

1.5.5

Multi-Instance GPU

Date published: 2026-09-24

Date modified:

CLOUDERA

Legal Notice

© Cloudera Inc. 2026. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

About Multi-Instance GPU (MIG).....	4
Prerequisites for using Multi-Instance GPU (MIG).....	4
Using Multi-Instance GPU (MIG).....	5
User Management.....	5
Selecting the Multi-Instance GPU (MIG) cluster strategy.....	6
Understanding the Nodes panel.....	6
Configuring the GPU Modals.....	8
Saving the Multi-Instance GPU (MIG) configuration.....	12
Switching cluster strategies.....	16
API support for the Multi-Instance GPU (MIG).....	19
Setting up Single or Multi-Rail cluster.....	23
Setting up Host node.....	23
Setting up single or multi rail for homogenous HGX cluster.....	24
Troubleshooting GPU and Network operators.....	33

About Multi-Instance GPU (MIG)

The partitioning option for NVIDIA Multi-Instance GPUs is now supported. For better utilization, you can partition your GPUs into smaller slices.

By default, a workload would use the entire GPU, but if the GPU is partitioned (or sliced) to smaller pieces, then the workload would just use a smaller portion of the GPU and multiple workloads could be running on the same GPU.



Note: This feature is only supported on Cloudera Embedded Container Service clusters. OpenShift Container Platform is not supported.

Prerequisites for using Multi-Instance GPU (MIG)

Review the prerequisites for using the Multi-Instance GPU on your Cloudera Embedded Container Service clusters.

The Cloudera Embedded Container Service users must:

- Install the NVIDIA driver on every MIG-capable node. Use a version verified by NVIDIA and listed in the [GPU Operator Component Matrix](#) under **GPU operator v26.3.2**, which cdp-private Cloudera Data Services on premises 1.5.5-SP4 installs:
 - [595.71.05](#)
 - [595.58.03](#)
 - [590.48.01](#)
 - [580.173.02](#) (R)
 - [580.167.08](#)
 - [580.159.04](#)
 - [580.159.03](#)
 - [580.126.20](#) (D)
 - [570.211.01](#)
 - [535.309.01](#)
 - [535.288.01](#)
- Add the MIG capable nodes to the cluster.



Important: If added later, then these MIG capable nodes will be discovered automatically by the MIG feature.

Prerequisites to review for the Cloudera Embedded Container Service clusters



Caution: Users must not make MIG-capable nodes into Cloudera Embedded Container Service server/master nodes. Doing so applies a taint that prevents NVIDIA-related pods from starting on those nodes. You must retain the MIG-capable nodes as Cloudera Embedded Container Service agent/worker nodes.

Before upgrading to Cloudera Data Services on premises 1.5.5 SP4 on Cloudera Embedded Container Service, you must:

- [uninstall the NVIDIA GPU Operator](#)
- [uninstall the NVIDIA NicClusterPolicy and Network Operator](#)

Additionally, **on every MIG capable node** clean up the following directories:

```
rm -rf /etc/cdi/*
rm -f /etc/nvidia-container-runtime/config.toml
rm -rf /var/run/cdi/*
```

```
rm -rf /usr/local/nvidia/toolkit
```

The cdp-private installer installs the GPU operator to Helm release gpu-operator and Kubernetes namespace gpu-operator.



Important: MIG is not supported on OpenShift Container Platform.

Using Multi-Instance GPU (MIG)

Review the multiple operational phases of the MIG feature.

In the Cloudera Management Console instance, under the Administration page, a new MIG Configuration tab is available.

Cluster Strategy
Choose how Multi-Instance GPU (MIG) configuration profiles are distributed across cluster nodes to optimize workload allocation, partition hardware resources, and maintain consistent node behavior across your environment.

☒ **Mixed** Current
Allow different MIG profiles on different GPUs.

☐ **Single**
Apply a single MIG profile uniformly across all GPUs.

☐ **None**
Disable MIG on all nodes in the cluster.

Nodes

Name	Status	Overview
ecs-sxm-0631.ecs.mtc.comops.rock	Success	None of 8 GPUs are configured.
ecs-sxm-0633.ecs.mtc.comops.rock	Success	All 8 GPUs are configured.

User Management

Review the Multi-Instance GPU (MIG) user management roles.

In the Cloudera Management Console instance, under the User Management page, a couple of roles are introduced:

☐	MigConfigManager	Grants permission to view and apply NVIDIA MIG configuration for MIG-capable nodes.	Policies
☐	MigConfigViewer	Grants permission to view NVIDIA MIG configuration for MIG-capable nodes.	Policies

- MigConfigManager
- MigConfigViewer

Note the following information related to the MIG user management roles:

- If a user does not have any of these two roles, then the MIG Configuration tab does not show up on the Administration page.
- If a user has been assigned the MigConfigViewer role:
 - It covers the READ operations.
 - Users with this role can open the MIG Configuration tab, and inspect the content, but cannot modify it.
 - By default, EnvironmentUser and DataSteward users have this role assigned.
- If a user has been assigned the MigConfigManager role:
 - It includes everything from MigConfigViewer (READ operations) and includes the SAVE operations.

- Users can open the MIG Configuration tab, inspect the content, and also modify and save it.
- By default, PowerUser and EnvironmentAdmin users have this role assigned.

Selecting the Multi-Instance GPU (MIG) cluster strategy

You can select the MIG cluster strategy based on your requirements.

Before you begin

Make sure you have the required roles to make these changes.

On the MIG Configuration tab, MIG supports three types of strategies.

Cluster Strategy

Choose how Multi-Instance GPU (MIG) configuration profiles are distributed across cluster nodes to optimize workload allocation, partition hardware resources, and maintain consistent node behavior across your environment.

☒ Mixed Current
 Allow different MIG profiles on different GPUs.

☐ Single
 Apply a single MIG profile uniformly across all GPUs.

☐ None
 Disable MIG on all nodes in the cluster.

- **Mixed** - Indicates that every GPU can be sliced differently within a node. Users are allowed to set up the nodes in a heterogeneous manner.
- **Single** - Indicates that GPUs can be sliced, but on a node every GPU must be sliced uniformly. For example, if a GPU has 7 x 1g.10gb slices set, then all the other GPUs in the node must have these exact slices. Different nodes can use different slicing.
- **None** - Indicates that MIG is disabled on the whole cluster. Users will not be able to slice up the GPUs with this strategy. It does not mean that workloads cannot run on these GPUs, but these would use the entire GPU.



Note: Users can switch between strategies, any time.

Understanding the Nodes panel

Within the Cluster Strategy option, you can locate the Nodes panel. This panel lists all the MIG capable nodes.

Under the Cluster Strategy option, you can locate the Nodes panel. This panel lists all the MIG capable nodes.

Each node lists its available GPUs, and for each GPU the currently associated slices.

As an example, the following illustration contains a cluster that has two MIG capable nodes:

- ecs-sxm-0631.ecs.mc.comops.rock
- ecs-sxm-0633.ecs.mc.comops.rock

Nodes

 Search

	Name	Status	Overview
▼	ecs-sxm-0631.ecs.mc.comops.rock	✔ Success	None of 8 GPUs are configured.
▼	ecs-sxm-0633.ecs.mc.comops.rock	✔ Success	All 8 GPUs are configured.

< 1 > 10 / page ▼

These two nodes are identical, have the exact same GPUs: 8 x NVIDIA H100 80GB HBM3.

ecs-sxm-0631.ecs.mc.comops.rocks Success None of 8 GPUs are configured.

GPUs [Configure On Every GPU](#)

#	Model	Slices
0	NVIDIA H100 80GB HBM3	No slices configured. Configure
1	NVIDIA H100 80GB HBM3	No slices configured. Configure
2	NVIDIA H100 80GB HBM3	No slices configured. Configure
3	NVIDIA H100 80GB HBM3	No slices configured. Configure
4	NVIDIA H100 80GB HBM3	No slices configured. Configure
5	NVIDIA H100 80GB HBM3	No slices configured. Configure
6	NVIDIA H100 80GB HBM3	No slices configured. Configure
7	NVIDIA H100 80GB HBM3	No slices configured. Configure

ecs-sxm-0633.ecs.mc.comops.rocks Success All 8 GPUs are configured.

GPUs [Configure On Every GPU](#)

#	Model	Slices
0	NVIDIA H100 80GB HBM3	1 * 1g.10gb+me 2 * 1g.20gb Configure
1	NVIDIA H100 80GB HBM3	1 * 3g.40gb 2 * 1g.20gb Configure
2	NVIDIA H100 80GB HBM3	1 * 3g.40gb 2 * 1g.20gb Configure
3	NVIDIA H100 80GB HBM3	1 * 3g.40gb 2 * 1g.20gb Configure
4	NVIDIA H100 80GB HBM3	1 * 3g.40gb 2 * 1g.20gb Configure
5	NVIDIA H100 80GB HBM3	1 * 3g.40gb 2 * 1g.20gb Configure
6	NVIDIA H100 80GB HBM3	1 * 3g.40gb 2 * 1g.20gb Configure
7	NVIDIA H100 80GB HBM3	1 * 3g.40gb 2 * 1g.20gb Configure

Additional information about these slices and how these should be read:

- NVIDIA H100 80GB HBM3 has capacity: **80gb memory and 7 compute units.**
- Slices like: 1 * 3g.40gb + 2 * 1g.20gb implies that we split that GPU to 3 slices overall:
 - 1 * 3g.40gb: This single slice would use 40GB memory and 3 compute units
 - 2 * 1g.20gb: This yields 2 slices of the same size, each would have 20GB memory and 1 compute unit
 - This uses up the whole GPU memory: 40gb + 2 * 20gb = 80gb
 - However, not all the compute units are used: 3 + 2 * 1 = 5. This GPU has 7 compute units, so 2 would be left unused.
 - Because these slices exhaust the GPU capacity, there is no option to add more slices to this GPU.
- There is a special slice, that is visible on the screenshot for ecs-sxm-0633.ecs.mc.comops.rocks GPU 0: 1g.10gb+me. This is a special slice and only 1 could be used on a GPU, you cannot have 2 slices of the same. The +me means to include media engines such as: NVDEC, JPEG, OFA.

Configuring the GPU Modals

Select slice profiles per GPU or per Multi-Instance GPU (MIG) node.

Using the Configure option

Using any Configure button, you can specify the slices for a specific GPU. Using the modal window that pops up, you can configure specific slices.

Configure GPU 3

×

Set the MIG layout for GPU 3. Saving replaces all slices currently configured on this GPU. Remove every slice from the layout preview to disable MIG on this GPU.

Instance Count		Slice Profile	
1	*	Select	+ Add

Layout Preview

1 * 3g.40gb ×

2 * 1g.20gb ×

Save

Cancel

The user can add new slices to the layout, or remove existing ones. The UI validates if the user exceeds the capacity of the GPU, for instance:

Configure GPU 3



Set the MIG layout for GPU 3. Saving replaces all slices currently configured on this GPU. Remove every slice from the layout preview to disable MIG on this GPU.

Instance Count

Slice Profile

5

*

1g.10gb



+ Add

This configuration exceeds the GPU capacity.

Layout Preview

1 * 3g.40gb ×

Save

Cancel

In the previous illustration, having 5 * 1g.10gb and 1 * 3g.40gb would exceed both the compute units and memory of the GPU. The modal detects this behavior and warns the user.

If you want to **disable MIG** on a GPU then you must remove every slice:

Configure GPU 3



Set the MIG layout for GPU 3. Saving replaces all slices currently configured on this GPU. Remove every slice from the layout preview to disable MIG on this GPU.

Instance Count

Slice Profile

*

+ Add

Layout Preview

No slices configured. Saving will disable MIG on this GPU.

Save

Cancel

The UI alerts the user that having no slices will disable MIG on a specific GPU. It does NOT imply that the GPU cannot be used but indicates that if a workload uses this GPU, then it would use the entire GPU.

Note that the Instance Count field upper limit is capped at the maximum count of the selected Slice Profile. For instance, as an example for GPU NVIDIA H100 80GB HBM3

- For Slice Profile 1g.10gb, the maximum amount of slices you can have is 7. Otherwise you would exceed the compute unit capacity of the GPU. Therefore, Instance Count field upper limit is set to 7.
- For Slice Profile 1g.10gb+me, you can only have 1, so Instance Count field upper limit is set to 1.



Note: Other GPUs can have different slices and max amounts.

After setting the slices in this modal and clicking on Save, the UI marks that GPU as Modified:

3	NVIDIA H100 80GB HBM3	Modified	1 * 3g.40gb	2 * 1g.20gb	7 * 1g.10gb	Configure	Revert
---	-----------------------	----------	-------------	-------------	-------------	-----------	--------

It implies what slices would be removed and what would be added. User can use the Revert button to revert back to the original slices for that particular GPU. Or they can use the Configure button again to overwrite the slices for that GPU.

Configuring on Every GPU Modal

You can use the Configure on Every GPU button at the node level:

GPUs		Configure On Every GPU	Revert
#	Model	Slices	
0	NVIDIA H100 80GB HBM3	1 * 1g.10gb+me 2 * 1g.20gb	Configure
1	NVIDIA H100 80GB HBM3	1 * 3g.40gb 2 * 1g.20gb	Configure
2	NVIDIA H100 80GB HBM3	1 * 3g.40gb 2 * 1g.20gb	Configure
3	NVIDIA H100 80GB HBM3 Modified	1 * 3g.40gb 2 * 1g.20gb 7 * 1g.10gb	Configure Revert
4	NVIDIA H100 80GB HBM3	1 * 3g.40gb 2 * 1g.20gb	Configure
5	NVIDIA H100 80GB HBM3	1 * 3g.40gb 2 * 1g.20gb	Configure
6	NVIDIA H100 80GB HBM3	1 * 3g.40gb 2 * 1g.20gb	Configure
7	NVIDIA H100 80GB HBM3	1 * 3g.40gb 2 * 1g.20gb	Configure

By using the Configure On Every GPU option, the user can select what slices to use on every GPU on this node. A similar modal window pops up, where the slices can be specified for every GPU on this node:

Configure Every GPU ✕

Set the same MIG layout for every GPU on this node. Saving replaces all slices currently configured on each GPU. Remove every slice from the layout preview to disable MIG on every GPU.

Instance Count

Slice Profile

▼

+ Add

Layout Preview

1 * 3g.40gb ✕

1 * 4g.40gb ✕

Save

Cancel

Clicking Save will overwrite the MIG layout for every GPU on that node:

GPUs [Configure On Every GPU](#) [Revert](#)

#	Model	Slices		
0	NVIDIA H100 80GB HBM3 Modified	1 * 1g.10gb+me 2 * 1g.20gb 1 * 3g.40gb 1 * 4g.40gb	Configure	Revert
1	NVIDIA H100 80GB HBM3 Modified	1 * 3g.40gb 2 * 1g.20gb 1 * 4g.40gb	Configure	Revert
2	NVIDIA H100 80GB HBM3 Modified	1 * 3g.40gb 2 * 1g.20gb 1 * 4g.40gb	Configure	Revert
3	NVIDIA H100 80GB HBM3 Modified	1 * 3g.40gb 2 * 1g.20gb 1 * 4g.40gb	Configure	Revert
4	NVIDIA H100 80GB HBM3 Modified	1 * 3g.40gb 2 * 1g.20gb 1 * 4g.40gb	Configure	Revert
5	NVIDIA H100 80GB HBM3 Modified	1 * 3g.40gb 2 * 1g.20gb 1 * 4g.40gb	Configure	Revert
6	NVIDIA H100 80GB HBM3 Modified	1 * 3g.40gb 2 * 1g.20gb 1 * 4g.40gb	Configure	Revert
7	NVIDIA H100 80GB HBM3 Modified	1 * 3g.40gb 2 * 1g.20gb 1 * 4g.40gb	Configure	Revert

The Revert button at the node level behaves in an analogous manner. It reverts back every GPU on this node to the original MIG configuration.

Saving the Multi-Instance GPU (MIG) configuration

Once you complete making all the MIG configuration changes on the UI, you may want to save it.

Before you begin

Note that:

A Revert button next to the Save button that restores the entire cluster (every node and GPU) to the original cluster-wide MIG configuration in the UI. No backend changes occur.

ec2-sam-0631.ecs.mc.comops.rocks [Modified](#) [Success](#) 6 GPUs are configured, 2 GPUs not configured.

GPUs [Configure On Every GPU](#) [Revert](#)

#	Model	Slices		
0	NVIDIA H100 80GB HBM3 Modified	4 * 1g.20gb	Configure	Revert
1	NVIDIA H100 80GB HBM3 Modified	3 * 2g.20gb	Configure	Revert
2	NVIDIA H100 80GB HBM3 Modified	2 * 3g.40gb	Configure	Revert
3	NVIDIA H100 80GB HBM3 Modified	1 * 4g.40gb 2 * 1g.10gb	Configure	Revert
4	NVIDIA H100 80GB HBM3 Modified	1 * 1g.10gb 1 * 1g.20gb 1 * 2g.20gb	Configure	Revert
5	NVIDIA H100 80GB HBM3 Modified	1 * 1g.10gb+me 1 * 3g.40gb 1 * 2g.20gb 1 * 1g.10gb	Configure	Revert
6	NVIDIA H100 80GB HBM3 Modified	No slices configured.	Configure	
7	NVIDIA H100 80GB HBM3 Modified	No slices configured.	Configure	

ec2-sam-0633.ecs.mc.comops.rocks [Modified](#) [Success](#) All 8 GPUs are configured.

GPUs [Configure On Every GPU](#) [Revert](#)

#	Model	Slices		
0	NVIDIA H100 80GB HBM3 Modified	1 * 1g.10gb+me 2 * 1g.20gb 1 * 3g.40gb 1 * 4g.40gb	Configure	Revert
1	NVIDIA H100 80GB HBM3 Modified	1 * 3g.40gb 2 * 1g.20gb 1 * 4g.40gb	Configure	Revert
2	NVIDIA H100 80GB HBM3 Modified	1 * 3g.40gb 2 * 1g.20gb 1 * 4g.40gb	Configure	Revert
3	NVIDIA H100 80GB HBM3 Modified	1 * 3g.40gb 2 * 1g.20gb 1 * 4g.40gb	Configure	Revert
4	NVIDIA H100 80GB HBM3 Modified	1 * 3g.40gb 2 * 1g.20gb 1 * 4g.40gb	Configure	Revert
5	NVIDIA H100 80GB HBM3 Modified	1 * 3g.40gb 2 * 1g.20gb 1 * 4g.40gb	Configure	Revert
6	NVIDIA H100 80GB HBM3 Modified	1 * 3g.40gb 2 * 1g.20gb 1 * 4g.40gb	Configure	Revert
7	NVIDIA H100 80GB HBM3 Modified	1 * 3g.40gb 2 * 1g.20gb 1 * 4g.40gb	Configure	Revert

< 1 > 10 / page v

[Save](#) [Revert](#)

Clicking Save opens a modal window, where the MIG configuration application can be selected:

Apply MIG Configuration ✕

Applying this configuration updates the MIG profile on affected nodes. For each affected node, the system will cordon the node, evict GPU-consuming workloads, apply the MIG profile, then uncordon the node. Non-GPU workloads on the node are not affected. Choose how many nodes to process at once.

☒ **Apply In Parallel Batches**

Process nodes in parallel within each batch; batches run one after another.

Batch Size

☐ **Apply To All Nodes At Once**

Process all affected nodes in parallel.

☐ **Apply One Node At A Time**

Process affected nodes sequentially to minimize disruption.

Save

Cancel

In this modal, the user can select how they want to apply the MIG configuration:

- **Apply In Parallel Batches:** This is the default option. Select this option to enable how many affected nodes should be processed in parallel. It combines the benefits of the other two options: parallel processing, but in a rolling fashion. Having too many nodes processed at once can clog the backend system.
- **Apply To All Nodes At Once:** Processing is done simultaneously for every node.



Caution: Processing many nodes at once could clog the backend system.

- **Apply One Node At A Time:** Nodes are processed one-by-one.
- The modal text describes what applying the MIG configuration stands for: For each affected node, the system will cordon the node, evict GPU-consuming workloads, apply the MIG profile, and then uncordon the node. Apply workflow is the same for every node.
- The UI controls what nodes will be sent for processing:
 - Typically only the modified nodes will be sent for processing, the other nodes will be left untouched (or unsent).
 - If the cluster-wide strategy was modified, then every node will be sent for processing, even if a certain node did not require any MIG configuration changes.

- When the MIG feature is used the very first time, every node will be sent for processing.

Polling the MIG configuration progress

After selecting the proper MIG config application method and clicking on Save, the UI initiates a request to the backend. The backend validates the request, then it starts applying the MIG configuration in the background. Then the UI starts polling the backend for retrieving the progress about the MIG config application.

The top screenshot shows the 'MIG Configuration' page in the AWS Management Console. A success message indicates that the MIG configuration request has been successfully validated and submitted. The 'Cluster Strategy' section shows the 'Mixed' option selected. The 'Nodes' table lists two nodes, both with a 'Success' status.

The bottom screenshot shows the 'Applying MIG Configuration' progress bar at 25%. The 'Cluster Strategy' section remains the same. The 'Nodes' table shows the same two nodes, but their status is now 'Awaiting Profile Propagation'.

Name	Status	Overview
ecs-sxm-0631.ecs.mc.comops.rock	Success	6 GPUs are configured, 2 GPUs not configured.
ecs-sxm-0633.ecs.mc.comops.rock	Success	All 8 GPUs are configured.

Name	Status	Overview
ecs-sxm-0631.ecs.mc.comops.rock	Awaiting Profile Propagation	25%
ecs-sxm-0633.ecs.mc.comops.rock	Awaiting Profile Propagation	25%

Administration

Diagnostic Data Authentication CA Certificates Databases Alerts Network Metrics Backup And Restore MIG Configuration

Applying MIG Configuration

A configuration is currently being applied across the cluster. Further configuration changes are temporarily disabled until this operation completes.

69%

Cluster Strategy

Choose how Multi-Instance GPU (MIG) configuration profiles are distributed across cluster nodes to optimize workload allocation, partition hardware resources, and maintain consistent node behavior across your environment.

Mixed Current

Allow different MIG profiles on different GPUs.

Single

Apply a single MIG profile uniformly across all GPUs.

None

Disable MIG on all nodes in the cluster.

Nodes

Q Search

Name	Status	Overview
ecs-sxm-0631.ecs.mc.comops.rocks	Evicting GPU Workloads	62%
ecs-sxm-0633.ecs.mc.comops.rocks	Applying Profile	75%

< 1 > 10 / page

Save

Revert

Every modification action (Save, Revert) is disabled while the MIG configuration change is in progress. If a MIG configuration change is being applied, then no other MIG configuration change request can be sent in because the backend rejects it, and also the UI forbids such.

When every MIG configuration has been applied on every MIG node, the UI will reflect the new MIG configuration. If any error was detected, then the UI will display it. Error details will be displayed in the Overview section of the node, and status will be displayed as Failed.



Note: If there was a cluster-wide issue (not node related), then this cluster-wide issue would be displayed at the same location as the progress banner during the save operation.

Administration

Diagnostic Data Authentication CA Certificates Databases Alerts Network Metrics Backup And Restore MIG Configuration

Cluster Strategy

Choose how Multi-Instance GPU (MIG) configuration profiles are distributed across cluster nodes to optimize workload allocation, partition hardware resources, and maintain consistent node behavior across your environment.

Mixed Current

Allow different MIG profiles on different GPUs.

Single

Apply a single MIG profile uniformly across all GPUs.

None

Disable MIG on all nodes in the cluster.

Nodes

Q Search

Name	Status	Overview
ecs-sxm-0631.ecs.mc.comops.rocks	Success	6 GPUs are configured, 2 GPUs not configured.
ecs-sxm-0633.ecs.mc.comops.rocks	Success	All 8 GPUs are configured.

< 1 > 10 / page

Save

Revert

ecs-sxm-0631.ecs.mc.comops.rocks

Success

6 GPUs are configured, 2 GPUs not configured.

GPUs

Configure On Every GPU

#	Model	Slices
0	NVIDIA H100 80GB HBM3	4 * 1g.20gb Configure
1	NVIDIA H100 80GB HBM3	3 * 2g.20gb Configure
2	NVIDIA H100 80GB HBM3	2 * 3g.40gb Configure
3	NVIDIA H100 80GB HBM3	3 * 1g.10gb 1 * 4g.40gb Configure
4	NVIDIA H100 80GB HBM3	1 * 2g.20gb 1 * 1g.20gb 1 * 1g.10gb Configure
5	NVIDIA H100 80GB HBM3	1 * 1g.10gb 1 * 2g.20gb 1 * 3g.40gb 1 * 1g.10gb+me Configure
6	NVIDIA H100 80GB HBM3	No slices configured. Configure
7	NVIDIA H100 80GB HBM3	No slices configured. Configure

ecs-sxm-0633.ecs.mc.comops.rocks

Success

All 8 GPUs are configured.

GPUs

Configure On Every GPU

#	Model	Slices
0	NVIDIA H100 80GB HBM3	1 * 4g.40gb 1 * 3g.40gb Configure
1	NVIDIA H100 80GB HBM3	1 * 4g.40gb 1 * 3g.40gb Configure
2	NVIDIA H100 80GB HBM3	1 * 4g.40gb 1 * 3g.40gb Configure
3	NVIDIA H100 80GB HBM3	1 * 4g.40gb 1 * 3g.40gb Configure
4	NVIDIA H100 80GB HBM3	1 * 4g.40gb 1 * 3g.40gb Configure
5	NVIDIA H100 80GB HBM3	1 * 4g.40gb 1 * 3g.40gb Configure
6	NVIDIA H100 80GB HBM3	1 * 4g.40gb 1 * 3g.40gb Configure
7	NVIDIA H100 80GB HBM3	1 * 4g.40gb 1 * 3g.40gb Configure

Switching cluster strategies

You can switch the cluster strategies based on your need to slice the GPUs accordingly.

Switching to Single strategy

Switching to Single has no significant change. It highlights what change is required, in order to switch to single strategy. For example, ecs-sxm-0631.ecs.mc.comops.rocks needs change, because it does not have uniform slices.

Cluster Strategy

Choose how Multi-Instance GPU (MIG) configuration profiles are distributed across cluster nodes to optimize workload allocation, partition hardware resources, and maintain consistent node behavior across your environment.

☐ Mixed **Current**
Allow different MIG profiles on different GPUs.

☒ Single
Apply a single MIG profile uniformly across all GPUs.

☐ None
Disable MIG on all nodes in the cluster.

Nodes

Search

	Name	Status	Overview
▼	ecs-sxm-0631.ecs.mc.comops.rocks	Needs Change	6 GPUs are configured, 2 GPUs not configured.
▼	ecs-sxm-0633.ecs.mc.comops.rocks	Success	All 8 GPUs are configured.

ecs-sxm-0631.ecs.mc.comops.rocks **Needs Change** Success 6 GPUs are configured, 2 GPUs not confi

GPUs [Configure On Every GPU](#)

#	Model	Slices
0	NVIDIA H100 80GB HBM3 Needs Change	4 * 1g.20gb
1	NVIDIA H100 80GB HBM3 Needs Change	3 * 2g.20gb
2	NVIDIA H100 80GB HBM3 Needs Change	2 * 3g.40gb
3	NVIDIA H100 80GB HBM3 Needs Change	3 * 1g.10gb 1 * 4g.40gb
4	NVIDIA H100 80GB HBM3 Needs Change	1 * 2g.20gb 1 * 1g.20gb 1 * 1g.10gb
5	NVIDIA H100 80GB HBM3 Needs Change	1 * 1g.10gb 1 * 2g.20gb 1 * 3g.40gb 1 * 1g.10gb+me
6	NVIDIA H100 80GB HBM3 Needs Change	No slices configured.
7	NVIDIA H100 80GB HBM3 Needs Change	No slices configured.

ecs-sxm-0633.ecs.mc.comops.rocks conforms already with single strategy, it has uniform slices.

ecs-sxm-0633.ecs.mc.comops.rocks Success All 8 GPUs are configured.

GPUs [Configure On Every GPU](#)

#	Model	Slices
0	NVIDIA H100 80GB HBM3	1 * 4g.40gb 1 * 3g.40gb
1	NVIDIA H100 80GB HBM3	1 * 4g.40gb 1 * 3g.40gb
2	NVIDIA H100 80GB HBM3	1 * 4g.40gb 1 * 3g.40gb
3	NVIDIA H100 80GB HBM3	1 * 4g.40gb 1 * 3g.40gb
4	NVIDIA H100 80GB HBM3	1 * 4g.40gb 1 * 3g.40gb
5	NVIDIA H100 80GB HBM3	1 * 4g.40gb 1 * 3g.40gb
6	NVIDIA H100 80GB HBM3	1 * 4g.40gb 1 * 3g.40gb
7	NVIDIA H100 80GB HBM3	1 * 4g.40gb 1 * 3g.40gb



Note: The Configure button per GPU disappeared and only the Configure On Every GPU button is available.

Configuring individual GPUs does not apply for Single strategy, because you must have uniform slices for every GPU on a node.



Caution: For the MIG single strategy, it NOT a valid configuration to define multiple different slice profiles per GPU, for example: 1×3g.40gb + 1×4g.40gb on every GPU on a node. In Cludera Data Services on premises 1.5.5 SP4 release, the MIG Configuration tab, the backend, and also the GPU operator itself allows such, and reports success. However, note that, under the hood this is **NOT** a valid configuration and this scenario must be avoided.

While using the single strategy users must select only one slice profile (for example: 2×3g.40gb on every GPU on a node) and should **NOT** select multiple different slice profiles.

Switching to None strategy

By switching to None, the UI auto disables MIG on every GPU.

Cluster Strategy

Choose how Multi-Instance GPU (MIG) configuration profiles are distributed across cluster nodes to optimize workload allocation, partition hardware resources, and maintain consistent node behavior across your environment.

☐ Mixed

☒ Current

Allow different MIG profiles on different GPUs.

☐ Single

Apply a single MIG profile uniformly across all GPUs.

☒ None

Disable MIG on all nodes in the cluster.

Nodes

	Name	Status	Overview
▼	ecs-sxm-0631.ecs.mc.comops.rock	Success	6 GPUs are configured, 2 GPUs not configured.
▼	ecs-sxm-0633.ecs.mc.comops.rock	Success	All 8 GPUs are configured.

<

1

>

10 / page

ecs-sxm-0631.ecs.mc.comops.rock

Disabled

Success

6 GPUs are configured, 2 GPUs not configured.

GPUs

#	Model	Slices
0	NVIDIA H100 80GB HBM3	4 * 1g.20gb
1	NVIDIA H100 80GB HBM3	3 * 2g.20gb
2	NVIDIA H100 80GB HBM3	2 * 3g.40gb
3	NVIDIA H100 80GB HBM3	3 * 1g.10gb 1 * 4g.40gb
4	NVIDIA H100 80GB HBM3	1 * 2g.20gb 1 * 1g.20gb 1 * 1g.10gb
5	NVIDIA H100 80GB HBM3	1 * 1g.10gb 1 * 2g.20gb 1 * 3g.40gb 1 * 1g.10gb+me
6	NVIDIA H100 80GB HBM3	
7	NVIDIA H100 80GB HBM3	

ecs-sxm-0633.ecs.mc.comops.rock

Disabled

Success

All 8 GPUs are configured.

GPUs

#	Model	Slices
0	NVIDIA H100 80GB HBM3	1 * 4g.40gb 1 * 3g.40gb
1	NVIDIA H100 80GB HBM3	1 * 4g.40gb 1 * 3g.40gb
2	NVIDIA H100 80GB HBM3	1 * 4g.40gb 1 * 3g.40gb
3	NVIDIA H100 80GB HBM3	1 * 4g.40gb 1 * 3g.40gb
4	NVIDIA H100 80GB HBM3	1 * 4g.40gb 1 * 3g.40gb
5	NVIDIA H100 80GB HBM3	1 * 4g.40gb 1 * 3g.40gb
6	NVIDIA H100 80GB HBM3	1 * 4g.40gb 1 * 3g.40gb
7	NVIDIA H100 80GB HBM3	1 * 4g.40gb 1 * 3g.40gb

There is no Configure, nor Configure On Every GPU buttons for none strategy. MIG is disabled on every GPU within the cluster. Saving the changes will ensure MIG would be disabled cluster-wide.

API support for the Multi-Instance GPU (MIG)

APIs that support the MIG feature.

Two new services are created and supported for this feature.

- config
- config-api



Note: These APIs are limited only to the MIG feature.

Read APIs

POST /api/v1/config/describeMigClusterSettings

Describe cluster-wide Nvidia MIG settings.

- **Description:** Returns the Multi-Instance GPU (MIG) strategy and MIG profiles ConfigMap name, defined in the GPU Operator ClusterPolicy. Unset fields are returned as empty strings when the ClusterPolicy exists.
- **Right:** config/describeMigClusterSettings
- **Example input:** {}
- **Example response:** {"migStrategy":"mixed","migProfilesConfigMap":"cdp-mig-profiles"}

POST /api/v1/config/listMigCapableNodeNames

List MIG-capable nodes.

- **Description:** Lists all node names that have MIG-capable GPUs, such as HGX nodes.
- **Right:** config/listMigCapableNodeNames
- **Example input:** {}
- **Example response:** {"nodeNames": ["ecs-sxm-0631.ecs.mc.comops.rocks", "ecs-sxm-0633.ecs.mc.comops.rocks"] }

POST /api/v1/config/listMigNodeDetails

List MIG node details.

- **Description:** Lists detailed MIG inventory for the requested nodes, or all MIG-capable nodes if the input node list is empty. Requests containing unknown or non-MIG-capable node names return HTTP 200 with error details on those specific node objects. The entire call fails only for cluster-level errors, such as inability to list nodes.
- **Right:** config/listMigNodeDetails
- **Example input:** {"nodes":["ecs-sxm-0631.ecs.mc.comops.rocks","ecs-sxm-0633.ecs.mc.comops.rocks"]}
- **Example response:**

```
{ "nodes": [ { "name": "ecs-sxm-0631.ecs.mc.comops.rocks",
               "appliedProfile": { "name": "cdp-0c431696", "state": "succ
               ess",
               "slices": [ { "gpuIndex": 0, "slice": { "size": "1g.20gb",
               "count": 4 } },
               { "gpuIndex": 1, "slice": { "size": "2g.20gb", "count": 3
               } },
               { "gpuIndex": 2, "slice": { "size": "3g.40gb", "count": 2 }
               },
               { "gpuIndex": 3, "slice": { "size": "1g.10gb", "count": 3
               } },
               { "gpuIndex": 3, "slice": { "size": "4g.40gb", "count": 1 }
               },
               ],
               },
               ],
               }
```

```

    },
    { "gpuIndex": 4, "slice": { "size": "1g.20gb", "count": 1 } },
    { "gpuIndex": 4, "slice": { "size": "1g.10gb", "count": 1 } },
    { "gpuIndex": 5, "slice": { "size": "1g.10gb", "count": 1 } },
    { "gpuIndex": 5, "slice": { "size": "2g.20gb", "count": 1 } },
    { "gpuIndex": 5, "slice": { "size": "3g.40gb", "count": 1 } },
    { "gpuIndex": 5, "slice": { "size": "1g.10gb+me", "count": 1 } } ],
    "availableSlices": [ { "availableOnGpuIndices": [ 0, 1, 2, 3, 4, 5, 6, 7 ],
    "slice": { "size": "1g.10gb", "count": 7 } },
    { "availableOnGpuIndices": [ 0, 1, 2, 3, 4, 5, 6, 7 ],
    "slice": { "size": "1g.10gb+me", "count": 1 } },
    { "availableOnGpuIndices": [ 0, 1, 2, 3, 4, 5, 6, 7 ],
    "slice": { "size": "1g.20gb", "count": 4 } },
    { "availableOnGpuIndices": [ 0, 1, 2, 3, 4, 5, 6, 7 ],
    "slice": { "size": "2g.20gb", "count": 3 } },
    { "availableOnGpuIndices": [ 0, 1, 2, 3, 4, 5, 6, 7 ],
    "slice": { "size": "3g.40gb", "count": 2 } },
    { "availableOnGpuIndices": [ 0, 1, 2, 3, 4, 5, 6, 7 ],
    "slice": { "size": "4g.40gb", "count": 1 } },
    { "availableOnGpuIndices": [ 0, 1, 2, 3, 4, 5, 6, 7 ],
    "slice": { "size": "7g.80gb", "count": 1 } } ],
    "errors": [], "gpus": [ { "gpuIndex": 0, "name": "NVIDIA
H100 80GB HBM3",
    "uuid": "GPU-4d40c852-2779-7071-e53e-19b907692111" },
    { "gpuIndex": 1, "name": "NVIDIA H100 80GB HBM3",
    "uuid": "GPU-788ceb31-bcde-eb50-7ac2-07862c5b9970" },
    { "gpuIndex": 2, "name": "NVIDIA H100 80GB HBM3",
    "uuid": "GPU-4a878356-b30d-efcc-24b3-f4475cd1b671" },
    { "gpuIndex": 3, "name": "NVIDIA H100 80GB HBM3",
    "uuid": "GPU-8e65736b-5df6-4e47-40ad-15efab6a10b3" },
    { "gpuIndex": 4, "name": "NVIDIA H100 80GB HBM3",
    "uuid": "GPU-5a07dfe6-b3d1-4ecc-491b-848b8713c365" },
    { "gpuIndex": 5, "name": "NVIDIA H100 80GB HBM3",
    "uuid": "GPU-67cb5a19-0abc-57e2-527e-c0a1caee841f" },
    { "gpuIndex": 6, "name": "NVIDIA H100 80GB HBM3",
    "uuid": "GPU-4afba924-6414-dcf1-98f1-54dd0111f39a" },
    { "gpuIndex": 7, "name": "NVIDIA H100 80GB HBM3",
    "uuid": "GPU-4db81d0b-d1bf-af8f-dc02-9d37a9299fdd" } ],
    "gpuProductLabels": { "nvidia.com/mig-1g.20gb.product": "NV
IDIA-H100-80GB-HBM3-MIG-1g.20gb",
    "nvidia.com/mig-4g.40gb.product": "NVIDIA-H100-80GB-HBM3-MI
G-4g.40gb",
    "nvidia.com/gpu.product": "NVIDIA-H100-80GB-HBM3",
    "nvidia.com/mig-3g.40gb.product": "NVIDIA-H100-80GB-HBM3-MI
G-3g.40gb",
    "nvidia.com/mig-1g.10gb.me.product": "NVIDIA-H100-80GB-HBM3-
MIG-1g.10gb.me",
    "nvidia.com/mig-2g.20gb.product": "NVIDIA-H100-80GB-HBM3-
MIG-2g.20gb",
    "nvidia.com/mig-1g.10gb.product": "NVIDIA-H100-80GB-HBM3-MI
G-1g.10gb" } },
    { "name": "ecs-sxm-0633.ecs.mc.comops.rocks",
    "appliedProfile": { "name": "cdp-03912a68",
    "state": "success", "slices": [ { "gpuIndex": 0,
    "slice": { "size": "4g.40gb", "count": 1 } } ],

```

```

    { "gpuIndex": 0, "slice": { "size": "3g.40gb", "count": 1 }
  },
  { "gpuIndex": 1, "slice": { "size": "4g.40gb", "count": 1
} }},
    { "gpuIndex": 1, "slice": { "size": "3g.40gb",
"count": 1 } }},
    { "gpuIndex": 2, "slice": { "size": "4g.40gb", "count": 1 }
  },
    { "gpuIndex": 2, "slice": { "size": "3g.40gb", "count": 1
} }},
    { "gpuIndex": 3, "slice": { "size": "4g.40gb", "count": 1 }
  },
    { "gpuIndex": 3, "slice": { "size": "3g.40gb", "count": 1
} }},
    { "gpuIndex": 4, "slice": { "size": "4g.40gb", "count": 1 }
  },
    { "gpuIndex": 4, "slice": { "size": "3g.40gb", "count": 1
} }},
    { "gpuIndex": 5, "slice": { "size": "4g.40gb", "count": 1 }
  },
    { "gpuIndex": 5, "slice": { "size": "3g.40gb", "count": 1
} }},
    { "gpuIndex": 6, "slice": { "size": "4g.40gb", "count": 1 }
  },
    { "gpuIndex": 6, "slice": { "size": "3g.40gb", "count": 1
} }},
    { "gpuIndex": 7, "slice": { "size": "4g.40gb", "count": 1 }
  },
    { "gpuIndex": 7, "slice": { "size": "3g.40gb", "count": 1
} } ] },
  "availableSlices": [ { "availableOnGpuIndices": [ 0, 1, 2,
3, 4, 5, 6, 7 ],
    "slice": { "size": "1g.10gb", "count": 7 } },
    { "availableOnGpuIndices": [ 0, 1, 2, 3, 4, 5, 6, 7 ],
    "slice": { "size": "1g.10gb+me", "count": 1 } },
    { "availableOnGpuIndices": [ 0, 1, 2, 3, 4, 5, 6, 7 ],
    "slice": { "size": "1g.20gb", "count": 4 } },
    { "availableOnGpuIndices": [ 0, 1, 2, 3, 4, 5, 6, 7 ],
    "slice": { "size": "2g.20gb", "count": 3 } },
    { "availableOnGpuIndices": [ 0, 1, 2, 3, 4, 5, 6, 7 ],
    "slice": { "size": "3g.40gb", "count": 2 } },
    { "availableOnGpuIndices": [ 0, 1, 2, 3, 4, 5, 6, 7 ],
    "slice": { "size": "4g.40gb", "count": 1 } },
    { "availableOnGpuIndices": [ 0, 1, 2, 3, 4, 5, 6, 7 ],
    "slice": { "size": "7g.80gb", "count": 1 } } ],
  "errors": [], "gpus": [ { "gpuIndex": 0, "name": "NVIDIA H100 80GB HBM3", "uuid": "GPU-5a61d619-287c-0786-6fc0-1e51930e5812" },
    { "gpuIndex": 1, "name": "NVIDIA H100 80GB HBM3", "uuid": "GPU-6a09ddb0-e0ac-9072-d892-34c5fe1e52b4" },
    { "gpuIndex": 2, "name": "NVIDIA H100 80GB HBM3", "uuid": "GPU-c067fb97-2dd7-0642-f2a7-78ea17323265" },
    { "gpuIndex": 3, "name": "NVIDIA H100 80GB HBM3", "uuid": "GPU-aab83db8-2110-c274-993b-9136c1a391c5" },
    { "gpuIndex": 4, "name": "NVIDIA H100 80GB HBM3", "uuid": "GPU-2c06ab8f-e69b-d5a1-bfbb-10f75cd0b774" },
    { "gpuIndex": 5, "name": "NVIDIA H100 80GB HBM3", "uuid": "GPU-f0554c20-5a8e-6115-2e28-d2a7d209a230" },
    { "gpuIndex": 6, "name": "NVIDIA H100 80GB HBM3", "uuid": "GPU-4c643312-3bb2-7772-b2c7-cf6a5a305dcf" },
    { "gpuIndex": 7, "name": "NVIDIA H100 80GB HBM3", "uuid": "GPU-976d2e7f-c223-66a3-3c51-8c9b135ea5b4" } ],
  "gpuProductLabels": { "nvidia.com/mig-4g.40gb.product": "NVIDIA-H100-80GB-HBM3-MIG-4g.40gb",
    "nvidia.com/gpu.product": "NVIDIA-H100-80GB-HBM3",

```

```
"nvidia.com/mig-3g.40gb.product": "NVIDIA-H100-80GB-HBM3-MIG-3g.40gb" } } ] }
```

POST /api/v1/config/describeMigApplyWorkflow

Describe the current or most recent MIG apply backend workflow.

- **Description:** Returns the status of an ongoing MIG apply workflow or, when idle, the details of the most recent completed run — including per-node phases and execution results.
- **Right:** config/describeMigApplyWorkflow
- **Example input:** {}
- **Example response:** { "workflowId": "9866ba80-6cf6-4919-a2d8-87d4fa057d13", "workflowType": "PROFILE_ONLY", "targetStrategy": "mixed", "running": false, "startedAt": "2026-08-18T22:46:24.520747766Z", "completedAt": "2026-08-18T22:54:46.365941614Z", "overallStatus": "SUCCESS", "nodes": [{ "nodeName": "ecs-sxm-0633.ecs.mc.comops.rocks", "phase": "COMPLETED", "profileName": "cdp-03912a68", "state": "success" }, { "nodeName": "ecs-sxm-0631.ecs.mc.comops.rocks", "phase": "COMPLETED", "profileName": "cdp-0c431696", "state": "success" }] }

Save MIG API

POST /api/v1/config/applyMigConfig

Save and validate a MIG configuration for cluster nodes.

- **Description:** Validates the requested MIG strategy and per-node slices, then reconfigures the affected nodes via custom mig-parted configs. Nodes whose configuration is unchanged may be omitted. Returns immediately with IN_PROGRESS; poll describeMigApplyWorkflow for per-node progress.
- **Right:** config/applyMigConfig
- **Example input:**

```
{ "strategy": "mixed", "batchSize": 0, "nodes": [ { "nodeName": "ecs-axm-0631.ecs.mc.comops.rocks",
  "gpus": [ { "gpuIndex": 0, "slices": [ { "size": "1g.20gb",
    "count": 4 } ] },
    { "gpuIndex": 1, "slices": [ { "size": "2g.20gb", "count":
3 } ] },
    { "gpuIndex": 2, "slices": [ { "size": "3g.40gb", "count":
2 } ] },
    { "gpuIndex": 3, "slices": [ { "size": "1g.10gb", "count": 3
  },
    { "size": "4g.40gb", "count": 1 } ] },
    { "gpuIndex": 4, "slices": [ { "size": "2g.20gb", "count":
1 },
    { "size": "1g.20gb", "count": 1 }, { "size": "1g.10gb", "count":
1 } ] },
    { "gpuIndex": 5, "slices": [ { "size": "1g.10gb", "count":
1 },
    { "size": "2g.20gb", "count": 1 }, { "size": "3g.40gb", "count":
1 },
    { "size": "1g.10gb+me", "count": 1 } ] },
    { "gpuIndex": 6, "slices": [ ] },
    { "gpuIndex": 7, "slices": [ ] } ] },
  { "nodeName": "ecs-axm-0633.ecs.mc.comops.rocks",
    "gpus": [ { "gpuIndex": 0, "slices": [ { "size": "4g.40gb",
      "count": 1 },
        { "size": "3g.40gb", "count": 1 } ] },
        { "gpuIndex": 1, "slices": [ { "size": "4g.40gb", "count":
1 }, { "size": "3g.40gb", "count": 1 } ] },
        { "gpuIndex": 2, "slices": [ { "size": "4g.40gb", "count": 1
  },
```

```
    { "size": "2g.40gb", "count": 1 } ] },  
1 },  
    { "gpuIndex": 3, "slices": [ { "size": "4g.40gb", "count":  
  
        { "size": "3g.40gb", "count": 1 } ] },  
1 },  
    { "gpuIndex": 4, "slices": [ { "size": "4g.40gb", "count":  
  
        { "size": "3g.40gb", "count": 1 } ] },  
1 },  
    { "gpuIndex": 5, "slices": [ { "size": "4g.40gb", "count":  
  
        { "size": "3g.40gb", "count": 1 } ] },  
},  
    { "gpuIndex": 6, "slices": [ { "size": "4g.40gb", "count": 1  
  
        { "size": "3g.40gb", "count": 1 } ] },  
1 },  
    { "gpuIndex": 7, "slices": [ { "size": "4g.40gb", "count":  
  
        { "size": "3g.40gb", "count": 1 } ] } ] } ] }
```

- ```
{ "status": "IN_PROGRESS", "message": "MIG reconfiguration accepted and is being applied. Poll describeMigApplyWorkflow for workflow progress." }
```

## Setting up Single or Multi-Rail cluster

Nvidia Hyperscale Graphics eXtension (HGX), a modular hardware platform and server specification designed for high-performance computing (HPC) and artificial intelligence (AI) data centers.

Cloudera Control Plane provisions the GPU and Network Operator control loops directly onto the cluster. This is supported only on the Cloudera Embedded Container Service clusters.

You must perform the following actions:

1. Setup the host node requirements
2. Single/Multi rail setup for homogenous HGX cluster

## Setting up Host node

As the driver management is turned off in the Cloudera Control Plane deployment, you must execute the following setup on all worker nodes prior to joining the cluster.

## Before you begin

Refer: [CUDA Installation Guide for Linux](#).

For Platform support information (Cloudera deploys gpu-operator 26.3.2): Refer:[Platform Support instructions](#).

## Procedure

1. To enable Kernel Dependencies and EPEL, run the following commands:

Worker nodes require kernel development packages matching the exact active kernel version.

```
Register RHEL 9 Repositories (skip on Rocky/AlmaLinux 9)
sudo subscription-manager repos --enable=rhel-9-for-x86_64-appstream-rpms
sudo subscription-manager repos --enable=rhel-9-for-x86_64-baseos-rpms
sudo subscription-manager repos --enable=codeready-builder-for-rhel-9-x86_64-rpms

Install matching kernel development headers
sudo dnf install -y kernel-devel-$(uname -r) kernel-headers EPEL
```

```
Add CUDA & Toolkit Repositories
sudo dnf config-manager --add-repo https://developer.download.nvidia.com/
compute/cuda/repos/rhel9/x86_64/cuda-rhel9.repo
curl -s -L https://nvidia.github.io/libnvidia-container/stable/rpm/nvidia-
container-toolkit.repo | \
 sudo tee /etc/yum.repos.d/nvidia-container-toolkit.repo
```

## 2. Install the Driver and FabricManager

You must install drivers natively on the host.

```
Install open-dkms NVIDIA driver & FabricManager
sudo dnf module enable -y nvidia-driver:open-dkms
sudo dnf install -y nvidia-open nvidia-fabricmanager

Enable FabricManager (Mandatory for HGX A100/H100/B200 NVSwitch systems)
sudo systemctl enable --now nvidia-fabricmanager
```

## 3. Link must be up and running. You must run this on every HGX node:

```
rdma link show
```

```
Example OUTPUT
link mlx5_0/1 subnet_prefix fe80:0000:0000:0000 lid 28 sm_lid 28 lmc 0
state ACTIVE physical_state LINK_UP
link mlx5_1/1 subnet_prefix fe80:0000:0000:0000 lid 11 sm_lid 28 lmc 0
state ACTIVE physical_state LINK_UP
link mlx5_2/1 subnet_prefix fe80:0000:0000:0000 lid 26 sm_lid 28 lmc 0
state ACTIVE physical_state LINK_UP
link mlx5_3/1 subnet_prefix fe80:0000:0000:0000 lid 8 sm_lid 28 lmc 0 s
tate ACTIVE physical_state LINK_UP
link mlx5_4/1 subnet_prefix fe80:0000:0000:0000 lid 33 sm_lid 28 lmc 0 s
tate ACTIVE physical_state LINK_UP
link mlx5_5/1 subnet_prefix fe80:0000:0000:0000 lid 5 sm_lid 28 lmc 0 st
ate ACTIVE physical_state LINK_UP
link mlx5_6/1 subnet_prefix fe80:0000:0000:0000 lid 35 sm_lid 28 lmc 0 st
ate ACTIVE physical_state LINK_UP
link mlx5_7/1 subnet_prefix fe80:0000:0000:0000 lid 25 sm_lid 28 lmc 0 st
ate ACTIVE physical_state LINK_UP
```

### What to do next

Set up the Single/Multi Rail for homogeneous HGX cluster

## Setting up single or multi rail for homogenous HGX cluster

Setting up the HGX cluster involves multiple operational phases that includes discovering rails, defining YAML configuration, applying the changes, and verifying configurations.

### Discovering the rails

Make sure that before running this command you substitute:

- HGX node names to: <node-0> <node-1>
- Username to: <username>

```
for NODE in <node-0> <node-1>; do
 echo "=== $NODE ==="
 ssh -o StrictHostKeyChecking=no <username>@$NODE \
```

```

'printf "%-6s %-14s %-14s %-14s %-9s %-9s\n" "GPU" "pf-name" "network-t
ype" "nic-vendor-id" "totalvfs" "state"
ibdev2netdev 2>/dev/null | while read -r m _ _ _ nd _; do
 case $m in mlx5_[0-7])
 t=$(cat /sys/class/net/$nd/type 2>/dev/null)
 v=$(cat /sys/class/net/$nd/device/vendor 2>/dev/null | sed "s/0x//")

 vfs=$(cat /sys/class/net/$nd/device/sriov_totalvfs 2>/dev/null ||
echo "-")
 st=$(cat /sys/class/net/$nd/operstate 2>/dev/null || echo "-")
 case $t in 32) l=InfiniBand;; 1) l=Ethernet;; *) l=unknown;; esac
 printf "%-6s %-14s %-14s %-14s %-9s %-9s\n" "GPU${m#mlx5_}" "$nd" "$
l" "$v" "$vfs" "$st"
 ;; esac
 done'
echo
done

```

# Example OUTPUT per node

=== <node-0> ===

| GPU  | pf-name  | network-type | nic-vendor-id | totalvfs | state |         |
|------|----------|--------------|---------------|----------|-------|---------|
| GPU0 | ibp15s0  | InfiniBand   | 15b3          | 8        | up    | # rail0 |
| GPU1 | ibp20s0  | InfiniBand   | 15b3          | 8        | up    | # rail1 |
| GPU2 | ibp28s0  | InfiniBand   | 15b3          | 8        | up    | # ...   |
| GPU3 | ibp102s0 | InfiniBand   | 15b3          | 8        | up    |         |
| GPU4 | ibp155s0 | InfiniBand   | 15b3          | 8        | up    |         |
| GPU5 | ibp170s0 | InfiniBand   | 15b3          | 8        | up    |         |
| GPU6 | ibp187s0 | InfiniBand   | 15b3          | 8        | up    |         |
| GPU7 | ibp218s0 | InfiniBand   | 15b3          | 8        | up    | # rail7 |



**Note:** For the next steps, the generated output is required to create the YAML configurations. Before proceeding, make sure that the output is identical for every HGX node (homogenous HGX setup).

- Same GPU <-> NIC layout
- Same pf-name
- Same network-type (IB vs Ethernet)
- Same nic-vendor-id / totalvfs
- state can be different, however if you want to set up:
  - **single-rail:** state is up for GPU0 is enough on every HGX node.
  - **multi-rail:** state is up for every GPU on every HGX node.



**Important:** Currently, the heterogeneous HGX setup is not supported.

Refer to the [NVIDIA documentation](#) to seek directions for any kind of setup.

### YAML configuration templates per rail

The YAML configuration template that must be defined per rail. It is <highlighted> and # commented what values need to be substituted from the previous section.

```

--- IPPool: allocates IPs for this rail ---
apiVersion: nv-ipam.nvidia.com/v1alpha1
kind: IPPool
metadata:
 name: rail<N>-pool # <- index
 namespace: network-operator
spec:
 subnet: "10.0.<N>.0/24" # <- index
 perNodeBlockSize: 20

```

```

gateway: "10.0.<N>.1" # <- index

--- SrioVIBNetwork: exposes the rail as a Network Attachment Definition -
--
apiVersion: sriovnetwork.openshift.io/v1
Use SrioVIBNetwork for InfiniBand. Use SrioVNetwork for Ethernet.
kind: <SrioVIBNetwork OR SrioVNetwork>
metadata:
 name: rail<N>-network # <- index
 namespace: network-operator
spec:
 vlan: 0 # <- Ethernet only – DELETE this line for InfiniBand
 linkState: enable # <- InfiniBand only – DELETE this line for Ethernet
 networkNamespace: "default"
 resourceName: "sriovlegacy<N>" # <- index
 ipam: |-
 {
 "type": "nv-ipam",
 "poolName": "rail<N>-pool" # <- index
 }

--- SrioVNetworkNodePolicy: configures the VFs on the physical function -
--
apiVersion: sriovnetwork.openshift.io/v1
kind: SrioVNetworkNodePolicy
metadata:
 name: rail<N>-policy # <- index
 namespace: network-operator
spec:
 deviceType: netdevice
 mtu: 1500
 nicSelector:
 vendor: "<nic-vendor-id>" # <- e.g.: 15b3
 # for single-rail use the pf-name of GPU0,
 # for multi-rail use the pf-name of the corresponding GPU
 pfNames:
 - <pf-name> # <- e.g.: ibp15s0
 nodeSelector:
 # e.g.: feature.node.kubernetes.io/pci-15b3.present: "true"
 feature.node.kubernetes.io/pci-<nic-vendor-id>.present: "true" # <- e.g.
: 15b3
 numVfs: <totalvfs> # <- number of VFs to create per PF, e.g.: 8
 linkType: <IB OR eth> # <- InfiniBand | Ethernet (from network-type)
 priority: 90
 isRdma: true
 resourceName: sriovlegacy<N> # <- index

```

### Configuring single-rail

Use the following information:

- You need to define only one instance of the YAML configuration template from above.
- Save this in a file on the Cloudera Embedded Container Service master node, and call it all-rails-combined.yaml.
- Remove every # helper comment.

### Configuring multi-rail

Use the following information:

You need to define as many instances of the YAML configuration template as many rails the discovery command has identified in the previous section. In this example, the rail count is 8, so the YAML configuration template from above should be repeated 8 times for each rail.

- Combine each configuration template instance into an all-rails-combined.yaml file and insert it on the Cloudera Embedded Container Service master node, add a --- separator after each instance.
- For every new rail, increment the index <N>, commence from N=0.
- For the pf-name in SrioNetworkNodePolicy, use the pf-name for the corresponding GPU.
- Remove every # helper comment.

### Applying the defined configurations

Before you start, you must login as root and set kubectl and KUBECONFIG. Perform the actions in the order of flow. Login to Cloudera Embedded Container Service master node and set kubectl command, and KUBECONFIG.



**Note:** The commands to be executed require root privileges:

```
sudo su
alias kubectl=/var/lib/rancher/rke2/bin/kubectl
export KUBECONFIG=/etc/rancher/rke2/rke2.yaml
```

### Disabling Operator Drain

Because of the Longhorn's instance manager disruption policy, it is not possible to evict the instance-manager Pods from the Multi-Instance GPU (MIG) capable nodes.

Therefore you must disable automatic draining:

```
kubectl patch sriovoperatorconfig default -n network-operator --type=merge \
 -p '{"spec":{"disableDrain":true}}'
```

```
Example OUTPUT
sriovoperatorconfig.sriovnetwork.openshift.io/default patched
```

### Applying the configuration changes

You must apply the changes:

```
kubectl apply -f all-rails-combined.yaml
```

```
Example OUTPUT
ippool.nv-ipam.nvidia.com/rail0-pool created
sriovibnetwork.sriovnetwork.openshift.io/rail0-network created

sriovnetworknodepolicy.sriovnetwork.openshift.io/rail0-policy created
...
ippool.nv-ipam.nvidia.com/rail7-pool created
sriovibnetwork.sriovnetwork.openshift.io/rail7-network created
sriovnetworknodepolicy.sriovnetwork.openshift.io/rail7-policy created
```

### Synchronizing the changes

Allow about 15 minutes and run the following command on a periodic basis:

```
kubectl get sriovnetworknodestate -n network-operator
```

```
Successful reconciliation OUTPUT looks like the following for every HGX node:
* SYNC STATUS + DESIRED SYNC STATE =
* InProgress + Reboot_Required
* Succeeded + Reboot_Required
```

```
* Succeeded + Idle
* CURRENT SYNC STATE = idle
NAME SYNC STATUS DESIRED SYNC STATE CURRENT SYNC STATE AGE
<node-0> Succeeded Idle Idle 10h
<node-1> InProgress Reboot_Required Idle 10h
```

### Rebooting every HGX Node

You must review and finalise as to which HGX node reboot scenario to proceed with.

- **Sequential** (safest and recommended):
  - Reboot one HGX node at a time.
- **Parallel** (only if ALL true):
  - No active GPU workloads on HGX nodes.
  - No critical Longhorn volumes with replicas only on those nodes.
  - Full HGX downtime is acceptable.
- **Batched parallel** (middle-ground):
  - Split HGX nodes into small batches (for example, two at a time).
  - Reboot each batch in parallel. After one batch has finished, verify and then do the same for the next batch.
- After every reboot session, you must perform a verification on the HGX node(s).

The reboot commands to be executed on each HGX node:

```
NODE=<node-0>
kubectl cordon $NODE
ssh <username>@$NODE 'sudo reboot'
Wait some time (about 15 minutes) after HGX node reboot...
kubectl uncordon $NODE
```

### Verifying the change requests

Use the following information to verify the changes

```
kubectl get sriovnetworknodepolicy,sriovibnetwork,sriovnetwork,ippools.nv-ip
am.nvidia.com -n network-operator
```

```
Example successful OUTPUT for Infiniband
For Ethernet, you should see SriovNetwork CRs, instead of SriovIBNetwork
NAME AGE
sriovnetworknodepolicy.sriovnetwork.openshift.io/rail0-policy 44m
...
sriovnetworknodepolicy.sriovnetwork.openshift.io/rail7-policy 44m

NAME AGE
sriovibnetwork.sriovnetwork.openshift.io/rail0-network 44m
...
sriovibnetwork.sriovnetwork.openshift.io/rail7-network 44m

NAME SUBNET GATEWAY BLOCK SIZE 10.0.0.1 20
ippool.nv-ipam.nvidia.com/rail0-pool 10.0.0.0/24 10.0.0.1 20
...
ippool.nv-ipam.nvidia.com/rail7-pool 10.0.7.0/24 10.0.7.1 20
```

## Verifying the Network Access Devices

Use the following information to verify the changes

```
kubectl get net-attach-def -n default -o json | \
jq -r '.items[] | select(.metadata.name | test("^rail")) | .metadata.name
as $n |
(.spec.config | fromjson) | "\($n)\t\(.type)\t\(.link_state // "n/a")"'
```

```
Example successful OUTPUT for Infiniband
For Ethernet, the middle column should show 'sriov'
rail0-network ib-sriov enable
...
rail7-network ib-sriov enable
```

## Confirming that the operator has successfully reconciled

Use the following information to verify the changes

```
kubectl get sriovnetworknodestate -n network-operator
```

```
Example successful OUTPUT
NAME SYNC STATUS DESIRED SYNC STATE CURRENT SYNC STATE AGE
<node-0> Succeeded Idle Idle 13h
<node-1> Succeeded Idle Idle 13h
```

## Confirming if the SR-IOV operator applied the policy correctly on each node

Before you start, you must substitute the HGX node names to: <node-0> <node-1>.

```
for NODE in <node-0> <node-1>; do
echo "=== $NODE ==="
kubectl get sriovnetworknodestate $NODE -n network-operator \
-o jsonpath='{range .status.interfaces[*]}{.name}{": numVfs="}{.numVfs}
{"\n"}{end}{'
done
```

```
Example successful OUTPUT
=== <node-0> ===
ibp15s0: numVfs=8
ibp20s0: numVfs=8
ibp28s0: numVfs=8
ibp102s0: numVfs=8
ibp155s0: numVfs=8
ibp170s0: numVfs=8
ibp187s0: numVfs=8
ibp218s0: numVfs=8
=== <node-1> ===
ibp15s0: numVfs=8
ibp20s0: numVfs=8
ibp28s0: numVfs=8
ibp102s0: numVfs=8
ibp155s0: numVfs=8
ibp170s0: numVfs=8
ibp187s0: numVfs=8
ibp218s0: numVfs=8
```

### Confirming if the Kubernetes scheduler can actually detect and assign the VFs to pods

Before you start, you must substitute the HGX node names to: <node-0> <node-1>.

```
for NODE in <node-0> <node-1>; do
 echo "=== $NODE ==="
 kubectl get node $NODE -o json | \
 jq '.status.allocatable | with_entries(select(.key | contains("sriov"))
)'
done
```

```
Example successful OUTPUT
=== <node-0> ===
{
 "nvidia.com/sriovlegacy0": "8",
 "nvidia.com/sriovlegacy1": "8",
 "nvidia.com/sriovlegacy2": "8",
 "nvidia.com/sriovlegacy3": "8",
 "nvidia.com/sriovlegacy4": "8",
 "nvidia.com/sriovlegacy5": "8",
 "nvidia.com/sriovlegacy6": "8",
 "nvidia.com/sriovlegacy7": "8"
}
=== <node-1> ===
{
 "nvidia.com/sriovlegacy0": "8",
 "nvidia.com/sriovlegacy1": "8",
 "nvidia.com/sriovlegacy2": "8",
 "nvidia.com/sriovlegacy3": "8",
 "nvidia.com/sriovlegacy4": "8",
 "nvidia.com/sriovlegacy5": "8",
 "nvidia.com/sriovlegacy6": "8",
 "nvidia.com/sriovlegacy7": "8"
}
```

### Verifying GPU Operator pods

You must verify that every GPU operator Pod STATUS is Running/Completed:

```
kubectl get pods -n gpu-operator
```

### Ensuring single/multi rail is functional

Substitute the values highlighted in the code and run the following script:

```
#!/bin/bash
=====
multirail_connectivity_test_template.sh
#
Cross-node multi-rail SR-IOV/RDMA connectivity test.
Schedules one pod per GPU node, attaches all rails, pings across every rail.
#
This is the "fill-in-the-blanks" version: the logic is fixed, you only edit
the CONFIG block below. Every line you must change is marked # <-- CHANGE
=====
=====
```

```

CONFIG -- edit everything in this block to match your cluster

kubectl + kubeconfig. On embedded ECS/RKE2 these are the on-node defaults;
on a normal kubeconfig set K=kubectl and KC=$HOME/.kube/config.
K=/var/lib/rancher/rke2/bin/kubectl # <-- CHANGE if kubectl
is elsewhere
KC=/etc/rancher/rke2/rke2.yaml # <-- CHANGE to your kubeconfig path

Container image for the test pods. Needs sh + ping + ls (busybox is enough).
In an air-gapped cluster point this at YOUR registry mirror.
IMG="<YOUR_REGISTRY>/busybox:latest" # <-- CHANGE to a reachable image

The two GPU nodes to test between (must be different nodes).
Get names with: $K --kubeconfig=$KC get nodes
NODE_A="<GPU_NODE_A>" # <-- CHANGE to your 1st GPU node name
NODE_B="<GPU_NODE_B>" # <-- CHANGE to your 2nd GPU node name

Namespace holding the rail NetworkAttachmentDefinitions and where pods run.
NS=default # <-- CHANGE if your rail NADs live elsewhere

Comma-separated list of rail NAD names to attach.
List them with: $K --kubeconfig=$KC get net-attach-def -n $NS
NETS="rail0-network,rail1-network,rail2-network,rail3-network,rail4-network,rail5-network,rail6-network,rail7-network" # <-- CHANGE to your rail NADs
The SR-IOV device-plugin resources each rail consumes, one "name: \"1\"" per
rail. Find them with: $K --kubeconfig=$KC get node $NODE_A -o json | grep -i sriov
The number of lines here MUST match the number of rails in NETS above.
RESOURCE_LIMITS=$(cat <<'LIMITS'
 nvidia.com/sriovlegacy0: "1"
 nvidia.com/sriovlegacy1: "1"
 nvidia.com/sriovlegacy2: "1"
 nvidia.com/sriovlegacy3: "1"
 nvidia.com/sriovlegacy4: "1"
 nvidia.com/sriovlegacy5: "1"
 nvidia.com/sriovlegacy6: "1"
 nvidia.com/sriovlegacy7: "1"
LIMITS
)
^-- CHANGE the resource names/count above to match your rails

Rail count (used only for the ping loop) and the IPv4 prefix your rail IPAM
hands out. Example: pools 10.0.0.0/24 .. 10.0.7.0/24 -> prefix "10.0" and
the per-rail octet is the rail index. Adjust the regex if your pools differ.
RAIL_COUNT=8 # <-- CHANGE to your number of rails
RAIL_IP_PREFIX="10.0" # <-- CHANGE to your rail subnet prefix


```

```
END CONFIG -- no changes needed below this line

run() {
 sudo "$K" --kubeconfig="$KC" "$@"; # drop "sudo" if not needed for kubec
onfig
}
ex() {
 run exec "$1" -c t -n "$NS" -- sh -c "$2" 2>&1;
}
cleanup() {
 run delete pod mr-a mr-b -n "$NS" --ignore-not-found --grace-period=0 --
force >/dev/null 2>&1
}

cleanup
for P in "mr-a $NODE_A" "mr-b $NODE_B"; do
 set -- $P
 cat <<YAML | run apply -f - >/dev/null
apiVersion: v1
kind: Pod
metadata: { name: $1, namespace: $NS, annotations: { k8s.v1.cni.cncf.io/net
works: "$NETS" } }
spec:
 nodeName: $2
 restartPolicy: Never
 containers:
 - name: t
 image: $IMG
 command: ["sh", "-c", "sleep 3600"]
 securityContext: { runAsUser: 0, capabilities: { add: ["IPC_LOCK", "NE
T_ADMIN", "NET_RAW"] } }
 resources:
 limits:
$RESOURCE_LIMITS
YAML
done
echo "--- wait for Running ---"
for i in $(seq 1 24); do
 a=$(run get pod mr-a -n "$NS" -o jsonpath='{.status.phase}' 2>/dev/null)
 b=$(run get pod mr-b -n "$NS" -o jsonpath='{.status.phase}' 2>/dev/null)
 ["$a" = "Running"] && ["$b" = "Running"] && break; sleep 5
done
run get pods mr-a mr-b -n "$NS" -o wide

echo "=== mr-a secondary interfaces (net1..netN) count ==="
ex mr-a "ls /sys/class/net/ | grep -c net"

echo "=== CROSS-NODE PING on ALL $RAIL_COUNT rails (mr-a -> mr-b) ==="
ASTAT=$(run get pod mr-a -n "$NS" -o jsonpath='{.metadata.annotations.k8s\
.v1\.cni\.cncf\.io/network-status}')
BSTAT=$(run get pod mr-b -n "$NS" -o jsonpath='{.metadata.annotations.k8s\
.v1\.cni\.cncf\.io/network-status}')
for n in $(seq 0 $((RAIL_COUNT-1))); do
 # Pull mr-b's IP on rail $n out of its Multus network-status annotation.
 bip=$(echo "$BSTAT" | grep -oE "${RAIL_IP_PREFIX}\.${n}\.[0-9]+" | head
-1)
 res=$(ex mr-a "ping -c 2 -w 6 $bip >/dev/null 2>&1 && echo OK || echo FAI
L")
 echo "rail${n} -> ${bip}: ${res}"
done
```

**cleanup**

```
Example successful OUTPUT
NAME READY STATUS RESTARTS AGE IP NODE
mr-a 1/1 Running 0 6s 10.42.1.145 <node-0>
mr-b 1/1 Running 0 6s 10.42.2.121 <node-1>
=== mr-a secondary interfaces (net0..net7) count ===
8
=== CROSS-NODE PING on ALL 8 rails (mr-a -> mr-b) ===
rail0 -> 10.0.0.81: OK
rail1 -> 10.0.1.81: OK
rail2 -> 10.0.2.81: OK
rail3 -> 10.0.3.81: OK
rail4 -> 10.0.4.81: OK
rail5 -> 10.0.5.81: OK
rail6 -> 10.0.6.81: OK
rail7 -> 10.0.7.81: OK
```

## Troubleshooting GPU and Network operators

Review some of the troubleshooting scenarios related to the GPU and Network operators that you might encounter.

### Unresolvable CDI devices

After an HGX node reboot, if the GPU operator Pods fail to start due to the following issue (you can locate it among the Pod events):

```
failed to inject CDI devices: unresolvable CDI devices management.nvidia.com
/gpu=all
```

### Procedure

1. Run the following command on the affected HGX node where the Pods are running:

```
sudo /usr/local/nvidia/toolkit/nvidia-ctk cdi generate \
--vendor=management.nvidia.com \
--device-name-strategy=type-index \
--output=/var/run/cdi/management.nvidia.com-gpu.yaml
```

2. Later delete the affected Pods on the HGX node.

### The /dev/nvidia-uvm is missing

After an HGX node reboot, if the GPU operator Pods are not able to start because /dev/nvidia-uvm is missing, then run the following commands on the affected HGX node.

```
sudo modprobe nvidia
sudo modprobe nvidia-uvm
sudo nvidia-modprobe -u -c=0
```

Later delete the affected Pods on the HGX node.

### Multi-Node GPUDirect RDMA resource creation and memory registration

You might encounter an error or failure scenario during **Multi-Node GPUDirect RDMA** resource creation and memory registration step.

Use the following information to set up the Safety Valve with the given values:

1. From the Cloudera Manager # Containerized Cluster # ECS service # Configuration tab.
2. Input the Safety Valve name: `rke2_agent_systemd_dropin_safety_valve`.
3. Value:

```
[Service]
LimitMEMLOCK=infinity
```

Setting this safety valve will cause configuration staleness and all the Cloudera Embedded Container Service agent roles must be restarted for these changes to take effect.