NVIDIA DGX BasePOD

Deployment Guide

Featuring NVIDIA DGX A100 System

DG-11192-003 V2

BCM 10.23.09

Abstract

Artificial intelligence (AI) infrastructure requires significant compute resources to train the latest state-of-the-art models efficiently, often requiring multiple nodes running in a distributed cluster.

While cloud computing provides an easy on-ramp to train AI models, many enterprises require an on-premises data center for a variety of technical or business reasons.

Building AI infrastructure on-premises can be a complex and confusing process. Careful planning and coordination will make the cluster deployment and the job of the cluster administrators tasked with the day-to-day operations easier.

NVIDIA DGX BasePOD™ provides the underlying infrastructure and software to accelerate deployment and execution of these new AI workloads. By building upon the success of NVIDIA DGX™ systems, DGX BasePOD is a prescriptive AI infrastructure for enterprises, eliminating the design challenges, lengthy deployment cycle, and management complexity traditionally associated with scaling AI infrastructure.

The DGX BasePOD is built upon NVIDIA DGX A100 systems, which offer unprecedented compute performance with eight NVIDIA A100 Tensor Core GPUs connected with NVIDIA NVLink® and NVIDIA NVSwitch™ technologies for fast inter-GPU communication.

Powered by NVIDIA Base Command™, DGX BasePOD provides the essential foundation for AI development optimized for the enterprise.

Contents

Chapter 1. Architecture

1.1 Hardware Overview

The DGX BasePOD consists of compute nodes, five control plane servers (two for cluster management and three Kubernetes (K8s) control plane nodes), as well as associated storage and networking infrastructure.

An overview of the hardware is in Table 1. Details about the hardware that can be used and how it should be cabled are given in the *NVIDIA DGX BasePOD Reference Architecture*.

This deployment guide describes the steps necessary for configuring and testing a four-node DGX BasePOD after the physical installation has taken place. Minor adjustments to specific configurations will be needed for DGX BasePOD deployments of different sizes, and to tailor for different customer environments, but the overall procedure described in this document should be largely applicable to any DGX BasePOD with NVIDIA A100 deployments.

Table 1. DGX BasePOD components

Component	Technology
Compute nodes	DGX A100 system
Compute fabric	NVIDIA Quantum QM8700 HDR 200 Gbps InfiniBand
Management fabric	NVIDIA SN4600 switches
Storage fabric	NVIDIA SN4600 switches for Ethernet attached storage NVIDIA Quantum QM8700 HDR 200 Gb/s for InfiniBand attached storage
Out-of-band management fabric	NVIDIA SN2201 switches
Control plane	Minimum Requirements (each server): > 64-bit x86 processor, AMD EPYC 7272 or equivalent > 256 GB memory > 1 TB SSD

	> Two 100 Gbps network ports

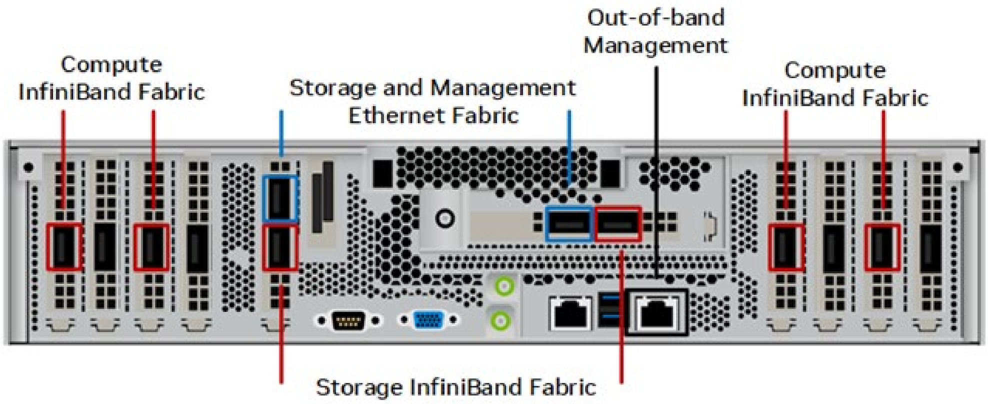
1.2 Networking

This section covers the DGX system network ports and an overview of the networks used by DGX BasePOD.

1.2.1 DGX A100 System Network Ports

Figure 1 shows the rear of the DGX A100 system with the network port configuration used in this solution guide.

Figure 1. DGX A100 system rear


Compute InfiniBand Fabric
Storage and Management Ethernet Fabric
Out-of-band Management
Compute InfiniBand Fabric
Storage InfiniBand Fabric

The following ports are selected for DGX BasePOD networking:

- Four single-port ConnectX-6 cards are used for the InfiniBand compute fabric, two on either side of the chassis (marked in red).
- Two ports of the dual-port ConnectX-6 cards are configured as a bonded Ethernet interface for in-band management and storage networks. These are the bottom port from slot 4 and the right port from slot 5 (marked in blue).
- BMC network access is provided through the out-of-band network (marked in gray).

The networking ports and their mapping are described in the Network Ports section of the *NVIDIA DGX A100 System User Guide*.

1.2.2 DGX BasePOD Network Overview

There are four networks in a DGX BasePOD configuration:

- `internalnet`—Network used exclusively within the cluster, for storage and in-band management.
- `externalnet`—Network connecting the DGX BasePOD to an external network, such as a corporate or campus network.
- `ipminet`—Network for out of band management, connecting BMCs.
- `ibnet`—InfiniBand network connecting all DGX systems' ConnectX-6 Compute Fabric HCAs.

These are shown in Figure 3.

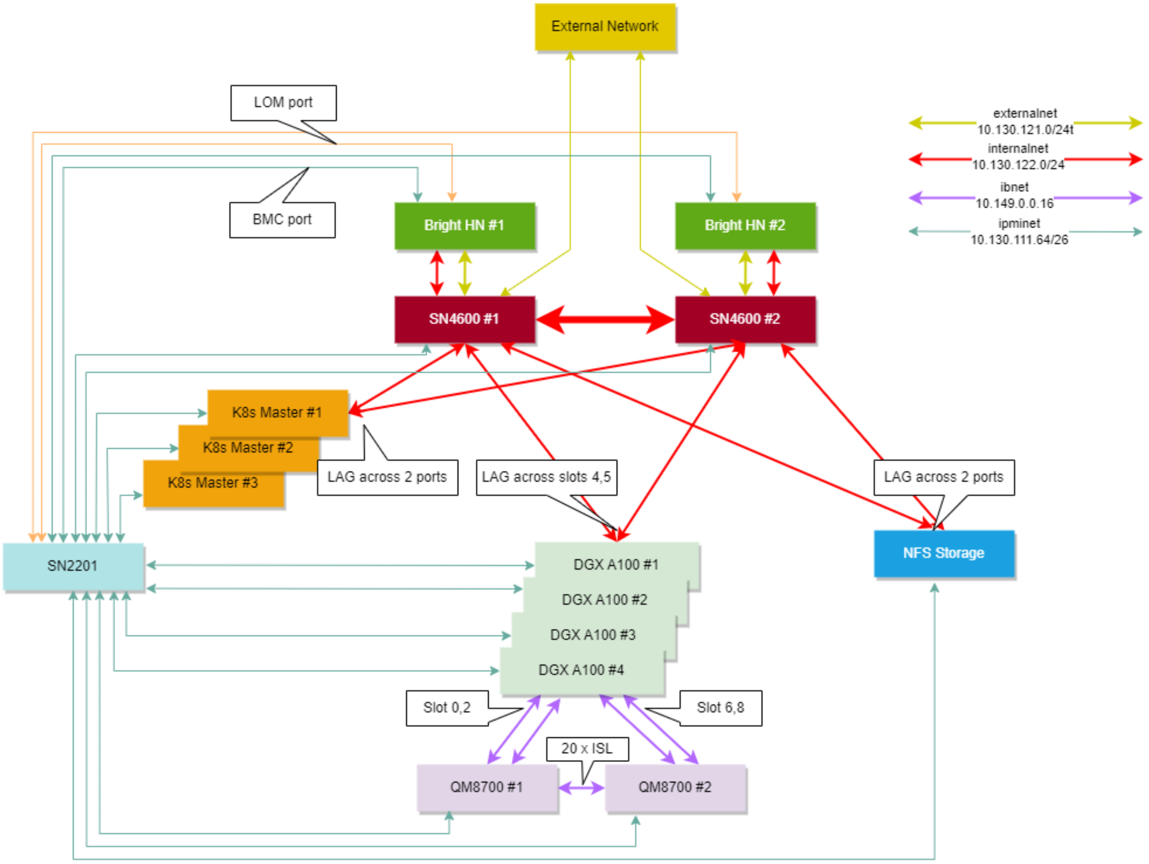


Figure 2. Network design and topology diagram

1.2.3 internalnet and externalnet

`internalnet` uses VLAN 122 and `externalnet` uses VLAN 121. Both VLANs are configured on the SN4600 switches, which are the backbone of the DGX BasePOD Ethernet networking. Each DGX system connects to the SN4600 switches with a bonded interface that consists of two physical interfaces; slot 4 bottom port (storage 4-2) and slot 5 right port (storage 5-2) as described in the Network Ports section of the *NVIDIA DGX A100 System User Guide.*

The K8s control plane nodes and the NFS storage device have a similar bonded interface configuration connected to SN4600 switches. Two SN4600 switches with Multi-chassis Link Aggregation (MLAG) provides the redundancy for DGX systems, K8s controller nodes, and other devices with bonded interfaces. Trunk mode is used to bond the interface with VLAN 122 as its native VLAN. Access mode is used on the port connected to the BCM head node. BGP protocols used between interfaces are described in Table 2. All connected subnets are redistributed into BGP.

Table 2. BGP protocols

Protocol	Description
BGP	Used as required for routing between switches
iBGP	Configured between the two SN4600s using the MLAG `peerlink.4094` interface
eBGP	Configured between the uplink (SN4600) and IPMI (SN2201) switches

1.2.4 ipminet

On the `ipminet` switches, the gateway for VLAN 111 is configured and all the ports connected to the end hosts are configured as access ports for VLAN 111. Each BCM head node requires two interfaces connected to the IPMI switch; the first for the IPMI interface of the host and the second to be used as HOST OS's direct access to the IPMI subnet. Uplinks are connected to TOR-01 and TOR-02 using unnumbered eBGP. All connected subnets are redistributed into BGP. IPMI switches can also be uplinked to a separate management network if required, rather than the TOR switches; still IPMI subnet route must be advertised to the in-band network so that BCM can control hosts using the IPMI network.

1.2.5 ibnet

For the `ibnet`, NICs on physical DGX `slot 0` and `2` are connected to QM8700-1 InfiniBand switch; and the NICs on physical DGX `slot 6` and `8` are connected to QM8700-2 InfiniBand switch. To manage the InfiniBand fabric, a subnet manager is required; one of the 8700 switches must be configured as the subnet manager.

The networking ports and their mapping are described in the Network Ports section of the *NVIDIA DGX A100 System User Guide*.

1.3 Software

Base Command Manager (BCM) is a key software component of DGX BasePOD. BCM is used to provision the OS on all hosts, deploy K8s, optionally deploy Jupyter, and provide monitoring and visibility of the cluster health.

An instance of BCM runs on a pair of head nodes in an High Availability (HA) configuration and is connected to all other nodes in the DGX BasePOD.

DGX systems within a DGX BasePOD have a DGX OS image installed by BCM. Similarly, the K8s control plane nodes are imaged by BCM with an Ubuntu LTS version equivalent to that of the DGX OS and the head nodes themselves.

1.3.1 Kubernetes (K8s)

K8s is an open-source platform for automating deployment, scaling, and operations of application containers across clusters of hosts. With K8s, it is possible to:

- Scale applications on the fly.
- Seamlessly update running services.
- Optimize hardware availability by using only the needed resources.

The cluster manager provides the administrator with the required packages, allows K8s to be set up, and manages and monitors K8s.

1.3.2 Jupyter (Optional)

BCM can optionally deploy and manage Jupyter, consisting of four major components and several extensions. The major components are: Jupyter Notebook, JupyterLab, JupyterHub, and Jupyter Enterprise Gateway.

These are the Jupyter extensions that BCM deploys:

- Template specialization extension—create a custom Jupyter kernel without editing text files.
- Job management extension—manage jobs from within the Jupyter interface.
- VNC extension—interact with the X display of the execution server (including the desktop) from within the Jupyter interface.
- K8s operators—Jupyter kernel, PostgreSQL, and Spark operators.
- Jupyter dev server—Proxy server that enables developing applications in alternative editors while the computational workload is proxied to their Jupyter notebook running on the cluster.

1.4 Storage

An NFS solution is required for a highly available (HA) BCM installation, and the required export path for that is described in this DGX BasePOD document. A DGX BasePOD typically also includes dedicated storage, but the configuration of that is outside the scope of this document. Contact the vendor of the storage solution being used for instructions on configuring the high performance- storage portions of a DGX BasePOD.

Chapter 2. Deployment

Deployment of a DGX BasePOD involves pre-setup, deployment, and use of BCM to provision the K8s cluster, and optionally deploy Jupyter.

1. Prepare the infrastructure.

   Physical installation should be completed before using this document, along with capturing information about the intended deployment in a site survey. Refer to Appendix A for the example site survey used by this document.

2. Configure the networking switches.

   Refer to Appendix B for the example configuration used by this document. Specifics on connecting to and configuring the switches can be found in their associated user guides.

3. Configure the NFS solution.

   a. As stated in Section 1.4, NFS configuration steps are not in scope for this document.

   b. This DGX BasePOD deployment uses the path `/var/nfs/general`, which is the NFS export path provided in Table 3 of the Site Survey.

   c. Use the following parameters for the NFS server export file `/etc/exports`

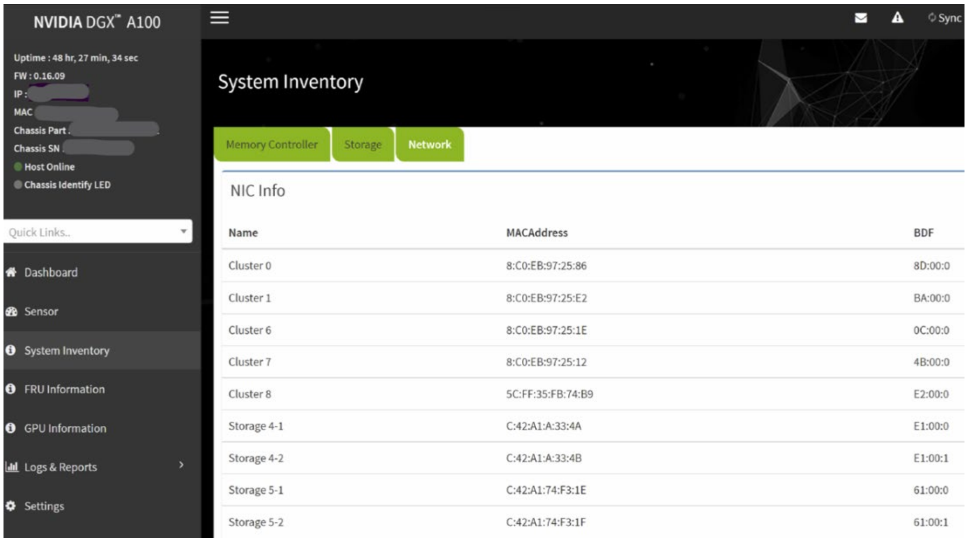
```
/var/nfs/general *(rw,sync,no_root_squash,no_subtree_check)
```

4. Set the DGX BIOS so that the DGX systems PXE boot by default.

   BCM requires DGX systems to PXE boot.

   d. Connect to the BMC of the DGX system.

e. In the `Network` tab of the `System Inventory` window, locate the MAC addresses for the `Storage 4-2` and `Storage 5-2` interfaces.



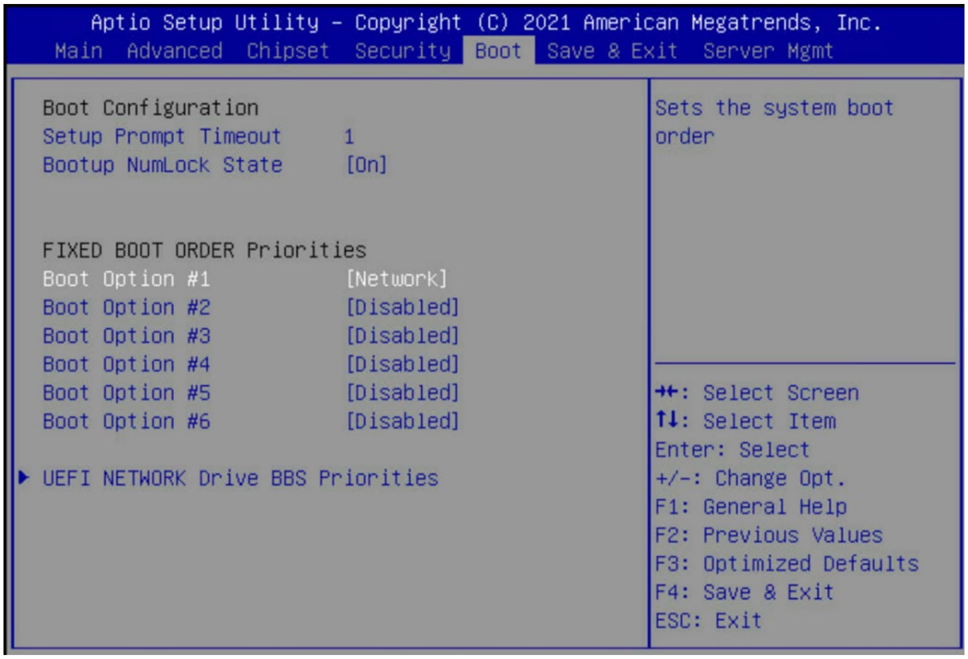
Uptime : 48 hr, 27 min, 34 sec
FW : 0.16.09
IP : [redacted]
MAC [redacted]
Chassis Part [redacted]
Chassis SN [redacted]
● Host Online
● Chassis Identify LED

System Inventory

NIC Info

Name	MACAddress	BDF
Cluster 0	8:C0:EB:97:25:86	8D:00:0
Cluster 1	8:C0:EB:97:25:E2	BA:00:0
Cluster 6	8:C0:EB:97:25:1E	0C:00:0
Cluster 7	8:C0:EB:97:25:12	4B:00:0
Cluster 8	5C:FF:35:FB:74:B9	E2:00:0
Storage 4-1	C:42:A1:A:33:4A	E1:00:0
Storage 4-2	C:42:A1:A:33:4B	E1:00:1
Storage 5-1	C:42:A1:74:F3:1E	61:00:0
Storage 5-2	C:42:A1:74:F3:1F	61:00:1

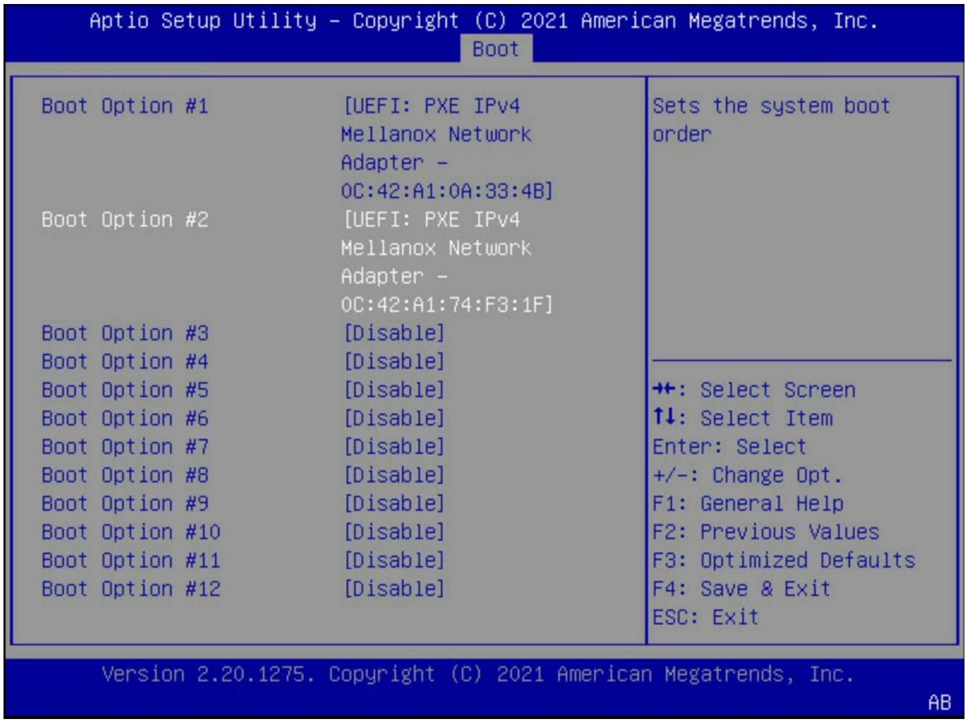
f. In the DGX A100 system BIOS, configure `Boot Option #1` to be `[NETWORK]`. Set other Boot devices to `[DISABLED]`.



```
        Aptio Setup Utility – Copyright (C) 2021 American Megatrends, Inc.
     Main  Advanced  Chipset  Security  Boot  Save & Exit  Server Mgmt

  Boot Configuration                         Sets the system boot
  Setup Prompt Timeout       1               order
  Bootup NumLock State       [On]


  FIXED BOOT ORDER Priorities
  Boot Option #1             [Network]
  Boot Option #2             [Disabled]
  Boot Option #3             [Disabled]
  Boot Option #4             [Disabled]      →←: Select Screen
  Boot Option #5             [Disabled]      ↑↓: Select Item
  Boot Option #6             [Disabled]      Enter: Select
                                             +/-: Change Opt.
▶ UEFI NETWORK Drive BBS Priorities          F1: General Help
                                             F2: Previous Values
                                             F3: Optimized Defaults
                                             F4: Save & Exit
                                             ESC: Exit
```

g. Disable PXE boot devices except for `Storage 4-2` and `Storage 5-2`.

   Set them to use IPv4.

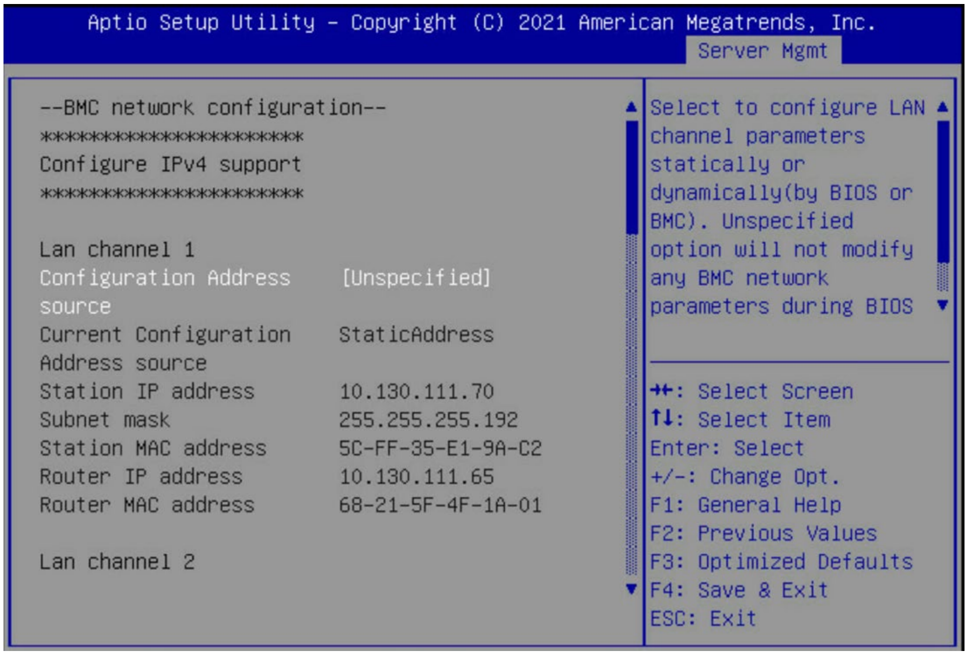


```
        Aptio Setup Utility - Copyright (C) 2021 American Megatrends, Inc.
                                     Boot

  Boot Option #1           [UEFI: PXE IPv4              Sets the system boot
                           Mellanox Network             order
                           Adapter -
                           0C:42:A1:0A:33:4B]
  Boot Option #2           [UEFI: PXE IPv4
                           Mellanox Network
                           Adapter -
                           0C:42:A1:74:F3:1F]
  Boot Option #3           [Disable]
  Boot Option #4           [Disable]
  Boot Option #5           [Disable]                    →←: Select Screen
  Boot Option #6           [Disable]                    ↑↓: Select Item
  Boot Option #7           [Disable]                    Enter: Select
  Boot Option #8           [Disable]                    +/-: Change Opt.
  Boot Option #9           [Disable]                    F1: General Help
  Boot Option #10          [Disable]                    F2: Previous Values
  Boot Option #11          [Disable]                    F3: Optimized Defaults
  Boot Option #12          [Disable]                    F4: Save & Exit
                                                        ESC: Exit

          Version 2.20.1275. Copyright (C) 2021 American Megatrends, Inc.
                                                                            AB
```

h. Configure a static IP address for the BMC.

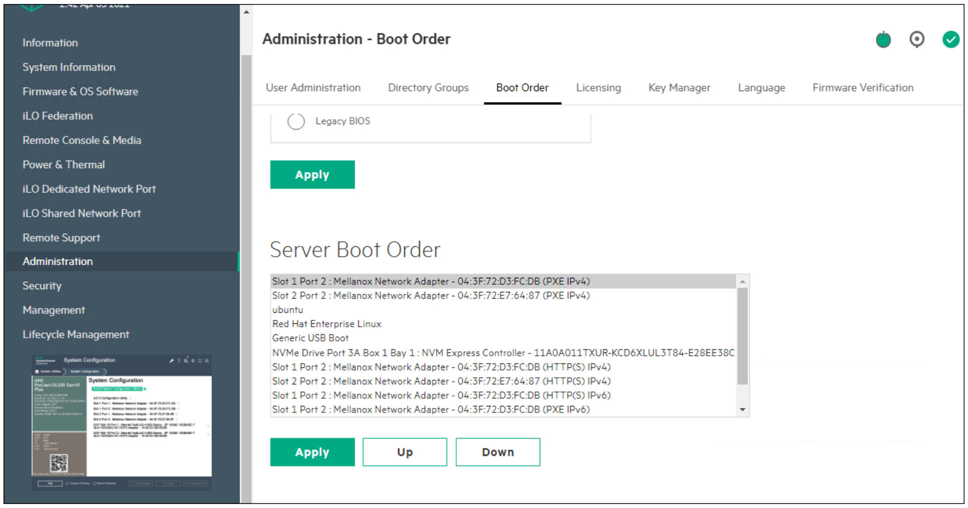
   Navigate to the `Server Mgmt` tab of the BIOS, enter the `BMC network configuration menu`, then set the IPv4 `Lan channel 1` Configuration Address Source option to `StaticAddress`, enter the IP address, subnet, and gateway/router information.

5. Ensure that the Network boot option is configured as the primary boot option for the K8s control plane nodes that are to be used for this cluster.



```
        Aptio Setup Utility - Copyright (C) 2021 American Megatrends, Inc.
                                                   Server Mgmt

  --BMC network configuration--                         Select to configure LAN
  **********************                                channel parameters
  Configure IPv4 support                                statically or
  **********************                                dynamically(by BIOS or
                                                        BMC). Unspecified
  Lan channel 1                                         option will not modify
  Configuration Address    [Unspecified]                any BMC network
  source                                                parameters during BIOS
  Current Configuration    StaticAddress
  Address source
  Station IP address       10.130.111.70                →←: Select Screen
  Subnet mask              255.255.255.192              ↑↓: Select Item
  Station MAC address      5C-FF-35-E1-9A-C2            Enter: Select
  Router IP address        10.130.111.65                +/-: Change Opt.
  Router MAC address       68-21-5F-4F-1A-01            F1: General Help
                                                        F2: Previous Values
  Lan channel 2                                         F3: Optimized Defaults
                                                        F4: Save & Exit
                                                        ESC: Exit
```

This is an example of a system that will boot from the network with `Slot 1 Port 2` and `Slot 2 Port 2`.



Administration - Boot Order

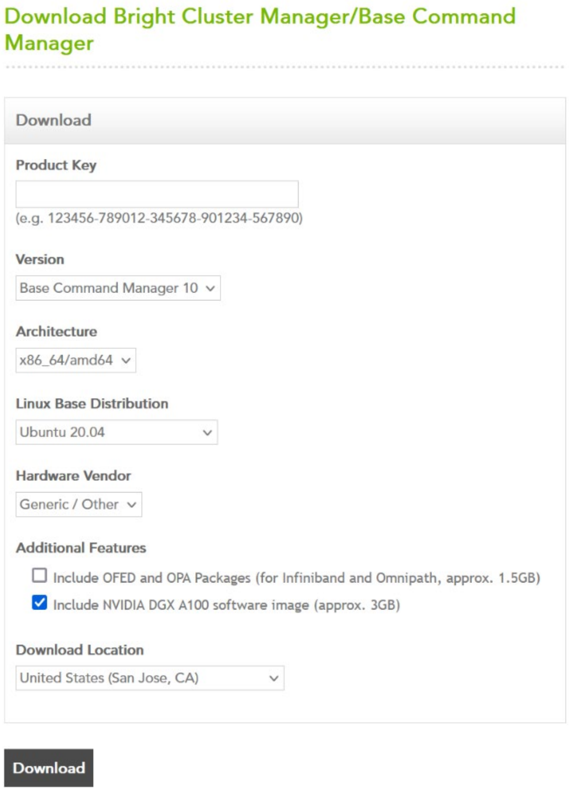
○ Legacy BIOS

Server Boot Order

Slot 1 Port 2 : Mellanox Network Adapter - 04:3F:72:D3:FC:DB (PXE IPv4)
Slot 2 Port 2 : Mellanox Network Adapter - 04:3F:72:E7:64:87 (PXE IPv4)
ubuntu
Red Hat Enterprise Linux
Generic USB Boot
NVMe Drive Port 3A Box 1 Bay 1 : NVM Express Controller - 11A0A011TXUR-KCD6XLUL3T84-E28EE38C
Slot 1 Port 2 : Mellanox Network Adapter - 04:3F:72:D3:FC:DB (HTTP(S) IPv4)
Slot 2 Port 2 : Mellanox Network Adapter - 04:3F:72:E7:64:87 (HTTP(S) IPv4)
Slot 1 Port 2 : Mellanox Network Adapter - 04:3F:72:D3:FC:DB (HTTP(S) IPv6)
Slot 1 Port 2 : Mellanox Network Adapter - 04:3F:72:D3:FC:DB (PXE IPv6)

6. Download a BCM ISO from the Bright Cluster Manager/Base Command Manager download site.

   Select Base Command Manager 10, Ubuntu 20.04, and check the `Include NVIDIA DGX A100 Software image` checkbox.



Download Bright Cluster Manager/Base Command Manager

Product Key

(e.g. 123456-789012-345678-901234-567890)

Version: Base Command Manager 10

Architecture: x86_64/amd64

Linux Base Distribution: Ubuntu 20.04

Hardware Vendor: Generic / Other

Additional Features

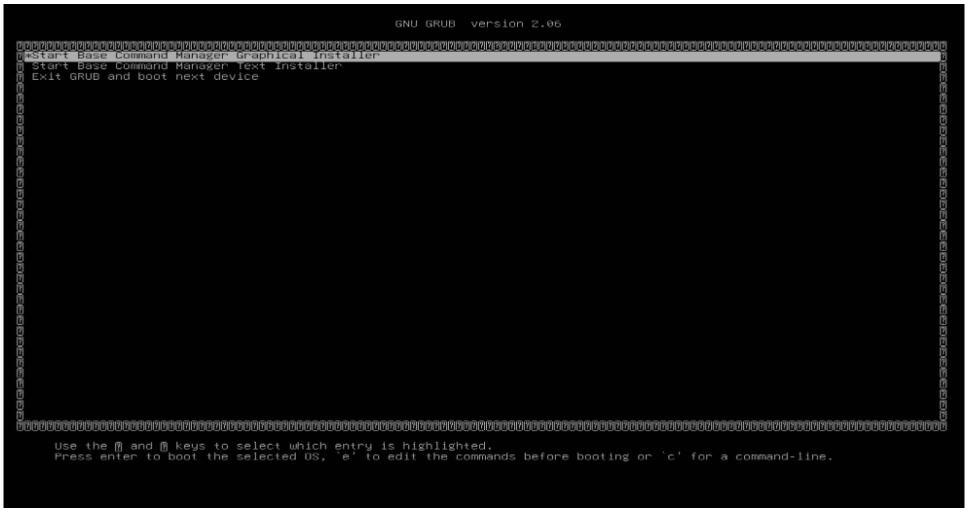
☐ Include OFED and OPA Packages (for Infiniband and Omnipath, approx. 1.5GB)
☑ Include NVIDIA DGX A100 software image (approx. 3GB)

Download Location: United States (San Jose, CA)

7. Burn the ISO to a DVD or to a bootable USB device.

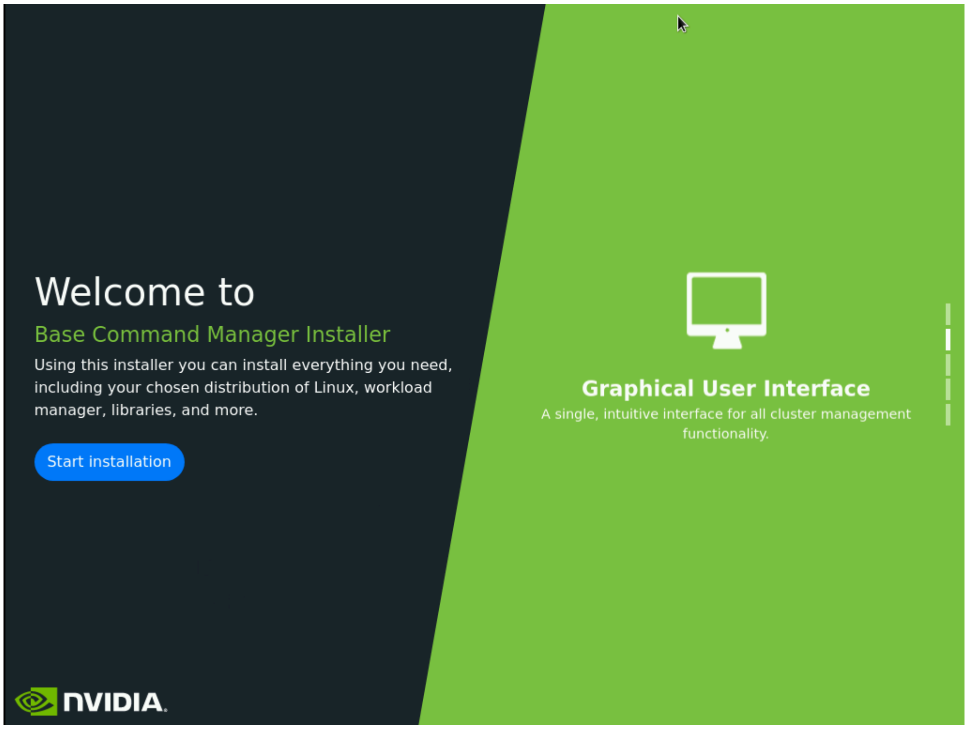
   It can also be mounted as virtual media and installed using the BMC. The specific mechanism for the latter will vary by vendor.

8. Ensure that the BIOS of the target head node is configured in UEFI mode and that its boot order is configured to boot the media containing the BCM installer image.
9. Boot the installation media.
10. At the grub menu, choose `Start Base Command Manager Graphical Installer`.



```
                              GNU GRUB  version 2.06

 *Start Base Command Manager Graphical Installer
  Start Base Command Manager Text Installer
  Exit GRUB and boot next device

     Use the ▯ and ▯ keys to select which entry is highlighted.
     Press enter to boot the selected OS, `e' to edit the commands before booting or `c' for a command-line.
```

11. Select `Start installation` on the splash screen.



Welcome to

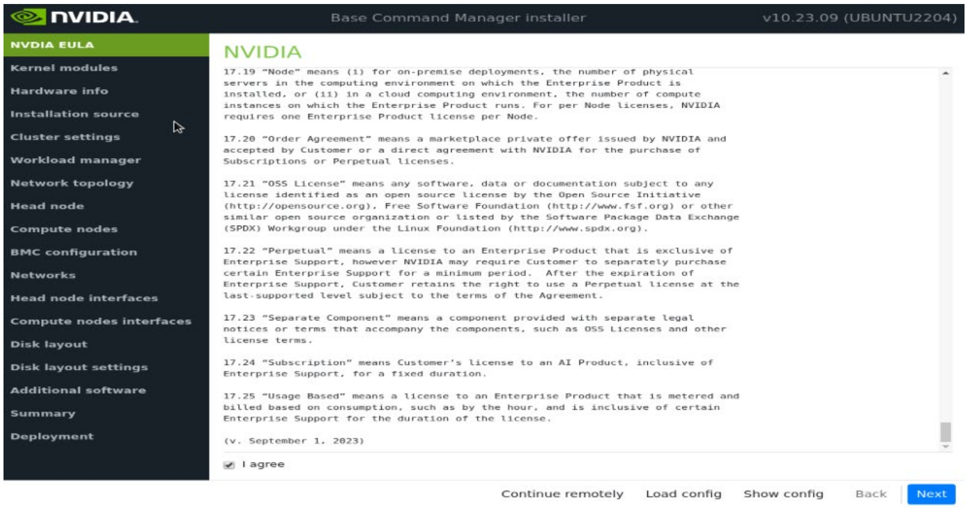
Base Command Manager Installer

Using this installer you can install everything you need, including your chosen distribution of Linux, workload manager, libraries, and more.

Graphical User Interface

A single, intuitive interface for all cluster management functionality.

12. Accept the terms of the NVIDIA EULA by checking `I agree` and then select `Next`.



NVIDIA

17.19 “Node” means (i) for on-premise deployments, the number of physical servers in the computing environment on which the Enterprise Product is installed, or (ii) in a cloud computing environment, the number of compute instances on which the Enterprise Product runs. For per Node licenses, NVIDIA requires one Enterprise Product license per Node.

17.20 “Order Agreement” means a marketplace private offer issued by NVIDIA and accepted by Customer or a direct agreement with NVIDIA for the purchase of Subscriptions or Perpetual licenses.

17.21 “OSS License” means any software, data or documentation subject to any license identified as an open source license by the Open Source Initiative (http://opensource.org), Free Software Foundation (http://www.fsf.org) or other similar open source organization or listed by the Software Package Data Exchange (SPDX) Workgroup under the Linux Foundation (http://www.spdx.org).

17.22 “Perpetual” means a license to an Enterprise Product that is exclusive of Enterprise Support, however NVIDIA may require Customer to separately purchase certain Enterprise Support for a minimum period. After the expiration of Enterprise Support, Customer retains the right to use a Perpetual license at the last-supported level subject to the terms of the Agreement.

17.23 “Separate Component” means a component provided with separate legal notices or terms that accompany the components, such as OSS Licenses and other license terms.

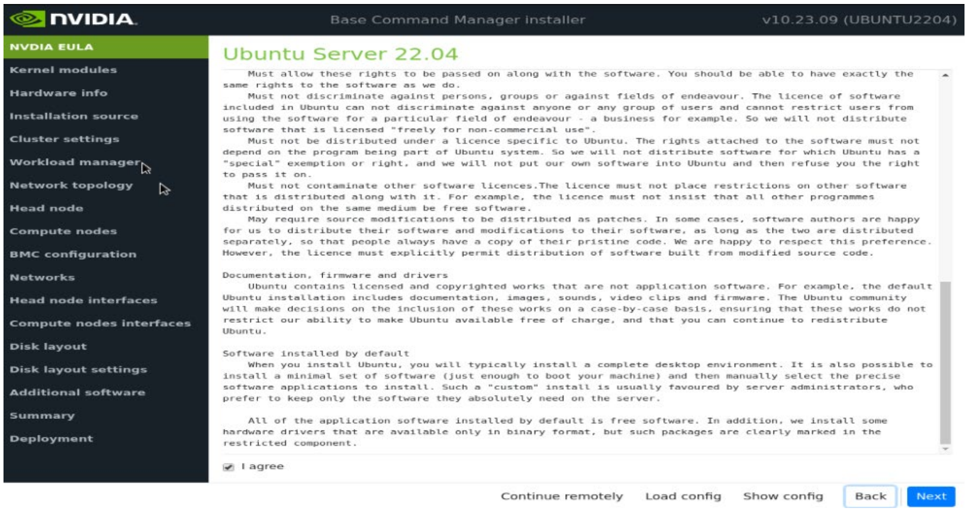
17.24 “Subscription” means Customer's license to an AI Product, inclusive of Enterprise Support, for a fixed duration.

17.25 “Usage Based” means a license to an Enterprise Product that is metered and billed based on consumption, such as by the hour, and is inclusive of certain Enterprise Support for the duration of the license.

(v. September 1, 2023)

☑ I agree

13. Accept the terms of the Ubuntu Server UELA by checking `I agree` and then select `Next`.



Ubuntu Server 22.04

Must allow these rights to be passed on along with the software. You should be able to have exactly the same rights to the software as we do.

Must not discriminate against persons, groups or against fields of endeavour. The licence of software included in Ubuntu can not discriminate against anyone or any group of users and cannot restrict users from using the software for a particular field of endeavour - a business for example. So we will not distribute software that is licensed "freely for non-commercial use".

Must not be distributed under a licence specific to Ubuntu. The rights attached to the software must not depend on the program being part of Ubuntu system. So we will not distribute software for which Ubuntu has a "special" exemption or right, and we will not put our own software into Ubuntu and then refuse you the right to pass it on.

Must not contaminate other software licences.The licence must not place restrictions on other software that is distributed along with it. For example, the licence must not insist that all other programmes distributed on the same medium be free software.

May require source modifications to be distributed as patches. In some cases, software authors are happy for us to distribute their software and modifications to their software, as long as the two are distributed separately, so that people always have a copy of their pristine code. We are happy to respect this preference. However, the licence must explicitly permit distribution of software built from modified source code.

Documentation, firmware and drivers

Ubuntu contains licensed and copyrighted works that are not application software. For example, the default Ubuntu installation includes documentation, images, sounds, video clips and firmware. The Ubuntu community will make decisions on the inclusion of these works on a case-by-case basis, ensuring that these works do not restrict our ability to make Ubuntu available free of charge, and that you can continue to redistribute Ubuntu.

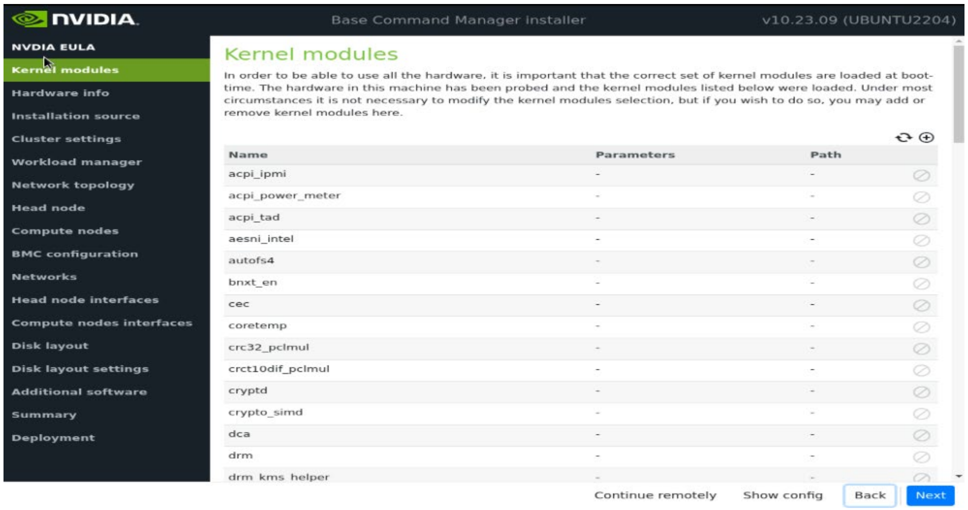
Software installed by default

When you install Ubuntu, you will typically install a complete desktop environment. It is also possible to install a minimal set of software (just enough to boot your machine) and then manually select the precise software applications to install. Such a "custom" install is usually favoured by server administrators, who prefer to keep only the software they absolutely need on the server.

All of the application software installed by default is free software. In addition, we install some hardware drivers that are available only in binary format, but such packages are clearly marked in the restricted component.

☑ I agree

14. Unless instructed otherwise, select `Next` without modifying the kernel modules to be loaded at boot time.



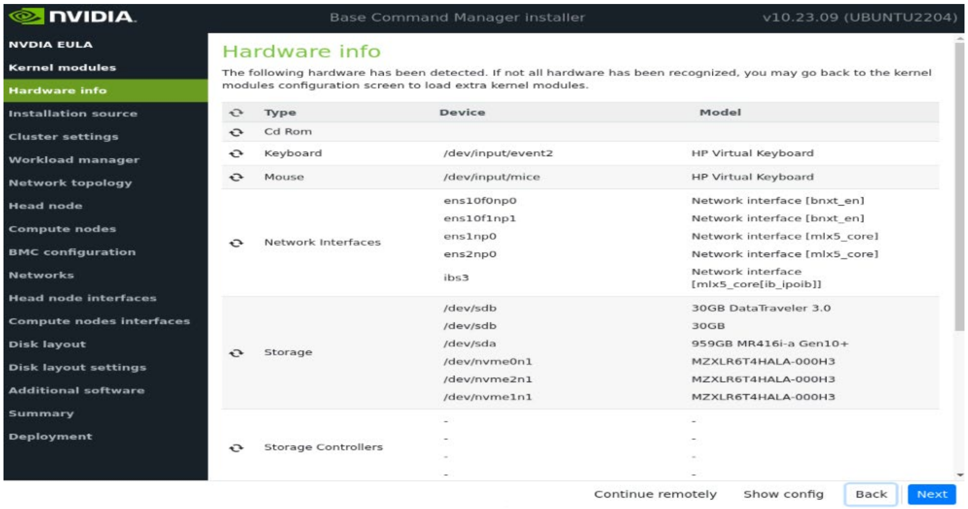
Kernel modules

In order to be able to use all the hardware, it is important that the correct set of kernel modules are loaded at boot-time. The hardware in this machine has been probed and the kernel modules listed below were loaded. Under most circumstances it is not necessary to modify the kernel modules selection, but if you wish to do so, you may add or remove kernel modules here.

Name	Parameters	Path
acpi_ipmi	-	-
acpi_power_meter	-	-
acpi_tad	-	-
aesni_intel	-	-
autofs4	-	-
bnxt_en	-	-
cec	-	-
coretemp	-	-
crc32_pclmul	-	-
crct10dif_pclmul	-	-
cryptd	-	-
crypto_simd	-	-
dca	-	-
drm	-	-
drm_kms_helper	-	-

15. Verify the `Hardware info` is correct and then select `Next`.

    For example, that the target storage device and the cabled host network interfaces are present (in this case three NVMe drives are the target storage device, and `ens1np0` and `ens2np01` are the cabled host network interfaces).



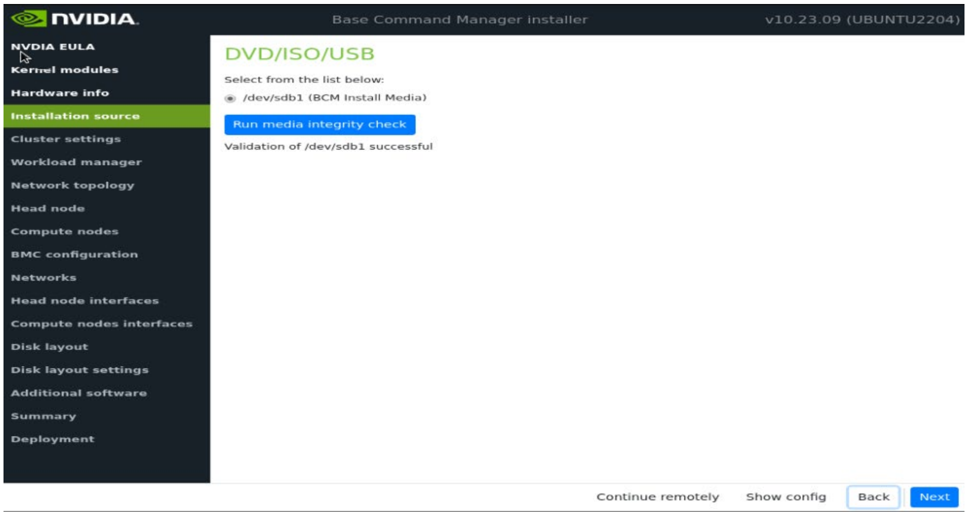
Hardware info

The following hardware has been detected. If not all hardware has been recognized, you may go back to the kernel modules configuration screen to load extra kernel modules.

Type	Device	Model
Cd Rom		
Keyboard	/dev/input/event2	HP Virtual Keyboard
Mouse	/dev/input/mice	HP Virtual Keyboard
Network Interfaces	ens10f0np0 ens10f1np1 ens1np0 ens2np0 ibs3	Network interface [bnxt_en] Network interface [bnxt_en] Network interface [mlx5_core] Network interface [mlx5_core] Network interface [mlx5_core[ib_ipoib]]
Storage	/dev/sdb /dev/sdb /dev/sda /dev/nvme0n1 /dev/nvme2n1 /dev/nvme1n1	30GB DataTraveler 3.0 30GB 959GB MR416i-a Gen10+ MZXLR6T4HALA-000H3 MZXLR6T4HALA-000H3 MZXLR6T4HALA-000H3
Storage Controllers	- - - -	- - - -

17. On the Installation source screen, choose the appropriate source and then select Next.

    Running a media integrity check is optional.



DVD/ISO/USB

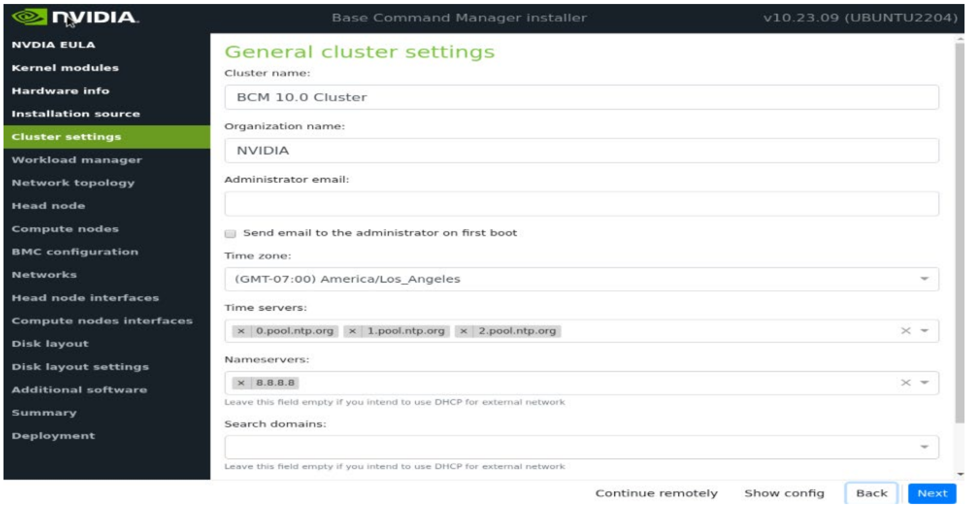
Select from the list below:

● /dev/sdb1 (BCM Install Media)

Validation of /dev/sdb1 successful

18. On the `Cluster settings` screen, enter the required information and then select `Next`.

    Enter information from the site survey. An example site survey is in Appendix A.



General cluster settings

Cluster name: BCM 10.0 Cluster

Organization name: NVIDIA

Administrator email:

□ Send email to the administrator on first boot

Time zone: (GMT-07:00) America/Los_Angeles

Time servers: × 0.pool.ntp.org × 1.pool.ntp.org × 2.pool.ntp.org

Nameservers: × 8.8.8.8

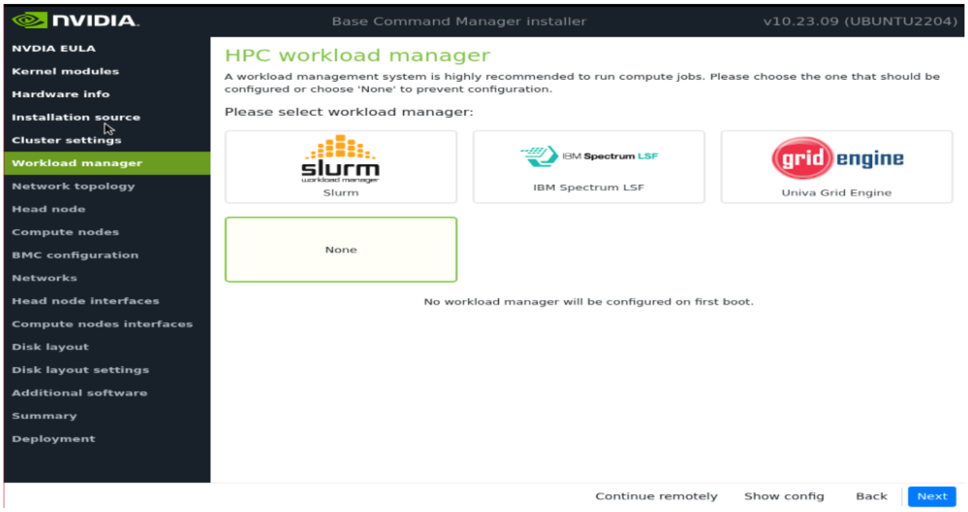
Leave this field empty if you intend to use DHCP for external network

Search domains:

Leave this field empty if you intend to use DHCP for external network

19. On the `Workload manager` screen, choose `None` and then select `Next`.

    After head node installation, K8s will be deployed for container orchestration.



HPC workload manager

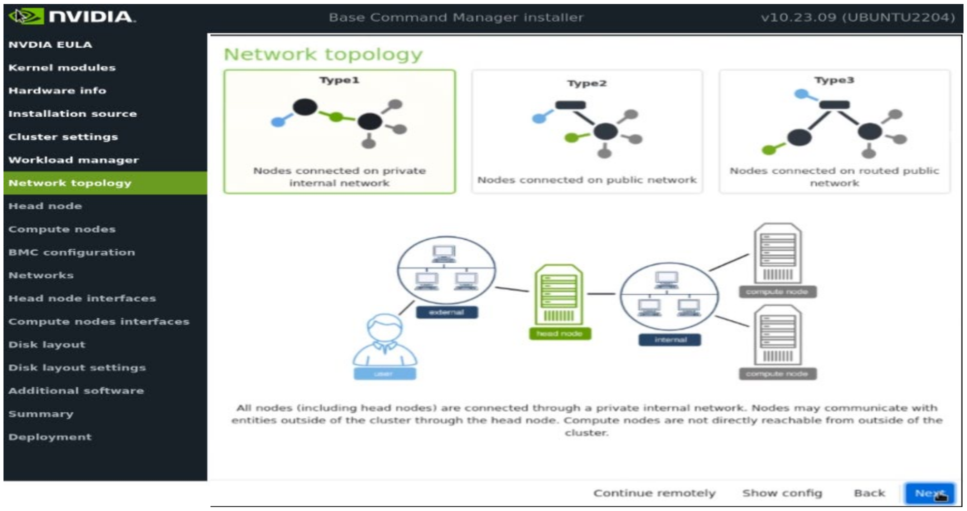
A workload management system is highly recommended to run compute jobs. Please choose the one that should be configured or choose 'None' to prevent configuration.

Please select workload manager:

No workload manager will be configured on first boot.

20. On the `Network topology` screen, choose the network topology Type 1 and then select `Next`.

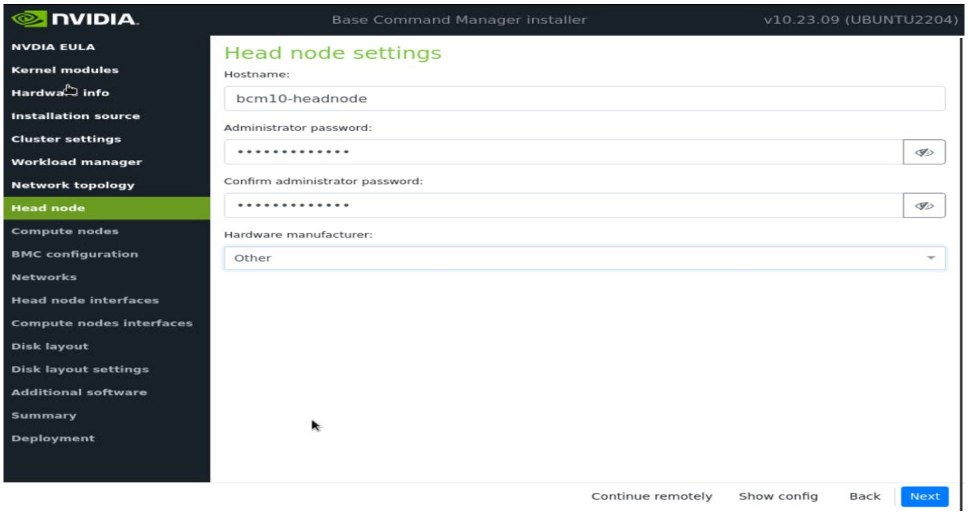
    In a DGX BasePOD architecture, the cluster nodes are connected to the head node over the internal network, with the head node serving as their default gateway.



Network topology

All nodes (including head nodes) are connected through a private internal network. Nodes may communicate with entities outside of the cluster through the head node. Compute nodes are not directly reachable from outside of the cluster.

21. On the `Head node` screen, enter the `Hostname`, `Administrator password`, choose `Other` for `Hardware manufacturer`, and then select `Next`.



Head node settings

Hostname: bcm10-headnode

Administrator password: •••••••••••••

Confirm administrator password: •••••••••••••

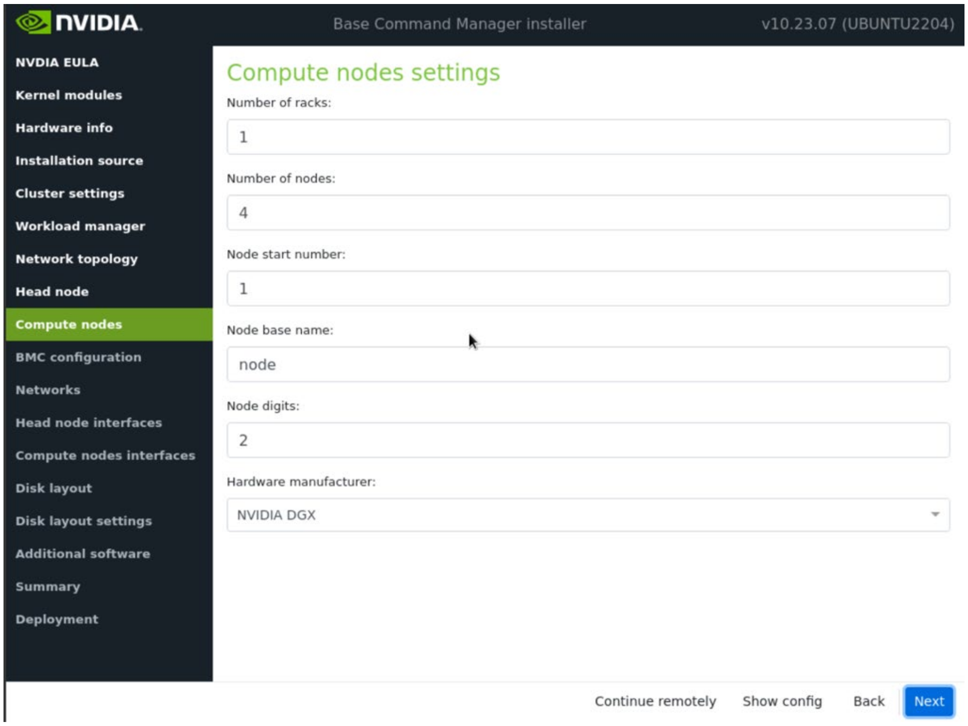
Hardware manufacturer: Other

22. Configure the `Compute nodes` screen.

    Set the `Number of nodes` to `4`.

    Set `Node digits` to `2`.

    Set `Hardware manufacturer` to `NVIDIA DGX`.



Compute nodes settings

Number of racks: 1

Number of nodes: 4

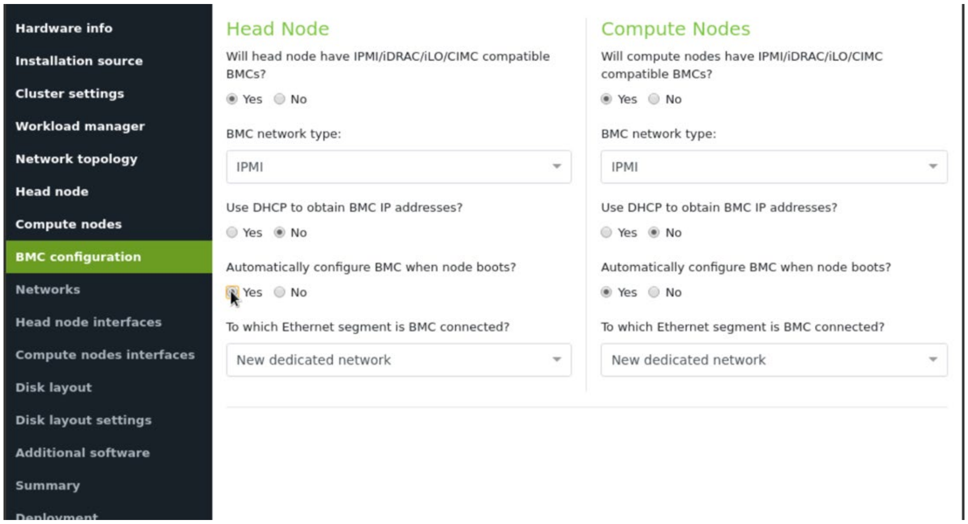
Node start number: 1

Node base name: node

Node digits: 2

Hardware manufacturer: NVIDIA DGX

23. Configure `BMC configuration` screen.
    i. Select `Yes` for both the `Head Node` and the `Compute Nodes`.
    j. Select `IPMI` from the `BMC network type` select lists for both the `Head Node` and the `Compute Nodes`.
    k. Select `No` to the DHCP question for both node types.
    l. Select `Yes` for `Automatically configure BMC when node boots?`.
    m. Select `New dedicated network` from the `To which Ethernet segment is BMC connected?` list.



Head Node

Will head node have IPMI/iDRAC/iLO/CIMC compatible BMCs?
● Yes ○ No

BMC network type:
IPMI

Use DHCP to obtain BMC IP addresses?
○ Yes ● No

Automatically configure BMC when node boots?
● Yes ○ No

To which Ethernet segment is BMC connected?
New dedicated network

Compute Nodes

Will compute nodes have IPMI/iDRAC/iLO/CIMC compatible BMCs?
● Yes ○ No

BMC network type:
IPMI

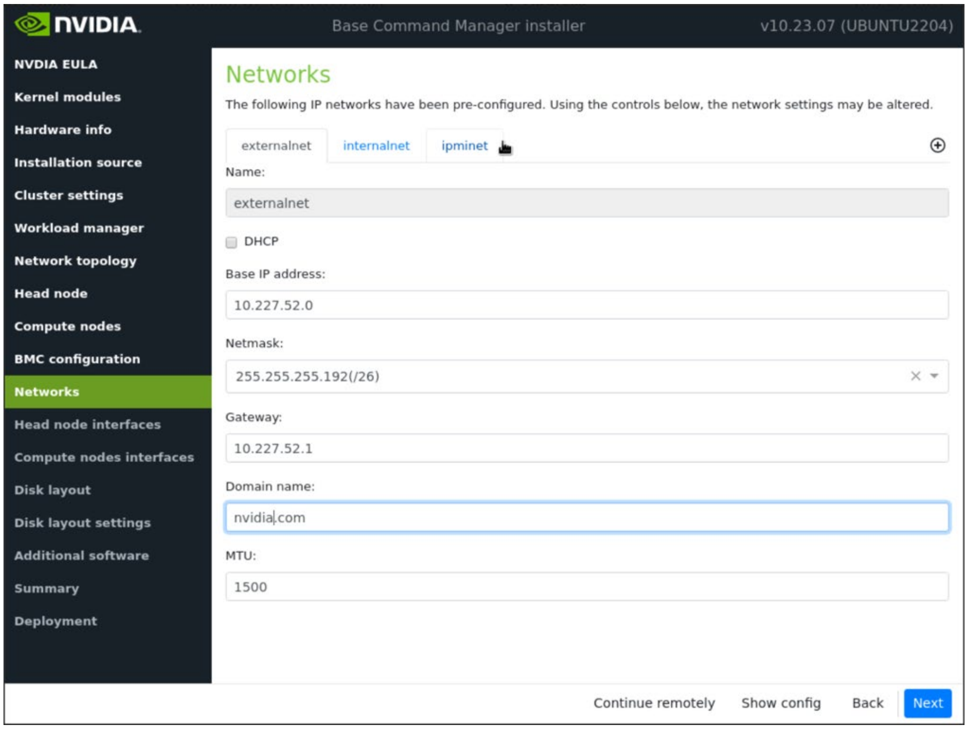
Use DHCP to obtain BMC IP addresses?
○ Yes ● No

Automatically configure BMC when node boots?
● Yes ○ No

To which Ethernet segment is BMC connected?
New dedicated network

24. Configure the `Networks` screens.
    n. `externalnet`

Set the `Base IP address`, `Netmask`, `Gateway`, and `Domain name` according to the site survey.



Networks

The following IP networks have been pre-configured. Using the controls below, the network settings may be altered.

Name: externalnet

□ DHCP

Base IP address: 10.227.52.0

Netmask: 255.255.255.192(/26)

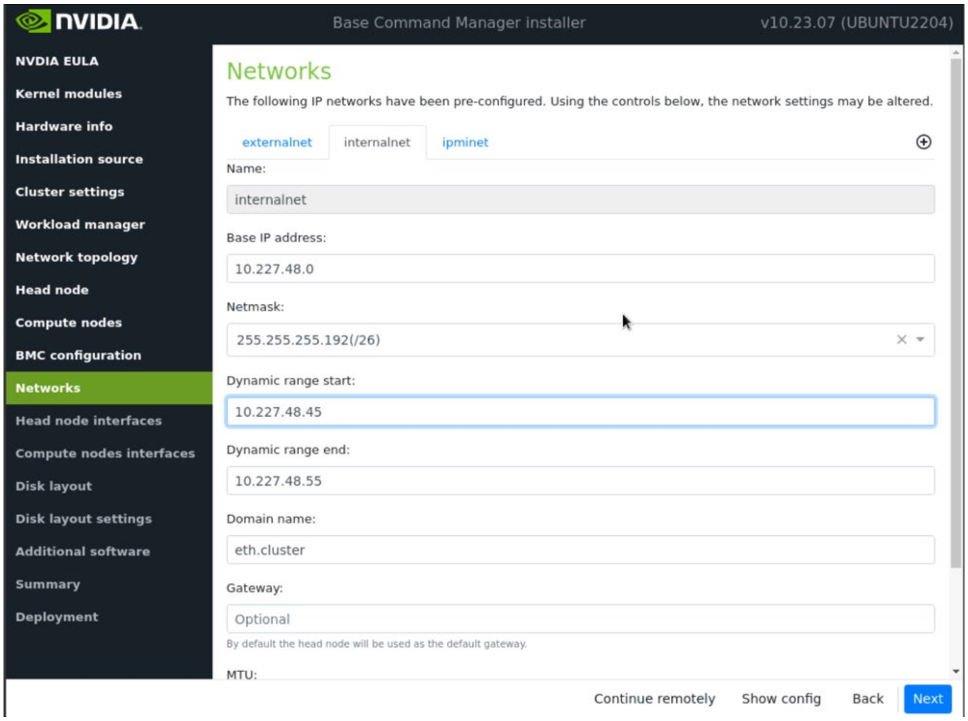
Gateway: 10.227.52.1

Domain name: nvidia.com

MTU: 1500

o. `internalnet`

Set the `Base IP address` and `Netmask` according to the site survey.



Networks

The following IP networks have been pre-configured. Using the controls below, the network settings may be altered.

Name: internalnet

Base IP address: 10.227.48.0

Netmask: 255.255.255.192(/26)

Dynamic range start: 10.227.48.45

Dynamic range end: 10.227.48.55

Domain name: eth.cluster

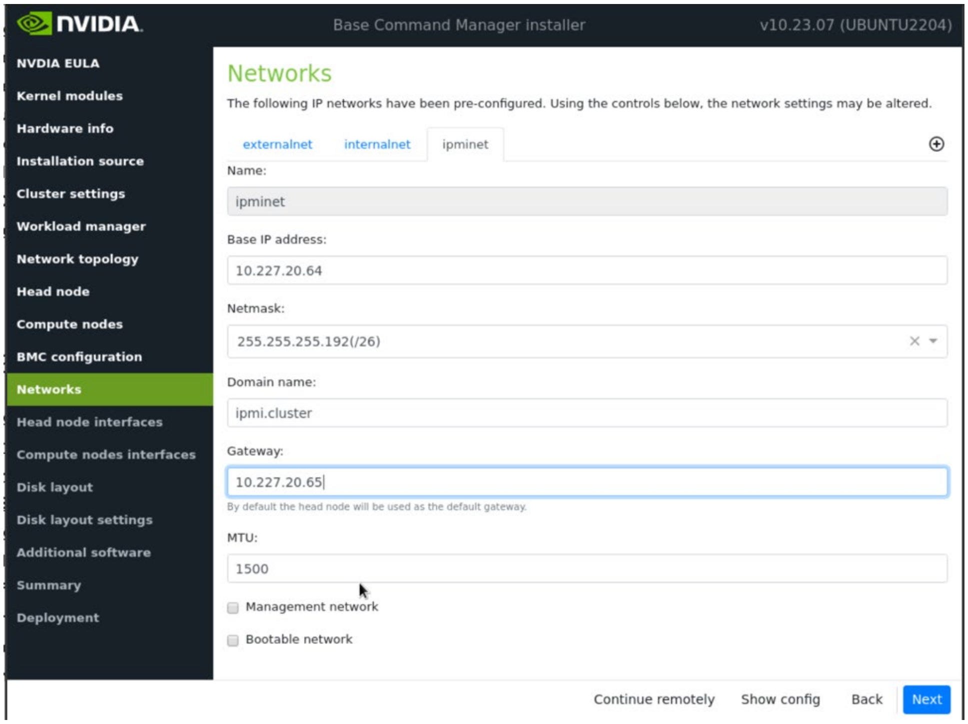
Gateway: Optional

By default the head node will be used as the default gateway.

MTU:

p. `ipminet`

Set the `Base IP address`, `Netmask`, and `Gateway` according to the site survey.



Networks

The following IP networks have been pre-configured. Using the controls below, the network settings may be altered.

Name: ipminet

Base IP address: 10.227.20.64

Netmask: 255.255.255.192(/26)

Domain name: ipmi.cluster

Gateway: 10.227.20.65

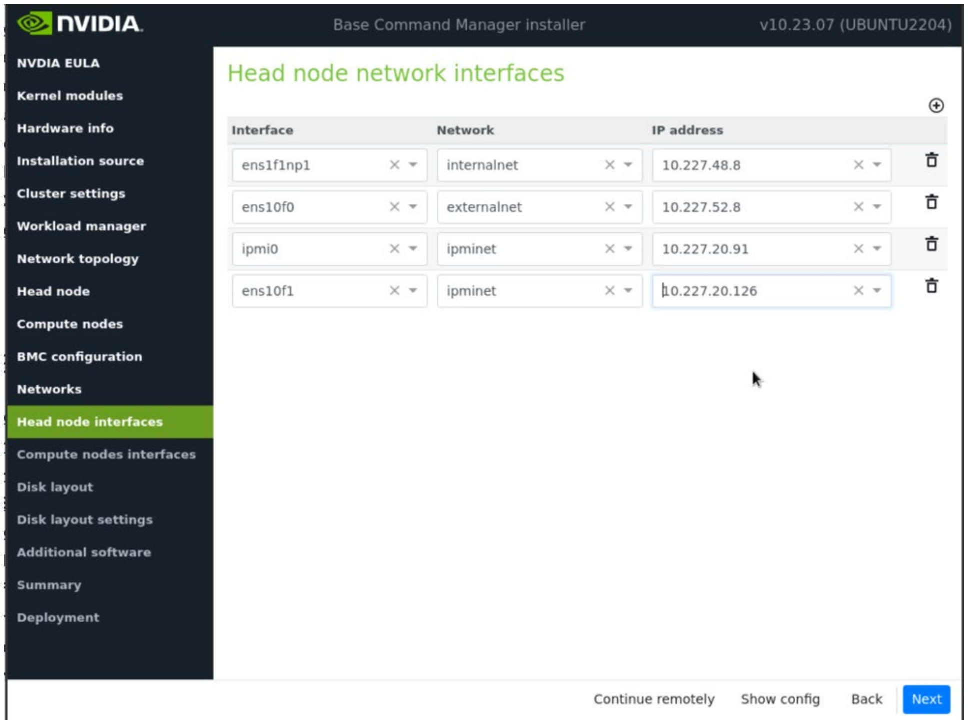
By default the head node will be used as the default gateway.

MTU: 1500

□ Management network

□ Bootable network

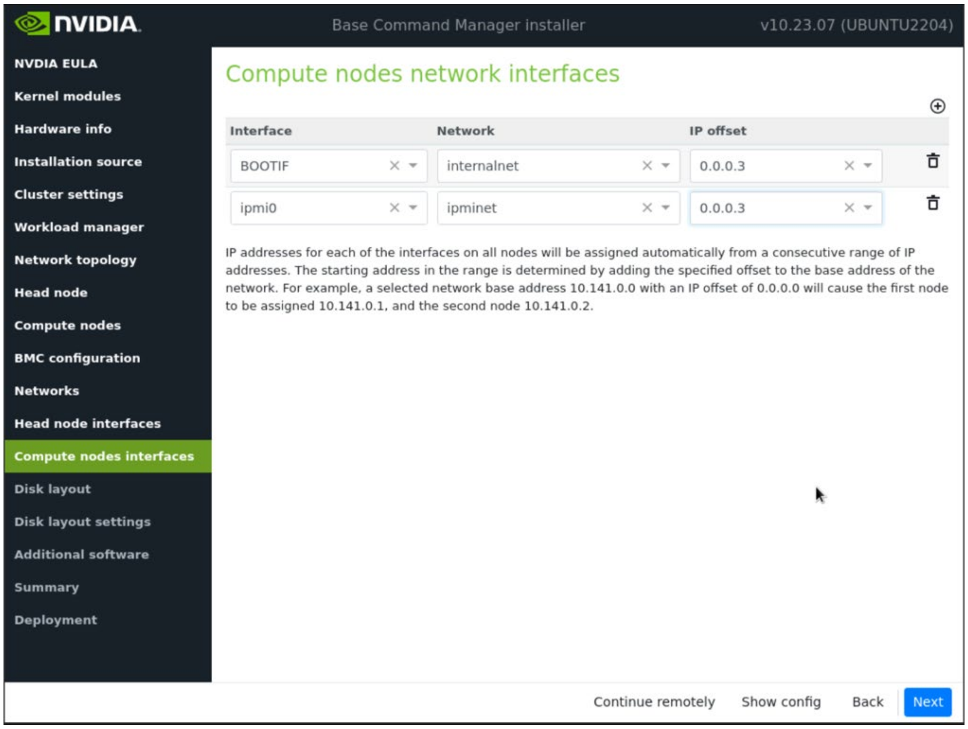
25. Configure the `Head node interfaces` according to the site survey and then select `Next`.



Head node network interfaces

Interface	Network	IP address
ens1f1np1	internalnet	10.227.48.8
ens10f0	externalnet	10.227.52.8
ipmi0	ipminet	10.227.20.91
ens10f1	ipminet	10.227.20.126

26. Configure the offset for `BOOTIF` and `ipmi0` to `0.0.0.3` on the `Compute nodes network interfaces` screen and then select `Next`.

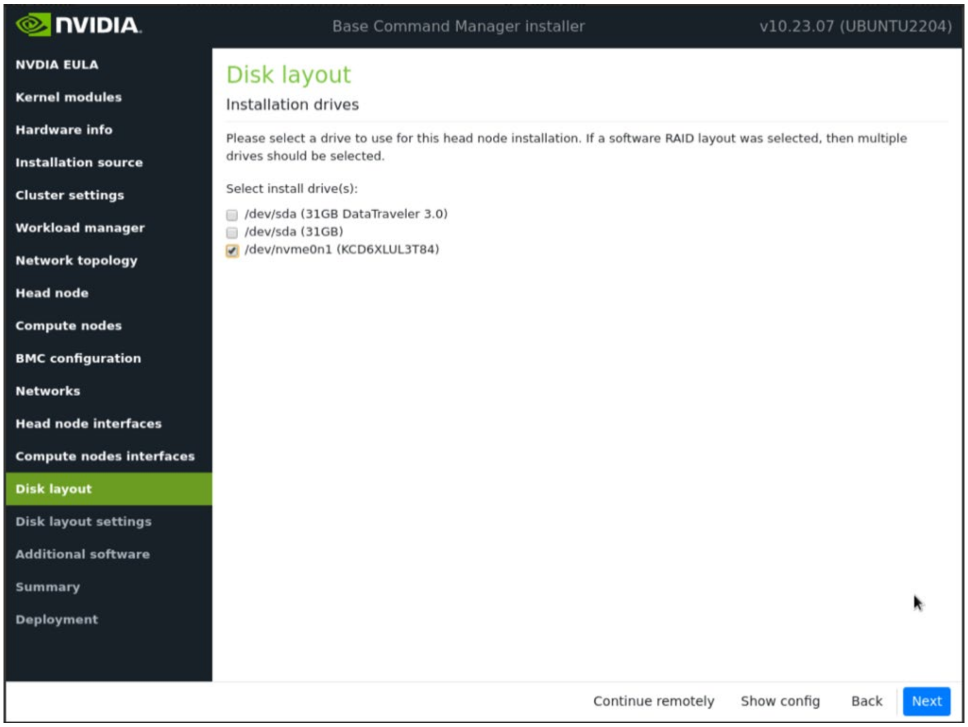


Compute nodes network interfaces

Interface	Network	IP offset
BOOTIF	internalnet	0.0.0.3
ipmi0	ipminet	0.0.0.3

IP addresses for each of the interfaces on all nodes will be assigned automatically from a consecutive range of IP addresses. The starting address in the range is determined by adding the specified offset to the base address of the network. For example, a selected network base address 10.141.0.0 with an IP offset of 0.0.0.0 will cause the first node to be assigned 10.141.0.1, and the second node 10.141.0.2.

27. Configure the installation drive(s) on the `Disk layout` screen and then select `Next`.



Disk layout

Installation drives

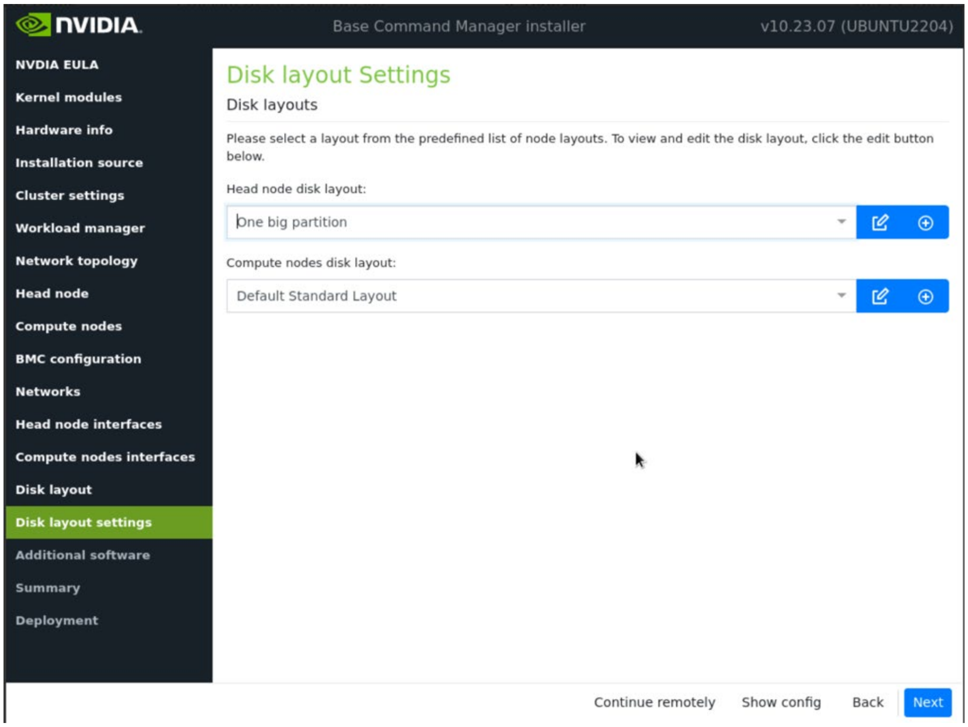
Please select a drive to use for this head node installation. If a software RAID layout was selected, then multiple drives should be selected.

Select install drive(s):

☐ /dev/sda (31GB DataTraveler 3.0)
☐ /dev/sda (31GB)
☑ /dev/nvme0n1 (KCD6XLUL3T84)

28. Configure the `Disk layout Settings` screen and then select `Next`.

Set the `Head node disk layout` to `One big partition` and the `Compute nodes disk layout` to `Default Standard Layout`.



Disk layout Settings

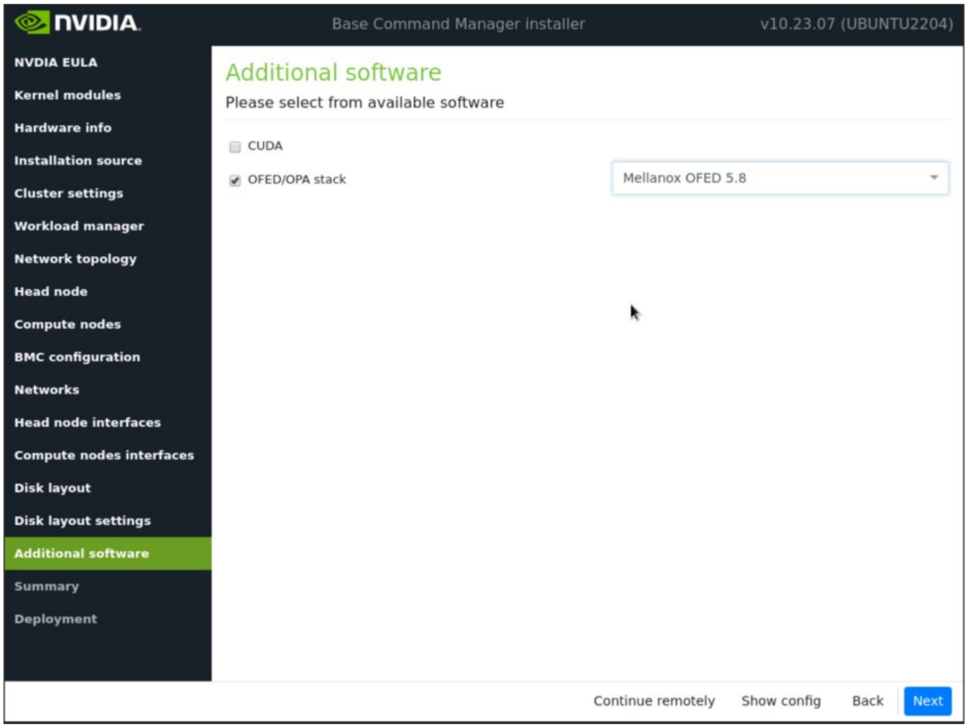
Disk layouts

Please select a layout from the predefined list of node layouts. To view and edit the disk layout, click the edit button below.

Head node disk layout: One big partition

Compute nodes disk layout: Default Standard Layout

29. Check the `OFED/OPA stack` box and choose `Mellanox OFED 5.8` on the `Additional software` screen and then select `Next`.



Additional software

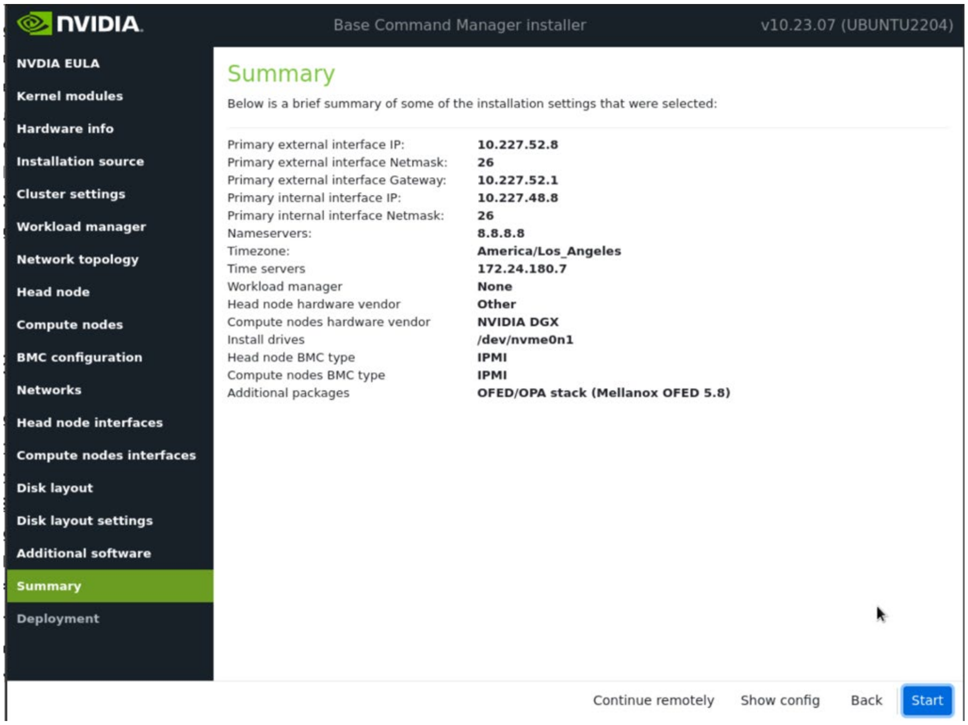
Please select from available software

☐ CUDA

☑ OFED/OPA stack — Mellanox OFED 5.8

30. Review the information on the `Summary` screen.

The Summary screen provides an opportunity to confirm the head node and basic cluster configuration before deployment begins. If anything does not match expectations, use the `Back` button to navigate to the appropriate screen to correct any mistake.



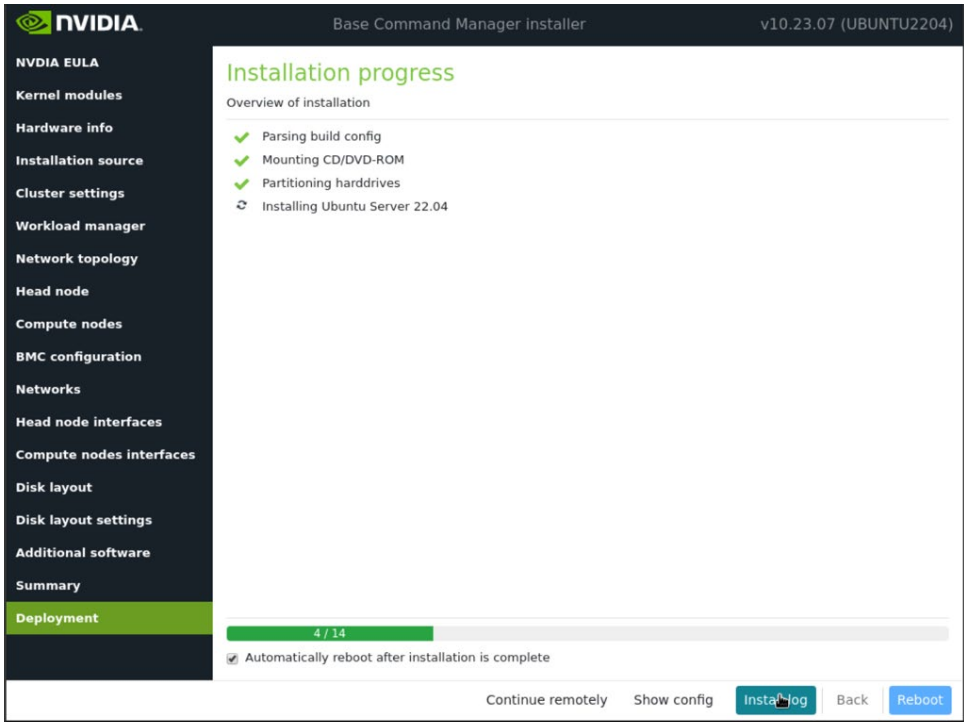
Summary

Below is a brief summary of some of the installation settings that were selected:

Primary external interface IP:	10.227.52.8
Primary external interface Netmask:	26
Primary external interface Gateway:	10.227.52.1
Primary internal interface IP:	10.227.48.8
Primary internal interface Netmask:	26
Nameservers:	8.8.8.8
Timezone:	America/Los_Angeles
Time servers	172.24.180.7
Workload manager	None
Head node hardware vendor	Other
Compute nodes hardware vendor	NVIDIA DGX
Install drives	/dev/nvme0n1
Head node BMC type	IPMI
Compute nodes BMC type	IPMI
Additional packages	OFED/OPA stack (Mellanox OFED 5.8)

31. Configure the `Deployment` screen and then select `Reboot`.

    Check the `Automatically reboot after installation is complete` checkbox to reboot the host upon successful completion of the deployment. Select `Install log` to see a summary of the installation.



Installation progress

Overview of installation

✓ Parsing build config
✓ Mounting CD/DVD-ROM
✓ Partitioning harddrives
⟳ Installing Ubuntu Server 22.04

☑ Automatically reboot after installation is complete

2.1 Cluster Configuration

1. Log in to the BCM head node assigned to `externalnet`.

```
ssh <externalnet>
```

2. Install the cluster license by running the `request-license` command.

   Because HA is used, specify the MAC address of the first NIC of the secondary head node so that it can also serve the BCM licenses in the event of a failover.

   This example is for a head node with Internet access. For air-gapped clusters, see "Off-cluster WWW access" in Section 4.3.3 of the *NVIDIA Base Command Manager Installation Manual*.

```
# request-license
Product Key (XXXXXX-XXXXXX-XXXXXX-XXXXXX-XXXXXX): 123456-123456-123456-123456
…
```

3. Backup the default software image.

   The backup image can be used to create additional software images.

```
# cmsh
% softwareimage
% clone default-image default-image-orig
```

% commit

> **Note**: This document uses `#` to indicate commands executed as the root user on a head node, and `%` to indicate commands executed within `cmsh`. The prompt change is in the preceding block. If it is unclear where a command is being executed, check the prompt that precedes it.

Wait for the ramdisk to be regenerated and the following text to be displayed.

```
Wed Jul 26 09:00:53 2023 [notice] bcm10-headnode: Initial ramdisk for image default-image-
orig was generated successfully
```

4. Backup the DGX software image.

   The backup image can be used to create additional software images.

```
% softwareimage
% clone dgx-os-6.0-a100-image dgx-os-6.0-a100-image-orig
% commit
```

Wait for the ramdisk to be regenerated and the following text to be displayed.

```
Wed Jul 26 09:01:11 2023 [notice] bcm10-headnode: Initial ramdisk for image dgx-a100-image-
orig was generated successfully
```

5. Create the K8s software image by cloning the default software image.

   This software image will be further configured and provisioned onto the K8s control plane nodes. Wait for the ramdisk to be regenerated.

```
% softwareimage
% clone default-image k8s-master-image
% commit
```

6. Add the required kernel modules to the `k8s-master-image` software image.

```
% /
% softwareimage
% use k8s-master-image
% kernelmodules
% add mlx5_core
% add bonding
% softwareimage commit
```

7. Create the `k8s-master` node category and assign the `k8s-master-image` software image to it.

   All nodes assigned to the `k8s-master` category will be provisioned with the `k8s-master-image` software image.

```
% category
% clone default k8s-master
% set softwareimage k8s-master-image
% commit
```

8. Create the DGX nodes.

   `node01` was created during head node installation. Clone `node01` to create the DGX nodes, which will initially be named `node02`, `node03`, `node04`, and `node05`.

```
% device
% foreach --clone node01 -n node02..node05 ()
% commit
```

9. Rename the DGX nodes so they are more easily identified later.

```
% use node02
% set hostname dgx01
% use node03
% set hostname dgx02
% use node04
% set hostname dgx03
% use node05
% set hostname dgx04
% device commit
```

10. Clone `node01` to create the K8s control plane nodes, which will initially be named `node05`, `node06` and `node07`.

```
% device
% foreach --clone node01 -n node06..node08 ()
% commit
```

11. Rename the K8s control plane nodes so they are more easily identifiable.

```
% device
% use node06
% set hostname knode01
% use node07
% set hostname knode02
% use node08
% set hostname knode03
% device commit
```

12. Rename `node01`.

    The purpose of this step is to specify that `node01` is only a template.

```
% device
% use node01
% set hostname template01
% commit
```

13. Assign the DGX nodes to the correct node category – dgx-a100.

```
% foreach -n dgx01..dgx04 (set category dgx-a100)
```

14. Assign the K8S nodes to the `k8s-master` node category.

```
% foreach -n knode01..knode03 (set category k8s-master)
% commit
```

15. Check the nodes and their categories.

    Extra options are used for `device list` to make the format more readable.

```
% device list -f hostname:20,category:10,ip:20,status:15
hostname (key)       category   ip                   status
-------------------- ---------- -------------------- ---------------
bcm10-headnode                  10.227.48.8          [   UP   ]
dgx01                dgx-a100   10.227.48.5          [  DOWN  ]
dgx02                dgx-a100   10.227.48.6          [  DOWN  ]
dgx03                dgx-a100   10.227.48.7          [  DOWN  ]
dgx04                dgx-a100   10.227.48.4          [  DOWN  ]
knode01              k8s-master 10.227.48.4          [  DOWN  ]
knode02              k8s-master 10.227.48.4          [  DOWN  ]
knode03              k8s-master 10.227.48.4          [  DOWN  ]
template01           default    10.227.48.4          [  DOWN  ]
```

2.1.1 Network Configuration

1. Add a Network for InfiniBand (`ibnet`).

```
% network
% add ibnet
% set domainname ibnet.cluster.local
% set baseaddress 10.126.0.0
% set netmaskbits 16
% set mtu 2048
% commit
% add ibnet
```

2. Verify the results.

```
% list -f name:20,type:10,netmaskbits:10,baseaddress:15,domainname:20
name (key)           type       netmaskbit baseaddress     domainname
-------------------- ---------- ---------- --------------- --------------------
externalnet          External   26         10.227.52.0     nvidia.com
globalnet            Global     0          0.0.0.0         cm.cluster
ibnet                Internal   16         10.126.0.0      ibnet.cluster.local
internalnet          Internal   26         10.227.48.0     eth.cluster
ipminet              Internal   26         10.227.20.64    ipmi.cluster
```

3. Ensure that head node interfaces are configured correctly.

```
% device
% use bcm10-headnode
% interfaces
% list
Type         Network device name  IP               Network          Start if
------------ -------------------- ---------------- ---------------- --------
bmc          ipmi0                10.227.20.91     ipminet          always
physical     ens10f0              10.227.52.8      externalnet      always
physical     ens10f1              10.227.20.126    ipminet          always
physical     ens1f1np1 [prov]     10.227.48.8      internalnet      always
```

4. If any interfaces are missing or unconfigured, add any missing devices and configure their network as appropriate.
5. Reboot the head node if the network interfaces were changed.

```
% /
% device
% use bcm10-headnode
% reboot
Reboot in progress for: bcm10-headnode
```

2.1.2 Configure Disk Layouts for Node Categories

Part of using BCM for managing nodes in a DGX BasePOD is to define the disk partitions. Each DGX BasePOD node category includes K8s control plane and DGX node categories. The DGX categories are pre-configured with the correct disk partitions out of the box.

These steps detail how to configure the disk layout for the k8s-master category.

1. Augment the `disksetup` of the k8s-master category.

For the K8s control plane nodes, an EFI System Partition of 100 MB is created at the start of the disk, with the remainder of the disk dedicated to the OS as a single large partition. Note that this disk setup does not have a swap partition.

The configuration file references `/dev/nvme0n1` as the block device used. This may need to be changed to match the specific device name used on systems intended as K8s control plane nodes.

Save the following text to `/cm/local/apps/cmd/etc/htdocs/disk-setup/k8s-disksetup.xml`, factoring in any necessary changes specific to the target systems as noted.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<diskSetup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <device>

    <blockdev>/dev/nvme0n1</blockdev>

    <partition id="a0" partitiontype="esp">
      <size>100M</size>
      <type>linux</type>
      <filesystem>fat</filesystem>
      <mountPoint>/boot/efi</mountPoint>
      <mountOptions>defaults,noatime,nodiratime</mountOptions>
    </partition>
    <partition id="a1">
      <size>max</size>
      <type>linux</type>
      <filesystem>xfs</filesystem>
      <mountPoint>/</mountPoint>
      <mountOptions>defaults,noatime,nodiratime</mountOptions>
    </partition>
  </device>
</diskSetup>
```

2. Assign this disk layout to the `k8s-master` node category.

```
$ cmsh
% category
% use k8s-master
% set disksetup /cm/local/apps/cmd/etc/htdocs/disk-setup/k8s-disksetup.xml
% commit
```

2.1.3 Configure Node Network Interfaces

2.1.3.1 Configure BCM to Allow MAC Addresses to PXE Boot

1. Use the root (not `cmsh`) shell.
2. In `/cm/local/apps/cmd/etc/cmd.conf`, uncomment the `AdvancedConfig` parameter.

```
AdvancedConfig = { "DeviceResolveAnyMAC=1" } # modified value
```

3. Restart the `CMDaemon` to enable reliable PXE booting from bonded interfaces.

```
# systemctl restart cmd
```

Restarting the CMDaemon will disconnect the cmsh session. Type connect to reconnect after the CMDaemon has restarted. Or enter exit and then restart cmsh.

2.1.3.2 Configure Provisioning Interfaces on the DGX Nodes

The steps that follow are performed on the head node and should be run on all DGX systems.

Note: Double check the MAC address for each interface, and the IP address for the bond0 interface. Mistakes here will be difficult to diagnose.

1. Use a cmsh for loop to quickly add the new physical interfaces and the bond0 interface. This will update all four DGX A100 systems.

```
# cmsh
% device
% foreach -n dgx01..dgx04 (interfaces; add physical enp225s0f1; add physical enp97s0f1; add
physical enp225s0f1np1; add physical enp97s0f1np1; commit)
% foreach -n dgx01..dgx04 (interfaces; add bond bond0; set interfaces enp225s0f1 enp97s0f1
enp225s0f1np1 enp97s0f1np1; set network internalnet; set mode 4; set options miimon=100;
commit)
```

2. Set the physical interface MAC addresses as appropriate, and set the ipmi0 and bond0 interfaces if they should be changed—this must be repeated on each DGX system (a single system shown here).

```
# cmsh
% device
% use dgx01
% interfaces
% set enp225s0f1 mac B8:CE:F6:2F:08:69
% set enp97s0f1 mac B8:CE:F6:2D:0E:A7
% set enp225s0f1np1 mac B8:CE:F6:2F:08:69
% set enp97s0f1np1 mac B8:CE:F6:2D:0E:A7
% set ipmi0 ip 10.227.20.69
% set bond0 ip 10.227.48.13
% commit
% list
Type         Network device name  IP               Network          Start if
------------ -------------------- ---------------- ---------------- --------
bmc          ipmi0                10.227.20.69     ipminet          always
physical     BOOTIF [prov]        10.227.48.4      internalnet      always
bond         bond0                10.227.48.13     internalnet      always
physical     enp225s0f1           0.0.0.0                           always
physical     enp225s0f1np1        0.0.0.0                           always
physical     enp97s0f1            0.0.0.0                           always
physical     enp97s0f1np1         0.0.0.0                           always
```

3. Using a foreach loop, the bond0 interface as the provisioninginterface and remove bootif.

```
% /        # go to top level of cmsh
% device
% foreach -n dgx01..dgx04 (set provisioninginterface bond0; commit; interfaces; remove
bootif; commit)
```

4. Verify the configuration.

```
% device
% use dgx01
% get provisioninginterface
bond0
% interfaces
% list
Type         Network device name    IP               Network          Start if
------------ ---------------------- ---------------- ---------------- --------
bmc          ipmi0                  10.227.20.69     ipminet          always
bond         bond0 [prov]           10.227.48.13     internalnet      always
physical     enp225s0f1 (bond0)     0.0.0.0                           always
physical     enp225s0f1np1 (bond0)  0.0.0.0                           always
physical     enp97s0f1 (bond0)      0.0.0.0                           always
physical     enp97s0f1np1 (bond0)   0.0.0.0                           always
```

2.1.3.3 Configure Provisioning Interfaces on the K8s Nodes

All the following steps in this section must be run for each of the three K8s nodes.

1. Use a cmsh for loop to quickly add the new physical interfaces and the bond0 interface. This will update all three knodes.

```
% /        # got to top level of CMSH
% device
% foreach -n knode01..knode03 (interfaces; add physical ens1f1; add physical ens2f1; add
physical ens1f1np1; add physical ens2f1np1; commit)
% foreach -n knode01..knode03 (interfaces; add bond bond0; set interfaces ens1f1np1
ens2f1np1 ens1f1 ens2f1; set network internalnet; set mode 4; set options miimon=100)
```

2. Set the physical interface MAC addresses as appropriate, and set the ipmi0 and bond0 interfaces if they should be changed—this must be repeated on each knode system (a single system shown here).

```
% /
% device
% use knode01
% interfaces
% set ens1f1 mac 04:3F:72:E7:64:97
% set ens1f1np1 mac 04:3F:72:E7:64:97
% set ens2f1 mac 0C:42:A1:79:9B:15
% set ens2f1np1 mac 0C:42:A1:79:9B:15
% add bond bond0
% set ipmi0 ip 10.227.20.80
% set bond0 ip 10.227.48.30
% list
% commit

Type         Network device name  IP               Network          Start if
------------ -------------------- ---------------- ---------------- --------
bmc          ipmi0                10.227.20.80     ipminet          always
physical     BOOTIF [prov]        10.227.48.4      internalnet      always
bond         bond0                10.227.48.30     internalnet      always
physical     ens1f1 (bond0)       0.0.0.0                           always
physical     ens1f1np1 (bond0)    0.0.0.0                           always
physical     ens2f1 (bond0)       0.0.0.0                           always
physical     ens2f1np1 (bond0)    0.0.0.0                           always
```

3. Set the bond0 interface as the provisioninginterface, and remove bootif – a for loop should be used here again.

```
% /
% device
% foreach -n knode01..knode03 (set provisioninginterface bond0; commit; interfaces; remove
bootif; commit)
```

2.1.3.4 Configure InfiniBand Interfaces on DGX Nodes

The following procedure adds four physical InfiniBand interfaces, and must be run for each DGX node.

1. Use a cmsh for loop to quickly add the new physical Infiniband interfaces. This will update all four DGX nodes.

```
% /       # got to top level of CMSH
% device
% foreach -n dgx01..dgx04 (interfaces; add physical ibp12s0; set network ibnet; add
physical ibp141s0; set network ibnet; add physical ibp186s0; set network ibnet; add
physical ibp75s0; set network ibnet; commit)
```

2. Set the ip addresses for each physical Infiniband interface—this will need to be repeated on each DGX system (a single system shown here).

```
% /       # go to top level of CMSH
% device
% use dgx01
% interfaces
% set ibp12s0 ip 10.126.0.13
% set ibp141s0 ip 10.126.2.13
% set ibp186s0 ip 10.126.3.13
% set ibp75s0 ip 10.126.1.13
% commit
% list
Type         Network device name    IP               Network          Start if
------------ ---------------------- ---------------- ---------------- --------
bmc          ipmi0                  10.227.20.69     ipminet          always
bond         bond0 [prov]           10.227.48.13     internalnet      always
physical     enp225s0f1 (bond0)     0.0.0.0                           always
physical     enp225s0f1np1 (bond0)  0.0.0.0                           always
physical     enp97s0f1 (bond0)      0.0.0.0                           always
physical     enp97s0f1np1 (bond0)   0.0.0.0                           always
physical     ibp12s0                10.126.0.13      ibnet            always
physical     ibp141s0               10.126.2.13      ibnet            always
physical     ibp186s0               10.126.3.13      ibnet            always
physical     ibp75s0                10.126.1.13      ibnet            always
```

2.1.3.5 Identify the Cluster Nodes

1. Identify the nodes by setting the MAC address for the provisioning interface for each node to the MAC address listed in the site survey.

```
% device
% set dgx01 mac b8:ce:f6:2f:08:69
% set dgx02 mac 0c:42:a1:54:32:a7
% set dgx03 mac 0c:42:a1:0a:7a:51
% set dgx04 mac 1c:34:da:29:17:6e
% set knode01 mac 04:3F:72:E7:64:97
% set knode02 mac 04:3F:72:D3:FC:EB
% set knode03 mac 04:3F:72:D3:FC:DB
% foreach -c dgx-a100,k8s-master (get mac)
```

```
B8:CE:F6:2F:08:69
0C:42:A1:54:32:A7
0C:42:A1:0A:7A:51
1C:34:DA:29:17:6E
04:3F:72:E7:64:97
04:3F:72:D3:FC:EB
04:3F:72:D3:FC:DB
```

2. If all the MAC addresses are set properly, commit the changes.

   ```
   % device commit
   % quit
   % commit
   ```

2.2 Power On and Provision Cluster Nodes

Now that the required post-installation configuration has been completed, it is time to power on and provision the cluster nodes. After the initial provisioning, power control will be available from within BCM—using the `cmsh` or Base View. But for this initial provisioning it is necessary to power them on outside of BCM (that is, using the power button or a KVM).

It will take several minutes for the nodes to go through their BIOS. After that, the node status will progress as the nodes are being provisioned. Watch the `/var/log/messages` and `/var/log/node-installer` log files to verify that everything is proceeding smoothly.

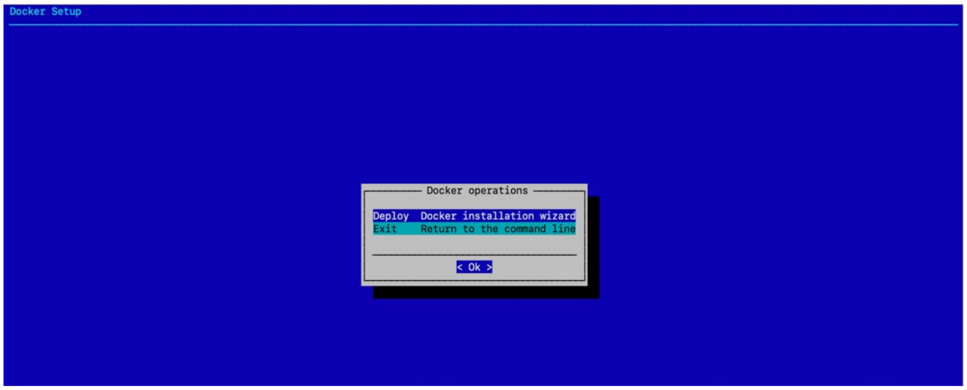
2.3 Deploy Docker

Install docker on the head node and K8s control plane nodes so that users can use docker functions on those nodes, for example, build containers.

1. Run the `cm-docker-setup` CLI wizard on the head node as the root user.

```
# cm-docker-setup
```

2. Choose `Deploy` to continue.

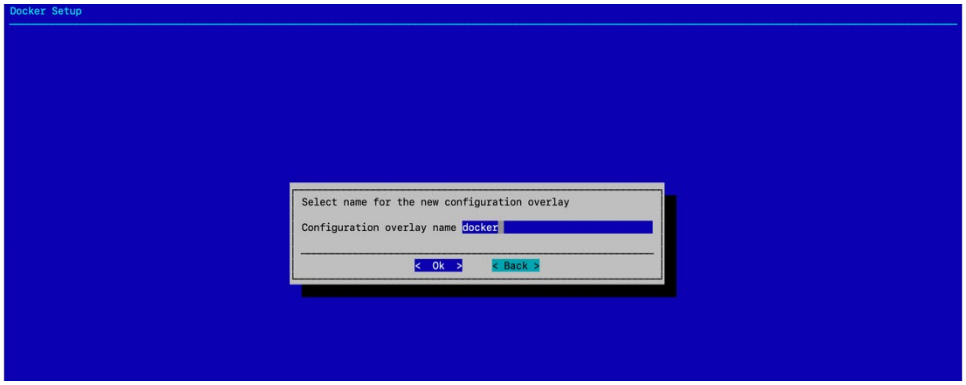


Docker Setup

Docker operations

Deploy Docker installation wizard
Exit Return to the command line

3. By default, the wizard will create a `docker` configuration overlay. This assigns the `Docker::Host` role to the nodes selected in the wizard.

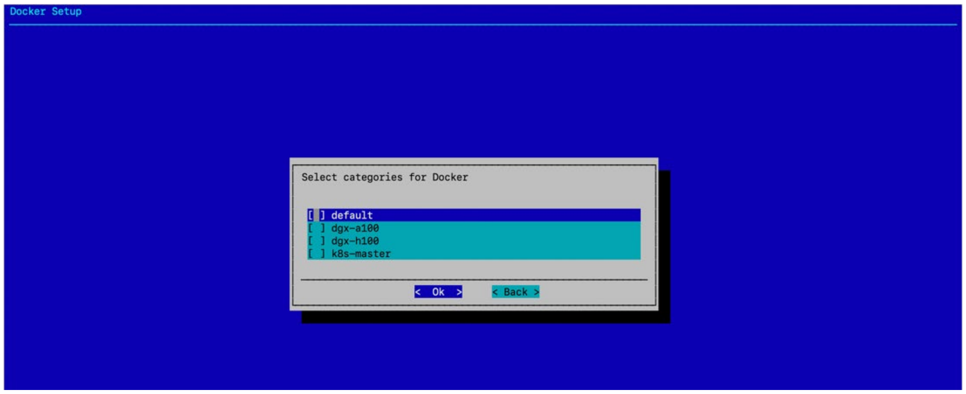


Docker Setup

Select name for the new configuration overlay

Configuration overlay name docker

4. Leave items unselected in the screen because individual nodes will be specified in the next step.

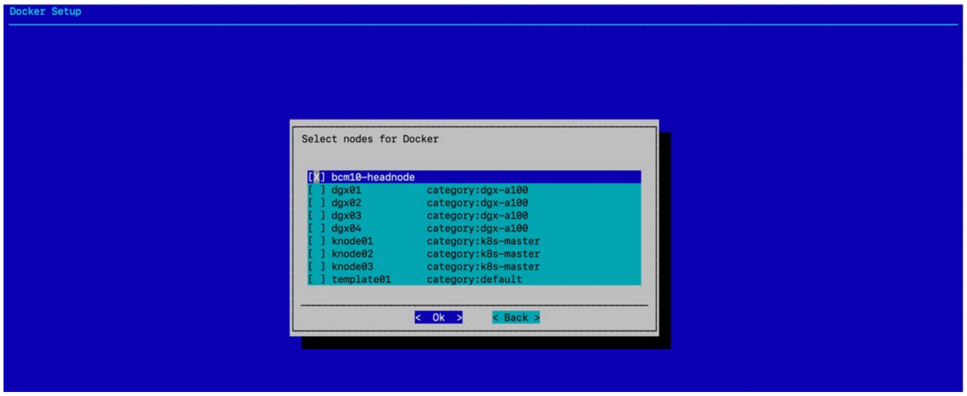


Docker Setup

Select categories for Docker

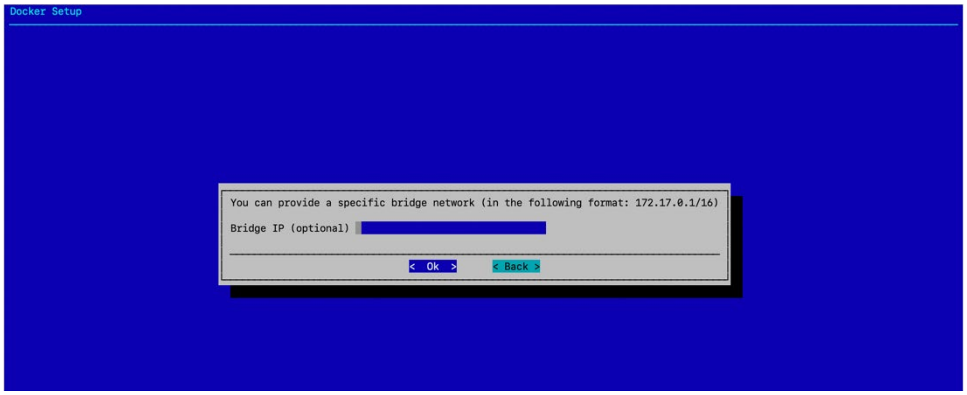
[ ] default
[ ] dgx-a100
[ ] dgx-h100
[ ] k8s-master

5. `bcm10-headnode` is selected to install docker.



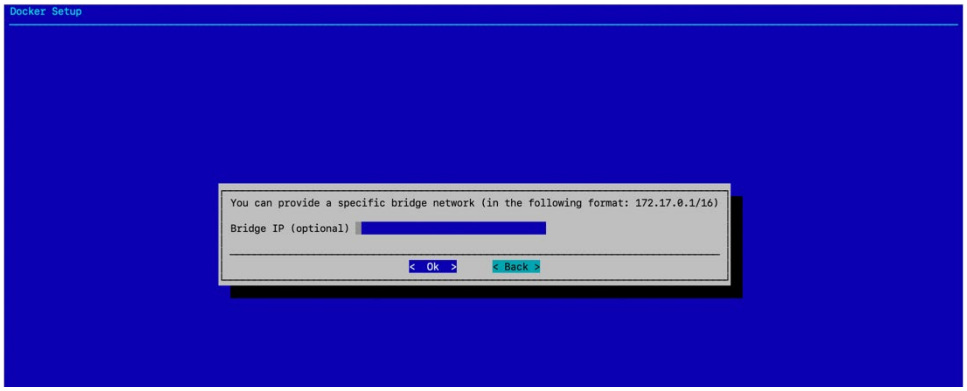
```
Docker Setup

Select nodes for Docker

[X] bcm10-headnode
[ ] dgx01              category:dgx-a100
[ ] dgx02              category:dgx-a100
[ ] dgx03              category:dgx-a100
[ ] dgx04              category:dgx-a100
[ ] knode01            category:k8s-master
[ ] knode02            category:k8s-master
[ ] knode03            category:k8s-master
[ ] template01         category:default

          <  Ok  >     < Back >
```

6. Optionally, specify a specific Docker bridge network. If you choose not to specify a bridge network, the default value of 172.17.0.0/16 will be used.



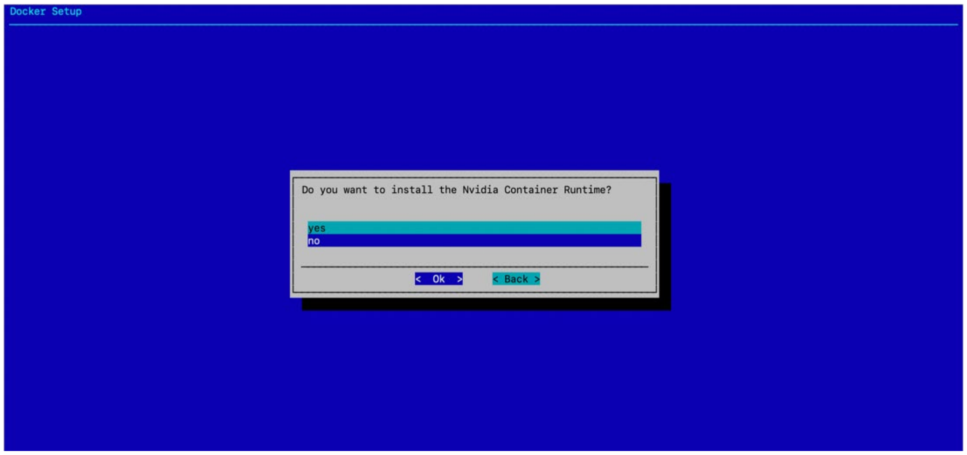
```
Docker Setup

You can provide a specific bridge network (in the following format: 172.17.0.1/16)

Bridge IP (optional)

          <  Ok  >     < Back >
```

7. Enter any local Docker repositories on this next screen.



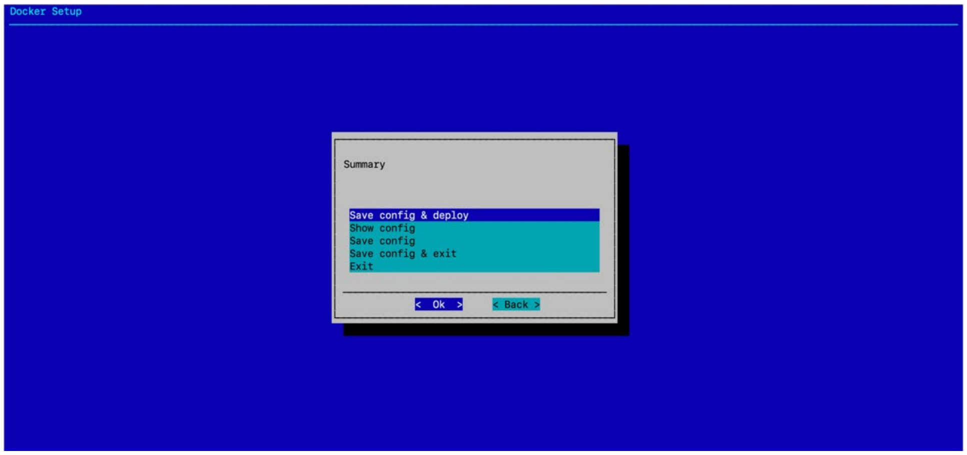
```
Docker Setup

You can provide a specific bridge network (in the following format: 172.17.0.1/16)

Bridge IP (optional)

          <  Ok  >     < Back >
```

8. Do not install the NVIDIA Container Runtime on the head node since there is no GPU on that node.



```
Docker Setup

Do you want to install the Nvidia Container Runtime?

yes
no
```

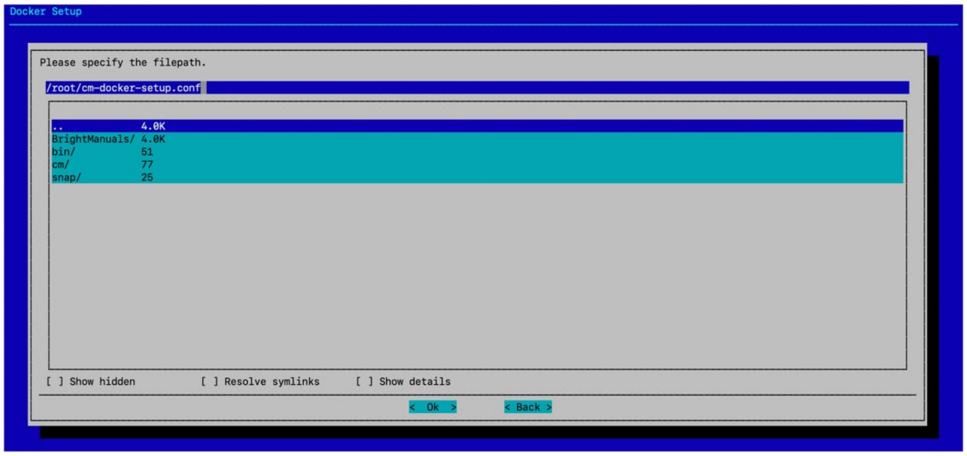
9. Select `Save config & deploy` to save the deployment configuration.



```
Docker Setup

Summary

Save config & deploy
Show config
Save config
Save config & exit
Exit
```

10. By default, the Docker wizard will save the deployment configuration in `/root/cm-docker-setup.conf`.

    This configuration file can be used to redeploy Docker in the future. Select `Ok` to start the installation.



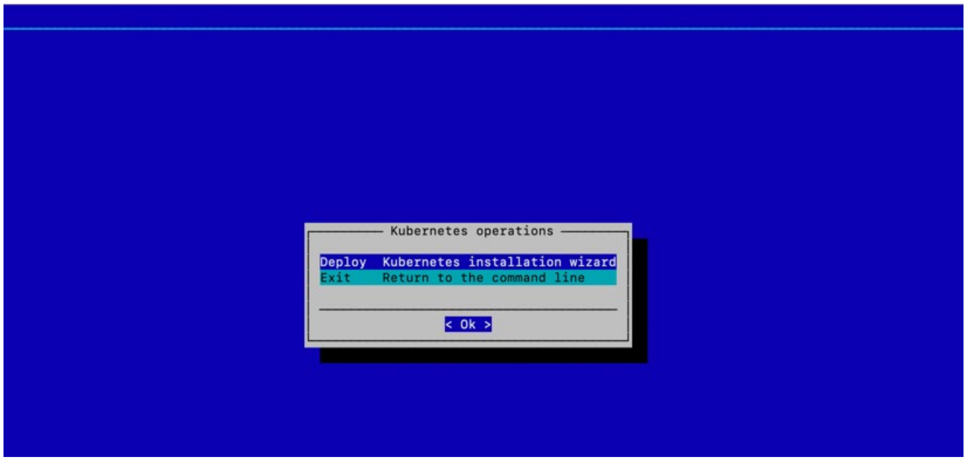
```
Docker Setup

Please specify the filepath.

/root/cm-docker-setup.conf

..             4.0K
BrightManuals/ 4.0K
bin/           51
cm/            77
snap/          25

[ ] Show hidden          [ ] Resolve symlinks          [ ] Show details
```

2.4 Deploy K8s

1. Run the `cm-kubernetes-setup` CLI wizard as the root user on the head node.

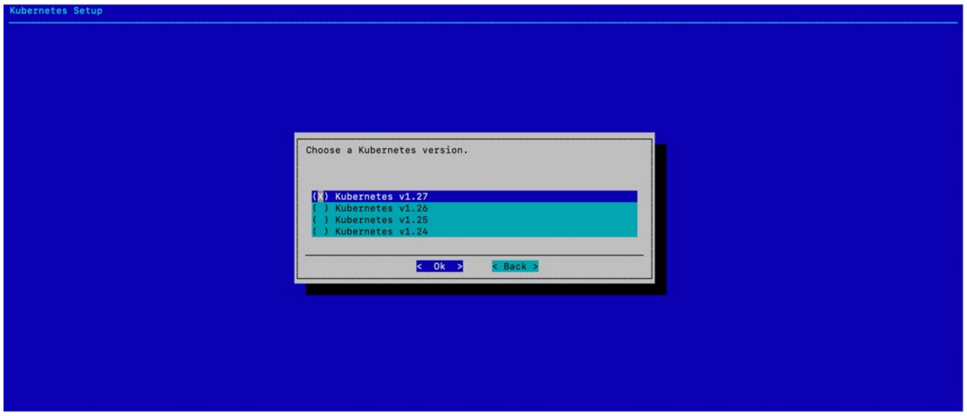
```
# cm-kubernetes-setup
```

2. Choose `Deploy` to start the deployment.

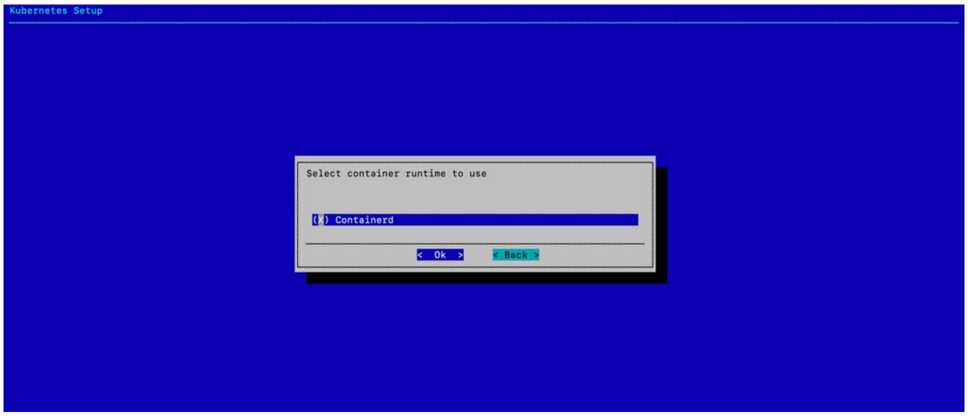


```
Kubernetes operations
Deploy  Kubernetes installation wizard
Exit    Return to the command line
< Ok >
```

3. Select K8s version v.1.27 in the dialog that appears next and select Ok to continue.



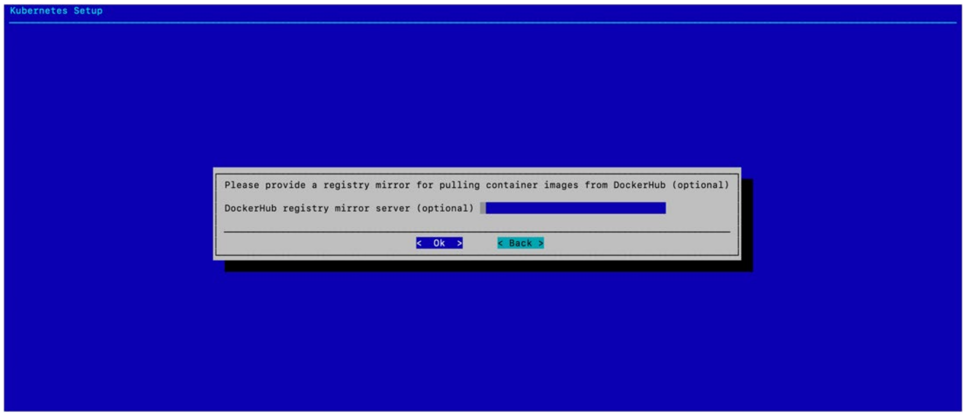
```
Kubernetes Setup
Choose a Kubernetes version.
(X) Kubernetes v1.27
( ) Kubernetes v1.26
( ) Kubernetes v1.25
( ) Kubernetes v1.24
<  Ok  >    < Back >
```



```
Kubernetes Setup
Select container runtime to use
(X) Containerd
<  Ok  >    < Back >
```

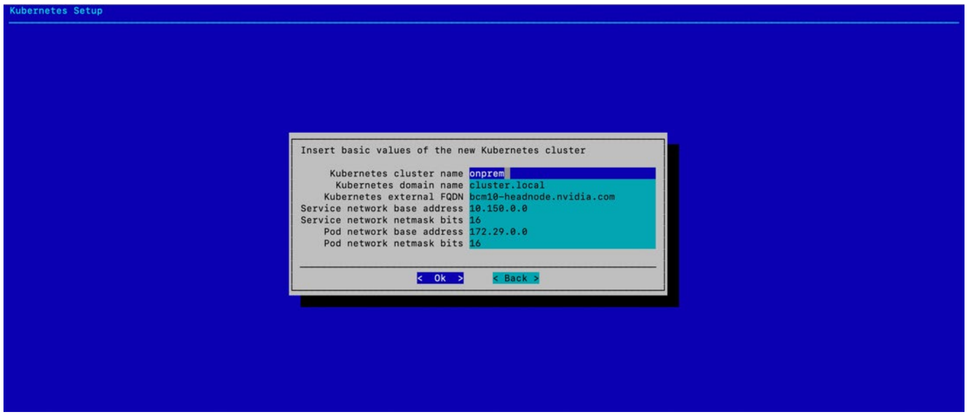
4. Select Ok to confirm the `Containerd` container runtime.

5. Fill in the optional DockerHub registry mirror endpoint if necessary—otherwise, select Ok to continue.



```
Kubernetes Setup

Please provide a registry mirror for pulling container images from DockerHub (optional)

DockerHub registry mirror server (optional)
```

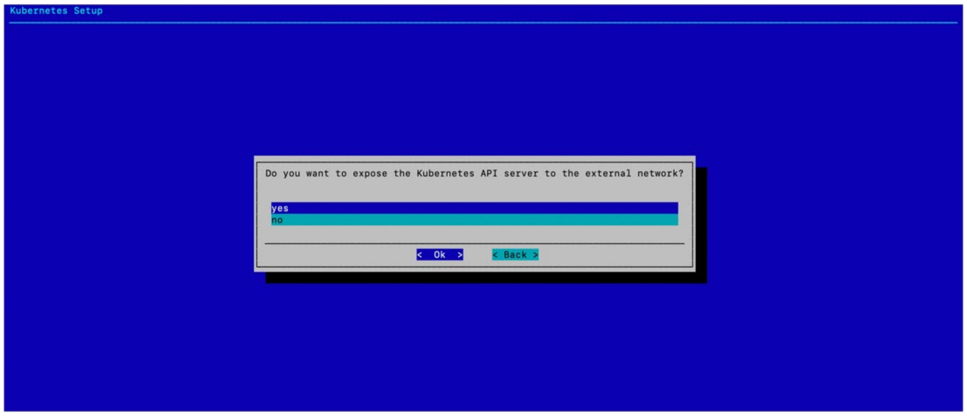
6. Accept the default settings for this K8s cluster—to match the naming chosen for this guide, change the K8s cluster name to onprem.



```
Kubernetes Setup

Insert basic values of the new Kubernetes cluster

      Kubernetes cluster name onprem
       Kubernetes domain name cluster.local
     Kubernetes external FQDN bcm10-headnode.nvidia.com
 Service network base address 10.150.0.0
 Service network netmask bits 16
     Pod network base address 172.29.0.0
     Pod network netmask bits 16
```

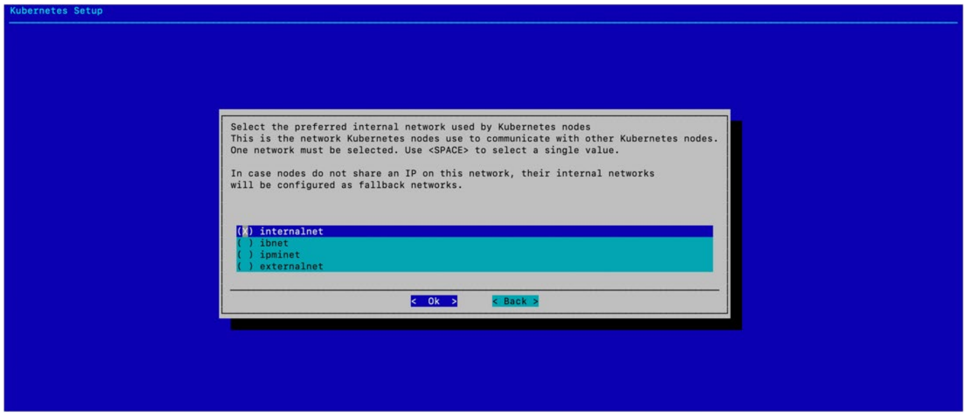
7. Select `yes` to expose the K8s API server on the head node.

   This allows users to use the K8s cluster from the head node.



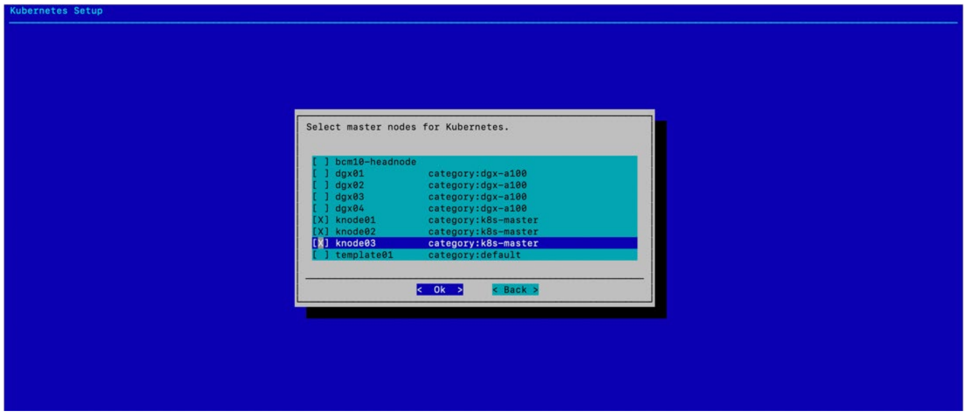
```
Do you want to expose the Kubernetes API server to the external network?

yes
no
```

8. Select `internalnet` since the K8s control plane nodes and the DGX nodes, which are the K8s worker nodes, are all connected on `internalnet`.



```
Select the preferred internal network used by Kubernetes nodes
This is the network Kubernetes nodes use to communicate with other Kubernetes nodes.
One network must be selected. Use <SPACE> to select a single value.

In case nodes do not share an IP on this network, their internal networks
will be configured as fallback networks.

(X) internalnet
( ) ibnet
( ) ipminet
( ) externalnet
```

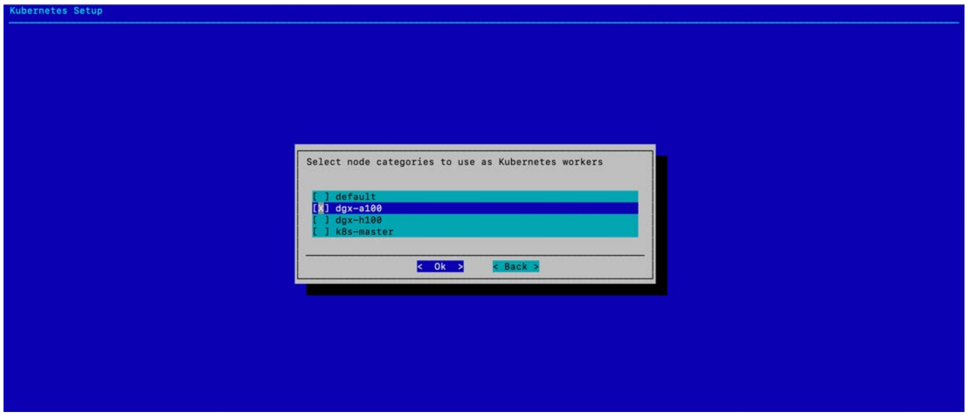
9. Select all three K8s control plane nodes: `knode01`, `knode02`, and `knode03`.



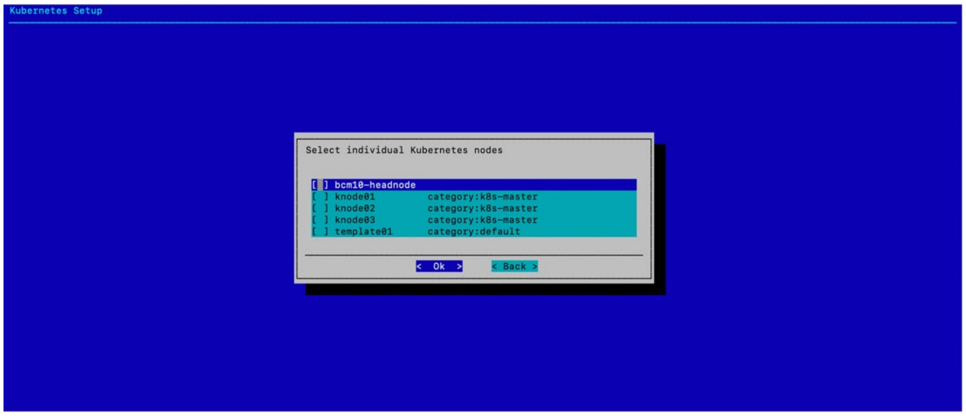
```
Kubernetes Setup

Select master nodes for Kubernetes.

[ ] bcm10-headnode
[ ] dgx01            category:dgx-a100
[ ] dgx02            category:dgx-a100
[ ] dgx03            category:dgx-a100
[ ] dgx04            category:dgx-a100
[X] knode01          category:k8s-master
[X] knode02          category:k8s-master
[X] knode03          category:k8s-master
[ ] template01       category:default
```

10. Select `dgx-a100` for the worker node category.

11. Do not select any individual Kubernetes nodes, and select Ok to continue.

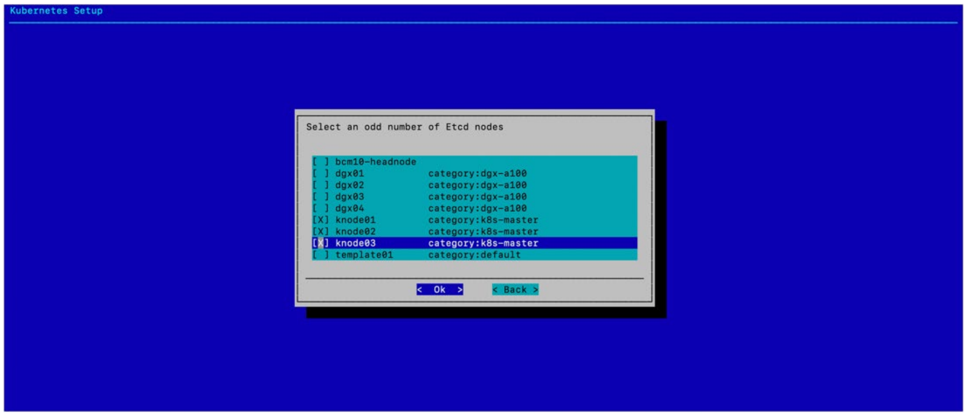


```
Kubernetes Setup

Select node categories to use as Kubernetes workers

[ ] default
[X] dgx-a100
[ ] dgx-h100
[ ] k8s-master
```



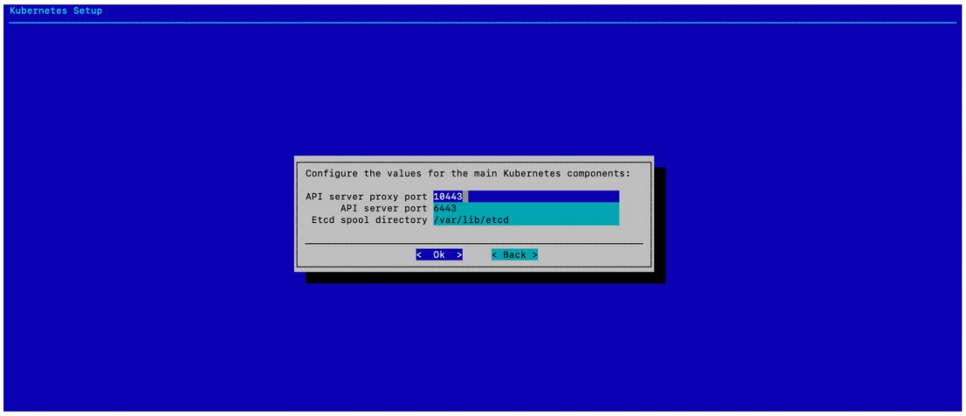
```
Kubernetes Setup

Select individual Kubernetes nodes

[ ] bcm10-headnode
[ ] knode01          category:k8s-master
[ ] knode02          category:k8s-master
[ ] knode03          category:k8s-master
[ ] template01       category:default
```

12. Select all three K8s control plane nodes: `knode01`, `knode02`, and `knode03` for the `Etcd` nodes.



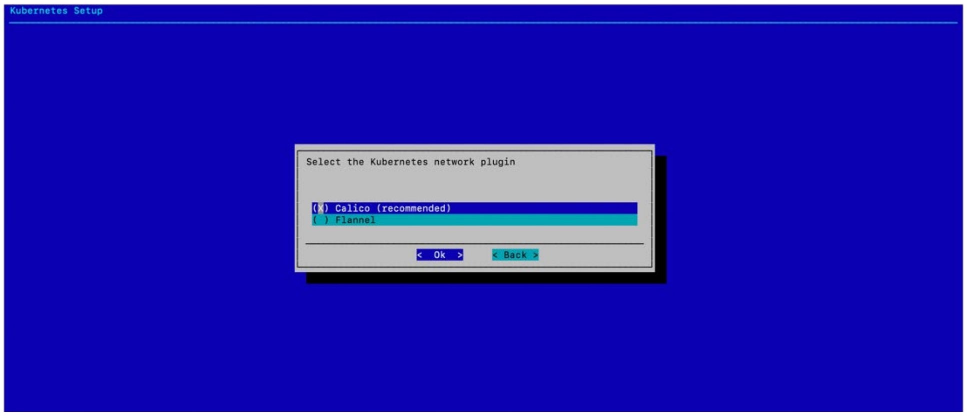
```
Kubernetes Setup

Select an odd number of Etcd nodes

[ ] bcm10-headnode
[ ] dgx01              category:dgx-a100
[ ] dgx02              category:dgx-a100
[ ] dgx03              category:dgx-a100
[ ] dgx04              category:dgx-a100
[X] knode01            category:k8s-master
[X] knode02            category:k8s-master
[X] knode03            category:k8s-master
[ ] template01         category:default
```

13. Accept the default values for the main Kubernetes components unless the organization requires specific ports.



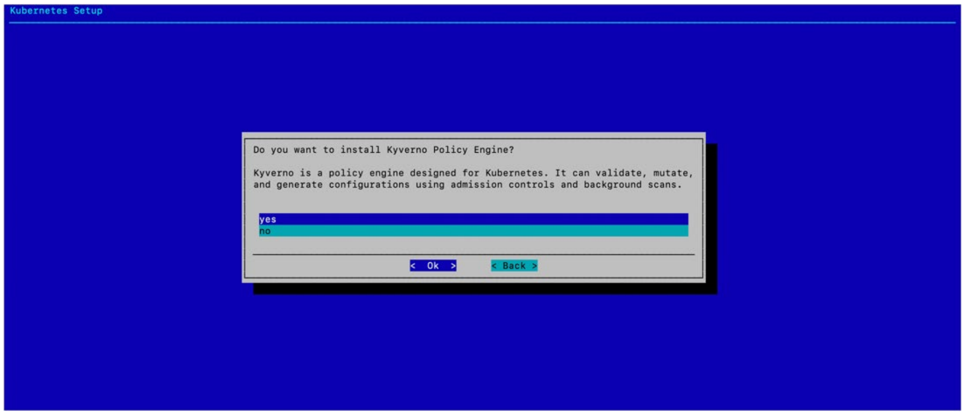
```
Kubernetes Setup

Configure the values for the main Kubernetes components:

API server proxy port 10443
      API server port 6443
 Etcd spool directory /var/lib/etcd
```

14. Select the `Calico network plugin` when prompted.



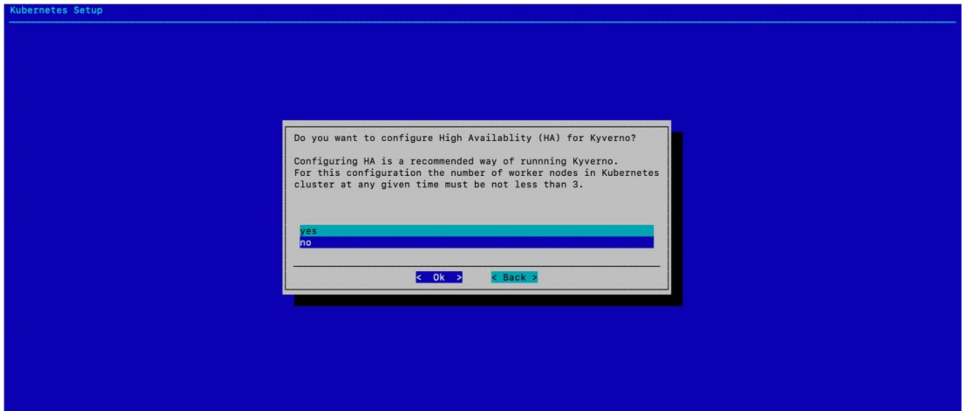
```
Select the Kubernetes network plugin

(X) Calico (recommended)
( ) Flannel
```

15. Choose yes to install the Kyverno policy engine and then select Ok.



```
Do you want to install Kyverno Policy Engine?

Kyverno is a policy engine designed for Kubernetes. It can validate, mutate,
and generate configurations using admission controls and background scans.

yes
no
```

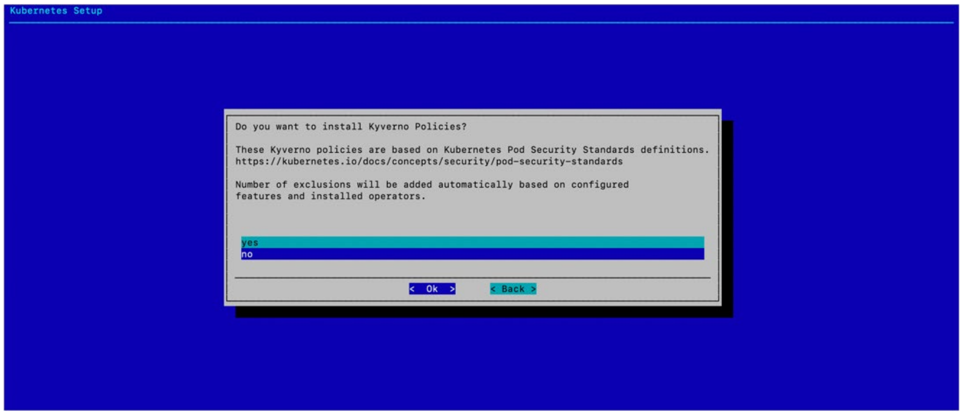
16. Choose no to decline to configure HA for Kyverno and then select Ok.



```
Do you want to configure High Availablity (HA) for Kyverno?

Configuring HA is a recommended way of runnning Kyverno.
For this configuration the number of worker nodes in Kubernetes
cluster at any given time must be not less than 3.

yes
no
```

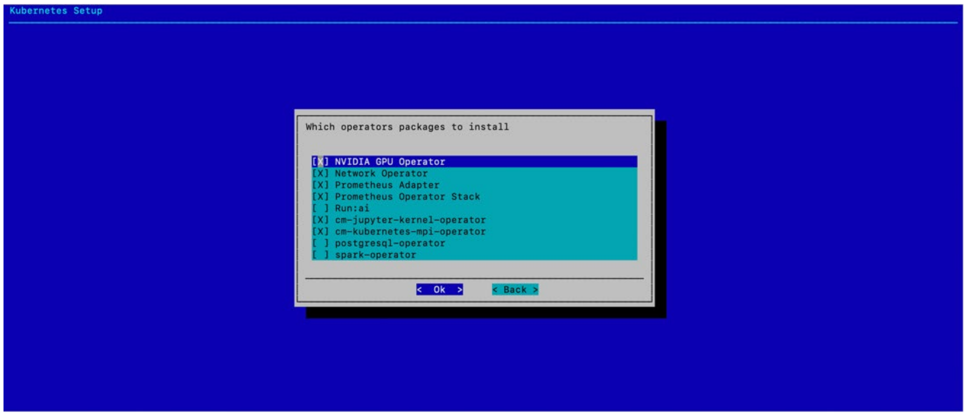
17. Choose whether to install Kyverno Policies and then select Ok.

    Unless required for the configuration, choose no.



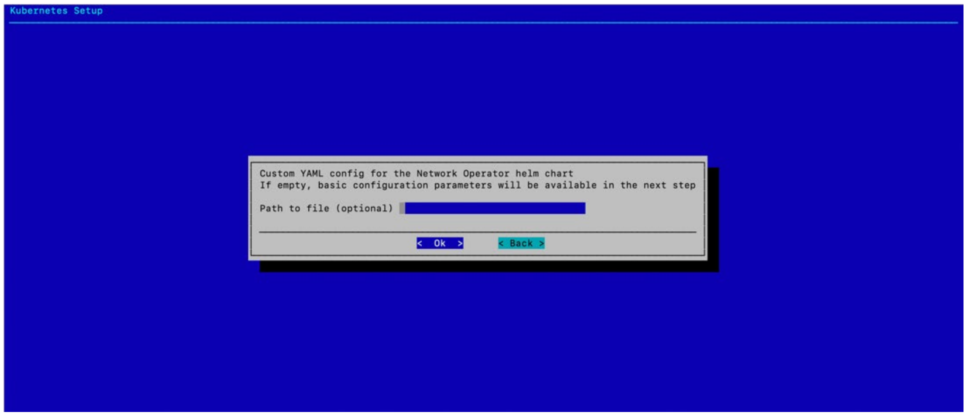
```
Do you want to install Kyverno Policies?

These Kyverno policies are based on Kubernetes Pod Security Standards definitions.
https://kubernetes.io/docs/concepts/security/pod-security-standards

Number of exclusions will be added automatically based on configured
features and installed operators.

yes
no
```

18. Select the following operators: `NVIDIA GPU Operator`, `Network Operator`, `Prometheus Adapter`, `Prometheus Operator Stack`, `cm-jupyter-kernel-operator`, and the `cm-kubernetes-mpi-operator` to install.



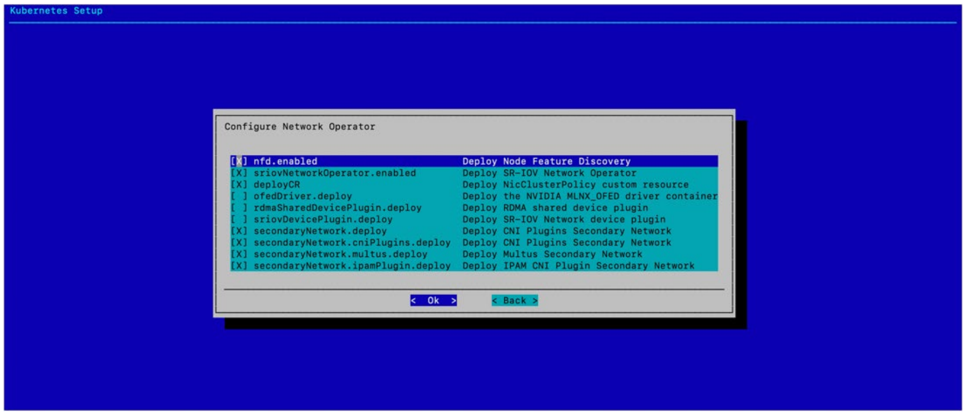
```
Which operators packages to install

[X] NVIDIA GPU Operator
[X] Network Operator
[X] Prometheus Adapter
[X] Prometheus Operator Stack
[ ] Run:ai
[X] cm-jupyter-kernel-operator
[X] cm-kubernetes-mpi-operator
[ ] postgresql-operator
[ ] spark-operator
```

19. Skip the optional YAML config for the Network Operator helm chart.



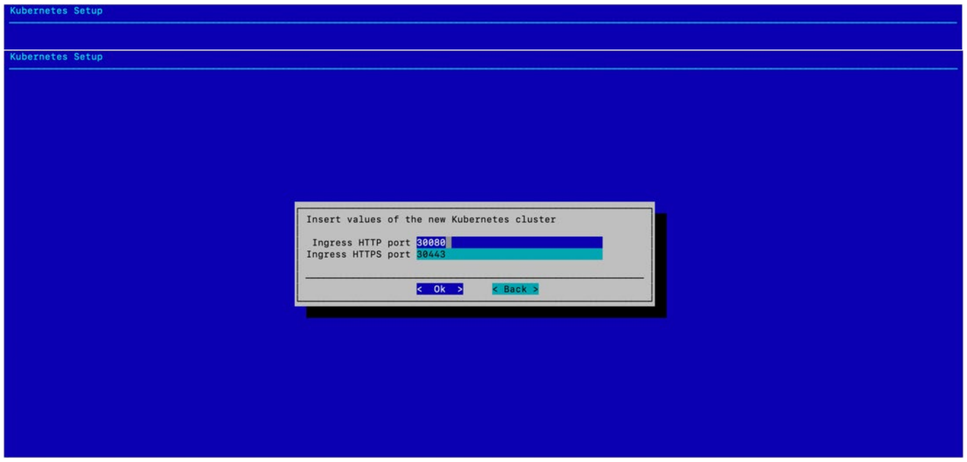
```
Custom YAML config for the Network Operator helm chart
If empty, basic configuration parameters will be available in the next step

Path to file (optional)
```

20. Configure the Network Operator by selecting nfs.enabled, sriovNetworkOperator.enabled, deployCR, secondaryNetwork.deploy, secondaryNetwork.cniPlugins.deploy, secondaryNetwork.multus.deploy, and secondaryNetwork.ipamPlugin.deploy.



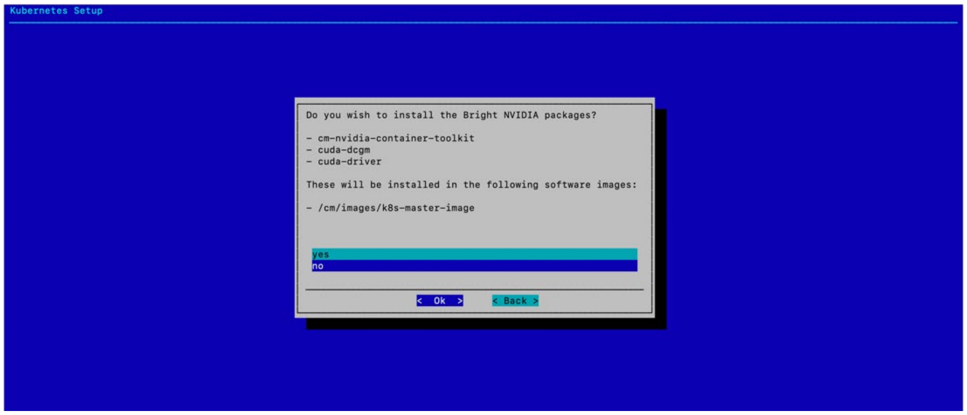
```
Kubernetes Setup

Configure Network Operator

[X] nfd.enabled                          Deploy Node Feature Discovery
[X] sriovNetworkOperator.enabled         Deploy SR-IOV Network Operator
[X] deployCR                             Deploy NicClusterPolicy custom resource
[ ] ofedDriver.deploy                    Deploy the NVIDIA MLNX_OFED driver container
[ ] rdmaSharedDevicePlugin.deploy        Deploy RDMA shared device plugin
[ ] sriovDevicePlugin.deploy             Deploy SR-IOV Network device plugin
[X] secondaryNetwork.deploy              Deploy CNI Plugins Secondary Network
[X] secondaryNetwork.cniPlugins.deploy   Deploy CNI Plugins Secondary Network
[X] secondaryNetwork.multus.deploy       Deploy Multus Secondary Network
[X] secondaryNetwork.ipamPlugin.deploy   Deploy IPAM CNI Plugin Secondary Network
```

21. Select the `Ingress Controller (Nginx)`, `Kubernetes Dashboard`, `Kubernetes Metrics Server`, and `Kubernetes State Metrics` to deploy.
22. Select the defaults unless specific ingress ports are to be used.



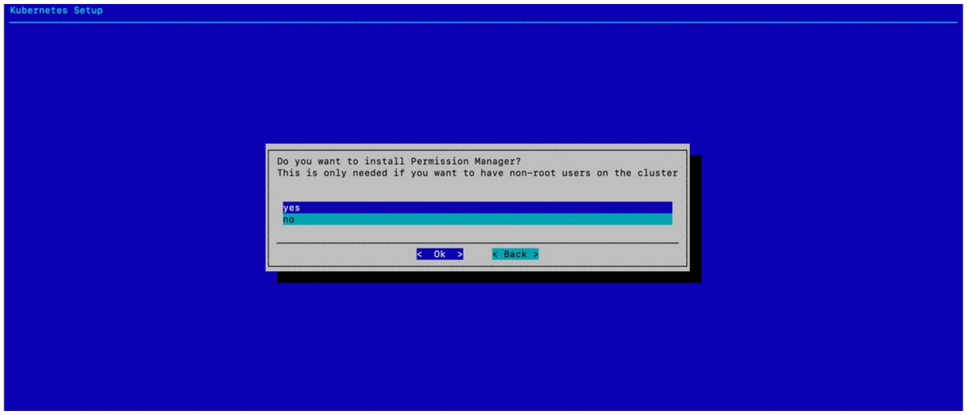
```
Kubernetes Setup
Kubernetes Setup

Insert values of the new Kubernetes cluster

 Ingress HTTP port 30080
Ingress HTTPS port 30443
```

23. Select `no` since the K8s control plane nodes do not have GPUs.



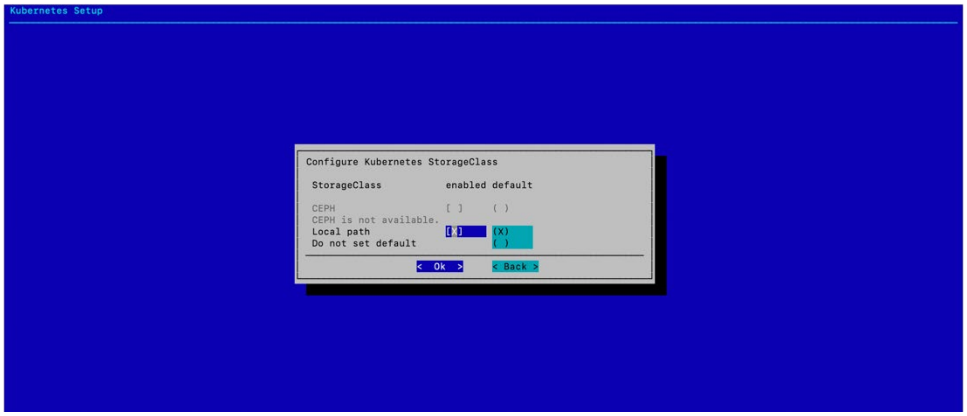
```
Do you wish to install the Bright NVIDIA packages?

- cm-nvidia-container-toolkit
- cuda-dcgm
- cuda-driver

These will be installed in the following software images:

- /cm/images/k8s-master-image


yes
no
```

24. Select `yes` to deploy the `Permission Manager`.



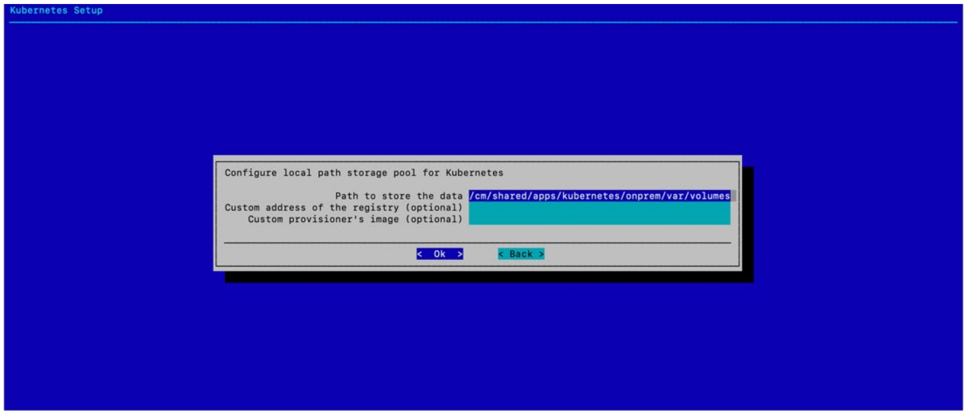
```
Do you want to install Permission Manager?
This is only needed if you want to have non-root users on the cluster

yes
no
```

25. Select both `enabled` and `default` for the `Local path` storage class.



```
Configure Kubernetes StorageClass

 StorageClass          enabled default

 CEPH                  [ ]     ( )
 CEPH is not available.
 Local path            [X]     (X)
 Do not set default            ( )
```

26. Accept the default data storage path and leave the other two fields blank, which is the default.

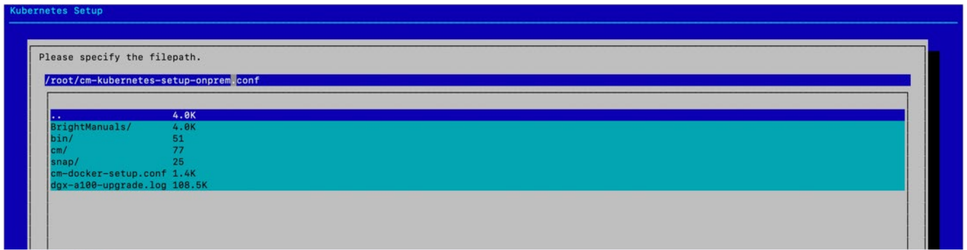


```
Kubernetes Setup

Configure local path storage pool for Kubernetes

             Path to store the data /cm/shared/apps/kubernetes/onprem/var/volumes
Custom address of the registry (optional)
    Custom provisioner's image (optional)
```

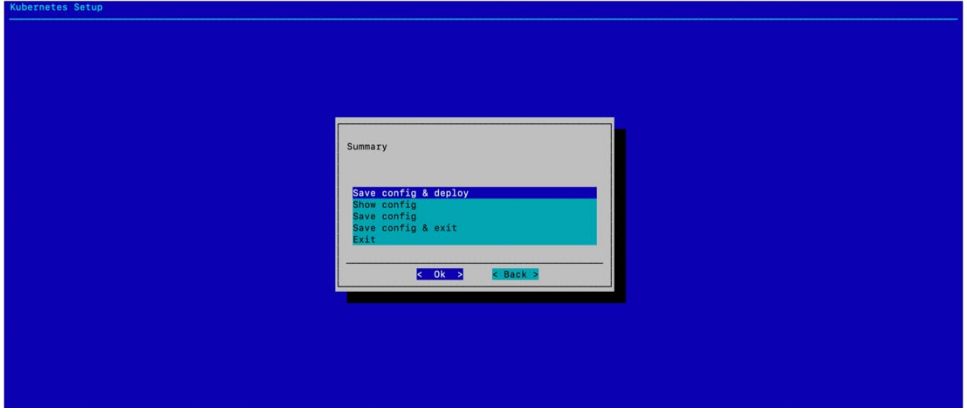
27. Select `Save config & deploy`.
28. Change the filepath to `/root/cm-kubernetes-setup-onprem.conf` and select Ok.

    This file can be used to redeploy K8s or copied and modified to deploy additional K8s clusters.

    Wait for the installation to finish.



```
Kubernetes Setup

Please specify the filepath.

/root/cm-kubernetes-setup-onprem.conf

..                    4.0K
BrightManuals/        4.0K
bin/                  51
cm/                   77
snap/                 25
cm-docker-setup.conf  1.4K
dgx-a100-upgrade.log  108.5K
```



```
Kubernetes Setup

Summary

Save config & deploy
Show config
Save config
Save config & exit
Exit
```

29. Verify that the K8s cluster is installed properly.

```
# module load kubernetes/onprem/1.27.4-00
# kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:10443
CoreDNS is running at https://127.0.0.1:10443/api/v1/namespaces/kube-system/services/kube-
dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

# kubectl get nodes
NAME      STATUS   ROLES                  AGE     VERSION
dgx01     Ready    worker                 7m36s   v1.27.4
dgx02     Ready    worker                 7m37s   v1.27.4
dgx03     Ready    worker                 7m36s   v1.27.4
dgx04     Ready    worker                 7m37s   v1.27.4
knode01   Ready    control-plane,master   7m59s   v1.27.4
knode02   Ready    control-plane,master   7m26s   v1.27.4
knode03   Ready    control-plane,master   7m25s   v1.27.4
```

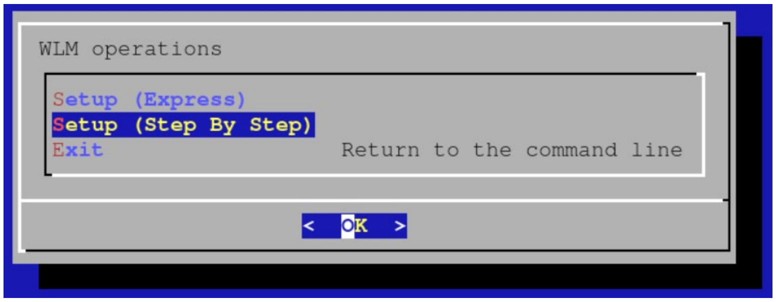
2.5 Deploying Slurm

A workload management system is helpful to be able to schedule jobs on a cluster of nodes. The steps below describe how to set up Slurm in such a way so that GPUs have to be explicitly requested. This way it becomes much easier to share GPU and CPU compute resources among many people without having users get in each other’s way.

1. Start the interactive setup tool.

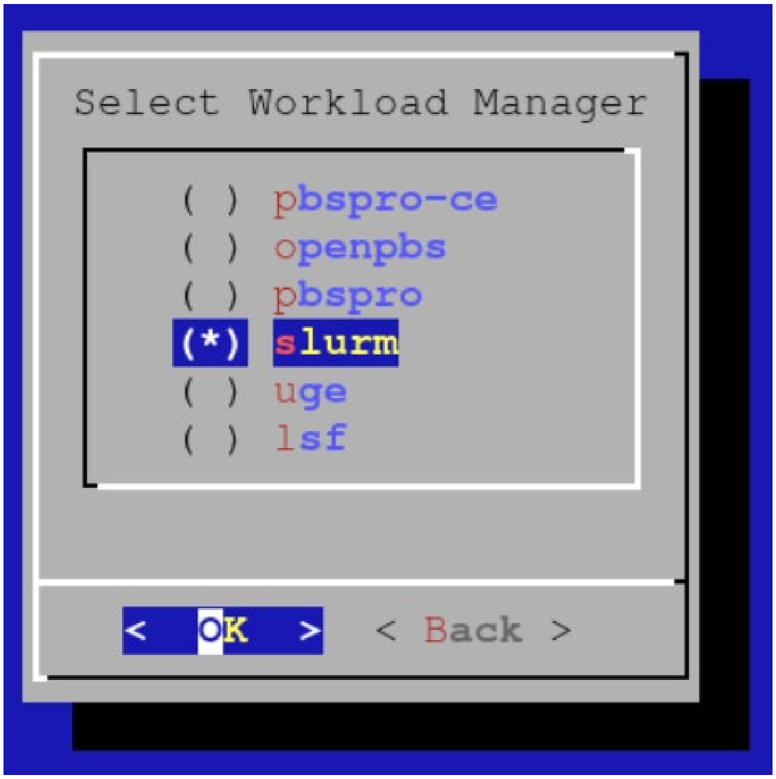
```
# cm-wlm-setup --disable --wlm-cluster-name=slurm --yes-i-really-mean-it
# cm-wlm-setup
```

2. Choose `Setup (Step By Step)` and then select `OK`.



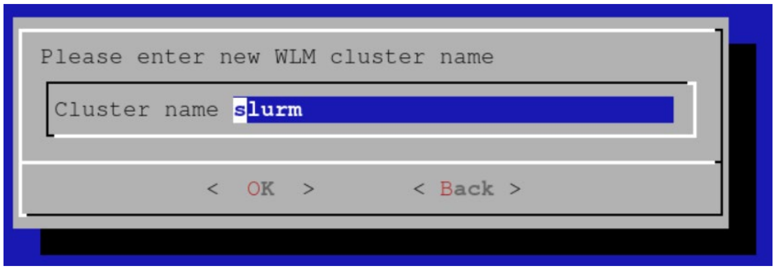
```
WLM operations

Setup (Express)
Setup (Step By Step)
Exit                       Return to the command line

                 <  OK  >
```

3. Choose `slurm` as the workload management system and then select OK.



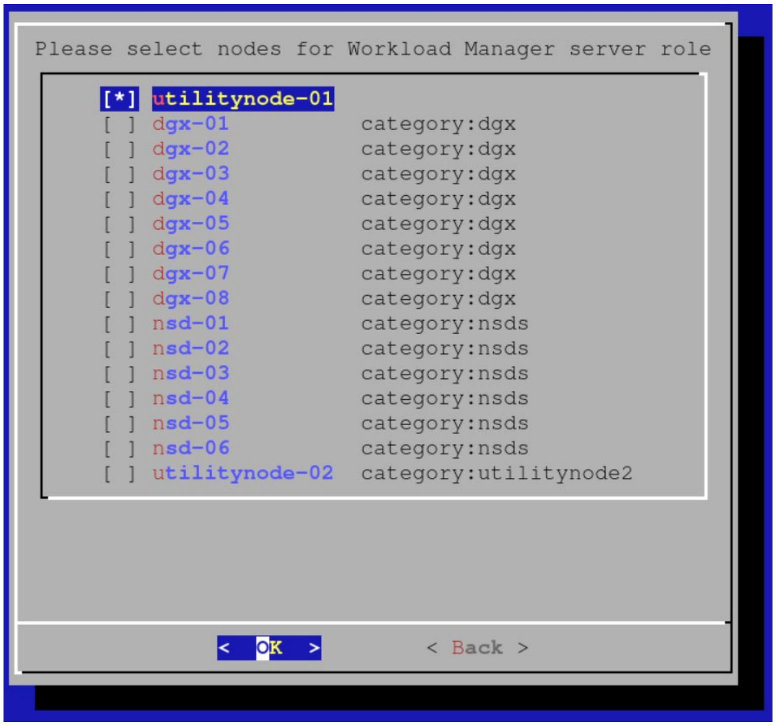
```
Select Workload Manager

    ( ) pbspro-ce
    ( ) openpbs
    ( ) pbspro
    (*) slurm
    ( ) uge
    ( ) lsf

<  OK  >   < Back >
```

4. Enter the name for the Slurm cluster and then select OK.



```
Please enter new WLM cluster name

Cluster name slurm
```

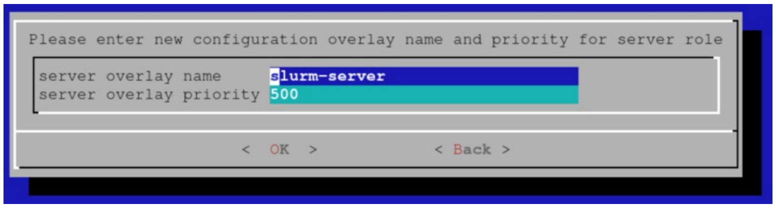
5. Choose the head node as the Slurm server.



```
Please select nodes for Workload Manager server role

    [*] utilitynode-01
    [ ] dgx-01          category:dgx
    [ ] dgx-02          category:dgx
    [ ] dgx-03          category:dgx
    [ ] dgx-04          category:dgx
    [ ] dgx-05          category:dgx
    [ ] dgx-06          category:dgx
    [ ] dgx-07          category:dgx
    [ ] dgx-08          category:dgx
    [ ] nsd-01          category:nsds
    [ ] nsd-02          category:nsds
    [ ] nsd-03          category:nsds
    [ ] nsd-04          category:nsds
    [ ] nsd-05          category:nsds
    [ ] nsd-06          category:nsds
    [ ] utilitynode-02  category:utilitynode2
```

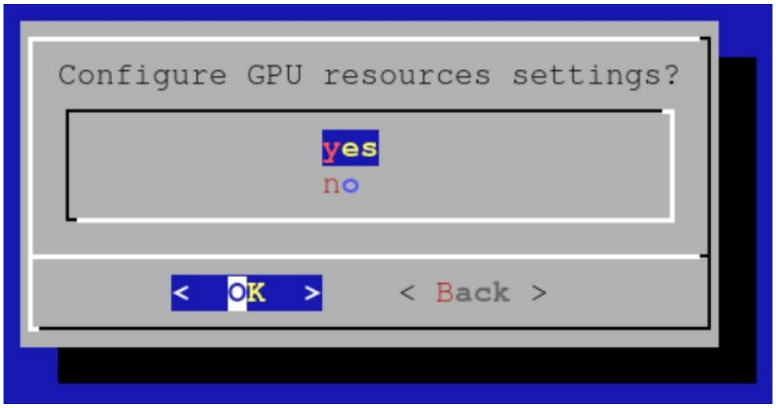
6. Set the overlay name and priority for the configuration overlay and then select OK.

This example uses the defaults.



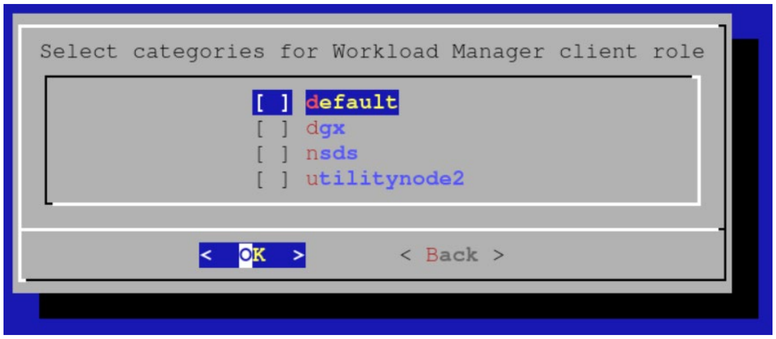
```
Please enter new configuration overlay name and priority for server role

server overlay name       slurm-server
server overlay priority   500
```

7. Choose yes to configure GPU resources and then select `OK`.



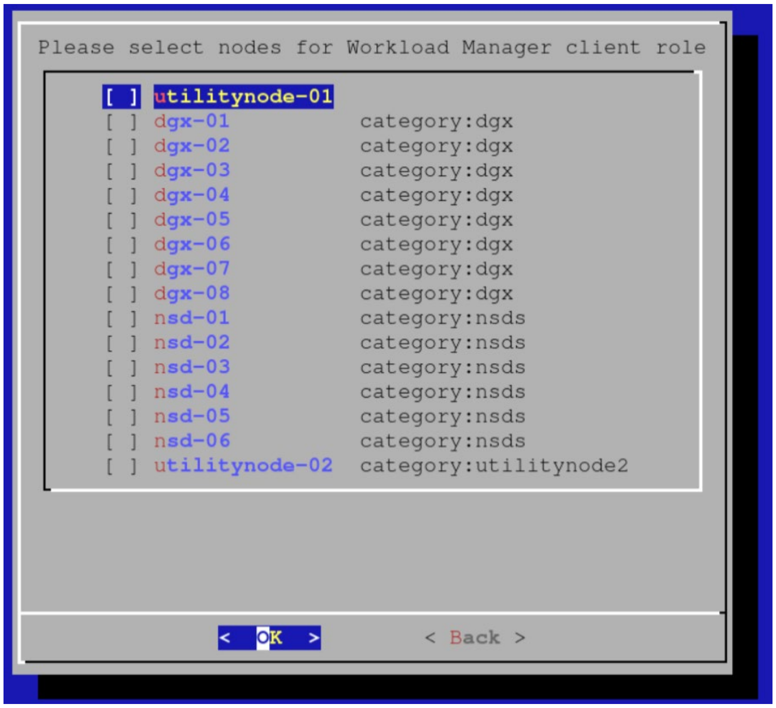
```
Configure GPU resources settings?

                yes
                no

        <  OK  >      < Back >
```

8. Initially `cm-wlm-setup` will set up Slurm clients *without* GPUs. Assuming that there are no nodes to be set up without GPUs, unselect all categories and press OK.



```
Select categories for Workload Manager client role

            [ ] default
            [ ] dgx
            [ ] nsds
            [ ] utilitynode2

        <  OK  >        < Back >
```

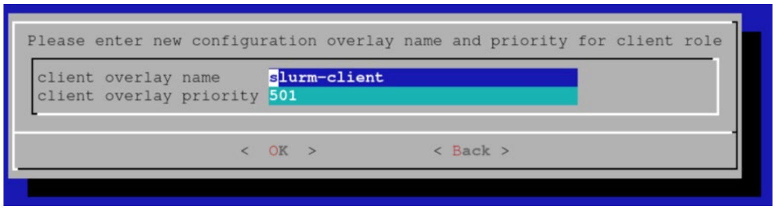
9. Assuming there are no compute nodes without GPUs, leave all the options unselected at the following screen and press OK.



```
Please select nodes for Workload Manager client role

     [ ] utilitynode-01
     [ ] dgx-01          category:dgx
     [ ] dgx-02          category:dgx
     [ ] dgx-03          category:dgx
     [ ] dgx-04          category:dgx
     [ ] dgx-05          category:dgx
     [ ] dgx-06          category:dgx
     [ ] dgx-07          category:dgx
     [ ] dgx-08          category:dgx
     [ ] nsd-01          category:nsds
     [ ] nsd-02          category:nsds
     [ ] nsd-03          category:nsds
     [ ] nsd-04          category:nsds
     [ ] nsd-05          category:nsds
     [ ] nsd-06          category:nsds
     [ ] utilitynode-02  category:utilitynode2

        <  OK  >         < Back >
```

10. Enter the overlay name and priority for Slurm clients *without* GPUs and then select `OK`.

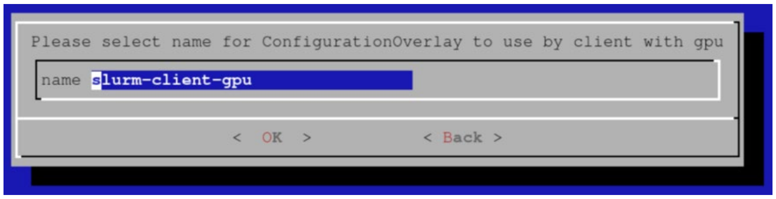
    The example uses the defaults.



```
Please enter new configuration overlay name and priority for client role

client overlay name      slurm-client
client overlay priority  501
```

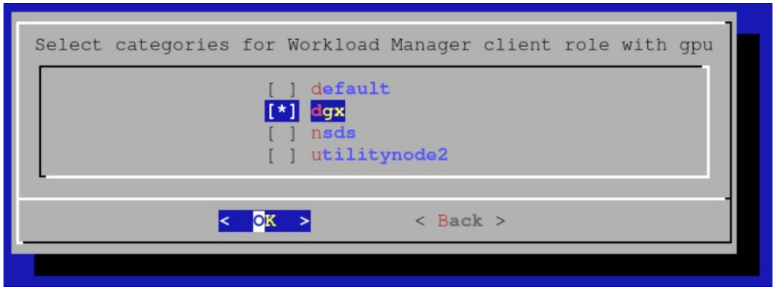
11. Choose a suitable name for the configuration overlay of Slurm clients *with* GPUs.

    The example uses the defaults.



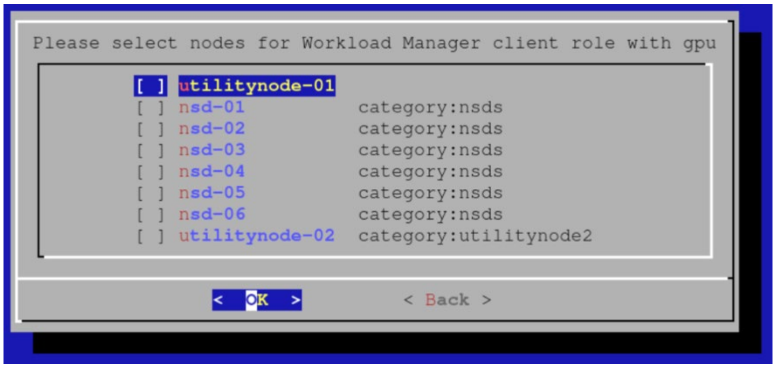
```
Please select name for ConfigurationOverlay to use by client with gpu

name slurm-client-gpu
```

12. Select the categories of compute nodes *with* GPUs that you would like to include in the configuration overlay that was created in the previous step.



```
Select categories for Workload Manager client role with gpu

[ ] default
[*] dgx
[ ] nsds
[ ] utilitynode2
```

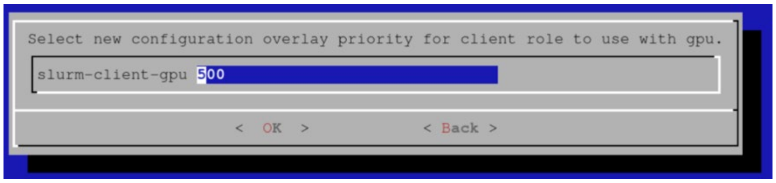
13. Select any additional nodes *with* GPUs that should be added to the configuration overlay.



```
Please select nodes for Workload Manager client role with gpu

[ ] utilitynode-01
[ ] nsd-01            category:nsds
[ ] nsd-02            category:nsds
[ ] nsd-03            category:nsds
[ ] nsd-04            category:nsds
[ ] nsd-05            category:nsds
[ ] nsd-06            category:nsds
[ ] utilitynode-02    category:utilitynode2
```

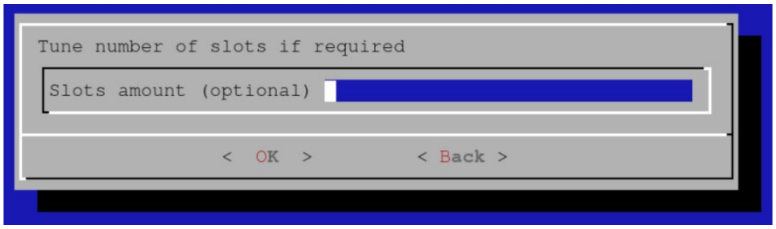
14. Select a priority for the configuration overlay.

    The example uses the default.



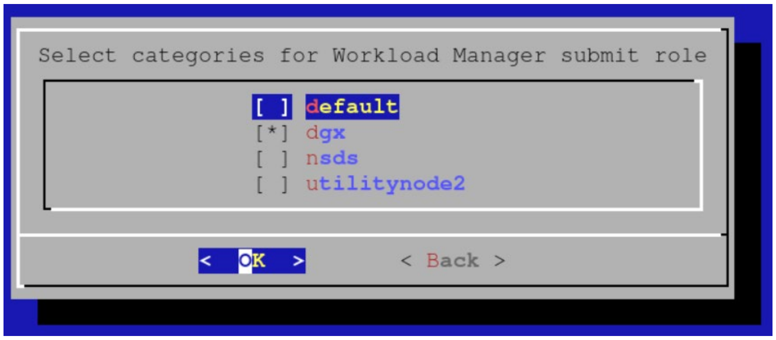
```
Select new configuration overlay priority for client role to use with gpu.

slurm-client-gpu 500
```

15. Leave the number of slots unconfigured.



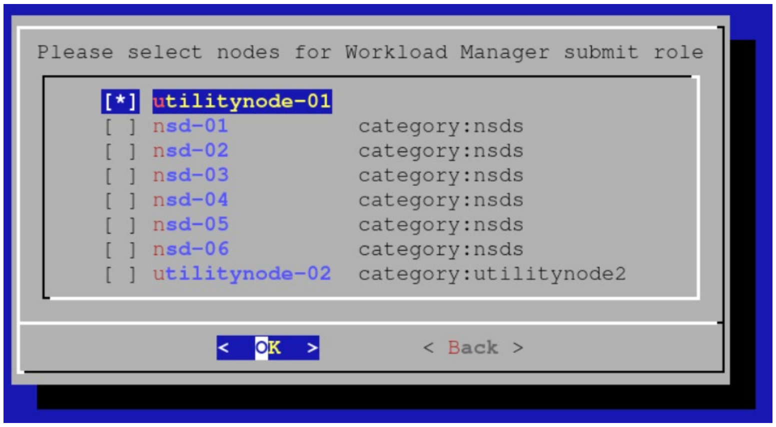
```
Tune number of slots if required

Slots amount (optional)
```

16. Select the category of GPU compute nodes as nodes from which jobs will be submitted. If you have a category of login nodes, you will want to add it as well. We will add the head node in the next screen:



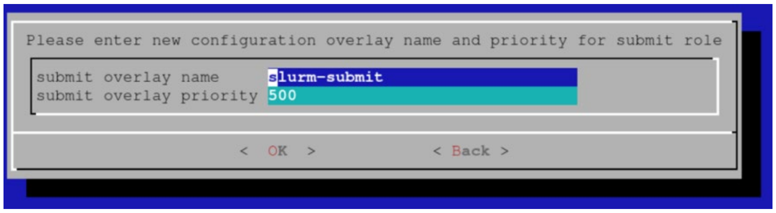
```
Select categories for Workload Manager submit role

[ ] default
[*] dgx
[ ] nsds
[ ] utilitynode2
```

17. Select additional nodes from where you will be submitting jobs (e.g. head node of the cluster).



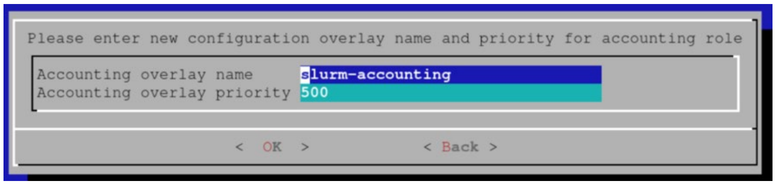
```
Please select nodes for Workload Manager submit role

[*] utilitynode-01
[ ] nsd-01          category:nsds
[ ] nsd-02          category:nsds
[ ] nsd-03          category:nsds
[ ] nsd-04          category:nsds
[ ] nsd-05          category:nsds
[ ] nsd-06          category:nsds
[ ] utilitynode-02  category:utilitynode2
```

18. Choose a name for the configuration overlay of submit hosts (the defaults will be fine):



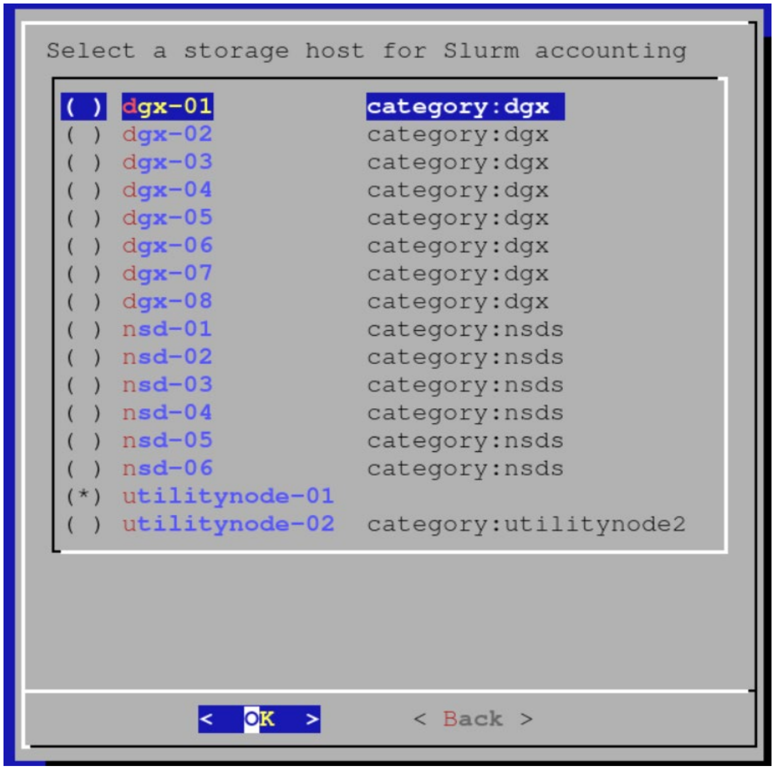
```
Please enter new configuration overlay name and priority for submit role

submit overlay name      slurm-submit
submit overlay priority  500
```

19. Choose a name for the configuration overlay of accounting nodes.



```
Please enter new configuration overlay name and priority for accounting role

Accounting overlay name      slurm-accounting
Accounting overlay priority  500

                  <  OK  >              < Back >
```

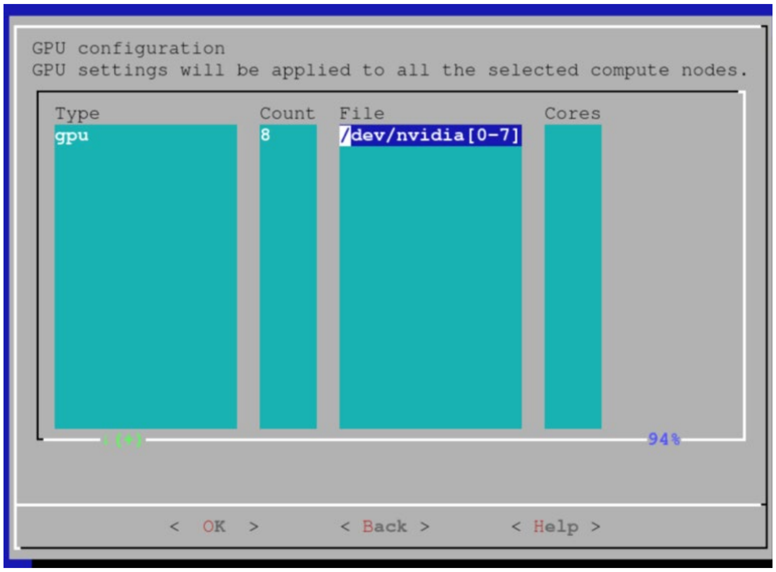
20. Select the head node as the accounting node.



```
Select a storage host for Slurm accounting

( ) dgx-01            category:dgx
( ) dgx-02            category:dgx
( ) dgx-03            category:dgx
( ) dgx-04            category:dgx
( ) dgx-05            category:dgx
( ) dgx-06            category:dgx
( ) dgx-07            category:dgx
( ) dgx-08            category:dgx
( ) nsd-01            category:nsds
( ) nsd-02            category:nsds
( ) nsd-03            category:nsds
( ) nsd-04            category:nsds
( ) nsd-05            category:nsds
( ) nsd-06            category:nsds
(*) utilitynode-01
( ) utilitynode-02    category:utilitynode2

          <  OK  >        < Back >
```

21. Add the 8 GPUs in each node as GPU resources that can be requested.

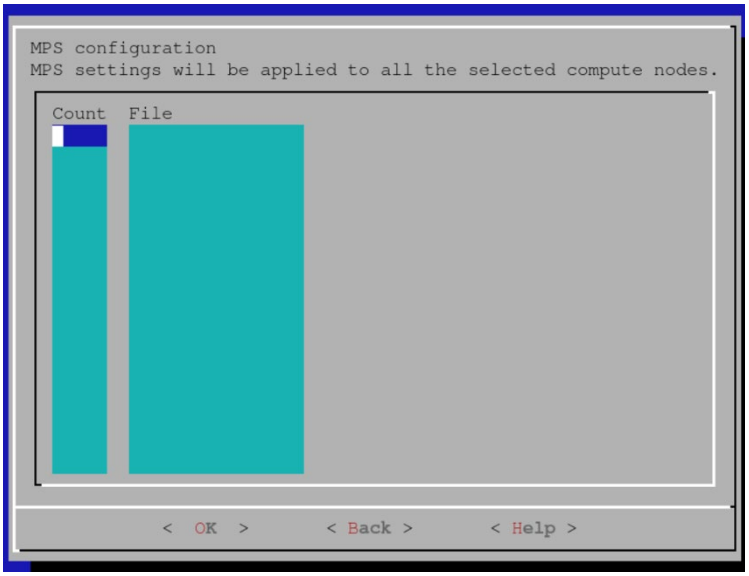
It is also possible to rely on the Slurm GPU autodetect capabilities. Consult the BCM documentation for details.



```
GPU configuration
GPU settings will be applied to all the selected compute nodes.

  Type            Count  File             Cores
  gpu             8      /dev/nvidia[0-7]

      <[*]>                                        94%

         <  OK  >       < Back >       < Help >
```

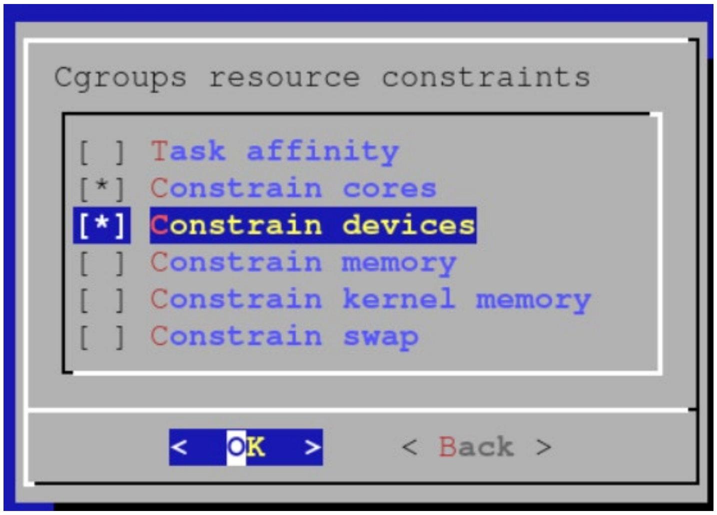
22. Unless CUDA Multi Process Management (MPS) will be used, leave the MPS settings empty.

    If MPS is to be configured, some additional setup steps will be needed to start/stop the MPS daemon through the prolog/epilog.



```
MPS configuration
MPS settings will be applied to all the selected compute nodes.

Count  File
```

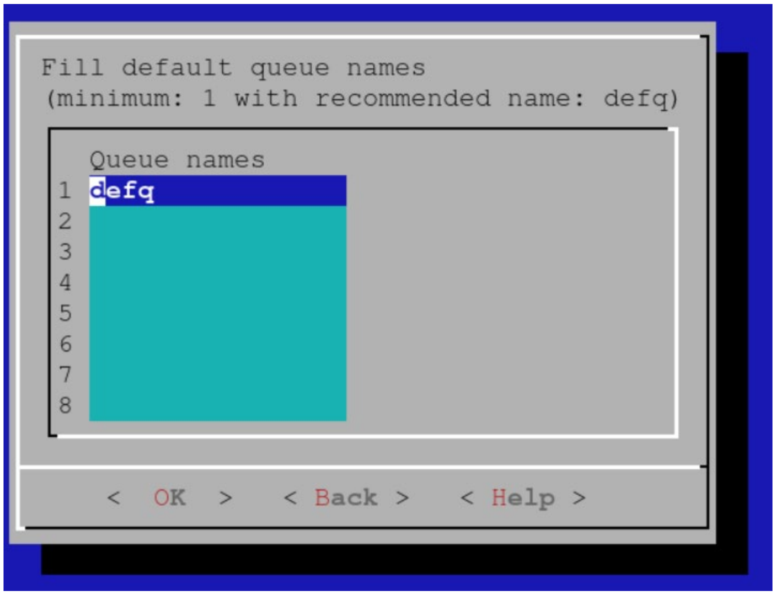
23. Enable the following cgroup resource constraints to make sure that jobs cannot use CPU cores or GPUs that they did not request:



```
Cgroups resource constraints

[ ] Task affinity
[*] Constrain cores
[*] Constrain devices
[ ] Constrain memory
[ ] Constrain kernel memory
[ ] Constrain swap
```

24. Create a default queue.

    More queues can always be defined later:

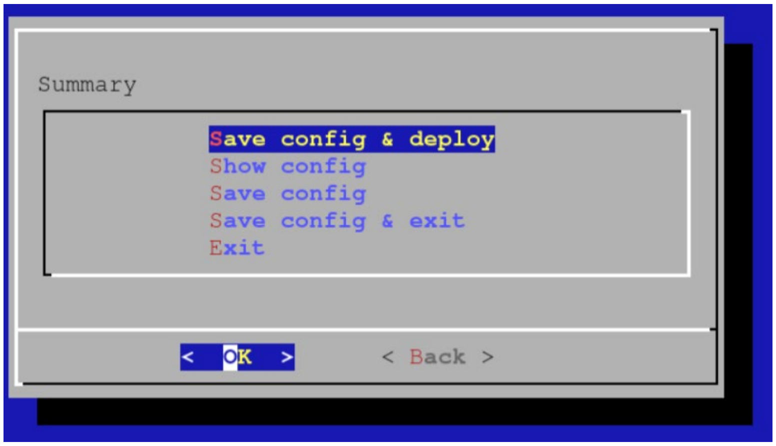


Fill default queue names
(minimum: 1 with recommended name: defq)

Queue names

1 defq

25. Choose `Save config & deploy` and then select `OK`.



26. Store the configuration for later.
27. After the setup completes, you will want to reboot all compute nodes using `cmsh`.

```
device power reset -c dgx
```

28. After the nodes come back up, you can verify that Slurm is working properly by checking:

```
[root@utilitynode-01 ~]# sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
defq*        up   infinite      8   idle dgx-[01-08]
```

29. By default, Slurm is configured to not allow multiple jobs on the same node. To change this behavior and allow (for example) a maximum of 8 simultaneous jobs to run on a single node.

```
[root@utilitynode-01 ~]# cmsh
[utilitynode-01]% wlm use slurm
[utilitynode-01->wlm[slurm]]% jobqueue
[utilitynode-01->wlm[slurm]->jobqueue]% use defq
[utilitynode-01->wlm[slurm]->jobqueue[defq]]% get oversubscribe
NO
[utilitynode-01->wlm[slurm]->jobqueue[defq]]% set oversubscribe YES:8
[utilitynode-01->wlm[slurm]->jobqueue*[defq*]]% commit
[utilitynode-01->wlm[slurm]->jobqueue[defq]]%
```

30. To verify that GPU reservation is working, first try allocating no GPUs.

```
[root@utilitynode-01 ~]# srun nvidia-smi
No devices were found
srun: error: dgx-06: task 0: Exited with exit code 6
[root@utilitynode-01 ~]#
```

31. Then try allocating, e.g., two GPUs.

```
[root@utilitynode-01 ~]# srun --gres=gpu:2 nvidia-smi
Thu Mar  4 08:50:44 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 450.102.04   Driver Version: 450.102.04   CUDA Version: 11.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  A100-SXM4-40GB      On   | 00000000:07:00.0 Off |                    0 |
| N/A   30C    P0    54W / 400W |      0MiB / 40537MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+
|   1  A100-SXM4-40GB      On   | 00000000:0F:00.0 Off |                    0 |
| N/A   30C    P0    53W / 400W |      0MiB / 40537MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

2.6 (Optional) Deploy Jupyter

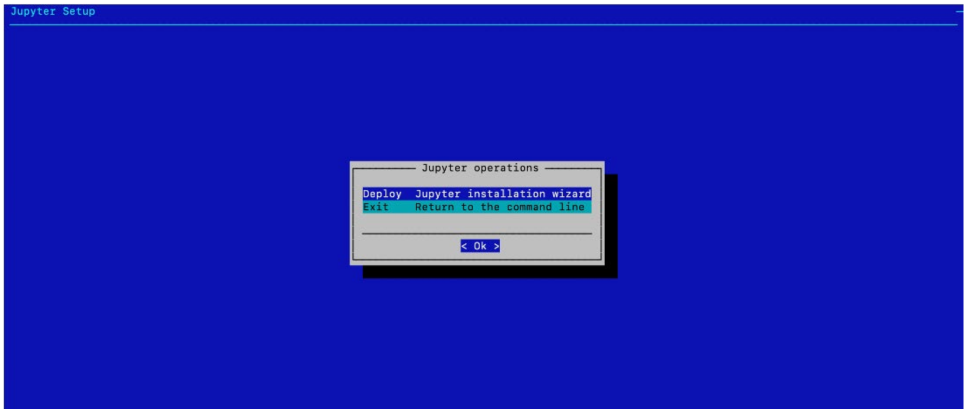
BCM provides a robust and popular Jupyter integration. Because the Jupyter integration distributes the kernel across the cluster through the HPC workload management system or Kubernetes, Jupyter is generally installed on the head node or on a login node.

2.6.1 Install Jupyter Using the CLI Wizard

1. Run the `cm-jupyter-setup` CLI wizard on the head node as the root user.

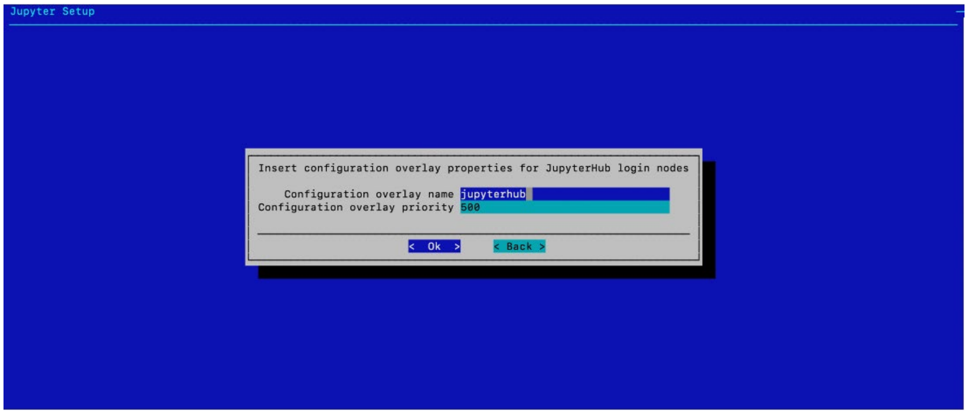
   ```
   # cm-jupyter-setup
   ```

2. Choose `Deploy` to continue.


Jupyter Setup
Jupyter operations
Deploy   Jupyter installation wizard
Exit   Return to the command line
< Ok >

3. Specify the overlay name and priority for the JupyterHub login nodes.

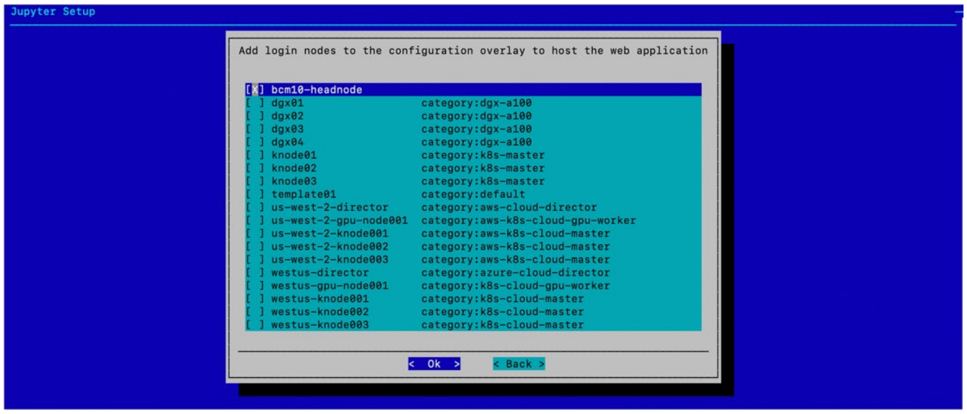
   By default, the Jupyter wizard will create a configuration overlay named `jupyterhub` with a priority of `500`. Use the defaults unless there is an existing `jupyterhub` overlay.



```
Insert configuration overlay properties for JupyterHub login nodes

   Configuration overlay name jupyterhub
Configuration overlay priority 500
```

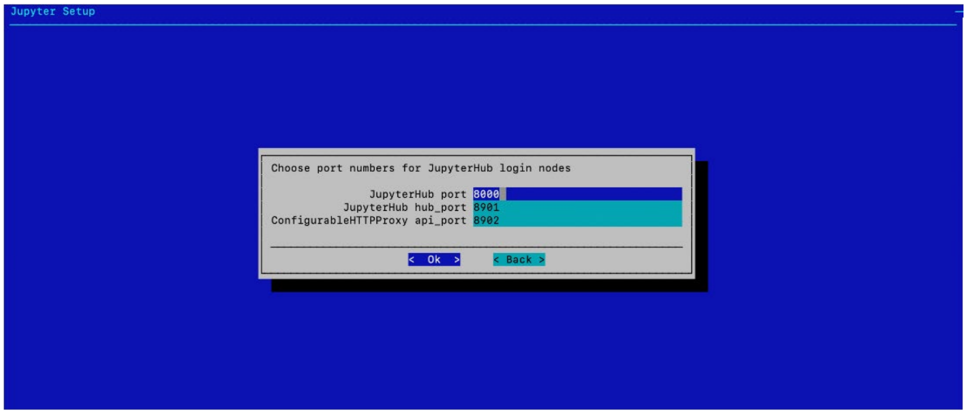
4. Select `bcm10-headnode` and then `Ok`.

   After HA is configured, the `cm-jupyterhub` service will be set to always run on the active head node.



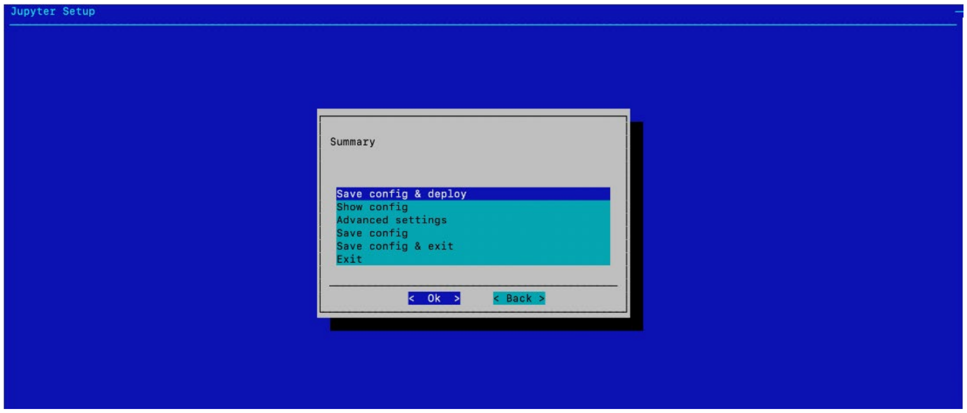
```
Add login nodes to the configuration overlay to host the web application

[X] bcm10-headnode
[ ] dgx01                  category:dgx-a100
[ ] dgx02                  category:dgx-a100
[ ] dgx03                  category:dgx-a100
[ ] dgx04                  category:dgx-a100
[ ] knode01                category:k8s-master
[ ] knode02                category:k8s-master
[ ] knode03                category:k8s-master
[ ] template01             category:default
[ ] us-west-2-director     category:aws-cloud-director
[ ] us-west-2-gpu-node001  category:aws-k8s-cloud-gpu-worker
[ ] us-west-2-knode001     category:aws-k8s-cloud-master
[ ] us-west-2-knode002     category:aws-k8s-cloud-master
[ ] us-west-2-knode003     category:aws-k8s-cloud-master
[ ] westus-director        category:azure-cloud-director
[ ] westus-gpu-node001     category:k8s-cloud-gpu-worker
[ ] westus-knode001        category:k8s-cloud-master
[ ] westus-knode002        category:k8s-cloud-master
[ ] westus-knode003        category:k8s-cloud-master
```

5. Select the default ports of `8000`, `8901`, and `8902` and select `Ok`.
   Users will access it on the active head node on port `8000`.



```
Jupyter Setup

Choose port numbers for JupyterHub login nodes

             JupyterHub port 8000
         JupyterHub hub_port 8901
ConfigurableHTTPProxy api_port 8902
```

6. Select `Save config & deploy` and then `Ok`.

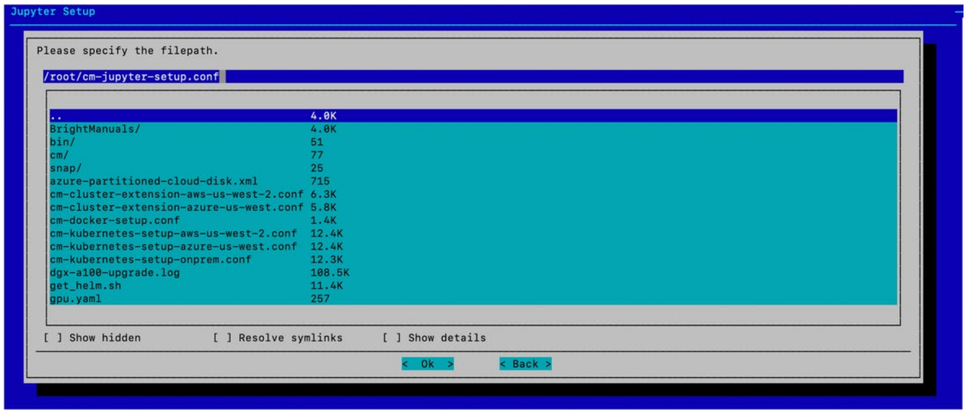


Jupyter Setup

Summary

7. Select `Ok` to start the installation.

   By default, the Jupyter wizard will save the deployment configuration in `/root/cm-jupyter-setup.conf`. This configuration file can be used to redeploy Jupyter in the future.

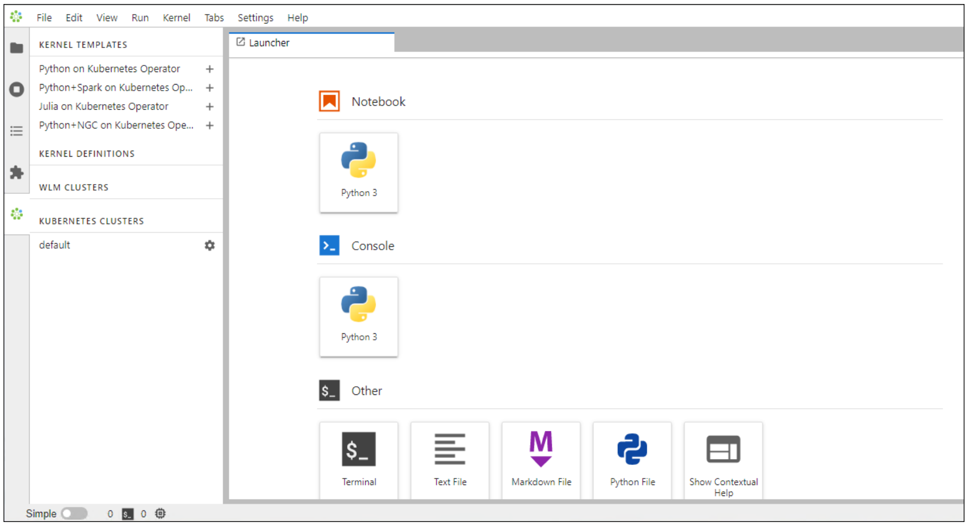


```
Jupyter Setup

Please specify the filepath.

/root/cm-jupyter-setup.conf

..                                       4.0K
BrightManuals/                           4.0K
bin/                                     51
cm/                                      77
snap/                                    25
azure-partitioned-cloud-disk.xml         715
cm-cluster-extension-aws-us-west-2.conf  6.3K
cm-cluster-extension-azure-us-west.conf  5.8K
cm-docker-setup.conf                     1.4K
cm-kubernetes-setup-aws-us-west-2.conf   12.4K
cm-kubernetes-setup-azure-us-west.conf   12.4K
cm-kubernetes-setup-onprem.conf          12.3K
dgx-a100-upgrade.log                     108.5K
get_helm.sh                              11.4K
gpu.yaml                                 257

[ ] Show hidden          [ ] Resolve symlinks       [ ] Show details

                        <  Ok  >       < Back >
```

8. When the installation completes, the `cm-jupyter` service will automatically be started on the selected node.

   All users in the cluster (except the root user) will be able to login to Jupyterhub using a web browser at `http://<head-node-ip or FQDN>:8000`.

   Example: http://10.227.52.254:8000



9. If needed, a test user can be created with the following command:

```
# cmsh -c "user; add jupyterhubuser; set password jupyterhubuser; commit"
```

10. Add the user to K8s.

```
# cm-kubernetes-setup --add-user jupyterhubuser --operators cm-jupyter-kernel-operator
```

Chapter 3. High Availability

This section covers how to configure high availability (HA) using `cmha-setup` CLI wizard.

1. Ensure that both head nodes are licensed.

   We provided the MAC address for the secondary head when we installed the cluster license (Section 3.2.2.1).

   ```
   % main licenseinfo  | grep ^MAC
   MAC address / Cloud ID               04:3F:72:E7:67:07|14:02:EC:DA:AF:18
   ```

2. Configure the NFS shared storage.

   Mounts configured in `fsmounts` will be automatically mounted by the `CMDaemon`.

   ```
   % device
   % use master
   % fsmounts
   % add /nfs/general
   % set device 10.227.48.252:/var/nfs/general
   % set filesystem nfs
   % commit
   % show
   Parameter                        Value
   ---------------------------- ------------------------------------------------
   Device                           10.227.48.252:/var/nfs/general
   Revision
   Filesystem                       nfs
   Mountpoint                       /nfs/general
   Dump                             no
   RDMA                             no
   Filesystem Check                 NONE
   Mount options                    defaults
   ```

3. Verify that the shared storage is mounted.

   ```
   # mount | grep '/nfs/general'
   10.227.48.252:/var/nfs/general on /nfs/general type nfs4
   (rw,relatime,vers=4.2,rsize=1048576,wsize=1048576,namlen=255,hard,proto=tcp,timeo=600,retra
   ns=2,sec=sys,clientaddr=10.130.12210.227.48_lock=none,addr=10.130.122.252)10.227.48
   ```

4. Verify that head node has power control over the cluster nodes.

```
% device
% power -c dgx,k8s-master status
[basepod-head1->device]% power -c dgx,k8s-master status
ipmi0 .................... [    ON     ] dgx01
ipmi0 .................... [    ON     ] dgx02
ipmi0 .................... [    ON     ] dgx03
ipmi0 .................... [    ON     ] dgx04
ipmi0 .................... [    ON     ] knode01
ipmi0 .................... [    ON     ] knode02
ipmi0 .................... [    ON     ] knode03
[basepod-head1->device]%
```

5. Power off the cluster nodes.

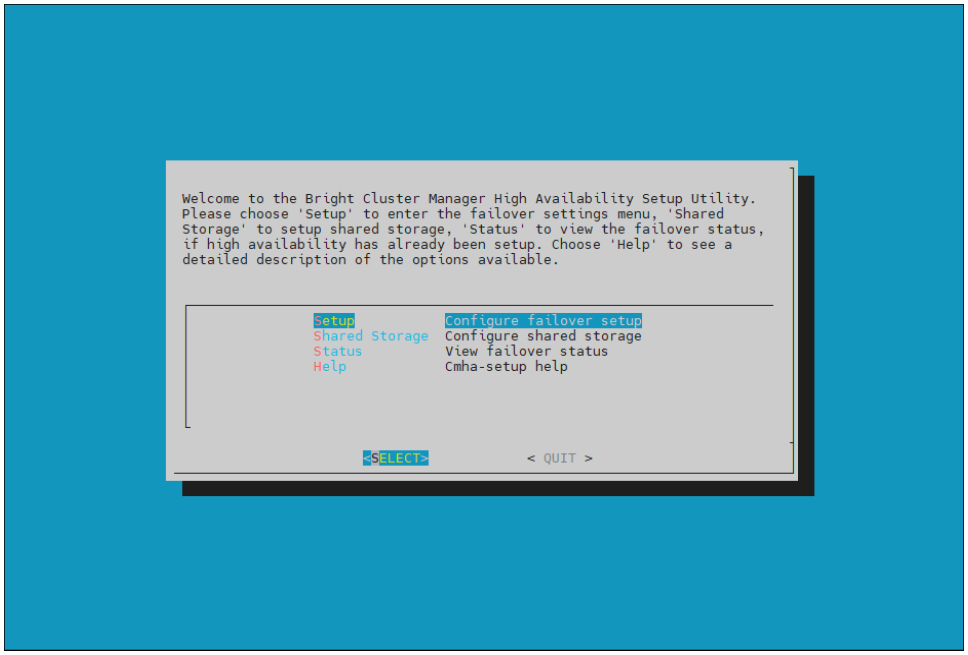
   The cluster nodes must be powered off before configuring HA.

```
% power -c k8s-master,dgx off
ipmi0 .................... [    OFF    ] knode01
ipmi0 .................... [    OFF    ] knode02
ipmi0 .................... [    OFF    ] knode03
ipmi0 .................... [    OFF    ] dgx01
ipmi0 .................... [    OFF    ] dgx02
ipmi0 .................... [    OFF    ] dgx03
ipmi0 .................... [    OFF    ] dgx04
```

6. Start the `cmha-setup` CLI wizard as the root user on the primary head node.

```
# cmha-setup
```

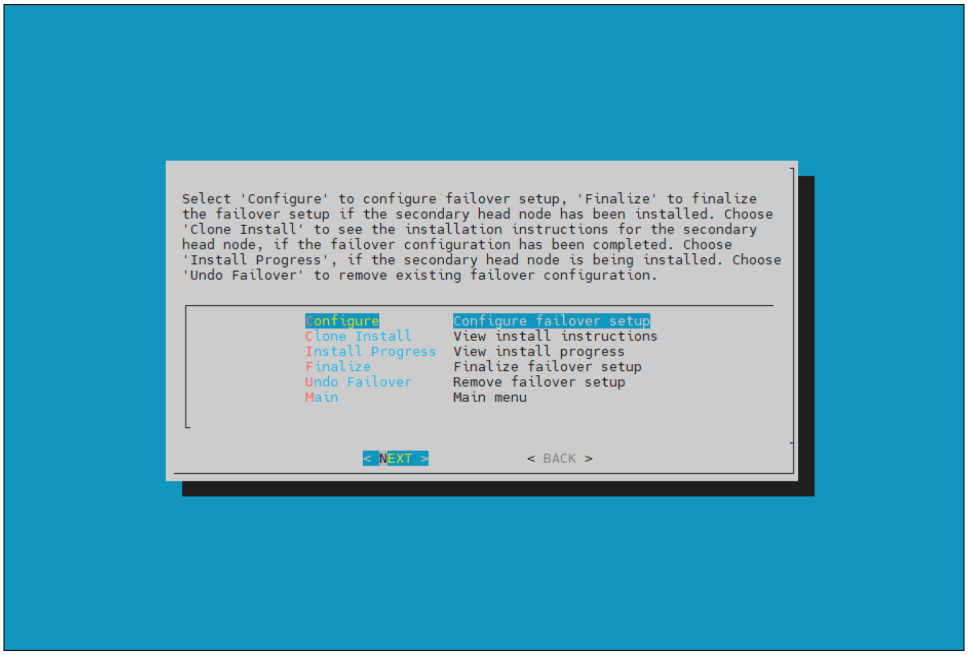
7. Choose `Setup` and then select `SELECT`.



Welcome to the Bright Cluster Manager High Availability Setup Utility. Please choose 'Setup' to enter the failover settings menu, 'Shared Storage' to setup shared storage, 'Status' to view the failover status, if high availability has already been setup. Choose 'Help' to see a detailed description of the options available.

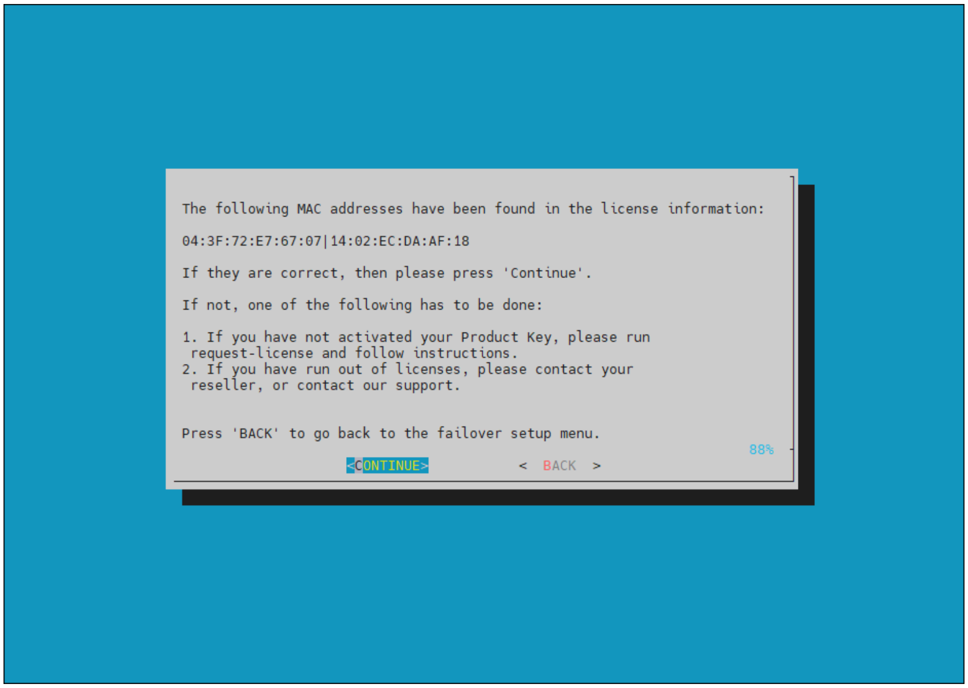
Setup   Configure failover setup
Shared Storage Configure shared storage
Status   View failover status
Help   Cmha-setup help
<SELECT>
< QUIT >

8. Choose `Configure` and then select `NEXT`



```
Select 'Configure' to configure failover setup, 'Finalize' to finalize
the failover setup if the secondary head node has been installed. Choose
'Clone Install' to see the installation instructions for the secondary
head node, if the failover configuration has been completed. Choose
'Install Progress', if the secondary head node is being installed. Choose
'Undo Failover' to remove existing failover configuration.

               Configure          Configure failover setup
               Clone Install      View install instructions
               Install Progress   View install progress
               Finalize           Finalize failover setup
               Undo Failover      Remove failover setup
               Main               Main menu

                     < NEXT >               < BACK >
```

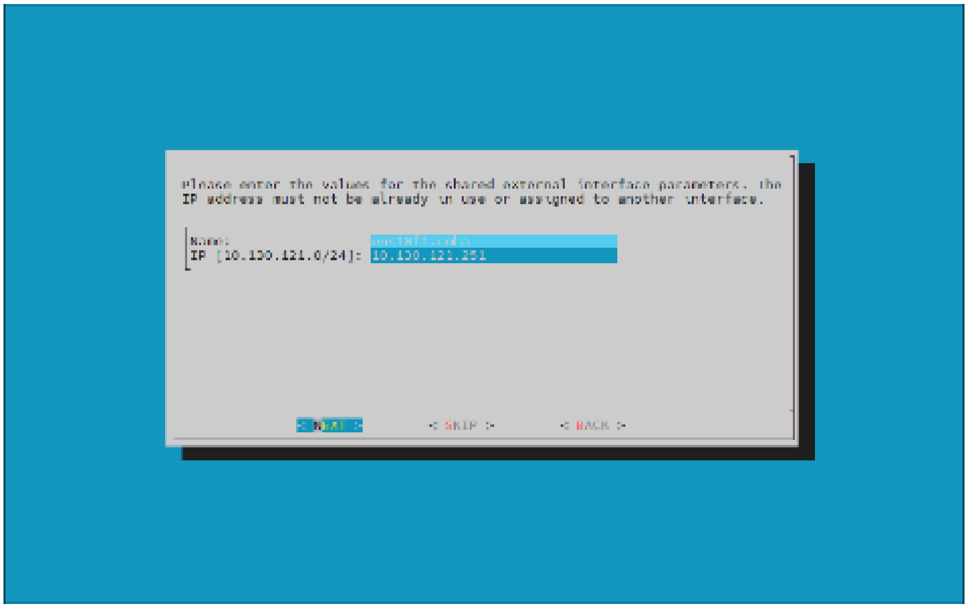
9. Verify that the cluster license information found by the wizard is correct and then select `CONTINUE`.



```
The following MAC addresses have been found in the license information:

04:3F:72:E7:67:07|14:02:EC:DA:AF:18

If they are correct, then please press 'Continue'.

If not, one of the following has to be done:

1. If you have not activated your Product Key, please run
 request-license and follow instructions.
2. If you have run out of licenses, please contact your
 reseller, or contact our support.


Press 'BACK' to go back to the failover setup menu.
                                                                    88%
                    <CONTINUE>              <  BACK  >
```

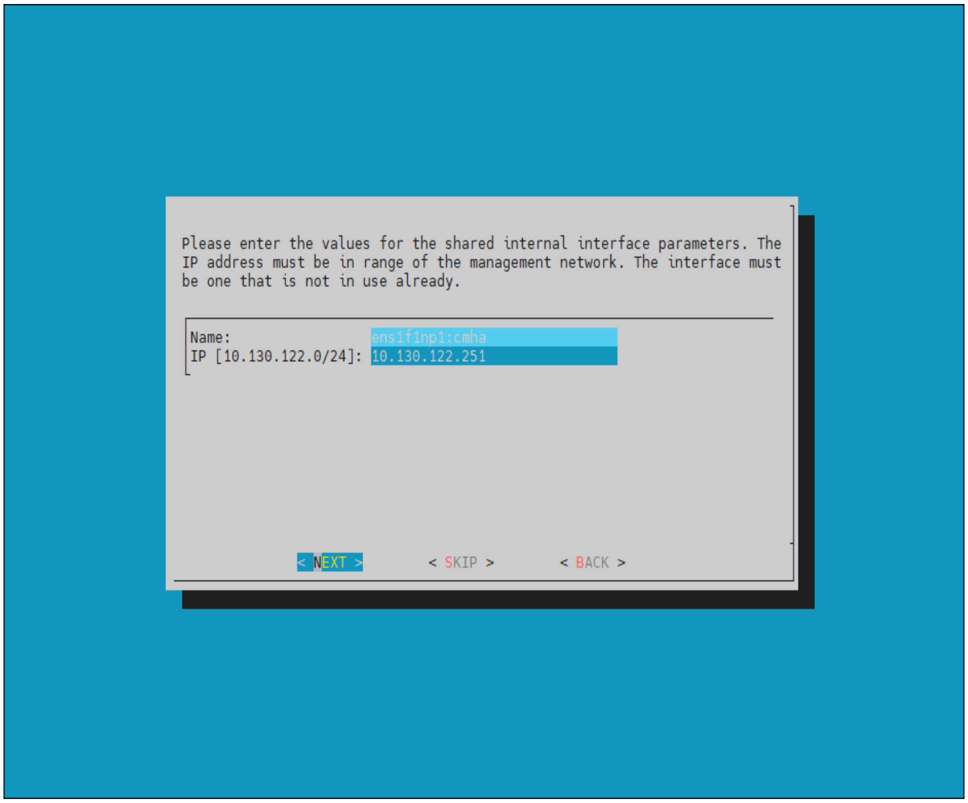
10. Configure an external virtual IP address to be used by the active head node in the HA configuration and then select NEXT.

    This will be the IP that should always be used for accessing the active head nodes.



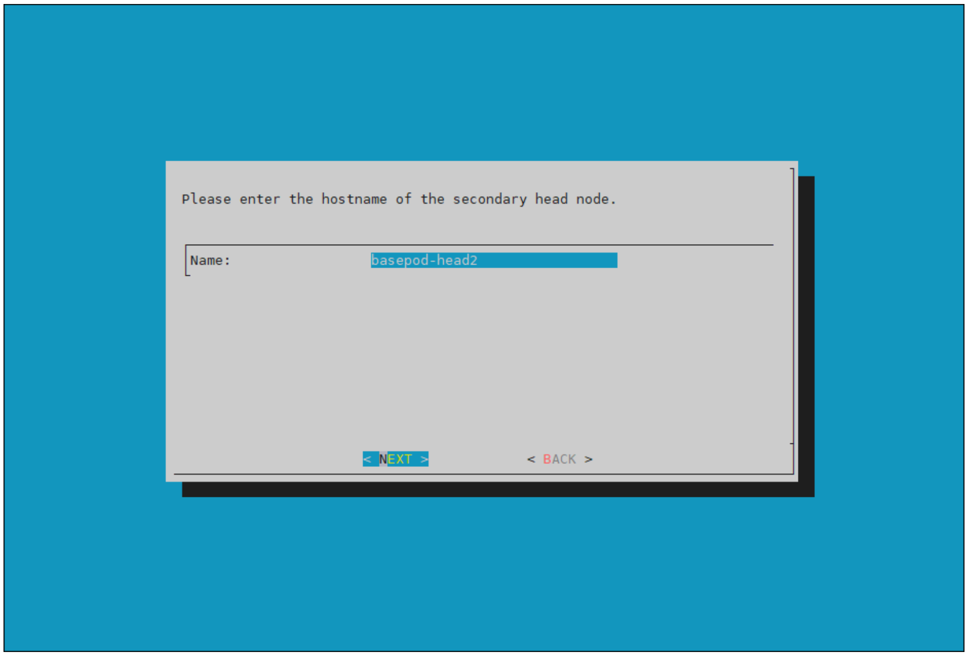
```
Please enter the values for the shared external interface parameters. The
IP address must not be already in use or assigned to another interface.

Name:                 ens10f1:cmha
IP [10.130.121.0/24]: 10.130.121.251
```

11. Provide an internal virtual IP address that will be used by the active head node in the HA configuration and then select NEXT.



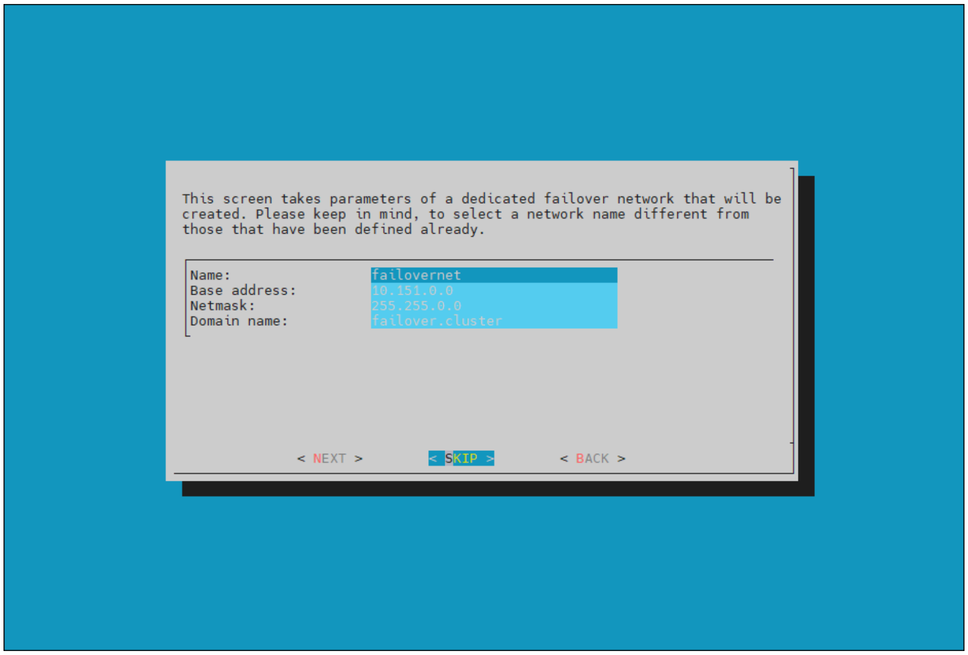
```
Please enter the values for the shared internal interface parameters. The
IP address must be in range of the management network. The interface must
be one that is not in use already.

Name:                 ens1f1np1:cmha
IP [10.130.122.0/24]: 10.130.122.251
```

12. Provide the name of the secondary head node and then select `NEXT`.



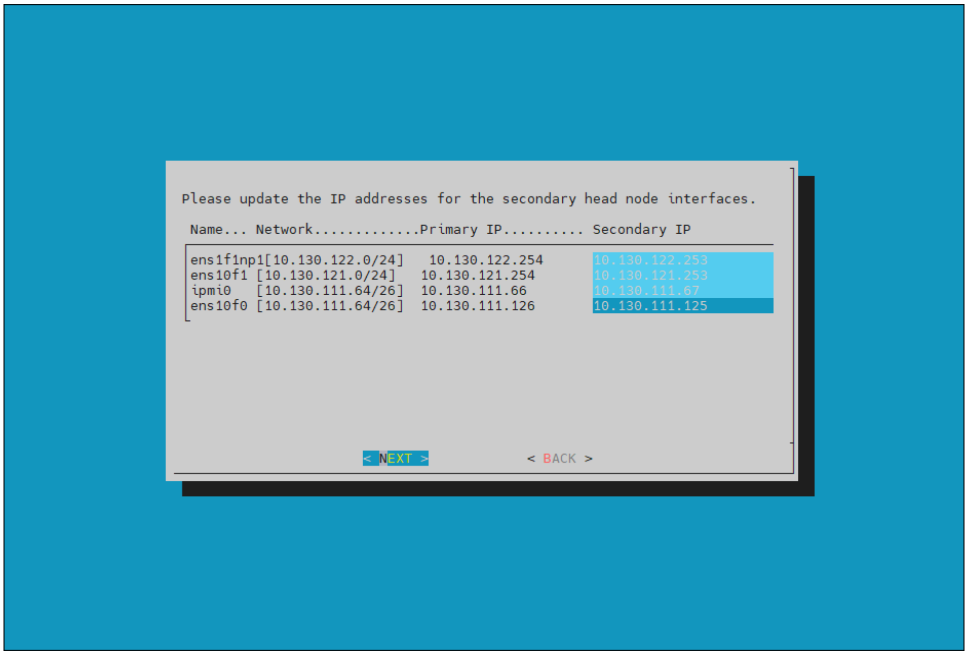
```
Please enter the hostname of the secondary head node.

Name:                      basepod-head2
```

13. DGX BasePOD uses the internal network as the failover network, so select `SKIP`.



```
This screen takes parameters of a dedicated failover network that will be
created. Please keep in mind, to select a network name different from
those that have been defined already.

Name:                      failovernet
Base address:              10.151.0.0
Netmask:                   255.255.0.0
Domain name:               failover.cluster
```

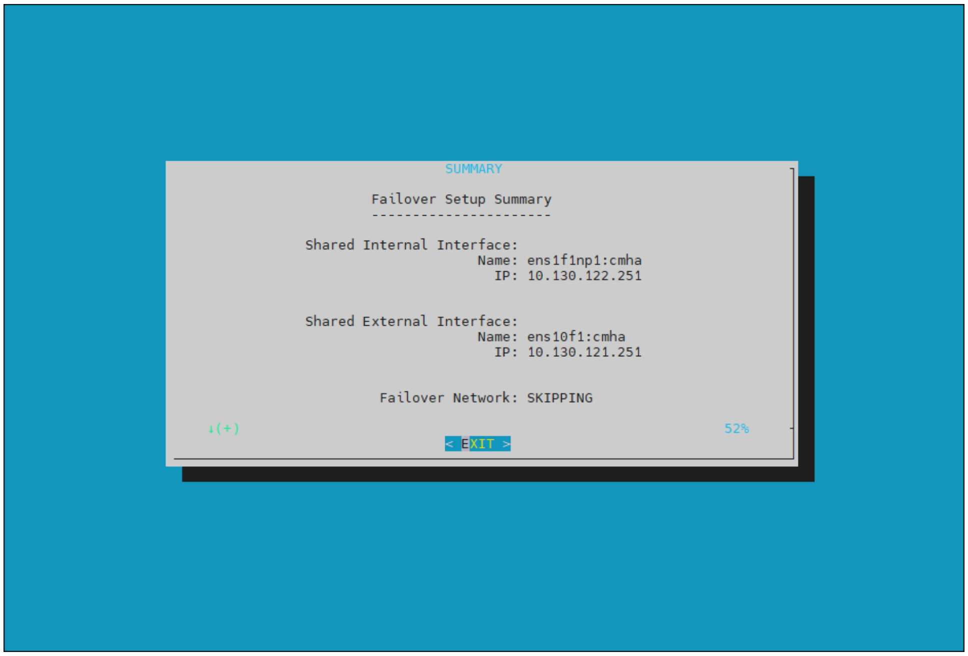
14. Configure the IP addresses for the secondary head node and then select `NEXT`



```
Please update the IP addresses for the secondary head node interfaces.

 Name... Network............Primary IP.......... Secondary IP

 ens1f1np1[10.130.122.0/24]   10.130.122.254    10.130.122.253
 ens10f1 [10.130.121.0/24]   10.130.121.254     10.130.121.253
 ipmi0   [10.130.111.64/26]  10.130.111.66      10.130.111.67
 ens10f0 [10.130.111.64/26]  10.130.111.126     10.130.111.125

                       < NEXT >              < BACK >
```

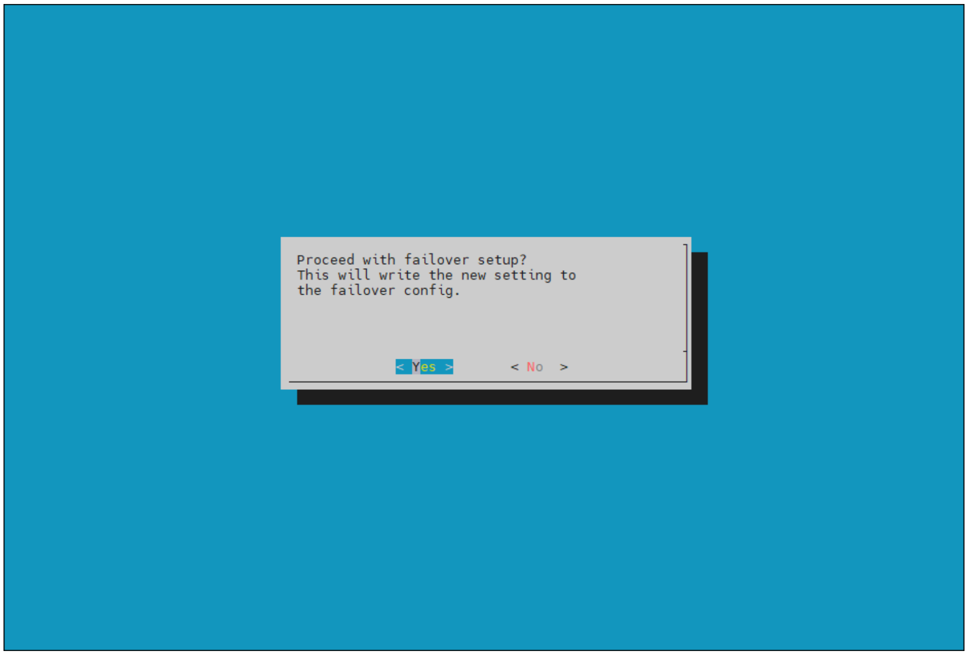
15. Review the summary of the configuration and then select `NEXT`.

    This screen shoes the VIP that will be assigned to the internal and external interfaces.



```
                                 SUMMARY

                          Failover Setup Summary
                          ----------------------

                  Shared Internal Interface:
                                      Name: ens1f1np1:cmha
                                        IP: 10.130.122.251


                  Shared External Interface:
                                      Name: ens10f1:cmha
                                        IP: 10.130.121.251


                           Failover Network: SKIPPING

     ↓(+)                                                           52%
                                < EXIT >
```

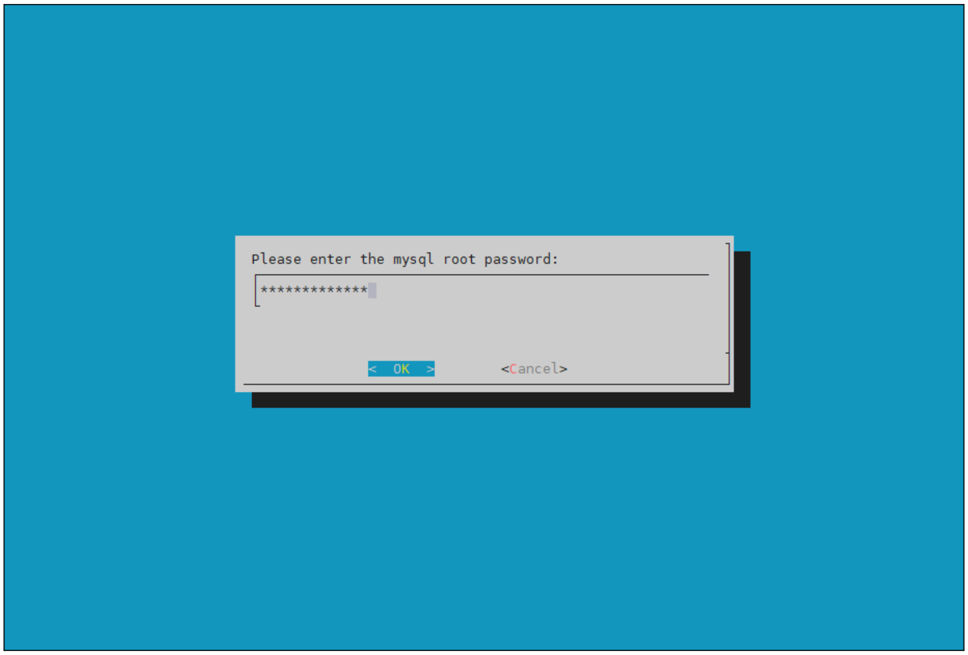
16. Select Yes to proceed with the failover configuration.



```
Proceed with failover setup?
This will write the new setting to
the failover config.
```

17. Enter the MySQL root password and then select OK.

    This should be the same as the root password.

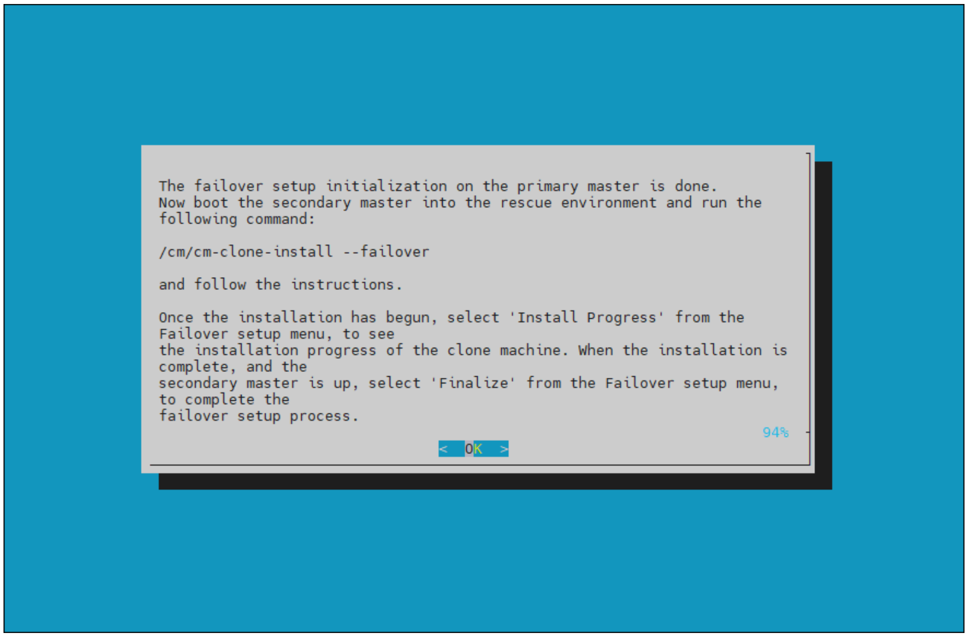


```
Please enter the mysql root password:
*************
```

18. The wizard implements the first steps in the HA configuration. If all the steps show `OK`, press `ENTER` to continue. The progress is shown below:

```
Initializing failover setup on master............. [  OK  ]
Updating shared internal interface................ [  OK  ]
Updating shared external interface................ [  OK  ]
Updating extra shared internal interfaces......... [  OK  ]
Cloning head node................................. [  OK  ]
Updating secondary master interfaces.............. [  OK  ]
Updating Failover Object.......................... [  OK  ]
Restarting cmdaemon............................... [  OK  ]
Press any key to continue
```

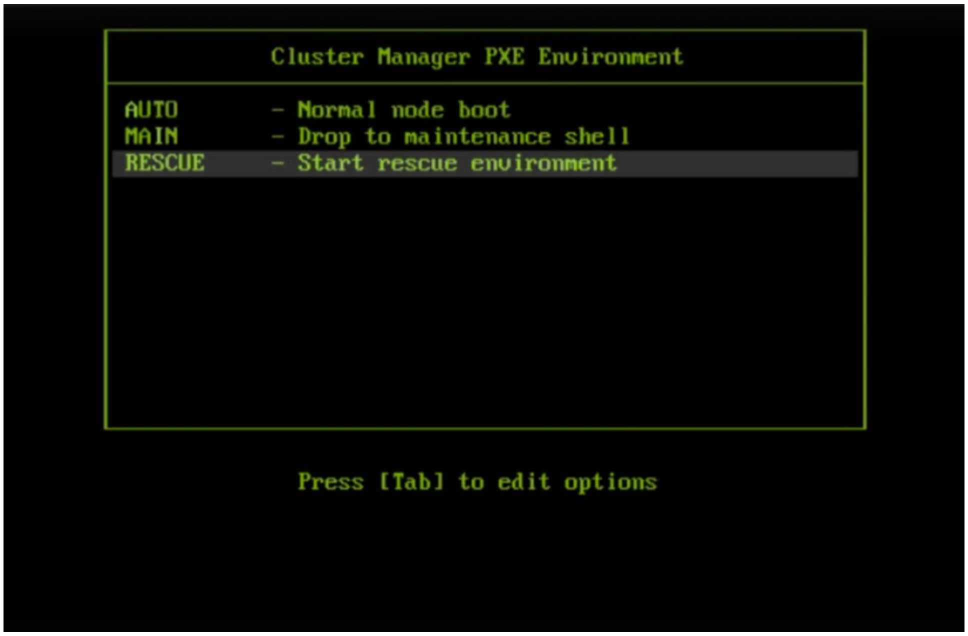
19. When the failover setup installation on the primary master is complete, select `OK` to exit the wizard.



```
The failover setup initialization on the primary master is done.
Now boot the secondary master into the rescue environment and run the
following command:

/cm/cm-clone-install --failover

and follow the instructions.

Once the installation has begun, select 'Install Progress' from the
Failover setup menu, to see
the installation progress of the clone machine. When the installation is
complete, and the
secondary master is up, select 'Finalize' from the Failover setup menu,
to complete the
failover setup process.
```

20. PXE boot the secondary head node and then select `RESCUE` from the grub menu.

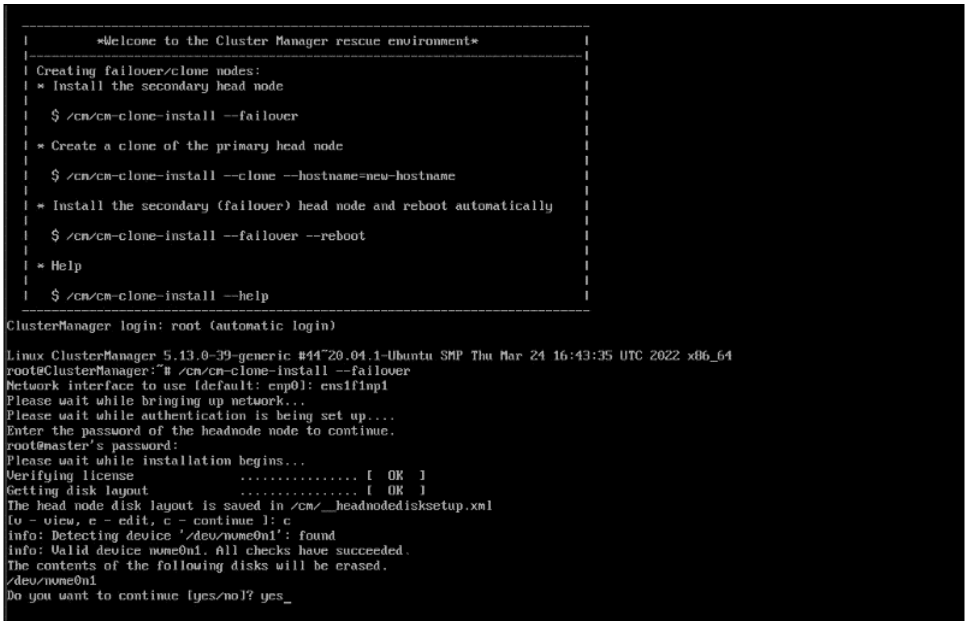
    Since this is the initial boot of this node, it must be done outside of Base Command Manager (BMC or physical power button).



```
Cluster Manager PXE Environment

AUTO      - Normal node boot
MAIN      - Drop to maintenance shell
RESCUE    - Start rescue environment

Press [Tab] to edit options
```

21. After the secondary head node has booted into the rescue environment, run the `/cm/cm-clone-install –failover` command, then enter `yes` when prompted.

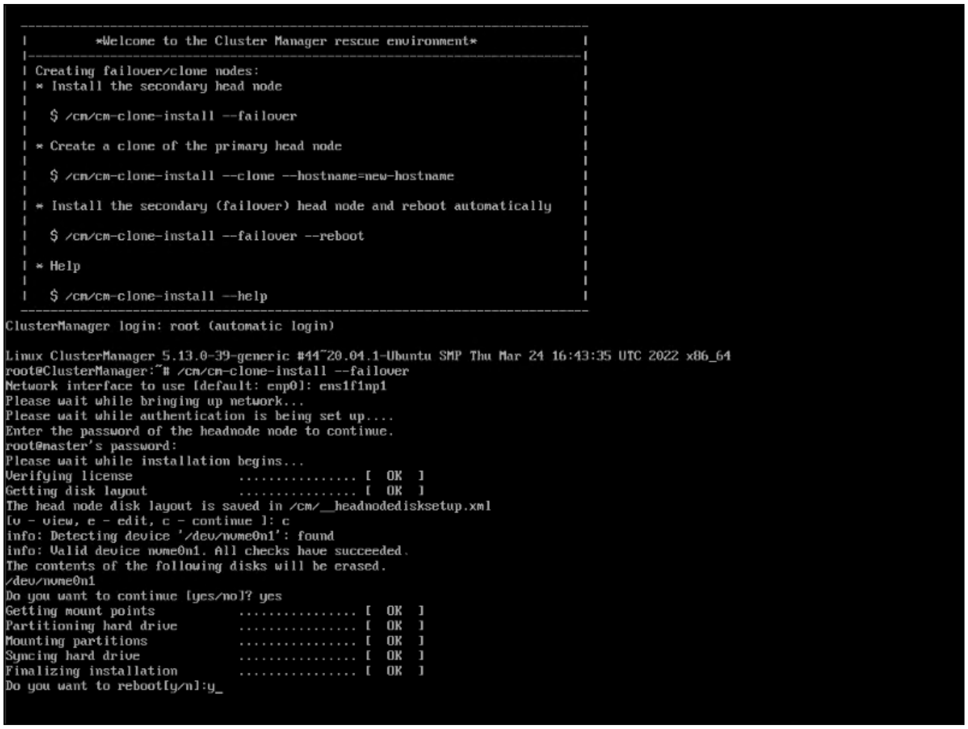
    The secondary head node will be cloned from the primary.



```
 ---------------------------------------------------------------------------
 |         *Welcome to the Cluster Manager rescue environment*             |
 |-------------------------------------------------------------------------|
 | Creating failover/clone nodes:                                          |
 | * Install the secondary head node                                       |
 |                                                                         |
 |   $ /cm/cm-clone-install --failover                                     |
 |                                                                         |
 | * Create a clone of the primary head node                               |
 |                                                                         |
 |   $ /cm/cm-clone-install --clone --hostname=new-hostname                |
 |                                                                         |
 | * Install the secondary (failover) head node and reboot automatically   |
 |                                                                         |
 |   $ /cm/cm-clone-install --failover --reboot                            |
 |                                                                         |
 | * Help                                                                  |
 |                                                                         |
 |   $ /cm/cm-clone-install --help                                         |
 ---------------------------------------------------------------------------
ClusterManager login: root (automatic login)

Linux ClusterManager 5.13.0-39-generic #44~20.04.1-Ubuntu SMP Thu Mar 24 16:43:35 UTC 2022 x86_64
root@ClusterManager:~# /cm/cm-clone-install --failover
Network interface to use [default: enp0]: ens1f1np1
Please wait while bringing up network...
Please wait while authentication is being set up....
Enter the password of the headnode node to continue.
root@master's password:
Please wait while installation begins...
Verifying license              ................ [  OK  ]
Getting disk layout            ................ [  OK  ]
The head node disk layout is saved in /cm/__headnodedisksetup.xml
[v - view, e - edit, c - continue ]: c
info: Detecting device '/dev/nvme0n1': found
info: Valid device nvme0n1. All checks have succeeded.
The contents of the following disks will be erased.
/dev/nvme0n1
Do you want to continue [yes/no]? yes_
```

22. When cloning is completed, enter `y` to reboot the secondary head node.

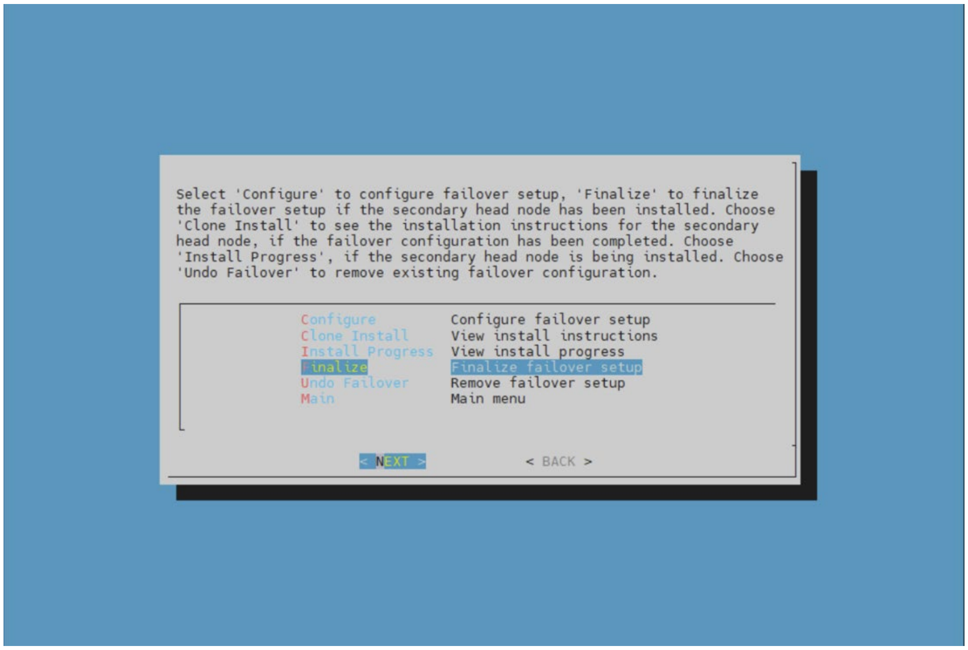
    The secondary must boot from its hard drive. PXE boot should not be enabled.



```
 -----------------------------------------------------------------------------
 |         *Welcome to the Cluster Manager rescue environment*               |
 |---------------------------------------------------------------------------|
 | Creating failover/clone nodes:                                            |
 | * Install the secondary head node                                         |
 |                                                                           |
 |   $ /cm/cm-clone-install --failover                                       |
 |                                                                           |
 | * Create a clone of the primary head node                                 |
 |                                                                           |
 |   $ /cm/cm-clone-install --clone --hostname=new-hostname                  |
 |                                                                           |
 | * Install the secondary (failover) head node and reboot automatically     |
 |                                                                           |
 |   $ /cm/cm-clone-install --failover --reboot                              |
 |                                                                           |
 | * Help                                                                    |
 |                                                                           |
 |   $ /cm/cm-clone-install --help                                           |
 -----------------------------------------------------------------------------
ClusterManager login: root (automatic login)

Linux ClusterManager 5.13.0-39-generic #44~20.04.1-Ubuntu SMP Thu Mar 24 16:43:35 UTC 2022 x86_64
root@ClusterManager:~# /cm/cm-clone-install --failover
Network interface to use [default: enp0]: ens1f1np1
Please wait while bringing up network...
Please wait while authentication is being set up....
Enter the password of the headnode node to continue.
root@master's password:
Please wait while installation begins...
Verifying license              ................ [  OK  ]
Getting disk layout            ................ [  OK  ]
The head node disk layout is saved in /cm/__headnodedisksetup.xml
[v - view, e - edit, c - continue ]: c
info: Detecting device '/dev/nvme0n1': found
info: Valid device nvme0n1. All checks have succeeded.
The contents of the following disks will be erased.
/dev/nvme0n1
Do you want to continue [yes/no]? yes
Getting mount points           ................ [  OK  ]
Partitioning hard drive        ................ [  OK  ]
Mounting partitions            ................ [  OK  ]
Syncing hard drive             ................ [  OK  ]
Finalizing installation        ................ [  OK  ]
Do you want to reboot[y/n]:y_
```

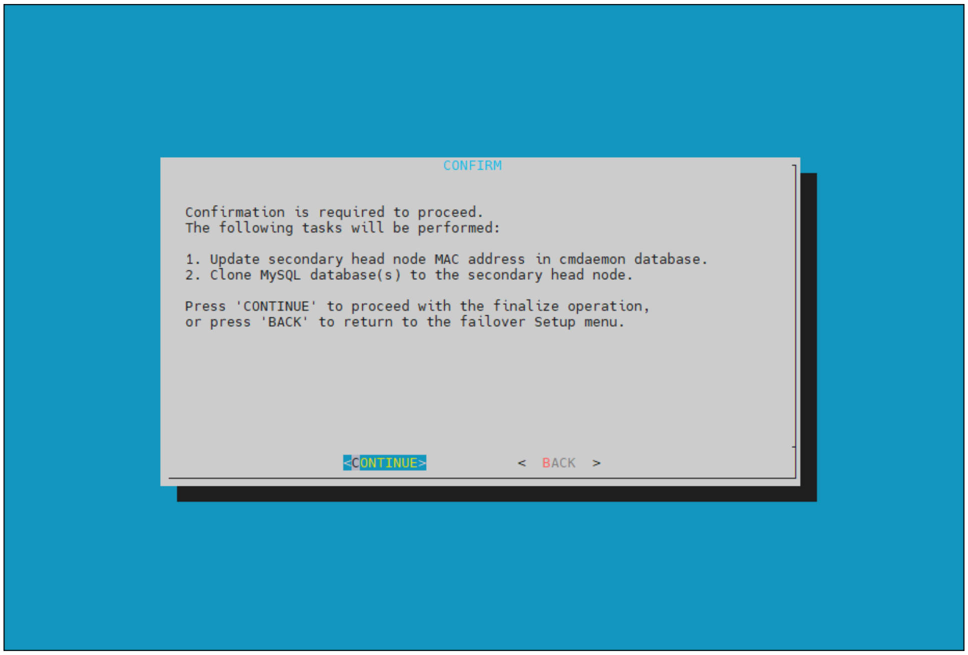
23. Wait for the secondary head node to reboot and then continue the HA setup procedure on the primary head node.
24. Choose `finalize` from the `cmha-setup` menu and then select `NEXT`

    This will clone the MySQL database from the primary to the secondary head node.



Select 'Configure' to configure failover setup, 'Finalize' to finalize the failover setup if the secondary head node has been installed. Choose 'Clone Install' to see the installation instructions for the secondary head node, if the failover configuration has been completed. Choose 'Install Progress', if the secondary head node is being installed. Choose 'Undo Failover' to remove existing failover configuration.

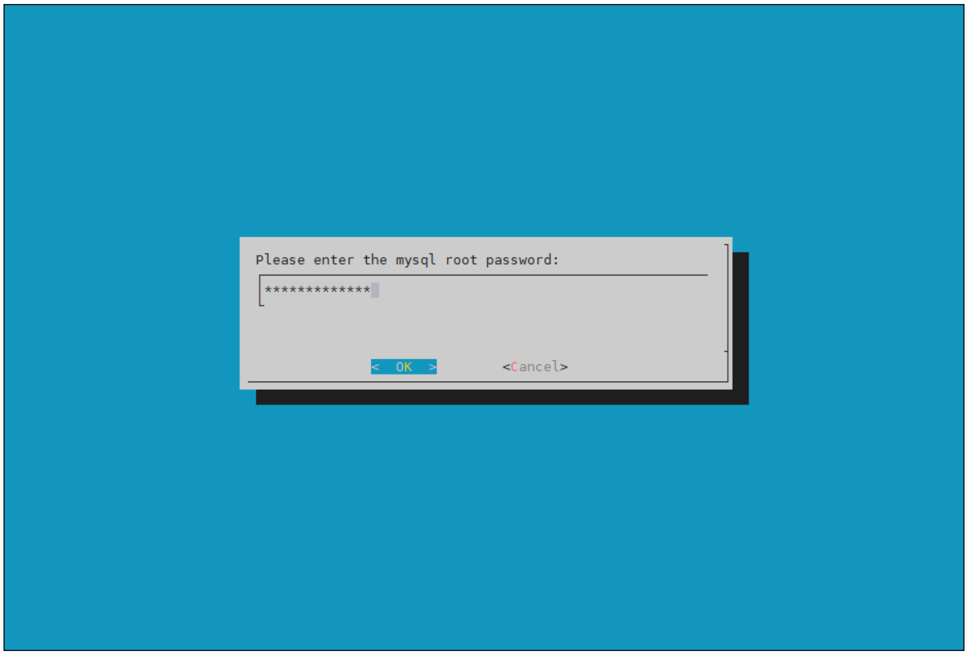
25. Select `CONTINUE` on the confirmation screen.



```
CONFIRM

Confirmation is required to proceed.
The following tasks will be performed:

1. Update secondary head node MAC address in cmdaemon database.
2. Clone MySQL database(s) to the secondary head node.

Press 'CONTINUE' to proceed with the finalize operation,
or press 'BACK' to return to the failover Setup menu.

<CONTINUE>            <  BACK  >
```

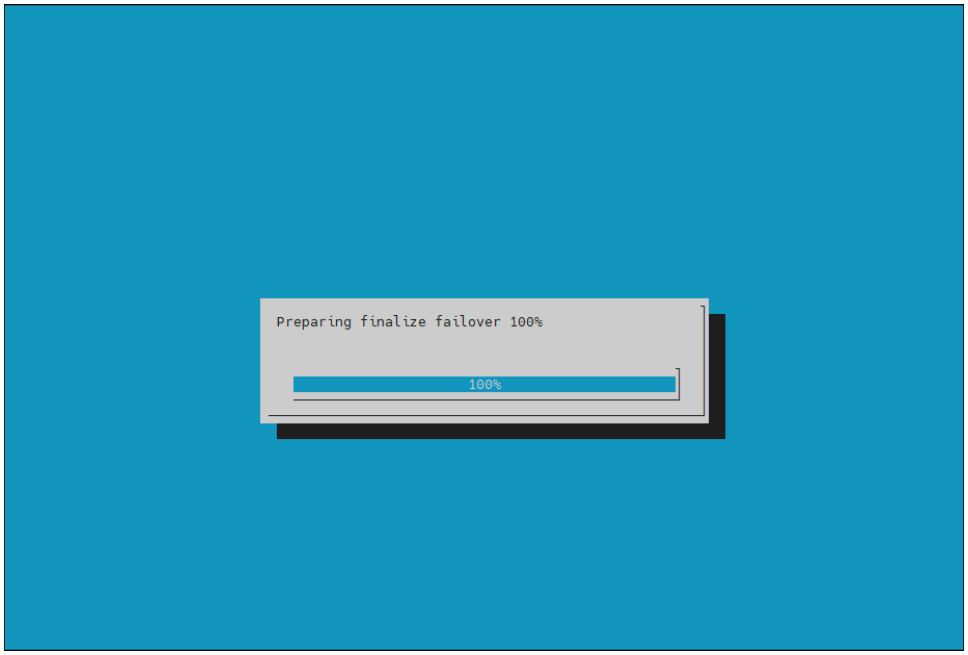
26. Enter the MySQL root password and then select `OK`.

    This should be the same as the root password.



```
Please enter the mysql root password:
*************

<  OK  >          <Cancel>
```

27. The `cmha-setup` wizard continues. Press `ENTER` to continue when prompted.



Preparing finalize failover 100%

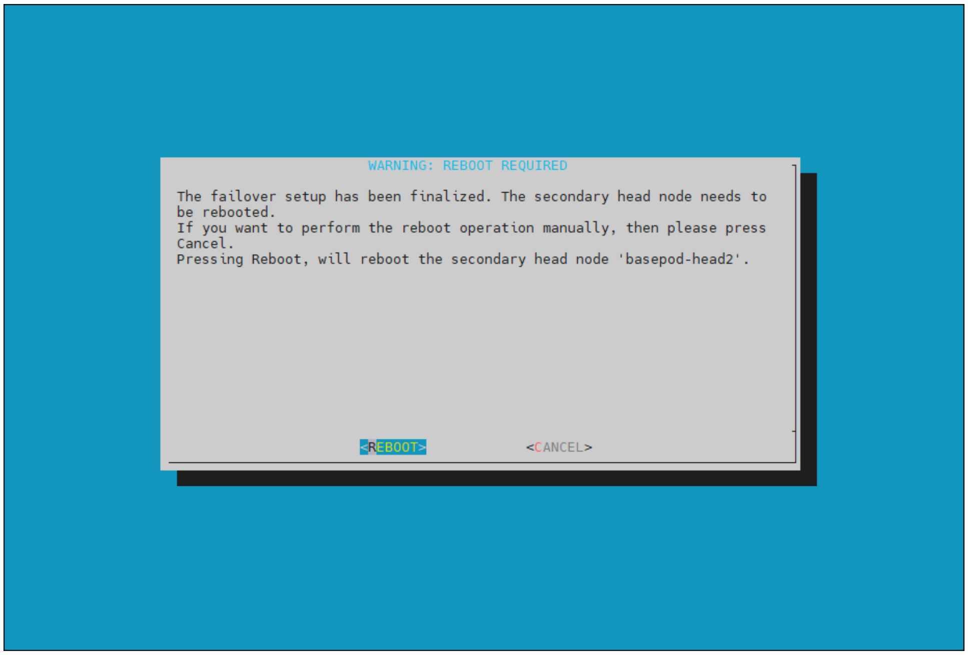
100%

The progress is shown below:

```
Updating secondary master mac address............. [  OK  ]
Initializing failover setup on basepod-head2....... [  OK  ]
Stopping cmdaemon................................. [  OK  ]
Cloning cmdaemon database......................... [  OK  ]
Checking database consistency..................... [  OK  ]
Starting cmdaemon, chkconfig services............. [  OK  ]
Cloning workload manager databases................ [  OK  ]
Cloning additional databases...................... [  OK  ]
Update DB permissions............................. [  OK  ]
Checking for dedicated failover network........... [  OK  ]
Press any key to continue
```

28. Select `REBOOT` when the `WARNING: REBOOT REQUIRED` screen is shown.

    Wait for the secondary head node to reboot before continuing.



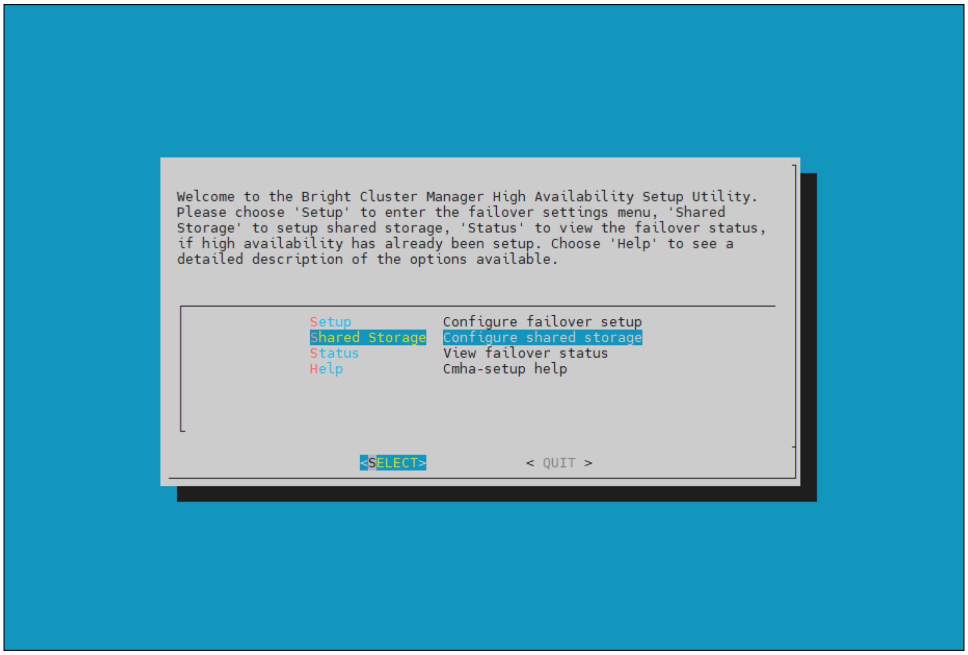
```
WARNING: REBOOT REQUIRED

The failover setup has been finalized. The secondary head node needs to
be rebooted.
If you want to perform the reboot operation manually, then please press
Cancel.
Pressing Reboot, will reboot the secondary head node 'basepod-head2'.
```

29. The secondary head node is now UP.

```
% device list -f hostname:20,category:12,ip:20,status:15
hostname (key)       category     ip                   status
-------------------- ---------- -------------------- ---------------
basepod-head1                     10.227.48.254        [    UP    ]
basepod-head2                     10.227.48.253        [    UP    ]
knode01              k8s-master   10.227.48.9          [   DOWN   ]
knode02              k8s-master   10.227.48.10         [   DOWN   ]
knode03              k8s-master   10.227.48.11         [   DOWN   ]
dgx01                dgx          10.227.48.5          [   DOWN   ]
dgx02                dgx          10.227.48.6          [   DOWN   ]
dgx03                dgx          10.227.48.7          [   DOWN   ]
dgx04                dgx          10.227.48.8          [   DOWN   ]
```

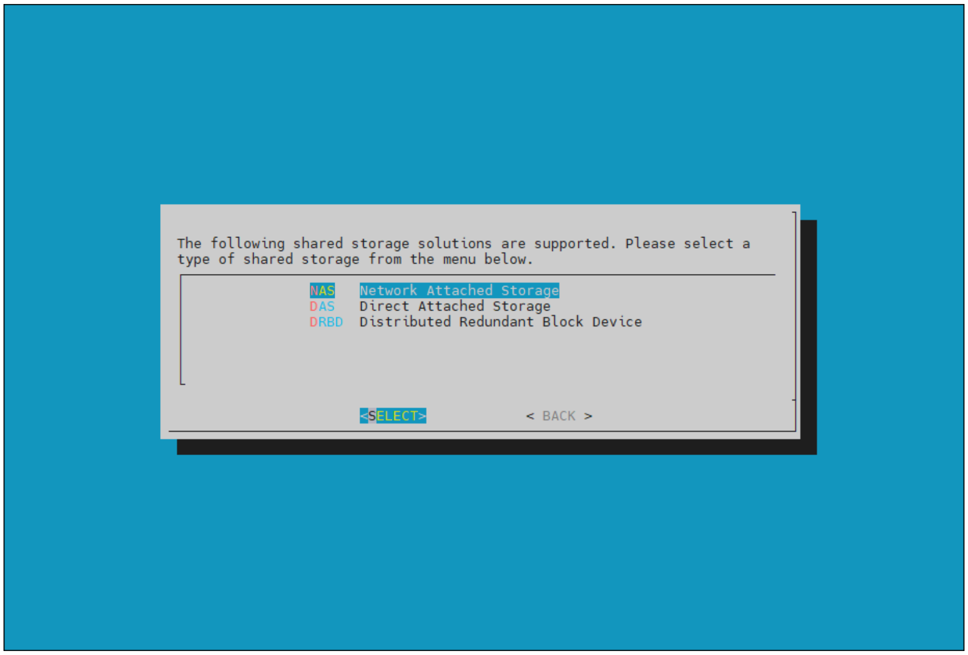
30. Choose `Shared Storage` from the `cmha-setup` menu and select `SELECT`.

    In this final HA configuration step, `cmha-setup` will copy the `/cm/shared` and `/home` directories to the shared storage, and it configures both head nodes and all cluster nodes to mount it.



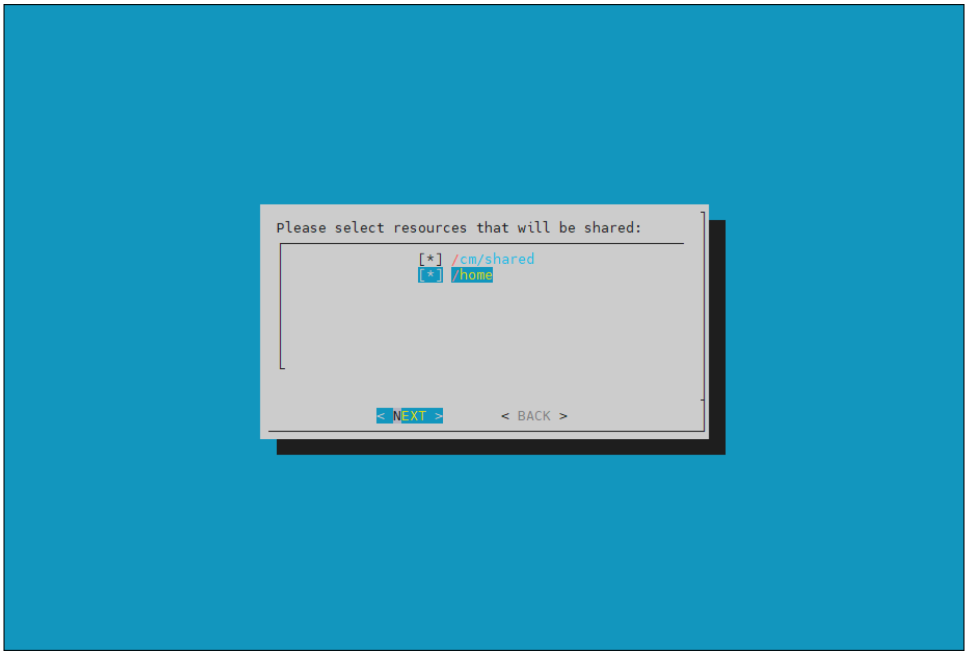
```
Welcome to the Bright Cluster Manager High Availability Setup Utility.
Please choose 'Setup' to enter the failover settings menu, 'Shared
Storage' to setup shared storage, 'Status' to view the failover status,
if high availability has already been setup. Choose 'Help' to see a
detailed description of the options available.

                Setup           Configure failover setup
                Shared Storage  Configure shared storage
                Status          View failover status
                Help            Cmha-setup help

                    <SELECT>            < QUIT >
```

31. Choose `NAS` and then select `SELECT`.



```
The following shared storage solutions are supported. Please select a
type of shared storage from the menu below.

                NAS   Network Attached Storage
                DAS   Direct Attached Storage
                DRBD  Distributed Redundant Block Device

                    <SELECT>            < BACK >
```

32. Choose `/cm/shared` and `/home` and then select `NEXT`.

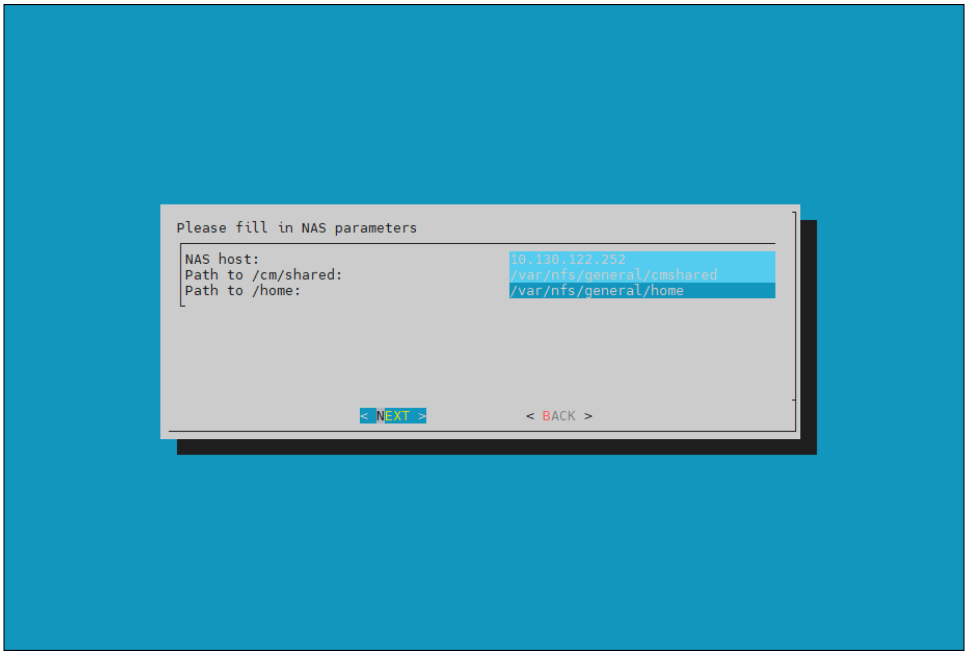


Please select resources that will be shared:

```
[*] /cm/shared
[*] /home
```

33. Provide the IP address of the NAS host, the paths for the `/cm/shared` and `/home` directories should be copied to on the shared storage and then select `NEXT`

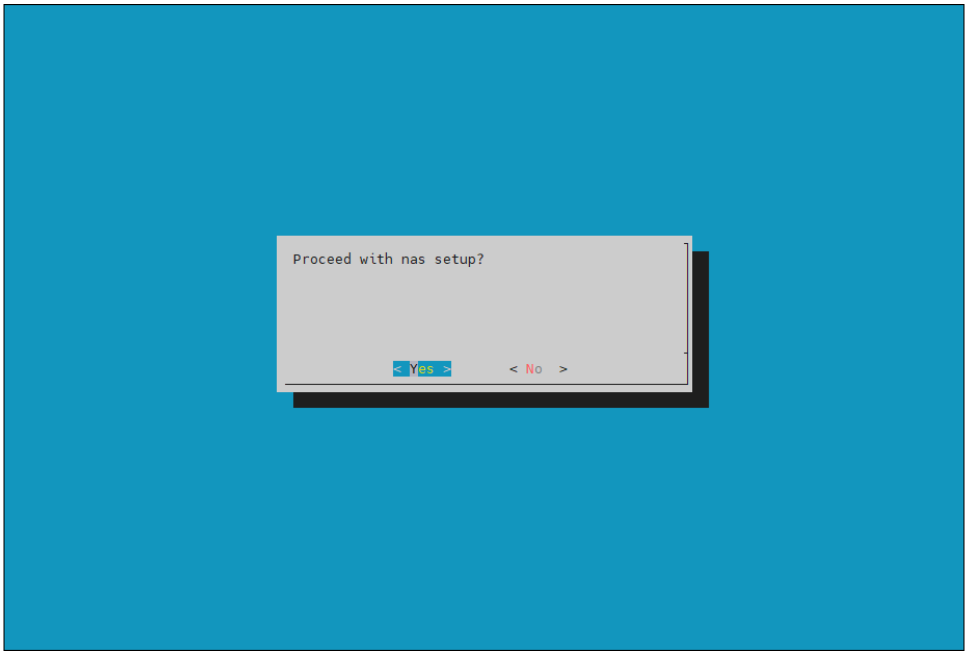
    The values are from In this case, /`var/nfs/general` is exported, so the `/cm/shared` directory will be copied to `10.227.48.252:/var/nfs/general/cmshared`, and it will be mounted over `/cm/shared` on the cluster nodes.



Please fill in NAS parameters

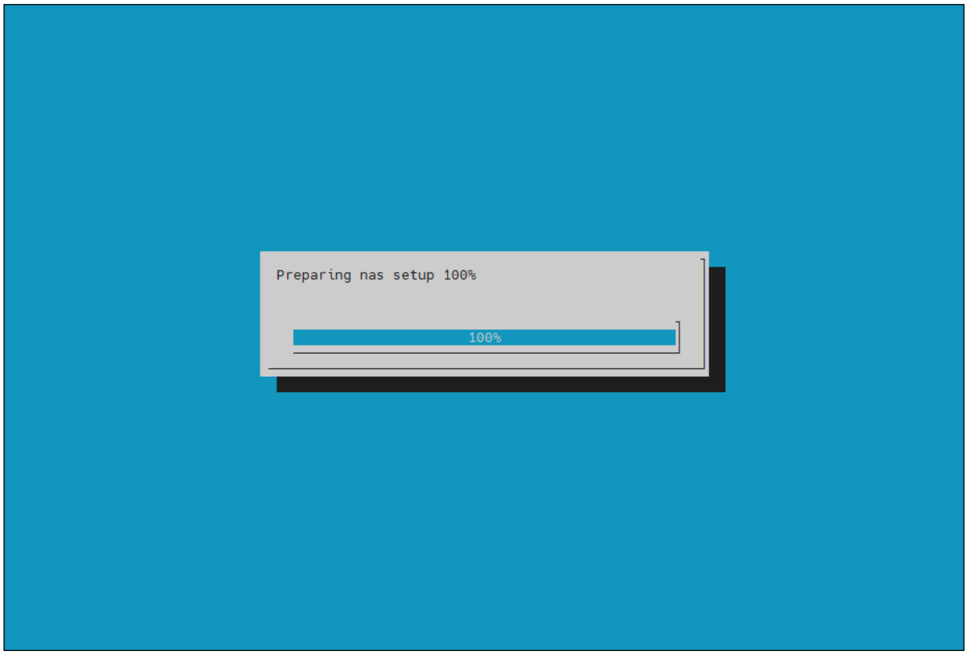
```
NAS host:                              10.130.122.252
Path to /cm/shared:                    /var/nfs/general/cmshared
Path to /home:                         /var/nfs/general/home
```

34. The wizard shows a summary of the information that it has collected. Press `ENTER` to continue.
35. Select YES when prompted to proceed with the setup.



```
Proceed with nas setup?
```

36. The `cmha-setup` wizard proceeds with its work. When it completes, select `ENTER` to finish the HA setup.

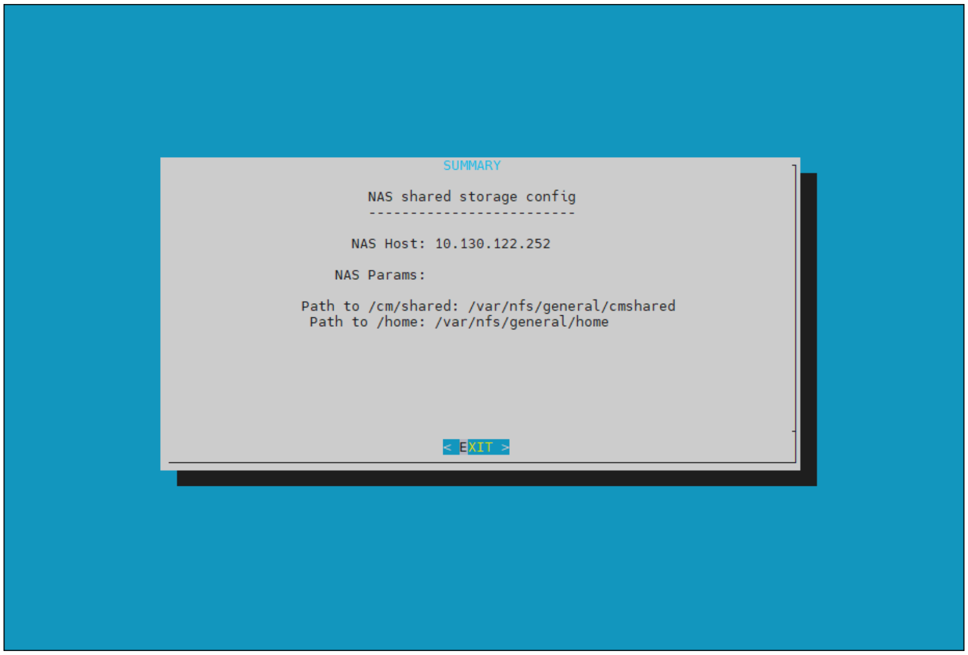


```
Preparing nas setup 100%
```

The progress is shown below:

```
Copying NAS data.................................. [  OK  ]
Mount NAS storage................................. [  OK  ]
Remove old fsmounts............................... [  OK  ]
Add new fsmounts.................................. [  OK  ]
Remove old fsexports.............................. [  OK  ]
Write NAS mount/unmount scripts................... [  OK  ]
Copy mount/unmount scripts........................ [  OK  ]
Press any key to continue
```

37. `cmha-setup` is now complete. Select `EXIT` to return to the shell prompt.



```
SUMMARY

NAS shared storage config
-------------------------

NAS Host: 10.130.122.252

NAS Params:

Path to /cm/shared: /var/nfs/general/cmshared
Path to /home: /var/nfs/general/home
```

3.1.1 Verify the HA Setup

1. Run the `cmha status` command to verify that the failover configuration is correct and working as expected.

   Note that the command tests the configuration from both directions: from the primary head node to the secondary, and from the secondary to the primary. The active head node is indicated by an asterisk.

```
# cmha status
Node Status: running in active mode

basepod-head1* -> basepod-head2
  mysql         [  OK  ]
  ping          [  OK  ]
  status        [  OK  ]

basepod-head2 -> basepod-head1*
  mysql         [  OK  ]
  ping          [  OK  ]
  status        [  OK  ]
```

2. Verify that the `/cm/shared` and `/home` directories are being mounted from the NAS server.

```
# mount
. . . some output omitted . . .
10.227.48.252:/var/nfs/general/cmshared on /cm/shared type nfs4
(rw,relatime,vers=4.2,rsize=32768,wsize=32768,namlen=255,hard,proto=tcp,timeo=600,retrans=2
,sec=sys,clientaddr=10.130.12210.227.48_lock=none,addr=10.130.122.252)10.227.48
10.227.48.252:/var/nfs/general/home on /home type nfs4
(rw,relatime,vers=4.2,rsize=32768,wsize=32768,namlen=255,hard,proto=tcp,timeo=600,retrans=2
,sec=sys,clientaddr=10.130.12210.227.48_lock=none,addr=10.130.122.252)10.227.48
```

3. Login to the head node to be made active and run `cmha makeactive`.

```
# ssh basepod-head2
# cmha makeactive
=========================================================================
This is the passive head node. Please confirm that this node should become
the active head node. After this operation is complete, the HA status of
the head nodes will be as follows:

basepod-head2 will become active head node (current state: passive)
basepod-head1 will become passive head node (current state: active)
=========================================================================

Continue(c)/Exit(e)? c

Initiating failover.............................. [  OK  ]

basepod-head2 is now active head node, makeactive successful
```

4. Run the `cmha status` command again to verify that the secondary head node has become the active head node.

```
# cmha status
Node Status: running in active mode

basepod-head2* -> basepod-head1
  mysql         [  OK  ]
  ping          [  OK  ]
  status        [  OK  ]

basepod-head1 -> basepod-head2*
  mysql         [  OK  ]
  ping          [  OK  ]
  status        [  OK  ]
```

5. Manually failover back to the primary head node.

```
# ssh basepod-head1
# cmha makeactive

===========================================================================
This is the passive head node. Please confirm that this node should become
the active head node. After this operation is complete, the HA status of
the head nodes will be as follows:

basepod-head1 will become active head node (current state: passive)
basepod-head2 will become passive head node (current state: active)
===========================================================================

Continue(c)/Exit(e)? c

Initiating failover.............................. [  OK  ]

basepod-head1 is now active head node, makeactive successful
```

6. Run `cmsh status` again to verify that the primary head node has become the active head node.

```
# cmha status
Node Status: running in active mode

basepod-head1* -> basepod-head2
  mysql         [  OK  ]
  ping          [  OK  ]
  status        [  OK  ]

basepod-head2 -> basepod-head1*
  mysql         [  OK  ]
  ping          [  OK  ]
  status        [  OK  ]
```

7. Power on the cluster nodes.

```
# cmsh -c “power -c k8s-master,dgx on”
ipmi0 .................... [   ON    ] knode01
ipmi0 .................... [   ON    ] knode02
ipmi0 .................... [   ON    ] knode03
ipmi0 .................... [   ON    ] dgx01
ipmi0 .................... [   ON    ] dgx02
ipmi0 .................... [   ON    ] dgx03
ipmi0 .................... [   ON    ] dgx04
```

8. (Optionally) Configure Jupyter HA

   If Jupyter was deployed on the primary head node before HA was configured, configure the Jupyter service to run on the active head node.

```
% device
% use basepod-head1
% services
% use cm-jupyterhub
% show
Parameter                        Value
-------------------------------- --------------------------------------------
Revision
Service                          cm-jupyterhub
Run if                           ALWAYS
Monitored                        yes
Autostart                        yes
Timeout                          -1
Belongs to role                  yes
Sickness check script
Sickness check script timeout    10
Sickness check interval          60
```

9. Set the `runif` parameter to `active`.

```
% set runif active
% commit

% show
Parameter                        Value
-------------------------------- --------------------------------------------
Revision
Service                          cm-jupyterhub
Run if                           ACTIVE
Monitored                        yes
Autostart                        yes
Timeout                          -1
Belongs to role                  yes
Sickness check script
Sickness check script timeout    10
Sickness check interval          60
```

10. Configure the Jupyter service on the secondary head node.

```
% device
% use basepod-head2
% services
% use cm-jupyterhub
% set runif active
```

Chapter 4. Basic User Management

BCM uses its own LDAP service to manage users and groups with a centralized LDAP database server running on the head node, and not by entries in `/etc/passwd` or `/etc/group` files. An external LDAP server can be setup for authentication services to replace the existing BCM LDAP service, but it is outside of the scope of this document.

Only the basic user management tasks are outlined in this guide to provide a starting point. Refer to the *Base Command Manager Administrator Manual* for complete options and additional details.

Although user management can be done in both `cmsh` and Base View, `cmsh` is used in this chapter.

4.1 Configuring a User

1. Add a user (`userone` in this case).

```
# cmsh
% user
% add userone
% set password 7adGnv0!K
% commit
```

2. `userone` will reset the password after successfully logging in.

```
userone@basepod-head2:~$ passwd
(current) LDAP Password:
New password:
Retype new password:
passwd: password updated successfully
userone@basepod-head2:~$
```

3. Use `show` to view user parameters and values.

```
[basepod-head2->user[userone]]% show
Parameter                        Value
-------------------------------- --------------------------------------------
Accounts
Managees
Name                             userone
Primary group                    userone
Revision
Secondary groups
ID                               1004
Common name                      userone
Surname                          userone
Group ID                         1004
Login shell                      /bin/bash
Home directory                   /home/ userone
Password                         *********
email
Profile
Create cmjob certificate         no
Write ssh proxy config           no
Shadow min                       0
Shadow max                       999999
Shadow warning                   7
Inactive                         0
Last change                      2022/10/20
Expiration date                  2037/12/31
Project manager                  <submode>
Notes                            <0B>
```

4. Use `set` to change parameters.

```
[basepod-head2->user[userone]]% set
commonname              expirationdate          id                      name
profile                 shadowmax               surname
createcmjobcertificate  groupid                 inactive                notes
projectmanager          shadowmin               writesshproxyconfig
email                   homedirectory           loginshell              password
revision                shadowwarning
```

4.1.1 Procedures to Remove a User

This block of code will delete a user.

```
# cmsh
% user
% remove userone
% commit
```

Adding the `-d` option to `remove` will also delete the home directory.

4.2 Adding a User to K8s

To use K8s services, a user must also be added to the K8s cluster.

Add each K8s user with `cm-kubernetes-setup`.

```
root@basepod-head1:~# cm-kubernetes-setup --add-user userone
Connecting to CMDaemon
Executing 10 stages
################### Starting execution for 'Kubernetes Setup'
  - kubernetes
  - docker
## Progress: 0
#### stage: kubernetes: Get Kube Cluster
## Progress: 10
#### stage: kubernetes: Check Permissions User Chart
## Progress: 20
#### stage: kubernetes: Check User
## Progress: 30
#### stage: kubernetes: Check Add User
## Progress: 40
#### stage: kubernetes: Check Namespace Does Not Exist
## Progress: 50
#### stage: kubernetes: Check Cluster Admin Has No Operators
## Progress: 60
#### stage: kubernetes: Deploy user
User userone created successfully!
## Progress: 70
#### stage: kubernetes: List Installed Operators
## Progress: 80
#### stage: kubernetes: Update Operator Permissions
## Progress: 90
#### stage: kubernetes: Log Text
User added successfully!
## Progress: 100

Took:     00:06 min.
Progress: 100/100
################### Finished execution for 'Kubernetes Setup', status: completed

Kubernetes Setup finished!
```

4.3 Removing a User from K8s

To remove a user (`userone`) from K8s, execute this command:

```
# cm-kubernetes-setup --remove-user userone
```

The user will no longer be able to use the K8s service.

If an attempt is made, this error message will be shown:

```
Error from server (Forbidden): nodes is forbidden: User "userone" cannot list resource
"nodes" in API group "" at the cluster scope
```

Appendix A. Site Survey

The tables in this section represent responses to a completed site survey and are used as examples in this deployment guide.

Table 3. General information

Item	Value
NFS server IP	10.227.48.252
NFS server export	/var/nfs/general
Head node drive path	nvme0n1
Cluster name	BasePOD
Organization	NVIDIA
Timezone	US/Los_Angeles
Nameservers	8.8.8.8
Search domains	example.com

Table 4. BCM head node information

Item	Value
Head node 1 name	`basepod-head1`
Head node 1 and 2 administrator password	`ExamplePassword1234!@#$`
Head node 1 BMC IP (`ipminet`)	`10.227.20.66`
Head node 1 Ethernet device (`externalnet`)	`enp10`
Head node 1 IP (`externalnet`)	`10.227.52.254`
Head node 1 Ethernet device (`internalnet`)	`enp10`
Head node 1 IP (`internalnet`)	`10.227.48.254`
Head node 2 name	`basepod-head2`
Head node 2 BMC IP (`ipminet`)	`10.227.20.67`
Head node 2 Ethernet device (`externalnet`)	`enp10`
Head node 2 IP (`externalnet`)	`10.227.52.253`
Head node 2 Ethernet device (`internalnet`)	`enp10`
Head node 2 IP (`internalnet`)	`10.227.48.253`

Table 5. Network information

Item	Value
K8s node name template	knode##
ipminet base IP	10.227.20.64
ipminet netmask	255.255.255.192
ipminet gateway	10.227.20.65
ipminet switch ASN	
internalnet switch #1 ASN (for externalnet)	
internalnet switch #2 ASN (for externalnet)	
externalnet base IP	10.227.52.0
externalnet netmask	255.255.255.0
externalnet gateway	10.227.52.1
Domain	example.com
internalnet base IP	10.227.48.0
internalnet netmask	255.255.255.0
ibnet base IP	10.149.0.0
ibnet netmask	255.255.0.0

Table 6. DGX node information

Item	Value
DGX Node 1 name	dgx01
DGX Node 1 MAC (enp225s0f0/Management)	B8:CE:F6:2F:08:69
DGX Node 1 MAC (enp97s0f0/Management)	B8:CE:F6:2D:0E:A7
DGX Node 2 name	dgx02
DGX Node 2 MAC (enp225s0f0/Management)	B8:CE:F6:2F:08:69
DGX Node 2 MAC (enp97s0f0/Management)	B8:CE:F6:2D:0E:A7
DGX Node 3 name	dgx03
DGX Node 3 MAC (enp225s0f0/Management)	B8:CE:F6:2F:08:69
DGX Node 3 MAC (enp97s0f0/Management)	B8:CE:F6:2D:0E:A7
DGX Node 4 name	dgx04
DGX Node 4 MAC (enp225s0f0/Management)	B8:CE:F6:2F:08:69
DGX Node 4 MAC (enp97s0f0/Management)	B8:CE:F6:2D:0E:A7

Table 7. K8s node information

Item	Value
K8S Node 1 interface 1 (Management)	`ens1f1np1`
K8S Node 1 interface 2 (Management)	`ens2f1np1`
K8S Node 1 interface 1 (Management) MAC	`04:3F:72:E7:64:97`
K8S Node 1 interface 2 (Management) MAC	`0C:42:A1:79:9B:15`
K8S Node 2 interface 1 (Management)	`ens1f1np1`
K8S Node 2 interface 2 (Management)	`ens2f1np1`
K8S Node 2 interface 1 (Management) MAC	`04:3F:72:E7:64:97`
K8S Node 2 interface 2 (Management) MAC	`0C:42:A1:79:9B:15`
K8S Node 3 interface 1 (Management)	`ens1f1np1`
K8S Node 3 interface 2 (Management)	`ens2f1np1`
K8S Node 3 interface 1 (Management) MAC	`04:3F:72:E7:64:97`
K8S Node 3 interface 2 (Management) MAC	`0C:42:A1:79:9B:15`

Appendix B. Switch Configurations

Switch configuration files are captured in this section.

B.1 SN4600 #1 (In-band Management Switch)

```
# Note Make sure to update the IP addresses in the sample config below
# eth0 mgmt interface configs
nv set interface eth0 ip address 10.227.20.78/26
nv set interface eth0 ip gateway 10.227.20.65
nv set interface eth0 ip vrf mgmt
nv set interface eth0 type eth

# Creating SVI interfaces and adding the VLANs to the bridge
# Note Make sure to update the IP addresses
nv set bridge domain br_default vlan 122
nv set bridge domain br_default vlan 121
nv set interface vlan121 type svi
nv set interface vlan121 ip vrr mac-address 44:38:39:ff:ff:ff
nv set interface vlan121 ip vrr address 10.227.52.1/26
nv set interface vlan121 ip address 10.227.52.2/26
nv set interface vlan122 type svi
nv set interface vlan122 ip vrr mac-address 44:38:39:ff:ff:ff
nv set interface vlan122 ip vrr address 10.227.48.1/26
nv set interface vlan122 ip address 10.227.48.2/26

# MLAG configs
nv set interface peerlink bond member swp57
nv set interface peerlink bond member swp58
nv set interface peerlink type peerlink
nv set interface peerlink.4094 base-interface peerlink
nv set interface peerlink.4094 type sub
nv set interface peerlink.4094 vlan 4094
nv set mlag backup 10.227.20.79 vrf mgmt
nv set mlag enable on
nv set mlag mac-address 44:38:39:ff:ff:ff
nv set mlag peer-ip linklocal
nv set mlag priority 2048

### bcm headnode 01
nv set interface swp1 bridge domain br_default access 121
### bcm headnode 02
nv set interface swp2 bridge domain br_default access 121

### bond4-20 used to connect to DGX and Kubernetes master nodes
nv set interface bond4 bond member swp4
nv set interface bond4 bond mlag id 4
nv set interface bond4 bridge domain br_default untagged 122
nv set interface bond4 bridge domain br_default vlan all
nv set interface bond4 bond mlag enable on
nv set interface bond4 bond lacp-bypass on
nv set interface bond5 bond member swp5
nv set interface bond5 bond mlag id 5
nv set interface bond5 bridge domain br_default untagged 122
nv set interface bond5 bridge domain br_default vlan all
nv set interface bond5 bond mlag enable on
nv set interface bond5 bond lacp-bypass on
nv set interface bond6 bond member swp6
nv set interface bond6 bond mlag id 6
nv set interface bond6 bridge domain br_default untagged 122
nv set interface bond6 bridge domain br_default vlan all
nv set interface bond6 bond mlag enable on
nv set interface bond6 bond lacp-bypass on
nv set interface bond7 bond member swp7
nv set interface bond7 bond mlag id 7
nv set interface bond7 bridge domain br_default untagged 122
nv set interface bond7 bridge domain br_default vlan all
nv set interface bond7 bond mlag enable on
nv set interface bond7 bond lacp-bypass on
nv set interface bond8 bond member swp8
nv set interface bond8 bond mlag id 8
nv set interface bond8 bridge domain br_default untagged 122
nv set interface bond8 bridge domain br_default vlan all
nv set interface bond8 bond mlag enable on
nv set interface bond8 bond lacp-bypass on
nv set interface bond9 bond member swp9
nv set interface bond9 bond mlag id 9
nv set interface bond9 bridge domain br_default untagged 122
nv set interface bond9 bridge domain br_default vlan all
nv set interface bond9 bond mlag enable on
nv set interface bond9 bond lacp-bypass on
nv set interface bond10 bond member swp10
nv set interface bond10 bond mlag id 10
nv set interface bond10 bridge domain br_default untagged 122
nv set interface bond10 bridge domain br_default vlan all
nv set interface bond10 bond mlag enable on
nv set interface bond10 bond lacp-bypass on
nv set interface bond11 bond member swp11
nv set interface bond11 bond mlag id 11
nv set interface bond11 bridge domain br_default untagged 122
nv set interface bond11 bridge domain br_default vlan all
nv set interface bond11 bond mlag enable on
nv set interface bond11 bond lacp-bypass on
nv set interface bond12 bond member swp12
nv set interface bond12 bond mlag id 12
nv set interface bond12 bridge domain br_default untagged 122
nv set interface bond12 bridge domain br_default vlan all
nv set interface bond12 bond mlag enable on
nv set interface bond12 bond lacp-bypass on
nv set interface bond13 bond member swp13
nv set interface bond13 bond mlag id 13
nv set interface bond13 bridge domain br_default untagged 122
nv set interface bond13 bridge domain br_default vlan all
nv set interface bond13 bond mlag enable on
nv set interface bond13 bond lacp-bypass on
nv set interface bond14 bond member swp14
nv set interface bond14 bond mlag id 14
nv set interface bond14 bridge domain br_default untagged 122
nv set interface bond14 bridge domain br_default vlan all
nv set interface bond14 bond mlag enable on
nv set interface bond14 bond lacp-bypass on
nv set interface bond15 bond member swp15
nv set interface bond15 bond mlag id 15
nv set interface bond15 bridge domain br_default untagged 122
nv set interface bond15 bridge domain br_default vlan all
nv set interface bond15 bond mlag enable on
nv set interface bond15 bond lacp-bypass on
nv set interface bond16 bond member swp16
nv set interface bond16 bond mlag id 16
nv set interface bond16 bridge domain br_default untagged 122
nv set interface bond16 bridge domain br_default vlan all
nv set interface bond16 bond mlag enable on
nv set interface bond16 bond lacp-bypass on
nv set interface bond17 bond member swp17
nv set interface bond17 bond mlag id 17
nv set interface bond17 bridge domain br_default untagged 122
nv set interface bond17 bridge domain br_default vlan all
nv set interface bond17 bond mlag enable on
nv set interface bond17 bond lacp-bypass on
nv set interface bond18 bond member swp18
nv set interface bond18 bond mlag id 18
nv set interface bond18 bridge domain br_default untagged 122
nv set interface bond18 bridge domain br_default vlan all
nv set interface bond18 bond mlag enable on
nv set interface bond18 bond lacp-bypass on
nv set interface bond19 bond member swp19
nv set interface bond19 bond mlag id 19
nv set interface bond19 bridge domain br_default untagged 122
nv set interface bond19 bridge domain br_default vlan all
nv set interface bond19 bond mlag enable on
nv set interface bond19 bond lacp-bypass on
nv set interface bond20 bond member swp20
nv set interface bond20 bond mlag id 20
nv set interface bond20 bridge domain br_default untagged 122
nv set interface bond20 bridge domain br_default vlan all
nv set interface bond20 bond mlag enable on
nv set interface bond20 bond lacp-bypass on

### BGP unnumbered configuration (NOTE: no IPs need to be configured on the BGP
interfaces, when using BGP unnumbered)
nv set router bgp autonomous-system 4200000003
nv set router bgp enable on
nv set router bgp router-id 10.227.20.78
nv set vrf default router bgp address-family ipv4-unicast enable on
nv set vrf default router bgp address-family ipv4-unicast redistribute connected enable
on
nv set vrf default router bgp enable on
nv set vrf default router bgp neighbor swp64 remote-as external
nv set vrf default router bgp neighbor swp64 type unnumbered
nv set vrf default router bgp neighbor swp63 remote-as external
nv set vrf default router bgp neighbor swp63 type unnumbered
nv set vrf default router bgp neighbor swp60 remote-as external
nv set vrf default router bgp neighbor swp60 type unnumbered
nv set vrf default router bgp neighbor peerlink.4094 remote-as internal
nv set vrf default router bgp neighbor peerlink.4094 type unnumbered

### apply and save configuration
nv config diff
nv config apply
nv config save
```

B.2 SN4600 #2 (In-band Management Switch)

```
# Note Make sure to update the IP addresses in the sample config below
# eth0 mgmt interface configs
nv set interface eth0 ip address 10.227.20.79/26
nv set interface eth0 ip gateway 10.227.20.65
nv set interface eth0 ip vrf mgmt
nv set interface eth0 type eth

# Creating SVI interfaces and adding the VLANs to the bridge
# Note Make sure to update the IP addresses
nv set bridge domain br_default vlan 122
nv set bridge domain br_default vlan 121
nv set interface vlan121 type svi
nv set interface vlan121 ip vrr mac-address 44:38:39:ff:ff:ff
nv set interface vlan121 ip vrr address 10.227.52.1/26
nv set interface vlan121 ip address 10.227.52.3/26
nv set interface vlan122 type svi
nv set interface vlan122 ip vrr mac-address 44:38:39:ff:ff:ff
nv set interface vlan122 ip vrr address 10.227.48.1/26
nv set interface vlan122 ip address 10.227.48.3/26

# MLAG configs
nv set interface peerlink bond member swp57
nv set interface peerlink bond member swp58
nv set interface peerlink type peerlink
nv set interface peerlink.4094 base-interface peerlink
nv set interface peerlink.4094 type sub
nv set interface peerlink.4094 vlan 4094
nv set mlag backup 10.227.20.78 vrf mgmt
nv set mlag enable on
nv set mlag mac-address 44:38:39:ff:ff:ff
nv set mlag peer-ip linklocal

### bcm headnode 01
nv set interface swp1 bridge domain br_default access 122
### bcm headnode 02
nv set interface swp2 bridge domain br_default access 122

### bond4-20 used to connect to DGX and Kubernetes master nodes
nv set interface bond4 bond member swp4
nv set interface bond4 bond mlag id 4
nv set interface bond4 bridge domain br_default untagged 122
nv set interface bond4 bridge domain br_default vlan all
nv set interface bond4 bond mlag enable on
nv set interface bond4 bond lacp-bypass on
nv set interface bond5 bond member swp5
nv set interface bond5 bond mlag id 5
nv set interface bond5 bridge domain br_default untagged 122
nv set interface bond5 bridge domain br_default vlan all
nv set interface bond5 bond mlag enable on
nv set interface bond5 bond lacp-bypass on
nv set interface bond6 bond member swp6
nv set interface bond6 bond mlag id 6
nv set interface bond6 bridge domain br_default untagged 122
nv set interface bond6 bridge domain br_default vlan all
nv set interface bond6 bond mlag enable on
nv set interface bond6 bond lacp-bypass on
nv set interface bond7 bond member swp7
nv set interface bond7 bond mlag id 7
nv set interface bond7 bridge domain br_default untagged 122
nv set interface bond7 bridge domain br_default vlan all
nv set interface bond7 bond mlag enable on
nv set interface bond7 bond lacp-bypass on
nv set interface bond8 bond member swp8
nv set interface bond8 bond mlag id 8
nv set interface bond8 bridge domain br_default untagged 122
nv set interface bond8 bridge domain br_default vlan all
nv set interface bond8 bond mlag enable on
nv set interface bond8 bond lacp-bypass on
nv set interface bond9 bond member swp9
nv set interface bond9 bond mlag id 9
nv set interface bond9 bridge domain br_default untagged 122
nv set interface bond9 bridge domain br_default vlan all
nv set interface bond9 bond mlag enable on
nv set interface bond9 bond lacp-bypass on
nv set interface bond10 bond member swp10
nv set interface bond10 bond mlag id 10
nv set interface bond10 bridge domain br_default untagged 122
nv set interface bond10 bridge domain br_default vlan all
nv set interface bond10 bond mlag enable on
nv set interface bond10 bond lacp-bypass on
nv set interface bond11 bond member swp11
nv set interface bond11 bond mlag id 11
nv set interface bond11 bridge domain br_default untagged 122
nv set interface bond11 bridge domain br_default vlan all
nv set interface bond11 bond mlag enable on
nv set interface bond11 bond lacp-bypass on
nv set interface bond12 bond member swp12
nv set interface bond12 bond mlag id 12
nv set interface bond12 bridge domain br_default untagged 122
nv set interface bond12 bridge domain br_default vlan all
nv set interface bond12 bond mlag enable on
nv set interface bond12 bond lacp-bypass on
nv set interface bond13 bond member swp13
nv set interface bond13 bond mlag id 13
nv set interface bond13 bridge domain br_default untagged 122
nv set interface bond13 bridge domain br_default vlan all
nv set interface bond13 bond mlag enable on
nv set interface bond13 bond lacp-bypass on
nv set interface bond14 bond member swp14
nv set interface bond14 bond mlag id 14
nv set interface bond14 bridge domain br_default untagged 122
nv set interface bond14 bridge domain br_default vlan all
nv set interface bond14 bond mlag enable on
nv set interface bond14 bond lacp-bypass on
nv set interface bond15 bond member swp15
nv set interface bond15 bond mlag id 15
nv set interface bond15 bridge domain br_default untagged 122
nv set interface bond15 bridge domain br_default vlan all
nv set interface bond15 bond mlag enable on
nv set interface bond15 bond lacp-bypass on
nv set interface bond16 bond member swp16
nv set interface bond16 bond mlag id 16
nv set interface bond16 bridge domain br_default untagged 122
nv set interface bond16 bridge domain br_default vlan all
nv set interface bond16 bond mlag enable on
nv set interface bond16 bond lacp-bypass on
nv set interface bond17 bond member swp17
nv set interface bond17 bond mlag id 17
nv set interface bond17 bridge domain br_default untagged 122
nv set interface bond17 bridge domain br_default vlan all
nv set interface bond17 bond mlag enable on
nv set interface bond17 bond lacp-bypass on
nv set interface bond18 bond member swp18
nv set interface bond18 bond mlag id 18
nv set interface bond18 bridge domain br_default untagged 122
nv set interface bond18 bridge domain br_default vlan all
nv set interface bond18 bond mlag enable on
nv set interface bond18 bond lacp-bypass on
nv set interface bond19 bond member swp19
nv set interface bond19 bond mlag id 19
nv set interface bond19 bridge domain br_default untagged 122
nv set interface bond19 bridge domain br_default vlan all
nv set interface bond19 bond mlag enable on
nv set interface bond19 bond lacp-bypass on
nv set interface bond20 bond member swp20
nv set interface bond20 bond mlag id 20
nv set interface bond20 bridge domain br_default untagged 122
nv set interface bond20 bridge domain br_default vlan all
nv set interface bond20 bond mlag enable on
nv set interface bond20 bond lacp-bypass on

### BGP unnumbered configuration (NOTE: no IPs need to be configured on the BGP
interfaces, when using BGP unnumbered)
nv set router bgp autonomous-system 4200000003
nv set router bgp enable on
nv set router bgp router-id 10.227.20.79
nv set vrf default router bgp address-family ipv4-unicast enable on
nv set vrf default router bgp address-family ipv4-unicast redistribute connected enable
on
nv set vrf default router bgp enable on
nv set vrf default router bgp neighbor swp63 remote-as external
nv set vrf default router bgp neighbor swp63 type unnumbered
nv set vrf default router bgp neighbor swp64 remote-as external
nv set vrf default router bgp neighbor swp64 type unnumbered
nv set vrf default router bgp neighbor swp60 remote-as external
nv set vrf default router bgp neighbor swp60 type unnumbered
nv set vrf default router bgp neighbor peerlink.4094 remote-as internal
nv set vrf default router bgp neighbor peerlink.4094 type unnumbered

### apply and save configuration
nv config diff
nv config apply
nv config save
```

B.3 SN2201 (Out-of-band Management Switch)

```
#eth0
nv set interface eth0 ip address 10.227.20.77/26
nv set interface eth0 ip gateway 10.227.20.65
nv set interface eth0 ip vrf mgmt
nv set interface eth0 type eth

# Creating SVI interfaces and adding the VLANs to the bridge
# Note Make sure to update the IP addresses
nv set interface vlan111 ip address 10.227.20.65/26
nv set bridge domain br_default vlan 111

### BGP configurations
nv set router bgp autonomous-system 4200000004
nv set router bgp enable on
nv set router bgp router-id 10.227.20.77
nv set vrf default router bgp address-family ipv4-unicast enable on
nv set vrf default router bgp address-family ipv4-unicast redistribute connected enable
on
nv set vrf default router bgp enable on
nv set vrf default router bgp neighbor swp51 remote-as external
nv set vrf default router bgp neighbor swp51 type unnumbered
nv set vrf default router bgp neighbor swp52 remote-as external
nv set vrf default router bgp neighbor swp52 type unnumbered

# interfaces connected to IPMI interfaces of different servers
nv set interface swp1-40 bridge domain br_default access 111

### apply and save configuration
nv config diff
nv config apply
nv config save
```

B.4 Ethernet Network Configuration Verifications

Some of the lines in the output have been truncated for readability.

```
### MLAG verifications:
root@TOR-01:mgmt:~# net show clag
The peer is alive
     Our Priority, ID, and Role: 2048 48:b0:2d:cc:b2:bc primary
    Peer Priority, ID, and Role: 32768 48:b0:2d:ca:93:7b secondary
          Peer Interface and IP: peerlink.4094 fe80::4ab0:2dff:feca:937b (linklocal)
                      Backup IP: 10.227.20.79 vrf mgmt (active)
                     System MAC: 44:38:39:ff:ff:ff
root@TOR-01:mgmt:~#
CLAG Interfaces
Our Interface   Peer Interface  CLAG Id   Conflicts        Proto-Down Reason
```

```
-------------   -------------   -------   --------------   -----------------
       bond10   -                 10        -                      -
       bond11   -                 11        -                      -
       bond12   -                 12        -                      -
       bond13   bond13            13        -                      -
       bond14   -                 14        -                      -
       bond15   -                 15        -                      -
       bond16   bond16            16        -                      -
       bond17   -                 17        -                      -
       bond18   -                 18        -                      -
       bond19   -                 19        -                      -
       bond20   -                 20
       bond4   bond4              4         -                      -
       bond5   -                  5         -                      -
       bond6   bond6              6         -                      -
       bond7   -                  7         -                      -
       bond8   -                  8         -                      -
       bond9   -                  9         -                      -

### verifying an access port (below is a access port with VLAN set to 111)

network-admin@IPMI-01:mgmt:~$ net show int swp1
    Name  MAC                Speed  MTU   Mode
--  ----  -----------------  -----  ----  ---------
UP  swp1  68:21:5f:4f:14:81  1G     1500  Access/L2

Alias
-----
bcm-bootstrap:eth0

All VLANs on L2 Port
--------------------
111

Untagged
--------
111

### verifying a bonded interface in trunk mode (note the native VLAN is set to 122)
network-admin@TOR-01:mgmt:~$ net show int bond9
    Name    MAC                Speed  MTU   Mode
--  ------  -----------------  -----  ----  -------
UP  bond9  1c:34:da:29:17:54  100G   9216  802.3ad

Bond Details
------------------  --------
Bond Mode:          802.3ad
Load Balancing:     layer3+4
Minimum Links:      1
LACP Sys Priority:
LACP Rate:          1
LACP Bypass:        Active

All VLANs on L2 Port
--------------------
1,121,122
```

```
Untagged
--------
122

#### BGP verifications

cumulus@TOR-01:mgmt:~$ net show bgp summary
show bgp ipv4 unicast summary
=============================
BGP router identifier 10.227.20.78, local AS number 4200000003 vrf-id 0
BGP table version 3
RIB entries 5, using 1000 bytes of memory
Peers 4, using 91 KiB of memory

Neighbor              V         AS   MsgRcvd   MsgSent   TblVer  InQ OutQ  Up/Down
State/PfxRcd    PfxSnt
TOR-02(peerlink.4094)  4 4200000003      2705      2706        0    0    0 02:15:00
Spine-02(swp64)        4 4200000002      2655      2656        0    0    0 02:12:29
Spine-01(swp63)        4 4200000001      2753      2754        0    0    0 02:17:22
IPMI-01(swp60)         4 4200000004      2480      2482        0    0    0 02:03:48

Total number of neighbors 4


cumulus@TOR-02:mgmt:~$ net show bgp summary
show bgp ipv4 unicast summary
=============================
BGP router identifier 10.227.20.79, local AS number 4200000003 vrf-id 0
BGP table version 3
RIB entries 5, using 1000 bytes of memory
Peers 4, using 91 KiB of memory

Neighbor              V         AS   MsgRcvd   MsgSent   TblVer  InQ OutQ  Up/Down
State/PfxRcd   PfxSnt
TOR-01(peerlink.4094) 4 4200000003      2692      2692        0    0    0 02:14:20
Spine-01(swp63)        4 4200000001      2692      2692        0    0    0 02:14:21
Spine-02(swp64)        4 4200000002      2641      2642        0    0    0 02:11:48
IPMI-01(swp60)         4 4200000004      2467      2469        0    0    0 02:03:07

Total number of neighbors 4

cumulus@IPMI-01:mgmt:~$ net show bgp summary
show bgp ipv4 unicast summary
=============================
BGP router identifier 10.227.20.77, local AS number 4200000004 vrf-id 0
BGP table version 3
RIB entries 5, using 1000 bytes of memory
Peers 2, using 46 KiB of memory

Neighbor        V         AS   MsgRcvd   MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd
PfxSnt
TOR-01(swp51)    4 4200000003      2495      2494        0    0    0 02:04:30
TOR-02(swp52)   4 4200000003      2495      2494        0    0    0 02:04:30

Total number of neighbors 2
```

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation ("NVIDIA") makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

Trademarks

Copyright

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
http://www.nvidia.com