NVIDIA Enterprise Partners

Deployment Guide for NVIDIA Enterprise Reference Architectures – Upstream Kubernetes

April-2026

NVIDIA Enterprise Reference Architecture Notice & Disclaimers

NVIDIA Enterprise Reference Architecture documentation is NVIDIA confidential information and embodies substantial intellectual property and engineering know-how. NVIDIA grants you limited permissions to use NVIDIA Enterprise Reference Architecture to create NVIDIA platform-enabled data center clusters with NVIDIA technologies. You may not use the NVIDIA Enterprise Reference Architecture to develop competing products or technologies or assisting a third party in such activities. Except as expressly stated in this Notice & Disclaimer statement, no other license or right is granted to you by implication, estoppel or otherwise. NVIDIA objects to and rejects any additional or different terms you may propose.

NVIDIA Enterprise Reference Architecture is subject to change without notice. NVIDIA technologies may require enabled hardware, software, or service activation. Your costs and results may vary. This document is not a commitment to develop, release, or deliver any technology, code, or functionality. NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice. It is your sole responsibility to evaluate and determine the applicability of any information contained in this document and ensure that the product is suitable and fit for the application planned by you. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party or a license from NVIDIA.

NVIDIA products or services are subject to the applicable NVIDIA product or sales terms and conditions that are provided in conjunction with the relevant NVIDIA product or service. NVIDIA's provision of these resources does not expand or otherwise alter NVIDIA's applicable warranties or warranty disclaimers for the NVIDIA products or services.

You will not use the NVIDIA Enterprise Reference Architecture or NVIDIA's confidential information to: (i) identify or support an assertion or potential assertion of any intellectual property rights (including patent, copyright, or trade secret); or (ii) prepare, file, amend, or prosecute any patent applications. You agree to grant NVIDIA a limited, worldwide, nonexclusive, perpetual, irrevocable, non-sublicensable, nontransferable, royalty-free, fully-paid license to any patent claim that you may file after accessing NVIDIA Enterprise Reference Architecture that covers the subject matter disclosed herein.

Warranty Disclaimer. NVIDIA Enterprise Reference Architecture materials disclosed are provided "AS IS". To the maximum extent permitted by applicable law, NVIDIA disclaims all warranties and representations of any kind, whether express, implied, or statutory, relating to or arising under this agreement, including, without limitation, the warranties of title, noninfringement, merchantability, fitness for a particular purpose, usage of trade, and course of dealing.

Limitation of Liability. To the maximum extent permitted by applicable law, in no event will NVIDIA be liable for any (i) indirect, punitive, special, incidental, or consequential damages, or (ii) damages for the (a) cost of procuring substitute goods or (b) loss of profits, revenues, use, data, or goodwill arising out of or related to this document, whether based on breach of contract, tort (including negligence), strict liability, or otherwise, and even if NVIDIA has been advised of the possibility of such damages and even if a your remedies fail their essential purpose. Additionally, to the maximum extent permitted by applicable law, NVIDIA's total cumulative aggregate liability for any and all liabilities, obligations, or claims arising out of or related to this document will not exceed one-hundred U.S. dollars (US$100).

NVIDIA, NVIDIA HGX, NVIDIA DGX SuperPOD, NVIDIA DGX Cloud, NVIDIA ConnectX, NVIDIA BlueField, NVIDIA Spectrum, NVIDIA NVLink, NVIDIA Base Command, NVIDIA CUDA, NVIDIA Magnum IO, NVIDIA NetQ, and the NVIDIA logo are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

© 2025 NVIDIA Corporation & Affiliates. All rights reserved.

Jack Liu

Contents

Overview

This Deployment Guide for NVIDIA Enterprise Reference Architectures provides a step-by-step process to deploy Upstream Kubernetes (K8s) on a cluster running NVIDIA Base Command Manager (BCM) 10.x. Kubernetes is required for scalable AI inference and machine learning workloads in enterprise environments.

While multiple Kubernetes distributions exist, including commercial and community supported options, this guide focuses specifically on upstream Kubernetes, as it is the deployment option available through NVIDIA BCM. The full BCM Deployment Guide is available on the NVIDIA Enterprise Reference Architecture Docs webpage under Deployment.

The following steps support successfully deploying a standard Kubernetes cluster environment by using BCM 10:

- Master nodes as K8s Master
- Compute nodes as K8s Workers
- Ready for application deployment and cluster management
- Extensible with GPU support, private registries, ingress controllers, and more

Prerequisites

Before proceeding with the Kubernetes deployment, ensure the following components and configurations are in place for a seamless setup

- **BCM Cluster**
  - A running BCM 10 cluster (see BCM Deployment Guide)
  - Master and compute nodes configured
  - DNS, time synchronization, and user authentication functional

- **Software & Licenses**
  - Kubernetes integration module provided by BCM 10
  - Valid BCM license
  - Software image and category configured (see BCM Deployment Guide, Section: Create Custom SoftwareImage and Category)

- **Network Environment**
  - Management and internal cluster networks configured
  - Internet access to public image registries (e.g., DockerHub), or a private local registry configured

Installation

This section provides a step-by-step process to successfully install Kubernetes via BCM on a standard cluster based off the Enterprise RA Software BOM:

1. Log into BCM headnode as **root**

```
root@pdx-289-bcm-01:~# cm-kubernetes-setup

Connecting to CMDaemon
```

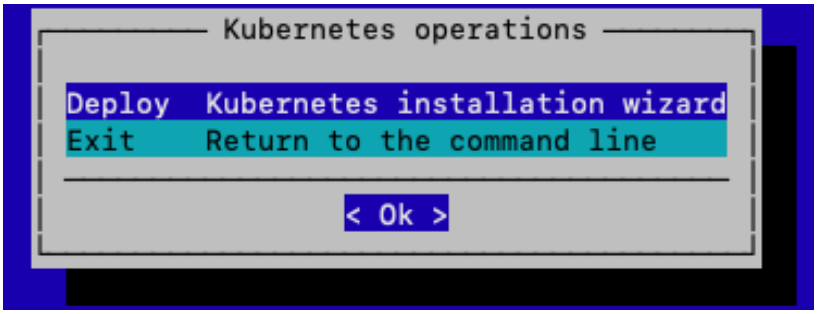
2. Use **cmsh > device > list** to confirm the nodes you want deployed (can be various)

```
[pdx-289-bcm-01->device]% list
Type             Hostname (key)       MAC               Category           IP              Network      Status
---------------- -------------------- ----------------- ------------------ --------------- ------------ ----------
HeadNode         pdx-289-bcm-01       36:02:97:77:0B:38                    10.187.1.252    internalnet  [   UP   ]
HeadNode         pdx-289-bcm-02       46:39:CE:5B:1E:73                    10.187.1.253    internalnet  [   UP   ]
PhysicalNode     cisco-k8s-worker-01  1A:E5:B3:2B:DE:8C  cisco-k8s-worker  10.187.1.181    internalnet  [   UP   ]
PhysicalNode     cisco-k8s-worker-02  4A:BA:43:AF:F7:C0  cisco-k8s-worker  10.187.1.213    internalnet  [   UP   ]
PhysicalNode     dell-k8s-master-01   02:4E:21:06:4B:A9  dell-k8s-master   10.187.1.241    internalnet  [   UP   ]
PhysicalNode     dell-k8s-master-02   8E:0D:6D:77:4A:58  dell-k8s-master   10.187.1.242    internalnet  [   UP   ]
PhysicalNode     dell-k8s-master-03   6A:66:7B:51:1B:F7  dell-k8s-master   10.187.1.243    internalnet  [   UP   ]
PhysicalNode     dell-k8s-worker-01   EA:EA:6A:F8:33:5E  dell-k8s-worker   10.187.1.194    internalnet  [   UP   ]
PhysicalNode     dell-k8s-worker-02   8E:90:43:BE:65:C5  dell-k8s-worker   10.187.1.179    internalnet  [   UP   ]
PhysicalNode     lenovo-k8s-worker-01 F2:6B:32:B4:5B:A2  lenovo-k8s-worker 10.187.1.177    internalnet  [   UP   ]
PhysicalNode     lenovo-k8s-worker-02 36:F4:7C:6C:C6:C0  lenovo-k8s-worker 10.187.1.141    internalnet  [   UP   ]
```

3. Run **cm-kubernetes-setup** command

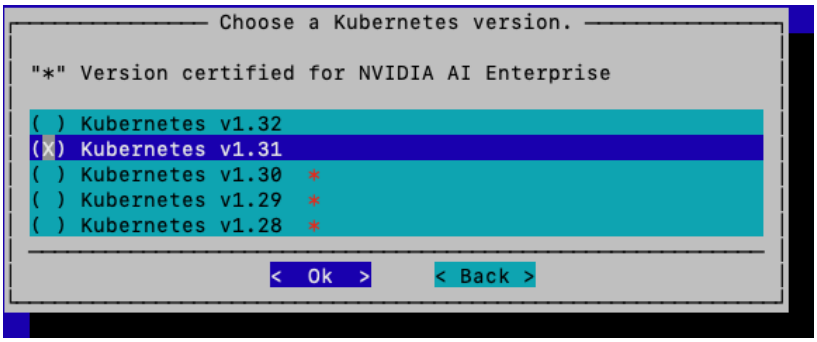
```
root@pdx-289-bcm-01:# cm-kubernetes-setup
```

4. Wizard screen



   a. Select **Deploy**, and click **OK**

5. Choose a target Kubernetes Version

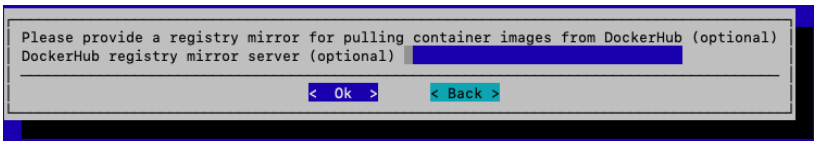


   a. Use the **space** key to check selected version

   b. Click **OK**

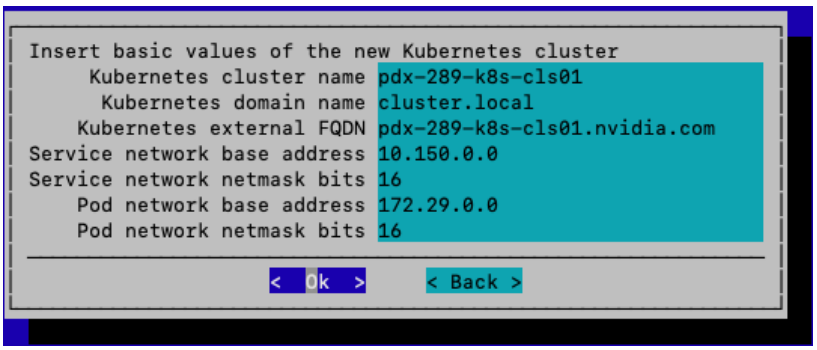
> **Note**: According to the Enterprise RA Software BOM, we select Kubernetes v1.31. The version may differ for your specific deployment. See Appendix A.

6. Type in docker registry mirror server address if needed (optional)



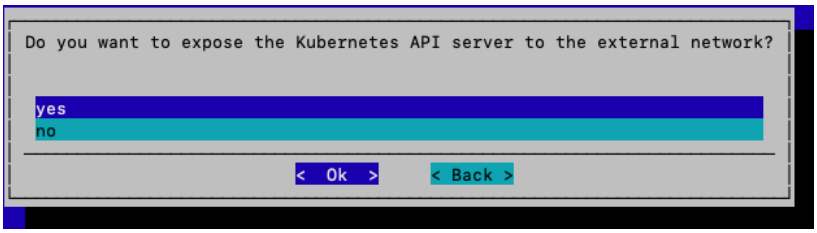
   a. This part can be skipped by clicking **OK**

7. Give a cluster name, domain name, external FQDN and desired IP addresses



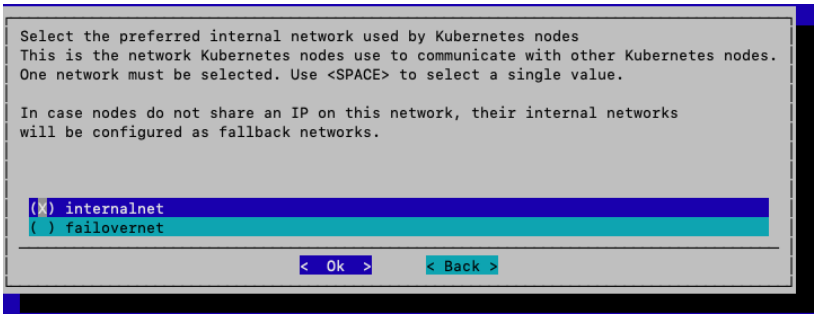
   a. Type in all information required, then click **OK**

8. Expose Kubernetes API server to external network if needed



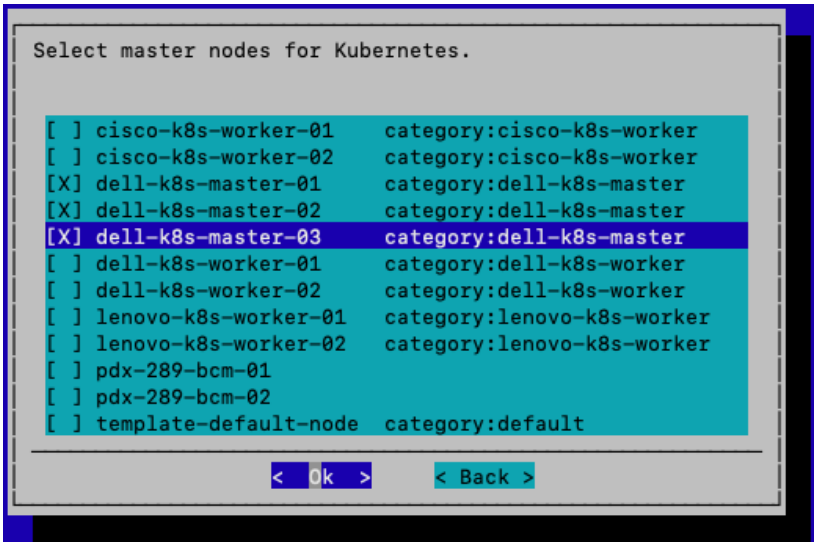
   a. Select **yes**, and click **OK**

9. Select preferred internal network used by Kubernetes nodes



   a. Choose **internalnet**, and click **OK**

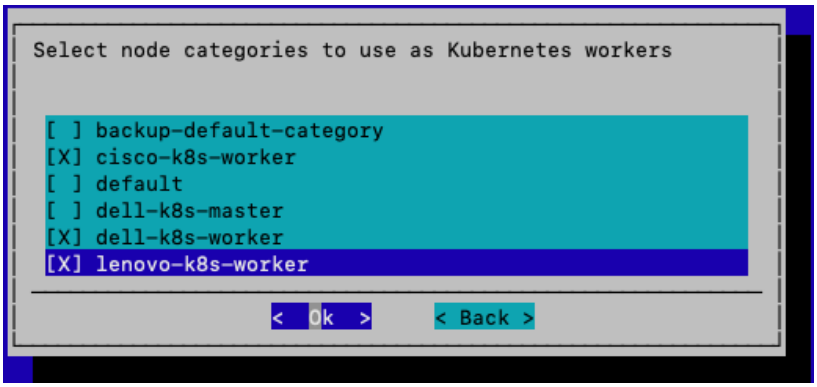
10. Selecting master nodes



   a. Use **space** key to toggle **master** nodes that are already deployed

   b. Click **OK**

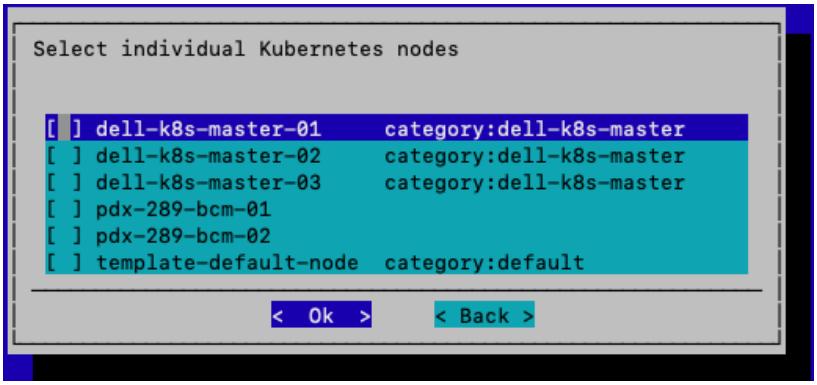
11. Choosing Kubernetes workers categories



a. Choose the category that was previously configured (Prerequisites)

b. Toggle with **space** key, then click **OK**

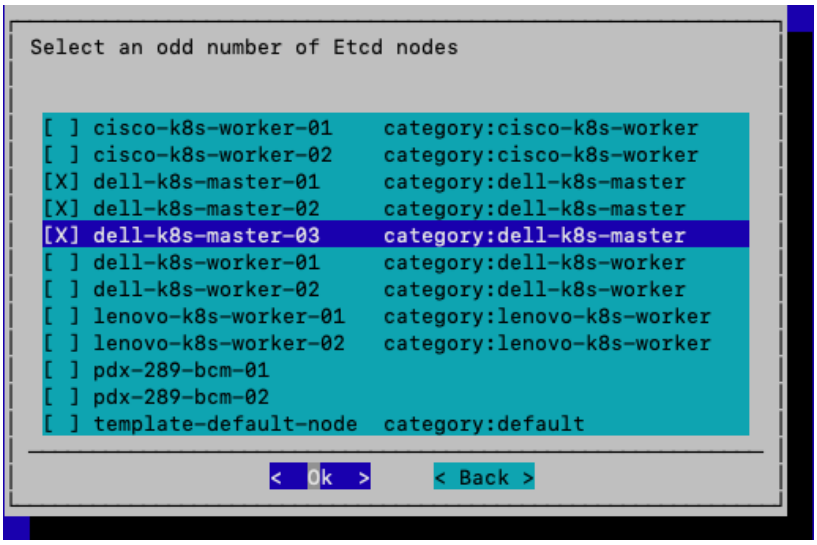
12. Choosing specific nodes if there are any



a. Any specific node included in the worker category can be chosen. In this case, none are available.

b. Click **OK**

13. Choosing Etcd nodes



a. Toggle all three master nodes as Etcd nodes

b. Click **OK**

Note: etcd is the distributed key-value database that stores all Kubernetes cluster state (pods, services, secrets, configs).

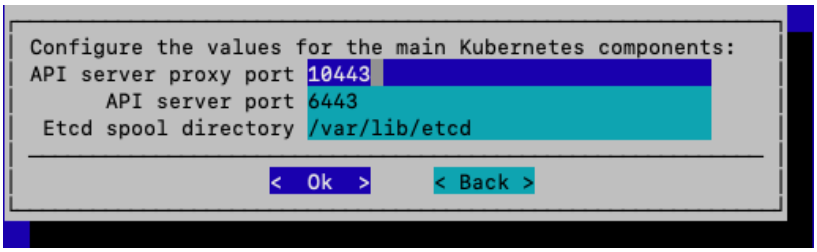
For **Small/medium cluster (≤ 200 worker nodes):**

- Just let etcd run on the **control plane nodes** (usually 3)
- Each control plane node runs both kube-apiserver and etcd

For **Larger cluster (≥ 200+ workers nodes):**

- Create 3–5 **dedicated etcd nodes**
- Deploy 3 or 5 separate nodes only running etcd

14. Configuring API Server ports

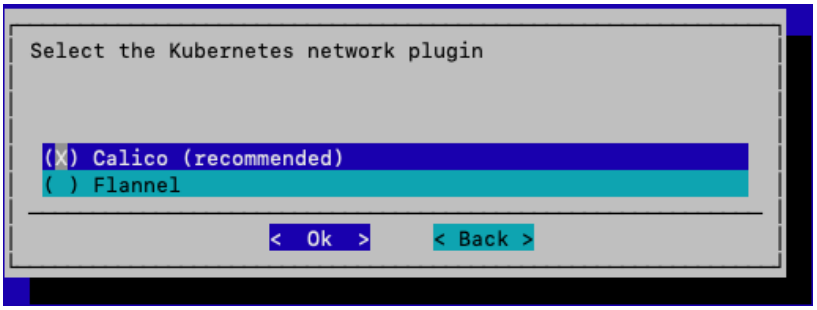


a. There is only one API server in this case

b. Use default ports assigned, click **OK**

Note: If there are multiple Kubernetes clusters, please assign a unique port number for each API server.

15. Choosing CNI (**Container Network Interface**) plugins

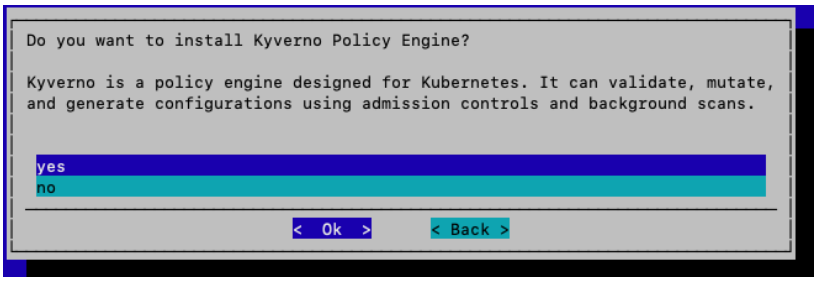


a. Toggle recommended **Calico** as CNI, click **OK**

Note:

- Kubernetes itself doesn't handle pod networking directly — instead it relies on **CNI plugins** to set up networking for pods and services.
- A CNI plugin is responsible for:
  - Assigning IP addresses to pods
  - Setting up routes so pods can talk to each other across nodes
  - Enforcing network policies (firewalls, isolation, etc.)

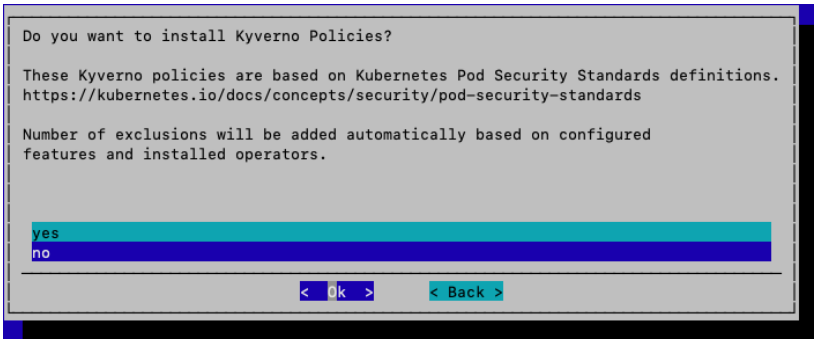
16. Installing Kyverno policy engine



a. Select **yes**, and click **OK**

Note: **Kyverno** is a **Kubernetes-native policy engine**. It enables **policy-as-code** using **YAML**, avoiding the need for new languages

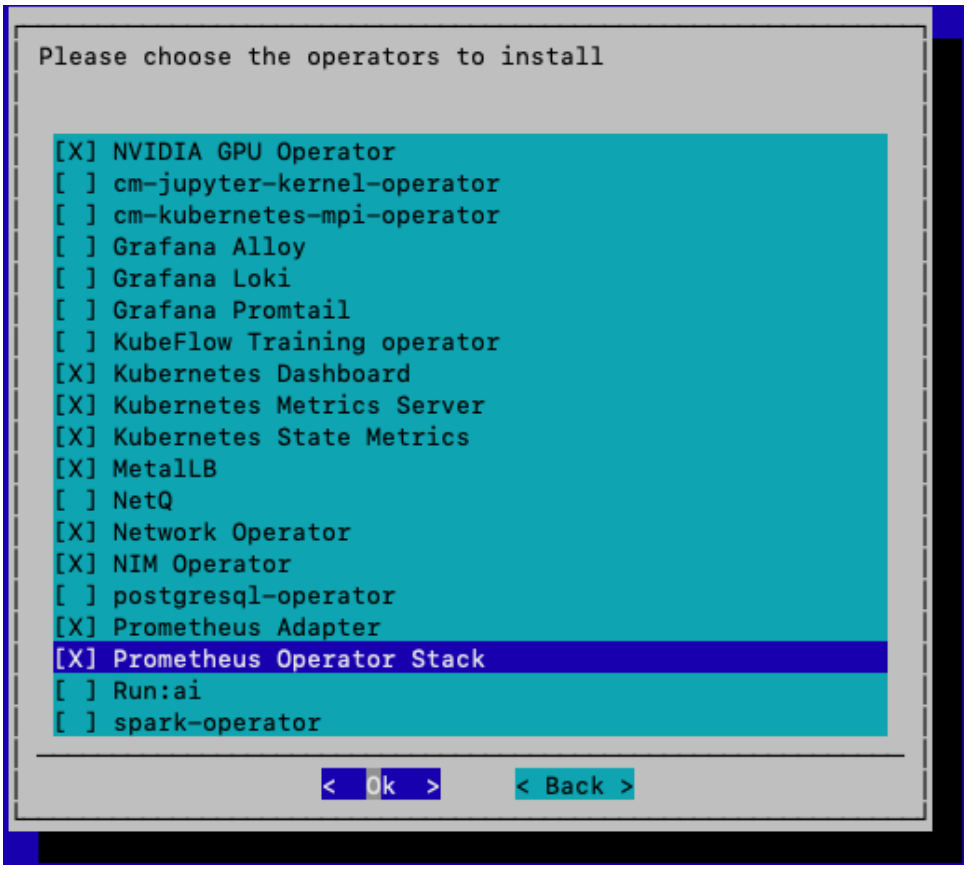
17. Installing Kyverno policies



a. This will be configured later, choose **no**, and click **OK**

Note: The base installation can be performed without policy initialization, with policy Custom Resource Definitions (**CRD**) applied subsequently through standard Kubernetes workflows.

18. Choosing the operators to install



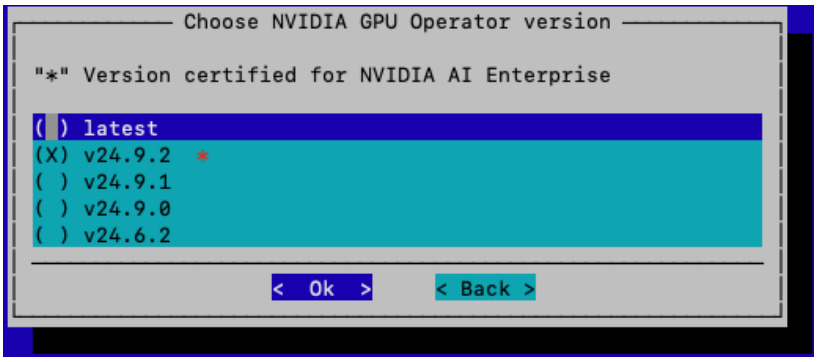
As per the Enterprise RA Software BOM, we install following operators:

- NVIDIA GPU Operator
- Kubernetes Dashboard
- Kubernetes Metric Server
- Kubernetes State Metrics
- MetalLB
- Network Operator
- NIM Operator
- Prometheus Adapter
- Prometheus Operator Stack

a. Toggle selected Operators by **Space** key

b. Click **OK**

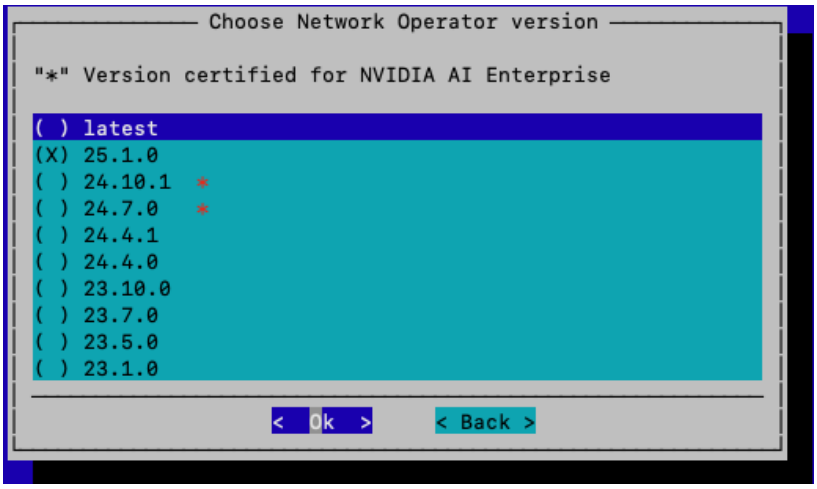
19. Choosing NVIDIA GPU Operator version



According to the Enterprise RA Software BOM, we chose v24.9.2

a. Toggle **v24.9.2**, and click **OK**

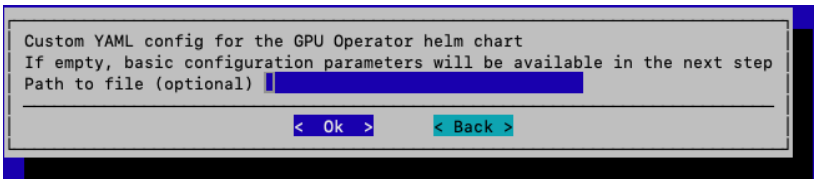
20. Choosing Network Operator version



According to the Enterprise RA Software BOM, version v25.1.0 is selected.

a. Toggle **v25.1.0**, and click **OK**

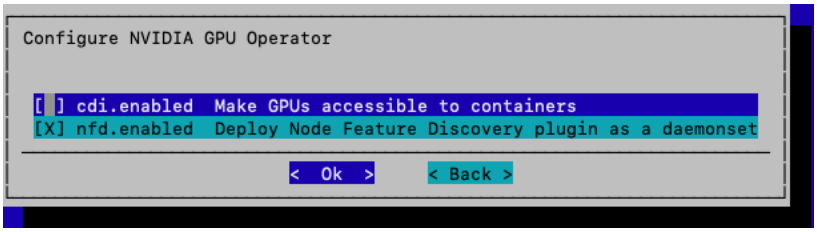
21. Customizing YAML file for GPU Operator



If a custom YAML file is available, this is where the value can be passed to the Helm chart. In this case, standard configuration parameters are used.

a. Leave it empty, and click **OK**

22. Configuring NVIDIA GPU Operator



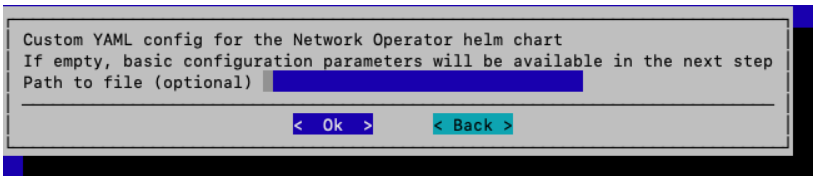
In this case, direct access to the GPU is not required. Only discovering is needed.

a. Toggle **nfd.enabled**

b. Click **OK**

Note: In the special case that direct access to GPU is needed, e.g. upgrading GPU Firmware, then **cdi** needs to be toggled.

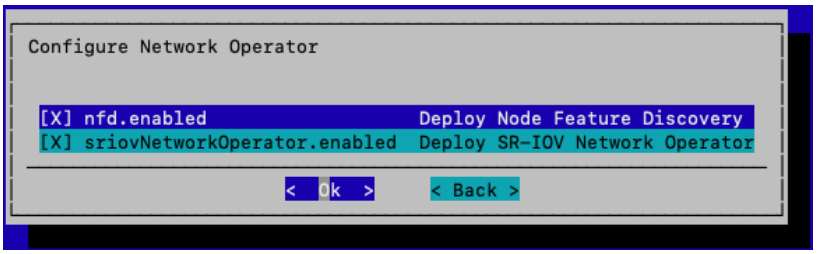
23. Customizing YAML file for Network Operator



If a custom YAML file is available, this is where the value can be passed to the Helm chart. In this case, standard configuration parameters are used.

a. Leave it empty, and click **OK**

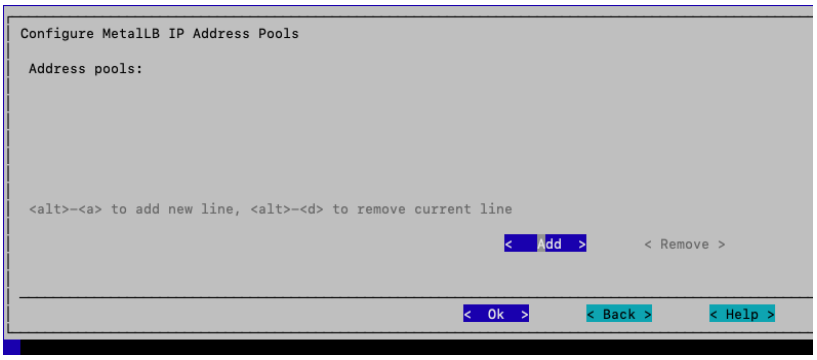
24. Configuring Network Operator



Both options need to be enabled: network discovery is required, and SR-IOV is necessary to perform RDMA and related functions.

    a. Toggle **both**, and click **OK**

25. Creating a MetalLB IP address Pools



    a. Click **Add**, and click **OK**

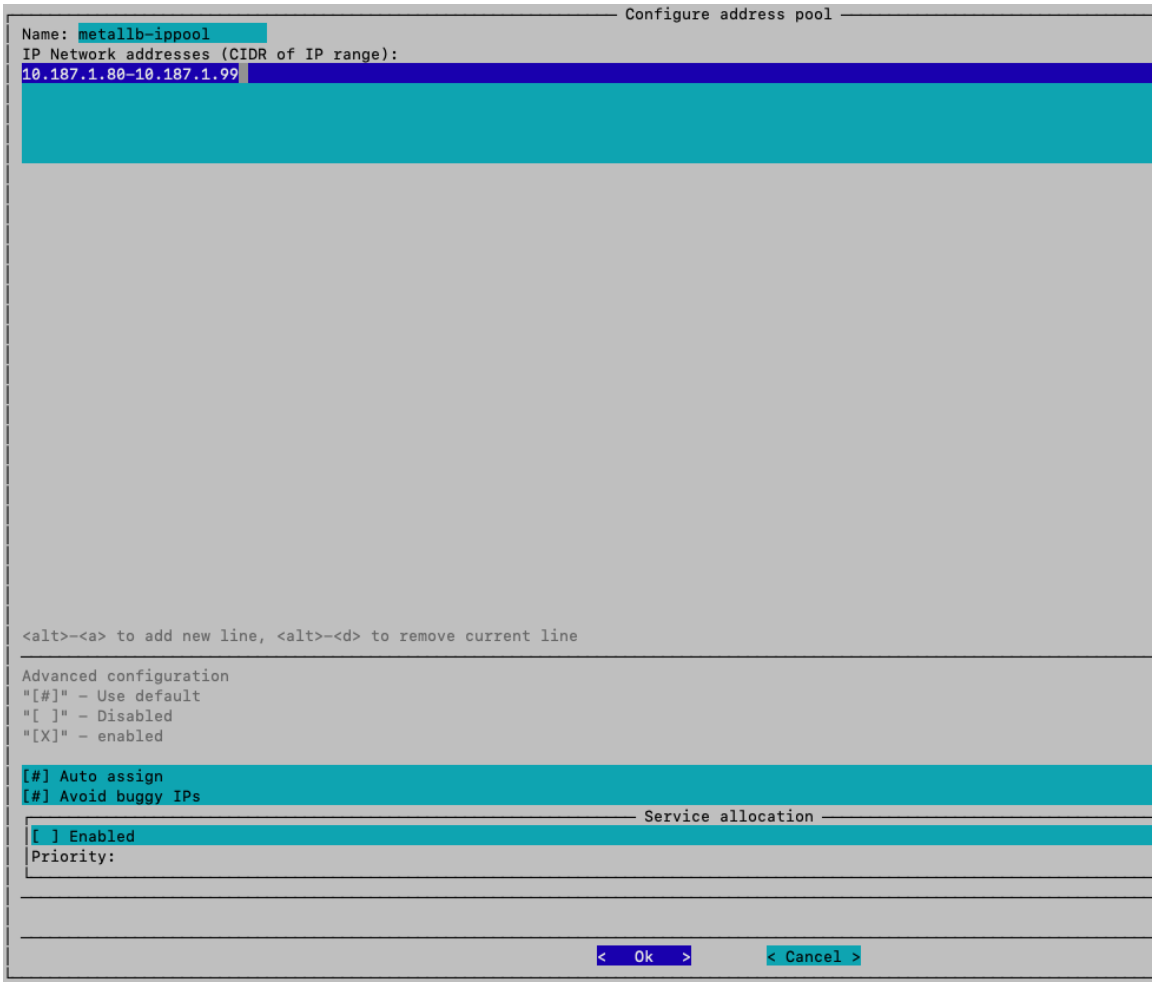
Note: **MetalLB** is a **load-balancer** implementation for bare-metal Kubernetes clusters.

- In cloud environments (AWS, GCP, Azure), Kubernetes Services of type LoadBalancer automatically fetch a cloud provider's load balancer.
- On **bare-metal or self-hosted clusters**, there's no built-in load balancer. That's where **MetalLB** comes in — it provides a way to assign external IPs to services and handle traffic routing.

Key Features:

1. **Implements Service type LoadBalancer**
   - Exposes a Service to the outside world with an external IP.
2. **Two operating modes**:
   - **ARP / Layer 2 mode**
     - Simple: one node advertises the external IP using ARP/NDP.
     - Easy to set up but doesn't support advanced routing.
   - **BGP mode (Layer 3)**
     - MetalLB peers with the network routers using **BGP**.
     - Scales better, provides true load balancing across nodes, integrates into enterprise networks.
3. **Supports address pools**
   - It is possible to configure IP ranges (like 192.168.1.240-192.168.1.250) that MetalLB assigns to Services.
4. **Works with standard Kubernetes resources**
   - No need for special CRDs for basic use — it extends existing Service semantics.

26. Configuring MetalLB IP address pool

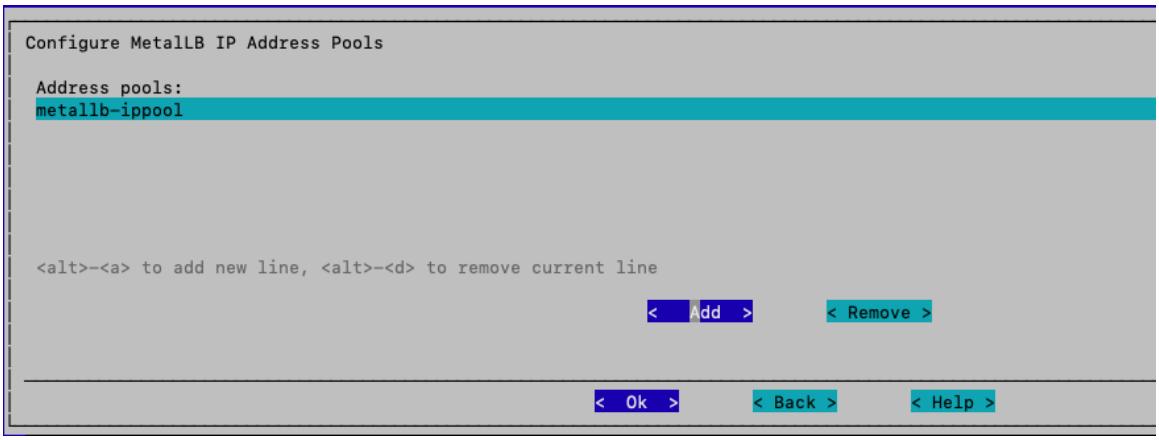


a. Type in a **name** for the ip address pool

b. Type in an **ip range** that was reserved for MetalLB

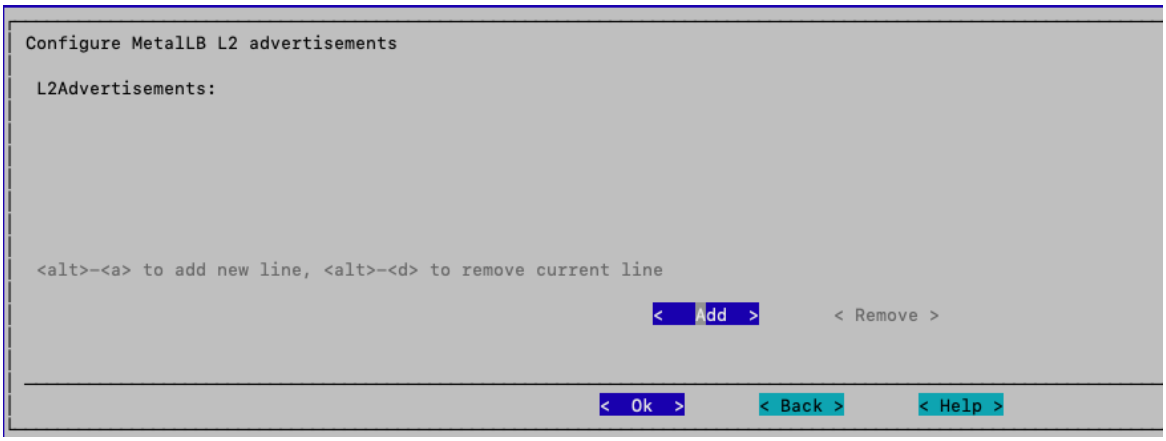
c. Click **OK**

27. Confirming the IP address pool created



a. The name of IP address pool should appear here and click **OK**

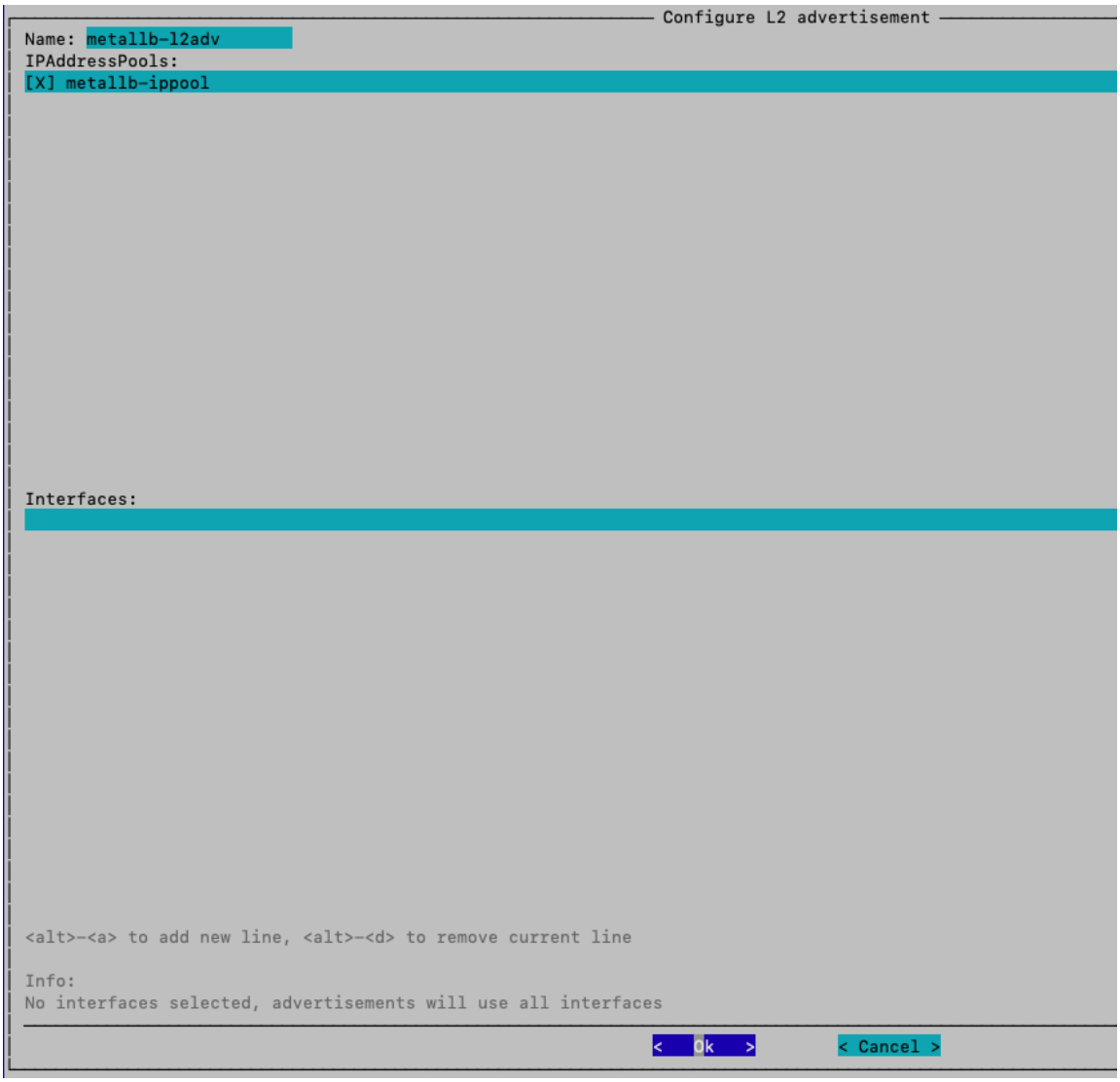
28. Creating MetalLB L2 advertisements



a. Click **Add**, and click **OK**

	Note: An L2Advertisement is not typically required in this scenario. If a dedicated service network is being used, then L2Advertisement becomes necessary. In this case, since both the Kubernetes control plane and the services are running on the same network, explicit advertisement is not required. However, it is being configured as a best practice.

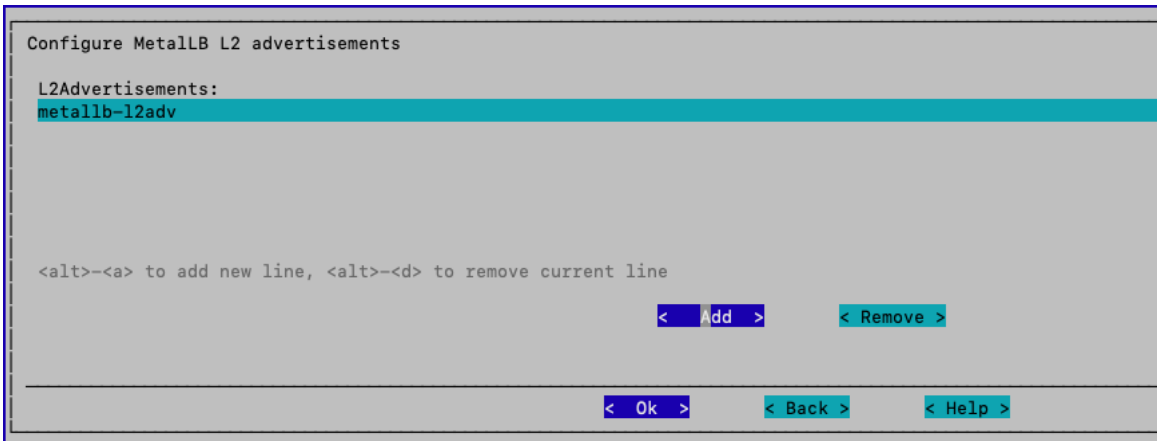
29. Configuring MetaLB L2 advertisements



a. Type in a **name** for L2 advertisements

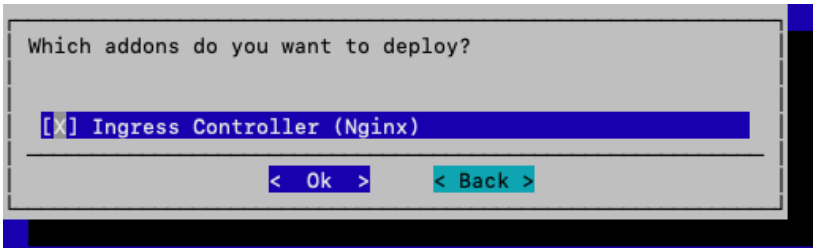
b. Toggle the IP address pool that was previously created, and click **OK**

30. Confirming the L2 advertisements



a. The name of L2 advertisements should be appeared, and click **OK**

31. Installing Ingress Controller



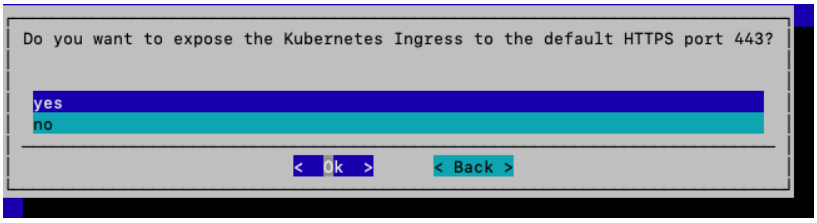
Instead of directly exposing NGINX via Service, use **Ingress resources** with the NGINX Ingress Controller for routing multiple apps behind one IP/hostname.

a. Toggle it, and click **OK**

Note:

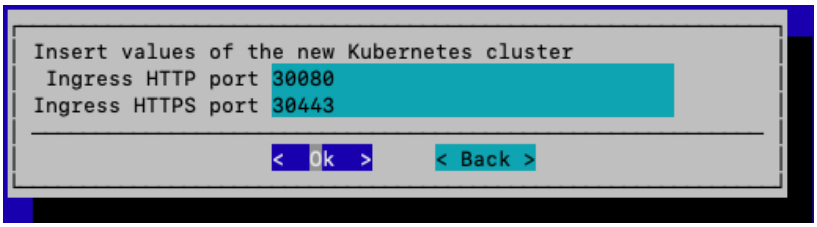
- NGINX **Service** = exposes NGINX pods internally/externally.
- NGINX **Ingress Controller** = smart entrypoint that routes multiple domains/paths to different services.
- In production environment, it is best to almost always use **Ingress** instead of exposing each Service separately.

Exposing Kubernetes Ingress to default HTTPS port



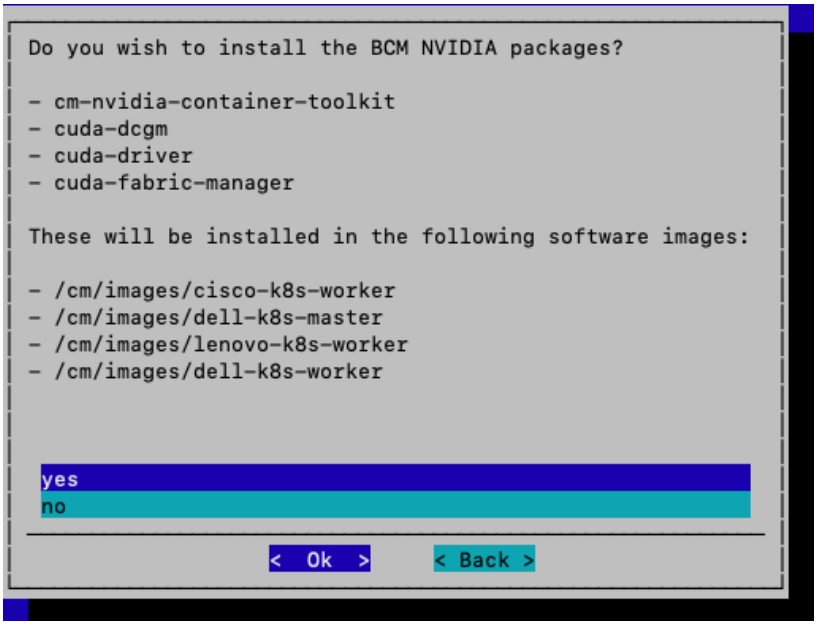
a. Choose **yes**, and click **OK**

33. Configuring Ingress port for Kubernetes Cluster



a. Keep **default** values, and click **OK**

34. Installing BCM NVIDIA Packages

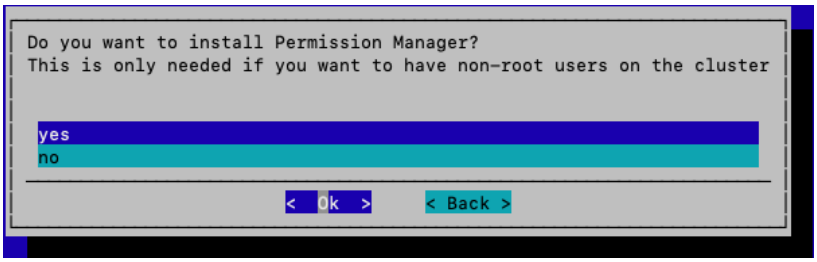


Not all packages are needed. However, container-toolkit runtime is required for the GPU (worker) nodes.

a. Choose **yes**, and click **OK**

Note: On newer version of BCM (BCM 11.x), it is possible to pick and choose packages individually.

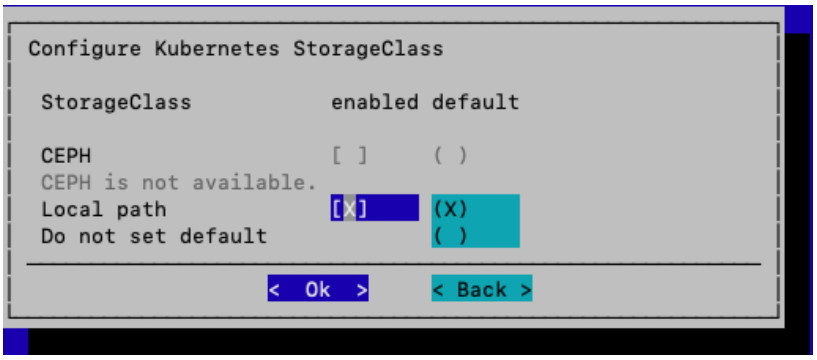
35. Installing Permission Manager



This is necessary for creating users, manage users, groups, and ServiceAccounts, it's also useful for shared clusters among multiple teams.

a. Choose **yes**, and click **OK**

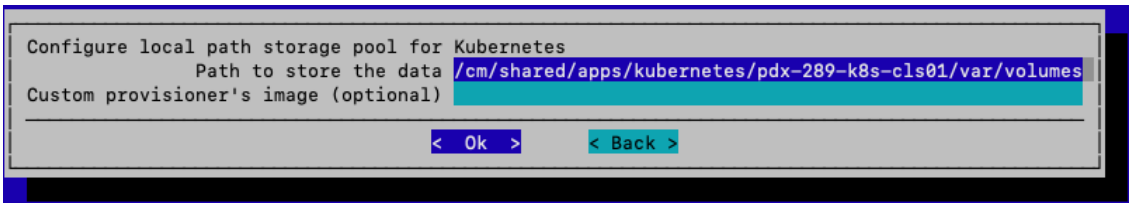
36. Configuring Kubernetes StorageClass



In this case, there aren't any other StorageClass on the local path.

a. Toggle local path, and click **OK**

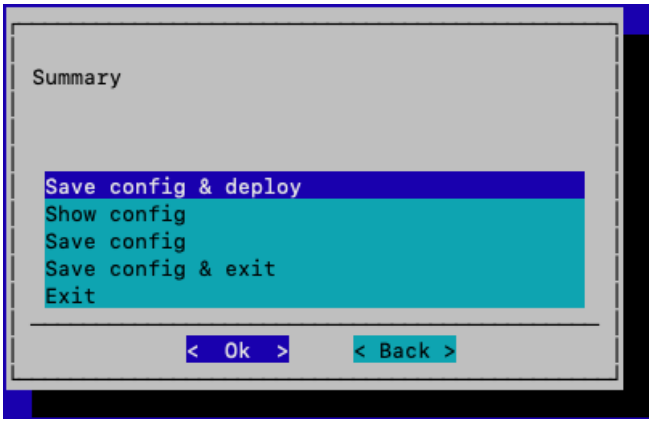
37. Configuring local path storage pool for Kubernetes



In this case, use default value.

a. Click **OK**

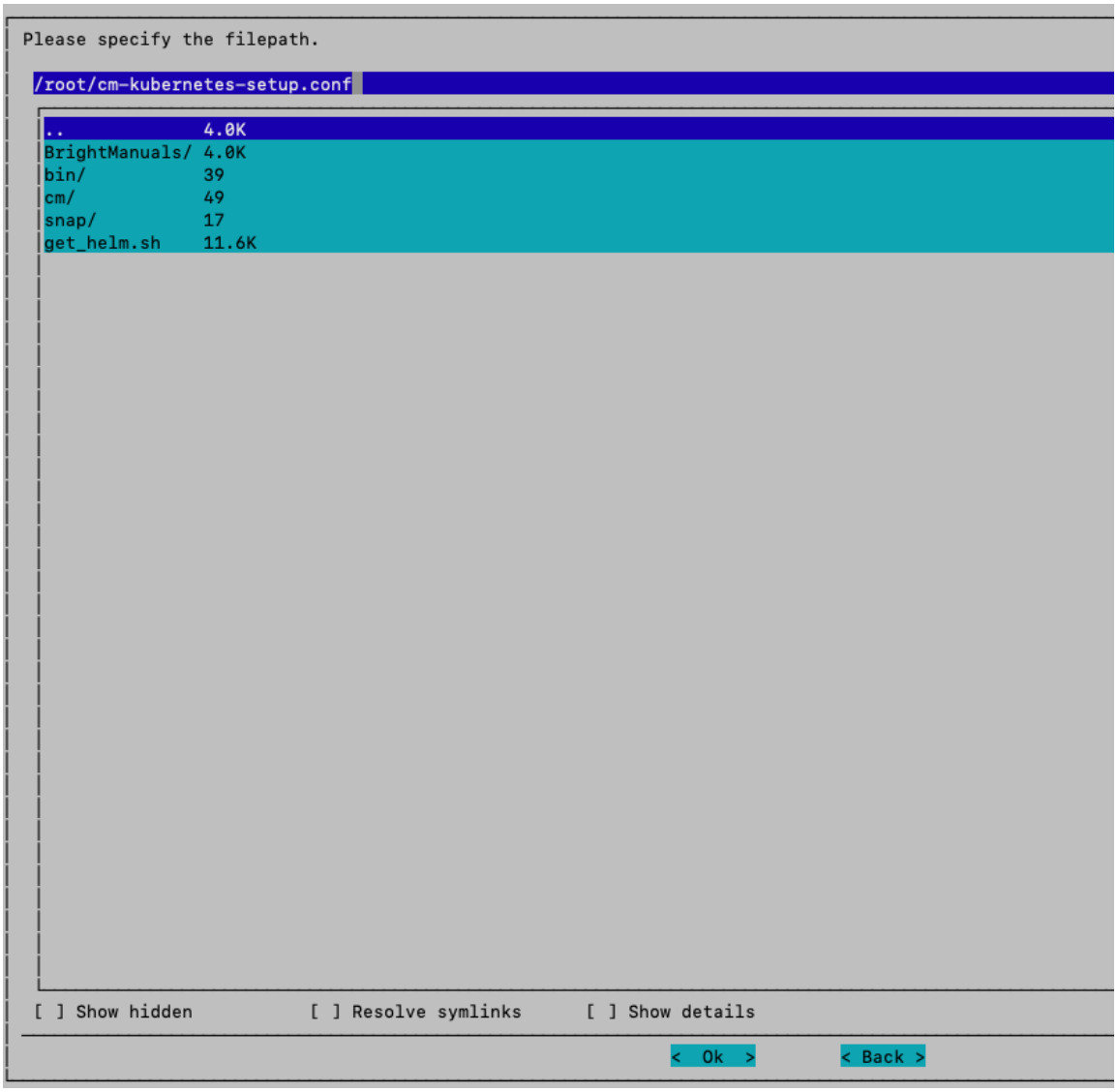
38. Summary



a. Toggle **Save config & deploy**

b. Click **OK**

39. Saving the config file



A .conf file will be saved into the root folder.

a. **Name** the config file

b. Click **OK**

40. Rebooting

```
The reset process does not clean your kubeconfig files and you must remove them manually.
Please, check the contents of the $HOME/.kube/config file.

## Progress: 50
#### stage: kubernetes: Find Files On Headnodes
#### stage: kubernetes: Remove Files On Headnodes
## Progress: 51
#### stage: kubernetes: Create Etcd CA
Sending request to recreate the CA to cmd on pdx-289-bcm-cls01
## Progress: 53
#### stage: kubernetes: Collection Update Provisioners
#### stage: kubernetes: Collection Images Updater
## Progress: 54
#### stage: kubernetes: Apply Sysctl Config
## Progress: 55
#### stage: kubernetes: Collection Nodes Reboot
cisco-k8s-worker-01: reboot requested
cisco-k8s-worker-02: reboot requested
dell-k8s-master-01: reboot requested
dell-k8s-master-02: reboot requested
dell-k8s-master-03: reboot requested
dell-k8s-worker-01: reboot requested
dell-k8s-worker-02: reboot requested
lenovo-k8s-worker-01: reboot requested
lenovo-k8s-worker-02: reboot requested
Press ctrl+c to abort waiting and continue with deployment
Waiting for nodes to start reboot
Going to wait up to 15 minutes for the nodes to come back up.
```

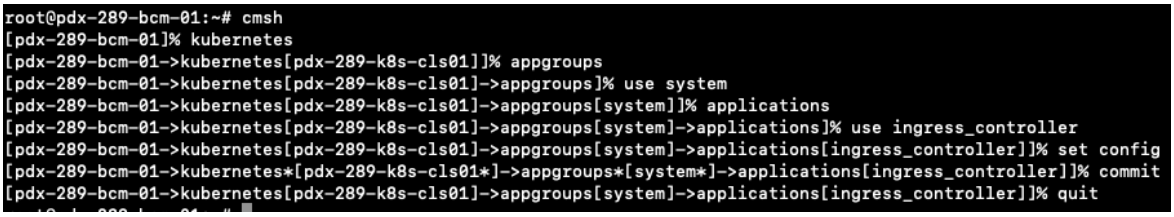
All master nodes and worker nodes will be deployed and rebooted automatically.

Note: If there is an error during deployment, a command line will pop up and ask what to do. Try Retry “r” first, sometimes it’s just a connection issue to pull the packages, if it doesn’t resolve it, enter “d” for Debugging to see where the issue is.

```
Undo/Abort/Skip/Retry/Info/Debug/Remote debug: u/a/s/r/i/d/D:
```

41. Deploying MetalLB Load Balancer service

The cm-kubernetes-setup wizard sets up the Ingress Controller to operate as a NodePort service. After deployment, deploy a MetalLB LoadBalancer service to make the Ingress Controller accessible externally. This assigns an external IP address, enabling traffic from outside the Kubernetes cluster to reach the Controller.



a. Log in BCM head node

b. Type in **cmsh**,

c. Inside cmsh, type in **kubernetes**, then **appgroups**

d. Inside **appgroups**, type in **use system**

e. Inside **system**, type in **applications**

f. Inside **applications**, type in **use ingress_controller**

g. Inside ingress_controller, type in **set config**

h. Looking for section of port type: **NodePort**, change it to **LoadBalancer**

i. **Save** the changes**, commit** and **quit**

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/instance: ingress-nginx
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/part-of: ingress-nginx
    app.kubernetes.io/version: 1.12.1
  name: ingress-nginx-controller
  namespace: ingress-nginx
spec:
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - appProtocol: http
    name: http
    port: 80
    nodePort: ${CM_KUBE_INGRESS_HTTP_PORT}
    protocol: TCP
    targetPort: http
  - appProtocol: https
    name: https
    port: 443
    nodePort: ${CM_KUBE_INGRESS_HTTPS_PORT}
    protocol: TCP
    targetPort: https
  selector:
    app.kubernetes.io/component: controller
    app.kubernetes.io/instance: ingress-nginx
    app.kubernetes.io/name: ingress-nginx
  type: LoadBalancer
---
```

To verify changes, ingress controller nginx service external-IP will changes from “none” to the first IP address that was previously set in the MetalLB address pool. The pod IP address will change from internal IP to external IP.

j. Type **kubectl get svc -n ingress-nginx** to check ingress controller nginx service

```
root@pdx-289-bcm-01:~# kubectl get svc -n ingress-nginx
NAME                                 TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)
ingress-nginx-controller             LoadBalancer   10.150.4.4       10.187.1.80   80:30080/TCP,443:30443/TCP
ingress-nginx-controller-admission   ClusterIP      10.150.123.149   <none>        443/TCP
```

k. Or type **kubectl get ingress -A** to check pod IP address

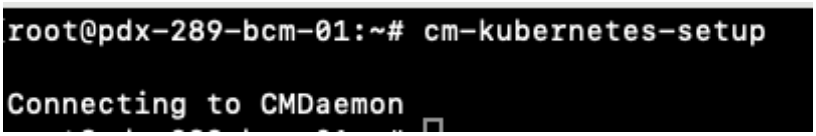
```
root@pdx-289-bcm-01:~# kubectl get ingress -A
NAMESPACE              NAME                   CLASS    HOSTS                                        ADDRESS       PORTS
kubernetes-dashboard   kube-dashboard         <none>   *                                            10.187.1.80   80
kubernetes-dashboard   kubernetes-dashboard   <none>   dashboard.pdx-289-k8s-cls01.nvidia.com       10.187.1.80   80
prometheus             monitoring-ingress     nginx    *                                            10.187.1.80   80
prometheus             prometheus-grafana     <none>   *                                            10.187.1.80   80
```

Uninstallation

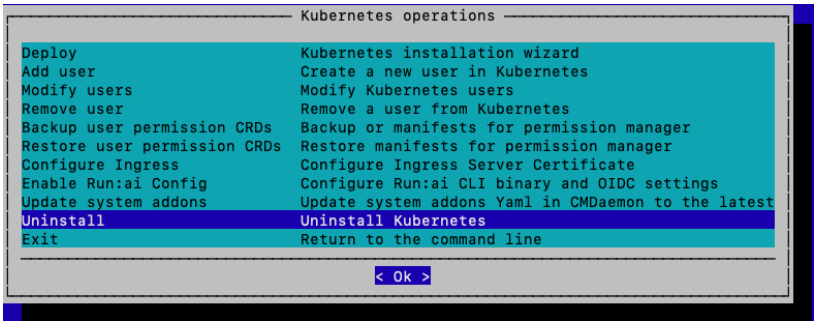
This section describes how to uninstall the previous setup, in the case that some of the components are broken; or a fresh re-install is needed.

1. Run **cm-kubernetes-setup** command as root user on BCM headnode

```
root@pdx-289-bcm-01:# cm-kubernetes-setup
```

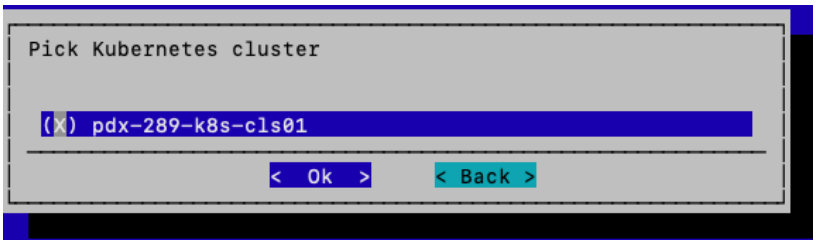


2. Wizard screen



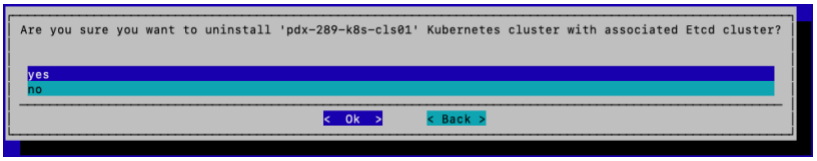
   a. Select **Uninstall**, and click **OK**

3. Choosing the cluster



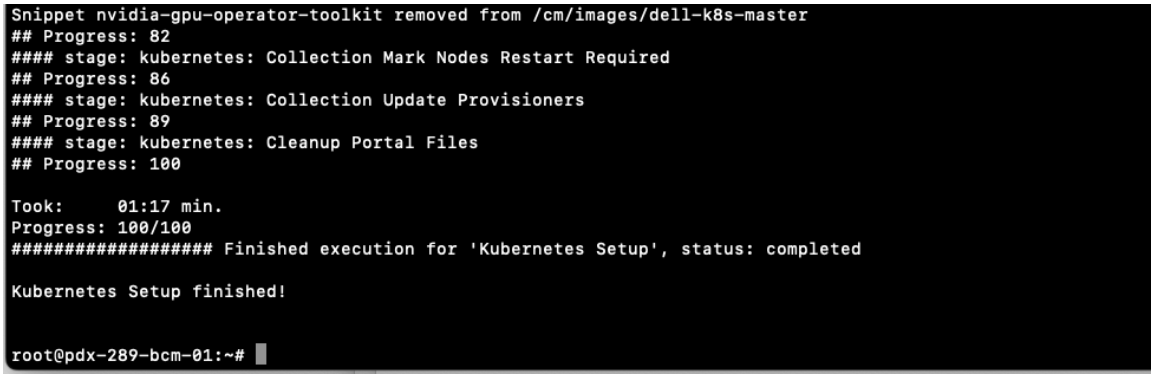
   a. Toggle the cluster we want to uninstall, and click **OK**

4. Confirming the cluster



   a. Toggle **yes**, and click **OK**

5. Uninstall finished



6. Rebooting the nodes

```
 cisco-k8s-worker   10.187.1.232    internalnet     [    UP    ], restart required (K+
 cisco-k8s-worker   10.187.1.210    internalnet     [    UP    ], restart required (K+
 dell-k8s-master    10.187.1.241    internalnet     [    UP    ], restart required (K+
 dell-k8s-master    10.187.1.242    internalnet     [    UP    ], health check failed+
 dell-k8s-master    10.187.1.243    internalnet     [    UP    ], restart required (K+
 dell-k8s-worker    10.187.1.224    internalnet     [    UP    ], restart required (K+
 dell-k8s-worker    10.187.1.175    internalnet     [    UP    ], restart required (K+
 lenovo-k8s-worker  10.187.1.125    internalnet     [    UP    ], restart required (K+
 lenovo-k8s-worker  10.187.1.193    internalnet     [    UP    ], restart required (K+
```

```
]% reboot -c dell-k8s-master
 dell-k8s-master-01..dell-k8s-master-03
]% reboot -c dell-k8s-master
5 [notice] pdx-289-bcm-01: dell-k8s-master-01 [   DOWN   ], pingable, restart required (Kubernetes cleanup)
5 [notice] pdx-289-bcm-01: dell-k8s-master-03 [   DOWN   ], pingable, restart required (Kubernetes cleanup)
5 [notice] pdx-289-bcm-01: dell-k8s-master-02 [   DOWN   ], health check failed, pingable, restart required
```

Summary

The document describes step by step process require to deploy Upstream Kubernetes (K8s) on a cluster running Base Command Manager (BCM) 10. The instruction provided covers cluster preparation, control plane and worker node selection, networking configuration, and add-on setup. By following the wizard, a fully functional Kubernetes cluster is provisioned and integrated into BCM with minimal manual intervention. The guide serves as a reference for quickly reproducing a standard Kubernetes environment within existing BCM deployments.

Note: For more detailed setup guide for Kubernetes deployment, refer to NVIDIA Base Command Manager Containerization Manual.

Appendix

Enterprise RA Software BOM

Category	Product	Version
Operating System	Ubuntu	22.04.04 (AMD64)
Orchestration	Base Command Manager	10.0 (10.25.03)
Container Runtime	Container	1.7.23
Container Orchestration	Kubernetes	1.31.5
Container Network Interface (CNI)	Calico	3.27.4
Load Balancer - API Service Gateway	Nginx	1.18.0
Load Balancer – Network Services	MetalLB	0.14.8
Ingress Controller	Nginx	1.12.0
Package Manager	Helm	3.16.0
Operator	GPU Operator	24.9.2
Operator	Network Operator	25.1.0
Operator	NIM Operator	1.0.1
RBAC	Permission Manager	0.5.1
Observability	Prometheus	2.55.0
Observability	Grafana	11.2.2
Storage	NFS Provisioner	4.0.2