



Multi-Tenant Virtualization with vCluster, KubeVirt and Lightbits

Date: Sep 29, 2026

Author: Rob Bloemendal

Abstract

Organizations that host workloads for many teams or customers want three things from their platform: one place to run both containers and virtual machines (VMs), strong isolation between tenants, and the freedom to maintain hardware without interrupting anyone's workloads.

This paper shows how vCluster, KubeVirt and Lightbits software-defined NVMe® over TCP storage work together to deliver all three. vCluster gives every tenant its own Kubernetes control plane and dedicated nodes. KubeVirt runs VMs alongside containers on those nodes. Lightbits SDS™ provides one shared, high-performance block storage pool, whose ReadWriteMany volumes let running VMs move between nodes with no downtime.

We describe the architecture, how tenants stay isolated, how live migration works, and where this approach fits for service providers and enterprise platform teams. A reference deployment, validated on a three-node environment, is included in the appendix.

Table of contents

Abstract	1
Executive Summary	3
The Challenges Platform Teams Face	3
How it Works	4
Keeping Every Tenant Isolated	5
Zero-Downtime Maintenance with Live Migration	6
Where It Fits	6
Getting started	7
Summary and Next Steps	7
Appendix A: Reference Deployment	8
A.1 Tenant Clusters with Private Nodes	8
A.2 Node Preparation	9
A.3 Lightbits Projects and Storage Per Tenant	10
Step 1: Create a Project and Credential per Tenant.	10
Step 2: Issue a JWT Per Tenant.	11
Step 3: Store the JWT as a Secret in the Tenant Cluster.	11
Step 4: Create the StorageClass.	11
A.4 KubeVirt and CDI	12
A.5 Test VM and Live Migration	13
Appendix B: Deployment Tips	16
Appendix C: Readiness Checklist	17
About Lightbits Labs™	17

Executive Summary

With vCluster, KubeVirt and Lightbits SDS, you can give every tenant a private Kubernetes cluster that runs containers and virtual machines side by side, on shared storage, with zero-downtime maintenance.

For your tenants, it feels like having a cluster of their own: their own control plane, their own nodes, and the freedom to install the tools they need. For your platform team, it means one storage pool to run, one operating model for VMs and containers, and hardware maintenance that no longer needs a maintenance window.

What you gain:

- **Isolation your tenants can trust.** Each tenant has a separate control plane, dedicated compute and its own storage credentials.
- **VMs that keep running.** Running VMs move between nodes during maintenance or rebalancing, with no reboot and no data copy.
- **Simpler, more efficient storage.** One disaggregated Lightbits NVMe/TCP pool serves every tenant over standard Ethernet, so compute nodes need no local disks for VM data.
- **A path off legacy virtualization.** VMs and containers share one Kubernetes-based platform, instead of two separate stacks.

The Challenges Platform Teams Face

Most teams hosting many tenants end up trading isolation against cost, or containers against VMs, because the usual options each fall short somewhere.

Approach	Tenant Isolation	VMs and Containers on One Platform	Live Migration
Namespaces in one shared cluster	Limited: tenants share one API server and cluster-wide objects	Yes	Yes, with shared storage
A physical cluster per tenant	Strong	Yes	Yes, with shared storage
A separate hypervisor platform next to Kubernetes	Strong	No: two stacks to run and staff	Yes
vCluster + KubeVirt + Lightbits	Strong: own control plane and nodes	Yes	Yes, built-in



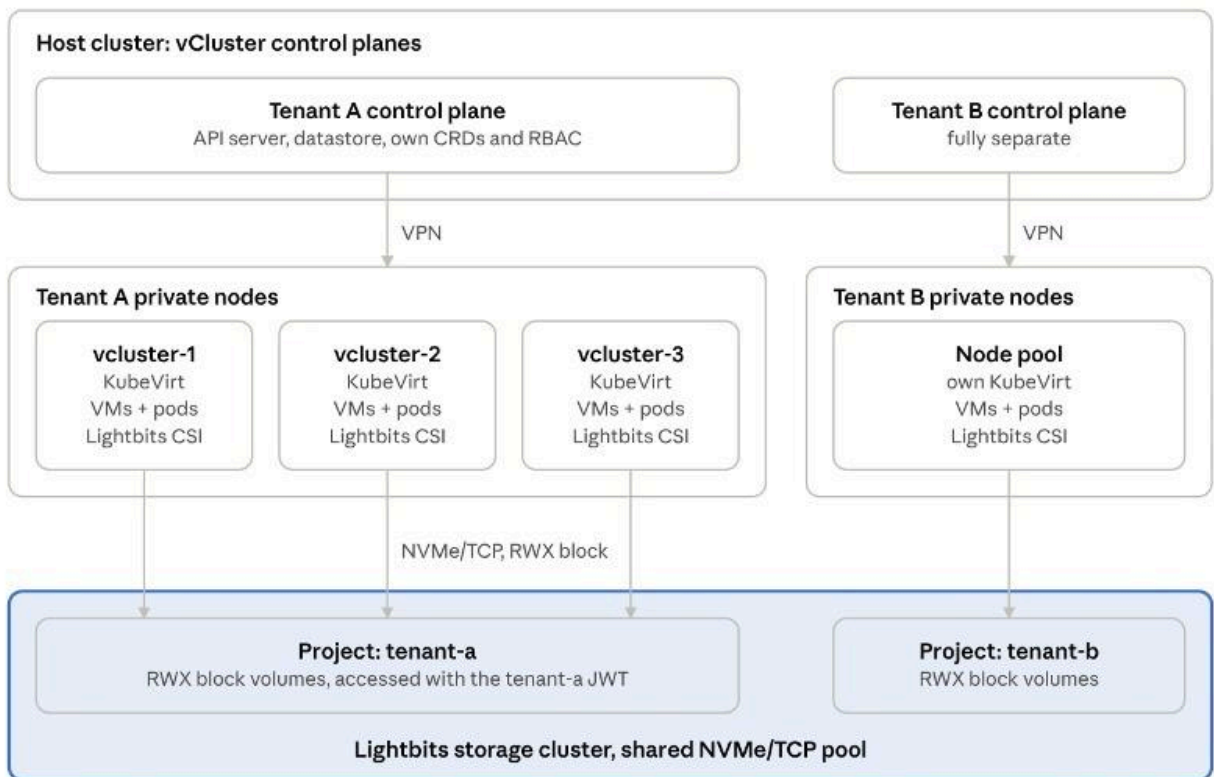
Shared namespaces are economical, but tenants cannot install their own operators or custom resources, and one tenant's changes can affect others. A physical cluster per tenant isolates well, but every extra control plane adds servers and operational effort.

Storage is often the hidden blocker. To move a running VM, both the old and the new node must be able to reach its disk at the same time. With local disks or single-attach storage, that means copying the disk during the move or restarting the VM, which is exactly the downtime tenants want to avoid.

How it Works

Three proven building blocks, each doing one job well: vCluster gives tenants their own clusters, KubeVirt runs their VMs, and Lightbits stores their data.

Tenants get their own control plane and nodes, and share one Lightbits pool



Solution architecture · 2 tenants, 1 Lightbits cluster

Each tenant's control plane runs on a shared host cluster, while the tenant's workloads run on nodes dedicated to that tenant. Every node connects to the same Lightbits storage cluster over NVMe/TCP.

Building Block	What it Does for You
vCluster with private nodes	Gives each tenant a complete Kubernetes cluster, with its own API server and dedicated worker nodes, at a fraction of the cost of separate physical clusters
KubeVirt	Runs VMs as native Kubernetes workloads, so tenants manage VMs and containers with the same tools
Lightbits SDS	Provides fast, shared block storage over standard Ethernet, with volumes every node can attach, which is what makes live migration possible

Because each tenant has its own cluster, each tenant can also choose its own versions of KubeVirt and other operators, and upgrade on its own schedule.

Keeping Every Tenant Isolated

Each tenant gets full administrative control of its own cluster, while staying completely separated from every other tenant at four layers.

Layer	How Tenants are Separated	What Each Tenant is Free to Do
Control plane	A dedicated vCluster API server and datastore per tenant	Manage its own namespaces, access rights, custom resources and operators
Compute	Nodes dedicated to one tenant, so tenants never share a server or hypervisor	Decide where its pods and VMs run across its nodes
Network	Separate pod and service address ranges, and a network layer that runs only on the tenant's nodes	Define its own services and network policies
Storage	Per-tenant Lightbits credentials and storage classes	Create, size and manage its own volumes

Your platform team keeps ownership of the host cluster and the Lightbits storage cluster, and tenants never receive credentials for either.



On the storage side, each tenant maps to its own Lightbits project. The tenant's cluster holds a token that is valid only for that project, so a tenant can only see, create and attach its own volumes, even though all tenants share one Lightbits cluster. Optional per-tenant quality-of-service policies cap IOPS and bandwidth, so one busy tenant cannot slow down the others.

Zero-Downtime Maintenance with Live Migration

With Lightbits SDS, a running VM can move to another node while its applications keep running, because the VM's disk never has to move.

Lightbits volumes can be attached to two nodes at the same time. So when a VM moves, only its memory and CPU state travel across the network, while its data stays safely in place on the Lightbits cluster. There is no disk copy, and no reboot for the guest.

A migration happens in four stages:

1. **Prepare.** The platform starts a new VM instance on the target node, and the target node attaches the same Lightbits volume.
2. **Copy.** The VM's memory is copied to the target node while the VM keeps running.
3. **Switch.** The VM pauses for a moment while the last changes transfer, then resumes on the target node.
4. **Clean up.** The old instance is removed and the source node releases the volume.

This changes how you run your platform:

- **Maintenance without maintenance windows.** Draining a node for patching or hardware work moves its VMs automatically instead of shutting them down.
- **Balanced capacity.** Busy VMs can be moved to less loaded nodes while they run.
- **Fewer tenant conversations.** Planned work on the infrastructure no longer needs to be scheduled with every tenant.

To be migration-ready, a VM keeps its disks on Lightbits shared volumes, uses the standard pod network connection, and does not depend on devices tied to a single server. The nodes in a tenant's pool also need compatible CPUs and some spare memory to receive a VM. The reference deployment in the appendix sets all of this up.

Where It Fits

This approach suits any organization that hosts VMs and containers for more than one team or customer.

Who	What they gain
Service providers and MSPs	A multi-tenant Kubernetes and VM hosting offer, with strong isolation and one shared storage pool to operate



Who	What they gain
Enterprise platform teams	A private cloud where each business unit gets its own cluster, without a physical cluster per unit
Teams moving off legacy virtualization	A path to run existing VMs on Kubernetes next to new containerized applications, with live migration kept
Development and test environments	A dedicated, disposable cluster per team or project, backed by the same production-grade storage

Getting started

You can bring this platform up in four phases, and the Lightbits team can support you through each one as part of a proof of concept.

1. **Plan.** Decide how tenants map to node pools, and size the Lightbits cluster for your combined VM and container workloads.
2. **Build the tenant clusters.** Create a vCluster with private nodes and a matching Lightbits project for each tenant, and connect the tenant's nodes to its project.
3. **Enable virtualization.** Install KubeVirt in each tenant cluster and configure Lightbits shared block volumes for VM disks.
4. **Validate.** Boot a test VM, live-migrate it between nodes, and drain a node to confirm VMs move without downtime.

The appendix walks through a reference deployment of all four phases on a three-node environment, including the settings that matter most.

Summary and Next Steps

vCluster, KubeVirt and Lightbits let you offer every tenant a private cluster for VMs and containers, on one shared high-performance storage pool, with maintenance that never interrupts their workloads.

vCluster delivers the isolation of separate clusters without the cost of separate hardware for every control plane. KubeVirt brings VMs onto the same platform as containers. Lightbits shared NVMe/TCP volumes tie it together: data stays in place, VMs move freely, and node maintenance no longer means downtime.

Ready to see it in your environment? Contact your Lightbits representative to plan a proof of concept, starting from the reference deployment in the appendix.

Appendix A: Reference Deployment

This appendix shows how the platform was deployed for one tenant on three nodes: vcluster-1, vcluster-2 and vcluster-3. It is meant as a starting point for your own proof of concept; values such as address ranges and names can be adapted to your environment.

A.1 Tenant Clusters with Private Nodes

Each tenant gets its own vCluster, in its own namespace on the host cluster, with its own set of private nodes. This reference uses two tenants:

Tenant	vCluster Name	Host Namespace	Private Nodes	Pod CIDR	Service CIDR
Tenant A	tenant-a	vcluster-tenant-a	vcluster-1, vcluster-2, vcluster-3	10.64.0.0/16	10.128.0.0/16
Tenant B	tenant-b	vcluster-tenant-b	tenant-b-1, tenant-b-2, tenant-b-3	10.65.0.0/16	10.129.0.0/16

Non-overlapping address ranges per tenant keep routing and troubleshooting simple when tenants share the same physical network.

The configuration for each tenant differs only in its address ranges:

```
Shell
# tenant-a.yaml
privateNodes:
  enabled: true
  vpn:
    enabled: true

networking:
  podCIDR: 10.64.0.0/16
  serviceCIDR: 10.128.0.0/16

# tenant-b.yaml
privateNodes:
  enabled: true
  vpn:
    enabled: true

networking:
```

```
podCIDR: 10.65.0.0/16
serviceCIDR: 10.129.0.0/16
```

The tenant clusters are created on the host cluster:

```
vcluster create tenant-a --namespace vcluster-tenant-a --values tenant-a.yaml
```

```
vcluster create tenant-b --namespace vcluster-tenant-b --values tenant-b.yaml
```

Each tenant's nodes then join their own vCluster, using the join command that vCluster generates for that tenant (vcluster token create, run while connected to that tenant). A node joins exactly one tenant, which is what keeps compute separated.

With private nodes, KubeVirt is installed inside each tenant cluster, so no extra vCluster setting is needed for virtualization.

A.2 Node Preparation

Each node needs hardware virtualization. A quick check:

Shell

```
ls -l /dev/kvm
grep -cE 'vmx|svm' /proc/cpuinfo # greater than 0
```

When the nodes are themselves VMs, nested virtualization is enabled on the underlying hypervisor.

For Lightbits block volumes, containerd is configured to pass device ownership to the pod's user. This is the one setting most likely to be missed:

Shell

```
# /etc/containerd/config.toml, containerd 1.x (config version 2)
[plugins."io.containerd.grpc.v1.cri"]
  device_ownership_from_security_context = true

# /etc/containerd/config.toml, containerd 2.x (config version 3)
[plugins."io.containerd.cri.v1.runtime"]
  device_ownership_from_security_context = true
```

```
sudo systemctl restart containerd
sudo containerd config dump | grep device_ownership # shows = true
```

A.3 Lightbits Projects and Storage Per Tenant

On the Lightbits side, every tenant maps to its own Lightbits project. A project is a tenant-specific sandbox: API calls made with a tenant's credentials can only see and change volumes in that tenant's project. Multi-tenancy must be enabled on the Lightbits cluster, and management traffic then uses the TLS-protected grpcs protocol.

Tenant	Lightbits Project	Credential ID	JWT role	Kubernetes Secret (in the Tenant Cluster)
Tenant A	tenant-a	tenant-a-key	tenant-a:admin	lightbits-jwt in kube-system
Tenant B	tenant-b	tenant-b-key	tenant-b:admin	lightbits-jwt in kube-system

Step 1: Create a Project and Credential per Tenant.

The platform team runs this once per tenant with lbcli, as a Lightbits cluster administrator. The tenant's public key is uploaded as its credential, and the private key stays with the platform team for signing the tenant's JWT:

```
Shell
# Key pair per tenant
openssl genrsa -out tenant-a.key 2048
openssl rsa -in tenant-a.key -pubout -out tenant-a.pub

# Project and credential in Lightbits
lbcli create project --name tenant-a --description "Tenant A"
lbcli create credential --project-name tenant-a --id tenant-a-key \
  --type rsa256pubkey tenant-a.pub

# Repeat for tenant-b
```



Step 2: Issue a JWT Per Tenant.

Each tenant's JWT is signed with its private key (RS256) and carries these claims:

```
Shell
// header
{ "alg": "RS256", "kid": "tenant-a:tenant-a-key", "typ": "JWT" }
// payload
{ "aud": "LightOS", "iss": "platform-team", "sub": "tenant-a-csi",
  "roles": ["tenant-a:admin"], "iat": <now>, "exp": <expiry> }
```

The kid ties the token to the tenant's credential, and the roles claim limits it to the tenant's own project. Choose an expiry that matches your credential-rotation policy.

Step 3: Store the JWT as a Secret in the Tenant Cluster.

The Lightbits CSI driver reads the JWT from a Secret of type `lightbitslabs.com/jwt`:

```
Shell
apiVersion: v1
kind: Secret
metadata:
  name: lightbits-jwt
  namespace: kube-system
  type: lightbitslabs.com/jwt
data:
  jwt: <base64 of tenant-a JWT> # echo -n $JWT | base64 -w0
```

Step 4: Create the StorageClass.

The StorageClass names the tenant's project and points every CSI operation at the tenant's Secret. The Lightbits CSI driver itself is deployed inside each tenant cluster:

```
Shell
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
```

```
name: lightbits-vm
annotations:
  storageclass.kubernetes.io/is-default-class: "true"
provisioner: csi.lightbitslabs.com
allowVolumeExpansion: true
parameters:
  mgmt-endpoint: 10.10.0.1:443,10.10.0.2:443,10.10.0.3:443
  mgmt-scheme: grpcs
  project-name: tenant-a
  replica-count: "3"
  compression: enabled
  qos-policy-name: "" # optional: per-tenant QoS policy
  csi.storage.k8s.io/controller-publish-secret-name: lightbits-jwt
  csi.storage.k8s.io/controller-publish-secret-namespace: kube-system
  csi.storage.k8s.io/node-stage-secret-name: lightbits-jwt
  csi.storage.k8s.io/node-stage-secret-namespace: kube-system
  csi.storage.k8s.io/node-publish-secret-name: lightbits-jwt
  csi.storage.k8s.io/node-publish-secret-namespace: kube-system
  csi.storage.k8s.io/provisioner-secret-name: lightbits-jwt
  csi.storage.k8s.io/provisioner-secret-namespace: kube-system
  csi.storage.k8s.io/controller-expand-secret-name: lightbits-jwt
  csi.storage.k8s.io/controller-expand-secret-namespace: kube-system
```

For Tenant B, the same Secret and StorageClass are created in the tenant-b cluster, with project-name: tenant-b and the tenant-b JWT. Because both tenants use the same StorageClass name, the same VM manifests work in every tenant cluster.

A quick check that isolation works:

```
Shell
kubectl get secret,sc # lightbits-jwt and lightbits-vm exist
lbcli list volumes --project-name tenant-a # tenant-a volumes only
lbcli list volumes --project-name tenant-b # tenant-b volumes only
```

The mgmt-endpoint list names all Lightbits servers for high availability. See the [Lightbits CSI Secret and StorageClass reference](#) for every parameter.

A.4 KubeVirt and CDI

KubeVirt runs the VMs, and the Containerized Data Importer (CDI) loads disk images into Lightbits volumes:

Shell

```
export KV=$(curl -s
https://storage.googleapis.com/kubevirt-prow/release/kubevirt/kubevirt/stable.txt)
kubectl create -f
https://github.com/kubevirt/kubevirt/releases/download/${KV}/kubevirt-operator.yaml
1
kubectl create -f
https://github.com/kubevirt/kubevirt/releases/download/${KV}/kubevirt-cr.yaml
kubectl -n kubevirt wait kv kubevirt --for condition=Available --timeout=10m

# Move VMs during node drains instead of stopping them
kubectl -n kubevirt patch kubevirt kubevirt --type=merge -p \
  '{"spec":{"configuration":{"evictionStrategy":"LiveMigrate"}}}'

export CDI=$(basename $(curl -s -w '%{redirect_url}'
https://github.com/kubevirt/containerized-data-importer/releases/latest))
kubectl create -f
https://github.com/kubevirt/containerized-data-importer/releases/download/${CDI}/c
di-operator.yaml
kubectl create -f
https://github.com/kubevirt/containerized-data-importer/releases/download/${CDI}/c
di-cr.yaml
kubectl wait cdi cdi --for condition=Available --timeout=10m

# VM disks on Lightbits as shared block volumes
kubectl patch storageprofile lightbits-vm --type=merge -p \

'{"spec":{"claimPropertySets":[{"accessModes":["ReadWriteMany"],"volumeMode":"Bloc
k"}]}}'
```

A.5 Test VM and Live Migration

The test VM uses an Ubuntu 24.04 cloud image and a Lightbits shared volume:

Shell

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: test-vm
spec:
  runStrategy: Always
  dataVolumeTemplates:
```

```
- metadata:
  name: test-vm-disk
  spec:
    storage:
      storageClassName: lightbits-vm
      resources:
        requests:
          storage: 10Gi
        source:
          http:
            url:
https://cloud-images.ubuntu.com/noble/current/noble-server-cloudimg-amd64.img
  template:
    spec:
      domain:
        cpu: { cores: 2 }
        memory: { guest: 2Gi }
      devices:
        disks:
          - { name: root, disk: { bus: virtio } }
          - { name: cloudinit, disk: { bus: virtio } }
        interfaces:
          - { name: default, masquerade: {} }
      networks:
        - { name: default, pod: {} }
      volumes:
        - name: root
      dataVolume: { name: test-vm-disk }
        - name: cloudinit
      cloudInitNoCloud:
        userData: |
          #cloud-config
          ssh_authorized_keys:
            - ssh-ed25519 AAAA... admin@example

kubectl apply -f test-vm.yaml
kubectl get dv,vmi -o wide -w # disk import completes, then the VM runs
virtctl migrate test-vm
kubectl get vmim -w # migration completes
kubectl get vmi test-vm -o wide # the VM now runs on another node
```

Before the migration, the test-vm is located on server vcluster-3.

Local Cluster ✓ Connected ID: loft-cluster

Create Pod

Filter by name, id or namespace... All namespaces Columns

☐	Name	Namespace	Status	Ready	Events	Restarts	Node	Created +
☐	virt-launcher-test-vm-2v672	default	Running	2/2	10	0	vcluster-3	3 minutes ago

To migrate the test-vm to another node in the cluster:

Shell

```
virtctl migrate test-vm
```

Output:

```
VM test-vm was scheduled to migrate
```

The VM is live migrated to vcluster-1

Local Cluster ✓ Connected ID: loft-cluster

Create Pod

Filter by name, id or namespace... All namespaces Columns

☐	Name	Namespace	Status	Ready	Events	Restarts	Node	Created +
☐	virt-launcher-test-vm-zlwrr	default	PodInitializing	0/2	2	0	vcluster-1	3 seconds ago
☐	virt-launcher-test-vm-2v672	default	Running	2/2	10	0	vcluster-3	3 minutes ago

And the migration is completed to node vcluster-1

Local Cluster ✓ Connected ID: loft-cluster

Create Pod

Filter by name, id or namespace... All namespaces Columns

☐	Name	Namespace	Status	Ready	Events	Restarts	Node	Created +
☐	virt-launcher-test-vm-zlwrr	default	Running	2/2	10	0	vcluster-1	19 seconds ago
☐	virt-launcher-test-vm-2v672	default	Completed	0/2	11	0	vcluster-3	3 minutes ago

Appendix B: Deployment Tips

These are the situations we came across while building the reference deployment, with the quickest way to resolve each.

What You May See	Likely Reason	Resolution
A disk import stays in progress, with restarts climbing	The import pod stops right after starting	Its log (kubectll logs <importer-pod>) shows the reason, usually one of the cases below
Permission denied on /dev/cdi-block-volume	containerd on that node does not pass block-device ownership to the pod	Apply the containerd setting from A.2 on every node and restart containerd
The import works on some nodes but not others	The setting is missing on the node the pod landed on	The NODE column of kubectll get pods -o wide shows which node to check
A scratch volume stays Pending	No default storage class for temporary import space	Mark the Lightbits storage class as default, as in A.3
Import timeouts or no such host	Pods cannot reach the image download site	Check DNS and internet access from pods, or configure a proxy in CDI
VM creation or the VM console hangs	The tenant control plane cannot reach KubeVirt on the nodes	Check the vCluster VPN connection between control plane and nodes
A VM cannot be migrated	Its disk, network or devices tie it to one node	Use Lightbits shared volumes and the standard pod network connection

For login on Ubuntu cloud images, the default user is ubuntu. SSH keys in cloud-init are recommended, as cloud images disable SSH password login.

Appendix C: Readiness Checklist

A tenant environment is ready for production workloads when these checks pass:

- ☐ Hardware virtualization (/dev/kvm) is available on every node
- ☐ The containerd device-ownership setting is active on every node
- ☐ KubeVirt runs on every node, and KubeVirt and CDI report Available
- ☐ VM disks default to Lightbits shared block volumes
- ☐ A test VM boots and is reachable
- ☐ The test VM live-migrates to another node without a reboot
- ☐ Draining a node moves its VMs to the remaining nodes
- ☐ One tenant's storage credentials cannot see another tenant's volumes

About Lightbits Labs™

Lightbits Labs (Lightbits®) invented the NVMe over TCP storage protocol, which is natively built into its industry-leading block storage, and introduced the first KV cache optimization engine for AI acceleration. Lightbits data infrastructure solutions are engineered to deliver unmatched high performance and maximum hardware efficiency for LLM inference, real-time analytics, and transactional workloads. Lightbits is backed by enterprise technology leaders [Cisco Investments, Dell Technologies Capital, Intel Capital, J.P. Morgan, Lenovo, and Micron] and is on a mission to deliver best-in-class, cost-efficient software solutions for performance-sensitive workloads at scale.

 www.lightbitslabs.com

 info@lightbitslabs.com

US Offices
1830 The Alameda,
San Jose, CA 95126, USA

Israel Office
17 Atir Yeda Street,
Kfar Saba 4464313, Israel

The information in this document and any document referenced herein is provided for informational purposes only, is provided as is and with all faults and cannot be understood as substituting for customized service and information that might be developed by Lightbits Labs Ltd for a particular user based upon that user's particular environment. Reliance upon this document and any document referenced herein is at the user's own risk.

The software is provided "As is", without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and non-infringement. In no event shall the contributors or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the software or the use or other dealings with the software.

Unauthorized copying or distributing of included software files, via any medium is strictly prohibited.

COPYRIGHT© 2026 LIGHTBITS LABS LTD. - ALL RIGHTS RESERVED

LBWP28/2026/10