NVIDIA DGX BasePOD

Deployment Guide
Featuring NVIDIA DGX A100 System
Abstract

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

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

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

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

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

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

Chapter 1. Architecture.............................................................................................................................. 1  
1.1 Hardware Overview.......................................................................................................................... 1  
1.2 Networking ........................................................................................................................................ 3  
1.2.1 DGX A100 System Network Ports................................................................................... 3  
1.2.2 DGX BasePOD Network Overview.................................................................................... 4  
1.2.3 internalnet and externalnet.............................................................................................. 4  
1.2.4 ipminet........................................................................................................................................ 5  
1.2.5 ibnet............................................................................................................................................. 5  
1.3 Software ............................................................................................................................................. 6  
1.3.1 Kubernetes (K8s).................................................................................................................... 6  
1.3.2 Jupyter (Optional)................................................................................................................... 6  
1.4 Storage................................................................................................................................................ 7  

Chapter 2. Deployment............................................................................................................................... 8  
2.1 Cluster Configuration ......................................................................................................................... 24  
2.1.1 Network Configuration...................................................................................................... 27  
2.1.2 Configure Disk Layouts for Node Categories.................................................................... 27  
2.1.3 Configure Node Network Interfaces............................................................................ 28  
2.1.3.1 Configure BCM to Allow MAC Addresses to PXE Boot................................ 28  
2.1.3.2 Configure Provisioning Interfaces on the DGX Nodes......................................... 29  
2.1.3.3 Configure Provisioning Interfaces on the K8s Nodes.................................. 30  
2.1.3.4 Configure InfiniBand Interfaces on DGX Nodes............................................ 31  
2.1.3.5 Identify the Cluster Nodes...................................................................................... 31  
2.2 Power On and Provision Cluster Nodes................................................................................... 32  
2.3 Deploy Docker................................................................................................................................ 33  
2.4 Deploy K8s...................................................................................................................................... 36  
2.5 Deploying Slurm.............................................................................................................................. 46  
2.6 (Optional) Deploy Jupyter............................................................................................................. 56  
2.6.1 Install Jupyter Using the CLI Wizard............................................................................ 56  

Chapter 3. High Availability .................................................................................................................... 60  
3.1.1 Verify the HA Setup............................................................................................................ 77  

Chapter 4. Basic User Management ................................................................................................... 80  
4.1 Configuring a User........................................................................................................................... 80  
4.1.1 Procedures to Remove a User ........................................................................................ 81  
4.2 Adding a User to K8s..................................................................................................................... 82
4.3 Removing a User from K8s ........................................................................................................ 83

Appendix A. Site Survey ................................................................................................................ iv

Appendix B. Switch Configurations .............................................................................................. ix

B.1 SN4600 #1 (In-band Management Switch) ........................................................................ ix
B.2 SN4600 #2 (In-band Management Switch) ........................................................................ xii
B.3 SN2201 (Out-of-band Management Switch) ...................................................................... xv
B.4 Ethernet Network Configuration Verifications .................................................................... xv
Chapter 1. Architecture

1.1 Hardware Overview

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

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

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

Table 1. DGX BasePOD components

<table>
<thead>
<tr>
<th>Component</th>
<th>Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute nodes</td>
<td>DGX A100 system</td>
</tr>
<tr>
<td>Compute fabric</td>
<td>NVIDIA Quantum QM8700 HDR 200 Gbps InfiniBand</td>
</tr>
<tr>
<td>Management fabric</td>
<td>NVIDIA SN4600 switches</td>
</tr>
<tr>
<td>Storage fabric</td>
<td>NVIDIA SN4600 switches for Ethernet attached storage</td>
</tr>
<tr>
<td></td>
<td>NVIDIA Quantum QM8700 HDR 200 Gb/s for InfiniBand attached storage</td>
</tr>
<tr>
<td>Out-of-band management fabric</td>
<td>NVIDIA SN2201 switches</td>
</tr>
<tr>
<td>Control plane</td>
<td>Minimum Requirements (each server):</td>
</tr>
<tr>
<td></td>
<td>&gt;  64-bit x86 processor, AMD EPYC 7272 or equivalent</td>
</tr>
<tr>
<td></td>
<td>&gt;  256 GB memory</td>
</tr>
<tr>
<td></td>
<td>&gt;  1 TB SSD</td>
</tr>
</tbody>
</table>
Two 100 Gbps network ports
1.2 Networking

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

1.2.1 DGX A100 System Network Ports

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

Figure 1. DGX A100 system rear

The following ports are selected for DGX BasePOD networking:

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

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

There are four networks in a DGX BasePOD configuration:

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

These are shown in Figure 3.

![Network design and topology diagram](image)

**Figure 2. Network design and topology diagram**

1.2.3 internalnet and externalnet

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

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

**Table 2. BGP protocols**

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BGP</td>
<td>Used as required for routing between switches</td>
</tr>
<tr>
<td>iBGP</td>
<td>Configured between the two SN4600s using the MLAG peer1nkn.4094 interface</td>
</tr>
<tr>
<td>eBGP</td>
<td>Configured between the uplink (SN4600) and IPMI (SN2201) switches</td>
</tr>
</tbody>
</table>

### 1.2.4 ipminet

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

### 1.2.5 ibnet

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

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

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

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

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

1.3.1 Kubernetes (K8s)

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

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

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

1.3.2 Jupyter (Optional)

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

These are the Jupyter extensions that BCM deploys:

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

An NFS solution is required for a highly available (HA) BCM installation, and the required export path for that is described in this DGX BasePOD document. A DGX BasePOD typically also includes dedicated storage, but the configuration of that is outside the scope of this document. Contact the vendor of the storage solution being used for instructions on configuring the high performance-storage portions of a DGX BasePOD.
Deployment of a DGX BasePOD involves pre-setup, deployment, and use of BCM to provision the K8s cluster, and optionally deploy Jupyter.

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

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

3. Configure the NFS solution.
   a. As stated in Section 1.4, NFS configuration steps are not in scope for this document.
   b. This DGX BasePOD deployment uses the path `/var/nfs/general`, which is the NFS export path provided in Table 3 of the Site Survey.
   c. Use the following parameters for the NFS server export file `/etc/exports`:
      ```
      /var/nfs/general *(rw,sync,no_root_squash,no_subtree_check)
      ```

4. Set the DGX BIOS so that the DGX systems PXE boot by default.
   BCM requires DGX systems to PXE boot.
   d. Connect to the BMC of the DGX system.
e. In the Network tab of the System Inventory window, locate the MAC addresses for the Storage 4-2 and Storage 5-2 interfaces.

f. In the DGX A100 system BIOS, configure Boot Option #1 to be [NETWORK]. Set other Boot devices to [DISABLED].
g. Disable PXE boot devices except for Storage 4-2 and Storage 5-2. Set them to use IPv4.

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

5. Ensure that the Network boot option is configured as the primary boot option for the K8s control plane nodes that are to be used for this cluster.
This is an example of a system that will boot from the network with **Slot 1 Port 2** and **Slot 2 Port 2**.

6. Download a BCM ISO from the [Bright Cluster Manager/Base Command Manager download site](https://example.com). Select Base Command Manager 10, Ubuntu 20.04, and check the **Include NVIDIA DGX A100 Software image** checkbox.
7. Burn the ISO to a DVD or to a bootable USB device. It can also be mounted as virtual media and installed using the BMC. The specific mechanism for the latter will vary by vendor.

8. Ensure that the BIOS of the target head node is configured in UEFI mode and that its boot order is configured to boot the media containing the BCM installer image.

9. Boot the installation media.

10. At the grub menu, choose Start Base Command Manager Graphical Installer.

11. Select Start installation on the splash screen.
12. Accept the terms of the NVIDIA EULA by checking I agree and then select Next.

13. Accept the terms of the Ubuntu Server UELA by checking I agree and then select Next.
14. Unless instructed otherwise, select **Next** without modifying the kernel modules to be loaded at boot time.

15. Verify the Hardware info is correct and then select **Next**.
   For example, that the target storage device and the cabled host network interfaces are present (in this case three NVMe drives are the target storage device, and `ens1np0` and `ens2np0` are the cabled host network interfaces).
17. On the Installation source screen, choose the appropriate source and then select Next.

Running a media integrity check is optional.

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

Enter information from the site survey. An example site survey is in Appendix A.
19. On the **Workload manager** screen, choose **None** and then select **Next**.

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

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

In a DGX BasePOD architecture, the cluster nodes are connected to the head node over the internal network, with the head node serving as their default gateway.
21. On the Head node screen, enter the Hostname, Administrator password, choose Other for Hardware manufacturer, and then select Next.

22. Configure the Compute nodes screen.
   - Set the Number of nodes to 4.
   - Set Node digits to 2.
   - Set Hardware manufacturer to NVIDIA DGX.
23. Configure BMC configuration screen.
   i. Select Yes for both the Head Node and the Compute Nodes.
   j. Select IPMI from the BMC network type select lists for both the Head Node and the Compute Nodes.
   k. Select No to the DHCP question for both node types.
   l. Select Yes for Automatically configure BMC when node boots.
   m. Select New dedicated network from the To which Ethernet segment is BMC connected? list.

24. Configure the Networks screens.
   n. externalnet
Set the Base IP address, Netmask, Gateway, and Domain name according to the site survey.

- **internalnet**

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

- **ipminet**
Set the Base IP address, Netmask, and Gateway according to the site survey.
25. Configure the Head node interfaces according to the site survey and then select Next.

26. Configure the offset for BOOTIF and ipmi0 to 0.0.0.3 on the Compute nodes network interfaces screen and then select Next.
27. Configure the installation drive(s) on the Disk layout screen and then select Next.

28. Configure the Disk layout Settings screen and then select Next.
   Set the Head node disk layout to One big partition and the Compute nodes disk layout to Default Standard Layout.
29. Check the OFED/OPA stack box and choose Mellanox OFED 5.8 on the Additional software screen and then select Next.

30. Review the information on the Summary screen.

The Summary screen provides an opportunity to confirm the head node and basic cluster configuration before deployment begins. If anything does not match expectations, use the Back button to navigate to the appropriate screen to correct any mistake.
31. Configure the Deployment screen and then select Reboot.
Check the Automatically reboot after installation is complete checkbox to reboot
the host upon successful completion of the deployment. Select Install log to see a
summary of the installation.

2.1 Cluster Configuration

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

2. Install the cluster license by running the `request-license` command.
Because HA is used, specify the MAC address of the first NIC of the secondary head
node so that it can also serve the BCM licenses in the event of a failover.
This example is for a head node with Internet access. For air-gapped clusters, see
“Off-cluster WWW access” in Section 4.3.3 of the NVIDIA Base Command Manager

   ```
   # request-license
   Product Key (XXXXX-XXXXX-XXXXX-XXXXX-XXXXX): 123456-123456-123456-123456
   ```

3. Backup the default software image.
The backup image can be used to create additional software images.

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

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

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

4. Backup the DGX software image.

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

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

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

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

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

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

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

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

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

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

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

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

8. Create the DGX nodes.

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

% device
% foreach --clone node01 -n node02..node05 ()
% commit
9. Rename the DGX nodes so they are more easily identified later.

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

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

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

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

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

12. Rename node01.

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

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

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

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

14. Assign the K8s nodes to the k8s-master node category.

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

15. Check the nodes and their categories.

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

```bash
% device list -f hostname:20,category:10,ip:20,status:15
```

<table>
<thead>
<tr>
<th>hostname (key)</th>
<th>category</th>
<th>ip</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>bcm10-headnode</td>
<td></td>
<td>10.227.48.8</td>
<td>[ UP   ]</td>
</tr>
<tr>
<td>dgx01</td>
<td>dgx-a100</td>
<td>10.227.48.5</td>
<td>[ DOWN ]</td>
</tr>
<tr>
<td>dgx02</td>
<td>dgx-a100</td>
<td>10.227.48.6</td>
<td>[ DOWN ]</td>
</tr>
<tr>
<td>dgx03</td>
<td>dgx-a100</td>
<td>10.227.48.7</td>
<td>[ DOWN ]</td>
</tr>
<tr>
<td>dgx04</td>
<td>dgx-a100</td>
<td>10.227.48.4</td>
<td>[ DOWN ]</td>
</tr>
<tr>
<td>knode01</td>
<td>k8s-master</td>
<td>10.227.48.4</td>
<td>[ DOWN ]</td>
</tr>
<tr>
<td>knode02</td>
<td>k8s-master</td>
<td>10.227.48.4</td>
<td>[ DOWN ]</td>
</tr>
<tr>
<td>knode03</td>
<td>k8s-master</td>
<td>10.227.48.4</td>
<td>[ DOWN ]</td>
</tr>
<tr>
<td>template01</td>
<td>default</td>
<td>10.227.48.4</td>
<td>[ DOWN ]</td>
</tr>
</tbody>
</table>
2.1.1 Network Configuration

1. Add a Network for InfiniBand (ibnet).

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

2. Verify the results.

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

3. Ensure that head node interfaces are configured correctly.

   % device
   % use bcm10-headnode
   % interfaces
   % list

   Type         Network device name  IP               Network          Start if
   --           ----------------------- ----------------- -------------------
   bmc          ipmi0                10.227.20.91     ipminet          always
   physical     ens10f0              10.227.52.8      externalnet      always
   physical     ens10f1              10.227.20.126    ipminet          always
   physical     ens1f1np1 [prov]     10.227.48.8      internalnet      always

4. If any interfaces are missing or unconfigured, add any missing devices and configure their network as appropriate.

5. Reboot the head node if the network interfaces were changed.

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

2.1.2 Configure Disk Layouts for Node Categories

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

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

1. Augment the disksetup of the k8s-master category.
For the K8s control plane nodes, an EFI System Partition of 100 MB is created at the start of the disk, with the remainder of the disk dedicated to the OS as a single large partition. Note that this disk setup does not have a swap partition.

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

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

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<diskSetup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <device>
    <blockdev>/dev/nvme0n1</blockdev>
    <partition id="a0" partitiontype="esp">
      <size>100M</size>
      <type>linux</type>
      <filesystem>fat</filesystem>
      <mountPoint>/boot/efi</mountPoint>
      <mountOptions>defaults,noatime,nodiratime</mountOptions>
    </partition>
    <partition id="a1">
      <size>max</size>
      <type>linux</type>
      <filesystem>xfs</filesystem>
      <mountPoint>/</mountPoint>
      <mountOptions>defaults,noatime,nodiratime</mountOptions>
    </partition>
  </device>
</diskSetup>
```

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

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

### 2.1.3 Configure Node Network Interfaces

#### 2.1.3.1 Configure BCM to Allow MAC Addresses to PXE Boot

1. Use the root (not cmsh) shell.

2. In /cm/local/apps/cmd/etc/cmd.conf, uncomment the AdvancedConfig parameter.

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

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

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

### 2.1.3.2 Configure Provisioning Interfaces on the DGX Nodes

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

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

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

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

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

   ```
   # cmsh
   % device
   % use dgx01
   % interfaces
   % set enp225s0f1 mac B8:CE:F6:2F:08:69
   % set enp97s0f1 mac B8:CE:F6:2D:0E:A7
   % set enp225s0f1np1 mac B8:CE:F6:2F:08:69
   % set enp97s0f1np1 mac B8:CE:F6:2D:0E:A7
   % set ipmi0 ip 10.227.20.69
   % set bond0 ip 10.227.48.13
   % commit
   % list
   ```

<table>
<thead>
<tr>
<th>Type</th>
<th>Network device name</th>
<th>IP</th>
<th>Network</th>
<th>Start if</th>
</tr>
</thead>
<tbody>
<tr>
<td>bmc</td>
<td>ipmi0</td>
<td>10.227.20.69</td>
<td>ipminet</td>
<td>always</td>
</tr>
<tr>
<td>physical</td>
<td>BOOTIF [prov]</td>
<td>10.227.48.4</td>
<td>internalnet</td>
<td>always</td>
</tr>
<tr>
<td>bond</td>
<td>bond0</td>
<td>10.227.48.13</td>
<td>internalnet</td>
<td>always</td>
</tr>
<tr>
<td>physical</td>
<td>enp225s0f1</td>
<td>0.0.0.0</td>
<td>internalnet</td>
<td>always</td>
</tr>
<tr>
<td>physical</td>
<td>enp225s0f1np1</td>
<td>0.0.0.0</td>
<td>internalnet</td>
<td>always</td>
</tr>
<tr>
<td>physical</td>
<td>enp97s0f1</td>
<td>0.0.0.0</td>
<td>internalnet</td>
<td>always</td>
</tr>
<tr>
<td>physical</td>
<td>enp97s0f1np1</td>
<td>0.0.0.0</td>
<td>internalnet</td>
<td>always</td>
</tr>
</tbody>
</table>

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

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

4. Verify the configuration.

   ```
   % device
   % use dgx01
   ```
Configure Provisioning Interfaces on the K8s Nodes

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

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

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

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

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

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

   ```
   % /
   ```
2.1.3.4 Configure InfiniBand Interfaces on DGX Nodes

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

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

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

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

   ```
   % /         # go to top level of CMSH
   % device
   % use dgx01
   % interfaces
   % set ibp12s0 ip 10.126.0.13
   % set ibp141s0 ip 10.126.2.13
   % set ibp186s0 ip 10.126.3.13
   % set ibp75s0 ip 10.126.1.13
   % commit
   ```

<table>
<thead>
<tr>
<th>Type</th>
<th>Network device name</th>
<th>IP</th>
<th>Network</th>
<th>Start if</th>
</tr>
</thead>
<tbody>
<tr>
<td>bmc</td>
<td>ipmi0</td>
<td>10.227.20.69</td>
<td>ipminet</td>
<td>always</td>
</tr>
<tr>
<td>bond</td>
<td>bond0 [prov]</td>
<td>10.227.48.13</td>
<td>internalnet</td>
<td>always</td>
</tr>
<tr>
<td>physical</td>
<td>enp225s0f1 (bond0)</td>
<td>0.0.0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>physical</td>
<td>enp225s0f1np1 (bond0)</td>
<td>0.0.0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>physical</td>
<td>enp97s0f1 (bond0)</td>
<td>0.0.0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>physical</td>
<td>enp97s0f1np1 (bond0)</td>
<td>0.0.0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>physical</td>
<td>ibp12s0</td>
<td>10.126.0.13</td>
<td>ibnet</td>
<td>always</td>
</tr>
<tr>
<td>physical</td>
<td>ibp141s0</td>
<td>10.126.2.13</td>
<td>ibnet</td>
<td>always</td>
</tr>
<tr>
<td>physical</td>
<td>ibp186s0</td>
<td>10.126.3.13</td>
<td>ibnet</td>
<td>always</td>
</tr>
<tr>
<td>physical</td>
<td>ibp75s0</td>
<td>10.126.1.13</td>
<td>ibnet</td>
<td>always</td>
</tr>
</tbody>
</table>

2.1.3.5 Identify the Cluster Nodes

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

   ```
   % device
   % set dgx01 mac b8:ce:f6:2f:08:69
   % set dgx02 mac c0:42:a1:54:32:a7
   % set dgx03 mac 0c:42:a1:0a:7a:51
   % set dgx04 mac 1c:34:da:29:17:6e
   % set knode01 mac 04:3f:72:e7:64:97
   % set knode02 mac 04:3f:72:d3:fe:eb
   % set knode03 mac 04:3f:72:d3:fc:db
   % foreach -c dgx-a100,k8s-master (get mac)
   ```
2. If all the MAC addresses are set properly, commit the changes.
   % device commit
   % quit
   % commit

2.2 Power On and Provision Cluster Nodes

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

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

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

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

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

2. Choose Deploy to continue.

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

4. Leave items unselected in the screen because individual nodes will be specified in the next step.
5. bcm10-headnode is selected to install docker.

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

7. Enter any local Docker repositories on this next screen.
8. Do not install the NVIDIA Container Runtime on the head node since there is no GPU on that node.

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

10. By default, the Docker wizard will save the deployment configuration in /
    /root/cm-docker-setup.conf.
    This configuration file can be used to redeploy Docker in the future. Select ok to start
    the installation.
2.4 Deploy K8s

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

2. Choose Deploy to start the deployment.

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

4. Select Ok to confirm the Containerd container runtime.
5. Fill in the optional DockerHub registry mirror endpoint if necessary—otherwise, select Ok to continue.

6. Accept the default settings for this K8s cluster—to match the naming chosen for this guide, change the K8s cluster name to onprem.
7. Select yes to expose the K8s API server on the head node. This allows users to use the K8s cluster from the head node.

8. Select `internalnet` since the K8s control plane nodes and the DGX nodes, which are the K8s worker nodes, are all connected on `internalnet`. 
9. Select all three K8s control plane nodes: knode01, knode02, and knode03.

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

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

12. Select all three K8s control plane nodes: knode01, knode02, and knode03 for the Etcd nodes.
13. Accept the default values for the main Kubernetes components unless the organization requires specific ports.
14. Select the Calico network plugin when prompted.

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

16. Choose no to decline to configure HA for Kyverno and then select Ok.
17. Choose whether to install Kyverno Policies and then select Ok. Unless required for the configuration, choose no.

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

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


22. Select the defaults unless specific ingress ports are to be used.
23. Select no since the K8s control plane nodes do not have GPUs.

24. Select yes to deploy the Permission Manager.

25. Select both enabled and default for the Local path storage class.
26. Accept the default data storage path and leave the other two fields blank, which is 
the default.
27. Select Save config & deploy.
28. Change the filepath to `/root/cm-kubernetes-setup-onprem.conf` and select Ok. 
This file can be used to redeploy K8s or copied and modified to deploy additional K8s 
clusters.
Wait for the installation to finish.
29. Verify that the K8s cluster is installed properly.

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

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

```
# kubectl get nodes
<table>
<thead>
<tr>
<th>NAME</th>
<th>STATUS</th>
<th>ROLES</th>
<th>AGE</th>
<th>VERSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>dgx01</td>
<td>Ready</td>
<td>worker</td>
<td>7m36s</td>
<td>v1.27.4</td>
</tr>
<tr>
<td>dgx02</td>
<td>Ready</td>
<td>worker</td>
<td>7m37s</td>
<td>v1.27.4</td>
</tr>
<tr>
<td>dgx03</td>
<td>Ready</td>
<td>worker</td>
<td>7m36s</td>
<td>v1.27.4</td>
</tr>
<tr>
<td>dgx04</td>
<td>Ready</td>
<td>worker</td>
<td>7m37s</td>
<td>v1.27.4</td>
</tr>
<tr>
<td>knode01</td>
<td>Ready</td>
<td>control-plane,master</td>
<td>7m59s</td>
<td>v1.27.4</td>
</tr>
<tr>
<td>knode02</td>
<td>Ready</td>
<td>control-plane,master</td>
<td>7m26s</td>
<td>v1.27.4</td>
</tr>
<tr>
<td>knode03</td>
<td>Ready</td>
<td>control-plane,master</td>
<td>7m25s</td>
<td>v1.27.4</td>
</tr>
</tbody>
</table>
```

2.5 Deploying Slurm

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

1. Start the interactive setup tool.

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

2. Choose Setup (Step By Step) and then select OK.
3. Choose `slurm` as the workload management system and then select OK.
4. Enter the name for the Slurm cluster and then select OK.

5. Choose the head node as the Slurm server.

6. Set the overlay name and priority for the configuration overlay and then select OK. This example uses the defaults.
7. Choose yes to configure GPU resources and then select OK.

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

9. Assuming there are no compute nodes without GPUs, leave all the options unselected at the following screen and press OK.
10. Enter the overlay name and priority for Slurm clients *without* GPUs and then select OK. The example uses the defaults.

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

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

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

14. Select a priority for the configuration overlay. The example uses the default.
15. Leave the number of slots unconfigured.

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

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

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

20. Select the head node as the accounting node.

21. Add the 8 GPUs in each node as GPU resources that can be requested.
   It is also possible to rely on the Slurm GPU autodetect capabilities. Consult the BCM documentation for details.
22. Unless CUDA Multi Process Management (MPS) will be used, leave the MPS settings empty.

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

23. Enable the following cgroup resource constraints to make sure that jobs cannot use CPU cores or GPUs that they did not request:
24. Create a default queue.
   More queues can always be defined later:

![Queue selection screen]

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

![Save config & deploy screen]

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

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

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

```
[root@utilitynode-01 ~]# sinfo
PARTITION  AVAIL  TIMELIMIT  NODES  STATE  NODELIST
  defq*    up    infinite   8    idle  dgx-[01-08]
```
29. By default, Slurm is configured to not allow multiple jobs on the same node. To change this behavior and allow (for example) a maximum of 8 simultaneous jobs to run on a single node.

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

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

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

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

   [root@utilitynode-01 ~]# srun --gres=gpu:2 nvidia-smi

---

Thu Mar 4 08:50:44 2021

+-----------------------------------------------------------------------------+
| NVIDIA-SMI 450.102.04   Driver Version: 450.102.04   CUDA Version: 11.0     |
+-----------------------------------------------------------------------------+
| GPU Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap| Memory-Usage | GPU-Util  Compute M. |
| N/A   30C    P0    54W / 400W | 0MiB / 40537MiB | 0%      Default |
| N/A   30C    P0    53W / 400W | 0MiB / 40537MiB | 0%      Default |
+-----------------------------------------------------------------------------+

| Processes: |
| GPU GI CI PID Type Process name GPU Memory |
| ID ID Usage |
+-----------------------------------------------------------------------------+
| No running processes found |
+-----------------------------------------------------------------------------+
2.6  (Optional) Deploy Jupyter

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

2.6.1  Install Jupyter Using the CLI Wizard

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

2. Choose Deploy to continue.
3. Specify the overlay name and priority for the JupyterHub login nodes. By default, the Jupyter wizard will create a configuration overlay named `jupyterhub` with a priority of 500. Use the defaults unless there is an existing `jupyterhub` overlay.

4. Select `bcm10-headnode` and then Ok. After HA is configured, the `cm-jupyterhub` service will be set to always run on the active head node.
5. Select the default ports of 8000, 8901, and 8902 and select Ok. Users will access it on the active head node on port 8000.

6. Select Save config & deploy and then Ok.
7. Select `Ok` to start the installation.
   By default, the Jupyter wizard will save the deployment configuration in `/root/cm-jupyter-setup.conf`. This configuration file can be used to redeploy Jupyter in the future.

8. When the installation completes, the `cm-jupyter` service will automatically be started on the selected node.
   All users in the cluster (except the root user) will be able to login to Jupyterhub using a web browser at `http://<head-node-ip or FQDN>:8000`.
   Example: `http://10.227.52.254:8000`

9. If needed, a test user can be created with the following command:
   ```bash
   # cmsh -c "user; add jupyterhubuser; set password jupyterhubuser; commit"
   ```

10. Add the user to K8s.
    ```bash
    # cm-kubernetes-setuo --add-user jupyterhubuser --operators cm-jupyter-kernel-operator
    ```
Chapter 3. High Availability

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

1. Ensure that both head nodes are licensed.
   We provided the MAC address for the secondary head when we installed the cluster license (Section 3.2.2.1).
   
   % main licenseinfo | grep ^MAC

2. Configure the NFS shared storage.
   Mounts configured in fsmounts will be automatically mounted by the CMDaemon.
   
   % device
   % use master
   % fsmounts
   % add /nfs/general
   % set device 10.227.48.252:/var/nfs/general
   % set filesystem nfs
   % commit
   % show

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device</td>
<td>10.227.48.252:/var/nfs/general</td>
</tr>
<tr>
<td>Revision</td>
<td></td>
</tr>
<tr>
<td>Filesystem</td>
<td>nfs</td>
</tr>
<tr>
<td>Mountpoint</td>
<td>/nfs/general</td>
</tr>
<tr>
<td>Dump</td>
<td>no</td>
</tr>
<tr>
<td>RDMA</td>
<td>no</td>
</tr>
<tr>
<td>Filesystem Check</td>
<td>NONE</td>
</tr>
<tr>
<td>Mount options</td>
<td>defaults</td>
</tr>
</tbody>
</table>

3. Verify that the shared storage is mounted.

   # mount | grep '/nfs/general'
   10.227.48.252:/var/nfs/general on /nfs/general type nfs4
   (rw, relatime, vers=4.2, rsize=1048576, wsize=1048576, namlen=255, hard, proto=tcp, timeo=600, retran
   ns=2, sec=sys, clientaddr=10.130.122.10.227.48._lock=none, addr=10.130.122.252)10.227.48
4. Verify that head node has power control over the cluster nodes.

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

5. Power off the cluster nodes.

The cluster nodes must be powered off before configuring HA.

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

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

```
# cmha-setup
```

7. Choose Setup and then select SELECT.
8. Choose Configure and then select NEXT

![Configure screen](image1.png)

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

![License information](image2.png)
10. Configure an external virtual IP address to be used by the active head node in the HA configuration and then select NEXT.

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

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

13. DGX BasePOD uses the internal network as the failover network, so select SKIP.
14. Configure the IP addresses for the secondary head node and then select NEXT.

15. Review the summary of the configuration and then select NEXT. This screen shows the VIP that will be assigned to the internal and external interfaces.
16. Select Yes to proceed with the failover configuration.

17. Enter the MySQL root password and then select OK. This should be the same as the root password.
18. The wizard implements the first steps in the HA configuration. If all the steps show OK, press ENTER to continue. The progress is shown below:

- Initializing failover setup on master .......... [ OK ]
- Updating shared internal interface .......... [ OK ]
- Updating shared external interface .......... [ OK ]
- Updating extra shared internal interfaces .... [ OK ]
- Cloning head node .............................. [ OK ]
- Updating secondary master interfaces ........ [ OK ]
- Updating Failover Object ...................... [ OK ]
- Restarting cmdaemon ........................... [ OK ]

Press any key to continue.

19. When the failover setup installation on the primary master is complete, select OK to exit the wizard.
20. PXE boot the secondary head node and then select RESCUE from the grub menu. Since this is the initial boot of this node, it must be done outside of Base Command Manager (BMC or physical power button).

21. After the secondary head node has booted into the rescue environment, run the `/cm/cm-clone-install -failover` command, then enter yes when prompted. The secondary head node will be cloned from the primary.
22. When cloning is completed, enter `y` to reboot the secondary head node. The secondary must boot from its hard drive. PXE boot should not be enabled.

```
ClusteringManager login: root (automatic login)
Linux ClusterManager 5.13.0-99-generic #1 SMP Thu Apr 24 16:43:35 UTC 2022 x86_64
node/ClusterManager: # source /etc/ClusterManagerliste/ha
Welcome to the Cluster Manager menu environment:
| 1 | Creating failover/cron lates:               |
| 2 | Install the secondary head node             |
| 3 | Source close-install --failover             |
| 4 | Create a clone of the primary head node      |
| 5 | Source close-install --hostname=hostname     |
| 6 | Install the secondary (failover) head node and reboot automatically    |
| 7 | Source close-install --failover --reboot    |
| 8 | Help                                       |
| 9 | Source close-install --help                  |
```

23. Wait for the secondary head node to reboot and then continue the HA setup procedure on the primary head node.

24. Choose finalize from the cmha-setup menu and then select NEXT. This will clone the MySQL database from the primary to the secondary head node.
25. Select **CONTINUE** on the confirmation screen.

26. Enter the MySQL root password and then select **OK**.  
   This should be the same as the root password.
27. The cmha-setup wizard continues. Press ENTER to continue when prompted.

The progress is shown below:

- Updating secondary master mac address.............. [ OK ]
- Initializing failover setup on basepod-head2........ [ OK ]
- Stopping cmdaemon.................................. [ OK ]
- Cloning cmdaemon database.......................... [ OK ]
- Checking database consistency...................... [ OK ]
- Starting cmdaemon, chkconfig services.............. [ OK ]
- Cloning workload manager databases................. [ OK ]
- Cloning additional databases....................... [ OK ]
- Update DB permissions.............................. [ OK ]
- Checking for dedicated failover network............ [ OK ]
- Press any key to continue
28. Select **REBOOT** when the **WARNING: REBOOT REQUIRED** screen is shown. Wait for the secondary head node to reboot before continuing.

![Screenshot of WARNING: REBOOT REQUIRED message]

29. The secondary head node is now UP.

```bash
% device list -f hostname:20,category:12,ip:20,status:15

+-----------------------+--------------+-----------------+--------+
| hostname (key)         | category     | ip              | status |
+-----------------------+--------------+-----------------+--------+
| basepod-head1          |              | 10.227.48.254   | [ UP ] |
| basepod-head2          |              | 10.227.48.253   | [ UP ] |
| knode01                | k8s-master   | 10.227.48.9     | [ DOWN ]|
| knode02                | k8s-master   | 10.227.48.10    | [ DOWN ]|
| knode03                | k8s-master   | 10.227.48.11    | [ DOWN ]|
| dgx01                  | dgx          | 10.227.48.5     | [ DOWN ]|
| dgx02                  | dgx          | 10.227.48.6     | [ DOWN ]|
| dgx03                  | dgx          | 10.227.48.7     | [ DOWN ]|
| dgx04                  | dgx          | 10.227.48.8     | [ DOWN ]|
```
30. Choose Shared Storage from the cmha-setup menu and select SELECT.
In this final HA configuration step, cmha-setup will copy the /cm/shared and /home directories to the shared storage, and it configures both head nodes and all cluster nodes to mount it.

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

33. Provide the IP address of the NAS host, the paths for the `/cm/shared` and `/home` directories should be copied to on the shared storage and then select NEXT. The values are from In this case, `/var/nfs/general` is exported, so the `/cm/shared` directory will be copied to 10.227.48.252:/var/nfs/general/cmshared, and it will be mounted over `/cm/shared` on the cluster nodes.
34. The wizard shows a summary of the information that it has collected. Press ENTER to continue.

35. Select YES when prompted to proceed with the setup.

36. The cmha-setup wizard proceeds with its work. When it completes, select ENTER to finish the HA setup.
The progress is shown below:

<table>
<thead>
<tr>
<th>Task</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copying NAS data</td>
<td>OK</td>
</tr>
<tr>
<td>Mount NAS storage</td>
<td>OK</td>
</tr>
<tr>
<td>Remove old fmounts</td>
<td>OK</td>
</tr>
<tr>
<td>Add new fmounts</td>
<td>OK</td>
</tr>
<tr>
<td>Remove old fexports</td>
<td>OK</td>
</tr>
<tr>
<td>Write NAS mount/unmount scripts</td>
<td>OK</td>
</tr>
<tr>
<td>Copy mount/unmount scripts</td>
<td>OK</td>
</tr>
</tbody>
</table>

Press any key to continue

37. cmha-setup is now complete. Select EXIT to return to the shell prompt.
3.1.1 Verify the HA Setup

1. Run the `cmha status` command to verify that the failover configuration is correct and working as expected.
   
   Note that the command tests the configuration from both directions: from the primary head node to the secondary, and from the secondary to the primary. The active head node is indicated by an asterisk.

   ```
   # cmha status
   Node Status: running in active mode
   
   basepod-head1* -> basepod-head2
   mysql     [ OK ]
   ping      [ OK ]
   status    [ OK ]
   
   basepod-head2 -> basepod-head1*
   mysql     [ OK ]
   ping      [ OK ]
   status    [ OK ]
   ```

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

   ```
   # mount
   . . . some output omitted . . .
   10.227.48.252:/var/nfs/general/cmshared on /cm/shared type nfs4
   (rw,relatime,vers=4.2,rsize=32768,wsize=32768,namlen=255,hard,proto=tcp,timeo=600,retrans=2
   ,sec=sys,clientaddr=10.130.122.252,addr=10.130.122.252)10.227.48
   10.227.48.252:/var/nfs/general/home on /home type nfs4
   (rw,relatime,vers=4.2,rsize=32768,wsize=32768,namlen=255,hard,proto=tcp,timeo=600,retrans=2
   ,sec=sys,clientaddr=10.130.122.252,addr=10.130.122.252)10.227.48
   ```

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

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

   basepod-head2 will become active head node (current state: passive)
   basepod-head1 will become passive head node (current state: active)
   
  Continue(c)/Exit(e)? c
   
   Initiating failover......................... [ OK ]
   ```

   ```
   basepod-head2 is now active head node, makeactive successful
   ```
4. Run the `cmha status` command again to verify that the secondary head node has become the active head node.

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

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

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

5. Manually failover back to the primary head node.

```
# ssh basepod-head1
# cmha makeactive

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

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

Continue(c)/Exit(e)? c

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

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

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

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

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

basepod-head2 -> basepod-head1*
  mysql  [ OK ]
  ping   [ OK ]
  status [ OK ]
```
7. Power on the cluster nodes.

   # cmsh -c "power -c k8s-master,dgx on"

   ipmi0  .................... [   ON    ] knode01
   ipmi0  .................... [   ON    ] knode02
   ipmi0  .................... [   ON    ] knode03
   ipmi0  .................... [   ON    ] dgx01
   ipmi0  .................... [   ON    ] dgx02
   ipmi0  .................... [   ON    ] dgx03
   ipmi0  .................... [   ON    ] dgx04

8. (Optionally) Configure Jupyter HA

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

   % device
   % use basepod-head1
   % services
   % use cm-jupyterhub
   % show

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

9. Set the runif parameter to active.

   % set runif active
   % commit

   % show

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

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

    % device
    % use basepod-head2
    % services
    % use cm-jupyterhub
    % set runif active
Chapter 4. Basic User Management

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

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

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

4.1 Configuring a User

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

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

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

```
userone@basepod-head2:~$ passwd
(current) LDAP Password:
New password:
Retype new password:
passwd: password updated successfully
userone@basepod-head2:~$
```
3. Use show to view user parameters and values.

```
[basedpod-head2->user[userone]]% show
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accounts</td>
<td></td>
</tr>
<tr>
<td>Managees</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>userone</td>
</tr>
<tr>
<td>Primary group</td>
<td>userone</td>
</tr>
<tr>
<td>Revision</td>
<td></td>
</tr>
<tr>
<td>Secondary groups</td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>1004</td>
</tr>
<tr>
<td>Common name</td>
<td>userone</td>
</tr>
<tr>
<td>Surname</td>
<td>userone</td>
</tr>
<tr>
<td>Group ID</td>
<td>1004</td>
</tr>
<tr>
<td>Login shell</td>
<td>/bin/bash</td>
</tr>
<tr>
<td>Home directory</td>
<td>/home/ userone</td>
</tr>
<tr>
<td>Password</td>
<td>*********</td>
</tr>
<tr>
<td>email</td>
<td></td>
</tr>
<tr>
<td>Create cmjob certificate</td>
<td>no</td>
</tr>
<tr>
<td>Write ssh proxy config</td>
<td>no</td>
</tr>
<tr>
<td>Shadow min</td>
<td>0</td>
</tr>
<tr>
<td>Shadow max</td>
<td>999999</td>
</tr>
<tr>
<td>Shadow warning</td>
<td>7</td>
</tr>
<tr>
<td>Inactive</td>
<td>0</td>
</tr>
<tr>
<td>Last change</td>
<td>2022/10/20</td>
</tr>
<tr>
<td>Expiration date</td>
<td>2037/12/31</td>
</tr>
<tr>
<td>Project manager</td>
<td>&lt;submode&gt;</td>
</tr>
<tr>
<td>Notes</td>
<td>&lt;0B&gt;</td>
</tr>
</tbody>
</table>

4. Use set to change parameters.

```
[basedpod-head2->user[userone]]% set
```

<table>
<thead>
<tr>
<th>commonname</th>
<th>expirationdate</th>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>profile</td>
<td>shadowmax</td>
<td>surname</td>
<td></td>
</tr>
<tr>
<td>createcmjobcertificate</td>
<td>groupid</td>
<td>inactive</td>
<td>notes</td>
</tr>
<tr>
<td>projectmanager</td>
<td>shadowmin</td>
<td>writesshproxyconfig</td>
<td></td>
</tr>
<tr>
<td>email</td>
<td>homedirectory</td>
<td>loginshell</td>
<td>password</td>
</tr>
<tr>
<td>revision</td>
<td>shadowwarning</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.1.1 Procedures to Remove a User

This block of code will delete a user.

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

Adding the -d option to remove will also delete the home directory.
4.2 Adding a User to K8s

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

Add each K8s user with cm-kubernetes-setup.

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

Took: 00:06 min.
Progress: 100/100

######################## Finished execution for 'Kubernetes Setup', status: completed

Kubernetes Setup finished!
4.3 Removing a User from K8s

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

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

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

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

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

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

**Table 3. General information**

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NFS server IP</td>
<td>10.227.48.252</td>
</tr>
<tr>
<td>NFS server export</td>
<td>/var/nfs/general</td>
</tr>
<tr>
<td>Head node drive path</td>
<td>nvme0n1</td>
</tr>
<tr>
<td>Cluster name</td>
<td>BasePOD</td>
</tr>
<tr>
<td>Organization</td>
<td>NVIDIA</td>
</tr>
<tr>
<td>Timezone</td>
<td>US/Los_Angeles</td>
</tr>
<tr>
<td>Nameservers</td>
<td>8.8.8.8</td>
</tr>
<tr>
<td>Search domains</td>
<td>example.com</td>
</tr>
</tbody>
</table>
Table 4. BCM head node information

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Head node 1 name</td>
<td>basepod-head1</td>
</tr>
<tr>
<td>Head node 1 and 2 administrator password</td>
<td>ExamplePassword1234!@#$</td>
</tr>
<tr>
<td>Head node 1 BMC IP (ipminet)</td>
<td>10.227.20.66</td>
</tr>
<tr>
<td>Head node 1 Ethernet device (externalnet)</td>
<td>enp10</td>
</tr>
<tr>
<td>Head node 1 IP (externalnet)</td>
<td>10.227.52.254</td>
</tr>
<tr>
<td>Head node 1 Ethernet device (internalnet)</td>
<td>enp10</td>
</tr>
<tr>
<td>Head node 1 IP (internalnet)</td>
<td>10.227.48.254</td>
</tr>
<tr>
<td>Head node 2 name</td>
<td>basepod-head2</td>
</tr>
<tr>
<td>Head node 2 BMC IP (ipminet)</td>
<td>10.227.20.67</td>
</tr>
<tr>
<td>Head node 2 Ethernet device (externalnet)</td>
<td>enp10</td>
</tr>
<tr>
<td>Head node 2 IP (externalnet)</td>
<td>10.227.52.253</td>
</tr>
<tr>
<td>Head node 2 Ethernet device (internalnet)</td>
<td>enp10</td>
</tr>
<tr>
<td>Head node 2 IP (internalnet)</td>
<td>10.227.48.253</td>
</tr>
</tbody>
</table>
Table 5. Network information

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>K8s node name template</td>
<td>knode##</td>
</tr>
<tr>
<td>ipminet base IP</td>
<td>10.227.20.64</td>
</tr>
<tr>
<td>ipminet netmask</td>
<td>255.255.255.192</td>
</tr>
<tr>
<td>ipminet gateway</td>
<td>10.227.20.65</td>
</tr>
<tr>
<td>ipminet switch ASN</td>
<td></td>
</tr>
<tr>
<td>internalnet switch #1 ASN (for externalnet)</td>
<td></td>
</tr>
<tr>
<td>internalnet switch #2 ASN (for externalnet)</td>
<td></td>
</tr>
<tr>
<td>externalnet base IP</td>
<td>10.227.52.0</td>
</tr>
<tr>
<td>externalnet netmask</td>
<td>255.255.255.0</td>
</tr>
<tr>
<td>externalnet gateway</td>
<td>10.227.52.1</td>
</tr>
<tr>
<td>Domain</td>
<td>example.com</td>
</tr>
<tr>
<td>internalnet base IP</td>
<td>10.227.48.0</td>
</tr>
<tr>
<td>internalnet netmask</td>
<td>255.255.255.0</td>
</tr>
<tr>
<td>ibnet base IP</td>
<td>10.149.0.0</td>
</tr>
<tr>
<td>ibnet netmask</td>
<td>255.255.0.0</td>
</tr>
<tr>
<td>Item</td>
<td>Value</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>DGX Node 1 name</td>
<td>dgx01</td>
</tr>
<tr>
<td>DGX Node 1 MAC (enp225s0f0/Management)</td>
<td>B8:CE:F6:2F:08:69</td>
</tr>
<tr>
<td>DGX Node 1 MAC (enp97s0f0/Management)</td>
<td>B8:CE:F6:2D:0E:A7</td>
</tr>
<tr>
<td>DGX Node 2 name</td>
<td>dgx02</td>
</tr>
<tr>
<td>DGX Node 2 MAC (enp225s0f0/Management)</td>
<td>B8:CE:F6:2F:08:69</td>
</tr>
<tr>
<td>DGX Node 2 MAC (enp97s0f0/Management)</td>
<td>B8:CE:F6:2D:0E:A7</td>
</tr>
<tr>
<td>DGX Node 3 name</td>
<td>dgx03</td>
</tr>
<tr>
<td>DGX Node 3 MAC (enp225s0f0/Management)</td>
<td>B8:CE:F6:2F:08:69</td>
</tr>
<tr>
<td>DGX Node 3 MAC (enp97s0f0/Management)</td>
<td>B8:CE:F6:2D:0E:A7</td>
</tr>
<tr>
<td>DGX Node 4 name</td>
<td>dgx04</td>
</tr>
<tr>
<td>DGX Node 4 MAC (enp225s0f0/Management)</td>
<td>B8:CE:F6:2F:08:69</td>
</tr>
<tr>
<td>DGX Node 4 MAC (enp97s0f0/Management)</td>
<td>B8:CE:F6:2D:0E:A7</td>
</tr>
</tbody>
</table>
### Table 7. K8s node information

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>K8S Node 1 interface 1 (Management)</td>
<td>ens1f1np1</td>
</tr>
<tr>
<td>K8S Node 1 interface 2 (Management)</td>
<td>ens2f1np1</td>
</tr>
<tr>
<td>K8S Node 1 interface 1 (Management) MAC</td>
<td>04:3F:72:E7:64:97</td>
</tr>
<tr>
<td>K8S Node 1 interface 2 (Management) MAC</td>
<td>0C:42:A1:79:9B:15</td>
</tr>
<tr>
<td>K8S Node 2 interface 1 (Management)</td>
<td>ens1f1np1</td>
</tr>
<tr>
<td>K8S Node 2 interface 2 (Management)</td>
<td>ens2f1np1</td>
</tr>
<tr>
<td>K8S Node 2 interface 1 (Management) MAC</td>
<td>04:3F:72:E7:64:97</td>
</tr>
<tr>
<td>K8S Node 2 interface 2 (Management) MAC</td>
<td>0C:42:A1:79:9B:15</td>
</tr>
<tr>
<td>K8S Node 3 interface 1 (Management)</td>
<td>ens1f1np1</td>
</tr>
<tr>
<td>K8S Node 3 interface 2 (Management)</td>
<td>ens2f1np1</td>
</tr>
<tr>
<td>K8S Node 3 interface 1 (Management) MAC</td>
<td>04:3F:72:E7:64:97</td>
</tr>
<tr>
<td>K8S Node 3 interface 2 (Management) MAC</td>
<td>0C:42:A1:79:9B:15</td>
</tr>
</tbody>
</table>
Appendix B. Switch Configurations

Switch configuration files are captured in this section.

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

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

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

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

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

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

### BGP unnumbered configuration (NOTE: no IPs need to be configured on the BGP interfaces, when using BGP unnumbered)

nv set router bgp autonomous-system 4200000003
nv set router bgp enable on
nv set router bgp router-id 10.227.20.78
nv set vrf default router bgp address-family ipv4-unicast enable on
nv set vrf default router bgp address-family ipv4-unicast redistribute connected enable on
nv set vrf default router bgp enable on
nv set vrf default router bgp neighbor swp64 remote-as external
nv set vrf default router bgp neighbor swp64 type unnumbered
nv set vrf default router bgp neighbor swp63 remote-as external
nv set vrf default router bgp neighbor swp63 type unnumbered
nv set vrf default router bgp neighbor swp60 remote-as external
nv set vrf default router bgp neighbor swp60 type unnumbered
nv set vrf default router bgp neighbor peerlink.4094 remote-as internal
nv set vrf default router bgp neighbor peerlink.4094 type unnumbered

### apply and save configuration
nv config diff
nv config apply
B.2 SN4600 #2 (In-band Management Switch)

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

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

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

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

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

### BGP unnumbered configuration (NOTE: no IPs need to be configured on the BGP interfaces, when using BGP unnumbered)

nv set router bgp autonomous-system 4200000003
nv set router bgp enable on
nv set router bgp router-id 10.227.20.79
nv set vrf default router bgp address-family ipv4-unicast enable on
nv set vrf default router bgp address-family ipv4-unicast redistribute connected enable on
nv set vrf default router bgp enable on
nv set vrf default router bgp neighbor swp63 remote-as external
nv set vrf default router bgp neighbor swp63 type unnumbered
nv set vrf default router bgp neighbor swp64 remote-as external
nv set vrf default router bgp neighbor swp64 type unnumbered
nv set vrf default router bgp neighbor swp60 remote-as external
nv set vrf default router bgp neighbor swp60 type unnumbered
nv set vrf default router bgp neighbor peerlink.4094 remote-as internal
nv set vrf default router bgp neighbor peerlink.4094 type unnumbered

### apply and save configuration

nv config diff
nv config apply
nv config save
B.3  SN2201 (Out-of-band Management Switch)

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

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

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

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

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

B.4  Ethernet Network Configuration Verifications

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

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

root@TOR-01:mgmt:~#
```

### CLAG Interfaces

<table>
<thead>
<tr>
<th>Our Interface</th>
<th>Peer Interface</th>
<th>CLAG Id</th>
<th>Conflicts</th>
<th>Proto-Down Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### verifying an access port (below is a access port with VLAN set to 111)

```bash
network-admin@IPMI-01:mgmt:~$ net show int swp1

<table>
<thead>
<tr>
<th>Name</th>
<th>MAC</th>
<th>Speed</th>
<th>MTU</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1G</td>
<td>1500</td>
<td>Access/L2</td>
</tr>
</tbody>
</table>

Alias
-----

bcm-bootstrap:eth0

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

Untagged
------

111

### verifying a bonded interface in trunk mode (note the native VLAN is set to 122)

```bash
network-admin@TOR-01:mgmt:~$ net show int bond9

<table>
<thead>
<tr>
<th>Name</th>
<th>MAC</th>
<th>Speed</th>
<th>MTU</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>100G</td>
<td>9216</td>
<td>802.3ad</td>
</tr>
</tbody>
</table>

Bond Details
-------------

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

All VLANs on L2 Port
------------------
1,121,122
## BGP verifications

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

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

Total number of neighbors 4
```

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

```bash
<table>
<thead>
<tr>
<th>Neighbor</th>
<th>V</th>
<th>AS</th>
<th>MsgRcvd</th>
<th>MsgSent</th>
<th>TblVer</th>
<th>InQ</th>
<th>OutQ</th>
<th>Up/Down</th>
<th>State/PfxRcd</th>
<th>PfxSnt</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOR-01(peerlink.4094)</td>
<td>4</td>
<td>4200000003</td>
<td>2692</td>
<td>2692</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>02:14:20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spine-01(swp63)</td>
<td>4</td>
<td>4200000001</td>
<td>2641</td>
<td>2642</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>02:11:48</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPMI-01(swp60)</td>
<td>4</td>
<td>4200000004</td>
<td>2467</td>
<td>2469</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>02:03:07</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Total number of neighbors 4
```

cumulus@IPMI-01:mgmt:~$ net show bgp summary
show bgp ipv4 unicast summary
=======================================
BGP router identifier 10.227.20.77, local AS number 4200000004 vrf-id 0
BGP table version 3
RIB entries 5, using 1000 bytes of memory
Peers 4, using 46 KiB of memory

```bash
<table>
<thead>
<tr>
<th>Neighbor</th>
<th>V</th>
<th>AS</th>
<th>MsgRcvd</th>
<th>MsgSent</th>
<th>TblVer</th>
<th>InQ</th>
<th>OutQ</th>
<th>Up/Down</th>
<th>State/PfxRcd</th>
<th>PfxSnt</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOR-01(swp51)</td>
<td>4</td>
<td>4200000003</td>
<td>2495</td>
<td>2494</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>02:04:30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOR-02(swp52)</td>
<td>4</td>
<td>4200000003</td>
<td>2495</td>
<td>2494</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>02:04:30</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Total number of neighbors 2
Notice
This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation ("NVIDIA") makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality. NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete. NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document. No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA. Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

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

Trademarks
NVIDIA, the NVIDIA logo, DGX, DGX NVIDIA BasePOD, and NVIDIA Base Command Manager are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright
© 2023 NVIDIA Corporation. All rights reserved.