NVIDIA DGX BasePOD: Deployment Guide Featuring NVIDIA DGX H200/H100 Systems

Release latest

NVIDIA Corporation

Jun 16, 2025

Overview

Document Number	NA
Publication Date	2025-2-22

Tip

The NVIDIA DGX BasePOD: Deployment Guide Featuring NVIDIA DGX H200 and H100 Systems is also available as a PDF.

Chapter 1. Introduction

Artificial Intelligence (AI) infrastructure requires significant compute resources to operate 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 reference design 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 H200/H100 systems, which offer unprecedented compute performance with eight NVIDIA H200/H100 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.

Chapter 2. Hardware Overview

DGX BasePOD deployment in this example 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 2.1. 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 deployments.

Table 2.1: DGX BasePOD components

Component	Technology
Compute nodes	DGX H200/H100 system
Compute fabric	NVIDIA Quantum QM9700 InfiniBand switches
Management fabric	NVIDIA SN4600C switches
Storage fabric	Option 1: NVIDIA SN4600C switches for Ethernet attached storage Option 2: NVIDIA Quantum QM9700 switches for InfiniBand attached storage
Out-of-band management fabric	NVIDIA SN2201 switches
Control plane and workload management nodes	Minimum Requirements (each server): > 64-bit x86 processor, AMD EPYC 7272 or equivalent > 256 GB memory > 1 TB SSD > Two 100 Gbps network ports

Chapter 3. Networking

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

3.1. DGX H200/H100 System Network Ports

Figure 3.1 shows the physical layout of the back of the DGX H200/H100 system.

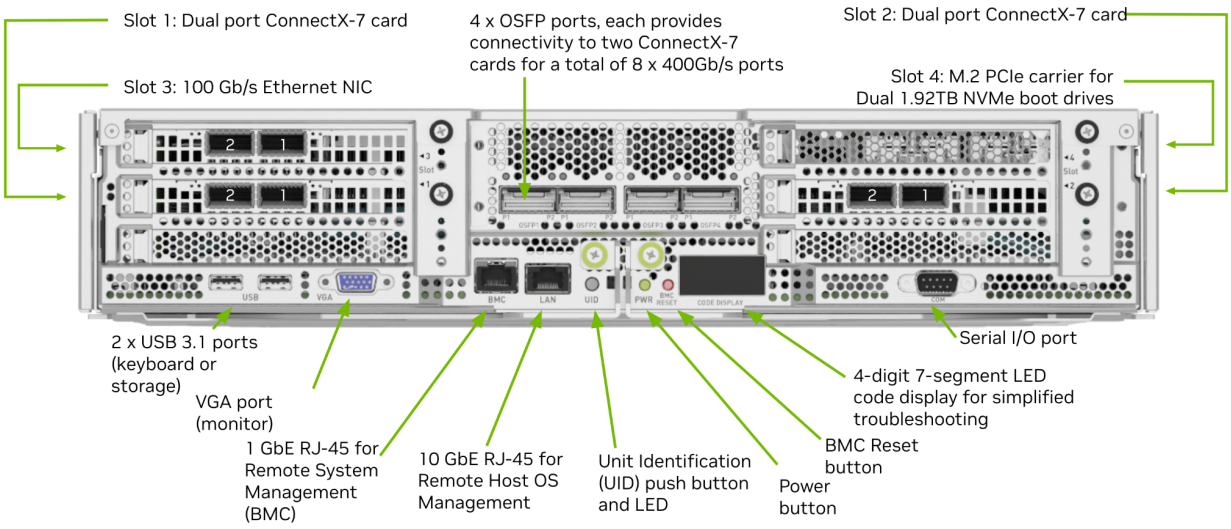


Figure 3.1: Physical layout of the back of the DGX H200/H100 system

Figure 3.2 shows how the DGX H200/H100 network ports are used in this deployment guide.

The following ports are selected for DGX BasePOD networking:

- Eight ports in four OSFP connections are used for the InfiniBandcompute fabric (marked in green).
- Two ports of the dual-port ConnectX-7 cards are configured as abonded Ethernet interface for in-band management and storage ethernet networks. These are the left ports (port 2) from slot 1 and slot 2 (marked in blue).
- Optional: Two ports of the dual-port ConnectX-7 cards when InfinIBand storage is used instead of Ethernet. These are the right ports (port 1) from slot 1 and slot 2 (marked in red).
- BMC network access is provided through the out-of-band network (marked in purple).

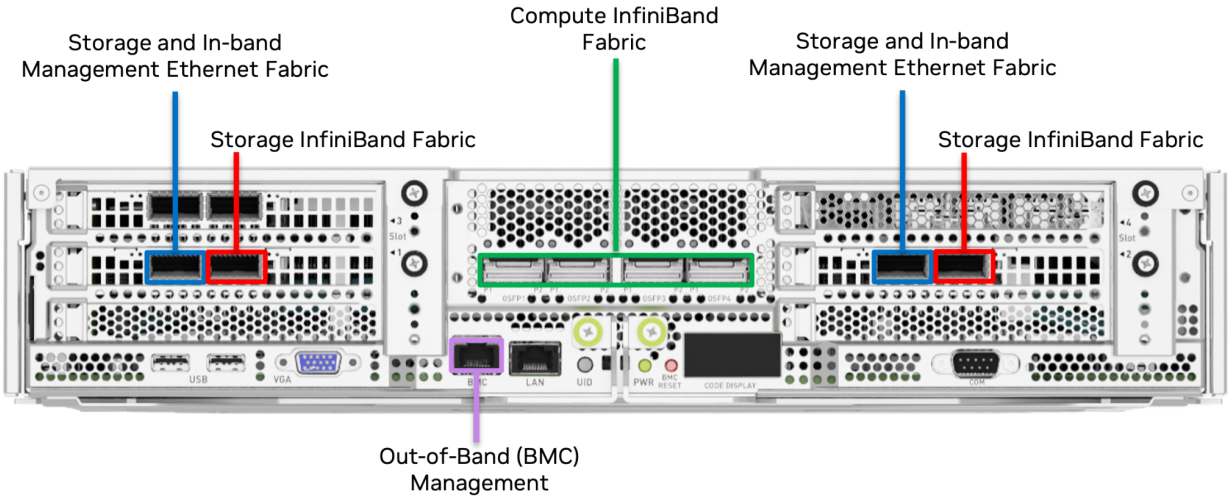


Figure 3.2: DGX H200/H100 network ports

The networking ports and their mapping are described in detail in the Network Ports section of the NVIDIA DGX H200/H100 System User Guide.

3.2. DGX BasePOD Network Overview

There are four networks in a DGX BasePOD configuration:

- managementnet (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.
- oobmanagementnet (ipminet)—Network for out of band management, connecting BMCs.
- computenet (ibnet)—InfiniBand network connecting all DGX systems' ConnectX-7 Compute Fabric HCAs.

These are shown in Figure 3.3.

3.3. Managementnet and External networks

managementnet (internalnet) and externalnet are configured on the SN4600C switches, which are the backbone of the DGX BasePOD Ethernet networking. Each DGX system connects to the SN4600C switches with a bonded interface that consists of two physical interfaces; slot 1 port 2 (storage 1-2) and slot 2 port 2 (storage 2-2) as described in the Network Ports section of the NVIDIA DGX H200/H100 System User Guide.

The K8s (Workload Manager nodes in the topology diagram) nodes and the NFS storage device have a similar bonded interface configuration connected to SN4600C switches. Two SN4600C switches with Multi-chassis Link Aggregation (MLAG) provides the redundancy for DGX systems, K8s nodes, and other devices with bonded interfaces.In this deployment guide, BGP (Border Gateway Protocol) is used for network connectivity between the managementnet (internalnet) and externalnet networks.

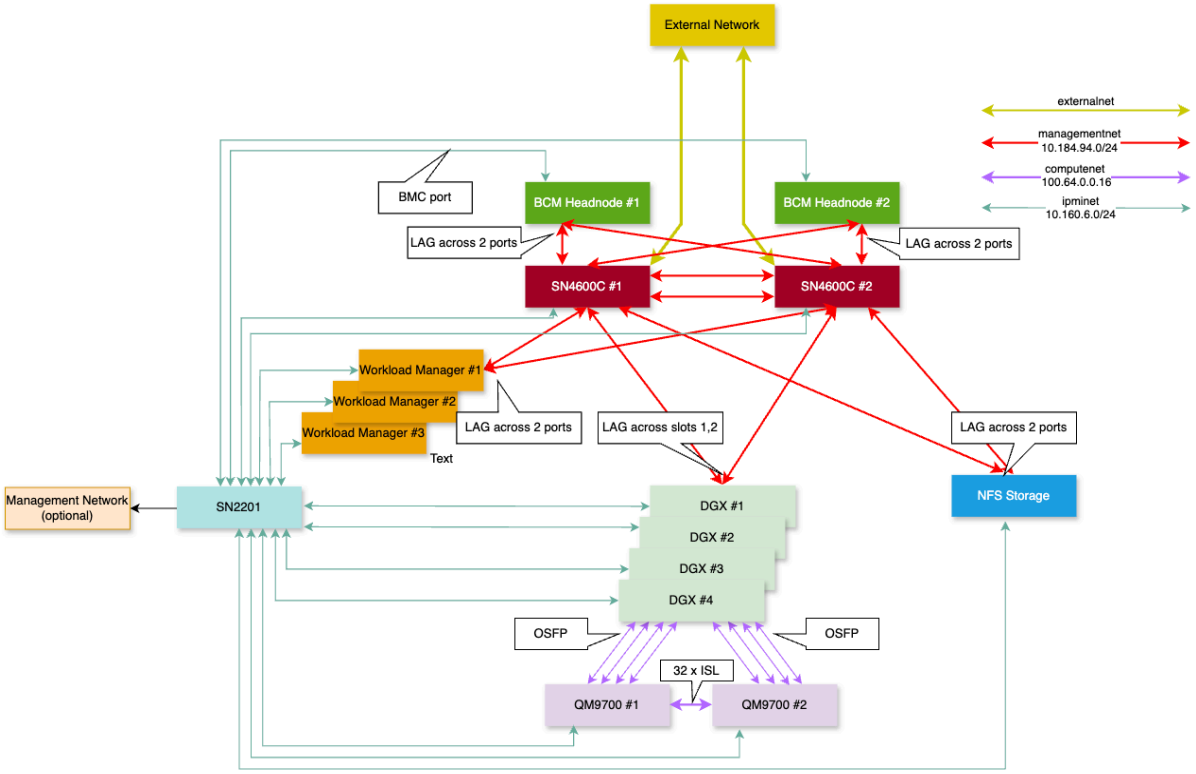


Figure 3.3: Network design and topology diagram.

Table 3.1: BGP Protocols

Protocol	Description
eBGP	Used as required for routing between switches and customer network
iBGP	Configured between the two SN4600C switches using the MLAG peerlink.4094 interface

3.4. oobmanagementnet (ipminet)

On the oobmanagementnet(ipminet) SN2201 switches, all the switch ports connected to the end hosts are configured as access ports. Each BCM headnode has its BMC interface connected to the IPMI switch. Uplinks are connected to SN4600C switches.

3.5. externalnet uplink

All connected subnets are redistributed into BGP. IPMI switches can also be uplinked to a separate management network if required, rather than the SN4600C switches; still IPMI subnet route must be advertised to the in-band network so that BCM can control hosts using the IPMI network.

3.6. computenet (ibnet)

For the computenet (ibnet), 4 ports of the DGX OSFP ports are connected to QM9700-1 InfiniBand switch, and 4 ports are connected to QM9700-2 InfiniBand switch. To manage the InfiniBand fabric, at least one subnet manager is required to be enabled on the QM9700 switches.

The networking ports and their mapping are described in the Network Ports section of the NVIDIA DGX H200/H100 System User Guide.

Chapter 4. Software Overview

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

An instance of BCM runs on a pair of head nodes in a 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 (workload manager) nodes are imaged by BCM with an Ubuntu LTS version equivalent to that of the DGX OS and the head nodes themselves.

4.1. Kubernetes (K8s)

K8s is a 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.

Chapter 5. NFS 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 Deployment Guide. A DGX BasePOD typically also includes dedicated storage, but the configuration of that is outside the scope of this document. Contact the DGX BasePOD certified storage vendor being used for instructions on configuring the high performance storage portions of a DGX BasePOD.

Chapter 6. BCM Introduction

This chapter details deploying the NVIDIA Base Command Manager (BCM) on NVIDIA DGX BasePOD™ configurations.

Physical installation and network switch configuration must be completed before deploying BCM. In addition, information about the intended deployment should be recorded in the Site Survey. A sample Site Survey can be found in the Appendix.

It's essential to get familiar with the DGX BasePOD reference architecture, DGX H200/H100 motherboard connections and its network port designations before you proceed with this deployment guide:

- DGX BasePOD reference architecture
- DGX H200/H100 Motherboard connections
- DGX H200/H100 Network port designations

The following reference documents can be helpful during the deployment:

- BCM Installation Manual
- BCM Administrator Manual
- DGX OS User Guide
- Cumulus User Guide (for SN4600C and SN2201 switches)
- InfiniBand NVOS with QM9700 switches
- MLNX-OS with QM9700 switches

Chapter 7. Network Deployment

It's essential to get familiar with the DGX BasePOD Reference Architecture before proceeding

Note

Before powering on any of the switches, ensure physical serial port connections have been established, then proceed to power on all the switches.

7.1. SN4600C – managementnet ethernet switches

The SN4600c managementnet fabric provides connectivity for inband management and provisioning of the nodes. The key configuration requirements are

- MLAG between the two SN4600C switches
- L3 SVI/VRRP for all the pod ethernet networks
- Each headnode / K8s node / DGX is dual homed to the SN4600C switches via bond interface
- External connectivity to customer network, using customer specified routing arrangements, like BGP (Border Gateway Protocol) or static or other dynamic routing protocols
- Link to IPMI Network for BCM to access node BMCs, either direct or indirect via customer network.

7.1.1. SN4600C-1 Reference Configuration

```
# Basic management configuration
nv set system hostname 4600C-1
#
# Create SVIs for Internal/Management Network with VRRP as FHRP
nv set bridge domain br_default vlan 102
nv set interface vlan102 type svi
nv set interface vlan102 ip vrr mac-address 00:00:5E:00:01:01
nv set interface vlan102 ip vrr address 10.184.94.1/24
nv set interface vlan102 ip address 10.184.94.2/24
nv set interface vlan102 ip vrr state up
# Repeat the same for other SVI interfaces
```

(continues on next page)

(continued from previous page)

```
# Configure MLAG
# Define inter-chassis peerlink etherchannel/bond
nv set interface peerlink bond member swp63,swp64
nv set interface peerlink type peerlink
#
# Loopback for BGP/MLAG backup routing
nv set interface lo ip address 10.160.254.22
#
# Configure Peerlink L3 parameters
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 10.160.254.23
nv set mlag enable on
nv set mlag mac-address 44:38:39:ff:00:02
nv set mlag peer-ip linklocal
# MAG Primary
nv set mlag priority 2048
# Example port configuration for head nodes (BCM, Kube)
# BCM Head Nodes
nv set interface bond1 bond member swp1
nv set interface bond1 description "BCM Headnode 1"
nv set interface bond1 bond mlag id 1
nv set interface bond1 bridge domain br_default access 102
nv set interface bond1 bond mlag enable on
nv set interface bond1 bond lacp-bypass on
# Repeat for other management/workloads/compute nodes
#
# Uplink to the customer network.
# Example configuration with BGP unnumbered
nv set router bgp autonomous-system 4200004001
nv set router bgp enable on
nv set router bgp router-id 10.160.254.22
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
# Uplinks via swp50
nv set vrf default router bgp neighbor swp50 type unnumbered
# Peering to MLAG peer switch
nv set vrf default router bgp neighbor peerlink.4094 remote-as internal
nv set vrf default router bgp neighbor peerlink.4094 type unnumbered
```

Refer to the appendix for complete switch configuration.

7.1.2. SN4600C-2 Reference Configuration

Same as SN4600C-1, with the following changes

```
# Basic management configuration
nv set system hostname 4600C-2
```

(continues on next page)

(continued from previous page)

```
#
# Create SVIs - Internal/Management Network with VRRP as FHRP
nv set bridge domain br_default vlan 102
nv set interface vlan102 type svi
nv set interface vlan102 ip vrr mac-address 00:00:5E:00:01:01
nv set interface vlan102 ip vrr address 10.184.94.1/24
nv set interface vlan102 ip address 10.184.94.3/24
nv set interface vlan102 ip vrr state up
#
# Configure MLAG
# Define inter-chassis peerlink etherchannel/bond
#
# BGP/MLAG backup routing loopback
nv set interface lo ip address 10.160.254.23
#
# Configure Peerlink L3 parameters
nv set mlag backup 10.160.254.22
nv set mlag mac-address 44:38:39:ff:00:02
# MLAG Secondary
nv set mlag priority 4096
#
# Example port configuration - head nodes (BCM, Kube)
# same as 4600-1
#
# Uplink to the customer network.
# Same as 4600-1
```

Refer to the appendix for complete switch configuration.

You can verify the MLAG status using the following command

```
root@mgmt-net-leaf-1:mgmt:/home/cumulus# clagctl
The peer is alive
     Our Priority, ID, and Role: 2048 9c:05:91:dd:cc:28 primary
    Peer Priority, ID, and Role: 2048 9c:05:91:f1:73:28 secondary
          Peer Interface and IP: peerlink.4094 fe80::9e05:91ff:fef1:7328
↪(linklocal)
                     Backup IP: 10.160.254.23 vrf mgmt (inactive)
                   System MAC: 44:38:39:ff:0a:00

CLAG Interfaces
Our Interface      Peer Interface     CLAG Id   Conflicts              Proto-
↪Down Reason
----------------   ----------------   -------   --------------------   -------
↪----------
           bond1   -                  1         -                      -
          bond10   -                  10        -                      -
          bond11   -                  11        -                      -
          bond12   -                  12        -                      -
          bond13   -                  13        -                      -
          bond14   -                  14        -                      -
```

For troubleshooting, you can use the consistency check command. Here is an example output from a

working MLAG pair.

```
cumulus@mgmt-net-leaf-2:mgmt:~$ nv show mlag consistency-checker global
Parameter               LocalValue                 PeerValue
 ↪Conflict  Summary
----------------------  -------------------------  -------------------------
 ↪--------  -------
anycast-ip              -                          -
 ↪-
bridge-priority         32768                      32768
 ↪-
bridge-stp-mode         rstp                       rstp
 ↪-
bridge-stp-state        on                         on
 ↪-
bridge-type             vlan-aware                 vlan-aware
 ↪-
clag-pkg-version        1.6.0-cl5.11.0u2           1.6.0-cl5.11.0u2
 ↪-
clag-protocol-version   1.7.0                      1.7.0
 ↪-
peer-ip                 fe80::9e05:91ff:fedd:cc28  fe80::9e05:91ff:fedd:cc28
 ↪-
peerlink-bridge-member  Yes                        Yes
 ↪-
peerlink-mtu            9216                       9216
 ↪-
peerlink-native-vlan    1                          1
 ↪-
peerlink-vlans          1, 100->102                1, 100->102
 ↪-
redirect2-enable        yes                        yes
 ↪-
system-mac              44:38:39:ff:0a:00          44:38:39:ff:0a:00
 ↪-
```

7.2. SN2201 – IPMI Switch for Out-of-Band Management

All the BMCs are in the same oobmanagementnet subnet, configure all switch ports connected to the BMCs to be under the same VLAN.The oobmanagementnet should be accessible from the managementnet to allow the BCM headnodes to control the BMCs. In this example, the oobmanagementnet is routed via the managementnet SN4600C switches. It is recommended to add an additional uplink to the customer's OOB network.

Example Configuration for the SN2201 switch.

```
nv set system hostname IPMI-SW
#<Basic management configuration>
#
# VLAN - BMC ports. Adjust according to the customer
```

(continues on next page)

(continued from previous page)

```
specification
nv set bridge domain br_default vlan 101
#
# Enable the BMC Ports to the Access VLAN
#
nv set interface swp1-48 bridge domain br_default
nv set bridge domain br_default untagged 1
nv set interface swp1-48
nv set interface swp1-48 link state up
nv set interface swp1-48 description "BMC Ports"
nv set interface swp1-48 bridge domain br_default access 101
#
# Uplink to customer OOB/PIMI Network
# In this example the uplink is a layer 2 trunk with etherchannel/bond.
# Adjust according to the customer specification
nv set interface swp49-50 link state up
nv set interface bond1 bond member swp49,swp50
nv set interface bond1 bridge domain br_default untagged 1
nv set interface bond1 bridge domain br_default vlan all
```

Refer to the appendix for complete switch configuration.

Reference: Cumulus Network configuration Guide.

You can also use NVIDIA Air to simulate and model the network configuration.

Once the SN2201 switches have been successfully configured, verify that all devices out of band management interfaces are reachable from the network. (i.e. make sure you can access the BMC/iLO/iDRAC).

7.3. Computenet Configuration

Before powering on any of the QM9700 switches in the Compute or Storage switch stacks ensure that serial port connectivity can be established (either via remote serial concentrator or physically interfacing with the serial port of the switch), then proceed to power on all Compute & Storage switches.

7.4. QM9700 IB Switches

We recommend configuring the InfiniBand switches with subnet manager HA enabled.

Example configuration

7.4.1. QM-9700-1

```
ib sm
ib sm virt enable
ib smnode 9700-1 create
ib smnode 9700-1 enable
```

(continues on next page)

(continued from previous page)

```
ib smnode 9700-1 sm-priority 15
ib ha infiniband-default ip <HA VIP> <mask>
```

7.4.2. QM-9700-2

```
ib sm virt enable
ib smnode 9700-1 create
ib smnode 9700-1 enable
ib smnode 9700-1 sm-priority 15
```

Verify IB SM HA status using the following command

```
QM9700-1[infiniband-default: master] # show ib smnodes
HA state of switch infiniband-default:
IB Subnet HA name: infiniband-default
HA IP address    : 10.185.230.247/22
Active HA nodes  : 2

HA node local information:
  Name       : 9700-2 (active)
  SM-HA state: standby
  SM Running : stopped
  SM Enabled : disabled
  SM Priority: 0
  IP         : 10.185.230.243

HA node local information:
  Name       : 9700-1 (active)  <--- (local node)
  SM-HA state: master
  SM Running : running
  SM Enabled : enabled - master
  SM Priority: 15
  IP         : 10.185.231.43
```

Refer to the *Appendix* for complete switch configuration.

Reference: Nvidia QM9700 InfiniBand Switch user manual

7.4.3. InfiniBand/Ethernet Storage Fabric Specific Configurations

A DGX BasePOD typically also includes dedicated storage, but the configuration 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 8. BCM Headnodes Pre-install Preparation

8.1. NFS server

NFS is used for BCM headnode HA. User home directories (`/home`) and shared data directories (`/cm_shared`, which includes files such as the DGX OS image) must be shared between head nodes and are stored on an NFS filesystem that both headnodes mount.

Because DGX BasePOD does not mandate the nature of the NFS storage, the configuration is outside the scope of this document. This DGX BasePOD deployment uses the NFS export path provided in the BasePOD site survey.

The parameters below are recommended for the NFS server export (for example, as specified in the file `/etc/exports`). In particular, the exported NFS directory must be mountable read-write, and files must be allowed to be owned by UID 0 (`root`); these are indicated by the `rw` and `no_root_squash` directives in the example below.

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

8.2. Verify DNS & NTP servers

Make sure the DNS and NTP servers are reachable from within the cluster environment.

8.3. Cluster Nodes Configuration

As cluster nodes – control plane (BCM headnodes), workload management nodes, DGX's, compute & storage fabric switches are racked and cabled. It is recommended to configure each appliance's BIOS & out of band management interface (sometimes referred to as a BMC, IPMI, etc) ahead of time before the installation of BCM.

Once all cluster nodes have been successfully configured verify that all cluster nodes' out of band management interfaces are reachable from within the working cluster network space. (i.e. make sure you can load access the BMC/iLO/iDRAC).

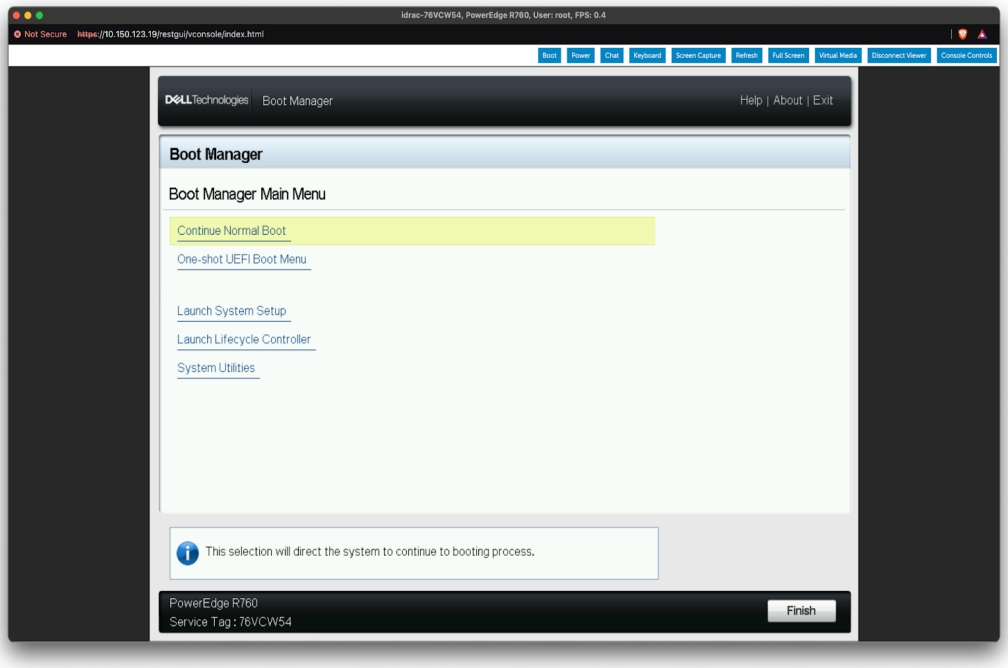
8.3.1. DGX BIOS Config and Network Interface Boot Order

Refer to the DGX System User Guide for specific steps on changing the boot order to use the 2 primary in-band interfaces to PXE boot first.

8.3.2. Control Plane and Workload Management Nodes

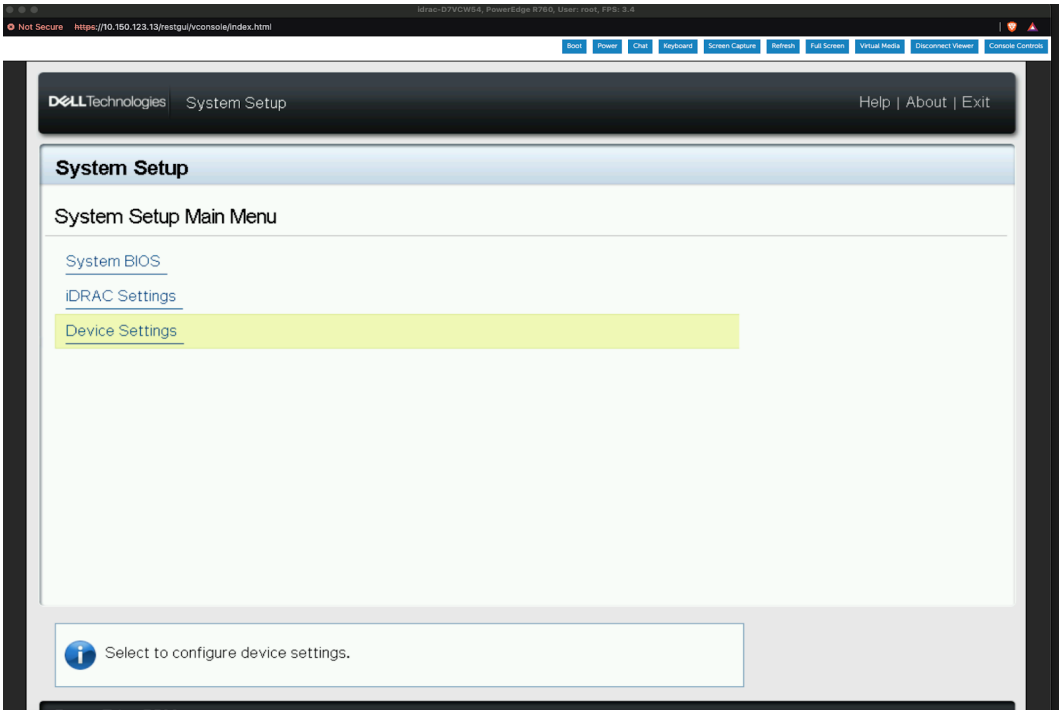
The following is an example for Dell appliances.

Refer to the below references for configuring an appliance with iDRAC9:

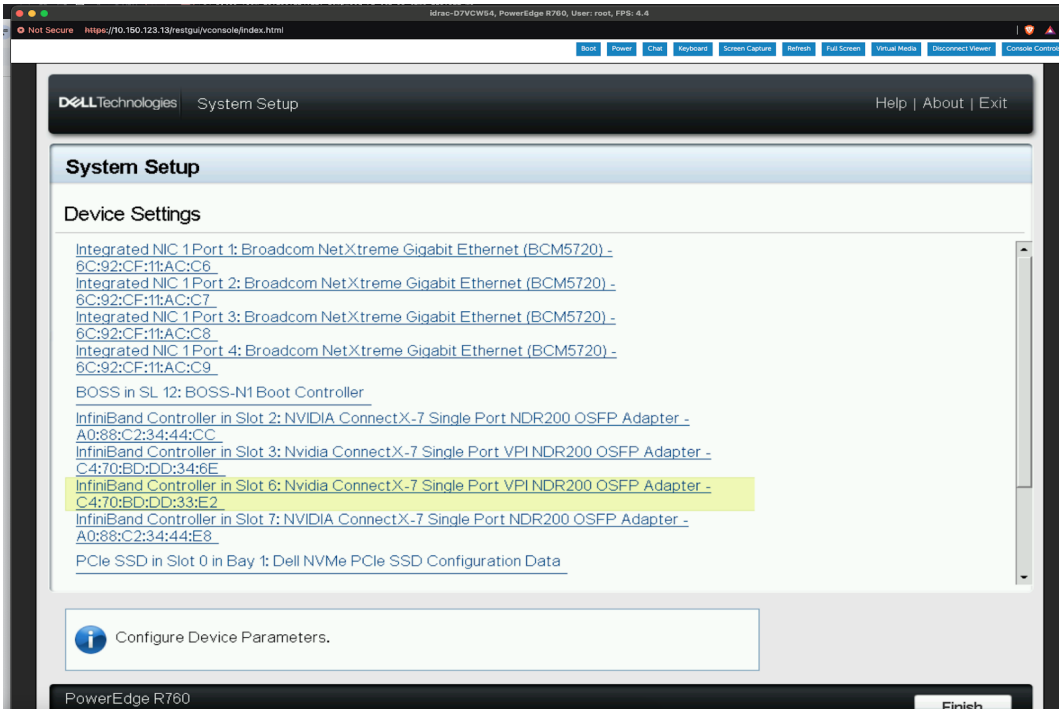


Interrupt the boot cycle to enter the Boot Manager and select “Launch System Setup”.

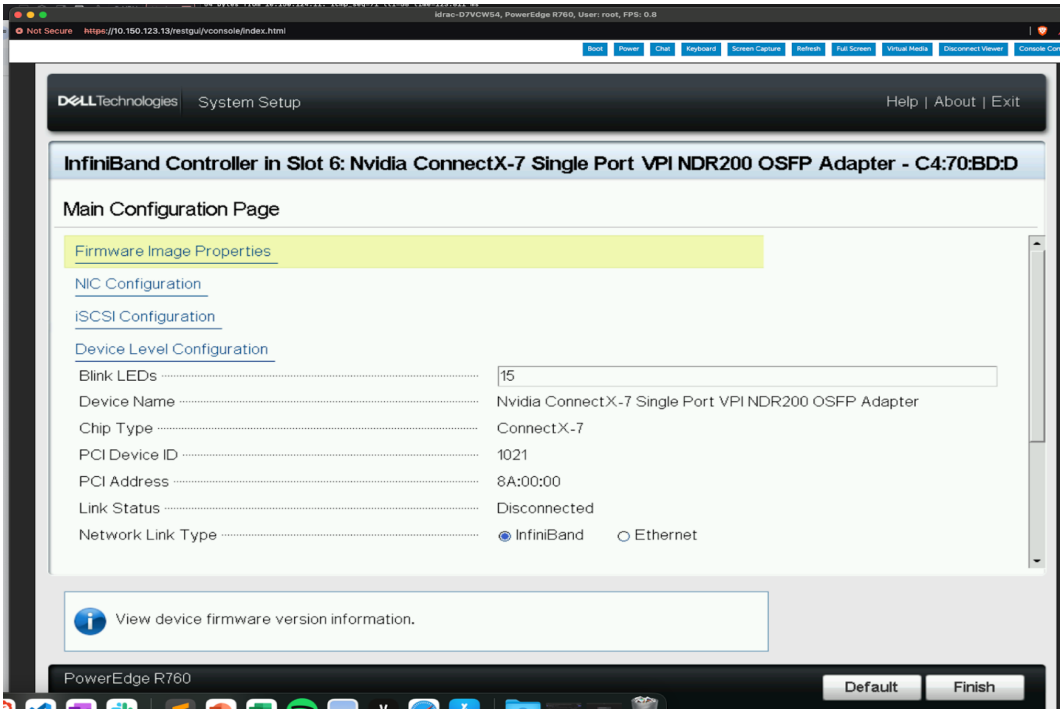
Next select “Device Settings”.



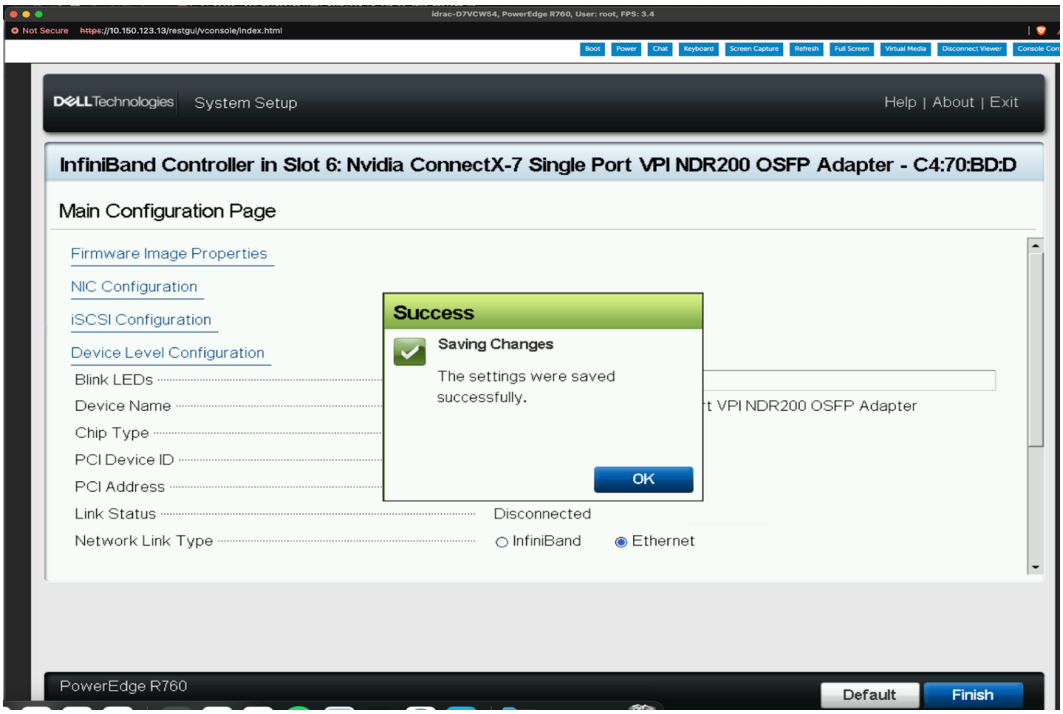
Select the Card that needs the mode flipped from Infiniband (IB Mode) to Ethernet (ETH Mode).



On this screen change the “Network Link Type” from “Infiniband” to “Ethernet” and select “Finish”.

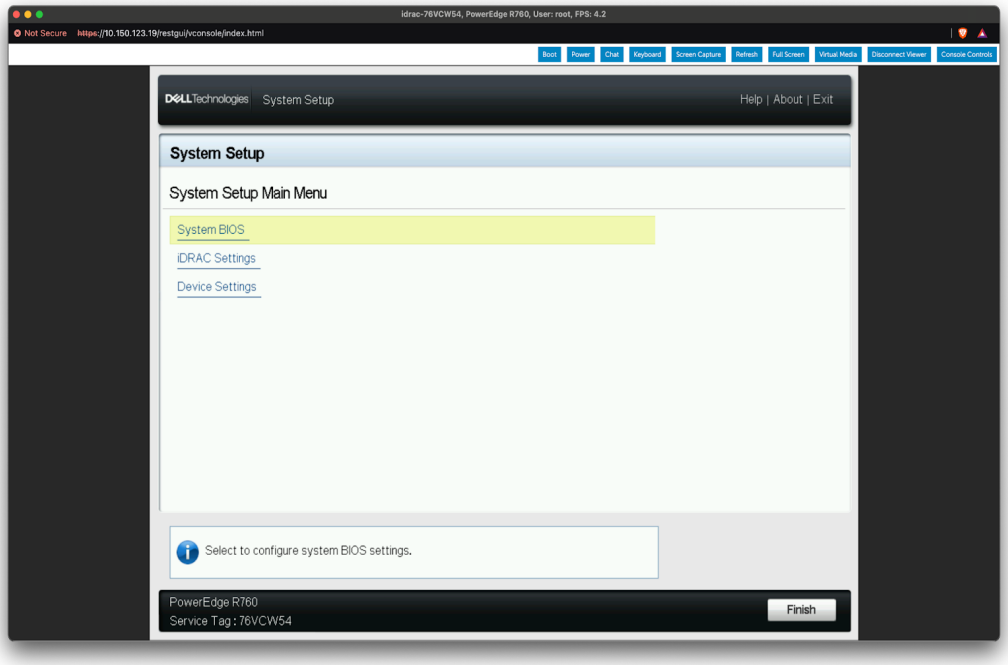


On seeing the following confirmation message, the card is now in Ethernet mode. Click OK.

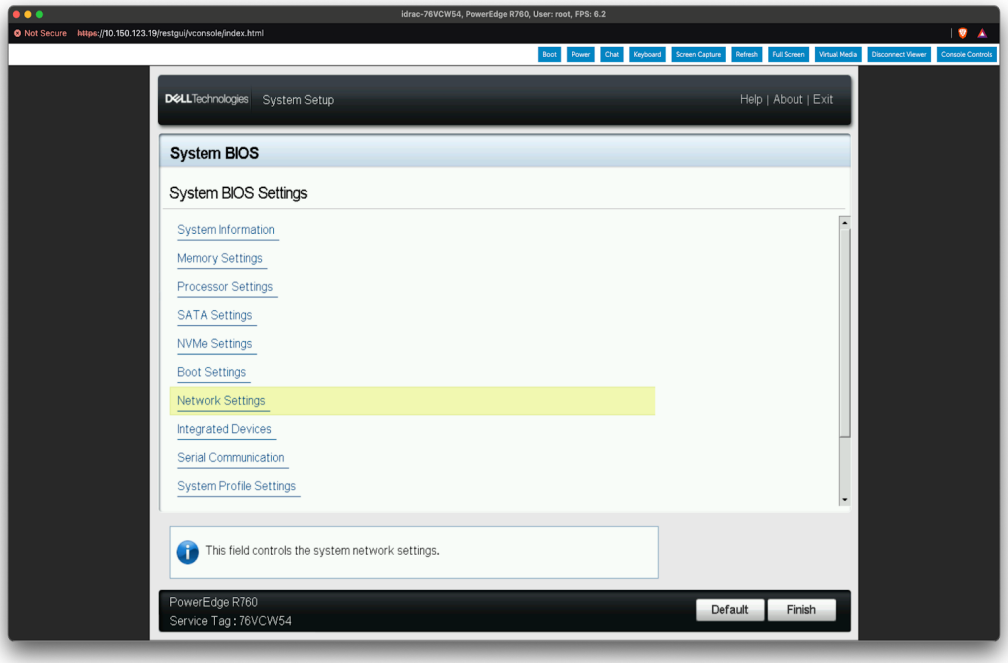


After confirming the CX card ports are in the correct mode we can proceed to configure the boot order.

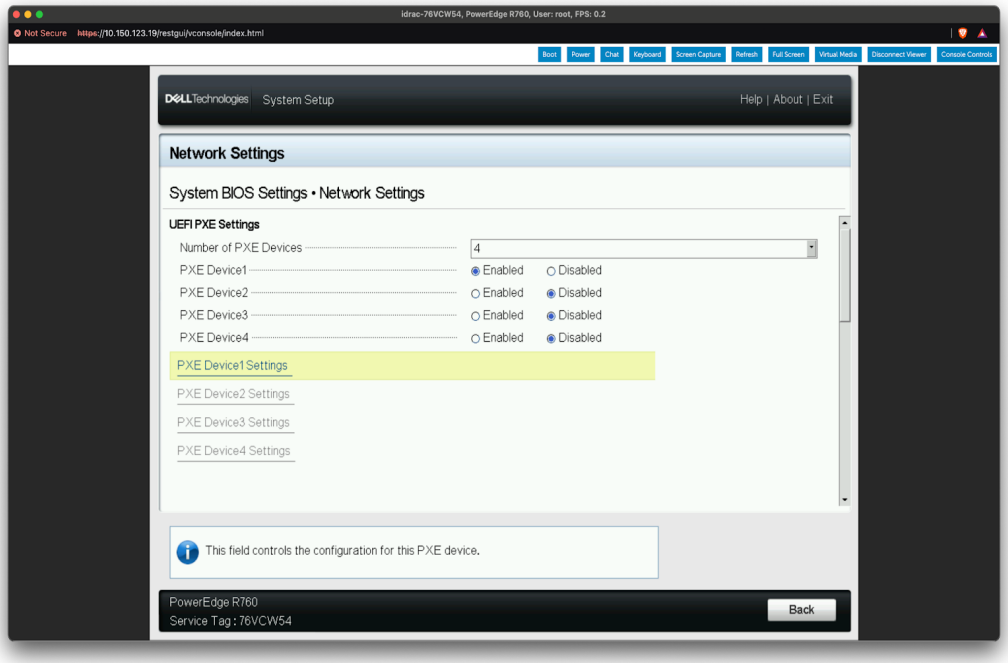
Return to the “System Setup” screen and select “System BIOS”.

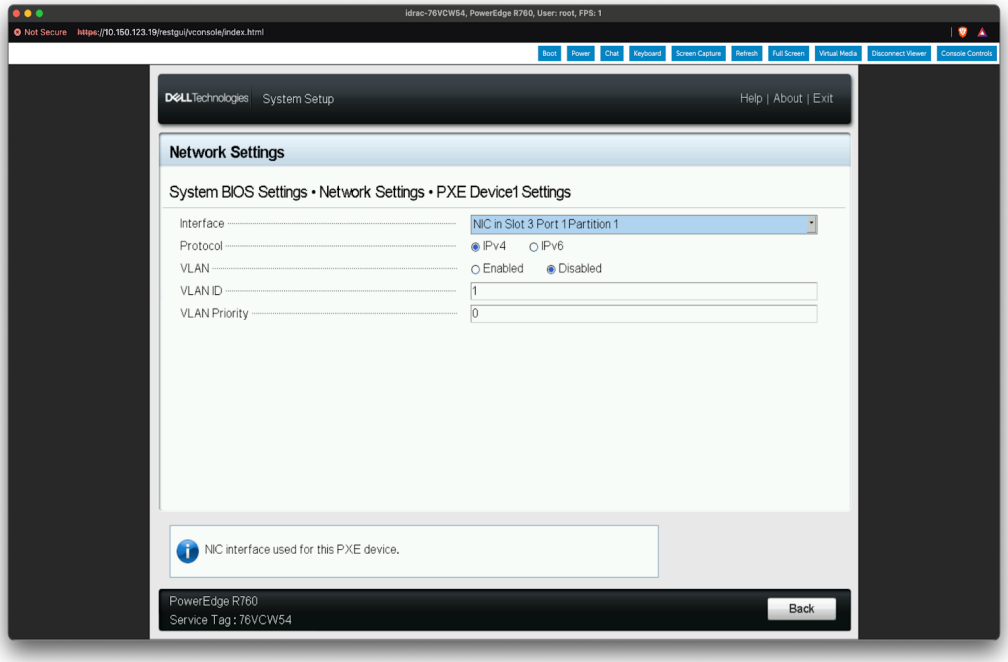


Select “Network Settings”.

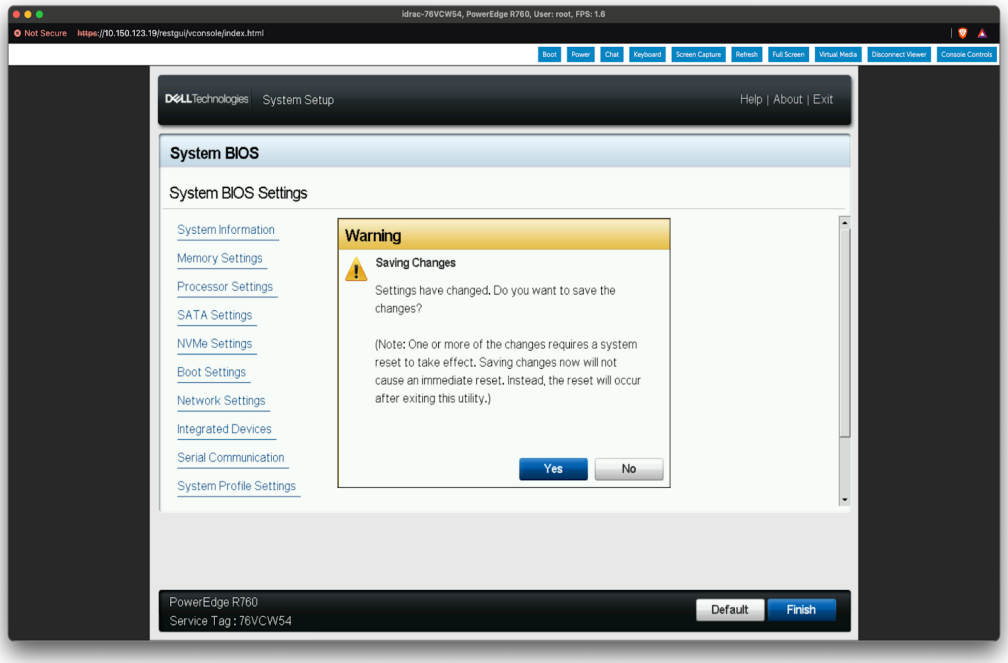


Enable a minimum of 2 PXE Devices, then define each PXE Device Setting such that the In-Band network port is selected as one of the 2 PXE boot interfaces.

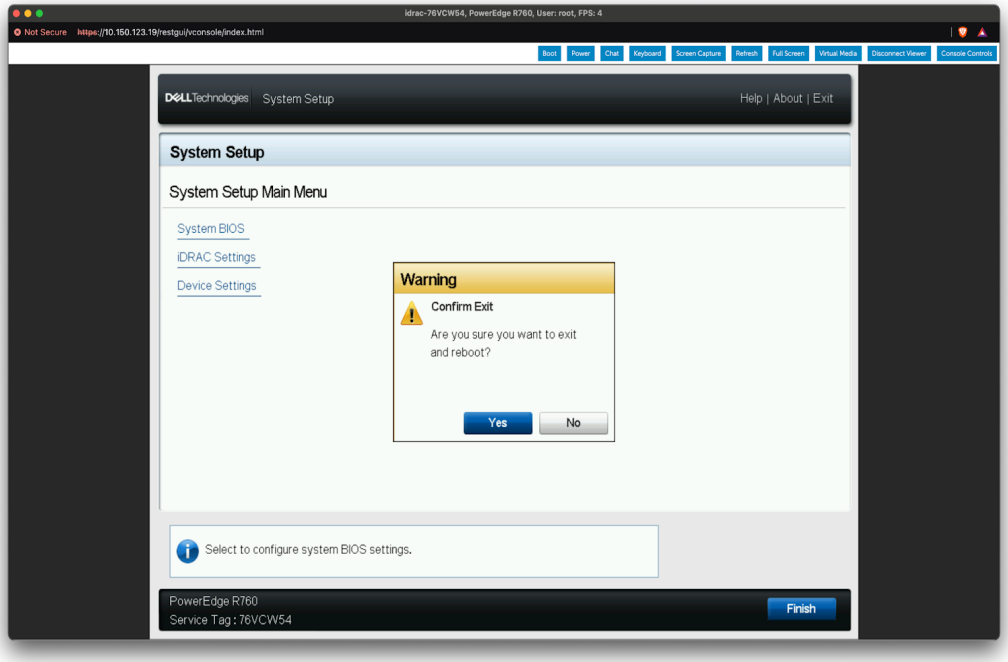




With both interfaces defined as PXE Devices click ”Back” to return to the ”System BIOS” screen with a ”Warning - Save Changes” prompt. Select ”Yes” to confirm saving the changes. Then click ”Finish“ to return to the System Setup Main Menu.



Select “Finish”, and on the “Warning - Confirm Exit” prompt select “Yes” to confirm the appliance re-boot.



8.3.3. Other Branded Appliances

On the ConnectX card that facilitates the In-Band network connections for the management nodes, ensure to set the port mode to Ethernet (not InfiniBand)

If the Connect-X card mode is not correctly set to Ethernet mode, the appliance will fail to communicate on the In-Band network.

The card's port mode can be modified by temporarily booting the appliance to a linux environment to install the NVIDIA Firmware Tools application which can flip the port mode using the below EXAMPLE command:

```
mlxconfig -d /dev/mst/mt4119_pciconf0 set LINK_TYPE_P1=2 LINK_TYPE_P2=2
```

Note

The specified command needs to be used with the correct device id, do not run the below example *as is* on a production system. Refer to NVIDIA Firmware Tools for usage details.

8.3.4. RAID/Storage Configuration

If available, configure the hardware RAID controller and disks to minimum RAID level 1 using the appliance's BMC or BIOS. The procedure varies depending on the appliance vendor and RAID controller. Refer to the specific vendor documentation for the configuration procedure.

Chapter 9. BCM Headnodes Installation

9.1. Download the Base Command Manager ISO

Download the BCM ISO image from the BCM website.

Be sure to select the following options for the download:

```
Version: Base Command Manager 10
Architecture: x86_64/amd64
Linux Base Distribution: Ubuntu 22.04
Hardware Vendor: NVIDIA DGX
Additional Features: Include MOFED Packages & Include NVIDIA DGX OS
software images for DGX H100 & DGX A100
```

Validate the downloaded file by verifying the MD5 checksum

```
$md5sum bcm-10.0-ubuntu2204-dgx-os-6.3.iso

66ecc05da5b0ed89cf365a168af86ec9 bcm-10.0-ubuntu2204-dgx-os-6.3.iso
```

Burn the BCM ISO to a DVD or to a bootable USB device. The ISO can also be mounted as virtual media and installed using appliance BMC Virtual Console.

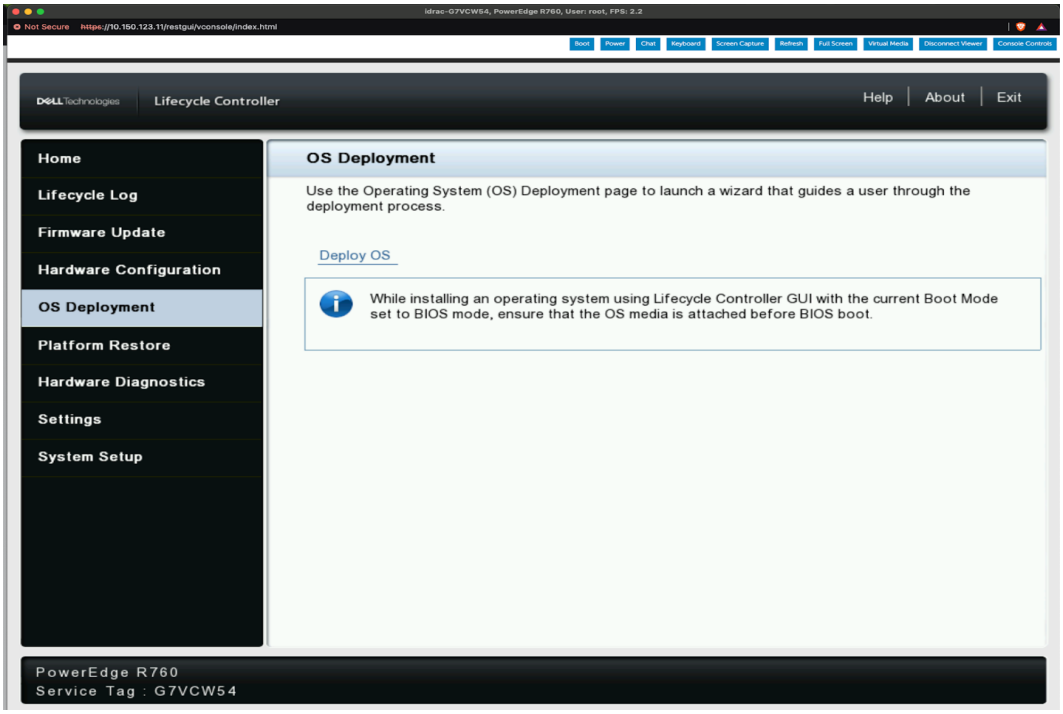
9.2. Primary Headnode preparation

Before starting the installation process it's important to ensure the Headnode's storage media are all in a freshly wiped state, and perform a BIOS configuration default. This resets all boot devices/order priorities.

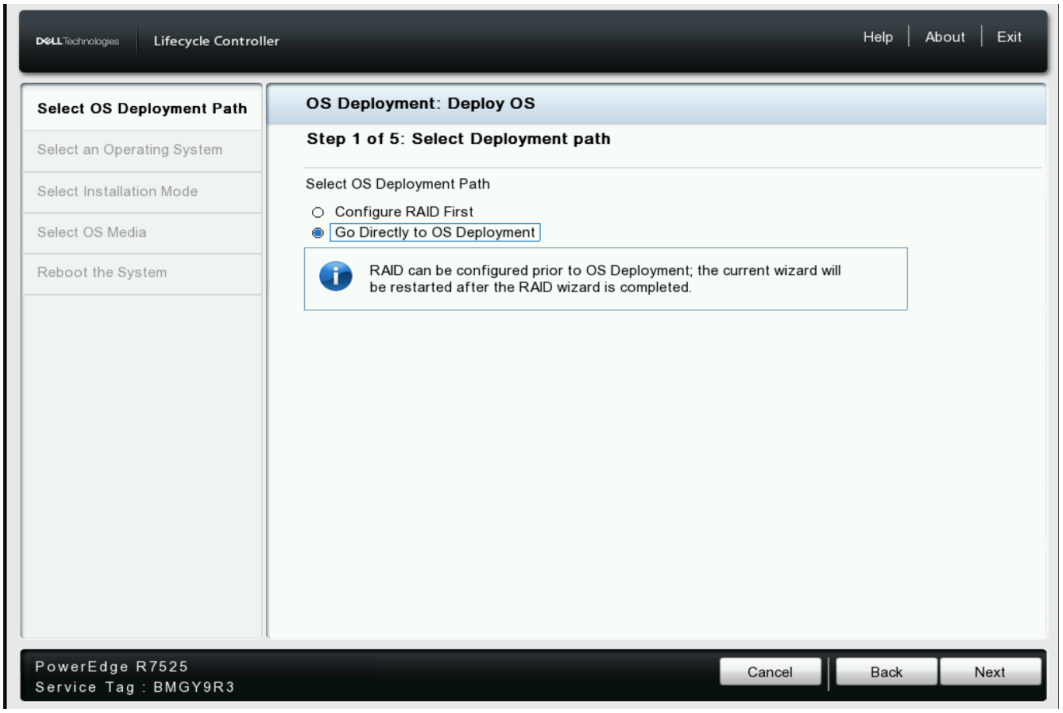
The following is an example of Dell appliances.

9.3. Dell appliances with iDRAC9

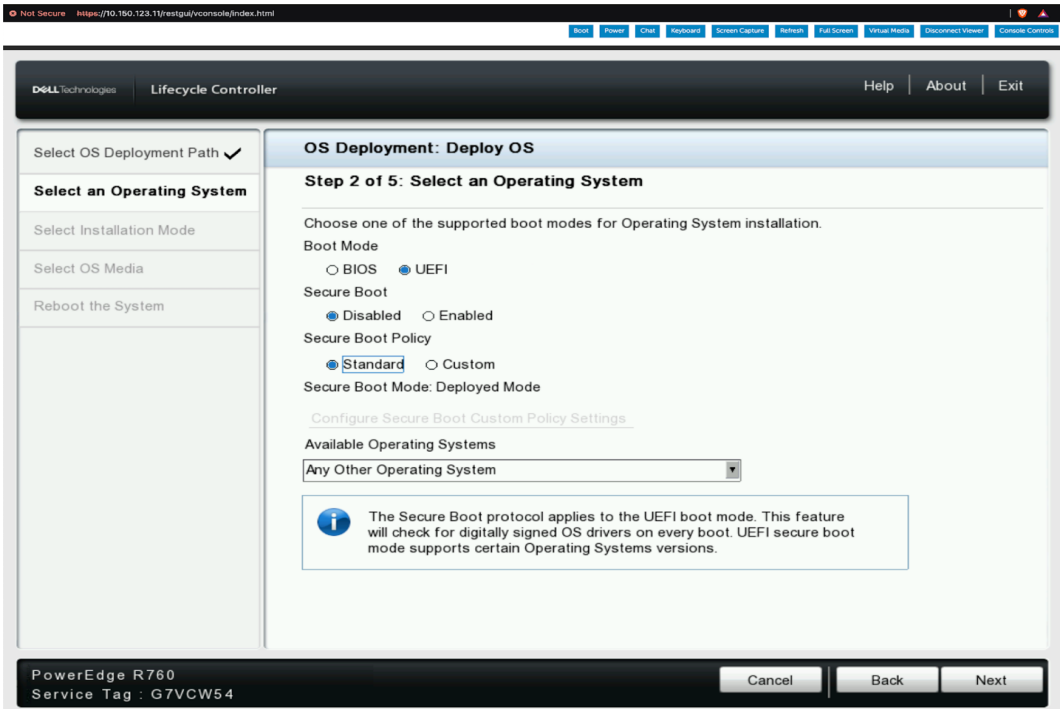
Access the appliance's iDRAC9 web portal, click the boot button and select Lifecycle Controller. Power cycle the appliance and boot into the Lifecycle Controller. Select "OS Deployment" on the left side of the screen and then click "Deploy OS".



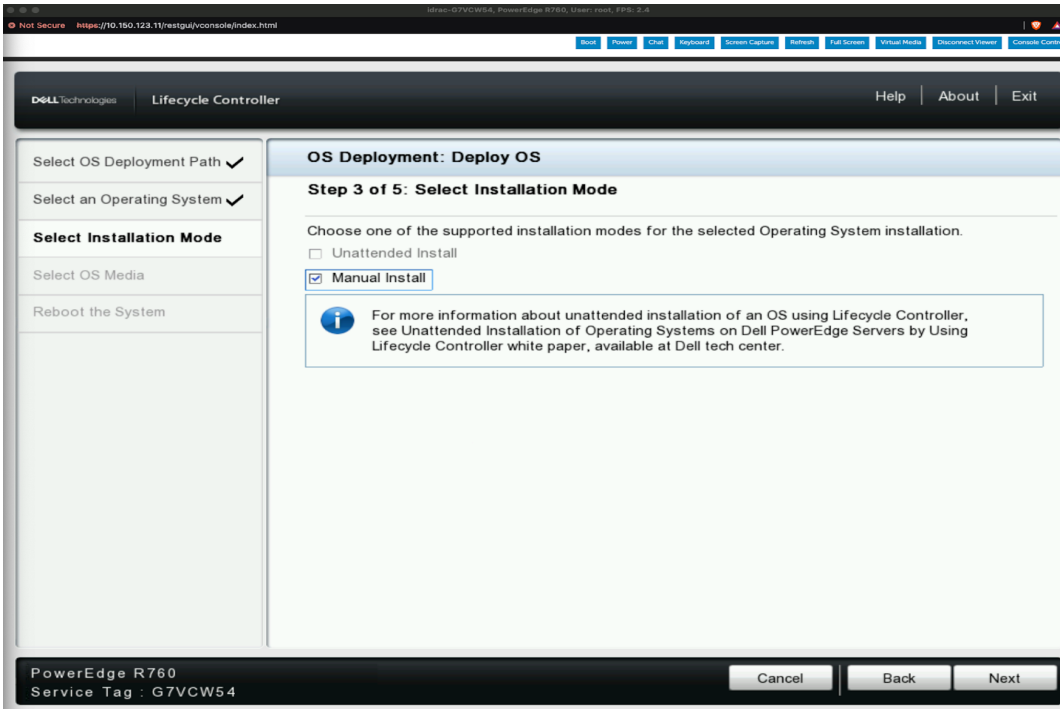
Select “Go Directly to OS Deployment” then click “Next”.



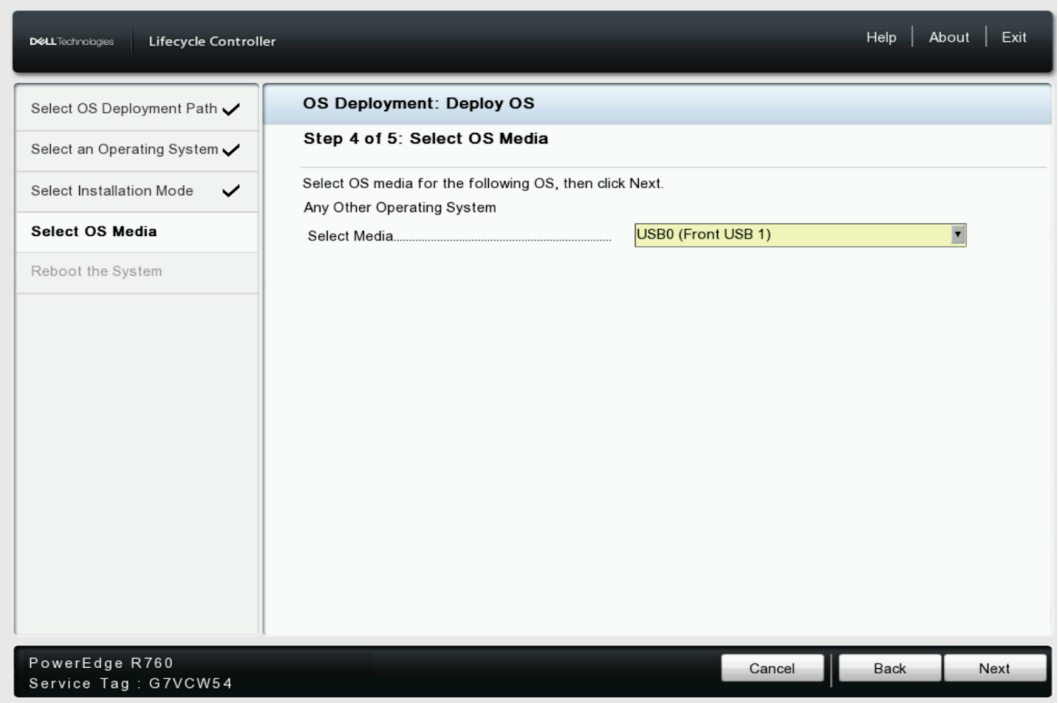
Ensure that Boot Mode is set to UEFI, Secure Boot is Disabled, Secure Boot Policy is set to Standard, and lastly that “Any Other Operating System” is set for the Available Operating System. Click “Next” to proceed.



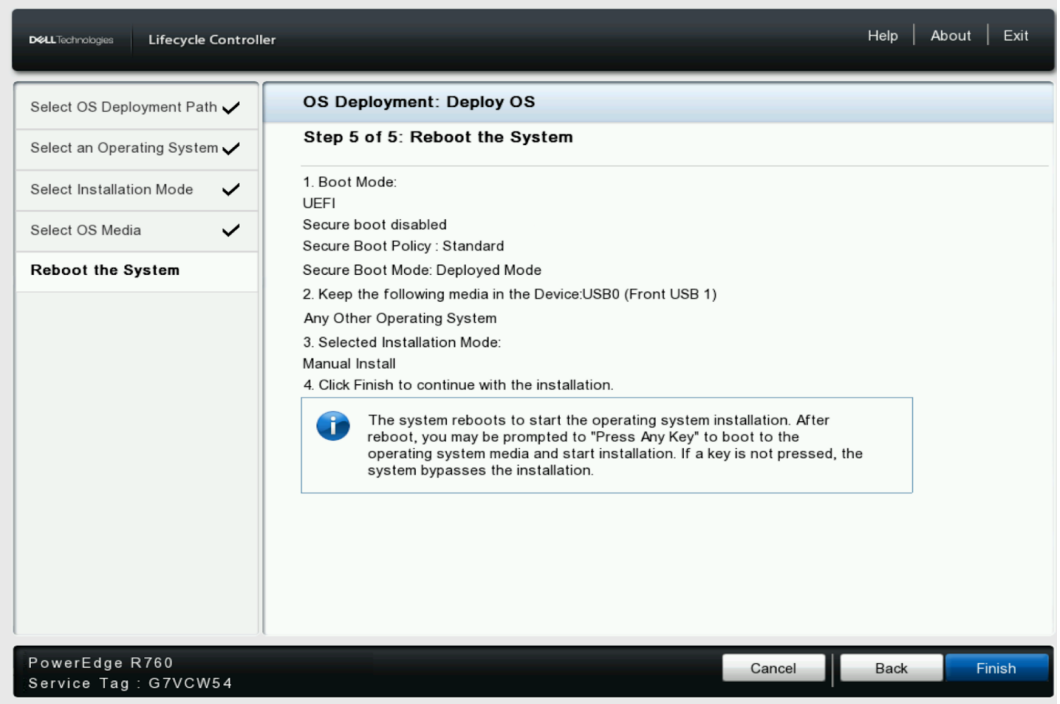
Next select the option for “Manual Install” and click “Next”.



Proceed to choose the appropriate Media/Virtual Media containing the BCM10 Installation ISO and then select “Next”.



Confirm the selected options, if any adjustments need to be made click the “Back” button to return to the appropriate screen to make a correction. If all options have been confirmed as correct select “Finish”.



The Dell appliance will proceed to boot as normal.

9.4. Other Vendor Appliances

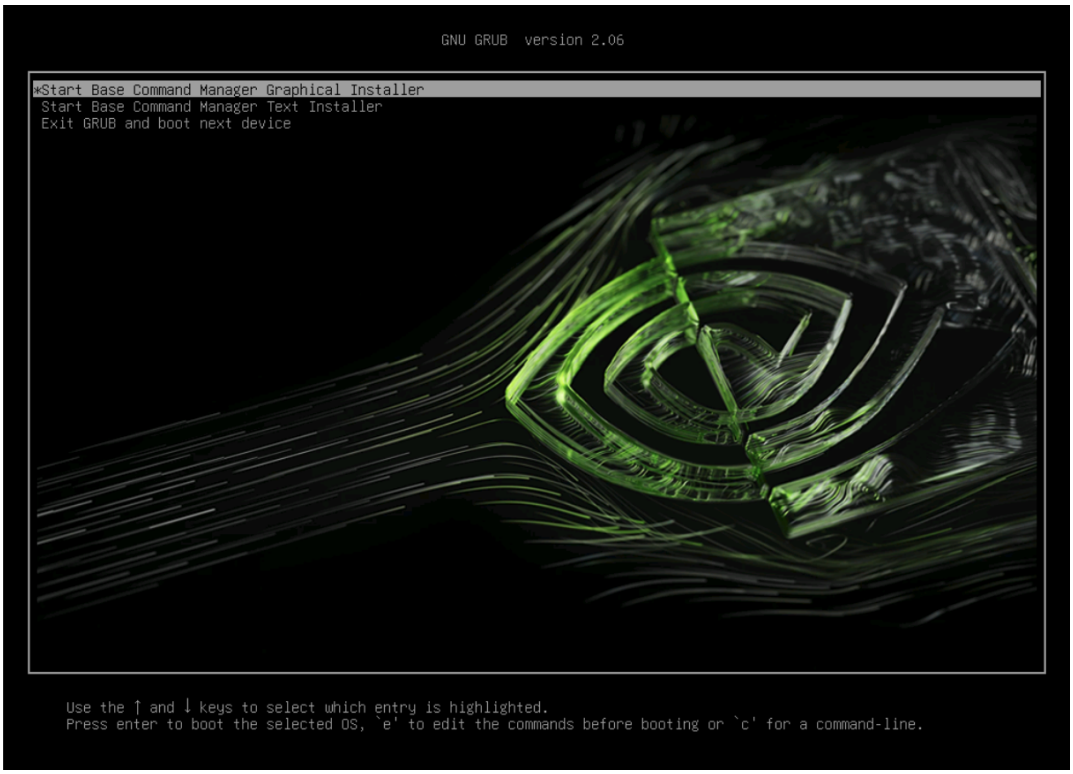
Attach the BCM10 installation media to the designated Headnode1 appliance.

Power on Headnode1 and proceed to boot from the BCM installation media.The specific procedure may vary by vendor, follow the respective vendor's user manual for details.

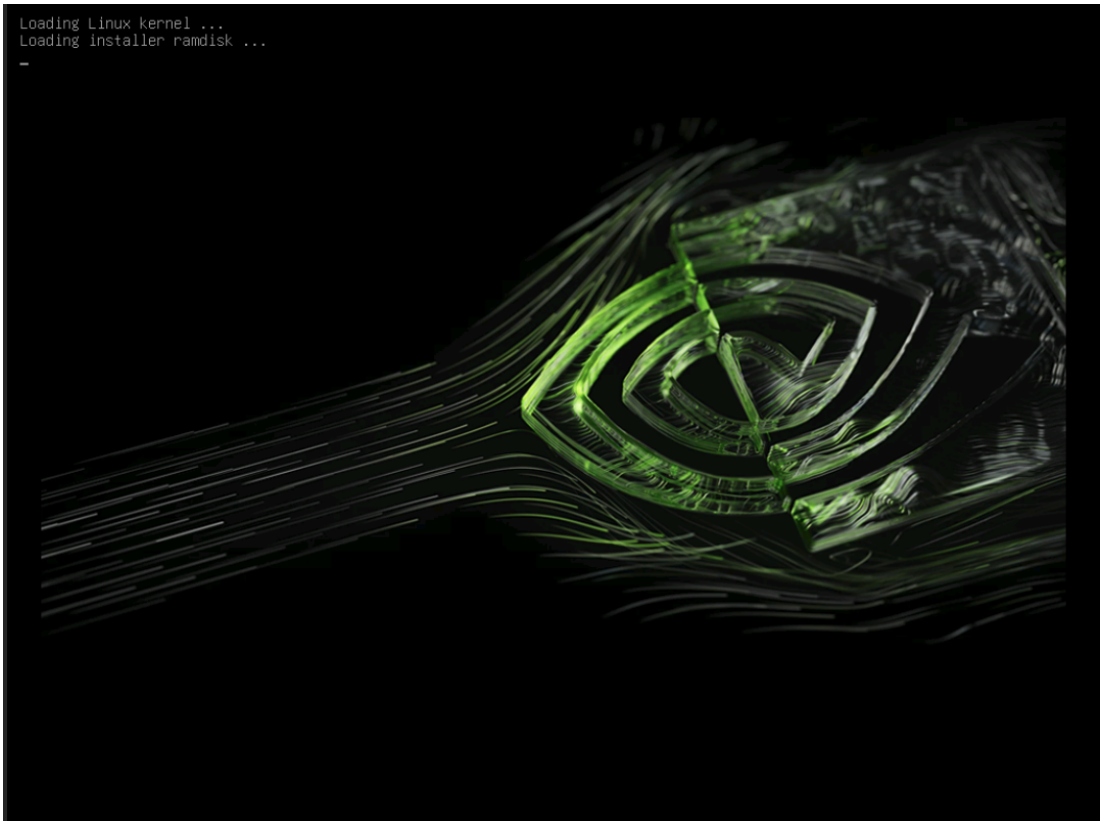
9.5. Booting the Base Command Manager Graphical Installer

After booting from the BCM ISO, at the grub menu, highlight **Start Base Command Manager Graphical Installer** using the arrow keys then press enter/return to select the option.

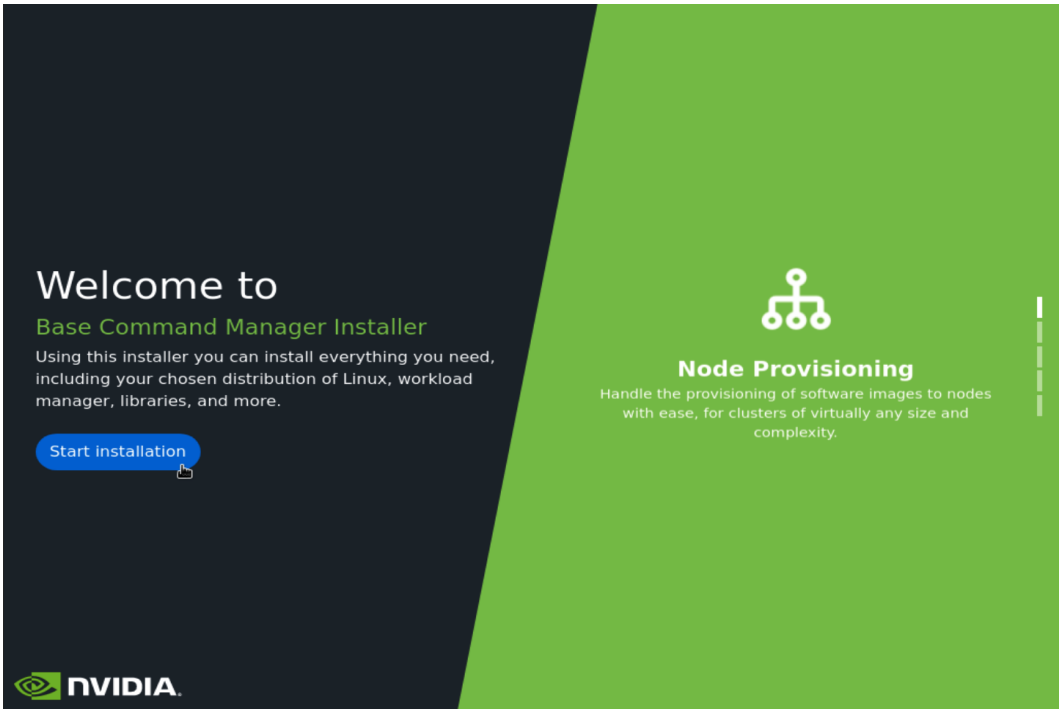
This step has an automated countdown timer, to interrupt the timer simply use the up or down arrow key.



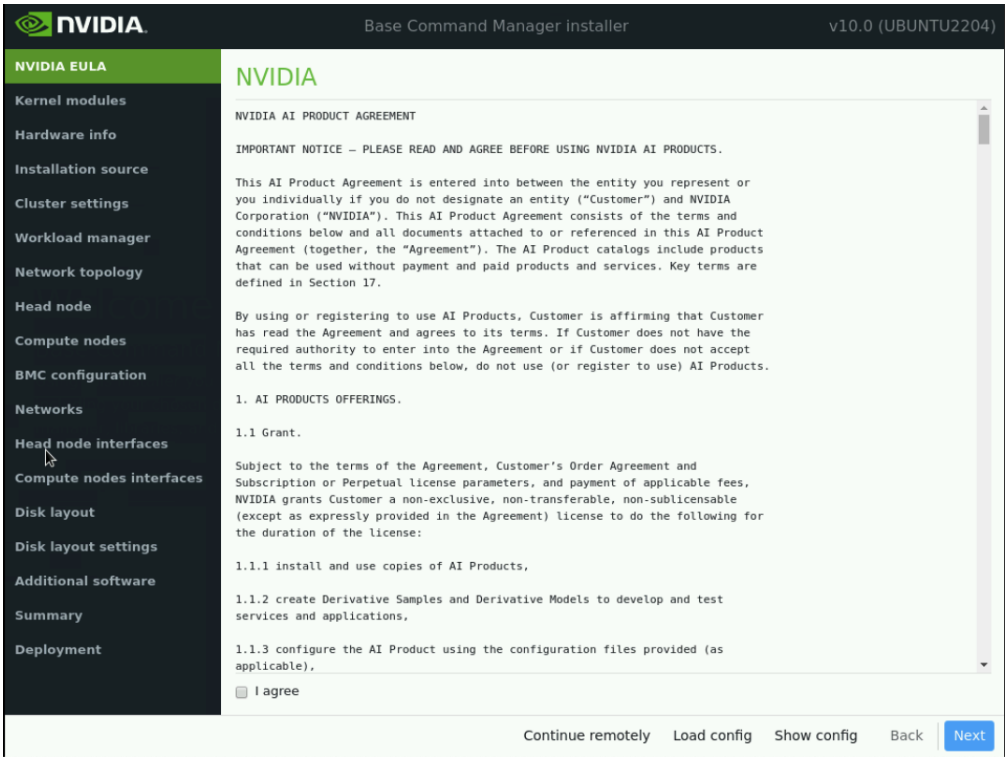
If you see the following after selecting "Start Base Command Manager Graphical Installer" this is expected and patience is needed while the installer loads up.



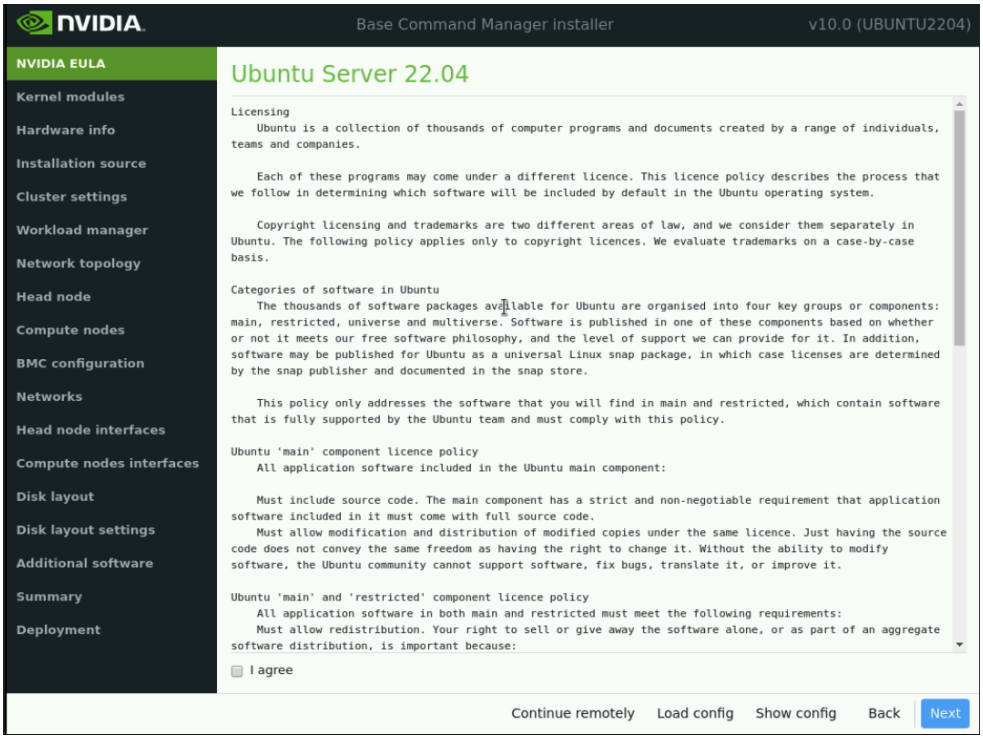
From here we can proceed to use the mouse to click Start installation on the Installer splash screen.



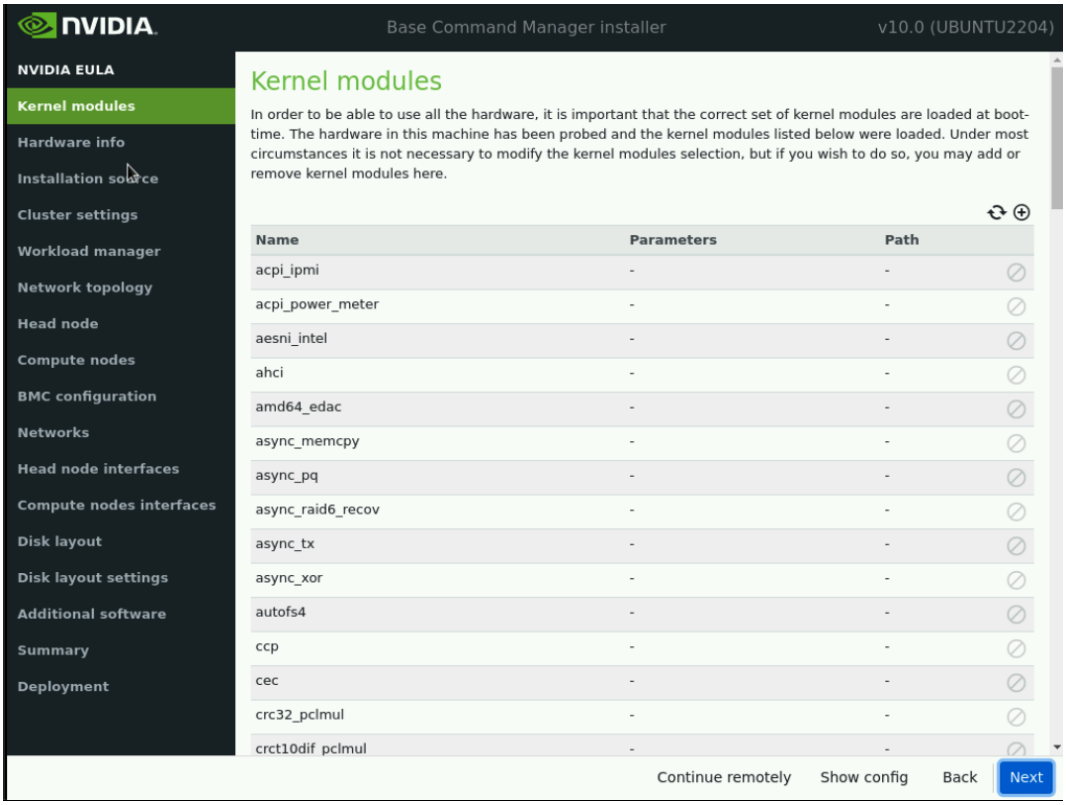
Accept the terms of the **NVIDIA EULA** by checking **I agree** and then select **Next**.



Accept the terms of the **Ubuntu Server EULA** by checking **I agree** and then select **Next**.



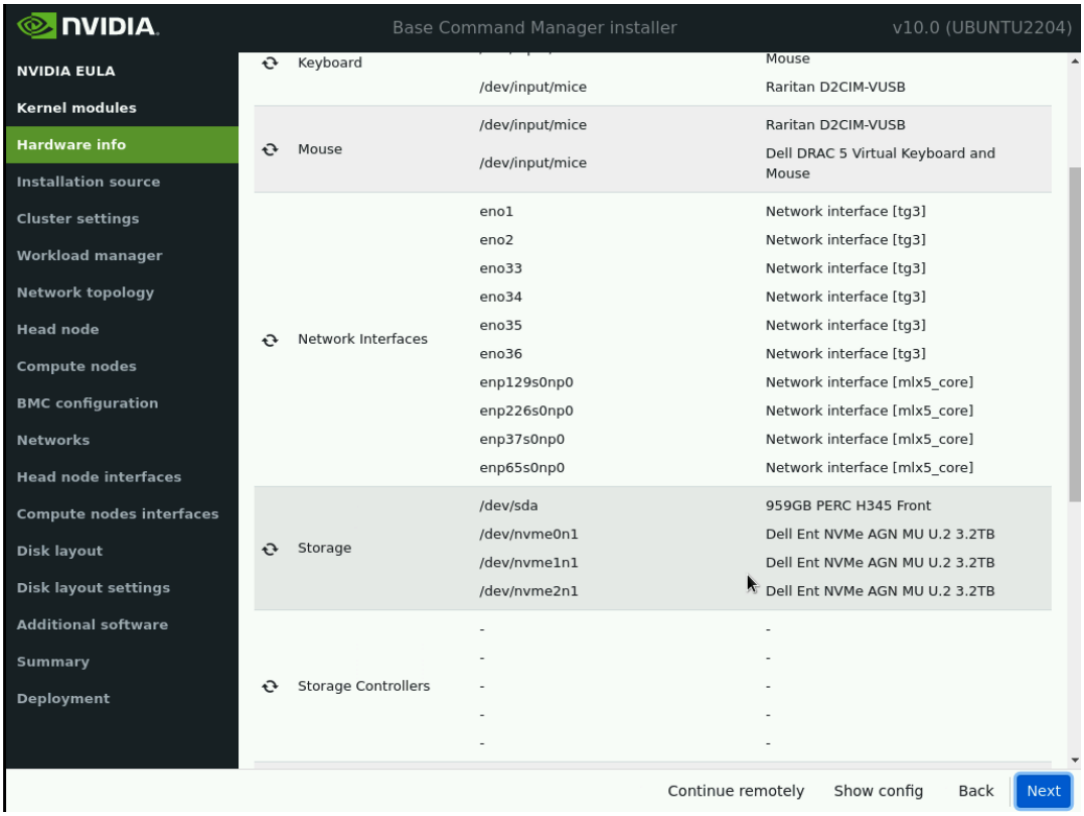
Unless instructed otherwise, select **Next** without modifying the **Kernel modules** to be loaded at boot time.



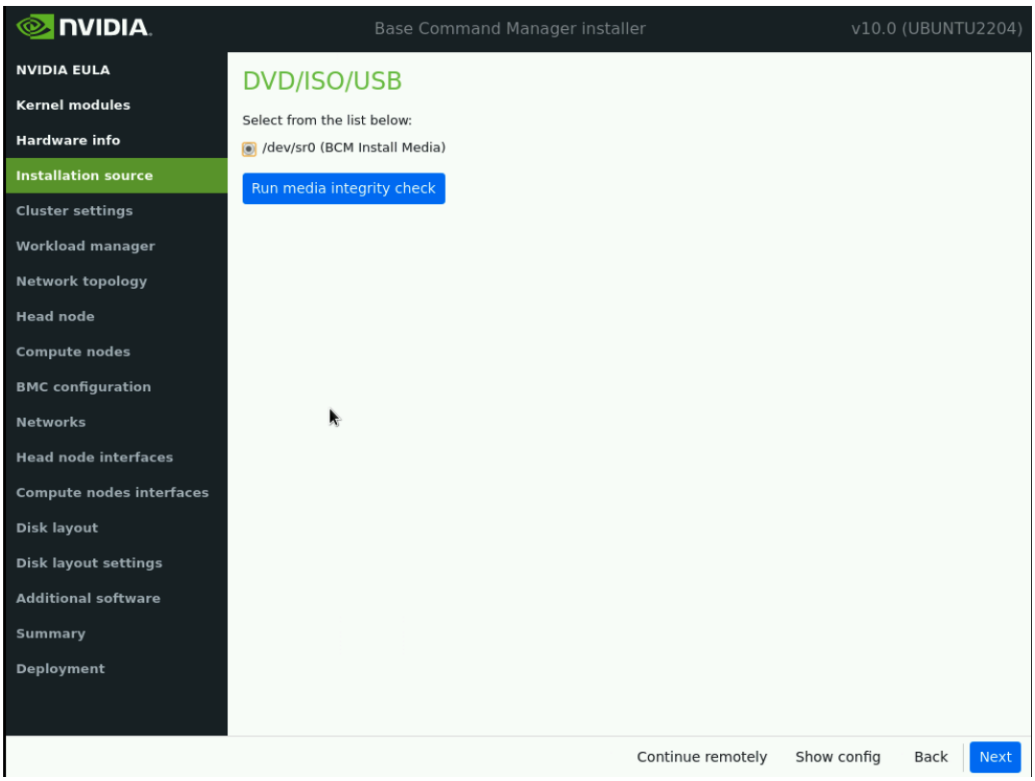
Verify the Primary Headnode **Hardware info** is correct and then select **Next**.

The key components that need to be validated are as follows:

- Network interfaces - Verify that a minimum of 2 Ethernet mode interfaces are detected, this is typically indicated via the device naming convention.
- Devices starting with e = ethernet, i = InfiniBand
- Storage devices - It is advisable to install the operating system on a redundant storage device, such as a hardware or software RAID array.

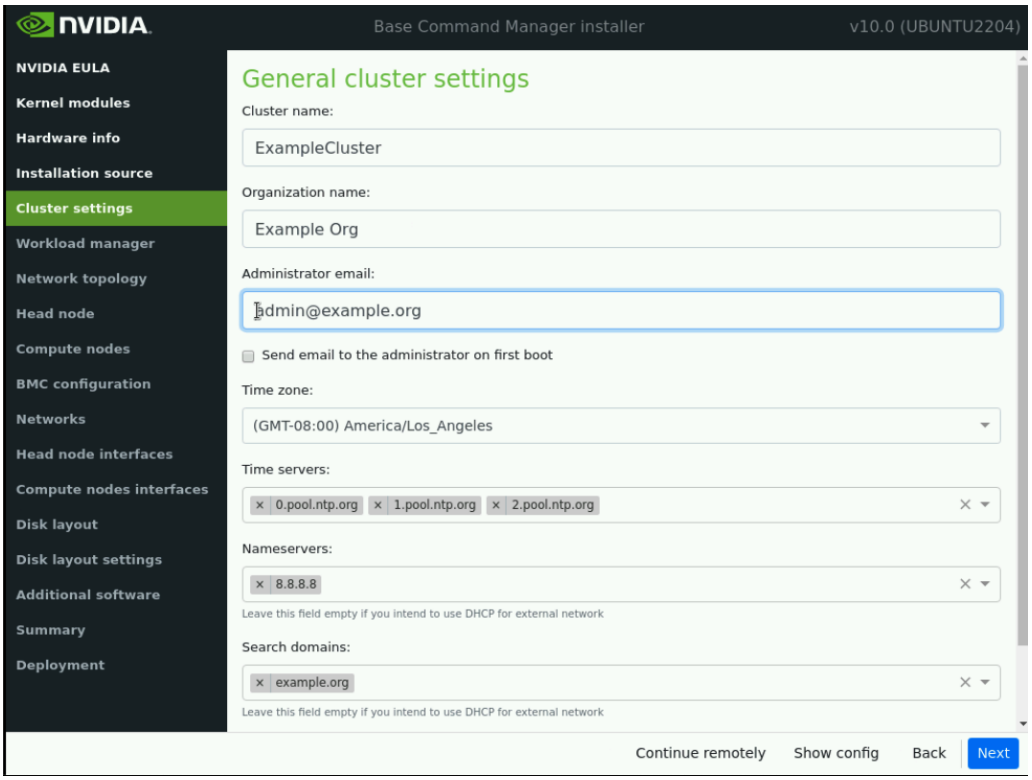


On the **Installation source** screen, choose the appropriate source for the installation media and then select **Next**.

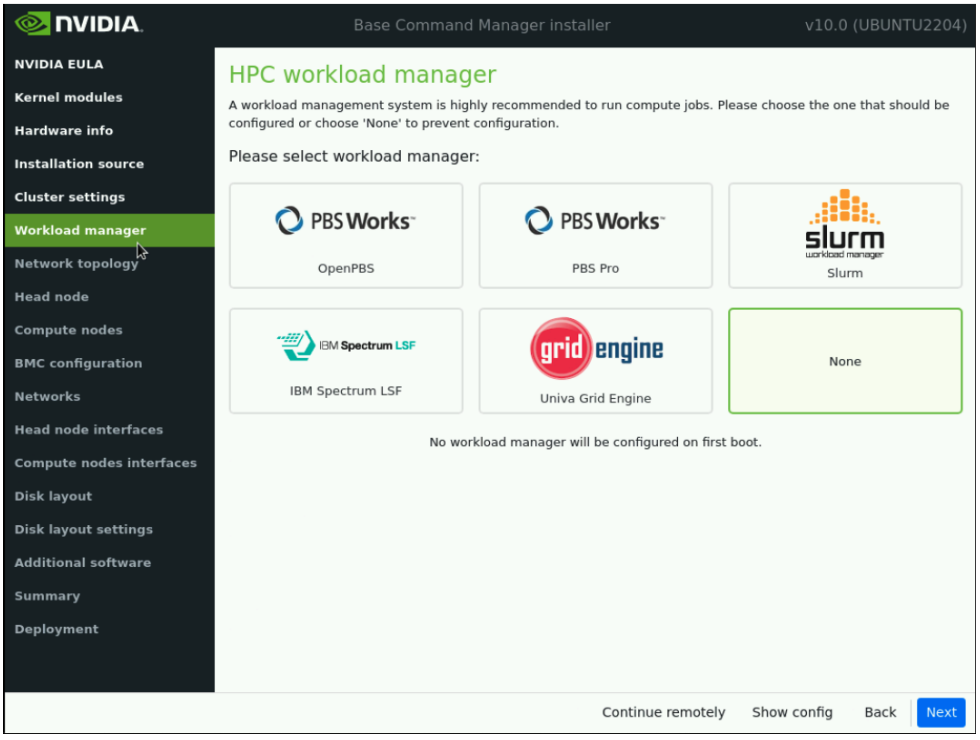


On the **General cluster settings** screen, enter the required information according to the Site Survey

and then select **Next**.



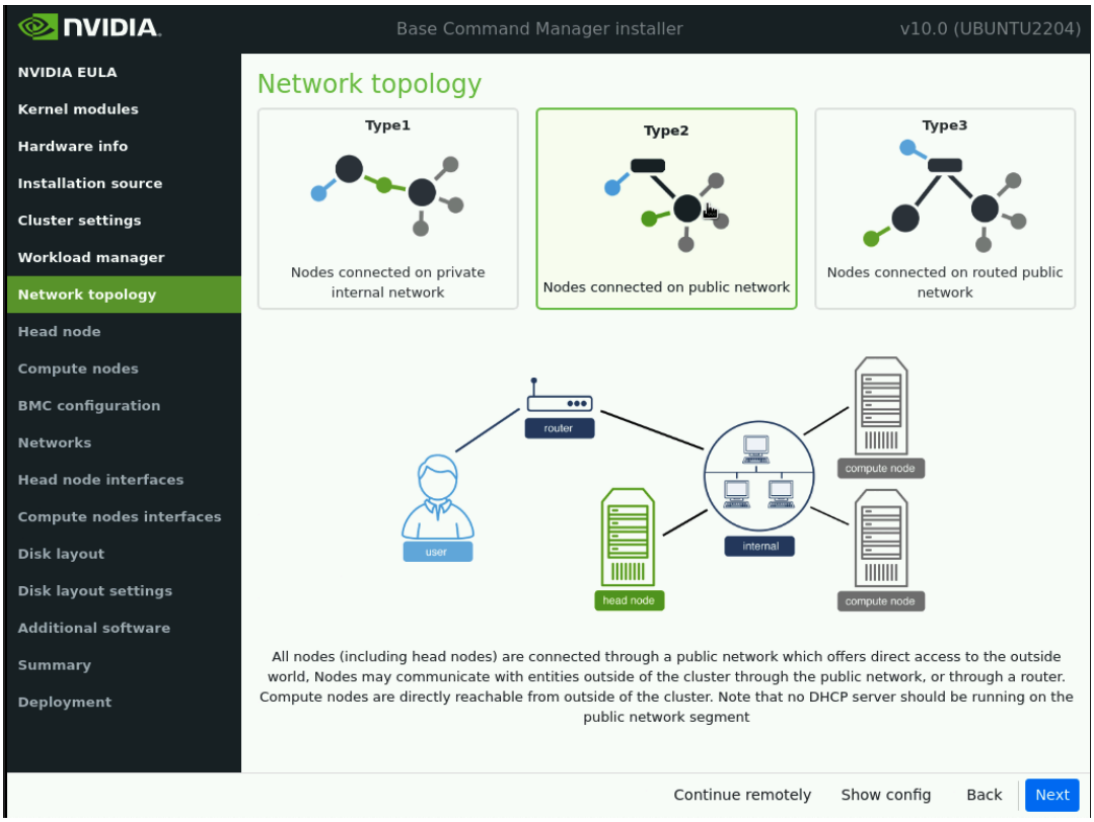
On the **Workload manager** screen, choose **None** and then select **Next**.



On the **Network topology** screen, choose the network type for the data center environment and then select **Next**.

Note

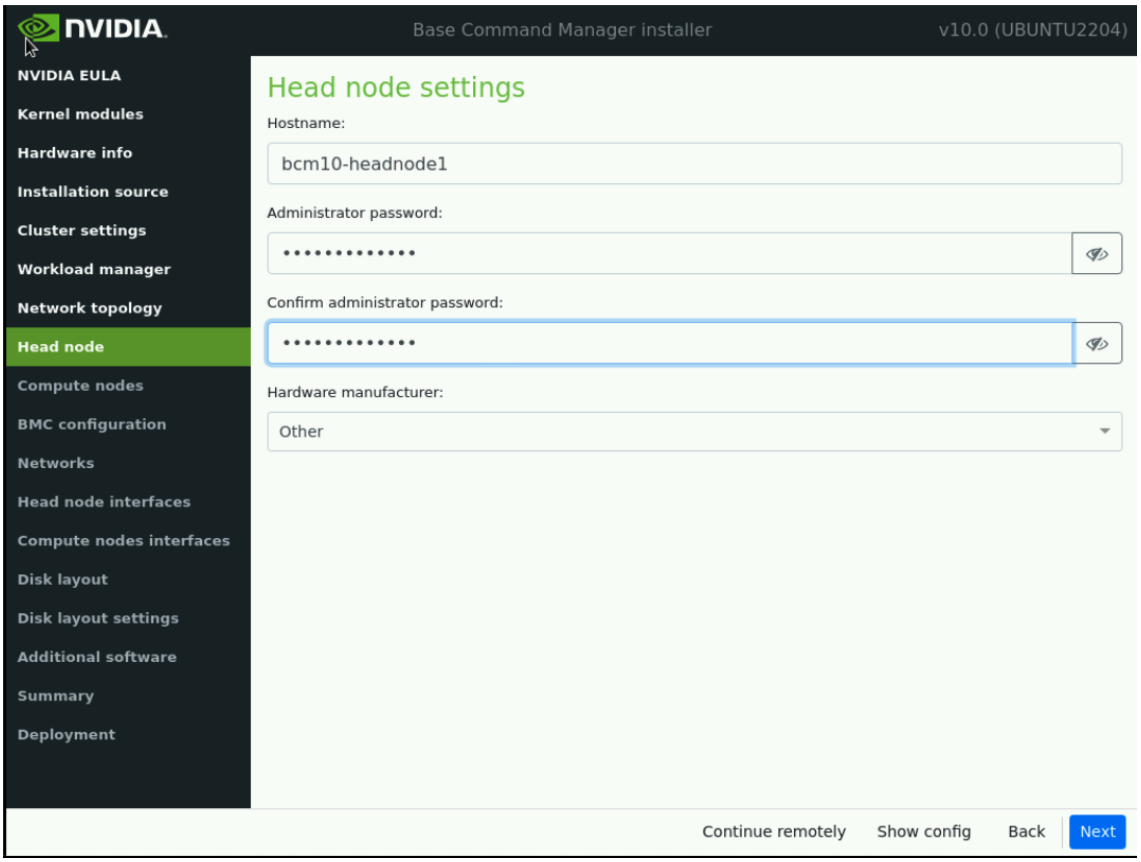
In this deployment example we are using a type 2 network. More information on the different types of networks can be found in the BCM Installation Manual



Update head node settings

On the **Head node** screen enter the Hostname and Administrator password as defined on the **Site Survey**.

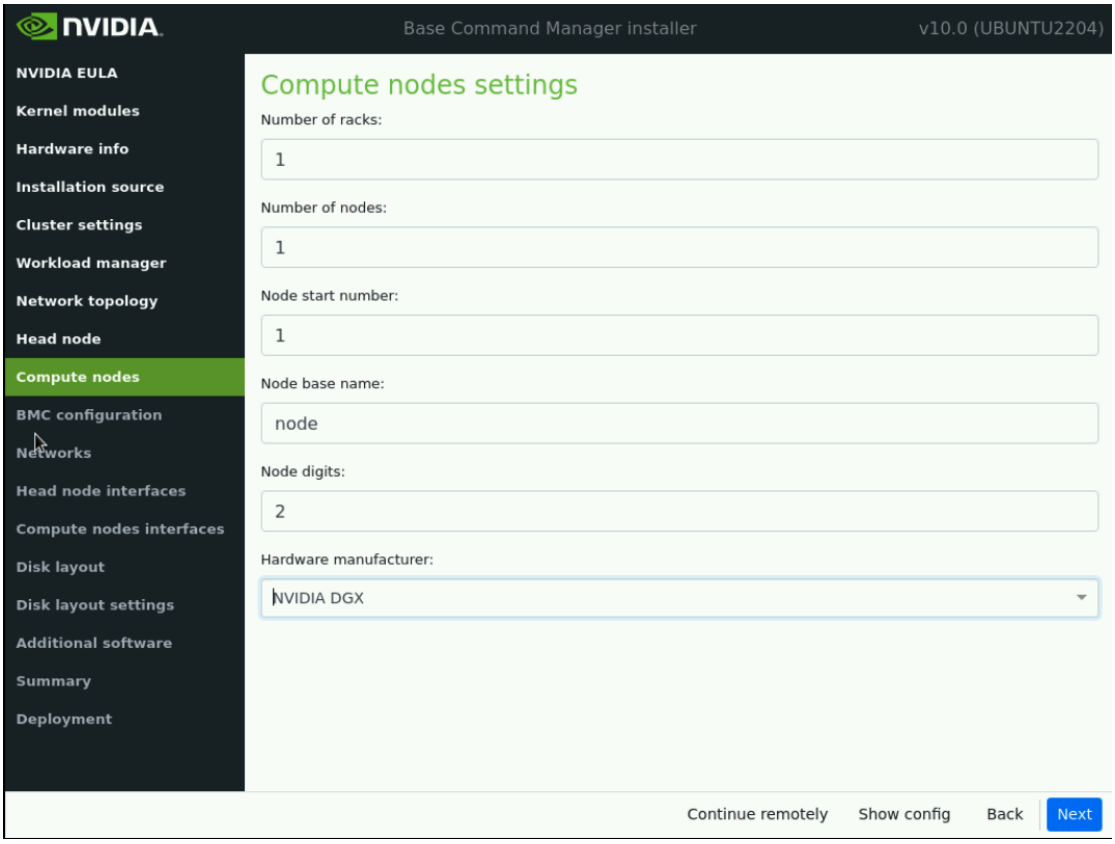
Choose Other for Hardware manufacturer, and then select **Next**.



Update compute node settings

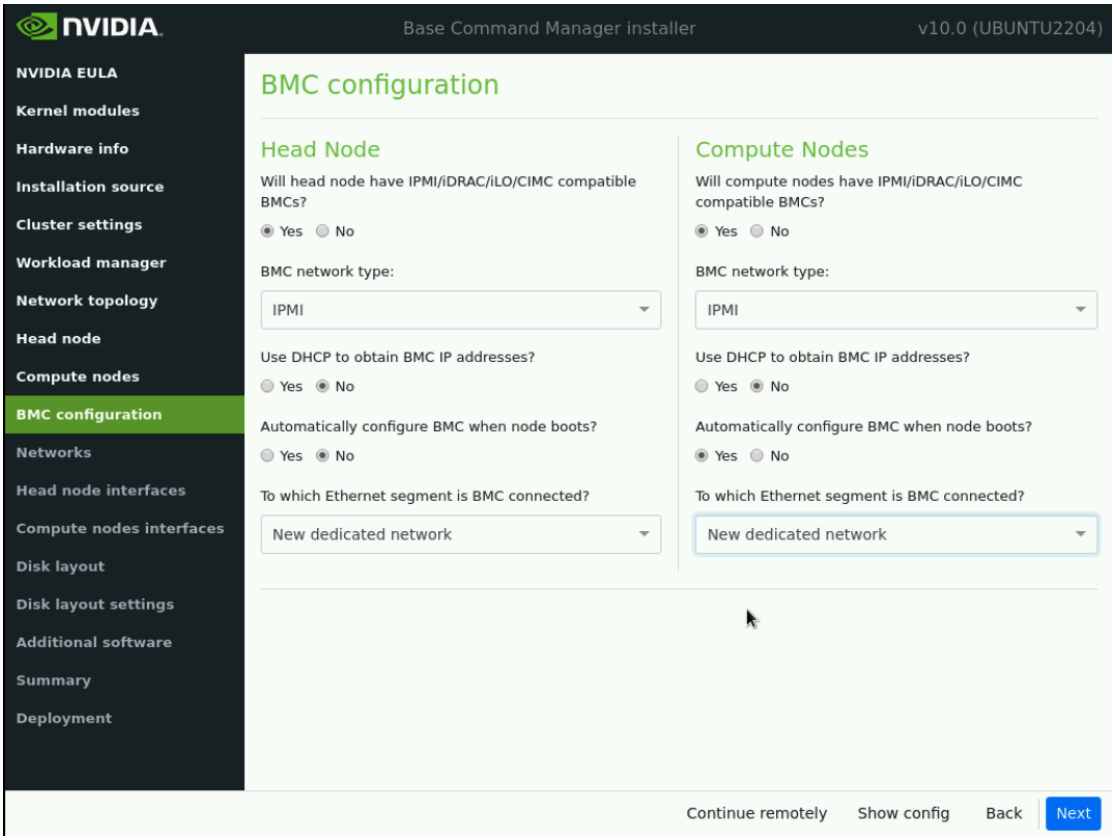
On the **Compute nodes** screen, update node digits from 3 to 2 and select **Next**.

This will populate what will be referred to as a “template” node with the name of node01, which will be modified to create the appropriate DGX and workload management node identities.

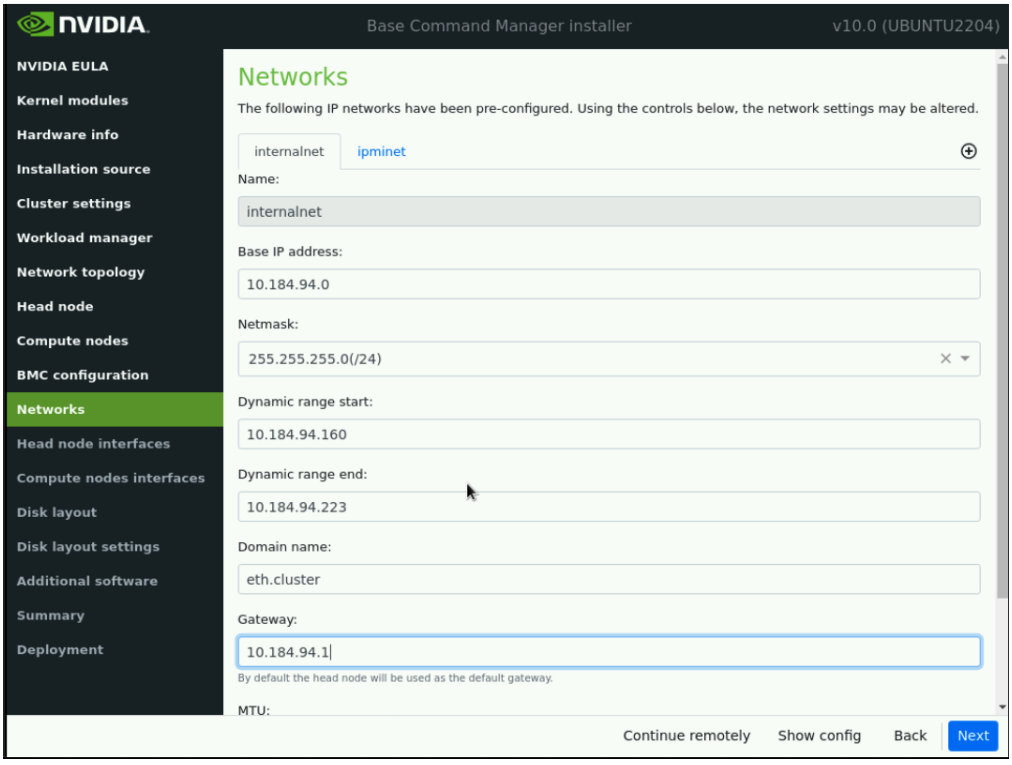


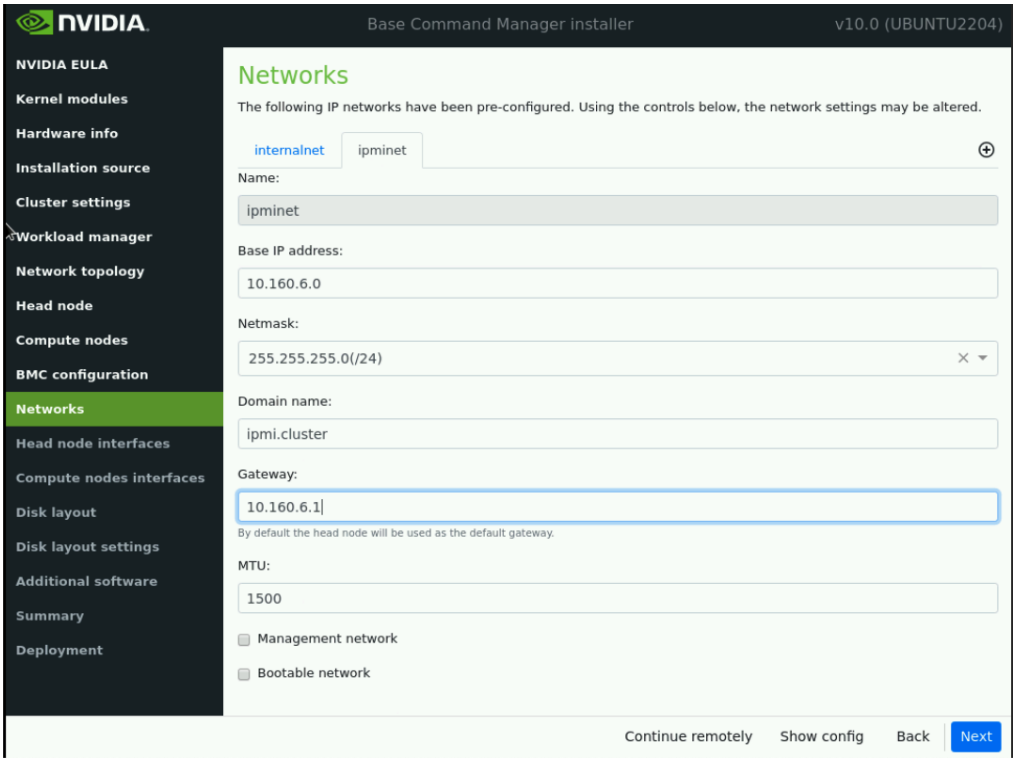
On the BMC Configuration screen, choose Yes for both Head Node and Compute Nodes and populate the following values for both the Head & Compute nodes

1. BMC network type select IPMI from the dropdown menu.
2. Choose No for “Use DHCP to obtain BMC IP addresses?”
3. For the Head node, select No for “Automatically configure BMC when node boots?”. Select Yes for Compute nodes.
4. Lastly select “New dedicated network” from the dropdown list for “To which Ethernet segment is BMC connected?”.



Since a Type 2 network was specified and “New dedicated network” was selected in the prior step for IPMI, there will be a total of 2 networks defined: managementnet (internalnet) & oobmanagementnet (ipminet). Proceed to populate both network definitions with the defined values in the Site Survey.

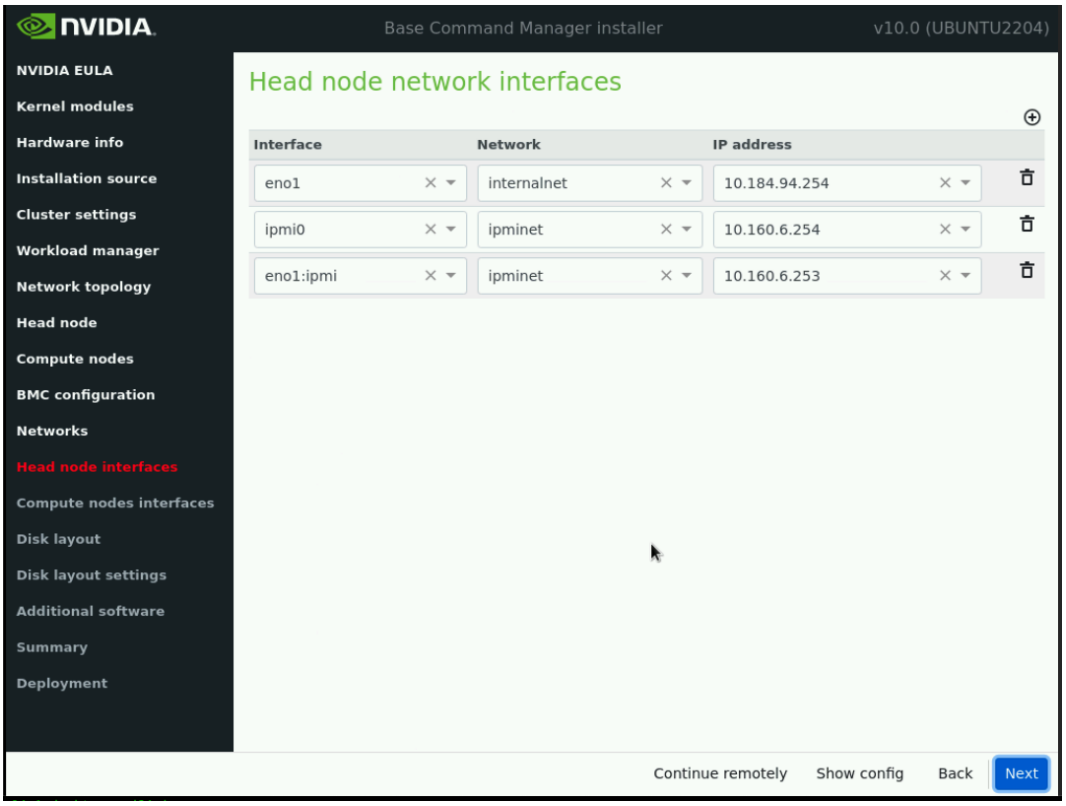




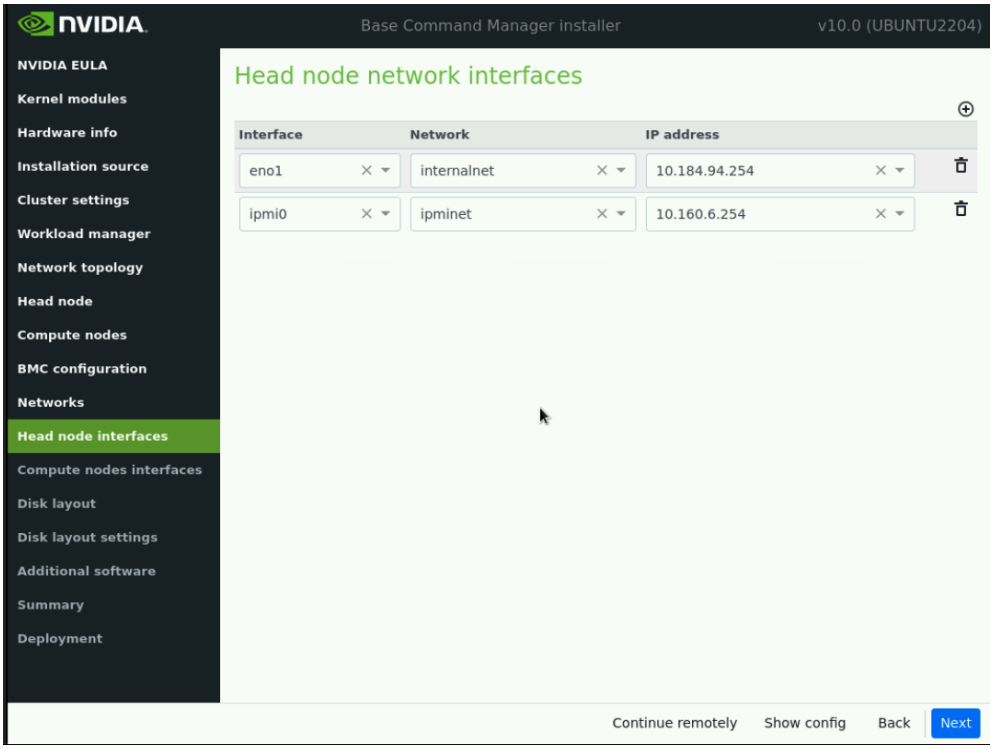
On the Head node interfaces screen, ensure that the correct interface is configured (refer to site survey) with the head node's target managementnet (internalnet) IP.

We will also need to remove the IPMI alias interface that was defined by default, in our example this interface is ens18:ipmi.

In other scenarios you will be looking for an interface name that ends with ":ipmi".



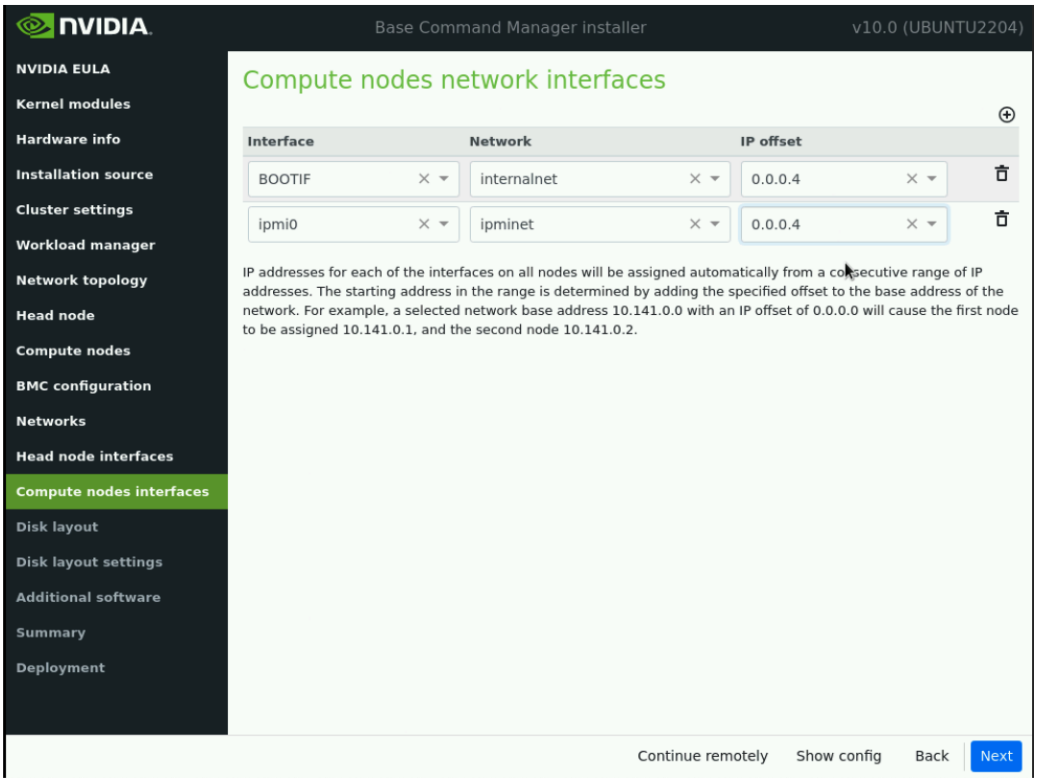
After deleting the alias interface ens18:ipmi.



On the Compute node interfaces screen, update the IP offset for both listed items, and then select Next. Here we are setting the IP offset for any nodes that get provisioned into the cluster later in the deployment process. The offset effectively blocks off the first *n* number of IP's from the specified

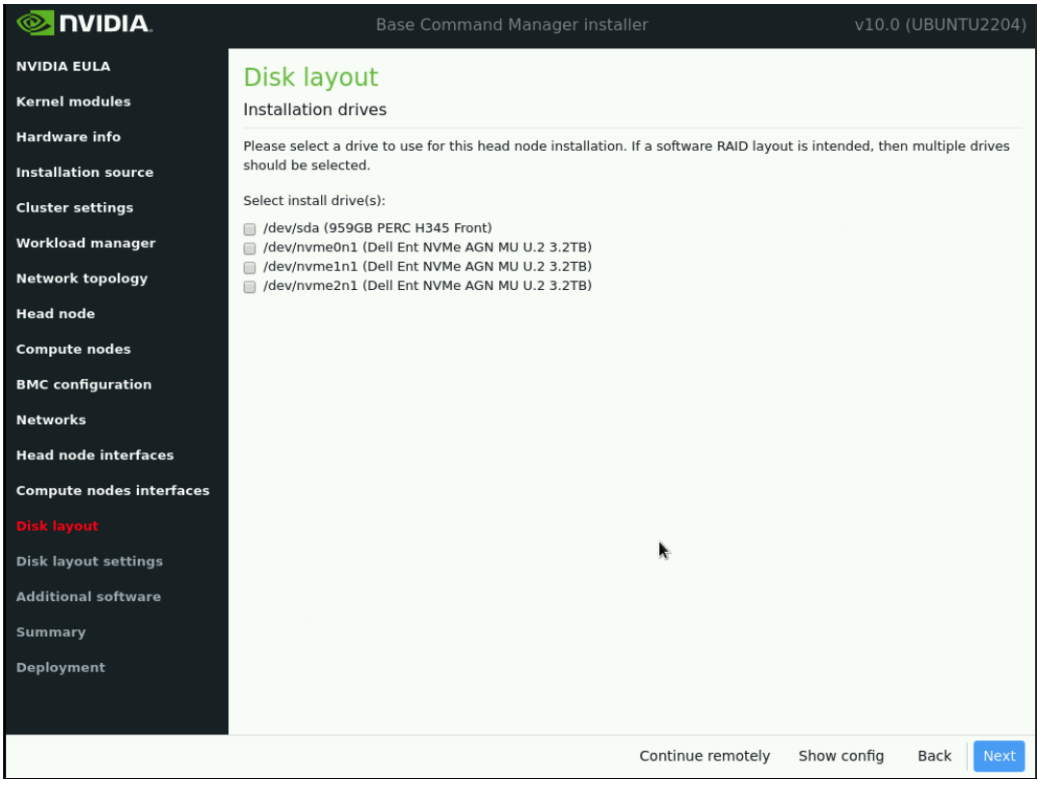
network.

In our example we have it set to 0.0.0.4 for managementnet (internalnet) (10.141.225.0/16) which means the first IP available out of our managementnet (internalnet) ip range will be 10.141.225.4 instead of the expected 10.141.225.1 address. The offset allows the reserved IP addresses to be used for gateways, VRRP etc within the network subnet.

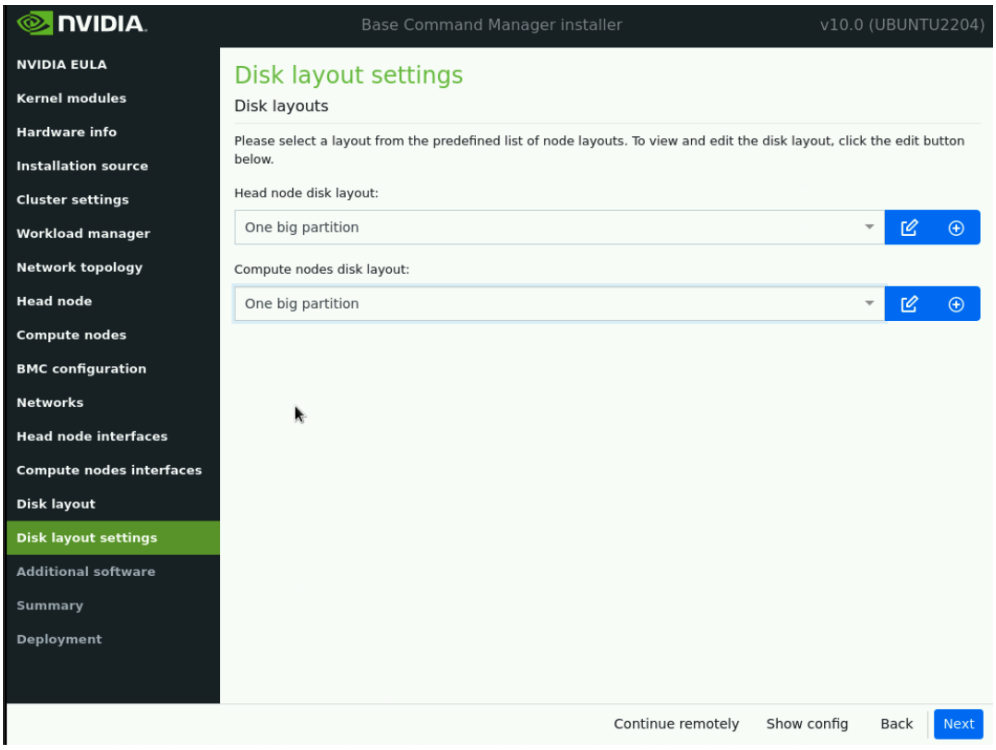


On the Disk layout screen, select the target install location (in this case /dev/sda) and then select Next.

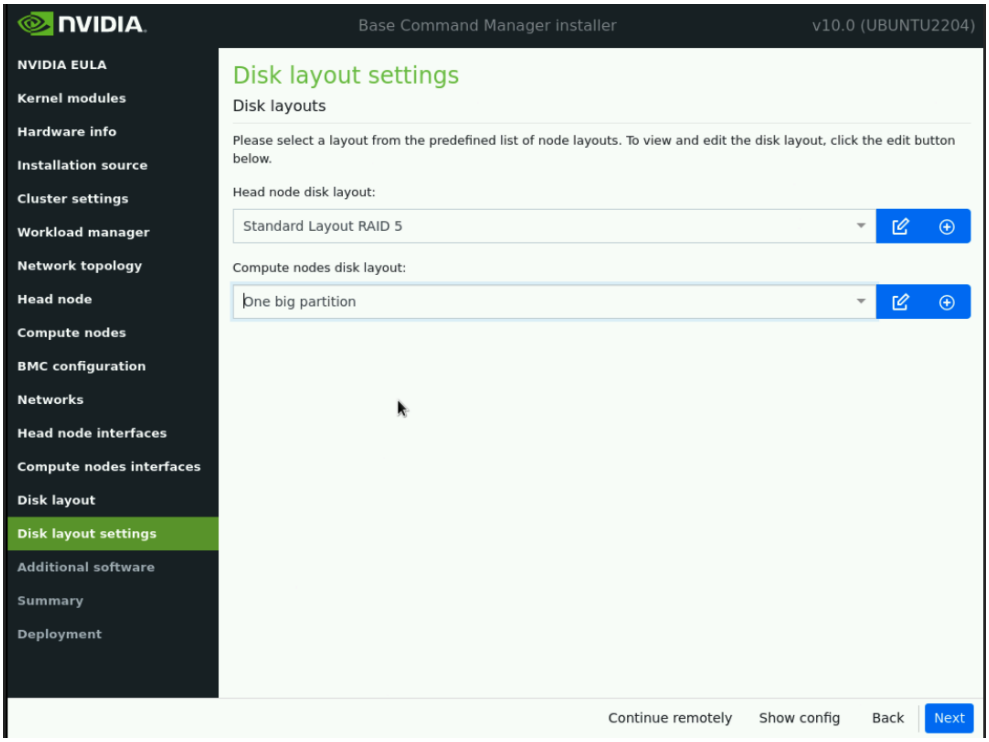
When selecting the target installation location be sure to use a storage device with a minimum of RAID1 redundancy.



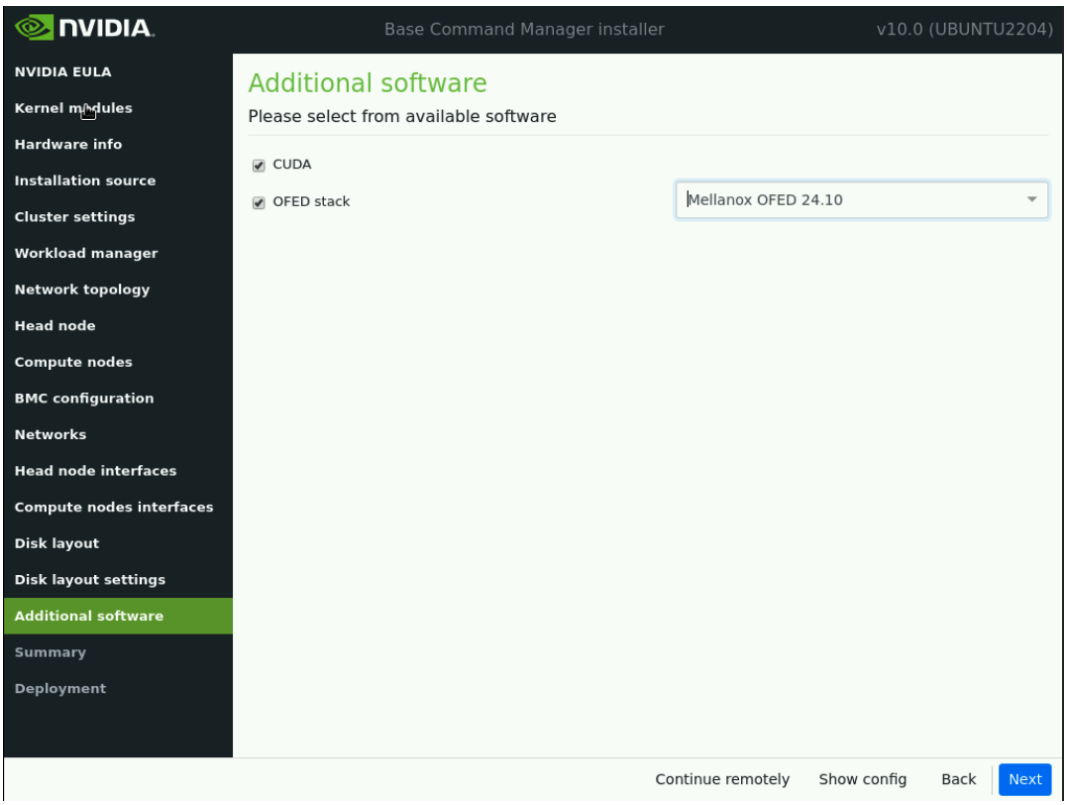
On the Disk layout settings screen, if hardware RAID storage was selected in the prior step accept defaults and then select Next.



If software RAID is the storage option, ensure that “One big partition RAID1” or “One big partition RAID5” is selected for Head node disk layout.



In the Additional software screen, select the newest version of OFED that is compatible with the DGX and select Next.

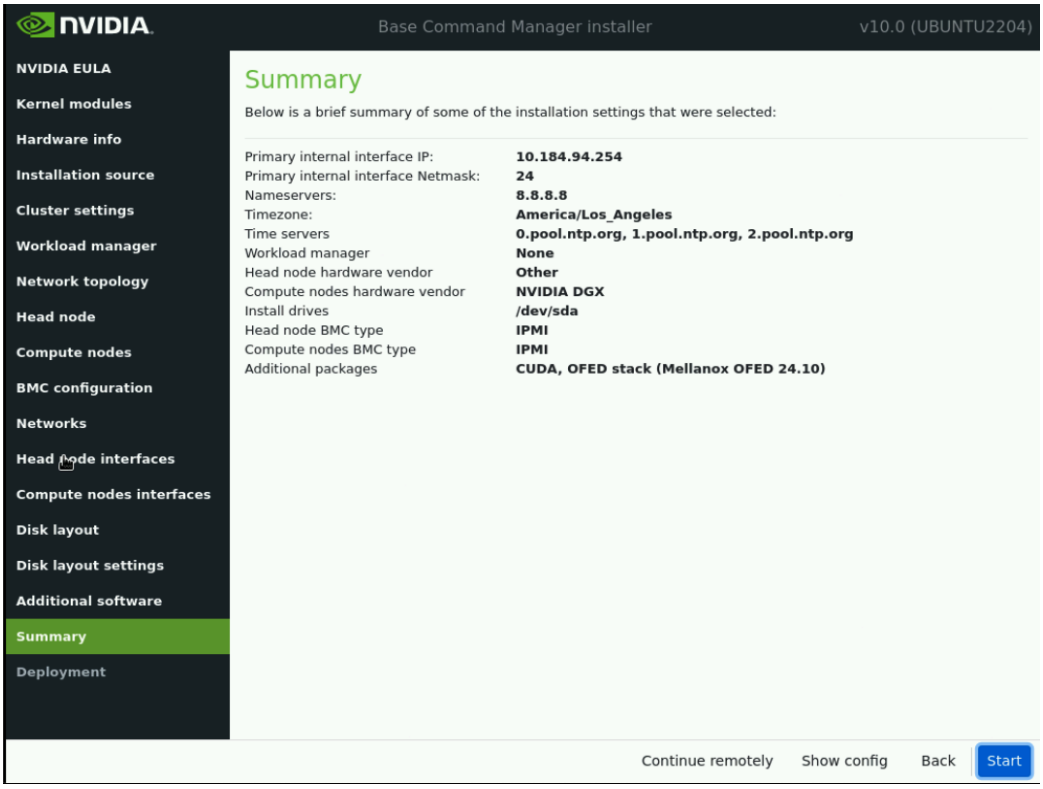


Confirm the information on the Summary screen and then select Next.

The Summary screen provides an opportunity to confirm the Head node/basic cluster configuration

before installation begins.

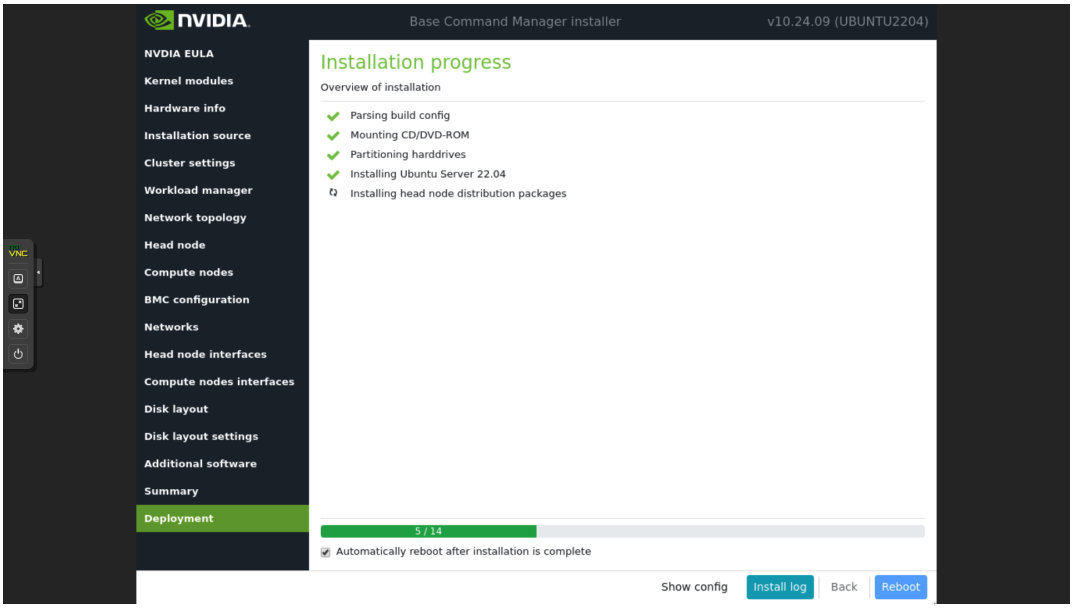
If values do not match site survey, use the back button to navigate to the appropriate screen to correct any errors.



Monitor the progress of the installation, once the deployment is complete, select Reboot.

Note

You can tick the “Automatically reboot after installation is complete” box to have the headnode automatically reboot after the installation completes.



9.6. Login to the Headnode

Once the headnode has finished rebooting, ssh to it using the root credentials.

9.7. Update BCM

Use `apt update` followed by `apt upgrade` to get the latest version of tools/utilities. Reboot the system if prompted.

9.8. Activate the BCM Cluster License

License the cluster by running the request-license command and providing the product key and other pieces of information as per the site survey.

```
#request-license
Product Key (XXXXXX-XXXXXX-XXXXXX-XXXXXX-XXXXXX):
Country Name (2 letter code): US
State or Province Name (full name): California
Locality Name (e.g. city): Santa Clara
Organization Name (e.g. company): NVIDIA
Organizational Unit Name (e.g. department): Demo
Cluster Name: Demo Cluster
Private key data saved to /cm/local/apps/cmd/etc/cluster.key.new

Warning: Permanently added 'bcm10-headnode' (ED25519) to the list of known
 ↪hosts.
MAC Address of primary head node (bcm10-headnode) for ens3f1np1
 ↪[08:C0:EB:F5:72:0F]:
```

(continues on next page)

(continued from previous page)

```
If setting up a second headnode for HA, enter the mac address for it's
 ↪primary in-band interface.
Will this cluster use a high-availability setup with 2 head nodes? [y/N] y
MAC Address of secondary head node for eth0 [XX:XX:XX:XX:XX:XX]:
 ↪5c:6f:69:24:dd:54

Certificate request data saved to /cm/local/apps/cmd/etc/cluster.csr.new
Submit certificate request to http://licensing.brightcomputing.com/licensing/
 ↪index.cgi ? [Y/n] Y

Contacting http://licensing.brightcomputing.com/licensing/index.cgi...

License granted.
License data was saved to /cm/local/apps/cmd/etc/cluster.pem.new
Install license? [Y/n] Y
========= Certificate Information ========
Version:                    10
Edition:                    Advanced
OEM:                        NVIDIA
Common name:                Demo Cluster
Organization:               NVIDIA
Organizational unit:        Demo
Locality:                   Santa Clara
State:                      California
Country:                    US
Serial:                     2369865
Starting date:              04/Oct/2023
Expiration date:            01/Sep/2024
MAC address / Cloud ID:     08:C0:EB:F5:72:0F|5C:6F:69:24:DD:54
Licensed tokens:            8192
Pay-per-use nodes:          Yes
Accounting & Reporting:     Yes
Allow edge sites:           Yes
License type:               Free
==========================================

Is the license information correct ? [Y/n] Y
Backup directory of old license: /var/spool/cmd/backup/certificates/2024-05-
 ↪31_08.25.05
Installed new license
Revoke all existing cmd certificates

Waiting for CMDaemon to stop: OK

Installing admin certificates

Waiting for CMDaemon to start: OK
mysql: [Warning] Using a password on the command line interface can be
 ↪insecure.
Copy cluster certificate to 3 images / node-installers
Copy cluster certificate to /cmimages/default-image//cm/local/apps/cmd/etc/
 ↪cluster.pem
```

(continues on next page)

(continued from previous page)

```
Copy cluster certificate to /cm/node-installer//cm/local/apps/cmd/etc/cluster.
↪pem
Copy cluster certificate to /cmimages/dgx-os-6.3-h100-image//cm/local/apps/
↪cmd/etc/cluster.pem
Copy cluster certificate to /cmimages/dgx-os-6.3-a100-image//cm/local/apps/
↪cmd/etc/cluster.pem
mysql: [Warning] Using a password on the command line interface can be
↪insecure.

Regenerating certificates for users

New license was installed. In order to allow compute nodes to obtain a new
node certificate, all compute nodes must be rebooted.

Please issue the following command to reboot all compute nodes:
     pdsh -g computenode reboot
```

9.9. Enable Bonding on the Headnode

Note

We recommended that you perform this from a remote/physical KVM, not via SSH. Before attempting the following steps on the headnode, verify the headnode's out of band management or BMC interface/remote/physical KVM is reachable and in service.

In the event a mistake is made here, remote access will temporarily be lost to the host OS, and the out of band management or BMC interface or remote console/crash cart would be the only way to rectify the problem.

In this step, we'll clear the managementnet (internalnet) interface IP which was assigned to the primary interface during the installation and assign it to the newly created bond interface with the network interfaces. Refer to site survey for the network interface names/MAC addresses.

Login to headnode and run Cluster Manager Shell (cmsh).

```
root@HEAD-01:~# cmsh
[bcm10-headnode]% device
[bcm10-headnode->device]% use bcm10-headnode1
[bcm10-headnode->device[bcm10-headnode]]% interfaces
[bcm10-headnode1->device[bcm10-headnode1]->interfaces]% list
Type         Network device name  IP              Network          Start if
-------------------------------------------------------------------------
bmc          ipmi0                <Change IP>     ipminet          always
physical     ens3f1np1 [prov]     <Change IP>     internalnet
[bcm10-headnode->device[bcm10-headnode]->interfaces]% clear ens3f1np1 ip
[bcm10-headnode->device*[bcm10-headnode*]->interfaces*]% clear ens3f1np1
↪network
[bcm10-headnode->device*[bcm10-headnode*]->interfaces*]% add physical ens2np0
[bcm10-headnode->device*[bcm10-headnode*]->interfaces*[ens2np0*]]% set mac
```

(continues on next page)

(continued from previous page)

```
↪88:00:00:00:18:d8
[bcm10-headnode->device*[bcm10-headnode*]->interfaces*[ens2np0*]]% add bond
↪bond0
[bcm10-headnode->device*[bcm10-headnode*]->interfaces*[bond0*]]% append
↪interfaces ens3f1np1 ens2np0
[bcm10-headnode->device*[bcm10-headnode*]->interfaces*[bond0*]]% set mode 1
[bcm10-headnode->device*[bcm10-headnode*]->interfaces*[bond0*]]% set network
↪managementnet
[bcm10-headnode->device*[bcm10-headnode*]->interfaces*[bond0*]]% set ip 10.
↪133.4.24
[bcm10-headnode->device*[bcm10-headnode*]->interfaces*[bond0*]]% ..
[bcm10-headnode->device*[bcm10-headnode*]->interfaces*]% ..
[bcm10-headnode->device*[bcm10-headnode*]]% set provisioninginterface bond0
[bcm10-headnode->device*[bcm10-headnode*]]% commit
```

Verify the IP connectivity to the BCM headnode using ping/ssh before proceeding.

Chapter 10. Cluster Bring Up

This section addresses configuration steps to be performed on Base Command Manager headnode1.

10.1. Enable DeviceResolveAnyMAC

The following section enables provisioning of the bonded interfaces on downstream appliances/nodes.

This process enables failover PXE booting for bonded interfaces.

Edit `/cm/local/apps/cmd/etc/cmd.conf` and add the following line

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

Example:

```
nano /cm/local/apps/cmd/etc/cmd.conf
GNU nano 6.2
# Set one or more advanced config parameters, only do this when needed
# AdvancedConfig = { "param=value", "param=value" }
AdvancedConfig = { "DeviceResolveAnyMAC=1" } # modified value
```

Once the above parameter has been saved restart the CMDaemon

```
root@bcm10-headnode:~# systemctl restart cmd
```

10.2. Define Cluster Networks

Next we'll add and configure the additional networks needed for BasePOD.

Refer to Site Survey for the details.

10.2.1. Change network names

First we will change the default network names to align with the names defined in Networking section under Overview.

```
root@bcm10-headnode1:~# cmsh
[bcm10-headnode1]% network
[bcm10-headnode1->network]% list
```

(continues on next page)

(continued from previous page)

```
Name (key) Type Netmask bits Base address Domain name IPv6
------------------------------------------------------------
globalnet Global 0 0.0.0.0 cm.cluster
internalnet Internal 24 10.184.94.0 eth.cluster
ipminet Internal 24 10.160.6.0 ipmi.cluster
[bcm10-headnode1->network]% use internalnet
[bcm10-headnode1->network[internalnet]]% set name managementnet
[bcm10-headnode1->network*[managementnet*]]% ..
[bcm10-headnode1->network*]% use ipminet
[bcm10-headnode1->network*[ipminet]]% set name oobmanagementnet
[bcm10-headnode1->network*[oobmanagementnet*]]% ..
[bcm10-headnode1->network*]% commit
Successfully committed 2 Networks
[bcm10-headnode1->network]% list
Name (key) Type Netmask bits Base address Domain name IPv6
------------------------------------------------------------
globalnet Global 0 0.0.0.0 cm.cluster
managementnet Internal 24 10.184.94.0 eth.cluster
oobmanagementnet Internal 24 10.160.6.0 ipmi.cluster
```

10.2.2. Computenet Config

Starting with computenet, to facilitate gpu to gpu RDMA communication.

```
root@bcm10-headnode1:~# cmsh
[bcm10-headnode1]% network
[bcm10-headnode1]% add computenet
[bcm10-headnode1->network*[computenet*]]% set domainname ib.compute
[bcm10-headnode1->network*[computenet*]]% set baseaddress 100.126.0.0
[bcm10-headnode1->network*[computenet*]]% set netmaskbits 16
[bcm10-headnode1->network*[computenet*]]% commit
```

10.2.3. Storagenet Config

Ethernet (tcp) Storage

BasePOD typically has Ethernet attached Block Storage solutions to the managementnet (internalnet).

In such scenarios it is not necessary to define any additional networks.

10.2.4. IB Storage

In the event IB Storage is attached to the cluster an additional Infiniband network will need to be defined using the following commands.

```
[bcm10-headnode1->network[computenet]]% clone computenet storagenet
[bcm10-headnode1->network*[storagenet*]]% set domainname ib.storage
[bcm10-headnode1->network*[storagenet*]]% set baseaddress 100.127.0.0
[bcm10-headnode1->network*[storagenet*]]% commit
```

Verify using cmsh CLI

```
[bcm10-headnode1]% home;network;list -f
name:20,type:10,netmaskbits:10,baseaddress:15,domainname:20
name (key) type netmaskbit baseaddress domainname
-------------------------------------------------------
computenet Internal 16 100.64.0.0 ib.compute
managementnet Internal 24 10.184.94.0 eth.cluster
oobmanagementnet Internal 24 10.160.6.0 ipmi.cluster
```

10.3. Enable Out-of-band Management of Cluster Nodes

Set the BMC Username and Password for all BCM managed nodes

```
cmsh
[bcm10-headnode1]% partition
[bcm10-headnode1->partition[base]% bmcsettings
[bcm10-headnode1->partition[base]->bmcsettings]% set username bright
[bcm10-headnode1->partition[base]->bmcsettings*]% set password
↪FUNKYpassW0rdGo3sH3r3
[bcm10-headnode1->partition[base]->bmcsettings*]% commit
```

10.4. DGX Node Bringup

10.4.1. Software Image Setup for DGX's

Next we'll create a backup image of the DGX software image on the headnode.

This is a safety step which lets us make changes to the in-use image, and revert back to a factory DGX OS image in the event that something goes wrong.

```
cmsh
[bcm10-headnode]% softwareimage
[bcm10-headnode->softwareimage]% clone dgx-os-6.2-h100-image dgx-os-6.2-h100-
↪image-orig
[bcm10-headnode->softwareimage*[dgx-os-6.2-h100-image-orig*]]% commit
```

10.4.2. DGX Node category setup

Next, we're going to define the DGX node identities in BCM.

All of the DGX nodes in the DGX BasePOD will be named using the following naming convention "dgx-xx", this helps differentiate them from the other nodes.

We'll first start by defining dgx-01's node identity and DGX node category

```
cmsh
[bcm10-headnode]% device
[bcm10-headnode->device]% clone node01 dgx-01
[bcm10-headnode->device*[dgx-01*]]% set category dgx-h100
[bcm10-headnode->device*[dgx-01*]]% commit
```

10.4.3. DGX Interface Definitions

Consult the site survey for the specific interface/IP addresses to assign to DGX nodes.

First we'll define the BMC and managementnet bond interfaces

```
[bcm10-headnode1->device*[dgx-01*]]% interfaces
[bcm10-headnode1->device*[dgx-01*]->interfaces]% set ipmi0 ip 10.160.6.31
[bcm10-headnode1->device*[dgx-01*]->interfaces*]% set ipmi0 network
↪oobmanagementnet
[bcm10-headnode1->device*[dgx-01*]->interfaces*]%
[bcm10-headnode1->device*[dgx-01*]->interfaces*[ipmi0*]]% add physical
↪enp170s0f1np1; add physical enp41s0f1np1
[bcm10-headnode1->device*[dgx-01*]->interfaces*[enp41s0f1np1*]]% add bond
↪bond0 10.133.15.31 managementnet
[bcm10-headnode1->device*[dgx-01*]->interfaces*[bond0*]]% append interfaces
↪enp170s0f1np1 enp41s0f1np1
[bcm10-headnode1->device*[dgx-01*]->interfaces*[bond0*]]% ..
[bcm10-headnode1->device*[dgx-01*]->interfaces*]% remove bootif
[bcm10-headnode1->device*[dgx-01*]->interfaces*]% ..
[bcm10-headnode1->device*[dgx-01*]]% set provisioninginterface bond0
[bcm10-headnode1->device*[dgx-01*]]% commit
```

Now add the ib interface definitions for the compute fabric

```
[bcm10-headnode->device*[dgx-01*]->interfaces[bond0]]% add physica ibp220s0
↪100.126.0.31 computenet
[bcm10-headnode->device*[dgx-01*]->interfaces*[ibp154s0*]]% foreach -o
↪ibp220s0 ibp154s0 ibp206s0 ibp192s0 ibp24s0 ibp64s0 ibp79s0 ibp94s0 ()
[bcm10-headnode->device*[dgx-01*]->interfaces*]% set ibp154s0 ip 100.126.1.31
[bcm10-headnode->device*[dgx-01*]->interfaces*]% set ibp206s0 ip 100.126.2.31
[bcm10-headnode->device*[dgx-01*]->interfaces*]% set ibp192s0 ip 100.126.3.31
[bcm10-headnode->device*[dgx-01*]->interfaces*]% set ibp79s0 ip 100.126.4.31
[bcm10-headnode->device*[dgx-01*]->interfaces*]% set ibp64s0 ip 100.126.5.31
[bcm10-headnode->device*[dgx-01*]->interfaces*]% set ibp94s0 ip 100.126.6.31
[bcm10-headnode->device*[dgx-01*]->interfaces*]% set ibp24s0 ip 100.126.7.31
[bcm10-headnode->device*[dgx-01*]->interfaces*]% commit
```

10.4.4. Defining the storage fabric:

10.4.5. Ethernet (tcp)

For Ethernet attached block storage solutions no additional network interface definitions are necessary as the interface used will be the bond0 bonded interface.

10.4.6. Infiniband (o2ib)

For Infiniband attached block storage solutions run the following commands to define the 2 additional storagenet interfaces for the DGX appliance.

```
[bcm10-headnode->device[dgx-01]->interfaces]% add physical ibp170s0f0 100.127.
↪0.31 storagenet
[bcm10-headnode->device*[dgx-01*]->interfaces*[ibp170s0f0*]]% add physical
↪ibp41s0f0 100.127.1.31 storagenet
[bcm10-headnode->device*[dgx-01*]->interfaces*[ibp41s0f0*]]% commit
[bcm10-headnode->device[dgx-01]->interfaces[ibp41s0f0]]% exit
```

10.4.7. Define DGX-01's MAC address

Here we are going to assign MAC addresses to the two managementnet (internalnet) attached interfaces that belong to bond0.

When assigning MAC address there is no specific order/requirement of which MAC goes to which enumerated interface name, so long as both MAC addresses are recorded for the appropriate DGX node identity

Refer to site survey for interface MAC details.

```
[bcm10-headnode->device[dgx-01]->interfaces]% set enp170s0f1np1 mac
↪94:6D:00:00:00:FB
[bcm10-headnode->device*[dgx-01*]->interfaces*]% set enp41s0f1np1 mac
↪94:00:00:00:74:0B
[bcm10-headnode->device*[dgx-01*]->interfaces*]% exit
[bcm10-headnode->device*[dgx-01*]]% set mac 94:6D:00:00:00:FB
[bcm10-headnode->device*[dgx-01*]]% commit
```

10.4.8. Test Provisioning of DGX-01

DGX-01 is now ready to be provisioned. It can be powered on using the physical power button, by using the BMC, or via IPMI tool command from the Headnode.

```
[bcm10-headnode->device[dgx-01]]% power on
ipmi0 ..................... [ ON ] dgx-01
```

OR

```
root@HEAD-01:~# module load ipmitool
ipmitool -I lanplus -U <BMC User> -P <pass> -H 10.160.6.31 power on
```

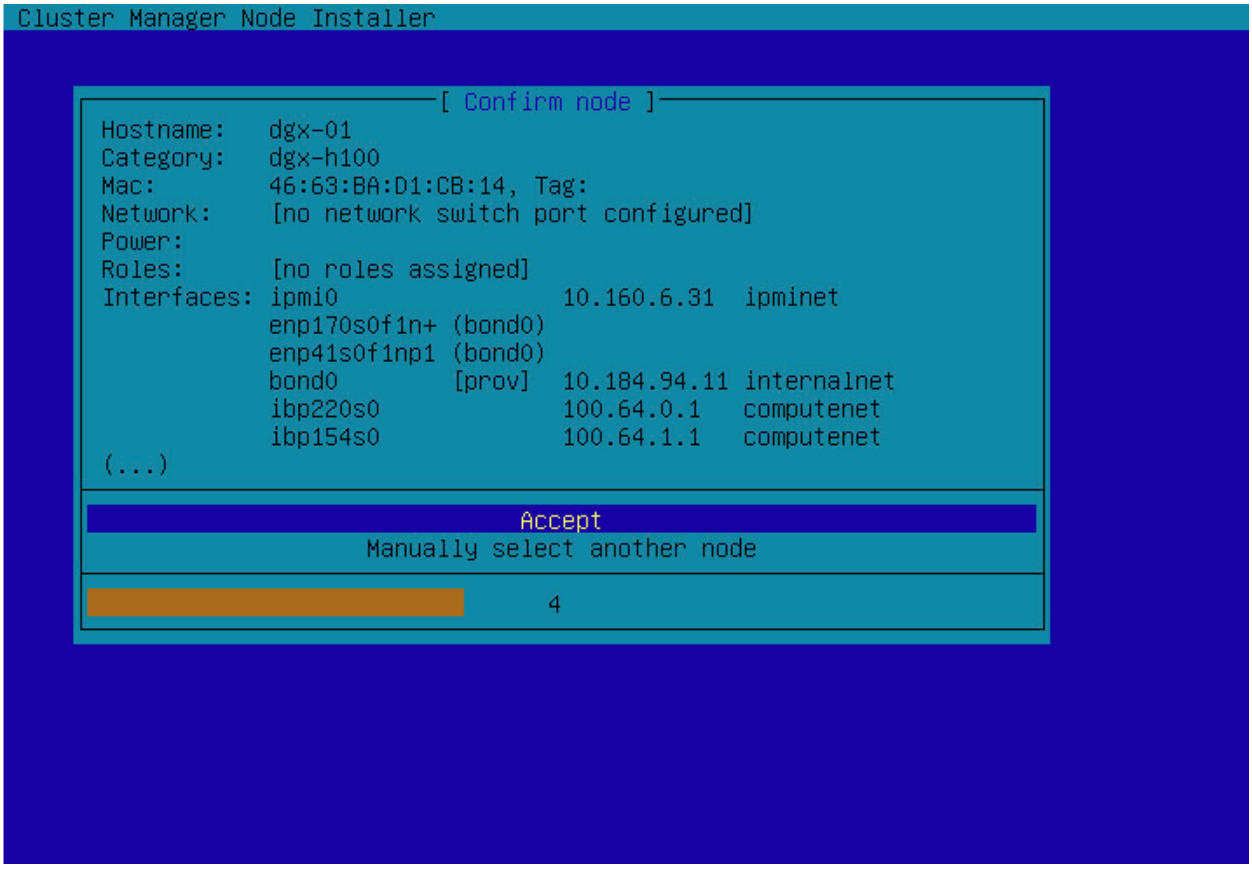
This DGX bootup process will take several minutes for it to go through POST. You can monitor the progress from a KVM or via BMC Virtual Console.

If DGX-01's boot options were properly configured in the BIOS (i.e PXE boot as the first boot option with the proper interface) the node should proceed to attempt PXE booting.

The DGX will load an installer environment to help facilitate the provisioning process and finally load into the Cluster Manager Node Installer environment.

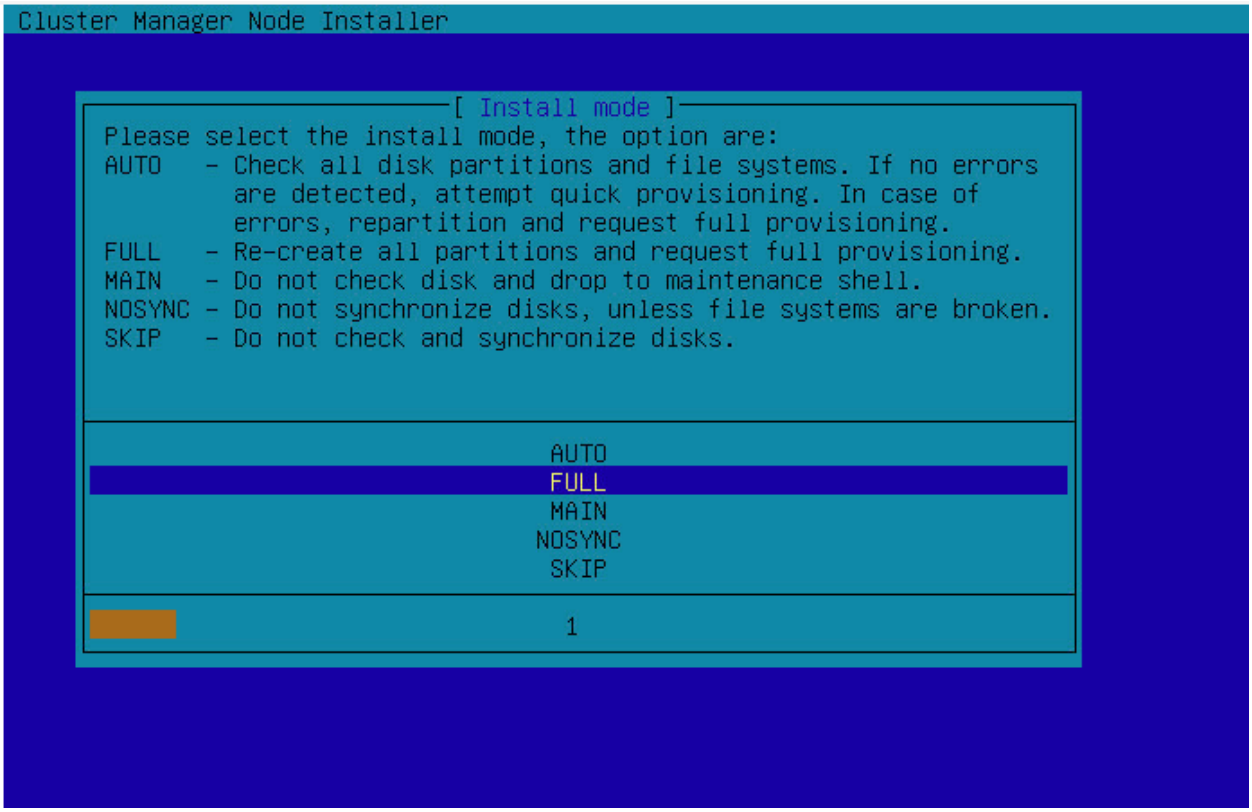
In the event that the DGX successfully identifies itself, you will see the following automated “Confirm node” prompt. The timer will expire and then proceed to provision the DGX with the displayed identity.

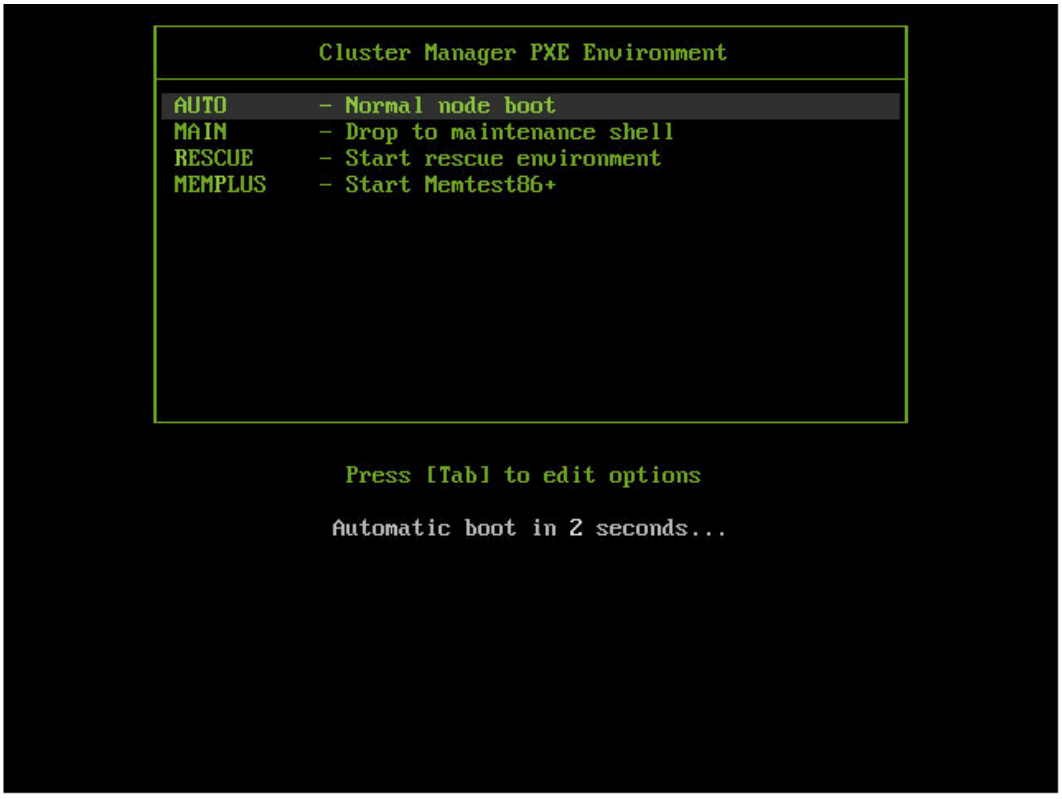
If required, validate the DGX hostname/category/MAC/network IPs with the site survey.



The next screen shows when a DGX appliance has successfully PXE booted.

This state is fully automated and no user intervention is required here.





From here the DGX will proceed to provision itself via the served identity from the BCM headnode.

As the DGX progresses through the PXE boot provisioning process, you can monitor the progress from the headnode via cmsh.

Once DGX-01 successfully shows a status of UP we can proceed to clone this node identity for the remaining needed DGX's, in this example we are adding 3 additional nodes for a total of 4.

```
[bcm10-headnode->device[dgx-01]]% ls
Type Hostname (key) MAC Category Ip Network Status
--------------------------------------------------------
HeadNode bcm10-headnode BC:00:00:00:43:45 10.133.11.51 managementnet [UP]
PhysicalNode dgx-01 94:00:00:00:91:FB dgx-h100 10.133.15.31 dgxnet [UP]
PhysicalNode node001 00:00:00:00:00:00 default 10.133.11.1 managementnet [
↪DOWN ]
[bcm10-headnode->device[dgx-01]]% foreach -o dgx-01 -n dgx-02..dgx-04 () --
↪next-ip
[bcm10-headnode->device*]% commit
Successfully committed 3 Devices
```

Note

In the event the provisioning attempt fails or encounters problems refer to /var/log/messages and /var/log/node-installer log files to further diagnose the provisioning issue.

Set the MAC addresses for each of the new nodes. Repeat the steps below for each new DGX node, refer to the site survey for the details.

```
[bcm10-headnode->device]% use dgx-02; interfaces
[bcm10-headnode->device[dgx-02]->interfaces]% set enp170s0f1np1 mac
↪94:6D:00:00:00:FD
[bcm10-headnode->device*[dgx-02*]->interfaces*]% set enp41s0f1np1 mac
↪94:6D:00:00:00:FE
[bcm10-headnode->device*[dgx-02*]->interfaces*]% exit
[bcm10-headnode->device*[dgx-02*]]% set mac 94:6D:00:00:00:FD
[bcm10-headnode->device*[dgx-02*]]% commit
```

Proceed to power on and provision the remaining DGX nodes into the BCM Cluster.

You can verify the provisioning progress/status using cmsh

```
[bcm10-headnode1]% device;list
Type Hostname (key) MAC Category IP Network Status
--------------------------------------------------------
HeadNode bcm10-headnode1 84:16:0C:AD:DA:DE 10.184.94.254 managementnet [UP ],
↪health check unknow+
PhysicalNode dgx-01 94:6D:AE:AA:13:C9 dgx-h100 10.184.94.11 managementnet [
↪UP ], health check failed+
PhysicalNode dgx-02 A0:88:C2:A3:44:E5 dgx-h100 10.184.94.12 managementnet [
↪UP ], health check unknown
PhysicalNode dgx-03 94:6D:AE:1C:80:CD dgx-h100 10.184.94.13 managementnet
↪[INSTALLING] (provis+
PhysicalNode dgx-04 A0:88:C2:04:70:A1 dgx-h100 10.184.94.14 managementnet [
↪UP ], health check unknown
```

Chapter 11. BCM HA

We will be configuring BCM head node high availability next by provisioning the second BCM head-node

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

```
% device
% power -c dgx-h100 status
[-head1->device]% power -c dgx-h100 status
ipmi0 ..................... [ ON ] bcm-dgx-h100-01
ipmi0 ..................... [ ON ] bcm-dgx-h100-02
ipmi0 ..................... [ ON ] bcm-dgx-h100-03
ipmi0 ..................... [ ON ] bcm-dgx-h100-04
[bcm-head-01->device]%
```

Power off the cluster nodes.

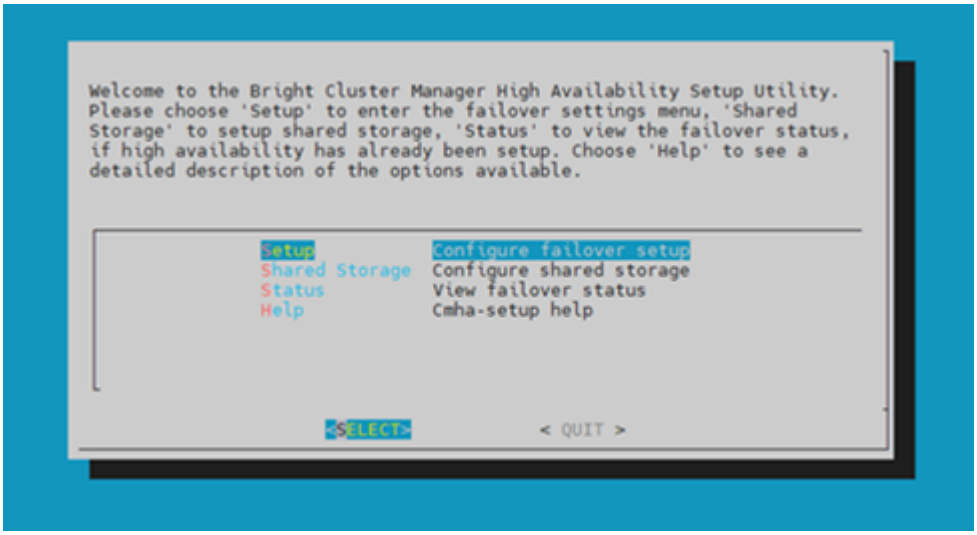
The cluster nodes must be powered off before configuring HA.

```
% power -c dgx-h100 off
ipmi0 ..................... [ OFF ] bcm-dgx-h100-01
ipmi0 ..................... [ OFF ] bcm-dgx-h100-02
ipmi0 ..................... [ OFF ] bcm-dgx-h100-03
ipmi0 ..................... [ OFF ] bcm-dgx-h100-04
```

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

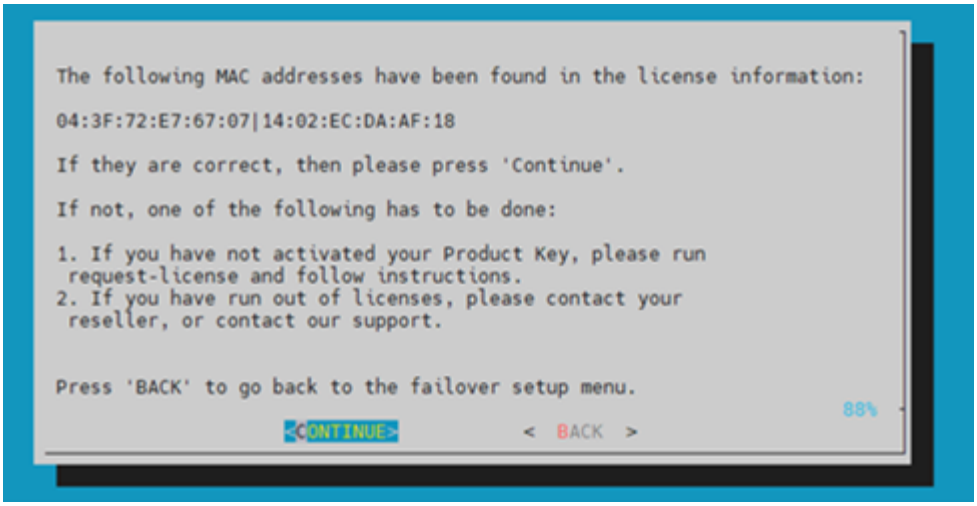
```
#cmha-setup
```

Choose Setup and then select SELECT.



Choose Configure and then select NEXT.

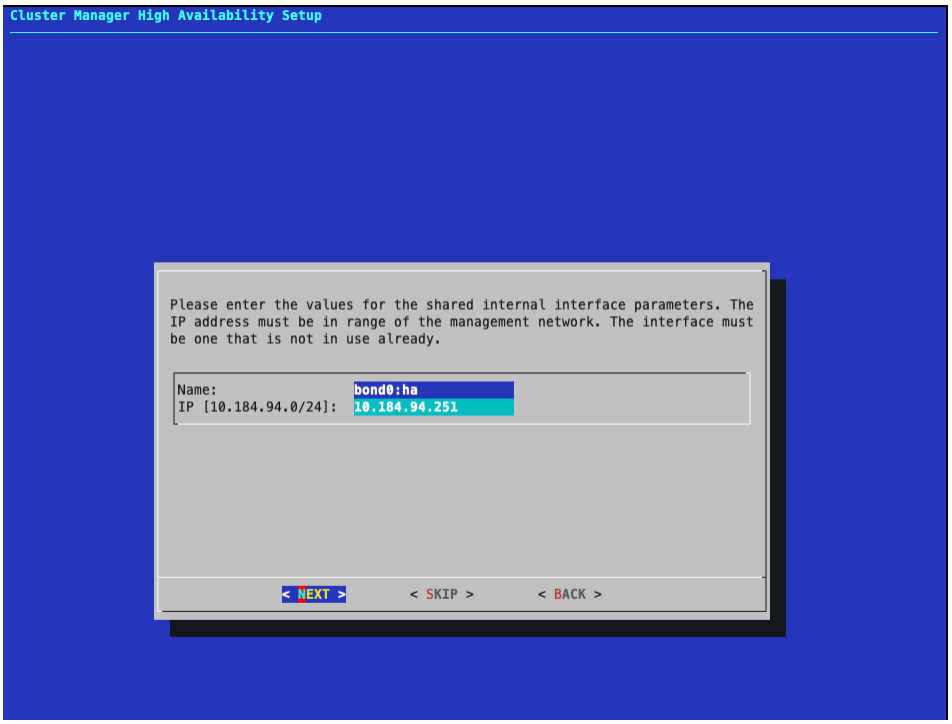
Verify that the cluster license information/MAC found in cmha-setup screen is correct and then select CONTINUE.



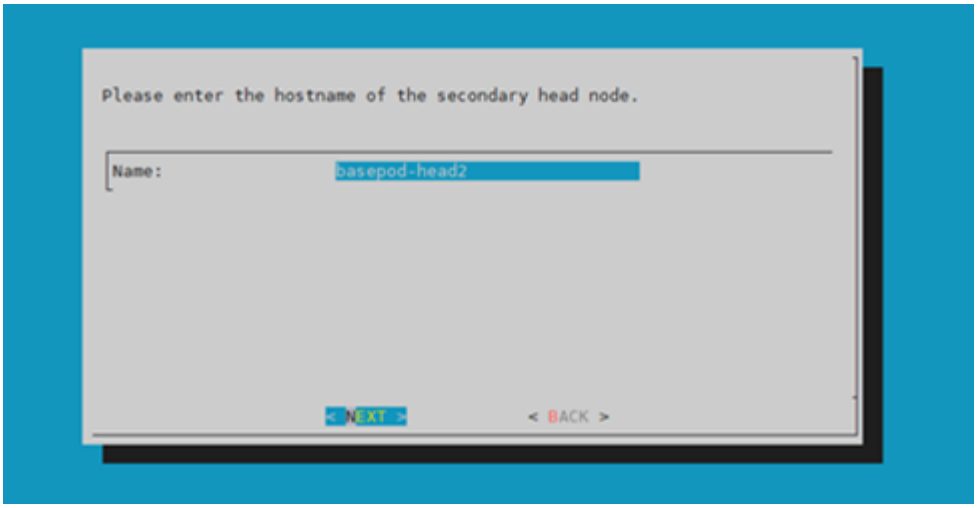
Configure an external Virtual IP address (obtained from Site Survey) that will be used by the active head node in the HA configuration and then select NEXT.

Note

This will be the IP that should always be used for accessing the active head nodes once HA configuration is complete.



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



Populate the failover network details as prescribed in the Site Survey

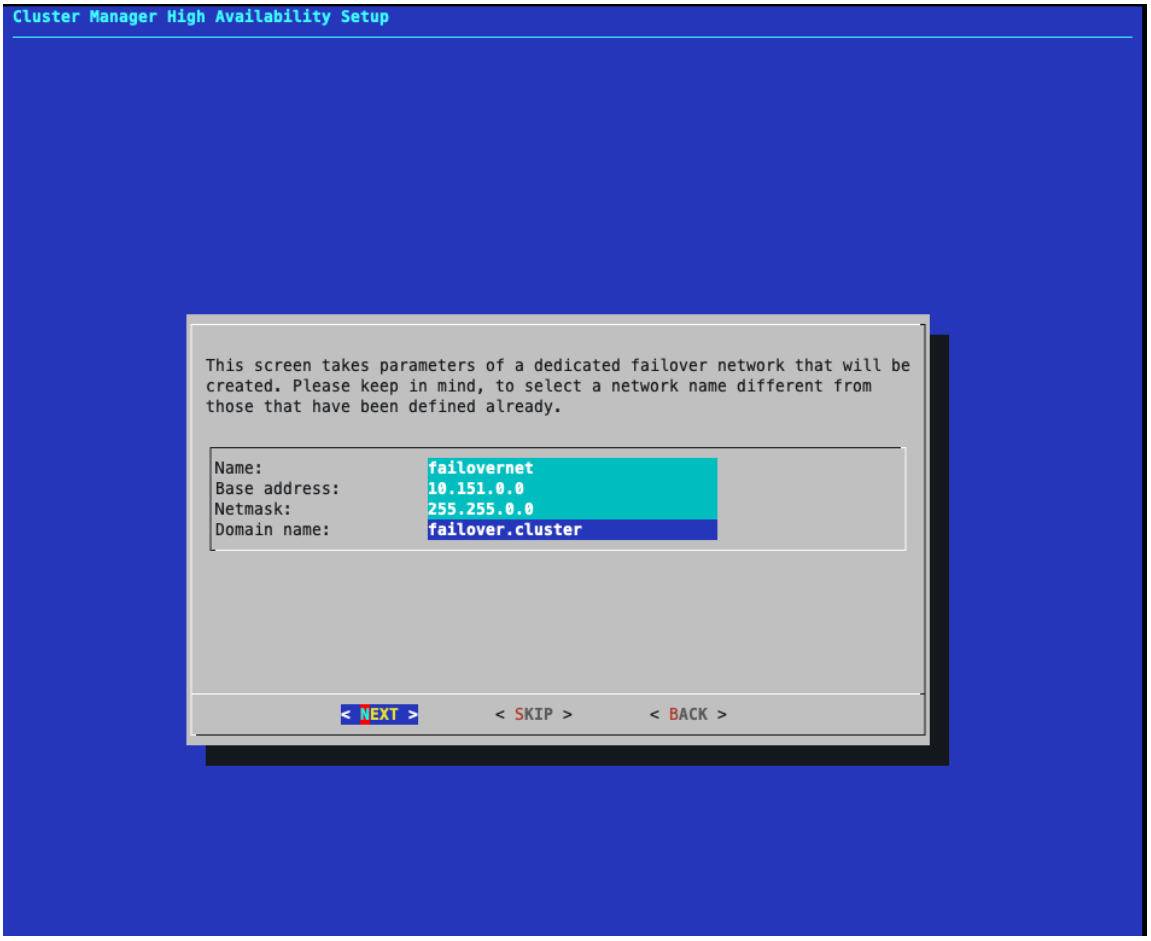
There are two options for the failover network:

1. Using dedicated interfaces on the BCM head nodes or
2. Utilizing the existing managementnet (internalnet).

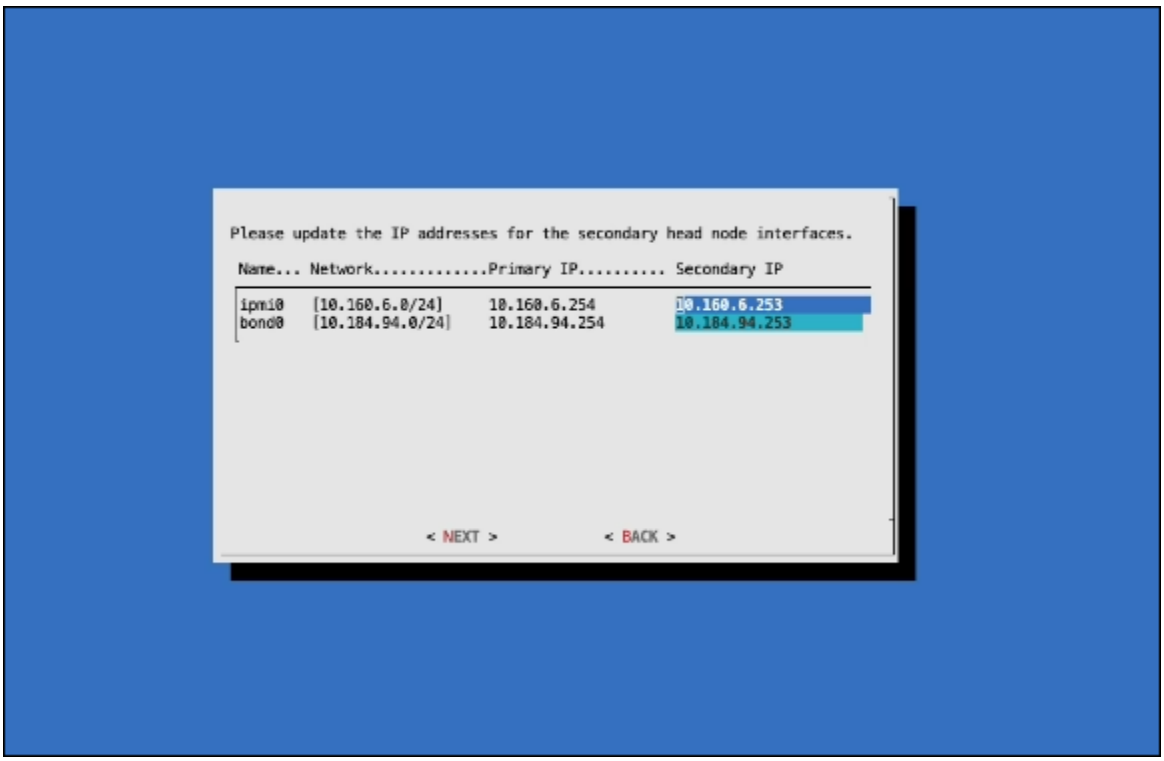
If you are using a dedicated network interface, select the chosen failover interface and provide the IP information for the dedicated network.

In this example, we are using the internal network as the failover network by skipping dedicated failover interface configuration.

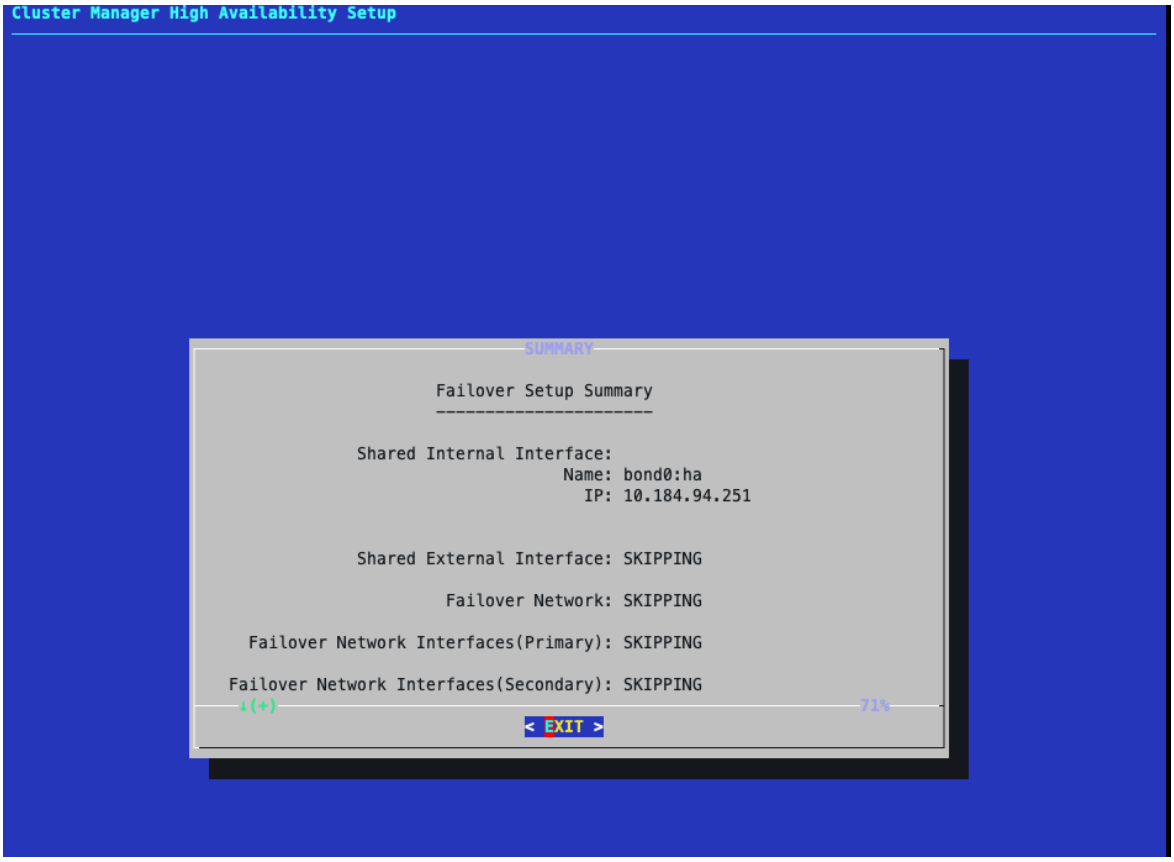
Refer to the BCM admin guide for BCM HA failover network configuration options.



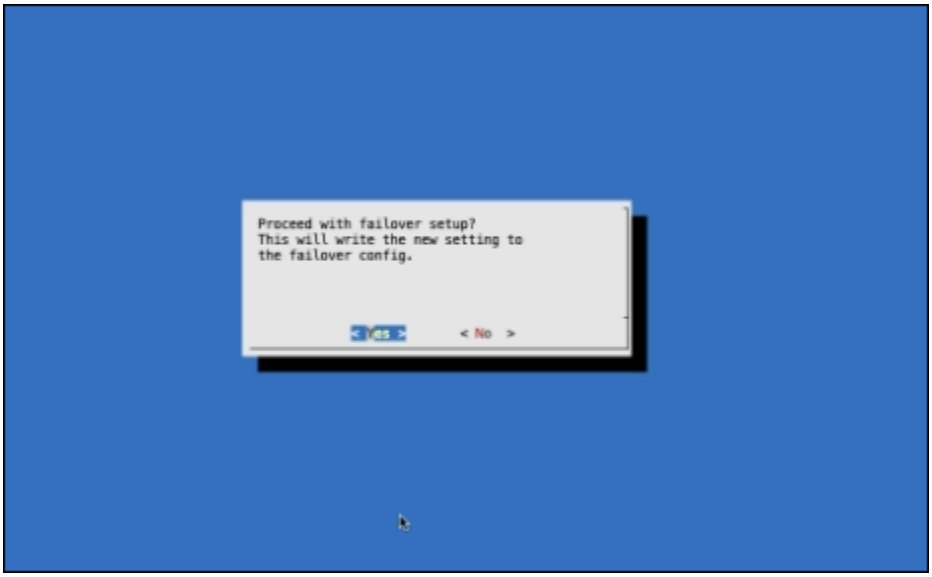
Configure the IP addresses for the secondary head node that the wizard is about to create and then select NEXT.



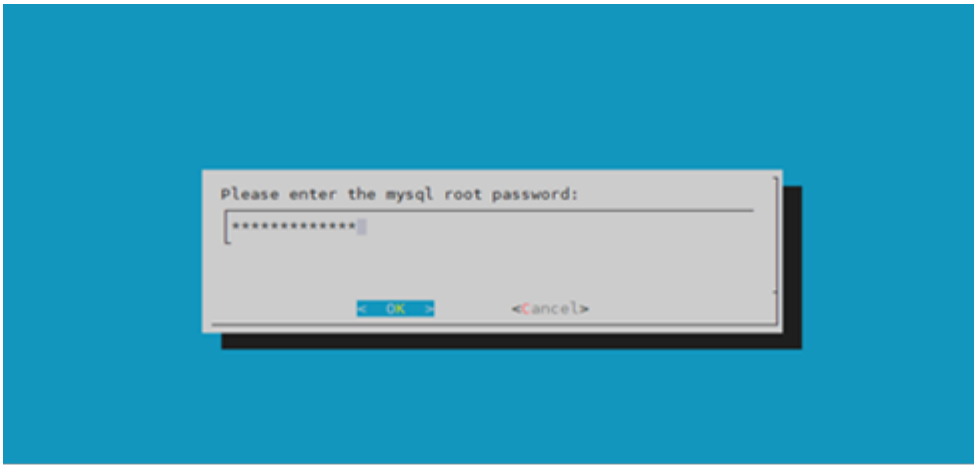
The wizard shows a summary of the information that it has collected. The VIP will be assigned to the internal and external interfaces, respectively.



Select Yes to proceed with the failover configuration.



Enter the BCM root password and then select OK.

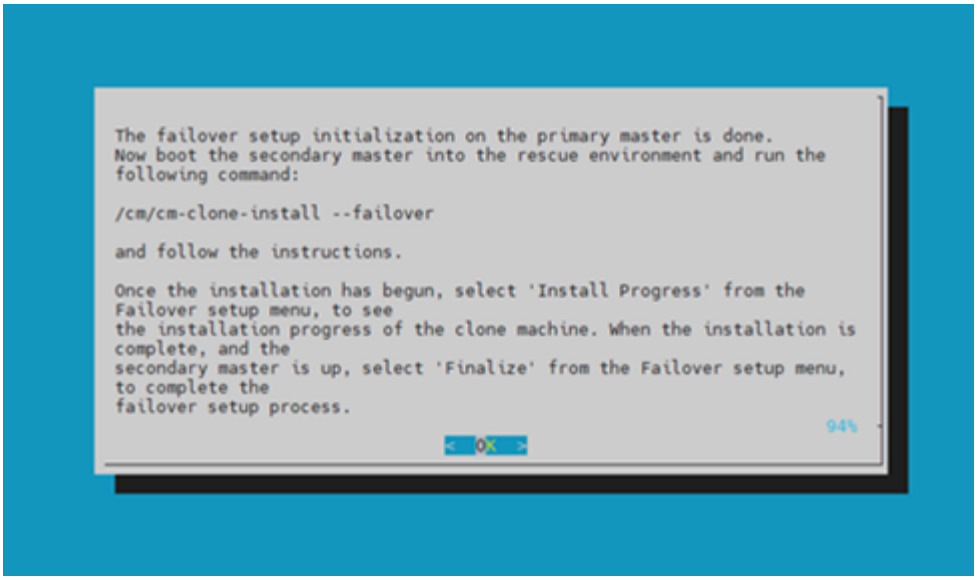


The wizard implements the first steps in the HA configuration. If all the steps show OK, press ENTER to continue. The progress is shown here.

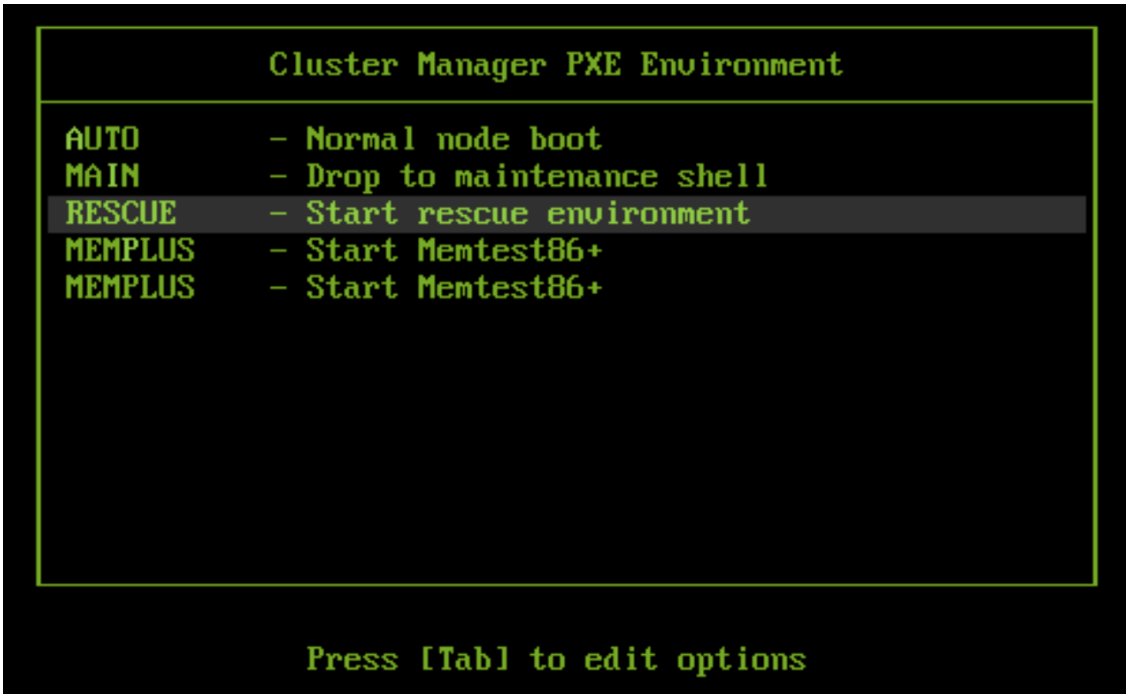
```
Initializing failover setup on master..... [ OK ]
Updating shared internal interface..... [ OK ]
Updating shared external interface..... [ OK ]
Updating extra shared internal interfaces..... [ OK ]
Cloning head node..... [ OK ]
Updating secondary master interfaces..... [ OK ]
Updating Failover Object.....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

When the failover setup installation on the primary master is complete, select OK to exit the wizard to the main HA setup screen

We will come back to the HA setup screen once the secondary headnode is added to the cluster.



Power up and PXE boot the secondary head node and then select RESCUE from the grub menu.

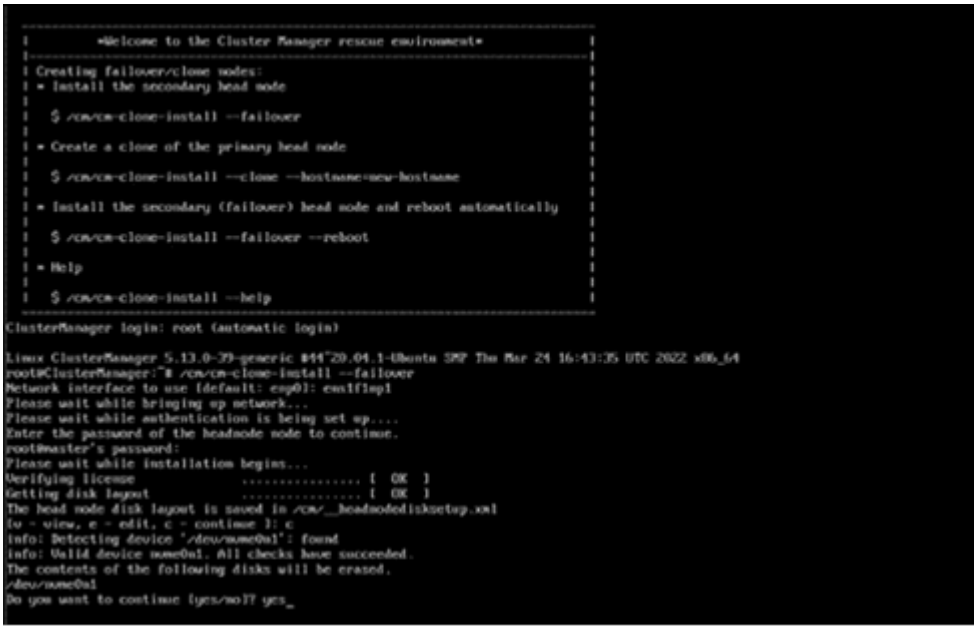


After the secondary head node has booted into the rescue environment, run the

```
/cm/cm-clone-install --failover
```

command, then enter YES when prompted.

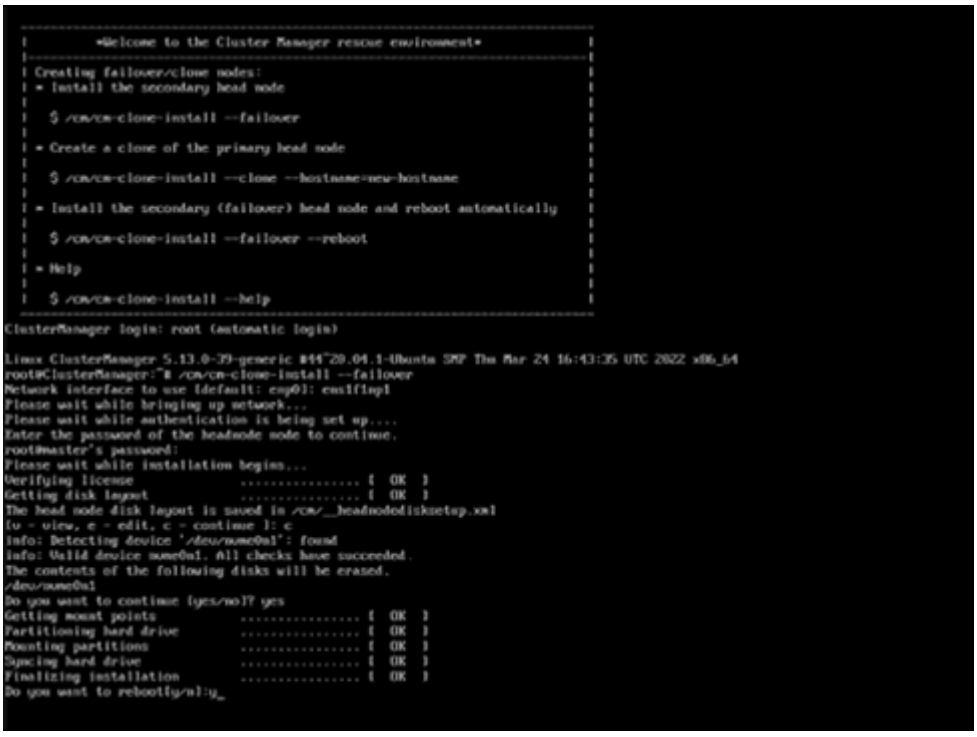
The secondary head node will be cloned from the primary.



Specify the inband interface on the secondary node

Press c to save the disk layout when prompted

When cloning is completed, enter y to reboot the secondary head node. The secondary must be set to boot from its hard drive. PXE boot should not be enabled. Use the appliance BMC/BIOS to change the boot order. Wait for the secondary head node to reboot.

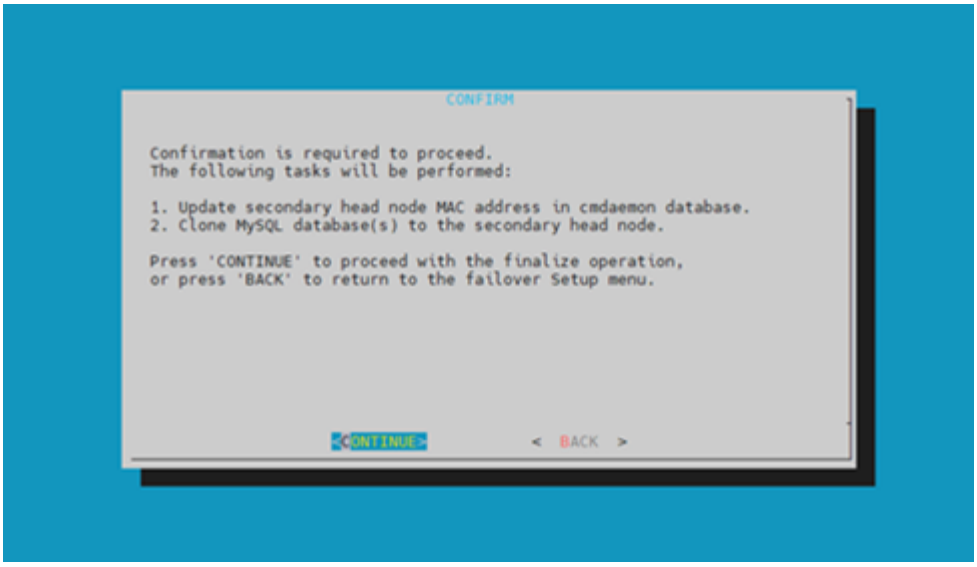


Then continue the HA setup procedure on the primary head node.

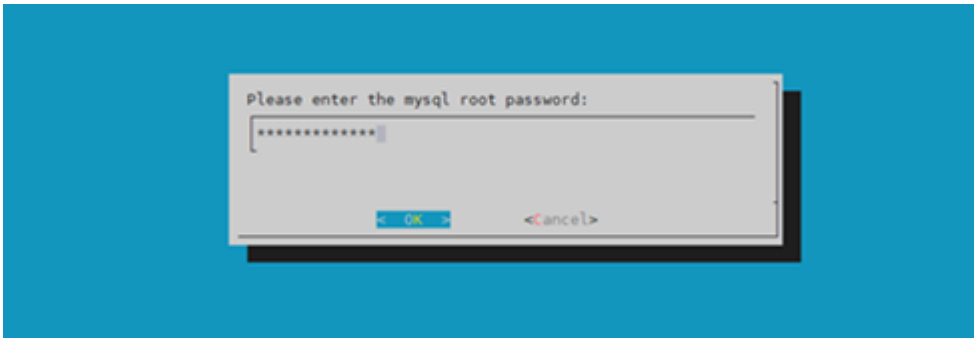
On the primary headnode, in the HA setup screen, select Finalize from the cmha-setup menu and then select NEXT.

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

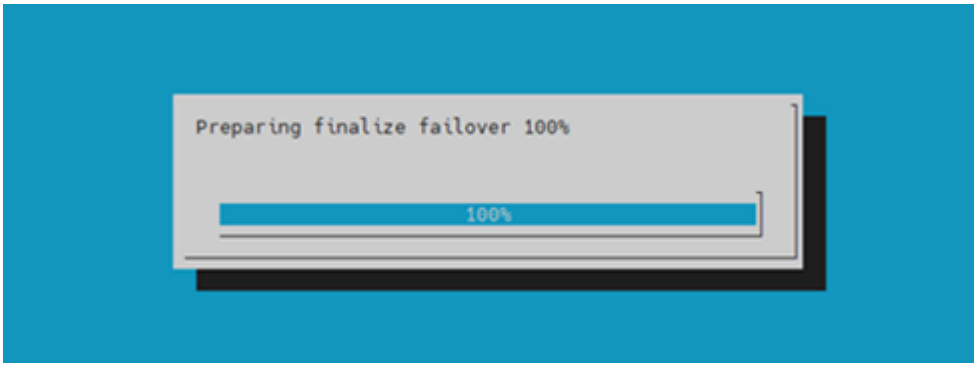
Select CONTINUE on the confirmation screen.



Enter the root password and then select OK.



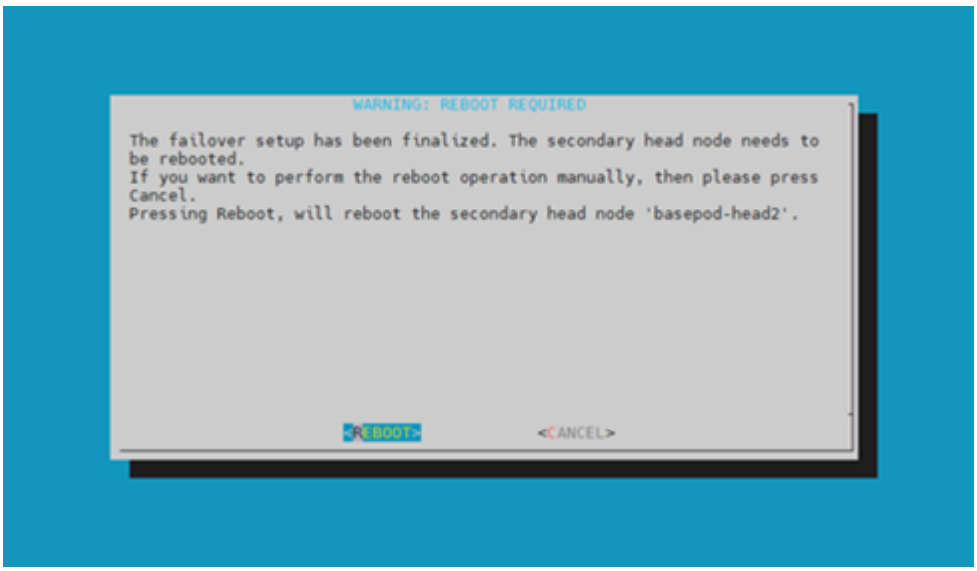
The cmha-setup wizard continues. Press ENTER to continue when prompted.



The progress is shown in the console during the process.

```
Updating secondary master mac address.............. [ OK ]
Initializing failover setup on bcm-head-02......... [ 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
```

The Finalize step is now completed. Select REBOOT and wait for the secondary head node to reboot.



You can verify the secondary node's status from the primary head node using cmsh

```
[bcm10-headnode1]% device list -f
hostname:20,category:12,ip:20,status:15
hostname (key) category ip status
------------------------------------------------------------------
bcm-head-01 10.130.122.254 [ UP ]
bcm-head-02 10.130.122.253 [ UP ]
```

(continues on next page)

(continued from previous page)

```
bcm-dgx-h100-01 dgx-h100 10.130.122.5 [ DOWN ]
bcm-dgx-h100-02 dgx-h100 10.130.122.6 [ DOWN ]
bcm-dgx-h100-03 dgx-h100 10.130.122.7 [ DOWN ]
bcm-dgx-h100-04 dgx-h100 10.130.122.8 [ DOWN ]
```

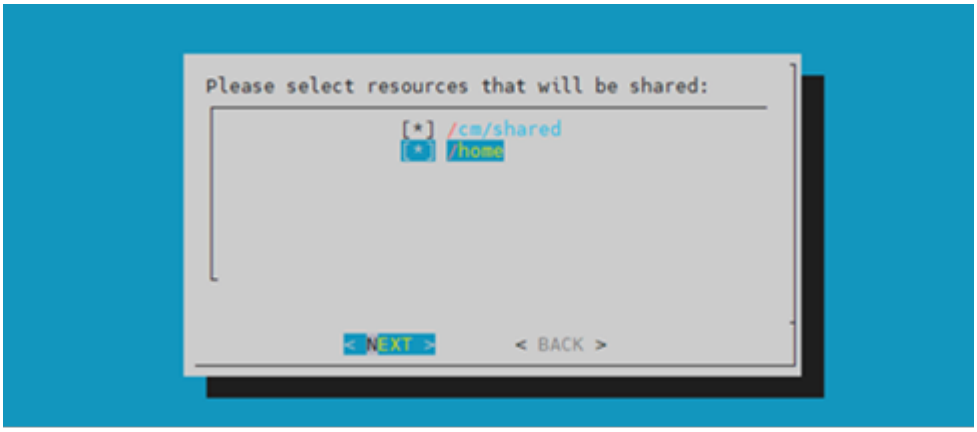
Chapter 12. Setup shared NFS storage.

On the primary headnode, in the HA setup screen, select Shared Storage from the cmha-setup menu and then select SELECT.

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

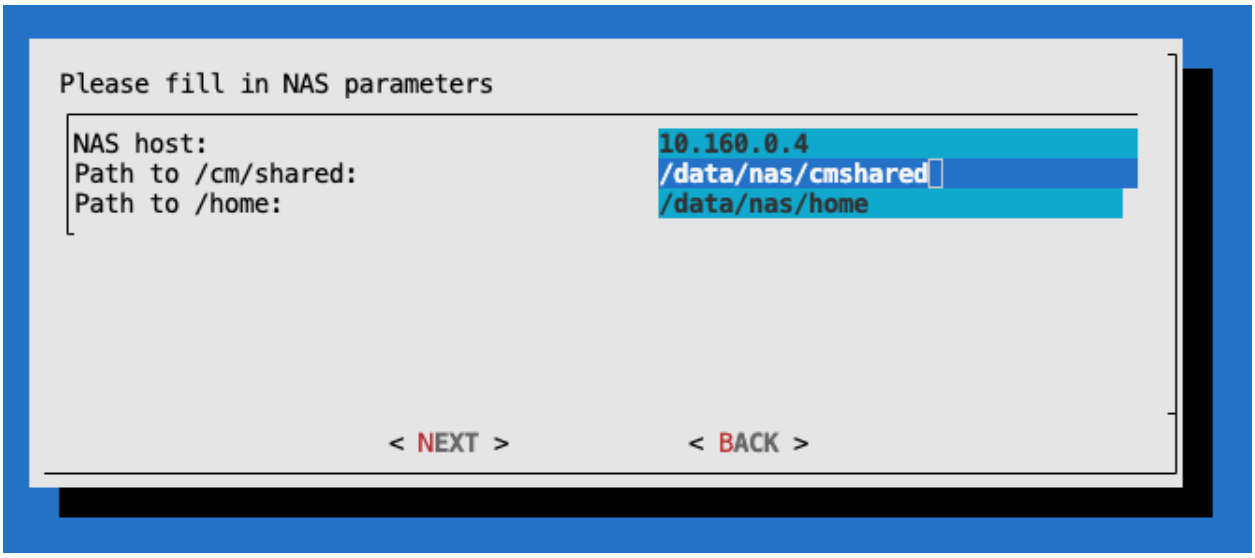
Choose NAS and then select SELECT.

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

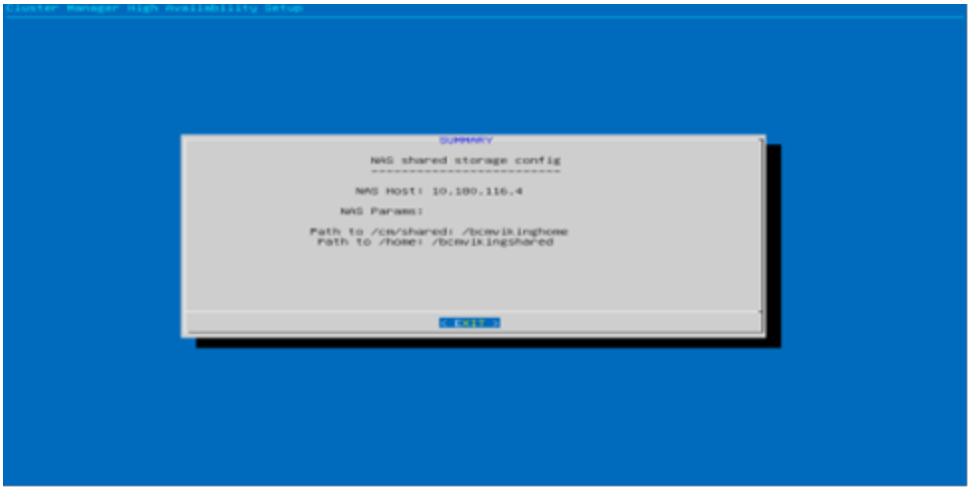


Provide the IP address of the NFS host and the NFS Shared paths for `/cmshared` and `/home` folders and then select NEXT.

Refer to the site survey for the details.

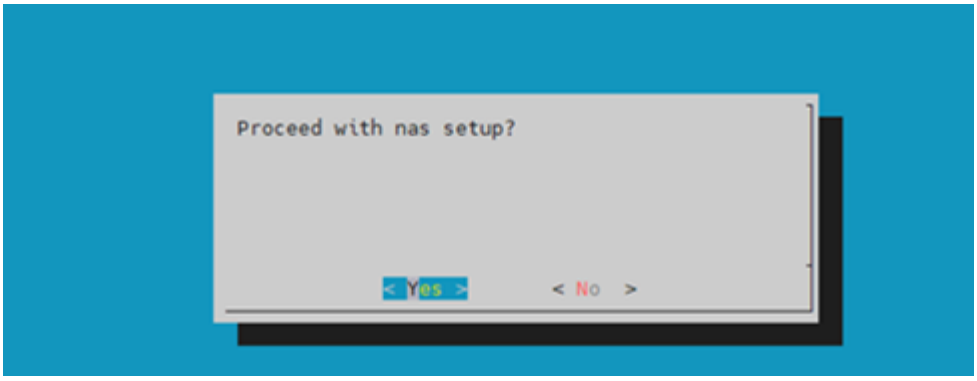


The wizard shows a summary of the information that it has collected. Select EXIT to continue.

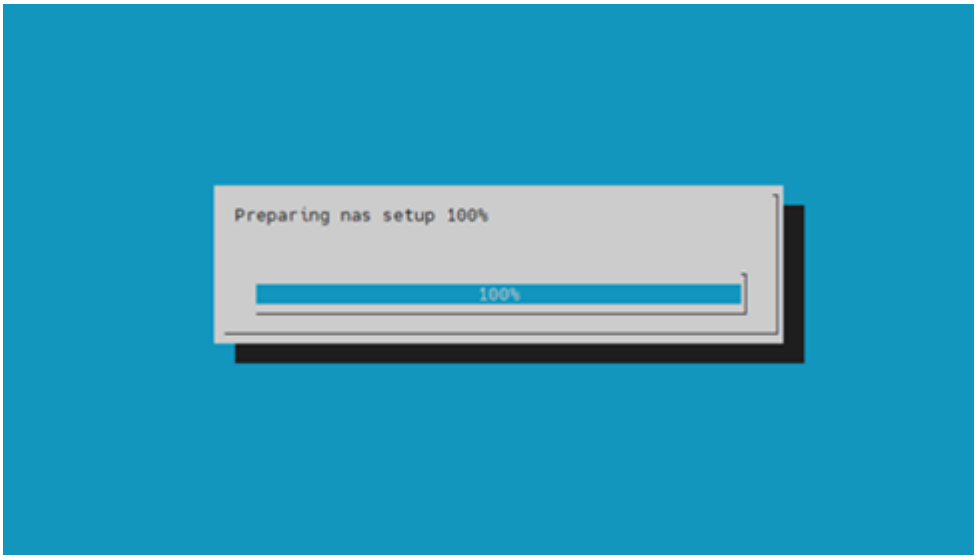


When asked to proceed with the NAS setup, select Yes to continue.

This will initiate a copy and update the NFS mount on the head nodes.



The cmha-setup wizard proceeds with its work.

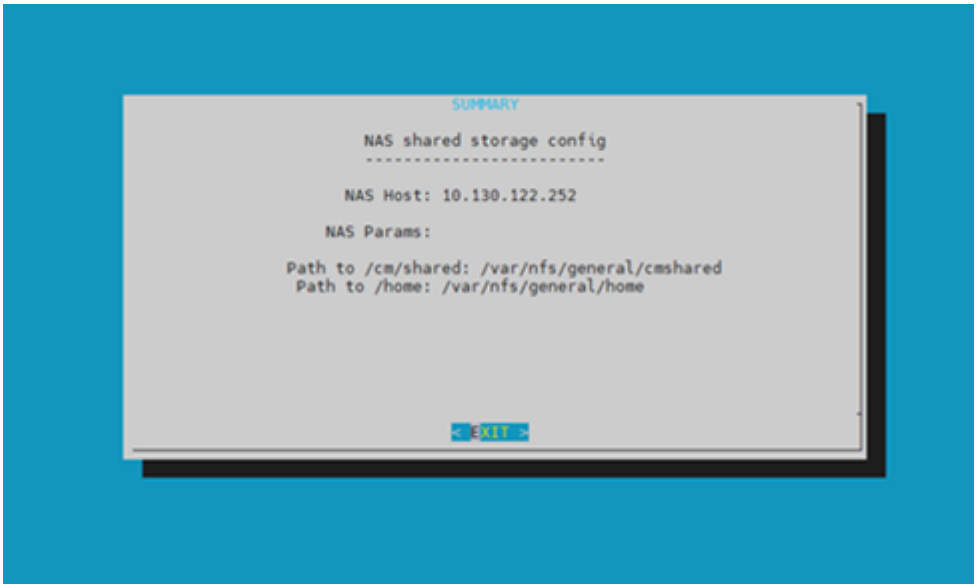


When setup completes, select ENTER to finish HA setup.

The progress is shown here:

```
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
```

cmha-setup is now complete. EXIT the wizard to return to the shell prompt on the primary headnode



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

The active head node is indicated by an asterisk. Here is an example of a working HA configuration.

```
# cmha status
Node Status: running in active mode
bcm-head-01* -> bcm-head-02
failoverping [ OK ]
mysql [ OK ]
ping [ OK ]
status [ OK ]
bcm-head-02 -> bcm-head-01*
failoverping [ OK ]
mysql [ OK ]
ping [ OK ]
status [ OK ]
```

Verify that the `/cm/shared` and `/home` directories are mounted from the NAS server. On the BCM head-node, run the following command.

```
 root@bcm10-headnode1:~# mount | grep nfs
 . . . some output omitted . . .
  10.130.122.252:/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=10.130.122.253,local_lock=none,
 ↪addr=10.130.122.252)
 10.130.122.252:/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=10.130.122.253,local_lock=none,
 ↪addr=10.130.122.252)
```

Chapter 13. Testing BCM HA

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

```
# ssh bcm-head-02
# 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:
bcm-head-02 will become active head node (current state: passive)
bcm-head-01 will become passive head node (current state: active)
================================================================================
Continue(c)/Exit(e)? c
Initiating failover..............................[ OK ]
bcm-head-02 is now active head node, makeactive successful
```

Run the cmsh status command again to verify that the secondary head node has become the active head node.The active head node is indicated by an asterisk.

```
# cmha status
Node Status: running in active mode
bcm-head-02\* -> bcm-head-01
failoverping [ OK ]
mysql [ OK ]
ping [ OK ]
status [ OK ]
bcm-head-01 -> bcm-head-02\*
failoverping [ OK ]
mysql [ OK ]
ping [ OK ]
status [ OK ]
```

Manually failover back to the primary head node by running cmha makeactive

```
# ssh bcm10-headnode1
# 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:
bcm-head-01 will become active head node (current state: passive)
bcm-head-02 will become passive head node (current state: active)
```

(continues on next page)

(continued from previous page)

```
================================================================================
Continue(c)/Exit(e)? c
Initiating failover..............................  [  OK  ]
bcm-head-01 is now active head node, makeactive successful
```

Run cmha status command again to verify that the primary head node has become the active head node. Then proceed to power on the cluster nodes

```
# cmha status
Node Status: running in active mode
bcm-head-01\* -> bcm-head-02
failoverping [ OK ]
mysql [ OK ]
ping [ OK ]
status [ OK ]
bcm-head-02 -> bcm-head-01\*
failoverping [ OK ]
mysql [ OK ]
ping [ OK ]
status [ OK ]
```

Power on the cluster nodes.

```
#cmsh -c "device ; power -c dgx-h100 on"
ipmi0 ..................... [  ON  ] bcm-dgx-h100-01
ipmi0 ..................... [  ON  ] bcm-dgx-h100-02
ipmi0 ..................... [  ON  ] bcm-dgx-h100-03
ipmi0 ..................... [  ON  ] bcm-dgx-h100-04
```

This concludes the setup and verification of HA.

Chapter 14. Deploy Kubernetes

14.1. Kubernetes Node Setup

This section outlines the configuration steps required on the BCM head node to provision the 3 Kubernetes control nodes

First, we will create the software image and define the category for the kubernetes nodes and assign the disklayout.

Add necessary kernel modules for bonding and mellanox drivers.

```
cmsh
[bcm10-headnode1]% softwareimage
[bcm10-headnode1->softwareimage]% clone default-image k8s-control-plane-image
[bcm10-headnode1->softwareimage*[k8s-control-plane-image*]]% commit
[bcm10-headnode1->softwareimage[k8s-control-plane-image]]%kernelmodules
[bcm10-headnode1->softwareimage[k8s-control-plane-image]->kernelmodules]% add
↪mlx5_core
[bcm10-headnode1->softwareimage[k8s-control-plane-image]->kernelmodules]% add
↪bonding
[bcm10-headnode1->softwareimage*[k8s-control-plane-image*]->
↪kernelmodules*[mlx5_core*]]%commit
[bcm10-headnode1->softwareimage[k8s-control-plane-image]]% category
[bcm10-headnode1->category]% clone default k8s-control-plane
[bcm10-headnode1->category*[k8s-control-plane*]]% set softwareimage k8s-
↪control-plane-image
[bcm10-headnode1->category*[k8s-control-plane*]]% set disksetup /cm/local/
↪apps/cmd/etc/htdocs/disk-setup/x86_64-slave-one-big-partition-ext4.xml
[bcm10-headnode1->category*[k8s-control-plane*]]% commit
```

Create the first kubernetes node - knode-01- by cloning the default node01 and move it to the k8s-control-plane category.

```
cmsh
[bcm10-headnode1]% device
[bcm10-headnode1->device]% clone node01 knode-01
[bcm10-headnode1->device*[knode-01*]]% set category k8s-control-plane
[bcm10-headnode1->device*[knode-01*]]% commit
```

Add and configure the IPMI (BMC) and managementnet (internalnet) bond interfaces.

Note

The name of the interfaces will change depending on the hardware vendor of the node appliance.

```
#cmsh
[bcm10-headnode1]% device
[bcm10-headnode1->device]% use knode-01
[bcm10-headnode1->device[knode-01]]% interfaces
[bcm10-headnode1->device[knode-01]->interfaces]% add bmc ipmi0 10.160.6.4
↪ipminet
[bcm10-headnode1->device*[knode-01*]->interfaces*[ipmi0*]]% commit
[bcm10-headnode1->device[knode-01]->interfaces]% add physica ens2f1np1; add
↪physical ens1f1np1
[bcm10-headnode1->device*[knode-01*]->interfaces*[ens1f1np1*]]% commit
[bcm10-headnode1->device[knode-01]->interfaces]% add bond bond0 10.184.94.4
↪managementnet
[bcm10-headnode1->device*[knode-01*]->interfaces*[bond0*]]% append interfaces
↪ens2f1np1 ens1f1np1
[bcm10-headnode1->device*[knode-01*]->interfaces*[bond0*]]% remove bootif
[bcm10-headnode1->device*[knode-01*]->interfaces*[bond0*]]% ..
[bcm10-headnode1->device*[knode-01*]->interfaces*]% ..
[bcm10-headnode1->device*[knode-01*]]% set provisioninginterface bond0
[bcm10-headnode1->device*[knode-01*]]% commit
```

Clone knode-01 to create the two additional knodes.

```
[bcm10-headnode1]% device
[bcm10-headnode1->device]% foreach --clone knode-01 -n knode-02..knode-03 --
↪next-ip ()
[bcm10-headnode1->device*]% commit
```

Set the MAC addresses for each of the knodes so that they can PXE boot from BCM. Refer to the site survey for details.

```
cmsh
[bcm10-headnode1]% device
[bcm10-headnode1->device]% use knode-01
[bcm10-headnode1->device[knode-01]]% interfaces
[bcm10-headnode1->device[knode-01]->interfaces]% set ens2f1np1 mac
↪00:CC:EE:FF:77:88
[bcm10-headnode1->device*[knode-01*]->interfaces*]% set ens1f1np1 mac
↪00:CC:EE:FF:77:99
[bcm10-headnode1->device*[knode-01*]->interfaces*]% exit
[bcm10-headnode1->device*[knode-01*]]% set mac 00:CC:EE:FF:77:88
[bcm10-headnode1->device*[knode-01*]]% commit
```

Repeat these steps for all 3 knodes, to get their interface/MAC mapping in BCM for PXE boot/provisioning.

Power on and BCM will provision the kubernetes nodes with PXE boot.

```
cmsh
```

(continues on next page)

(continued from previous page)

```
[bcm10-headnode1]% device
[bcm10-headnode1->device]% power on -c k8s-control-plane
```

Ensure all the nodes are up

```
[bcm10-headnode1->device]% list -c k8s-control-plane
Type Hostname (key) MAC Category IP Network Status
--------------------------------------------------------
PhysicalNode Knode1 9E:1D:D5:41:E4:96 k8s-control-plane 10.184.94.4
 ↪managementnet [ UP ]
PhysicalNode Knode2 52:FD:DB:48:5C:4D k8s-control-plane 10.184.94.5
 ↪managementnet [ UP ]
PhysicalNode Knode3 C2:6F:59:F9:75:6C k8s-control-plane 10.184.94.6
 ↪managementnet [ UP ]
```

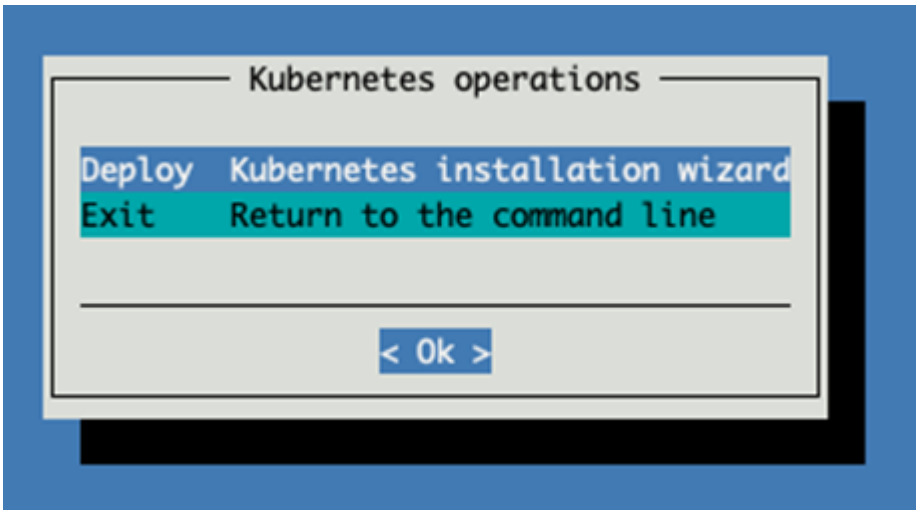
14.2. Kubernetes Deployment

This section addresses configuration steps to be performed on the BCM head node.

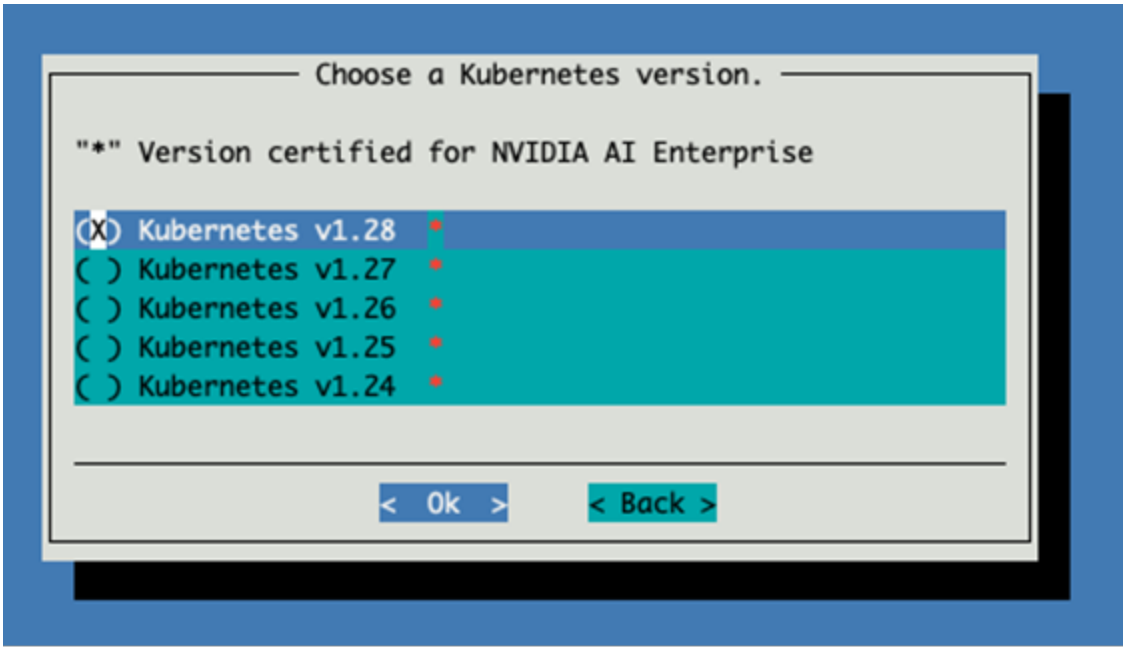
In the root shell, run the Kubernetes setup script.

```
cm-kubernetes-setup
```

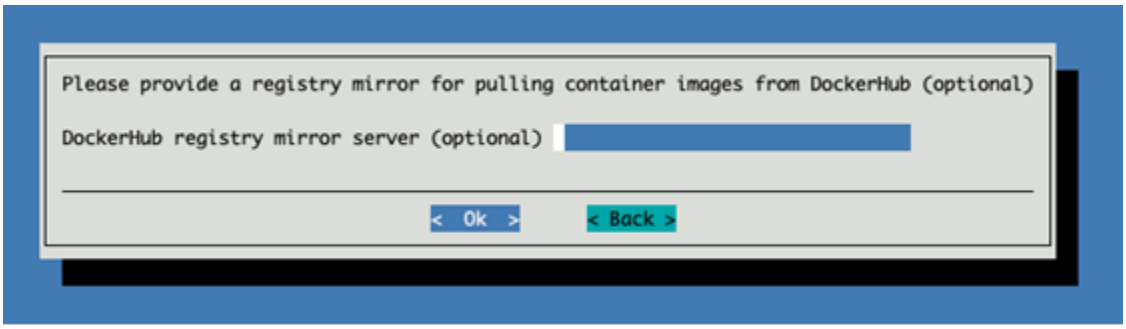
Select Deploy to continue



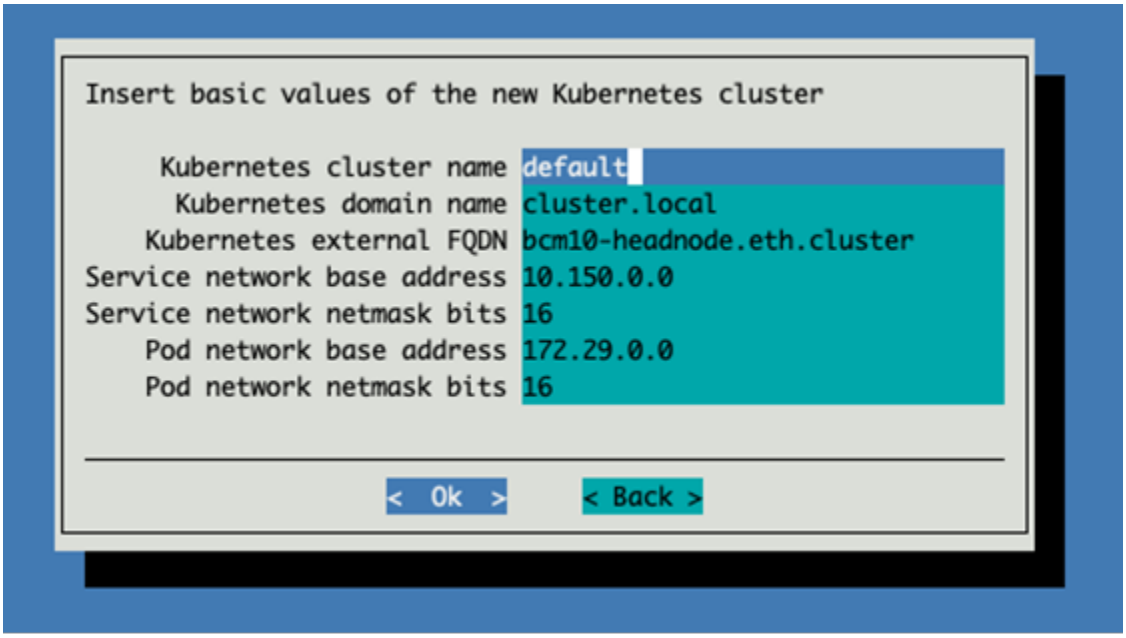
Select the newest version certified for NVIDIA AI Enterprise (i.e. version with *).



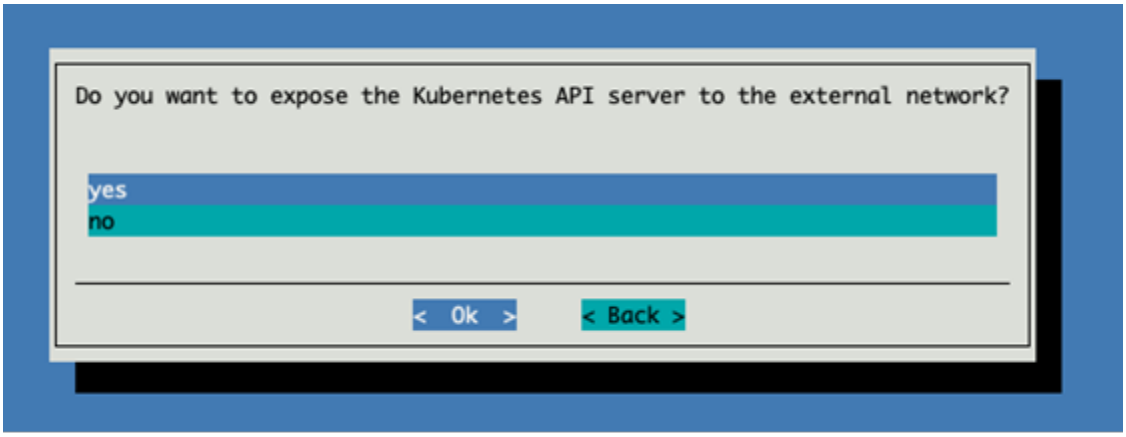
(Optional) Enter a private registry server here if required.



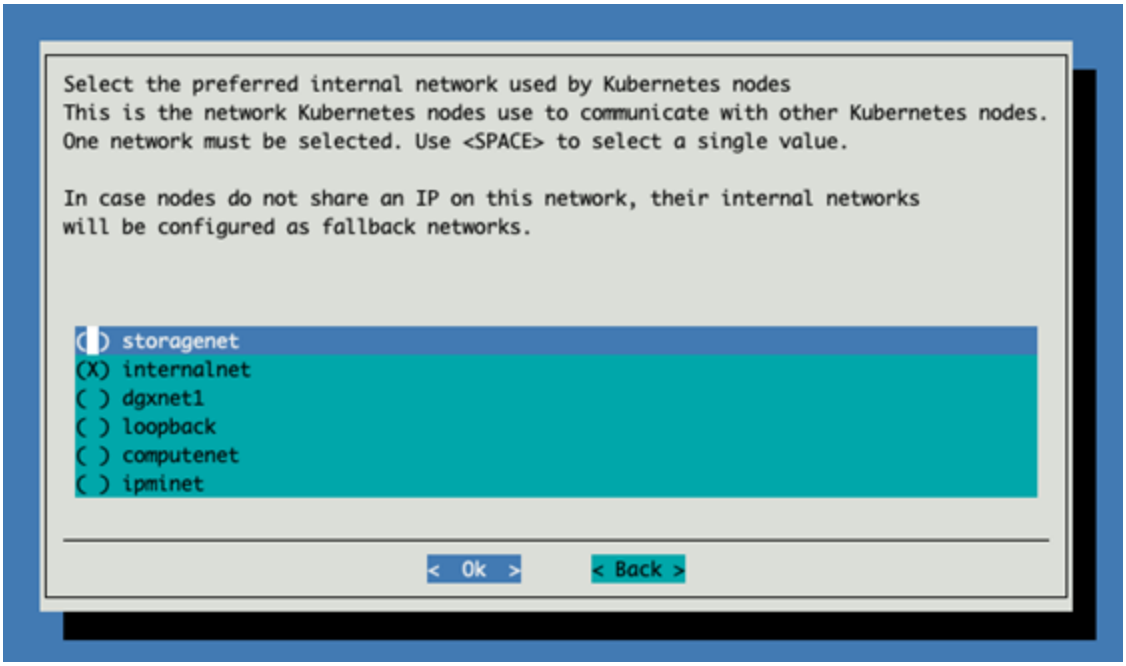
Keep the default settings for the cluster.



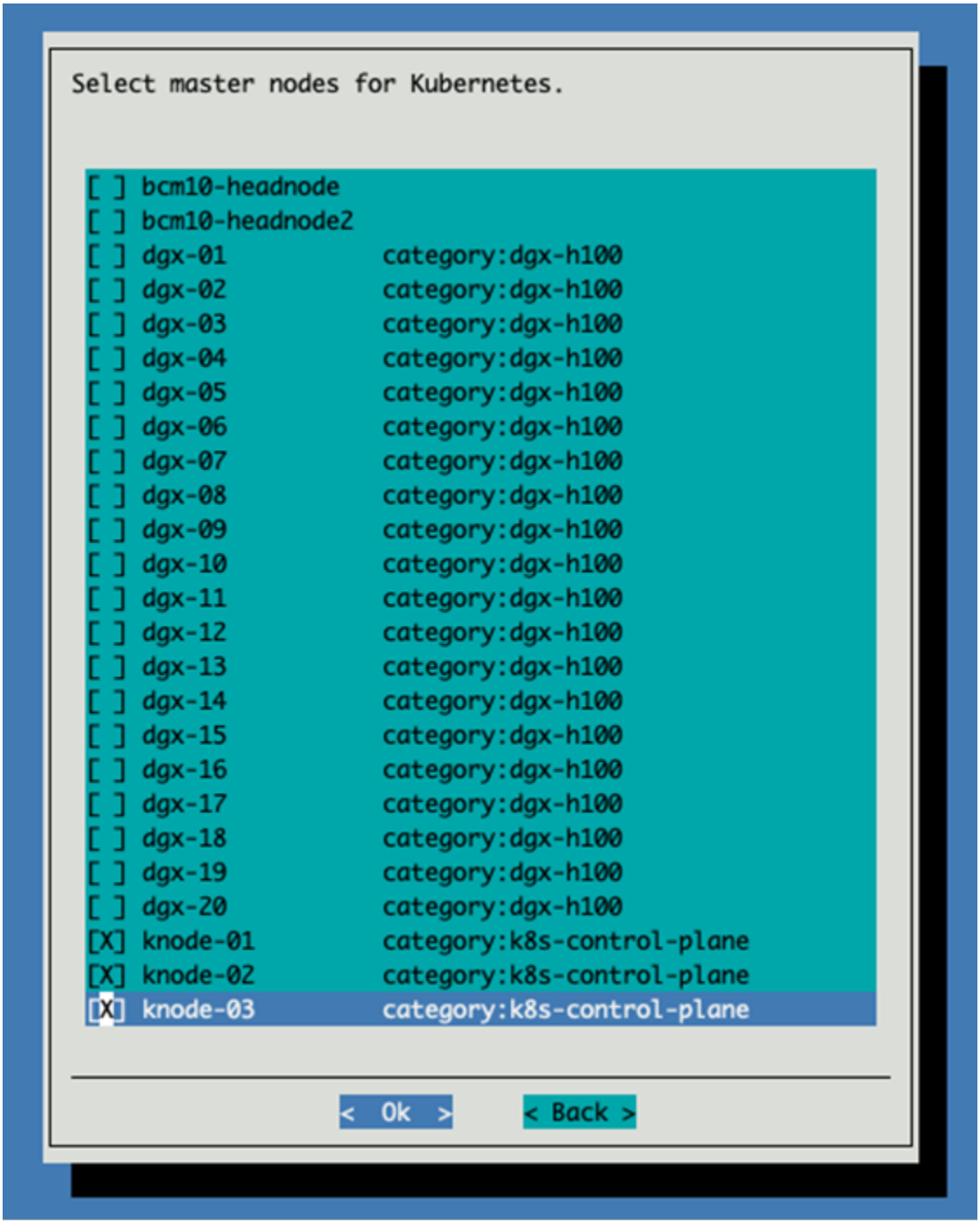
Select yes to allow the cluster to be used from the headnode.



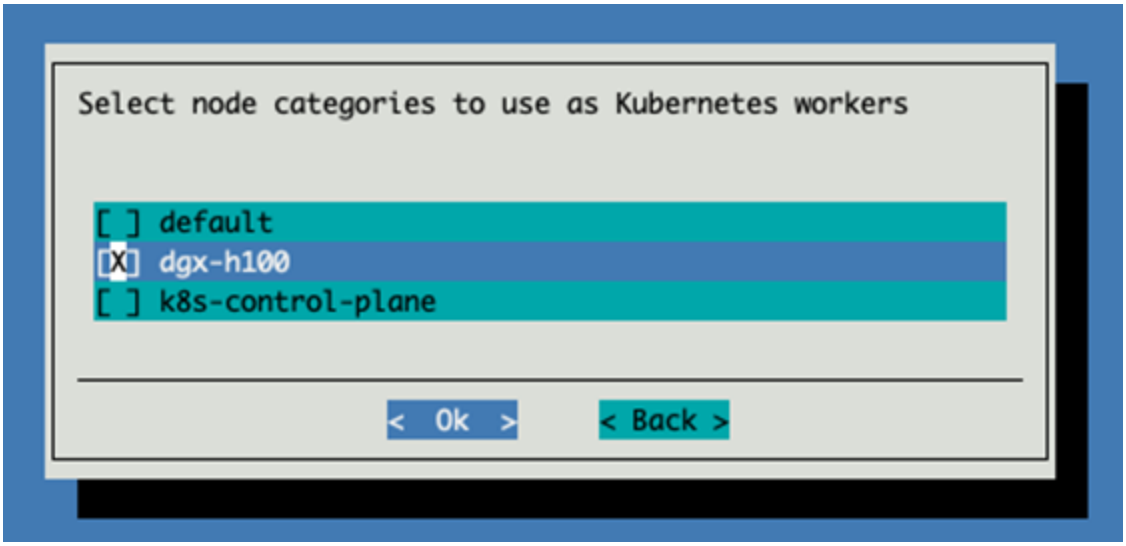
Select managementnet (internalnet) for kubernetes networking.



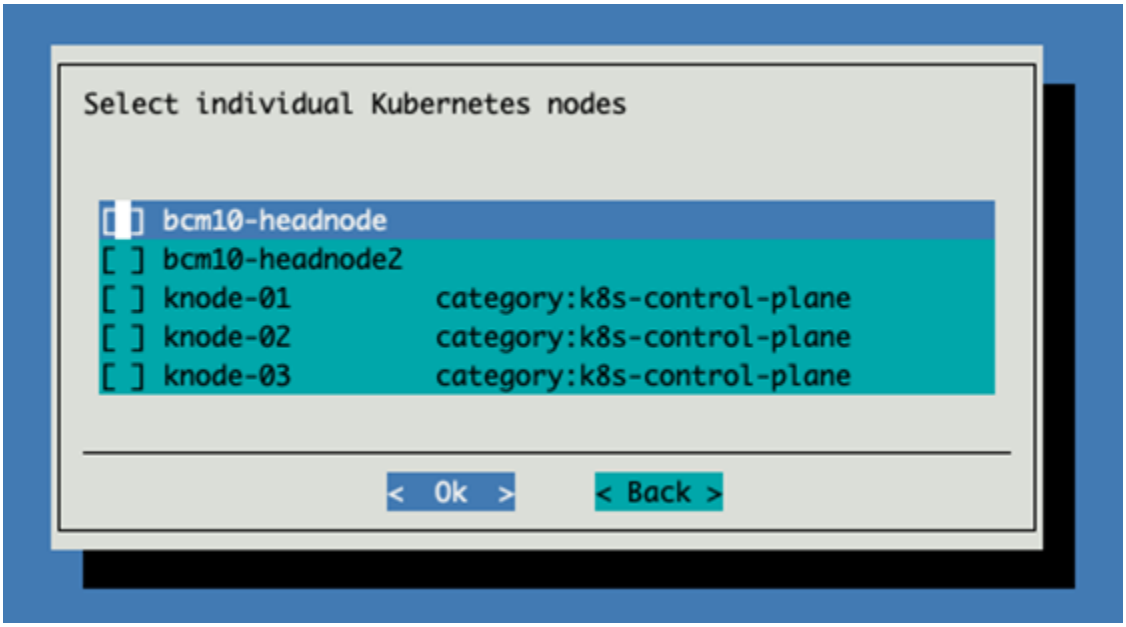
Select the 3 knodes we just provisioned for the control plane.



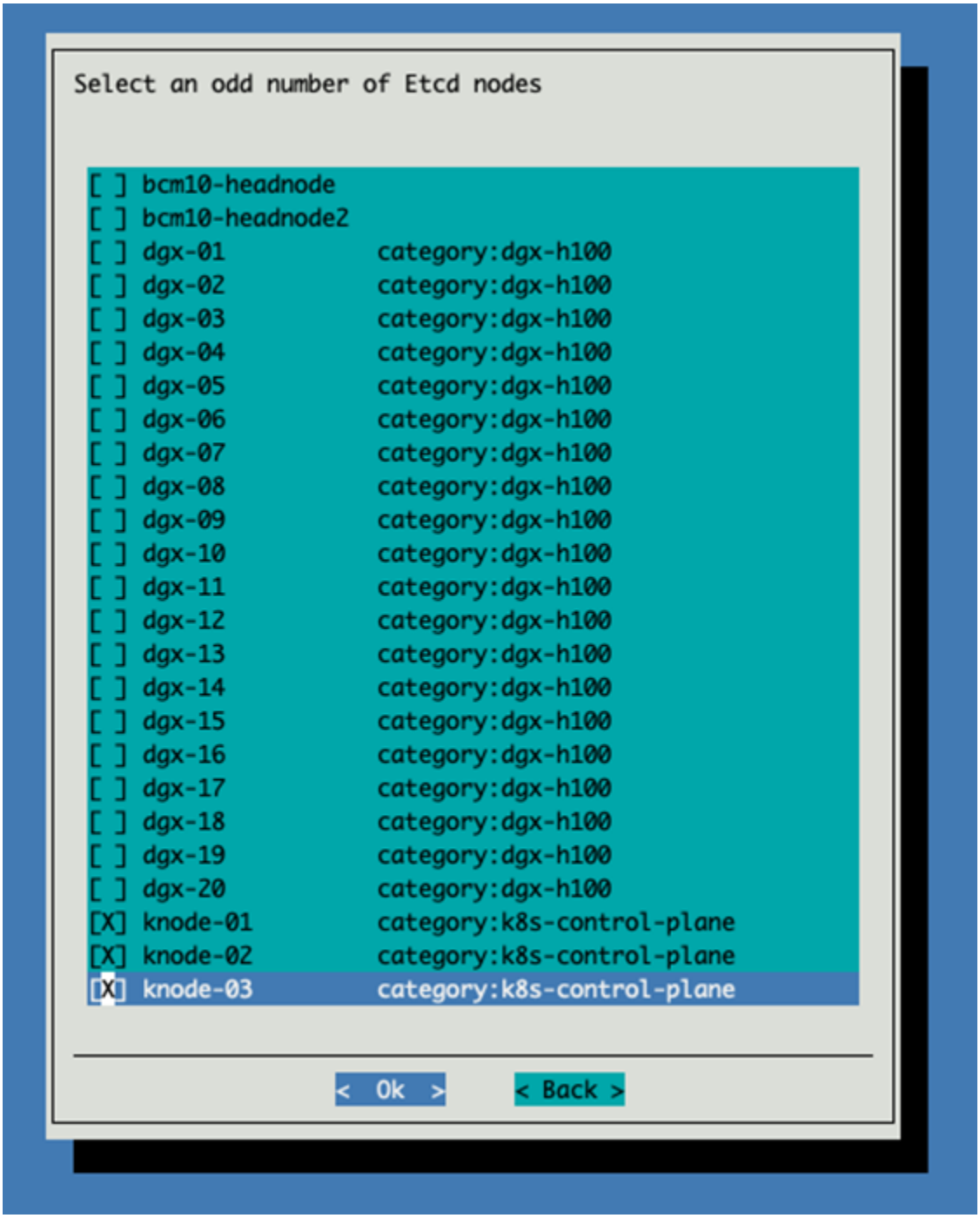
Select the dgx-h100 category for Kubernetes workers.



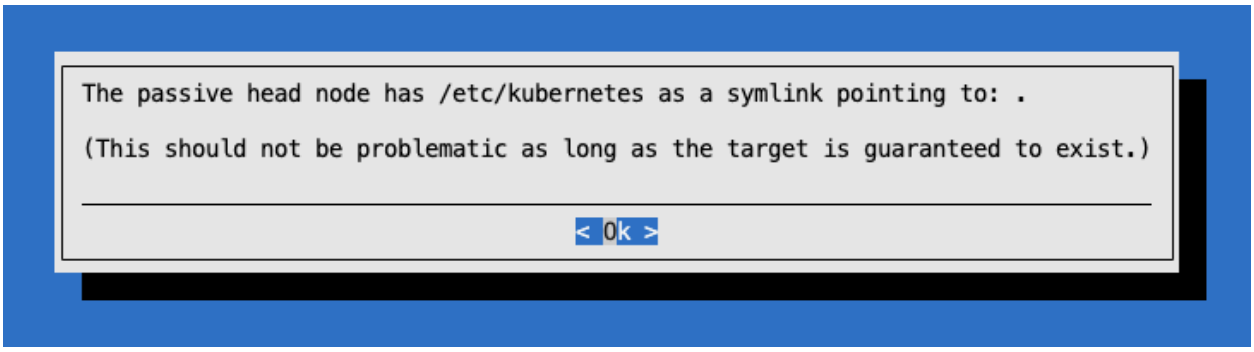
Skip selecting individual worker nodes.



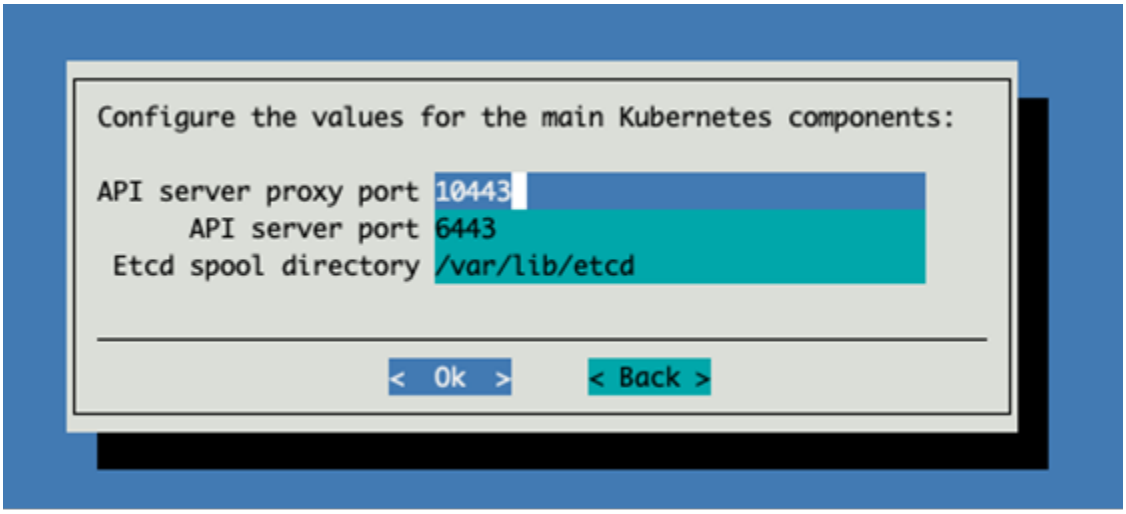
Select the 3 kubernetes nodes to be etcd nodes.



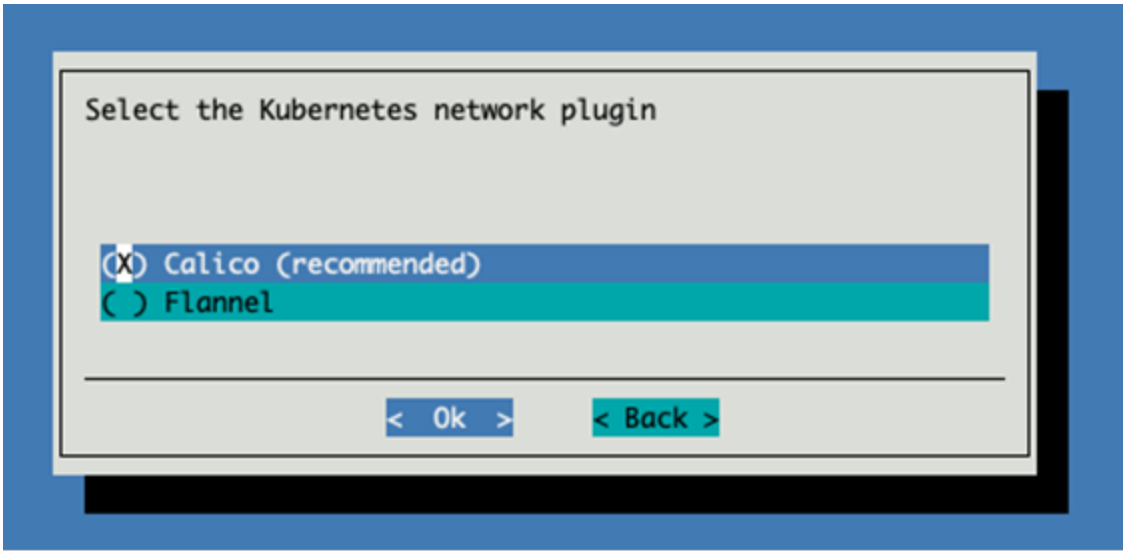
Click “OK” to the symlink dialogue



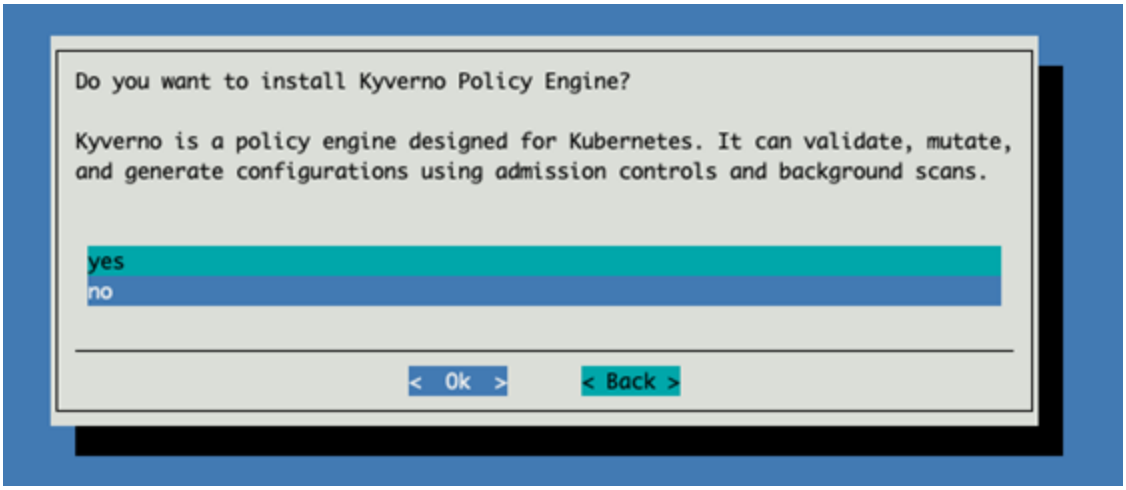
Accept the default values for the main Kubernetes components.



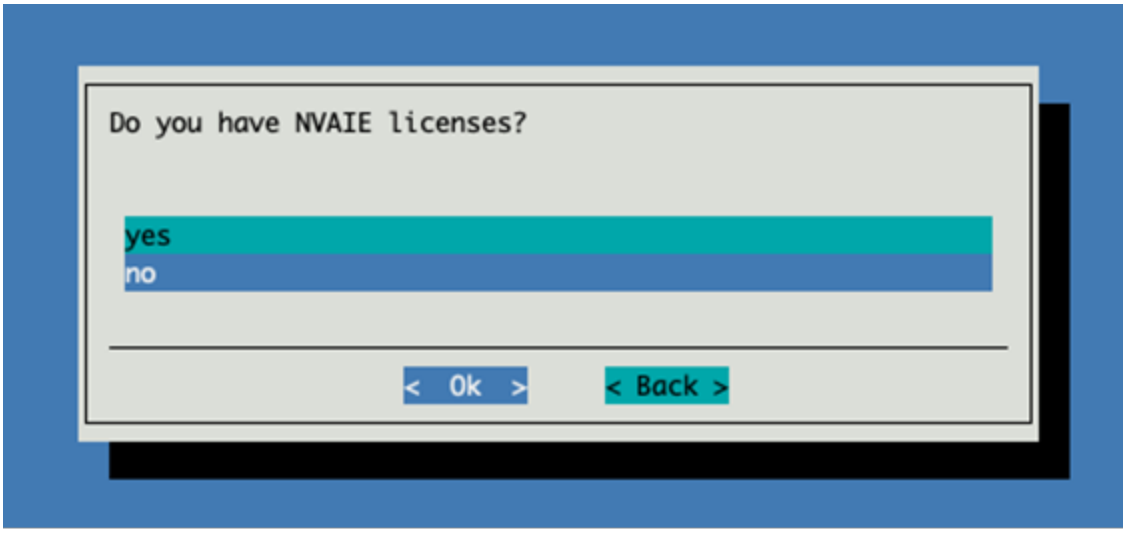
Select the Calico CNI plugin.



Select no for installing the Kyverno Policy Engine.



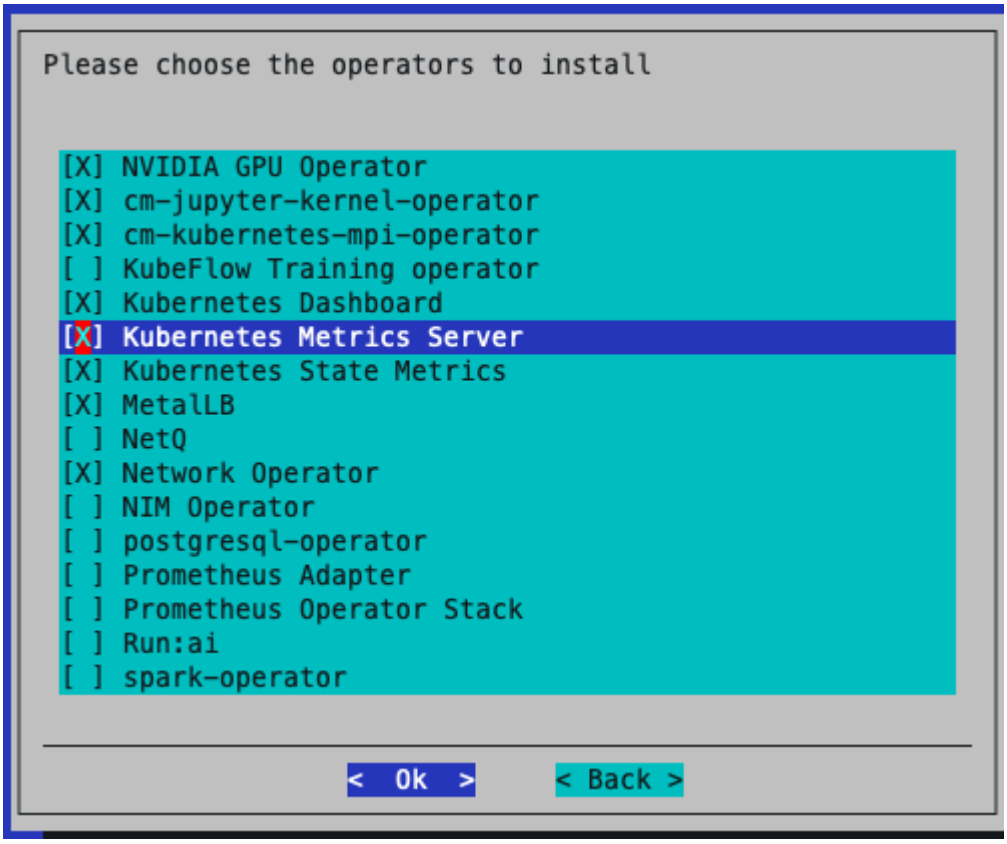
(Optional) If an NVAIE license has been provided, select yes and enter the details on the following page. Otherwise, select no.



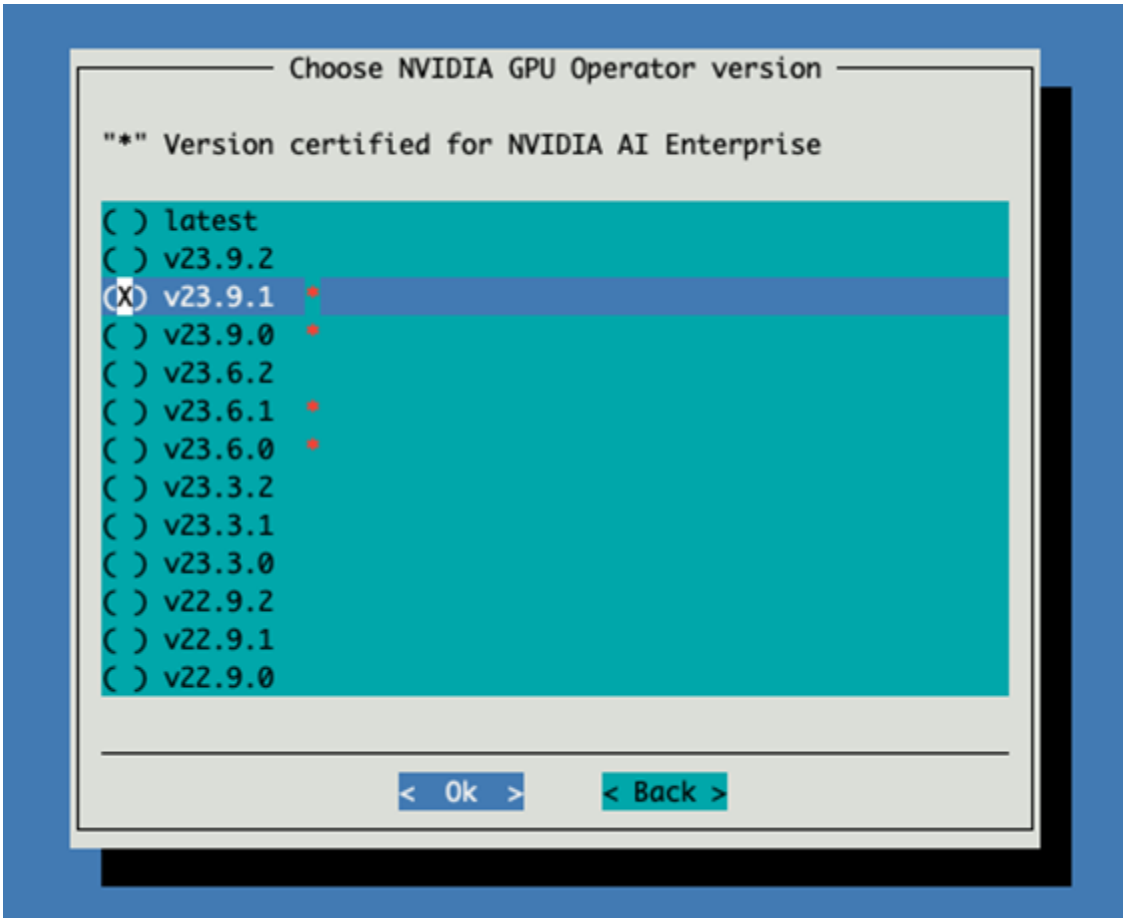
Select the following operators to install:

- NVIDIA GPU Operator
- cm-jupyter-kernel-operator
- cm-kubernetes-mpi-operator
- Network Operator
- Kubeflow Training Operator
- Prometheus Adapter
- Prometheus Operator Stack

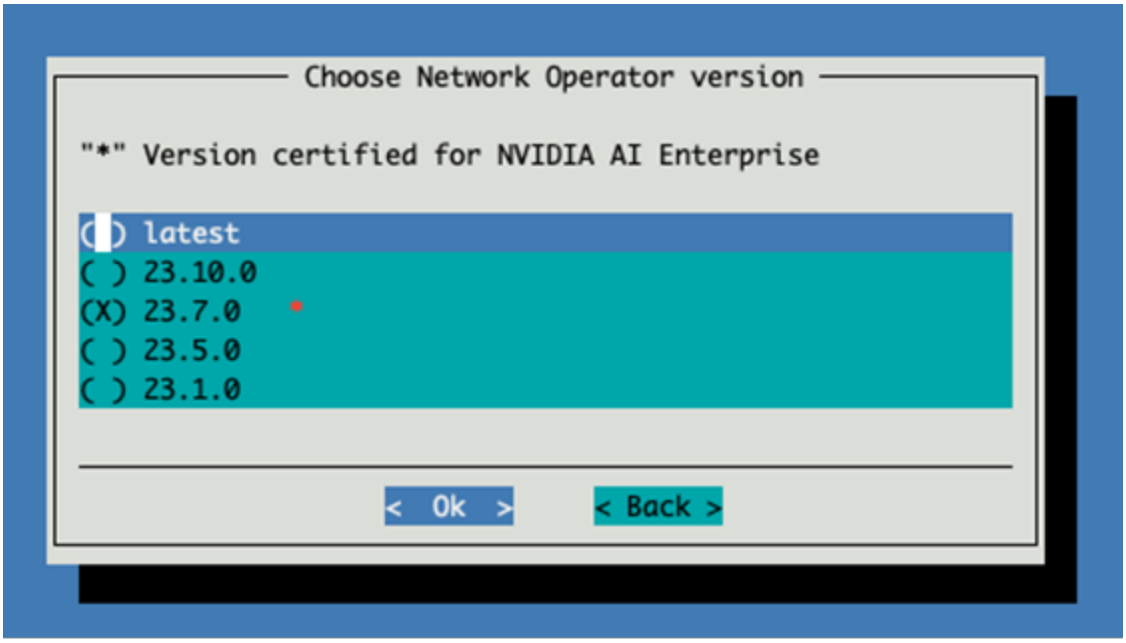
Optional - MetalLB if you are planning to expose services directly from the DGX nodes, like NIM/inference workloads



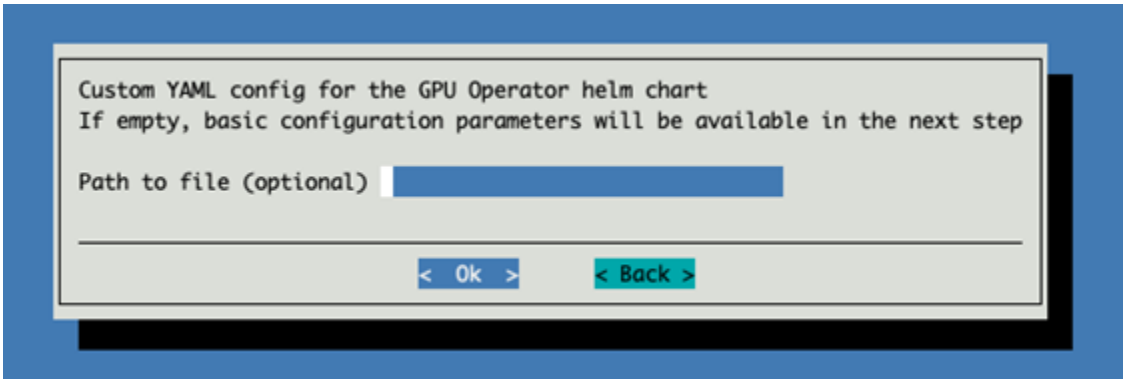
Select the latest version certified for NVIDIA AI Enterprise.



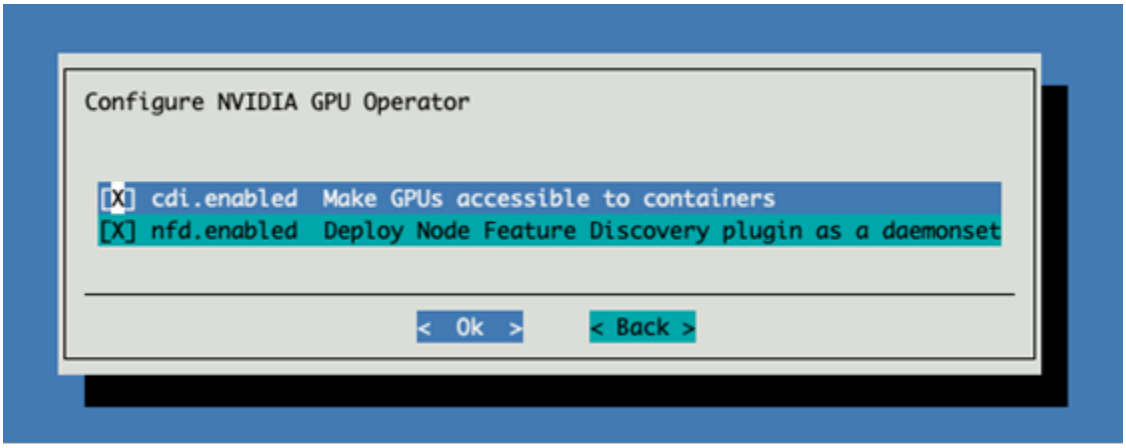
Select the latest version of network operator certified for NVIDIA AI Enterprise.



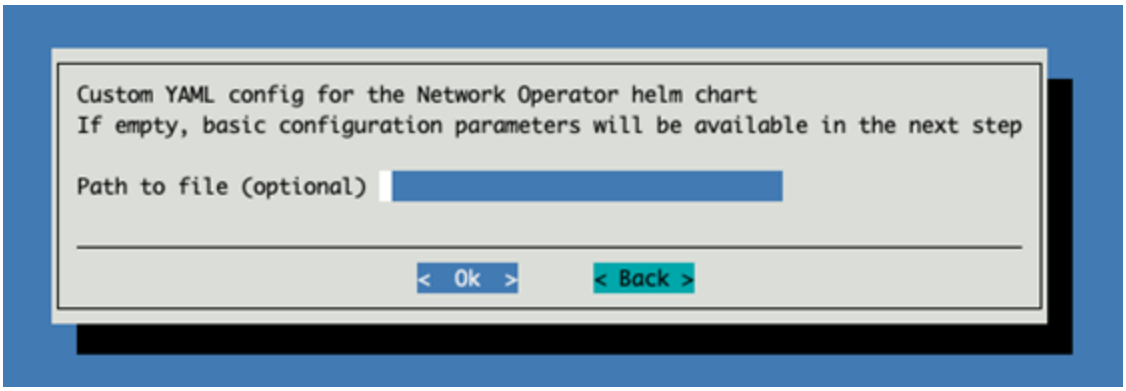
Leave the custom YAML file blank for the GPU Operator.



Enable CDI (container device interface) and NFD (node feature discovery)



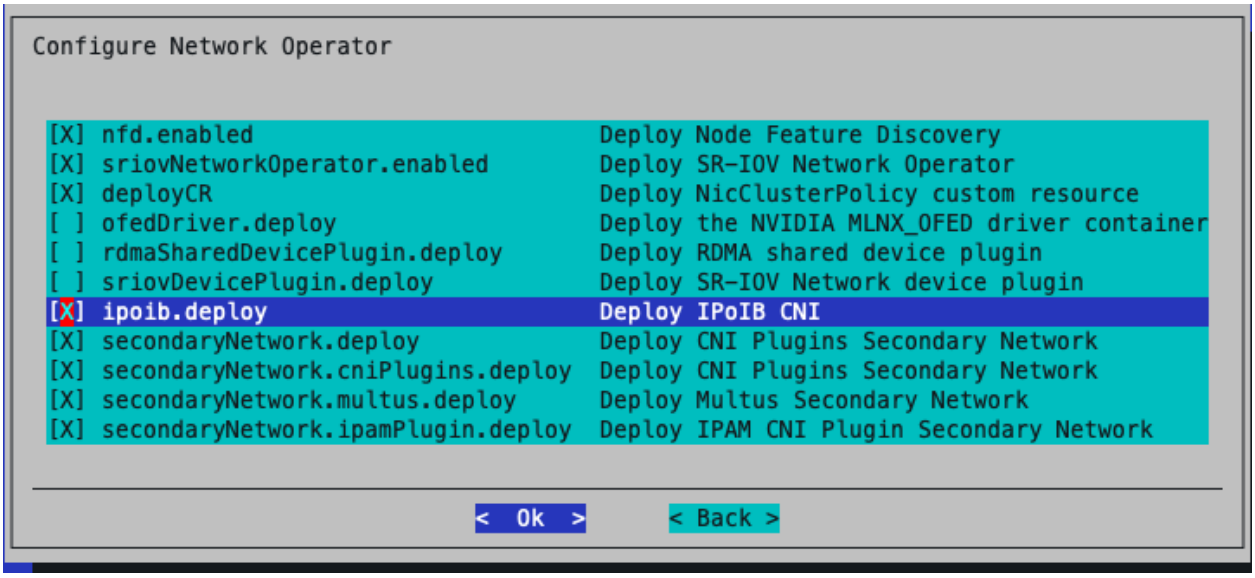
Leave the Network Operator custom YAML configuration file name as blank.



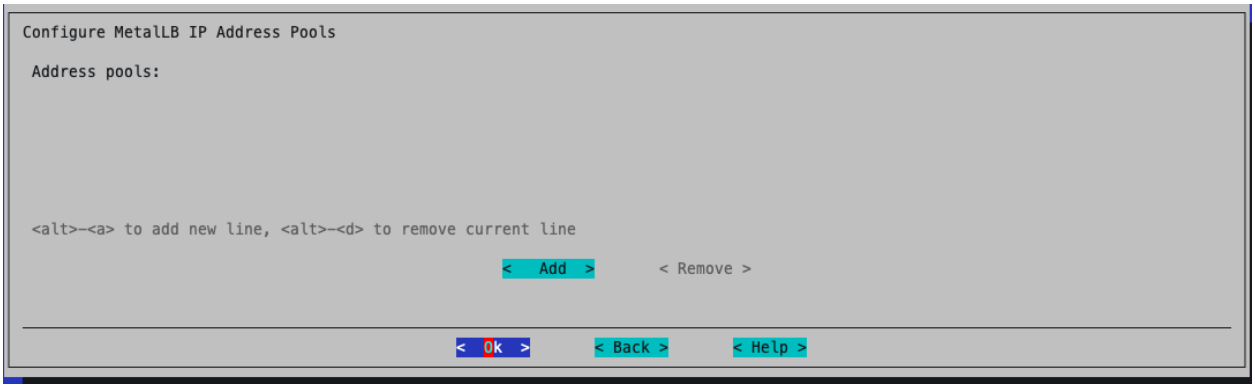
Select the following configuration options for the Network Operator:

- NFD - Node Feature discovery
- SRIOV - SRIOV Network Operator
- CR - Deploy NIC Cluster Policy CR
- IPoIB - Enable IP over Infiniband CNI

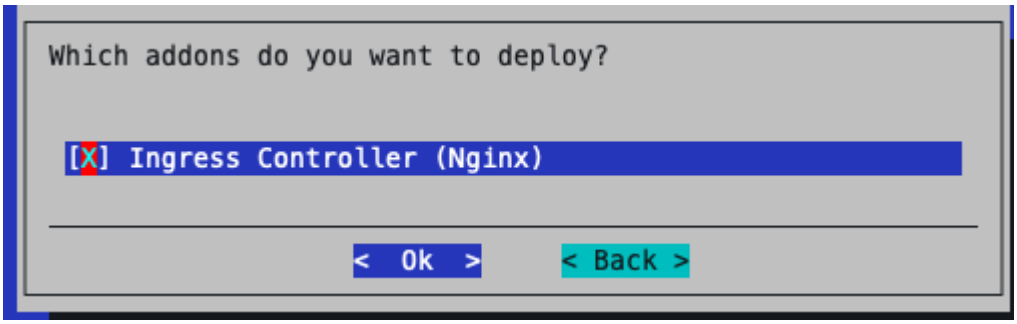
All secondaryNetwork Components - To deploy RDMA interface to the containers.



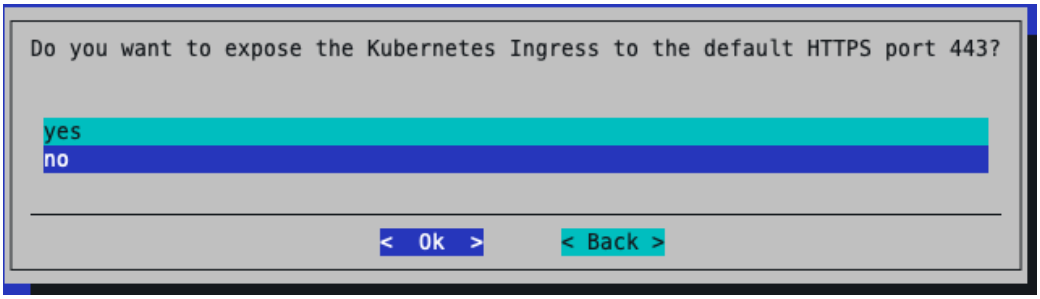
If MetalLB is enabled, define the MetalLB IP pool. Allocate a free IP range for MetalLB from the internal network range or as appropriate. Refer to the MetalLB configuration guide for further details.



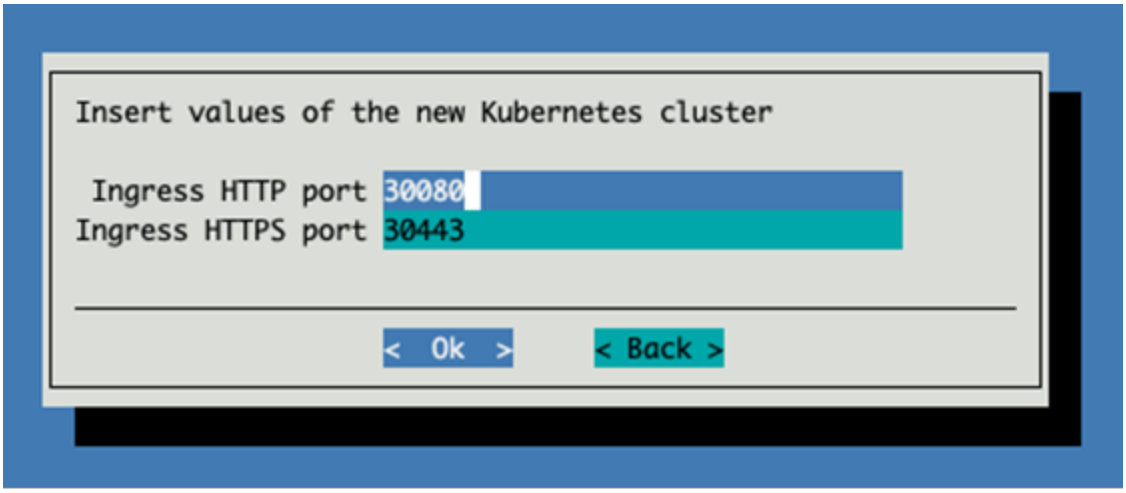
Deploy all the addons.



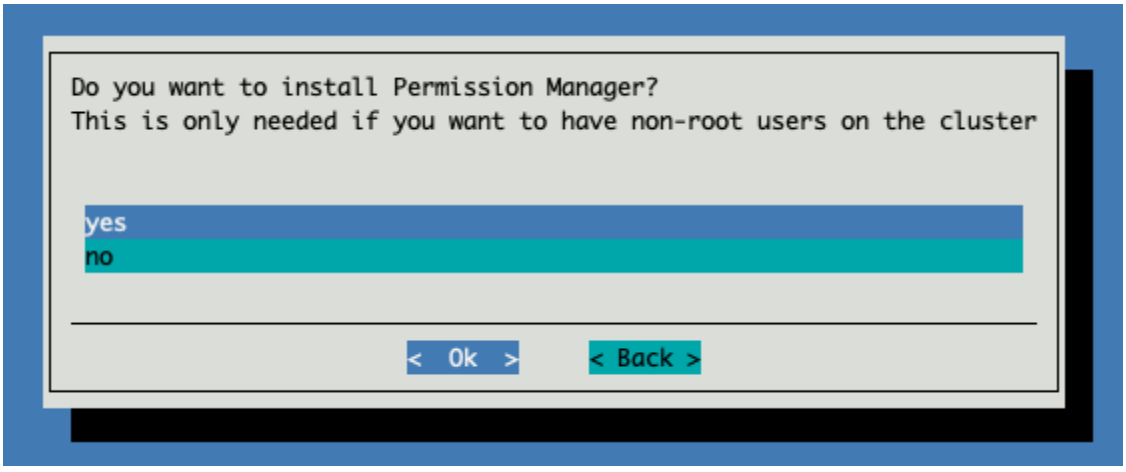
Select Yes to the Ingress default port



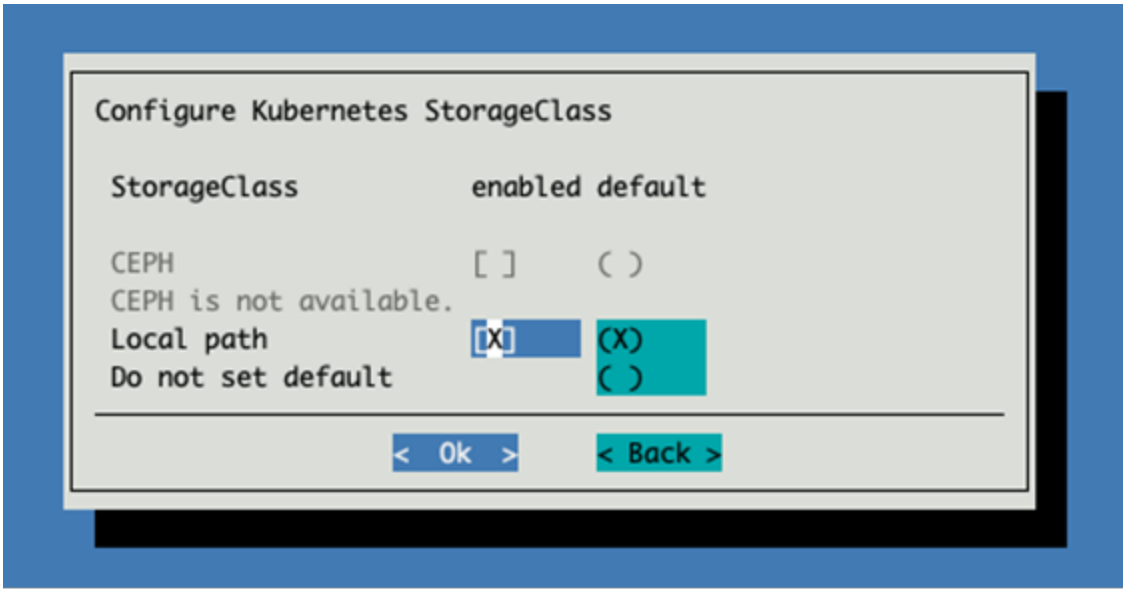
Leave the ports as default.



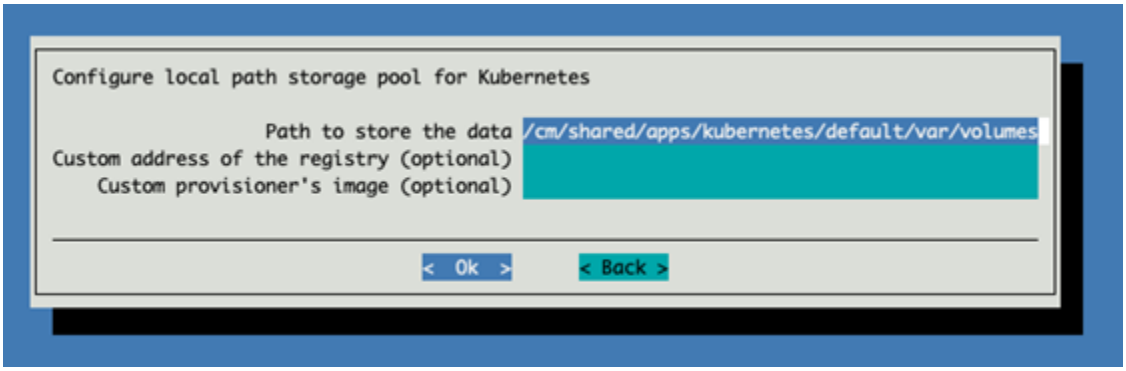
Choose **yes** to install the Permission Manager.



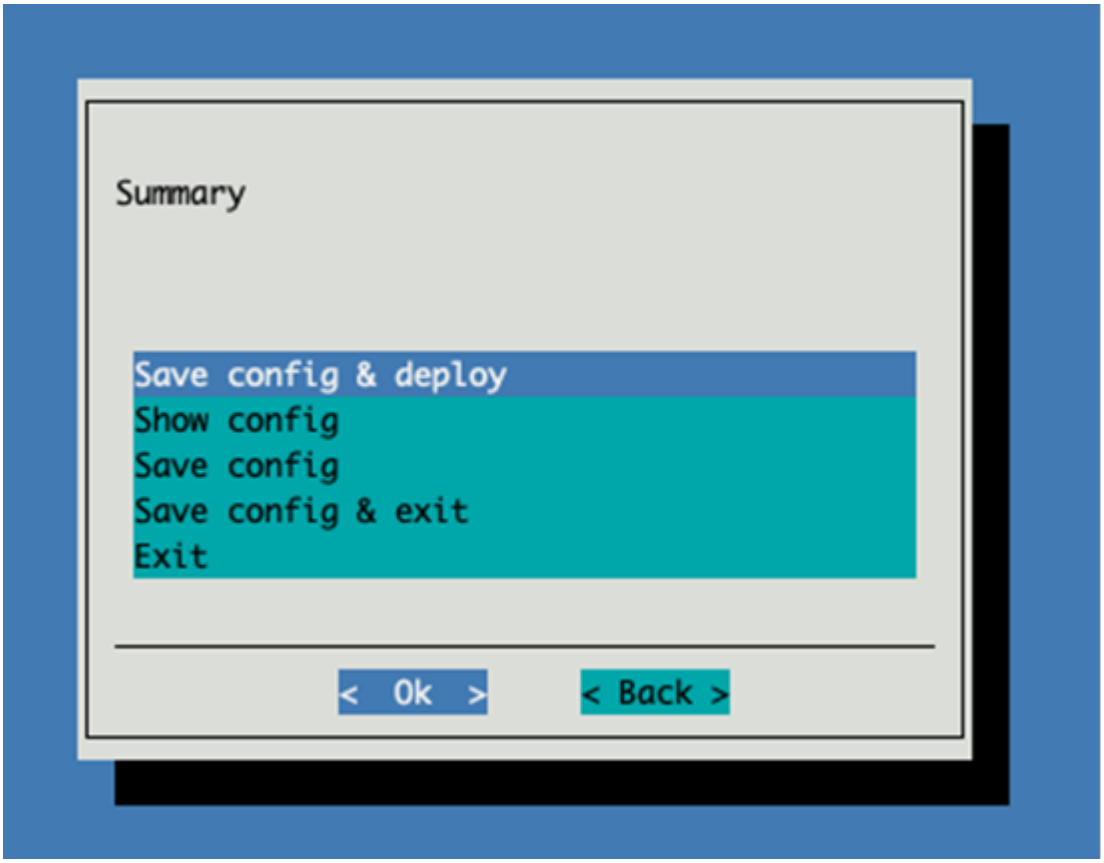
Select both enabled and default for the local storage path.



Leave the storage path as default.



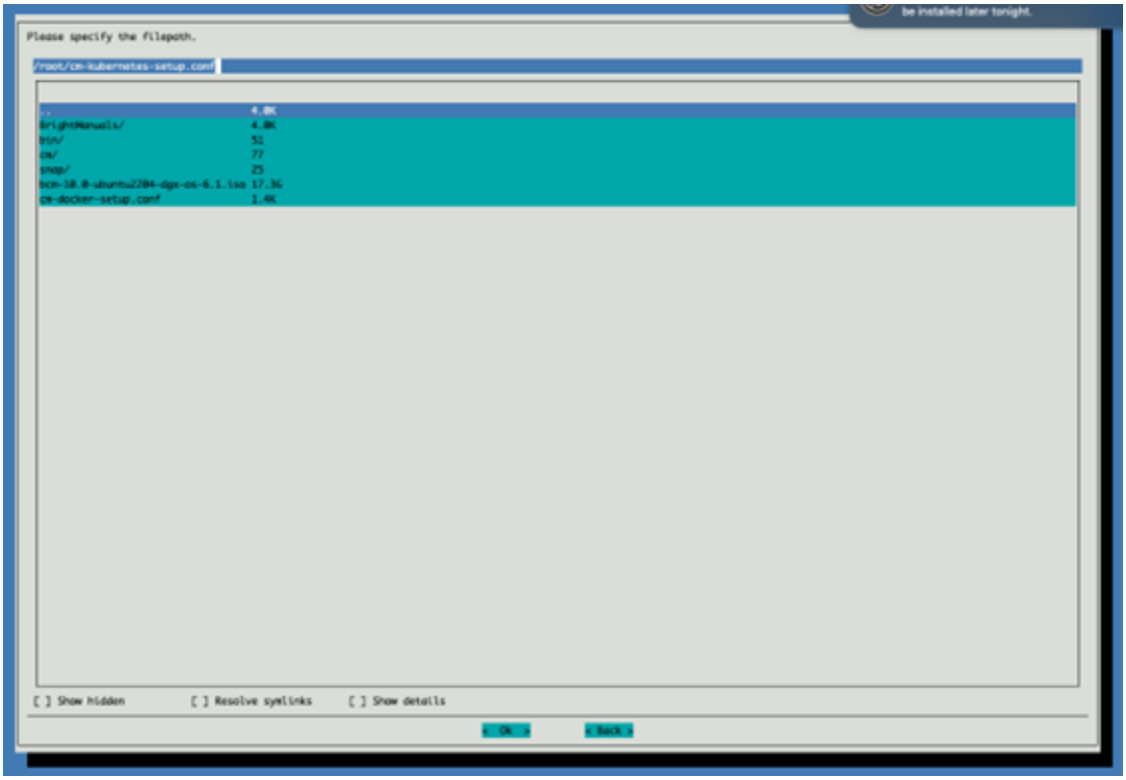
Choose to save config and deploy.



Keep the default file path for the config file and continue.

Note

This file contains the configuration for the kubernetes cluster.



BCM will deploy kubernetes to the nodes, once the Kubernetes setup has completed, verify that all the nodes are online.

```
root@bcm10-headnode1:~# kubectl get node -o wide
NAME STATUS ROLES AGE VERSION INTERNAL-IP EXTERNAL-IP OS-IMAGE KERNEL-VERSION
 ↪CONTAINER-RUNTIME
dgx-01 Ready worker 14d v1.30.9 10.184.94.11 <none> Ubuntu 22.04.4 LTS 5.15.0-
 ↪1063-nvidia containerd://1.7.21
dgx-02 Ready worker 14d v1.30.9 10.184.94.12 <none> Ubuntu 22.04.4 LTS 5.15.0-
 ↪1063-nvidia containerd://1.7.21
dgx-03 Ready worker 14d v1.30.9 10.184.94.13 <none> Ubuntu 22.04.4 LTS 5.15.0-
 ↪1063-nvidia containerd://1.7.21
dgx-04 Ready worker 14d v1.30.9 10.184.94.14 <none> Ubuntu 22.04.4 LTS 5.15.0-
 ↪1063-nvidia containerd://1.7.21
knode1 Ready control-plane,master 14d v1.30.9 10.184.94.4 <none> Ubuntu 22.04.
 ↪4 LTS 5.15.0-113-generic containerd://1.7.21
knode2 Ready control-plane,master 14d v1.30.9 10.184.94.5 <none> Ubuntu 22.04.
 ↪4 LTS 5.15.0-113-generic containerd://1.7.21
knode3 Ready control-plane,master 14d v1.30.9 10.184.94.6 <none> Ubuntu 22.04.
 ↪4 LTS 5.15.0-113-generic containerd://1.7.21
```

Validate the Kubernetes cluster by checking that the pods are in the “Running” state and ensuring that both the GPU operator and network operator pods are active.

```
root@bcm10-headnode1:~$ kubectl get pods -A | grep "network\|gpu"
gpu-operator gpu-feature-discovery-7l4bz 1/1 Running 0 13h
gpu-operator gpu-feature-discovery-bpzzq 1/1 Running 0 13h
– Output removed for brevity –
network-operator cni-plugins-ds-gx97k 1/1 Running 0 13h
```

(continues on next page)

(continued from previous page)

```
network-operator cni-plugins-ds-hw6sr 1/1 Running 0 13h
network-operator kube-multus-ds-kkbwz 1/1 Running 0 13h
```

14.3. Add a kubernetes user

Add a new user named ‘k8suser’ in BCM

```
root@bcm10-headnode1:~# cmsh
[bcm10-headnode1]% user;list
Name (key) ID (key) Primary group Secondary groups

------------------------------------------------------

cmsupport 1000      cmsupport
[bcm10-headnode1]% user
[bcm10-headnode1->user]% add k8suser
[bcm10-headnode1->user*[k8suser*]]% set password bcm123
[bcm10-headnode1->user*[k8suser*]]% commit
```

Add a new user to Kubernetes:

```
cm-kubernetes-setup --add-user k8suser
```

Switch to the new user:

```
su - k8suser
```

Chapter 15. Compute network/ IB Interfaces Configuration

15.1. Validate IB/Compute interfaces

Reference: Configuring SR-IOV (InfiniBand) section in OFED user guide

Check Physical link type (LINK_TYPE_P1), SRIOV(SRIOV_EN) state and VF count (NUM_OF_VF) configuration for the DGX Compute interfaces.

Physical link should be type 1 for InfiniBand, SRIOV in enabled state with 8 VFs

On the BCM headnode, run cmsh

```
root@bcm10-headnode1:~# cmsh
[bcm10-headnode1]% device
[bcm10-headnode1->device]%
[bcm10-headnode1->device]%pexec -c dgx-h100 -j "for i in dc 9a ce c0 4f 40 5e
↪18 ; do mst start; mlxconfig -d $i:00.0 q; done | grep -e \"SRIOV_EN\\|LINK_
↪TYPE\\|NUM_OF_VFS\";"
[dgx-01..dgx-04]
NUM_OF_VFS 8
SRIOV_EN True(1)
LINK_TYPE_P1 IB(1)
NUM_OF_VFS 8
SRIOV_EN True(1)
LINK_TYPE_P1 IB(1)
NUM_OF_VFS 8
SRIOV_EN True(1)
LINK_TYPE_P1 IB(1)
NUM_OF_VFS 8
SRIOV_EN True(1)
LINK_TYPE_P1 IB(1)
NUM_OF_VFS 8
SRIOV_EN True(1)
LINK_TYPE_P1 IB(1)
NUM_OF_VFS 8
SRIOV_EN True(1)
LINK_TYPE_P1 IB(1)
NUM_OF_VFS 8
SRIOV_EN True(1)
```

(continues on next page)

(continued from previous page)

```
LINK_TYPE_P1 IB(1)
NUM_OF_VFS 8
SRIOV_EN True(1)
LINK_TYPE_P1 IB(1)
```

Refer to DGX H100 user guide for interface name/PCI address mapping

If SR-IOV is enabled, the interface type is InfiniBand, and eight or more VFs are already configured, proceed to section Configure SR-IOV NetworkNodePolicy CR

If SRIOV/IB/VFs are not configured, enable them using the following command

```
[bcm10-headnode1->device]% pexec -c dgx-h100 -j "for i in dc 9a ce c0 4f 40
↪5e 18 ; do mst start; mlxconfig -d $i:00.0 -y set SRIOV_EN=1 NUM_OF_VFS=8
↪LINK_TYPE_P1=1 ; done"
Device #1:
----------
Device type: ConnectX7
Name: MCX750500B-0D00_Ax
Description: Nvidia adapter card with four ConnectX-7; each 400Gb/s NDR
IB; PCIe 5.0 x32; PCIe switch; secured boot; No Crypto
Device: 18:00.0
Configurations: Next Boot New
SRIOV_EN True(1) True(1)
NUM_OF_VFS 8 8
LINK_TYPE_P1 IB(1) IB(1)
Apply new Configuration? (y/n) [n] : y
Applying... Done!

-I- Please reboot the machine to load new configurations.
```

Reboot the DGX nodes from BCM to apply the changes.

```
[bcm10-headnode1->device]% reboot -c dgx-h100
```

Wait for the nodes to come back up

```
[bcm10-headnode1->device]% list -c dgx-h100
Type Hostname (key) MAC Category IP Network Status
-------------------------------------------------------
PhysicalNode dgx-01 5A:9F:F8:65:70:C4 dgx-h100 10.184.94.11 managementnet [
↪UP ]
PhysicalNode dgx-02 1E:BB:21:13:FF:60 dgx-h100 10.184.94.12 managementnet [
↪UP ]
PhysicalNode dgx-03 AA:4C:1C:14:84:1F dgx-h100 10.184.94.13 managementnet [
↪UP ]
PhysicalNode dgx-04 06:9E:B9:10:E5:DD dgx-h100 10.184.94.14 managementnet [
↪UP ]
```

Configure 8 VFs per IB interface

```
[bcm10-headnode1->device] pexec -c dgx-h100 -j "for i in 0 3 4 5 6 9 10 11;
↪do echo 8 > /sys/class/infiniband/mlx5_${i}/device/sriov_numvfs; done"
[dgx-01..dgx-04]
```

Check IB Interface status on the DGX nodes.

```
[bcm10-headnode1->device]% pexec -c dgx-h100 -j "for i in 0 3 4 5 6 9 10 11;
↪do ibstat -d mlx5_${i} \| grep -i \\"mlx5_\\|state\\|infiniband\"; done"
[dgx-01..dgx-04]
CA 'mlx5_0'
State: Active
Physical state: LinkUp
Link layer: InfiniBand
CA 'mlx5_3'
State: Active
Physical state: LinkUp
Link layer: InfiniBand
CA 'mlx5_4'
State: Active
Physical state: LinkUp
Link layer: InfiniBand
CA 'mlx5_5'
State: Active
Physical state: LinkUp
Link layer: InfiniBand
CA 'mlx5_6'
State: Active
Physical state: LinkUp
Link layer: InfiniBand
CA 'mlx5_9'
State: Active
Physical state: LinkUp
Link layer: InfiniBand
CA 'mlx5_10'
State: Active
Physical state: LinkUp
Link layer: InfiniBand
CA 'mlx5_11'
State: Active
Physical state: LinkUp
Link layer: InfiniBand
```

All interfaces should be in Active and physical layers in LinkUp state.

Verify VFs under each IB interface PCI device, for e.g. 18:00.0 to 18:00.7 for OSFP4, Port 2, mlx5_0

```
[bcm10-headnode1->device]% pexec -c dgx-h100 -j "lspci \| grep ConnectX"
[dgx-01..dgx-04]
16:00.0 PCI bridge: Mellanox Technologies MT2910 Family [ConnectX-7 PCIe
Bridge]
17:00.0 PCI bridge: Mellanox Technologies MT2910 Family [ConnectX-7 PCIe
Bridge]
17:02.0 PCI bridge: Mellanox Technologies MT2910 Family [ConnectX-7 PCIe
Bridge]
18:00.0 Infiniband controller: Mellanox Technologies MT2910 Family
[ConnectX-7]
18:00.1 Infiniband controller: Mellanox Technologies ConnectX Family
mlx5Gen Virtual Function
```

(continues on next page)

(continued from previous page)

```
18:00.2 Infiniband controller: Mellanox Technologies ConnectX Family
mlx5Gen Virtual Function
18:00.3 Infiniband controller: Mellanox Technologies ConnectX Family
mlx5Gen Virtual Function
18:00.4 Infiniband controller: Mellanox Technologies ConnectX Family
mlx5Gen Virtual Function
18:00.5 Infiniband controller: Mellanox Technologies ConnectX Family
mlx5Gen Virtual Function
18:00.6 Infiniband controller: Mellanox Technologies ConnectX Family
mlx5Gen Virtual Function
18:00.7 Infiniband controller: Mellanox Technologies ConnectX Family
mlx5Gen Virtual Function
18:01.0 Infiniband controller: Mellanox Technologies ConnectX Family
mlx5Gen Virtual Function
```

15.2. Configure SR-IOV NetworkNodePolicy CR

Create a file named 'sriov-ib-network-node-policy.yaml' with the following information:

```
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetworkNodePolicy
metadata:
  name: ibp24s0
  namespace: network-operator
spec:
  deviceType: netdevice
  nodeSelector:
    feature.node.kubernetes.io/network-sriov.capable: "true"
  nicSelector:
    vendor: "15b3"
    pfNames: ["ibp24s0"]
  linkType: ib
  isRdma: true
  numVfs: 8
  priority: 90
  resourceName: resibp24s0

---
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetworkNodePolicy
metadata:
  name: ibp64s0
  namespace: network-operator
spec:
  deviceType: netdevice
  nodeSelector:
    feature.node.kubernetes.io/network-sriov.capable: "true"
  nicSelector:
    vendor: "15b3"
    pfNames: ["ibp64s0"]
```

(continues on next page)

(continued from previous page)

```
  linkType: ib
  isRdma: true
  numVfs: 8
  priority: 90
  resourceName: resibp64s0

---
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetworkNodePolicy
metadata:
  name: ibp79s0
  namespace: network-operator
spec:
  deviceType: netdevice
  nodeSelector:
    feature.node.kubernetes.io/network-sriov.capable: "true"
  nicSelector:
    vendor: "15b3"
    pfNames: ["ibp79s0"]
  linkType: ib
  isRdma: true
  numVfs: 8
  priority: 90
  resourceName: resibp79s0

---
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetworkNodePolicy
metadata:
  name: ibp94s0
  namespace: network-operator
spec:
  deviceType: netdevice
  nodeSelector:
    feature.node.kubernetes.io/network-sriov.capable: "true"
  nicSelector:
    vendor: "15b3"
    pfNames: ["ibp94s0"]
  linkType: ib
  isRdma: true
  numVfs: 8
  priority: 90
  resourceName: resibp94s0

---
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetworkNodePolicy
metadata:
  name: ibp154s0
  namespace: network-operator
spec:
  deviceType: netdevice
```

(continues on next page)

(continued from previous page)

```
  nodeSelector:
    feature.node.kubernetes.io/network-sriov.capable: "true"
  nicSelector:
    vendor: "15b3"
    pfNames: ["ibp154s0"]
  linkType: ib
  isRdma: true
  numVfs: 8
  priority: 90
  resourceName: resibp154s0

---
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetworkNodePolicy
metadata:
  name: ibp192s0
  namespace: network-operator
spec:
  deviceType: netdevice
  nodeSelector:
    feature.node.kubernetes.io/network-sriov.capable: "true"
  nicSelector:
    vendor: "15b3"
    pfNames: ["ibp192s0"]
  linkType: ib
  isRdma: true
  numVfs: 8
  priority: 90
  resourceName: resibp192s0

---
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetworkNodePolicy
metadata:
  name: ibp206s0
  namespace: network-operator
spec:
  deviceType: netdevice
  nodeSelector:
    feature.node.kubernetes.io/network-sriov.capable: "true"
  nicSelector:
    vendor: "15b3"
    pfNames: ["ibp206s0"]
  linkType: ib
  isRdma: true
  numVfs: 8
  priority: 90
  resourceName: resibp206s0

---
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetworkNodePolicy
```

(continues on next page)

(continued from previous page)

```
metadata:
  name: ibp220s0
  namespace: network-operator
spec:
  deviceType: netdevice
  nodeSelector:
    feature.node.kubernetes.io/network-sriov.capable: "true"
  nicSelector:
    vendor: "15b3"
    pfNames: ["ibp220s0"]
  linkType: ib
  isRdma: true
  numVfs: 8
  priority: 90
  resourceName: resibp220s0

---
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetworkNodePolicy
metadata:
  name: ibp24s0
  namespace: network-operator
spec:
  deviceType: netdevice
  nodeSelector:
    feature.node.kubernetes.io/network-sriov.capable: "true"
  nicSelector:
    vendor: "15b3"
    pfNames: ["ibp24s0"]
  linkType: ib
  isRdma: true
  numVfs: 8
  priority: 90
  resourceName: resibp24s0

---
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetworkNodePolicy
metadata:
  name: ibp64s0
  namespace: network-operator
spec:
  deviceType: netdevice
  nodeSelector:
    feature.node.kubernetes.io/network-sriov.capable: "true"
  nicSelector:
    vendor: "15b3"
    pfNames: ["ibp64s0"]
  linkType: ib
  isRdma: true
  numVfs: 8
  priority: 90
```

(continues on next page)

(continued from previous page)

```
  resourceName: resibp64s0

---
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetworkNodePolicy
metadata:
  name: ibp79s0
  namespace: network-operator
spec:
  deviceType: netdevice
  nodeSelector:
    feature.node.kubernetes.io/network-sriov.capable: "true"
  nicSelector:
    vendor: "15b3"
    pfNames: ["ibp79s0"]
  linkType: ib
  isRdma: true
  numVfs: 8
  priority: 90
  resourceName: resibp79s0

---
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetworkNodePolicy
metadata:
  name: ibp94s0
  namespace: network-operator
spec:
  deviceType: netdevice
  nodeSelector:
    feature.node.kubernetes.io/network-sriov.capable: "true"
  nicSelector:
    vendor: "15b3"
    pfNames: ["ibp94s0"]
  linkType: ib
  isRdma: true
  numVfs: 8
  priority: 90
  resourceName: resibp94s0

---
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetworkNodePolicy
metadata:
  name: ibp154s0
  namespace: network-operator
spec:
  deviceType: netdevice
  nodeSelector:
    feature.node.kubernetes.io/network-sriov.capable: "true"
  nicSelector:
    vendor: "15b3"
```

(continues on next page)

(continued from previous page)

```
    pfNames: ["ibp154s0"]
  linkType: ib
  isRdma: true
  numVfs: 8
  priority: 90
  resourceName: resibp154s0

---
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetworkNodePolicy
metadata:
  name: ibp192s0
  namespace: network-operator
spec:
  deviceType: netdevice
  nodeSelector:
    feature.node.kubernetes.io/network-sriov.capable: "true"
  nicSelector:
    vendor: "15b3"
    pfNames: ["ibp192s0"]
  linkType: ib
  isRdma: true
  numVfs: 8
  priority: 90
  resourceName: resibp192s0

---
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetworkNodePolicy
metadata:
  name: ibp206s0
  namespace: network-operator
spec:
  deviceType: netdevice
  nodeSelector:
    feature.node.kubernetes.io/network-sriov.capable: "true"
  nicSelector:
    vendor: "15b3"
    pfNames: ["ibp206s0"]
  linkType: ib
  isRdma: true
  numVfs: 8
  priority: 90
  resourceName: resibp206s0

---
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetworkNodePolicy
metadata:
  name: ibp220s0
  namespace: network-operator
spec:
```

(continues on next page)

(continued from previous page)

```
  deviceType: netdevice
  nodeSelector:
    feature.node.kubernetes.io/network-sriov.capable: "true"
  nicSelector:
    vendor: "15b3"
    pfNames: ["ibp220s0"]
  linkType: ib
  isRdma: true
  numVfs: 8
  priority: 90
  resourceName: resibp220s0
```

Create the CRD using

```
kubectl apply –f sriov-ib-network-node-policy.yaml
```

Create SR-IOV IB Network CR

Create another file named ‘sriov-ib-network.yaml’ with the following information:

```
---
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovIBNetwork
metadata:
  name: ibp24s0
  namespace: network-operator
spec:
  ipam: |
    {
      "type": "whereabouts",
      "datastore": "kubernetes",
      "kubernetes": {
        "kubeconfig": "/etc/cni/net.d/whereabouts.d/whereabouts.kubeconfig"
      },
      "range": "192.168.1.0/24",
      "log_file": "/var/log/whereabouts.log",
      "log_level": "info"
    }
  resourceName: resibp24s0
  linkState: enable
  networkNamespace: default

---
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovIBNetwork
metadata:
  name: ibp64s0
  namespace: network-operator
spec:
  ipam: |
    {
      "type": "whereabouts",
      "datastore": "kubernetes",
```

(continues on next page)

(continued from previous page)

```
      "kubernetes": {
        "kubeconfig": "/etc/cni/net.d/whereabouts.d/whereabouts.kubeconfig"
      },
      "range": "192.168.2.0/24",
      "log_file": "/var/log/whereabouts.log",
      "log_level": "info"
    }
  resourceName: resibp64s0
  linkState: enable
  networkNamespace: default

---
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovIBNetwork
metadata:
  name: ibp79s0
  namespace: network-operator
spec:
  ipam: |
    {
      "type": "whereabouts",
      "datastore": "kubernetes",
      "kubernetes": {
        "kubeconfig": "/etc/cni/net.d/whereabouts.d/whereabouts.kubeconfig"
      },
      "range": "192.168.3.0/24",
      "log_file": "/var/log/whereabouts.log",
      "log_level": "info"
    }
  resourceName: resibp79s0
  linkState: enable
  networkNamespace: default

---
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovIBNetwork
metadata:
  name: ibp94s0
  namespace: network-operator
spec:
  ipam: |
    {
      "type": "whereabouts",
      "datastore": "kubernetes",
      "kubernetes": {
        "kubeconfig": "/etc/cni/net.d/whereabouts.d/whereabouts.kubeconfig"
      },
      "range": "192.168.4.0/24",
      "log_file": "/var/log/whereabouts.log",
      "log_level": "info"
    }
  resourceName: resibp94s0
```

(continues on next page)

(continued from previous page)

```
  linkState: enable
  networkNamespace: default

---
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovIBNetwork
metadata:
  name: ibp154s0
  namespace: network-operator
spec:
  ipam: |
    {
      "type": "whereabouts",
      "datastore": "kubernetes",
      "kubernetes": {
        "kubeconfig": "/etc/cni/net.d/whereabouts.d/whereabouts.kubeconfig"
      },
      "range": "192.168.5.0/24",
      "log_file": "/var/log/whereabouts.log",
      "log_level": "info"
    }
  resourceName: resibp154s0
  linkState: enable
  networkNamespace: default

---
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovIBNetwork
metadata:
  name: ibp192s0
  namespace: network-operator
spec:
  ipam: |
    {
      "type": "whereabouts",
      "datastore": "kubernetes",
      "kubernetes": {
        "kubeconfig": "/etc/cni/net.d/whereabouts.d/whereabouts.kubeconfig"
      },
      "range": "192.168.6.0/24",
      "log_file": "/var/log/whereabouts.log",
      "log_level": "info"
    }
  resourceName: resibp192s0
  linkState: enable
  networkNamespace: default

---
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovIBNetwork
metadata:
  name: ibp206s0
```

(continues on next page)

(continued from previous page)

```
  namespace: network-operator
spec:
  ipam: |
    {
      "type": "whereabouts",
      "datastore": "kubernetes",
      "kubernetes": {
        "kubeconfig": "/etc/cni/net.d/whereabouts.d/whereabouts.kubeconfig"
      },
      "range": "192.168.7.0/24",
      "log_file": "/var/log/whereabouts.log",
      "log_level": "info"
    }
  resourceName: resibp206s0
  linkState: enable
  networkNamespace: default

---
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovIBNetwork
metadata:
  name: ibp220s0
  namespace: network-operator
spec:
  ipam: |
    {
      "type": "whereabouts",
      "datastore": "kubernetes",
      "kubernetes": {
        "kubeconfig": "/etc/cni/net.d/whereabouts.d/whereabouts.kubeconfig"
      },
      "range": "192.168.8.0/24",
      "log_file": "/var/log/whereabouts.log",
      "log_level": "info"
    }
  resourceName: resibp220s0
  linkState: enable
  networkNamespace: default
```

Apply the CRD using

```
kubectl apply –f sriov-ib-network.yaml
```

Restart the services to apply the changes:

```
pdsh -g category=k8s-control-plane service containerd restart
pdsh -g category=k8s-control-plane service kubelet restart
```

Chapter 16. Validate the GPU status/health

Using cmsh run the following command

```
[bcm10-headnode1->device]% pexec -c dgx-h100 "nvsm show gpus \| grep -e \\
↪"GPU.\" -e \\"Health\""
[dgx-01] :
/systems/localhost/gpus/GPU0
Inventory_UUID = GPU-3a713db1-ff94-3f28-de11-2d6449b8d35f
Stats_UtilGPU = 0%
Status_Health = OK
/systems/localhost/gpus/GPU0/health
Health = OK
/systems/localhost/gpus/GPU1
Inventory_UUID = GPU-bc00e4ef-fb74-10c8-fa6a-a8c243304e14
Stats_UtilGPU = 0%
Status_Health = OK
/systems/localhost/gpus/GPU1/health
Health = OK
/systems/localhost/gpus/GPU2
Inventory_UUID = GPU-4f2e5158-c8be-b50e-2d99-e5380b4a8236
Stats_UtilGPU = 0%
Status_Health = OK
/systems/localhost/gpus/GPU2/health
Health = OK
/systems/localhost/gpus/GPU3
Inventory_UUID = GPU-0a8a7416-4c04-5038-19ca-345db5a1a0ad
Stats_UtilGPU = 0%
Status_Health = OK
/systems/localhost/gpus/GPU3/health
Health = OK
/systems/localhost/gpus/GPU4
Inventory_UUID = GPU-bbd31c7c-7845-d48a-df3b-1523adf86ee6
Stats_UtilGPU = 0%
Status_Health = OK
/systems/localhost/gpus/GPU4/health
Health = OK
/systems/localhost/gpus/GPU5

 output omitted for brevity
```

Ensure all GPUs are healthy.

Chapter 17. Validate the system topology/NVlink

Using cmsh run the following command

```
root@bcm10-headnode1:~# cmsh
[bcm10-headnode1]%device
[bcm10-headnode1]% pexec -c dgx-h100 -j "nvidia-smi topo -m"
bcm10-headnode1->device]% pexec -c dgx-h100 -j "nvidia-smi topo -m"
[dgx-01..dgx-04]
76 NICs found in the topology, only displaying 56 in the matrix.
            GPU0    GPU1    GPU2    GPU3    GPU4    GPU5    GPU6    GPU7
 ↪NIC0    NIC1    NIC2    NIC3    NIC4    NIC5    NIC6    NIC7    NIC8
 ↪NIC9    NIC10   NIC11   NIC12   NIC13   NIC14   NIC15   NIC16   NIC17
 ↪NIC18   NIC19   NIC20   NIC21   NIC22   NIC23   NIC24   NIC25   NIC26
 ↪NIC27   NIC28   NIC29   NIC30   NIC31   NIC32   NIC33   NIC34   NIC35
 ↪NIC36   NIC37   NIC38   NIC39   NIC40   NIC41   NIC42   NIC43   NIC44
 ↪NIC45   NIC46   NIC47   NIC48   NIC49   NIC50   NIC51   NIC52   NIC53
 ↪NIC54   NIC55   CPU Affinity    NUMA Affinity   GPU NUMA ID
GPU0         X          NV18        NV18    NV18    NV18    NV18    NV18
 ↪NV18    PXB     NODE    NODE    NODE    NODE    NODE    SYS     SYS     SYS
 ↪    SYS     PXB     PXB     PXB     PXB     PXB     PXB     PXB     PXB
 ↪NODE    NODE    NODE    NODE    NODE    NODE    NODE    NODE    NODE
 ↪NODE    NODE    NODE    NODE    NODE    NODE    NODE    NODE    NODE
 ↪NODE    NODE    NODE    NODE    NODE    NODE    SYS     SYS     SYS     SYS
 ↪    SYS     SYS     SYS     SYS     SYS     SYS     SYS     SYS     SYS
 ↪SYS     0-55,112-167    0               N/A
GPU1        NV18     X          NV18        NV18    NV18    NV18    NV18
 ↪NV18    NODE    NODE    NODE    PXB     NODE    NODE    SYS     SYS     SYS
 ↪    SYS     NODE    NODE    NODE    NODE    NODE    NODE    NODE    NODE
 ↪PXB     PXB     PXB     PXB     PXB     PXB     PXB     PXB     NODE
 ↪NODE    NODE    NODE    NODE    NODE    NODE    NODE    NODE    NODE
 ↪NODE    NODE    NODE    NODE    NODE    NODE    SYS     SYS     SYS     SYS
 ↪    SYS     SYS     SYS     SYS     SYS     SYS     SYS     SYS     SYS
 ↪SYS     0-55,112-167    0               N/A
GPU2        NV18    NV18     X          NV18        NV18    NV18    NV18
 ↪NV18    NODE    NODE    NODE    NODE    PXB     NODE    SYS     SYS     SYS
 ↪    SYS     NODE    NODE    NODE    NODE    NODE    NODE    NODE    NODE
 ↪NODE    NODE    NODE    NODE    NODE    NODE    NODE    NODE    PXB     PXB
 ↪    PXB     PXB     PXB     PXB     PXB     PXB     NODE    NODE    NODE
 ↪NODE    NODE    NODE    NODE    NODE    SYS     SYS     SYS     SYS     SYS
```

(continues on next page)

(continued from previous page)

```
↪     SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS
↪0-55,112-167    0               N/A
GPU3         NV18     NV18     NV18      X          NV18         NV18     NV18
↪NV18     NODE     NODE     NODE     NODE     NODE     PXB      SYS      SYS      SYS
↪     SYS      NODE     NODE     NODE     NODE     NODE     NODE     NODE     NODE
↪NODE     NODE     NODE     NODE     NODE     NODE     NODE     NODE     NODE
↪NODE     NODE     NODE     NODE     NODE     NODE     NODE     PXB      PXB      PXB
↪     PXB      PXB      PXB      PXB      PXB      SYS      SYS      SYS      SYS
↪SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS
↪     0-55,112-167    0               N/A
GPU4         NV18     NV18     NV18     NV18      X          NV18         NV18
↪NV18     SYS      SYS      SYS      SYS      SYS      SYS      PXB      NODE
↪NODE     NODE     SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS
↪     SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS
↪SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS
↪     SYS      SYS      SYS      SYS      SYS      PXB      PXB      PXB      PXB
↪PXB      PXB      PXB      PXB      NODE     NODE     NODE     NODE     NODE
↪NODE     56-111,168-223  1               N/A
GPU5         NV18     NV18     NV18     NV18     NV18      X          NV18
↪NV18     SYS      SYS      SYS      SYS      SYS      SYS      NODE     NODE
↪NODE     PXB      SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS
↪     SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS
↪SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS
↪     SYS      SYS      SYS      SYS      SYS      NODE     NODE     NODE     NODE
↪NODE     NODE     NODE     NODE     PXB      PXB      PXB      PXB      PXB      PXB
↪     56-111,168-223  1               N/A
GPU6         NV18     NV18     NV18     NV18     NV18     NV18      X          NV18
↪       SYS      SYS      SYS      SYS      SYS      SYS      NODE     NODE     NODE
↪   NODE     SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS
↪SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS
↪     SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS
↪SYS      SYS      SYS      SYS      SYS      NODE     NODE     NODE     NODE
↪NODE     NODE     NODE     NODE     NODE     NODE     NODE     NODE     NODE
↪NODE     56-111,168-223  1               N/A
GPU7         NV18     NV18     NV18     NV18     NV18     NV18     NV18      X
↪   SYS SYS      SYS      SYS      SYS      SYS      NODE     NODE     NODE     NODE
↪   SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS
↪SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS
↪     SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS
↪SYS      SYS      SYS      SYS      NODE     NODE     NODE     NODE     NODE
↪NODE     NODE     NODE     NODE     NODE     NODE     NODE     NODE     NODE     56-
↪111,168-223  1               N/A
NIC0         PXB      NODE     NODE     NODE     SYS      SYS      SYS      SYS
↪X       NODE     NODE     NODE     NODE     NODE     SYS      SYS      SYS      SYS
↪    PIX      PIX      PIX      PIX      PIX      PIX      PIX      PIX      NODE
↪NODE     NODE     NODE     NODE     NODE     NODE     NODE     NODE     NODE
↪NODE     NODE     NODE     NODE     NODE     NODE     NODE     NODE     NODE
↪NODE     NODE     NODE     NODE     NODE     SYS      SYS      SYS      SYS      SYS
↪     SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS      SYS
NIC1         NODE     NODE     NODE     NODE     SYS      SYS      SYS      SYS
↪NODE      X       PIX      NODE     NODE     NODE     SYS      SYS      SYS      SYS
↪     NODE     NODE     NODE     NODE     NODE     NODE     NODE     NODE     NODE
```

(continues on next page)

(continued from previous page)

```
↪NODE     NODE     NODE     NODE     NODE     NODE     NODE     NODE     NODE
↪NODE     NODE     NODE     NODE     NODE     NODE     NODE     NODE     NODE
↪NODE     NODE     NODE     NODE     NODE     SYS      SYS      SYS      SYS      SYS
↪      SYS
```

Verify the NVLink (NV18) status for GPU to GPU interconnect

Reference: Nvidia System Management Interface

Chapter 18. Validate the GPU/RDMA access within the container

18.1. Validate GPU access from container

Create a file named 'gpu-test.yaml' with the following information

```
apiVersion: v1
kind: Pod
metadata:
  name: gpu-test
spec:
  restartPolicy: OnFailure
  containers:
    - name: cuda-pod
      image: nvcr.io/nvidia/cuda:12.6.3-runtime-ubuntu22.04
      imagePullPolicy: IfNotPresent
      command: ["/bin/sh"]
      args: ["-c", "nvidia-smi"]
      resources:
            limits:
            nvidia.com/gpu: 8
```

With the Job file created we can now get ready to execute the job by loading the Kubernetes Module using the following command.

```
module load kubernetes
```

Now that the environment module is loaded we're finally ready to run the example job:

```
kubectl apply -f gpu-test.yaml
```

Next we'll need to monitor the progress of the job using the following command:

```
k8suser@bcm10-headnode1:~$ kubectl get pods
NAME          READY STATUS  RESTARTS AGE
nvidia-smi-test 1/1 Running 0        8m20s
```

Once the job has finished, verify the results using kubectl logs:

The log file should list all the 8 GPUs in the node where the job was executed.

```
k8suser@bcm10-headnode1:~$ kubectl logs nvidia-smi-test
Wed Feb 12 23:49:34 2025
+-----------------------------------------------------------------------------
↪------------+
| NVIDIA-SMI 550.90.07              Driver Version: 550.90.07      CUDA
↪Version: 12.4     |
|-----------------------------------------+------------------------+----------
↪------------+
| 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  NVIDIA H100 80GB HBM3          On  |   00000000:1B:00.0 Off |
↪          0 |
| N/A   27C    P0             69W /  700W |       1MiB /  81559MiB |      0%
↪    Default |
|                                         |                        |
↪   Disabled |
+-----------------------------------------+------------------------+----------
↪------------+
|   1  NVIDIA H100 80GB HBM3          On  |   00000000:43:00.0 Off |
↪          0 |
| N/A   28C    P0             70W /  700W |       1MiB /  81559MiB |      0%
↪    Default |
|                                         |                        |
↪   Disabled |
+-----------------------------------------+------------------------+----------
↪------------+
|   2  NVIDIA H100 80GB HBM3          On  |   00000000:52:00.0 Off |
↪          0 |
| N/A   31C    P0             69W /  700W |       1MiB /  81559MiB |      0%
↪    Default |
|                                         |                        |
↪   Disabled |
+-----------------------------------------+------------------------+----------
↪------------+
|   3  NVIDIA H100 80GB HBM3          On  |   00000000:61:00.0 Off |
↪          0 |
| N/A   29C    P0             72W /  700W |       1MiB /  81559MiB |      0%
↪    Default |
|                                         |                        |
↪   Disabled |
+-----------------------------------------+------------------------+----------
↪------------+
|   4  NVIDIA H100 80GB HBM3          On  |   00000000:9D:00.0 Off |
↪          0 |
| N/A   28C    P0             69W /  700W |       1MiB /  81559MiB |      0%
↪    Default |
|                                         |                        |
```

(continues on next page)

(continued from previous page)

```
↪   Disabled |
+-----------------------------------------+------------------------+----------
↪------------+
|   5  NVIDIA H100 80GB HBM3          On  |   00000000:C3:00.0 Off |
↪           0 |
| N/A   26C    P0             70W /  700W |       1MiB /  81559MiB |      0%
↪    Default |
|                                         |                        |
↪   Disabled |
+-----------------------------------------+------------------------+----------
↪------------+
|   6  NVIDIA H100 80GB HBM3          On  |   00000000:D1:00.0 Off |
↪           0 |
| N/A   29C    P0             69W /  700W |       1MiB /  81559MiB |      0%
↪    Default |
|                                         |                        |
↪   Disabled |
+-----------------------------------------+------------------------+----------
↪------------+
|   7  NVIDIA H100 80GB HBM3          On  |   00000000:DF:00.0 Off |
↪           0 |
| N/A   29C    P0             68W /  700W |       1MiB /  81559MiB |      0%
↪    Default |
|                                         |                        |
↪   Disabled |
+-----------------------------------------+------------------------+----------
↪------------+
```

Reference: Nvidia System Management Interface

18.2. Validate the RDMA Network from the container

Create a test file named 'ib-network-validation.yaml' with the following information:

```
apiVersion: v1
kind: Pod
metadata:
  name: network-validation-pod
  annotations:
    k8s.v1.cni.cncf.io/networks: ibp192s0,ibp206s0,ibp154s0,ibp220s0,ibp24s0,
↪ibp64s0,ibp79s0,ibp94s0
spec:
  containers:
    - name: network-validation-pod
      image: docker.io/deepops/nccl-tests:latest
      imagePullPolicy: IfNotPresent
      command:
        - sh
```

(continues on next page)

(continued from previous page)

```
      - -c
      - sleep inf
    securityContext:
      capabilities:
        add: ["IPC_LOCK"]
    resources:
      requests:
          nvidia.com/resibp192s0: "1"
          nvidia.com/resibp206s0: "1"
          nvidia.com/resibp154s0: "1"
          nvidia.com/resibp220s0: "1"
          nvidia.com/resibp24s0: "1"
          nvidia.com/resibp64s0: "1"
          nvidia.com/resibp79s0: "1"
          nvidia.com/resibp94s0: "1"
      limits:
        nvidia.com/resibp192s0: "1"
        nvidia.com/resibp206s0: "1"
        nvidia.com/resibp154s0: "1"
        nvidia.com/resibp220s0: "1"
        nvidia.com/resibp24s0: "1"
        nvidia.com/resibp64s0: "1"
        nvidia.com/resibp79s0: "1"
        nvidia.com/resibp94s0: "1"
```

Apply the 'ib-network-validation.yaml' file:

```
kubectl apply –f ib-network-validation.yaml
```

Confirm the pod is in running state

```
k8suser@bcm10-headnode1:~$ k get pods
NAME READY STATUS RESTARTS AGE
network-validation-pod 1/1 Running 0 5s
```

Verify the InfiniBand/RDMA interfaces are available in the container

```
root@bcm10-headnode1:~# kubectl exec -it network-validation-pod --
/usr/sbin/ibdev2netdev
mlx5_14 port 1 ==> net5 (Up)
mlx5_25 port 1 ==> net6 (Up)
mlx5_33 port 1 ==> net7 (Up)
mlx5_43 port 1 ==> net8 (Up)
mlx5_49 port 1 ==> net3 (Up)
mlx5_58 port 1 ==> net1 (Up)
mlx5_61 port 1 ==> net2 (Up)
mlx5_70 port 1 ==> net4 (Up)
```

Delete the test pod

```
root@bcm10-headnode1:~# kubectl delete pod network-validation-pod

pod "network-validation-pod" deleted
```

Chapter 19. Validate the node level NCCL test with 8 GPUs

Create a yaml named nccl-local.yaml with the following content

```
apiVersion: v1
kind: Pod
metadata:
  name: nccl-local-test
spec:
  restartPolicy: Never
  containers:
    - name: nvidia-smi
      image: docker.io/deepops/nccl-tests:2312
      command: ["/bin/bash", "-c", "nvidia-smi && tail -f /dev/null"]
      resources:
        limits:
          nvidia.com/gpu: 8  # Request 8 GPUs
  nodeSelector:
    nvidia.com/gpu.present: "true"  # Ensure it runs on a node with a GPU
```

Start the container with

```
kubectl apply -f nccl-local.yaml
```

Verify the container is running

```
k8suser@bcm10-headnode1:~$ kubectl get pods
NAME READY STATUS RESTARTS AGE
nccl-local-test 1/1 Running 0 17s
```

Login to the container and run NCCL broadcast perf with 8 GPUs

Reference: NCCL primitives

Reference: NCCL Tests

```
root@nccl-local-test:/workspace# broadcast_perf -b 64k -e 2G  -f 2  -g 8
# nThread 1 nGpus 8 minBytes 65536 maxBytes 2147483648 step: 2(factor) warmup
 ↪iters: 5 iters: 20 agg iters: 1 validation: 1 graph: 0
#
# Using devices
#  Rank  0 Group  0 Pid    323 on nccl-local-test device  0 [0x1b] NVIDIA
```

(continues on next page)

(continued from previous page)

```
↪H100 80GB HBM3
#  Rank  1 Group  0 Pid    323 on nccl-local-test device  1 [0x43] NVIDIA
↪H100 80GB HBM3
#  Rank  2 Group  0 Pid    323 on nccl-local-test device  2 [0x52] NVIDIA
↪H100 80GB HBM3
#  Rank  3 Group  0 Pid    323 on nccl-local-test device  3 [0x61] NVIDIA
↪H100 80GB HBM3
#  Rank  4 Group  0 Pid    323 on nccl-local-test device  4 [0x9d] NVIDIA
↪H100 80GB HBM3
#  Rank  5 Group  0 Pid    323 on nccl-local-test device  5 [0xc3] NVIDIA
↪H100 80GB HBM3
#  Rank  6 Group  0 Pid    323 on nccl-local-test device  6 [0xd1] NVIDIA
↪H100 80GB HBM3
#  Rank  7 Group  0 Pid    323 on nccl-local-test device  7 [0xdf] NVIDIA
↪H100 80GB HBM3
#
#                                                              out-of-place
↪                  in-place
#       size         count      type   redop    root     time   algbw   busbw
↪#wrong     time   algbw   busbw #wrong
#        (B)    (elements)                               (us)  (GB/s)  (GB/s)
↪            (us)  (GB/s)  (GB/s)
       65536         16384     float    none       0    28.77    2.28    2.28
↪      0    27.55    2.38    2.38      0
      131072         32768     float    none       0    30.42    4.31    4.31
↪      0    30.28    4.33    4.33      0
      262144         65536     float    none       0    35.31    7.42    7.42
↪      0    33.20    7.90    7.90      0
      524288        131072     float    none       0    42.05   12.47   12.47
↪      0    43.85   11.96   11.96      0
     1048576        262144     float    none       0    53.98   19.42   19.42
↪      0    52.38   20.02   20.02      0
     2097152        524288     float    none       0    54.29   38.63   38.63
↪      0    54.07   38.79   38.79      0
     4194304       1048576     float    none       0    56.67   74.01   74.01
↪      0    55.61   75.43   75.43      0
     8388608       2097152     float    none       0    60.71  138.17  138.17
↪      0    62.26  134.74  134.74      0
    16777216       4194304     float    none       0    82.29  203.88  203.88
↪      0    82.32  203.80  203.80      0
    33554432       8388608     float    none       0    130.4  257.30  257.30
↪      0    130.1  257.98  257.98      0
    67108864      16777216     float    none       0    225.7  297.33  297.33
↪      0    226.5  296.35  296.35      0
   134217728      33554432     float    none       0    411.5  326.17  326.17
↪      0    412.5  325.41  325.41      0
   268435456      67108864     float    none       0    780.8  343.80  343.80
↪      0    783.4  342.64  342.64      0
   536870912     134217728     float    none       0   1508.3  355.95  355.95
↪      0   1513.4  354.74  354.74      0
  1073741824     268435456     float    none       0   2975.8  360.82  360.82
↪      0   2977.4  360.64  360.64      0
```

(continues on next page)

(continued from previous page)

```
  2147483648     536870912     float    none       0   5896.4  364.20  364.20
↪      0   5911.1  363.29  363.29      0
# Out of bounds values : 0 OK
# Avg bus bandwidth    : 175.205
#
root@nccl-local-test:/workspace# exit
exit
```

Chapter 20. Validate the cluster level NCCL test with 4 nodes and 32 GPUs

Create a test file named 'nccl-test.yaml' with the contents shown below. In this example we are running NCCLtest across 4 DGX nodes, over a total of 32 GPUs, so "-np" is set to "4" in the mpirun command.

```
apiVersion: kubeflow.org/v2beta1
kind: MPIJob
metadata:
  name: nccltest
spec:
  slotsPerWorker: 1
  runPolicy:
    cleanPodPolicy: Running
  mpiReplicaSpecs:
    Launcher:
      replicas: 1
      template:
         spec:
           imagePullSecrets:
           - name: ngc-registry-default
           containers:
           - image: docker.io/deepops/nccl-tests:2312
             name: nccltest
             imagePullPolicy: IfNotPresent
             command:
             - sh
             - "-c"
             - |
               /bin/bash << 'EOF'
               mpirun --allow-run-as-root \
                 -np 4 \
                 -bind-to none -map-by slot \
                 -mca pml ob1 \
                 -mca btl ^openib \
                 -mca btl_tcp_if_include 192.168.0.0/16 \
                 -mca oob_tcp_if_include 172.29.0.0/16 \
                 all_reduce_perf_mpi -b 8 -e 16G -f2 -g 8 \
                 && sleep infinity
```

(continues on next page)

(continued from previous page)

```
              EOF
    Worker:
      replicas: 4
      template:
        metadata:
          annotations:
            k8s.v1.cni.cncf.io/networks: ibp192s0,ibp206s0,ibp154s0,ibp220s0,
↪ibp24s0,ibp64s0,ibp79s0,ibp94s0
        spec:
          imagePullSecrets:
          - name: ngc-registry-default
          containers:
          - image: docker.io/deepops/nccl-tests:2312
            name: nccltest
            imagePullPolicy: IfNotPresent
            securityContext:
              capabilities:
                add: [ "IPC_LOCK" ]
            resources:
              requests:
                nvidia.com/resibp192s0: "1"
                nvidia.com/resibp206s0: "1"
                nvidia.com/resibp154s0: "1"
                nvidia.com/resibp220s0: "1"
                nvidia.com/resibp24s0: "1"
                nvidia.com/resibp64s0: "1"
                nvidia.com/resibp79s0: "1"
                nvidia.com/resibp94s0: "1"
                nvidia.com/gpu: 8
              limits:
                nvidia.com/resibp192s0: "1"
                nvidia.com/resibp206s0: "1"
                nvidia.com/resibp154s0: "1"
                nvidia.com/resibp220s0: "1"
                nvidia.com/resibp24s0: "1"
                nvidia.com/resibp64s0: "1"
                nvidia.com/resibp79s0: "1"
                nvidia.com/resibp94s0: "1"
                nvidia.com/gpu: 8
```

Run the 'nccl-test.yaml' file:

```
kubectl apply -f nccl-test.yaml
```

Monitor the progress of the job: Wait till the pods are "Running"

```
kubectl get pods
root@bcm10-headnode1:~# k get pods
NAME READY STATUS RESTARTS AGE
nccltest-launcher-8znll 1/1 Running 3 (54s ago) 87s
nccltest-worker-0 1/1 Running 0 87s
nccltest-worker-1 1/1 Running 0 87s
```

(continues on next page)

(continued from previous page)

```
nccltest-worker-2 1/1 Running 0 87s
nccltest-worker-3 1/1 Running 0 87s
```

Once all are in Running state, (takes ~90 seconds), verify the results:

```
kubectl logs nccltest-launcher-NNNNN
```

For example

```
root@bcm10-headnode1:~#kubectl logs -f nccltest-launcher-8znll
#
# Using devices
#  Rank  0 Group  0 Pid     43 on nccltest-worker-0 device  0 [0x1b] NVIDIA
 ↪H100 80GB HBM3
#  Rank  1 Group  0 Pid     43 on nccltest-worker-0 device  1 [0x43] NVIDIA
 ↪H100 80GB HBM3
#  Rank  2 Group  0 Pid     43 on nccltest-worker-0 device  2 [0x52] NVIDIA
 ↪H100 80GB HBM3
#  Rank  3 Group  0 Pid     43 on nccltest-worker-0 device  3 [0x61] NVIDIA
 ↪H100 80GB HBM3
#  Rank  4 Group  0 Pid     43 on nccltest-worker-0 device  4 [0x9d] NVIDIA
 ↪H100 80GB HBM3
#  Rank  5 Group  0 Pid     43 on nccltest-worker-0 device  5 [0xc3] NVIDIA
 ↪H100 80GB HBM3
#  Rank  6 Group  0 Pid     43 on nccltest-worker-0 device  6 [0xd1] NVIDIA
 ↪H100 80GB HBM3
#  Rank  7 Group  0 Pid     43 on nccltest-worker-0 device  7 [0xdf] NVIDIA
 ↪H100 80GB HBM3
#  Rank  8 Group  0 Pid     43 on nccltest-worker-1 device  0 [0x1b] NVIDIA
 ↪H100 80GB HBM3
#  Rank  9 Group  0 Pid     43 on nccltest-worker-1 device  1 [0x43] NVIDIA
 ↪H100 80GB HBM3
#  Rank 10 Group  0 Pid     43 on nccltest-worker-1 device  2 [0x52] NVIDIA
 ↪H100 80GB HBM3
#  Rank 11 Group  0 Pid     43 on nccltest-worker-1 device  3 [0x61] NVIDIA
 ↪H100 80GB HBM3
#  Rank 12 Group  0 Pid     43 on nccltest-worker-1 device  4 [0x9d] NVIDIA
 ↪H100 80GB HBM3
#  Rank 13 Group  0 Pid     43 on nccltest-worker-1 device  5 [0xc3] NVIDIA
 ↪H100 80GB HBM3
#  Rank 14 Group  0 Pid     43 on nccltest-worker-1 device  6 [0xd1] NVIDIA
 ↪H100 80GB HBM3
#  Rank 15 Group  0 Pid     43 on nccltest-worker-1 device  7 [0xdf] NVIDIA
 ↪H100 80GB HBM3
#  Rank 16 Group  0 Pid     43 on nccltest-worker-2 device  0 [0x1b] NVIDIA
 ↪H100 80GB HBM3
#  Rank 17 Group  0 Pid     43 on nccltest-worker-2 device  1 [0x43] NVIDIA
 ↪H100 80GB HBM3
#  Rank 18 Group  0 Pid     43 on nccltest-worker-2 device  2 [0x52] NVIDIA
 ↪H100 80GB HBM3
#  Rank 19 Group  0 Pid     43 on nccltest-worker-2 device  3 [0x61] NVIDIA
 ↪H100 80GB HBM3
```

(continues on next page)

(continued from previous page)

```
#  Rank 20 Group  0 Pid     43 on nccltest-worker-2 device  4 [0x9d] NVIDIA
 ↪H100 80GB HBM3
#  Rank 21 Group  0 Pid     43 on nccltest-worker-2 device  5 [0xc3] NVIDIA
 ↪H100 80GB HBM3
#  Rank 22 Group  0 Pid     43 on nccltest-worker-2 device  6 [0xd1] NVIDIA
 ↪H100 80GB HBM3
#  Rank 23 Group  0 Pid     43 on nccltest-worker-2 device  7 [0xdf] NVIDIA
 ↪H100 80GB HBM3
#  Rank 24 Group  0 Pid     43 on nccltest-worker-3 device  0 [0x1b] NVIDIA
 ↪H100 80GB HBM3
#  Rank 25 Group  0 Pid     43 on nccltest-worker-3 device  1 [0x43] NVIDIA
 ↪H100 80GB HBM3
#  Rank 26 Group  0 Pid     43 on nccltest-worker-3 device  2 [0x52] NVIDIA
 ↪H100 80GB HBM3
#  Rank 27 Group  0 Pid     43 on nccltest-worker-3 device  3 [0x61] NVIDIA
 ↪H100 80GB HBM3
#  Rank 28 Group  0 Pid     43 on nccltest-worker-3 device  4 [0x9d] NVIDIA
 ↪H100 80GB HBM3
#  Rank 29 Group  0 Pid     43 on nccltest-worker-3 device  5 [0xc3] NVIDIA
 ↪H100 80GB HBM3
#  Rank 30 Group  0 Pid     43 on nccltest-worker-3 device  6 [0xd1] NVIDIA
 ↪H100 80GB HBM3
#  Rank 31 Group  0 Pid     43 on nccltest-worker-3 device  7 [0xdf] NVIDIA
 ↪H100 80GB HBM3
#
#                                                              out-of-place
 ↪                  in-place
#       size         count      type   redop    root     time   algbw   busbw
 ↪#wrong     time   algbw   busbw #wrong
#        (B)    (elements)                               (us)  (GB/s)  (GB/s)
 ↪            (us)  (GB/s)  (GB/s)
           8             2     float     sum      -1    222.7    0.00    0.00
 ↪     0    52.78    0.00    0.00      0
          16             4     float     sum      -1    45.53    0.00    0.00
 ↪     0    45.37    0.00    0.00      0
          32             8     float     sum      -1    43.96    0.00    0.00
 ↪     0    46.49    0.00    0.00      0
          64            16     float     sum      -1    44.13    0.00    0.00
 ↪     0    45.86    0.00    0.00      0
         128            32     float     sum      -1    44.73    0.00    0.01
 ↪     0    44.28    0.00    0.01      0
         256            64     float     sum      -1    66.13    0.00    0.01
 ↪     0    43.62    0.01    0.01      0
         512           128     float     sum      -1    51.52    0.01    0.02
 ↪     0    46.60    0.01    0.02      0
        1024           256     float     sum      -1    50.13    0.02    0.04
 ↪     0    48.78    0.02    0.04      0
        2048           512     float     sum      -1    52.50    0.04    0.08
 ↪     0    48.21    0.04    0.08      0
        4096          1024     float     sum      -1    50.20    0.08    0.16
 ↪     0    50.25    0.08    0.16      0
        8192          2048     float     sum      -1    56.57    0.14    0.28
```

(continues on next page)

(continued from previous page)

```
↪      0    50.76    0.16    0.31      0
       16384          4096     float     sum      -1    56.42    0.29    0.56
↪      0    53.13    0.31    0.60      0
       32768          8192     float     sum      -1    55.20    0.59    1.15
↪      0    53.74    0.61    1.18      0
       65536         16384     float     sum      -1    69.90    0.94    1.82
↪      0    68.49    0.96    1.85      0
      131072         32768     float     sum      -1    63.59    2.06    3.99
↪      0    83.64    1.57    3.04      0
      262144         65536     float     sum      -1    74.29    3.53    6.84
↪      0    77.09    3.40    6.59      0
      524288        131072     float     sum      -1    78.86    6.65   12.88
↪      0    82.36    6.37   12.33      0
     1048576        262144     float     sum      -1    91.09   11.51   22.30
↪      0    88.45   11.85   22.97      0
     2097152        524288     float     sum      -1    159.7   13.13   25.44
↪      0    139.5   15.03   29.13      0
     4194304       1048576     float     sum      -1    151.1   27.75   53.77
↪      0    159.6   26.28   50.91      0
     8388608       2097152     float     sum      -1    197.8   42.41   82.17
↪      0    203.7   41.19   79.80      0
    16777216       4194304     float     sum      -1    254.9   65.81  127.50
↪      0    254.6   65.89  127.66      0
    33554432       8388608     float     sum      -1    364.0   92.19  178.61
↪      0    462.5   72.56  140.58      0
    67108864      16777216     float     sum      -1    562.0  119.41  231.35
↪      0    550.7  121.87  236.12      0
   134217728      33554432     float     sum      -1    982.6  136.59  264.64
↪      0    969.6  138.42  268.19      0
   268435456      67108864     float     sum      -1   1842.7  145.67  282.24
↪      0   1874.8  143.18  277.41      0
   536870912     134217728     float     sum      -1   3578.6  150.02  290.67
↪      0   3580.5  149.94  290.52      0
  1073741824     268435456     float     sum      -1   7421.0  144.69  280.34
↪      0   7059.9  152.09  294.67      0
  2147483648     536870912     float     sum      -1    14051  152.84  296.12
↪      0    14061  152.72  295.90      0
  4294967296    1073741824     float     sum      -1    28187  152.37  295.22
↪      0    28145  152.60  295.67      0
  8589934592    2147483648     float     sum      -1    56366  152.39  295.26
↪      0    56564  151.86  294.23      0
 17179869184    4294967296     float     sum      -1   112687  152.46  295.38
↪      0   112808  152.29  295.07      0
# Out of bounds values : 0 OK
# Avg bus bandwidth    : 94.9049
```

Chapter 21. Site Survey

21.1. Sample Site Survey

General Information	
Country Name	US
State/Province	California
Locality	Santa Clara
Organization Name	Example Org
Administrator Email	admin@example.org
Organizational Unit	Demo
Cluster Name	ExampleCluster
Head Node Shared IP (HA Virtual IP)	10.184.94.251
Add Failover Network?	
NFS Server IP	10.160.0.4
NAS Path to /cm/shared	/nfs/data/nas/cmshared
NAS Path to /home	/nfs/data/nas/home
Timezone	US/Los_Angeles
Network Topology	Type 2
IP Offset (Compute Nodes)	0.0.0.3
Partition Type	One Big Partition
OFED Stack Version	Mellanox OFED 23.10
OOB Management BMC Username	root
BCM Head Node Admin Username	root
BCM Head Node Admin Password	

Network Information				
DGX BasePOD RA Name	BCM Network Name	Network Address (Base IP Address)	Netmask (/Netmaskbits)	Gateway
Compute Fabric	computenet	100.64.0.0	255.255.0.0 (/16)	—
Management & Storage Fabric	managementnet (internalnet)	10.184.94.0	255.255.255.0 (/24)	10.184.94.1
OOB Management Fabric	oobmanagementnet (ipminet)	10.160.6.0	255.255.255.0 (/24)	10.160.6.1
IB Storage Fabric	storagenet	–	–	
Name Servers	Search Domains	Time Servers		
8.8.8.8	example.org	time.nist.gov		

BCM Head Node Information					Managementnet	
Name/Unic ID	Hostname	BMC IP (oobmanagementnet)	BMC Credentia	Node IP (managementnet)	MAC 1 (enp37s0np0	MAC 2 (enp226s0np0)
Head1	bcm10-headnode1	10.160.6.254		10.184.94.254	E8:EB:D3:09:2	E8:EB:D3:09:26:34
Head2	bcm10-headnode2	10.160.6.253		10.184.94.253	E8:EB:D3:09:2	E8:EB:D3:09:26:44

DGX Node Information (1)					Managementnet	
Name/Unic ID	Hostname	BMC IP (oobmanagementnet)	BMC Credentials	Node IP (managementnet)	MAC 1 (enp37s0np0)	MAC 2 (enp226s0np0)
DGX-01	dgx-01	10.160.6.31		10.184.94.11	94:6D:AE:AA:1	94:6D:AE:AA:14:89
DGX-02	dgx-02	10.160.6.32		10.184.94.12	A0:88:C2:A3:4	A0:88:C2:A3:4B:05
DGX-03	dgx-03	10.160.6.33		10.184.94.13	94:6D:AE:1C:8	94:6D:AE:1C:80:7D
DGX-04	dgx-04	10.160.6.34		10.184.94.14	A0:88:C2:04:7	A0:88:C2:04:60:E1

DGX Node Information (2)		computenet							
Name/Uniqu ID	Host-name	ibp220s(	ibp154s(	ibp206s(	ibp192s(	ibp79s0	ibp64s0	ibp94s0	ibp24s0
DGX-01	dgx-01	100.64.0	100.64.1	100.64.2	100.64.3	100.64.4	100.64.5	100.64.6	100.64.7.1
DGX-02	dgx-02	100.64.0	100.64.1	100.64.2	100.64.3	100.64.4	100.64.5	100.64.6	100.64.7.2
DGX-03	dgx-03	100.64.0	100.64.1	100.64.2	100.64.3	100.64.4	100.64.5	100.64.6	100.64.7.3
DGX-04	dgx-04	100.64.0	100.64.1	100.64.2	100.64.3	100.64.4	100.64.5	100.64.6	100.64.7.4

Kubernetes Node Information				ManagementNet		
Name/Unic ID	Host-name	BMC IP (oobman-agementnet)	BMC Cre-dentials	Node IP (man-agementnet)	MAC 1 (enp37s0np0	MAC 2 (enp226s0np0)
Knode1	k8s-control-01	10.160.6.4		10.184.94.4	10:70:FD:73:7	10:70:FD:73:7B:D6
Knode2	k8s-control-02	10.160.6.5		10.184.94.5	10:70:FD:73:7	10:70:FD:73:7C:C6
Knode3	k8s-control-03	10.160.6.6		10.184.94.6	E8:EB:D3:09:2	B8:CE:F6:63:ED:36

21.2. Blank Site Survey

General Information
Country Name
State/Province
Locality
Organization Name
Administrator Email
Organizational Unit
Cluster Name
Head Node Shared IP (HA Virtual IP)
Add Failover Network?
NFS Server IP
NAS Path to /cm/shared
NAS Path to /home
Timezone
Network Topology
IP Offset (Compute Nodes)
Partition Type
OFED Stack Version
OOB Management BMC Username
OOB Management BMC Password

Network Information				
DGX BasePOD RA Name	BCM Network Name	Network Address	Netmask (/Netmaskbits)	Gateway
Compute Fabric	computenet			
Management & Storage Fabric	managementnet (internalnet)			
OOB Management Fabric	oobmanagementnet (ipminet)			
IB Storage Fabric	storagenet			
Name Servers	Search Domains	Time Servers		

BCM Head Node Information						
Name/Unic ID	Host-name	BMC IP (oobman-agementnet)	BMC Cre-dentials	Node IP (man-agementnet)	MAC 1 (enp37s0np0)	MAC 2 (enp226s0np0)
Head1						
Head2						

DGX Node Information (1) | Managementnet						
Name/Unic ID	Host-name	BMC IP (oobman-agementnet)	BMC Cre-dentials	Node IP (man-agementnet)	MAC 1 (enp37s0np0)	MAC 2 (enp226s0np0
DGX-01						
DGX-02						
DGX-03						
DGX-04						

DGX Node Information (2)									
Name/Unique ID	Host-name	ibp220s(	ibp154s(	ibp206s(	ibp192s(	ibp79s(	ibp64s(	ibp94s(	ibp24s0
DGX-01									
DGX-02									
DGX-03									
DGX-04									

Kubernetes Node Information					
Name/Unique ID	Host-name	BMC IP (oobmanage-mentnet)	BMC Cre-dentials	Node IP (manage-mentnet)	MAC 1 (enp37s0np0)
Knode1					
Knode2					
Knode3					

Chapter 22. Switch Configurations

22.1. SN2201 (oobmanagementnet) Switch Configuration

```
nv set bridge domain br_default vlan 101
nv set interface bond1 bond member swp49
nv set interface bond1 bond member swp50
nv set interface bond1 bridge domain br_default untagged 1
nv set interface bond1 bridge domain br_default vlan all
nv set interface bond1 type bond
nv set interface eth0 ip address dhcp
nv set interface eth0 ip vrf mgmt
nv set interface eth0 type eth
nv set interface swp1-48 bridge domain br_default access 101
nv set interface swp1-48 description 'BMC Ports'
nv set interface swp1-50 link state up
nv set interface swp1-50 type swp
nv set service ntp mgmt server 0.cumulusnetworks.pool.ntp.org
nv set service ntp mgmt server 1.cumulusnetworks.pool.ntp.org
nv set service ntp mgmt server 2.cumulusnetworks.pool.ntp.org
nv set service ntp mgmt server 3.cumulusnetworks.pool.ntp.org
nv set system aaa class nvapply action allow
nv set system aaa class nvapply command-path / permission all
nv set system aaa class nvshow action allow
nv set system aaa class nvshow command-path / permission ro
nv set system aaa class sudo action allow
nv set system aaa class sudo command-path / permission all
nv set system aaa role nvue-admin class nvapply
nv set system aaa role nvue-monitor class nvshow
nv set system aaa role system-admin class nvapply
nv set system aaa role system-admin class sudo
nv set system aaa user cumulus full-name cumulus,,,
nv set system aaa user cumulus hashed-password '*'
nv set system aaa user cumulus role system-admin
nv set system api state enabled
nv set system config auto-save state enabled
nv set system control-plane acl acl-default-dos inbound
nv set system control-plane acl acl-default-whitelist inbound
nv set system hostname IPMI-Basepod-01
```

(continues on next page)

(continued from previous page)

```
nv set system reboot mode cold
nv set system ssh-server permit-root-login enabled
nv set system ssh-server state enabled
nv set system ssh-server vrf mgmt
nv set system timezone America/Los_Angeles
nv set system wjh channel forwarding trigger l2
nv set system wjh channel forwarding trigger l3
nv set system wjh channel forwarding trigger tunnel
nv set system wjh enable on
```

22.2. SN4600C-1 (managementnet) Switches Configuration

22.2.1. SN4600C-1 Configuration

```
nv set bridge domain br_default vlan 100-102
nv set interface bond1 bond member swp1
nv set interface bond1 bond mlag id 1
nv set interface bond1-11,13-48,51 bond lacp-bypass on
nv set interface bond1-48 bridge domain br_default access 102
nv set interface bond1-48,51 bond mlag enable on
nv set interface bond1-48,51 type bond
nv set interface bond2 bond member swp2
nv set interface bond2 bond mlag id 2
nv set interface bond3 bond member swp3
nv set interface bond3 bond mlag id 3
nv set interface bond4 bond member swp4
nv set interface bond4 bond mlag id 4
nv set interface bond5 bond member swp5
nv set interface bond5 bond mlag id 5
nv set interface bond6 bond member swp6
nv set interface bond6 bond mlag id 6
nv set interface bond7 bond member swp7
nv set interface bond7 bond mlag id 7
nv set interface bond8 bond member swp8
nv set interface bond8 bond mlag id 8
nv set interface bond9 bond member swp9
nv set interface bond9 bond mlag id 9
nv set interface bond10 bond member swp10
nv set interface bond10 bond mlag id 10
nv set interface bond11 bond member swp11
nv set interface bond11 bond mlag id 11
nv set interface bond12 bond member swp12
nv set interface bond12 bond mlag id 12
nv set interface bond13 bond member swp13
nv set interface bond13 bond mlag id 13
nv set interface bond14 bond member swp14
nv set interface bond14 bond mlag id 14
```

(continues on next page)

(continued from previous page)

```
nv set interface bond15 bond member swp15
nv set interface bond15 bond mlag id 15
nv set interface bond16 bond member swp16
nv set interface bond16 bond mlag id 16
nv set interface bond17 bond member swp17
nv set interface bond17 bond mlag id 17
nv set interface bond18 bond member swp18
nv set interface bond18 bond mlag id 18
nv set interface bond19 bond member swp19
nv set interface bond19 bond mlag id 19
nv set interface bond20 bond member swp20
nv set interface bond20 bond mlag id 20
nv set interface bond21 bond member swp21
nv set interface bond21 bond mlag id 21
nv set interface bond22 bond member swp22
nv set interface bond22 bond mlag id 22
nv set interface bond23 bond member swp23
nv set interface bond23 bond mlag id 23
nv set interface bond24 bond member swp24
nv set interface bond24 bond mlag id 24
nv set interface bond25 bond member swp25
nv set interface bond25 bond mlag id 25
nv set interface bond26 bond member swp26
nv set interface bond26 bond mlag id 26
nv set interface bond27 bond member swp27
nv set interface bond27 bond mlag id 27
nv set interface bond28 bond member swp28
nv set interface bond28 bond mlag id 28
nv set interface bond29 bond member swp29
nv set interface bond29 bond mlag id 29
nv set interface bond30 bond member swp30
nv set interface bond30 bond mlag id 30
nv set interface bond31 bond member swp31
nv set interface bond31 bond mlag id 31
nv set interface bond32 bond member swp32
nv set interface bond32 bond mlag id 32
nv set interface bond33 bond member swp33
nv set interface bond33 bond mlag id 33
nv set interface bond34 bond member swp34
nv set interface bond34 bond mlag id 34
nv set interface bond35 bond member swp35
nv set interface bond35 bond mlag id 35
nv set interface bond36 bond member swp36
nv set interface bond36 bond mlag id 36
nv set interface bond37 bond member swp37
nv set interface bond37 bond mlag id 37
nv set interface bond38 bond member swp38
nv set interface bond38 bond mlag id 38
nv set interface bond39 bond member swp39
nv set interface bond39 bond mlag id 39
nv set interface bond40 bond member swp40
nv set interface bond40 bond mlag id 40
```

(continues on next page)

(continued from previous page)

```
nv set interface bond41 bond member swp41
nv set interface bond41 bond mlag id 41
nv set interface bond42 bond member swp42
nv set interface bond42 bond mlag id 42
nv set interface bond43 bond member swp43
nv set interface bond43 bond mlag id 43
nv set interface bond44 bond member swp44
nv set interface bond44 bond mlag id 44
nv set interface bond45 bond member swp45
nv set interface bond45 bond mlag id 45
nv set interface bond46 bond member swp46
nv set interface bond46 bond mlag id 46
nv set interface bond47 bond member swp47
nv set interface bond47 bond mlag id 47
nv set interface bond48 bond member swp48
nv set interface bond48 bond mlag id 48
nv set interface bond51 bond member swp51
nv set interface bond51 bond mlag id 51
nv set interface bond51 bridge domain br_default untagged 1
nv set interface bond51 bridge domain br_default vlan all
nv set interface eth0 ip address dhcp
nv set interface eth0 ip vrf mgmt
nv set interface eth0 type eth
nv set interface lo ip address 10.160.254.22/32
nv set interface lo type loopback
nv set interface peerlink bond member swp63
nv set interface peerlink bond member swp64
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 interface swp49-50 type swp
nv set interface vlan101-102 ip vrr enable on
nv set interface vlan101-102 ip vrr mac-address 00:1c:73:aa:bb:04
nv set interface vlan101-102 ip vrr state up
nv set interface vlan101-102 type svi
nv set interface vlan101 ip address 10.160.6.2/24
nv set interface vlan101 ip vrr address 10.160.6.1/24
nv set interface vlan101 vlan 101
nv set interface vlan102 ip address 10.184.94.2/24
nv set interface vlan102 ip vrr address 10.184.94.1/24
nv set interface vlan102 vlan 102
nv set mlag backup 10.160.254.23
nv set mlag enable on
nv set mlag mac-address 44:38:39:FF:0A:00
nv set mlag peer-ip linklocal
nv set mlag priority 2048
nv set router bgp autonomous-system 4200120327
nv set router bgp enable on
nv set router bgp router-id 10.160.254.22
nv set router vrr enable on
nv set service ntp mgmt server 0.cumulusnetworks.pool.ntp.org
```

(continues on next page)

(continued from previous page)

```
nv set service ntp mgmt server 1.cumulusnetworks.pool.ntp.org
nv set service ntp mgmt server 2.cumulusnetworks.pool.ntp.org
nv set service ntp mgmt server 3.cumulusnetworks.pool.ntp.org
nv set system aaa class nvapply action allow
nv set system aaa class nvapply command-path / permission all
nv set system aaa class nvshow action allow
nv set system aaa class nvshow command-path / permission ro
nv set system aaa class sudo action allow
nv set system aaa class sudo command-path / permission all
nv set system aaa role nvue-admin class nvapply
nv set system aaa role nvue-monitor class nvshow
nv set system aaa role system-admin class nvapply
nv set system aaa role system-admin class sudo
nv set system aaa user cumulus full-name cumulus,,,
nv set system aaa user cumulus hashed-password '*'
nv set system aaa user cumulus role system-admin
nv set system api state enabled
nv set system config auto-save state enabled
nv set system control-plane acl acl-default-dos inbound
nv set system control-plane acl acl-default-whitelist inbound
nv set system hostname SN4600C-1
nv set system reboot mode cold
nv set system ssh-server permit-root-login enabled
nv set system ssh-server state enabled
nv set system ssh-server vrf mgmt
nv set system timezone America/Los_Angeles
nv set system wjh channel forwarding trigger l2
nv set system wjh channel forwarding trigger l3
nv set system wjh channel forwarding trigger tunnel
nv set system wjh enable on
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 peerlink.4094 remote-as internal
nv set vrf default router bgp neighbor peerlink.4094 timers connection-retry
 ↪10
nv set vrf default router bgp neighbor peerlink.4094 timers hold 10
nv set vrf default router bgp neighbor peerlink.4094 timers keepalive 3
nv set vrf default router bgp neighbor peerlink.4094 timers route-
 ↪advertisement auto
nv set vrf default router bgp neighbor peerlink.4094 type unnumbered
nv set vrf default router bgp neighbor swp49 remote-as external
nv set vrf default router bgp neighbor swp49 timers connection-retry 10
nv set vrf default router bgp neighbor swp49 timers hold 10
nv set vrf default router bgp neighbor swp49 timers keepalive 3
nv set vrf default router bgp neighbor swp49 timers route-advertisement auto
nv set vrf default router bgp neighbor swp49 type unnumbered
nv set vrf default router bgp neighbor swp50 remote-as external
nv set vrf default router bgp neighbor swp50 timers connection-retry 10
nv set vrf default router bgp neighbor swp50 timers hold 10
nv set vrf default router bgp neighbor swp50 timers keepalive 3
```

(continues on next page)

(continued from previous page)

```
nv set vrf default router bgp neighbor swp50 timers route-advertisement auto
nv set vrf default router bgp neighbor swp50 type unnumbered
```

22.2.2. SN4600C-2 Configuration

```
nv set bridge domain br_default vlan 100-102
nv set interface bond1 bond member swp1
nv set interface bond1 bond mlag id 1
nv set interface bond1-11,13-48,51 bond lacp-bypass on
nv set interface bond1-48 bridge domain br_default access 102
nv set interface bond1-48,51 bond mlag enable on
nv set interface bond1-48,51 type bond
nv set interface bond2 bond member swp2
nv set interface bond2 bond mlag id 2
nv set interface bond3 bond member swp3
nv set interface bond3 bond mlag id 3
nv set interface bond4 bond member swp4
nv set interface bond4 bond mlag id 4
nv set interface bond5 bond member swp5
nv set interface bond5 bond mlag id 5
nv set interface bond6 bond member swp6
nv set interface bond6 bond mlag id 6
nv set interface bond7 bond member swp7
nv set interface bond7 bond mlag id 7
nv set interface bond8 bond member swp8
nv set interface bond8 bond mlag id 8
nv set interface bond9 bond member swp9
nv set interface bond9 bond mlag id 9
nv set interface bond10 bond member swp10
nv set interface bond10 bond mlag id 10
nv set interface bond11 bond member swp11
nv set interface bond11 bond mlag id 11
nv set interface bond12 bond member swp12
nv set interface bond12 bond mlag id 12
nv set interface bond13 bond member swp13
nv set interface bond13 bond mlag id 13
nv set interface bond14 bond member swp14
nv set interface bond14 bond mlag id 14
nv set interface bond15 bond member swp15
nv set interface bond15 bond mlag id 15
nv set interface bond16 bond member swp16
nv set interface bond16 bond mlag id 16
nv set interface bond17 bond member swp17
nv set interface bond17 bond mlag id 17
nv set interface bond18 bond member swp18
nv set interface bond18 bond mlag id 18
nv set interface bond19 bond member swp19
nv set interface bond19 bond mlag id 19
nv set interface bond20 bond member swp20
nv set interface bond20 bond mlag id 20
```

(continues on next page)

(continued from previous page)

```
nv set interface bond21 bond member swp21
nv set interface bond21 bond mlag id 21
nv set interface bond22 bond member swp22
nv set interface bond22 bond mlag id 22
nv set interface bond23 bond member swp23
nv set interface bond23 bond mlag id 23
nv set interface bond24 bond member swp24
nv set interface bond24 bond mlag id 24
nv set interface bond25 bond member swp25
nv set interface bond25 bond mlag id 25
nv set interface bond26 bond member swp26
nv set interface bond26 bond mlag id 26
nv set interface bond27 bond member swp27
nv set interface bond27 bond mlag id 27
nv set interface bond28 bond member swp28
nv set interface bond28 bond mlag id 28
nv set interface bond29 bond member swp29
nv set interface bond29 bond mlag id 29
nv set interface bond30 bond member swp30
nv set interface bond30 bond mlag id 30
nv set interface bond31 bond member swp31
nv set interface bond31 bond mlag id 31
nv set interface bond32 bond member swp32
nv set interface bond32 bond mlag id 32
nv set interface bond33 bond member swp33
nv set interface bond33 bond mlag id 33
nv set interface bond34 bond member swp34
nv set interface bond34 bond mlag id 34
nv set interface bond35 bond member swp35
nv set interface bond35 bond mlag id 35
nv set interface bond36 bond member swp36
nv set interface bond36 bond mlag id 36
nv set interface bond37 bond member swp37
nv set interface bond37 bond mlag id 37
nv set interface bond38 bond member swp38
nv set interface bond38 bond mlag id 38
nv set interface bond39 bond member swp39
nv set interface bond39 bond mlag id 39
nv set interface bond40 bond member swp40
nv set interface bond40 bond mlag id 40
nv set interface bond41 bond member swp41
nv set interface bond41 bond mlag id 41
nv set interface bond42 bond member swp42
nv set interface bond42 bond mlag id 42
nv set interface bond43 bond member swp43
nv set interface bond43 bond mlag id 43
nv set interface bond44 bond member swp44
nv set interface bond44 bond mlag id 44
nv set interface bond45 bond member swp45
nv set interface bond45 bond mlag id 45
nv set interface bond46 bond member swp46
nv set interface bond46 bond mlag id 46
```

(continues on next page)

(continued from previous page)

```
nv set interface bond47 bond member swp47
nv set interface bond47 bond mlag id 47
nv set interface bond48 bond member swp48
nv set interface bond48 bond mlag id 48
nv set interface bond51 bond member swp51
nv set interface bond51 bond mlag id 51
nv set interface bond51 bridge domain br_default untagged 1
nv set interface bond51 bridge domain br_default vlan all
nv set interface eth0 ip address dhcp
nv set interface eth0 ip vrf mgmt
nv set interface eth0 type eth
nv set interface lo ip address 10.160.254.22/32
nv set interface lo type loopback
nv set interface peerlink bond member swp63
nv set interface peerlink bond member swp64
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 interface swp49-50 type swp
nv set interface vlan101-102 ip vrr enable on
nv set interface vlan101-102 ip vrr mac-address 00:1c:73:aa:bb:04
nv set interface vlan101-102 ip vrr state up
nv set interface vlan101-102 type svi
nv set interface vlan101 ip address 10.160.6.2/24
nv set interface vlan101 ip vrr address 10.160.6.1/24
nv set interface vlan101 vlan 101
nv set interface vlan102 ip address 10.184.94.2/24
nv set interface vlan102 ip vrr address 10.184.94.1/24
nv set interface vlan102 vlan 102
nv set mlag backup 10.160.254.23
nv set mlag enable on
nv set mlag mac-address 44:38:39:FF:0A:00
nv set mlag peer-ip linklocal
nv set mlag priority 2048
nv set router bgp autonomous-system 4200120327
nv set router bgp enable on
nv set router bgp router-id 10.160.254.22
nv set router vrr enable on
nv set service ntp mgmt server 0.cumulusnetworks.pool.ntp.org
nv set service ntp mgmt server 1.cumulusnetworks.pool.ntp.org
nv set service ntp mgmt server 2.cumulusnetworks.pool.ntp.org
nv set service ntp mgmt server 3.cumulusnetworks.pool.ntp.org
nv set system aaa class nvapply action allow
nv set system aaa class nvapply command-path / permission all
nv set system aaa class nvshow action allow
nv set system aaa class nvshow command-path / permission ro
nv set system aaa class sudo action allow
nv set system aaa class sudo command-path / permission all
nv set system aaa role nvue-admin class nvapply
nv set system aaa role nvue-monitor class nvshow
nv set system aaa role system-admin class nvapply
```

(continues on next page)

(continued from previous page)

```
nv set system aaa role system-admin class sudo
nv set system aaa user cumulus full-name cumulus,,,
nv set system aaa user cumulus hashed-password '*'
nv set system aaa user cumulus role system-admin
nv set system api state enabled
nv set system config auto-save state enabled
nv set system control-plane acl acl-default-dos inbound
nv set system control-plane acl acl-default-whitelist inbound
nv set system hostname SN4600C-2
nv set system reboot mode cold
nv set system ssh-server permit-root-login enabled
nv set system ssh-server state enabled
nv set system ssh-server vrf mgmt
nv set system timezone America/Los_Angeles
nv set system wjh channel forwarding trigger l2
nv set system wjh channel forwarding trigger l3
nv set system wjh channel forwarding trigger tunnel
nv set system wjh enable on
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 peerlink.4094 remote-as internal
nv set vrf default router bgp neighbor peerlink.4094 timers connection-retry
 ↪10
nv set vrf default router bgp neighbor peerlink.4094 timers hold 10
nv set vrf default router bgp neighbor peerlink.4094 timers keepalive 3
nv set vrf default router bgp neighbor peerlink.4094 timers route-
 ↪advertisement auto
nv set vrf default router bgp neighbor peerlink.4094 type unnumbered
nv set vrf default router bgp neighbor swp49 remote-as external
nv set vrf default router bgp neighbor swp49 timers connection-retry 10
nv set vrf default router bgp neighbor swp49 timers hold 10
nv set vrf default router bgp neighbor swp49 timers keepalive 3
nv set vrf default router bgp neighbor swp49 timers route-advertisement auto
nv set vrf default router bgp neighbor swp49 type unnumbered
nv set vrf default router bgp neighbor swp50 remote-as external
nv set vrf default router bgp neighbor swp50 timers connection-retry 10
nv set vrf default router bgp neighbor swp50 timers hold 10
nv set vrf default router bgp neighbor swp50 timers keepalive 3
nv set vrf default router bgp neighbor swp50 timers route-advertisement auto
nv set vrf default router bgp neighbor swp50 type unnumbered
```

22.3. QM9700 (computenet) Switches Configuration

22.3.1. QM9700-1

```
##
## Running database "initial"
## Generated at 2025/02/15 06:42:32 +0000
## Hostname: QM9700-1
## Product release: 3.12.1002
##

##
## Running-config temporary prefix mode setting
##
no cli default prefix-modes enable

##
## IB Partition configuration
##
   ib partition Default defmember full force

##
## Subnet Manager configuration
##
   ib sm virt enable

##
## IB ports configuration
##
   interface ib 1/1/1-1/1/2 mtu 4K
   interface ib 1/1/1-1/1/2 op-vls 8
   interface ib 1/1/1-1/1/2 speed sdr qdr fdr edr hdr ndr
   interface ib 1/1/1-1/1/2 width 7

##
## Network interface configuration
##
no interface mgmt0 dhcp
   interface mgmt0 ip address 10.185.231.43 /22
   management-lldp enable

##
## Other IP configuration
##
   hostname QM9700-1
   ip domain-list nvidia.com
   ip name-server 10.126.136.6
   ip route 0.0.0.0/0 10.185.228.1

##
## Other IPv6 configuration
##
no ipv6 enable

##
```

(continues on next page)

(continued from previous page)

```
## Local user account configuration
##
   username admin password 7 <>
   username monitor password 7 <>

##
## AAA remote server configuration
##
# ldap bind-password ********
# radius-server key ********
# tacacs-server key ********

##
## Password restriction configuration
##
no password hardening enable

##
## Network management configuration
##
# web proxy auth basic password ********

##
## X.509 certificates configuration
##
#
# Certificate name system-self-signed, ID
 ↪9f639fcad62931e3996712b59066cdda047fb176
# (public-cert config omitted since private-key config is hidden)


##
## IB nodename to GUID mapping
##
   ib ha infiniband-default ip 10.185.230.247 /22 force
   ib smnode CL-01 create
   ib smnode CL-01 enable
   ib smnode CL-01 sm-priority 15
##
## Persistent prefix mode setting
##
cli default prefix-modes enable
```

22.3.2. QM9700-2

```
##
## Running database "initial"
## Generated at 2025/02/15 06:41:54 +0000
## Hostname: QM9700-2
## Product release: 3.12.1002
```

(continues on next page)

(continued from previous page)

```
##

##
## Running-config temporary prefix mode setting
##
no cli default prefix-modes enable

##
## IB Partition configuration
##
   ib partition Default defmember full force

##
## Subnet Manager configuration
##
   ib sm virt enable

##
## Other IP configuration
##
   hostname QM9700-2

##
## Local user account configuration
##
   username admin password 7 <>
   username monitor password 7 <>

##
## AAA remote server configuration
##
# ldap bind-password ********
# radius-server key ********
# tacacs-server key ********

##
## Password restriction configuration
##
no password hardening enable

##
## Network management configuration
##
# web proxy auth basic password ********

##
## X.509 certificates configuration
##
#
# Certificate name system-self-signed, ID
↪146da5394146409cf2e60c4b7debbedd1e2e6ac4
# (public-cert config omitted since private-key config is hidden)
```

(continues on next page)

(continued from previous page)

```
##
## IB nodename to GUID mapping
##
   ib ha infiniband-default ip 10.185.230.247 /22 force
   ib smnode CL-01 create
   ib smnode CL-01 enable
   ib smnode CL-01 sm-priority 15
##
## Persistent prefix mode setting
##
cli default prefix-modes enable
```

Copyright