OpenShift Virtualization Architecture Guide

Hybrid Platforms Business Unit

version: 1.0.0

May 2025

Legal Notice

Copyright © 2025 Red Hat, Inc.
The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at http://creativecommons.org/licenses/by-sa/3.0/. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Red Hat

Table of Contents

Red Hat

About this Document

Purpose of this document

This document provides opinionated architectural guidelines for deploying Red Hat OpenShift as a platform for virtualization workloads using OpenShift Virtualization. The architecture design guidelines provide a target platform to host workloads that may currently reside on one of the following platforms:

- VMware vSphere Foundation
- Red Hat Virtualization
- OpenStack (any distribution including Red Hat OpenStack Platform)

This document is supplementary to the product documentation and provides opinionated suggestions and guidance for configuring OpenShift for hosting virtual machines (VMs). It provides an overview of the technology in use, design considerations, and prescriptive guidance for implementation. The guidelines in this document do not define a required configuration for support and are not exhaustive. Before implementing, consider the needs and requirements of your virtualized applications, and choose the configuration that best meets your needs.

Scope

This document provides an architectural overview and design guidance for deploying and operating OpenShift Virtualization in an enterprise environment. The content focuses on the infrastructure and platform architecture required to support VM workloads within an OpenShift cluster. Key considerations include the following:

- Node and hardware design for VM hosting.
- Networking configuration, including separation of traffic types.
- Considerations for storage performance and availability.
- Starting points for lifecycle and Day 2 operations for VM infrastructure.
- Dependencies on existing IT systems (e.g., DNS, authentication, etc.).

This guide does not provide step-by-step deployment instructions. Instead, this document is intended to help you make informed architectural decisions and align your platform design with virtualization requirements.

Assumptions

This architecture guide assumes that the reader has the following knowledge:

- Working knowledge of OpenShift on bare metal, including installation methods Installer Provisioned Infrastructure (IPI) / User Provisioned Infrastructure (UPI), node roles, and cluster operations.
- Familiarity with Kernel-based Virtual Machine (KVM) virtualization concepts, including CPU and memory tuning, hardware acceleration, and VM lifecycle management.
- Experience configuring enterprise networking and Virtual Local Area Network (VLAN) segmentation for virtualized environments, including familiarity with NMstate Operator.
- Understanding of persistent storage in OpenShift, particularly Container Storage Interface (CSI) drivers, storage classes, and shared storage concepts needed for live migration.
- Awareness of OpenShift Virtualization capabilities and administrative workflows, including the use of the OpenShift console. CLI experience such as oc and/or virtctl is not expected. A working knowledge of YAML is helpful.

To learn more about OpenShift and OpenShift Virtualization, Red Hat Learning Services offers multiple resources, including a guided learning path, the learning hub, and instructor-led courses, such as DO180, DO280, DO316, DO322, and DO380.

Target audience

This guide is intended for the following audiences:

- Enterprise architects who design hybrid infrastructure platforms that unify container and virtual machine workloads.
- Platform engineers and SRE teams who are responsible for scaling and operating OpenShift clusters with virtualized workloads.
- Infrastructure and virtualization architects who are evaluating OpenShift Virtualization as a strategic platform for consolidating VMs.
- Cloud and datacenter teams who are transitioning from traditional hypervisors (e.g., VMware, KVM/libvirt) to a cloud-native virtualization model.
- Security and compliance stakeholders who are assessing how OpenShift Virtualization aligns with existing controls, audit frameworks, and infrastructure segmentation.

Overview

OpenShift Virtualization, a feature included with OpenShift, uses Kernel-based Virtual Machine (KVM) the trusted Linux kernel hypervisor, delivering enterprise-grade virtualization, for virtual machines, as native objects within OpenShift. Built on the proven foundation of KVM, which is known for robust performance, scalability, and stability. OpenShift Virtualization provides a modern platform tailored specifically for virtualization administrators.

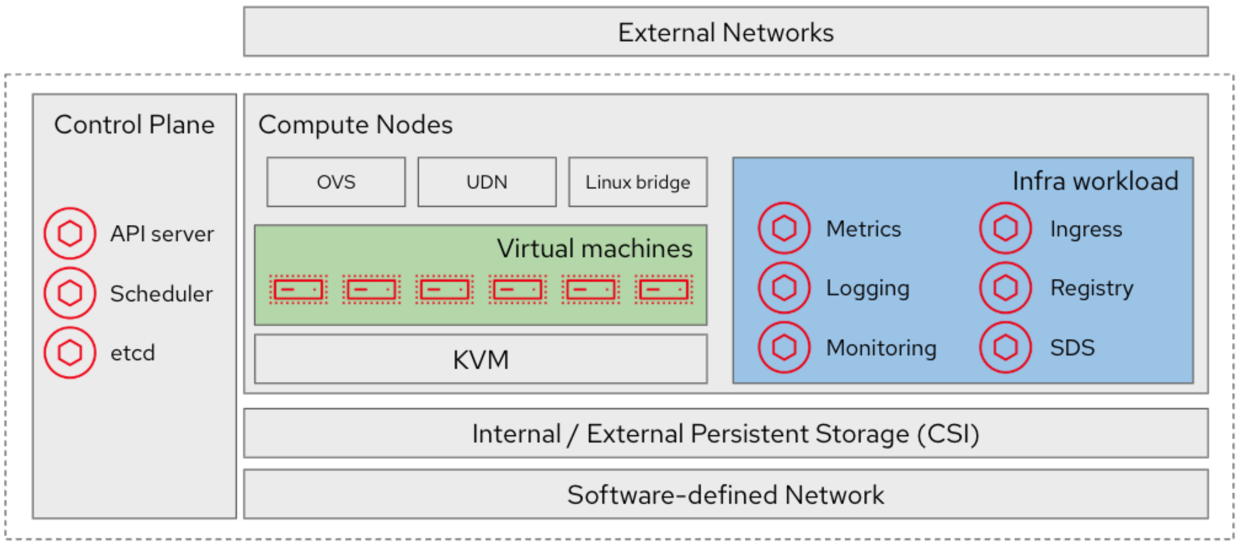
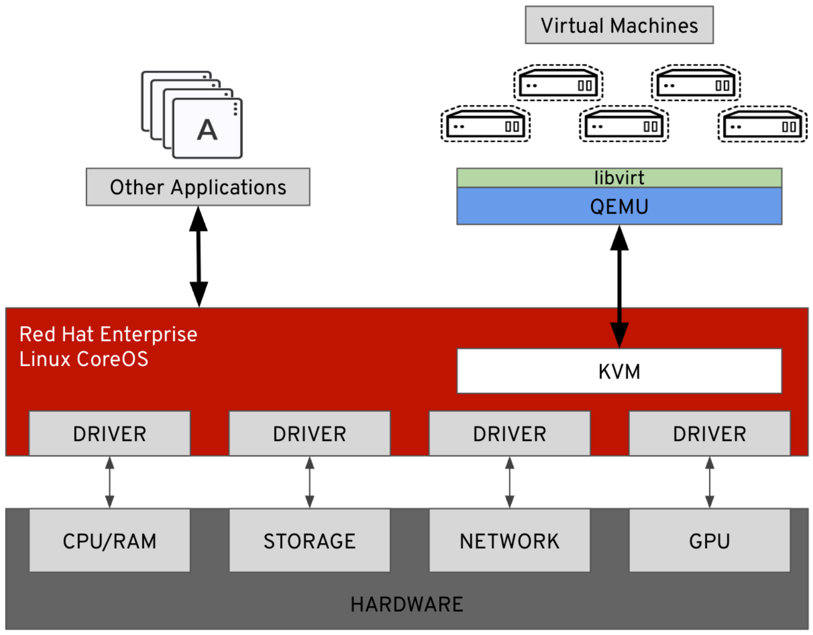


Diagram: OpenShift Virtualization cluster architecture

KVM has been integral to the Red Hat Enterprise Linux kernel for over 15 years, reliably powering production workloads across platforms including Red Hat Virtualization, Red Hat OpenStack Platform, and Red Hat Enterprise Linux. OpenShift Virtualization utilizes this same mature and trusted KVM, along with QEMU and libvirt, to ensure the consistent high-performance and reliability that administrators expect. Consequently, OpenShift functions effectively as a robust, stable, and scalable type-1 hypervisor, that is ideal for enterprise virtualization environments.



Virtual machines within OpenShift Virtualization benefit from sophisticated scheduling and management capabilities, including host affinity, resource awareness, load balancing, and built-in high availability. These features simplify administration and optimize resource utilization, ensuring that virtual machines run efficiently and reliably at scale.

Networking capabilities include connection to the integrated cluster software-defined network (SDN), enabling secure and controlled access to virtual machines through configurable network policies. This facilitates secure and reliable connectivity for virtual machines, with flexible ingress and egress options that align with enterprise security requirements. Additionally, more traditional connectivity options are available using Open vSwitch (OVS) for VMs connecting to VLANs and other logical networks, SR-IOV, and OpenShift's user-defined networking (UDN) as an evolving alternative.

OpenShift Virtualization supports diverse deployment scenarios. Scenarios include self-managed deployments using bare metal hosts on-premises, and cloud deployments on Amazon Web Services (AWS), either through self-managed clusters or managed Red Hat OpenShift Service on AWS (ROSA). Support on other hypervisor clouds is in development as of this writing. This flexibility enables organizations to adopt a virtualization platform that supports their specific infrastructure strategy and operational needs.

OpenShift Editions

Red Hat OpenShift Virtualization is a feature of Red Hat OpenShift. It is included with all bare metal editions of OpenShift, including OpenShift Virtualization Engine, OpenShift Kubernetes Engine (OKE), OpenShift Container Platform (OCP), and OpenShift Platform Plus.

- **Red Hat OpenShift Kubernetes Engine**: A hybrid cloud, enterprise Kubernetes runtime engine that provides core Red Hat OpenShift functionality to deploy and run applications, which you install and manage in a datacenter, public cloud, or edge environment.
- **Red Hat OpenShift Container Platform**: A full-featured hybrid cloud, enterprise Kubernetes application platform used to build, deploy, and run applications, which you install and manage in a datacenter, public cloud, and edge environment.
- **Red Hat OpenShift Platform Plus**: A hybrid cloud platform that allows enterprises to build, deploy, run, and manage intelligent applications at scale across multiple clusters and cloud environments. Multiple layers of security, manageability, and automation provide consistency throughout the software supply chain. OpenShift Platform Plus subscriptions are available for x86-based clusters only.
- **Red Hat OpenShift Virtualization Engine**: A bare metal-only, virtualization infrastructure offering based on Red Hat OpenShift and the open source kernel-based virtual machines (KVM) hypervisor, purposely-built to provide enterprises with a reliable, enterprise-grade solution for deploying, managing, and scaling VMs. A subset of OpenShift functionality, this edition of OpenShift is targeted at virtual machine-only workloads, with infrastructure services supported in containers only (i.e., no end-user application containers support).

The implementation guidance and information provided within this document applies to all editions of OpenShift, however some add-on features, implemented as OpenShift Operators, may not be included with all editions.

To learn more about the OpenShift editions, what is included with each, and what workloads are permissible, refer to the subscription guide.

Hyperscaler-hosted OpenShift Virtualization

Red Hat OpenShift, with its integrated OpenShift Virtualization capabilities, offers a unified platform for running both containerized and traditional virtual machine workloads. Bare metal OpenShift, required for OpenShift Virtualization, is available in a variety of deployment models to meet diverse infrastructure needs, including fully managed offerings on major hyperscalers such as AWS and self-managed using AWS, IBM Cloud, and Oracle Cloud Infrastructure. This flexibility allows organizations to standardize their application platform across environments, whether in the data center or in the cloud, while using familiar KVM-based virtualization technology.

This architecture guide focuses specifically on self-managed, on-premises deployments of OpenShift Virtualization. These environments provide full administrative control over infrastructure, making them ideal for organizations with stringent compliance, data locality, or performance requirements. By deploying OpenShift Virtualization on-premises, enterprises can modernize their virtualization stack while maintaining tight integration with existing networks, storage systems, and operational processes. The content that follows is focused on helping architects and administrators plan, deploy, and operate OpenShift Virtualization in these self-managed environments effectively.

Scalable multi-cluster virtualization

This document is focused on the architectural considerations, decisions, and options for an individual cluster, however most deployments will consist of multiple clusters with workloads distributed across them. As OpenShift Virtualization environments grow in scale and complexity, effective management and automation become essential. Red Hat provides two key platforms to support these needs: Advanced Cluster Management (ACM) for centralized cluster governance, and Ansible Automation Platform (AAP) for automating VM and guest OS operations. Together, they enable a layered management approach—ACM oversees the OpenShift infrastructure, while AAP orchestrates actions within and across workloads.

Red Hat Advanced Cluster Management

Red Hat Advanced Cluster Management (ACM) is a centralized management solution that provides unified visibility, governance, and lifecycle control across multiple OpenShift clusters, including those used for OpenShift Virtualization. It enables administrators to view and manage clusters across data centers, edge locations, and cloud environments through a single console. With ACM, virtualization architects can automate cluster creation and scaling, standardize configurations through policy enforcement, and monitor health, performance, and compliance of both the cluster and workloads across all managed clusters. This is especially valuable in environments where virtual machine workloads are distributed across multiple OpenShift Virtualization clusters and operational consistency is critical.

For OpenShift Virtualization, ACM helps streamline both platform and workload management at scale. Administrators can use ACM to apply consistent security and network policies, monitor VM health and usage metrics through integrated observability tools, and manage the placement of workloads across clusters based on resource availability or compliance requirements. ACM's integration with OpenShift Virtualization includes tailored dashboards for monitoring resource utilization of individual VMs, logical groups of VMs and cluster nodes, and more, such as VM rightsizing reports. In short, ACM enhances the operational efficiency, governance, and scalability of virtualized infrastructure built on OpenShift.

Red Hat Ansible Automation Platform

Red Hat Ansible Automation Platform (AAP) is a powerful enterprise automation framework that enables IT teams to standardize, orchestrate, and scale infrastructure and application workflows across hybrid environments. For OpenShift Virtualization architects, Ansible provides a flexible and agentless way to automate tasks such as virtual machine provisioning, guest OS configuration, patching, and post-deployment customization. Its ability to interface with OpenShift Virtualization APIs and resources allows administrators to define declarative playbooks that manage VMs alongside traditional infrastructure, ensuring consistent and repeatable operations across environments.

Ansible Automation Platform also supports infrastructure-as-code practices through its integration with Git, enabling version-controlled automation workflows and collaborative operations. With features like event driven automation, role-based access, job scheduling, and detailed logging, virtualization teams can extend automation beyond VM lifecycle management to include network configuration, storage provisioning, and application deployment within guest operating systems. This improves operational efficiency, reduces manual intervention, and helps enforce organizational policies for workload consistency.

When combined with Red Hat Advanced Cluster Management (ACM), Ansible Automation Platform complements ACM's cluster-focused capabilities by addressing the automation needs inside the cluster. While ACM centrally manages OpenShift cluster lifecycles, governance, and policy enforcement across fleets, AAP handles workload-specific automation, including VM provisioning, guest OS configuration, and day-2 operations within virtual machines. Together, they offer a full-stack automation solution—ACM for orchestrating and securing the OpenShift infrastructure, and AAP for customizing and managing the workloads running within it. This integration is especially useful in large-scale or multi-cluster OpenShift Virtualization environments where consistency, compliance, and operational velocity are key.

Architectural considerations and guidance

OpenShift Virtualization introduces new architectural requirements and operational challenges beyond a standard OpenShift deployment. Virtual machines are stateful, require large amounts of resources when compared to typical container-based applications, and depend on the platform to provide important functionality and capabilities. OpenShift Virtualization clusters need to be architected to provide resilient, reliable, and performant compute, network, and storage resources to meet the needs of the applications.

Additionally, administrators and users need access to tools and resources to deploy and manage the VMs, identify and remediate performance problems, secure the platform and the VMs, manage the infrastructure, and automate as much as possible to facilitate multi-cluster operations. The core features of OpenShift, including the use of Operators to extend capabilities, enable efficient and effective management and use of the cluster for virtual machines.

This section outlines architectural considerations for deploying and operating OpenShift Virtualization on bare metal infrastructure, including design guidance and recommendations where applicable.

Cluster architecture

Node role planning strategy

In OpenShift, nodes are assigned roles that define the workloads that they are expected to run. On bare metal infrastructure, where hardware must be purposefully allocated, planning node roles is an important architectural decision that affects scalability, resiliency, and performance isolation.

OpenShift clusters have the following primary node role types:

- Control plane nodes host the core components that manage the OpenShift cluster. These include the API server, scheduler, controller manager, and `etcd`. These services are critical to cluster availability and consistency. Control plane nodes are typically isolated from other workloads to ensure consistent performance and fault tolerance.

- Infrastructure nodes are an optional node type that host platform-level workloads supporting cluster functionality. They are not part of the core control plane. Infrastructure nodes include ingress controllers, the logging and monitoring functions, Operator controllers, and more.
- Compute nodes, also commonly called workers, are where the workload is hosted. These are general-purpose nodes where the VMs hosting your applications are run.

For more information about what is, and is not allowed to run on each node type, and the subscription requirements for each node role type, refer to the subscription guide.

OpenShift allows multiple roles to be assigned to the same node, enabling more flexible hardware utilization. Common role assignments include the following role combinations:

- Control plane and infrastructure nodes. Co-hosting infrastructure components with the control plane can reduce the total number of physical nodes while still maintaining functional separation from user application workloads.
- Control plane and compute nodes, also known as a schedulable control plane. This configuration places workloads, such as infra components and VMs, on the same physical nodes as the control plane components.
- Infrastructure and compute nodes. Without creating dedicated infrastructure nodes and assigning eligible applications to them, this is the default configuration for infrastructure workload.

When different workloads run on control plane nodes, there is a risk that the control plane nodes will encounter resource contention issues. You can help mitigate this risk by configuring cluster and node critical pods, such as the control plane services, to have a high priority and/or priority class. In the event that contention occurs, this will result in other pods being removed from the node before the control plane pods are targeted for termination. It is important to use monitoring and alerting to quickly identify and remediate any resource contention issues for control plane nodes.

In general, it’s common to isolate the control plane, infrastructure, and compute nodes to improve operational resilience, security boundaries, and workload predictability. However, bare metal servers often have significantly more resources that may be stranded and unused with single purpose nodes. Consider the impact to the cluster and the virtualized applications if the control plane node is affected by resource contention, then decide if the inefficient use of hardware is worth the trade off of potentially experiencing resource contention.

Topology awareness and failure domains

When deploying OpenShift Virtualization, it is recommended to define failure domains that align with the physical infrastructure of the environment. This improves workload availability and resiliency by enabling the OpenShift scheduler to make topology-aware placement decisions for virtual machines and containerized applications.

By applying topology labels to nodes, workloads can be spread across distinct physical boundaries. Examples of topology labels include `topology.kubernetes.io/zone` or custom labels such as rack, chassis, or room. These labels can then be referenced in affinity and anti-affinity rules, ensuring that workloads are distributed across failure domains.

Examples of infrastructure failure domains include:

- Multi-rack deployments. Using three or more racks to distribute nodes with combined roles assigned, particularly the control plane node, mitigates against top-of-rack switch failure, PDU failure, and other issues that can affect an entire rack of equipment.
- Network architecture. Leaf-Spine (CLOS), top-of-rack switch pairing, and other network implementations can affect the failure domain(s) to consider for workloads. This includes logical network separation, such as VLANs. For example, a VLAN used for accessing a handful of VMs being impacted by a network failure will have a dramatically different impact than the VLAN used for accessing IP-based storage used by all VM disks that suddenly become non-functional.
- Environmental zones. Clusters spread across multiple “server rooms” or “closets” may have different factors affecting power and cooling capacity and reliability. These may constitute secondary considerations for failure domains, particularly at edge and remote sites with potentially unreliable services.

Control plane high availability

The control plane is critical to the cluster, without it VMs cannot be created or managed, workload cannot be scheduled, and cluster and node configuration cannot be applied. It is vital that the control plane remains available at all times.

While OpenShift supports a single-node control plane option, this is not recommended for most deployments. Even at edge and other constrained sites, a minimum of three nodes is recommended, whenever possible, to provide resiliency for the control plane. If a single node cluster is unavoidable, then it’s recommended to use external tools for monitoring to detect failure and use automation to recover and restart virtual machines.

The most common OpenShift deployment type uses three control plane nodes. This architecture allows for one node to be unavailable, because of updates, maintenance, or failure, at any point in time and is suitable for most deployments. It also balances performance requirements for transaction latency by limiting the number of cluster members who need to acknowledge any writes.

However, OpenShift Virtualization requires bare metal, which often has different service level agreements (SLAs) for recovery than a virtual cluster. Where a virtual cluster can replace a lost or missing node in minutes or hours, a bare metal cluster might take days or weeks for the hardware to be fixed or replaced. During this time, a three node cluster would not have any additional redundancy for failure, or even control plane nodes being unavailable updates.

When an extended time to recover from a failed control plane node is a risk factor, then it is recommended to consider a 5-node control plane, which is supported in OpenShift 4.17 and later. A 5-node control plane provides the ability to have up to two control plane members unavailable at any time, making node failure, updates, configuration changes, and other expected - and unexpected - actions easier to accommodate. However, this configuration does come at the cost of additional resources needed for control plane services.

Importantly, a 5-node control plane does not meaningfully change the resiliency or survivability of a cluster stretched across two sites when compared to a 3 or 4 node control plane. In all instances, one site will have the majority, or quorum, of nodes and should that site be lost, then the cluster will be in a read-only state until quorum is regained.

Using a hosted control plane with a bare metal cluster

Hosted Control Planes (HCP) is an OpenShift architecture that separates the Kubernetes control plane from the worker nodes of the cluster. In a traditional OpenShift deployment, the control plane, comprising the API server, scheduler, controller manager, and etcd, runs directly on a set of dedicated nodes within the same cluster as the workloads. Control plane nodes may also host virtual machines if desired, which is common with bare metal clusters. With HCP, however, the control plane is deployed to a separate management cluster, allowing the workload cluster to consist solely of compute nodes, and infrastructure nodes if they are being used. This decoupling reduces the control plane footprint on the virtualization cluster, streamlines cluster deployment, and simplifies lifecycle operations such as upgrades and recovery.

While HCP is commonly associated with managed or cloud-hosted OpenShift environments, it is also a valid and fully supported architecture for self-managed, on-premises deployments, including bare metal. This model is useful when managing multiple OpenShift clusters under a centralized administrative team, as it consolidates control plane hosting and improves operational consistency. For organizations deploying OpenShift Virtualization across several clusters, HCP can improve resource efficiency by freeing up physical nodes for workloads and reducing the operational burden of managing multiple independent control planes.

However, this architecture guide primarily focuses on traditional cluster topologies where the control plane resides within the same cluster as the compute nodes. This model remains widely applicable for environments that prioritize local control, minimize inter-cluster dependencies, or operate in disconnected or edge data centers where centralized control plane hosting may not be practical.

Logical separation of compute resources for scheduling purposes

An OpenShift cluster is a single entity that encompasses all of the cluster nodes and virtual machines within. This is different from how other virtualization solutions create logical "clusters" of compute resources, which are the boundary for some features and capabilities, such as live migration, and also used to apply configuration common to all the members.

Despite this difference in how OpenShift works, creating a logical grouping of compute resources for scheduling virtual machines is still a valid and appropriate approach for many use cases. Some example reasons to create logical groups of servers are:

- Different hardware generations with different CPU models and other performance characteristics.
- Distinct hardware capabilities, such as nodes that have GPUs, SR-IOV NICs, etc.
- Dedicating resources to specific consumers.
- Software and operating system vendor licensing requirements.

Within OpenShift there are already logical groupings of server resources through the use of machine sets and machine config pools. Both of these refer to how the cluster manages the configuration of those nodes. It's reasonable, and expected, to have different machine config pools based on some of the same reasons as listed above, but these are not used when scheduling workload.

These groups are implemented using the following OpenShift features:

- Labels are key-value pairs assigned to nodes that provide a flexible way of expressing attributes such as hardware characteristics (e.g. server generation), location (e.g. room or rack), or workload roles (e.g. database VMs only or reserved for a specific team).

Through a node selector or affinity rule, the VM declares that it wants to run on nodes matching the label. As a result, labels attract the workload to the node.

- Taints are applied to nodes to restrict which workloads can be scheduled on them, effectively marking the node as reserved or specialized. VMs must have a toleration defined to be scheduled to a tainted node. As a result, taints repel workloads from the node.

Importantly, the node labels and taints can be changed at any time, meaning that a node can move from one logical group to another with no configuration changes other than the label. Additionally, nodes can have more than one label applied, which means that they can also belong to more than one logical resource group. This provides significant flexibility to how resources are managed and assigned to users and groups.

After labels and, optionally, taints are defined for the nodes, assigning workloads to them is done using node selector(s) and affinity rules, collectively known as scheduling hints, configured on the virtual machines. If the logical group of compute resources has a taint present, the virtual machine(s) also need a toleration. The selectors, affinity rules, and tolerations can be configured on virtual machines at any time using the UI, CLI, and/or other automation tools. There are two additional methods that automatically apply the configuration when the VM is created:

- Using project level configuration, where a node selector is automatically applied for all virtual machines in the project. Because these are largely transparent to the user and cannot be removed on an individual VM basis, using them judiciously is suggested.
- Using VM templates or instance types, where the definition used to create the VM already includes scheduling hints

Cluster capabilities

OpenShift supports a modular installation model, sometimes referred to as "Composable OpenShift", in which certain core platform components, known as cluster capabilities, can be selectively disabled during installation. This allows for leaner cluster footprints, particularly in constrained or specialized environments. The set of configurable capabilities changes with each minor release of OpenShift 4. It's important to review the documentation to determine what is available with your version and to review each capability to identify if it’s needed.

It is recommended to enable, at a minimum, the baremetal, MachineAPI, console, CSISnapshot, and NodeTuning capabilities for all OpenShift Virtualization clusters. If there is no specific reason to exclude a capability, it is generally recommended to leave it enabled to preserve full platform functionality and avoid unintended limitations.

Software-defined networks

A software defined network (SDN) is required for all OpenShift clusters, even when there is no intention of connecting virtual machines to the SDN or using its other features. The SDN is, by default, the network used for live migration traffic, node and VM health checks, internal service communication, and much more.

As of OpenShift 4.12, Open Virtual Networking - Kubernetes (OVN-Kubernetes) is the default SDN, however multiple other SDN options from partners, such as Tigera, Cisco, and Juniper, are supported with OpenShift and virtual machines. See https://red.ht/workswithvirt for a full list of available partner SDN options.

This section focuses on OVN-Kubernetes. For details about other implementation options, refer to the documentation for that vendor.

Beyond being necessary, some OVN-Kubernetes configuration choices at install time also have an impact on cluster scalability. Specifically, the cluster network Classless Inter-Domain Routing (CIDR) and host prefix affect the maximum number of nodes in the cluster and the maximum number of Pods per node.

The cluster network CIDR is expressed as a subnet with a prefix length. By default these values are 10.128.0.0/14 and a host prefix of /23, which means that each node in the cluster receives a /23 from the /14 IP range. This has two important ramifications:

- The maximum number of Pods that a single node can host is limited by the number of IPs available in the host prefix. A prefix of /23 means the maximum is 512. This value cannot be changed after cluster deployment.
- The maximum number of nodes in the cluster is determined by the number of host prefixes which can be allocated from the cluster network CIDR range. With the defaults, a /14 can have 512 /23 networks created, which means that the cluster would be limited to 512 nodes. This value cannot be changed after cluster deployment.

If you anticipate needing more than 500 Pods per node or more than 500 nodes in the cluster, you must configure a larger cluster network CIDR and/or a smaller host prefix during installation. For example:

- A host prefix of /22 allows 1024 Pods per node, but only 256 nodes in the cluster with the default /14 cluster network.
- A /13 cluster network supports 1024 nodes with the default /23 host network.

Before increasing these values, compare the supported maximums for both cluster size and VMs-per-node against your needs and determine whether you need more than one cluster. Incorrect sizing during installation may require a cluster redeployment to resolve. Initial planning is essential for long-term scalability.

Additional considerations for the cluster network IP range include the following:

- The same IP range can be reused across multiple clusters, unless you intend to use Submariner, or a similar capability, to provide cross-cluster network connectivity.
- The range should not overlap with other IP space external to the cluster, which SDN-connected VMs or Pods may need to connect to.

Disconnected and partially disconnected clusters

Disconnected clusters are those with no internet connectivity for retrieving images and updates from Red Hat registries and catalogs. This deployment model is common in highly secure or sensitive environments and depends on mirroring OpenShift content from an internet-connected system, which is then transferred into the disconnected infrastructure. All OpenShift subscriptions include a limited entitlement to the mirror registry, a containerized version of Quay, specifically for supporting these offline mirroring workflows.

Partially disconnected clusters use a local mirror for OpenShift content but maintain selective internet access. This model offers several benefits: only core images and frequently used Operators need to be mirrored thus reducing storage requirements compared to fully disconnected mirrors, and the cluster can still leverage Red Hat Insights. Insights provides proactive analytics and real-time recommendations to identify and remediate potential issues across the OpenShift infrastructure. To maximize these benefits, it is strongly recommended to keep telemetry enabled when possible.

It is generally recommended to use a local mirror of the core OpenShift images and content for the purpose of installing and updating clusters. This reduces reliance on remote Red Hat-hosted registries, minimizes WAN bandwidth usage, and significantly accelerates operations like image pulls, which is particularly important when there are multiple clusters deployed.

Stretched cluster

A stretched cluster is an OpenShift cluster in which control plane nodes are distributed across two or more physical locations. These locations may range from separate server rooms within the same building to entirely different data centers or geographic sites. With some constraints around latency and connectivity requirements, this architecture is fully supported for all workloads with OpenShift.

It is important to distinguish this model from remote worker node architectures. In a remote worker deployment, only the compute nodes (workers) reside in distant locations, often with limited or unreliable network links to the control plane. In contrast, a stretched cluster distributes the control plane members across locations. Both models are supported by OpenShift and OpenShift Virtualization, but they address different architectural needs and failure domains.

While this is supported, caution should be taken when considering this architecture. Stretched clusters often add significant complexity to the architecture of a cluster. The control plane, particularly etcd, is highly sensitive to latency, jitter, and even minor packet loss. Spanning the control plane across locations introduces dependencies on inter-site network stability and performance.

Carefully consider the reasons for choosing a stretched cluster, factoring in tradeoffs in complexity and configuration brittleness against potential availability improvements. For example, when using a stretched cluster both the network and storage must be available and identical across all sites. Furthermore, the control plane relies on a majority of nodes to be available to maintain quorum. With a two site deployment, one site will host the majority of quorum nodes, resulting in it being a single point of failure that will affect control plane availability. This does not change with four and five node control plane architectures.

Evaluate whether an alternative solution, such as using multiple clusters - one at each site - with storage replication is a better solution for your RTO and RPO requirements.

External network services

Bare metal OpenShift clusters rely on a set of critical external infrastructure services to function correctly throughout their lifecycle. Unlike cloud-based deployments, where many of these services are abstracted or managed by the infrastructure platform, bare metal clusters require deliberate planning, integration, and monitoring of these dependencies to ensure resilience and stability.

DNS

Domain Name System (DNS) is fundamental to OpenShift and critical to keep the cluster operational. External DNS is used for resolving control plane endpoints, such as the API, as well as routes for ingress traffic. Internal cluster components also rely heavily on service discovery via DNS, however this is hosted by the cluster and not an external dependency.

Architecturally, DNS services must be redundant, configured for both forward and reverse resolution, and be reachable by all nodes at all times.

DHCP

Depending on your install method and how you choose to add nodes to the cluster, Dynamic Host Configuration Protocol (DHCP) may be essential for node provisioning. DHCP is used by Installer-Provisioned Infrastructure (IPI) workflows and the PXE-based Red Hat Enterprise Linux CoreOS (RHCOS) install process. The DHCP server must be able to serve boot files (via TFTP or HTTP) and assign consistent IP addresses, especially if static IP reservations are not in use.

To maintain predictability and avoid IP conflicts, it is recommended to use DHCP reservations. Once nodes are provisioned, static addressing is suggested, either configured via NMstate or by setting DHCP lease time to infinite, to be used. In particular, it is very important that control plane node IP addresses on the machine network do not change unpredictably as this can lead to the loss of etcd quorum and control plane services being unavailable.

NTP

Accurate time synchronization is important to OpenShift for many reasons. Components such as etcd, the API server, and certificate services require synchronized clocks to function correctly. Time drift can lead to certificate validation errors, inconsistent logs, or failures in scheduling and coordination.

NTP should be configured with multiple, redundant sources. Cluster nodes should be configured, using MachineConfig, to ensure a consistent time source across the environment..

Load Balancer(s)

OpenShift supports both internal and external load balancer configurations, choosing the right approach at install time depends on expected operational requirements, in particular if the applications will make heavy use of cluster ingress services.

It's common to start with the VIP-based internal load balancer, particularly when other load balancer services are unavailable or not considered adequately reliable. This feature uses static virtual IPs managed by keepalived. It’s a lightweight solution that removes the need for external load balancer infrastructure, making it especially useful for getting started quickly and scenarios where high ingress throughput is not a requirement.

If hosted applications are expecting to make heavy use of the OpenShift Route/ingress capabilities, using one or more external load balancers is a better option. Whether implemented via hardware appliances like F5 or through open-source tools like HAProxy or NGINX, external load balancers provide proper distribution across multiple endpoints with active health checking and failover logic.

It is straightforward, and supported, to migrate to external load balancers post-deployment as requirements change. OpenShift allows the reconfiguration of API and ingress endpoints with minimal disruption to either traffic type. This phased approach works well in environments where early deployment speed is important but scaling and resilience will become priorities over time.

Authentication services

OpenShift’s internal authentication methods are limited to htpasswd, which is useful for small or non-production clusters, but does not easily scale nor provide mechanisms for password complexity, rotation, and other common requirements. Integrating with external integration services, such as Microsoft Active Directory, Red Hat Identity Management, and other OAuth/OIDC-based platforms lets administrators use common enterprise authentication services for their OpenShift cluster(s).

If/when authentication services are unreachable, logging into the cluster is not possible. It is important, and strongly recommended, to use certificate based authentication as a backup authentication mechanism for cluster administrators. This allows them to connect and manage the cluster even if other methods are not available.

Container registry

OpenShift relies on external container registries to provide the content required for cluster installation, updates, and the deployment of features, such as OpenShift Virtualization. While most container images are pulled from Red Hat–managed registries like quay.io and registry.redhat.io, relying solely on internet-based registries can introduce challenges, particularly in environments with limited WAN bandwidth or when managing updates across multiple clusters simultaneously. The full list of Red Hat container registries used for OpenShift cluster installs, updates, and feature deployment is in the documentation.

To address this, it is often beneficial to deploy an on-premises container registry. Hosting a local mirror of OpenShift content improves performance and reduces external network dependency. This is especially valuable in disconnected or bandwidth-constrained environments.

All OpenShift subscriptions include the mirror registry, a containerized and supported instance of Red Hat Quay, specifically intended for mirroring OpenShift content. However, OpenShift also supports integration with other container registries, allowing customers to use existing infrastructure when available.

Nodes

For specific guidance about hardware requirements refer to the product documentation. OpenShift inherits hardware support from the version of Red Hat Enterprise Linux (RHEL), which the RHCOS used by the cluster version is built from. You can find RHEL versions used by RHCOS in this KCS article.

Beyond the minimums described by the product documentation, there are no specific requirements for OpenShift Virtualization hardware. Hardware requirements are driven by the performance, capacity, density, and resiliency desired.

Sizing

Sizing depends on your environment and needs. You must plan appropriate sizing requirements for your environment. Compute, i.e. CPU and memory, capacity plays a significant factor in the overall capacity of each node in the cluster. Oversubscription of resources allows the administrator to choose to increase real utilization of the node at the risk of resource contention if too many virtual machines need the resources they've been assigned. Different workloads have different CPU and memory requirements at different ratios, if your applications skew toward using more than one or the other, consider configuring hardware in an equivalent ratio, such as 8GiB memory for every one CPU core, to avoid stranding resources.

Refer to the sizing guide for additional considerations regarding the CPU and memory configuration for hypervisor nodes.

Cluster homogeneity

Nodes in an OpenShift cluster do not need to be homogenous, meaning they can be of different make, model, capacity, and capabilities. However, each variance in hardware adds management and configuration overhead while also potentially impacting the ability to schedule and live migrate virtual machines.

Whenever possible, keep hardware identical across as many cluster nodes as is feasible. Understanding that hardware is often lifecycled in batches, not all at once, it's important to group homogenous nodes together for multiple reasons:

- Applying configuration, such as machine config and NMstate, relies on node labels. Nodes with the same physical configuration will need to have the same logical configuration applied.
- VM resources, particularly CPU, can be affected by different hardware generations. Setting a CPU model for a VM creates a lowest common denominator set of capabilities allowing it to have the broadest compatibility across the available nodes, however this often comes at the price of forfeiting performance and efficiency features of the newer hardware generation. Using node selectors to assign VMs to logical groupings of nodes allows the VMs to maximize the hardware feature set available to them.
- Different node configurations, particularly different generations, may have significant differences in performance that may lead to unexpected effects on the applications.

Importantly, these logical groupings of nodes can be implemented within the cluster - you do not need to have separate OpenShift Virtualization clusters for each hardware type. Assign labels, and taints if needed, to the nodes then use selectors, tolerations, and affinity rules to assign workload to them. You have the option to implement these customizations for your needs.

Reserved resources for node functions

In an OpenShift cluster, it's recommended that each node reserve a portion of CPU and memory resources for core system components, such as the kubelet, the container runtime, and other node-level services, to maintain sufficient resources to ensure node stability. However, the exact amount of resources required can vary significantly based on workload density, node size, and the specific components running on the node, making manual sizing potentially complex and error-prone.

To simplify this process, and ensure consistent node behavior across varying hardware profiles, we recommend to enable automatic system resource reservation by setting `autoSizingReserved: true` in the node's `MachineConfig`. When enabled, this setting allows the system to dynamically calculate and reserve appropriate CPU and memory resources for kubelet based on the total capacity of the node.

Network adapters and configuration

OpenShift typically uses one network interface, the one with an IP address on the machine network's CIDR, for all traffic, including node-to-node SDN traffic, pod-to-world traffic, IP-based storage mounted by CSI drivers, and communicating with the control plane. Virtualization often has significant requirements, both for latency and throughput, that require special consideration and identifying if additional configuration is needed.

A single network interface is not recommended, even if it is very high throughput, e.g. 40Gbps or 100Gbps. A single interface does not provide resiliency in the event of a cable, NIC, switch, or other failure.

Host network connections configured using a mode 4 (LACP) bond, balance-slb, or a mode 1 bond and composed of multiple network adapters (NICs) spanning redundant upstream network resources, is highly suggested. This allows network connectivity, and all of the functions depending on it, to continue to function in the event of NIC or upstream device failure.

Additional bonds of network adapters can optionally be used to further isolate different network traffic types. These include:

- Node-to-control plane and node-to-node SDN traffic. The SDN is used for communication between cluster functions and is also the default for live migration traffic. This functionality is required, and is critically important, so it's vital to avoid throughput contention and high latency.
- IP-based storage. Low latency network connectivity is important for the user and application experience with virtual machines. Throughput can vary based on multiple factors, but it's important to account for traffic that will be generated when guest OS services like antivirus and updates happen simultaneously across many VMs.
- Live migration, both compute and storage. Live migration can consume a significant amount of bandwidth, which is multiplied for each additional simultaneous migration happening. It is suggested to have dedicated NICs for this traffic. If that's not possible, consider limiting the number of simultaneous migrations per node. Limiting the throughput of live migration is not suggested as that may lead to issues where VMs with high memory churn will not be able to successfully migrate.

- Virtual machine import and migration network. Used by the Migration Toolkit for Virtualization, this optional network would be used to isolate the high throughput traffic when importing virtual machines from other hypervisors
- Backup. When performing backups, whether at the hypervisor level or from within the guest OS, substantial read (from the VM disk) and write (to the backup system) traffic will be generated. This does not require low latency network connectivity, however limiting the bandwidth available may extend the amount of time it takes to complete backups.
- Software-defined storage (SDS) replication. Many SDS offerings deployed to OpenShift utilized replication of data across multiple nodes to provide protection against failure. Low latency, high throughput network connectivity is extremely important to provide the performance expected. Consult with your storage vendor for detailed requirements and suggestions.
- VM and application traffic. This is the reason why the VMs exist - to host applications and provide that functionality across the network. The requirements are determined by the aggregate needs of all applications and guest operating system functions.

While each of these options can be isolated to separate NICs, this would require a substantial amount of network adapters. It's common to have SDN and control plane traffic on one pair of NICs, IP storage and/or backup and replication on another, live migration on a third, with VM and application traffic on a fourth set.

Multiple bonding modes are supported with OpenShift Virtualization, however mode 4 (LACP) is the typical recommendation in the absence of other requirements or constraints. LACP provides both improved resiliency for adapter link issues and increased total throughput for workload. Note that depending on the configuration of the LACP bond, point-to-point throughput of a single flow will be limited to the maximum throughput of a single member of the bond.

Managing host network configuration with NMstate

NMState is a declarative network configuration tool used in OpenShift to manage host-level networking. It abstracts the complexity of interface, VLAN, bond, bridge, and routing configuration into a consistent format, which is applied through the NetworkManager service. Depending on your install method, the NMstate Operator may not be installed and deployed by default, however it is highly recommended to use this Operator. In OpenShift, NMState's NodeNetworkConfigurationPolicy (nncp) resources define network settings across selected nodes in a controlled and predictable manner.

When using NMState, it is recommended to define all dependent network resources, such as the physical interfaces and bridges needed for VM connectivity or the interface(s) and bond used for IP-based storage connectivity, within the same configuration file. This ensures that changes are applied atomically, reducing the risk of misordered updates or transient states that could lead to configuration failing to apply. Keeping interdependent resources in a single configuration helps maintain idempotency and minimizes downtime during network reconfiguration, which is particularly important in clusters with strict availability requirements.

Local storage

Node storage is used for several important functions beyond installing RHCOS. This includes runtime storage for OS and platform features (images, log data, etc.), ephemeral volumes, etcd storage for control plane nodes, kubelet data, and more. This makes both the performance and capacity of the local storage important to the overall functionality. Note: this section does not refer to any secondary disks used by software-defined storage, the LVM Operator, or other features. This strictly refers to the disk(s) where RHCOS is installed and the `/var/lib/containers` mount point is.

Using flash based storage, for example NVMe or SSD, is recommended for all nodes and required for control plane nodes. Optionally, use software or hardware RAID to provide resiliency in the event of disk failure.

Local storage capacity available to RHCOS should follow documentation guidelines. Extra capacity is better than just-enough capacity. RHCOS takes action to shutdown workload, prune images, and remove orphaned volumes to protect the node disk(s) from reaching full capacity. The minimum recommendation is at least 250GiB of capacity. Using hardware or software RAID for the RHCOS disk is optional and can be used to increase resiliency of the server if wanted, however, be aware of any performance implications of the RAID configuration.

Red Hat Enterprise Linux CoreOS (RHCOS)

OpenShift Virtualization requires RHCOS compute nodes. Even though it is possible to deploy and use Red Hat Enterprise Linux (RHEL) compute nodes with OpenShift, these nodes are incompatible with OpenShift Virtualization. RHCOS is based on RHEL and designed to run containers with an integrated CRI-O runtime and dedicated container tools. The operating system uses rpm-ostree to facilitate being managed through the Machine Config Operator to deploy operating system updates and apply configuration declaratively.

To customize RHCOS, for example, to add drivers or other third-party OS-level tools, the RHCOS image layering feature must be used to modify the operating system and add the appropriate packages.

For more information on Red Hat Enterprise Linux CoreOS, see the documentation.

Maximum Pods per node

OpenShift's default maximum number of Pods per node is 250. The maximum number of Pods includes containerized application workloads, platform services, infrastructure agents, and virtual machines. These maximums apply regardless of which edition of OpenShift is used. Platform-level components, such as the kubelet, monitoring agents, CSI drivers, and logging or backup services, typically consume around 40–60 pods per node, depending on cluster configuration.

Changing the maximum number of Pods per node depends on the SDN configuration when the cluster is deployed. When using OVN-Kubernetes, the default for OpenShift since OpenShift v4.12, the clusterNetwork.hostPrefix value in the install-config.yaml indicates the size of the subnet assigned to each node. By default, this is a /23, which represents 512 IP addresses and is the maximum value that can be configured for the maximum Pods. To go beyond 512 Pods per node you must configure a larger host prefix for the cluster network. A value of /22 allows1024 IP addresses.

While increasing the pod limit is a supported configuration change, extremely high pod or VM density on a single node introduces operational complexity and resource contention risks. With OpenShift Virtualization, each virtual machine is scheduled as a single pod, but typically consumes more memory and CPU than a normal container. As a result, practical density for virtual machines is often limited by physical node capacity before Pod count becomes a constraint.

Changing the default from 250 to 500 is a straightforward process and generally recommended. However, it is important to carefully evaluate your requirements if you intend to host more than 350-400 virtual machines on a single hypervisor node. This is even more important when using OpenShift Kubernetes Engine (OKE) or OpenShift Container Platform (OCP) to co-host virtual machine and container-based applications on the same bare metal servers. Containerized applications tend to consume more Pods than virtual machines, where each VM is counted as a single Pod.

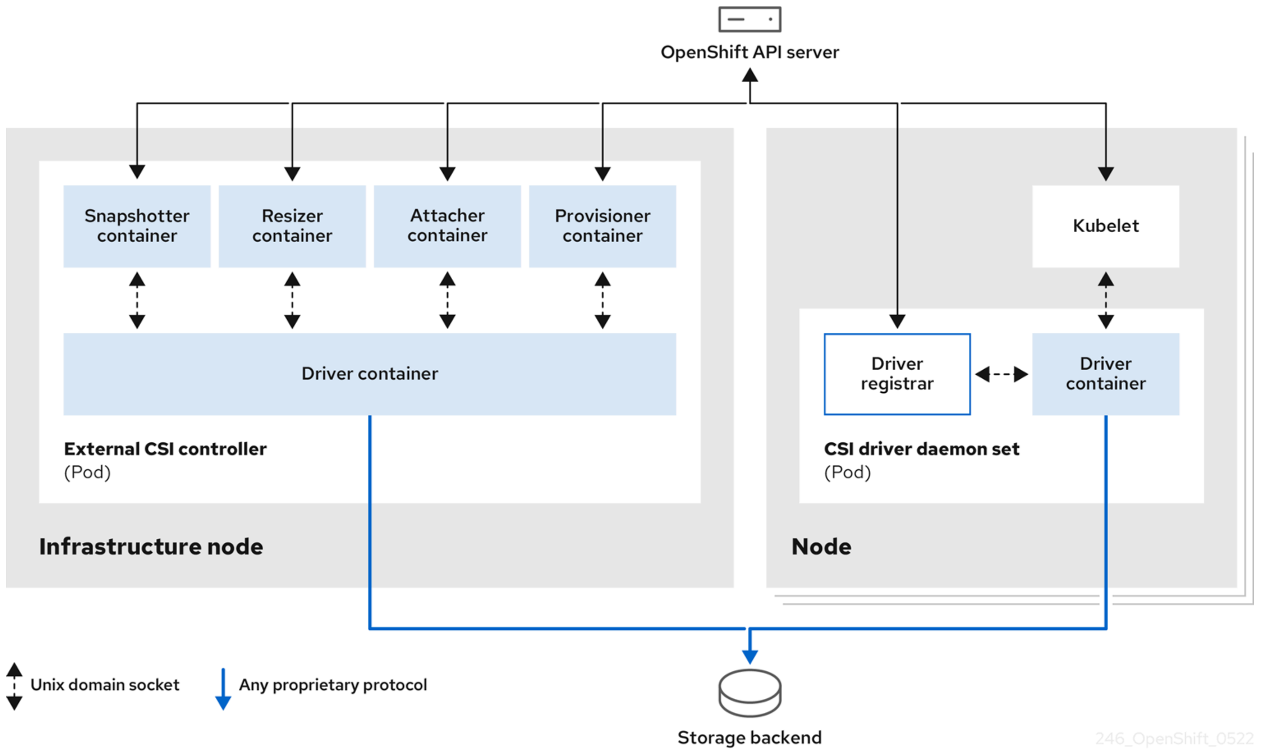
Shared storage for virtual machines

OpenShift Virtualization stores virtual machine disks using persistent volume claims (PVCs), where each VM disk has its own dedicated PVC. The PVCs are provisioned, on demand, by a CSI driver, which is a component provided by the storage vendor. The CSI driver simplifies how the containerized workload works with the storage vendor. The CSI driver abstracts the process of provisioning a volume on the backing storage device and mounting it to the node running the VM.

This is fundamentally different from the paradigm used by other virtualization solutions, where a single storage device, e.g. an iSCSI/FC LUN or NFS export, is used to store many virtual machine disks. In OpenShift Virtualization, by treating each virtual disk as an independent, managed resource, we enable fine-grained control, feature application, and lifecycle management tailored to the workload. This provides storage vendors the flexibility to expose differentiated capabilities (e.g., QoS, snapshots, cloning) per virtual disk.

Live migration of VMs requires that all VM disks for the VM being migrated use PVCs with the read-write-many (RWX) access mode. This is so that the disk can be mounted to both the source and destination hypervisor node during the migration. While both block (iSCSI, Fibre Channel, etc.) and file (NFS) protocols support RWX access, you must verify with your storage vendor any additional requirements or limitations specific to the storage device being used to host the virtual machine.

The following diagram shows a high-level overview of the CSI components in an OpenShift cluster. The CSI driver coordinates the creation and management of the volume on the storage device, mounting of the volume to the node hosting the virtual machine, and implementing features like snapshots and volume resizing at the storage volume level.



Multiple CSI drivers can be deployed to an OpenShift cluster to enable support of many different storage backends. This allows VMs to use various disk types, storage protocols, and access modes from different vendors to best suit the workload. This provides customers with maximum flexibility in selecting vendors to suit their specific needs.

The CSI driver abstraction relies on the storage provider to offer and implement advanced features that are beneficial to VMs. Examples include:

- Offloaded cloning provides the storage backend to create a copy of a PVC directly within the storage system, rather than performing a full read/write copy through the compute node. This is valuable when provisioning virtual machines from golden images or templates, because it significantly reduces clone time and avoids unnecessary input/output (I/O) overhead on the cluster.
- Volume snapshots provide the ability to capture a point-in-time state of the virtual machine disk. This enables rollbacks during application testing or upgrades, OS patching, or other major changes. The snapshot functionality must be implemented by the CSI driver and is a prerequisite for snapshot-based backup solutions.

- Storage capacity quotas, where the CSI driver can expose volume capacity tracking and enforce quotas. This helps ensure that VM PVCs do not exceed available backend capacity.
- Quality of Service (QoS) and other performance guarantees or limits. Some CSI drivers provide QoS parameters, such as Input/Output Operations Per Second (IOPS) limits, throughput guarantees, or latency thresholds, at the PVC level. These controls are especially useful for tiered storage environments or for isolating performance-sensitive VMs from noisy neighbors sharing the same backend.
- Multipath I/O for VM volumes uses redundant data paths between the node and the storage system. This enhances availability and performance, especially for block storage configurations used in high-throughput or mission-critical virtualization scenarios. The CSI driver must create, configure, and mount the backing volume for multipath.
- Other storage features, such as replication, deduplication, and compression, are transparent to OpenShift and the VMs. Some storage vendors choose to expose these features and make them configurable on a per PVC basis, via the storage class, and other storage vendors do not.

The presence or absence of these capabilities is entirely dependent on the storage backend and CSI driver implementation. Not all storage vendors support the same features, and some features may only be available under specific protocols (e.g., NFS vs iSCSI) or volume types (e.g., block vs file). The impact to the capacity and performance of the virtual machine(s) from each of these features is dependent on the storage provider as well. As a result, it is important to validate which of these features are supported by your storage platform, as support is not consistent across all vendors.

Network connectivity for virtual machines

Virtual machines always require connectivity to external systems, other applications, and/or end users. As a result, network connectivity is fundamental to the value of any virtualization platform. With OpenShift Virtualization, virtual machines have many different options for network connectivity, including:

- The cluster software-defined network (SDN), OVN-Kubernetes by default
- The service mesh, if one is deployed
- Direct external connectivity through Linux bridge and/or Open vSwitch (OVS)
- Direct external connections using SR-IOV and DPDK adapters

None of these connection methods is mandatory. For example it’s not required for all VMs to use the SDN, instead VMs can be connected to a mixture of one or more of the network types at any time. The result is a flexible networking model that enables a wide range of use cases and provides the architect many opportunities to optimize the networking configuration based on the needs of the applications.

Dedicated host NICs vs shared for VM traffic

When an OpenShift cluster is initially deployed, only the network interfaces associated with the node's machine IP address will be configured. In many cases, configuration involves a bonded interface composed of multiple Network Interface Cards (NICs), which OpenShift uses when creating the OVS used by OVN-Kubernetes. This SDN provides critical communication, including node-to-node and pod-to-pod, and can handle other cluster traffic such as virtual machine live migration and, when wanted, external access to VMs.

While it is technically possible to route VM traffic through the same OVS used by the SDN, this is not recommended. The SDN is essential to the health and functionality of the node, and reusing its resources for VM traffic introduces risk. Any misconfiguration or modification to the SDN OVS can disrupt inter-node communication or control plane availability and may complicate cluster upgrades.

Additionally, use caution when creating supplementary OVS or Linux bridge deployments that share physical interfaces or bonds with those used by the SDN. Although this configuration is supported, we discourage it for the same reasons: shared use of SDN-connected NICs increases the likelihood of performance contention and configuration errors that could impact the stability of the node’s core networking components.

Cluster SDN, Services, and Routes

OpenShift Virtualization uses the cluster software-defined network (SDN) to provide out-of-the-box connectivity for virtual machines, providing an easy way to communicate with other VMs and containerized applications within the cluster, and for outbound access to external networks. This requires no additional configuration from the users, or administrator, beyond optionally creating a Service to simplify discovery of the virtual machine for other SDN-connected VMs and containers. Additionally, Routes are able to be used to access VM-hosted applications using http(s) traffic on ports 80 and/or 443. However, for other traffic types and ports a Service with type LoadBalancer (typically backed by a solution like MetalLB), a host port, or a node port needs to be used to access the virtual machine.

Linux bridge vs Open vSwitch for external connectivity

For VM network interface adapters (NICs) which need to connect to external layer 2 (VLAN) networks, there are several options available in OpenShift Virtualization. The Linux bridge, used for decades, provides KVM-based virtual machines access to networks and continues to be a viable option within OpenShift Virtualization. The Linux bridge has limitations when compared to other options. Notably, it does not support platform-level security controls via network policy. However, Linux bridge often has the highest raw performance with the lowest overhead.

Open vSwitch (OVS) was also used for many years, particularly in the OpenStack community, to provide more feature rich connectivity for virtual machines. This includes the ability to use network policy to implement platform level rules for controlling connectivity. Network policy rules allow the user, or the administrator, to allow or deny the VM and/or other internal cluster or external resources to connect to the VMe. These rules can be applied to the virtual machine as a whole, the IP address, one or more subnets, and/or specific ports for connectivity to/from/between the VM.

Importantly, both options support native and tagged VLANs. It is generally recommended to use OVS as the default for external virtual machine networks.

Host-level VLAN adapters vs VLAN tagging using virtual switches

For some time it was common for administrators to create host-level VLAN interfaces on top of the NICs (or bonds) and then deploy Linux bridge (or OVS) configurations to those host-level interfaces. The following diagram shows this configuration. This continues to be possible, and fully supported, however, we do not recommend this configuration because of the increased configuration complexity and operational overhead that it introduces. Using this method generates a substantial amount of additional NMstate configuration that needs to be created, applied, and maintained. This complexity introduces risk of human error, particularly at scale.

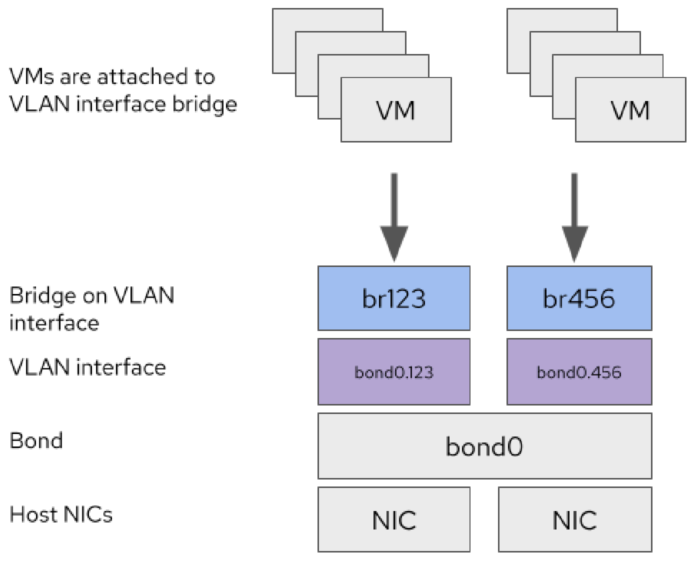


Diagram: Multiple bridges on VLAN interfaces.

In contrast, allowing the VLAN tags for VM traffic to be applied by a single OVS (or Linux bridge) on the NIC (or bond) dramatically simplifies the NMstate configuration needed. This results in only a bridge map per VLAN being needed on the OVS rather than a VLAN interface + bridge being needed per VLAN.

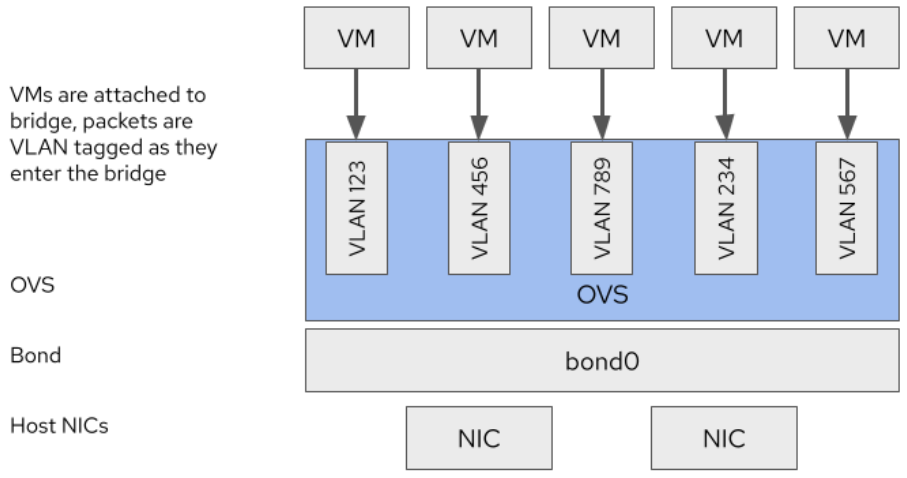


Diagram: One bridge applying VLAN tags.

Multus network attachment definitions

NMstate is used to configure the host network adapters and the OVS (or Linux bridge) devices to be used for VM networks. Multus provides the configuration needed by the container network interface (CNI) plugins to connect the virtual machine to that underlying network configuration. You can think of each network attachment definition as being a port group that is available either globally, when created in the default namespace, or only within the specific namespaces it's added to when not in the default namespace.

Importantly, VM network attachment definitions do not support the full set of CNI options available to other OpenShift-based workloads. This includes whereabouts, IpVlan, and MacVlan. Refer to the product documentation for specific support with each version.

Network policy

Network policy provides a declarative way to control traffic flow to and from virtual machines connected to the cluster SDN and/or external networks. It allows the users, and administrators, to enforce fine-grained, namespace-scoped network segmentation, sometimes referred to as micro-segmentation, for the workload. With regard to VMs, there are 2 types of policies which we need to be aware of:

1. `NetworkPolicy` rules are available to all workloads, VMs or Pods, which are connected to the SDN. These can be applied to traffic which is ingressing or egressing from the virtual machine, including traffic that is both internal and external to the cluster.
2. `MultiNetworkPolicy` is the set of rules which apply to non-SDN connected virtual machine NICs, such as those connected to a VLAN on an OVS bridge. Importantly, Linux bridge does not support the use of network policy rules.

Overall, the use of network policy is fully supported and provides a robust way to provide control of network traffic at the platform level, however be aware that more than one policy rule may be needed for VMs with multiple network connections. Optionally, Red Hat's Advanced Cluster Security provides a graphical management interface that displays traffic flows between virtual machines connected to the SDN and a simplified user experience for creating and managing network policy rules.

User-defined Networks (UDN)

User-defined networks (UDNs) are a new feature of OpenShift 4.18 that enable and empower both users and administrators to create network segments on demand, including private internal-only and public facing networks for virtual machines (and other cluster workload).

However, at the time of this writing the integration with virtual machines is nascent and, as a result, the suggestion continues to be to use OVS bridges for external VM connectivity unless you have a known issue or feature requirement that is only available with UDN.

UDN is expected to be the future of virtual machine connectivity, this guide will be updated as integration between UDN and virtual machines matures.

Virtual machine high availability

High availability (HA), in this section, refers to the ability to detect node failure and restart virtual machines. The topology awareness section discusses data center-level considerations for cluster resiliency and avoiding external failures, such as power loss or network switch failure, resulting in the entire cluster being unavailable.

Restarting VMs when a node fails happens in 3 phases - failure detection, fencing, and remediation - each with specific requirements and configuration within the cluster. Importantly, this configuration is not done automatically, the administrator must configure, at a minimum, the fencing mechanism to be used.

Failure detection

The process begins with the control plane detecting that a node is no longer responding. This typically occurs when the node misses a series of heartbeats or fails health checks reported by the kubelet. The node is marked as NotReady or Unknown, triggering availability workflows. Importantly, this state does not immediately trigger remediation—it simply identifies the node as potentially unhealthy.

OpenShift has several options for detecting node failure:

- Control plane status updates, a.k.a. Heartbeats. These happen when the node's kubelet performs a status update.
- Node health checks. These identify when the node is experiencing various conditions which affect its ability to host workload, such as being in a not ready state, memory pressure, and/or disk pressure.
- Machine health checks. This check type relies on Machine API to monitor the health of the server, which means that the nodes must be provisioned and managed using the bare metal Operator and a machine set.

Fencing

Before workloads can be safely rescheduled, OpenShift must ensure the failed node is not still running the virtual machines. If they are still running when the cluster attempts to restart them it can lead to corruption when two instances of the VM attempt to use the same disk.

Depending on the hardware, and how the nodes are managed by the cluster, there are several options for fencing:

- Self Node Remediation. This option is recommended for nodes which do not have a hardware management interface, e.g. BMC or iLO, or it is not accessible by the node(s) hosting the Operator.
- Fence Agents Remediation. This method uses the management interface, e.g. BMC or iLO, to power off the server. This is the recommended method when available.
- Machine Deletion Remediation. When using Machine API to deploy and manage cluster nodes, this option works by removing the node and provisioning a new one. This requires that the cluster use the bare metal cloud provider and be provisioned using IPI.

Remediation

After the failed node was fenced, guaranteeing no VMs are running, the cluster can safely reschedule and restart the workload. Remediation happens automatically following fencing by deleting the virtual machine instance (VMI) resources associated with VMs that were running on the failed node. This does not remove the VM definition nor any of its resources, it only indicates that it is no longer running and, therefore, needs to be restarted.

For additional details, see the Workload Availability for OpenShift documentation.

Security and compliance

OpenShift is designed with a layered security model that integrates with enterprise IT standards, providing robust mechanisms for access control, workload isolation, and system integrity. From its hardened container runtime and secure default configurations to its native support for encryption, policy enforcement, and auditability, OpenShift offers a comprehensive foundation for building secure, compliant infrastructure.

The OpenShift security documentation includes comprehensive guidance on how to use and configure security features of OpenShift. Furthermore, compliance guides for industry standards, such as the DISA STIG for OpenShift, provide specific guidance for meeting requirements.

Role-based Access Control

OpenShift's role-based access control (RBAC) provides a robust, and granular, method of applying, or removing, permissions for what users and service accounts are able to do within the platform. This is essential for virtualization environments, where access boundaries between teams, tenants, or workloads must be enforced across all types of actions, from basic VM manageability to core cluster configuration.

Importantly, integration with an enterprise authentication provider is recommended. This provides a scalable mechanism to control individual authentication, and also create groups of users against which cluster and project roles can be applied.

RBAC policies are designed to delegate responsibilities cleanly—for example, distinguishing between infrastructure administrators, platform operators, and application teams that create and use virtual machines and manage their own workloads.

The default configuration for an OpenShift cluster is for all authenticated users to receive the system:authenticated role, which has self-provisioner permissions so that they may create new projects and deploy workload. To control and manage general OpenShift permissions, use the standard cluster roles such as `cluster-admin` or `cluster-reader` to give users the ability to manage all aspects of the cluster or simply have view-only access to resources which don't explicitly belong to them.

OpenShift Virtualization also has cluster roles which control a user's access to virtualization resources and configuration. These roles are:

- `kubevirt.io:admin` cluster role grants users full permissions to all OpenShift Virtualization resources, including access to view and modify the runtime config of the CRD instance.
- `kubevirt.io:edit` cluster role gives users permissions to modify all virtual machine resources in the cluster. A user with this role can create, modify, and delete virtual machines across all projects in the cluster.
- `kubevirt.io:view` cluster role gives permissions to view all virtualization resources in the cluster. Users with this role do not have the ability to create, modify, or delete VMs. A view-only user would be able to see if a VM is running, but not be able to shutdown/power on nor access the console for the VM.

In general, we suggest that roles be used to delegate permissions and responsibilities to platform administrators and users. The specific implementation should be determined by your security policies and operational requirements, an example of this is to have 3 types of user. These users include the following:

- Cluster administrators, who have full access to all aspects of the cluster and its configuration.
- Virtualization administrators, who have access to manage virtualization resources such as boot images, instance types, templates, create/configure network attachment definitions, and other permissions as needed.
- Virtual machine users, who have the ability to create and manage VMs in their own project(s), but do not have access to other resources. Optionally, these users may need permission to create custom instance types, templates, and/or boot images within their project(s).

The OpenShift documentation contains additional information about RBAC, including how to extend the default roles with custom permissions and aggregate multiple roles for fine-grained control.

Compliance operator

The compliance operator automates the scanning, assessment, and remediation of cluster configurations against compliance frameworks, including CIS Benchmarks, NIST 800-53, PCI-DSS, HIPAA, and more, including custom profiles to fully tailor behavior to your requirements. It is built on top of the OpenSCAP framework and leverages Security Content Automation Protocol (SCAP) to evaluate the compliance state of the platform and its components.

The profiles do include OpenShift Virtualization components, however they may not be complete or adequate based on your specific requirements, particularly for all aspects of node configuration. It is important to review the compliance profiles to ensure that all aspects of your policies are addressed and, if not, creating custom profiles to augment the defaults.

Additionally, it's important to know that the compliance operator only addresses cluster configuration. It cannot scan and address issues within virtual machine guest operating systems. Tools such as Ansible Automation Platform and other compliance frameworks may be used from within the guest OS for these purposes.

Day two operations

After an OpenShift cluster is deployed, the focus shifts to Day 2 operations, i.e. the ongoing tasks required to maintain, update, and scale the platform and its workloads over time. These activities often involve complex and manual processes. These processes are greatly simplified in OpenShift through the use of Operators. Operators encapsulate operational knowledge into reusable automation, enabling platform components and features to self-manage, often with minimal intervention. From lifecycle management and remediating configuration drift, to observability and scaling, Operators provide a consistent and declarative approach that abstracts away much of the underlying complexity.

Node maintenance operator

The node maintenance operator simplifies the process of moving workload away from a node and marking it as unavailable for scheduling. Without the node maintenance operator this is a two step process of cordoning, where the node is marked unschedulable, and draining, where the workload is moved to other nodes in the cluster. The operator is not installed by default, it must be deployed post-install by a cluster administrator.

It is recommended to deploy and use this operator with OpenShift Virtualization to simplify node management.

Node disruption policies

Most changes made by the machine configuration operator to the configuration of a node will result in it being cordoned, drained, and rebooted. Philosophically, this is as a result of the appliance-like nature of RHCOS nodes, however it may not be desirable for the node to reboot for minor changes to the configuration, instead a service restart or reload may be all that is needed.

OpenShift introduced the concept of node disruption policies to empower administrators to apply policy for what actions are taken when configuration is changed. For example, this means that the administrator can configure the nodes to not reboot when a customized registries.conf file configuration is applied to the nodes and, instead, only a reload of the service happens.

If you have configuration which changes frequently, more frequently than you expect to apply updates to the cluster, then using node disruption policies is a viable option for managing extraneous reboots. However, it is recommended in most circumstances that the default configuration is left in place. If there is a need to control when reboots happen, the recommendation would be to pause the machine pools until all configuration changes have been applied so that only a single reboot is needed. Additionally, to further reduce node reboots, pausing the machine config pool and batch applying configuration updates in conjunction with a cluster update is a valid option.

Monitoring, metrics, and alerting

OpenShift Observability is the umbrella term used for the integrated monitoring, metrics, logging, and alerting framework within a cluster. Within OpenShift, this suite of tools uses Prometheus, Alertmanager, and Thanos to provide a default set of metrics and monitoring dashboards in the OpenShift console. Additionally, administrators may create custom dashboards by using the Graphana operator as desired. The data collected from the node exporters, among other places, is used to determine resource utilization, node and virtual machine health, and aid with troubleshooting efforts. It is vital to success that the observability tools are deployed and used within a cluster.

General recommendations for an OpenShift Virtualization deployment include:

- Review the Alertmanager configuration to, at a minimum, set notification rules and destinations.
- Consider using Advanced Cluster Manager (ACM) to extend the period of time which metrics are aggregated for trend analysis.
- Strongly consider deploying the optional network observability operator to gain additional insight and data around the node and virtual machine network resource usage.

The default resolution for metrics is based on the scrape interval for Prometheus, which is 15 seconds. This means that for assessing and troubleshooting performance metrics in real time you will need to utilize other methods, specifically connecting to the node via a debug pod or the virtual machine Pod's terminal to use more traditional tools like `top`. See the performance section for more details.

User workload monitoring

User workload monitoring is an optional component of the OpenShift metrics suite which allows users to define their own metrics endpoints to be scraped by Prometheus. This is useful for multiple reasons, including having a platform-level function for viewing overall metrics and health statistics for applications, virtual or container-based, deployed to the cluster. For pod-based applications, the metrics collected can also be used in conjunction with features such as the custom metrics autoscaler (KEDA) to scale an application up or down based on metrics beyond simple CPU and memory utilization.

Configuring user workload monitoring is not required for OpenShift Virtualization, however it may be useful for the application teams deploying and using virtual machines.

Log aggregation and analysis

The Cluster Logging Operator is deployed after OpenShift installation and provides centralized log collection and forwarding across the cluster, giving visibility into VM-related events. If you have an existing logging solution, configuring forwarding to that tool is sufficient. However, if you do not, it is strongly recommended to deploy OpenShift's log collection tool, Loki, and the appropriate plugin for the observability interface.

Cluster backup and recovery

The control plane, and specifically etcd, contains all of the configuration and desired state information for all aspects of an OpenShift cluster. This makes it the single authoritative source of truth for the configuration and also defines how the cluster Operators will attempt to configure the cluster. As a result, this is the most important and critical aspect of the cluster to backup and protect.

Both compute nodes, and individual control plane nodes, are largely disposable and can be removed and replaced with relative ease since their configuration is stored in, and applied by, the control plane. When a node fails, removing it from the cluster and replacing it with a new one is the fastest way to recover any workload that was hosted on the node. This is covered in more detail in the high availability section.

However, it is critical to understand that an etcd backup is not enough to recover a cluster which has lost the entire control plane. Plainly, in this event it is faster and more efficient to deploy a new cluster and redeploy the workload to the new cluster. It is not possible to restore an etcd backup from one cluster to a different cluster. Therefore, it is strongly recommended that regular backups of the configuration, to include machine config objects, NMstate configuration, storage classes and PVC definitions (but not data, which can and should be separately protected), virtual machine definitions, network attachment definitions, RBAC roles, Secrets, and all other relevant configurations, be created and stored in a protected location.

OADP provides a method of doing this that is included with OpenShift, where the administrator chooses the objects to protect, the destination S3-compatible storage, and the frequency to protect the configuration.

Alternatives to this include using external tools, such as Red Hat Advanced Cluster Management (ACM) or OpenShift GitOps, to manage and apply configuration for the cluster(s). This ensures that the configuration is stored in at least one external location and can be reapplied to a new cluster in order to facilitate rapid recovery from a full cluster loss.

Virtual machine backup and recovery

OpenShift's native functionality for performing virtual machine backup and restore is OpenShift API for Data Protection (OADP). OADP is a storage agnostic tool for performing backups of all OpenShift resources, including Secrets, ConfigMaps, PVCs, virtual machine definitions, and more. It is built upon Velero and uses an S3-compatible endpoint for storing the backup data. This tool is a useful starting point for doing basic virtual machine protection and, in the absence of an existing preferred backup solution, can adequately meet many recovery time and recovery point objectives (RTO, RPO) for smaller deployments.

However, it is generally recommended to use a partner backup suite for more robust capabilities that can integrate across the lifecycle of the VM, the guest OS, and any application(s) installed to the VM. For a list of backup software compatible with OpenShift Virtualization, visit https://red.ht/workswithvirt.

Updates

OpenShift's over-the–air update mechanism is, to the maximum extent possible, a hands-off experience for the administrator and users. The cluster operators manage updates to both themselves and the deployed instances of their operands (CRD instances) whenever possible.

As with any update or upgrade process, it's important to review the release notes thoroughly to identify any changes, deprecations, and new capabilities which affect your cluster(s) and the workload. Periodically APIs are removed from the OpenShift API server which can affect features and functionality, however these are well documented, including how to identify which APIs are being used and by which features or users. Be sure to follow the official upgrade guidance for supported z-stream (e.g. 4.17.19 to 4.18.6) update paths.

Compute node updates

The node operating system, RHCOS, is updated as a part of a cluster update. This is managed using the machine config operator and done transparently to both the administrator and the user, with only a reboot being needed at the end of the process. It is important to remember the time it takes for reboots to happen and factor this into the maintenance windows for your cluster(s). Importantly, this is also mitigated by the cluster automatically performing a cordon and drain on the node before a reboot occurs, which means the virtual machines will be non-disruptively live migrated to other nodes in the cluster.

The machine config operator applies updates to all machine config pools at the same time, which means that if you have more than one pool then multiple nodes may be unavailable for workload at the same time. By default, each machine config pool will apply updates to only a single node at a time. Updating nodes can be delayed, or prevented, indefinitely if there is not enough available resources in the cluster for the node to successfully drain. It is important to consider the capacity not just of the cluster as a whole, but for any scheduling groups, such as nodes with a common taint, when using labels + node selectors, and/or when using (anti)affinity rules for workloads.

If cluster updates are taking too long to complete it's possible to reduce the time needed by changing the parallelism of the updates being applied. This is configured on a per-machine pool basis where the administrator may set a specific number of nodes which can be unavailable at a time, e.g. 2, 5, etc., or a percentage of the overall node count, e.g. 15% or 50%.

Control plane updates

In most environments, the control plane and compute nodes are updated together as part of a unified upgrade to each OpenShift minor release (e.g., from 4.17 to 4.18). However, OpenShift also supports a control plane-only upgrade strategy, which can offer additional operational flexibility and options for managing maintenance windows.

The control plane must be updated serially, meaning that when an update across more than one version is needed the control plane must be updated to each intermediate version. However, compute nodes do not need to be updated in this way. Control plane-only updates are typically used when upgrading across multiple minor versions (e.g. 4.16 to 4.18) with the goal to avoid having to reboot compute nodes with each intermediate version. This has the benefit of fewer live migrations for VMs as the nodes are cordoned and drained while also significantly shortening the maintenance window required for the cluster since it only involves one reboot for a two version upgrade.

A control plane-only update is done by pausing the compute machine config pools and initiating a cluster update, resulting in only the control plane and its services being updated. Once the control plane has been updated, the machine config pools are unpaused and the compute nodes are updated directly to the new version. Importantly, this is only supported across a maximum of two versions at a time.

Operator updates

Features and functions of OpenShift are implemented and lifecycled using the operator paradigm. The operators are managed using Operator Lifecycle Manager to perform updates across them, including orchestrating the order in which updates need to be done to satisfy dependencies.

Feature operators can also prevent cluster updates from happening if they are not compatible. The cluster update mechanism will warn the administrator if any operators are not compatible with the desired version of OpenShift so that the administrator can then manually update those operators as needed.

The administrator can choose to set feature operators to have an automatic or manual update policy. When set to automatic, Operator Lifecycle Manager will apply updates to the operators when they become available so long as all other criteria are met. This can be convenient for ensuring that features are transparently at the latest version, however it can also introduce risk if you are uncomfortable using updates as soon as they are released. If that is the case, setting a manual update policy gives the administrator the option of when to update the operator and what version to update to.

Virtual machine performance

Hypervisor and virtual machine performance depends on multiple factors, including hardware capabilities, cluster and node configurations, and hypervisor settings, which all directly influence application performance within virtual machines. Refer to the OpenShift Virtualization performance and scaling guide for additional details.

Performance features and Operators

OpenShift provides advanced features that are designed to enhance virtual machine performance and provide allocation of dedicated resources when needed. To use these capabilities, you must deploy and configure specific OpenShift Operators and related performance features.

CPU manager and topology manager

CPU Manager provides exclusive CPU allocation, also known as CPU pinning, by dedicating specific CPUs to individual VMs. When used in combination with Topology Manager, it provides additional optimization by including NUMA (Non-Uniform Memory Access) node awareness. Topology Manager uses scheduling hints and KVM's capabilities to ensure that workloads have optimal access to memory and processor caches. This can reduce latency for sensitive workloads, particularly those requiring predictable CPU performance and exclusive compute allocation, such as the compute-exclusive instance type. It is recommended to deploy and configure both CPU Manager and Topology Manager to enable workloads with strict performance requirements.

Deployment and configuration of CPU Manager and Topology Manager are handled through machine config. While creating different configurations for specific groups of machines is supported, you must carefully consider whether to use this approach due to potential complications and limitations that it introduces. Differing configurations across nodes can restrict VM scheduling flexibility and limit live migration opportunities, because workloads might become limited in their ability to migrate between nodes with mismatched configurations.

When configuring Topology Manager, the recommended policy is either best-effort or single-numa-node. The best-effort policy attempts NUMA alignment but does not strictly enforce it, allowing the VM to schedule even when the requested NUMA alignment isn't possible. This flexibility is beneficial for general-purpose workloads and ensures broader node scheduling availability. The single-numa-node policy, however, enforces strict resource allocation, ensuring that VM resources, both CPU and memory, reside within a single NUMA node. This provides maximum performance predictability and is ideal for latency-sensitive workloads. However, it comes at the cost of scheduling flexibility, because the VM will not start if a single NUMA node cannot accommodate the requested resources fully.

Huge pages

Huge pages enable VMs to use larger memory allocation units, 2 MiB or 1 GiB pages, instead of the standard 4 KiB pages. Huge pages help improve performance and reduce latency by significantly reducing page table overhead and lowering the frequency of Translation Lookaside Buffer (TLB) misses. They are particularly beneficial for VMs hosting memory-intensive, latency-sensitive, or high-performance workloads.

OpenShift uses transparent huge pages (THP) by default, which relies on the kernel to automatically create and use 2 MiB huge pages whenever possible for applications, including

VMs, running in the cluster. As a result, in most cases VMs will experience the performance benefits of huge pages without performing additional configuration by the administrator or the application.

When using transparent huge pages, swap, when configured using wasp agent, will continue to work as expected. The system will "break" the 2 MiB pages into standard 4 KiB pages for this purpose. However, kernel samepage merging (KSM) will be less effective when enabled and used with transparent huge pages.

For workloads that need guaranteed access to high performance memory space, including applications using huge pages larger than 2 MiB or those that are sensitive to how THP allocates and manages the huge pages, then preallocating huge pages at boot time is the supported and recommended option. Additionally, any virtual machine that has a request for huge pages in its definition, regardless of the size of the huge pages, will require preallocated huge pages to meet that need.

Preallocated huge pages guarantee access to resources, however, they do come with some trade offs. Importantly, preallocated huge pages cannot be used by VMs which do not have an explicit request for huge pages, meaning that if no VMs are requesting the huge pages then the memory will not be used, regardless of other memory pressure on the node. Additionally, VMs using preallocated huge pages are not eligible for swap or KSM.

If no specific requirements for huge pages exist, then it is recommended to keep transparent huge pages enabled in your cluster(s). If there are specific workloads that need guaranteed access to huge pages or huge pages other than 2 MiB in size, then preallocate them at boot time and configure as needed. Consider identifying a subset of nodes for hosting these workloads to avoid excess memory being allocated to huge pages and not available for other virtual machines.

Swap

Traditionally, swap is disabled and not available for OpenShift workloads. However, VMs, via wasp agent, are able to be swapped, rather than shutdown and evicted or Out of Memory (OOM) terminated, when the host has been configured appropriately. Configuring the host involves using machine config to create a swap location, which could be a partition, a dedicated disk, or a file on an existing disk, and enable swap at the operating system level.

The amount of swap to configure on the node(s) is primarily dependent on two factors: the overcommitment level you're expecting and the overall utilization of the cluster.

- In clusters with both the high overcommitment ratio and high total utilization, more swap space is recommended as a result of fewer opportunities for the workload to be migrated to a node that is not experiencing memory pressure.
- Lower total utilization, but high overcommit ratio, could increase the risk for swapping, but only on one, or a small number, of hosts at a time. This means that a smaller amount of swap space is needed since it would be expected to be a temporary and localized impact to utilization space.
- Even when no swap usage is expected, it's still important to configure some capacity to account for short bursts in memory utilization. For example, when many VMs are powered-on simultaneously, to allow them to overflow into swap without triggering node memory pressure actions.

Swap device performance is also important to the overall experience and directly affects the VM and its application(s). Any memory being swapped has a significant performance impact for the VMs. You can mitigate the impact by using high performance flash-based disk, such as NVMe. Importantly, all VMs using swap on the node will share this device. Individual VMs do not have their own disk, local or PVC, to use for swap. This can have an effect on RHCOS and its node functionality, such as kubelet actions, if the swap location is the same disk that is used for RHCOS.

Avoiding resource contention

Contention for node resources, such as CPU or memory, has a significant impact on the workload's performance. OpenShift's features focus less on evenly balancing resource consumption across the cluster members, and focus more on contention avoidance whenever possible, relying on several remediation options.

Observability

Proactive observability is one of the most critical tools for effective resource management and contention avoidance in OpenShift Virtualization. Administrators must routinely use OpenShift's built-in observability features to analyze both historical trends and real-time metrics related to resource utilization and availability. Long-term monitoring helps inform capacity planning, while real-time visibility is essential for detecting emerging performance bottlenecks, before they affect VM workloads. It is important to enable and monitor Pressure Stall Information (PSI) metrics on all nodes. PSI exposes how CPU, memory, and I/O contention affect VMs, and other processes, on the host. Monitoring these signals provides critical insight into early symptoms of oversubscription and resource starvation. This enables administrators to intervene before contention escalates into service degradation or workload disruption.

Secondary scheduler and node feature rules

When a VM is "powered on" the VM is scheduled, by the control plane, to a specific node in the cluster. The scheduler analyzes many factors to identify a suitable node for the VM, including the virtual machine's CPU and memory resource requests, current node availability, and any defined (anti)affinity rules. However, beyond ensuring that sufficient resources are available and affinity rules are respected, the default scheduler does not consider overall node utilization or broader cluster resource utilization when making placement decisions. The primary objective is to avoid immediate contention rather than optimizing workload distribution or balancing long-term resource utilization across the cluster.

While this default behavior is typically sufficient, virtualization architects may encounter scenarios requiring more robust scheduling capabilities. These scenarios must take into account more resource factors, including factoring in real-time utilization of more resource metrics when making scheduling decisions. OpenShift's secondary scheduler capability, when used with plugins such as Trimaran, is able to factor-in these additional metrics and take into account node and cluster utilization when deciding where to schedule a VM.

For additional granularity to identify nodes with specialized hardware and/or configuration, use node feature rules to enable precise labeling of nodes possessing specialized hardware resources, such as GPUs, accelerators, or specific CPU features. This makes resources able to be specifically requested and accounted for when scheduling. Utilizing node feature rules, alongside Trimaran, can significantly enhance resource scheduling functionality, more effectively distribute workload. This can potentially improve the overall performance and efficiency of OpenShift virtualized environments. Configuring the secondary scheduler is not a default recommendation for all deployments. However, you may have the need to more evenly balance resources across nodes or consider factors beyond just CPU and memory that need to be used for scheduling.

Descheduler

The descheduler is an OpenShift feature designed to optimize workload distribution and resource utilization across cluster nodes by proactively triggering virtual machine live migrations. It continuously monitors node resource utilization, such as CPU and memory, against configurable thresholds. By default, it considers nodes overutilized when resource usage exceeds 50% and underutilized when usage is below 20%, however these values can be adjusted for the cluster. When both of these conditions are present, the descheduler initiates live migrations of virtual machines on overutilized nodes. If both conditions, underutilized nodes and overutilized nodes, are not present, then no action is taken. For example, if all nodes are above 20% utilization, then no underutilized nodes are present and no action is taken by the descheduler.

Using the descheduler with OpenShift Virtualization is recommended, as it helps maintain balanced resource utilization and attempts to avoid resource contention. After the descheduler is deployed and configured, the VM definitions must be updated to include an annotation indicating that they are eligible to be evicted. For existing VMs, this change can be applied using standard methods, however it is recommended to update any templates and/or instance type definitions to include the appropriate annotation. Specific VMs can be excluded from descheduler actions by removing the label.

Soft eviction

When node resources, specifically memory, reach certain utilization thresholds different actions are automatically taken by the cluster and/or the node to protect high-priority workloads and preserve stability. Importantly, there is a chain of escalating actions that happen when node memory resources approach being extinguished:

- The descheduler attempts to live migrate VMs away from nodes with utilization above its configured value when underutilized nodes are present.
- When configured, wasp agent swaps VMs using the configured system swap space. If the VMs continue to swap beyond the configured thresholds, then the VMs will be live migrated to another node.
- The soft eviction threshold attempts to safely shutdown the VM so that it can be rescheduled and restarted on a different node in the cluster.
- Reaching the hard eviction threshold results in workload being OOM_kill by the node. This is an immediate termination of the VM and is akin to a crash from the perspective of the guest operating system.

To avoid OOM_kill terminations it is recommended to configure a soft eviction threshold for memory utilization. Configure the threshold to leave enough resources / capacity before reaching the hard eviction threshold for the system to take action to safely shutdown VMs. Set the soft eviction threshold between 1 and 10% of the amount of node memory, depending on the size of your nodes.

Host memory optimization

The node operating system (RHEL CoreOS) and the hypervisor (KVM) have multiple features that increase the density of VM memory.

Kernel samepage merging (KSM)

Kernel samepage merging is a memory optimization technique used by the Linux kernel, and supported by OpenShift Virtualization, to reduce memory usage across VMs. It works by identifying identical memory pages in different VMs on the same host and consolidating them into a single, shared read-only page, allowing multiple VMs to reference the same memory content while preserving isolation. When a VM attempts to modify a shared page, a copy-on-write operation ensures memory integrity. KSM can be beneficial in environments with many VMs with similar applications and operating systems, such as VDI or dev/test clusters, as it improves memory density and allows for greater VM consolidation without overprovisioning physical RAM. However, there is increased CPU overhead needed by the scanning process and KSM does not work with huge pages.

KSM is not enabled by default with OpenShift Virtualization. You can enable KSM if you expect to have VMs with a high likelihood of identical memory and do not use huge pages. Transparent huge pages, which is enabled by default, affects the effectiveness of KSM in OpenShift Virtualization.

Free page reporting

Free page reporting is a memory management feature that allows guest VMs to proactively inform the hypervisor about unused memory pages. This enables more efficient memory reclamation and overcommitment. Traditionally, hypervisors rely on techniques like ballooning or kernel samepage merging to reclaim memory, which either require coordination with guest agents or rely on indirect methods like scanning for identical pages. Free page reporting improves upon this by enabling the guest kernel to mark unused pages that the hypervisor can then reclaim immediately, with minimal overhead. This is important in overcommitted clusters where multiple VMs are competing for physical RAM.

This feature is enabled by default with OpenShift Virtualization and VMs can be configured to use free page reporting to maximize its effectiveness. The default instance types and templates included with OpenShift Virtualization include the appropriate configuration.

Managing virtual machine configuration

Effectively managing VM configurations within OpenShift Virtualization involves utilizing a range of flexible, complementary methods. Administrators and users can specify VM configurations manually, utilize templates for rapid provisioning of standardized workloads, or use reusable constructs such as instance types and preferences to more easily manage resources and behavior across multiple VMs. The following items summarize the configuration options to consider when designing your VM environments.

- Manually, using the VM's definition to specify all of the details for each instance individually. While this provides maximum flexibility, it can lead to inconsistency and complexity at scale.
- Templates offer reusable blueprints to quickly provision new VMs with consistent initial configurations. When a VM is created from a template, a copy of the template configuration is applied, providing a standardized baseline. However, any adjustments to the template are only reflected in VMs newly created from the template and do not affect existing VMs. Changing configuration of existing VMs typically uses automation tools, such as Ansible Automation Platform.
- Instance types define standardized resource allocations, such as CPU topology, core counts, and memory size, which can be referenced by multiple VM definitions. Changes to an instance type automatically propagate to all associated VMs, facilitating consistency and simplifying resource management, particularly for environments with large numbers of VMs. Instance types primarily answer the question of "how much?", providing consistent resource sizing across VMs
- Preferences specify behavioral and device defaults for a VM. Most commonly, these define the low-level virtual hardware adjustment and tuning that is used on a per-guest operating system basis. For example, a preference specifies the device types or "model" to use for the virtual disk bus, network adapters, and more. Similar to instance types, preferences are added to a VM using a reference to the object.

Each approach serves a distinct purpose:

- Manual definitions maximize flexibility.
- Templates ensure consistent baseline provisioning.
- Instance types standardize resource allocations such as CPU and memory, and preferences manage device-level defaults and VM behaviors specific to the guest OS.

While there are no rules for managing configuration, a general guideline is that when only a single, or small number, of VMs need specific features adjusted, then it's often easiest to apply the configuration directly in the VM's definition. However, when the adjustments are more broad, affecting more than a handful of VMs, or to be made available to VM users who do not want to adjust low level configuration themselves, then adjusting the instance type, preference, or template to apply configuration is the ideal solution.

Default instance types

OpenShift Virtualization creates a default set of instance types that offer optimizations for the broad use cases associated with each instance type category. Each of these provides specific configuration for access to resources inline with the expected use case. Importantly, with the exception of the general purpose and overcommitted instance types, this means that additional features must be configured within the OpenShift cluster for the instance type to work. Using an instance type without the associated feature(s) configured will result in an unschedulable VM

- General Purpose instance types offer no specific optimizations for compute, memory, network, or storage resources. By default, VMs using these profiles follow the global overcommitment configuration.
    - Required features: none.
- Network instance types configure the VMs for better performance when they are heavy network consumers. This includes features such as dedicated huge pages, dedicated CPU pavement, and additional hypervisor-level configuration inline with network-optimized workload. This instance type is well suited for virtualized network functions (VNFs).
    - Required features: CPU manager, hugepages (1 GiB).
- Overcommitted instance types set a static memory overcommit percentage on the VM, which is implemented by reducing the VM's memory request by the specified amount. This provides flexibility for creating categories of performance expectations, e.g. the default is to not overcommit VM memory however selected VMs will overcommit according to their configuration, with higher overcommit percentage used for non-production applications.
    - Required features: none.
- Compute Exclusive provides exclusive CPU resources for the VM, including dedicated CPU placement, passthrough NUMA mapping for the guest operating system, and dedicated huge pages for memory. Furthermore, this instance type configures the hypervisor to separate CPU resources for running the VM, e.g. IO threads, from the VM's other processes, providing further improvements to resource access.
    - Required features: CPU manager, hugepages.

- Memory Intensive instance types use hugepages to allocate host memory resources for the VM.
    - Required features: hugepages (2 MiB).

Before using the resource-specific instance types, it is strongly recommended to review them to ensure that they align with your workload requirements. In particular, the CPU-to-memory ratio used by the instance type, and identify any configuration changes needed to accommodate VMs using the instance type. For example, the network instance types will require nodes to have an additional label applied. Also consider creating custom instance types with only the optimizations and/or sizing needed.

Disaster Recovery

In general, if separate availability zones cannot be guaranteed for the cluster control plane and worker nodes to assist with mitigation of a site failure, then the best option available is to deploy additional OpenShift clusters across multiple sites and establish a disaster recovery policy. In the policy, define the actions required to recover applications in the event of a cluster failing disaster at one site. In some cases this can be achieved by placing a global load balancer in front of a number of clusters at remote sites, ensuring that one cluster can still service applications in the case where its partner fails. However, this does require application awareness and monitoring to ensure that the correct instance is active and any persistent data used by the application is replicated appropriately.

To achieve this, Red Hat has made available the OpenShift APIs for Data Protection (OADP) as an operator in OpenShift to enable users to create backups and restore workload in OpenShift. For more details on OADP, please see the documentation available here.

The storage used for OpenShift Virtualization may offer the ability to replicate or otherwise protect the virtual machine disks and metadata. The disaster recovery guide offers several options for how to protect and recover virtual machines with varying recovery time object (RTO) and recovery point object (RPO) goals.

Additional options for backup, restore, and disaster recovery exist by making use of products from a number of Red Hat Partners that provide the ability to back up and restore stateful applications from one OpenShift cluster to another. Several of these partners focus on backup and recovery for applications in OpenShift and also currently support OpenShift Virtualization workloads. In addition to these partners that focus exclusively on backup and recovery operations, there are storage vendors with native backup functionality that also support OpenShift Virtualization.

For a list of validated disaster recovery partners, visit https://red.ht/workswithvirt, or contact your preferred storage or backup/recovery provider to confirm support for OpenShift Virtualization.

Summary

OpenShift Virtualization is a fully developed virtualization solution utilizing the type-1 KVM hypervisor from Red Hat Enterprise Linux with the powerful features and capabilities of OpenShift for managing hypervisor nodes, VMs, and their consumers. By providing a single control plane for managing both containers and VMs, OpenShift can streamline many administrative actions and assist with expediting the deployment of full application stacks that involve both of these resource types. Customers can also be assured of the value provided by OpenShift Virtualization in comparison to other enterprise virtualization solutions, as it also maintains many of the same hypervisor features expected of such a solution, such as live migration, virtual guest cloning, and integration with third-party tools for VM management. Carefully planning your implementation architecture is a critical step in a successful deployment of your virtualization solution. The following links and references provide additional information to consider for your implementation.

Additional Links and References

Documentation and KCS articles

- OpenShift Virtualization documentation
- OpenShift Virtualization - Cluster Sizing Guide
- OpenShift Virtualization - Tuning & Scaling Guide
- OpenShift Virtualization - Fencing and VM High Availability Guide
- Red Hat OpenShift Virtualization disaster recovery guide

Appendix A. Change log

May 2025 - version 1.0.0

- Initial release.