 24.04 just has CUDA 12.5, CUDA 12.6, CUDA 12.8, CUDA 12.9, CUDA-13.0, and CUDA 13.1 for `x86_64`.

Example: CUDA Packages Available For Ubuntu 24.04

At the time of writing of this section (April 2026), a cluster administrator can manage these packages for Ubuntu 24.04:

Package	Type	Description

`cuda12.5-toolkit`
`cuda12.6-toolkit`
`cuda12.8-toolkit`
`cuda12.9-toolkit`
`cuda13.0-toolkit`
`cuda13.1-toolkit`
	shared	CUDA math libraries and utilities

`cuda12.5-visual-tools`
`cuda12.6-visual-tools`
`cuda12.8-visual-tools`
`cuda12.9-visual-tools`
`cuda13.0-visual-tools`
`cuda13.1-visual-tools`
	shared	CUDA visual toolkit

`cuda12.5-sdk`
`cuda12.6-sdk`
`cuda12.8-sdk`
`cuda12.9-sdk`
`cuda13.0-sdk`
`cuda13.1-sdk`
	shared	CUDA software development kit

`nvidia-driver`**,†	local	CUDA GPU driver and libraries.
`cuda-dcgm`	local	CUDA Data Center GPU Manager (DCGM). This includes the `dcgmi` CLI tool.
`nvidia-fabricmanager`*,** `nvidia-fabricmanager-dev`*,**	local	Fabric Manager tools

* optional, intended only for hardware containing an NVSwitch, such as a DGX system

** installed by default on software images, not on head nodes

† the driver typically has a version number tag such as `-590`.

- 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
  - `nvidia-driver` (NVIDIA driver metapackage)
  - `cuda-dcgm` (NVIDIA Data Center GPU Manager)
  - `nvidia-fabricmanager` (Fabric Manager for NVSwitch based systems)
  - `nvidia-fabricmanager-dev` (Fabric Manager API headers and associated library)

  packages should be installed in a software image (section 2.1.2 of the *Administrator Manual*). By default, BCM software images already have these packages, which means that an explicit installation by the cluster administrator should not be required.

If a head node also accesses GPUs, then the `nvidia-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. For example, to select CUDA SDK version 13.0, the user can run:

Example

```
user@basecm11:~$ module add shared cuda13.0
```

Release notes and features list for the CUDA packages:

- Release notes for the NVIDIA toolkit can be found at: `https://docs.nvidia.com/cuda/cuda-toolkit-release-notes/index.html`
- The features list for the NVIDIA toolkit can be found at: `https://docs.nvidia.com/cuda/cuda-features-archive/index.html`

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

Package	Type	Description	Installation on GPU node
`cuda-dcgm-libs`	local	NVIDIA DCGM libraries	installed by default
`cuda-dcgm-devel`	local	NVIDIA DCGM development files	not installed by default
`cuda-dcgm-nvvs`	local	NVIDIA DCGM validation suite	installed by default, except for in RHEL8, and except for in a DGX node

At the time of writing of this paragraph (April 2026) the latest DCGM version supported by BCM was 4.5.2. Running a

- `dnf info cuda-dcgm; dnf list cuda-dcgm` for RHEL-family distributions
- `apt show -a cuda-dcgm` for Ubuntu distributions

should reveal the currently installed and available versions for the cluster.

- Release notes for the DCGM versions are available at `https://docs.nvidia.com/datacenter/dcgm/latest/release-notes/changelog.html`.

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 available for some distributions for some older CUDA versions, and provided by the package `cm-cub-cuda`.

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`*. Here 4 GPUs are detected:*

```
[root@node001]# { lspci | grep '3D controller: NVIDIA';} && echo "NVIDIA GPU hardware detected by kernel"
0008:01:00.0 3D controller: NVIDIA Corporation Device 2941 (rev a1)
0009:01:00.0 3D controller: NVIDIA Corporation Device 2941 (rev a1)
0018:01:00.0 3D controller: NVIDIA Corporation Device 2941 (rev a1)
0019:01:00.0 3D controller: NVIDIA Corporation Device 2941 (rev a1)
NVIDIA GPU hardware detected by kernel
[root@node001]#
```

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

The `Device 2941` part of the string can be tallied against the identifiers in the `README.txt.gz` file of the `nvidia-driver-580` package series, at the node where it has been installed.

Example

```
root@basecm11:~# zgrep 2941 /cm/images/default-image/usr/share/doc/nvidia-driver-580/README.txt.gz
 NVIDIA GB200                                            2941 10DE 2046  K
 NVIDIA GB200                                            2941 10DE 20CA  K
 NVIDIA GB200                                            2941 10DE 20D5  K
 NVIDIA GB200                                            2941 10DE 21C9  K
 NVIDIA GB200                                            2941 10DE 21CA  K
```

The output of the `lspci` command can be more directly informative for some hardware:

Example

```
root@basecm11:~# ssh node001 "lspci | grep 3D controller: NVIDIA"
00:06.0 3D controller: NVIDIA Corporation GA100 [A100 PCIe 40GB] (rev a1)
root@basecm11:~# zgrep A100 /cm/images/default-image/usr/share/doc/nvidia-driver-580/README.txt.gz \
| grep -i PCIE | grep 40GB
 NVIDIA A100-PCIE-40GB                                   20F1 10DE 145F  J
```

- Only after CUDA package installation has taken place, and after rebooting the node with the GPU, and allowing CMDaemon some time (a few minutes) to detect the change, are GPU details visible using the `sysinfo` command:

Example

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

```
[root@basecm11 ~]# cmsh
[basecm11]% device use node001
[basecm11->device[node001]]% sysinfo | grep GPU
```

Example

running `sysinfo` *on* `node001`*, via* `cmsh` *on the head node, after* `cuda-dcgm` *is ready. 4 GPUs are detected*

```
[basecm11->device[node001]]% sysinfo | grep GPU
[maple->device[dgx-gb200-n01-c1]]% sysinfo  | grep GPU
Motherboard Name          GB200 1CPU:2GPU Board PC
Number of GPUs            4
GPU[0-3] name             NVIDIA Graphics Device
GPU[0-3] power limit      1.20KW
GPU[0-3] BIOS             97.00.6C.00.03
GPU[0-3] driver version   570.82
GPU[0-3] nvlink           2 up
```

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 with a CPU 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 `nvidia-driver` package should be installed on the node on which the compilation is done, and the GPU-related services on the node, such as:
    - `nvidia-persistenced.service`
    - `nvidia-fabricmanager.service`
    - `nvidia-dcgm.service`

    should be disabled.

9.1.2 CUDA Package And Modules

The CUDA packages require access to the BCM repositories and to the distribution repositories.

The `nvidia-driver` packages replace the older `cuda-driver` packages that were used in versions of BCM prior to version 11. The newer packages do not compile the driver modules source at boot time, but during package installation. The newer packages are also open source.

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 install the CUDA 13.0 packages on the head using APT:

```
root@mycluster:~# apt install cuda13.0-toolkit cuda13.0-sdk cuda13.0-visual-tools
```

The `nvidia-driver-`, `cuda-dcgm`, `cuda-dcgm-nvvs`, and `nvidia-fabricmanager` are already installed by default on the default software image, which is `default-image` on a newly-installed default cluster.

Carrying out an explicit installation or `imageupdate` for these is therefore normally not needed.

In cases where it is required, then manual installation into a software image is described in section 9.4 of the *Administrator Manual*, while running the `imageupdate` command (section 5.6 of the *Administrator Manual*), for example on nodes 1 to 10, can be carried out with:

Example

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

If the CUDA packages have been set up on the cluster, then the end user typically just switches between CUDA environments with a module change (section 2.2 of the *Administrator Manual* and section 2.3 of the *User Manual*).

Example

```
module purge && module load shared cuda13.0/toolkit
```

9.2 Verifying CUDA

The `cuda*-sdk` packages provide CUDA SDK sources that 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 9.2).

An extensive method to verify that CUDA is working is to run the `verify_cuda<version>.sh` script, located in the CUDA SDK directory. The value of <*version*> can be `13.0`, for example.

This script first copies the CUDA SDK source to a local directory under `/tmp` or `/local`. It then builds CUDA test binaries. To build the makefiles, the compilation may need `cmake`, installed from the `cm-make` package into the image (section 9.4 of the *Administrator Manual*).

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_cuda<version>.sh -h`" is run.

A typical run output is as follows (some output elided):

Example

```
root@node001:~# module load shared cuda13.0/toolkit
root@node001:~# cd $CUDA_SDK
root@node001:/cm/shared/apps/cuda13.0/sdk/13.0.2# ./verify_cuda13.0.sh
Copy cuda13.0 sdk files to "/tmp/cuda13.0" directory.

cmake

make (may take a while)

Run multiple tests non-interactive ? (y/N)?
Executing: /tmp/cuda13.0/build/CMakeFiles/3.26.3/CMakeDetermineCompilerABI_CUDA.bin

80cmd: /tmp/cuda13.0/build/CMakeFiles/3.26.3/CMakeDetermineCompilerABI_CUDA.bin - rc : 0

Executing: /tmp/cuda13.0/build/CMakeFiles/FindOpenMP/ompver_C.bin

INFO:OpenMP-date[201511]
cmd: /tmp/cuda13.0/build/CMakeFiles/FindOpenMP/ompver_C.bin - rc : 0

Executing: /tmp/cuda13.0/build/CMakeFiles/FindOpenMP/ompver_CXX.bin

INFO:OpenMP-date[201511]
cmd: /tmp/cuda13.0/build/CMakeFiles/FindOpenMP/ompver_CXX.bin - rc : 0

Executing: /tmp/cuda13.0/build/Samples/0_Introduction/clock/clock

...

Test passed
cmd: /tmp/cuda13.0/build/Samples/6_Performance/cudaGraphsPerfScaling/cudaGraphsPerfScaling - rc : 0

All cuda13.0 compiled test programs can be found in the "/tmp/cuda13.0/build/" directory

Alternatively the "run_tests.py" can be run in the "/tmp/cuda13.0" directory.
For example:
  cd /tmp/cuda13.0; python3 run_tests.py --output ./test --dir ./build/Samples --config test_args.json

The "/tmp/cuda13.0" directory may take up a lot of diskspace.
```

```
Use "rm -rf /tmp/cuda13.0" to remove the data.
```

One of the binaries compiled is the `deviceQuery` command. This can be run 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). It can be run in CUDA 13 as follows:

```
root@node001:/cm/shared/apps/cuda13.0/sdk/13.0.2# /tmp/cuda13.0/Samples/1_Utilities/deviceQuery/deviceQuery
/tmp/cuda13.0/Samples/1_Utilities/deviceQuery/deviceQuery Starting...

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

Detected 1 CUDA Capable device(s)

Device 0: "NVIDIA A100-PCIE-40GB"
  CUDA Driver Version / Runtime Version          13.0 / 13.0
  CUDA Capability Major/Minor version number:    8.0
  Total amount of global memory:                 40441 MBytes (42405855232 bytes)
  (108) Multiprocessors, (064) CUDA Cores/MP:    6912 CUDA Cores
  GPU Max Clock rate:                            1410 MHz (1.41 GHz)
  Memory Clock rate:                             1215 Mhz
  Memory Bus Width:                              5120-bit
  L2 Cache Size:                                 41943040 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total shared memory per multiprocessor:        167936 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  2048
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 3 copy engine(s)
  Run time limit on kernels:                     No
  Integrated GPU sharing Host Memory:            No
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Enabled
  Device supports Unified Addressing (UVA):      Yes
  Device supports Managed Memory:                Yes
  Device supports Compute Preemption:            Yes
  Supports Cooperative Kernel Launch:            Yes
  Device PCI Domain ID / Bus ID / location ID:   0 / 0 / 6
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 13.0, CUDA Runtime Version = 13.0, NumDevs = 1
Result = PASS
root@node001:/cm/shared/apps/cuda13.0/sdk/13.0.2#
```

The query can detect if there is more than 1 GPU, and display output for each GPU that it finds.

The CUDA programming guide (`https://docs.nvidia.com/cuda/cuda-programming-guide/`) 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 INSTALLATION GUIDE FOR LINUX* at `https://docs.nvidia.com/cuda/pdf/CUDA_Installation_Guide_Linux.pdf`.

9.3 Verifying OpenCL

CUDA 12.8 contains an OpenCL compatible interface. To verify that OpenCL is working, or failing, the binaries can be compiled from the provided OpenCL source and run. For example:

Example

```
root@node001:~# module load shared cuda12.8/toolkit
root@node001 ~# cd $CUDA_SDK
root@node001:/cm/shared/apps/cuda12.8/sdk/12.8.1# cd opencl/OpenCL/ ; ls
bin  common  doc  Makefile  releaseNotesData  Samples.html  src
root@node001:/cm/shared/apps/cuda12.8/sdk/12.8.1/opencl/OpenCL# make
a - obj/release/oclUtils.cpp.o
make[1]: Leaving directory '/cm/shared/apps/cuda12.8/sdk/12.8.1/opencl/OpenCL/common'
make[1]: Entering directory '/cm/shared/apps/cuda12.8/sdk/12.8.1/opencl/shared'
ar: `u' modifier ignored since `D' is the default (see `U')
r - obj/x86_64/release/shrUtils.cpp.o
r - obj/x86_64/release/rendercheckGL.cpp.o
r - obj/x86_64/release/cmd_arg_reader.cpp.o
make[1]: Leaving directory '/cm/shared/apps/cuda12.8/sdk/12.8.1/opencl/shared'
make[1]: Entering directory '/cm/shared/apps/cuda12.8/sdk/12.8.1/opencl/OpenCL/src/oclBandwidthTest'
make[1]: Leaving directory '/cm/shared/apps/cuda12.8/sdk/12.8.1/opencl/OpenCL/src/oclBandwidthTest'
...
```

The binaries made using the `Makefile` can then be run from their directory:

```
root@node001:/cm/shared/apps/cuda12.8/sdk/12.8.1/opencl/OpenCL# cd bin/linux/release/
root@node001:/cm/shared/apps/cuda12.8/sdk/12.8.1/opencl/OpenCL/bin/linux/release# ls
oclBandwidthTest
oclBandwidthTest.txt
oclBlackScholes
oclConvolutionSeparable
oclCopyComputeOverlap
oclDCT8x8
oclDeviceQuery
oclDotProduct
oclDXTCompression
oclFDTD3d
oclHiddenMarkovModel
oclHistogram
oclInlinePTX
oclMatrixMul
oclMatVecMul
oclMersenneTwister
oclMultiThreads
oclQuasirandomGenerator
oclRadixSort
oclReduction
oclScan
oclSimpleMultiGPU
oclSortingNetworks
oclTranspose
```

```
oclTridiagonal
oclVectorAdd
root@node001:/cm/shared/apps/cuda12.8/sdk/12.8.1/opencl/OpenCL/bin/linux/release# for i in ocl*; do ./$i ; done
[oclBandwidthTest] starting...

./oclBandwidthTest Starting...

Running on...

Tesla V100-SXM3-32GB

Quick Mode

Host to Device Bandwidth, 1 Device(s), Paged memory, direct access
   Transfer Size (Bytes)        Bandwidth(MB/s)
   33554432                     5528.1

Device to Host Bandwidth, 1 Device(s), Paged memory, direct access
   Transfer Size (Bytes)        Bandwidth(MB/s)
   33554432                     3915.9

Device to Device Bandwidth, 1 Device(s)
   Transfer Size (Bytes)        Bandwidth(MB/s)
   33554432                     724307.1

[oclBandwidthTest] test results...
PASSED

> exiting in 3 seconds: 3...2...1...done!
```

...And so on. The other binaries listed should also run and pass....

9.4 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 deprecated. The diagnostics for CUDA 12.8 are run instead as part of the dcgmi diag command.

The package for NVVS is not installed by default. It is also not available for RHEL8 and derivatives.

If the package is not installed, then an attempt to run the dcgmi utility fails:

Example

```
[root@basecm11 ~] module load cuda-dcgm
[root@basecm11 ~]# dcgmi 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@basecm11 ~] module load cuda-dcgm
[root@basecm11 ~] 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@basecm11 ~] module load cuda-dcgm
[root@basecm11 ~]# dcgmi diag -r 3
Successfully ran diagnostic for group.
+---------------------------+-----------------------------------------------+
| Diagnostic                | Result                                        |
+===========================+===============================================+
|-----  Metadata  ----------+-----------------------------------------------|
| DCGM Version              | 4.1.1                                         |
| Driver Version Detected   | 570.82                                        |
| GPU Device IDs Detected   | 2941, 2941, 2941, 2941                        |
| Number of CPUs Detected   | 2                                             |
|-----  Deployment  --------+-----------------------------------------------|
| software                  | Pass                                          |
|                           | GPU0: Pass                                    |
|                           | GPU1: Pass                                    |
|                           | GPU2: Pass                                    |
|                           | GPU3: Pass                                    |
+-----  Hardware  ----------+-----------------------------------------------+
| memory                    | Skip                                          |
|                           | GPU0: Skip                                    |
|                           | GPU1: Skip                                    |
|                           | GPU2: Skip                                    |
|                           | GPU3: Skip                                    |
| diagnostic                | Skip                                          |
|                           | GPU0: Skip                                    |
|                           | GPU1: Skip                                    |
|                           | GPU2: Skip                                    |
|                           | GPU3: Skip                                    |
| nvbandwidth               | Skip                                          |
|                           | GPU0: Skip                                    |
|                           | GPU1: Skip                                    |
|                           | GPU2: Skip                                    |
|                           | GPU3: Skip                                    |
+-----  Integration  -------+-----------------------------------------------+
| pcie                      | Skip                                          |
|                           | GPU0: Skip                                    |
|                           | GPU1: Skip                                    |
|                           | GPU2: Skip                                    |
|                           | GPU3: Skip                                    |
+-----  Stress  ------------+-----------------------------------------------+
| memory_bandwidth          | Skip                                          |
|                           | GPU0: Skip                                    |
|                           | GPU1: Skip                                    |
|                           | GPU2: Skip                                    |
|                           | GPU3: Skip                                    |
| targeted_stress           | Skip                                          |
|                           | GPU0: Skip                                    |
|                           | GPU1: Skip                                    |
|                           | GPU2: Skip                                    |
|                           | GPU3: Skip                                    |
| targeted_power            | Skip                                          |
|                           | GPU0: Skip                                    |
|                           | GPU1: Skip                                    |
|                           | GPU2: Skip                                    |
|                           | GPU3: Skip                                    |
```

```
+--------------------------+----------------------------------------------+
```

9.5 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 3.17.2 of the *Administrator Manual*.

10

OFED Software Stack

This section explains how OFED (OpenFabrics Enterprise Distribution) packages are installed so that BCM 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*.

10.1 Choosing A Distribution Version, Or A Non-distribution Version, Ensuring The Kernel Matches, And Logging The Installation

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

BCM also allows NVIDIA's Mellanox OFED (sometimes called MOFED) software to be installed. Such separate, non-distribution OFED stacks can be more recent than the distribution version, which means that they can provide support for more recent hardware and firmware, as well as more features.

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

If NVIDIA AI Enterprise is to be used, then a matching, supported, version of OFED must be selected. A matching, supported OFED version can be found by contacting support via the BCM subscription at the NVIDIA Enterprise Support page at:

`https://www.nvidia.com/en-us/support/enterprise/`

Since June 2025, BCM no longer supports the non-distribution legacy BCM-packaged Mellanox OFED stack. These were packages that were provided from the BCM repository, with names such as `mlnx-ofed58`, `mlnx-ofed59-dgx-h100`, `mlnx-ofed59`, `mlnx-ofed23.04`, and `mlnx-ofed24.07`.

BCM has moved on to installing the non-distribution NVIDIA DOCA OFED stack directly (section 10.2), with the help of an installation script. The upstream NVIDIA DOCA OFED stack is a subset of the full DOCA software stack.

If there is no DOCA OFED stack available for the kernel in use, then falling back to a supported kernel is recommended.

When updating kernels on the head or the regular nodes, the updated NVIDIA DOCA OFED software stack must be reinstalled.

10.2 DOCA OFED Stack Installation Using The BCM DOCA OFED Installation Script

The set of DOCA OFED versions that can be installed can change with each BCM point release, as the DOCA LTS releases that BCM tracks are themselves updated. Running the installation script without arguments displays a usage help text that shows the versions the cluster currently offers. The same list is held in:

```
/cm/local/apps/cluster-tools/doca/etc/doca-ofed-versions.conf
```

At the time of writing of this section (August 2026), the file content is:

Example

```
root@basecm11:~# cat /cm/local/apps/cluster-tools/doca/etc/doca-ofed-versions.conf
# DOCA versions
2.9.4
3.2.2
```

The versions in the file on the actual cluster are the ones that BCM has install-tested against the supported Linux distribution versions. Restricting the choice in this way avoids a version being installed that is not supported on the cluster, which can leave the cluster in a broken state. An administrator may add a version to the file, but a version added in that way has not been tested by BCM.

If a required version is not offered by the cluster, then support should be contacted via the BCM subscription at the NVIDIA Enterprise Support page at:

`https://www.nvidia.com/en-us/support/enterprise/`

For DOCA OFED drivers, installation can be carried out for BCM with these methods:

- during the initial cluster installation (figure 3.23)
- on a running cluster by running the BCM DOCA installation script, which comes with the BCM `cluster-tools` package. This method is covered in this section.
- on a running cluster by creating an new image for nodes using the `cm-create-image` or `cm-image` tools with the `--extra-pkg-group` option. This method is covered in page 548 of the *Administrator Manual*.

The installation script is located at:

```
root@basecm11:~# /cm/local/apps/cluster-tools/bin/cm-doca-ofed-install.sh
```

Running the script without any arguments shows a usage help text.
The following arguments must be specified:

- `-v|--version`: This chooses which DOCA OFED version is to be used. The versions that can be given here are the ones the script itself lists.
- `-h` or `-s`: these select whether the DOCA OFED installation is carried out on the head node or on a software image

Example

Installing on the head node, where `<version>` is replaced by one of the versions the script lists:

```
/cm/local/apps/cluster-tools/bin/cm-doca-ofed-install.sh -v <version> -h
```

The following argument for the installation script is optional:

- `--cmdvd`: this option makes the script install from a BCM install medium rather than from the NVIDIA DOCA package repositories, which is useful for air-gapped deployments. BCM 11 install media carry DOCA OFED packages, so pointing the option at one finds the packages directly.

The installation log is recorded in:
`/var/log/cm-doca-ofed.log`
while the version installed is recorded in
`/etc/cm-ofed`

11

Burning Nodes

The *burn framework* is a component of NVIDIA Base Command Manager 11 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.

11.1 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 11.3.2.

11.2 Burn Configurations

11.2.1 Burn Configuration Simple XML Example

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 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"/>
      <test name="sleep" args="10"/>
    </phase>

    <phase name="02-disks">
      <test name="disktest" args="30"/>
      <test name="mce_check" endless="1"/>
    </phase>

  </pre-install>

  <post-install>

    <phase name="03-hpl">
      <test name="hpl"/>
      <test name="mce_check" endless="1"/>
    </phase>

    <phase name="04-compile">
      <test name="compile" args="6"/>
      <test name="mce_check" endless="1"/>
    </phase>

  </post-install>

</burnconfig>
```

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.

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.

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.

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.

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 11.2). 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.

11.2.2 Burn Configuration XML Schema Definition

In BCM, burn configuration setups such as in section 11.2.1 have their global structure defined using an XML schema definition, which is installed on the head node in `/cm/local/apps/cmd/etc/htdocs/xsd/burnconfig.xsd`.

The XML schema definition can also be viewed with:

Example

```
[root@basecm11 ]# cmsh -c "main xsdschema burnconfig" | less
```

11.3 Running A Burn Configuration

Burn configurations can be viewed and executed from `cmsh`.

11.3.1 Burn Configuration And Execution In `cmsh`

Burn Configuration File Settings

From `cmsh`, the burn configurations can be accessed from `partition` mode as follows:

Example

```
[basecm11]% partition use base
[basecm11->partition[base]]% burnconfigs
[basecm11->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

```
[basecm11->partition[base]->burnconfigs]% use default-destructive
[basecm11->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

```
[basecm11->partition[base]->burnconfigs]% add boxburn
[basecm11->partition[base]->burnconfigs*[boxburn*]]% set xml /tmp/im.xml
```

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.

```
[head1->device[node005]]% burn
Name:     burn - Node burn control

Usage:    burn [OPTIONS] status
          burn [OPTIONS] start
          burn [OPTIONS] stop
          burn [OPTIONS] log

Options:  -n, --nodes <node>
               List of nodes, e.g. node001..node015,node020..node028,node030
               or ^/some/file/containing/hostnames

          -g, --group <group>
               Include all nodes that belong to the node group, e.g. testnodes or test01,test03

          -c, --category <category>
               Include all nodes that belong to the category, e.g. default or default,gpu

          -r, --rack <rack>
               Include all nodes that are located in the given rack, e.g rack01
               or rack01..rack04
```

```
-h, --chassis <chassis>
     Include all nodes that are located in the given chassis, e.g chassis01
     or chassis03..chassis05

-e, --overlay <overlay>
     Include all nodes that are part of the given overlay, e.g overlay1
     or overlayA,overlayC

-m, --image <image>
     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:

    ```
    [basecm11->device]% burn --config boxburn -n node007 start
    Power reset nodes
    [basecm11->device]%
    ipmi0 ..................... [  RESET  ] node007
    Fri Nov  3 ... [notice] basecm11: node007 [ DOWN ]
    [basecm11->device]%
    Fri Nov  3 ... [notice] basecm11: node007 [ INSTALLING ] (node installer started)
    [basecm11->device]%
    Fri Nov  3 ... [notice] basecm11: node007 [ INSTALLING ] (running burn in tests)
    ...
    ```

  - The "`burn stop`" command only needs to be given the nodes it operates on, for example:

    ```
    [basecm11->device]% burn -n node007 stop
    ```

  - The "`burn status`" command:
    * may be given the nodes for which the status is to be found, for example:

```
[basecm11->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
...--------- ------------------------------------------------------------
...0
...0
...0         /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/bu+
```

  - 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`: These options allow burn commands to 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
  - `-p|--path`: This shows the burn log path, by default (section 11.4) on the head node under a directory named after the mac address `/var/spool/burn/`<*mac*>.

    Example

    ```
    [basecm11->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

```
[basecm11->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:

Description	Value	Meaning Here
The node name	`node001`	
The node tag	`(00000000a000)`	
Warnings since start of burn	`(0)`	
The current phase name	`02-disks`	Burn configuration phase being run is `02-disks`
Time since phase started	`00:02:58 (D:H:M)`	2 hours 58 minutes
State of current phase	`FAILED`	Failed in `02-disks`
burn test for MCE	`mce_check (SP)`	Started and Passed
burn test for disks	`disktest (SF,61)`	Started and Failed 61 is the speed, and is custom information
burn test kernel log monitor	`kmon (SP)`	Started and Passed

Each test in a phase uses these letters to display its status:

Letter	Meaning
S	started
W	warning
F	failed
P	passed

The "`burn log`" command output looks like the following (some output elided):

```
[basecm11->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
Thu ... 2012: node001 - burn-control: starting test /cm/shared/apps/cmburn/mce_check
Thu ... 2012: node001 - burn-control: starting test /cm/shared/apps/cmburn/kmon
Thu ... 2012: node001 - disktest: starting, threshold = 30 MB/s
Thu ... 2012: node001 - mce_check: checking for MCE's every minute
Thu ... 2012: node001 - kmon: kernel log monitor started
Thu ... 2012: node001 - disktest: detected 1 drives: sda
...
Thu ... 2012: node001 - disktest: drive sda wrote 81920 MB in 1278.13
Thu ... 2012: node001 - disktest: speed for drive sda was 64 MB/s -> disk passed
Thu ... 2012: node001 - burn-control: test disktest has ended, test FAILED
Thu ... 2012: node001 - burn-control: all non-endless test are done, terminating endless tests
Thu ... 2012: node001 - burn-control: asking test /cm/shared/apps/cmburn/kmon/kmon to terminate
Thu ... 2012: node001 - kmon: kernel log monitor terminated
Thu ... 2012: node001 - burn-control: test kmon has ended, test passed
Thu ... 2012: node001 - burn-control: asking test /cm/shared/apps/cmburn/mce_check/mce_check\
 to terminate
Thu ... 2012: node001 - mce_check: terminating
Thu ... 2012: node001 - mce_check: waiting for mce_check to stop
Thu ... 2012: node001 - mce_check: no MCE's found
Thu ... 2012: node001 - mce_check: terminated
Thu ... 2012: node001 - burn-control: test mce_check has ended, test passed
Thu ... 2012: node001 - burn-control: phase 02-disks FAILED
Thu ... 2012: node001 - burn-control: burn will terminate
```

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, 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/
```

11.3.2 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 11.1 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

  blog "$me: waiting for $me to stop"
  # wait for the main loop to die
  while [ -d /proc/`cat $spooldir/$me/pid` ]
  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

fi
```

11.3.3 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
```

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:

```
cd /var/spool/burn/48:5b:39:19:ff:b3
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
drwxr-xr-x 6 root root 4096 Jan 10 16:26 02-disks.1
```

3rd try, after failing again:

```
ls -ld 02-disks*/
drwxr-xr-x 6 root root 4096 Jan 10 16:59 02-disks
drwxr-xr-x 6 root root 4096 Jan 10 16:49 02-disks.1
drwxr-xr-x 6 root root 4096 Jan 10 16:26 02-disks.2
```

11.4 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.

11.4.1 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`
- `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 Base View, via the navigation path:

   `Devices > All Devices > Head Node > 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 Base View, then the change can be carried out from within the `fsexports` submode. Section 3.14.1 of the *Administrator Manual* gives more detail on similar examples of how to add such filesystem exports.

11.4.2 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 `cmsh`:

Example

```
[basecm11->device]% burn start --config default-destructive --edit -n node001
```

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 `mprime` test for about two minutes.

12

Installing And Configuring SELinux

12.1 Introduction

Security-Enhanced Linux (SELinux) can be enabled on selected nodes. If SELinux is enabled on a standard Linux operating system, then it is typically initialized in the kernel when booting from a hard drive. However, in the case of nodes provisioned by NVIDIA Base Command Manager, using 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 must therefore 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 for BCM 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.

SELinux is partially managed by BCM and can run on the head and regular nodes. The SELinux settings managed by CMDaemon (using `cmsh` or Base View) should not be managed by directly dealing with the node outside of CMDaemon management, as that can lead to an inconsistent knowledge of the SELinux settings by CMDaemon.

When first configuring SELinux to run with BCM on regular nodes, the nodes should be configured with permissive mode to ensure that the nodes work with applications. Troubleshooting permissive mode so that enforcing mode can be enabled is outside the scope of BCM support, unless the issue is demonstrably a BCM-related issue.

12.2 Enabling SELinux On RHEL8, Rocky 9

12.2.1 Setting SELinux Parameters

The `selinuxsettings` mode is available at node, category, or partition level.

Example

```
[root@basecm11 ~]# cmsh
[basecm11]% device use node001
[basecm11->device[node001]]% selinuxsettings
[basecm11->device[node001]->selinuxsettings]% show
Parameter                        Value
```

```
------------------------------- ------------------------------------------------
Initialize                      yes
Revision
Reboot after context restore    no
Allow NFS home directories      yes
Context action auto install     always
Context action full install     always
Context action nosync install   always
Mode                            permissive
Policy                          targeted
Key value settings              <submode>
```

The `Mode` can be set to `permissive`, `enforcing` or `disabled`. When starting the use of SELinux and establishing policies, it should be set to `permissive` to begin with, so that troubleshooting issues to do with running applications with `enforcing` mode can be examined.

The default SELinux configuration parameters are in `/cm/node-installer/scripts/node-installer.conf`, and that file remains unchanged by `cmsh` settings changes. The values of SELinux configuration parameters used from that file are however overridden by the corresponding `cmsh` settings.

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 values for *<distribution>* and *<architecture>* can take the values outlined on (page 558 of the *Administrator Manual*).

12.2.2 Setting Up On The Head Node

The following procedure can be run on both head nodes to configure SELinux on Rocky 9, so that the SELinux enforcing mode is enabled:

```
[root@basecm11 ~]# cmsh
[basecm11]% device use master
[basecm11->device[basecm11]]% selinuxsettings
[basecm11->device[basecm11]->selinuxsettings]% set mode enforcing
[basecm11->device*[basecm11*]->selinuxsettings*]% commit
[basecm11->device[basecm11]->selinuxsettings]% quit

[root@basecm11 ~]# touch /.autorelabel
[root@basecm11 ~]# reboot
...
[root@basecm11 ~]# getenforce
Enforcing
```

12.2.3 Setting Up On The Regular Nodes

A similar pattern can be repeated on the regular nodes.

Configuring The SELinux Settings

The SELinux settings can be configured at partition, category, or node level.

Nodes that are to use SELinux can be placed in a category, `secategory`. The `secategory` category can be created by cloning it from the `default` category that comes with a newly-installed cluster:

Example

```
[root@basecm11 ~]# cmsh
[basecm11]% category
[basecm11->category]% list
Name (key)              Software image          Nodes
```

```
------------------------ ------------------------ --------
default                  default-image            4
[basecm11->category]% clone default secategory; commit
```

The SELinux settings can then be configured for the newly-cloned category.

Example

```
[basecm11->category]% use secategory; selinuxsettings
[basecm11->category[secategory]->selinuxsettings]% keyvaluesettings
[basecm11->category*[secategory*]->selinuxsettings*->keyvaluesettings*]% set domain_can_mmap_files 1
[basecm11->category*[secategory*]->selinuxsettings*->keyvaluesettings*]% exit
[basecm11->category*[secategory*]->selinuxsettings*]% set mode<tab><tab>
disabled    enforcing   permissive
[basecm11->category*[secategory*]->selinuxsettings*]% set mode permissive    #for now, to debug apps
[basecm11->category*[secategory*]->selinuxsettings*]% commit
```

The `domain_can_mmap_files` boolean setting is needed to allow SELinux policies to revalidate some kinds of file access in memory.

Configuring The SELinux Image To Be Used

The problem with SELinux file contexts on the node when again provisioning an image: If the image that is used to provision the node still has the old file attributes, and if the provisioning mode is FULL or AUTO (section 5.4.4 of the *Administrator Manual*), then image sync during node-installer provisioning results in reverting the SELinux file contexts that have been set up on the running node back to the old state.

Thus, for FULL or AUTO provisioning modes, if `default-image` has no SELinux file security contexts, then the SELinux file security contexts vanish after the node comes back up after a reboot.

Creating a new image and using `setfiles` to set up SELinux file contexts on the new image: One good way to have a node come up with SELinux file contexts, is to set up the image that is provisioned so that the image has the contexts already.

This can be configured by first cloning the image, with:

Example

```
[basecm11->category[secategory]->selinuxsettings]% softwareimage
[basecm11->softwareimage]% list
Name (key)           Path                          Kernel version                Nodes
-------------------- ----------------------------- ----------------------------- --------
default-image        /cm/images/default-image      5.14.0-284.11.1.el9_2.x86_64  5
[basecm11->softwareimage]% clone default-image selinux-image; commit
...
...[notice] basecm11: Initial ramdisk for image selinux-image was generated successfully
```

Then, after `selinux-image` has been generated, the contexts can be set up in the new image with the SELinux `setfiles` command, using the `-r` option to set the root path:

Example

```
[basecm11->softwareimage]% quit
[root@basecm11 ~]# setfiles -r /cm/images/selinux-image \
/etc/selinux/targeted/contexts/files/file_contexts /cm/images/selinux-image/
[root@basecm11 ~]# setfiles -r /cm/images/selinux-image \
/etc/selinux/targeted/contexts/files/file_contexts.local /cm/images/selinux-image/
```

If the image is updated in the future with new packages, or new files, then the `setfiles` commands in the preceding example must be run again to set the file contexts.

Organizing the nodes and setting them up with the newly-created SELinux image: Nodes in the category can be listed with:

```
[basecm11->category[secategory]]% listnodes
```
...lists the nodes in that category...

Nodes can be placed in the category from `device` mode. For example, `node001`, `node002`, and `node003` can be configured with:

```
[basecm11->category[secategory]]% device
[basecm11->device]% foreach -n node001..node003 (set category secategory)
```

If the nodes in the category `secategory` are to run file systems with SELinux file contexts, then the image generated for this earlier on, `selinux-image`, can be committed to that category with:

Example

```
[basecm11->category[secategory]]% set softwareimage selinux-image
[basecm11->category*[secategory*]]% commit
```

Booting The Image For SELinux On The Regular Node

The category can then be rebooted to check that all is working correctly.

Example

```
[basecm11->category[secategory]]% device reboot -c secategory
Reboot in progress for: node001..node003
...
```
... reboot takes place...
```
[basecm11->category[secategory]]% !ssh node001 "getenforce"
Permissive
[basecm11->category[secategory]]% !ssh node001 "ls -Z"

system_u:object_r:admin_home_t:s0 original-ks.cfg
system_u:object_r:admin_home_t:s0 rpmbuild
```

A

Other Licenses, Subscriptions, Or Support Vendors

NVIDIA Base Command Manager comes with enough software to allow it to work with no additional commercial requirements other than its own. However, BCM 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.

Software	URL
Workload managers	
LSF	`http://www.ibm.com/systems/platformcomputing/products/lsf/`
Distributions	
Suse	`http://www.suse.com`
Red Hat	`http://www.redhat.com`
Compilers	
Intel	`https://software.intel.com/en-us/intel-sdp-home`
Miscellaneous	
Amazon AWS	`http://aws.amazon.com`

B

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 NVIDIA Base Command 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 25 GB is the minimum needed. 256GB should always be fine at the time of writing of this section (October 2023). 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 6GB, 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:

$\texttt{RAM used for non} - \texttt{BCM system processes} + 50\texttt{MB} + (\texttt{number of nodes} \times 10\texttt{kB})$.

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:

$\texttt{space needed by operating system without cluster manager} + 5\texttt{GB per regular node image} + (100\texttt{kB} \times \texttt{number of metrics and health checks} \times \texttt{number of devices})$

A device means any item seen as a device by CMDaemon. A list of devices can be seen by `cmsh` under its `device` mode. 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:

$\texttt{RAM used for normal system process} + 100\texttt{MB} + (\texttt{number of nodes} \times 1.8\texttt{MB})$

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:

Number of nodes	CMDaemon + MySQL RAM/GB	CMDaemon RAM/GB	Disk Use/GB
1000	16	2	10
2000	32	4	20
5000	64	10	50

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.

C

NVIDIA BCM Scope Of Support

C.1 Included In BCM Support

NVIDIA Base Command Manager (BCM) technical support has a standard scope of coverage that includes the following:

C.1.1 Support For Cluster Installation

The BCM support team will help install BCM on a supported Linux base distribution, using any of the documented BCM features.

If the add-on installation method is used (as opposed to the bare-metal installation method), then installation on top of a default minimal installation of the base Linux distribution is supported. A default minimal installation of the Linux base distribution means selecting a minimal set of packages during the installation and leaving all settings at their defaults.

C.1.2 Support For Cluster Configuration

The BCM support team can help create configurations to meet the specific needs of a customer using any of the documented BCM features.

If Security-Enhanced Linux (SELinux) is enabled on the head node or compute nodes and a problem arises, then a customer may be asked to demonstrate that the problem also exists when SELinux is disabled, in order to confirm that the problem is not caused by an SELinux policy configuration.

C.1.3 Support For Integration Of Cluster In Site Infrastructure

Site infrastructures vary greatly, and so support is not provided for the third-party hardware and software that may be involved. Nevertheless, the BCM support team will do its best to provide support on matters related to cluster integration within a site's infrastructure.

Information on how to accomplish many integration tasks is provided in the BCM product manuals and in the Knowledge Articles of the NVIDIA Enterprise Support Portal, and is often all that is needed to get the job done. If not, then the BCM support team may be able to help. The BCM support team will ask that the customer demonstrate how to accomplish the aspect of integration on an ordinary machine running the same Linux distribution as the cluster. Once the BCM support team has that information, it can provide instructions on how to accomplish the same task using BCM.

C.1.4 Support For Integration With Microsoft Active Directory

NVIDIA provides documentation that describes how to integrate the BCM cluster with Active Directory; however, that integration requires help from the Windows Server administrators. It is the responsibility of the BCM administrator to communicate with the Windows Server administrators to get the required information for the chosen integration method.

Debugging Active Directory integration issues is beyond the scope of standard support.

C.1.5 Support For First-Party Hardware Only

The BCM support team does not provide support for third-party hardware issues. When it is unclear whether an issue is due to hardware or software problems, the BCM support team will work with a customer to determine the source of the problem.

C.1.6 Support For Software Upgrades

Supported customers receive access to all new software releases, and the BCM support team manages any issues that arise as a result of an upgrade.

To identify versions of its software, BCM uses pairs of numbers (e.g., 8.0 or 9.2). The first number denotes the major version of the software and the second number denotes the minor version.

An upgrade within the major version (e.g., from 9.1 to 9.2) is called a minor upgrade. An upgrade from one major version to the next (e.g., from 8.2 to 9.0) is called a major upgrade.

Minor upgrades and recent major upgrades can be done in place. The BCM support team will enable smooth transitions during software upgrades.

C.2 Excluded From BCM Support

Support coverage for BCM excludes the following:

C.2.1 Help With Third-Party Software

Third-party software is all software not developed by NVIDIA, even though it may be packaged or integrated with BCM. Examples include, but may not be limited to, the Linux kernel, all software belonging to the Linux base distribution (e.g., Red Hat), Open MPI, ScaleMP, and workload management systems such as Slurm and LSF.

An exception will be made if a customer demonstrates that NVIDIA packaged an application incorrectly, or if the customer demonstrates that the integration in the stock user environment is incorrect.

C.2.2 Help With User Applications

No support can be provided on issues related to compiling or running user applications, i.e., applications that are not packaged as part of BCM and which are installed onto a cluster by a customer or other individual.

If a customer suspects the issue relates to an NVIDIA product, then the problem must be demonstrated using a small application that will allow NVIDIA engineers to reproduce the problem.

C.2.3 Help With Workload Management

No support will be provided on issues relating to the workload management system. The only exception is during the initial installation and configuration and in cases where a customer demonstrates that an issue is caused by incorrect integration between BCM and the workload manager.

C.2.4 Help With Performance Issues

No support will be provided to trace performance issues in the cluster. The only exceptions are performance issues related to software components developed by NVIDIA.

C.2.5 Root-Cause Analysis

The NVIDIA support engineers will do their best to determine the cause of a failure, but cannot guarantee that all issues can be traced down to a root cause.

C.2.6 Phone Support

Phone support is not provided by the BCM support team. This is because support via a written ticket system is generally more efficient, as it usually allows better tracking, more precision in conveying

meaning, easier information gathering, and a more optimal saving of information for later internal reference and search queries.

C.3 Support Outside Of The Standard Scope—Getting Professional Services

Support for items outside of the standard scope are considered professional services.

The BCM support team normally differentiates between

- regular support (customer has a question or problem that requires an answer or resolution), and
- professional services (customer asks for the team to do something or asks the team to provide some service).

Professional services outside of the standard scope of support can be purchased via the NVIDIA Enterprise Services page at:

`https://www.nvidia.com/en-us/support/enterprise/services/`