SUSE Virtualization, Portworx by Everpure, Supermicro Compact Edge Servers

SUSE Virtualization with Portworx Enterprise by Everpure in a Two-Node with Arbiter Topology

Deliver Highly Available, Cloud-Native Virtualization at the Edge

SUSE Virtualization
Portworx by Everpure
Supermicro Compact Edge Servers

Suresh S, Partner Solution Architect (SUSE)
Gopala Krishnan, Partner Solution Architect (SUSE)
Terry Smith, Partner Solution Innovation Director (SUSE)
Bhumitra Nagar, Member of Technical Staff (Everpure)

SUSE Virtualization with Portworx Enterprise by Everpure in a Two-Node with Arbiter Topology

Deliver Highly Available, Cloud-Native Virtualization at the Edge

Date: 2026-06-17

Summary

This document provides step-by-step technical guidance for deploying Portworx Enterprise in a Two-Node with Arbiter (TNA) topology on SUSE Virtualization, providing highly available, cloud-native virtualization for edge workloads.

Contents

1 Introduction

Organizations increasingly need to extend their application environments beyond centralized data centers to edge and far-edge locations, where infrastructure constraints, operational complexity, and cost sensitivity are significantly higher. These environments require solutions that deliver enterprise-grade availability and performance within a minimal hardware footprint. This reference configuration addresses these challenges by providing a consolidated architecture that runs both virtualized and containerized workloads on a unified platform, at the edge. This eliminates the separate infrastructure stacks traditional approaches require, reducing costs, operational overhead, and governance complexity while maintaining resilience and scalability.

This document presents a validated reference architecture that combines SUSE Virtualization, Portworx by Everpure, and Supermicro Compact Edge Servers technologies to support modern edge use cases. By leveraging a Two-Node with Arbiter (TNA) topology, the solution delivers high availability using only two primary nodes and a lightweight arbiter node, reducing infrastructure requirements without compromising business continuity.

1.1 Scope

This document provides a validated, reference design for deploying SUSE Virtualization with Portworx Enterprise using a Two-Node with Arbiter (TNA) topology. It describes the overall solution architecture and explains how the core components work together to support both virtualized and containerized workloads in edge and far-edge environments. It is intended to serve as a practical framework that organizations can use to implement a consistent, resilient, and cost-efficient platform tailored to their specific operational needs.

This reference design was validated with SUSE Virtualization (Harvester) 1.7, Portworx Enterprise by Everpure 3.6, and Supermicro Compact Edge Servers (SYS-E403-14B-FRN2T). The design and implementation guidance, however, should require little to no modification to apply to newer versions and models.

1.2 Audience

This document is intended for the following roles:

- Infrastructure Architects
- Platform Engineers

- System Administrators
- DevOps Engineers

To effectively follow this guide, readers should have

- a working knowledge of Kubernetes, including persistent storage concepts such as storage classes and persistent volumes.
- familiarity with virtualization technologies and Portworx Enterprise by Everpure.
- an understanding of distributed storage concepts, including replication and quorum.
- experience with Linux system administration.
- a basic understanding of networking concepts relevant to clustered environments.

1.3 Acknowledgments

The authors wish to thank Supermicro for providing access to hardware for the validation of this solution.

2 Business aspect

Organizations are increasingly extending their infrastructure to edge and distributed environments, where they must operate with limited resources while maintaining high availability and consistent performance. These environments introduce challenges related to infrastructure footprint, operational complexity, and the need to efficiently support both virtualized and containerized workloads.

This solution addresses these challenges by providing a unified platform based on SUSE Virtualization, Portworx by Everpure, and Supermicro Compact Edge Servers. This Two-Node with Arbiter (TNA) architecture enables organizations to reduce infrastructure requirements while maintaining resilience and simplifying operations, making it well suited for cost-efficient edge and far-edge deployments.

2.1 Challenge

Organizations operating at the edge face a compounded set of challenges that traditional infrastructure models are not designed to solve, including:

- **Separate stacks for VMs and containers:** Legacy virtualization platforms and container platforms are managed independently, doubling operational overhead, licensing costs, and training requirements.
- **Three-node HA storage mandates:** Conventional distributed storage systems require a minimum of three full storage nodes to achieve quorum and tolerate a single-node failure. At the edge, provisioning three heavy-weight nodes per site is prohibitively expensive.
- **Operational complexity at scale:** Enterprises with hundreds of edge sites cannot staff each with a skilled infrastructure engineer. Automation, consistency, and simplified deployment procedures are non-negotiable.

2.2 Value

This TNA architecture enables Portworx to deliver highly available storage for SUSE Virtualization VMs and containers using only two storage nodes and one lightweight arbiter (witness) node. The arbiter node does not store workload data. Instead, it stores only metadata, allowing it to participate in KVDB quorum decisions. This dramatically lowers the hardware footprint while providing resilient, production-grade storage for workloads.

2.3 Use cases

The combination of SUSE Virtualization, Portworx by Everpure, and Supermicro Compact Edge Servers addresses several critical enterprise use cases:

- **Application Modernization**
  Many organizations operate a mixed portfolio: legacy applications that are tightly coupled to virtual machines, and modern microservices written for Kubernetes. Migrating everything to containers in a single step can be risky and impractical. SUSE Virtualization allows both types of workloads to co-exist on the same cluster, with Portworx providing consistent storage semantics for both VM block volumes and container PersistentVolumeClaims.
- **Edge and Far-Edge Cost Optimization**

A retail chain with 500 stores, or a telecom operator with 1,000 base stations, would prefer to avoid the expense of three-server storage clusters at every location. With this TNA architecture, each site requires only two NVMe-equipped servers and one lightweight arbiter (witness) node. This can reduce per-site storage infrastructure cost by 25-35 percent.

- **High Availability Without Complexity**
  The Portworx storage layer performs synchronous data replication between the two storage worker nodes. If one node fails, the surviving node has a complete copy of all data and continues serving storage I/O without interruption. The arbiter ensures that the surviving node can make authoritative decisions without requiring a third storage node.

3 Architecture

The solution integrates SUSE Virtualization with Portworx Enterprise by Everpure on a Kubernetes-based platform to provide a unified infrastructure for running both virtualized and containerized workloads in edge and far-edge environments. It uses a Two-Node with Arbiter (TNA) topology, consisting of two worker nodes that provide compute and storage capabilities and serve as Kubernetes control-plane nodes, and a third arbiter (witness) node that participates in cluster quorum without hosting application workloads or storing data.

SUSE Virtualization enables the deployment and management of virtual machines alongside containerized applications within the same platform. Portworx Enterprise by Everpure provides container-native storage services, including persistent volumes, data replication, and high availability storage. Supporting this, Kubernetes orchestrates the deployment, scheduling, and lifecycle management of both workloads and storage services across the cluster.

Infrastructure hardware consists of compact edge servers with local NVMe storage and integrated networking, providing a resilient platform for deploying virtualized and containerized workloads in edge and far-edge environments.

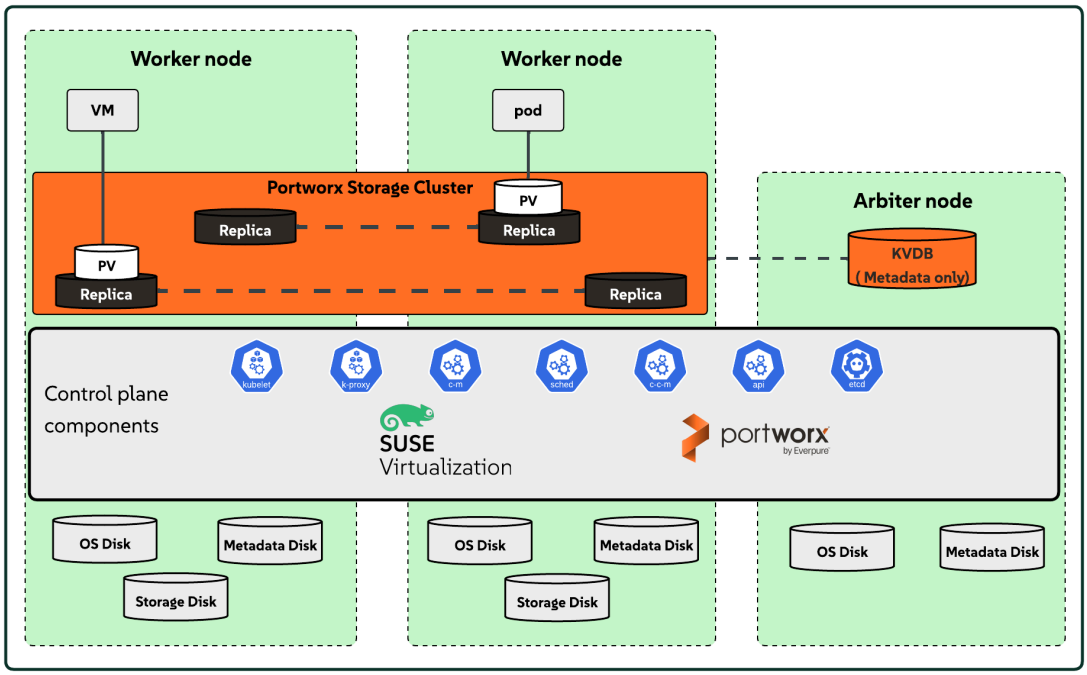

FIGURE 1: SUSE VIRTUALIZATION AND PORTWORX TWO NODE WITH ARBITER TOPOLOGY

Note

TNA Quorum Logic: Portworx requires a majority of key-value database (KVDB) members to agree before making cluster state changes.

In a TNA topology, a quorum of two is always achievable as long as either of the following conditions is met:

- Both storage nodes are functioning.
- One storage node and the arbiter node are functioning.

This quorum logic enables the cluster to survive a single storage node failure without split-brain scenarios or data loss.

3.1 SUSE Virtualization

SUSE Virtualization (https://www.suse.com/products/rancher/virtualization/) is a lightweight, hyperconverged infrastructure solution, enabling the management of virtual machines and containers side-by-side on a unified, cloud-native platform. To ensure a secure and optimized base,

SUSE Virtualization is deployed on SUSE Linux Micro, an immutable, enterprise-grade operating system purpose-built for containerized and virtual workloads. Furthermore, the platform integrates natively with SUSE Rancher Prime, equipping the solution with centralized authentication, access control, observability, and built-in security. This empowers operators to manage diverse workloads across data center and edge environments using a consistent operational model.

3.2 Portworx Enterprise by Everpure

Portworx (https://portworx.com/products/portworx-enterprise/) serves as the enterprise-grade storage layer for both VM and container workloads. It ensures consistent storage reliability across on-premises, hybrid, and edge deployments by delivering persistent volumes, high availability, automated failover, disaster recovery, performance optimization, and encryption. A critical component to this deployment is the Portworx Operator. Deployed into the SUSE Virtualization cluster, the Portworx Operator is designed to efficiently automate and manage the installation, configuration, and ongoing upgrade workflows of all Portworx components.

3.3 Supermicro Compact Edge Servers

Supermicro Compact Edge Servers (https://www.supermicro.com/en/products/edge/compact-edge-servers) provides the physical compute foundation for this reference design. The SYS-E403-14B-FRN2T (https://store.supermicro.com/us_en/iot-embedded-sys-e403-14b-frn2t.html) is a compact, short-depth edge server that is purpose-built for deployment in space- and power-constrained environments. This platform delivers enterprise-grade compute performance in a wall-mountable, fanless-compatible form factor suited for edge and far-edge locations. Each node is equipped with

- Intel Xeon 6700/6500 series processors,
- dual hot-swap NVMe bays,
- dual 10 GbE network ports,
- and a dedicated IPMI/BMC management interface

to provide the storage I/O throughput, network redundancy, and out-of-band manageability required to run production-grade Kubernetes and virtualized workloads at the edge. Although all three nodes use the same server platform, the arbiter node is configured with a different storage profile to handle quorum and metadata responsibilities, while the primary nodes provide the cluster's storage capacity.

4 Implementation considerations

The TNA topology introduces two implementation details that must be addressed.

1. StorageCluster Health-Check Annotations
   Portworx will fail to initialize if the Portworx preflight checker and health-check subsystem count fewer than three nodes with storage capacity.
   Because the arbiter intentionally hosts only metadata, these checks must be bypassed with annotations with a `StorageCluster` resource:

   ```
   metadata:
     annotations:
       portworx.io/health-check: "skip"
       portworx.io/preflight-check: "skip"
   ```

2. Disk Identification and Mixed Layouts
   On bare-metal servers, NVMe device identifiers (such as `nvme0n1` or `nvme1n1`) are determined by system firmware and PCIe slot positioning. This means that the identifiers can change. Because Portworx requires persistent identifiers, use persistent disk identifiers from the `/dev/disk/by-id/` directory (or file system UUIDs) when referencing storage devices. For example, use `/dev/disk/by-id/nvme-eui.36363530599182970025384e00000001-part1` and `/dev/disk/by-id/nvme-eui.36363530599182970025384e00000001-part2` instead of transient device names such as `/dev/nvme1n1p1` and `/dev/nvme1n1p2`. This ensures reliable disk mapping across reboots and helps prevent storage initialization failures.

5 Deployment

Portworx Enterprise by Everpure is a cloud-native storage and data management platform for Kubernetes. It offers native Container Storage Interface (CSI) integration, providing drivers that expose block and file storage systems to workloads running on Kubernetes, such as the contain-

ers and virtual machines in the SUSE Virtualization environment. The Portworx Operator is designed to efficiently manage the installation and upgrade workflow of all Portworx components. The following pages provide guidance for configuring and installing Portworx Enterprise via the Portworx Operator.

5.1 Installing and configuring SUSE Virtualization

1. Download the full ISO image for the supported SUSE Virtualization (Harvester) version from the Harvester GitHub repository (https://github.com/harvester/harvester/releases).
   Always use the full ISO (not the net-install variant) to ensure all container images critical for edge environments are available even in restricted Internet access environments.

2. Mount the ISO via IPMI Virtual Media (recommended) using the Supermicro BMC management port, or flash it to a USB drive that you can directly plug into the system.
   See SUSE Virtualization: Installation Methods (https://documentation.suse.com/cloudnative/virtualization/v1.7/en/installation-setup/methods/iso-install.html) for more information.

3. Enable UEFI boot mode, Intel VT-x, and VT-d in the Supermicro BIOS before booting.
   Legacy BIOS boot is deprecated in version 1.7 and later.

4. On the first node, select `Create a new Harvester cluster`.
   On subsequent nodes, select `Join an existing Harvester cluster` using the Cluster VIP join token generated after the first node boots.
   Assign the Witness role to the third node to designate it as a storageless, etcd arbiter.
   See SUSE Virtualization: Witness Node (https://documentation.suse.com/cloudnative/virtualization/v1.7/en/hosts/witness-node.html) for details.

5. When selecting disks, assign only the OS NVMe as the installation disk.
   The data NVMe must be left untouched by the Harvester installer, it will be partitioned manually for Portworx in the next section.

6. Assign static IPs to all nodes during installation.
   Node IPs cannot be changed after cluster formation without breaking the cluster.

7. Bond both 10 GbE ports on the node into a single management bond (`mgmt-bo`) for network redundancy.
   Physical switch ports must be configured as trunk ports.

See SUSE Virtualization: Cluster Network (https://documentation.suse.com/cloudnative/virtualization/v1.7/en/networking/cluster-network.html) for networking configuration details.

8. Configure a reliable NTP server during installation, as accurate time synchronization is mandatory for Kubernetes cluster stability.

5.2 Preparing for Portworx installation

Before deploying the Portworx Operator, three preparatory tasks must be completed on each storage worker node: partitioning the disks, wiping residual file system partition signatures, and installing the Augeas binary. These steps address hardware-level disk layout requirements and OS immutability, respectively.

5.2.1 Partitioning the disks

Portworx requires dedicated partitions on each data disk, one small partition for internal KVDB metadata and one large partition for storage pool data. A third minimal partition is created at the start of the disk as a protective alignment buffer. This must be performed on the designated data disk of each storage worker node before Portworx installation.

1. Create the three partitions using `parted`.

```
parted /dev/nvme1n1 \
    mklabel gpt \
    unit MiB \
    mkpart primary 1 2 \
    mkpart primary 2 65538 \
    mkpart primary 65538 100% \
    print \
    quit
```

2. Inform the kernel of the new partition table.

```
partprobe /dev/nvme1n1
```

3. Verify the partition layout with `lsblk`.

```
lsblk /dev/nvme1n1
```

You should see three partitions:

```
NAME                         MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINTS
nvme1n1                      259:0    0 894.3G  0 disk
├─nvme1n1p1                  259:1    0     1M  0 part
├─nvme1n1p2                  259:2    0    64G  0 part
└─nvme1n1p3                  259:3    0 830.3G  0 part
```

This partition layout maps directly to the `StorageCluster spec.nodes` configuration.

Important

Repeat this partitioning procedure on the designated data disk of each of the other nodes before proceeding. Do not perform this procedure on the OS boot disk. Verify the correct device with `lsblk` before executing `parted`.

5.2.2 Wiping partition signatures

After partitioning, wipe any residual file system signatures from the `p2` and `p3` partitions to ensure Portworx initializes them cleanly.

```
wipefs -a /dev/nvme1n1p2
wipefs -a /dev/nvme1n1p3
```

Confirm both partitions are clean (empty output from `wipefs` indicates no signatures remain):

```
wipefs /dev/nvme1n1p2 && echo "P2 CLEAN"
P2 CLEAN

wipefs /dev/nvme1n1p3 && echo "P3 CLEAN"
P3 CLEAN
```

5.2.3 Installing Augeas

SUSE Virtualization features SUSE Linux Micro as the host operating system. This provides an immutable root file system for added security and stability. During installation, the Portworx `px-storev2` engine performs a runtime dependency check that attempts to install the `augeas` package. Since `transactional-update` is not supported in SUSE Virtualization and `/usr/bin` is read-only, this action fails and Portworx pods enter a `CrashLoopBackOff` state.

Resolve this by manually extracting the `augeas` binary from an openSUSE Tumbleweed RPM and installing it into `/usr/local/bin`, which is writable on SUSE Virtualization and already included in the system `PATH` variable.

1. Download the Augeas RPM into a temporary directory.

   ```
   cd /tmp

   curl -LO \
   https://download.opensuse.org/tumbleweed/repo/oss/x86_64/augeas-<version>.x86_64.rpm
   ```

   Be sure to replace `<version>` in the above command to match the augeas package name.

2. Extract the RPM contents.

   ```
   mkdir -p /tmp/augeas-extract

   cd /tmp/augeas-extract

   rpm2cpio /tmp/augeas-<version>.x86_64.rpm | cpio -idm 2>/dev/null
   ```

3. Install the binaries into the writable path.

   ```
   for b in augtool augmatch augparse augprint fadot; do
     [ -f /tmp/augeas-extract/usr/bin/$b ] \
       && install -m 0755 /tmp/augeas-extract/usr/bin/$b /usr/local/bin/$b \
       && echo "Installed: $b" \
       || echo "Skipped: $b"
   done
   Installed: augtool
   ```

4. Verify the installation.

   ```
   which augtool && augtool --version && echo "=== SUCCESS ==="
   ```

   ```
   /usr/local/bin/augtool
   augtool 1.14.1
   === SUCCESS ===
   ```

Important

This installation must be repeated on both storage nodes and the arbiter node before applying the Portworx Operator. The binaries persist across pod restarts but may need to be reinstalled after a SUSE Virtualization OS upgrade. Be sure to include this step in your upgrade runbook.

5.3 Configuring the Portworx Operator

1. Log in to https://central.portworx.com/
2. Select *Portworx Enterprise* > *Generate Spec*.
3. Choose the supported Portworx Version and select `Others (or Generic/Upstream Kubernetes)` as the Platform, since this deployment uses local NVMe drives, not an Everpure FlashArray back-end.
4. Click *Customize* at the bottom of the menu.
5. After the Portworx version is validated, set the Kubernetes version to match the version running on your SUSE Virtualization cluster. Verify that the selected Portworx and Kubernetes versions are supported. Provide the desired namespace name (for example, `portworx`) and select **Built-in etcd** as the KVDB option.

Tip

You can verify compatibility between the Portworx Enterprise version and Kubernetes (RKE2) version by consulting the Portworx support matrix (https://docs.portworx.com/portworx-enterprise/support-matrix/supported-kubernetes-versions) and https://documentation.suse.com/cloudnative/virtualization/latest/en/installation-setup/requirements.html (https://www.suse.com/suse-harvester/support-matrix/all-supported-versions/).

For example, Portworx Enterprise 3.6 supports Kubernetes 1.34, which is included with SUSE Virtualization 1.7.

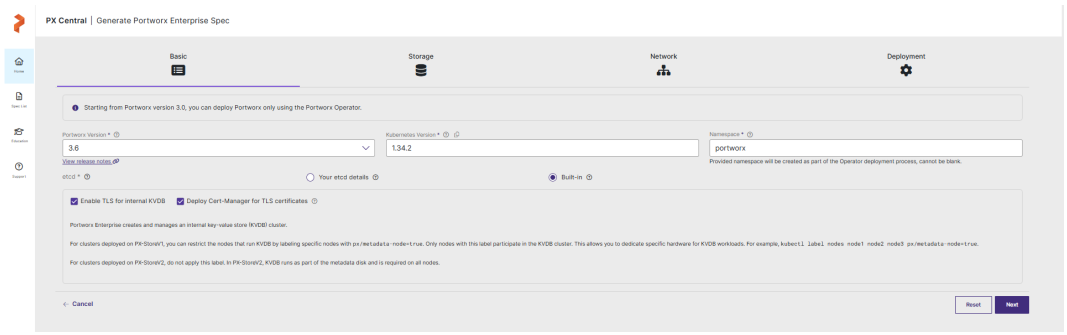

FIGURE 2: SUSE VIRTUALIZATION AND PORTWORX BASIC CONFIGURATION WITH BUILT-IN KVDB

6. In the Storage section, select Cloud Drive type as “None / Use Existing Disks” since Portworx will consume locally attached NVMe devices. Do not use the global "Use All Disks" option. Explicit per-node device paths will be defined directly in the StorageCluster manifest after download. Enable PX-StoreV2 as the storage engine. Set the maximum storage nodes per availability zone to 2, reflecting the two full storage worker nodes in the TNA topology (the arbiter node must not be counted as a storage node).

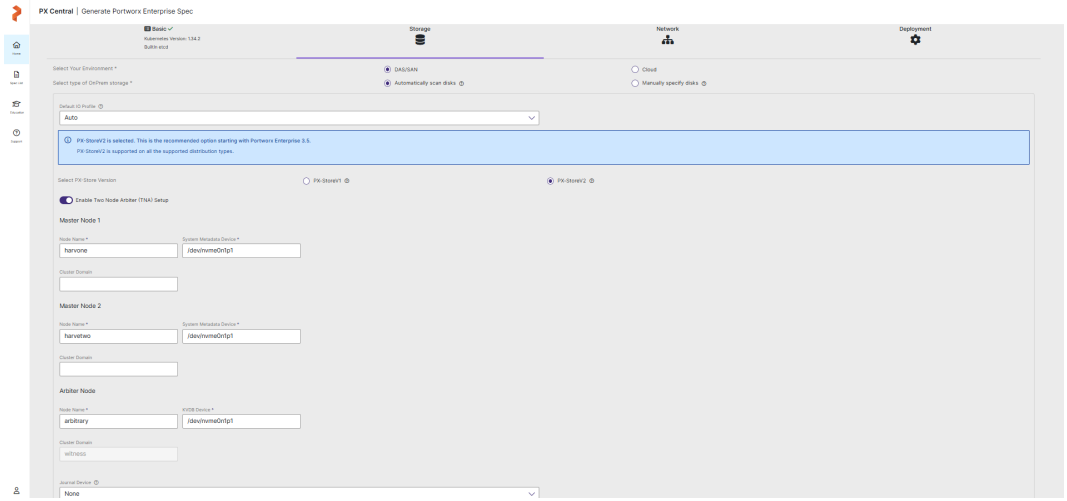

FIGURE 3: SUSE VIRTUALIZATION AND PORTWORX STORAGE CONFIGURATION

7. In the Network section, retain the default Portworx service port of 9001 unless your edge network environment requires a different port.

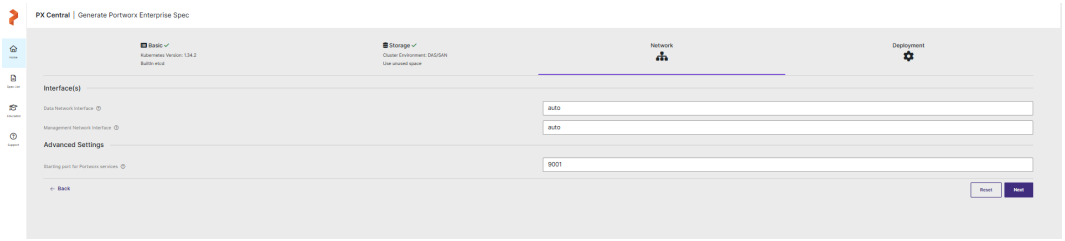

FIGURE 4: SUSE VIRTUALIZATION AND PORTWORX NETWORK CONFIGURATION

8. In the Customize section, select “Rancher Kubernetes Engine 2 (RKE2)” to match the Kubernetes engine embedded in SUSE Virtualization.

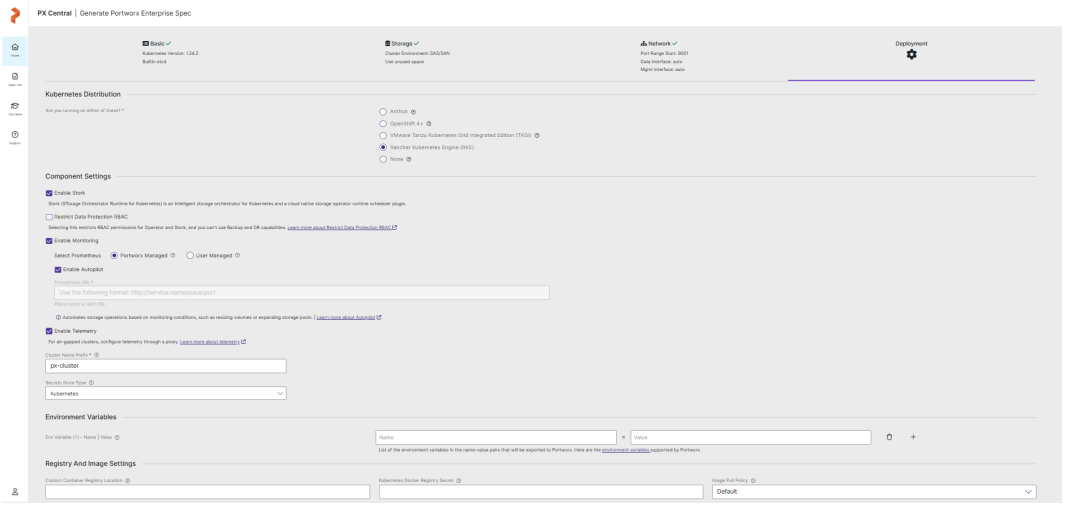

FIGURE 5: SUSE VIRTUALIZATION AND PORTWORX RKE2 CUSTOMIZATION

9. The spec is generated next and kubectl commands appear in the UI. Do not apply them yet. First, download and edit the Operator YAML to add the SUSE Virtualization-specific tolerations for node-role.kubernetes.io/control-plane (NoSchedule) and node-role.kubernetes.io/etcd (NoExecute). Without these, Portworx pods will not schedule on the arbiter or control-plane nodes. Save the commands for reference before making these edits.

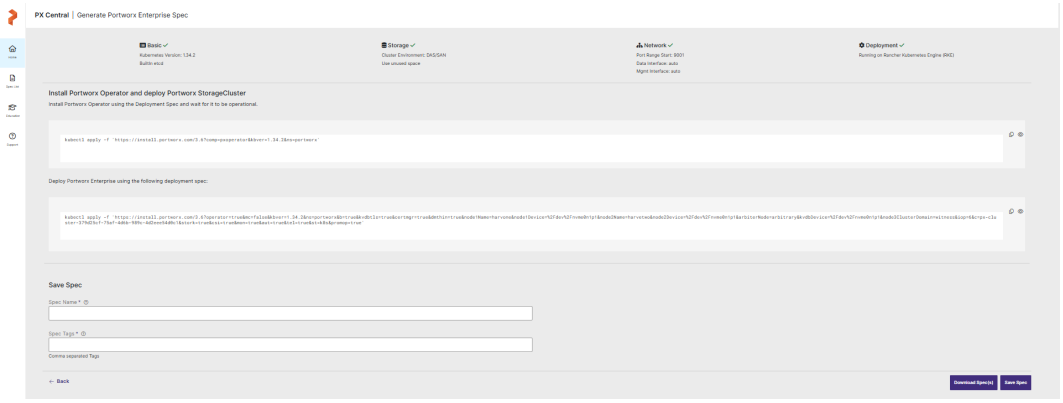

FIGURE 6: SUSE VIRTUALIZATION AND PORTWORX OPERATOR SPEC GENERATION

5.4 Deploying Portworx

Deploy Portworx involves three main steps:

1. Installing the Portworx Operator: The Portworx Operator efficiently automates and manages installation, configuration, and ongoing upgrade workflows for all Portworx components.
2. Creating a Stork Snapshot StorageClass: Stork adds support for orchestrating volume snapshots through Kubernetes, allowing users to take snapshots of PVCs and restore those snapshots to other PVCs through Kubernetes.
3. Creating the Portworx StorageCluster: The StorageCluster object acts as a definition for the Portworx cluster and provides a Kubernetes-native experience, allowing you to manage your Portworx cluster like other Kubernetes applications.

These steps are covered in detail in the following sections.

5.4.1 Installing the Portworx Operator

1. Log in to the SUSE Virtualization management node command line interface (CLI) with root access.
2. Download the Portworx Operator manifest.

```
curl -o portworx-operator.yaml 'https://install.portworx.com/<px-version>?
comp=pxoperator&kbver=<k8s-version>&ns=portworx'
```

Be sure to replace `<px-version>` and `<k8s-version>` in the above command with the supported versions you are using. For example, '3.6' and '1.34.2', respectively.

3. Apply the manifest to install the Portworx Operator.

```
kubectl apply -f portworx-operator.yaml
```

You should see the following output confirming the Operator is deployed.

```
namespace/portworx created
serviceaccount/portworx-operator created
clusterrole.rbac.authorization.k8s.io/portworx-operator created
clusterrolebinding.rbac.authorization.k8s.io/portworx-operator created
deployment.apps/portworx-operator created
```

4. Verify that the Operator pod reaches `Running` state before proceeding.

```
kubectl get pods -n portworx
```

```
NAME                                READY   STATUS    RESTARTS   AGE
portworx-operator-f466b7f5b-l5586   1/1     Running   0          2m
```

5.4.2 Creating the Stork Snapshot StorageClass

1. Create `stork-snapshot-sc.yaml` with the following contents:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: stork-snapshot-sc
  annotations:
    cdi.harvesterhci.io/storageProfileVolumeModeAccessModes: |
      {"Block":["ReadWriteOnce"]}
provisioner: stork-snapshot
allowVolumeExpansion: true
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

2. Apply the StorageClass.

```
kubectl apply -f stork-snapshot-sc.yaml
```

```
storageclass.storage.k8s.io/stork-snapshot-sc created
```

5.4.3 Creating the Portworx StorageCluster

1. Download the StorageCluster manifest.

```
curl -o new_stc.yaml 'https://install.portworx.com/<px-version>?
operator=true&mc=false&kbver=<k8s-version>&ns=portworx&b=true&iop=6&mz=2&c=px-
cluster-tna&stork=true&csi=true&mon=true&tel=true&st=k8s&promop=true'
```

   Be sure to replace `<px-version>` and `<k8s-version>` in the above command with the supported versions you are using. For example, '3.6' and '1.34.2', respectively.

2. Open `new_stc.yaml` and apply the following TNA-specific modifications before deploying. For reference, see example file *Section 12.1.1, "Sample StorageCluster manifest for TNA"*.

   a. Add the following annotations under `metadata.annotations` to prevent false node-count failures, since the arbiter is storageless:

```
portworx.io/health-check: "skip"
portworx.io/preflight-check: "skip"
```

   b. Under `spec` add the following tolerations block to allow Portworx agent pods to be scheduled on the arbiter node.

```
tolerations:
  - key: "node-role.kubernetes.io/control-plane"
    operator: "Exists"
    effect: "NoSchedule"
  - key: "node-role.kubernetes.io/etcd"
    operator: "Exists"
    effect: "NoExecute"
```

3. Apply the edited `StorageCluster` manifest.

```
kubectl apply -f new_stc.yaml -n portworx
```

   You should see:

```
storagecluster.core.libopenstorage.org/px-cluster-tna created
```

4. Monitor the Portworx pods as they initialize.
   Full start-up typically takes 3–5 minutes on NVMe hardware.

```
kubectl get pods -n portworx -w
```

```
NAME                                READY   STATUS    RESTARTS   AGE
```

```
portworx-operator-f466b7f5b-l5586   1/1     Running   0          10m
px-cluster-tna-25284                1/1     Running   0          4m
px-cluster-tna-5cmlx                1/1     Running   0          4m
```

5.5 Updating the CSI driver configuration

The final step is to configure SUSE Virtualization to use the Portworx cluster.

1. Log in to the SUSE Virtualization UI.
2. Navigate to *Advanced* > *Settings*.
3. Find and select *csi-driver-config*.
4. Select # > *Edit Setting* to access the configuration options.
5. Set the Provisioner to the third-party CSI driver, `pxd.portworx.com`.
6. Configure the Volume Snapshot Class Name to `px-csi-snapclass`.
   This setting points to the name of the `VolumeSnapshotClass` used for creating volume snapshots or VM snapshots.

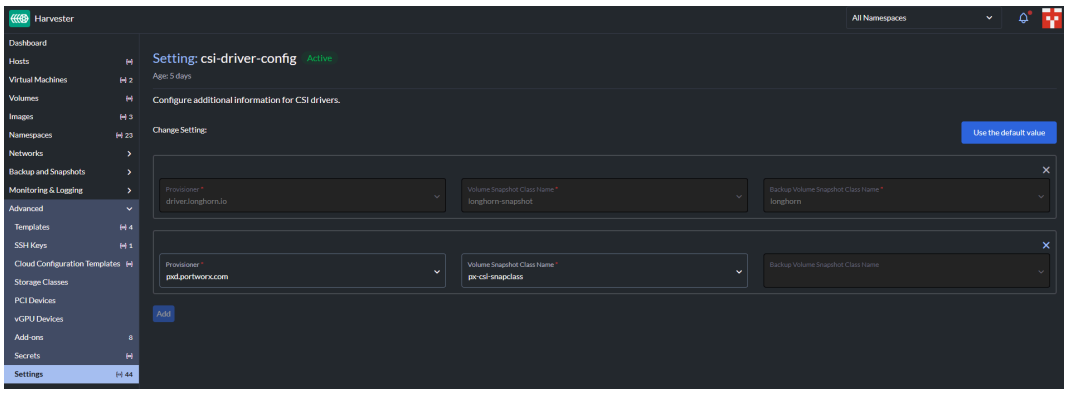

FIGURE 7: SUSE VIRTUALIZATION AND PORTWORX VOLUME SNAPSHOT CONFIGURATION

7. Find and select *csi-online-expand-validation*.
8. Select # > *Edit Setting* to access the configuration options.
9. Set the Provisioner to the third-party CSI driver, `pxd.portworx.com`, and the value to `True`.

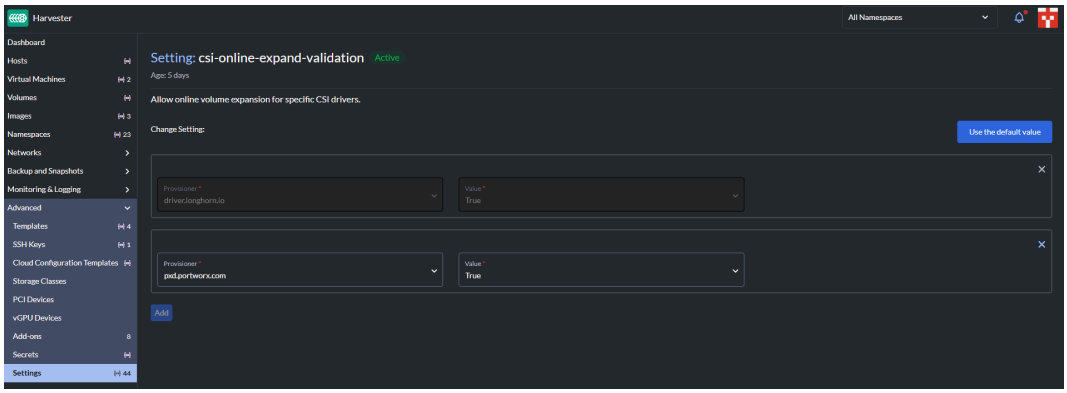

FIGURE 8: SET PROVISIONER TO PXD.PORTWORX.COM AND ENABLE TRUE

6 Validating the TNA deployment

You can check that the Portworx cluster is operational in the TNA topology with the steps below.

1. Verify that all Portworx pods are in the `Running` state with no `CrashLoopBackOff` or `Error` conditions.

```
kubectl get pods -n portworx
```

```
NAMESPACE   NAME                                READY   STATUS    RESTARTS     AGE
portworx    portworx-operator-f466b7f5b-l5586   1/1     Running   1(9m6s)      60m
portworx    px-cluster-tna-25284                1/1     Running   0            32m
portworx    px-cluster-tna-5cmlx                1/1     Running   0            32m
```

2. Check that `pxctl status` reports the correct TNA topology with the two storage nodes and one storageless arbiter node online.

```
pxctl status
```

```
Status: PX is operational
Telemetry: Healthy
Metering: Disabled or Unhealthy
License: Trial (expires in 27 days)
Node ID: a910fb93-e66f-45d0-830f-015b11c5a657
        IP: 172.24.131.45
        Local Storage Pool: 1 pool
        POOL    IO_PRIORITY     RAID_LEVEL      USABLE  USED    STATUS  ZONE
 REGION
        0       HIGH            raid0           811 GiB 36 GiB  Online  default
 default
```

```
        Local Storage Devices: 1 device
        Device  Path            Media Type              Size            Last-Scan
        0:0     /dev/nvme1n1p3  STORAGE_MEDIUM_NVME     830 GiB         09 May 26
 14:30 UTC
        total                   -                       830 GiB
        Cache Devices:
         * No cache devices
        Metadata Device:
        1       /dev/nvme1n1p2  STORAGE_MEDIUM_NVME     64 GiB
        * Internal kvdb on this node is using this dedicated metadata device to
 store its data.

Cluster Summary
        Cluster ID: px-cluster-tna
        Cluster UUID: 43513271-7de9-4ea6-85ea-ec599476ba92
        Scheduler: kubernetes
        Total Nodes: 3 node(s) with storage (3 online)

        IP              ID                                      SchedulerNodeName
     Auth            StorageNode     Used    Capacity        Status  StorageStatus
 Version         Kernel               OS
        172.24.131.46   f11b2ad2-6029-405d-bda3-7dece346178f    harvtwo
     Disabled        Yes(PX-StoreV2) 36 GiB  811 GiB         Online  Up
 3.6.0.0-a81cf43 6.4.0-36-default     Harvester v1.7.0-rc5
        172.24.131.47   b893ea8a-141b-47d3-baa8-6073e01f53ec    arbiter
     Disabled        Yes(PX-StoreV2) 0 B     0 B             Online  No Storage
 3.6.0.0-a81cf43 6.4.0-36-default     Harvester v1.7.0-rc5
        172.24.131.45   a910fb93-e66f-45d0-830f-015b11c5a657    harvone
     Disabled        Yes(PX-StoreV2) 36 GiB  811 GiB         Online  Up (This node)
 3.6.0.0-a81cf43 6.4.0-36-default     Harvester v1.7.0-rc5

Global Storage Pool
        Total Used      :  72 GiB
        Total Capacity  :  1.6 TiB

Collected at: 2026-05-11 10:19:51 UTC
```

Notice in the individual node details that:

- `harvone` and `harvetwo` (the storage nodes) show a StorageStatus of Up with PX-StoreV2 engine, 811 GiB of total capacity.
- `arbiter`: Online with `No Storage`, participating in KVDB quorum as a storageless arbiter, exactly as designed.
- `Arbiter` (the witness node) shows Online with a StorageStatus of No Storage, since it only participates in KVDB quorum.

Note

The `pxctl status` output reports three online Portworx cluster, including the arbiter node. However, the arbiter is a storageless witness node that participates only in KVDB quorum and metadata services. Storage capacity is provided solely by `harvone` and `harvtwo`.

6.1 Resilience validation

Validate TNA high availability by testing the following two failure scenarios with active VM workloads.

6.1.1 Scenario 1: Storage Node Failure with Active VMs

1. Create six, Portworx-backed VMs (vm1, vm2, …, vm6) distributed across both nodes, harvone and harvtwo.
2. Launch a heartbeat logger in a VM (vm1) on harvone so that it writes to the Portworx-backed volume every second.
3. Shut down harvone abruptly.

   ```
   harvone:~ # shutdown -h now
   ```

   When Kubernetes detects harvone as NotReady, it triggers live migration for all VMs that were running on it. Within approximately 2 minutes, you should see all VMs are Running on harvtwo.

4. Verify that harvone is marked as Offline.

```
harvetwo:~ # pxctl status
```

```
Status: PX is operational
Total Nodes: 3 node(s) with storage (2 online)

harvtwo    Online   Up (This node)
arbiter    Online   No Storage
harvone    Offline  Down
```

5. Verify that all VMs are running on harvtwo.

```
harvetwo:~ # kubectl get vmi -A -o wide
```

```
NAMESPACE   NAME        PHASE     IP           NODENAME   READY   LIVE-MIGRATABLE
default     vm1         Running   10.52.1.72   harvtwo    True    True
default     vm2         Running   10.52.1.73   harvtwo    True    True
default     vm3         Running   10.52.1.63   harvtwo    True    True
default     vm4         Running   10.52.1.61   harvtwo    True    True
default     vm5         Running   10.52.1.69   harvtwo    True    True
default     vm6         Running   10.52.1.66   harvtwo    True    True
```

6. Check the heartbeat log from inside vm1 to determine the observed I/O interruption across the entire failover event.

```
vm1:~ # awk '...' /root/hbtest/hb.log
```

```
Downtime: 1 seconds
From 1778873153 to 1778873155

TOTAL downtime: 1 seconds
```

Note

In the authors' validation testing, the full recovery timeline from node failure until all VMs were confirmed as Running and Ready on harvtwo was approximately 2 minutes.

6.1.2 Scenario 2: Cascading Failure: Arbiter Then Storage Node

1. With the cluster fully healthy and all six VMs running across both nodes, shut down the arbiter node (arbiter) first. Verify that both storage nodes remain operational and the cluster continues serving workloads normally the two storage nodes together hold majority quorum without the arbiter.

```
harvone:~ # kubectl get nodes -o wide
```

```
NAME        STATUS     ROLES                AGE   VERSION
arbiter     NotReady   etcd                 11d   v1.34.2+rke2r1
harvone     Ready      control-plane,etcd   11d   v1.34.2+rke2r1
harvtwo     Ready      control-plane,etcd   11d   v1.34.2+rke2r1
```

2. With the arbiter already offline, launch a heartbeat logger inside a VM (vm1) running on harvone so that it writes to its Portworx-backed volume every second.

3. Now shut down harvone abruptly, creating the cascading failure condition.

```
harvone:~ # shutdown -h now
```

4. When Kubernetes detects harvone as NotReady, it triggers live migration for all VMs that were running on it. Verify that all VMs are Running on harvtwo.

```
harvetwo:~ # kubectl get vmi -A -o wide
```

```
NAMESPACE   NAME        PHASE     IP            NODENAME   READY   LIVE-MIGRATABLE
default     vm1         Running   10.52.1.112   harvtwo    True    True
default     vm2         Running   10.52.1.113   harvtwo    True    True
default     vm3         Running   10.52.1.110   harvtwo    True    True
default     vm4         Running   10.52.1.111   harvtwo    True    True
default     vm5         Running   10.52.1.109   harvtwo    True    True
default     vm6         Running   10.52.1.66    harvtwo    True    True
```

5. Verify that Portworx remains operational on harvtwo with harvone shown as Offline.

```
harvetwo:~ # pxctl status
```

```
Status: PX is operational
Total Nodes: 3 node(s) with storage (2 online)
harvtwo    Online   Up (This node)
arbiter  Offline  No Storage
```

```
harvone    Offline  Down
```

6. Check the heartbeat log from inside vm1 to determine the observed I/O interruption across the entire cascading failure and recovery window.

```
vm1:~ # awk '...' /root/hbtest/hb.log
```

```
Downtime: 1 seconds
Downtime: 1 seconds
Downtime: 1 seconds
Downtime: 1 seconds
Downtime: 1 seconds
Downtime: 1 seconds
Downtime: 1 seconds
Downtime: 1 seconds
TOTAL downtime: 8 seconds
```

Note

In the authors' validation testing, the total observed I/O interruption across the cascading failure event was 8 seconds. The additional interruption compared to Scenario 1 reflects the time required for etcd to detect the second failure and for harvtwo to re-establish leadership without the arbiter.

During this period, the Kubernetes API server briefly reported `etcdserver: no leader` while leadership was being re-established. This is expected behaviour in a cascading failure scenario. Once harvtwo assumed sole etcd authority, the API server recovered and all VM workloads resumed normally on harvtwo. The two storage nodes together hold majority quorum (2 of 3 KVDB members) and can continue operating without the arbiter.

7 Upgrading and Lifecycle Management

Lifecycle management is an important consideration to ensure continuous platform interoperability, maintain data availability for stateful workloads, and close critical security gaps. The components of this solution run natively on a unified Kubernetes-based control plane, making coordinated lifecycle management essential for maintaining compatibility and operational stability.

Proper lifecycle management impacts this solution across several key operational areas:

7.1 Sustaining Hyperconverged Interoperability

- **API synchronization:** SUSE Virtualization and Portworx Enterprise by Everpure interact natively through the Container Storage Interface (CSI). Maintaining supported software versions helps prevent data-path failures caused by incompatible Kubernetes dependencies.
- **Operating system compatibility:** SUSE Virtualization nodes utilize SUSE Linux Micro. Lifecycle management helps ensure that Portworx kernel components remain compatible with the host operating system.

7.2 Guaranteeing Zero-Downtime Operations

- **Live VM migrations:** Keeping both platforms updated helps maintain advanced day-2 operations, such as live migration of running virtual machines between hosts without service interruption.
- **Zero-downtime data upgrades:** Modern Portworx capabilities enable rolling storage updates without resetting storage services or disrupting active virtual disks.

7.3 Activating Modern Resilience Features

- **Disaster recovery enhancements:** Advanced capabilities such as synchronous and asynchronous disaster recovery configurations depend on compatible versions of Stork and Portworx Enterprise.
- **Dynamic storage scaling:** New releases continue to introduce automation and scalability improvements that simplify storage expansion as infrastructure requirements grow.

7.4 Mitigating Risks and Lowering TCO

- **Automated security patches:** Regular lifecycle maintenance helps ensure that infrastructure components receive security updates and vulnerability fixes.
- **Workflow efficiency:** Standardized lifecycle management processes reduce operational overhead and improve consistency across distributed deployments.

For supported upgrade paths, compatibility requirements, and operational guidance, refer to the following documentation:

- SUSE Virtualization Documentation: https://documentation.suse.com/cloudnative/virtualization/
- Portworx Enterprise by Everpure Documentation: https://portworx.com/products/portworx-enterprise/

8 Summary

This document provides a reference design with step-by-step, implementation guidance for integrating SUSE Virtualization and Portworx Enterprise by Everpure in a Two-Node with Arbiter (TNA) topology on Supermicro Compact Edge Servers. The architecture delivers highly available, enterprise-grade storage at the edge using only two full storage nodes and one lightweight arbiter node to dramatically reduce hardware costs while preserving production-level resilience.

Organizations extending workloads beyond centralized data centers face a familiar dilemma: how do you maintain high availability, minimize hardware expenditure, and operate with limited on-site expertise. Traditional storage HA mandates at least three full storage nodes. The TNA architecture dissolves this constraint by leveraging an arbiter node, carrying no workload data while participating only in quorum decisions. This can dramatically cut hardware costs at scale while retaining full fault-tolerance.

By combining SUSE Virtualization, Portworx, and Supermicro Compact Edge Servers in this TNA topology, organizations create a scalable, resilient, and cost-efficient solution for edge and other use cases.

9 Frequently Asked Questions (FAQs)

1. **What exactly is the Two-Node with Arbiter (TNA) topology?**
   TNA is a Portworx deployment topology designed for resource-constrained environments. It consists of exactly two full storage worker nodes that hold application data with synchronous replication between them, plus one lightweight arbiter node. The arbiter participates in the internal Portworx KVDB quorum as a tie-breaker during network partitions or node failures. This means the cluster can achieve a quorum majority (2 of 3 KVDB members) even when one storage node is unavailable. The arbiter does not store any application data, making it suitable for deployment on a smaller, less expensive server.

2. **Why is third-party storage support important for SUSE Virtualization?**
   Broad third-party storage support gives customers the flexibility to leverage existing storage investments and select solutions that best match their specific performance, cost, and operational requirements. It eliminates vendor lock-in, enabling organizations to adopt best-of-breed storage technologies, such as Portworx for cloud-native HA storage, without being constrained to a single vendor's ecosystem. This is particularly valuable in edge environments, where cost sensitivity and integration with legacy infrastructure are primary concerns.

3. **Can the arbiter node be virtualized?**
   Yes. The arbiter node's role is computationally lightweight and it only participates in KVDB quorum decisions. It does not serve storage I/O or run application workloads. This makes it a good candidate for virtualization, a small server, or an existing edge gateway appliance, provided it meets the minimum supported hardware requirements for SUSE Virtualization (https://documentation.suse.com/cloudnative/virtualization/latest/en/installation-setup/requirements.html). Because the arbiter stores only KVDB metadata and quorum information, it does not require the same storage capacity as the Portworx storage nodes.

4. **What happens if both a storage node and the arbiter fail simultaneously?**
   With only one of three KVDB members available, quorum cannot be achieved, and the cluster enters a read-only protective state to prevent data divergence (split-brain). This is the correct and expected behavior. Recovery requires restoring at least one failed member. For edge environments, maintain an out-of-band management path (IPMI/BMC) to each node to facilitate remote recovery.

10 References

- SUSE Virtualization Documentation https://www.suse.com/products/rancher/virtualization/
- Portworx Enterprise Documentation https://portworx.com/products/portworx-enterprise/
- Portworx Central (Operator/StorageCluster manifest generation)https://central.portworx.com/
- RKE2 Documentation https://documentation.suse.com/cloudnative/rke2/
- openSUSE Augeas RPM Repository https://download.opensuse.org/tumbleweed/repo/oss/x86_64/

11 Glossary

Term	Definition
TNA	Two-Node with Arbiter. A Portworx deployment topology using 2 storage nodes + 1 storageless arbiter for HA quorum.
HCI	Hyperconverged Infrastructure. A software-defined architecture that combines compute, storage, and networking in a single system.
Arbiter	Also called Witness Node. A storageless Portworx node that participates in KVDB quorum decisions without holding application data.
KVDB	Key-Value Database. Portworx's internal distributed store for cluster metadata and state management.
CSI	Container Storage Interface. A Kubernetes-standard API for storage drivers to expose persistent storage to containerized workloads.
RKE2	Rancher's hardened Kubernetes distribution, used as the container orchestration engine inside SUSE Virtualization.
SUSE Linux Micro	SUSE Linux Micro. An immutable, minimal OS base with a read-only root filesystem, used by SUSE Virtualization.

Term	Definition
PX-StoreV2	Portworx storage engine version 2. Used for NVMe-optimized, high-performance block storage in this deployment.
Augeas	A configuration management library. Required by Portworx `px-storev2` for OS-level configuration parsing.
toleration	A Kubernetes pod specification field that allows a pod to schedule onto nodes with matching taints.
taint	A Kubernetes node property that repels pods unless those pods carry a matching tolerance.
NVMe	Non-Volatile Memory Express. A high-performance storage protocol for SSD and flash storage over PCIe.

12 Appendix

The following sections provide additional information and resources.

12.1 Sample files and output

12.1.1 Sample StorageCluster manifest for TNA

```
apiVersion: core.libopenstorage.org/v1
kind: StorageCluster
metadata:
  name: px-cluster-tna-gk
  namespace: portworx
  annotations:
    portworx.io/install-source: "https://install.portworx.com/3.6?
operator=true&mc=false&kbver=1.34.2&ns=portworx&b=true&kvdbtls=true&certmgr=true&dmthin=true&node1Name=
%2Fdev%2Fnvme0n1&node1ClusterDomain=Node1&node2Name=harvetwo&node2Device=
%2Fdev%2Fnvme1n1&node2ClusterDomain=Node2&arbiterNode=arbiter&kvdbDevice=
%2Fdev%2Fnvme1n1&node3ClusterDomain=witness&iop=6&c=px-cluster-
tna&stork=true&csi=true&mon=true&aut=true&tel=true&st=k8s&promop=true"
    portworx.io/health-check: "skip"
    portworx.io/preflight-check: "skip"
    portworx.io/misc-args: "-rt_opts small_conf=1 -T px-storev2"

spec:
  image: docker.io/portworx/oci-monitor:3.6.0
  imagePullPolicy: Always

  kvdb:
    internal: true
    enableTLS: false

  secretsProvider: k8s

  nodes:
    - clusterDomain: "Node1"
      selector:
        nodeName: "harvone"
      storage:
        systemMetadataDevice: /dev/nvme1n1p1
        devices:
          - /dev/nvme1n1p2
        useAll: false
```

```
  - clusterDomain: "Node2"
    selector:
      nodeName: "harvtwo"
    storage:
      systemMetadataDevice: /dev/nvme1n1p1
      devices:
        - /dev/nvme1n1p2
      useAll: false

  - clusterDomain: "witness"
    selector:
      nodeName: "arbiter"
    storage:
      kvdbDevice: /dev/nvme1n1p2
      useAll: false

placement:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
        - matchExpressions:
            - key: px/enabled
              operator: NotIn
              values:
                - "false"
  tolerations:
    - key: "node-role.kubernetes.io/etcd"
      operator: "Exists"
      effect: "NoExecute"
    - key: "node-role.kubernetes.io/control-plane"
      operator: "Exists"
      effect: "NoSchedule"
    - key: "node-role.kubernetes.io/master"
      operator: "Exists"
      effect: "NoSchedule"

stork:
  enabled: true
  args:
    webhook-controller: "true"

autopilot:
  enabled: true
  prometheusAlerts: true
  providers:
    - name: default
```

```
      type: prometheus
      params:
        url: http://px-prometheus:9090

runtimeOptions:
  default-io-profile: "6"

csi:
  enabled: true
  installSnapshotController: true

monitoring:
  telemetry:
    enabled: true
    metricsCollector:
      enabled: true
  prometheus:
    enabled: true
    exportMetrics: true

certManager:
  enabled: false

deleteStrategy:
  type: UninstallAndWipe
  ignoreVolumes: true

revisionHistoryLimit: 10
startPort: 9001

updateStrategy:
  rollingUpdate:
    disruption:
      allow: true
    maxUnavailable: 1
  type: RollingUpdate
```

Note

The sample configuration explicitly defines the storage devices used by each node by setting `useAll: false` and specifying the metadata and data devices. If you want Portworx to automatically consume all available storage devices on a node, set `useAll: true` and omit the explicit device list. Review the available storage devices carefully before enabling this option.

12.2 Server hardware specifications

All three nodes in this reference deployment are built on the Supermicro SYS-E403-14B-FRN2T, a compact IoT/Edge SuperServer from Supermicro's Compact Edge Server product line. This platform was selected for its short-depth form factor, NVMe storage support, and suitability for edge and far-edge environments where rack space and power budgets are constrained.

Component	Specification	Qty	Notes
Server Platform	Supermicro SYS-E403-14B-FRN2T	3	2x Storage Workers (`harveone`, `harvetwo`) + 1x Arbiter (`arbiter`)
Form Factor	Compact / Short-Depth (10.5" x 4.62" x 16")	—	Wall-mountable; suitable for shallow racks and edge cabinets
CPU	Intel Xeon 6700/6500 Series (P-core or E-core)	1/node	Up to 300W TDP with air cooling
RAM	DDR5 RDIMM, 8 DIMM slots per node	—	Up to 2TB per node; configured per workload requirements
NVMe Drive Bays	2x Front Hot-swap 2.5" NVMe	2/node	One bay for OS; one bay for Portworx data disk
SATA Drive Bays	2x Internal Fixed 2.5" SATA	2/node	Available for additional storage; not used in this deployment
Network Ports	2x 10GbE RJ45 LAN	2/node	One port for cluster/storage traffic; one for management
Management Port	Dedicated IPMI / BMC	1/node	Out-of-band management for remote recovery at edge sites
PCIe Expansion	3x PCIe 5.0 x16 FHFL	—	Available for GPU/NIC expansion; not used in this deployment
Display / USB	1x VGA, 4x USB 3.2 Gen1, 2x USB 2.0	—	Local console access during initial setup

Note

The arbiter node participates only in KVDB quorum and does not host workloads or contribute to storage capacity. Therefore, it can be sized smaller than the storage nodes. However, it must still meet the minimum supported hardware requirements for SUSE Virtualization (https://documentation.suse.com/cloudnative/virtualization/latest/en/installation-setup/requirements.html).

12.3 Per-node storage layout

Each storage worker node (`harveone`, `harvetwo`) carries two 2.5" NVMe drives. The data NVMe is partitioned into three GPT partitions as described in the Pre-Installation Preparation section. The arbiter node uses the same server platform but is configured as a storageless KVDB witness. It does not contribute NVMe capacity to the Portworx storage pool and can be provisioned with a smaller NVMe drive.

Partition	Size	Portworx Role	StorageCluster Field
`p1`	~1 MiB	GPT alignment buffer (unused by Portworx)	Not referenced
`p2`	~64 GiB	Portworx KVDB metadata store	`systemMetadataDevice`
`p3`	Remaining space	Portworx storage pool data	`devices`

13 Legal notice

14 GNU Free Documentation License

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all

Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

```
Copyright (c) YEAR YOUR NAME.
   Permission is granted to copy, distribute and/or modify this document
   under the terms of the GNU Free Documentation License, Version 1.2
```

```
   or any later version published by the Free Software Foundation;
   with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
   A copy of the license is included in the section entitled “GNU
   Free Documentation License”.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “ with…Texts.” line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
   Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.