REFERENCE ARCHITECTURE

Boosting Data Scientist Productivity with NVIDIA Run:ai, Pure Storage Portworx and FlashBlade

Contents

Executive Summary

Machine learning (ML) and artificial intelligence (AI) are revolutionizing industries by enhancing efficiency, streamlining operations, and unlocking new growth opportunities. However, the technical complexity of managing massive data sets, training sophisticated models, and addressing high-throughput data requires extensive resource management on complex AI infrastructure. Successful deployment of modern ML and GenAI applications must address some of today's most common resource management challenges.

Utilization

Problem: Static allocation of GPUs can lead to resource wastage, as GPUs may remain idle when not in use.

Fairness

Problem: Unbalanced access to GPU resources can lead to inefficient distribution, where some users or workloads monopolize resources while others face shortages.

Efficiency

Problem: Low GPU utilization rates, typically ranging between 25-45%, can lead to increased costs and wasted capacity.

Scalability

Problem: Manual scheduling and orchestration methods are not sustainable as workload demands increase, making it difficult to manage GPU and storage allocations effectively across large-scale environments.

Together, these challenges result in increased operational costs and prolonged training times that slow down AI development cycles. To address these issues, Pure Storage and Run:ai offers an integrated solution. Pure Storage FlashBlade delivers high-performance, scalable storage designed for modern AI and ML workloads, ensuring seamless data access and accelerated processing speeds. Portworx from Pure Storage enables simple statefulness and management for containers. It provides robust, scalable, and highly available storage and data management for Kubernetes with automated operations, dynamic provisioning, snapshots, cloning, and cross-node/cluster replication. Run:ai dynamically allocates GPUs based on job priority, ensures efficient scheduling of workloads, and prevents resource conflicts in multi-tenant environments.

Solution Overview

In today's competitive landscape, having a deep understanding of the AI tech stack is critical. The AI tech stack is a structural framework made up of independent layers, each covering a critical function for system efficiency and effectiveness. This structured architecture includes the various hardware, software, and services required to develop, deploy, and scale AI applications.

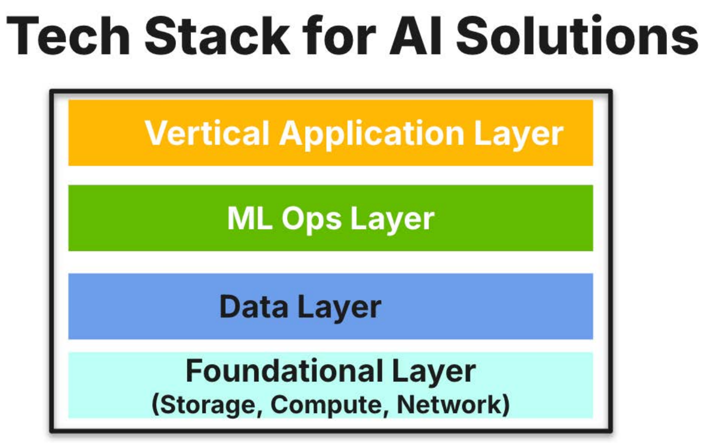

Tech Stack for AI Solutions
Vertical Application Layer
ML Ops Layer
Data Layer
Foundational Layer
(Storage, Compute, Network)

FIGURE 1 Figure 1. Tech stack for AI solutions

Each layer in the stack represents a critical function required to build and operate AI models efficiently.

1. **Foundational layer**: This is the base layer for AI infrastructure which has storage systems, computational resources (GPUs, CPUs, TPUs), and networking solutions to ensure smooth data transfer and model execution. This is where Run:ai can play a role in effectively managing GPU resources.
2. **Data layer**: This layer focuses on data management and orchestration, ensuring that raw data is ingested, cleaned, labeled, and stored properly before being used in ML workflows.
3. **MLOps layer**: This layer focuses on model and experiment tracking, model evaluation, and deployment of production models.
4. **Vertical Application layer**: This layer represents AI applications and industry-specific full-stack solutions. This finishes the structure, where all the bricks come together to create a functional AI-powered application.

Where Run:ai Fits in AI Stack Providers

The AI ecosystem refers to the tools, technologies, and infrastructure required to build, deploy, and manage AI and ML workloads. It includes multiple layers from MLOps to CHIP/Network Fabric to ensure that AI models are efficiently developed, scaled, and maintained.

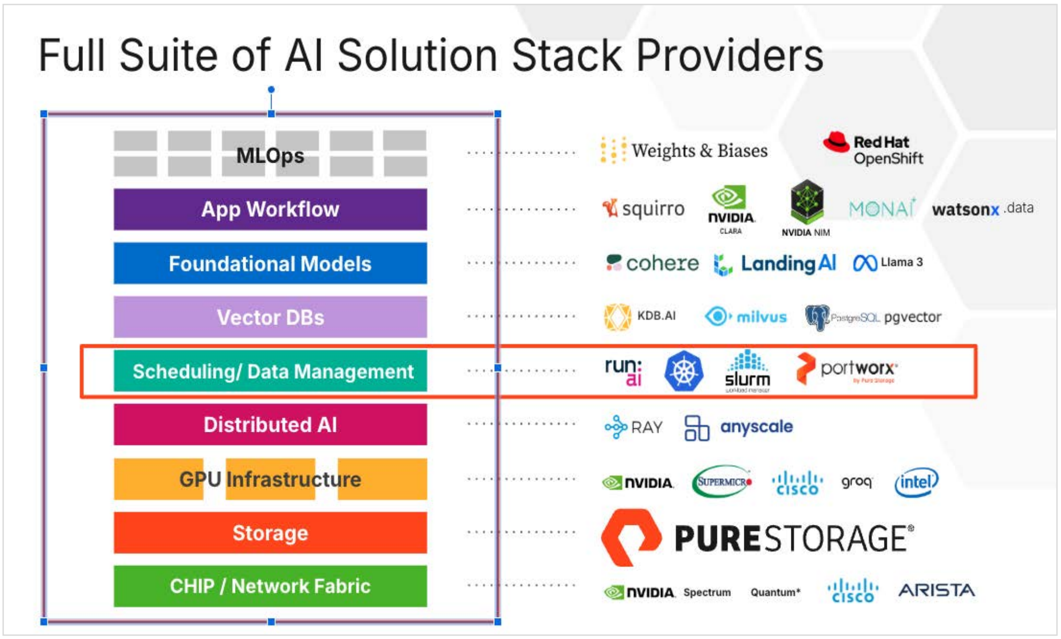


FIGURE 2 Figure 2. Full suite of AI solution stack providers

Technology Overview

The integrated solution was designed with the capability to resolve past challenges effectively. The main components of the solution are:

- **NVIDIA GPU/networking**: Integrating NVIDIA GPUs with NVIDIA networking switches enables a high-performance AI infrastructure by combining powerful compute capabilities with ultra-low latency data transfers. This seamless integration ensures optimal GPU utilization, preventing data bottlenecks and significantly accelerating AI training and inference workloads.
- **Pure Storage FlashBlade**: Delivers high-performance, scalable storage to handle massive datasets efficiently, enabling seamless data access and rapid processing.
- **Run:ai**: Provides intelligent resource management and orchestration, maximizing GPU utilization and ensuring optimal workload distribution. In collaboration with NVIDIA, Run:ai has developed solutions that leverage advanced networking technologies to enhance AI workload performance.
- **Portworx**: Delivers intelligent, enterprise-grade Kubernetes storage and data management with automated operations, data mobility, and high availability to optimize performance and scalability for AI workloads like retrieval-augmented generation (RAG) and inference.
- **BCM**: NVIDIA Base Command Manager (BCM) streamlines the deployment, management, and monitoring of AI infrastructure, enabling organizations to efficiently orchestrate and scale AI workloads across clusters of any size, from a few nodes to thousands.

Together, these components create a powerful, on-premises platform—deployed with Kubernetes—that integrates storage, compute, and experiment management to drive innovation and efficiency in AI workflows.

Benefits of Run:ai, NVIDIA Certified Storage & Portworx in Training and Inference

Training: Run:ai revolutionizes AI training by offering dynamic GPU management, automated orchestration, and cost-efficient resource allocation, making it an essential tool for enterprises looking to scale their AI workloads. Integrating NVIDIA DGX SuperPOD with Run:ai creates a powerful AI training environment by combining high-performance computing with intelligent workload orchestration and job scheduling. SuperPOD provides scalable, multi-node GPU clusters, ideal for training large AI models, while Run:ai optimizes GPU utilization, job scheduling, and multi-tenant resource allocation. This combination ensures dynamic GPU sharing, automated workload distribution, and seamless scaling across training jobs, significantly accelerating AI model development while maximizing infrastructure efficiency. Portworx reduces training time by ensuring high-speed access to large datasets through optimized I/O paths and data locality features. By storing training data close to GPU/CPU resources, latency is minimized, enabling faster iteration cycles for ML engineers. For example, Portworx's Volume Placement Strategy (VPS) ensures pods are scheduled on nodes with local data, cutting data retrieval delays by up to 40%. This acceleration allows organizations like healthcare providers to train diagnostic models 2–3x faster, directly improving time to market for AI solutions.

Inference: As AI models become more complex, organizations face significant challenges. Integrating NVIDIA BasePOD and OVX with Run:ai enables the scaling of inference and RAG workloads. Run:ai optimizes inference clusters, allowing organizations of all sizes to deploy production models efficiently at scale. This optimization maximizes GPU utilization and provides user fairness through multi-tenant resource sharing, resource guarantees, and quotas. It ensures efficient inference scaling and deployment, enables concurrent RAG pipelines, and increases the number of users and models served. This integration allows enterprises to dynamically scale AI applications while minimizing compute costs and maximizing GPU utilization. For real-time inference workloads, Portworx delivers sub-millisecond latency through NVMe-optimized storage and configurable I/O profiles. This is critical for applications like autonomous vehicles or fraud detection systems, where delayed decisions have operational consequences. By maintaining persistent, low-latency access to model catalogs and vector databases, Portworx ensures inference engines operate at peak efficiency even during traffic spikes. Financial institutions leveraging these capabilities report 30% faster fraud detection responses, directly reducing revenue loss.

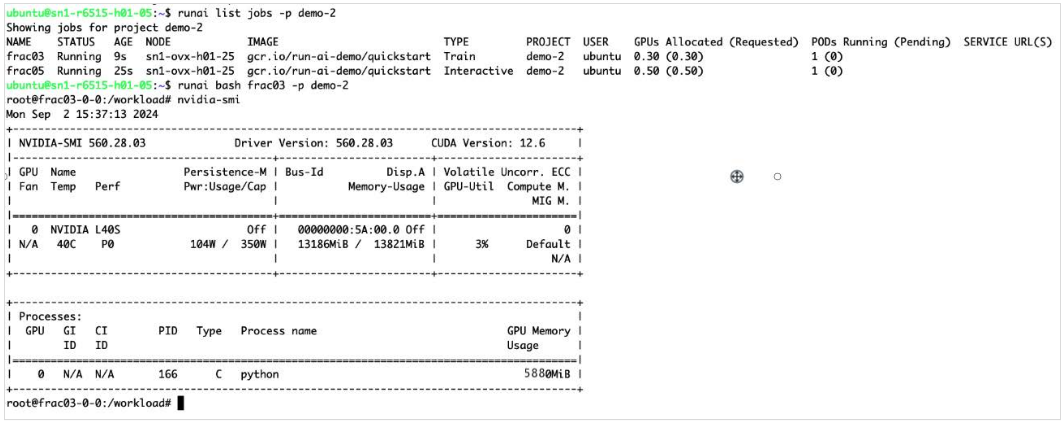
Key Solution Features

This integrated solution tackles the complexities of ML and GenAI workflows through optimized compute orchestration and scalable high-performance storage. Key benefits include:

1. Efficient Resource Utilization and Cost Reduction

Run:ai maximizes GPU efficiency through fractional GPU allocation, bin packing, dynamic GPU memory allocation, and dynamic quotas, ensuring that compute resources are fully utilized. These optimizations eliminate GPU idle time, lower infrastructure costs, and maximize return on investment (ROI):

- Data scientists can efficiently share GPUs, allocate the right amount of compute for their workloads, and over-provision GPUs when necessary for inference or interactive development environments like Jupyter Notebooks.
- MLOps/DevOps teams benefit from reduced GPU fragmentation, better resource overhead optimization, and prevention of unnecessary memory provisioning—leading to direct cost savings.



```
ubuntu@sn1-r6515-h01-05:~$ runai list jobs -p demo-2
Showing jobs for project demo-2
NAME    STATUS   AGE  NODE            IMAGE                            TYPE         PROJECT  USER    GPUs Allocated (Requested)  PODs Running (Pending)  SERVICE URL(S)
frac03  Running  9s   sn1-ovx-h01-25  gcr.io/run-ai-demo/quickstart  Train        demo-2   ubuntu  0.30 (0.30)                 1 (0)
frac05  Running  25s  sn1-ovx-h01-25  gcr.io/run-ai-demo/quickstart  Interactive  demo-2   ubuntu  0.50 (0.50)                 1 (0)
ubuntu@sn1-r6515-h01-05:~$ runai bash frac03 -p demo-2
root@frac03-0-0:/workload# nvidia-smi
Mon Sep  2 15:37:13 2024
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 560.28.03              Driver Version: 560.28.03      CUDA Version: 12.6     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA L40S                    Off |   00000000:5A:00.0 Off |                    0 |
| N/A   40C    P0            104W /  350W |   13186MiB /  13821MiB |      3%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                              GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|    0   N/A  N/A       166      C   python                                       5880MiB |
+-----------------------------------------------------------------------------------------+
root@frac03-0-0:/workload# █
```

FIGURE 3 Figure 3. List of jobs in Run:ai showing fractional GPU allocation

Run:ai addresses this challenge by dynamically orchestrating GPU usage across teams and projects, ensuring no compute power goes to waste, making it ideal for organizations with diverse AI workloads. Combining the capabilities of FlashBlade and Run:ai optimizes resource utilization and reduces unnecessary spending. FlashBlade reduces data access latency and supports high throughput, eliminating performance bottlenecks that could increase operational costs.

2. Simplified Resource Access, Improved Productivity, and Reduced IT Burden

Run:ai abstracts infrastructure complexity, allowing AI teams to focus on building and optimizing models instead of manually managing compute resources. With automated GPU discovery, allocation, and dynamic quotas, workflows become significantly more streamlined:

- Data scientists no longer need to spend time searching for available GPUs, speeding up experimentation and reducing bottlenecks in model development.
- MLOps/DevOps teams benefit from automated scheduling of distributed workloads, minimizing manual configurations and enabling them to focus on higher-value tasks like optimizing AI pipelines.
- IT administrators see a reduction in support tickets related to resource allocation, lowering operational overhead and making infrastructure management more efficient.

3. Fair and Equitable Resource Allocation

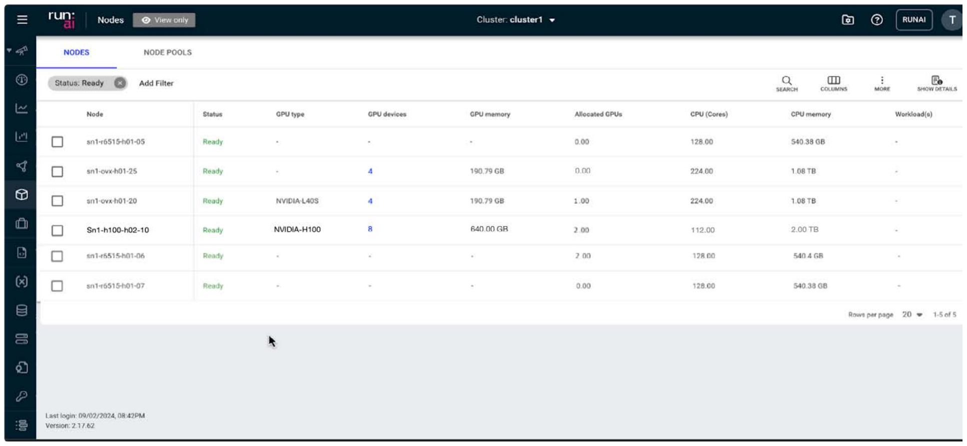
Run:ai ensures fair resource distribution using hierarchical policies and fairness algorithms that dynamically allocate GPU resources based on workload priority. High-priority tasks receive the necessary compute power, while lower-priority workloads still operate within designated resource limits.

IT administrators can configure custom resource-sharing policies that align with organizational goals, ensuring that critical workloads never face resource starvation while maintaining fair distribution among teams.

4. Centralized Visibility and Streamlined Infrastructure

Run:ai provides a single-pane-of-glass interface that consolidates compute usage metrics, job monitoring, and workload distribution across both on-premises and cloud environments. This unified visibility simplifies operations and enhances system-wide efficiency.

- MLOps/DevOps teams gain a centralized dashboard for monitoring GPU workloads, quickly identifying bottlenecks and optimizing resource consumption.
- IT administrators can reduce infrastructure silos, lower maintenance overhead, and create a more streamlined AI/ML compute environment, leading to greater operational efficiency and system reliability.

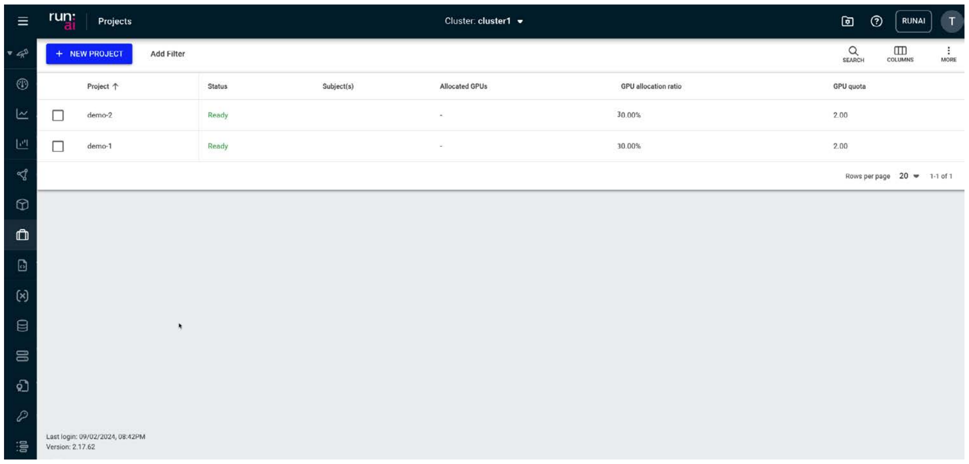


Node	Status	GPU type	GPU devices	GPU memory	Allocated GPUs	CPU (Cores)	CPU memory	Workload(s)
sn1-r6515-h01-05	Ready	-	-	-	0.00	128.00	540.38 GB	-
sn1-ovx-h01-25	Ready	-	4	190.79 GB	0.00	224.00	1.08 TB	-
sn1-ovx-h01-20	Ready	NVIDIA-L40S	4	190.79 GB	1.00	224.00	1.08 TB	-
Sn1-h100-h02-10	Ready	NVIDIA-H100	8	640.00 GB	2.00	112.00	2.00 TB	-
sn1-r6515-h01-06	Ready	-	-	-	2.00	128.00	540.4 GB	-
sn1-r6515-h01-07	Ready	-	-	-	0.00	128.00	540.38 GB	-

Rows per page 20 1-5 of 5

Last login: 09/02/2024, 08:42PM
Version: 2.17.62

FIGURE 4 Figure 4. Run:ai nodes dashboard showing available nodes



Project ↑	Status	Subject(s)	Allocated GPUs	GPU allocation ratio	GPU quota
demo-2	Ready		-	30.00%	2.00
demo-1	Ready		-	30.00%	2.00

Rows per page 20 1-1 of 1

Last login: 09/02/2024, 08:42PM
Version: 2.17.62

FIGURE 5 Figure 5. Run:ai projects view with resource allocation

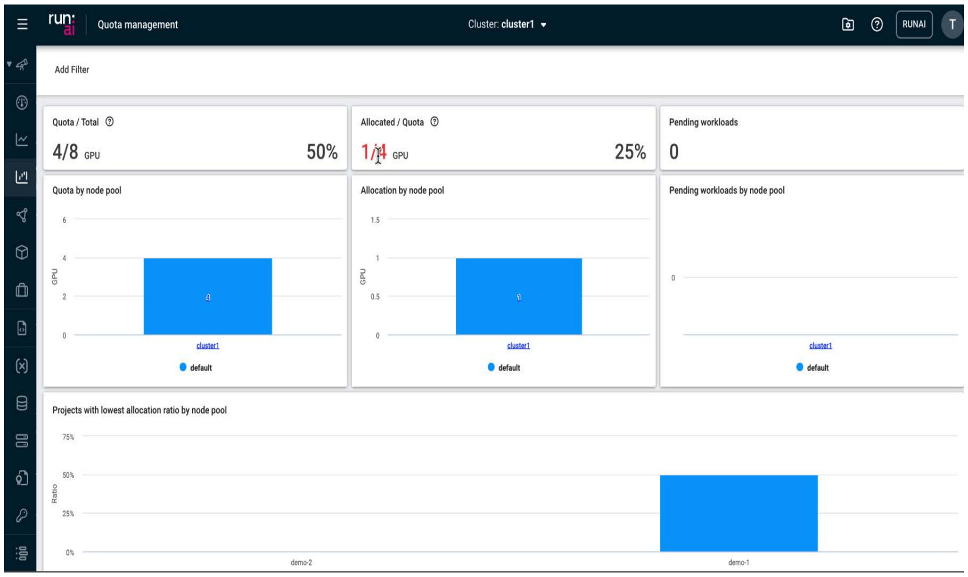


FIGURE 6 Figure 6. Run:ai

AI workloads, particularly in generative AI, demand rapid access to vast amounts of data, from large training datasets to complex model checkpoints. With support for NFS, S3, and SMB protocols, FlashBlade seamlessly integrates into diverse AI environments.

5. Scalable, High-performance Storage

AI and ML applications require high-speed, scalable storage solutions to manage vast datasets and optimize performance. FlashBlade delivers low-latency, high-throughput storage with NFS, S3, and SMB support, and is capable of handling billions of files and objects efficiently.

- Data scientists benefit from faster iteration cycles and reduced data access latency, accelerating AI model development.
- MLOps/DevOps teams experience consistent high throughput, ensuring that distributed training workloads and model serving remain smooth and efficient.
- IT administrators deploy a flexible, future-proof storage solution that scales with evolving AI workloads while maintaining peak performance.

6. Enterprise-grade Reliability

Portworx ensures continuous operations through:

- **Automatic failover**: High availability configurations maintain 99.99% uptime for mission-critical inference pipelines**.**
- **Snapshot-driven recovery**: ML model versions and training data can be restored in minutes rather than hours, minimizing downtime costs.

7. Cost Optimization

Portworx addresses two major cost drivers in AI infrastructure:

1. **GPU utilization**: By synchronizing data availability with compute resource scheduling, idle GPU time is reduced by up to 50%.
2. **Storage efficiency**: Deduplication and thin provisioning minimize redundant data copies, lowering storage costs by 40% for enterprises managing petabyte-scale datasets.

Solution Architecture

Recommended Hardware Architecture and Design

- **Accelerated computing resources**: NVIDIA DGX systems and NVIDIA OVX systems deliver the necessary computational power for AI workloads.
- **FlashBlade storage arrays**: FlashBlade//S500 provides a unified storage platform for NFS, S3, and SMB protocols, supporting large-scale vector databases and facilitating high-speed data retrieval. On a FlashBlade system, objects can be natively accessed via the S3 storage protocol, and simultaneously files can be accessed natively via the NFS or SMB storage protocols.
- **Portworx**: Portworx manages the data layer with enterprise-grade storage that ensures data persistence, mobility, and protection. Portworx extends the scheduling of Kubernetes to accelerate innovation and increase availability to the data needed to power RAG and inference workloads.
- **Network switches**: The SN3700 enables connectivity to endpoints at different speeds and carries a throughput of 6.4Tb/s, with a landmark 8.33Bpps processing capacity. As an ideal spine solution, the SN3700 offers maximum flexibility, with port speeds spanning from 10GbE to 200GbE per port.
- **BCM**: NVIDIA BCM is a comprehensive cluster management software that automates deployment, monitoring, and administration of heterogeneous AI and HPC clusters across data centers, edge, and hybrid-cloud environments. It streamlines GPU resource provisioning, workload management, and infrastructure monitoring, supporting seamless integration with NVIDIA hardware and software platforms.

Proper sizing and tuning of accelerated computing and storage resources are critical for optimizing performance. Detailed guidelines on configurations ensure that the platform meets specific workload requirements. In the testing and design of the Pure Storage and NVIDIA integrated solution, we used two nodes of NVIDIA OVX systems with one to 10 chassis FlashBlade//S500 systems, Portworx, NVIDIA BCM, and an NVIDIA Spectrum-2 SN3700 switch.

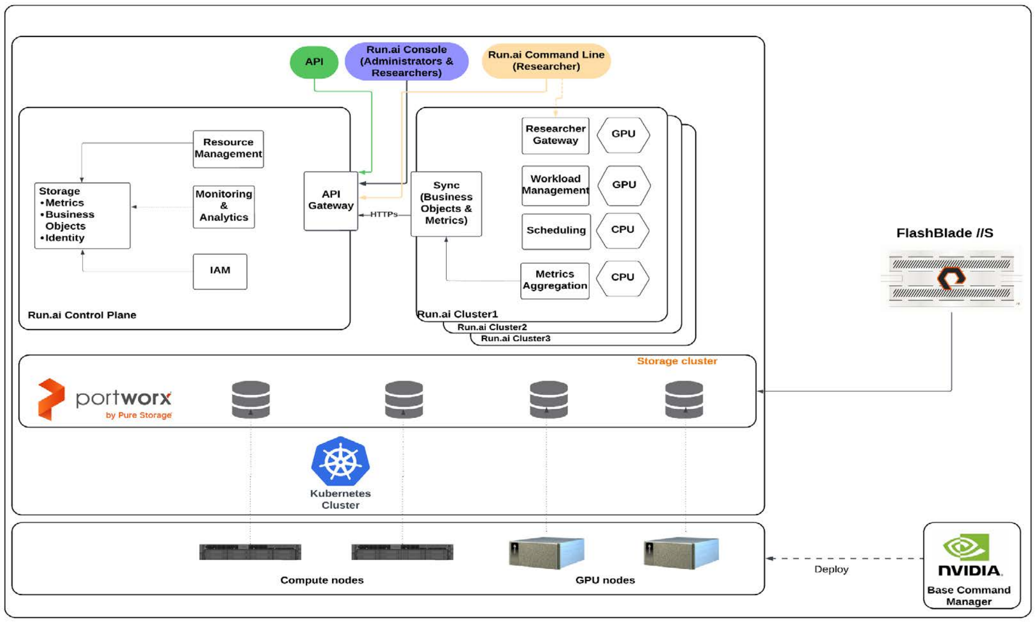


FIGURE 7 Architecture diagram

Deployment Guide

The process involves deploying Portworx, Run:ai, Kubernetes (K8s), and BCM stack in an environment with L40S, FBS200, and SN3700. The following walks through each software component setup in detail to help guide you in the deployment of this complex stack.

Infrastructure: Supermicro servers with LS40 GPU and Dell R6515. All nodes with one additional NVME drive.

High-level flow:

- Install Kubernetes cluster using BCM
- Deploy Portworx and create storage classes
- Integrate FlashBlade
- Install Run:ai

Prerequisites: Existing NVIDIA Base Command Manager and NVAIE deployment.

This guide assumes that NVIDIA AI Enterprise (NVAIE) and NVIDIA BCM are already deployed and operational using a Type 1 networking topology across your leaf-spine fabric. The steps below build on that foundation. We have tested this deployment with BCM version 10.0. The Kubernetes version installed by BCM is 1.31.8.

Software Deployment

Install Kubernetes Using BCM

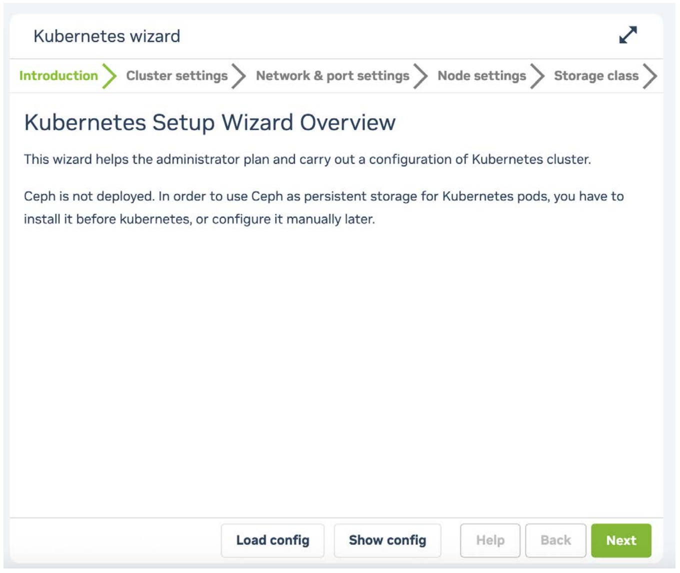
Node Preparation

It is assumed that all the nodes have a supported operating system (OS) installed and are shown online in the BCM device list.

All the worker nodes should have one NVMe drive other than the OS drive for Portworx installation.

Kubernetes Install Wizard

Log in to BCM and select Containers > Kubernetes > Kubernetes Wizard.
Introduction section:



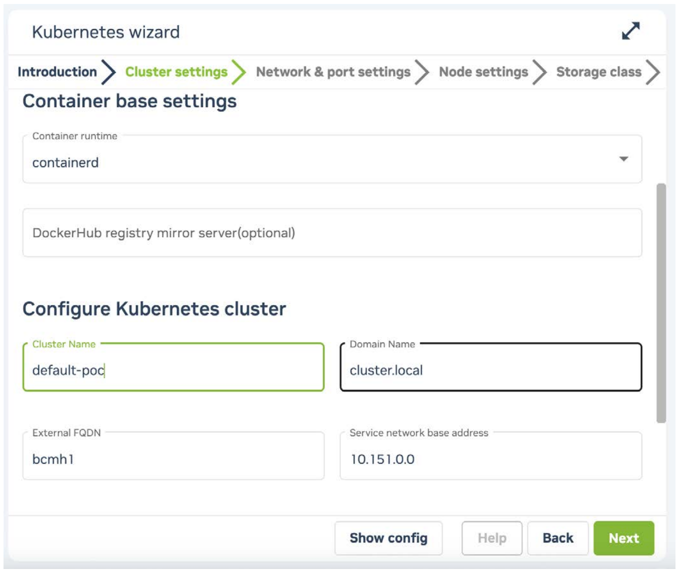
Kubernetes Setup Wizard Overview

This wizard helps the administrator plan and carry out a configuration of Kubernetes cluster.

Ceph is not deployed. In order to use Ceph as persistent storage for Kubernetes pods, you have to install it before kubernetes, or configure it manually later.

Click “Next.”

Cluster settings section:



Container base settings

Container runtime: containerd

DockerHub registry mirror server(optional)

Configure Kubernetes cluster

Cluster Name: default-poc

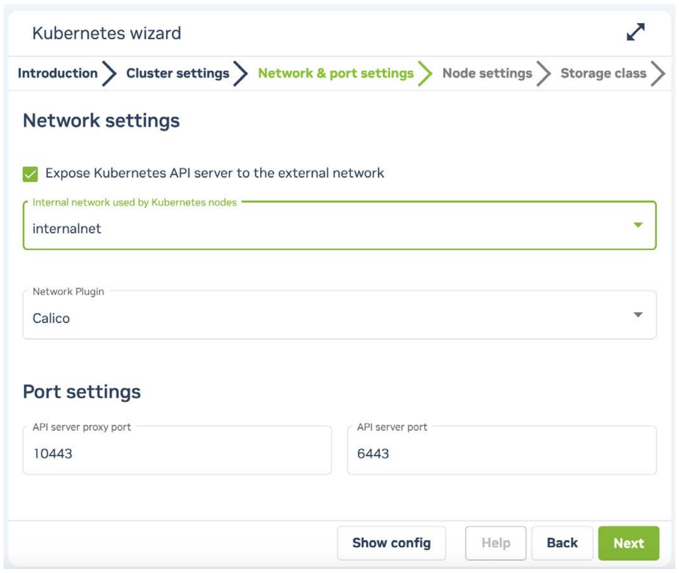
Domain Name: cluster.local

External FQDN: bcmh1

Service network base address: 10.151.0.0

Specify the “Cluster name” and click “Next.”

Network & port settings section:



Network settings

☑ Expose Kubernetes API server to the external network

Internal network used by Kubernetes nodes: internalnet

Network Plugin: Calico

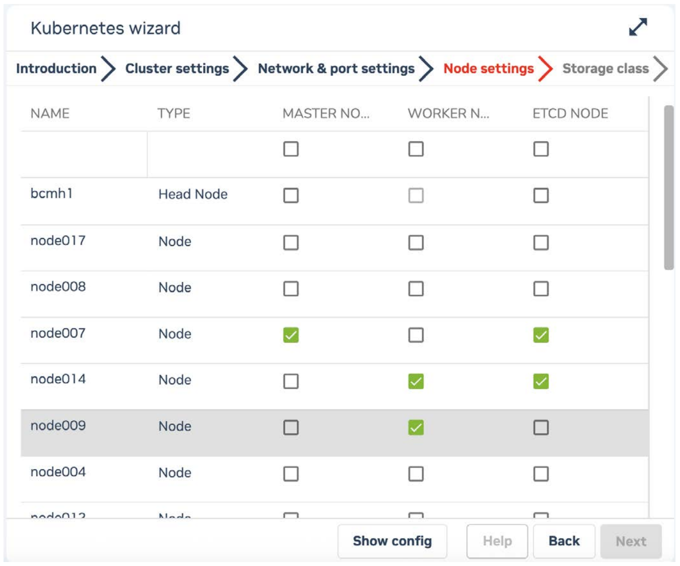
Port settings

API server proxy port: 10443

API server port: 6443

Select the “Internal network used by Kubernetes nodes.” Since we are using Type 1 BCM network topology, select “internalnet.” This is a private subnet where all nodes are configured.

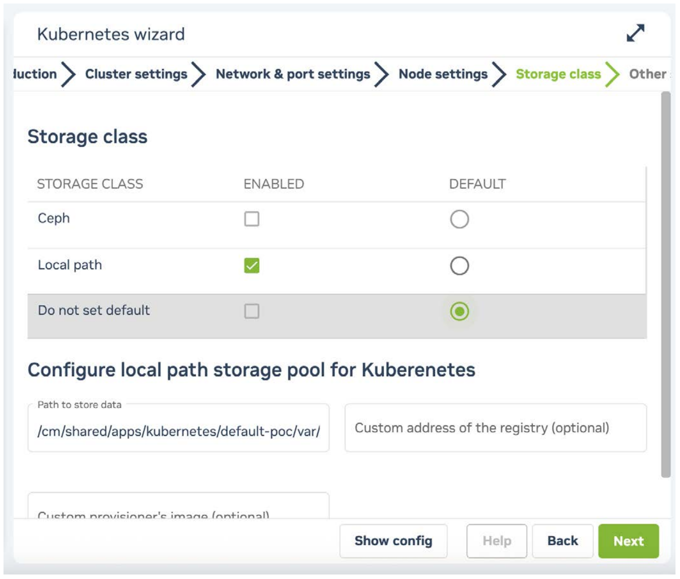
Node settings section:



NAME	TYPE	MASTER NO...	WORKER N...	ETCD NODE
		☐	☐	☐
bcmh1	Head Node	☐	☐	☐
node017	Node	☐	☐	☐
node008	Node	☐	☐	☐
node007	Node	✓	☐	✓
node014	Node	☐	✓	✓
node009	Node	☐	✓	☐
node004	Node	☐	☐	☐

Select the master, worker, and etcd nodes required for the Kubernetes cluster.

Storage class section:



Storage class

STORAGE CLASS	ENABLED	DEFAULT
Ceph	☐	○
Local path	✓	○
Do not set default	☐	●

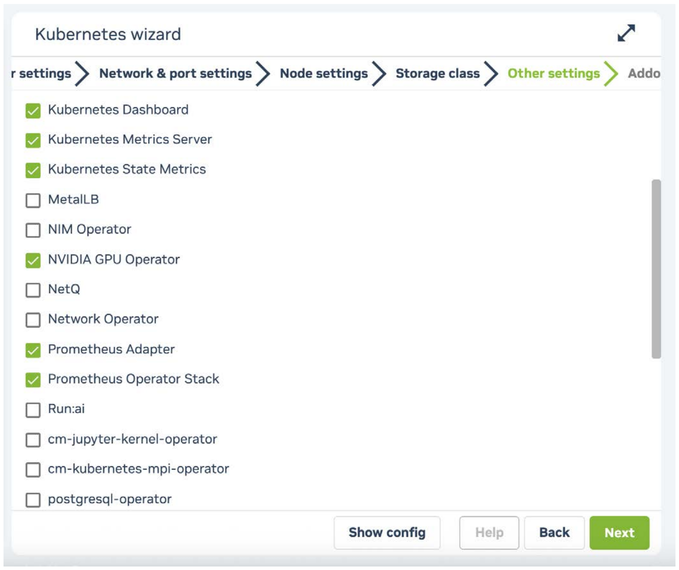
Configure local path storage pool for Kuberenetes

Path to store data: /cm/shared/apps/kubernetes/default-poc/var/

Custom address of the registry (optional)

Select “Local path” as the storage class and mark not to use it as default. We will set Portworx pools as the default storage class later in the installation.

Other settings section:

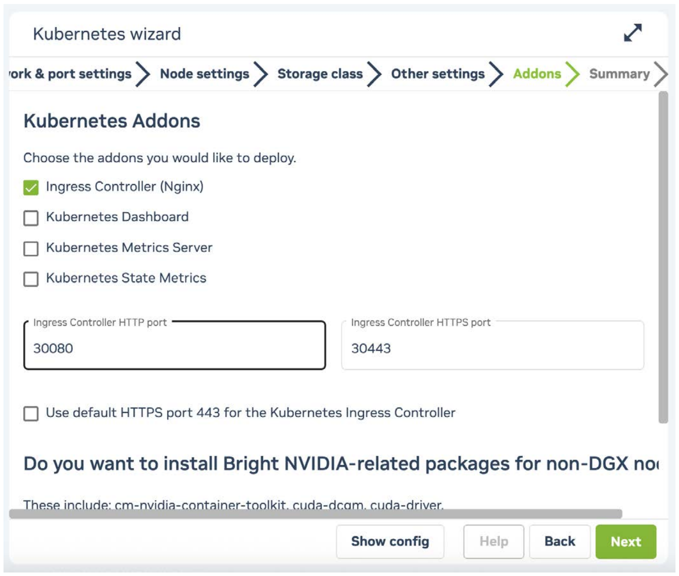


- [✓] Kubernetes Dashboard
- [✓] Kubernetes Metrics Server
- [✓] Kubernetes State Metrics
- [ ] MetalLB
- [ ] NIM Operator
- [✓] NVIDIA GPU Operator
- [ ] NetQ
- [ ] Network Operator
- [✓] Prometheus Adapter
- [✓] Prometheus Operator Stack
- [ ] Run:ai
- [ ] cm-jupyter-kernel-operator
- [ ] cm-kubernetes-mpi-operator
- [ ] postgresql-operator

Select the operators highlighted in the image above. Note: We are not using the Run:ai operator from this section.

Unselect “Install permissions manager,” because this installs the keyverno policy manager which is not tested by NVIDIA for Run:ai install.

Add-ons section:



Kubernetes Addons

Choose the addons you would like to deploy.

- [✓] Ingress Controller (Nginx)
- [ ] Kubernetes Dashboard
- [ ] Kubernetes Metrics Server
- [ ] Kubernetes State Metrics

Ingress Controller HTTP port: 30080

Ingress Controller HTTPS port: 30443

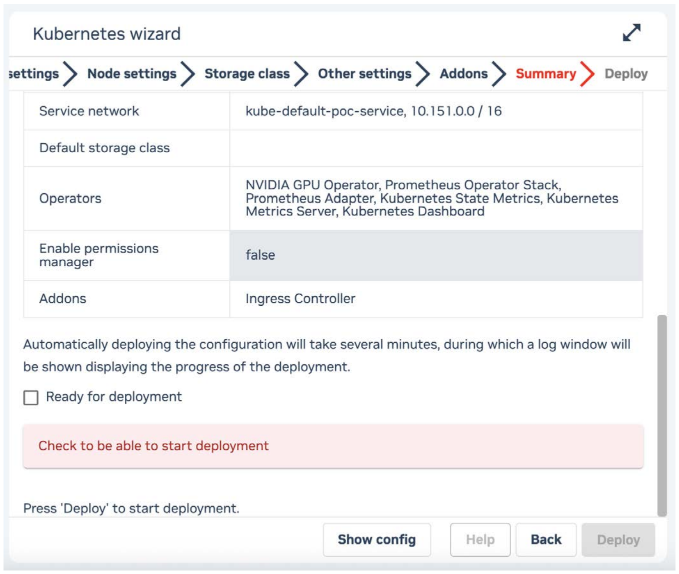
- [ ] Use default HTTPS port 443 for the Kubernetes Ingress Controller

Do you want to install Bright NVIDIA-related packages for non-DGX no

These include: cm-nvidia-container-toolkit, cuda-dcgm, cuda-driver,

Select “Ingress controller (Nginx).”

Summary section:



Kubernetes wizard

settings > Node settings > Storage class > Other settings > Addons > Summary > Deploy

Service network	kube-default-poc-service, 10.151.0.0 / 16
Default storage class	
Operators	NVIDIA GPU Operator, Prometheus Operator Stack, Prometheus Adapter, Kubernetes State Metrics, Kubernetes Metrics Server, Kubernetes Dashboard
Enable permissions manager	false
Addons	Ingress Controller

Automatically deploying the configuration will take several minutes, during which a log window will be shown displaying the progress of the deployment.

☐ Ready for deployment

Check to be able to start deployment

Press 'Deploy' to start deployment.

Select “Ready for deployment” and click “Deploy.”

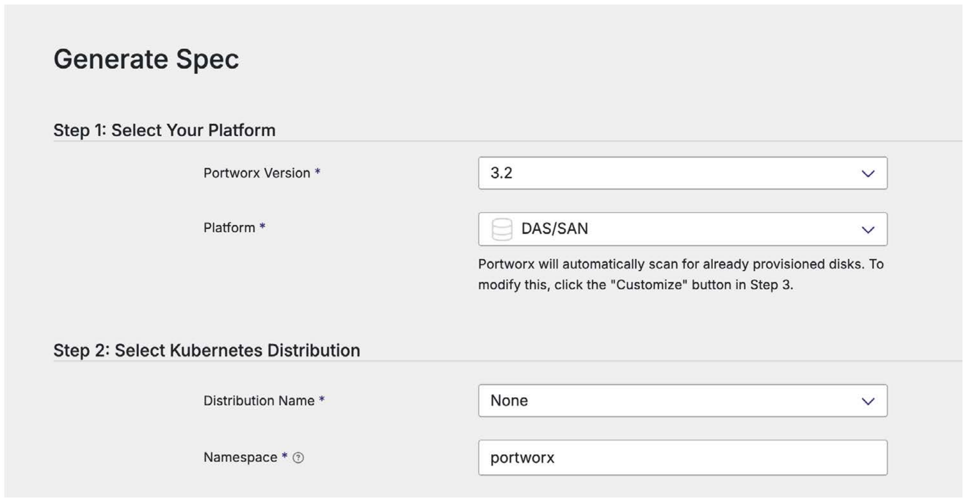
Install Portworx

Portworx is a software-defined storage platform designed specifically for containerized applications and microservices, primarily operating in environments like Kubernetes.

Prepare the Installation YAML from Portworx Central

Access https://central.portworx.com/ and create an account.

Click “Get started” and “Portworx Enterprise.”



Generate Spec

Step 1: Select Your Platform

Portworx Version *	3.2
Platform *	DAS/SAN

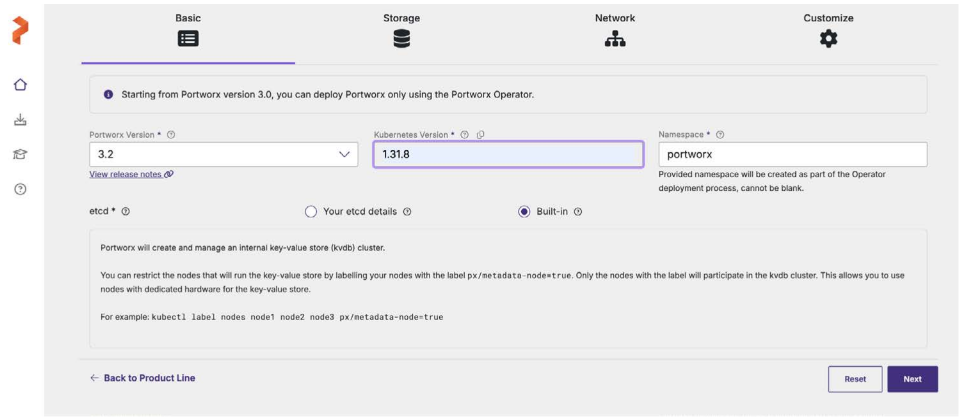
Portworx will automatically scan for already provisioned disks. To modify this, click the "Customize" button in Step 3.

Step 2: Select Kubernetes Distribution

Distribution Name *	None
Namespace *	portworx

Select the option as mentioned in the screenshot.

Click on customize.



Starting from Portworx version 3.0, you can deploy Portworx only using the Portworx Operator.

Portworx Version * : 3.2

View release notes

Kubernetes Version * : 1.31.8

Namespace * : portworx

Provided namespace will be created as part of the Operator deployment process, cannot be blank.

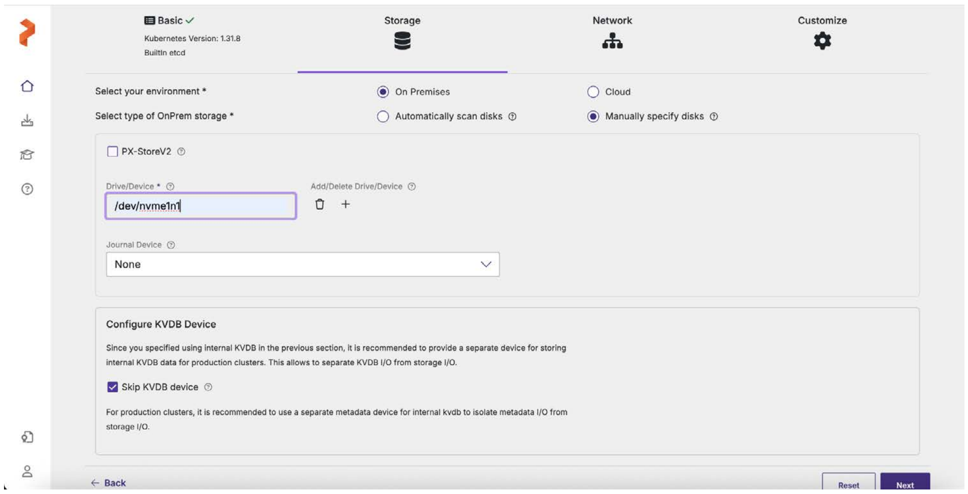
etcd * : ○ Your etcd details [✓ ● Built-in]

Portworx will create and manage an internal key-value store (kvdb) cluster.

You can restrict the nodes that will run the key-value store by labelling your nodes with the label `px/metadata-node=true`. Only the nodes with the label will participate in the kvdb cluster. This allows you to use nodes with dedicated hardware for the key-value store.

For example: `kubectl label nodes node1 node2 node3 px/metadata-node=true`

Type the Kubernetes version.



Kubernetes Version: 1.31.8

BuiltIn etcd

Select your environment * : [✓ ● On Premises] ○ Cloud

Select type of OnPrem storage * : ○ Automatically scan disks [✓ ● Manually specify disks]

☐ PX-StoreV2

Drive/Device * : /dev/nvme1n1

Add/Delete Drive/Device

Journal Device : None

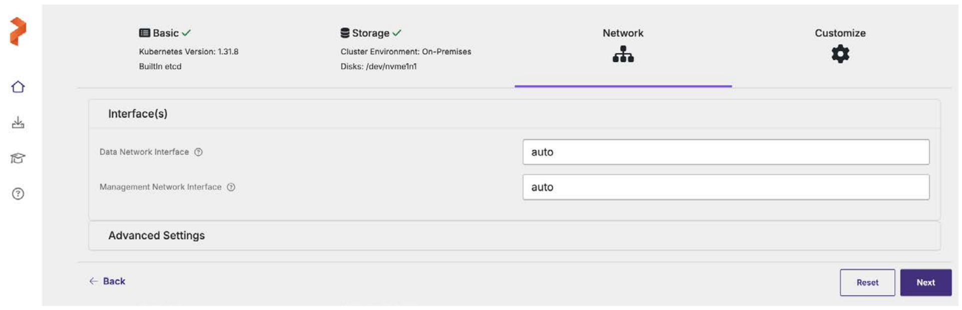
Configure KVDB Device

Since you specified using internal KVDB in the previous section, it is recommended to provide a separate device for storing internal KVDB data for production clusters. This allows to separate KVDB I/O from storage I/O.

☑ Skip KVDB device

For production clusters, it is recommended to use a separate metadata device for internal kvdb to isolate metadata I/O from storage I/O.

Specify the disk available on all worker nodes manually.



Kubernetes Version: 1.31.8

BuiltIn etcd

Cluster Environment: On-Premises

Disks: /dev/nvme1n1

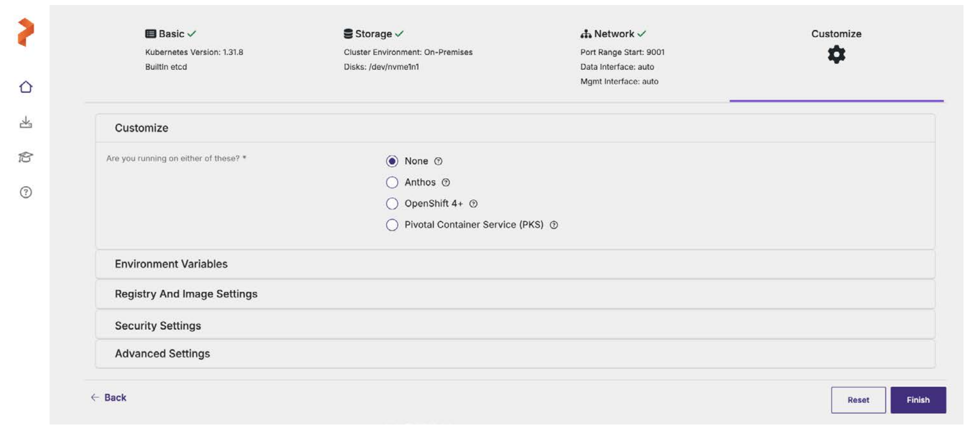
Interface(s)

Data Network Interface : auto

Management Network Interface : auto

Advanced Settings

Leave the network settings as default.



Kubernetes Version: 1.31.8
Builtin etcd

Cluster Environment: On-Premises
Disks: /dev/nvme1n1

Port Range Start: 9001
Data Interface: auto
Mgmt Interface: auto

Customize

Are you running on either of these? *

[✓ None] Anthos OpenShift 4+ Pivotal Container Service (PKS)

Environment Variables

Registry And Image Settings

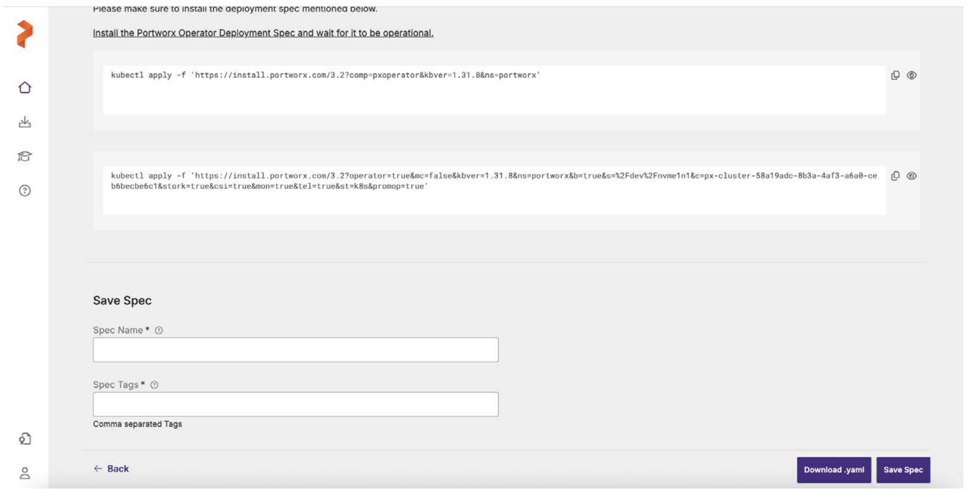
Security Settings

Advanced Settings

Leave the defaults and click “Finish.”

Copy the Portworx Operator installation command and run it.

```
kubectl apply -f 'https://install.portworx.com/3.2?comp=pxoperator&kbver=1.31.8&ns=portworx'
```



Please make sure to install the deployment spec mentioned below.

Install the Portworx Operator Deployment Spec and wait for it to be operational.

```
kubectl apply -f 'https://install.portworx.com/3.2?comp=pxoperator&kbver=1.31.8&ns=portworx'
```

```
kubectl apply -f 'https://install.portworx.com/3.2?operator=true&mc=false&kbver=1.31.8&ns=portworx&b=true&s=%2Fdev%2Fnvme1n1&c=px-cluster-58a19adc-8b3a-4af3-a6a0-ce
b6becbe6c1&stork=true&csi=true&mon=true&tel=true&st=k8s&promop=true'
```

Save Spec

Spec Name *

Spec Tags *

Comma separated Tags

Click on “Download.yaml.”

Disable Prometheus Exporters

Edit the YAML file (`portworx_enterprise.yaml`) downloaded in the previous step and change `enabled: false` under the `monitoring.prometheus` section.

Unset

```
    prometheus:
       enabled: false
       exportMetrics: true
```

Unset

```
   prometheus:
       enabled: false
       exportMetrics: true
```

Deploy Portworx

Deploy Portworx using the updated YAML file.

```
Unset kubectl apply -f portworx_enterprise.yaml
```

Change pvc-controller Ports

The default port used by pvc-controller conflicts with the port used by the Kubernetes control plan service. Hence, you can set a different port. This can be achieved by adding the following annotations for StorageCluster by editing it. kubectl -n portworx edit stc

Unset

```
     portworx.io/pvc-controller-port: "1111"
      portworx.io/pvc-controller-secure-port: "2222"
```

Clean Up Legacy Storage Classes

Update the StorageCluster and add the following annotations for StorageCluster by editing it.

```
kubectl -n portworx edit stc
```

```
Unset portworx.io/disable-storage-class: "true"
```

Update the Storage Cluster and Use the External Prometheus

Update the StorageCluster and update the Prometheus URL as shown below. We are using the Prometheus instance in the prometheus namespace deployed by BCM during initial Kubernetes deployment.

```
kubectl -n portworx edit stc
```

Unset

```
spec:
  autopilot:
    enabled: true
    providers:
    - name: default
      params:
        url: http://kube-prometheus-stack-prometheus.prometheus:9090
      type: prometheus
```

Enter the following kubectl get command and wait until all Portworx nodes show as Ready or Online in the output:

```
kubectl -n portworx get storagenodes -l name=portworx
```

2.6 Verify If All Pods Are Running

Enter the following kubectl get pods command to list and filter the results for Portworx pods:

```
kubectl get pods -n portworx -o wide | grep -e portworx -e px
```

Refer to the Portworx installation guide for more information.

Create Storage Class for Portworx Block Pool

px-block-sc.yaml:

Unset

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
 name: px-rep2
parameters:
 repl: "2"
 priority_io: "high"
provisioner: pxd.portworx.com
reclaimPolicy: Delete
volumeBindingMode: Immediate
allowVolumeExpansion: true
```

kubectl apply -f px-block-sc.yaml

Configure FlashBlade and Create Storage Class

Prerequisites

- Management VIP reachable from Kubernetes nodes.
- Create API token with minimum RBAC.
- Data VIP to access NFS.

Add FlashBlade

Create pure.json with the below content.

Unset

```
{
  "FlashBlades": [
    {
  "MgmtEndPoint": "10.x.x.x",
  "APIToken": "T-0xxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx",
  "NFSEndPoint": " 10.x.x.x",
 }
]
}
```

```
kubectl create secret generic px-pure-secret --namespace portworx --from-file=pure.json
```

FlashBlade Storage Classes

fb-sc.yaml:

Unset

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
 name: portworx-pso-fb-v3
provisioner: pxd.portworx.com
parameters:
 backend: "pure_file"
 pure_export_rules: "*(rw)"
mountOptions:
 - nfsvers=3
 - tcp
allowVolumeExpansion: true
```

kubectl apply -f fb-sc.yaml

Refer to Portworx documentation for more information.

Update Default Storage Class

Example command:

```
kubectl patch storageclass px-rep2 -p '{"metadata": {"annotations":{"storageclass.kubernetes.io/
is-default-class":"true"}}}'
```

```
kubectl patch storageclass local-path -p '{"metadata": {"annotations":{"storageclass.kubernetes.io/
is-default-class":"false"}}}'
```

BCM Head Node Configuration

After Kubernetes is installed through BCM, we need to configure some settings to ensure that the reverse proxy functions correctly.

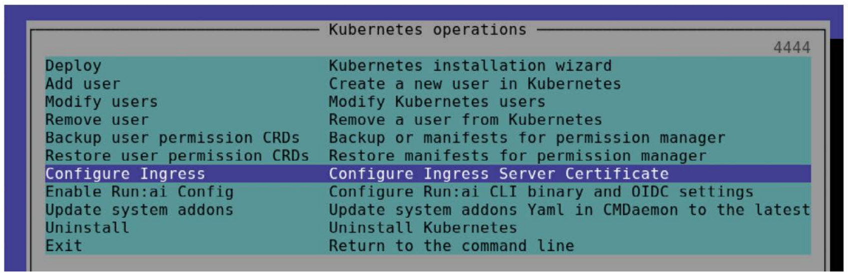
First, we are going to configure the Kubernetes ingress by adding the cert and key for the Run:ai domain. These are the same certificate and key that are used in the Run:ai setup.

Note: Please follow the company-specific steps to generate certificates. In our example, we're using "runai.puretec.purestorage.com" as the domain name.

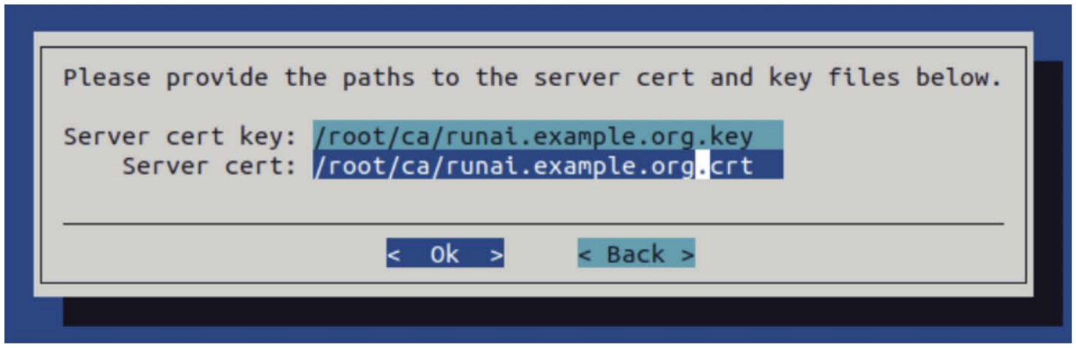
Update Ingress Server Certificate

Run:

Unset cm-kubernetes-setup



```
Kubernetes operations
                                                                        4444
Deploy                       Kubernetes installation wizard
Add user                     Create a new user in Kubernetes
Modify users                 Modify Kubernetes users
Remove user                  Remove a user from Kubernetes
Backup user permission CRDs  Backup or manifests for permission manager
Restore user permission CRDs Restore manifests for permission manager
Configure Ingress            Configure Ingress Server Certificate
Enable Run:ai Config         Configure Run:ai CLI binary and OIDC settings
Update system addons         Update system addons Yaml in CMDaemon to the latest
Uninstall                    Uninstall Kubernetes
Exit                         Return to the command line
```



```
Please provide the paths to the server cert and key files below.

Server cert key: /root/ca/runai.example.org.key
    Server cert: /root/ca/runai.example.org.crt

              <  Ok  >        < Back >
```

6.2 Update the Reverse Proxy

Now, we need to update the ingress controller to route HTTPS traffic to the head nodes. Substitute your control-plane nodes in the following commands.

Unset

```
cmsh

[bcmh1]% device use master

[bcmh1→device[bcmh1]]% roles

[bcmh1→device[bcmh1]→roles]% use nginx

[bcmh1→device[bcmh1]→roles[nginx]]% nginxreverseproxy

[bcmh1→device[bcmh1]→roles[nginx]→nginxreverseproxy]% add 443 node010 30443 'Run:ai'

[bcmh1→device*[bcmh1*]→roles*[nginx*]→nginxreverseproxy]% add 443 node011 30443 'Run:ai'

[bcmh1→device*[bcmh1*]→roles*[nginx*]→nginxreverseproxy]% commit
```

Install Run:ai

Run:ai provides GPU virtualization, queueing, and fair-share scheduling.

Create Secret

```
TOKEN=”NVIDIA RUNAI LICENSE KEY”

kubectl create ns runai-backend

kubectl create ns runai

kubectl create secret docker-registry runai-reg-creds --docker-server=https://runai.jfrog.io --docker-
username=self-hosted-image-puller-prod --docker-password=$TOKEN --docker-email=support@run.ai
--namespace=runai-backend

kubectl create secret docker-registry runai-reg-creds --docker-server=https://runai.jfrog.io --docker-
username=self-hosted-image-puller-prod --docker-password=$TOKEN --docker-email=support@run.ai
--namespace=runai
```

7.2 Create Cluster-wide CA Certificate

runai-ca.pem is the certificate in Step 6.

```
kubectl create secret generic runai-ca-cert --from-file=runai-ca.pem=/etc/ssl/certs/runai-ca.pem -n
runai-backend

kubectl create secret generic runai-ca-cert --from-file=runai-ca.pem=/etc/ssl/certs/runai-ca.pem -n runai

kubectl label secret runai-ca-cert -n runai run.ai/cluster-wide=true run.ai/name=runai-ca-cert
--overwrite
```

7.3 Install Run:ai Control Plane

Run the following commands to install the Run:ai control plane.

```
helm repo add runai-backend https://runai.jfrog.io/artifactory/cp-charts-prod

helm search repo -l runai-backend
```

Select the version you want to install. In this case we are using the latest version (2.21.14).

```
helm upgrade -i runai-backend -n runai-backend runai-backend/control-plane --version 2.21.14 --set global.
domain=runai.puretec.purestorage.com --set global.customCA.enabled=true
```

Make sure all the pods are in running mode in the runai-backend name space.

7.4 Install Run:ai Cluster

Update FQDN DNS record is pointing to the IP address of the BCM head node.

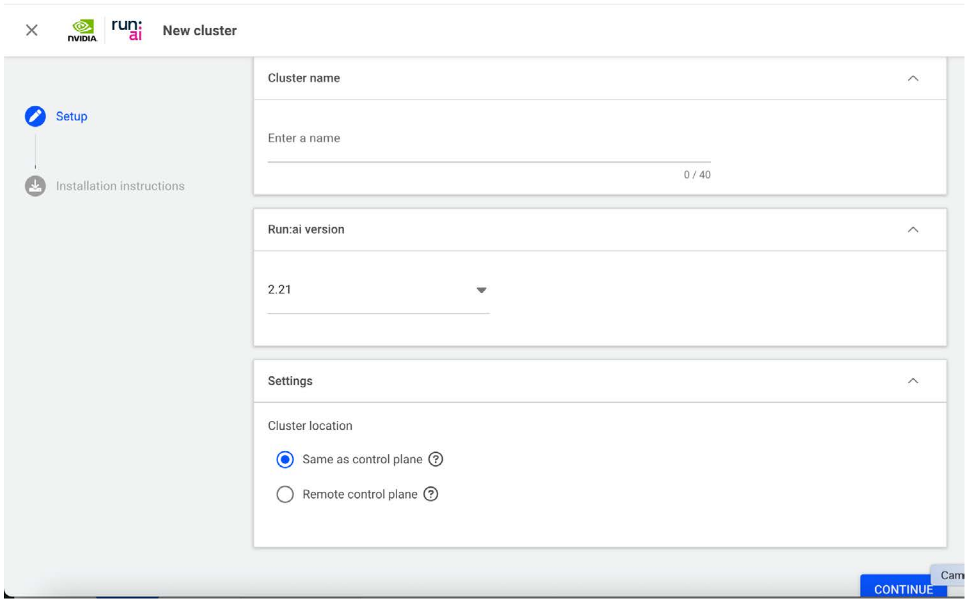
In our case, we added the following entry to /etc/hosts on a Mac:

10.21.0.x runai.puretec.purestorage.com

If all the components come up, we will be able to access the Run:ai control plane at https://runai.puretec.purestorage.com.

Log in with the default username and password (test@run.ai / Admin!234).

A new cluster creation wizard will open.



Cluster name

Enter a name 0 / 40

Run:ai version

2.21

Settings

Cluster location

[✓ Same as control plane]

Remote control plane

Once all the requirements are met, run the commands shown in the screen:

Example commands:

```
helm repo add runai https://runai.jfrog.io/artifactory/api/helm/op-charts-prod

helm repo update

helm upgrade -i runai-cluster runai/runai-cluster -n runai \

--set controlPlane.url=runai.puretec.purestorage.com \

--set controlPlane.clientSecret=BKMYbmBEaBwBrl02vZ3njMJOo5QrC5L4 \

--set cluster.uid=dcb41c3f-5858-46a9-af70-1b1b4cb59882 \

--set cluster.url=runai.puretec.purestorage.com --version=2.17.49 --create-namespace --set global.
customCA.enabled=true
```

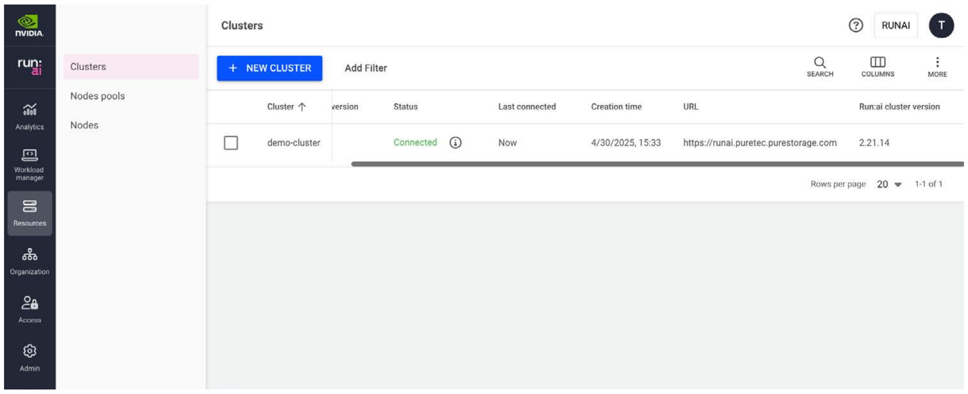
Note: Add "--set global.customCA.enabled=true" at the end of the helm upgrade command and run it.

7.5 Verify and Test GPU Workload

Enter the following kubectl get command and wait until all Run:ai cluster pods show as Running in the output:

```
kubectl -n runai get pods
```

Once all the pods are in running state, the cluster will show as Connected in the Run:ai resources UI.



Cluster	version	Status	Last connected	Creation time	URL	Run:ai cluster version
demo-cluster		Connected	Now	4/30/2025, 15:33	https://runai.puretec.purestorage.com	2.21.14

Rows per page 20 1-1 of 1

8. Setting Up Projects and Running Workloads

Once the cluster is in a connected state, we can create a project and run workloads.

This setup enables efficient monitoring, collaboration, and management of machine learning workflows within a secure on-premises infrastructure.

Comparison with Traditional Tools

Pure Storage and Run:ai form a powerful duo by redefining AI workflows and offering high-performance storage solutions by resolving critical limitations of traditional tools.

Run:ai vs. Traditional Resource Management Tools

Traditional tools lack the ability to dynamically allocate GPU resources across various workloads, leading to underutilization, static allocation, and inefficient scaling in hybrid cloud environments. Run:ai enables AI engineers to optimize GPU usage by allowing multiple workloads to share resources without idle time. Additionally, it seamlessly integrates with on-premises and cloud resources based on workload demands. As a result, Run:ai minimizes waste and ensures maximum return on investment in GPU infrastructure.

Pure Storage Platform (FlashBlade and Portworx) vs. Traditional Storage Solutions

Legacy storage systems struggle to meet AI workload demands due to limited scalability, high latency, and lack of multi-protocol support. FlashBlade addresses these challenges with high performance, delivering ultra-low latency and high throughput, exceptional scalability for billions of files and objects, and seamless multi-protocol integration with NFS, S3, and SMB. Its data-centric design optimizes unstructured data processing for faster model training and real-time analytics, while its future-proof infrastructure simplifies scaling and ensures long-term efficiency. Portworx can manage backend databases to power vector for RAG. Plus, the self-service capabilities of the platform accelerate deployment for data science and development teams by shortening training time and directly addressing GPU.

The Portworx-FlashBlade synergy addresses the scale, speed, and agility AI workloads require in a unified platform:

- **Unified data plane**: Consolidating training, inference, and backup on a single scalable platform.
- **Kubernetes-native agility**: Automating lifecycle management for dynamic AI workloads.
- **Future-proof architecture**: Supporting Ethernet-based NVIDIA SuperPODs and retrieval-augmented generation (RAG).

Together, this solution enables centralized collaboration, end-to-end visibility, and unparalleled efficiency, empowering AI teams to innovate faster while reducing costs.

Feature Validation and Observations

Run:ai offers several features for fine-grained resource management on a GPU cluster. The following features were validated on an AI stack with OVX, FlashBlade//S500, and SN3700 switches.

- **GPU allocation**: Users can submit multiple inference or training jobs with varying GPU allocation requirements, especially fractional GPU allocation and dynamic multi-instance GPU (MIG) utilization. The Run:ai scheduler allocates the prescribed quota in terms of GPU cores and memory for each user, providing strict resource guarantees. Increasing overall allocation efficiency by fractionalization increases FlashBlade's throughput demands and ensures GPU utilization is optimized.
- **Intelligent scheduling**: Users can submit multiple inference and training jobs and monitor how Run:ai handles quota workloads, fairness through preemption, and bin packing/spreading strategies. Intelligent scheduling helps organizations lower total cost of ownership by implementing borrowing, fairness, and bin packing tasks. Further, FlashBlade elevates enterprise workloads since GPU clusters can be borrowed across organizations to run jobs more efficiently.
- **Interactive build session validation**: Users can execute interactive build sessions such as debugging and real-time AI model adjustments using GPU orchestration in Run:ai.
- **Workload prioritization**: Users and administrators can prioritize and rank workloads in the queue for execution order.

Conclusion

The integration of Pure Storage FlashBlade, Portworx, BCM, and Run:ai provides a robust foundation for addressing the challenges of modern machine learning and generative AI development. By optimizing GPU allocation, enabling effective collaboration of resources through borrowing and pre-emption of GPU resources, and ensuring the scalability of compute and storage, organizations are empowered to innovate faster, reduce costs, and achieve greater efficiency across their AI projects.

Additional Resources

- Accelerate AI adoption with the Pure Storage Data Platform
- Explore Pure Storage FlashBlade//S500
- Read the Run:ai documentation

CM1082-02-en 05/25