



# OpenShift Virtualization

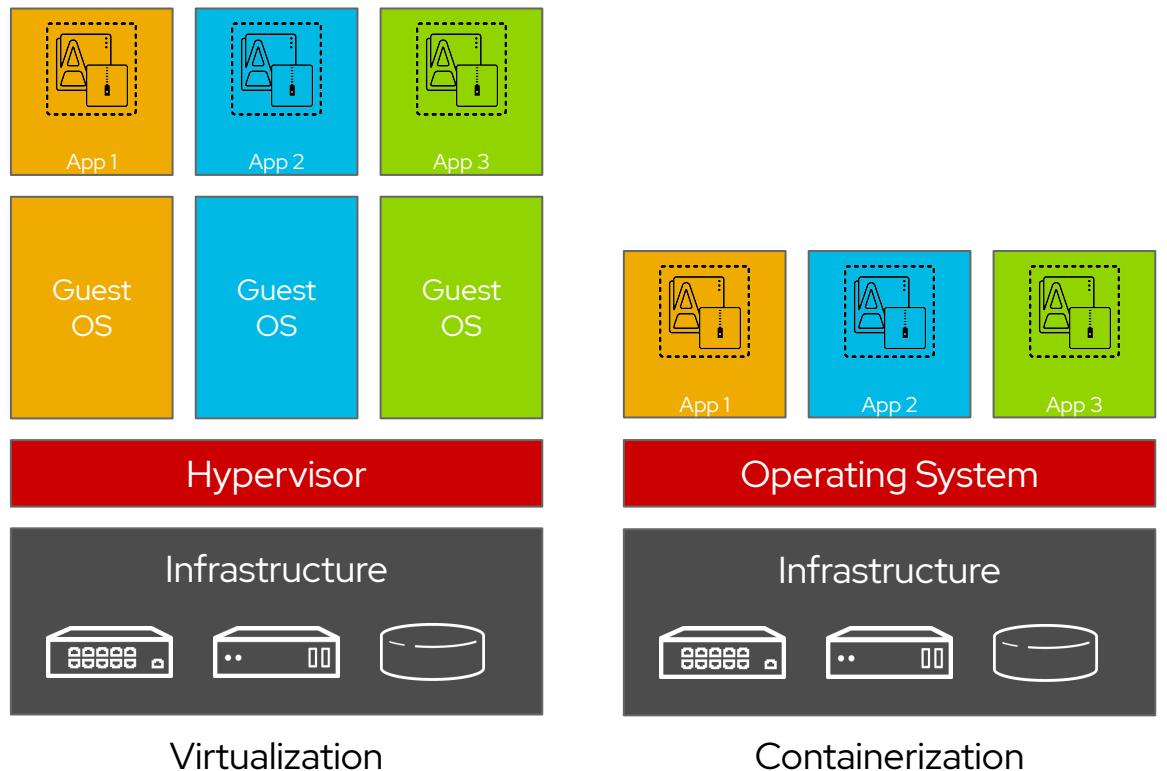
Technical presentation

Alfred Bach  
Principal Solution Architect  
EMEA Partner Enablement

# What is OpenShift Virtualization?

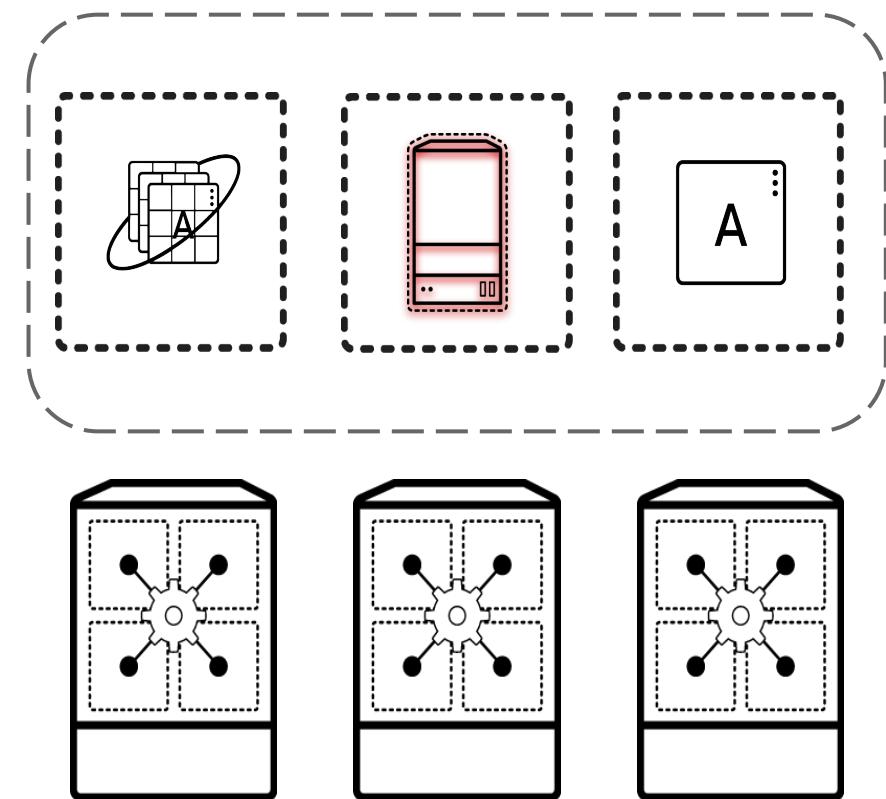
# Containers are not virtual machines

- Containers are process isolation
- Kernel namespaces provide isolation and cgroups provide resource controls
- No hypervisor needed for containers
- Contain only binaries, libraries, and tools which are needed by the application
- Ephemeral



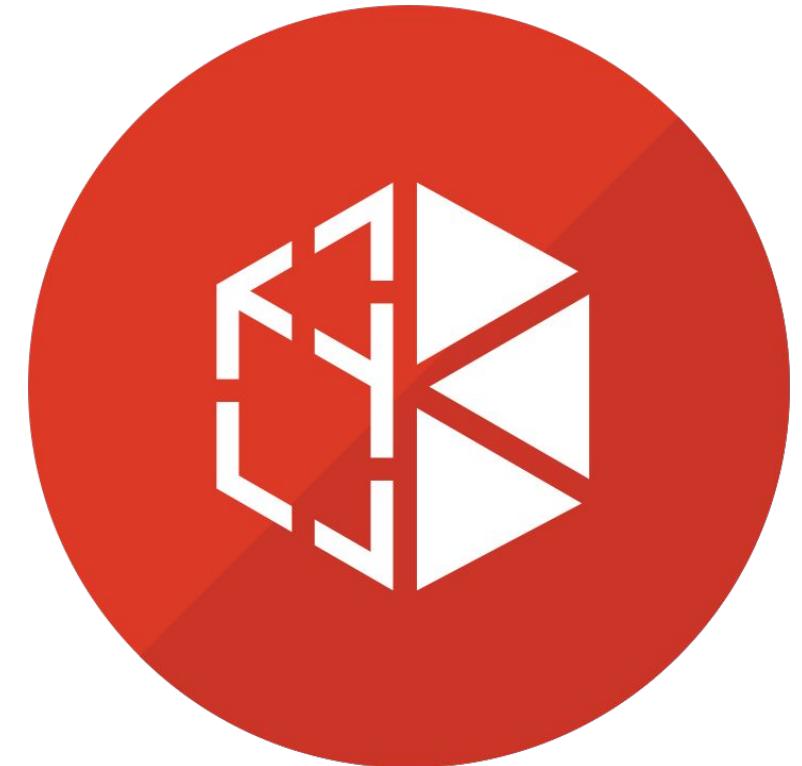
# Virtual machines can be put into containers

- A KVM virtual machine is a process
- Containers encapsulate processes
- Both have the same underlying resource needs:
  - Compute
  - Network
  - (sometimes) Storage



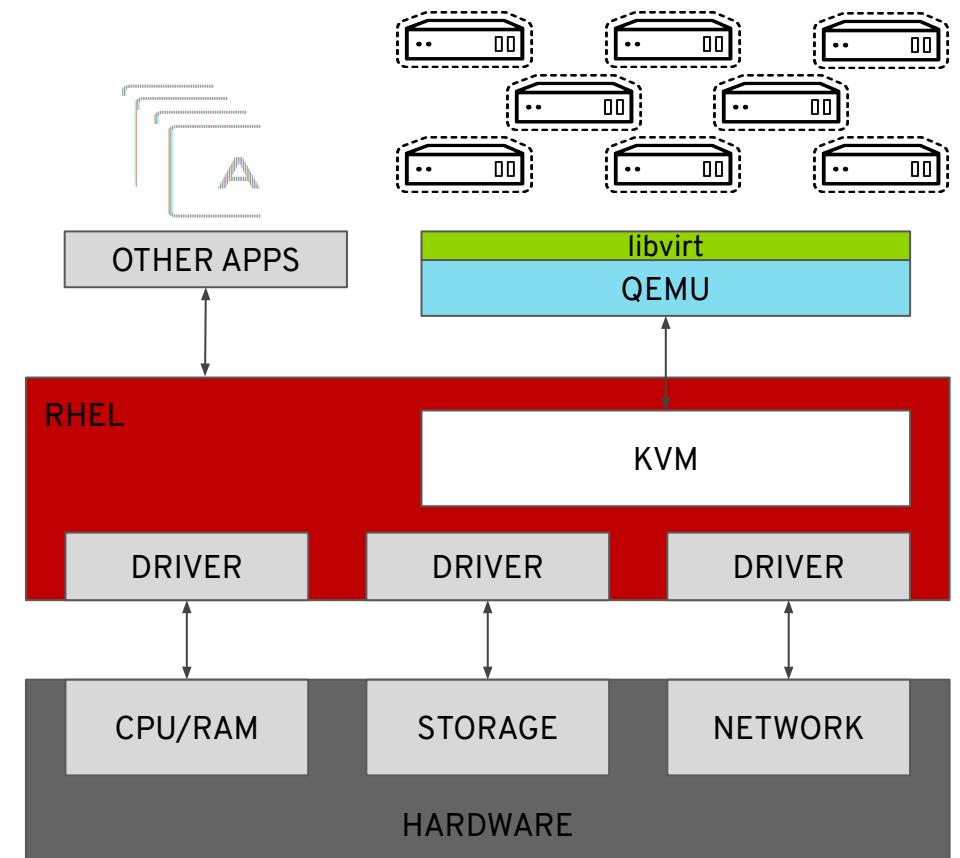
# OpenShift Virtualization

- Virtual machines
  - Running in containers
  - Using the KVM hypervisor
- Scheduled, deployed, and managed by Kubernetes
- Integrated with container orchestrator resources and services
  - Traditional Pod-like SDN connectivity and/or connectivity to external VLAN and other networks via multus
  - Persistent storage paradigm (PVC, PV, StorageClass)



# VM containers use KVM

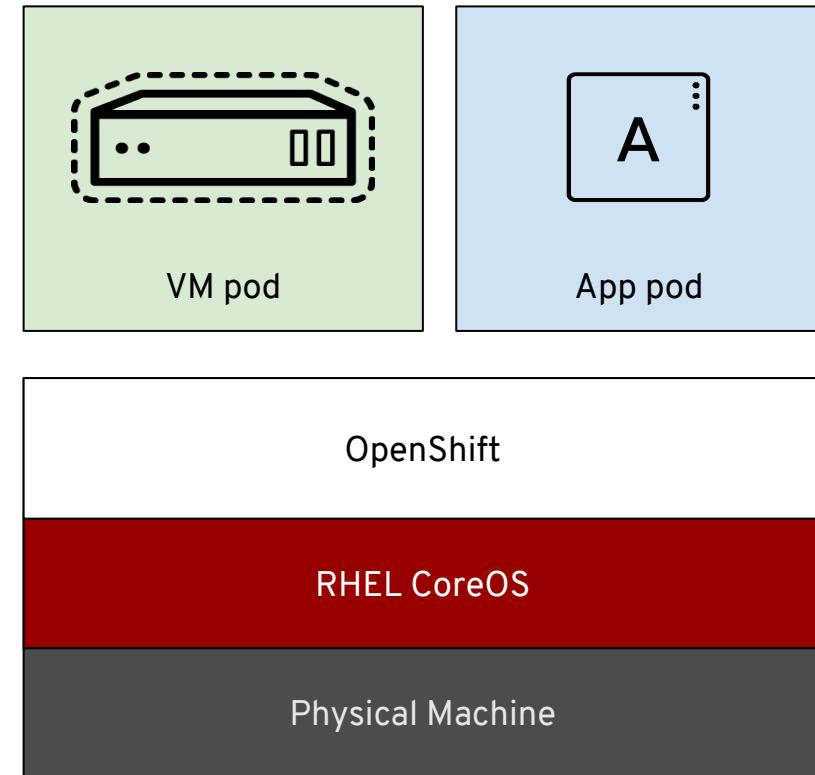
- OpenShift Virtualization uses KVM, the Linux kernel hypervisor
- KVM is a core component of the Red Hat Enterprise Linux kernel
  - KVM has 10+ years of production use: Red Hat Virtualization, Red Hat OpenStack Platform, and RHEL all leverage KVM, QEMU, and libvirt
- QEMU uses KVM to execute virtual machines
- libvirt provides a management abstraction layer



# Built with Kubernetes

# Virtual machines in a container world

- Provides a way to transition application components which can't be directly containerized into a Kubernetes system
  - Integrates directly into existing k8s clusters
  - Follows Kubernetes paradigms:
    - Container Networking Interface (CNI)
    - Container Storage Interface (CSI)
    - Custom Resource Definitions (CRD, CR)
- Schedule, connect, and consume VM resources as container-native

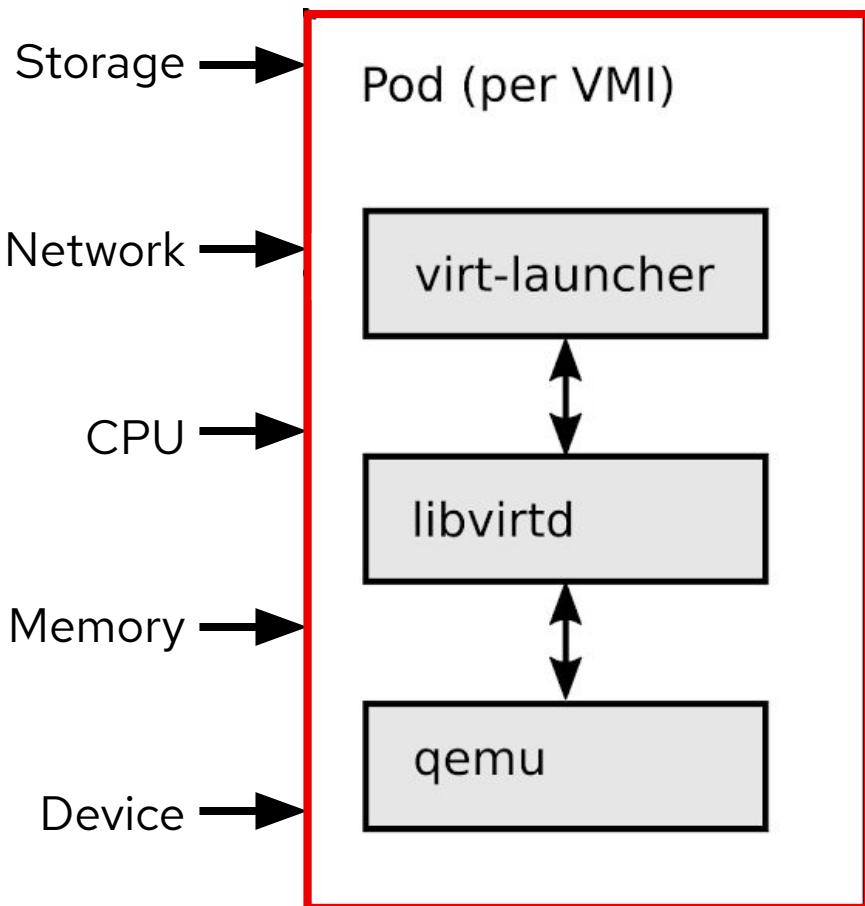


# Virtualization native to Kubernetes

- Operators are a Kubernetes-native way to introduce new capabilities
- New CustomResourceDefinitions (CRDs) for native VM integration, for example:
  - `VirtualMachine`
  - `VirtualMachineInstance`
  - `VirtualMachineInstanceMigration`
  - `DataVolume`

```
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachine
metadata:
  labels:
    app: demo
    flavor.template.kubevirt.io/small: "true"
  name: rhel
spec:
  dataVolumeTemplates:
  - apiVersion: cdi.kubevirt.io/v1alpha1
    kind: DataVolume
    metadata:
      creationTimestamp: null
      name: rhel-rootdisk
    spec:
      pvc:
        accessModes:
        - ReadWriteMany
      resources:
        requests:
          storage: 20Gi
      storageClassName: managed-nfs-storage
      volumeMode: Filesystem
```

# Containerized virtual machines



## Kubernetes resources

- Every VM runs in a launcher pod. The launcher process will supervise, using libvirt, and provide pod integration.

## Red Hat Enterprise Linux

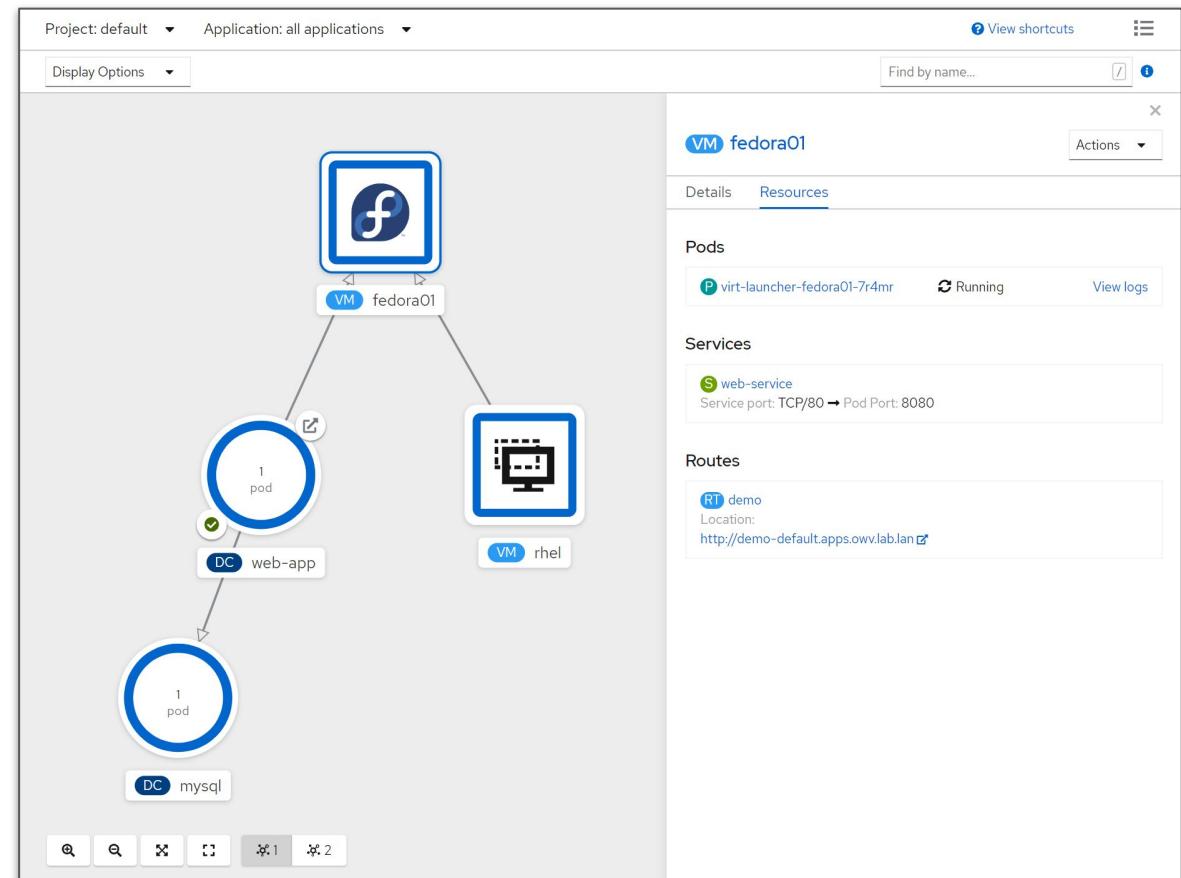
- libvirt and qemu from RHEL are mature, have high performance, provide stable abstractions, and have a minimal overhead.

## Security - Defense in depth

- Immutable RHCOS by default, SELinux MCS, plus KVM isolation - inherited from the Red Hat Portfolio stack

# Using VMs and containers together

- Virtual Machines connected to pod networks are accessible using standard Kubernetes methods:
  - Service
  - Route
  - Ingress
- Network policies apply to VM pods the same as application pods
- VM-to-pod, and vice-versa, communication happens over SDN or ingress depending on network connectivity



# Managed with OpenShift

# Virtual Machine Management

- Create, modify, and destroy virtual machines, and their resources, using the OpenShift web interface or CLI
- Use the `virtctl` command to simplify virtual machine interaction from the CLI

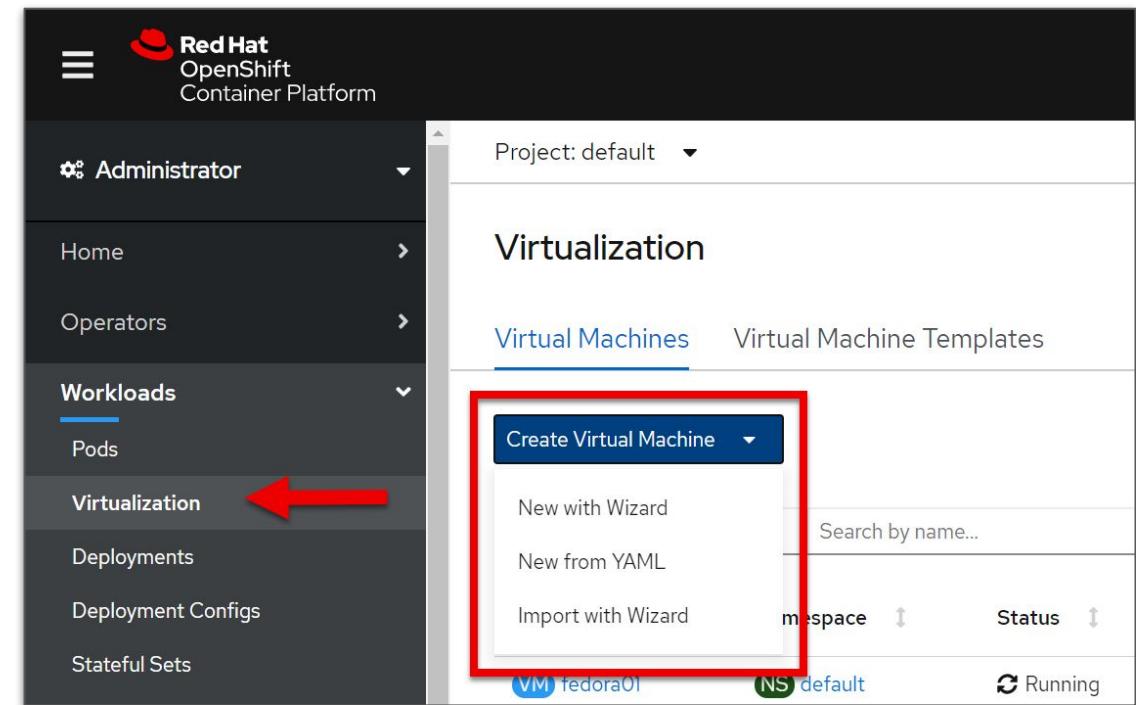
The screenshot shows the Red Hat OpenShift Container Platform web interface. The left sidebar has a dark theme with the Red Hat logo at the top. The 'Workloads' section is expanded, showing 'Virtualization' as the active category. The main content area is titled 'Virtualization' and shows a table of 'Virtual Machines'. The table includes columns for Name, Namespace, Status, Created, Node, and IP Address. There are four entries in the table:

Name	Namespace	Status	Created	Node	IP Address
VM fedora01	NS default	Running	Jul 9, 5:00 pm	N worker-0.ovw.lab.lan	10.131.0.74
VM rhel	NS default	Running	Jul 8, 4:18 pm	N worker-0.ovw.lab.lan	192.168.14.163/24, fe80::87cc:48e:1e2: 9d23/64
VM rhel01	NS default	Off	Jul 9, 4:58 pm		
VM windows2019	NS default	Running	Jul 9, 5:01 pm	N worker-1.ovw.lab.lan	10.128.2.52

# Create VMs

# Virtual Machine creation

- Streamlined and simplified creation via the GUI or create VMs programmatically using YAML
- Full configuration options for compute, network, and storage resources
  - Clone VMs from templates or import disks using DataVolumes
  - Pre-defined and customizable presets for CPU/RAM allocations
  - Workload profile to tune KVM for expected behavior
- Import VMs from VMware vSphere or Red Hat Virtualization



# Create Virtual Machine - General

- Source represents how the VM will boot
  - Boot via PXE, optionally diskless
  - URL will import a QCOW2 or raw disk image using a DataVolume
  - Container uses a container image, pulled from a registry, for the disk
  - Disk uses an existing PVC
- Flavor represents the preconfigured CPU and RAM assignments
  - Tiny = 1 vCPU and 1GB RAM, Small = 1 vCPU and 2GB RAM, etc.
- Workload profile defines the category of workload expected and is used to set KVM performance flags

The screenshot shows the 'Create Virtual Machine' wizard in progress, specifically the 'General' step (step 1). The interface includes a sidebar with steps 1 through 6: General, Networking, Storage, Advanced, Cloud-init, Virtual Hardware, Review, and Result. The 'General' step is selected. On the right, there are fields for 'Name' (with a required asterisk), 'Description', and 'Template' (showing 'No template available'). Below these are three dropdown menus:

- 1** 'Source' dropdown: Options include '--- Select Source ---', 'PXE', 'URL', 'Container', and 'Disk'. The 'Container' option is currently selected.
- 2** 'Flavor' dropdown: Options include 'Custom', '--- Select Flavor ---', 'Tiny', 'Small', 'Medium', 'Large', and 'Custom'. The 'Custom' option is currently selected.
- 3** 'Workload Profile' dropdown: Options include '--- Select Workload Profile ---', 'desktop', 'highperformance', and 'server'. The 'server' option is currently selected.

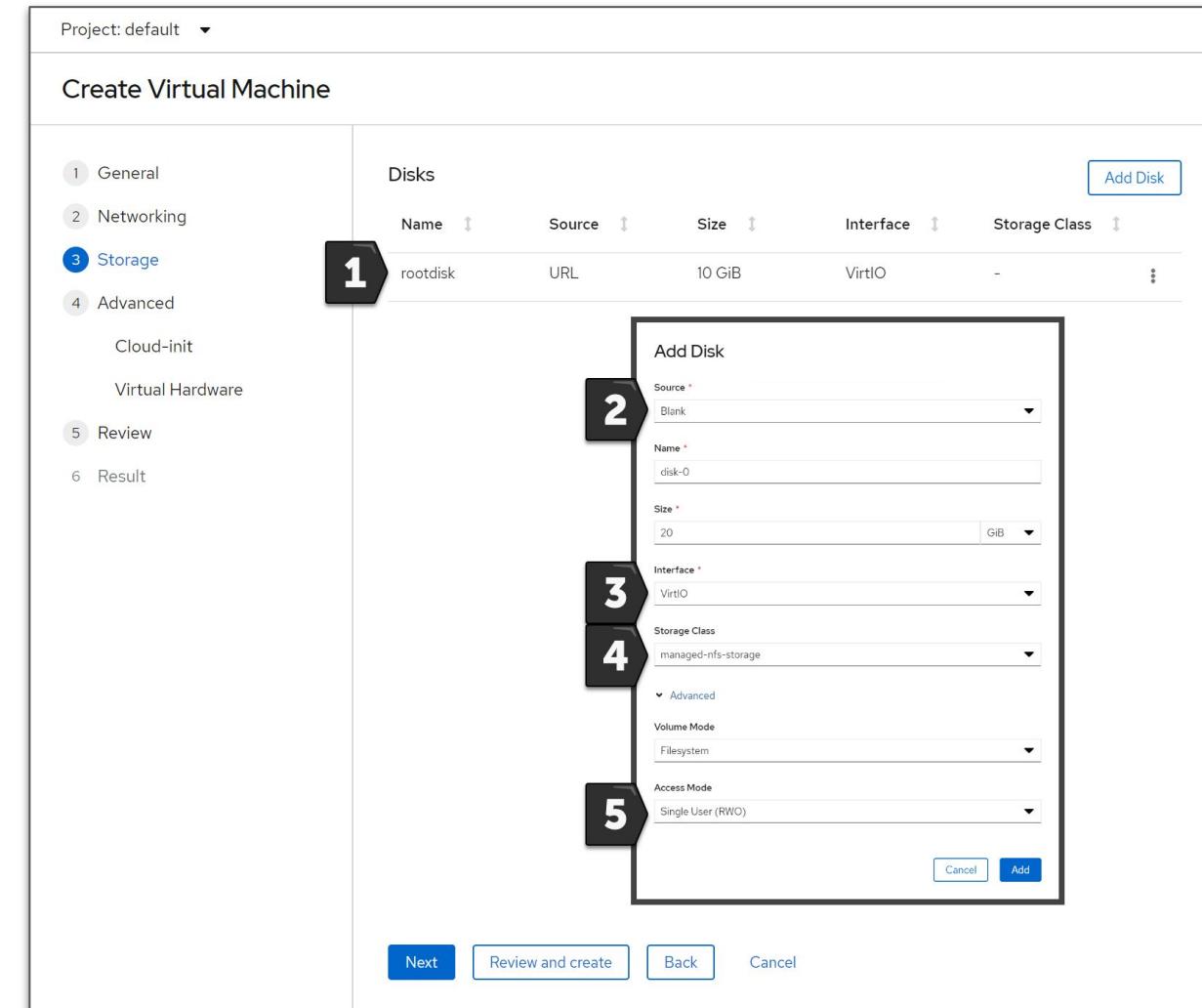
# Create Virtual Machine - Networks

- Add or edit network adapters
- One or more network connections
  - Pod network for the default SDN
  - Additional multus-based interfaces for specific connectivity
- Multiple NIC models for guest OS compatibility or paravirtualized performance with VirtIO
- Masquerade, bridge, or SR-IOV connection types
- MAC address customization if desired

The screenshot shows the 'Create Virtual Machine' wizard in progress, specifically the 'Networking' step (step 2). The main interface displays a table of existing network interfaces: 'nic-0' (Model: VirtIO, Network: Pod Networking, Type: masquerade). A large black arrow labeled '1' points to the 'Add Network Interface' button in the top right corner of the table header. An 'Add Network Interface' dialog box is overlaid on the right side of the screen, containing fields for Name (nic-1), Model (VirtIO), Network (host-br1), Type (bridge), and MAC Address (empty). The dialog has 'Cancel' and 'Add' buttons at the bottom. The wizard's navigation steps are visible on the left: General (1), Networking (2), Storage (3), Advanced (4), Cloud-init (5), Virtual Hardware (6), Review (5), and Result (6). Buttons at the bottom include 'Next', 'Review and create', 'Back', and 'Cancel'.

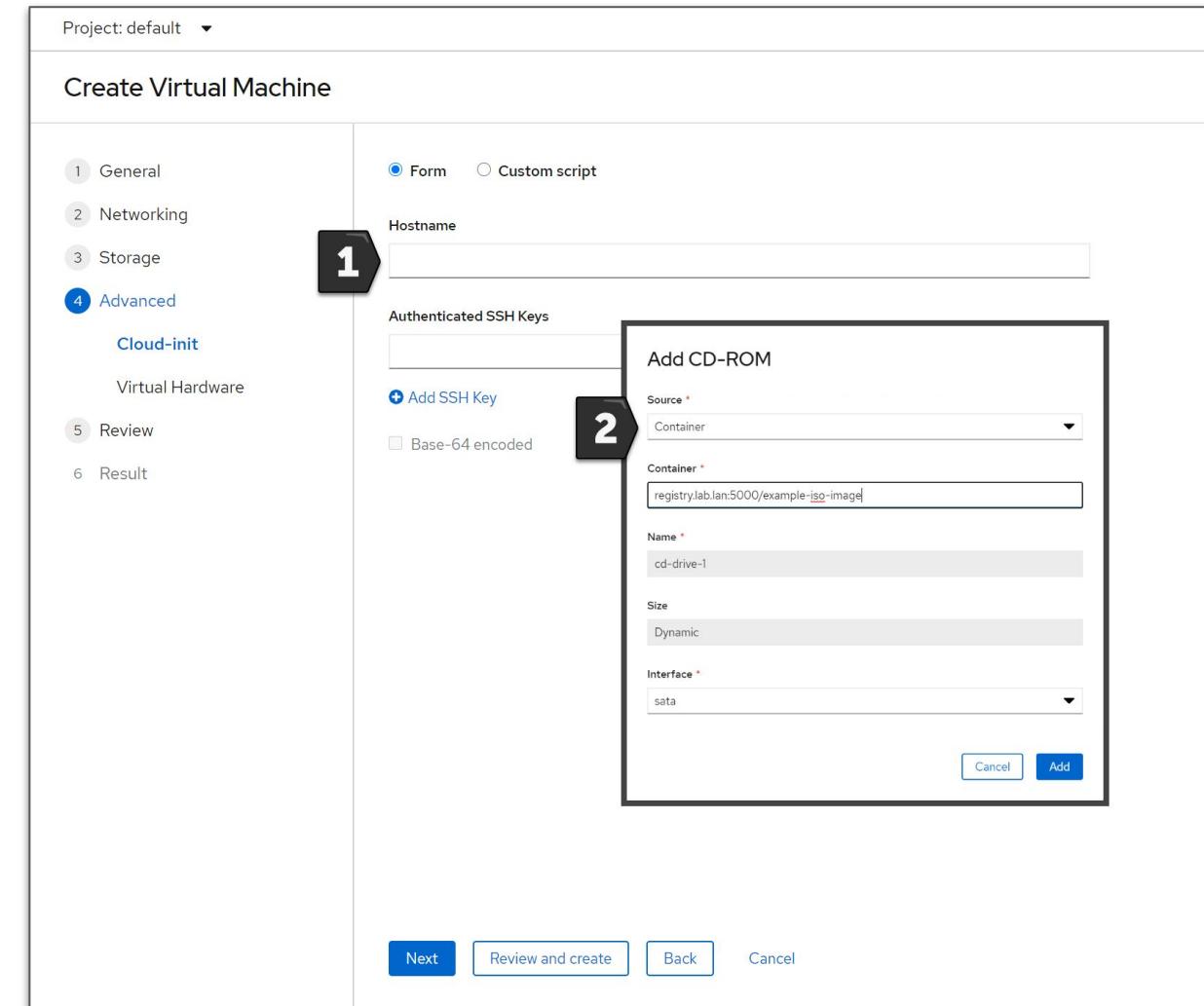
# Create Virtual Machine - Storage

- Add or edit persistent storage
- Disks can be sourced from
  - Imported QCOW2 or raw images
  - New or existing PVCs
  - Clone existing PVCs
- Use SATA/SCSI interface for compatibility or VirtIO for paravirtual performance
- For new or cloned disks, select from available storage classes
  - Customize volume and access mode as needed



# Create Virtual Machine - Advanced

- Customize the operating system deployment using cloud-init scripts
  - Guest OS must have cloud-init installed
  - RHEL, Fedora, etc. cloud images
- Attach ISOs to the VM CD/DVD drive
  - ISOs stored in container images (registry), existing PVC, or imported from URL



# Create Virtual Machine – Review

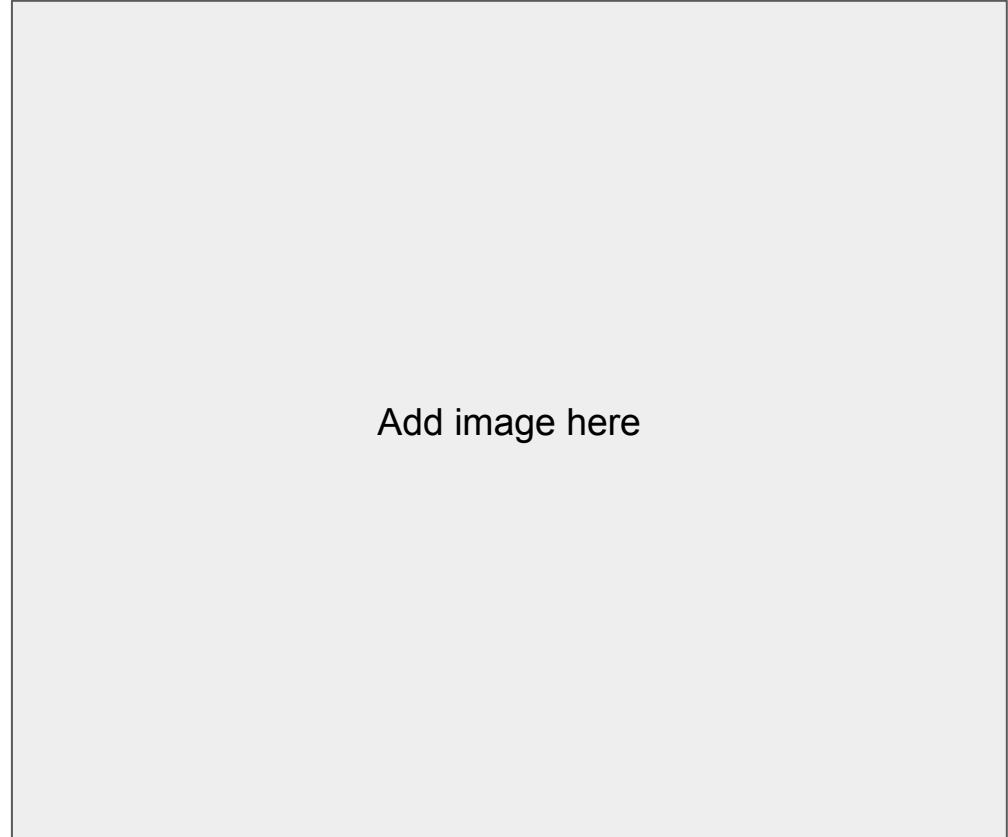
- A summary of the decisions made
- Warnings and other important information about the configuration of the VM are displayed
- Choose to automatically power on the VM after creation

The screenshot shows the Red Hat OpenShift Container Platform interface. On the left is a sidebar with navigation links: Home, Operators, Workloads (selected), Networking, Storage, Builds, Monitoring, Compute, User Management, and Administration. The main area is titled 'Create Virtual Machine' and has a sub-header 'Review and confirm your settings'. It lists six steps: 1 General, 2 Networking, 3 Storage, 4 Advanced, 5 Review (highlighted), and 6 Result. The 'General' section shows the VM name as 'rhel02', description as 'No description', source as 'URL', operating system as 'Red Hat Enterprise Linux 8.0 or higher', flavor as 'Small: 1 vCPU, 2 GiB Memory', and workload profile as 'desktop'. The 'Networking' section shows one interface named 'nic-0' with model 'VirtIO' and network 'Pod Networking'. The 'Storage' section contains a warning message: 'Some disks do not have a storage class defined. Default storage class managed-nfs-storage will be used.' It lists a disk named 'rootdisk' with source 'URL', size '10 GiB', interface 'VirtIO', storage class 'managed-nfs-storage', access mode 'Single User (RWO)', and volume mode 'Filesystem'. The 'Advanced' section shows 'Cloud Init' as 'Not Enabled' and a checkbox for 'Start virtual machine on creation' which is unchecked. At the bottom are 'Create Virtual Machine', 'Back', and 'Cancel' buttons.

# Import VMs

# Virtual Machine Import

- Wizard supports importing from VMware or Red Hat Virtualization
  - Single-VM workflow
- VMware import uses VDDK to expedite the disk import process
  - User is responsible for downloading the VDDK from VMware and adding it to a container image
- Credentials stored as Secrets
- **ResourceMapping** CRD configures default source -> destination storage and network associations

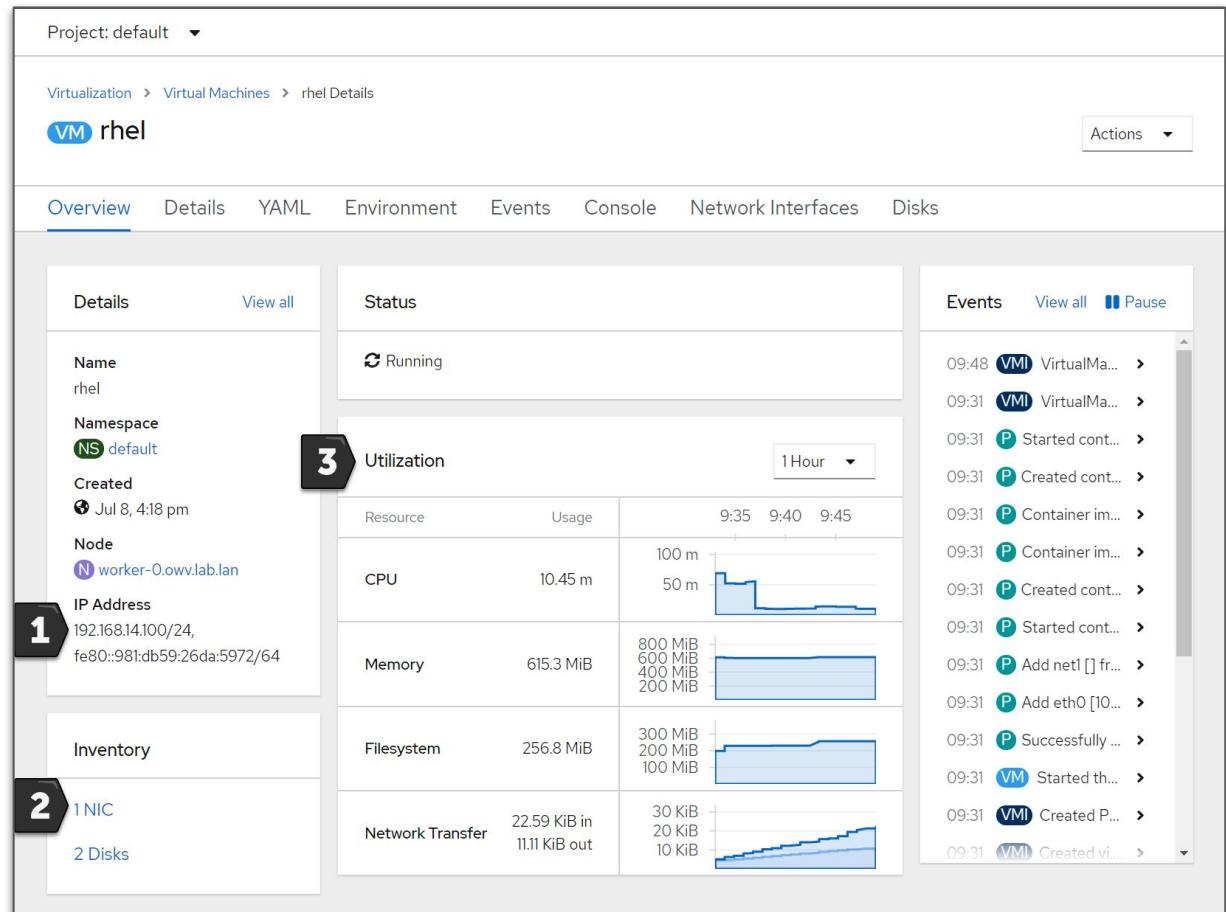


Add image here

# View / manage VMs

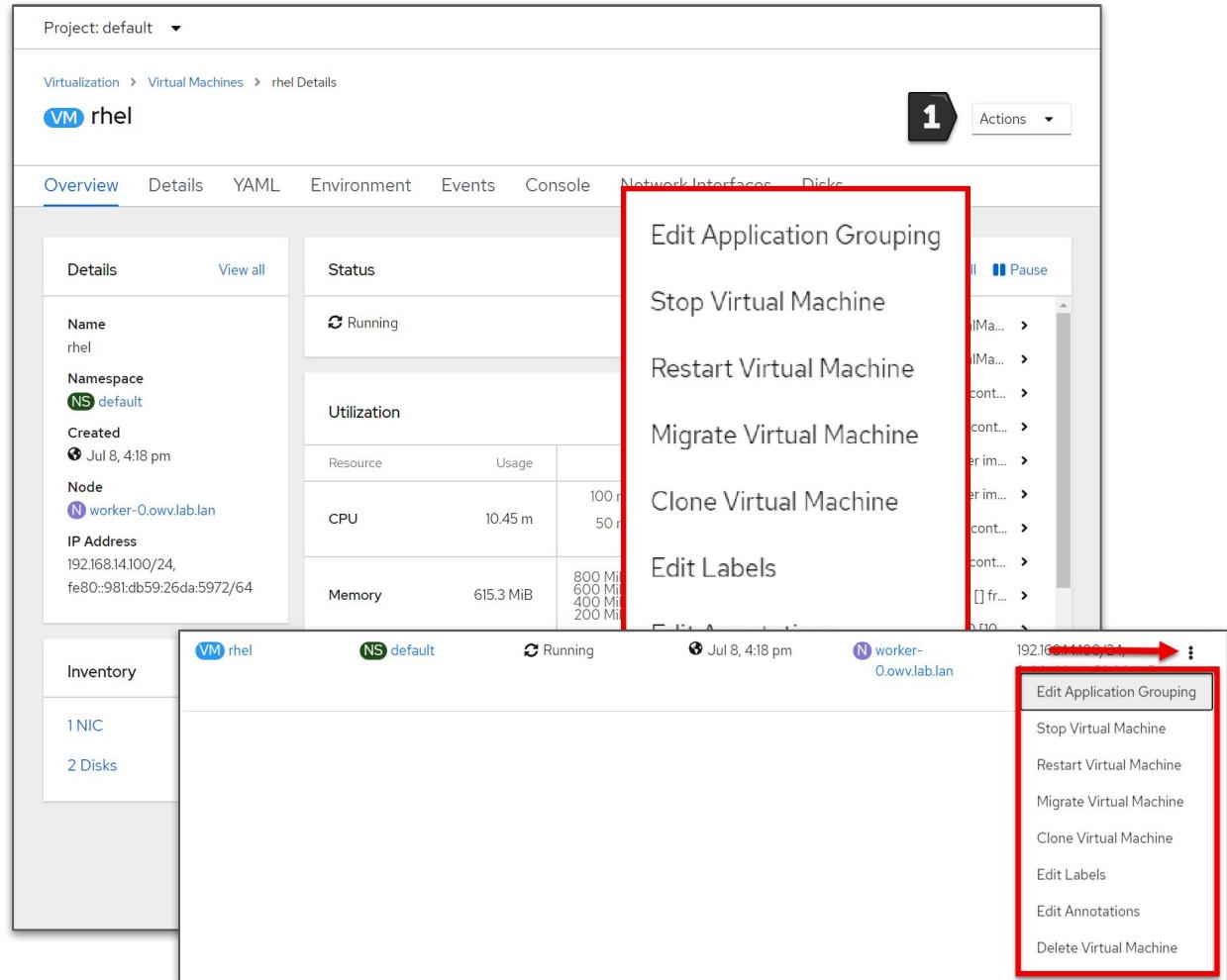
# Virtual Machine – Overview

- General overview about the virtual machine
- Information populated from guest when integrations are available
  - IP address
- Inventory quickly shows configured hardware with access to view/manage
- Utilization reporting for CPU, RAM, disk, and network



# Virtual Machine - Actions

- Actions menu allows quick access to common VM tasks
  - Start/stop/restart
  - Live migration
  - Clone
  - Edit application group, labels, and annotations
  - Delete
- Accessible from all tabs of VM details screen and the VM list



# Virtual Machine - Details

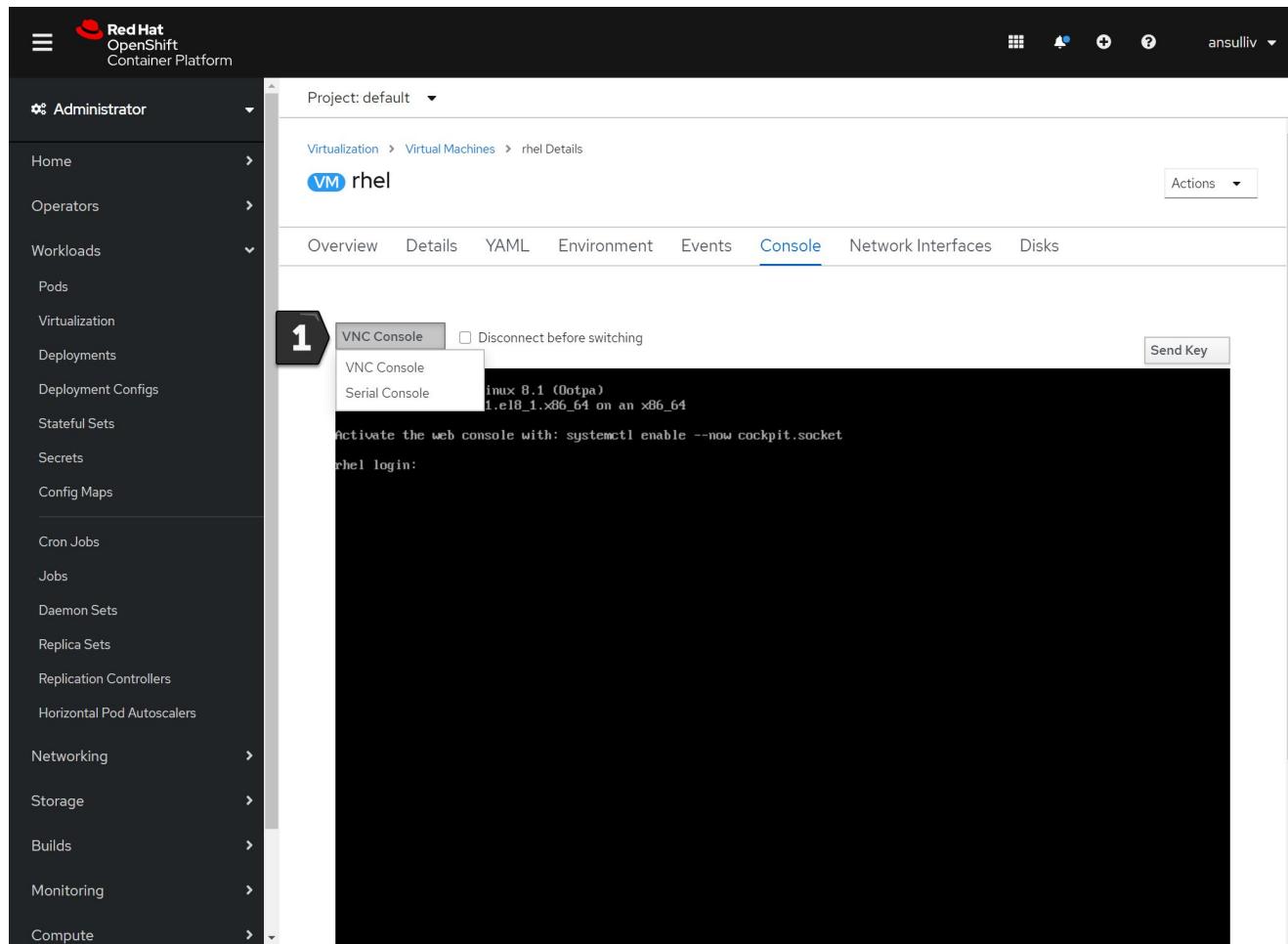
- Details about the virtual machine
  - Labels, annotations
  - Configured OS
  - Template used, if any
  - Configured boot order
  - Associated workload profile
  - Flavor
- Additional details about scheduling
  - Node selector, tolerations, (anti)affinity rules
- Services configured for the VM

The screenshot shows the Red Hat OpenShift Container Platform interface for viewing a Virtual Machine (VM) named 'rhel' in the 'default' project. The interface is dark-themed.

- 1 Labels:** Shows labels like app=rhel, flavor=template.kubevirt.io/small=true, os=template.kubevirt.io/rhel8.2=true, vm.kubevirt.io/template=rhel8-desktop-small-v0.10.0, vm.kubevirt.io/template.namespace=openshift, vm.kubevirt.io/template.revision=1, and vm.kubevirt.io/template.version=v0.11.2.
- 2 Operating System:** Red Hat Enterprise Linux 8.0 or higher.
- 3 Template:** Not available.
- 4 Boot Order:** 1.rootdisk (Disk).
- 5 Workload Profile:** desktop.
- 6 Flavor:** Small:1 vCPU, 2 GiB Memory. Dedicated Resources: No Dedicated resources applied.
- 7 Scheduling and resources requirements:** Node Selector: No selector. Tolerations: No Toleration rules. Affinity Rules: No Affinity rules.
- 8 Services:** No Services Found.

# Virtual Machine - Console

- Browser-based access to the serial and graphical console of the virtual machine
- Access the console using native OS tools, e.g. `virt-viewer`, using the `virtctl` CLI command
  - `virtctl console vmname`
  - `virtctl vnc vmname`



# Virtual Machine - Disks and NICs

- Add, edit, and remove NICs and disks for non-running virtual machines

The screenshot shows two side-by-side views of the Red Hat OpenShift Container Platform web interface, both for a virtual machine named 'rhel' in the 'default' project.

**Top View (Network Interfaces):**

- Header: Red Hat OpenShift Container Platform, Project: default, VM: rhel, Actions dropdown.
- Breadcrumbs: Virtualization > Virtual Machines > rhel Details.
- Navigation tabs: Overview, Details, YAML, Environment, Events, Console, **Network Interfaces**, Disks.
- Buttons: Add Network Interface.
- Table headers: Name, Model, Network, Type, MAC Address.
- Data row: nic-0, VirtIO, host-br1, bridge, -.

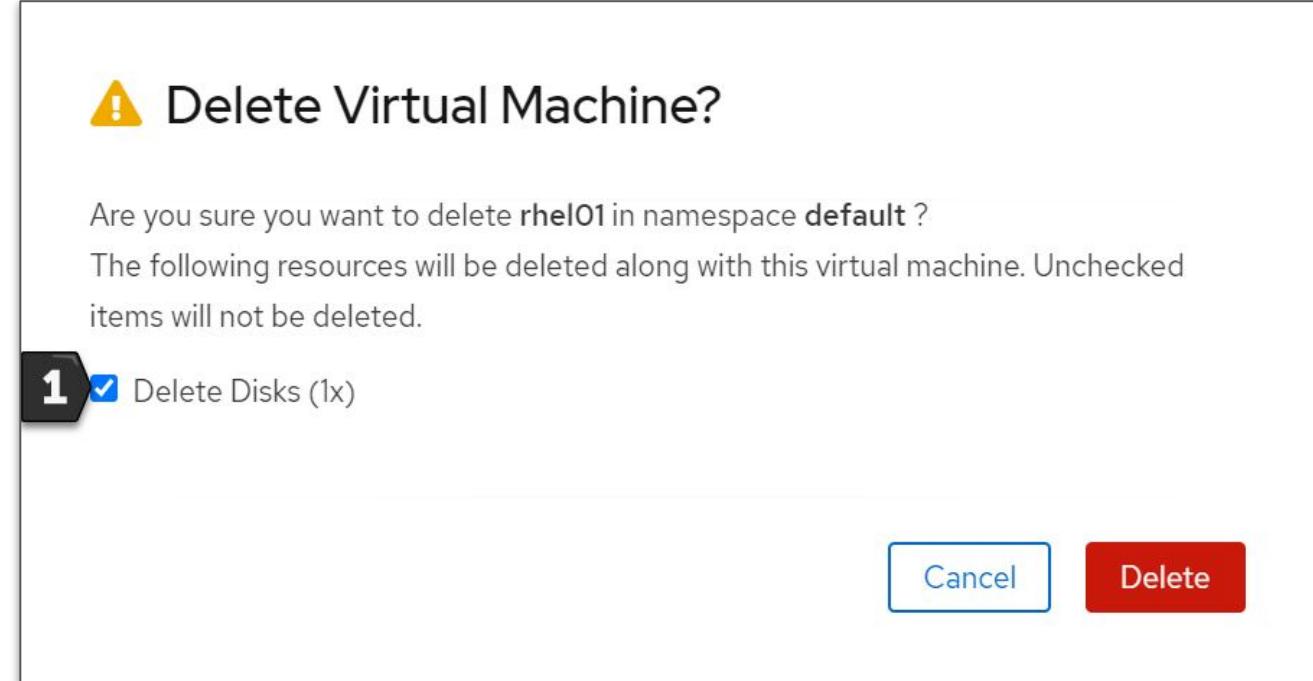
**Bottom View (Disks):**

- Header: Red Hat OpenShift Container Platform, Project: default, VM: rhel, Actions dropdown.
- Breadcrumbs: Virtualization > Virtual Machines > rhel Details.
- Navigation tabs: Overview, Details, YAML, Environment, Events, Console, Network Interfaces, **Disks**.
- Buttons: Add Disk, Filter, Search by name... input field.
- Table headers: Name, Source, Size, Interface, Storage Class.
- Data rows:
  - cloudinitdisk, Other, -, VirtIO, -.
  - rootdisk, URL, 20 GiB, VirtIO, managed-nfs-storage.

# Destroy VMs

# Destroying a Virtual Machine

- Deleting a VM removes the VM definition
  - Optionally delete PVC-backed disks associated with the VM
- Running VMs are terminated first
- Other associated resources, e.g. Services, are not affected



# Metrics

# Overview Virtual Machine metrics

- Summary metrics for 1, 6, and 24 hour periods are quickly viewable from the VM overview page
- Clicking a graph will display it enlarged in the metrics UI

The image shows two screenshots illustrating the monitoring of a virtual machine. The top screenshot is the 'Virtual Machines' details page for a VM named 'rhel' in the 'default' project. It displays various tabs like Overview, Details, YAML, Environment, Events, Console, Network Interfaces, and Disks. The 'Overview' tab is selected, showing the VM's status as 'Running'. Below the status, there's a 'Utilization' section with four charts: CPU, Memory, Filesystem, and Network Transfer. A red box highlights the CPU chart, which shows usage over a 1-hour period. The bottom screenshot is the 'Metrics' UI, specifically the Prometheus UI, showing a line graph of CPU usage over a 30m interval. The graph has a Y-axis ranging from 0 to 0.035 and an X-axis from 13:05 to 13:30. Below the graph, a table shows the query used: `pod:container_cpu_usage:sum{pod='virt-launcher-rhel-24wbf'}`. The table includes columns for Name, namespace, pod, prometheus, and Value, with one row showing the result for the virt-launcher-rhel-24wbf pod.

Project: default ▾

Virtualization > Virtual Machines > rhel Details

VM rhel

Actions ▾

Overview Details YAML Environment Events Console Network Interfaces Disks

Details View all

Name rhel

Status

Running

Events View all ▪ Pause

There are no recent events.

Utilization 1Hour ▾

Resource Usage 12:30 13:00

Resource	Usage	12:30	13:00
CPU	36.86 m	80 m	60 m
Memory	633 MiB	800 MiB	600 MiB
Filesystem	4.88 GiB	6 GiB	4 GiB
Network Transfer	228.8 KiB in 24.63 KiB out	300 KiB	200 KiB

Metrics Prometheus UI ▾

Actions ▾

Hide Graph

30m ▾ Reset Zoom

Insert Metric at Cursor ▾ Add Query Run Queries

pod:container\_cpu\_usage:sum{pod='virt-launcher-rhel-24wbf'}

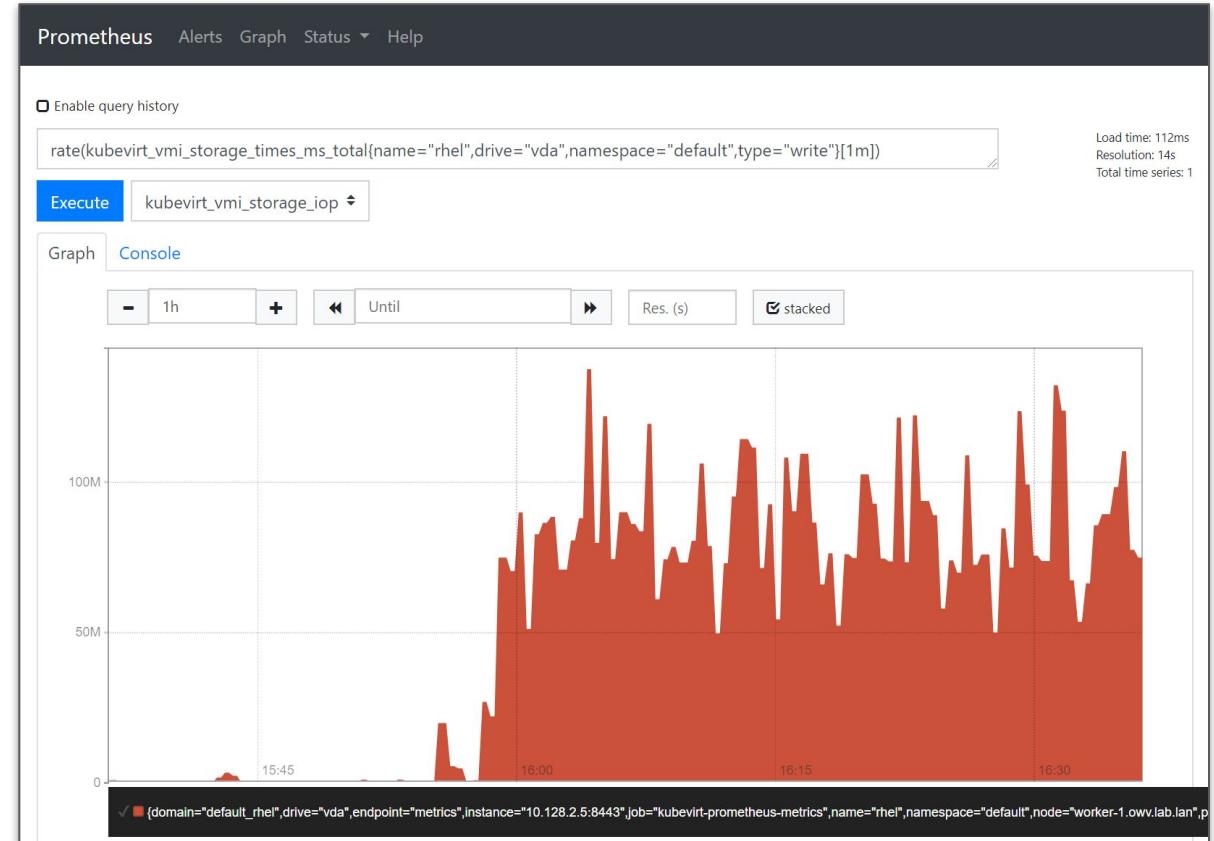
Name	namespace	pod	prometheus	Value
pod:container_cpu_usage:sum	default	virt-launcher-rhel-24wbf	openshift-monitoring/k8s	0.03392109586001908

1-1 of 1 ▾ << < 1 of 1 > >>

V00000000 Red Hat

# Detailed Virtual Machine metrics

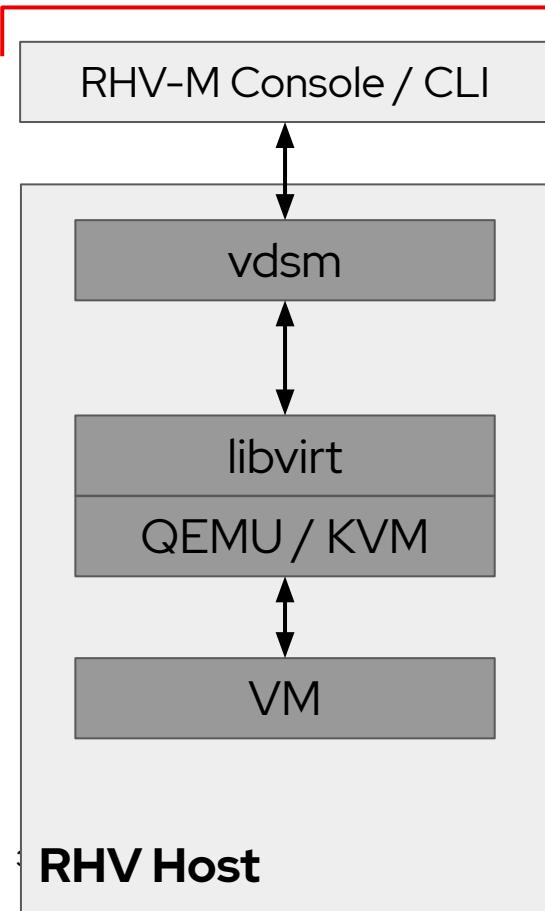
- Virtual machine, and VM pod, metrics are collected by the OpenShift metrics service
  - Available under the `kubevirt` namespace in Prometheus
- Available per-VM metrics include
  - Active memory
  - Active CPU time
  - Network in/out errors, packets, and bytes
  - Storage R/W IOPS, latency, and throughput
- VM metrics are for VMs, not for VM pods
  - Management overhead not included in output
  - Look at `virt-launcher` pod metrics for
- No preexisting Grafana dashboards



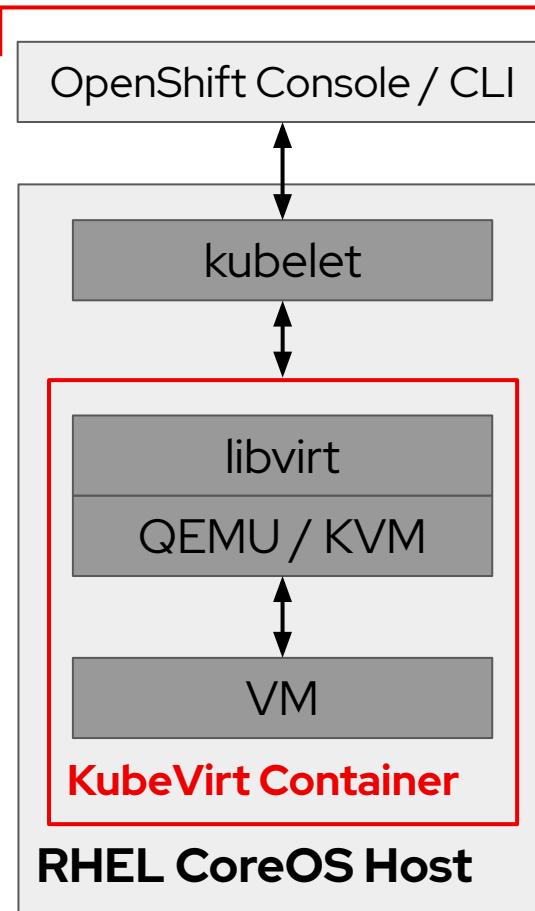
# Deeper into the technology

# Containerizing KVM

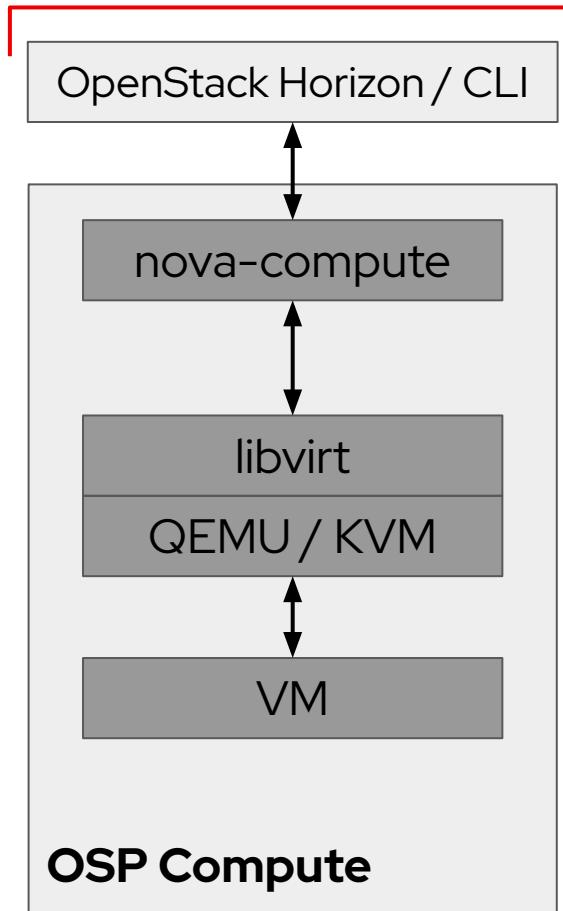
## Red Hat Virtualization



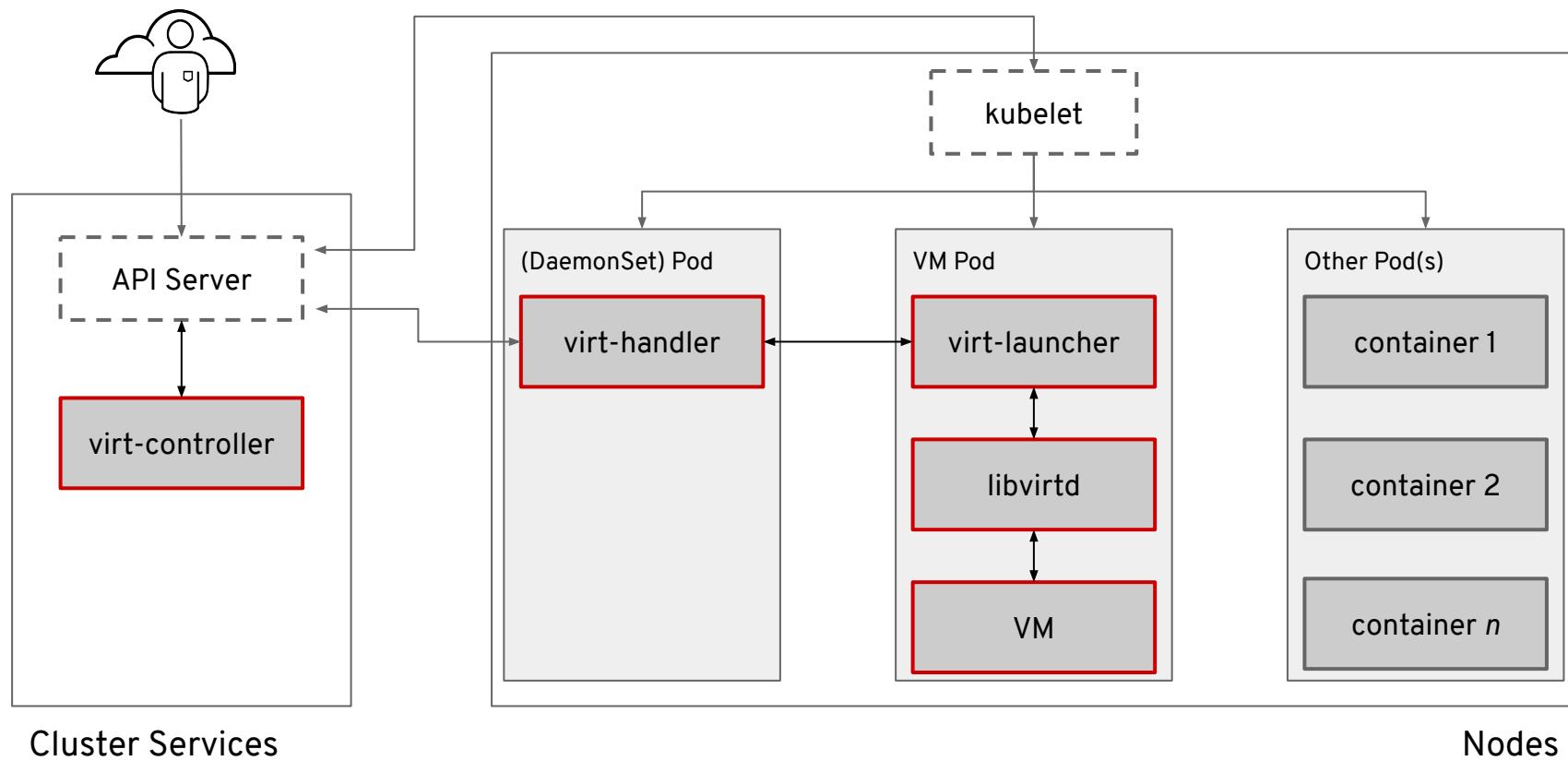
## OpenShift Virtualization



## Red Hat OpenStack Platform



# Architectural Overview



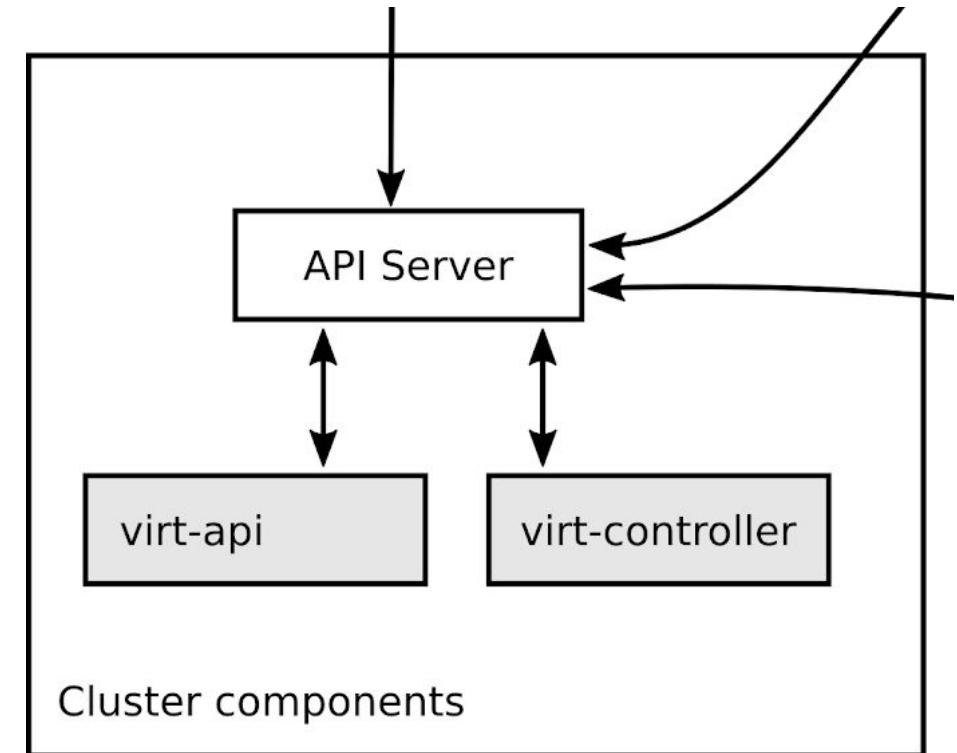
# Adding virtualization to the Kubernetes API

## CRD and aggregated API servers

- These are the ways to extend the Kubernetes API in order to support new entities
- For users, the new entities are indistinguishable from native resources

## Single API entry point for all workloads

- All workloads (containers, VMs, and serverless) are managed through a single API



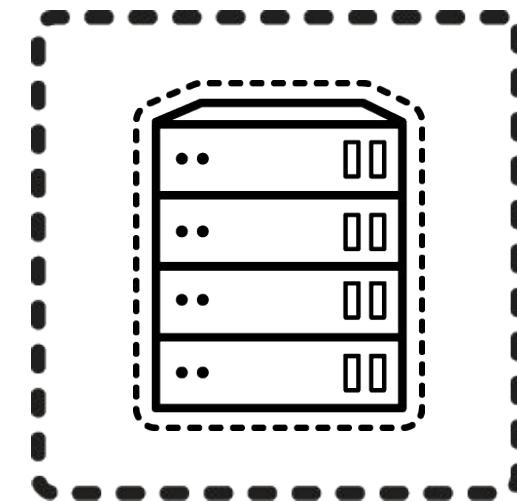
# OpenShift cluster architecture

- Everything everywhere - all 8 nodes are "workers"
  - Create the cluster with the control plane as schedulable
  - No dedicated infra nodes, no dedicated OCS nodes
  - Pros: no wasted resources
  - Cons: must pay for all cores of all nodes, extra effort should be taken to ensure pods have appropriate QoS to prevent resource contention exacerbating performance problems
- Shared control plane, dedicated + combined infra
  - Schedulable control plane
  - Dedicated infra nodes for registry, logging, metrics, and OCS
  - Pros: don't have to pay for infra node licenses
  - Cons: care needs to be taken to size nodes appropriately to not strand resources, e.g. "infra nodes are only 15% utilized, but we can't put workload on those nodes without paying for the OCP entitlements"
- Dedicated control plane, dedicated infra
  - Non-schedulable control plane
  - Dedicated infra nodes for registry, logging, metrics, and OCS
  - Pros: control plane resource isolation prevents contention from causing performance ripples
  - Cons: control plane nodes will almost certainly be dramatically under utilized, minimum 6 dedicated nodes (3 control plane, 3 infra)
- Shared control plane/worker/infra, dedicated OCS
  - Scheduleable control plane, no dedicated infra
  - Pros: isolates OCS for performance/scale reasons
  - Cons: same as above - care needs to be taken to protect control plane workloads, must pay for infra cores
- Dedicated everything
  - Dedicated control plane, infra, and OCS nodes
  - Pros: lots of isolation and protection for workloads
  - Cons: lots of potentially wasted resources (node right sizing is important!) and lots of nodes needed: 3 control plane, 3 OCS, 2 infra, + workers

# Virtual machines

# Containerized virtual machines

- Inherit many features and functions from Kubernetes
  - Scheduling, high availability, attach/detach resources
- Containerized virtual machines have the same characteristics as non-containerized
  - CPU, RAM, etc. limitations dictated by libvirt and QEMU
  - Linux and Windows guest operating systems
- Storage
  - Use Persistent Volumes Claims (PVCs) for VM disks
  - Containerized Data Importer (CDI) import VM images
- Network
  - Inherit pod network by default
  - Multus enables direct connection to external network



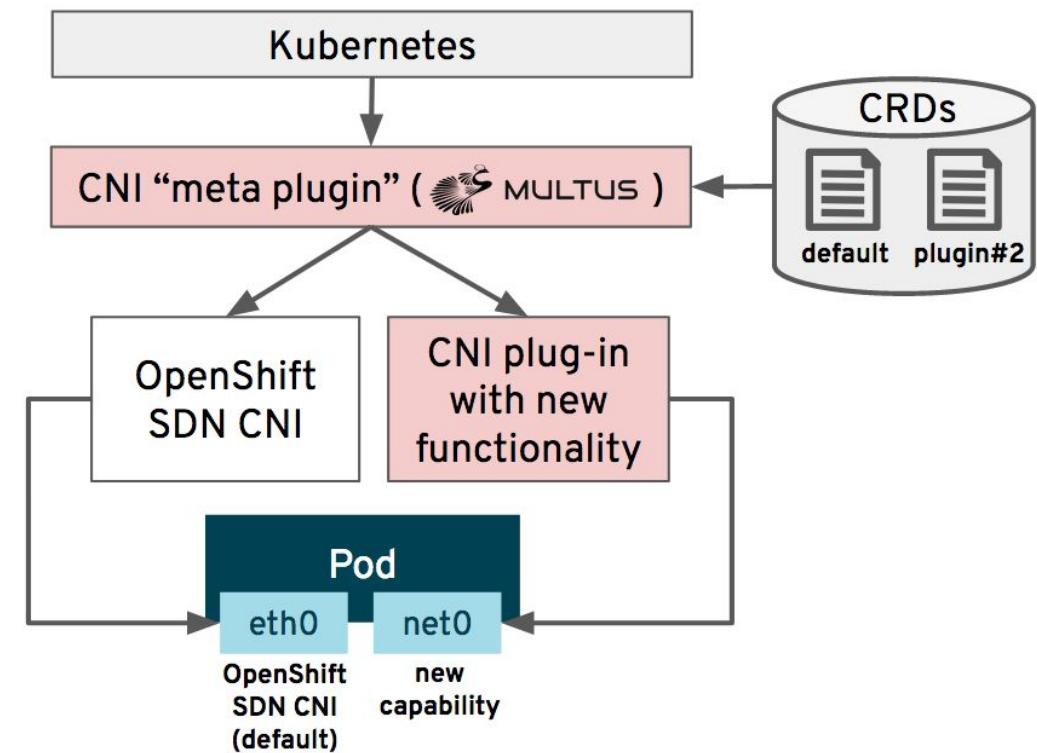
 **Red Hat**  
**OpenShift**

The Red Hat OpenShift logo consists of a red circular icon with a white 'O' shape inside, followed by the text "Red Hat" in a bold sans-serif font and "OpenShift" in a larger, bold sans-serif font below it.

# Network

# Virtual Machine Networking

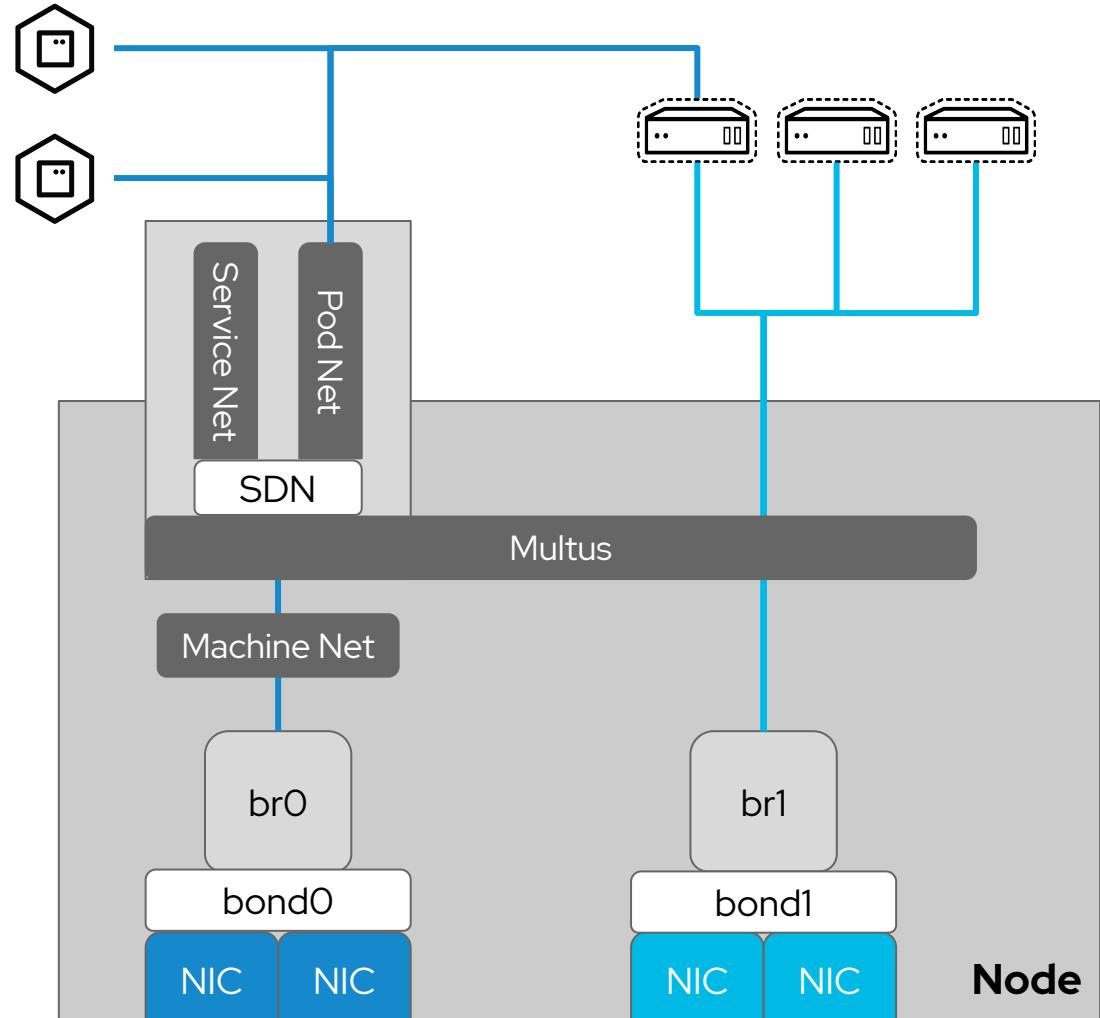
- Virtual machines optionally connect to the standard pod network
  - OpenShift SDN, OVNKubernetes, etc.
- Additional network interfaces accessible via Multus:
  - Bridge, SR-IOV
  - VLAN and other networks can be created using nmstate at the host level
- When using at least one interface on the default SDN, Service, Route, and Ingress configuration applies to VM pods the same as others



# Example host network configuration

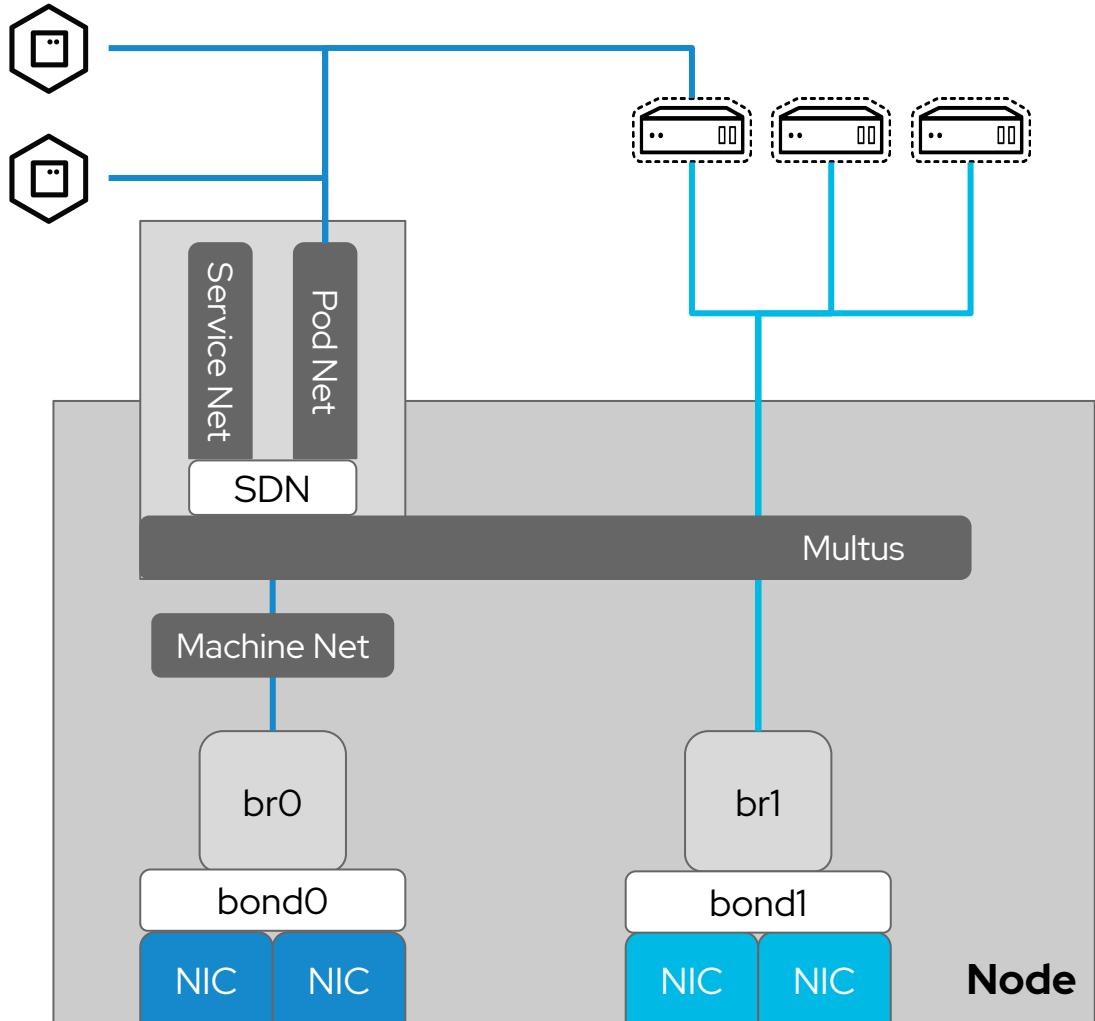
- Pod, service, and machine network are configured by OpenShift automatically
  - Use kernel parameters (dracut) for configuration at install
- Use `kubernetes-nmstate`, via the `nmstate` Operator, to configure additional host network interfaces
  - `bond1` and `br1` in the example to the right
- VM pods connect to one or more networks simultaneously

**The following slides show an example of how this setup is configured**

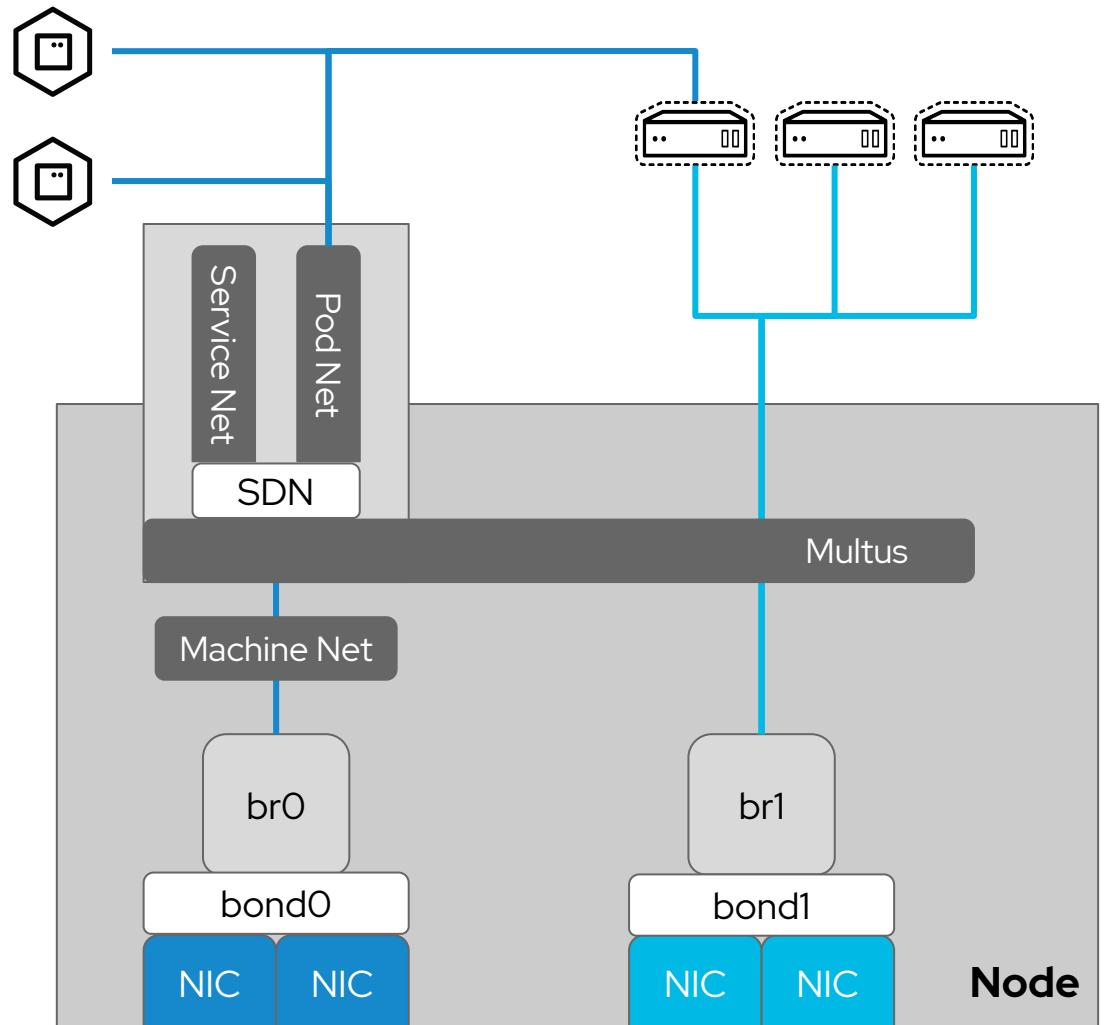


# Host bond configuration

- NodeNetworkConfigurationPolicy (NNCP)
  - Nmstate operator CRD
  - Configure host network using declarative language
- Applies to all nodes specified in the `nodeSelector`, including newly added nodes automatically
- Update or add new NNCPs for additional host configs



# Host bridge configuration



# Storage

# Virtual Machine Storage

- OpenShift Virtualization uses the Kubernetes PersistentVolume (PV) paradigm
- PVs can be backed by
  - In-tree iSCSI, NFS
  - CSI drivers
  - Local storage using host path provisioner
  - OpenShift Container Storage
- Dynamically or statically provisioned PVs
- RWX required for live migration
- Disks are attached using VirtIO or SCSI controllers
  - Connection order defined in the VM definition
- Boot order customized via VM definition

PersistentVolumeClaim Details

Name	rhel-rootdisk	Status	Bound
Namespace	NS default	Capacity	20Gi
Labels	app=containerized-data-importer	Access Modes	ReadWriteMany
Annotations	12 Annotations	Volume Mode	Filesystem
Label Selector	No selector	Storage Class	SC managed-nfs-storage
Created At	Jul 8, 4:18 pm	Persistent Volume	PV pvc-alaac411-2e46-495a-897e-cf3bc2442199
Owner	DV rhel-rootdisk		

# DataVolumes

- VM disks can be imported from multiple sources using DataVolumes, e.g. an HTTP(S) or S3 URL for a QCOW2 or raw disk image, optionally compressed
- DataVolumes are created view explicit object definition or as a part of the VM definition
- DataVolumes use the `ContainerizedDataImporter` to connect, download, and prepare the image for OpenShift Virtualization
- DataVolumes create PVCs based on defaults defined in the `kubevirt-storage-class-defaults` ConfigMap

```
1  dataVolumeTemplates:  
2    - apiVersion: cdi.kubevirt.io/v1alpha1  
3      kind: DataVolume  
4      metadata:  
5        creationTimestamp: null  
6        name: vm-rootdisk  
7      spec:  
8        pvc:  
9          accessModes:  
10            - ReadWriteMany  
11          resources:  
12            requests:  
13              storage: 20Gi  
14          storageClassName: my-storage-class  
15          volumeMode: Filesystem  
16        source:  
17          http:  
18            url: 'http://web.server/disk-image.qcow2'
```

# Comparing with traditional virtualization platforms

# Live Migration

- Live migration moves a virtual machine from one node to another in the OpenShift cluster
- Can be triggered via GUI, CLI, API, or automatically
- RWX storage is required, cannot use bridge connection to pod network
- Live migration is cancellable by deleting the API object
- Default maximum of five (5) simultaneous live migrations
  - Maximum of two (2) outbound migrations per node, 64MiB/s throughput each

Migration Reason	vSphere	RHV	OpenShift Virtualization
Resource contention	DRS	Cluster policy	Pod eviction policy, pod descheduler
Node maintenance	Maintenance mode	Maintenance mode	Maintenance mode, node drain

# VM scheduling

- VM scheduling follows pod scheduling rules
  - Node selectors
  - Taints / tolerations
  - Pod and node affinity / anti-affinity
- Kubernetes scheduler takes into account many additional factors
  - Resource load balancing - requests and reservations
  - CPU pinning, NUMA
  - Large / Huge page support for VM memory
- Resources are managed by Kubernetes
  - CPU and RAM requests match VM requirements - Guaranteed QoS by default
  - K8s QoS policy determines scheduling priority: **BestEffort** class is evicted before **Burstable** class, which is evicted before **Guaranteed** class

# Node Resource Management

- VM density is determined by multiple factors controlled at the cluster, OpenShift Virtualization, pod, and VM levels
- Pod QoS policy
  - Burstable ( $\text{limit} > \text{request}$ ) allows more overcommit, but may lead to more frequent migrations
  - Guaranteed ( $\text{limit} = \text{request}$ ) enables less overcommitment, but may have less physical resource utilization on the hosts
- Cluster Resource Override Operator provides global overcommit policy, can be customized per project for additional control
- VM pods request a small amount of additional memory, used for libvirt/QEMU overhead
  - Administrator can set this to be overcommitted
- Enable kernel same-page merging (KSM) by starting the daemon using a MachineConfig

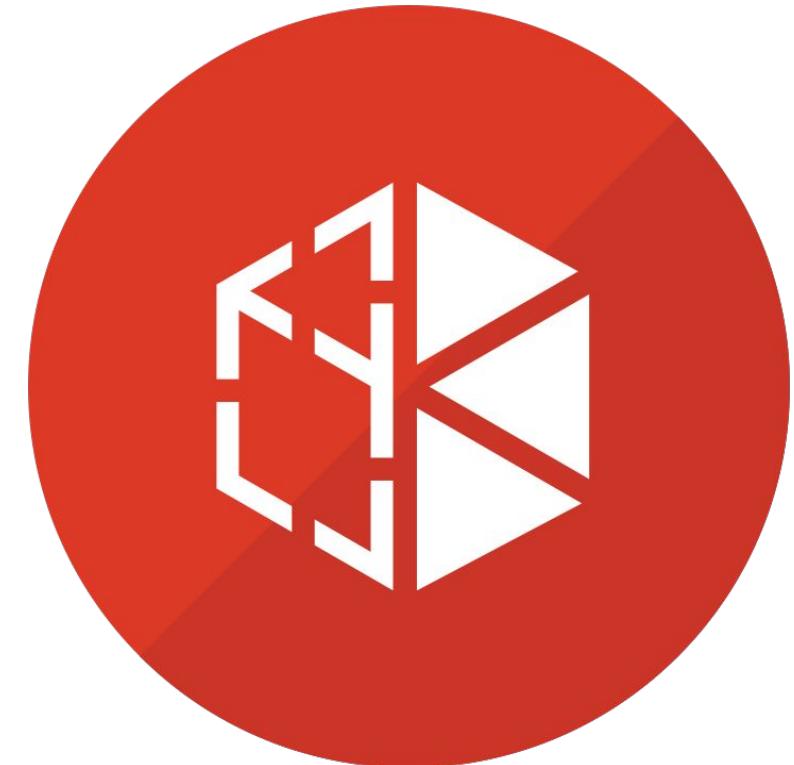
# High availability

- Node failure is detected by Kubernetes and results in the pods from the lost node being rescheduled to the surviving nodes
- VMs are not scheduled to nodes which have not had a heartbeat from `virt-handler`, regardless of Kubernetes node state
- Additional monitoring may trigger automated action to force stop the VM pods, resulting in rescheduling
  - May take up to 5 minutes for `virt-handler` and/or Kubernetes to detect failure
  - Liveness and Readiness probes may be configured for VM-hosted applications

# Runtime awareness

# Deploy and configure

- OpenShift Virtualization is deployed as an Operator utilizing multiple CRDs, ConfigMaps, etc. for primary configuration
- Many aspects are controlled by native Kubernetes functionality
  - Scheduling
  - Overcommitment
  - High availability
- Utilize standard Kubernetes / OpenShift practices for applying and managing configuration



# Compute configuration

- VM nodes should be physical with CPU virtualization technology enabled in the BIOS
  - Nested virtualization *works*, but is not supported
  - Emulation *works*, but is not supported
- Node labeler detects CPU type and labels nodes for compatibility and scheduling
- Configure overcommitment using native OpenShift functionality – Cluster Resource Override Operator
  - Optionally, customize the default project so that non-VM pods are not overcommitted
  - Customize projects hosting VMs for overcommit policy
- Enable KSM using MachineConfig, ballooning is not supported
- Apply Quota and LimitRange controls to projects with VMs to manage resource consumption

# Network configuration

- Apply traditional network architecture decision framework to OpenShift Virtualization
  - Resiliency, isolation, throughput, etc. determined by combination of application, management, storage, migration, and console traffic
  - Most clusters are not VM only, include non-VM traffic when planning
- Node interface on the **MachineNetwork** is used for “primary” communication, including SDN
  - This interface should be both resilient and high throughput
  - Used for migration and console traffic
  - Configure this interface at install time using kernel parameters, reinstall node if configuration changes
- Additional interfaces, whether single or bonded, may be used for traffic isolation, e.g. storage and VM traffic
  - Configure using nmstate Operator, apply configuration to nodes using selectors on NNCP

# Storage configuration

- Local storage may be utilized via the Host Path Provisioner
  - Local-only, non-shared storage means no live migration
- Create shared storage from local resources using OpenShift Container Storage
  - RWX file and block devices for live migration
- No preference for storage protocol, use what works best for the application(s)
- Storage backing PVs should provide adequate performance for VM workload
  - Monitor latency from within VM, monitor throughput from OpenShift
- For IP storage (NFS, iSCSI), consider using dedicated network interfaces
  - Will be used for all PVs, not just VM PVs
- Certified CSI drivers are recommended
  - No CSI snapshot integration
  - Non-certified work, but do not have same level of OpenShift testing

# Deploying a VM operating system

Creating virtual machines can be accomplished in multiple ways, each offering different options and capabilities

- Start by answering the question “Do I want to manage my VM like a container or a traditional VM?”
- Deploying the OS persistently, i.e. “I want to manage like a traditional VM”
  - Methods:
    - Import a disk with the OS already installed (e.g. cloud image) from a URL or S3 endpoint using a DataVolume, or via CLI using virtctl
    - Clone from an existing PVC or VM template
  - VM state will remain through reboots and, when using RWX PVCs, can be live migrated
- Deploying the OS non-persistently, i.e. “I want to manage like a container”
  - Methods:
    - Diskless, via PXE
    - Container image, from a registry
  - VM has no state, power off will result in disk reset. No live migration.
- Import disks deployed from a container image using CDI to make them persistent

# Deploying an application

Once the operating system is installed, the application can be deployed and configured several ways

- The application is pre-installed with the OS
  - This is helpful when deploying from container image or PXE as all components can be managed and treated like other container images
- The application is installed to a container image
  - Allows the application to be mounted to the VM using a secondary disk. Decouples OS and app lifecycle.  
When used with a VM that has a persistently deployed OS this breaks live migration
- The application is installed after OS is installed to a persistent disk
  - cloud-init - perform configuration operations on first boot, including OS customization and app deployment
  - SSH/Console - connect and administer the OS just like any other VM
  - Ansible or other automation - An extension of the SSH/console method, just automated

# Targeted Opportunities and Use Cases

# Targeted Opportunities and Use Cases - Developers

- Current, prospective OpenShift customers who have
  - Apps on containers and
  - Apps on VMs running RHEL and Windows guests
  - Developers with forward leaning and legacy app ownership
    - Value prop:
      - Standardize, accelerate app dev processes and delivery - NOW!
      - Modernize current apps running on VMs - NOW!
        - Convert to containers over time, or never
        - Migrate older Windows apps - NOW!
          - Same support matrix as RHV and RHOSP
      - Delivery “mixed” apps using containers, VMs - NOW!
    - Ask them:
      - What dev platforms, infrastructure are you using now?
      - How do you request and provision new VMs?
        - ***You can do that with OpenShift today.***

# Targeted Opportunities and Use Cases - Infra Owners, DevOps

- Current, prospective OpenShift customers who have
  - Apps on containers and
  - Apps on VMs running RHEL and Windows guests
  - Infra Owners, DevOps who support developers, container and VM management platforms
    - Value prop:
      - Simplify your data center
      - Standardize the services, offerings you provide
        - Security, stability of RH's shared KVM-based virtualization stack
  - Ask them:
    - What VM specific management platforms, infrastructure are you using now?
    - How do you service requests and provision new VMs?
      - Can this be automated or provided via self-service?
    - ***OpenShift can do that for you.***

# Targeted Workloads - supported in OpenShift 4.5 (1 of 2)

Workload <small>This list is not exhaustive; specific platforms, product names are provided as examples to guide discussions within target opportunities and use cases.</small>	RHEL Guests (x86 only)	Windows Guests (x86 only)	Supported
3rd party, pre-packaged apps on VMs	Yes	Yes	OpenShift 4.5
Custom apps running on VMs	Yes	Yes	OpenShift 4.5
IT Infra VMs - File Servers	Yes	Yes	OpenShift 4.5
IT Apps - MSFT Exchange Server		Yes	OpenShift 4.5
IT Apps - MSFT Sharepoint		Yes	OpenShift 4.5
IT Apps - JIRA	Yes	Yes	OpenShift 4.5
IT Apps - Wordpress	Yes	Yes	OpenShift 4.5
App Runtimes - JBoss/EAP	Yes	Yes	OpenShift 4.5
App Runtimes - WebSphere Application Server	Yes	Yes	OpenShift 4.5

v0000000



# Targeted Workloads - supported in OpenShift 4.5 (2 of 2)

Workload <small>This list is not exhaustive; specific platforms, product names are provided as examples to guide discussions within target opportunities and use cases.</small>	RHEL Guests (x86 only)	Windows Guests (x86 only)	Supported
App Runtimes - MSFT .net	Yes	Yes	OpenShift 4.5
App Runtimes - IIS		Yes	OpenShift 4.5
Databases - Oracle	Yes	Yes	OpenShift 4.5
Databases - MSFT SQL Server (non-clustered)	Yes	Yes	OpenShift 4.5
Databases - IBM DB2 LUW	Yes	Yes	OpenShift 4.5
Databases - MySQL (all derivatives, community and Enterprise editions)	Yes	Yes	OpenShift 4.5
Databases - MongoDB (all derivatives, community and Enterprise editions)	Yes	Yes	OpenShift 4.5

# Targeted Workloads - Roadmap

Workload <small>This list is not exhaustive; specific platforms, product names are provided as examples to guide discussions within target opportunities and use cases.</small>	RHEL Guests (x86 only)	Windows Guests (x86 only)	Supported
Big Data, Streaming, Collaborative Apps			Roadmap
Extreme Resource Requirements (SAP HANA)			Roadmap
Clustering and HA Requirements (Oracle RAC, MySQL Carrier Grade, Percona XtraDB, MSFT Cluster Server/Service)			Roadmap
Desktop Visualization, Technical Workstation (GPU enablement)			Roadmap

These workloads can be targeted as net new or to remain on RHV and then positioned for migration to OpenShift Virtualization when maturity allows.

# OpenShift Virtualization Roadmap

<b>Near Term</b> <small>(3-6 months)</small>		
CORE	CORE	CORE
NETWORK	NETWORK	NETWORK
STORAGE	STORAGE	STORAGE
<ul style="list-style-type: none"> <li>Streamline 'oc drain'</li> <li>Improve performance with Hyper-V enlightenments</li> <li>Improve golden image template workflow</li> <li>Sizing guidance for field</li> </ul>	<p><b>Virt gaps</b> - vCPU &amp; vNUMA pinning</p> <p><b>Security</b> - FIPS</p> <p><b>Developer</b></p> <ul style="list-style-type: none"> <li>Using VMs naturally in Developer Pipelines</li> </ul>	<ul style="list-style-type: none"> <li>OCP as on-prem infrastructure</li> <li><b>Cloud</b> - Bare Metal where supported by public cloud provider</li> <li><b>Mass migration</b> - VMw to OCP &amp; RHV to OCP</li> </ul>
<p><b>IPv6</b></p> <ul style="list-style-type: none"> <li>Dual stack</li> <li>Multicast</li> <li>IPv6 multi-address</li> <li>Enhance CNI test suite for VMs</li> </ul>	<ul style="list-style-type: none"> <li>Hotplug - NIC</li> </ul> <p><b>Ecosystem CNI certification</b></p> <ul style="list-style-type: none"> <li>Tigera - Calico</li> </ul>	<ul style="list-style-type: none"> <li>OVS HW offload</li> </ul> <p><b>Ecosystem CNI certification</b></p> <ul style="list-style-type: none"> <li>Nuage</li> <li>Juniper</li> </ul>
<ul style="list-style-type: none"> <li>Offline snapshots</li> <li>Hotplug disk PoC</li> <li>Performance and scale of VMs on OCS</li> <li>Enhance CNI test suite for VMs</li> </ul>	<ul style="list-style-type: none"> <li>Application consistent snapshots and cloning</li> <li>Improve default performance with io=native</li> </ul> <p><b>Backup and DR</b></p> <ul style="list-style-type: none"> <li>via OCS data protection</li> </ul>	<ul style="list-style-type: none"> <li><b>High Availability</b></li> </ul>

# Positioning, Selling the right Red Hat virtualization platform



**Red Hat**  
Virtualization



**Red Hat**  
OpenStack Platform



**Red Hat**  
OpenShift  
Virtualization

Understanding the customer problem to be solved	Traditional Virtualization	IaaS Platform Virtualization	Kubernetes Managed Virtualization
Main use cases	Virtualized workloads, apps running on-prem	Private cloud: NFV/telco, cloud native (w/ OpenShift), developer cloud, service provider, AI/ML/HPC, developer, edge computing	Mixed workloads running on containers, VMs
Target audience	Enterprises running VMs and those migrating VMs from VMw, not using or planning OCP, containers	Telcos, service providers, those already using OpenStack, enterprises using private cloud	Enterprises, service providers running VMs and those migrating VMs from VMw, using or planning OCP, containers.
Customer Use Cases	<a href="#"><u>Customer Presentation</u></a> <a href="#"><u>Customer Use Cases and References</u></a>	<a href="#"><u>Customer Presentation</u></a> <a href="#"><u>Customer Use Cases and References</u></a>	<a href="#"><u>Customer Presentation</u></a> Customer Use Cases and References
Technical Requirements	<a href="#"><u>Technical Specification for Compute, Network, Storage</u></a>	Technical Specification for Compute, Network, Storage	<a href="#"><u>Technical Specification for Compute, Network, Storage</u></a>
All Resources	<a href="#"><u>RHV on PnT Portal</u></a>	<a href="#"><u>RHOSP on PnT Portal</u></a>	<a href="#"><u>OpenShift on PnT Portal</u></a> 

# More information

- Documentation:
  - OpenShift Virtualization: <https://docs.openshift.com>
  - KubeVirt: <https://kubevirt.io>
- Demos and video resources: <http://demo.openshift.com>
- Labs and workshops: coming soon to RHPDS

# Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

 [linkedin.com/company/red-hat](https://www.linkedin.com/company/red-hat)

 [youtube.com/user/RedHatVideos](https://www.youtube.com/user/RedHatVideos)

 [facebook.com/redhatinc](https://www.facebook.com/redhatinc)

 [twitter.com/RedHat](https://twitter.com/RedHat)