HyperSDK

KubeVirt VM Deployment

Deploy migrated VMs into Kubernetes using KubeVirt — with HyperSDK's operator, CLI commands, and production-ready deployment patterns for running VMs alongside containers.

KubeVirt CRDs — Operator — Helm Chart — Carbon-Aware Scheduling

What is KubeVirt

Run traditional virtual machines on Kubernetes alongside your containerized workloads.

Kubernetes-Native VMs

KubeVirt extends Kubernetes with custom resource definitions (CRDs) that let you create, manage, and run virtual machines using the same kubectl and API workflows you use for pods and deployments.

CRD-Based Management

VMs are defined as YAML manifests — just like any other Kubernetes resource. Version controlled, GitOps-compatible, and managed through the standard Kubernetes API server.

VirtualMachine Resource

The VirtualMachine (VM) CRD defines a persistent VM with desired state. It manages the lifecycle: create, start, stop, restart, and migrate — while preserving the VM across reboots.

VirtualMachineInstance (VMI)

The VMI represents a running VM instance (similar to a Pod). It holds runtime state: IP address, guest agent info, and live migration status. Created automatically by the VM controller.

VM
Persistent Definition

VMI
Running Instance

VMIRS
ReplicaSet for VMs

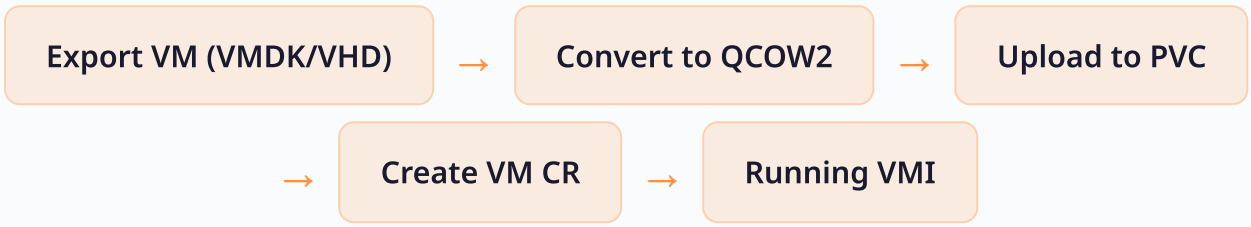
DV
DataVolume (Disk)

KubeVirt brings VMs into the Kubernetes ecosystem — unifying infrastructure management, networking (Multus, OVN), and storage (CSI) for both containers and virtual machines.

HyperSDK to KubeVirt Pipeline

End-to-end migration from any source hypervisor to a running KubeVirt VM.

Migration Pipeline



Step 1: Export

Use HyperSDK to export the VM from vSphere, Hyper-V, AWS, Azure, GCP, or any supported provider. Output is a VMDK, VHD, or QCOW2 disk image with metadata.

Step 2: Convert

Convert the disk to QCOW2 or RAW format using HyperSDK's native Go converters. QCOW2 is recommended for space efficiency; RAW for maximum I/O performance.

Step 3: Upload to PVC

Upload the disk image to a Kubernetes PersistentVolumeClaim using CDI (Containerized Data Importer). HyperSDK creates the DataVolume CR and monitors upload progress.

Step 4: Create VM

Generate and apply the VirtualMachine CR with CPU, memory, disk (PVC reference), network, and cloud-init configuration. The VM controller starts the instance automatically.

Source	Export Format	Target Format	Upload Method

vSphere	VMDK	QCOW2	CDI DataVolume
Hyper-V	VHDX	QCOW2	CDI DataVolume
AWS EC2	VMDK/RAW	QCOW2	CDI DataVolume
Azure	VHD	QCOW2	CDI DataVolume
GCP	VMDK/RAW	QCOW2	CDI DataVolume

Operator Integration

The hypersdk-operator brings declarative VM migration to Kubernetes with 4 CRDs.

⚙

hypersdk-operator

A Kubernetes operator built with controller-runtime that watches for HyperSDK custom resources and reconciles them into migration jobs, VM deployments, and status updates.

4 Custom Resource Definitions

VMExport (source VM export), VMConversion (disk format conversion), VMImport (PVC upload), and VMMigration (end-to-end orchestration combining all three steps).

🎯

Controller Reconciliation

Each CRD has a dedicated controller that watches for create/update events and drives the resource to its desired state. Status conditions report progress, errors, and completion.

Helm Chart

Deploy the operator with a single Helm install. Configurable values for image registry, resource limits, RBAC, and CRD installation. Supports air-gapped environments.

Operator CRDs

CRD	Purpose	Key Fields
VMExport	Export VM from source provider	provider, vmName, outputFormat, credentials

VMConversion	Convert disk format	sourceFormat, targetFormat, sourcePVC
VMImport	Upload disk to PVC via CDI	sourceURL, targetPVC, storageClass
VMMigration	End-to-end migration	source (provider config), target (KubeVirt config)

CLI Commands

hyperctl k8s operations for managing KubeVirt VMs from the command line.

Command	Description
hyperctl k8s -op vm-create	Create a new KubeVirt VM from a disk image or PVC
hyperctl k8s -op vm-list	List all KubeVirt VMs with status, IP, and resource usage
hyperctl k8s -op vm-start	Start a stopped VirtualMachine (creates VMI)
hyperctl k8s -op vm-stop	Gracefully stop a running VM (deletes VMI, preserves VM)
hyperctl k8s -op vm-clone	Clone a VM with new PVC and updated metadata
hyperctl k8s -op vm-snapshot	Create a VolumeSnapshot of the VM's PVC for backup

Example Workflows

Deploy from Export

hyperctl k8s -op vm-create --name web-server --disk /exports/web.qcow2 --cpu 4 --memory 8Gi --network default

Inventory Check

hyperctl k8s -op vm-list --namespace production --output table

Snapshot Before Update

Clone for Testing

```
hyperctl k8s -op vm-snapshot --
name db-server --snapshot-class
csi-snapclass
```

```
hyperctl k8s -op vm-clone --source
prod-app --name staging-app --
namespace staging
```

> *Every hyperctl k8s command works with standard kubeconfig files. Use --kubeconfig or KUBECONFIG environment variable to target any cluster.*

Production Deployment

Best practices for running KubeVirt VMs in production with HyperSDK.

High Availability

Deploy KubeVirt VMs with eviction strategies and live migration enabled. Use pod disruption budgets and node affinity rules to ensure VMs survive node maintenance and failures.

Resource Limits

Set CPU and memory requests/limits on VMs to prevent resource contention. Use Kubernetes resource quotas per namespace to enforce team-level capacity controls.

Monitoring

Integrate with Prometheus via KubeVirt's built-in metrics. Monitor VM CPU, memory, disk I/O, and network throughput. Grafana dashboards for VM fleet visibility.

Carbon-Aware Scheduling

HyperSDK's carbon-aware scheduler can defer VM migrations and batch jobs to low-carbon-intensity time windows. Integrates with Electricity Maps and WattTime APIs.

Production Checklist

1. **Storage Class Selection**

   Use CSI-backed storage with ReadWriteMany support for live migration. Ceph/Rook, Longhorn, or cloud-native CSI drivers recommended.

2. **Network Configuration**

   Configure Multus for secondary networks. Use SR-IOV for performance-sensitive workloads. Bridge networking for layer-2 connectivity.

3

Backup Strategy

Schedule regular VM snapshots with hyperctl k8s -op vm-snapshot. Integrate with Velero for cluster-level backup and disaster recovery.

4

Observability Stack

Deploy Prometheus + Grafana + alerting. Set up alerts for VM crash loops, disk pressure, and migration failures.

VMs + Containers on One Platform

HyperSDK and KubeVirt together let you consolidate legacy VMs and modern containers onto a single Kubernetes platform — with unified management, monitoring, and automation.