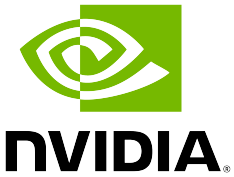

NVIDIA Base Command Manager 11

Upgrade Manual

Revision: 25e5965d7

Date: Thu Dec 11 2025

Table Of Contents

1 Introduction

This *Upgrade Manual* describes the procedure for upgrading a BCM (Base Command Manager) cluster that is being managed by BCM version 10, to a cluster managed by BCM version 11.

A clean installation of BCM is recommended instead of an upgrade because it is simpler.

If an upgrade is to be carried out, then reasonable familiarity with Base Command Manager is assumed. A good idea is to look up any unfamiliar concepts described in this manual in the other BCM manuals.

The *Upgrade Manual* should be read carefully before the upgrade

The BCM Support team can be contacted using `https://enterprise-support.nvidia.com/s/create-case` if there are BCM-related upgrade issues for which the documentation is insufficient.

1.1 Supported BCM Versions And Linux Distributions

1.1.1 Upgrades From The Following BCM Versions Are Supported:

- BCM 10

1.1.2 The Following Linux Distributions Are Supported:

- RedHat Enterprise Linux 8 (update 8.10 and later)
- RedHat Enterprise Linux 9 (update 9.5 and later)
- Rocky Linux 8 (update 8.10 and later)
- Rocky Linux 9 (update 9.5 and later)
- SuSE Linux Enterprise Server 15 (SLES15 SP6)
- Ubuntu 22.04
- Ubuntu 24.04

1.1.3 An Overview Of Software Upgrades In Relation To A Cluster

These three softwares can be considered when upgrading the main softwares for a cluster.

- The underlying (distribution) OS on the head node.
- The BCM software that manages the regular (compute) nodes and is on the regular nodes
- The underlying (distribution) OS on the regular nodes. That OS is what is provided by the software image.

Upgrading The Underlying OS Is Not Supported

The underlying OS is the software that runs the head node. This can be any of the distributions listed in section 2.1.1 of the *Installation Manual*. It is the distribution on which BCM runs. BCM in turn provides software images that are provisioned to the other nodes. The software images run on an underlying distribution OS that can be the same distribution as on the head node, but can also be of another distribution. The architecture can be `x86_64` or `aarch64`.

Upgrading the underlying OS, that is, upgrading from the existing major release of the OS upon which BCM runs, to the next major release of the OS (for example, RHEL8 to RHEL9, or Ubuntu 22.04 to Ubuntu 24.04) while keeping BCM as it is, is not supported. If the underlying OS is to be upgraded, then BCM must be reinstalled from scratch.

Upgrading BCM Is Supported

The discerning reader will have understood that upgrading the BCM cluster manager software from version 10 to 11 is the subject of this upgrade manual. The upgrade to BCM is carried out for the head node(s), and is carried out for the BCM used on the images that are used by the regular nodes. The upgrade process requires planning, preparation, and commitment, because once the upgrade is started—which is really when `cm-upgrade` is executed—rolling back is very hard. The parallel upgrade procedure (section 1.2) is recommended for cluster administrators who would like to check along each step of the way as they upgrade.

Upgrading The Software Images Is Supported

Creating software images for provisioning is described in section 9.6 of the *Administrator Manual*.

Upgrading the software images that are provisioned to other nodes is possible. This is however independent of a BCM upgrade. This means that, for example, an Ubuntu 22.04 image can be upgraded to an Ubuntu 24.04 image for a regular node while keeping the BCM 10 head node running Ubuntu 22.04. Sometimes that is all that the cluster administrator or users require, instead of upgrading BCM 10 to BCM 11.

An implication of the support for image OS upgrades independent of BCM upgrades is that a new DGX OS 7 image (which is based on Ubuntu 24.04) can be upgraded from a DGX OS 6 image (which requires Ubuntu 22.04 on the head node) in BCM 10 if the cluster has multiOS. However, only the B200 hardware is supported by DGX OS 7 in BCM 10, and it requires Ubuntu 24.04 on the head node. This means that DGX image upgrades from DGX OS 6 to DGX OS 7 for an Ubuntu 22.04 head node are not a supported image upgrade path in BCM 10.

Upgrading to BCM 11 and then setting up a new DGX 7 image within the new BCM 11 is recommended instead. DGX 7 images support more hardware, as listed in `https://docs.nvidia.com/dgx/dgx-os-7-user-guide/introduction.html#supported-nvidia-dgx-systems`.

BCM 11 clusters do not support DGX 6 images. This means that if a BCM 10 cluster with DGX OS 6 images is upgraded to BCM 11, then new DGX OS 7 images must be created when upgrading, by following the procedure in section 2.10.12 for a parallel upgrade, or in section 3.8.13 for an in-place upgrade (the idea behind a parallel upgrade and in-place upgrade is described in section 1.2).

When upgrading the software image, it is important to note that any configuration changes that were carried out on the old image need to have their equivalents or their analogous versions carried out for the new image.

1.2 Parallel Vs In-place Upgrades

There are two kinds of upgrades possible.

1. A **parallel upgrade**: This is when a clone is made of the active head node. This means that
   - For a cluster with one head node (a cluster with no HA failover configuration), the clone is a new clone of the head node, and that there are now two head nodes in existence—one active, and one clone

- For a cluster with two head nodes (a cluster with an HA failover, with one head as one active, and one head as the passive), the clone is a new clone of the active head node, and that there are now 3 head nodes in existence—one active, one passive, and one clone.

The clone is taken away and isolated from the network of the original cluster. Any passive is also taken to the new network and made the passive of the clone. The cloned system (that is: head node, or active head node and passive head node) is then upgraded.

Meanwhile, the original head node keeps managing the cluster, so that there is not such a severe pressure on getting the upgraded cluster up before a deadline. This means that if there are unexpected issues that take time to resolve during upgrade, then the original cluster and users remain unaffected.

After the required upgrade steps are completed for the isolated head node (or head node pair) that is undergoing the upgrade, then the unupgraded head node that is managing the cluster is replaced by the upgraded head node (or head node pair).

2. An **in-place upgrade**: This is when an upgrade is carried out using the existing head node or head nodes in a production cluster. The regular nodes need to be powered off when the upgrade is carried out, which means a significant downtime and can mean a severe pressure on the cluster administrator to get the upgraded cluster up before a deadline.

Chapter 2 covers the steps and considerations for the upgrade procedure that are specific to the parallel upgrade.

Chapter 3 covers the steps and considerations for the upgrade procedure that are specific to the in-place upgrade.

2 Parallel Upgrades

2.1 Introduction

In contrast with the in-place upgrade (Chapter 3), for a parallel upgrade the production cluster remains operational while the upgrade takes place. This is possible with the `cm-clone-install` and `cm-upgrade` functionality that allows the production (primary) head node to be cloned to a new node. The new node is upgraded (in parallel) outside of the cluster, and when the upgrade is completed, the old production primary head node is replaced with the (newly) upgraded head node. This last step to replace the head node will require a downtime.

Alternatively, the compute nodes can be gradually moved from the old production to the new parallel-upgrade setup, while the two setups co-exist.

2.2 Existing Cluster Health Check

It is a good idea to check that the existing cluster is in a healthy state before carrying out the upgrade.

The head nodes health checks in particular should be looked at. Unexpected health check failures should be examined, and if required, resolved. In some cases it is expected that health checks will be in a failed state. For example, if all the compute nodes are DOWN, then the schedulers health check fails.

The reason behind checking the health of the cluster before upgrading is that the health checks can detect failing services, overfull filesystems, unmounted filesystems, networking issues, time synchronization issues, and other issues. If these are unexpected then they can prevent the upgrade from completing successfully.

2.3 Preparations For The Parallel Upgrade Setup

The following steps need to be carried out for parallel upgrades. The steps marked with HA (High Availability) need to be carried out only on clusters with (head node) HA:

- Have the BCM product key and the MAC address of the spare node (which is to be used for cloning the primary active head node) ready, since they will need to be provided while cloning the primary head node to the spare node.
- (HA) Have also the MAC address of the secondary head node ready.
- Have the new IPs for the cloned primary head node on the external network(s) ready.
- (HA) The primary head node must be made the active head, if it is not already.
- (HA) Login to the secondary passive head node and stop CMDaemon (systemctl stop cmd). Note: make sure to stop CMDaemon on the passive secondary head node, and not on the primary active head node.

- (HA) On the same secondary passive head node, edit /etc/fstab and comment out the entry for `/cm/shared`, so that on the next boot the secondary passive head node does not mount `/cm/shared` from the file server.
- (HA) Shut down the secondary passive head node. The node should remain shut down until later when plugged in to the parallel setup.
- Plug the spare node to the internal network of the production cluster and PXE boot it to the BCM rescue environment (select RESCUE in the PXE boot menu, in the same way as performing a clone for an HA setup)
- Log in to the rescue environment and run cm-clone-install, providing the product key and the MAC address of this spare node. In the case of HA, the MAC address of the secondary head node must also be provided. Thus, for the non-HA case:

```
# /cm/cm-clone-install --clone --upgrade --productkey PRODUCT-KEY \
 --primarymac PRIMARY-MAC
```

and for the HA case:

```
# /cm/cm-clone-install --clone --upgrade --productkey PRODUCT-KEY \
 --primarymac PRIMARY-MAC --secondarymac SECONDARY-MAC
```

and follow the instructions. At the end of the cloning process, confirm shutting down the spare node.

In the preceding the values used for the primary and secondary MAC addresses are specified as follows:
`PRIMARY_MAC`: the MAC address of the management interface of the primary head node (i.e the one that is on the management network)
`SECONDARY_MAC`: the MAC address of the management interface of the secondary head node (i.e the one that is on the management network)

- Unplug the spare node from the production setup/networks and plug it in the parallel setup.
- In the case of `/cm/shared` on a file server (i.e. usually on an HA setup), login to the file server and make a copy of the `/cm/shared` to a new directory. Alternatively, the copy can be made to a new file server. Have the directory name and the (possibly new) file server host name ready.
- Boot the spare (a.k.a. cloned primary head) node as it is plugged in the parallel (isolated) setup. The node will boot with all of its network interfaces remaining DOWN.
- Login on console (ignoring possible messages about not-found modules in `/cm/shared`), and run /root/cm/update-cloned-db.py, providing the new location of `/cm/shared`:

```
# /root/cm/update-cloned-db.py --new-cm-shared NEW-CM-SHARED-DIRECTORY-NAME \
 --nas-server NAS-SERVER
```

which will setup in CMDaemon to mount the copy of `/cm/shared` made in the preceding session from NAS-SERVER:NEW-CM-SHARED-DIRECTORY-NAME

or, if `/cm/shared` is not on a NAS server (i.e. no copy of `/cm/shared` was made in the preceding session), then just run:

```
# /root/cm/update-cloned-db.py
```

The script finds values for MAC addresses, interfaces, and so on that came as part of the original primary head node, and converts those values to ones suitable for the clone.

- After /root/cm/update-cloned-db.py is completed, on the cloned primary head node still logged in on the console, by using "cmsh -r localhost", change the IPs of the primary head node (master) on the external network(s).
- In the case of HA: change also the shared IP (alias) of the secondary head node on the external network(s) to match the new shared IP(s) of the primary head node.
- If necessary, update also other settings such as power management settings or if desired fsmounts, to reflect that in this parallel setup the primary head node is a different physical machine, which is a clone of the original primary head node.
- Reboot, to allow for the settings to take effect. After the reboot, the (cloned) primary head node in the parallel setup will have the network interfaces UP.
- Ensure that this primary head node in the parallel setup has correctly mounted the copied `/cm/shared` from the file server, if a copy of `/cm/shared` was made in the preceding.

  If `/cm/shared` is not correctly mounted, then verify the fsmount settings for the primary head node, the secondary head node, as well as for the compute nodes (typically the compute nodes fsmount settings are set in the compute nodes categories) are correct and the head node is able to connect to the file server.
- (HA) Plug in the secondary head node to the parallel setup and boot it. After the secondary head node has booted, ensure it has mounted correctly the `/cm/shared` copy from the file server.
- (HA) On both the (cloned) primary head node and the secondary head node, ensure the active head node is the (cloned) primary head node, the failover is in a good state and the mysql replication is working by running (on both head nodes):

  ```
  # cmha status
  ```

  If any of the returned responses for `failoverping`, `mysql`, `ping`, or `status` do not report "OK", then ensure the network configuration and cabling allow the two head nodes in the parallel setup to communicate, and that they are isolated from the production setup. If this does not resolve the issue, then please contact support (`https://enterprise-support.nvidia.com/s/create-case`) for further assistance.
- Disable/remove the Base Command Manager integrations, such as with Kubernetes or WLM for some versions of BCM, as described in the important notes and other prerequisites in the preceding.

The notes in sections 2.4–2.8 should now be considered, before continuing on with the upgrade as described in sections 2.9– 2.10.

2.4 Important: Considerations For The Parallel Upgrade Setup

2.4.1 Configuration Changes And Services On The Production Cluster Desynchronized

The parallel upgrade procedure involves creating a clone of the primary head node and isolating the clone from the production cluster.

This means that after creating the clone of the primary head node, changes that are usually saved on the head node or in `/cm/shared` due to activities such as

- any further configuration changes on the production cluster,
- further collected monitoring data,
- other changes in settings or data values, such as are caused by the running of jobs, or by new WLM jobs' state data,

are not mirrored or synced to the cloned parallel head node, and are not mirrored or synced to the copy of the `/cm/shared` directory tree.

Similarly, services running on the original primary head node will also be started by the operating system on the cloned head node, but they will not be able to communicate with the production cluster nodes.

For example, if the cluster is running with Kubernetes integration, with only one control plane running on the head node, then the control plane on the production head node will continue to communicate with the worker nodes and maintain the Kubernetes cluster state. In the parallel setup, the Kubernetes control plane on the cloned head node cannot communicate with any node from the production cluster and will not be aware of the Kubernetes cluster state.

Therefore some special services running on the parallel-setup head node may not be able to take over from the respective services running on the production head node when the old production head node is to be shut down and the upgraded parallel-setup head node is to become the production head node. In these special cases, the administrator needs to take the necessary steps to migrate the services, or the administrator needs to perform an in-place upgrade.

2.4.2 Local Data On The Head Node Desynchronized

When the users' home directories (`/home`) or other data are not on a file server, then the administrator will most commonly need to take the necessary steps to sync the latest changes in the users' home, resp. other directories from the production setup to the parallel setup when the upgrade of the parallel setup is complete.

2.4.3 Dangling, Temporary, And Relocated Mounted Network File Systems

If the users' home directories are on a file server, then a manual sync of the data as pointed in the preceding is unnecessary, however the administrator must be aware that except for `/cm/shared` as noted below, the head node(s) in the parallel setup will retain their fsmount settings also during the upgrade. This means that also during the upgrade the head nodes from the parallel setup will attempt to mount the network file systems. If this behavior is not desired, the administrator can configure the parallel-upgrade network and cabling in such a way so that the file server(s) are also isolated from the parallel setup. Alternatively, the administrator can modify the fsmount settings after the primary head node is cloned when it is booted for the first time as described below, so that the head nodes in the parallel setup do not mount these network file systems.

On the other hand, if mounting the network file systems on the head nodes in the parallel setup is required, for example so that users can log in and test the upgraded setup while the production setup is still operational in parallel, then the network configuration/cabling should allow the head node to access the file servers. When the file servers are on an external network, usually no additional steps will be required since the head nodes in the parallel setup will have full access to the external networks.

If the file servers are on the internal network(s) however, because by default the cloned head node retains the same internal network IPs, and because the parallel setup's internal networks should be isolated from the production setup, by default the parallel-setup's head nodes cannot access the file servers or any other device on the production-cluster's internal network(s). Additional measures such as setting up intermediate servers or re-configuring the network settings will have to be taken to ensure the parallel-setup's nodes can access the data on the file servers when needed, or to make sure they have access to a copy of the data on different (possibly temporary) file servers. Such configuration changes are beyond the scope of this document, for further information please contact support (`https://enterprise-support.nvidia.com/s/create-case`) for further assistance.

A special case is `/cm/shared` mounted from a file server. The clone and upgrade scripts will detect that, and the administrator will be required to copy `/cm/shared` to a new location and provide the (new) path and file server host name. The new location can be a new directory on the same file server if the file server is on an external network. If the file server is on the internal network, the simplest solution is to set up a new or temporary file server to host the required for the parallel upgrade copy of `/cm/shared`.

2.4.4 Cluster Extension Upgrade Not Supported

Parallel upgrade of clusters extended to the cloud (Cluster Extension cloudbursting) is not supported. For these clusters only an in-place upgrade is possible, which involves the termination of all cloud nodes when the cluster extension is removed as part of the upgrade procedure.

2.4.5 Edge Setup Upgrade Not Supported

Parallel upgrade of clusters with an edge setup is not supported.

2.4.6 Prerequisites

- The cluster must have a previously installed and active subscription license.
  - Sites with hardware life licenses should contact support (`https://enterprise-support.nvidia.com/s/create-case`) for further assistance.
  - Before starting with the upgrade, the subscription license should first be unlocked via the self-service customer portal at `https://customer.brightcomputing.com/Customer-Portal?p=unlock`
- A spare node is required for cloning the original (a.k.a production) primary head node. The spare node must have comparable to the current head node hardware, such as number of network cards, number of disks and size, memory, etc. After the completion of the upgrade, the spare node will become the new primary head node for the upgraded cluster.
- In the case of head node HA, the secondary head node will need to be taken out of the production cluster so that it can also be upgraded. This means that for the time needed to perform the parallel upgrade, until all compute nodes are migrated to the upgraded setup, the old production cluster will run with only one head node.
- If `/cm/shared` will be unmounted during the upgrade then please make sure that the contents of the local `/cm/shared` are in sync with the remote copy.
- In the case of `/cm/shared` on a file server (such as in the case of head node HA), the file server must have enough space for creating a copy of the `/cm/shared` directory tree. Alternatively, `/cm/shared` can be copied to a new file server.
- While performing the upgrade, the cloned primary and—in the case of HA—the secondary head node(s) need to have isolated internal network(s) which do not clash with the internal

network(s) of the production cluster. The IP(s) of the upgraded head node(s) on the internal network(s) will remain the same. After the upgrade is completed, the old (not upgraded) primary head must be shut down and the upgraded head node(s) can be re-connected back in the cluster. Alternatively, compute nodes can be disconnected from the old (not upgraded) setup and then connected in the upgraded setup, which can also be performed gradually.

- There is no requirement to isolate the external network(s) for the head node(s) while performing the upgrade, which means the cloned head node can still have access to the Internet. However, the cloned primary head node will require a new set of IPs on the external network(s), which are different from the current IP(s) of the primary head node on the external network(s). It is also recommended to use static IPs, and not DHCP with dynamic DNS updates on the external network, because both the original production head node and its clone in the parallel setup will have the same hostname as defined in CMDaemon. The secondary head node does not require a new set of IPs because it is simply moved (and not cloned) from the production to the parallel setup.
- When upgrading the parallel setup, the head nodes require access to the new BCM package repositories. Alternatively, the BCM ISOs can be used as a source for the packages. Access to the base distro package repositories, using for example, the via internet or local repositories, is also required.

  Updates using the old BCM package repositories should be carried out only to bring the cluster up to date before running `cm-upgrade`.
- In the case of HA, the cloned primary and the secondary head nodes will require to be able to connect to the file server for their copied `/cm/shared` directory tree. This means that if the file server is on an internal network, then—because the internal networks must be isolated—a new file server must be setup and made available on the isolated internal network.

2.5 Important: Upgrading Particular Applications

2.5.1 Overview Of The Applications That `cm-upgrade` Does And Does Not Manage

Automatically Upgraded By `cm-upgrade`

The following software, if managed by BCM 10, is managed and upgraded automatically by `cm-upgrade` to BCM 11:

- Docker
- The `cm-docker-registry` package (Harbor upgrades (section 2.5.6) are not supported)
- Jupyter (the JupyterHub database requires some additional care (section 2.5.5).
- Base View: The regular version of Base View in BCM 10 is upgraded to a new version in BCM 11. Any monitoring dashboards from the regular BCM 10 Base View version are no longer used in the BCM 11 Base View version (section 2.5.2).
- NVIDIA drivers: The defaults will be installed in non-DGX images

Automatically Kept As Is By `cm-upgrade`

The following applications are allowed to remain during the upgrade, and are not automaticaly upgraded:

- WLMs (section 2.5.3)
  - Slurm

  - PBS Professional/OpenPBS
  - LSF

  The installation of workload managers is described in section 7.3 of the *Administrator Manual*, while the upgrade of the Slurm and PBS Professional workload managers is described in section 7.5 of the *Administrator Manual*.

- Kubernetes: Kubernetes installation is described in section 4.21 of the *Containerization Manual*.
- Run:Ai: Run:Ai installation options for BCM are described in section 6.8 of the *Containerization Manual*.
- NVIDIA Container Toolkit: An NVIDIA Container Toolkit installation can be carried out as part of Docker setup or a package installation (section 2.7 of the *Containerization Manual*).
- other Kubernetes operators: The installation of other Kubernetes operators is described in Chapter 6 of the *Containerization Manual*.

Manually Removed And Reinstalled During Upgrade (Not Managed By `cm-upgrade`)

- Harbor: As explained earlier, Harbor upgrades (section 2.5.6) are not supported. The Harbor package needs to be removed from BCM version 10, and redeployed (section 3.2 of the *Containerization Manual*) in BCM version 11.
- Cluster Extension: Needs to be removed and redeployed (Chapter 3 of the *Cloudbursting Manual*)

Removed From Support During Upgrade (Not Managed By `cm-upgrade`)

The following software is no longer managed by BCM in BCM version 11.

- (Altair) Grid Engine (AGE): BCM support for integration ended. Possible alternatives to the AGE WLM, that BCM supports, are Slurm, PBS Professional/OpenPBS, or LSF.
- Ceph: BCM support for integration ended.
- BeeGFS: BCM support for integration ended.

2.5.2 Upgrading Base View

BCM has Base View (section 2.4 of the *Administrator Manual*) as the browser GUI to carry out cluster management tasks. It is available as the package `base-view`.

Base View in BCM 10 had an additional, experimental and undocumented, Base View NG, available as the package `base-view-ng`. Base View NG uses a newer GUI toolkit from that of Base View in BCM 10.

Base View in BCM 11 is actually just a development of Base View NG from BCM 10, and so it uses that newer toolkit too.

The new Base View in BCM 11 means that an upgrade from BCM 10 to BCM 11 loses all the dashboards that users and administrators may have built for Base View in BCM 10.

It is therefore a good idea to make notes on the useful dashboards of BCM 10 so that analogous ones can be constructed for BCM 11. A useful transition may be to construct the analogues in Base View NG while still on BCM 10, before upgrading to BCM 11.

2.5.3 Upgrading Workload Managers (WLM)

Unsupported unversioned or unsupported versioned WLM packages are not removed automatically. They must be removed manually. If they are to be updated then it is recommended to install new, supported, versioned WLM packages.

The supported PBSPro versions are compatible across BCM 10 and BCM 11.

Upgrading WLMs In General From BCM 10

WLMs are not required to be disabled using `cm-wlm-setup` before upgrading the BCM version from 10 to 11.

Upgrading Slurm

Slurm versions older than Slurm 24.11 are not supported in BCM 11. When upgrading from BCM 10, Slurm must be upgraded to at least Slurm 24.11 in the BCM 10 repositories before moving to BCM 11.

For more information on upgrading Slurm, please refer to the KB article `https://kb.brightcomputing.com/knowledge-base/upgrading-slurm/`.

Packages: Software images now include the same packages as the head nodes. This is because starting with BCM 11, Slurm packages keep many things in `/cm/local`, instead of `/cm/shared`.

Slurm client packages, for example `slurm25.05-client`, are no longer available. So they are removed as part of the upgrade procedure.

There are now separate `slurmctld` and `slurmd` packages that provide the binary and systemd unit files for each service.

Path adjustments: There are some path adjustments for Slurm for the upgrade from BCM 10 to BCM 11, as summarized in the following table.

Slurm item	Version	Path
Prefix directory	BCM10	/cm/shared/apps/slurm
	BCM11	/cm/local/apps/slurm
Most configuration files	BCM10	/cm/shared/apps/slurm/var/etc/
	BCM11	/cm/shared/apps/slurm/etc
Munge configuration files	BCM10	/cm/shared/apps/slurm/var/etc/munge
	BCM11	/cm/shared/apps/slurm/var/etc/munge (unchanged)
Binaries	BCM10	/cm/shared/apps/slurm/current/bin
	BCM11	/cm/local/apps/slurm/current/bin
Template files such as: slurmdbd.conf.template, cgroup.conf.template	BCM10	/cm/shared/apps/slurm/var/etc
	BCM11	/cm/local/apps/slurm/current/templates
statesave files	BCM10	/cm/shared/apps/slurm/var/cm/statesave
	BCM11	/cm/shared/apps/slurm/statesave
Power scripts	BCM10	/cm/shared/apps/slurm/var/cm/power
	BCM11	/cm/local/apps/slurm/current/scripts/power
Enroot prolog	BCM10	/cm/shared/apps/slurm/var/cm
	BCM11	/cm/shared/apps/slurm/prologs/prolog-enroot.sh
Enroot epilog	BCM10	/cm/shared/apps/slurm/var/cm
	BCM11	/cm/shared/apps/slurm/epilogs/epilog-enroot.sh

Slurm DBD Performance Improvement

Installations from BCM 10.24.05 onward have MySQL configuration settings changes that improve the performance of the slurmdbd service. The changes are not automatically applied during updates of MySQL when updating from versions installed earlier than BCM 10.24.05.

To apply the improvement to a cluster that does not have this update, the MySQL configuration must be manually adjusted on the head nodes after `cm-upgrade` has completed. The /etc/my.cnf file on the head node, and also on the secondary head node in an HA-enabled cluster, should have the following settings updated:

```
innodb_buffer_pool_size=4096M
innodb_log_file_size=64M
innodb_lock_wait_timeout=900
```

The default `my.cnf` file is typically symlinked by the system utility `update-alternatives` in Ubuntu and SLES, (`alternatives` in RHEL derivatives), so how it is configured should also be considered.

Slurm Upgrades For DGX OS 6 To DGX OS 7

For DGX, BCM 10 supports only DGX OS 6 (an Ubuntu 22.04 variant), while BCM 11 supports only DGX OS 7 (an Ubuntu 24.04 variant). As part of the upgrade from BCM 10 to BCM 11, the DGX OS must be replaced with DGX OS 7. When replacing the DGX OS nodes from 6 to 7, a new software image (and `/cm/shared`) is created for the Ubuntu 2404 nodes.

If Slurm and Pyxis have been deployed in a BCM 10 DGX OS 6 image, then some manual configuration changes must be done to make these work right in the BCM 11 DGX OS 7 image. These can be carried out as follows:

Changes to the MUNGE key in `/cm/shared`: The key needs to have the correct permissions and ownership. If it does not have these, then a symptom of it is the message:

```
munged: Error: Keyfile is insecure: "/cm/shared/apps/slurm/var/munge/keys/
munge.key" should be owned  by UID 1 instead of UID 0
```

as seen in the systemd status output:

```
root@dgx-04:~# systemctl status munge.service
× munge.service - MUNGE authentication service
     Loaded: loaded (/usr/lib/systemd/system/munge.service; enabled;
         preset: enabled)
     Active: failed (Result: exit-code) since Fri 2025-10-31
         03:54:29 PDT; 26s ago
       Docs: man:munged(8)
    Process: 255424 ExecStart=/usr/sbin/munged --key-file=/cm/shared/
       apps/slurm/var/munge/keys/munge.key (code=exited, status=1/
       FAILURE)Oct 31 03:54:29 dgx-04 systemd[1]: Starting
    munge.service - MUNGE authentication service...
Oct 31 03:54:29 dgx-04 munged[255424]: munged: Error: Keyfile is insecure:
  "/cm/shared/apps/slurm/var/munge/keys/munge.key" should be owned by UID 1
  instead of UID 0
Oct 31 03:54:29 dgx-04 systemd[1]: munge.service: Control process exited,
  code=exited, status=1/FAILURE Oct 31 03:54:29 dgx-04 systemd[1]:
  munge.service: Failed with result 'exit-code'.
Oct 31 03:54:29 dgx-04 systemd[1]: Failed to start munge.service -
  MUNGE authentication service.
```

The key can be made consistent with that of the previous deployment. This can be done by changing the attributes of the files in the new DGX software image, for example `dgx-os-7.2-h100-image`, after mounting it in the appropriate shared directory on the new cluster:

Example

```
cm-chroot-sw-image -s /cm/shared-ubuntu2404-x86_64 \
/cm/images/dgx-os-7.2-h100-image
...
chown daemon:root /cm/shared/apps/slurm/var/munge/keys/munge.key
chmod 400 /cm/shared/apps/slurm/var/munge/keys/munge.key
```

Changes to the Pyxis and Enroot files: The files within the DGX OS 7 image can be configured as follows:

Copy the Pyxis plugin file to the new software image, for example: `dgx-os-7.2-h100-image`:

Example

```
cp /cm/local/apps/slurm/current/lib64/slurm/spank_pyxis.so \
 /cm/images/dgx-os-7.2-h100-image/cm/local/apps/slurm/current/lib64/slurm/
```

Copy the plugstack files in `/cm/shared` over from the Ubuntu 2204 (DGX OS 6) version to the Ubuntu 2404 (DGX OS 7) version:

Example

```
cp /cm/shared-ubuntu2204-x86_64/apps/slurm/etc/slurm/plugstack.conf \
   /cm/shared-ubuntu2404-x86_64/apps/slurm/etc/slurm/
cp /cm/shared-ubuntu2204-x86_64/apps/slurm/etc/slurm/plugstack.conf.d/pyxis.conf \
 /cm/shared-ubuntu2404-x86_64/apps/slurm/etc/slurm/plugstack.conf.d/
```

Copy the Enroot configurations in `/cm/shared` from the Ubuntu 2204 version to the Ubuntu 2404 version:

Example

```
cp /cm/shared-ubuntu2204-x86_64/apps/slurm/etc/enroot.conf \
 /cm/shared-ubuntu2404-x86_64/apps/slurm/etc/
cp /cm/shared-ubuntu2204-x86_64/apps/slurm/etc/enroot-sysctl.conf \
 /cm/shared-ubuntu2404-x86_64/apps/slurm/etc/
```

Symlinks for the epilog, prolog, and `enroot.conf` files are created for the new software image:

Example

```
cm-chroot-sw-img -s /cm/shared-ubuntu2404-x86_64 /cm/images/dgx-os-7.2-h100-image/
ln -s /cm/shared/apps/slurm/prologs/prolog-enroot.sh \
 /cm/local/apps/slurm/var/prologs/50-prolog-enroot.sh
ln -s /cm/shared/apps/slurm/epilogs/epilog-enroot.sh \
 /cm/local/apps/slurm/var/epilogs/50-epilog-enroot.sh
ln -s /cm/shared/apps/slurm/etc/enroot.conf /etc/enroot/enroot.conf
<ctrl-d>
```

The compute nodes can pick up the new configuration when they get rebooted.

PBS Professional Upgrades

PBS Professional versions older than PBS Pro 2022 are not supported in BCM 10. Below is an overview of the steps that can be taken to upgrade PBS Pro:

- Stop the `pbsserver` service on the head node

  ```
  cmsh% device
  cmsh% foreach -l pbsproserver ( services; stop pbsserver )
  ```

- Remove the old packages from the headnodes

```
$ yum/zypper/apt remove pbspro<old version>*
```

- Remove the old packages from the software images

```
$ cm-chroot-sw-img /cm/images/<software image>
...
[root@<software image> /]# yum/zypper/apt remove pbspro<old version>*
```

- Install the new packages in the headnode

```
$ yum/zypper/apt install pbspro<new version> pbspro<new version>-client
```

- Install the new packages in the software images

```
$ cm-chroot-sw-img /cm/images/<software image>
...
...
[root@<software image> /]# yum/zypper/apt install pbspro<new version>-client
```

- Change the version of the WLM in the `wlm` object

```
cmsh% wlm use pbspro
cmsh% set version <new version>
cmsh% commit
```

- Start `pbsserver`

```
cmsh% device
cmsh% foreach -l pbsproserver ( services; start pbsserver )
```

2.5.4 Upgrading The Kubernetes Package

Kubernetes deployments do not need removal before upgrading to BCM 11. The same version of Kubernetes runs on the BCM cluster after the upgrade.

An upgrade to Kubernetes (section 4.21 of the *Containerization Manual*) can be carried out after the BCM upgrade.

Kubernetes Package Upgrades For DGX Software Images

When upgrading DGX OS nodes from 6 to 7.2, a new software image (and `/cm/shared`) needs to be created (section 2.10.12), for the nodes which will be running Ubuntu 24.04.

Kubernetes on the new DGX image needs to be manually copied over from the old image. This can be done as follows:

- The `kubernetes.list` file should be copied over from the old image, <*old DGX image*>, to the new image, <*new DGX image*>, along with its associated keyring file:

  Example

```
[root@basecm11 ~]# cp /cm/images/<old DGX image>/etc/apt/sources.list.d/\
kubernetes.list /cm/images/<new DGX image>/etc/apt/sources.list.d/
[root@basecm11 ~]# cp /cm/images/<old DGX image>/etc/apt/keyrings/\
kubernetes-apt-keyring.gpg /cm/images/ <new DGX image>/etc/apt/keyrings/
```

- The packages for the existing version need to be placed in the new image (here it is called `new-dgx-image`):

```
[root@basecm11 ~]# cm-chroot-sw-img /cm/images/new-dgx-image
...
[root@new-dgx-image /]# apt-get install cm-kube-diagnose kubeadm kubectl \
kubelet kubernetes-cni cm-containerd nvidia-container-toolkit nginx \
libnginx-mod-stream
...
```

- The packages for the existing version need to be held. The following commands take care of that:

```
[root@new-dgx-image /]# apt-mark hold kubeadm
[root@new-dgx-image /]# apt-mark hold kubectl
[root@new-dgx-image /]# apt-mark hold kubelet
[root@new-dgx-image /]#   <ctrl-d>
```

The `<ctrl-d>` is to exit out of the `cm-chroot-sw-img` directory. The `cm-chroot-sw-img` command is described on page 534 of the *Administrator Manual*.

2.5.5 Upgrading The JupyterHub Database

The JupyterHub database schema must be manually upgraded in each JupyterHub node after the main package upgrade has completed. The following commands can be run to upgrade the JupyterHub database:

```
module load jupyter
jupyterhub upgrade-db--config=/cm/local/apps/jupyter/conf/jupyterhub_config.py \
  --db sqlite:////cm/local/apps/jupyter/run/jupyterhub.sqlite
```

2.5.6 Upgrading The Docker Registry And Harbor Packages

Harbor cannot be upgraded when BCM 10 is upgraded to BCM 11, its package can only be (re-)installed and (re-)deployed separately from the BCM upgrade.

If the Harbor packages were installed in BCM 10 to a node other than the head node, then a re-deployment to that node is required after the upgrade to BCM 11 for Harbor.

When upgrading a cluster from BCM 10 to BCM 11 the upgrade of Docker registry is supported.

2.6 Important: Package Upgrade Dependency Issues

The upgrade process will not only upgrade CMDaemon and its dependencies, but it will also upgrade other packages. This means that old packages will not be available from the new BCM repositories.

In some cases, this will require recompiling the user applications to use the upgraded versions of the compilers and the libraries. Also, the configurations of the old packages will not be copied automatically to the new packages, which means that the administrator will have to adjust the configuration from the old packages to suit the new packages manually.

Extra base distribution packages may be installed by `yum/zypper/apt-get` in order to resolve dependencies that might arise as a result of the upgrade. Hence the base distribution repositories must be reachable. This means that the clusters that run the Enterprise Linux distributions (RHEL and SLES) must be subscribed to the appropriate software channels.

Packages in `/cm/shared` are upgraded, but the administrator should be aware of the following:

- If `/cm/shared` is installed in the local partition, then the packages are upgraded. This may not be desirable for users that wish to retain the old behavior.
- If `/cm/shared` is mounted from a separate partition, then unmounting it will prevent upgrades to the mounted partition, but will allow new packages to be installed in `/cm/shared` within the local partition. This may be desirable for the administrator, who can later copy over updates from the local `/cm/shared` to the remote `/cm/shared` manually according to site-specific requirements. Since unmounting of mounted `/cm/shared` is carried out by default, a local `/cm/shared` will have files from any packages installed there upgraded. According to the yum database, the system is then upgraded even though the files are misplaced in the local partition. However, the newer packages can only be expected to work properly if their associated files are copied over from the local partition to the remote partition.

2.7 Known Issues

2.7.1 Kubernetes PVC Access Issue With The Shared Directory For Multidistro (MultiOS)

With a multidistro Kubernetes cluster, there can be a known issue with accessing `/cm/shared` across the various nodes.

Under What Conditions Is There An Issue?

The issue can occur when all of the following conditions apply:

1. The Kubernetes cluster (and worker nodes in particular) are spread across multiple DGX OS versions, due to one of the following situations:
   - (A) the nodes are running as multidistro nodes within the same Kubernetes cluster (running permanently as multidistro as is the intention, and not just transiently as part of an upgrade).
   - (B) the nodes are being upgraded, for example from Ubuntu 22 to Ubuntu 24
2. The *Local Path Provisioner* is provisioned and deployed in the Kubernetes cluster. This is a component running in Kubernetes that enables pods in every Kubernetes node to consume some kind of local storage. The `local-path-provisioner` parameter provides the local path `StorageClass`, which sets up a sub-directory under /cm/shared.
3. In this Kubernetes cluster, there is at least one pod using the local-path StorageClass using a PVC (PersistentVolumeClaim)

Why Is There An Issue?

The problem is that the `/cm/shared` directory is not unique across all nodes in a BCM cluster. The directory differs based on the DGX OS version. For example, on the head node, for the `/cm/shared` mount, you have:

- for Ubuntu 22 nodes:

  `/cm/shared-ubuntu2204-x86_64`

- for Ubuntu 24 nodes:

  `/cm/shared-ubuntu2404-x86_64`

During normal running, each DGX node only sees its specific `/cm/shared`.

Now, if a pod consuming a PVC from its local-path StorageClass is rescheduled to a node from the other DGX OS, then it cannot find the volume anymore, and it creates a new one.

Resolving The Issue For Persistent Mixed Distributions

A solution in the case of persistent mixed distributions on the nodes is to move all the pods to a new common shared directory for Kubernetes, independent of the directories offered for the specific DGX OS versions.

Steps:

1. If the cluster is not HA, then
   - a new `/cm/shared-kube` directory can be created
   - a new fsexport is created on the headnode:

     Example

     ```
     [head]% device use master; fsexports
     [head->device[head]->fsexports*]% add /cm/shared-kube@internalnet
     [head->devi...*[/cm/shared-kube@internalnet*]]% set path /cm/shared-kube
     [head->devi...*[/cm/shared-kube@internalnet*]]% set network internalnet
     [head->devi...*[/cm/shared-kube@internalnet*]]% set write yes
     [head->devi...*[/cm/shared-kube@internalnet*]]% commit
     [head->device[head]->fsexports[/cm/shared-kube@internalnet]]%
     ```

2. If the cluster is HA, then
   - a new directory or export can be created in the NAS to store the Kubernetes directory
   - if the headnode is a Kubernetes workers, or if the head node is a Kubernetes master that uses volumes using pods, then volumes will be used. Create an fsmount on the head nodes with:

     Example

     ```
     [head->device[head]->fsmounts]% add /cm/shared/apps/kubernetes
     [head->...red/apps/kubernetes*]]% set device 10.180.81.165:/brdkube
     [head->..*]->fsmounts*[/cm/shared/apps/kubernetes*]]% set filesystem nfs
     [head->device*[head*]->fsmounts*[/cm/shared/apps/kubernetes*]]% commit
     ```

3. Create an fsmount in the Kubernetes worker nodes categories (and/or nodes):

   Example

   ```
   [head->category[default]->fsmounts]% add /cm/shared/apps/kubernetes
   [head->...red/apps/kubernetes]]% set device $localnfsserver:/cm/shared-kube
   [head->...*]->fsmounts*[/cm/shared/apps/kubernetes*]]% set filesystem nfs
   [head->...*]->fsmounts*[/cm/shared/apps/kubernetes*]]% commit
   [head->category[default]->fsmounts[/cm/shared/apps/kubernetes]]%
   ```

```
```

4. Turn off all the Kubernetes worker nodes using `cmsh`

   This ensures that all pods are down, and not writing to the PVCs

5. On the head node, populate the whole directory with all the data from Kubernetes:

   Example

   ```
   rsync -ar /cm/shared-ubuntu2204-x86_64/apps/kubernetes/ \
   /cm/shared/apps/kubernetes/
   ```

6. If present, copy specific PVCs from Ubuntu 24.04 as well:

   Example

   ```
   rsync -ar /cm/shared-ubuntu2404-x86_64/apps/kubernetes/default/var/volumes/\
    /cm/shared/apps/kubernetes/var/volumes/
   ```

7. Turn on all the Kubernetes worker nodes
8. Ensure the Kubernetes worker nodes mount the new `/cm/shared/apps/kubernetes` directory

   Example

   ```
   mount | grep /cm/shared
   ```

9. Ensure all the pods are up and running again:

   Example

   ```
   kubectl get pods -A
   ```

10. Afterwards, clean up the old PVCs:

    Example

    ```
    rm -rf /cm/shared-ubuntu2204-x86_64/apps/kubernetes/default/var/volumes/
    rm -rf /cm/shared-ubuntu2404-x86_64/apps/kubernetes/default/var/volumes/
    ```

Resolving The Issue For The Upgrade Case Only

A solution in the case of an image upgrade, for example from DGX OS 6 to DGX OS 7, can be carried out as follows:

Before starting the upgrade of DGX OS version, you can move all PVCs from the old to the new /cm/shared.

We assume old nodes are running Ubuntu 22 and that they need to be upgraded to Ubuntu 24.

1. Turn off all the (Ubuntu 22.04) Kubernetes worker nodes via `cmsh`

   This ensures that all pods are down and not writing to the PVCs.

2. Upgrade DGX OS to Ubuntu 24.04

3. On the head node, copy all the PVCs to the other new /cm/shared

   Example

   ```
   rsync -ar  /cm/shared-ubuntu2204-x86_64/apps/kubernetes/default/var/volumes/ \
    /cm/shared-ubuntu2404-x86_64/apps/kubernetes/default/var/volumes/
   ```

4. Turn on all the Kubernetes worker nodes and check that all the pods are up and running again (`kubectl get pods -A`)

5. Afterwards, clean up the old PVCs:

   Example

   ```
   rm -rf /cm/shared-ubuntu2204-x86_64/apps/kubernetes/default/var/volumes/
   ```

Resolving The Issue If You Already Have Duplicated PVCs

A solution in the case of PVCs that have already been duplicated requires some finesse.

1. Detect all PVCs in the old and new `/cm/shared` and mapping between them. First look for current PVCs and ensure each one has the related directory in the new shared directory:

   - `kubectl get pvc -A`
   - `ls /cm/shared-ubuntu2404-x86_64/apps/kubernetes/default/var/volumes/`

2. Turn off the nodes

3. Backup or move the newly created PVCs, that most likely should be deleted:

   Example

   ```
   mv /cm/shared-ubuntu2404-x86_64/apps/kubernetes/default/var/volumes \
    /cm/shared-ubuntu2404-x86_64/apps/kubernetes/default/var/volumes-backup
   ```

4. Now, for each PVC, you have to find the original/correct one in the former shared directory. This is a manual process, and a bit of an art. You can consider:

   - The directory name which matches the Kubernetes namespace and PVC name
   - If there are 2 or more matching directories then:

- look for the latest modification date (`ls -lah`)
- check the overall size (`du -hs`). Usually the largest directory has more data and should be the most recent one

5. Run an `rsync` command per PVC:

Example

```
rsync -ar \
/cm/shared-ubuntu2204-x86_64/apps/kubernetes/default/var/volumes/\
pvc-98975be7-4030-4f11-8680-46a38829e2e7_runai-backend_data-runai-backend-\
redis-queue-master-0/ \
/cm/shared-ubuntu2404-x86_64/apps/kubernetes/default/var/volumes/\
pvc-98975be7-4030-4f11-8680-46a38829e2e7_runai-backend_data-runai-backend-\
redis-queue-master-0/
```

Note: the source and target directories are different (`...-ubuntu2204-...` to `...-ubuntu2404-...`). This is because the source should match what you have found out in the previous step, while the target should match what you listed initially.

6. Turn on nodes and verify pods are up and running

2.7.2 Miscellaneous Known Issues

- On clusters with Jupyter deployed, after the upgrade, users might find themselves being asked to login twice. The first time when the user will be asked to login to JupyterHub, and a second time, where the user is asked about a "CMDaemon login". In such cases, the user should once again present the credentials. This extra login request is harmless, and after the credentials are given a second time, it is unlikely that users will be queried again. The cause of this issue is that the Jupyterhub component that interacts with CMDaemon can lose connection to CMDaemon, causing it to prompt for an extra login.

- On some base distributions, depending on the version of the installed packages, when starting `cm-upgrade`, a YAMLLoadWarning message may be printed when its configuration files are loaded. The warning message can be safely ignored.

- In some cases because `/cm/shared` may be temporarily not accessible from the cloned primary head node when it boots for the first time, if workload managers are setup on the cluster, then CMDaemon may create and leave module files called `no-version`. For example:
`slurm/slurm/no-version`
To resolve the issue, the leftover `no-version` module files under `/cm/shared/modulefiles/` can be deleted.

- In some cases at the end of the head node or the software image(s) upgrade, the `cm-upgrade` script may report a list of packages that were not upgraded to their BCM 11 versions. This can occur in cases where the package manager was not able to resolve the list of packages to find a suitable candidate at the time of the upgrade of all packages.

  While `cm-upgrade` makes a best effort to swap or upgrade the packages to their BCM 11 versions, in some cases— notably on Ubuntu, it is also possible some packages are left in their 10.0 versions.

  The administrator is advised to manually swap the `cm10.0` packages with the `cm11.0` package.

- On clusters with cluster extension to AWS deployed, the cloud director and cloud nodes might fail to start after the upgrade. This can happen when there is an entry

  ```
  <blockdev mode="cloud">/dev/nvme1n1</blockdev>
  ```

  in the disksetup. The issue can be resolved by removing that entry from the disksetup for the cloud director and cloud nodes.

- On clusters that have PBSPro configured, and upgraded from BCM 10 to BCM 11, new jobs submitted by users will get job ids that were already used. This is particularly relevant for users that use BCM monitoring data for billing purposes.

- Disabling Kubernetes as part of the parallel-upgrade procedure results in an apparent failure during the stage when etcd data is cleaned up, because the compute nodes are not reachable from the parallel setup. This is expected since the parallel setup is isolated from the production setup. To resolve the issue, the administrator should answer a prompt so that the failed stage is skipped. It is also recommended to re-provision the compute nodes of the upgraded parallel setup before attempting to set up Kubernetes or etcd again, so that any leftover data from before the upgrade is cleaned during the provisioning.

- When upgrading the DGX OS 6 sofware image to the latest updates of DGX OS 6, there is a known issue with DKMS when a new kernel `linux-image-5.15.0-1090-nvidia` is installed. The DKMS post-install script fails for the new kernel. This is caused by duplicate entries for the `gdrdrv` module. This can be checked for with the following command:

  ```
  root@dgx-os-6:/# dkms status
  gdrdrv/2.4.4, 5.15.0-1063-nvidia, amd64: installed
  gdrdrv/2.4.4, 5.15.0-1090-nvidia, amd64: installed
  gdrdrv/2.4.4, 5.15.0-1090-nvidia, x86_64: built
  ```

  The last two statuses are for the same driver, and the last line is `built` rather than `installed`, which means it was not installed, and leads to the post-install script failure.

  This issue can be resolved by removing the duplicate entry and reconfiguring the package as follows:

  ```
  dkms remove gdrdrv/2.4.4 -a amd64 -k 5.15.0-1090-nvidia
  dpkg --configure -a
  ```

- When checking if the compute nodes are UP, in some cases `cm-upgrade` may wrongly report that some nodes are not DOWN. The report depends on how a node has been shut down, the state of the node not being updated correctly in the CMDaemon database. In such scenarios, after confirming that the nodes are indeed DOWN, re-run `cm-upgrade` with the `--skip-node-states-check` flag to skip the check for node states:

  Example

  ```
  cm-upgrade --skip-node-states-check
  ```

- Upgrading from BCM 10 to BCM 11 for DGX SuperPOD deployments requires filesystem exports to be explicitly set for the network `dgxnet` (page 64).

- The BCM machine learning repositories may still be referenced in the repository lists. They should be removed. For example, in the Ubuntu head node and images with:

```
rm /etc/apt/sources.list.d/cm-ml.list
```

2.8 Upgrading Using A BCM DVD/ISO

When using a BCM DVD/ISO to perform the upgrade, it is important to use a DVD/ISO that is not older than **11.30.0**. The DVD/ISO version can be found (assuming that the DVD/ISO is mounted under `/mnt/cdrom`) with a `find` command such as:

```
# find /mnt/cdrom -type d -name '11.30.0'
/mnt/cdrom/data/packages/11.30.0
```

2.9 Enable Upgrade

2.9.1 Enable The Upgrade Repo, And Install The Upgrade Package

Enabling the upgrade repository and installing the upgrade package can be carried out for the various distributions as follows:

RHEL8 derivatives:

```
# yum-config-manager \
  --add-repo http://support.brightcomputing.com/upgrade/11.0/rhel/8/updates
# yum --nogpgcheck install cm-upgrade
```

RHEL9 derivatives:

```
# yum-config-manager \
--add-repo http://support.brightcomputing.com/upgrade/11.0/rhel/9/updates
# yum --nogpgcheck install cm-upgrade
```

SLES 15:

```
# zypper addrepo \
http://support.brightcomputing.com/upgrade/11.0/sles/15/updates cm-upgrade-11
# zypper install cm-upgrade
```

Ubuntu 22.04:

```
# cat <<EOF > /etc/apt/sources.list.d/cm-upgrade-11.list
deb [trusted=yes] https://support.brightcomputing.com/upgrade/11.0/ubuntu/2204/ ./
EOF
# apt-get update
# apt-get install cm-upgrade
```

Ubuntu 24.04:

```
# cat <<EOF > /etc/apt/sources.list.d/cm-upgrade-11.list
deb [trusted=yes] https://support.brightcomputing.com/upgrade/11.0/ubuntu/2404/ ./
EOF
# apt-get update
```

```
# apt-get install cm-upgrade
```

2.9.2 Load The Environment Module

```
# module load cm-upgrade/11.0
```

2.10 Perform Upgrade

The steps for carrying out the upgrade are as follows:

2.10.1 Apply Updates To Head Node

- RHEL derivatives:

```
# yum update
```

- SLES derivatives:

```
# zypper up
```

- Ubuntu:

```
# apt-get update
# apt-get upgrade --with-new-pkgs
```

2.10.2 Apply Updates To Software Images

Before updating and upgrading a software image, it may be worth creating an image backup by cloning the image (page 211 of the *Administrator Manual*). If the cluster after the upgrade has an image that has an issue, then comparing it against the backup can help troubleshoot the issue.

For each software image, updates can be applied as follows:

- RHEL derivatives:

```
# yum --installroot /cm/images/<software image> update
```

- SLES derivatives:

```
# zypper --root /cm/images/<software image> up
```

- Ubuntu:

```
# cm-chroot-sw-img /cm/images/<software image>
# apt-get update
# apt-get upgrade --with-new-pkgs
# exit
```

2.10.3 Apply Updates To The `node-installer` Image

- RHEL derivatives:

```
# yum --installroot /cm/node-installer update
```

- SLES derivatives:

```
# zypper --root /cm/node-installer up
```

- Ubuntu:

```
# cm-chroot-sw-img /cm/node-installer
# apt-get update
# apt-get upgrade --with-new-pkgs
# exit
```

2.10.4 Upgrade Head Nodes To BCM 11

Important: this must be run on both head nodes in a high availability setup. Recommended: Upgrade the active head node first and then the passive head node.

Depending on availability, the upgrade can be run either over the network, or using a DVD/ISO:

- Upgrade using repositories accessible over the network

```
# cm-upgrade
```

- Upgrade using a BCM DVD/ISO

```
# cm-upgrade -b /root/bcm-11.0-ubuntu2404.iso
```

In an HA setup, after upgrading both the head nodes, resync the databases. Run the following from the active head node (it is very important to complete this step before moving to the next one):

```
# cmha dbreclone <secondary>
```

2.10.5 Reboot Head Node(s)

The head node(s) must be rebooted before proceeding to run the post-upgrade actions.

2.10.6 Post-upgrade Head Node Actions

Important: these must be run on both head nodes in an HA setup

```
# module load cm-upgrade/11

# cm-post-upgrade -m
```

2.10.7 Upgrade The Software Image(s) To BCM 11

Important: the upgrade to the software images must be done only on the active head node. Depending on availability, the upgrade can be run either over the network, or using a DVD/ISO:

- Upgrade using repositories accessible over the network

```
# cm-upgrade -i all
```

- Upgrade using a BCM DVD/ISO

```
# cm-upgrade -i all -b /root/bcm-11.0-ubuntu2404.iso
```

If the software images are not under the standard location—`/cm/images/` on the head node—then the option `-a` should be used:

```
# cm-upgrade -a /apps/images -i <name of software image>
# cm-upgrade -a /apps/images -i <name of software image> \
  -b /root/bcm-11.0-ubuntu2404.iso
```

If the software images are located under multiple locations, the target locations can be specified as a comma-separated list:

```
# cm-upgrade -a '/apps/images,/opt/images' -i all
```

2.10.8 Upgrade /cm/node-installer To BCM 11

This needs to be performed only on the primary active head node, either over the network, or using a BCM DVD/ISO.

- Upgrade using repositories accessible over the network:

```
# cm-upgrade -x
```

- Upgrade using a BCM DVD/ISO:

```
# cm-upgrade -x -b /root/bcm-11.0-ubuntu2404.iso
```

2.10.9 Post-upgrade Software Images Actions

Important: these must be run only on the active head node.

```
# cm-post-upgrade -i all
```

If the software images are not under the standard location, which is `/cm/images/` on the head node, then the option `-a` should be used:

```
# cm-post-upgrade -a /apps/images -i all
# cm-post-upgrade -a /apps/images -i <name of software image>
```

If the software images are located under multiple locations, then the target locations can be specified as a comma-separated list:

```
# cm-post-upgrade -a '/apps/images,/opt/images' -i all
```

2.10.10 NVIDIA Driver Installation

During the upgrade to BCM 11.0, `cm-upgrade` automatically installs NVIDIA GPU drivers on non-DGX software images. This is only carried out on software images. It is not carried out on the head nodes or the node-installer images.

Default NVIDIA Driver Version

By default, `cm-upgrade` installs `NVIDIA driver version 570 (Open Kernel Modules edition)`. The following components are installed:

- On Ubuntu 22.04 and 24.04: the package installed is the `nvidia-driver-570-open` meta-package plus additional components including:
  - Fabric Manager (for NVLink/NVSwitch management)
  - NSCQ library (NVIDIA NVSwitch Configuration and Query Library)
  - IMEX library (GPU memory sharing across different compute nodes in a multi-node system connected by NVLink)
  - NVSDM library (for monitoring NVSwitch devices on NVIDIA Blackwell systems)
- On RHEL 9/Rocky 9: a comprehensive set of packages based on the major 570 version. The package set installed includes:
  - DKMS and open kernel modules for dynamic kernel support
  - Core driver libraries and CUDA components
  - Fabric Manager for multi-GPU interconnect management
  - GUI tools (nvidia-settings, nvidia-xconfig) for X11 display configuration

  Most package names are generic (for example: nvidia-driver, nvidia-fabric-manager), with the exact version version determined by the repository configuration
- On SLES 15 SP6: the package installed is a set based on the major 570 version, and includes:
  - Open kernel modules (`nvidia-open-570`)
  - Fabric Manager and NSCQ library

  Packages are sourced from NVIDIA's CUDA repository (version-numbered). Native SLES packages use G06 generation naming, but we use NVIDIA's CUDA repository for consistency

Non-DGX Systems Only

NVIDIA driver installation is only carried out on non-DGX systems.

DGX systems have pre-installed NVIDIA drivers, which are maintained by NVIDIA's own update mechanisms, and should not managed by `cm-upgrade`. The upgrade script detects DGX systems by checking for the presence of `/etc/dgx-release` in the software image. If this file exists, then NVIDIA driver installation in the DGX system is skipped.

Changing the NVIDIA Driver Version

If you need to install a different version of the NVIDIA drivers, then you can modify the driver version either before or after the upgrade as follows:

- Pre-upgrade: Edit /cm/local/apps/cm-upgrade/11.0/conf/packages.yaml and modify the package versions listed under the `nvidia_driver_packages` section for your distribution.
- Post-upgrade: Use the standard package management tools (`yum`/`apt`/`zypper`) within the software image to remove the installed driver packages and install your preferred version.

2.10.11 (HA) Allow For The Provisioning Requests To The Secondary Head Node To Complete

Allow between 15 and 30 seconds for CMDaemon to schedule and start provisioning updates of `/tftpboot` and `/cm/node-installer` from the primary to the secondary head node. Then check and wait until the provisioning requests with the secondary head node as destination have completed using:

```
# cmsh -c 'softwareimage provisioningstatus -r'
```

If the provisioning requests with a destination node the secondary head node continue to be listed for a prolonged period of time and do not disappear from the list, then the `node-installer` and/or `tftpboot` on the secondary head node are out of sync from the primary head node, which can result in issues with provisioning of the compute nodes. Please contact support (`https://enterprise-support.nvidia.com/s/create-case`) for further assistance.

2.10.12 Enable Mixed OS Setup And Create A DGX OS 7 Image

On BCM 10 Ubuntu 22.04 clusters can use DGX OS 6 images. DGX OS 6 images are not supported in BCM 11.

A new DGX OS 7 software image must therefore be used. The image can be created using `cm-image` and `cm-create-image`. These are BCM utilities, and are described in section 9.6 of the *Administrator Manual*.

DGX OS 7 is based on Ubuntu 24.04. This then also means the mixed OS environment—multidistro or multiOS as described in section 9.7 of the *Administrator Manual*—must be enabled, to allow provisioning of DGX OS 7 images from the Ubuntu 22.04 head node of BCM 10.

Image creation can be carried out as follows:

1. A script may need to be created first:

   Example

   ```
   [root@basecm11 ~]# cat disable-upstream-repos.sh
   #!/bin/bash
   if [ -f "$1/etc/apt/sources.list.d/ubuntu.sources" ]; then
           mv $1/etc/apt/sources.list.d/ubuntu.sources \
           $1/etc/apt/sources.list.d/ubuntu.sources.dvd_bak
   fi
   ```

   The script is only needed for a cluster running a BCM version 11 earlier than 11.31.0. It works around a known issue that may arise from using an upstream repository in an inappropriate manner. For BCM 11.31.0 and later, the workaround is carried out automatically, and the script is not needed.

2. Use `cm-image` to create the appropriate the Ubuntu 24.04 base images and enable a mixed OS setup. The `-z` option that calls the script need only be used for BCM 11 installations earlier than BCM 11.31.0.

   Example

```
[root@basecm11 ~]# module load cm-image
[root@basecm11 ~]# cm-image --v create all -a x86_64 -d ubuntu2404 --source \
                /path/to/bcm11.0-ubuntu24.04-dgx-os-7.2.iso \
                -z disable-upstream-repos.sh
```

3. Use `cm-create-image` to create the new DGX OS image. The `-z` option that calls the script need only be used for BCM 11 installations earlier than BCM 11.31.0.

```
[root@basecm11 ~]# cm-create-image --cmdvd \
                /root/bcm-11.0-ubuntu2404-dgx-os-7.2.iso \
                --no-cm-cuda-repo --extra-pkg-group \
                doca_ofed_2.9.1-3.0.2 --dgx --dgx-type dgx_b200 \
                --imagename dgx-os-image-b200 \
                -z disable-upstream-repos.sh
```

The `--dgx-type` option (page 546 of the *Administrator Manual*) must be set for every type that is to be used as an image. The possible values are:

- `dgx_h100` for the H100 GPU
- `dgx_b200` for the B200 GPU
- `dgx_a100` for the A100 GPU
- `dgx_gb200` for the GB200 GPU
- `dgx_h200` for the H200 GPU

The cluster administrator should create a new category for each new image. Cloning the `default-image` image from the `default` category is a convenient way to do this since it already has the correct fsmounts. The image name should then be set for the category.

For example, for a B200 image `dbx-os-image-b200` that has been placed in /cm-images/:

Example

```
root@basecm11:~# cmsh
[basecm11]% category
[basecm11->category]% list
Name (key)              Software image          Nodes
----------------------- ----------------------- --------
default                 default-image           1
[basecm11->category]% clone default gb200
[basecm11->category*[gb200*]]% use gb200
[basecm11->category*[gb200*]]% set softwareimage dgx-os-image-b200 ; commit
...
```

The nodes that are to use the new image should then be assigned their image. For example, a `node001` can be assigned to the `gb200` category created in the preceding example with:

Example

```
[basecm11->device]% set node001 category gb200 ; commit
```

Configuration overlays that were used in the old category should also be migrated as needed to the new category.

Upgrading from BCM 10 to BCM 11 for DGX SuperPOD deployments requires NFS exports to be explicitly set for the network `dgxnet`, for `/cm/node-installer-ubuntu2404-x86_64` and `/cm/node-installer-ubuntu2404-x86_64/certificates`.

This can be configured with the following `cmsh` session on the head node:

Example

```
[root@basecm11 ~]# cmsh
[basecm11]% device use master
[basecm11->device[basecm11]]% fsexports
[basecm11->...->fsexports]% add /cm/node-installer-ubuntu2404-x86_64@dgxnet
[basecm11->...installer-ubuntu2404-x86_64@dgxnet*]]% set network dgxnet
[basecm11->...installer-ubuntu2404-x86_64@dgxnet*]]% commit
[basecm11->...installer-ubuntu2404-x86_64@dgxnet]]% ..
[basecm11->...rts]% add /cm/node-installer-ubuntu2404-x86_64/certificates@dgxnet
[basecm11->...-ubuntu2404-x86_64/certificates@dgxnet*]]% set network dgxnet
[basecm11->...-ubuntu2404-x86_64/certificates@dgxnet*]]% set write yes
[basecm11->...-ubuntu2404-x86_64/certificates@dgxnet*]]% commit
[basecm11->...-ubuntu2404-x86_64/certificates@dgxnet]]%
```

The same procedure must also be carried out for that secondary head node.

The new DGX OS image has no Kubernetes packages installed on it by default. Migrating an old Kubernetes installation over from the old image is described on page 49.

2.10.13 Upgrading CMDaemon Lite

The BCM 10 to BCM 11 upgrade supports CMDaemon Lite (section 2.6.7 of the *Administrator Manual*) running on cluster switches, such as Cumulus switches.

After the upgrade has been carried out on the head node and the images are ready for deployment, then the switches can be rebooted to pick up their new image.

- It is important not to reboot the switch that is being used to connect to the head node.

After the switches have rebooted, the compute nodes can be powered on.

In a cluster, mixing deployments of cm-lite-daemon for BCM 10 with those of BCM 11 is not supported.

This means that the CMDaemon Lite servers that are not managed by CMDaemon, for example PCs running CMDaemon Lite in a Python environment, also need to be upgraded when upgrading the cluster itself from BCM 10 to BCM 11. That must be done manually, by removing the BCM 10 package, and installing the BCM 11 package according to the environment and OS it is running in.

2.10.14 The Cluster Is Now Upgraded To BCM 11

It is recommended to provision the compute nodes at least once with BCM 11 before setting up any new integrations.

2.11 Simple Validation Checks

Some basic checks can be carried out to validate the cluster. If a check fails, then further investigation should be carried out.

2.11.1 Validating CMDaemon

The CMDaemon service status (section 2.6.1 of the *Administrator Manual*), can be checked with:

Example

```
root@basecm11:~# systemctl status cmd
  cmd.service - BCM daemon
     Loaded: loaded (/usr/lib/systemd/system/cmd.service; enabled; preset: enabled)
     Active: active (running) since Mon 2025-12-01 01:44:17 PST; 6h ago
   Main PID: 2547 (safe_cmd)
      Tasks: 77 (limit: 65535)
     Memory: 576.1M (high: 3.4G max: 3.6G available: 1.3G peak: 600.1M)
        CPU: 6min 56.963s
     CGroup: /system.slice/cmd.service
...
```

2.11.2 Validating High Availability

If the cluster has HA configured, then the cluster manager HA status (page 782 of the *Administrator Manual*), can be checked. For example, for head nodes `basecm11-a` and `basecm11-b` the response to the check would be:

Example

```
root@basecm11-a:~# cmha status
Node Status: running in active mode

basecm11-a* -> basecm11-b
  failoverping  [  OK  ]
  mysql         [  OK  ]
  ping          [  OK  ]
  status        [  OK  ]

basecm11-b -> basecm11-a*
  failoverping  [  OK  ]
  mysql         [  OK  ]
  ping          [  OK  ]
  status        [  OK  ]
```

2.11.3 Validating `cmsh` And Devices

A check on the device statuses can be run with:

Example

```
root@basecm11:~# cmsh -c "ds"
basecm11.............[   UP   ]
node001 .............[   UP   ]
...
```

2.11.4 Validating The Cluster Health

Health check values can be retrieved from devices in the BCM monitoring system with the help of the `latesthealtdata` command (section 10.6.3 of the *Administrator Manual*):

Example

```
[basecm11->device]% foreach -v * (latesthealthdata)
================================ basecm11 ====================================
```

```
Measurable          Parameter    Type          Value      Age         State      Info
------------------- ------------ ------------ ---------- ---------- ---------- ----------
ManagedServicesOk                Internal     PASS       1m 39s
chrootprocess                    OS           PASS       1.23s
cm-chroot-sw-img                 OS           PASS       1m 18s
cmsh                             Internal     PASS       25m 7s
cuda-dcgm                        OS           PASS       54.8s
defaultgateway                   Network      PASS       1m 8s
diskspace                        Disk         PASS       14s
dmesg                            OS           PASS       1m 51s
exports                          Disk         PASS       9.21s
interfaces                       Network      PASS       1m 7s
ldap                             OS           PASS       1m 37s
lustre                           Disk         PASS       58.8s
mounts                           Disk         PASS       1m 40s
mysql                            OS           PASS       53m 47s
ntp                              Internal     PASS       1m 49s
oomkiller                        OS           PASS       18s
ssh2node                         network      PASS       8m 58s
================================ node001 ===================================
Measurable          Parameter    Type          Value      Age         State      Info
------------------- ------------ ------------ ---------- ---------- ---------- ----------
ManagedServicesOk                Internal     PASS       2m 54s
cuda-dcgm                        OS           PASS       2m 9s
defaultgateway                   Network      PASS       2m 13s
diskspace                        Disk         PASS       3m 28s
dmesg                            OS           PASS       3m 5s
interfaces                       Network      PASS       2m 22s
ldap                             OS           PASS       2m 51s
lustre                           Disk         PASS       2m 13s
mounts                           Disk         PASS       2m 55s
ntp                              Internal     PASS       3m 3s
oomkiller                        OS           PASS       1m 32s
ssh2node                         network      PASS       8m 43s
...
```

2.11.5 Validating Base View

For a dotted quad IP address <*IP address*>, the cluster administrator can connect to:

`https://`<*IP address*>`:8081/base-view`

and try logging in. If there was non-administrator access before the upgrade, then that should also be checked.

2.11.6 Validating The State Of The Packages On The Cluster

The state of packages according to the package manager for the cluster can be checked in the head nodes and in the software images with:

Ubuntu Packages

```
apt-get check
dpkg --audit
```

RHEL-like Distribution Packages

```
yum check
rpm -Va --nofiles
```

SLES Packages

```
zypper verify
rpm -Va --nofiles
```

These commands should all normally give no output, other than processing messages in some cases.

2.11.7 Validating Slurm And Pyxis

Slurm and Pyxis validation on the cluster (page 356 of the *Administrator Manual*), can be carried out as follows:

Validating Slurm

```
root@basecm11:~# module load slurm
root@basecm11:~# scontrol show config
Configuration data as of 2025-12-02T04:06:05
AccountingStorageBackupHost = (null)
AccountingStorageEnforce = none
AccountingStorageHost   = master
AccountingStorageExternalHost = (null)
AccountingStorageParameters = (null)
AccountingStoragePort   = 6819
...
Slurmctld(primary) at basecm11 is UP
```

Validation Of Pyxis

```
root@basecm11:~# srun --container-image=ubuntu grep PRETTY /etc/os-release
pyxis: importing docker image: ubuntu
pyxis: imported docker image: ubuntu
PRETTY_NAME="Ubuntu 24.04.3 LTS"
root@basecm11:~#
```

2.11.8 Validating Kubernetes

The `kube-system` pods should be inspected to see if they are running along without any real issues:

Example

```
root@basecm11:~# kubectl get pods -n kube-system
NAME                                  READY   STATUS    RESTARTS   AGE
coredns-66bc5c9577-bv577              1/1     Running   0          4h24m
coredns-66bc5c9577-rsfpv              1/1     Running   0          4h24m
kube-apiserver-basecm11               1/1     Running   0          4h19m
kube-apiserver-node001                1/1     Running   2          4h24m
kube-controller-manager-basecm11      1/1     Running   0          4h23m
kube-controller-manager-node001       1/1     Running   2          4h25m
kube-proxy-2ckqr                      1/1     Running   0          4h24m
...
kube-scheduler-node001                1/1     Running   1          4h25m
kube-state-metrics-6b98df4478-nhkd2   1/1     Running   0          4h19m
metrics-server-f49fdc486-4ndfk        1/1     Running   0          4h19m
metrics-server-f49fdc486-5rvbf        1/1     Running   0          4h19m
```

Are the operators running along fine too?

```
root@basecm11:~# kubectl get pods --all-namespaces
NAMESPACE      NAME                                       READY   STATUS    RESTARTS   AGE
calico-system  calico-apiserver-577cdf67b5-645qn          1/1     Running   0          4h49m
```

```
...
calico-system  calico-typha-6495c7469f-gpztc              1/1     Running   0          4h49m
calico-system  calico-typha-6495c7469f-rp9rz              1/1     Running   0          4h49m
calico-system  goldmane-7b6b78f6f-tn65t                  1/1     Running   0          4h49m
calico-system  whisker-6c95b488ff-7s9pf                  2/2     Running   0          4h49m
cm             local-path-provisioner-675dc9fd86-b6fg6   1/1     Running   0          4h48m
cmkpm-system   cmkpm-controller-manager-66b99b6699-6k6gm  1/1     Running   0          4h48m
grafana        grafana-grafana-operator-c65d95b7f-c5p59  1/1     Running   0          4h47m
ingress-nginx  ingress-nginx-controller-68994db546-7c2h9  1/1     Running   0          4h44m
ingress-nginx  ingress-nginx-controller-68994db546-p6pjv  1/1     Running   0          4h44m
kube-system    coredns-66bc5c9577-bv577                  1/1     Running   0          4h51m
...
```

3 In-place Upgrades

3.1 Introduction

In contrast with a parallel upgrade (Chapter 2), with an in-place upgrade the production cluster is taken out of operation while the upgrade takes place.

This chapter covers the procedure for an in-place upgrade.

3.2 Existing Cluster Health Check

It is a good idea to check that the existing cluster is in a healthy state before carrying out the upgrade.

The head nodes health checks in particular should be looked at. Unexpected health check failures should be examined, and if required, resolved. In some cases it is expected that health checks will be in a failed state. For example, if all the compute nodes are DOWN, then the schedulers health check fails.

The reason behind checking the health of the cluster before upgrading is that the health checks can detect failing services, overfull filesystems, unmounted filesystems, networking issues, time synchronization issues, and other issues. If these are unexpected then they can prevent the upgrade from completing successfully.

3.3 Important: Upgrading Particular Applications

3.3.1 Overview Of The Applications That `cm-upgrade` Does And Does Not Manage

Automatically Upgraded By `cm-upgrade`

The following software, if managed by BCM 10, is managed and upgraded automatically by `cm-upgrade` to BCM 11:

- Docker
- The `cm-docker-registry` package (Harbor upgrades (section 3.3.6) are not supported)
- Jupyter (the JupyterHub database requires some additional care (section 3.3.5).
- Base View: The regular version of Base View in BCM 10 is upgraded to a new version in BCM 11. Any monitoring dashboards from the regular BCM 10 Base View version are no longer used in the BCM 11 Base View version (section 3.3.2).
- NVIDIA drivers: The defaults will be installed in non-DGX images

Automatically Kept As Is By `cm-upgrade`

The following applications are allowed to remain during the upgrade, and are not automaticaly upgraded:

- WLMs (section 3.3.3)
  - Slurm
  - PBS Professional/OpenPBS
  - LSF

  The installation of workload managers is described in section 7.3 of the *Administrator Manual*, while the upgrade of the Slurm and PBS Professional workload managers is described in section 7.5 of the *Administrator Manual*.
- Kubernetes: Kubernetes installation is described in section 4.21 of the *Containerization Manual*.
- Run:Ai: Run:Ai installation options for BCM are described in section 6.8 of the *Containerization Manual*.
- NVIDIA Container Toolkit: An NVIDIA Container Toolkit installation can be carried out as part of Docker setup or a package installation (section 2.7 of the *Containerization Manual*).
- other Kubernetes operators: The installation of other Kubernetes operators is described in Chapter 6 of the *Containerization Manual*.

Manually Removed And Reinstalled During Upgrade (Not Managed By `cm-upgrade`)

- Harbor: As explained earlier, Harbor upgrades (section 3.3.6) are not supported. The Harbor package needs to be removed from BCM version 10, and redeployed (section 3.2 of the *Containerization Manual*) in BCM version 11.
- Cluster Extension: Needs to be removed and redeployed (Chapter 3 of the *Cloudbursting Manual*)

Removed From Support During Upgrade (Not Managed By `cm-upgrade`)

The following software is no longer managed by BCM in BCM version 11.

- (Altair) Grid Engine (AGE): BCM support for integration ended. Possible alternatives to the AGE WLM, that BCM supports, are Slurm, PBS Professional/OpenPBS, or LSF.
- Ceph: BCM support for integration ended.
- BeeGFS: BCM support for integration ended.

3.3.2 Upgrading Base View

BCM has Base View (section 2.4 of the *Administrator Manual*) as the browser GUI to carry out cluster management tasks. It is available as the package `base-view`.

Base View in BCM 10 had an additional, experimental and undocumented, Base View NG, available as the package `base-view-ng`. Base View NG uses a newer GUI toolkit from that of Base View in BCM 10.

Base View in BCM 11 is actually just a development of Base View NG from BCM 10, and so it uses that newer toolkit too.

The new Base View in BCM 11 means that an upgrade from BCM 10 to BCM 11 loses all the dashboards that users and administrators may have built for Base View in BCM 10.

It is therefore a good idea to make notes on the useful dashboards of BCM 10 so that analogous ones can be constructed for BCM 11. A useful transition may be to construct the analogues in Base View NG while still on BCM 10, before upgrading to BCM 11.

3.3.3 Upgrading Workload Managers (WLM)

Unsupported unversioned or unsupported versioned WLM packages are not removed automatically. They must be removed manually. If they are to be updated then it is recommended to install new, supported, versioned WLM packages.

The supported PBSPro versions are compatible across BCM 10 and BCM 11.

Upgrading WLMs In General From BCM 10

WLMs are not required to be disabled using `cm-wlm-setup` before upgrading the BCM version from 10 to 11.

Upgrading Slurm

Slurm versions older than Slurm 24.11 are not supported in BCM 11. When upgrading from BCM 10, Slurm must be upgraded to at least Slurm 24.11 in the BCM 10 repositories before moving to BCM 11.

For more information on upgrading Slurm, please refer to the KB article `https://kb.brightcomputing.com/knowledge-base/upgrading-slurm/`.

Packages: Software images now include the same packages as the head nodes. This is because starting with BCM 11, Slurm packages keep many things in `/cm/local`, instead of `/cm/shared`.

Slurm client packages, for example `slurm25.05-client`, are no longer available. So they are removed as part of the upgrade procedure.

There are now separate `slurmctld` and `slurmd` packages that provide the binary and systemd unit files for each service.

Path adjustments: There are some path adjustments for Slurm for the upgrade from BCM 10 to BCM 11, as summarized in the following table.

Slurm item	Version	Path
Prefix directory	BCM10	/cm/shared/apps/slurm
	BCM11	/cm/local/apps/slurm
Most configuration files	BCM10	/cm/shared/apps/slurm/var/etc/
	BCM11	/cm/shared/apps/slurm/etc
Munge configuration files	BCM10	/cm/shared/apps/slurm/var/etc/munge
	BCM11	/cm/shared/apps/slurm/var/etc/munge (unchanged)
Binaries	BCM10	/cm/shared/apps/slurm/current/bin
	BCM11	/cm/local/apps/slurm/current/bin
Template files such as: slurmdbd.conf.template, cgroup.conf.template	BCM10	/cm/shared/apps/slurm/var/etc
	BCM11	/cm/local/apps/slurm/current/templates
statesave files	BCM10	/cm/shared/apps/slurm/var/cm/statesave
	BCM11	/cm/shared/apps/slurm/statesave
Power scripts	BCM10	/cm/shared/apps/slurm/var/cm/power
	BCM11	/cm/local/apps/slurm/current/scripts/power
Enroot prolog	BCM10	/cm/shared/apps/slurm/var/cm
	BCM11	/cm/shared/apps/slurm/prologs/prolog-enroot.sh
Enroot epilog	BCM10	/cm/shared/apps/slurm/var/cm
	BCM11	/cm/shared/apps/slurm/epilogs/epilog-enroot.sh

Slurm DBD Performance Improvement

Installations from BCM 10.24.05 onward have MySQL configuration settings changes that improve the performance of the slurmdbd service. The changes are not automatically applied during updates of MySQL when updating from versions installed earlier than BCM 10.24.05.

To apply the improvement to a cluster that does not have this update, the MySQL configuration must be manually adjusted on the head nodes after `cm-upgrade` has completed. The /etc/my.cnf file on the head node, and also on the secondary head node in an HA-enabled cluster, should have the following settings updated:

```
innodb_buffer_pool_size=4096M
innodb_log_file_size=64M
innodb_lock_wait_timeout=900
```

The default `my.cnf` file is typically symlinked by the system utility `update-alternatives` in Ubuntu and SLES, (`alternatives` in RHEL derivatives), so how it is configured should also be considered.

Slurm Upgrades For DGX OS 6 To DGX OS 7

For DGX, BCM 10 supports only DGX OS 6 (an Ubuntu 22.04 variant), while BCM 11 supports only DGX OS 7 (an Ubuntu 24.04 variant). As part of the upgrade from BCM 10 to BCM 11, the DGX OS must be replaced with DGX OS 7. When replacing the DGX OS nodes from 6 to 7, a new software image (and `/cm/shared`) is created for the Ubuntu 2404 nodes.

If Slurm and Pyxis have been deployed in a BCM 10 DGX OS 6 image, then some manual configuration changes must be done to make these work right in the BCM 11 DGX OS 7 image. These can be carried out as follows:

Changes to the MUNGE key in `/cm/shared`: The key needs to have the correct permissions and ownership. If it does not have these, then a symptom of it is the message:

```
munged: Error: Keyfile is insecure: "/cm/shared/apps/slurm/var/munge/keys/
munge.key" should be owned  by UID 1 instead of UID 0
```

as seen in the systemd status output:

```
root@dgx-04:~# systemctl status munge.service
× munge.service - MUNGE authentication service
     Loaded: loaded (/usr/lib/systemd/system/munge.service; enabled;
         preset: enabled)
     Active: failed (Result: exit-code) since Fri 2025-10-31
         03:54:29 PDT; 26s ago
       Docs: man:munged(8)
    Process: 255424 ExecStart=/usr/sbin/munged --key-file=/cm/shared/
       apps/slurm/var/munge/keys/munge.key (code=exited, status=1/
       FAILURE)Oct 31 03:54:29 dgx-04 systemd[1]: Starting
    munge.service - MUNGE authentication service...
Oct 31 03:54:29 dgx-04 munged[255424]: munged: Error: Keyfile is insecure:
  "/cm/shared/apps/slurm/var/munge/keys/munge.key" should be owned by UID 1
  instead of UID 0
Oct 31 03:54:29 dgx-04 systemd[1]: munge.service: Control process exited,
  code=exited, status=1/FAILURE Oct 31 03:54:29 dgx-04 systemd[1]:
  munge.service: Failed with result 'exit-code'.
Oct 31 03:54:29 dgx-04 systemd[1]: Failed to start munge.service -
  MUNGE authentication service.
```

The key can be made consistent with that of the previous deployment. This can be done by changing the attributes of the files in the new DGX software image, for example `dgx-os-7.2-h100-image`, after mounting it in the appropriate shared directory on the new cluster:

Example

```
cm-chroot-sw-image -s /cm/shared-ubuntu2404-x86_64 \
/cm/images/dgx-os-7.2-h100-image
...
chown daemon:root /cm/shared/apps/slurm/var/munge/keys/munge.key
chmod 400 /cm/shared/apps/slurm/var/munge/keys/munge.key
```

Changes to the Pyxis and Enroot files: The files within the DGX OS 7 image can be configured as follows:

Copy the Pyxis plugin file to the new software image, for example: `dgx-os-7.2-h100-image`:

Example

```
cp /cm/local/apps/slurm/current/lib64/slurm/spank_pyxis.so \
 /cm/images/dgx-os-7.2-h100-image/cm/local/apps/slurm/current/lib64/slurm/
```

Copy the plugstack files in `/cm/shared` over from the Ubuntu 2204 (DGX OS 6) version to the Ubuntu 2404 (DGX OS 7) version:

Example

```
cp /cm/shared-ubuntu2204-x86_64/apps/slurm/etc/slurm/plugstack.conf \
   /cm/shared-ubuntu2404-x86_64/apps/slurm/etc/slurm/
cp /cm/shared-ubuntu2204-x86_64/apps/slurm/etc/slurm/plugstack.conf.d/pyxis.conf \
 /cm/shared-ubuntu2404-x86_64/apps/slurm/etc/slurm/plugstack.conf.d/
```

Copy the Enroot configurations in `/cm/shared` from the Ubuntu 2204 version to the Ubuntu 2404 version:

Example

```
cp /cm/shared-ubuntu2204-x86_64/apps/slurm/etc/enroot.conf \
 /cm/shared-ubuntu2404-x86_64/apps/slurm/etc/
cp /cm/shared-ubuntu2204-x86_64/apps/slurm/etc/enroot-sysctl.conf \
 /cm/shared-ubuntu2404-x86_64/apps/slurm/etc/
```

Symlinks for the epilog, prolog, and `enroot.conf` files are created for the new software image:

Example

```
cm-chroot-sw-img -s /cm/shared-ubuntu2404-x86_64 /cm/images/dgx-os-7.2-h100-image/
ln -s /cm/shared/apps/slurm/prologs/prolog-enroot.sh \
 /cm/local/apps/slurm/var/prologs/50-prolog-enroot.sh
ln -s /cm/shared/apps/slurm/epilogs/epilog-enroot.sh \
 /cm/local/apps/slurm/var/epilogs/50-epilog-enroot.sh
ln -s /cm/shared/apps/slurm/etc/enroot.conf /etc/enroot/enroot.conf
<ctrl-d>
```

The compute nodes can pick up the new configuration when they get rebooted.

PBS Professional Upgrades

PBS Professional versions older than PBS Pro 2022 are not supported in BCM 10. Below is an overview of the steps that can be taken to upgrade PBS Pro:

- Stop the `pbsserver` service on the head node

  ```
  cmsh% device
  cmsh% foreach -l pbsproserver ( services; stop pbsserver )
  ```

- Remove the old packages from the headnodes

```
$ yum/zypper/apt remove pbspro<old version>*
```

- Remove the old packages from the software images

```
$ cm-chroot-sw-img /cm/images/<software image>
...
[root@<software image> /]# yum/zypper/apt remove pbspro<old version>*
```

- Install the new packages in the headnode

```
$ yum/zypper/apt install pbspro<new version> pbspro<new version>-client
```

- Install the new packages in the software images

```
$ cm-chroot-sw-img /cm/images/<software image>
...
...
[root@<software image> /]# yum/zypper/apt install pbspro<new version>-client
```

- Change the version of the WLM in the `wlm` object

```
cmsh% wlm use pbspro
cmsh% set version <new version>
cmsh% commit
```

- Start `pbsserver`

```
cmsh% device
cmsh% foreach -l pbsproserver ( services; start pbsserver )
```

3.3.4 Upgrading The Kubernetes Package

Kubernetes deployments do not need removal before upgrading to BCM 11. The same version of Kubernetes runs on the BCM cluster after the upgrade.

An upgrade to Kubernetes (section 4.21 of the *Containerization Manual*) can be carried out after the BCM upgrade.

Kubernetes Package Upgrades For DGX Software Images

When upgrading DGX OS nodes from 6 to 7.2, a new software image (and `/cm/shared`) needs to be created (section 3.8.13), for the nodes which will be running Ubuntu 24.04.

Kubernetes on the new DGX image needs to be manually copied over from the old image. This can be done as follows:

- The `kubernetes.list` file should be copied over from the old image, <*old DGX image*>, to the new image, <*new DGX image*>, along with its associated keyring file:

  Example

```
[root@basecm11 ~]# cp /cm/images/<old DGX image>/etc/apt/sources.list.d/\
kubernetes.list /cm/images/<new DGX image>/etc/apt/sources.list.d/
[root@basecm11 ~]# cp /cm/images/<old DGX image>/etc/apt/keyrings/\
kubernetes-apt-keyring.gpg /cm/images/ <new DGX image>/etc/apt/keyrings/
```

- The packages for the existing version need to be placed in the new image (here it is called `new-dgx-image`):

```
[root@basecm11 ~]# cm-chroot-sw-img /cm/images/new-dgx-image
...
[root@new-dgx-image /]# apt-get install cm-kube-diagnose kubeadm kubectl \
kubelet kubernetes-cni cm-containerd nvidia-container-toolkit nginx \
libnginx-mod-stream
...
```

- The packages for the existing version need to be held. The following commands take care of that:

```
[root@new-dgx-image /]# apt-mark hold kubeadm
[root@new-dgx-image /]# apt-mark hold kubectl
[root@new-dgx-image /]# apt-mark hold kubelet
[root@new-dgx-image /]#   <ctrl-d>
```

The `<ctrl-d>` is to exit out of the `cm-chroot-sw-img` directory. The `cm-chroot-sw-img` command is described on page 534 of the *Administrator Manual*.

3.3.5 Upgrading The JupyterHub Database

The JupyterHub database schema must be manually upgraded in each JupyterHub node after the main package upgrade has completed. The following commands can be run to upgrade the JupyterHub database:

```
module load jupyter
jupyterhub upgrade-db--config=/cm/local/apps/jupyter/conf/jupyterhub_config.py \
  --db sqlite:////cm/local/apps/jupyter/run/jupyterhub.sqlite
```

3.3.6 Upgrading The Docker Registry And Harbor Packages

Harbor cannot be upgraded when BCM 10 is upgraded to BCM 11, its package can only be (re-)installed and (re-)deployed separately from the BCM upgrade.

If the Harbor packages were installed in BCM 10 to a node other than the head node, then a re-deployment to that node is required after the upgrade to BCM 11 for Harbor.

When upgrading a cluster from BCM 10 to BCM 11 the upgrade of Docker registry is supported.

3.4 Important: Package Upgrade Dependency Issues

The upgrade process will not only upgrade CMDaemon and its dependencies, but it will also upgrade other packages. This means that old packages will not be available from the new BCM repositories.

In some cases, this will require recompiling the user applications to use the upgraded versions of the compilers and the libraries. Also, the configurations of the old packages will not be copied automatically to the new packages, which means that the administrator will have to adjust the configuration from the old packages to suit the new packages manually.

Extra base distribution packages may be installed by `yum/zypper/apt-get` in order to resolve dependencies that might arise as a result of the upgrade. Hence the base distribution repositories must be reachable. This means that the clusters that run the Enterprise Linux distributions (RHEL and SLES) must be subscribed to the appropriate software channels.

Packages in `/cm/shared` are upgraded, but the administrator should be aware of the following:

- If `/cm/shared` is installed in the local partition, then the packages are upgraded. This may not be desirable for users that wish to retain the old behavior.
- If `/cm/shared` is mounted from a separate partition, then unmounting it will prevent upgrades to the mounted partition, but will allow new packages to be installed in `/cm/shared` within the local partition. This may be desirable for the administrator, who can later copy over updates from the local `/cm/shared` to the remote `/cm/shared` manually according to site-specific requirements. Since unmounting of mounted `/cm/shared` is carried out by default, a local `/cm/shared` will have files from any packages installed there upgraded. According to the yum database, the system is then upgraded even though the files are misplaced in the local partition. However, the newer packages can only be expected to work properly if their associated files are copied over from the local partition to the remote partition.

3.5 Known Issues

3.5.1 Kubernetes PVC Access Issue With The Shared Directory For Multidistro (MultiOS)

With a multidistro Kubernetes cluster, there can be a known issue with accessing `/cm/shared` across the various nodes.

Under What Conditions Is There An Issue?

The issue can occur when all of the following conditions apply:

1. The Kubernetes cluster (and worker nodes in particular) are spread across multiple DGX OS versions, due to one of the following situations:
   - (A) the nodes are running as multidistro nodes within the same Kubernetes cluster (running permanently as multidistro as is the intention, and not just transiently as part of an upgrade).
   - (B) the nodes are being upgraded, for example from Ubuntu 22 to Ubuntu 24
2. The *Local Path Provisioner* is provisioned and deployed in the Kubernetes cluster. This is a component running in Kubernetes that enables pods in every Kubernetes node to consume some kind of local storage. The `local-path-provisioner` parameter provides the local path `StorageClass`, which sets up a sub-directory under /cm/shared.
3. In this Kubernetes cluster, there is at least one pod using the local-path StorageClass using a PVC (PersistentVolumeClaim)

Why Is There An Issue?

The problem is that the `/cm/shared` directory is not unique across all nodes in a BCM cluster. The directory differs based on the DGX OS version. For example, on the head node, for the `/cm/shared` mount, you have:

- for Ubuntu 22 nodes:

  `/cm/shared-ubuntu2204-x86_64`

- for Ubuntu 24 nodes:

  `/cm/shared-ubuntu2404-x86_64`

During normal running, each DGX node only sees its specific `/cm/shared`.

Now, if a pod consuming a PVC from its local-path StorageClass is rescheduled to a node from the other DGX OS, then it cannot find the volume anymore, and it creates a new one.

Resolving The Issue For Persistent Mixed Distributions

A solution in the case of persistent mixed distributions on the nodes is to move all the pods to a new common shared directory for Kubernetes, independent of the directories offered for the specific DGX OS versions.

Steps:

1. If the cluster is not HA, then
   - a new `/cm/shared-kube` directory can be created
   - a new fsexport is created on the headnode:

     Example

     ```
     [head]% device use master; fsexports
     [head->device[head]->fsexports*]% add /cm/shared-kube@internalnet
     [head->devi...*[/cm/shared-kube@internalnet*]]% set path /cm/shared-kube
     [head->devi...*[/cm/shared-kube@internalnet*]]% set network internalnet
     [head->devi...*[/cm/shared-kube@internalnet*]]% set write yes
     [head->devi...*[/cm/shared-kube@internalnet*]]% commit
     [head->device[head]->fsexports[/cm/shared-kube@internalnet]]%
     ```

2. If the cluster is HA, then
   - a new directory or export can be created in the NAS to store the Kubernetes directory
   - if the headnode is a Kubernetes workers, or if the head node is a Kubernetes master that uses volumes using pods, then volumes will be used. Create an fsmount on the head nodes with:

     Example

     ```
     [head->device[head]->fsmounts]% add /cm/shared/apps/kubernetes
     [head->...red/apps/kubernetes*]]% set device 10.180.81.165:/brdkube
     [head->..*]->fsmounts*[/cm/shared/apps/kubernetes*]]% set filesystem nfs
     [head->device*[head*]->fsmounts*[/cm/shared/apps/kubernetes*]]% commit
     ```

3. Create an fsmount in the Kubernetes worker nodes categories (and/or nodes):

   Example

   ```
   [head->category[default]->fsmounts]% add /cm/shared/apps/kubernetes
   [head->...red/apps/kubernetes]]% set device $localnfsserver:/cm/shared-kube
   [head->...*]->fsmounts*[/cm/shared/apps/kubernetes*]]% set filesystem nfs
   [head->...*]->fsmounts*[/cm/shared/apps/kubernetes*]]% commit
   [head->category[default]->fsmounts[/cm/shared/apps/kubernetes]]%
   ```

```
```

4. Turn off all the Kubernetes worker nodes using `cmsh`

   This ensures that all pods are down, and not writing to the PVCs

5. On the head node, populate the whole directory with all the data from Kubernetes:

   Example

   ```
   rsync -ar /cm/shared-ubuntu2204-x86_64/apps/kubernetes/ \
   /cm/shared/apps/kubernetes/
   ```

6. If present, copy specific PVCs from Ubuntu 24.04 as well:

   Example

   ```
   rsync -ar /cm/shared-ubuntu2404-x86_64/apps/kubernetes/default/var/volumes/\
    /cm/shared/apps/kubernetes/var/volumes/
   ```

7. Turn on all the Kubernetes worker nodes
8. Ensure the Kubernetes worker nodes mount the new `/cm/shared/apps/kubernetes` directory

   Example

   ```
   mount | grep /cm/shared
   ```

9. Ensure all the pods are up and running again:

   Example

   ```
   kubectl get pods -A
   ```

10. Afterwards, clean up the old PVCs:

    Example

    ```
    rm -rf /cm/shared-ubuntu2204-x86_64/apps/kubernetes/default/var/volumes/
    rm -rf /cm/shared-ubuntu2404-x86_64/apps/kubernetes/default/var/volumes/
    ```

Resolving The Issue For The Upgrade Case Only

A solution in the case of an image upgrade, for example from DGX OS 6 to DGX OS 7, can be carried out as follows:

Before starting the upgrade of DGX OS version, you can move all PVCs from the old to the new /cm/shared.

We assume old nodes are running Ubuntu 22 and that they need to be upgraded to Ubuntu 24.

1. Turn off all the (Ubuntu 22.04) Kubernetes worker nodes via `cmsh`

   This ensures that all pods are down and not writing to the PVCs.

2. Upgrade DGX OS to Ubuntu 24.04

3. On the head node, copy all the PVCs to the other new /cm/shared

   Example

   ```
   rsync -ar  /cm/shared-ubuntu2204-x86_64/apps/kubernetes/default/var/volumes/ \
    /cm/shared-ubuntu2404-x86_64/apps/kubernetes/default/var/volumes/
   ```

4. Turn on all the Kubernetes worker nodes and check that all the pods are up and running again (`kubectl get pods -A`)

5. Afterwards, clean up the old PVCs:

   Example

   ```
   rm -rf /cm/shared-ubuntu2204-x86_64/apps/kubernetes/default/var/volumes/
   ```

Resolving The Issue If You Already Have Duplicated PVCs

A solution in the case of PVCs that have already been duplicated requires some finesse.

1. Detect all PVCs in the old and new `/cm/shared` and mapping between them. First look for current PVCs and ensure each one has the related directory in the new shared directory:

   - `kubectl get pvc -A`
   - `ls /cm/shared-ubuntu2404-x86_64/apps/kubernetes/default/var/volumes/`

2. Turn off the nodes

3. Backup or move the newly created PVCs, that most likely should be deleted:

   Example

   ```
   mv /cm/shared-ubuntu2404-x86_64/apps/kubernetes/default/var/volumes \
    /cm/shared-ubuntu2404-x86_64/apps/kubernetes/default/var/volumes-backup
   ```

4. Now, for each PVC, you have to find the original/correct one in the former shared directory. This is a manual process, and a bit of an art. You can consider:

   - The directory name which matches the Kubernetes namespace and PVC name
   - If there are 2 or more matching directories then:

- look for the latest modification date (`ls -lah`)
- check the overall size (`du -hs`). Usually the largest directory has more data and should be the most recent one

5. Run an `rsync` command per PVC:

Example

```
rsync -ar \
/cm/shared-ubuntu2204-x86_64/apps/kubernetes/default/var/volumes/\
pvc-98975be7-4030-4f11-8680-46a38829e2e7_runai-backend_data-runai-backend-\
redis-queue-master-0/ \
/cm/shared-ubuntu2404-x86_64/apps/kubernetes/default/var/volumes/\
pvc-98975be7-4030-4f11-8680-46a38829e2e7_runai-backend_data-runai-backend-\
redis-queue-master-0/
```

Note: the source and target directories are different (`...-ubuntu2204-...` to `...-ubuntu2404-...`). This is because the source should match what you have found out in the previous step, while the target should match what you listed initially.

6. Turn on nodes and verify pods are up and running

3.5.2 Miscellaneous Known Issues

- On clusters with Jupyter deployed, after the upgrade, users might find themselves being asked to login twice. The first time when the user will be asked to login to JupyterHub, and a second time, where the user is asked about a "CMDaemon login". In such cases, the user should once again present the credentials. This extra login request is harmless, and after the credentials are given a second time, it is unlikely that users will be queried again. The cause of this issue is that the Jupyterhub component that interacts with CMDaemon can lose connection to CMDaemon, causing it to prompt for an extra login.

- On some base distributions, depending on the version of the installed packages, when starting `cm-upgrade`, a YAMLLoadWarning message may be printed when its configuration files are loaded. The warning message can be safely ignored.

- In some cases because `/cm/shared` may be temporarily not accessible from the cloned primary head node when it boots for the first time, if workload managers are setup on the cluster, then CMDaemon may create and leave module files called `no-version`. For example:
`slurm/slurm/no-version`
To resolve the issue, the leftover `no-version` module files under `/cm/shared/modulefiles/` can be deleted.

- In some cases at the end of the head node or the software image(s) upgrade, the `cm-upgrade` script may report a list of packages that were not upgraded to their BCM 11 versions. This can occur in cases where the package manager was not able to resolve the list of packages to find a suitable candidate at the time of the upgrade of all packages.

  While `cm-upgrade` makes a best effort to swap or upgrade the packages to their BCM 11 versions, in some cases— notably on Ubuntu, it is also possible some packages are left in their 10.0 versions.

  The administrator is advised to manually swap the `cm10.0` packages with the `cm11.0` package.

- On clusters with cluster extension to AWS deployed, the cloud director and cloud nodes might fail to start after the upgrade. This can happen when there is an entry

  ```
  <blockdev mode="cloud">/dev/nvme1n1</blockdev>
  ```

  in the disksetup. The issue can be resolved by removing that entry from the disksetup for the cloud director and cloud nodes.

- On clusters that have PBSPro configured, and upgraded from BCM 10 to BCM 11, new jobs submitted by users will get job ids that were already used. This is particularly relevant for users that use BCM monitoring data for billing purposes.

- Disabling Kubernetes as part of the parallel-upgrade procedure results in an apparent failure during the stage when etcd data is cleaned up, because the compute nodes are not reachable from the parallel setup. This is expected since the parallel setup is isolated from the production setup. To resolve the issue, the administrator should answer a prompt so that the failed stage is skipped. It is also recommended to re-provision the compute nodes of the upgraded parallel setup before attempting to set up Kubernetes or etcd again, so that any leftover data from before the upgrade is cleaned during the provisioning.

- When upgrading the DGX OS 6 sofware image to the latest updates of DGX OS 6, there is a known issue with DKMS when a new kernel `linux-image-5.15.0-1090-nvidia` is installed. The DKMS post-install script fails for the new kernel. This is caused by duplicate entries for the `gdrdrv` module. This can be checked for with the following command:

  ```
  root@dgx-os-6:/# dkms status
  gdrdrv/2.4.4, 5.15.0-1063-nvidia, amd64: installed
  gdrdrv/2.4.4, 5.15.0-1090-nvidia, amd64: installed
  gdrdrv/2.4.4, 5.15.0-1090-nvidia, x86_64: built
  ```

  The last two statuses are for the same driver, and the last line is `built` rather than `installed`, which means it was not installed, and leads to the post-install script failure.

  This issue can be resolved by removing the duplicate entry and reconfiguring the package as follows:

  ```
  dkms remove gdrdrv/2.4.4 -a amd64 -k 5.15.0-1090-nvidia
  dpkg --configure -a
  ```

- When checking if the compute nodes are UP, in some cases `cm-upgrade` may wrongly report that some nodes are not DOWN. The report depends on how a node has been shut down, the state of the node not being updated correctly in the CMDaemon database. In such scenarios, after confirming that the nodes are indeed DOWN, re-run `cm-upgrade` with the `--skip-node-states-check` flag to skip the check for node states:

  Example

  ```
  cm-upgrade --skip-node-states-check
  ```

- Upgrading from BCM 10 to BCM 11 for DGX SuperPOD deployments requires filesystem exports to be explicitly set for the network `dgxnet` (page 64).

- The BCM machine learning repositories may still be referenced in the repository lists. They should be removed. For example, in the Ubuntu head node and images with:

```
rm /etc/apt/sources.list.d/cm-ml.list
```

3.6 Upgrading Using A BCM DVD/ISO

When using a BCM DVD/ISO to perform the upgrade, it is important to use a DVD/ISO that is not older than **11.30.0**. The DVD/ISO version can be found (assuming that the DVD/ISO is mounted under `/mnt/cdrom`) with a `find` command such as:

```
# find /mnt/cdrom -type d -name '11.30.0'
/mnt/cdrom/data/packages/11.30.0
```

3.7 Enable Upgrade

3.7.1 Enable The Upgrade Repo, And Install The Upgrade Package

Enabling the upgrade repository and installing the upgrade package can be carried out for the various distributions as follows:

RHEL8 derivatives:

```
# yum-config-manager \
  --add-repo http://support.brightcomputing.com/upgrade/11.0/rhel/8/updates
# yum --nogpgcheck install cm-upgrade
```

RHEL9 derivatives:

```
# yum-config-manager \
--add-repo http://support.brightcomputing.com/upgrade/11.0/rhel/9/updates
# yum --nogpgcheck install cm-upgrade
```

SLES 15:

```
# zypper addrepo \
http://support.brightcomputing.com/upgrade/11.0/sles/15/updates cm-upgrade-11
# zypper install cm-upgrade
```

Ubuntu 22.04:

```
# cat <<EOF > /etc/apt/sources.list.d/cm-upgrade-11.list
deb [trusted=yes] https://support.brightcomputing.com/upgrade/11.0/ubuntu/2204/ ./
EOF
# apt-get update
# apt-get install cm-upgrade
```

Ubuntu 24.04:

```
# cat <<EOF > /etc/apt/sources.list.d/cm-upgrade-11.list
deb [trusted=yes] https://support.brightcomputing.com/upgrade/11.0/ubuntu/2404/ ./
EOF
# apt-get update
```

```
# apt-get install cm-upgrade
```

3.7.2 Load The Environment Module

```
# module load cm-upgrade/11.0
```

3.8 Perform Upgrade

The steps for carrying out the upgrade are as follows:

3.8.1 Power Off Nodes

- Power off regular nodes
- Terminate cloud nodes and cloud directors, and remove the cloud extension(s)

3.8.2 Apply Updates To Head Node

- RHEL derivatives:

```
# yum update
```

- SLES derivatives:

```
# zypper up
```

- Ubuntu:

```
# apt-get update
# apt-get upgrade --with-new-pkgs
```

3.8.3 Apply Updates To Software Images

Before updating and upgrading a software image, it may be worth creating an image backup by cloning the image (page 211 of the *Administrator Manual*). If the cluster after the upgrade has an image that has an issue, then comparing it against the backup can help troubleshoot the issue.

For each software image, updates can be applied as follows:

- RHEL derivatives:

```
# yum --installroot /cm/images/<software image> update
```

- SLES derivatives:

```
# zypper --root /cm/images/<software image> up
```

- Ubuntu:

```
# cm-chroot-sw-img /cm/images/<software image>
# apt-get update
# apt-get upgrade --with-new-pkgs
# exit
```

3.8.4 Apply Updates To The `node-installer` Image

- RHEL derivatives:

```
# yum --installroot /cm/node-installer update
```

- SLES derivatives:

```
# zypper --root /cm/node-installer up
```

- Ubuntu:

```
# cm-chroot-sw-img /cm/node-installer
# apt-get update
# apt-get upgrade --with-new-pkgs
# exit
```

3.8.5 Upgrade Head Nodes To BCM 11

Important: this must be run on both head nodes in a high availability setup. Recommended: Upgrade the active head node first and then the passive head node.

Depending on availability, the upgrade can be run either over the network, or using a DVD/ISO:

- Upgrade using repositories accessible over the network

```
# cm-upgrade
```

- Upgrade using a BCM DVD/ISO

```
# cm-upgrade -b /root/bcm-11.0-ubuntu2404.iso
```

In an HA setup, after upgrading both the head nodes, resync the databases. Run the following from the active head node (it is very important to complete this step before moving to the next one):

```
# cmha dbreclone <secondary>
```

3.8.6 Reboot Head Node(s)

The head node(s) must be rebooted before proceeding to run the post-upgrade actions.

3.8.7 Post-upgrade Head Node Actions

Important: these must be run on both head nodes in an HA setup

```
# module load cm-upgrade/11

# cm-post-upgrade -m
```

3.8.8 Upgrade The Software Image(s) To BCM 11

Important: the upgrade to the software images must be done only on the active head node. Depending on availability, the upgrade can be run either over the network, or using a DVD/ISO:

- Upgrade using repositories accessible over the network

```
# cm-upgrade -i all
```

- Upgrade using a BCM DVD/ISO

```
# cm-upgrade -i all -b /root/bcm-11.0-ubuntu2404.iso
```

If the software images are not under the standard location—`/cm/images/` on the head node—then the option `-a` should be used:

```
# cm-upgrade -a /apps/images -i <name of software image>
# cm-upgrade -a /apps/images -i <name of software image> \
  -b /root/bcm-11.0-ubuntu2404.iso
```

If the software images are located under multiple locations, the target locations can be specified as a comma-separated list:

```
# cm-upgrade -a '/apps/images,/opt/images' -i all
```

3.8.9 Upgrade /cm/node-installer To BCM 11

This needs to be performed only on the primary active head node, either over the network, or using a BCM DVD/ISO.

- Upgrade using repositories accessible over the network:

```
# cm-upgrade -x
```

- Upgrade using a BCM DVD/ISO:

```
# cm-upgrade -x -b /root/bcm-11.0-ubuntu2404.iso
```

3.8.10 Post-upgrade Software Images Actions

Important: these must be run only on the active head node.

```
# cm-post-upgrade -i all
```

If the software images are not under the standard location, which is `/cm/images/` on the head node, then the option `-a` should be used:

```
# cm-post-upgrade -a /apps/images -i all
# cm-post-upgrade -a /apps/images -i <name of software image>
```

If the software images are located under multiple locations, then the target locations can be specified as a comma-separated list:

```
# cm-post-upgrade -a '/apps/images,/opt/images' -i all
```

3.8.11 NVIDIA Driver Installation

During the upgrade to BCM 11.0, `cm-upgrade` automatically installs NVIDIA GPU drivers on non-DGX software images. This is only carried out on software images. It is not carried out on the head nodes or the node-installer images.

Default NVIDIA Driver Version

By default, `cm-upgrade` installs `NVIDIA driver version 570 (Open Kernel Modules edition)`. The following components are installed:

- On Ubuntu 22.04 and 24.04: the package installed is the `nvidia-driver-570-open` meta-package plus additional components including:
  - Fabric Manager (for NVLink/NVSwitch management)
  - NSCQ library (NVIDIA NVSwitch Configuration and Query Library)
  - IMEX library (GPU memory sharing across different compute nodes in a multi-node system connected by NVLink)
  - NVSDM library (for monitoring NVSwitch devices on NVIDIA Blackwell systems)
- On RHEL 9/Rocky 9: a comprehensive set of packages based on the major 570 version. The package set installed includes:
  - DKMS and open kernel modules for dynamic kernel support
  - Core driver libraries and CUDA components
  - Fabric Manager for multi-GPU interconnect management
  - GUI tools (nvidia-settings, nvidia-xconfig) for X11 display configuration

  Most package names are generic (for example: nvidia-driver, nvidia-fabric-manager), with the exact version version determined by the repository configuration
- On SLES 15 SP6: the package installed is a set based on the major 570 version, and includes:
  - Open kernel modules (`nvidia-open-570`)
  - Fabric Manager and NSCQ library

  Packages are sourced from NVIDIA's CUDA repository (version-numbered). Native SLES packages use G06 generation naming, but we use NVIDIA's CUDA repository for consistency

Non-DGX Systems Only

NVIDIA driver installation is only carried out on non-DGX systems.

DGX systems have pre-installed NVIDIA drivers, which are maintained by NVIDIA's own update mechanisms, and should not managed by `cm-upgrade`. The upgrade script detects DGX systems by checking for the presence of `/etc/dgx-release` in the software image. If this file exists, then NVIDIA driver installation in the DGX system is skipped.

Changing the NVIDIA Driver Version

If you need to install a different version of the NVIDIA drivers, then you can modify the driver version either before or after the upgrade as follows:

- Pre-upgrade: Edit /cm/local/apps/cm-upgrade/11.0/conf/packages.yaml and modify the package versions listed under the `nvidia_driver_packages` section for your distribution.
- Post-upgrade: Use the standard package management tools (`yum`/`apt`/`zypper`) within the software image to remove the installed driver packages and install your preferred version.

3.8.12 (HA) Allow For The Provisioning Requests To The Secondary Head Node To Complete

Allow between 15 and 30 seconds for CMDaemon to schedule and start provisioning updates of `/tftpboot` and `/cm/node-installer` from the primary to the secondary head node. Then check and wait until the provisioning requests with the secondary head node as destination have completed using:

```
# cmsh -c 'softwareimage provisioningstatus -r'
```

If the provisioning requests with a destination node the secondary head node continue to be listed for a prolonged period of time and do not disappear from the list, then the `node-installer` and/or `tftpboot` on the secondary head node are out of sync from the primary head node, which can result in issues with provisioning of the compute nodes. Please contact support (`https://enterprise-support.nvidia.com/s/create-case`) for further assistance.

3.8.13 Enable Mixed OS Setup And Create A DGX OS 7 Image

On BCM 10 Ubuntu 22.04 clusters can use DGX OS 6 images. DGX OS 6 images are not supported in BCM 11.

A new DGX OS 7 software image must therefore be used. The image can be created using `cm-image` and `cm-create-image`. These are BCM utilities, and are described in section 9.6 of the *Administrator Manual*.

DGX OS 7 is based on Ubuntu 24.04. This then also means the mixed OS environment—multidistro or multiOS as described in section 9.7 of the *Administrator Manual*—must be enabled, to allow provisioning of DGX OS 7 images from the Ubuntu 22.04 head node of BCM 10.

Image creation can be carried out as follows:

1. A script may need to be created first:

   Example

   ```
   [root@basecm11 ~]# cat disable-upstream-repos.sh
   #!/bin/bash
   if [ -f "$1/etc/apt/sources.list.d/ubuntu.sources" ]; then
           mv $1/etc/apt/sources.list.d/ubuntu.sources \
           $1/etc/apt/sources.list.d/ubuntu.sources.dvd_bak
   fi
   ```

   The script is only needed for a cluster running a BCM version 11 earlier than 11.31.0. It works around a known issue that may arise from using an upstream repository in an inappropriate manner. For BCM 11.31.0 and later, the workaround is carried out automatically, and the script is not needed.

2. Use `cm-image` to create the appropriate the Ubuntu 24.04 base images and enable a mixed OS setup. The `-z` option that calls the script need only be used for BCM 11 installations earlier than BCM 11.31.0.

   Example

```
[root@basecm11 ~]# module load cm-image
[root@basecm11 ~]# cm-image --v create all -a x86_64 -d ubuntu2404 --source \
                /path/to/bcm11.0-ubuntu24.04-dgx-os-7.2.iso \
                -z disable-upstream-repos.sh
```

3. Use `cm-create-image` to create the new DGX OS image. The `-z` option that calls the script need only be used for BCM 11 installations earlier than BCM 11.31.0.

```
[root@basecm11 ~]# cm-create-image --cmdvd \
                /root/bcm-11.0-ubuntu2404-dgx-os-7.2.iso \
                --no-cm-cuda-repo --extra-pkg-group \
                doca_ofed_2.9.1-3.0.2 --dgx --dgx-type dgx_b200 \
                --imagename dgx-os-image-b200 \
                -z disable-upstream-repos.sh
```

The `--dgx-type` option (page 546 of the *Administrator Manual*) must be set for every type that is to be used as an image. The possible values are:

- `dgx_h100` for the H100 GPU
- `dgx_b200` for the B200 GPU
- `dgx_a100` for the A100 GPU
- `dgx_gb200` for the GB200 GPU
- `dgx_h200` for the H200 GPU

The cluster administrator should create a new category for each new image. Cloning the `default-image` image from the `default` category is a convenient way to do this since it already has the correct fsmounts. The image name should then be set for the category.

For example, for a B200 image `dbx-os-image-b200` that has been placed in /cm-images/:

Example

```
root@basecm11:~# cmsh
[basecm11]% category
[basecm11->category]% list
Name (key)              Software image          Nodes
----------------------- ----------------------- --------
default                 default-image           1
[basecm11->category]% clone default gb200
[basecm11->category*[gb200*]]% use gb200
[basecm11->category*[gb200*]]% set softwareimage dgx-os-image-b200 ; commit
...
```

The nodes that are to use the new image should then be assigned their image. For example, a `node001` can be assigned to the `gb200` category created in the preceding example with:

Example

```
[basecm11->device]% set node001 category gb200 ; commit
```

Configuration overlays that were used in the old category should also be migrated as needed to the new category.

Upgrading from BCM 10 to BCM 11 for DGX SuperPOD deployments requires NFS exports to be explicitly set for the network `dgxnet`, for `/cm/node-installer-ubuntu2404-x86_64` and `/cm/node-installer-ubuntu2404-x86_64/certificates`.

This can be configured with the following `cmsh` session on the head node:

Example

```
[root@basecm11 ~]# cmsh
[basecm11]% device use master
[basecm11->device[basecm11]]% fsexports
[basecm11->...->fsexports]% add /cm/node-installer-ubuntu2404-x86_64@dgxnet
[basecm11->...installer-ubuntu2404-x86_64@dgxnet*]]% set network dgxnet
[basecm11->...installer-ubuntu2404-x86_64@dgxnet*]]% commit
[basecm11->...installer-ubuntu2404-x86_64@dgxnet]]% ..
[basecm11->...rts]% add /cm/node-installer-ubuntu2404-x86_64/certificates@dgxnet
[basecm11->...-ubuntu2404-x86_64/certificates@dgxnet*]]% set network dgxnet
[basecm11->...-ubuntu2404-x86_64/certificates@dgxnet*]]% set write yes
[basecm11->...-ubuntu2404-x86_64/certificates@dgxnet*]]% commit
[basecm11->...-ubuntu2404-x86_64/certificates@dgxnet]]%
```

The same procedure must also be carried out for that secondary head node.

The new DGX OS image has no Kubernetes packages installed on it by default. Migrating an old Kubernetes installation over from the old image is described on page 49.

3.8.14 Upgrading CMDaemon Lite

The BCM 10 to BCM 11 upgrade supports CMDaemon Lite (section 2.6.7 of the *Administrator Manual*) running on cluster switches, such as Cumulus switches.

After the upgrade has been carried out on the head node and the images are ready for deployment, then the switches can be rebooted to pick up their new image.

- It is important not to reboot the switch that is being used to connect to the head node.

After the switches have rebooted, the compute nodes can be powered on.

In a cluster, mixing deployments of cm-lite-daemon for BCM 10 with those of BCM 11 is not supported.

This means that the CMDaemon Lite servers that are not managed by CMDaemon, for example PCs running CMDaemon Lite in a Python environment, also need to be upgraded when upgrading the cluster itself from BCM 10 to BCM 11. That must be done manually, by removing the BCM 10 package, and installing the BCM 11 package according to the environment and OS it is running in.

3.8.15 The Cluster Is Now Upgraded To BCM 11

It is recommended to provision the compute nodes at least once with BCM 11 before setting up any new integrations.

3.9 Simple Validation Checks

Some basic checks can be carried out to validate the cluster. If a check fails, then further investigation should be carried out.

3.9.1 Validating CMDaemon

The CMDaemon service status (section 2.6.1 of the *Administrator Manual*), can be checked with:

Example

```
root@basecm11:~# systemctl status cmd
  cmd.service - BCM daemon
     Loaded: loaded (/usr/lib/systemd/system/cmd.service; enabled; preset: enabled)
     Active: active (running) since Mon 2025-12-01 01:44:17 PST; 6h ago
   Main PID: 2547 (safe_cmd)
      Tasks: 77 (limit: 65535)
     Memory: 576.1M (high: 3.4G max: 3.6G available: 1.3G peak: 600.1M)
        CPU: 6min 56.963s
     CGroup: /system.slice/cmd.service
...
```

3.9.2 Validating High Availability

If the cluster has HA configured, then the cluster manager HA status (page 782 of the *Administrator Manual*), can be checked. For example, for head nodes `basecm11-a` and `basecm11-b` the response to the check would be:

Example

```
root@basecm11-a:~# cmha status
Node Status: running in active mode

basecm11-a* -> basecm11-b
  failoverping  [  OK  ]
  mysql         [  OK  ]
  ping          [  OK  ]
  status        [  OK  ]

basecm11-b -> basecm11-a*
  failoverping  [  OK  ]
  mysql         [  OK  ]
  ping          [  OK  ]
  status        [  OK  ]
```

3.9.3 Validating `cmsh` And Devices

A check on the device statuses can be run with:

Example

```
root@basecm11:~# cmsh -c "ds"
basecm11.............[   UP   ]
node001 .............[   UP   ]
...
```

3.9.4 Validating The Cluster Health

Health check values can be retrieved from devices in the BCM monitoring system with the help of the `latesthealtdata` command (section 10.6.3 of the *Administrator Manual*):

Example

```
[basecm11->device]% foreach -v * (latesthealthdata)
================================ basecm11 ====================================
```

```
Measurable          Parameter    Type          Value      Age         State      Info
------------------- ------------ ------------ ---------- ---------- ---------- ----------
ManagedServicesOk                Internal     PASS       1m 39s
chrootprocess                    OS           PASS       1.23s
cm-chroot-sw-img                 OS           PASS       1m 18s
cmsh                             Internal     PASS       25m 7s
cuda-dcgm                        OS           PASS       54.8s
defaultgateway                   Network      PASS       1m 8s
diskspace                        Disk         PASS       14s
dmesg                            OS           PASS       1m 51s
exports                          Disk         PASS       9.21s
interfaces                       Network      PASS       1m 7s
ldap                             OS           PASS       1m 37s
lustre                           Disk         PASS       58.8s
mounts                           Disk         PASS       1m 40s
mysql                            OS           PASS       53m 47s
ntp                              Internal     PASS       1m 49s
oomkiller                        OS           PASS       18s
ssh2node                         network      PASS       8m 58s
================================ node001 ====================================
Measurable          Parameter    Type          Value      Age         State      Info
------------------- ------------ ------------ ---------- ---------- ---------- ----------
ManagedServicesOk                Internal     PASS       2m 54s
cuda-dcgm                        OS           PASS       2m 9s
defaultgateway                   Network      PASS       2m 13s
diskspace                        Disk         PASS       3m 28s
dmesg                            OS           PASS       3m 5s
interfaces                       Network      PASS       2m 22s
ldap                             OS           PASS       2m 51s
lustre                           Disk         PASS       2m 13s
mounts                           Disk         PASS       2m 55s
ntp                              Internal     PASS       3m 3s
oomkiller                        OS           PASS       1m 32s
ssh2node                         network      PASS       8m 43s
...
```

3.9.5 Validating Base View

For a dotted quad IP address <*IP address*>, the cluster administrator can connect to:

`https://`<*IP address*>`:8081/base-view`

and try logging in. If there was non-administrator access before the upgrade, then that should also be checked.

3.9.6 Validating The State Of The Packages On The Cluster

The state of packages according to the package manager for the cluster can be checked in the head nodes and in the software images with:

Ubuntu Packages

```
apt-get check
dpkg --audit
```

RHEL-like Distribution Packages

```
yum check
rpm -Va --nofiles
```

SLES Packages

```
zypper verify
rpm -Va --nofiles
```

These commands should all normally give no output, other than processing messages in some cases.

3.9.7 Validating Slurm And Pyxis

Slurm and Pyxis validation on the cluster (page 356 of the *Administrator Manual*), can be carried out as follows:

Validating Slurm

```
root@basecm11:~# module load slurm
root@basecm11:~# scontrol show config
Configuration data as of 2025-12-02T04:06:05
AccountingStorageBackupHost = (null)
AccountingStorageEnforce = none
AccountingStorageHost   = master
AccountingStorageExternalHost = (null)
AccountingStorageParameters = (null)
AccountingStoragePort   = 6819
...
Slurmctld(primary) at basecm11 is UP
```

Validation Of Pyxis

```
root@basecm11:~# srun --container-image=ubuntu grep PRETTY /etc/os-release
pyxis: importing docker image: ubuntu
pyxis: imported docker image: ubuntu
PRETTY_NAME="Ubuntu 24.04.3 LTS"
root@basecm11:~#
```

3.9.8 Validating Kubernetes

The `kube-system` pods should be inspected to see if they are running along without any real issues:

Example

```
root@basecm11:~# kubectl get pods -n kube-system
NAME                                  READY   STATUS    RESTARTS   AGE
coredns-66bc5c9577-bv577              1/1     Running   0          4h24m
coredns-66bc5c9577-rsfpv              1/1     Running   0          4h24m
kube-apiserver-basecm11               1/1     Running   0          4h19m
kube-apiserver-node001                1/1     Running   2          4h24m
kube-controller-manager-basecm11      1/1     Running   0          4h23m
kube-controller-manager-node001       1/1     Running   2          4h25m
kube-proxy-2ckqr                      1/1     Running   0          4h24m
...
kube-scheduler-node001                1/1     Running   1          4h25m
kube-state-metrics-6b98df4478-nhkd2   1/1     Running   0          4h19m
metrics-server-f49fdc486-4ndfk        1/1     Running   0          4h19m
metrics-server-f49fdc486-5rvbf        1/1     Running   0          4h19m
```

Are the operators running along fine too?

```
root@basecm11:~# kubectl get pods --all-namespaces
NAMESPACE      NAME                                       READY   STATUS    RESTARTS   AGE
calico-system  calico-apiserver-577cdf67b5-645qn          1/1     Running   0          4h49m
```

```
...
calico-system  calico-typha-6495c7469f-gpztc              1/1     Running   0          4h49m
calico-system  calico-typha-6495c7469f-rp9rz              1/1     Running   0          4h49m
calico-system  goldmane-7b6b78f6f-tn65t                  1/1     Running   0          4h49m
calico-system  whisker-6c95b488ff-7s9pf                  2/2     Running   0          4h49m
cm             local-path-provisioner-675dc9fd86-b6fg6   1/1     Running   0          4h48m
cmkpm-system   cmkpm-controller-manager-66b99b6699-6k6gm  1/1     Running   0          4h48m
grafana        grafana-grafana-operator-c65d95b7f-c5p59  1/1     Running   0          4h47m
ingress-nginx  ingress-nginx-controller-68994db546-7c2h9  1/1     Running   0          4h44m
ingress-nginx  ingress-nginx-controller-68994db546-p6pjv  1/1     Running   0          4h44m
kube-system    coredns-66bc5c9577-bv577                  1/1     Running   0          4h51m
...
```

3.10 In-place Upgrade Of An Edge Director Or A Compute Node

The recommended way of upgrading the compute nodes is to upgrade the head node and software image(s) as described in section 3.8, and to allow the node-installer (when the compute nodes are booted) to re-provision the compute nodes with the upgraded software. For clusters with an edge setup, this usually means a new edge ISO needs to be generated on the head node, transferred to the edge site, and then the edge director node needs to be reconfigured to boot from the ISO, which will start the node-installer.

In some special cases, however, the administrator may elect to manually carry out an in-place upgrade of a compute node or edge director, instead of using the node installer to automatically upgrade the node. With the manual in-place upgrade, the administrator sets up the node to receive the updates directly from the software image (through `imageupdate`) without the involvement of the node installer.

Other than the cases where the node installer does not start on (re)boot by default, such as for the edge director, the in-place upgrade of a compute node/edge director does not offer any advantage over the standard approach to simply upgrade the head node and the software image(s).

3.10.1 Important Note About Services Running On The In-place Upgraded Compute Node/Edge Director

An in-place upgrade of a compute node will generally restart only the CMDaemon service. Any other service that may be affected by the upgrade of the packages, especially in the case of packages in `/cm/shared` (such as WLM), which are upgraded as part of the head node(s) upgrade, will need to be restarted manually by the administrator, or a reboot will be required.

It is therefore recommended to use the standard approach to upgrade the software image(s) on the head node, boot the nodes with the node installer, and allow the node installer to upgrade and configure the compute nodes as usual.

Nevertheless the administrator may find it more convenient in special cases such as edge sites, which have edge directors, to perform an in-place upgrade. The edge directors nodes are typically configured to boot from the local disk and by default do not start the node installer on (re)boot. An in-place upgrade of the edge directors will then allow the administrator to upgrade the software on the directors without transferring ISOs and reconfiguring the directors to boot from ISOs. To address the issue with the running services, the edge directors can be rebooted at the end of the upgrade and while they will still not start the node installer, they will boot into the upgraded software.

3.10.2 In-place Edge Director Upgrade Procedure

Unless otherwise stated, all actions are performed locally on the edge director

Apply Updates To The Edge Director

- RHEL derivatives:

```
# yum update
```

- SLES derivatives:

```
# zypper up
```

- Ubuntu:

```
# apt-get update
# apt-get upgrade --with-new-pkgs
```

Back Up The File System

It is recommended to perform a file system backup of the edge director

Stop CMDaemon and `cm-nfs-checker` services

```
# systemctl stop cmd
# systemctl stop cm-nfs-checker
```

Upgrade The Head Node(s) As Described In The Preceding

Follow the steps to upgrade the head nodes, software image, and node installer as described in the preceding sections "Enable Upgrade" (section 3.7) and "Perform Upgrade" (section 3.8), and which are performed on the head node(s) of the cluster. Because CMDaemon on the edge director is stopped, the edge director does not need to be powered off while the upgrade of the head nodes takes place.

Upgrade CMDaemon To BCM 11

After the upgrade of the head nodes is completed, the package manager repositories files on the edge director must be manually updated to point to the BCM 11 repositories. This means that all occurrences of 10.0 in the repositories are replaced with with 11.0. Note: Leave the usernames such as in, for example, "`username=cm10user`", unchanged.

Then, upgrade CMDaemon as follows for the various distributions:

- RHEL derivatives:

  Update the `/etc/yum.repos.d/cm.repo` file to point to the 11.0 repositories.

  On RHEL, typically `yum`/`dnf` can manage to resolve most of the dependencies, and the administrator can execute:

```
# yum clean all
# rpm -e --nodeps cm-config-cm
# yum install cmdaemon
```

- SLES derivatives:

  Update the `/etc/zypp/repos.d/Cluster_Manager_Updates.repo` file to point to the 11.0 repositories.

  Execute

```
# zypper clean --all
# zypper install cmdaemon
```

A problem may be reported by zypper with upgrading cmdaemon to version 11.0, due to a requirement to install cm-config-cm version 11.0. Typically the first solution, Solution 1, offered by zypper can be selected, which is to deinstall a certain set of BCM packages (prefixed by cm9.2) in order to upgrade CMDaemon to version 11.0.

- Ubuntu:

  Update the /etc/apt/sources.list.d/cm.list file to point to the 11.0 repositories.

  Also update the /etc/apt/auth.conf.d/cm.conf file, to point to the 11.0 repositories.

  Create a new file /etc/apt/preferences.d/99-tmp-cmd-upgrade with content (without any space at the beginning of the lines):

```
Package: *
Pin: origin "updates.brightcomputing.com"
Pin-Priority: 1001
```

  The administrator may also find it convenient to choose a geographically-close mirror in the cm.list file. But note that the local change will be overwritten to the settings in the software image, once imageupdate is run in a step later on.

  Execute

```
# apt-get update
# apt-get install cmdaemon
```

Start CMDaemon

After CMDaemon has been upgraded to BCM 11, execute:

```
# systemctl daemon-reload
# systemctl start cm-nfs-checker
# systemctl start cmd
```

Update The Edge Director Image With imageupdate

On the head node, with cmsh, verify that the edge director is UP. If the edge director is not UP, then the in-place upgrade has not been successful, and the director will need to be upgraded by booting into to the node-installer.

With cmsh on the head node, execute imageupdate for the edge director. The administrator is advised to execute a dry-run first, then to use use the synclog command to review the result and verify no unexpected local data directories or files will be affected, and then to perform the real imageupdate:

```
cmsh% device use edge-director
cmsh% imageupdate
cmsh% synclog
cmsh% imageupdate -w
```

Update The Provisioners And Recreate The Ramdisks

Execute on the head node

```
cmsh% softwareimage updateprovisioners;
cmsh% fspart; trigger /tftpboot; trigger /cm/node-installer; trigger /cm/shared
```

Allow for the provisioning requests with a destination node the in-place upgraded director to complete by monitoring the provisioning status:

```
cmsh% softwareimage provisioningstatus -r
```

For the images served by the edge director, recreate the ramdisk

```
cmsh% softwareimage; createramdisk -d <image-name>
```

If needed, the in-place upgraded edge director can now also be rebooted. It is then upgraded and can provision the edge compute nodes.