Red Hat OpenShift

# OpenShift Virtualization

## Technical presentation

Alfred Bach
Principal Solution Architect
EMEA Partner Enablement

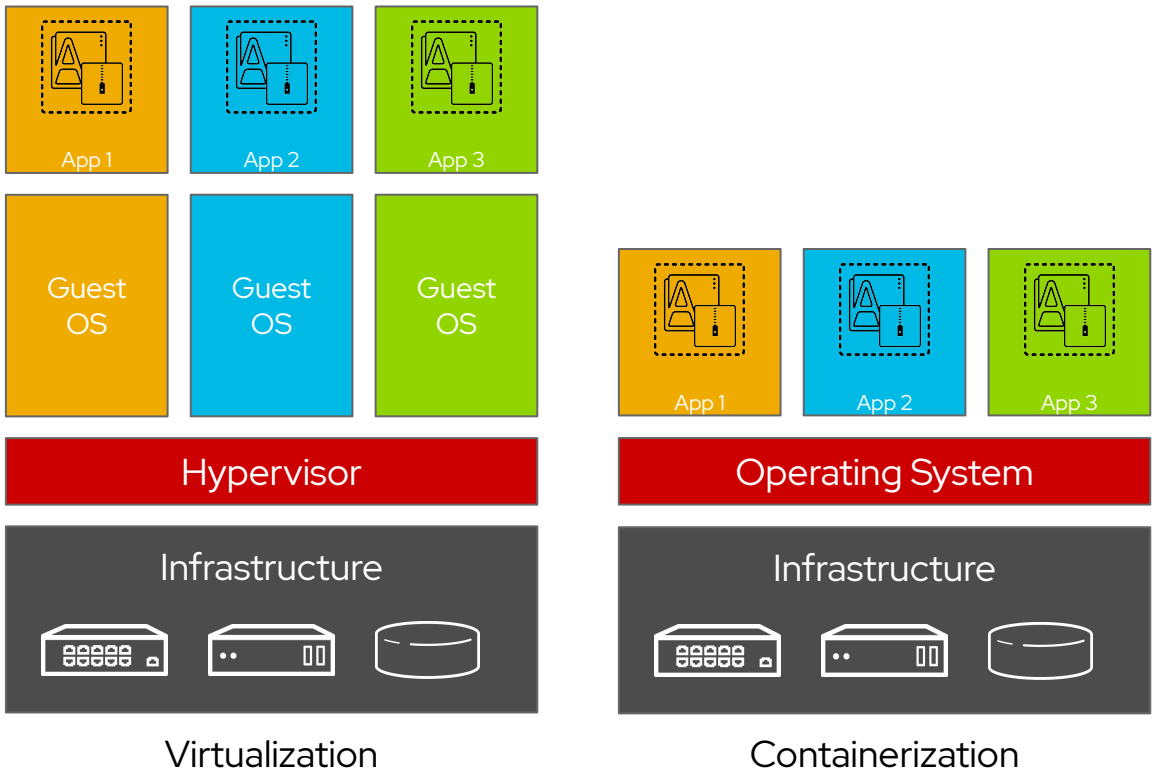


Red Hat

# What is OpenShift Virtualization?

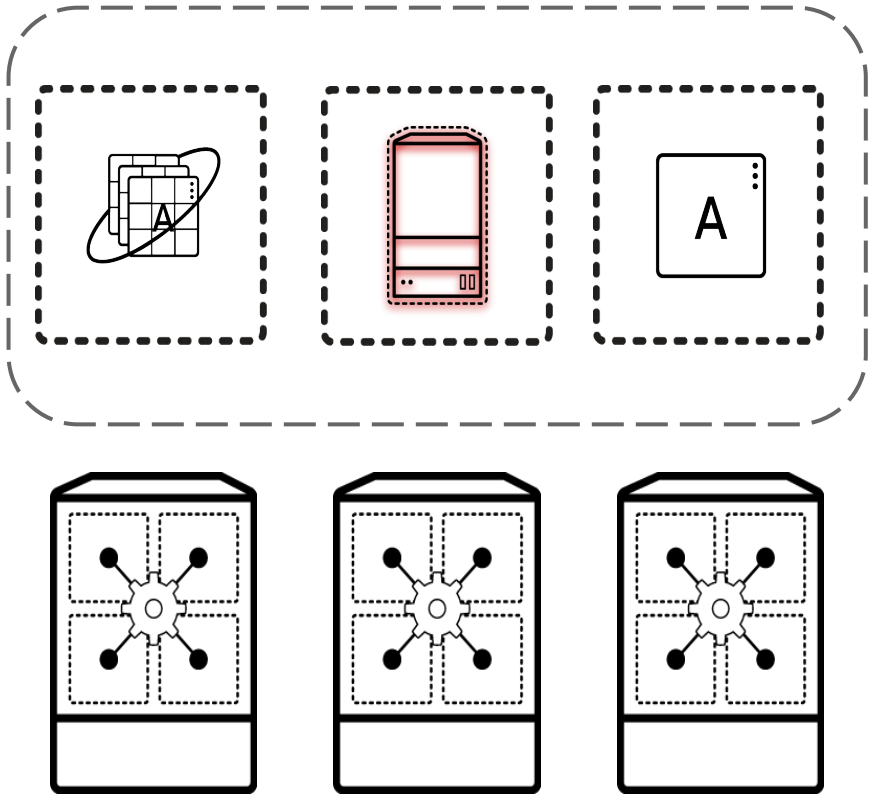
# Containers are not virtual machines

- 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



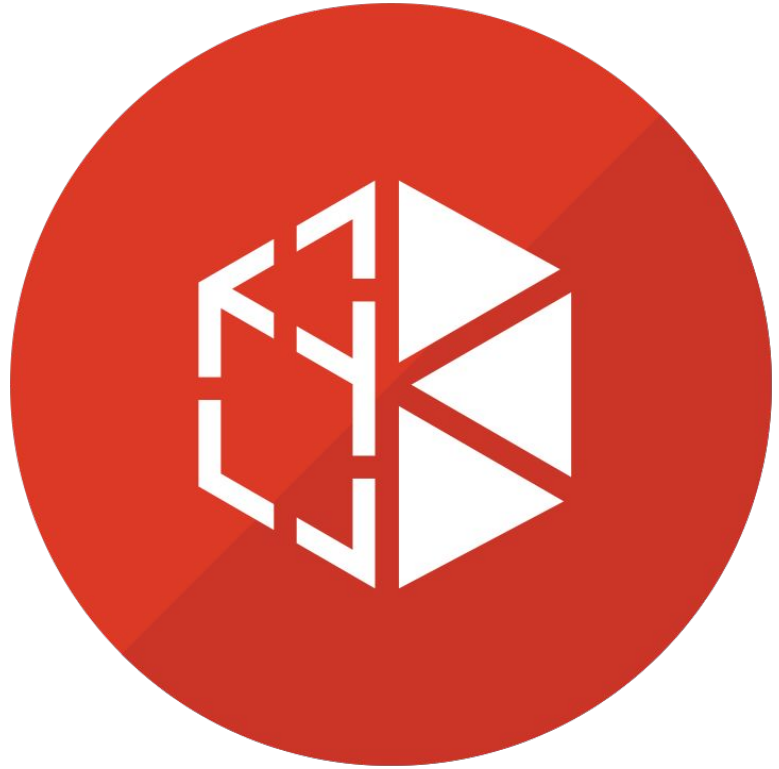
# Virtual machines can be put into containers

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



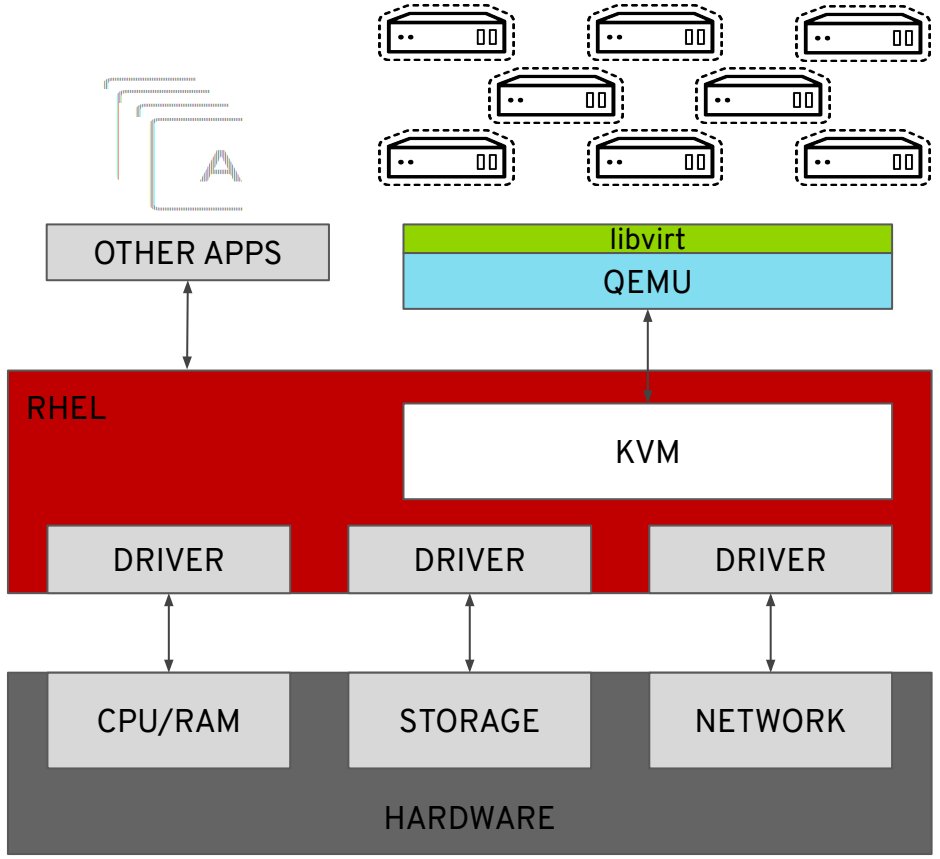
# OpenShift Virtualization

- 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

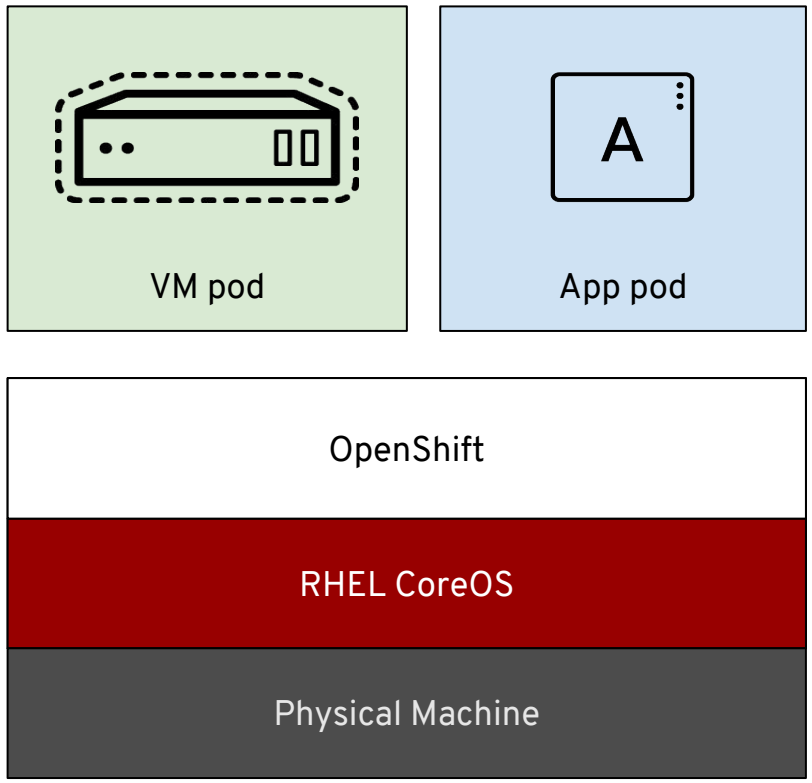
- OpenShift Virtualization uses KVM, the Linux kernel hypervisor
- KVM is a core component of the Red Hat Enterprise Linux kernel
  - KVM has 10+ years of production use: Red Hat Virtualization, Red Hat OpenStack Platform, and RHEL all leverage KVM, QEMU, and libvirt
- QEMU uses KVM to execute virtual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

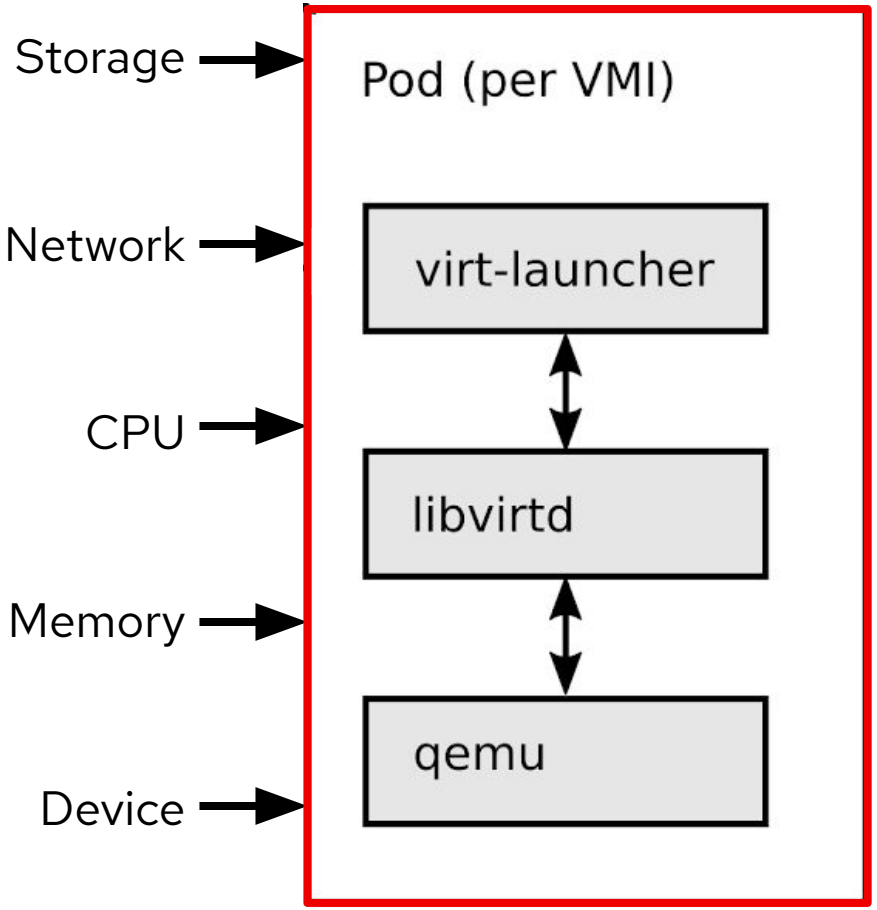


# Virtualization native to Kubernetes

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

```yaml
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**

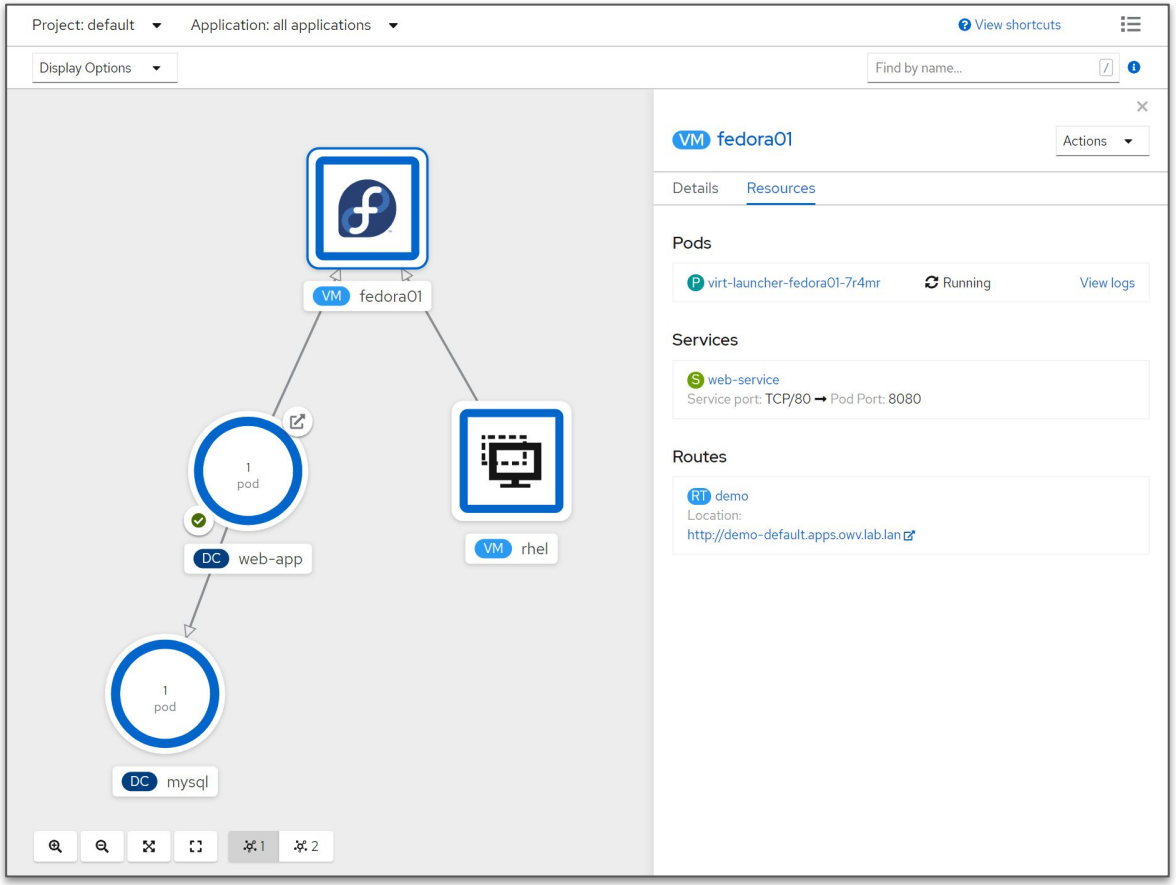
- libvirt and qemu from RHEL are mature, have high performance, provide stable abstractions, and have a minimal overhead.

**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





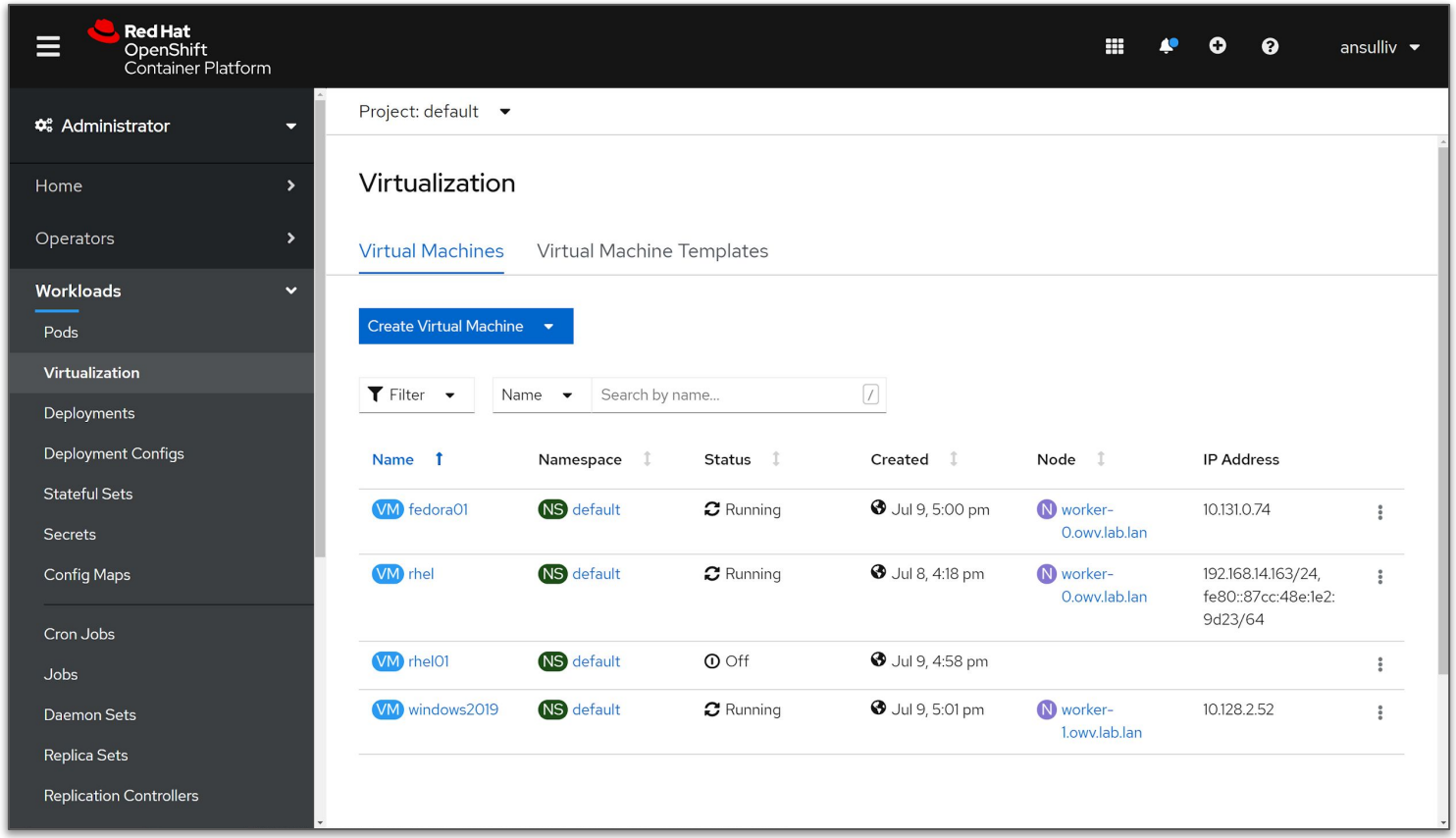



Red Hat

# 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





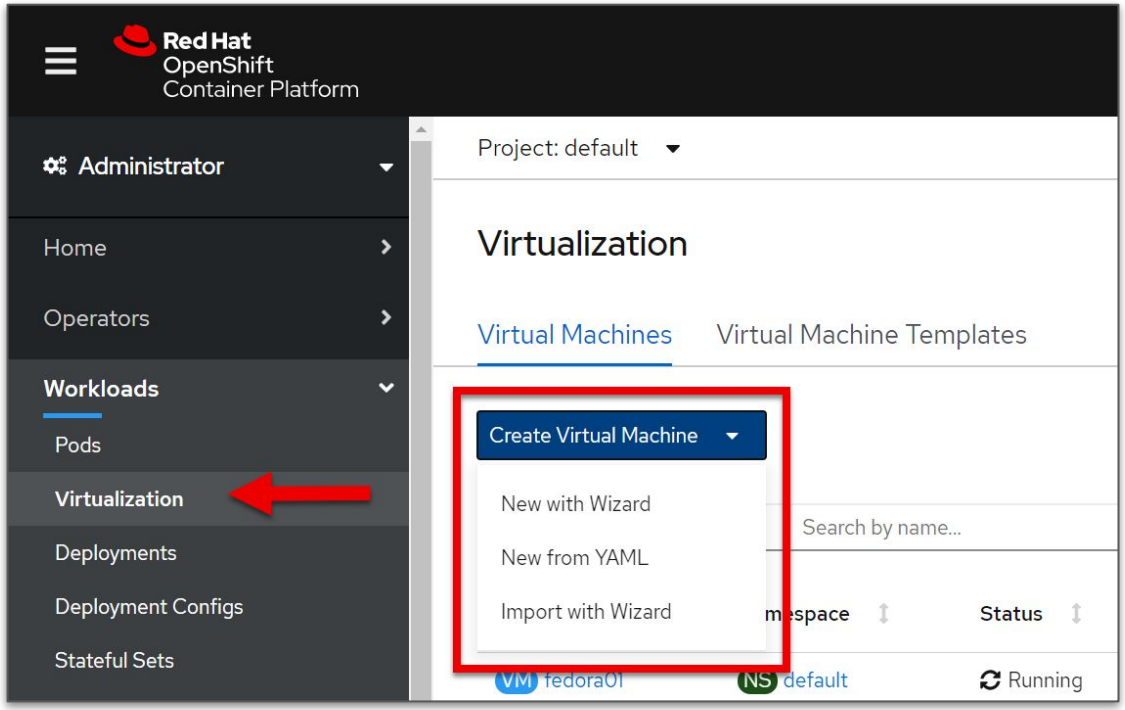



Red Hat

# 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



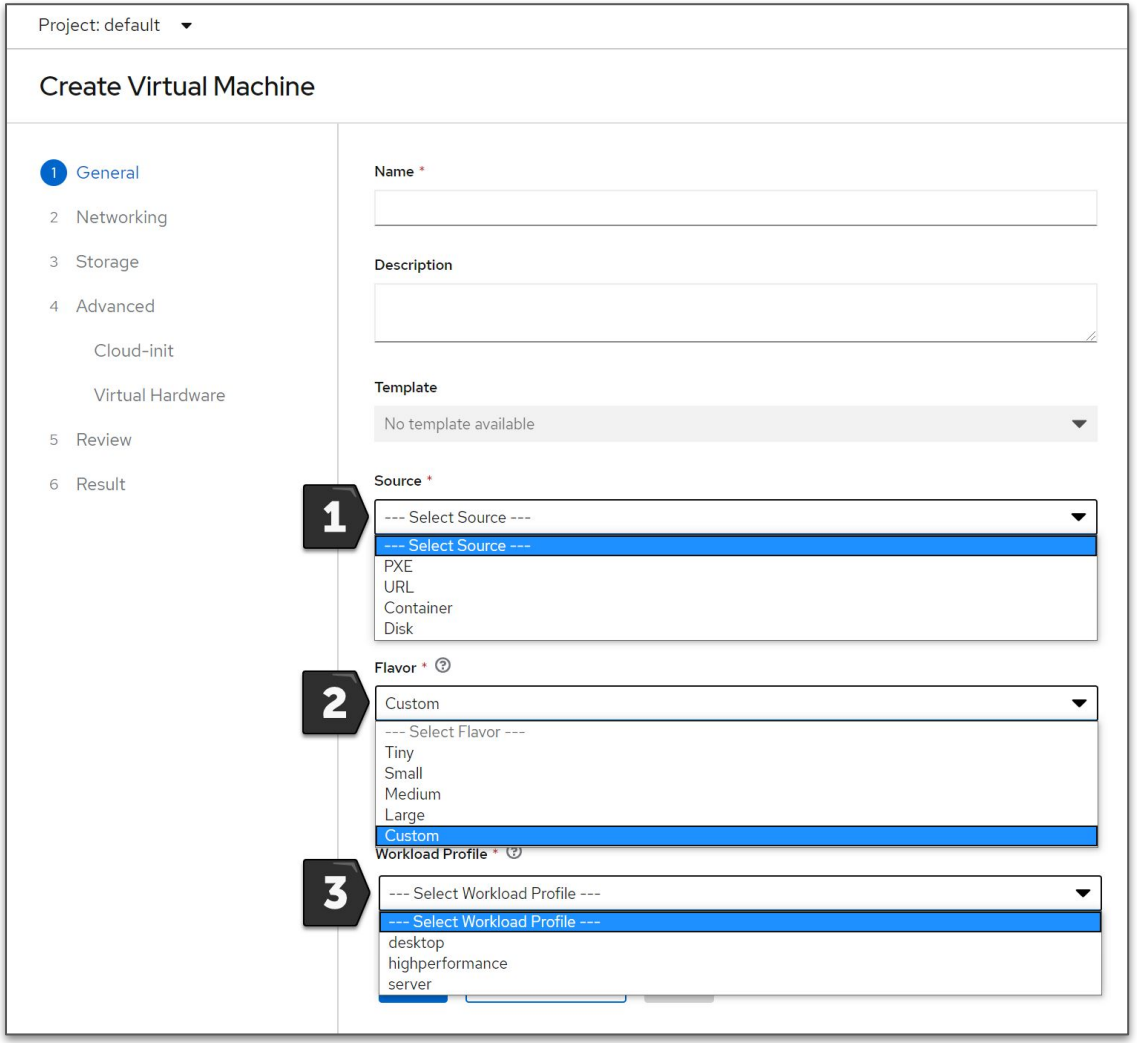





Red Hat

# 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



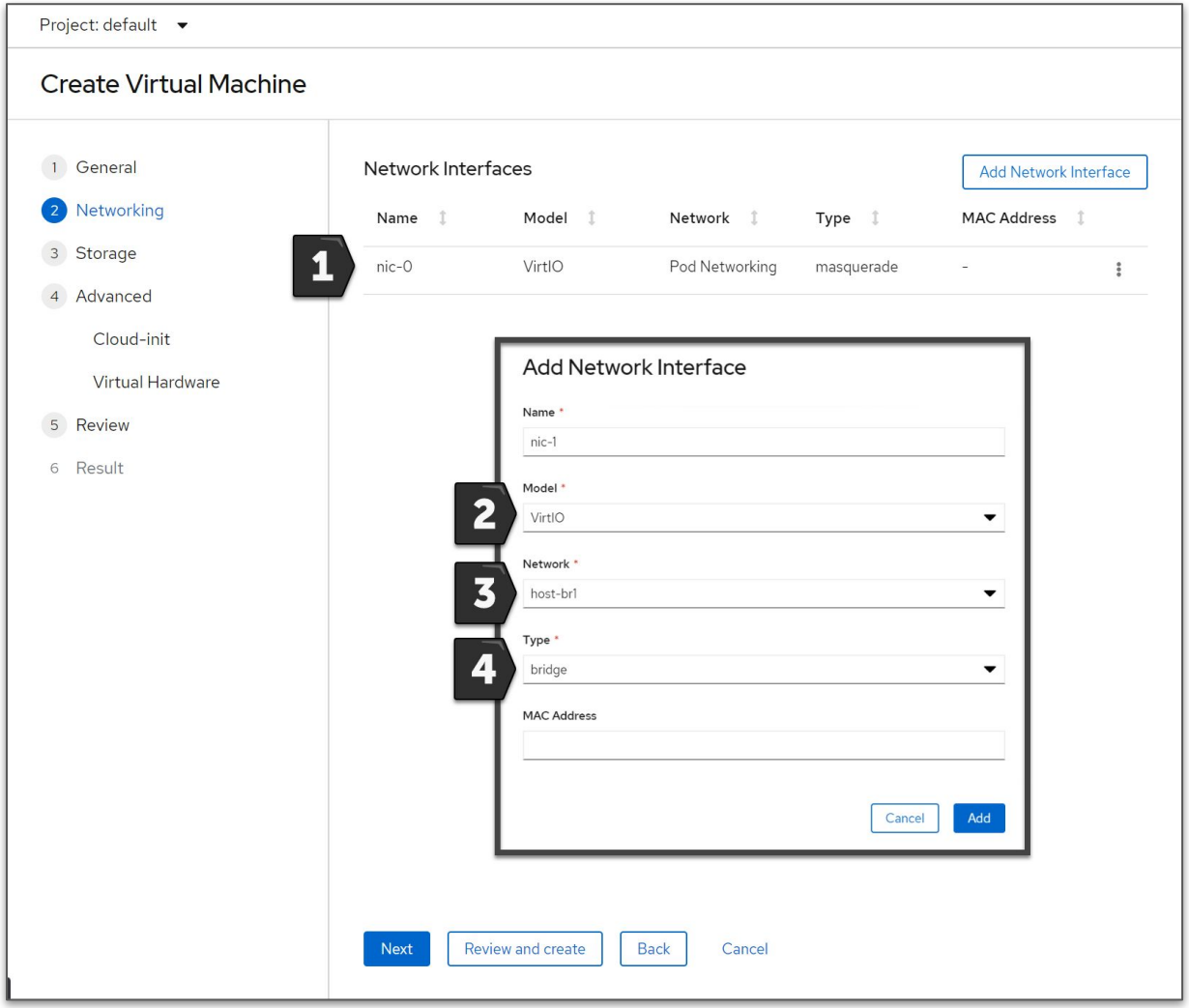





Red Hat

# 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



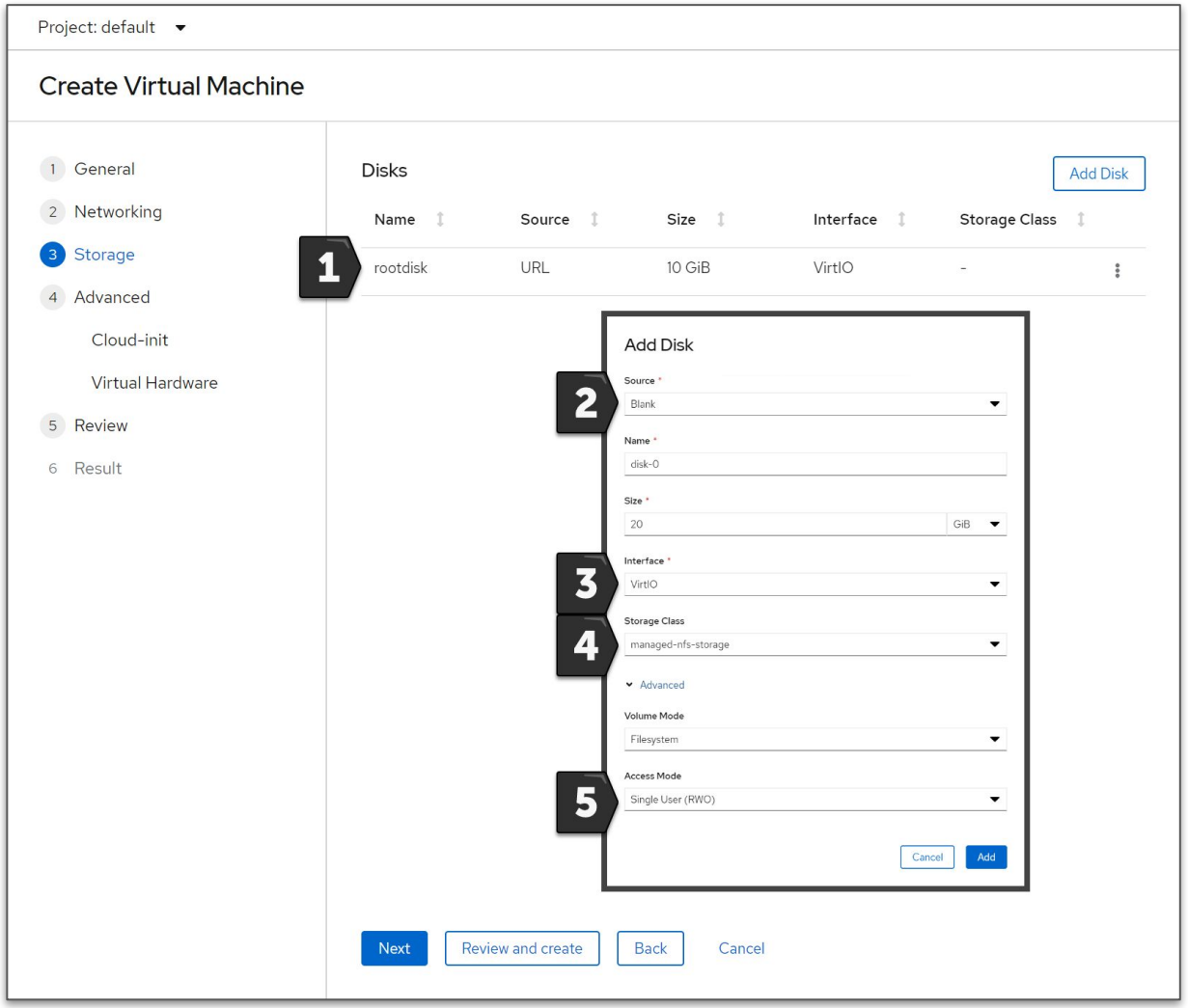





Red Hat

# 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



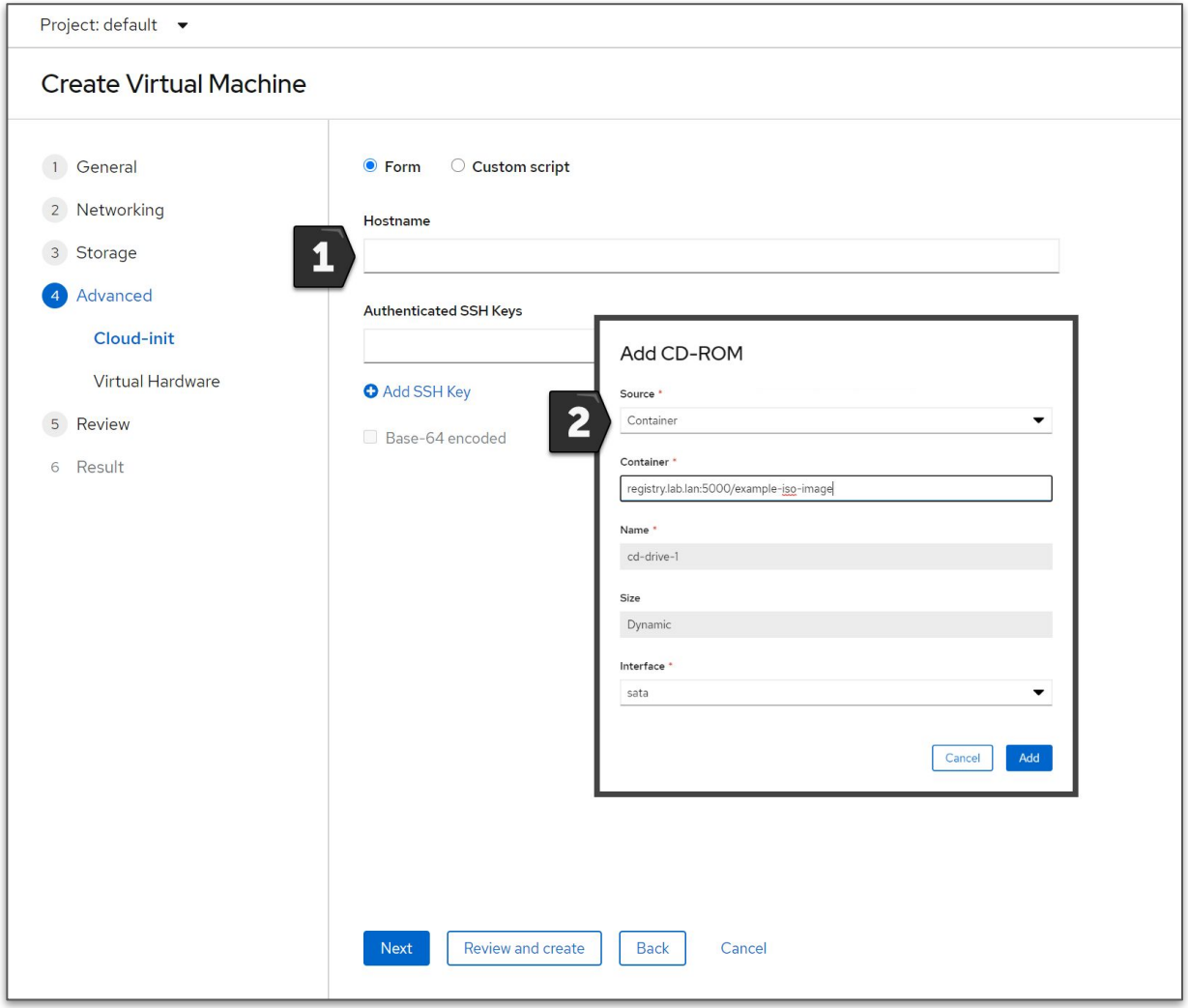





Red Hat

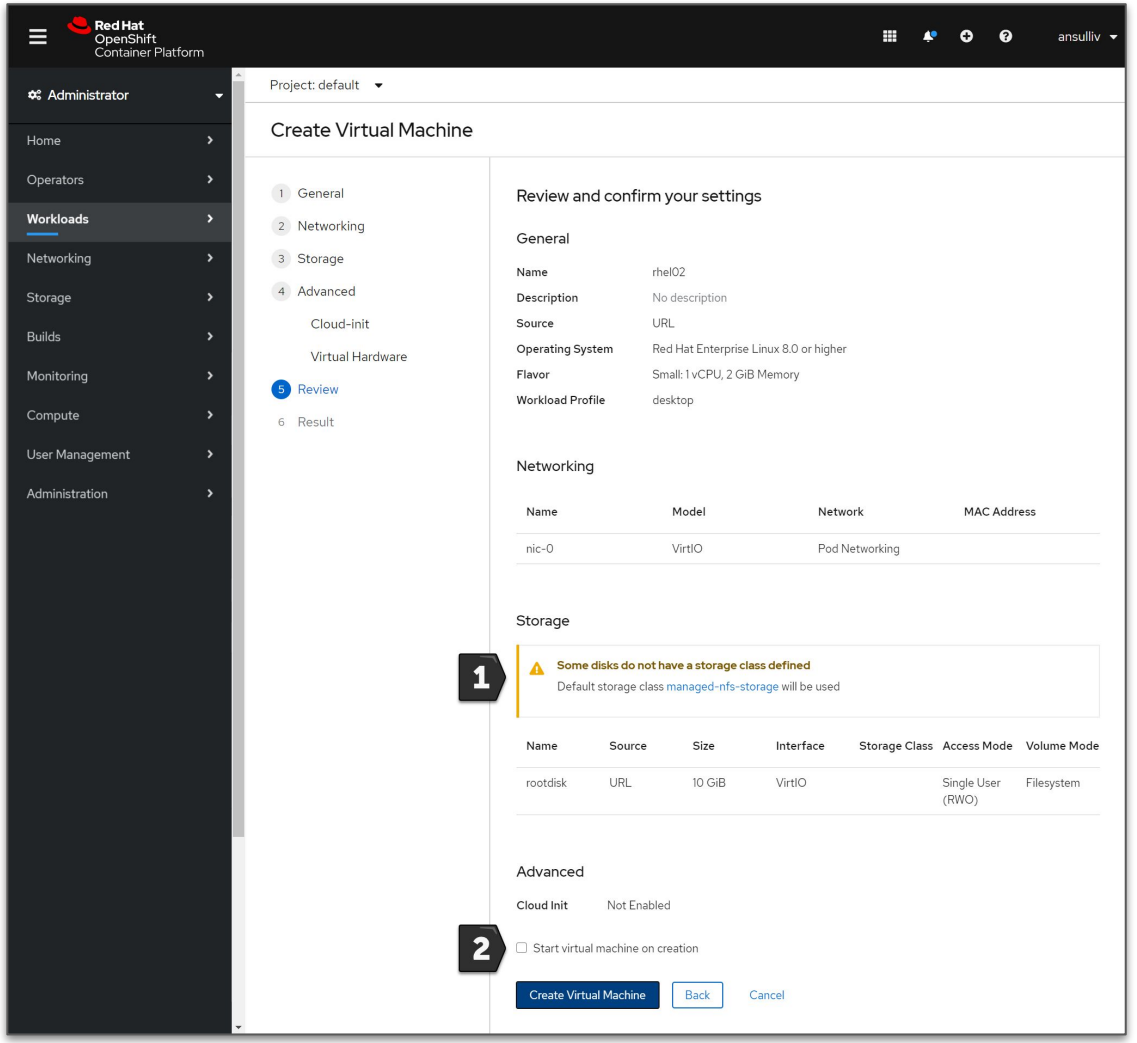
# Create Virtual Machine - Advanced

- 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








Red Hat

# Import VMs

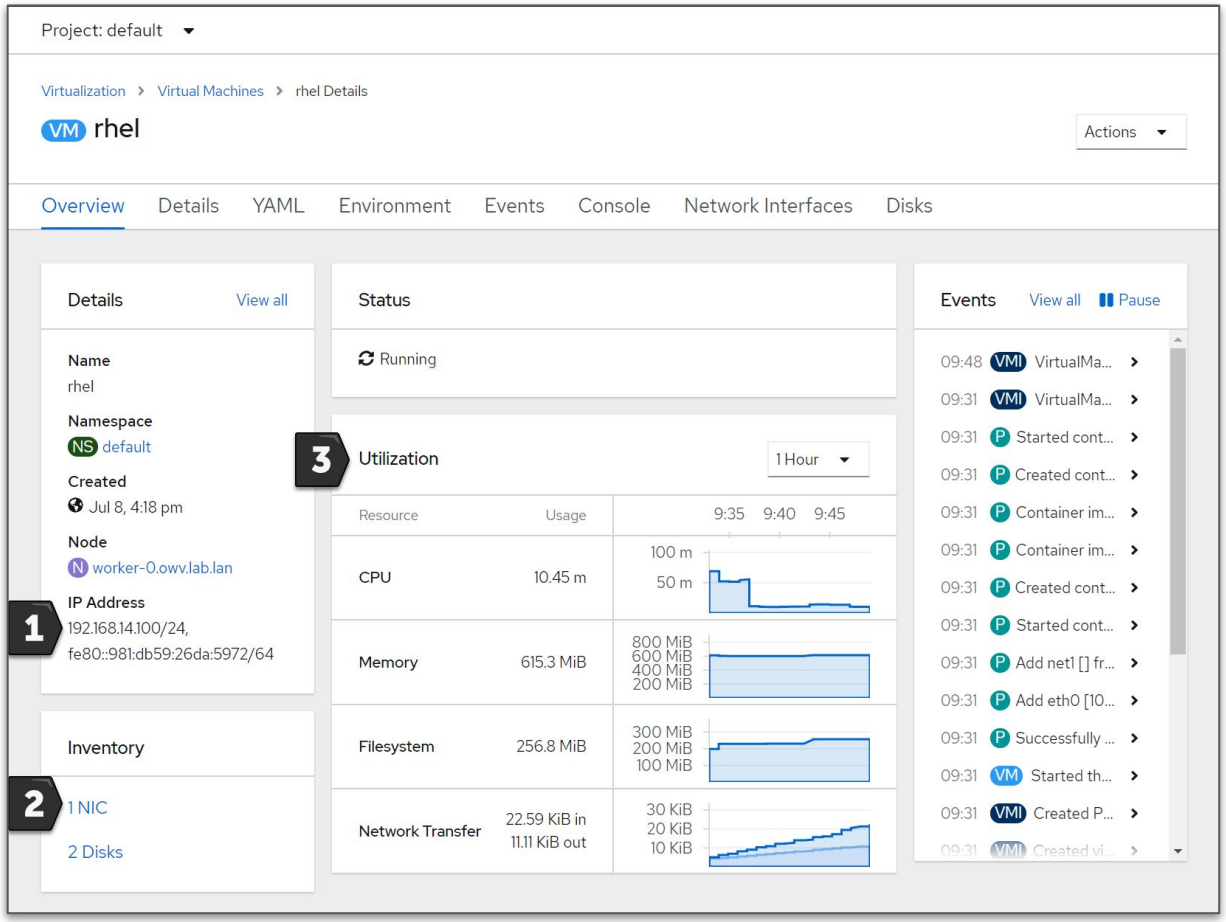
# Virtual Machine Import

- 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

# View / manage VMs

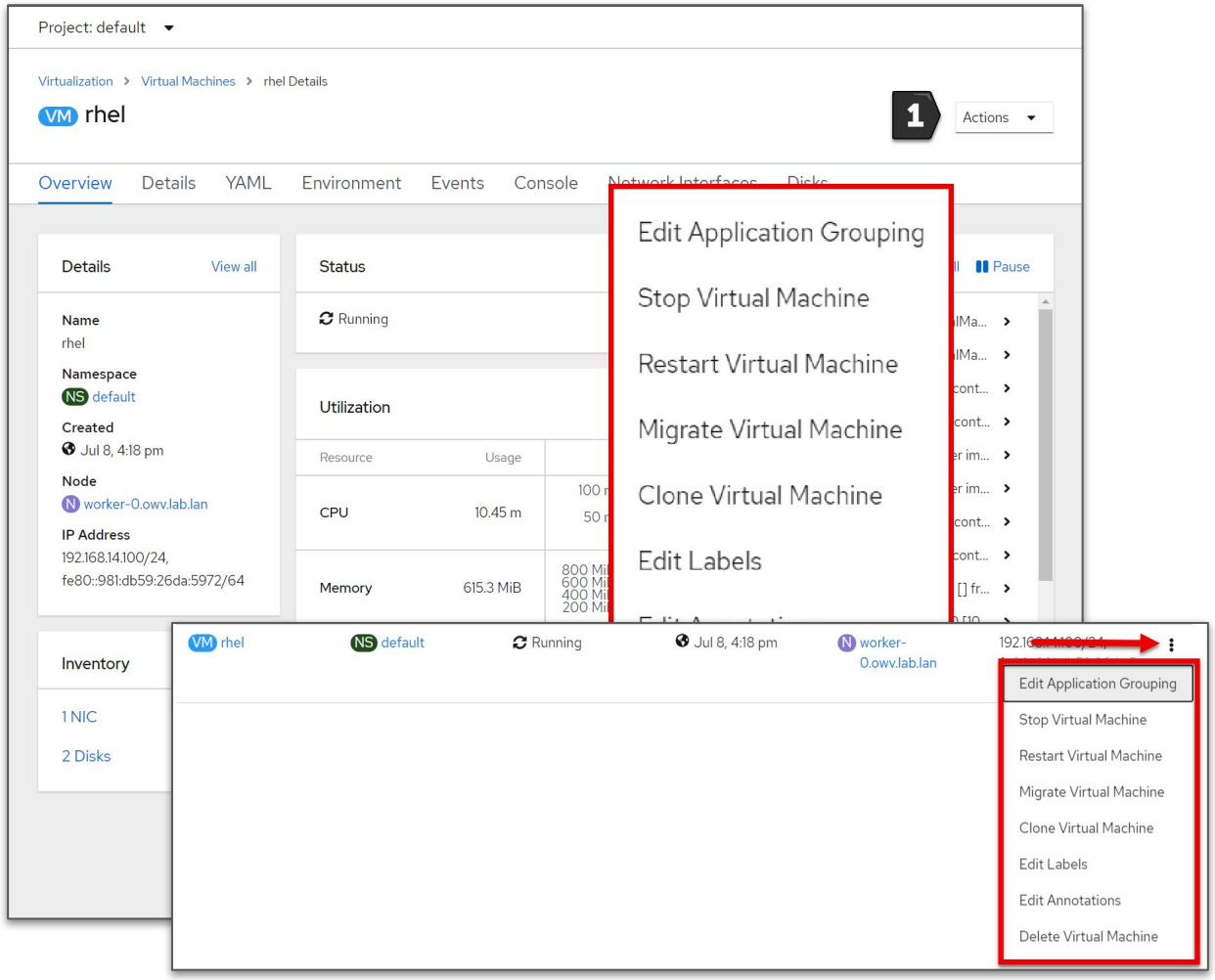
# Virtual Machine - Overview

- 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



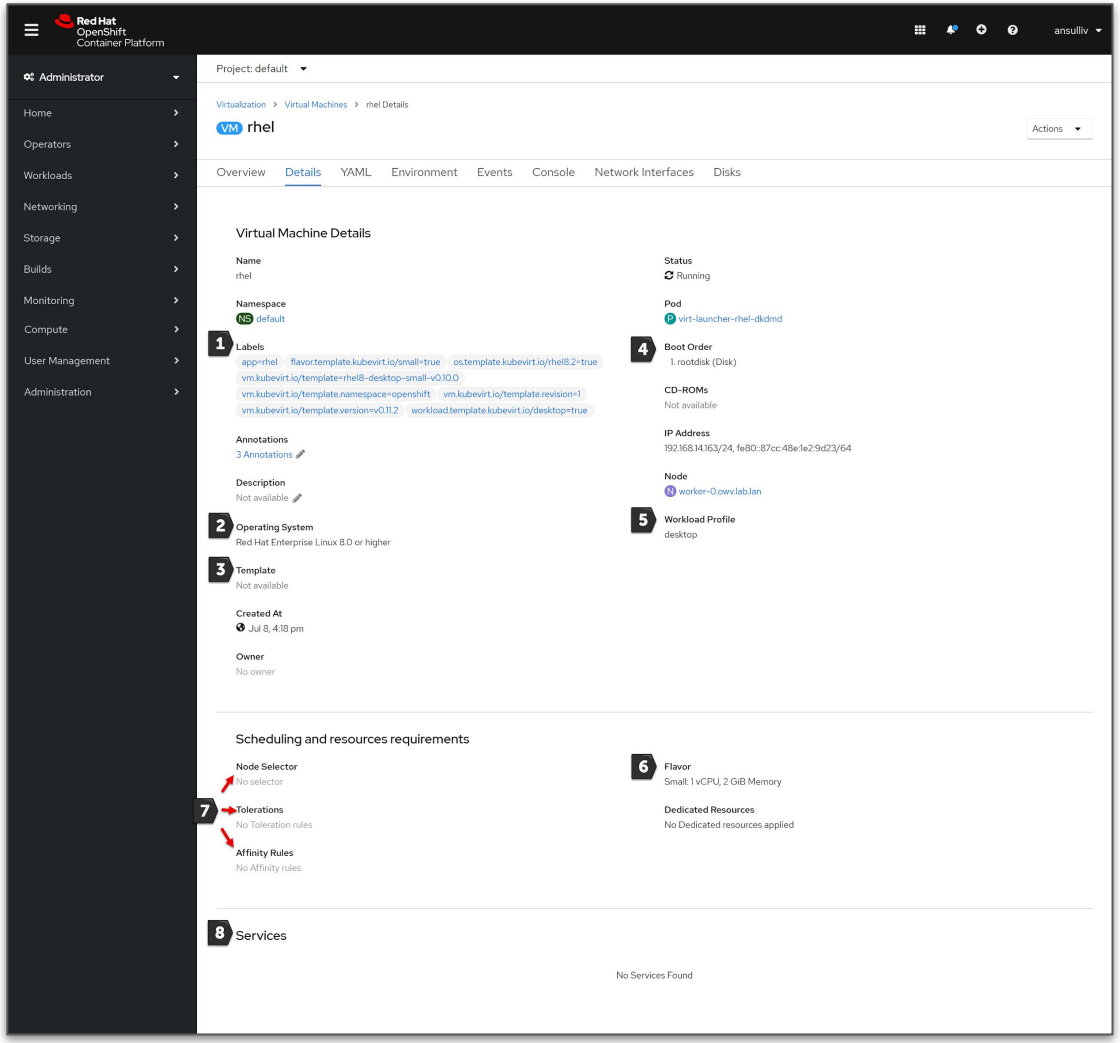





Red Hat

# 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



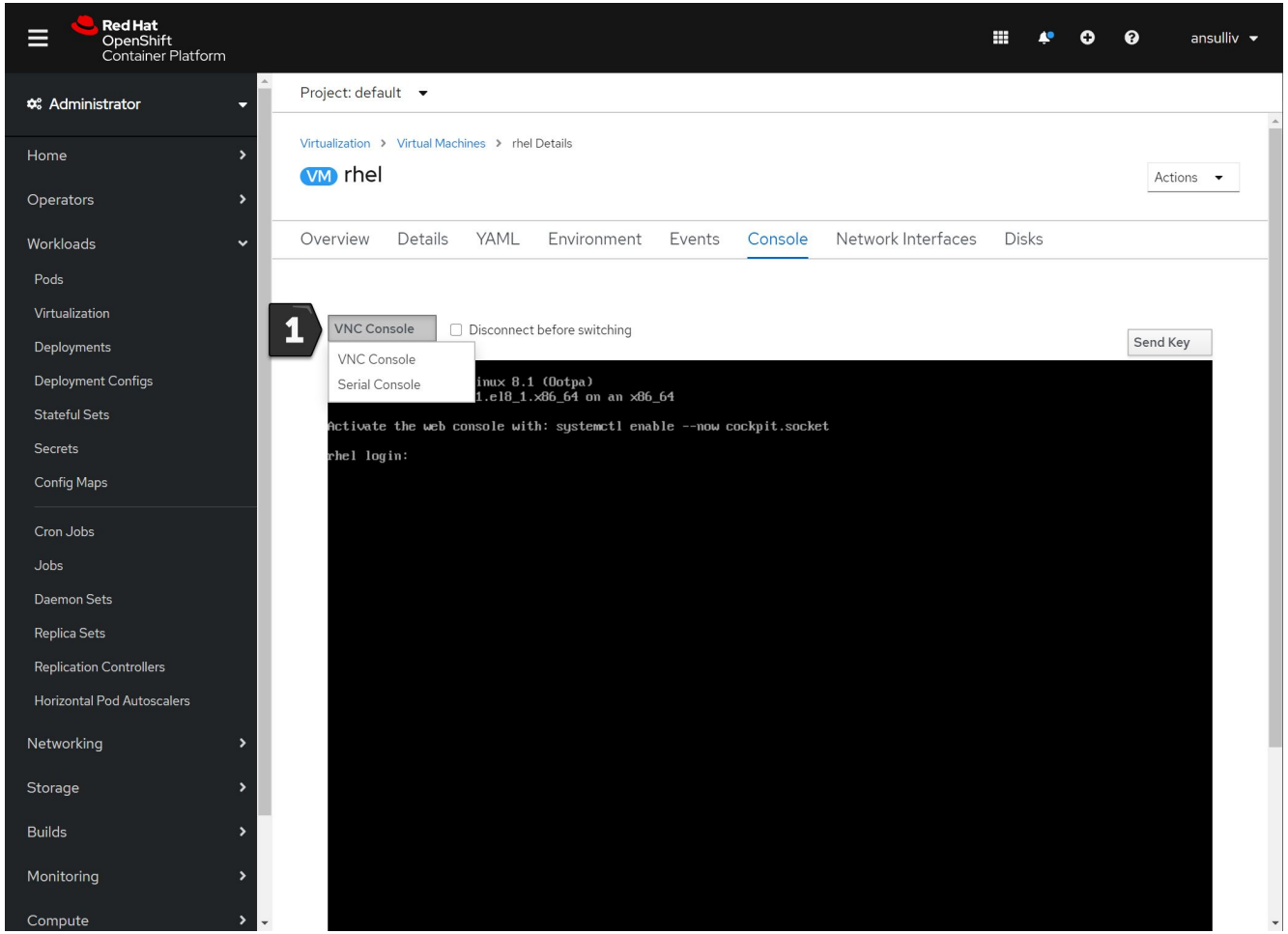





Red Hat

# 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`



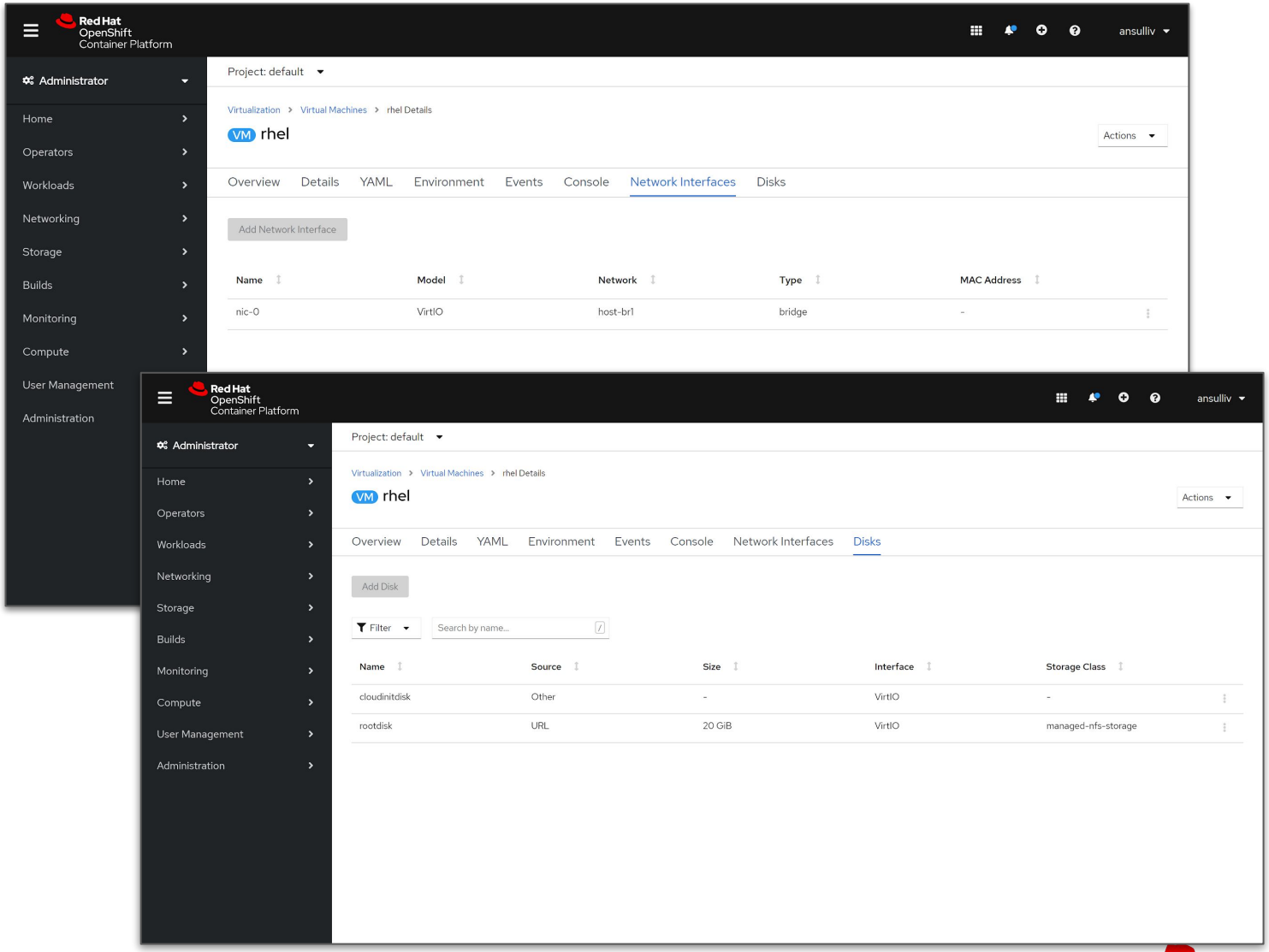





Red Hat

# Virtual Machine - Disks and NICs

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





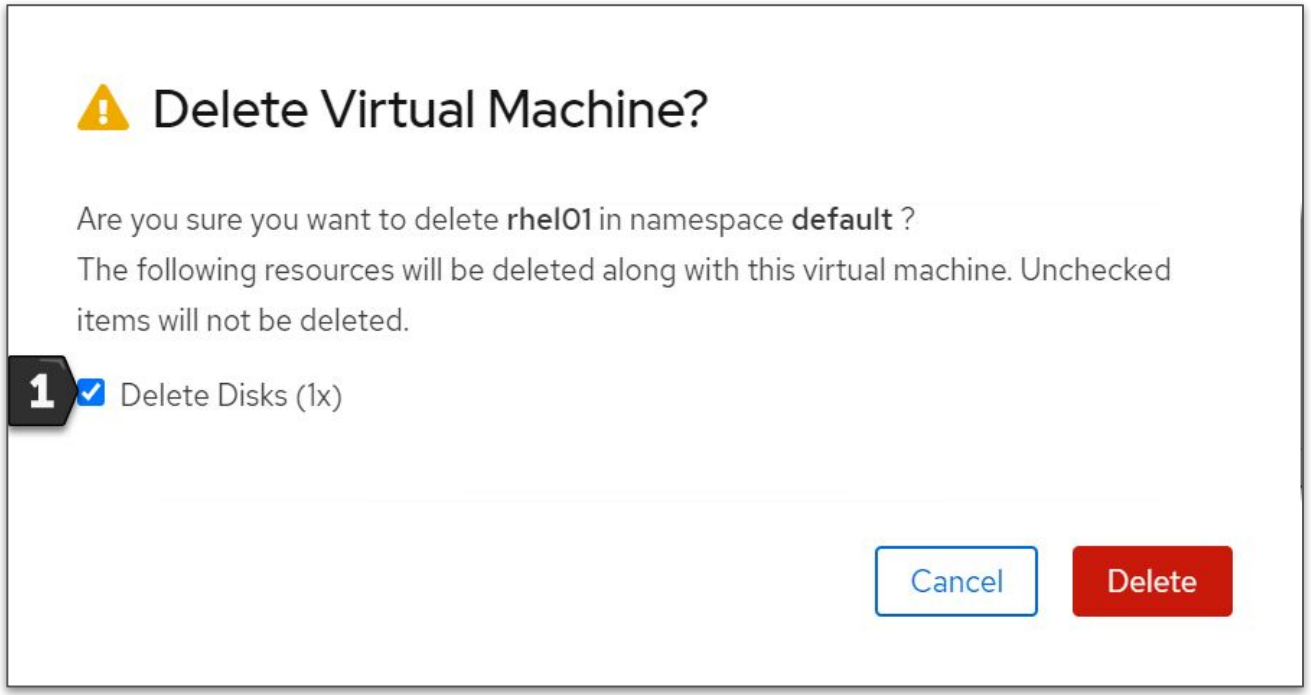



Red Hat

# Destroy VMs

# Destroying a Virtual Machine

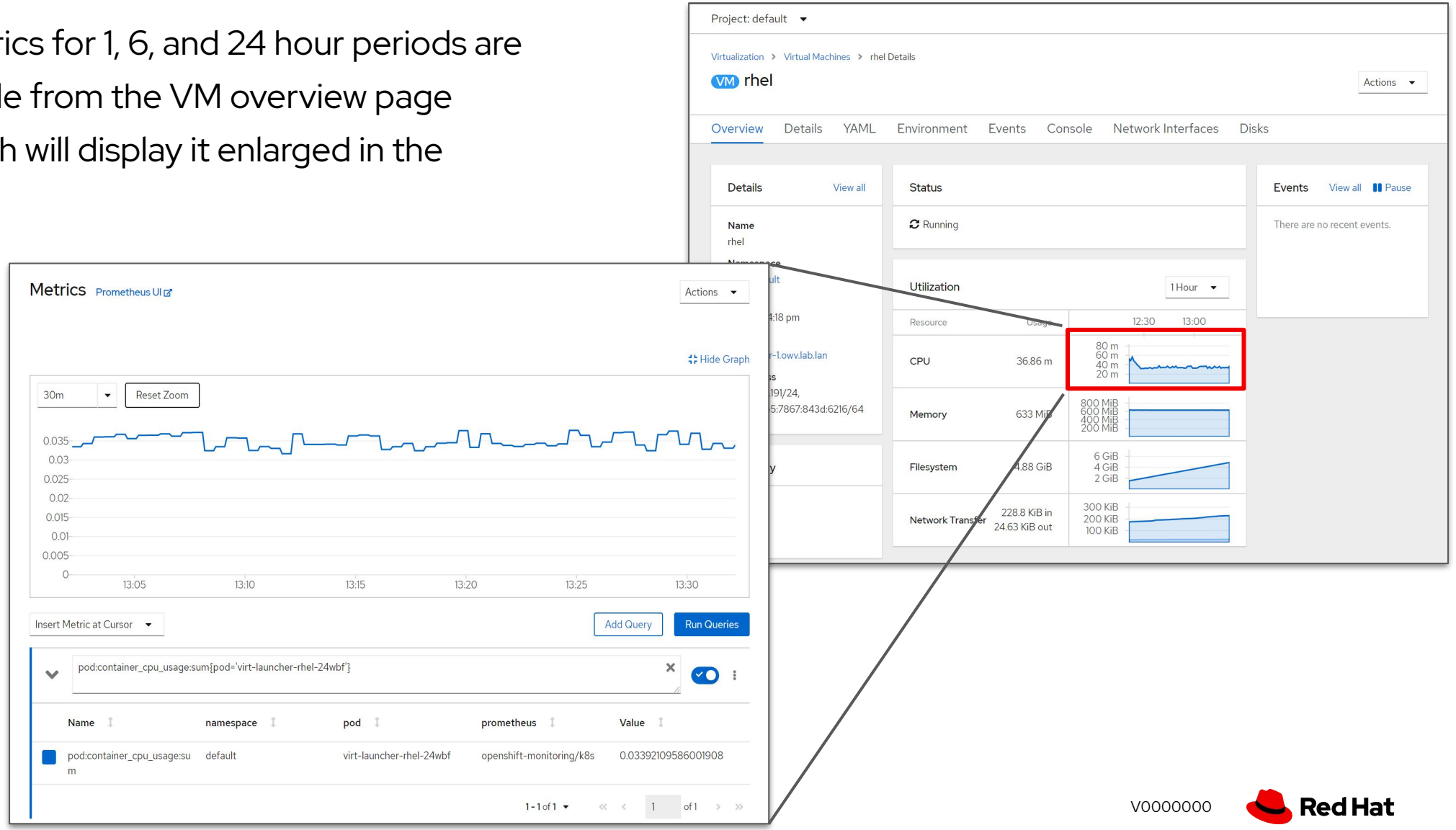
- Deleting a VM removes the VM definition
  - Optionally delete PVC-backed disks associated with the VM
- Running VMs are terminated first
- Other associated resources, e.g. Services, are not affected



# 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



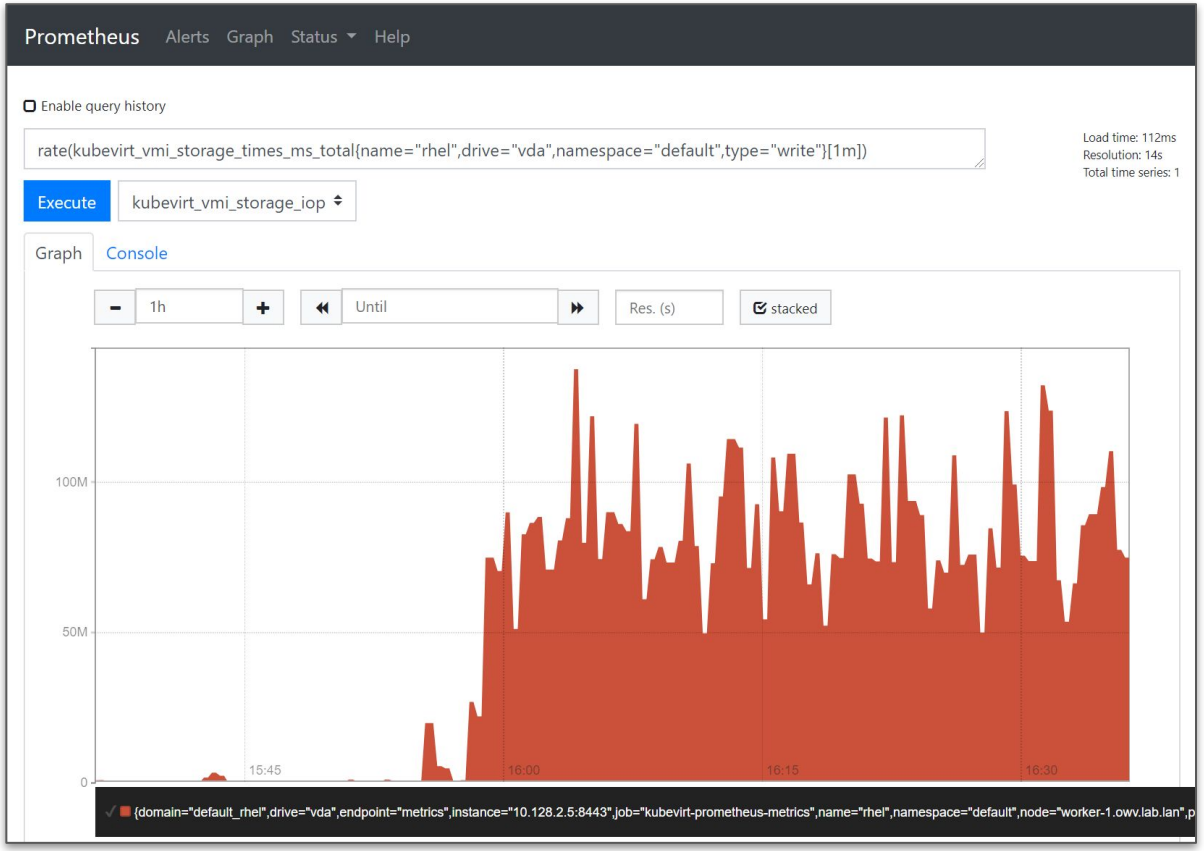





Red Hat

# 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



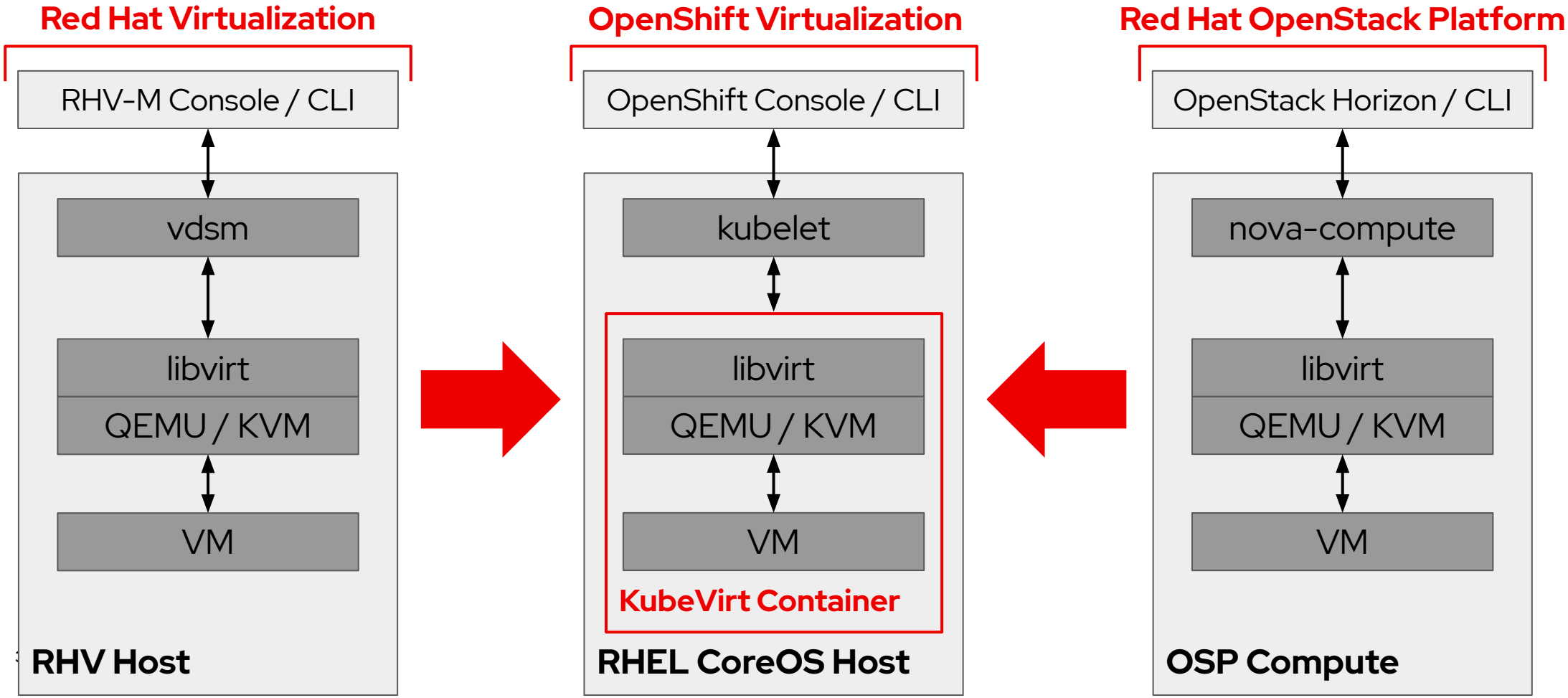





Red Hat

# Deeper into the technology

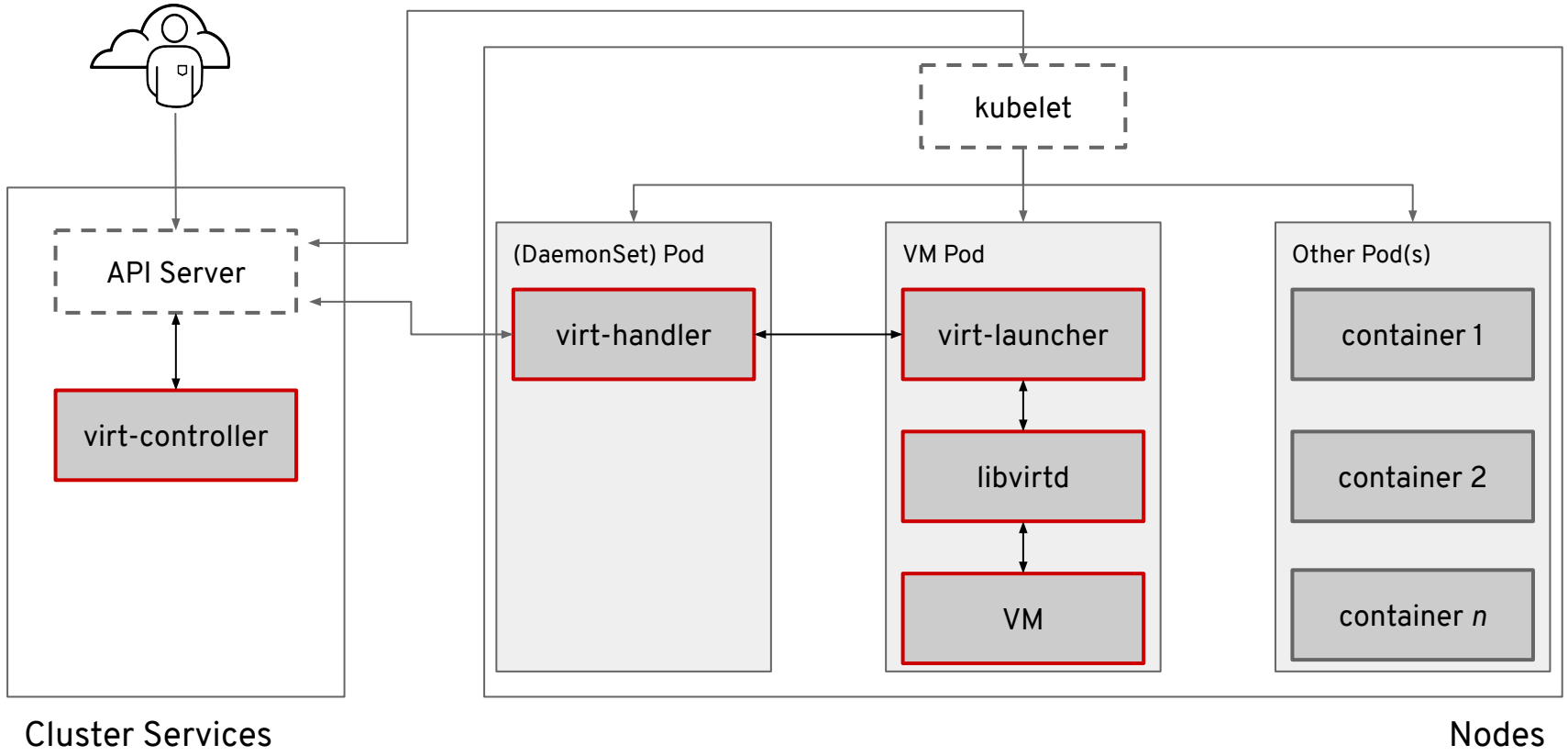
# Containerizing KVM






Red Hat

# Architectural Overview



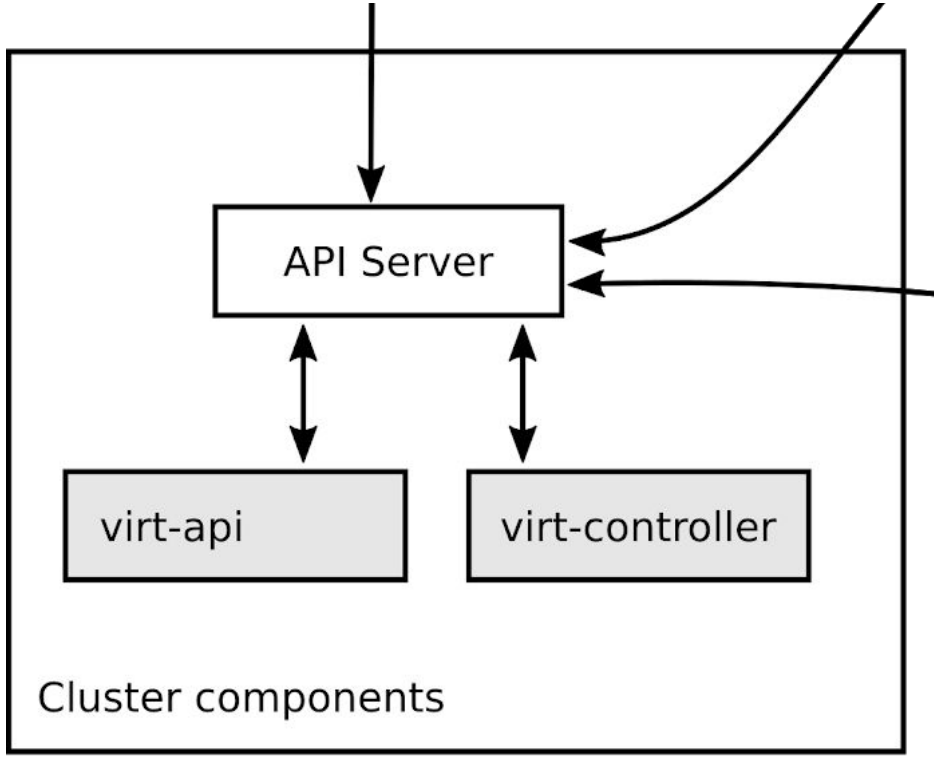
# Adding virtualization to the Kubernetes API

**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



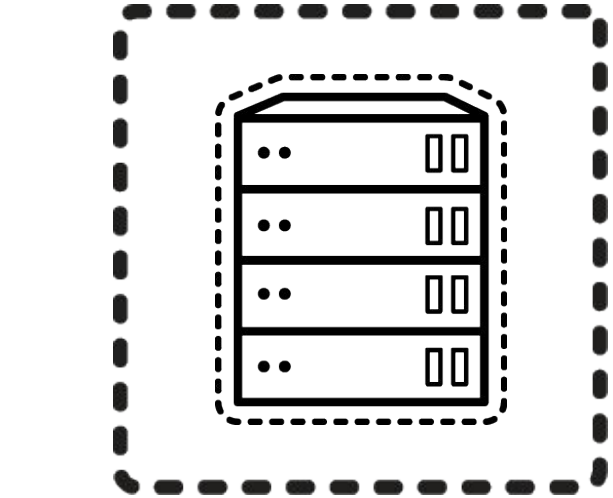
# OpenShift cluster architecture

- 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

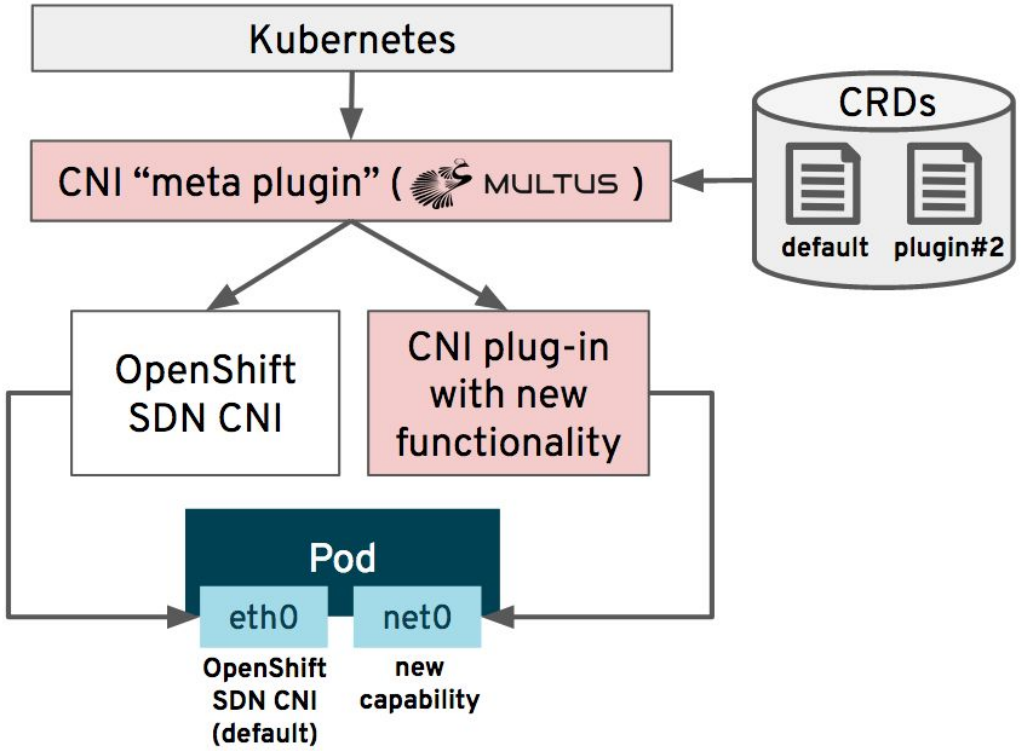


Red Hat OpenShift

# Network

# Virtual Machine Networking

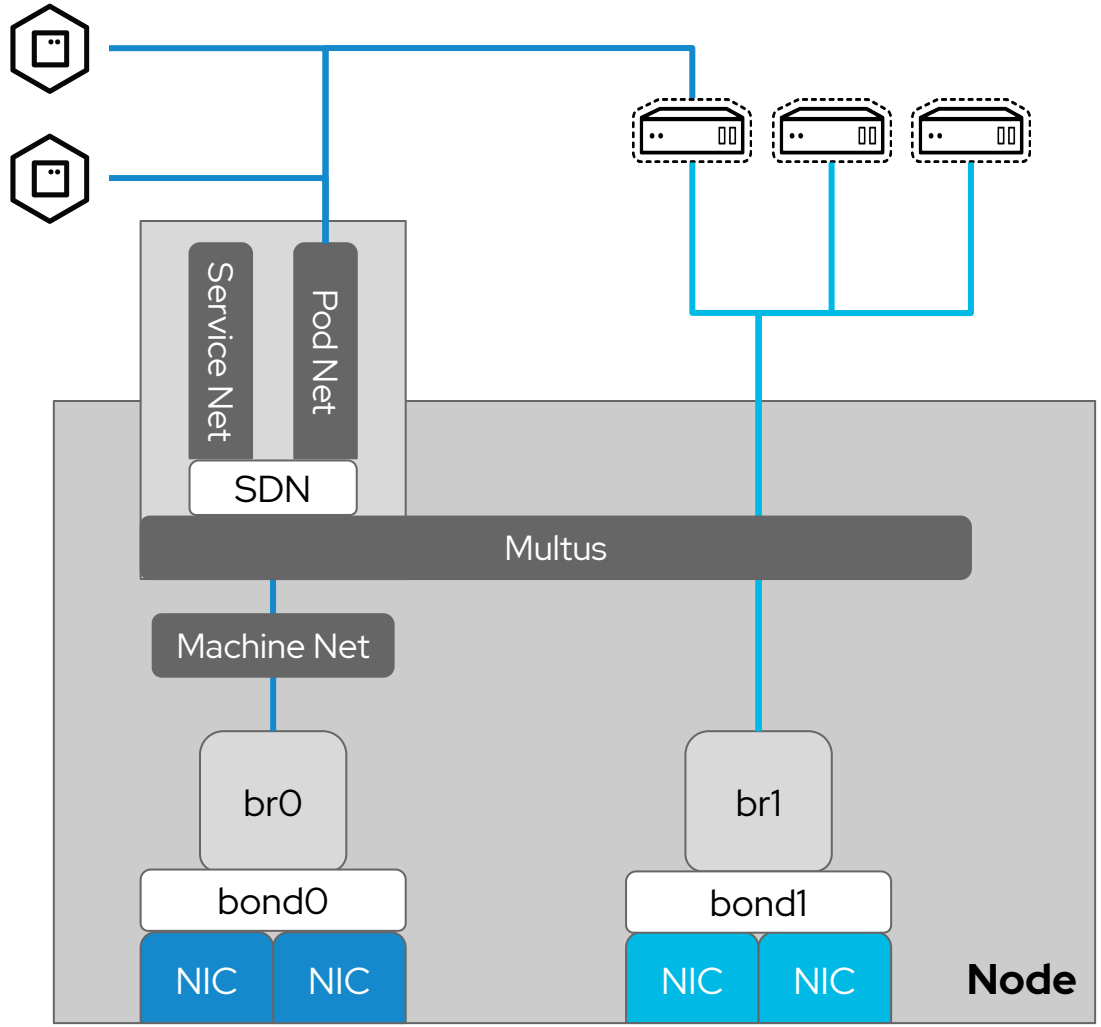
- Virtual machines optionally connect to the standard pod network
  - OpenShift SDN, OVNKubernetes, etc.
- Additional network interfaces accessible via Multus:
  - Bridge, SR-IOV
  - VLAN and other networks can be created using nmstate at the host level
- When using at least one interface on the default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



# Example host network configuration

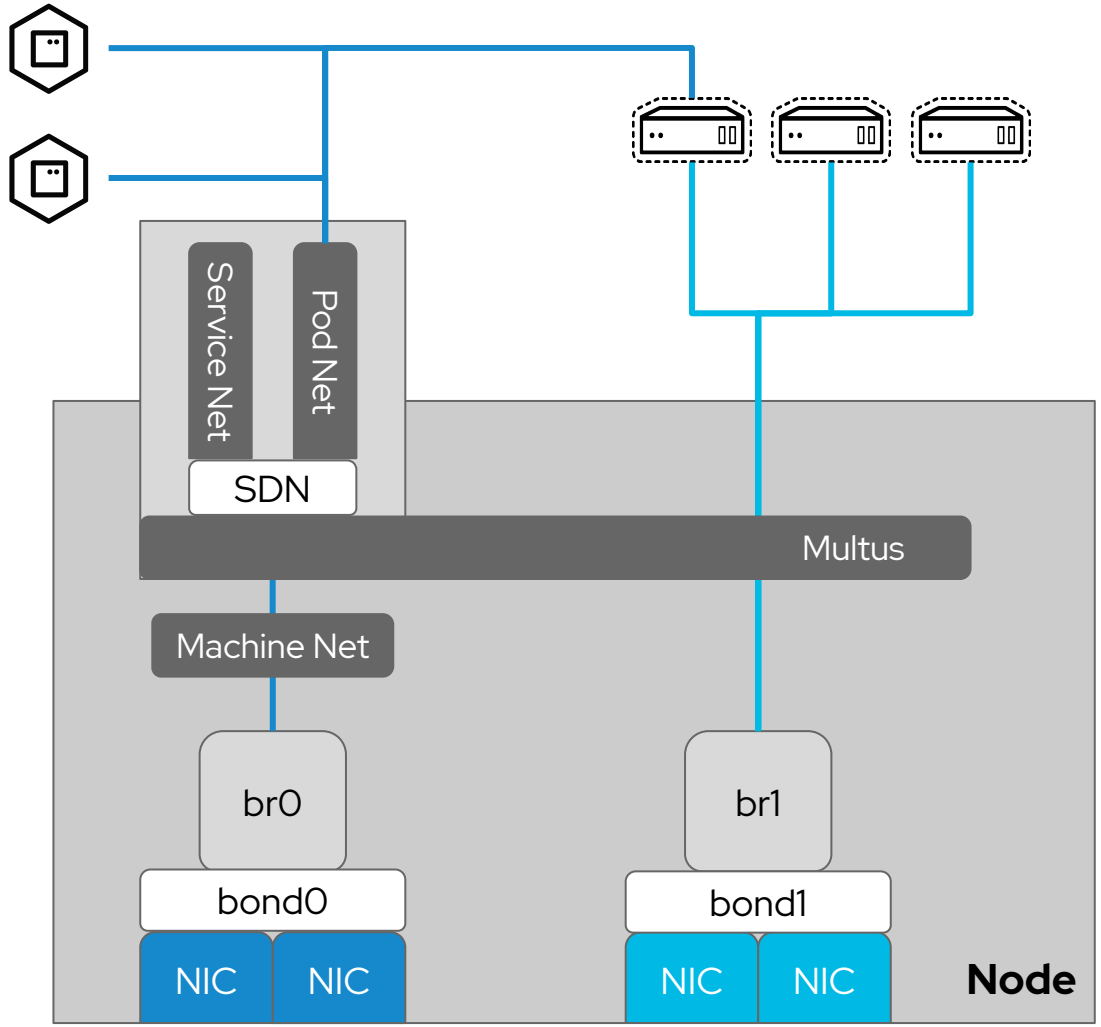
- Pod, service, and machine network are configured by OpenShift automatically
  - Use kernel parameters (dracut) for configuration at install
- Use `kubernetes-nmstate`, via the nmstate Operator, to configure additional host network interfaces
  - `bond1` and `br1` in the example to the right
- VM pods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

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

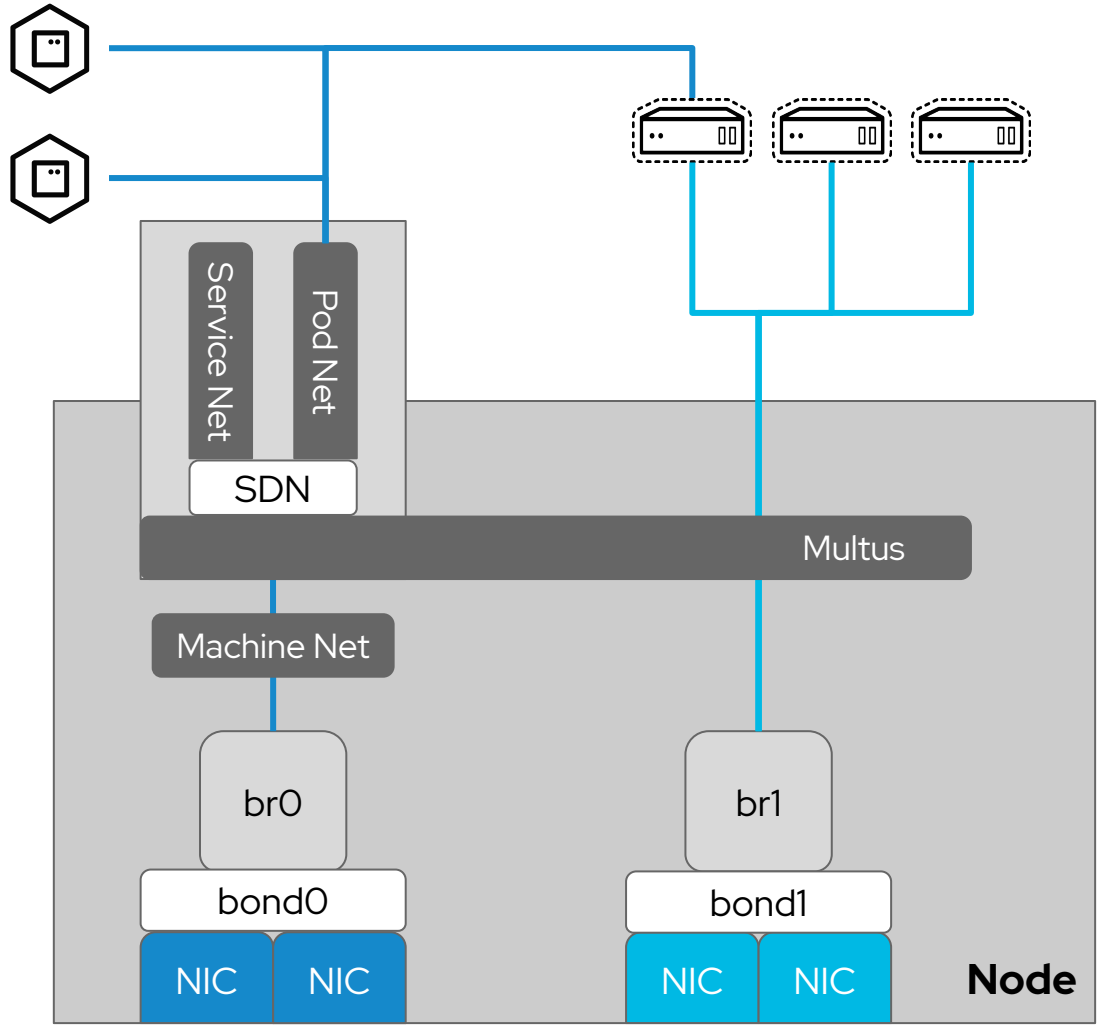


# Host bond configuration

- 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



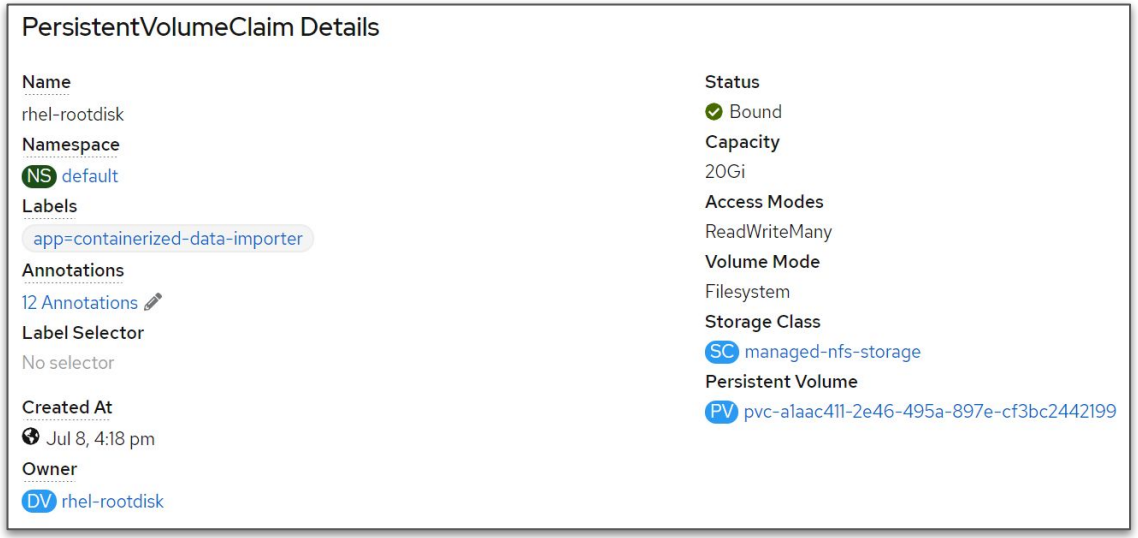
# Host bridge configuration



# 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



# 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

```yaml
dataVolumeTemplates:
  - apiVersion: cdi.kubevirt.io/v1alpha1
    kind: DataVolume
    metadata:
      creationTimestamp: null
      name: vm-rootdisk
    spec:
      pvc:
        accessModes:
          - ReadWriteMany
        resources:
          requests:
            storage: 20Gi
        storageClassName: my-storage-class
        volumeMode: Filesystem
      source:
        http:
          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 (limit > request)  allows more overcommit, but may lead to more frequent migrations
  - Guaranteed (limit = 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

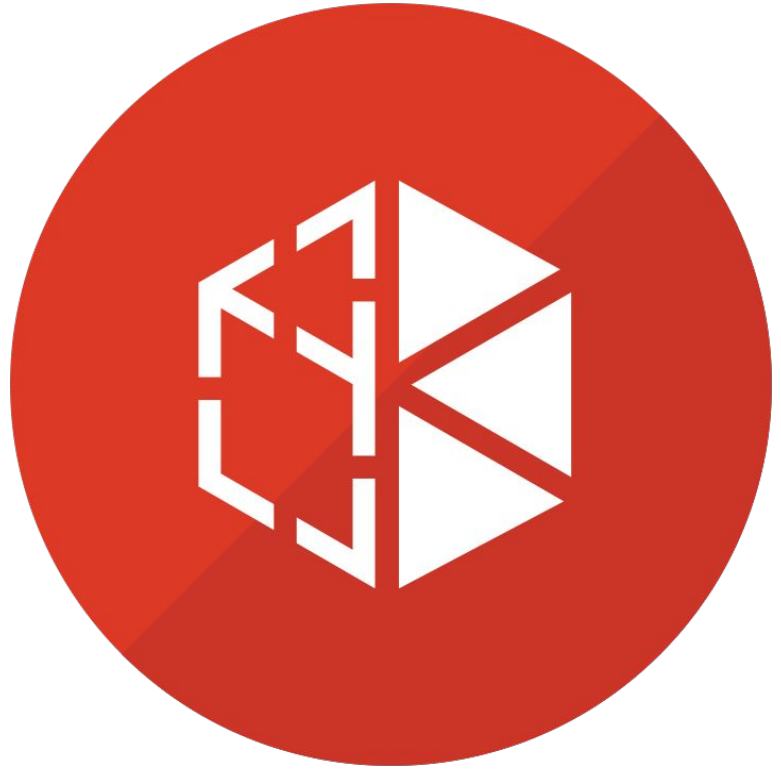
# High availability

- 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<br>This list is not exhaustive; specific platforms, product names are provided as examples to guide discussions within target opportunities and use cases. | 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 |

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

| Workload<br>This list is not exhaustive; specific platforms, product names are provided as examples to guide discussions within target opportunities and use cases. | 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<br>This list is not exhaustive; specific platforms, product names are provided as examples to guide discussions within target opportunities and use cases. | 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

## Near Term
(3-6 months)

**CORE**
- Streamline 'oc drain'
- Improve performance with Hyper-V enlightenments
- Improve golden image template workflow
- Sizing guidance for field

**NETWORK**

**IPv6**
- Dual stack
- Multicast
- IPv6 multi-address
- Enhance CNI test suite for VMs

**STORAGE**
- Offline snapshots
- Hotplug disk PoC
- Performance and scale of VMs on OCS
- Enhance CNI test suite for VMs

## Mid Term
(6-9 months)

**CORE**

**Virt gaps -** vCPU & vNUMA pinning

**Security** - FIPS

**Developer**
- Using VMs naturally in Developer Pipelines

**NETWORK**
- Hotplug - NIC

**Ecosystem CNI certification**
- Tigera - Calico

**STORAGE**
- Application consistent snapshots and cloning
- Improve default performance with io=native

**Backup and DR**
- via OCS data protection
- Ecosystem CSI certification

## Long Term
(9+ months)

**CORE**
- OCP as on-prem infrastructure
- **Cloud** - Bare Metal where supported by public cloud provider
- **Mass migration**- VMw to OCP & RHV to OCP

**NETWORK**
- OVS HW offload

**Ecosystem CNI certification**
- Nuage
- Juniper

**STORAGE**
- **High Availability**



Product Manager: Peter Lauterbach, Ronen Sde-Or, and Rob Young

Red Hat

# Positioning, Selling the right Red Hat virtualization platform

| Understanding the customer problem to be solved | Traditional Virtualization<br>Red Hat Virtualization | IaaS Platform Virtualization<br>Red Hat OpenStack Platform | Kubernetes Managed Virtualization<br>Red Hat OpenShift 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** | Customer Presentation<br><br>Customer Use Cases and References | Customer Presentation<br><br>Customer Use Cases and References | Customer Presentation<br><br>Customer Use Cases and References |
| **Technical Requirements** | Technical Specification for Compute, Network, Storage | Technical Specification for Compute, Network, Storage | Technical Specification for Compute, Network, Storage |
| **All Resources** | RHV on PnT Portal | RHOSP on PnT Portal | OpenShift on PnT Portal |





Red Hat

# 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


youtube.com/user/RedHatVideos


facebook.com/redhatinc


twitter.com/RedHat



Red Hat