Base Command Manager Administration

LAB GUIDE

Scope & Audience

This workbook covers administration tasks for Base Command Manager. It is intended for technical training students. The target audience for this course is system administrators that will help with the setup and upkeep of clusters using Base Command Manager.

Objectives

By the end of this workbook, students will be able to:

- Provision different types of nodes in the cluster.
- Manage software images.
- Configure and maintain node categories and groups.
- Run workloads on the cluster.
- Build custom monitoring.

Overview

Each student or group of students will be connecting to a virtual environment in the NVIDIA LaunchPad platform.

> **Notice**
> Please follow the instructions below carefully to successfully complete the practice.
> If you encounter technical issues, please contact the NVIDIA Academy team:
> academy-support@nvidia.com

Good Luck,
NVIDIA Academy Team

Revision #6.0 | August 2025

Table of Contents

Prerequisites and Guidelines

Notice
Please make sure you have reviewed and completed the following prerequisites before starting this hands-on lab guide.

There are no prerequisites for this lab.

Launchpad Lab Topology

The Base Command Manager lab environment is hosted on VMWare vSphere.

Lab components:

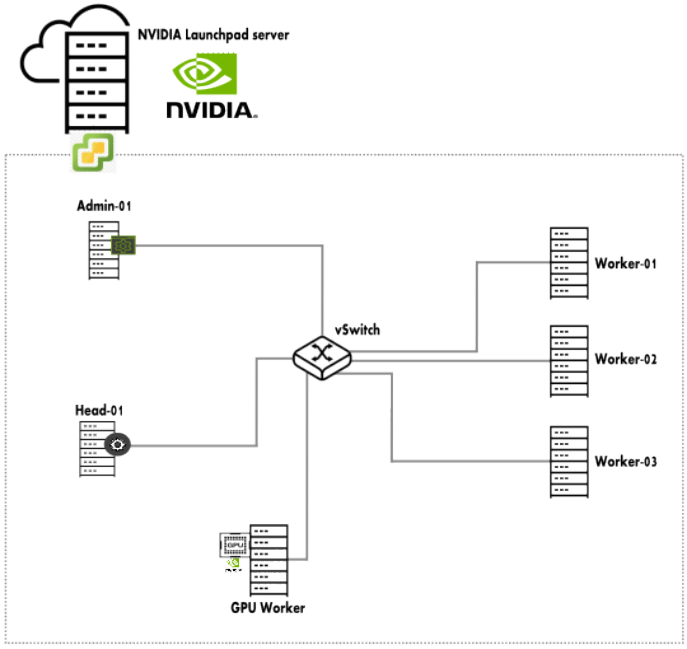
- **Admin Node**
  The Admin-01 node has a GUI through which you can login to other nodes and to the *Base View* app on the head node.

- **Head Node**
  The Head-01 node is the head node for the Base Command Manager cluster. The SSH connection to this node is used for access to CMSH.

- **Worker Nodes (node001-node003)**
  There are 3 worker nodes that are waiting to be provisioned.

- **GPU Worker Node (node004)**
  This node was deployed with an L40s vGPU, it is waiting to be provisioned.

The training lab is organized in the following topology:



Practice 0: File and Script Downloads

Practice Objectives:

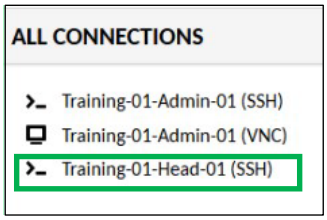
In this practice session you download scripts and files that will be used during the other practices:

You will:

- Login to the Base Command Manager Head Node.
- Download setup script.
- Run the setup script.

Task 1: Run the Setup Script

a. Login to the head node (**Head-01**).



b. The credentials are: username: **root** password: **bcm123.**

c. Once logged in, run the setup script:

```
./setup_lab.sh
```

```
./setup_lab.sh
Example output:
root@bcm:~# ./setup_lab.sh
Copying lab files
Creating 'nodes.csv' for Training-01
Done!
root@bcm:~#
```

Practice 1: Node Discovery and Provisioning

Practice Objectives:
In this practice session you will become familiar with the node discovery process and initial provisioning for Base Command Manager:

You will:

- Login to the Base Command Manager Head Node.
- Provision an unidentified node in the cluster.
- Provision a predefined node in the cluster.
- Get familiar with the Base View GUI and the CMSH CLI tool.

Task 1: Provision an Unidentified Node

a. Using the head node (**Head-01**) SSH, start CMSH:

```
cmsh
```

b. Enter the device mode:

```
device
```

c. List the nodes:

```
list
```

Example output:

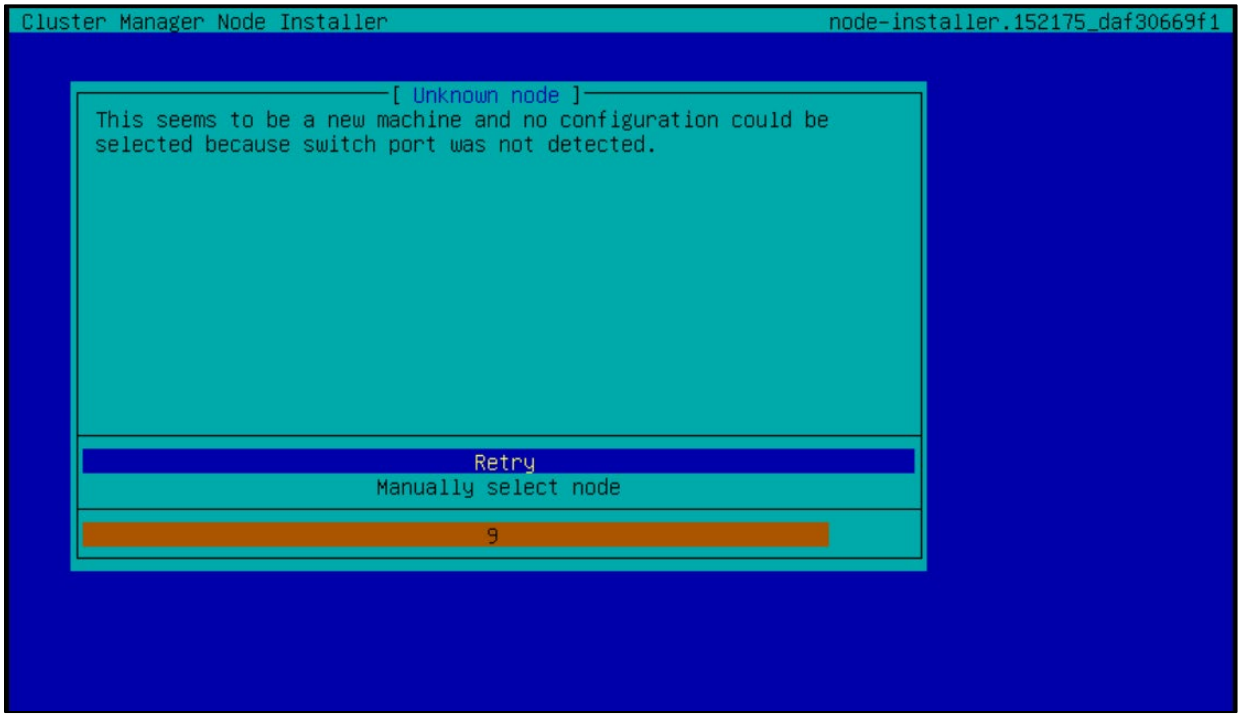
```
root@bcm:~# cmsh
[bcm]% device
[bcm->device]% list
Type                  Hostname (key)   MAC                Category
Ip              Network        Status
--------------------- --------------- ------------------ --------------- -
-------------- -------------- ----------------------
HeadNode              bcm              00:50:56:8A:7B:D1
10.141.255.254  Internalnet    [   UP   ]
PhysicalNode          node001          00:00:00:00:00:00  default
10.141.0.1      Internalnet    [  DOWN  ], unassigned
PhysicalNode          node002          00:00:00:00:00:00  default
10.141.0.2      Internalnet    [  DOWN  ], unassigned
PhysicalNode          node003          00:00:00:00:00:00  default
10.141.0.3      Internalnet    [  DOWN  ], unassigned
PhysicalNode          node004          00:00:00:00:00:00  default
10.141.0.4      Internalnet    [  DOWN  ], unassigned
[bcm->device]%
```

Ensure the head node is up and operational. The Status should be [UP].
If not, contact the instructor to assist with any issues.

d. The cluster nodes in the lab environment were pre-set for PXE boot.
Note: In your production environment, you may need to set the nodes to PXE boot.

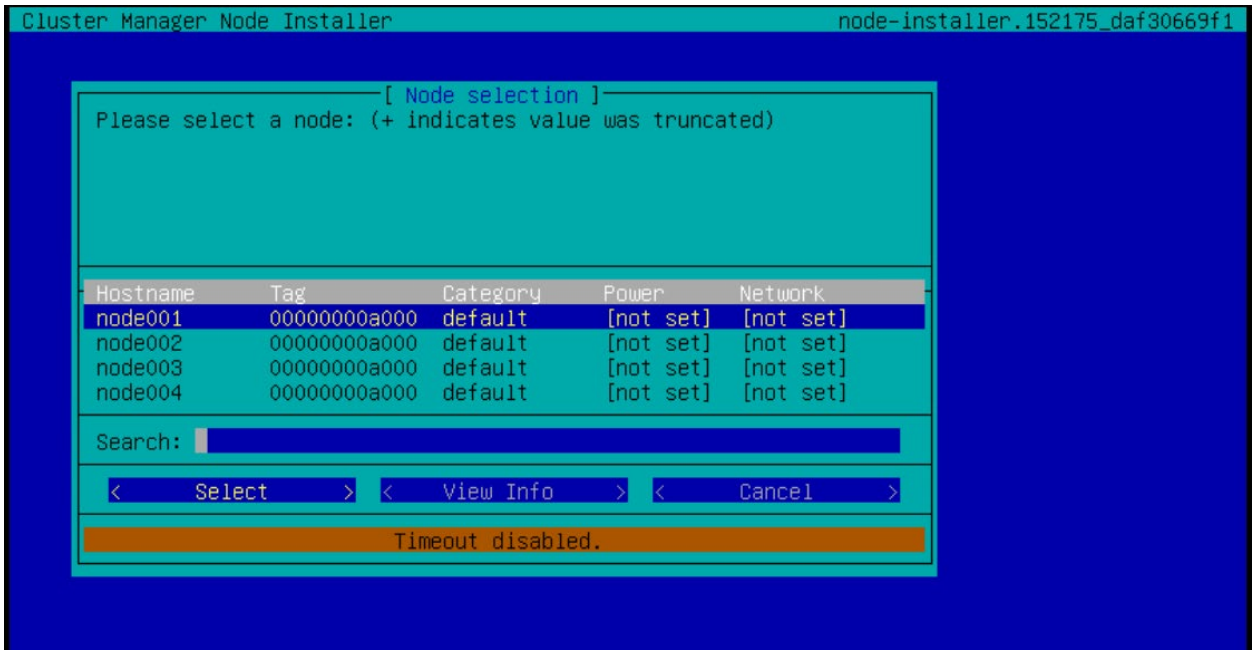
e. From the **Admin VNC desktop**, click the icon for **Worker01**.

f. Once open, you will see the following screen:

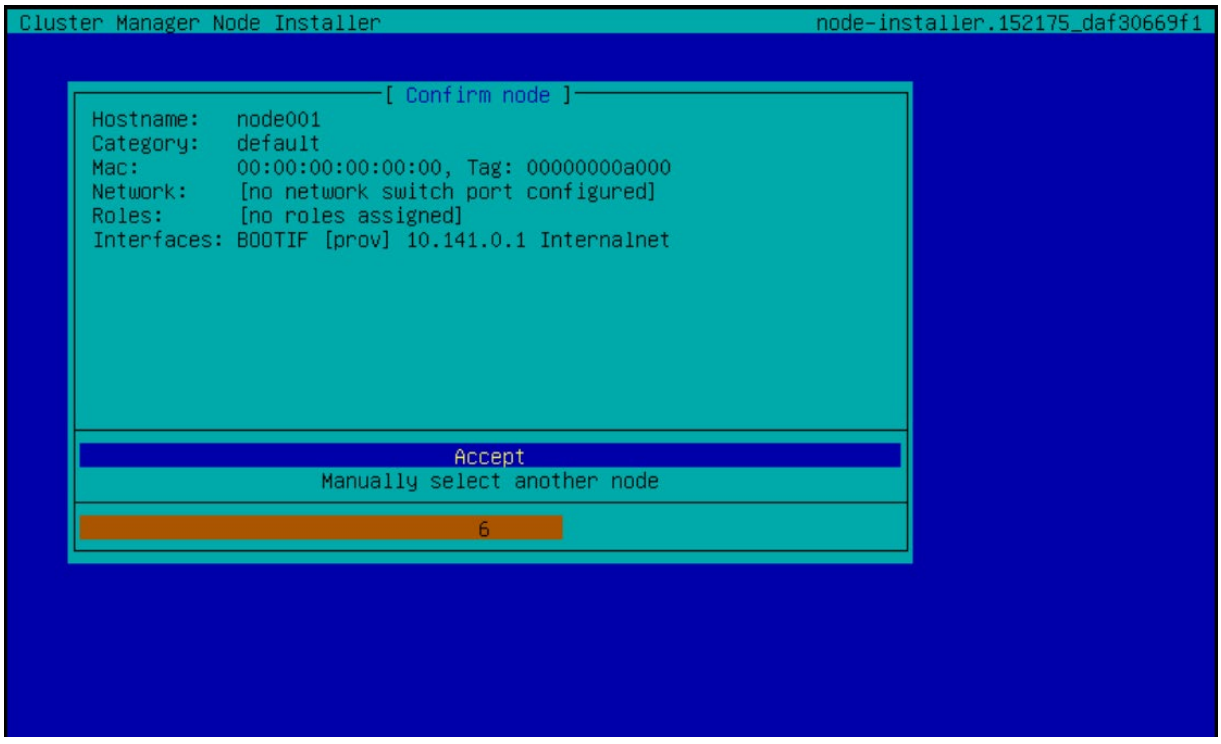


Using the keyboard, press the down arrow to highlight **Manually select node** and then press the **Enter** key.

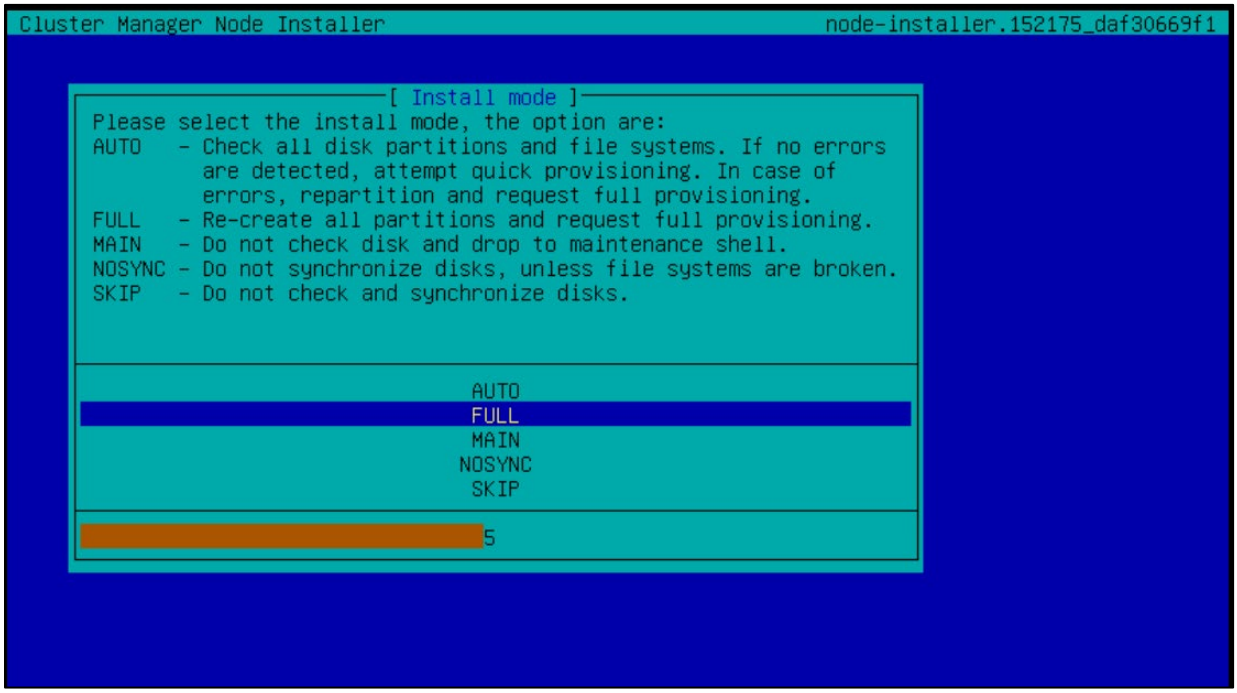
g. Using the keyboard arrow keys, highlight **node001** then press the **Tab** key to move and highlight the **Select** option. Press the **Enter** key to activate the **Select** button.



h. A screen will appear showing the **Hostname**, **Category**, **MAC** address and a few other parameters. The MAC address will be all zeros at this point.

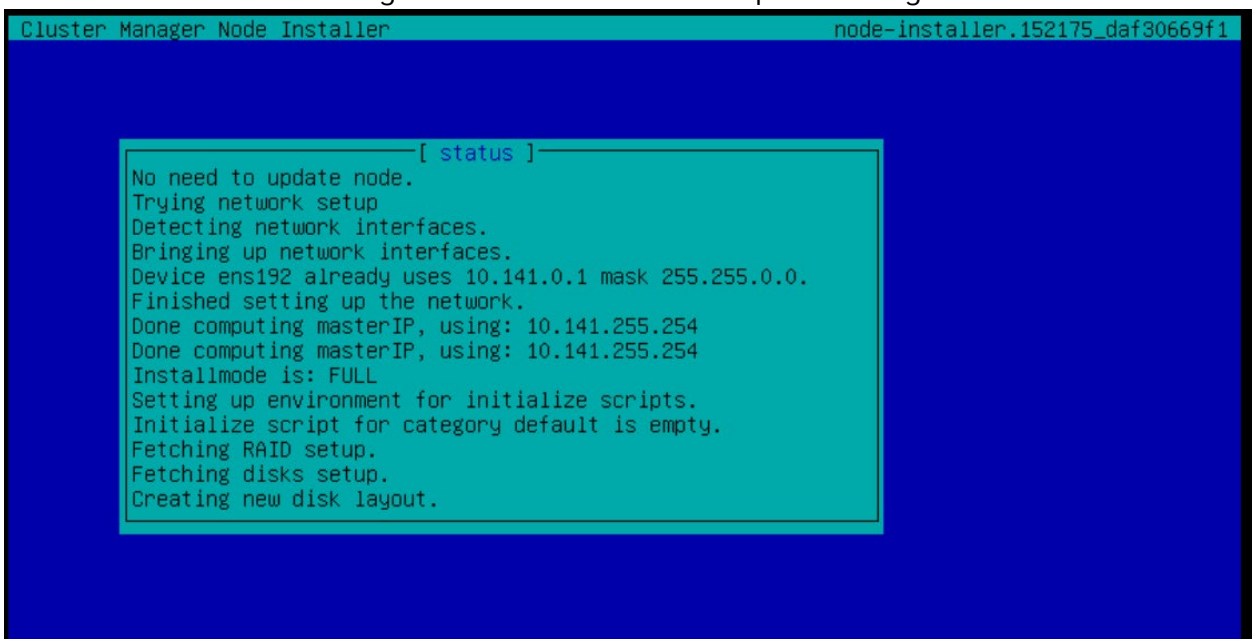


i. The node will contact the head node and then perform a quick restart.
Either press **Enter** to **Accept**, or simply allow the system to continue, you should get to the following screen



Since this is the first time the node is booting as part of the cluster, a **FULL** provisioning will occur.

j. Allow the count to time out or press the **Enter** key.
You should see the following screen that indicates the provisioning has started.



k. Return to CMSH (on the head node) to check the status of the compute node. There will be notifications that show the process of the node.

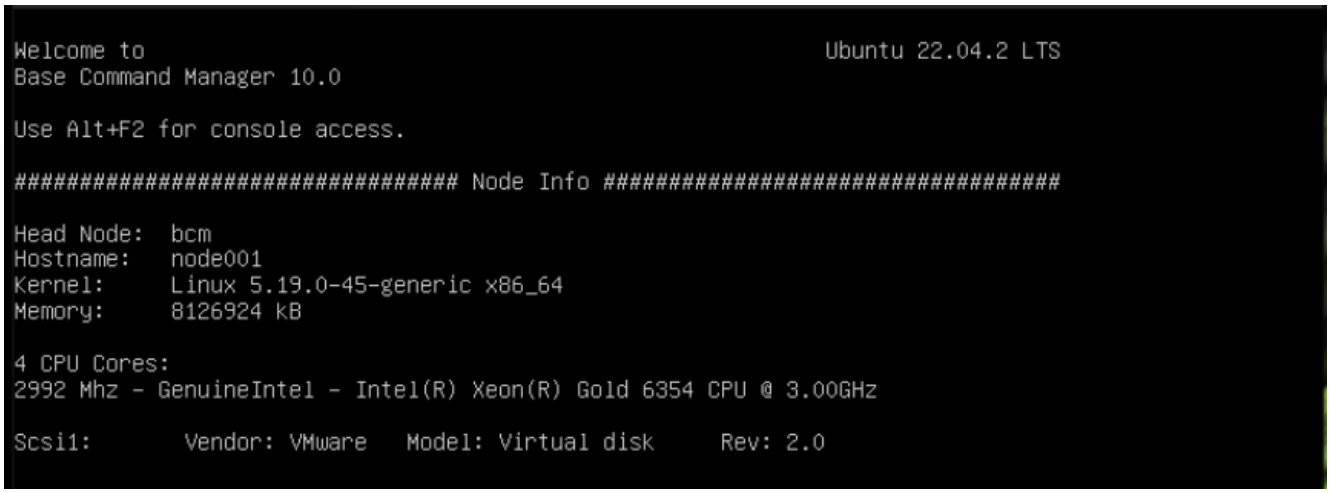
Use the device mode and the list command to see the status of the nodes.
It should show that the head node and **node001** are UP.

```
device; list
```

Example output:

```
root@bcm:~# cmsh
Thu May  2 08:35:52 2024 [notice] bcm: node001 [       BOOTING       ]
(ldlinux.e64 from bcm)
[bcm]%
Thu May  2 08:36:29 2024 [notice] bcm: node001 [     INSTALLING      ] (node
installer started)
[bcm]%
Thu May  2 08:36:50 2024 [notice] bcm: node001 [ INSTALLER_CALLINGINIT ]
(switching to local root)
[bcm]%
Thu May  2 08:36:56 2024 [notice] bcm: node001 [   UP   ]
[bcm]% device; list
Type                  Hostname (key)   MAC                Category
Ip              Network        Status
--------------------- ---------------- ------------------ ---------------- -
-------------- -------------- ----------------------
HeadNode              bcm              00:50:56:8A:7B:D1
10.141.255.254  Internalnet    [   UP   ]
PhysicalNode          node001          4E:56:44:41:01:01  default
10.141.0.1      Internalnet    [   UP   ]
PhysicalNode          node002          00:00:00:00:00:00  default
10.141.0.2      Internalnet    [  DOWN  ], unassigned
PhysicalNode          node003          00:00:00:00:00:00  default
10.141.0.3      Internalnet    [  DOWN  ], unassigned
PhysicalNode          node004          00:00:00:00:00:00  default
10.141.0.4      Internalnet    [  DOWN  ], unassigned
[bcm->device]%
```

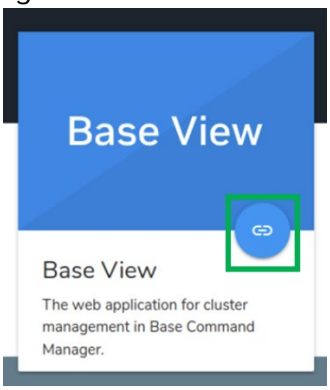
Once provisioned, the console for **node001** will look like the image below:



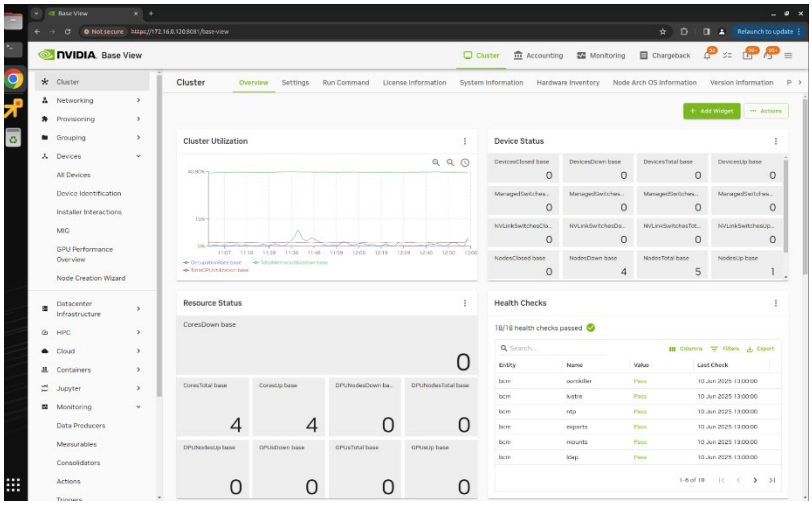
Task 2: Provision a Predefined Node Using Base View

a. For this exercise, the MAC address of a node will be entered into **Base View** for a particular node.
To access **Base View**, click **Head Node WebUI** icon on the **Admin VNC** desktop. This will open the Chrome browser.

b. Using the Chrome browser click **Base View** link to open **Base View**.



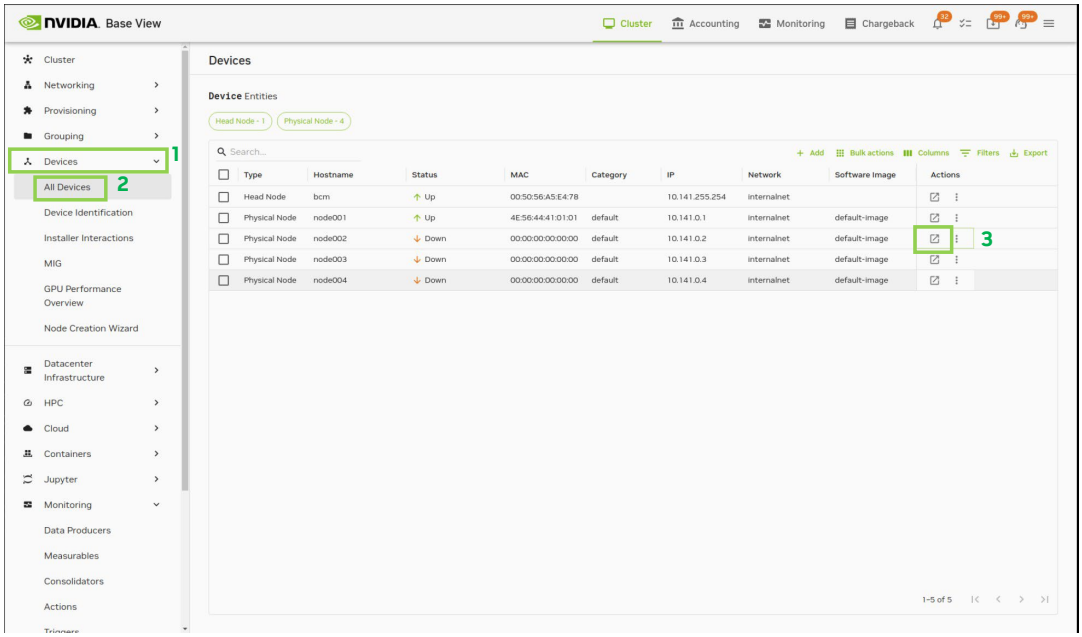
c. The credentials are: username: **root** password: **bcm123.**



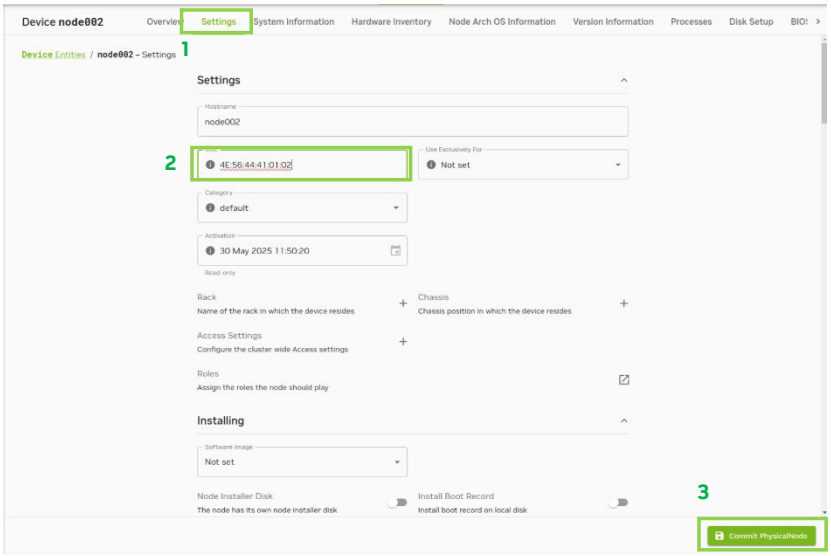
d. The MAC address for **node002** will be **4E:56:44:41:01:02**.

```
4E:56:44:41:01:02
```

e. Select **Devices -> All Devices** and then click on the **action** button at the end of the node002 row.

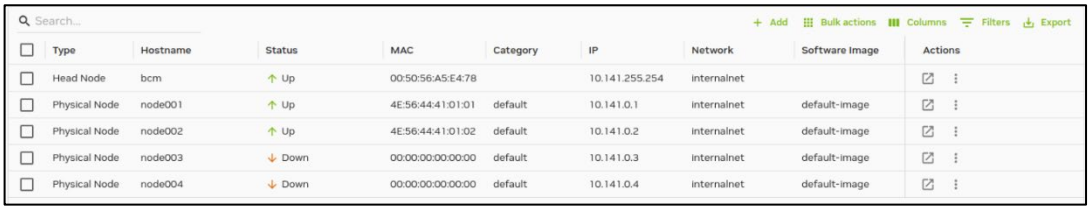


f. Click the **Settings** tab and enter the MAC address into the corresponding box.



Click the **Commit PhysicalNode** button on the lower right to finish.

g. Now that the MAC address is set, the node will be provisioned.
In **Base View**, the node should have a green up arrow in the status column when showing **Devices**.

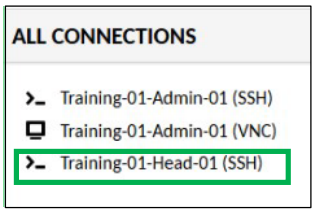


Type	Hostname	Status	MAC	Category	IP	Network	Software Image	Actions
Head Node	bcm	Up	00:50:56:A5:E4:78		10.141.255.254	internalnet		
Physical Node	node001	Up	4E:56:44:41:01:01	default	10.141.0.1	internalnet	default-image	
Physical Node	node002	Up	4E:56:44:41:01:02	default	10.141.0.2	internalnet	default-image	
Physical Node	node003	Down	00:00:00:00:00:00	default	10.141.0.3	internalnet	default-image	
Physical Node	node004	Down	00:00:00:00:00:00	default	10.141.0.4	internalnet	default-image	

Task 3: Provision a Predefined Node Using CMSH

a. For this exercise, the MAC address of two nodes will be entered into CMSH instead of **Base View**. The MAC address will be provided by a CSV file.
As with the **Base View** procedure, no user interaction with the compute nodes is required. This method can be used to quickly prepare a large number of compute nodes. Connect to the **Base Command Manager** head node (**Head-01**) via SSH. If still in CMSH, type `quit` to go back to the head node prompt.

```
quit
```



b. Run the **readmac** script:

```
./readmac.sh nodes.csv
Example output:
root@bcm:~# ./readmac.sh nodes.csv
Processing node: node003 with mac=4E:56:44:41:01:03
Processing node: node004 with mac=4E:56:44:41:01:04
Done!
```

c. Open CMSH to see that the nodes are being provisioned.

```
cmsh
```

From CMSH go to the mode:

```
device
```

Display information about the cluster devices:

```
list
```

Example output:

```
root@bcm:~# cmsh
[bcm]% device
[bcm->device]% list
Type                  Hostname (key)   MAC                Category
Ip              Network        Status
--------------------- ---------------- ------------------ ---------------- -
-------------- -------------- -----------------
HeadNode              bcm              4E:56:44:41:01:05
10.141.255.254  Internalnet    [   UP   ]
```

```
PhysicalNode           node001          4E:56:44:41:01:01  default
10.141.0.1      Internalnet    [   UP   ]
PhysicalNode           node002          4E:56:44:41:01:02  default
10.141.0.2      Internalnet    [   UP   ]
PhysicalNode           node003          4E:56:44:41:01:03  default
10.141.0.3      Internalnet    [   UP   ]
PhysicalNode           node004          4E:56:44:41:01:04  default
10.141.0.4      Internalnet    [   UP   ]
```

Appendix for Practice 1

readmac.sh

```
#! /bin/bash

# run this script as follows:
# readmac.sh nodes.csv

# read a line at a time, stripping out lines that begin with a comment
character
for line in `cat $1 |sed '/^#/ d'` ; do
   # read the node
   node=`echo $line | cut -d',' -f1`
   # read the mac
   mac=`echo $line | cut -d',' -f2`
   echo "Processing node: $node with mac=$mac"
   cmsh -c "device use $node; set mac $mac; commit"
   retcode=$?
   if [ $retcode -ne 0 ]; then
         echo "cmsh failed with exit code: $retcode"
         exit
   fi
done

echo "Done!"
```

nodes.csv

```
# Read this file with readmac.sh,
# node,mac
node003,4E:56:44:41:01:03
node004,4E:56:44:41:01:04
```

Practice 2: Software Images

Practice Objectives:

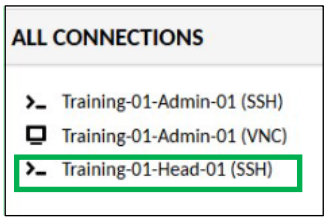
In this practice session you will become familiar with node images in Base Command Manager:

You will

- Clone the default image using Base View.
- Edit the default image and load kernel modules.
- Get familiar with the Base View event log.
- Create a Revision.

Task 1: Clone the Default Image

a. Connect to the **Base Command Manager** head node (**Head-01**) via SSH.



The credentials are: username: **root** password: **bcm123**

b. Enter **cmsh**.

```
cmsh
```

Enter the software image mode:

```
softwareimage
```

Print a list of software images:

```
list
```

Example output:

```
root@bcm:~# cmsh
[bcm]% softwareimage
[bcm->softwareimage]% list
Name (key)            Path                                    Kernel version
Nodes
--------------------- --------------------------------------- --------------
---- --------
default-image          /cm/images/default-image                5.15.0-113-
generic  4
dgx-os-6.3-a100-image  /cm/images/dgx-os-6.3-a100-image        5.15.0-1063-
nvidia  0
dgx-os-6.3-h100-image  /cm/images/dgx-os-6.3-h100-image        5.15.0-1063-
nvidia  0
[bcm->softwareimage]%
```

c. Create a new software image by cloning the default-image:

```
clone default-image default-image-backup
```

Commit the change, and the software image will be cloned:

```
commit
```

Example output:

```
[bcm->softwareimage]% clone default-image default-image-backup
[bcm->softwareimage*[default-image-backup*]]% commit
… [output truncated]
Mon Apr  1 15:07:00 2024 [notice] bcm: Initial ramdisk for image default-
image-backup was generated successfully
[bcm->softwareimage]% list
Name (key)            Path                                    Kernel version
Nodes
--------------------- --------------------------------------- --------------
---- -------
default-image          /cm/images/default-image                5.15.0-113-
generic  4
default-image-backup   /cm/images/default-image-backup         5.15.0-113-
generic  0
dgx-os-6.3-a100-image  /cm/images/dgx-os-6.3-a100-image        5.15.0-1063-
nvidia  0
dgx-os-6.3-h100-image  /cm/images/dgx-os-6.3-h100-image        5.15.0-1063-
nvidia  0
```

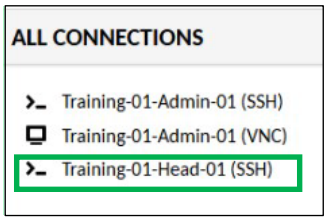
Wait for the message saying that the Initial **ramdisk** was generated successfully.
Use the **list** command to see the new software images.

```
list
```

Task 2: Edit the Default Image

a. Connect to the head node (**Head-01**) via SSH. If you are currently in CMSH, use the **`quit`** command to exit CMSH.

```
quit
```



b. Use the **`cm-chroot-sw-img /cm/images/default-image`** command to connect to the default image.
Notice the change in the command prompt from showing ~ (indicating the home directory) to showing / (indicating the root directory of the image).

```
cm-chroot-sw-img /cm/images/default-image
```

c. Create a file in the image and verify its existence by entering the following commands:

```
touch test.txt
```

```
ls
```

Example output:

```
root@default-image:/# touch test.txt
root@default-image:/# ls
bin   cm   etc   initrd.img  lib32  libx32  media  opt   root  sbin   snap
sys  tmp  var boot  dev  home  lib  lib64  local   mnt    proc  run   share
srv   test.txt  usr  vmlinuz
```

d. Close the chroot session:

```
exit
```

Notice the change in the directory indicated in the prompt (it changed from **@default-image:/ to @bcm:~**).

Example output:

```
root@default-image:/# exit
exit
… [output truncated]
root@bcm:~#
```

e. Now, we will edit the kernel modules in the default image.
Enter CMSH:

```
cmsh
```

f. Enter the software image mode:

```
softwareimage
```

g. Use the default image:

```
use default-image
```

h. Access the **kernelmodules** sub-mode:

```
kernelmodules
```

i. Get a list of the existing kernel modules in the **default-image**:

```
list
```

Note that the ***soundcore module*** is NOT on the list. We will use this module to demonstrate adding a module to the kernel.

Example output:

```
root@bcm:~# cmsh
[bcm]% softwareimage
[bcm->softwareimage]% use default-image
[bcm->softwareimage[default-image]]% kernelmodules
[bcm->softwareimage[default-image]->kernelmodules]% list
Module (key)             Parameters
------------------------ --------------------------------------------------
nfs
e1000
tg3
sata_sil
ext3
ext4
forcedeth
… [output truncated]
```

j. Add the **soundcore** module to the kernel:

```
add soundcore
```

k. Commit the changes:

```
commit
```

l. Display the updated list of modules in the **default-image**:

```
list
```

Verify that the ***soundcore module*** is now in the list.
Note: The **ramdisk** for the image will have to be generated after this change.

Example output:

```
[bcm->softwareimage[default-image]->kernelmodules]% add soundcore
[bcm->softwareimage*[default-image*]->kernelmodules*[soundcore*]]% commit
[bcm->softwareimage*[default-image*]->kernelmodules*[soundcore*]]% list
Module (key)             Parameters
------------------------ ------------------------------------------------
nfs
e1000
tg3
sata_sil
ext3
ext4
forcedeth
…
soundcore
[bcm->softwareimage[default-image]->kernelmodules]%
```

Task 3: Verify the changes

a. Connect to the head node (**Head-01**) via SSH and enter CMSH.

```
cmsh
```

b. Navigate to the **device** mode.

```
device
```

c. SSH into node001 from the head node.

```
rshell node001
```

d. List the files under the root folder, you should **NOT** see the ***test.txt*** file you created earlier.

```
ls /
```

Example output:

```
root@ bcm:~#cmsh
[bcm]% device
[bcm->device]% rshell node001
root@node001:~# ls /
bin   cm   etc   initrd.img  lib32  libx32  media  opt   root  sbin   snap
sys  usr  vmlinuz boot  dev  home  lib lib64  local   mnt    proc  run
share  srv   tmp  var
```

e. List the kernel modules and filter to display only ***soundcore***.
Similarly, you should **NOT** see the Kernel module you added earlier.

```
lsmod | grep soundcore
Example output:
root@node001:~# lsmod | grep soundcore
root@node001:~#
```

f. Exit the **rshell** and reboot **node001**:

```
exit
```

```
reboot node001
```

Example output:

```
root@node001:~# exit
[bcm->device]% reboot node001
```

g. You may notice this alert on the head node console.

```
Mon Jun 19 16:23:56 2023 [notice] bcm: node001 [  DOWN  ]
```

This indicates the node was restarted/shutdown.

h. Wait for the node to be reprovisioned, and you see the following messages in CMSH.

```
Mon Jun 19 16:25:39 2023 [notice] bcm: node001 [       BOOTING       ]
(ldlinux.e64 from bcm)
Mon Jun 19 16:26:15 2023 [notice] bcm: node001 [     INSTALLING      ] (node
installer started)
Mon Jun 19 16:26:37 2023 [notice] bcm: node001 [ INSTALLER_CALLINGINIT ]
(switching to local root)
Mon Jun 19 16:26:49 2023 [notice] bcm: node001 [   UP   ]
```

i. Connect to **node001**:

```
rshell node001
```

j. List the files under the root folder.
Now you should see the **test.txt** file you created earlier.

```
ls /
```

Example output:

```
root@node001:~# ls /
bin   cm   etc   initrd.img  lib32  libx32  media  opt   root  sbin   snap
sys tmp  var boot  dev  home  lib lib64  local   mnt    proc  run   share
srv  test.txt  usr  vmlinuz
```

k. List the kernel modules and filter to display only ***soundcore***.
You will see the kernel module you added earlier.

```
lsmod | grep soundcore
```

Example output:

```
root@node001:~# lsmod | grep soundcore
soundcore              16384  0
```

l. Leave the **rshell** session:

```
exit
```

m. Leave CMSH.

```
quit
```

Example output:

```
root@node001:~# exit
logout
[bcm->device]% quit
root@bcm:~#
```

Practice 3: Node Categories and Groups

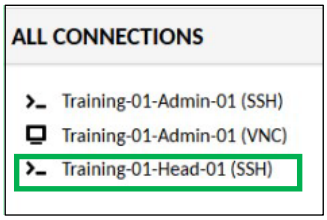
Practice objectives:

In this practice session you will become familiar with the node categories and groups used in Base Command Manager.

- You will create a new category.
- You will associate a node with a category.
- You will operate on multiple nodes simultaneously.

Task 1: Cloning a Node Category

a. Connect to the **Base Command Manager** head node (**Head-01**) via SSH.



The credentials are: username: **root** password: **bcm123.**

b. Enter CMSH.

```
cmsh
```

c. Enter the category mode:

```
category
```

d. Display the list of node categories:

```
list
```

e. List the nodes in the default category.
There should be 4 nodes in the category.

```
listnodes default
```

Example output:

```
root@bcm:~# cmsh
[bcm->category]% list
Name (key)               Software image           Nodes
------------------------ ------------------------ --------
default                  default-image            4
[bcm->category]% listnodes default
Type                  Hostname (key)   MAC                Category
Ip              Network        Status
--------------------- ---------------- ------------------ ---------------- -
-------------- -------------- ----------------
PhysicalNode          node001          4E:56:44:41:01:01  default
10.141.0.1      Internalnet    [   UP   ]
PhysicalNode          node002          4E:56:44:41:01:02  default
10.141.0.2      Internalnet    [   UP   ]
PhysicalNode          node003          4E:56:44:41:01:03  default
10.141.0.3      Internalnet    [   UP   ]
PhysicalNode          node004          4E:56:44:41:01:04  default
10.141.0.4      Internalnet    [   UP   ]
```

```
[bcm->category]%
```

f. Make a clone of the default category called **Lite** and commit the change.

```
clone default lite;commit
Example output:
[bcm->category]% clone default lite;commit
[bcm->category[lite]]%
```

g. Set the software image for the **Lite** category and commit the change.

```
set softwareimage default-image-backup;commit
```

h. Review the settings for the category:

```
show
```

Example output:

```
[bcm->category[lite]]% set softwareimage default-image-backup; commit
[bcm->category[lite]]% show
Parameter                        Value
-------------------------------- ------------------------------------------
------------
Name                             lite
Nodes                            0
… [output truncated]
Software image                   default-image-backup
… [output truncated]
```

i. Move **node003** to the new **Lite** category.
Enter the device mode:

```
device
```

j. Enter sub-mode for **node003**.

```
use node003
```

k. Change the category and commit the change:

```
set category Lite; commit
```

Example output:

```
[bcm->category[Lite]]% device
[bcm->device]% use node003
[bcm->device[node003]]% set category lite;commit
[bcm->device[node003]]%
```

l. List the devices and note that **node003** is now in the **Lite** category.

```
list
```

Example Output:

```
[bcm->device[node003]]% list
Type                    Hostname (key)   MAC                Category
Ip              Network         Status
-------------------- ---------------- ------------------ ---------------- -
-------------- -------------- ----------------
HeadNode                bcm              4E:56:44:41:01:05
10.141.255.254  Internalnet     [   UP    ]
PhysicalNode            node001          4E:56:44:41:01:01  default
10.141.0.1      Internalnet     [   UP    ]
PhysicalNode            node002          4E:56:44:41:01:02  default
10.141.0.2      Internalnet     [   UP    ]
PhysicalNode            node003          4E:56:44:41:01:03  Lite
10.141.0.3      Internalnet     [   UP    ]
PhysicalNode            node004          4E:56:44:41:01:04  default
10.141.0.4      Internalnet     [   UP    ]
[bcm->device[node003]]%
```

m. Provision the SW image to **node003**.
Without the **-w** flag, the command will do a dry run.
The **-w** flag ensures the node is provisioned.
Note the messages that give the provisioning status of the node.

```
imageupdate -n node003 -w
```

Example output:

```
[bcm->device[node003]]% imageupdate -n node003 -w
[bcm->device[node003]]%
Fri May 10 13:22:05 2024 [notice] bcm: Provisioning started: sending
bcm:/cm/images/default-image-backup to node003:/, mode UPDATE, dry run = no
[bcm->device[node003]]%
Fri May 10 13:22:13 2024 [notice] bcm: Provisioning completed: sent
bcm:/cm/images/default-image-backup to node003:/, mode UPDATE, dry run = no
imageupdate -n node003 -w [ COMPLETED ]
[bcm->device[node003]]%
```

n. Verify that **node003** has a different image than **node001** which is in the default category.
Use the **.. (or exit)** command to move up to the device mode.
Use the command **`get node00x softwareimage`** to see the information for node001 and node003.

```
..
```

```
get node001 softwareimage
```

```
get node003 softwareimage
```

Example output:

```
[bcm->device[node003]]% ..
[bcm->device]% get node001 softwareimage
default-image (category:default)
[bcm->device]% get node003 softwareimage
default-image-backup (category:Lite)
[bcm->device]%
```

o. Connect to **node001** from the head node and list the files in the root directory.
You should see the ***test.txt*** file that you created earlier.

```
rshell node001
```

```
ls /
```

Example output:

```
[bcm->device]% rshell node001
root@node001:~# ls /
bin   cm   etc   initrd.img  lib32  libx32  media  opt   root  sbin   snap
sys tmp  var boot  dev  home  lib lib64  local   mnt    proc  run   share
srv  test.txt  usr  vmlinuz
```

p. Exit **node001** and connect to **node003.**
List the files in the root directory.
Exit the **node003** when you are done.
You should **NOT** see the ***test.txt*** file you created earlier, since this file is not in the default-image-backup and therefore should not be present on **node003**.

```
exit
```

```
rshell node003
```

```
ls /
```

Example output:

```
root@node001:~# exit
logout
[bcm->device]% rshell node003
root@node003:~# ls /
bin   cm   etc   initrd.img  lib32  libx32  media  opt   root  sbin   snap
sys  usr  vmlinuz boot  dev  home  lib lib64  local   mnt    proc  run
share  srv   tmp  var
root@node003:~# exit
logout
[bcm->device]%
```

Task 2: Working with Multiple Nodes

a. Connect to the head node (**Head-01**) via SSH and enter CMSH.
   Navigate to the **category** mode and list the Node categories.

```
category;list
```

Example output:

```
[bcm]% category;list
Name (key)               Software image           Nodes
------------------------ ------------------------ --------
Lite                     default-image-backup     1
default                  default-image            3
dgx-a100                 dgx-os-6.3-a100-image    0
dgx-h100                 dgx-os-6.3-h100-image    0  Lite
default-image-backup     1
```

Each node category has an associated software image. The nodes in the category will inherit the software image from the node category.
Go to the device mode and check the software image for each node:
`get node00x softwareimage`
Example output:

```
[bcm->device]% get node001 softwareimage
default-image (category:default)
[bcm->device]% get node002 softwareimage
default-image (category:default)
[bcm->device]% get node003 softwareimage
default-image-backup (category:Lite)
[bcm->device]% get node004 softwareimage
default-image (category:default)
```

b. The "(**category:[sw-img]**)" information shows that the node inherits the software image from the node category.
Set the software image for all the nodes in the default node category to the **default-image-backup** category at the same time and commit the change.

```
foreach -c default (set softwareimage default-image-backup)
```

```
commit
```

Example output:

```
[bcm->device]% foreach -c default (set softwareimage default-image-backup)
[bcm->device*]% commit
Successfully committed 3 Devices
```

c. Confirm that the software image for these nodes is updated and does not come from the node category.
The output will show the result for each node the command is run on.

```
foreach -c default (get softwareimage)
Example output:
[bcm->device]% foreach -c default (get softwareimage)
default-image-backup
default-image-backup
default-image-backup
```

d. Clear the software image at the node level for all nodes.
This will cause the nodes to inherit the software image from the node category.
Use the following command to operate on all nodes in the range at the same time:

```
range -n  node001..node004
```

e. Clear the node level software image setting:

```
clear softwareimage
```

f. Commit and confirm the change:

```
commit; get softwareimage
```

Note that the nodes are now getting their software image from the node category.

Example output:

```
[bcm->device]% range -n node001..node004
```

```
[bcm->device{-n node001..node004}]% clear softwareimage
[bcm->device*{-n node001..node004}]% commit; get softwareimage
default-image (category:default)
default-image (category:default)
default-image-backup (category:Lite)
default-image (category:default)
```

g. Reboot the nodes to ensure they have the proper software image provisioned:

```
reboot
```

h. There will be updates on the boot and provisioning. Use the **quit** command to leave CMSH:

```
quit
```

Example output:

```
[bcm->device{-n node001..node004}]% reboot
Reboot in progress for: node001
Reboot in progress for: node002
Reboot in progress for: node003
Reboot in progress for: node004
[bcm->device{-n node001..node004}]% quit
root@bcm:~#
```

Practice 4: User and Group Management

Practice Objectives:

In this practice session you will become familiar with the setting up users and groups in Base Command Manager using CMSH and Base View:

You will:

- Add users using CMSH and Base View.
- Verify the users and set user parameters.
- Add and modify groups.
- Remove users and groups.

Task 1: Add a User

In this task we will use CMSH to add a user.

a. Enter the user mode:

```
user
```

b. Add a user named **slurmy**:

```
add slurmy
```

c. Commit the changes:

```
commit
```

d. Display all the users in the system:

```
list
```

Example output:

```
root@bcm:~# cmsh
[bcm]% user
[bcm->user]% add slurmy
[bcm->user*[slurmy*]]% commit
[bcm->user[slurmy]]% list
Name (key)       ID (key)         Primary group    Secondary groups
---------------- ---------------- ---------------- ----------------
cmsupport        1000             cmsupport
slurmy           1001             slurmy
[bcm->user[slurmy]]%
```

e. Display all parameters for the user that was just created:

```
show
```

f. The user password value is empty. A user must have a password to be able to log in. Set the password:

```
set password Welcome123!
```

g. Commit the change:

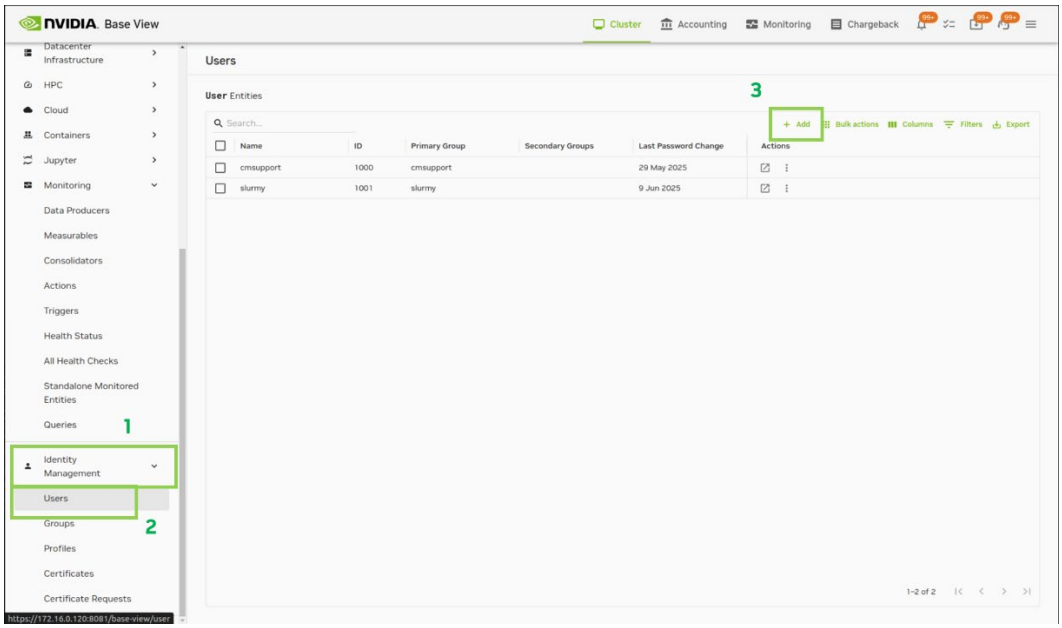
```
commit
```

Example output:

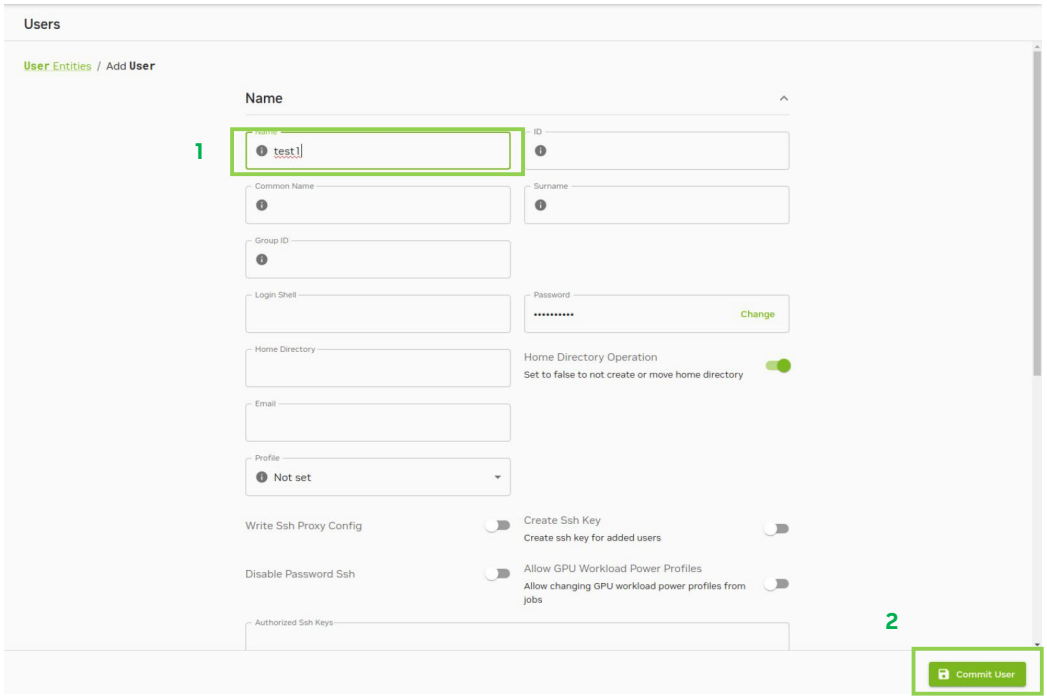
```
[bcm->user[slurmy]]% show
Parameter                      Value
------------------------------ ------------------------------------------
-
Accounts
```

```
Managees
Name                                    slurmy
Primary group                           1001
Revision
Secondary groups
ID                                      1001
… [ output truncated]
[bcm->user[slurmy]]%set password Welcome123!
[bcm->user*[slurmy*]]% commit
```

h. Use the Chrome browser on the **Admin** desktop to connect to **Base View**.
Add a user in **Base View** by selecting:
Identity Management->Users from the menu, and the **+ Add** button.



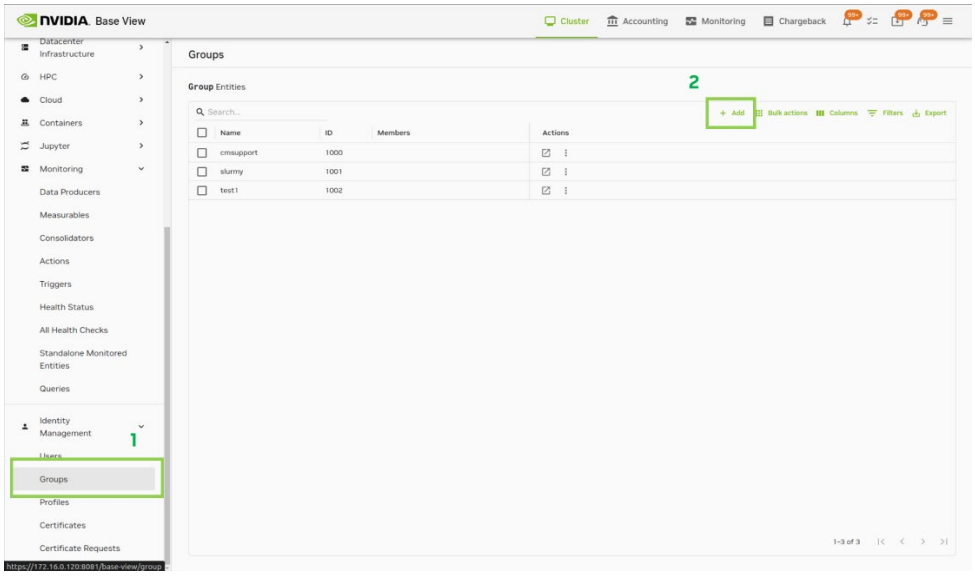
The **Name** field is the only required field. Once the name is input, click the **Commit User** button.
A user ID will be created and the other information (except for password) will fill in with default values.

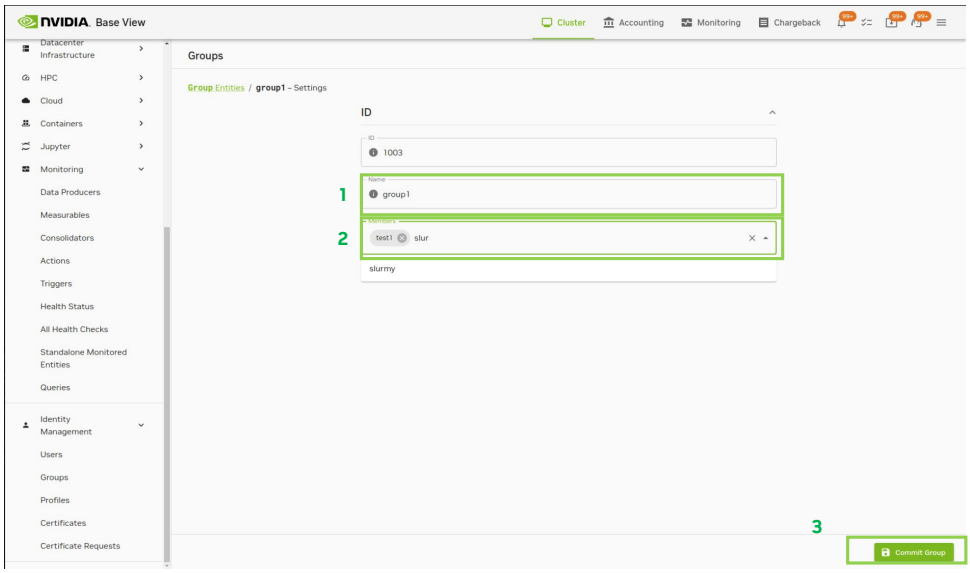


Task 2: Add a Group

a. Groups can be created in a similar way that users are created.
To add a group in **Base View**, click **Identity Management->Groups** from the menu, and the **+ Add** button. The group name is required.

- Create a Group named **group1**.
- Members can be added by typing the name in the box.
- Clicking the X icon deletes members from groups.
- Click the Commit Group button to save the group.





b. In CMSH, we use the group mode and add a new group using the commands below. Users can be added to the group using the **append** command and users can be removed from a group using the **removefrom** command.

c. Enter cmsh and enter the group mode.

```
cmsh
```

```
group
```

d. Add **group2** and commit the change:

```
add group2;commit
```

e. Append the created users to the new group and commit the change:

```
append members slurmy test1; commit
```

Example output:

```
root@bcm:~# cmsh
[bcm]% group
[bcm->group]% add group2; commit
[bcm->user[group2]]% append members slurmy test1; commit
[bcm->user[group2]]%
```

Task 3: Deleting users and groups

a. In CMSH, the **remove** command can be used to remove a user or group based on the mode (user mode or group mode respectively).
The groups and users can be removed in **Base View** by clicking the **Edit** button and then Delete.
Confirm by clicking the **Delete** button in the pop up.
Use either tool to delete the users and groups you created.
Removing user with CMSH:

```
cmsh
```

```
user
```

```
remove test1
```

```
commit
```

Example output:

```
root@bcm:~# cmsh
[bcm]% user
[bcm->user]% remove test1
[bcm->user*]% commit
Successfully removed 1 Users
Successfully committed 0 Users
```

Practice 5: Health Checks and Monitoring

Objectives:
In this practice session you will become familiar with using Base View for creating metrics and health checks and working with the monitoring tools and dashboards.

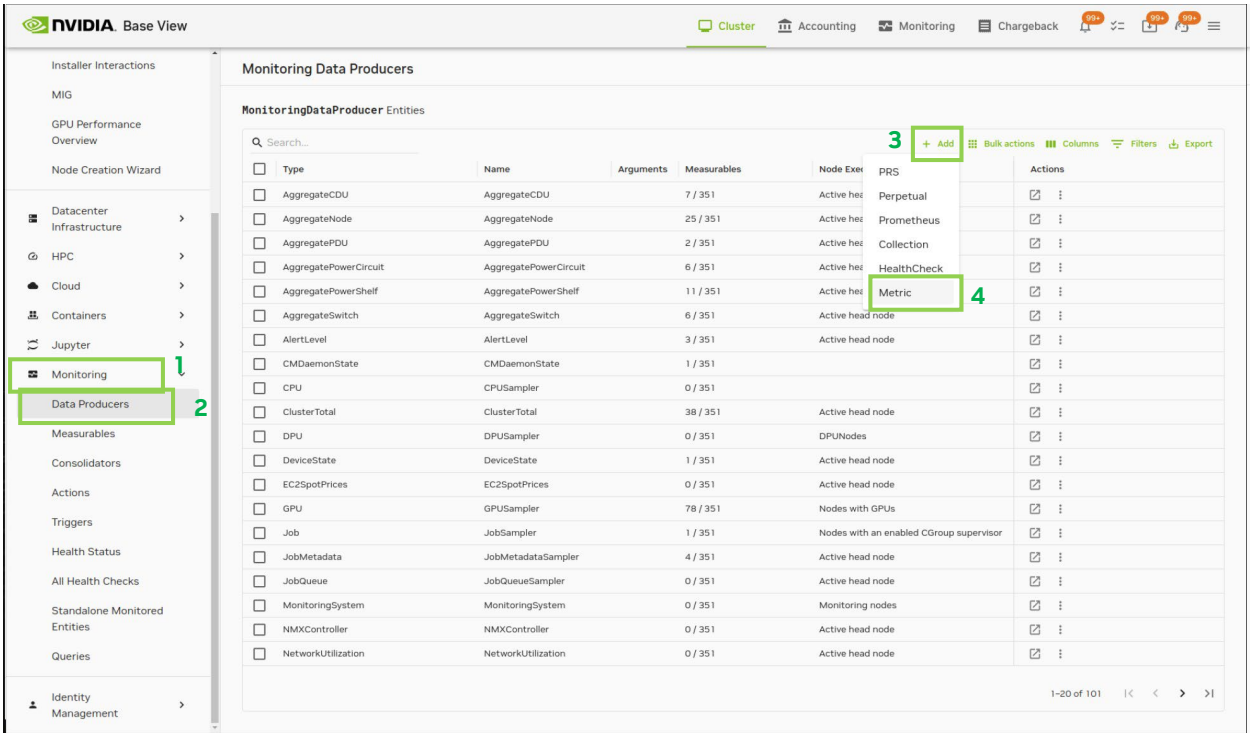
You will:

- Create a custom metric.
- Create a custom health check.
- Create a monitoring dashboard.

Task 1: Creating Custom Metrics, Health Checks, and Actions

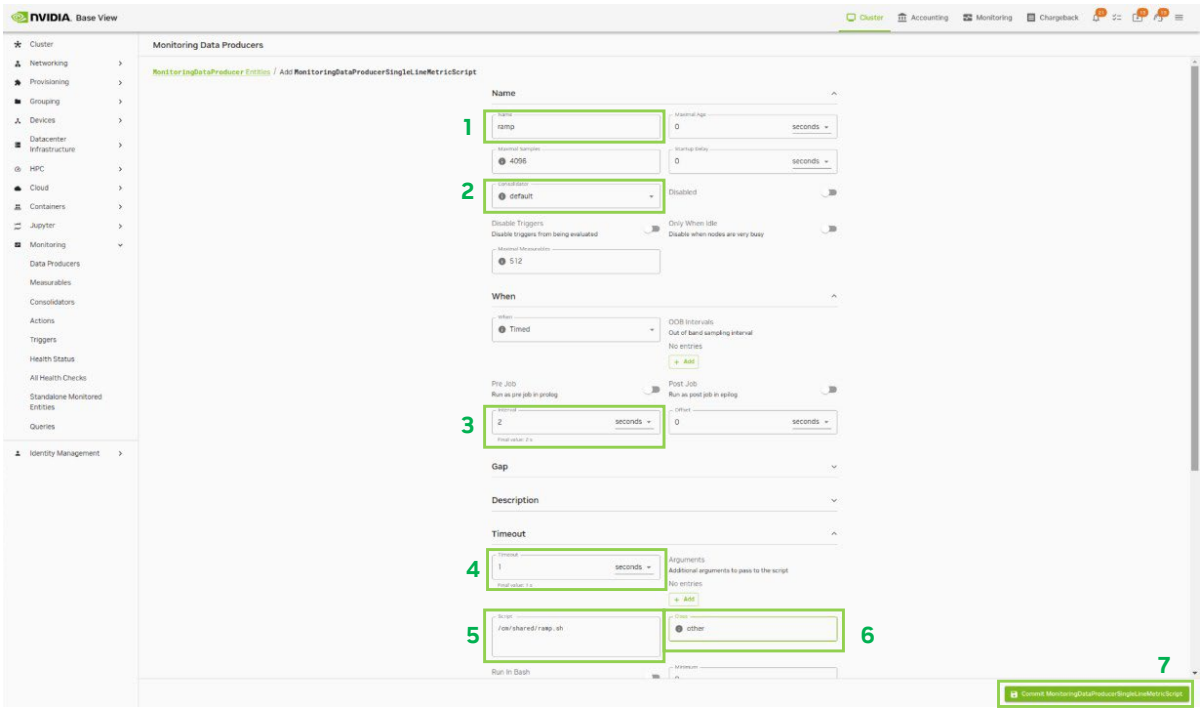
In this task we will use **Base View** to create a custom metric, health check, and action.

a. From the lefthand menu in **Base View**, click **Monitoring -> Data Producers**, and then the **+ Add** button.
Select **Metric** from the pop-up window.



b. Set the following parameters for the data producer:
   - In the **Name** section, **Name = ramp** and **Consolidator = default**.
   - In the **When** section, **Interval = 2 seconds**.
   - In the **Timeout** section, **Timeout=1 second** and **Script = /cm/shared/ramp.sh and Class = Other**.

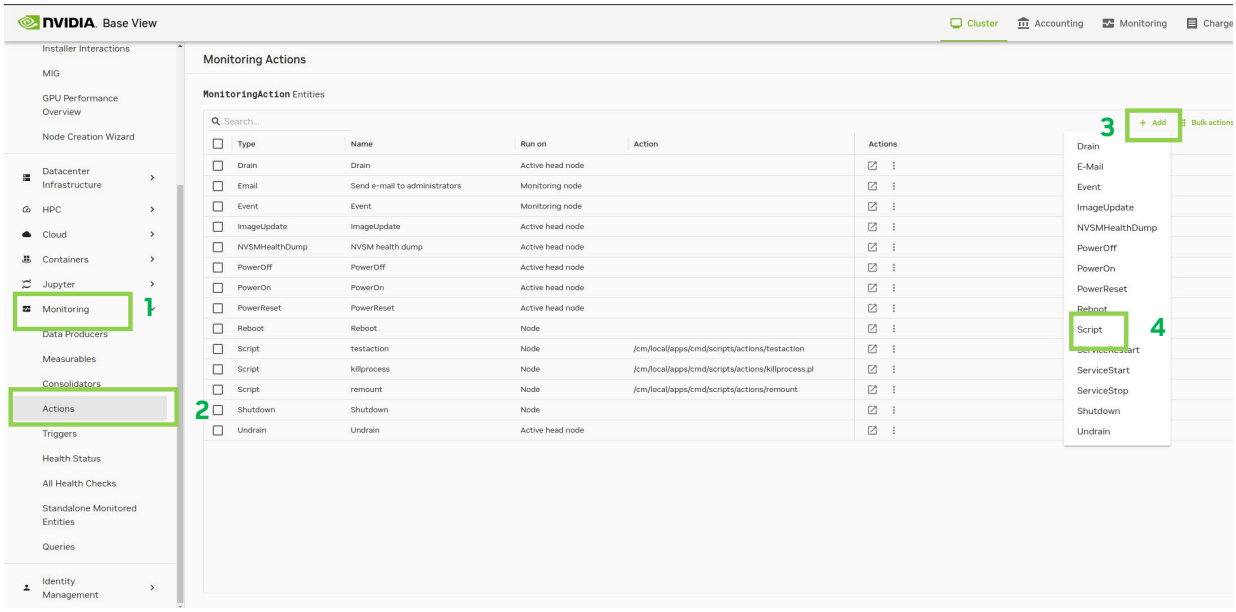
   Click the **Commit MonitoringDataProducerSingleLineMetricScript** button.



c. From the lefthand menu in **Base View**, click **Monitoring > Data Producers**, and then the **+ Add** button.
   Select **HealthCheck** from the pop-up window.
d. Set the following parameters for the data producer:
   - In the **Name** section, Name = **ramphealthcheck** and **Consolidator = default**.
   - In the **When** section, **Interval = 20**.
   - In the **Timeout** section **Timeout=5** and **Script = /cm/shared/rampcheck.sh and Class = Other**.

   Click the **Commit MonitoringDataProducerSingleLineHealthCheckScript** button.

e. Set up the action script.
   From the lefthand menu, click **Monitoring -> Actions** and then click the **+ Add** button.
   Click **Script** as the type.

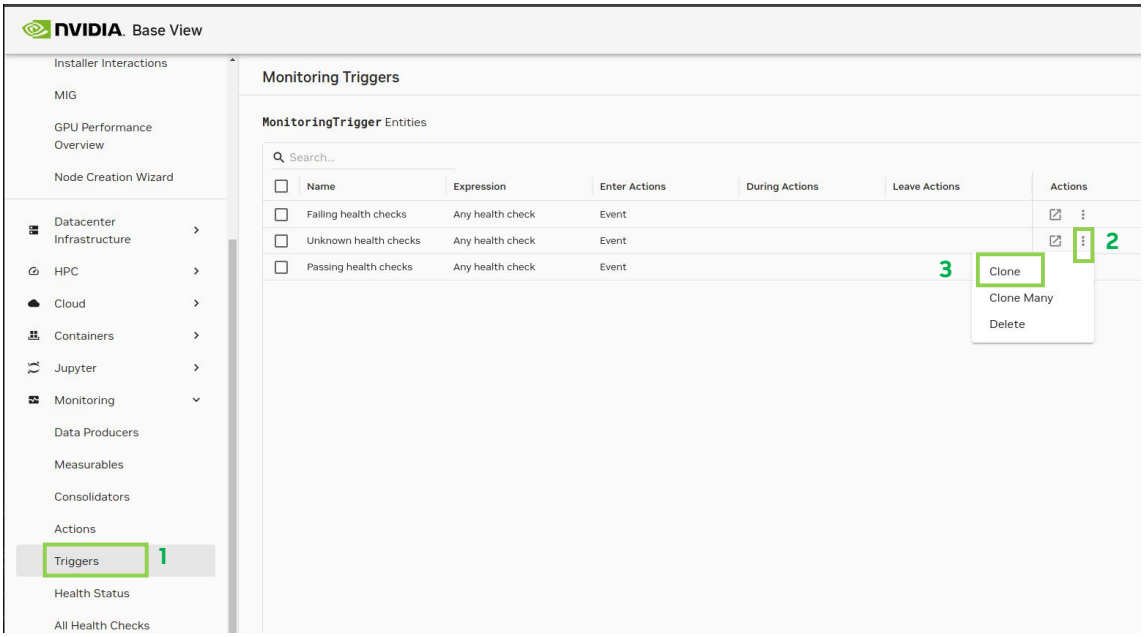


f. Set the following parameters for the action:

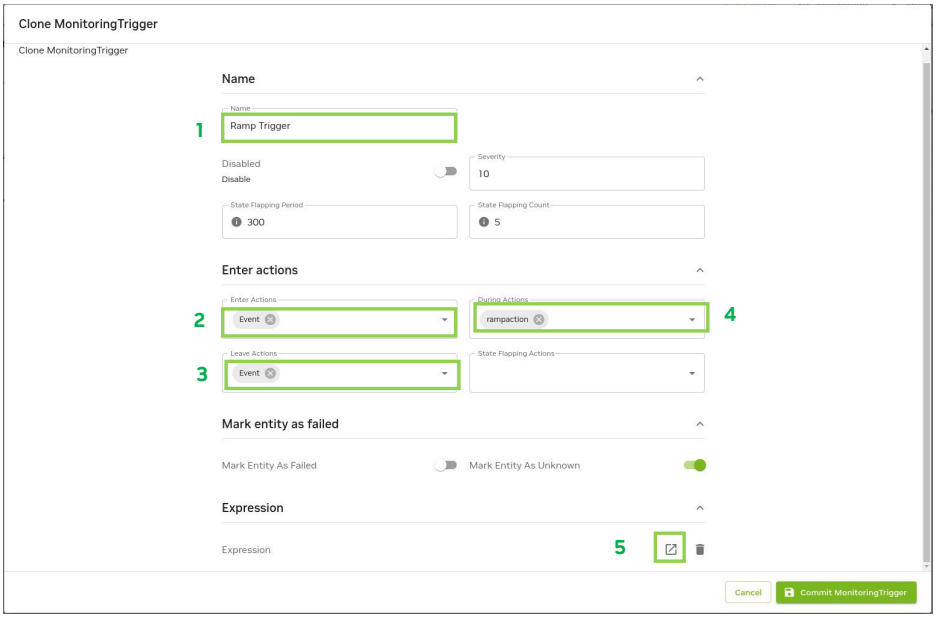
- In the **Name** section, **Name = rampaction**.
- In the **Script** section, **Script = /cm/shared/rampaction.sh**

Click the **Commit MonitoringScriptAction** button.

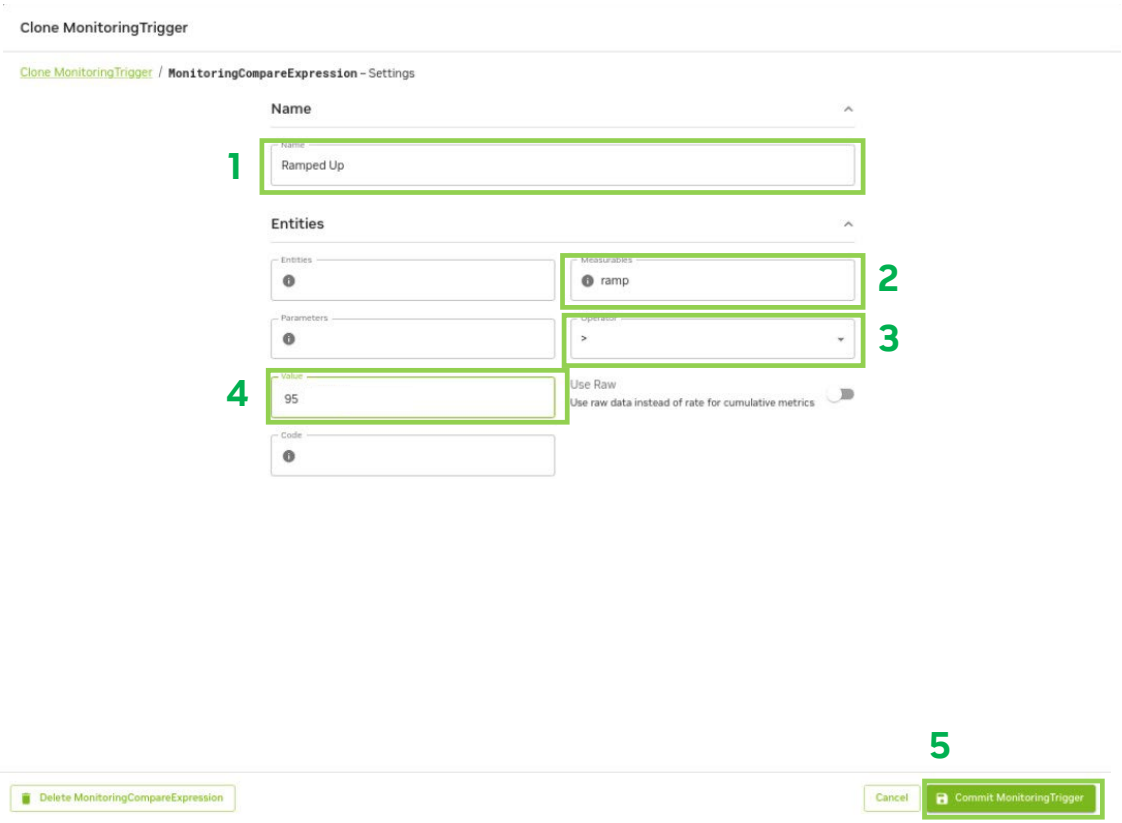
g. Select **Triggers** from the monitoring window and click to **Clone** the Unknown health checks trigger.



h. Edit the newly cloned trigger.
Set the name to **Ramp Trigger**.
Set the **Enter Actions** and **Leave Actions** to **Event**.
Set the **During Actions** to **rampaction**.
Click the **Edit** button for the **Expression**.

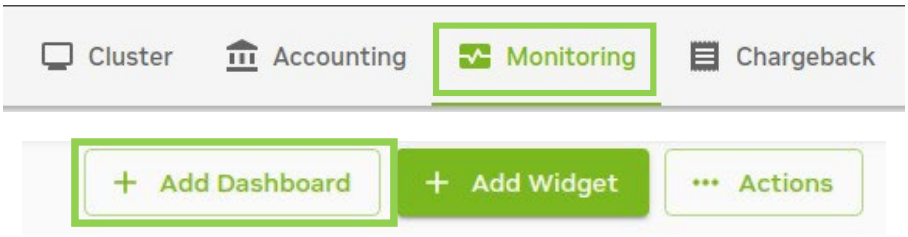


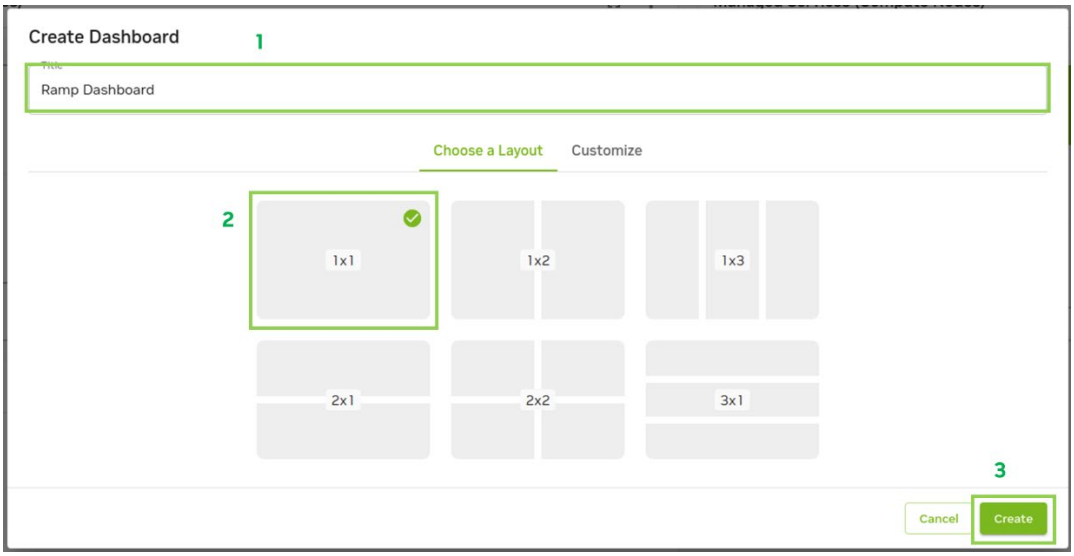
i. On the **Expression** screen, set the name to **Ramped Up**.
Set **Measurables** to **ramp**, select **>** for the **Operator**.
Set the **Value** to **95**.
Click the **Commit MonitoringTrigger** button.



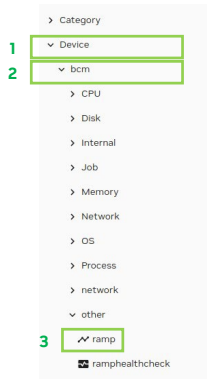
Task 2: Creating Monitoring Dashboard

a. Click the **Monitoring** icon in the top menu to view the dashboards.
Add a new **Dashboard** by clicking the **+ Add Dashboard** button shown on the monitoring window.
In the pop-up window, enter the title **Ramp Dashboard**, select **1x1** for the layout, and click the **Create** button.

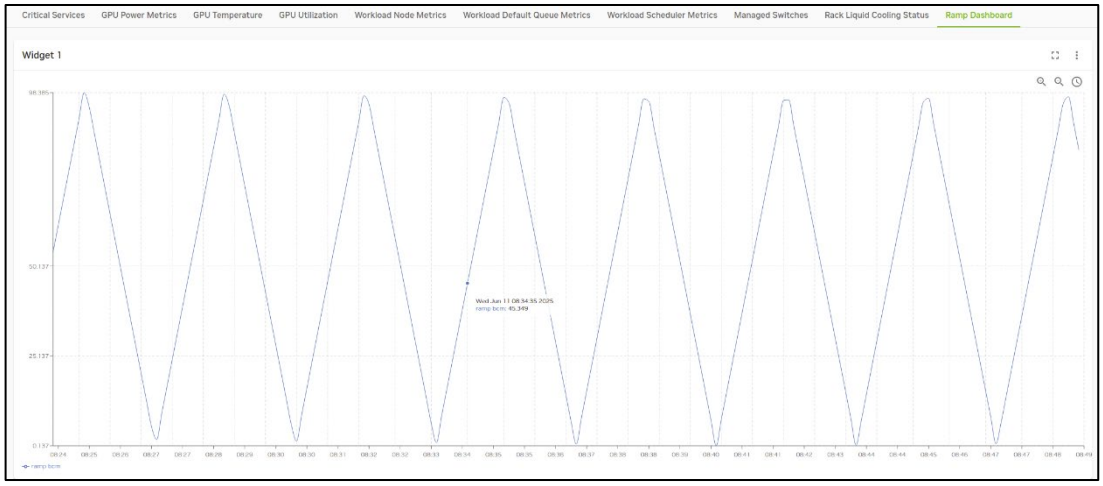




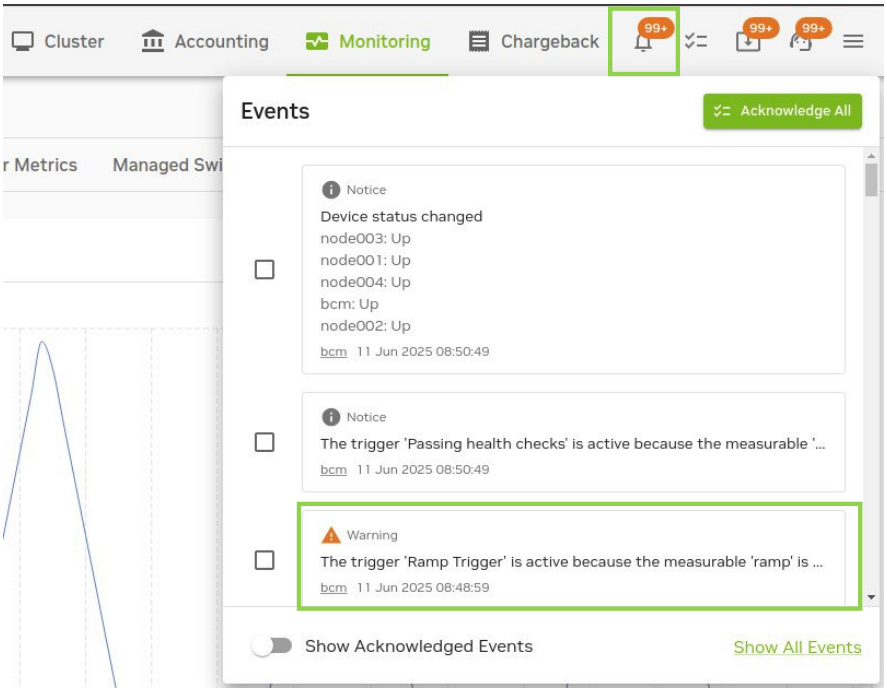
b. To add new data to the dashboard, drag the data source from the left menu to the widget on the new dashboard.
Click **Device -> bcm ->Other** and drag **ramp** to the **Dashboard**.



c. The dashboard settings can be used to adjust the data, remove existing dashboards, add new dashboards, etc.
Adjust the dashboards to see the **ramp**.



d. Click the **Notifications** icon in the top menu to view notifications.



Appendix for Practice 5

ramp.sh:

```
#! /bin/bash
# set the maximum value
max=100
# if our tracking file does not exist, set the value to zero
# and increment direction to positive
if [ ! -f "$HOME/ramp.dat" ] ; then
        value=0
        direction=1
else
# otherwise read the value and direction from the file
# they are stored as value,direction
        value=$(cat $HOME/ramp.dat | cut -d',' -f1)
        direction=$(cat $HOME/ramp.dat|cut -d',' -f2)
fi

# Change the direction when the absolute value equals the max
# NOTE: ${var#-} strips the minus from a string
# this equates to an abs() function
if [ "${value#-}" -ge "$max" ] ; then
        direction=$(( $direction * -1 ))
fi
#calculate the new value
value=$(( $value + $direction ))
# output the value
echo $value
# save value and direction for next time
echo "$value,$direction" > "$HOME/ramp.dat"
```

rampcheck.sh

```
#! /bin/bash
FAIL_LIMIT=90
# if the file does not exist, set the value to zero
if [ ! -f "$HOME/ramp.dat" ] ; then
    value=0
# otherwise read the value from the file

else
    value=$(cat $HOME/ramp.dat | cut –d',' -f1)
fi
# set the output based on the value
if [ "$value" -le "$FAIL_LIMIT" ] ; then
     echo 'PASS'
else
     echo 'FAIL'
fi
```

rampaction.sh

```
#! /bin/bash
# Save a string in a file.
echo "We had a failure on $(date)" >> "$HOME/rampfail.txt
```