Red Hat OpenShift

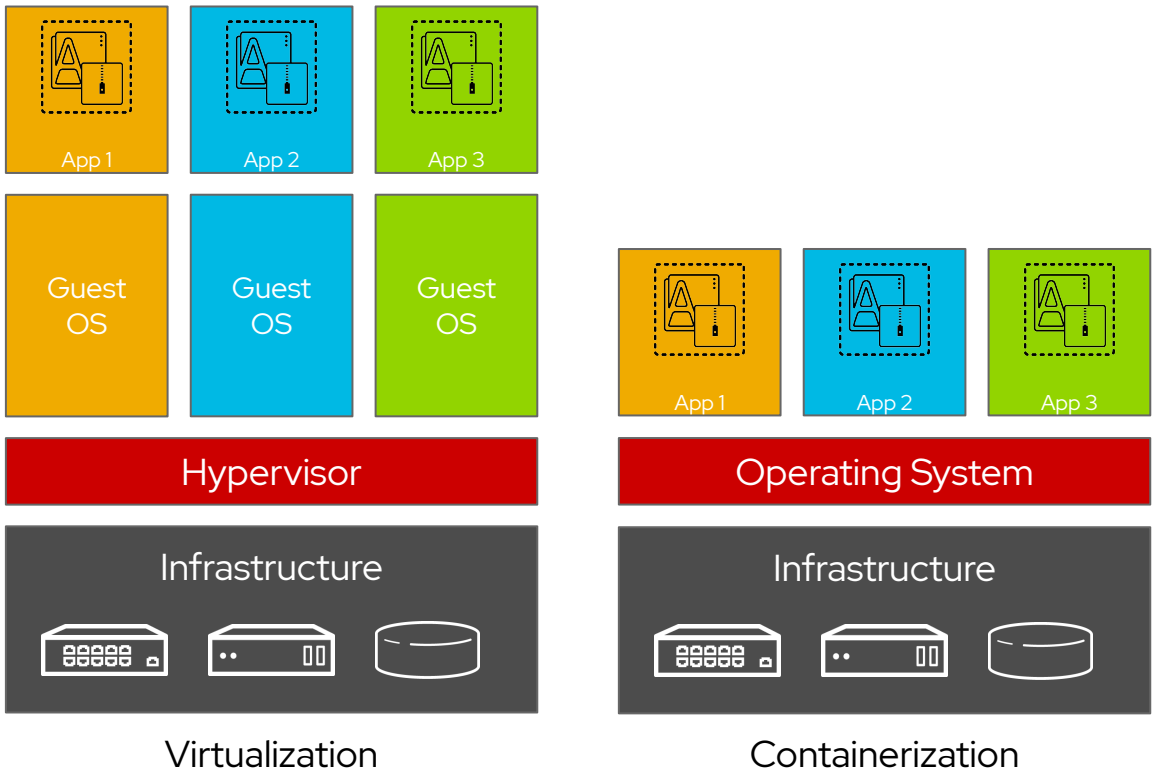
OpenShift Virtualization

Red Hat

What is OpenShift Virtualization?

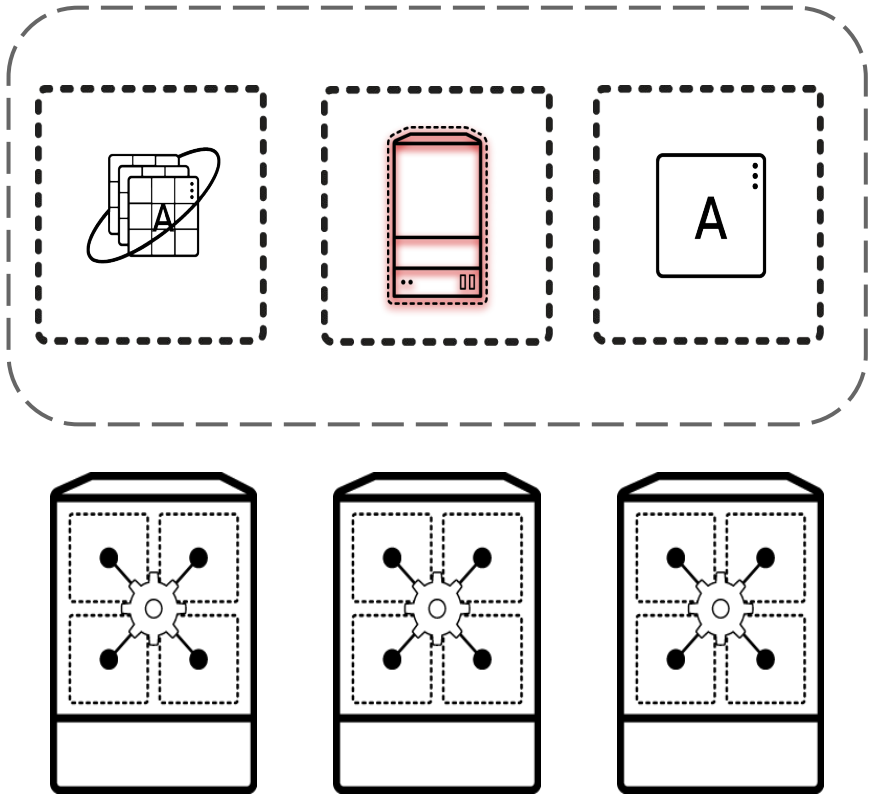
Containers are not virtual machines

- 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



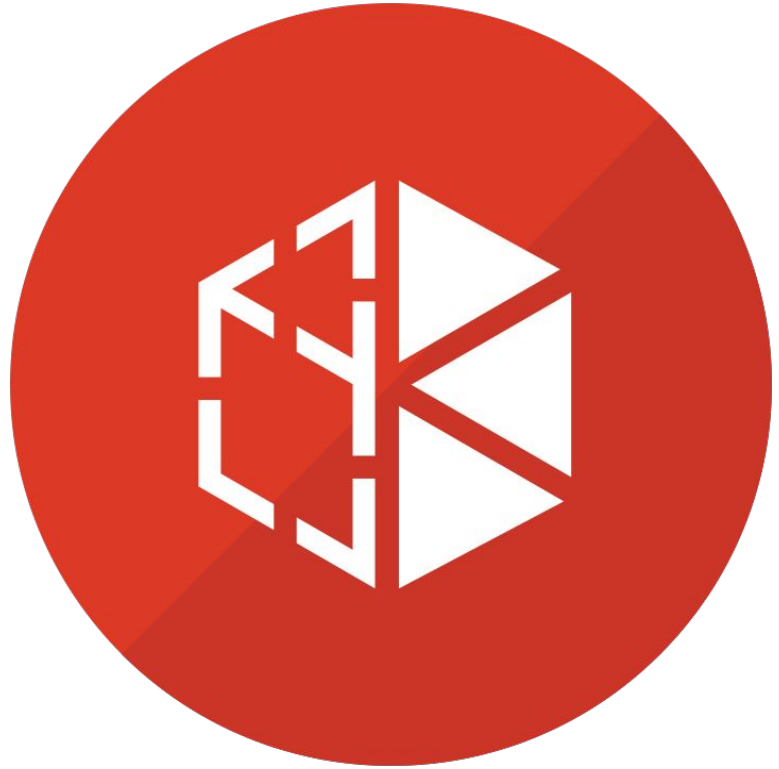
Virtual machines can be put into containers

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



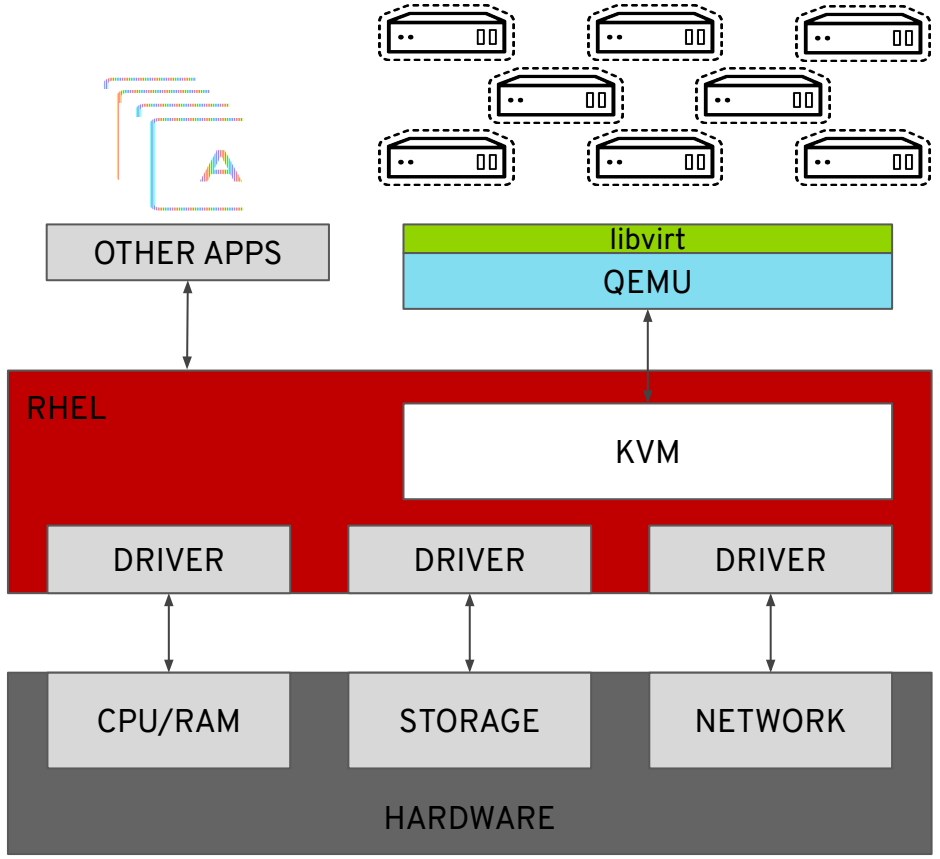
OpenShift Virtualization

- 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

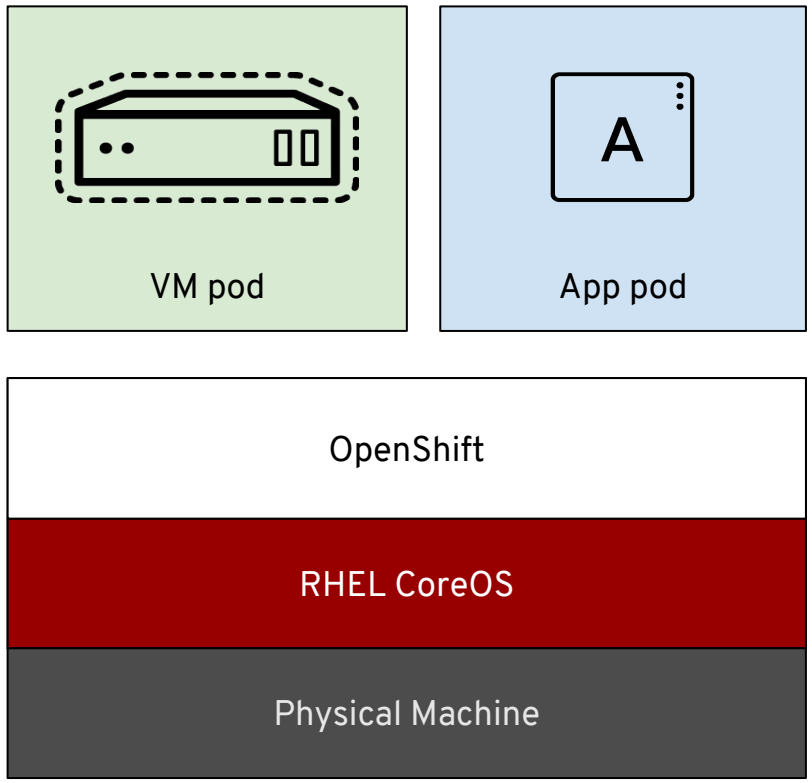
- OpenShift Virtualization uses KVM, the Linux kernel hypervisor
- KVM is a core component of the Red Hat Enterprise Linux kernel
  - KVM has 10+ years of production use: Red Hat Virtualization, Red Hat OpenStack Platform, and RHEL all leverage KVM, QEMU, and libvirt
- QEMU uses KVM to execute virtual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

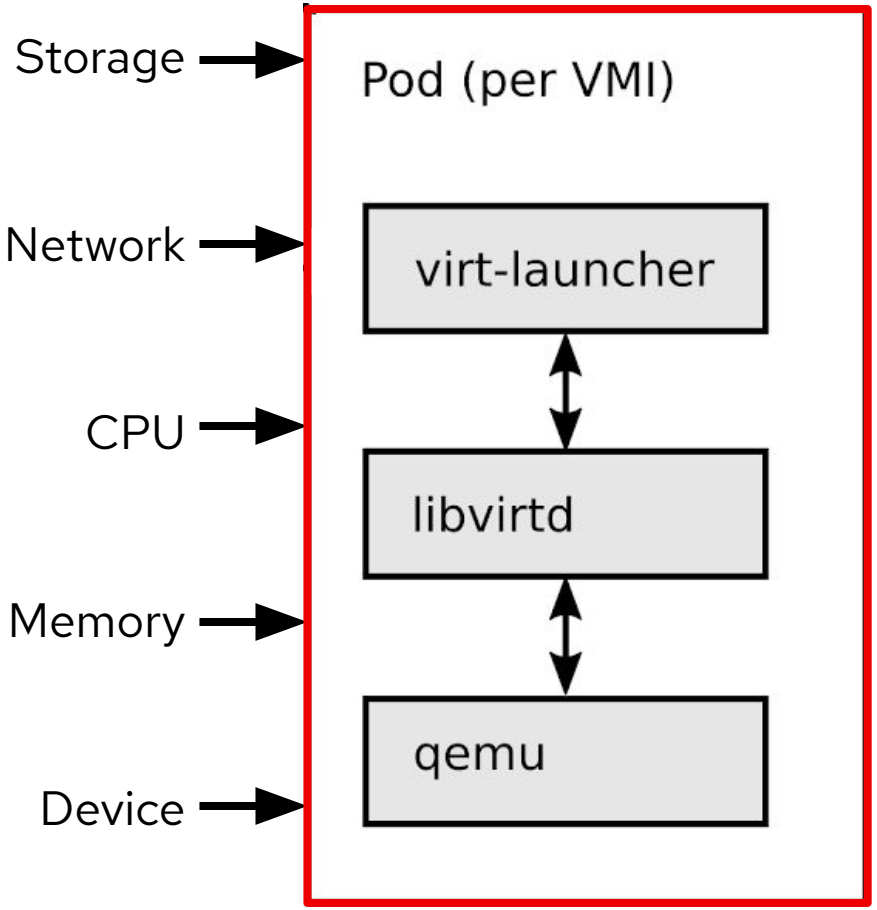


Virtualization native to Kubernetes

- 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

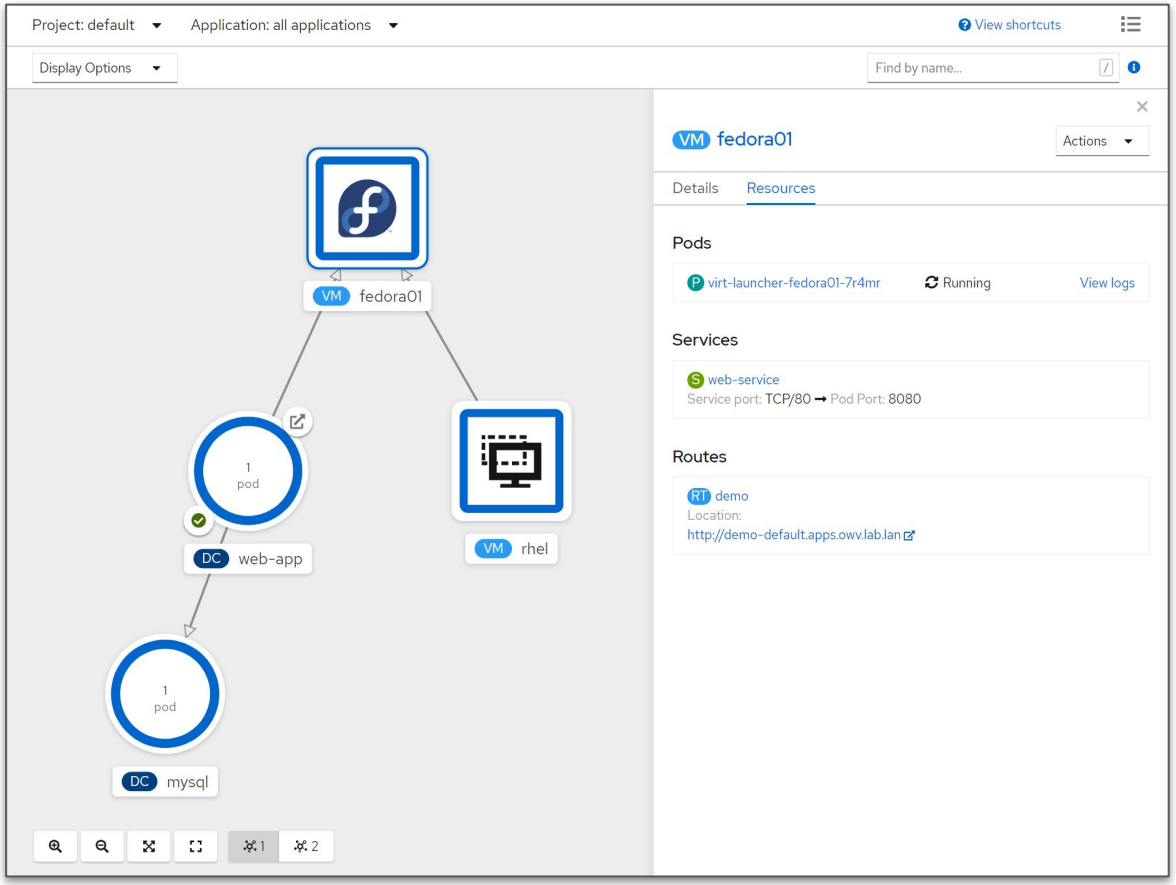
- libvirt and qemu from RHEL are mature, have high performance, provide stable abstractions, and have a minimal overhead.

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

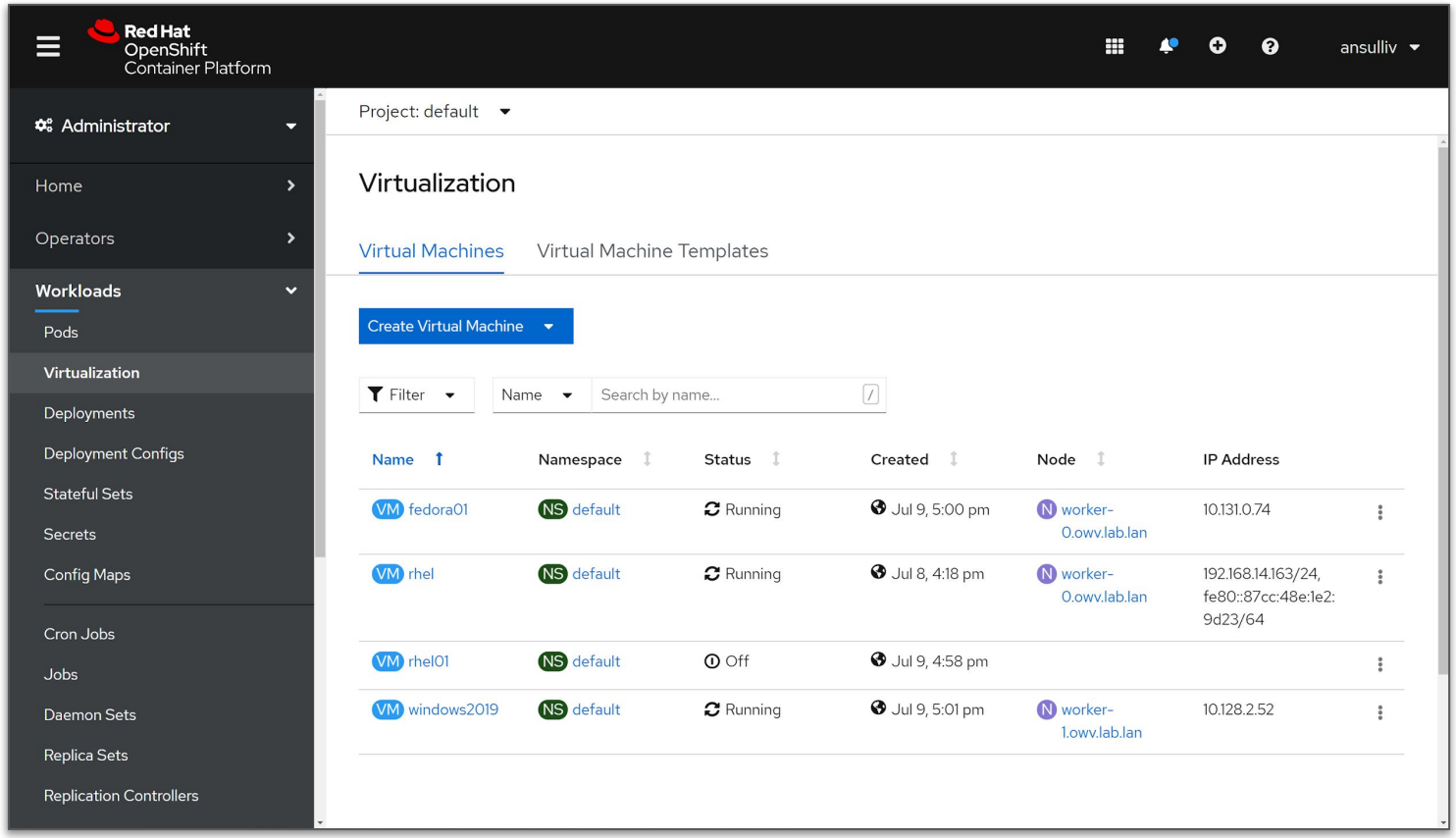


Red Hat

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

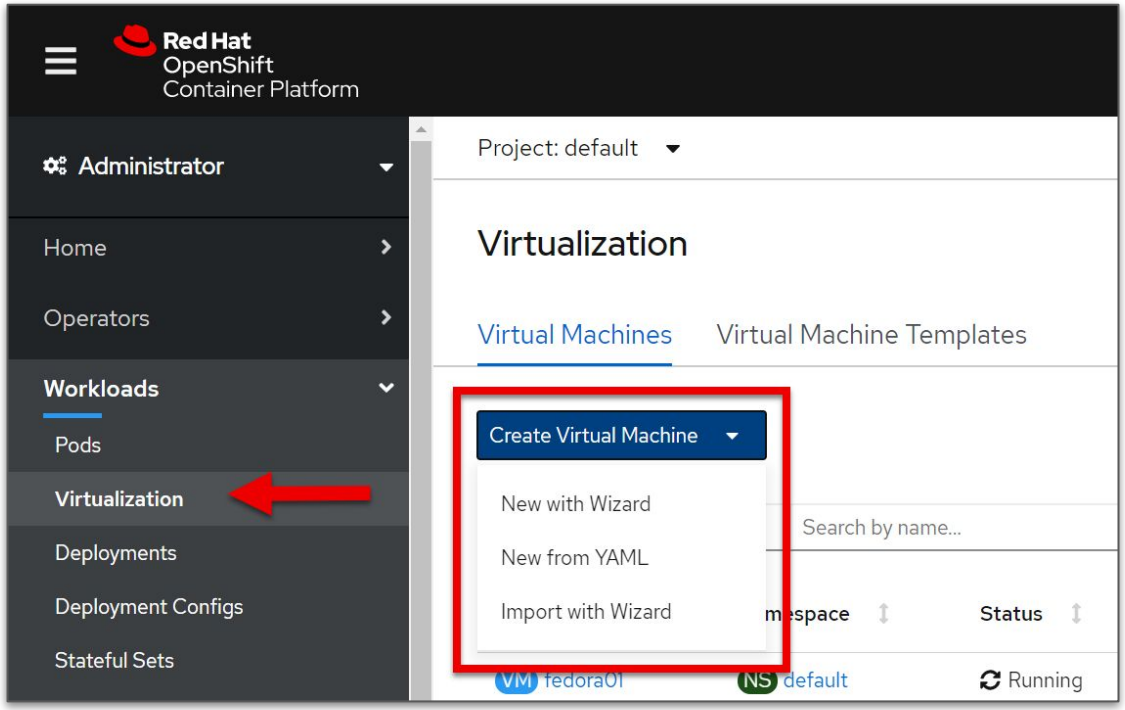


Red Hat

Create VMs

Virtual Machine creation

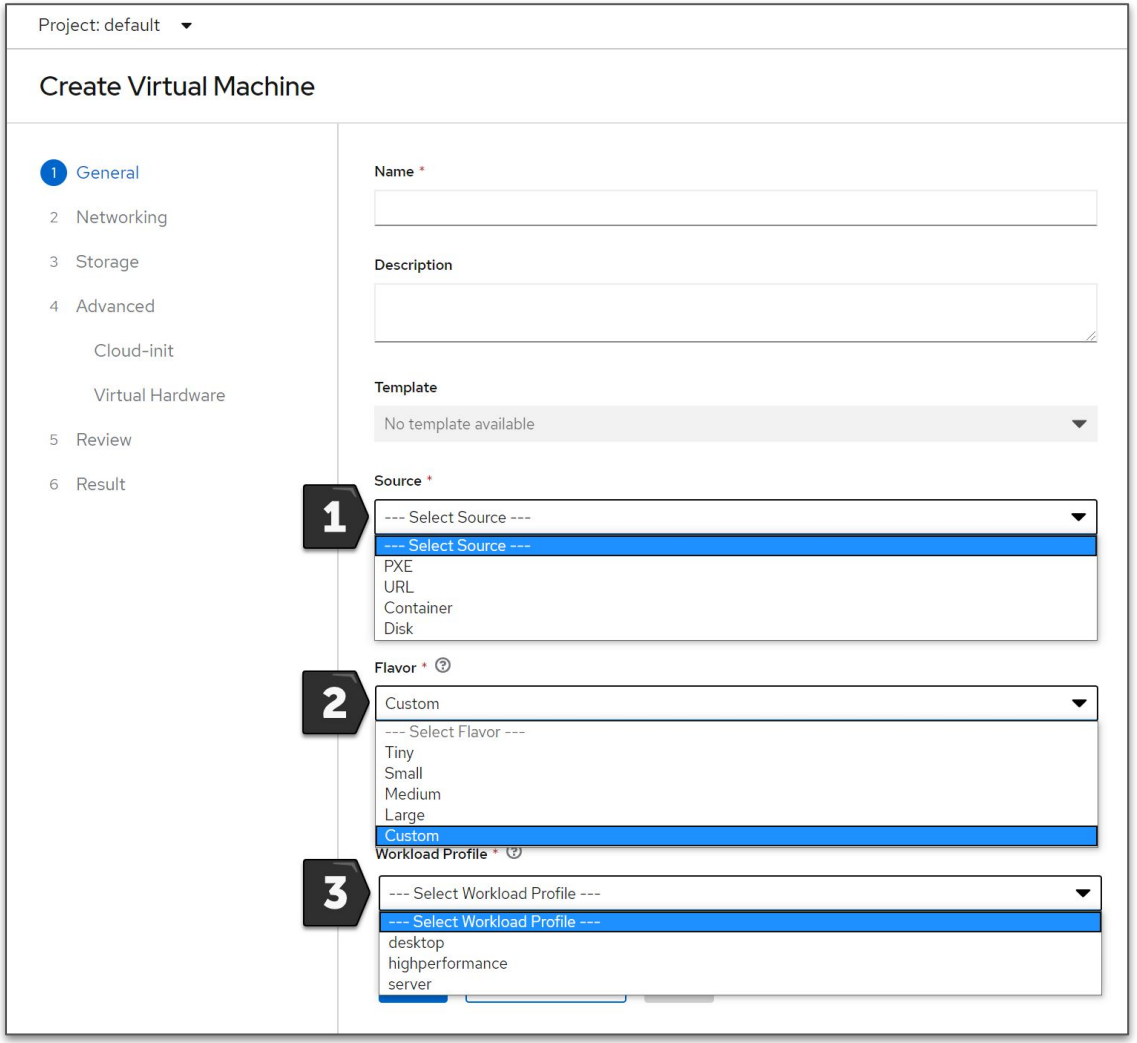
- Streamlined and simplified creation via the GUI or create VMs programmatically using YAML
- Full configuration options for compute, network, and storage resources
  - Clone VMs from templates or import disks using DataVolumes
  - Pre-defined and customizable presets for CPU/RAM allocations
  - Workload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



Red Hat

Create Virtual Machine - General

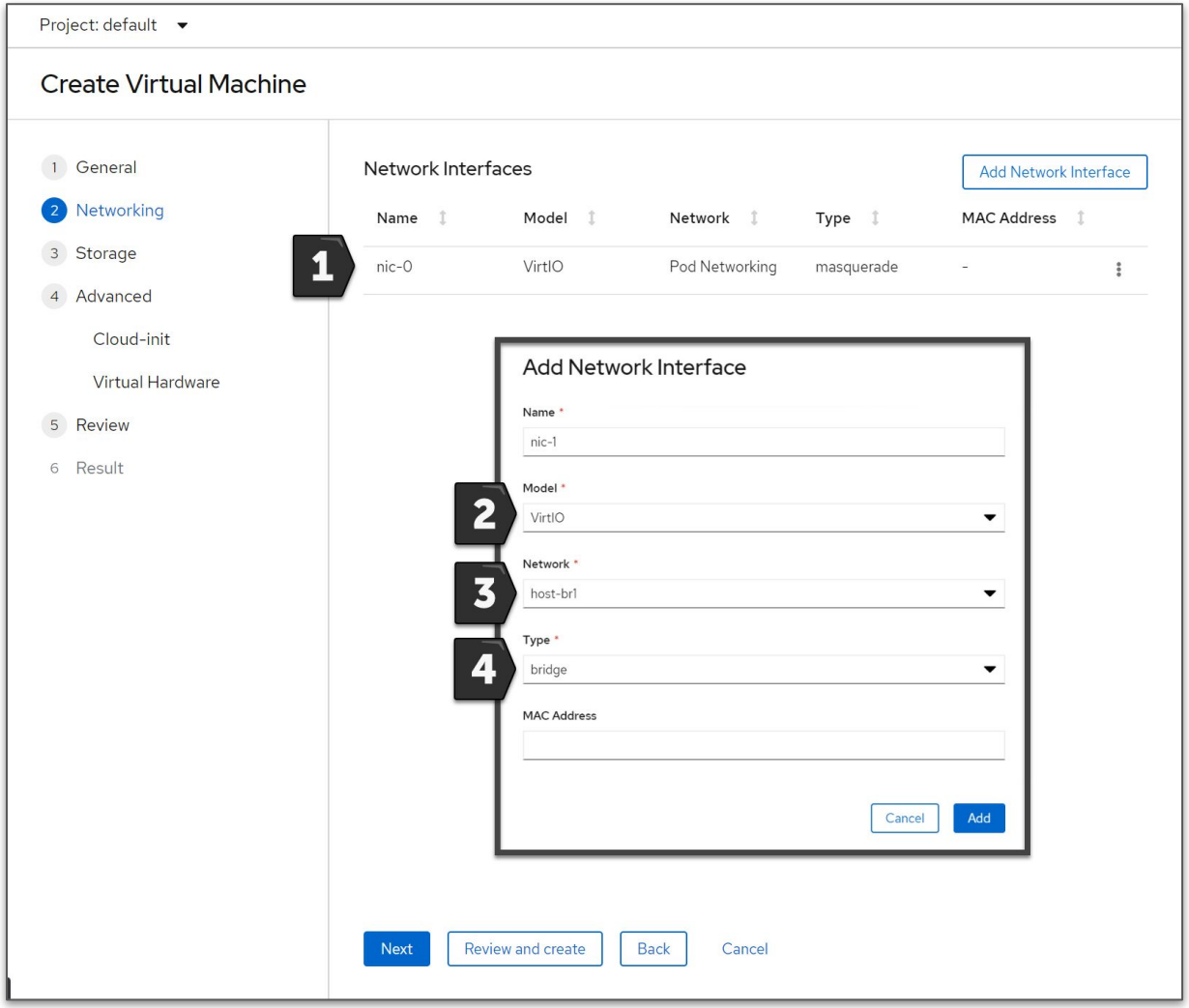
- Source represents how the VM will boot
  - Boot via PXE, optionally diskless
  - URL will import a QCOW2 or raw disk image using a DataVolume
  - Container uses a container image, pulled from a registry, for the disk
  - Disk uses an existing PVC
- Flavor represents the preconfigured CPU and RAM assignments
  - 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

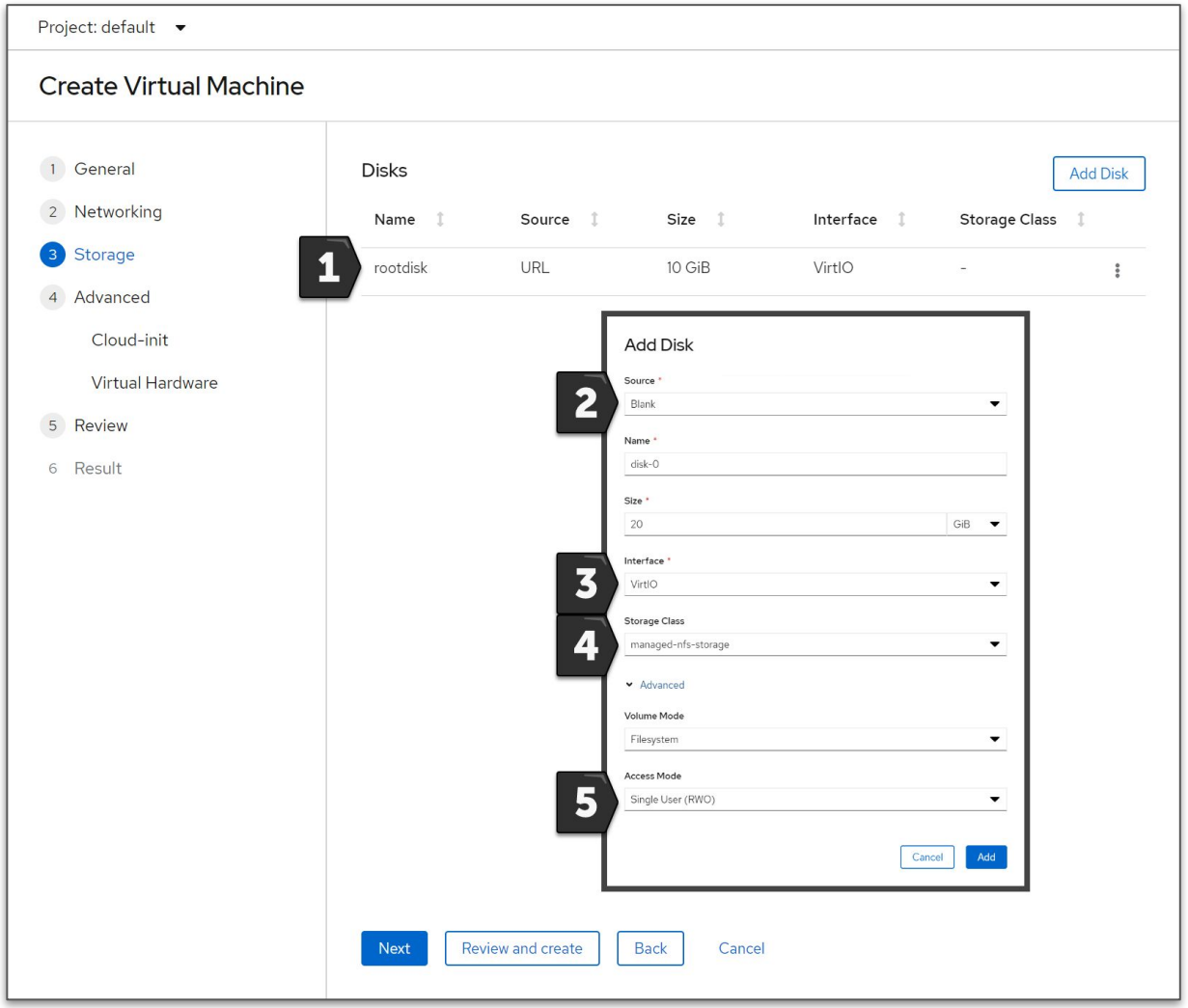
- Add or edit network adapters
- One or more network connections
  - Pod network for the default SDN
  - Additional multus-based interfaces for specific connectivity
- Multiple NIC models for guest OS compatibility or paravirtualized performance with VirtIO
- Masquerade, bridge, or SR-IOV connection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



Red Hat

Create Virtual Machine - Storage

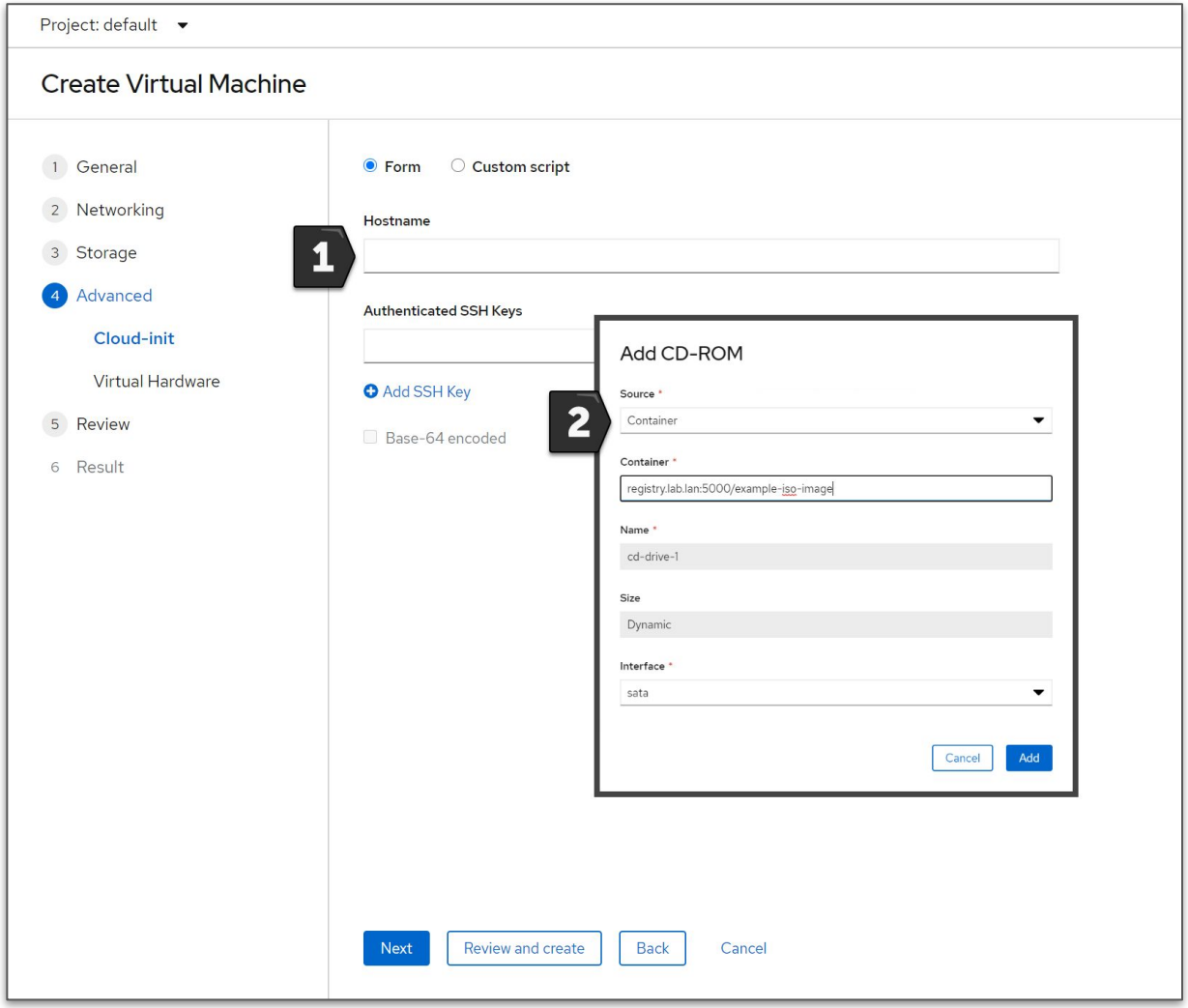
- Add or edit persistent storage
- Disks can be sourced from
  - Imported QCOW2 or raw images
  - New or existing PVCs
  - Clone existing PVCs
- Use SATA/SCSI interface for compatibility or VirtIO for paravirtual performance
- For new or cloned disks, select from available storage classes
  - Customize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



Red Hat

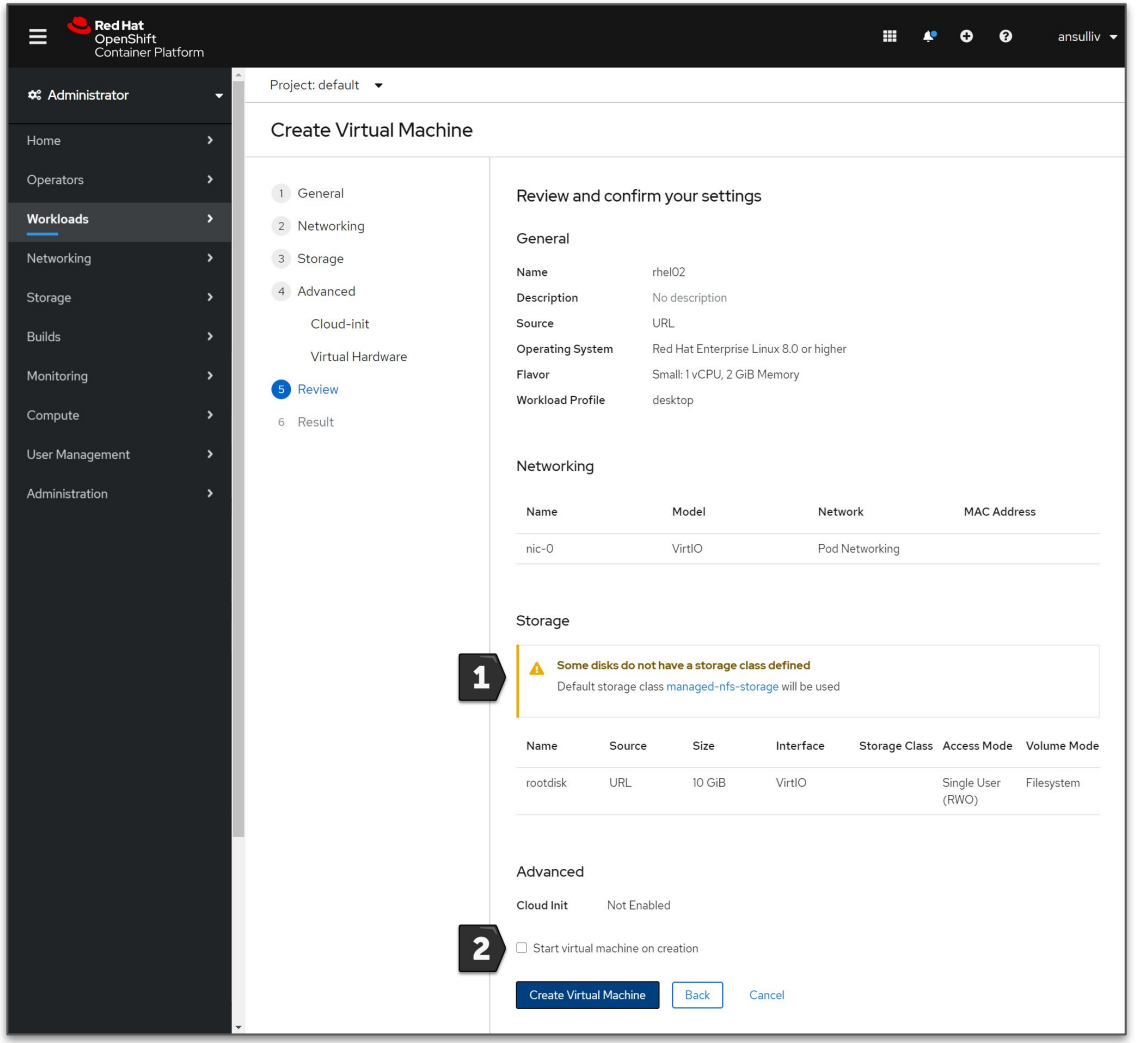
Create Virtual Machine - Advanced

- 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

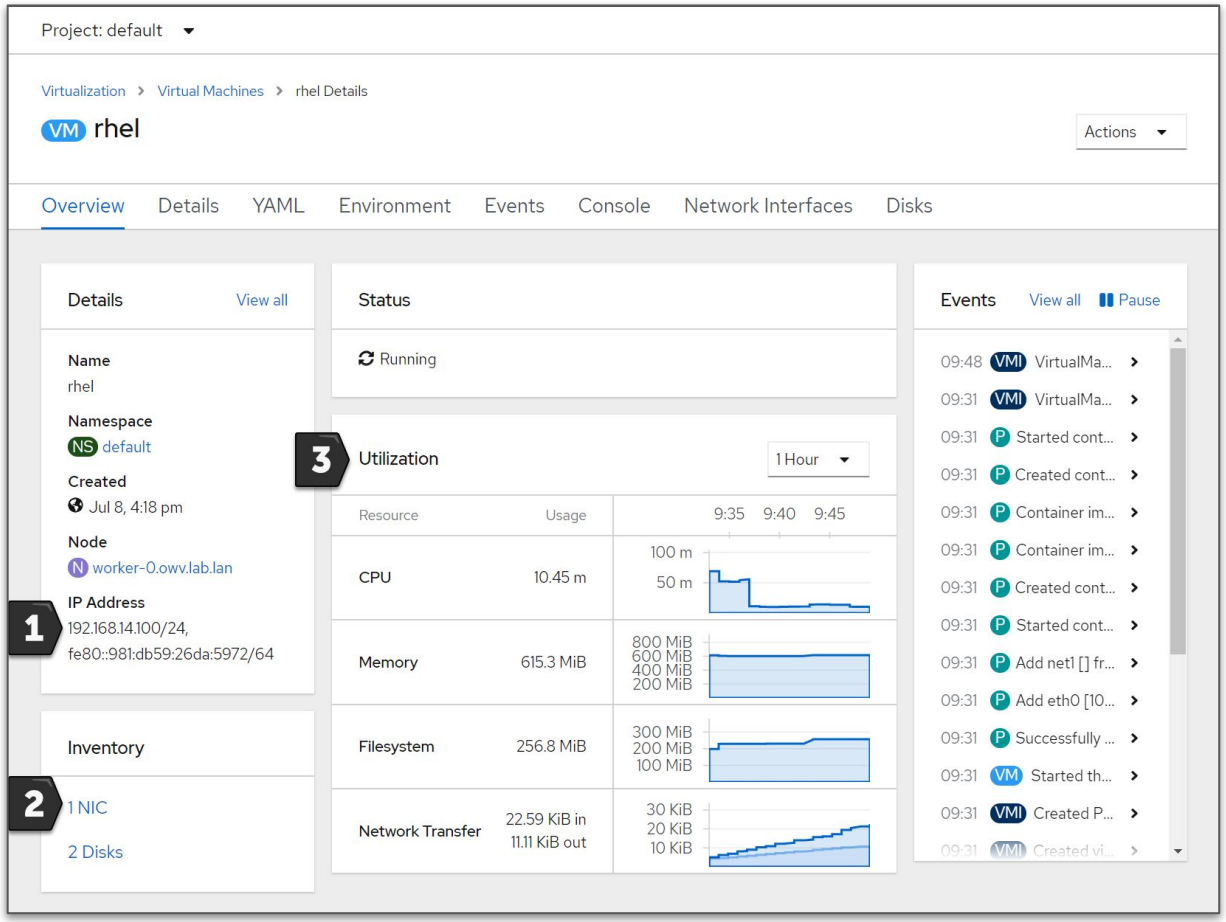
Virtual Machine Import

- 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

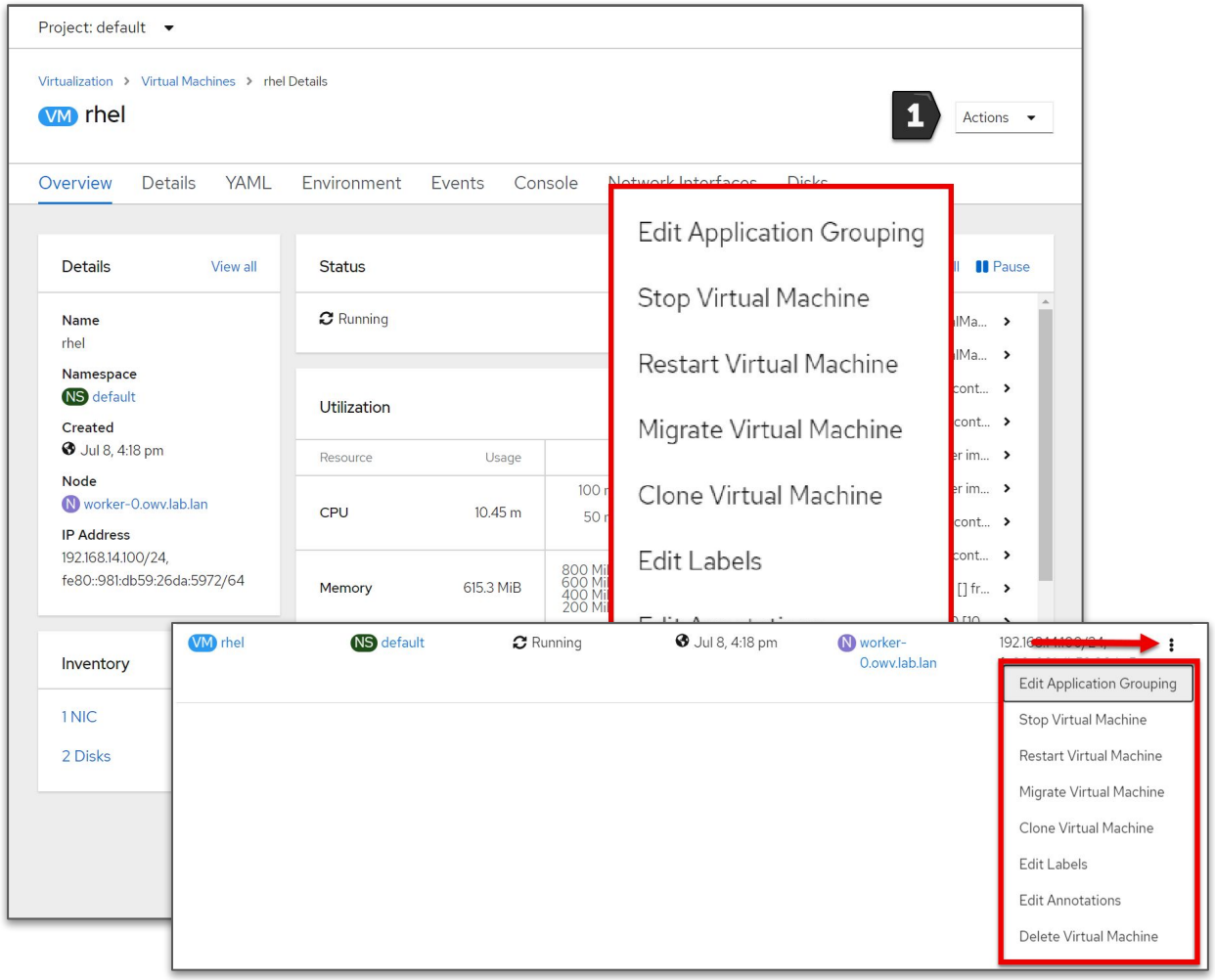
Virtual Machine - Overview

- 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

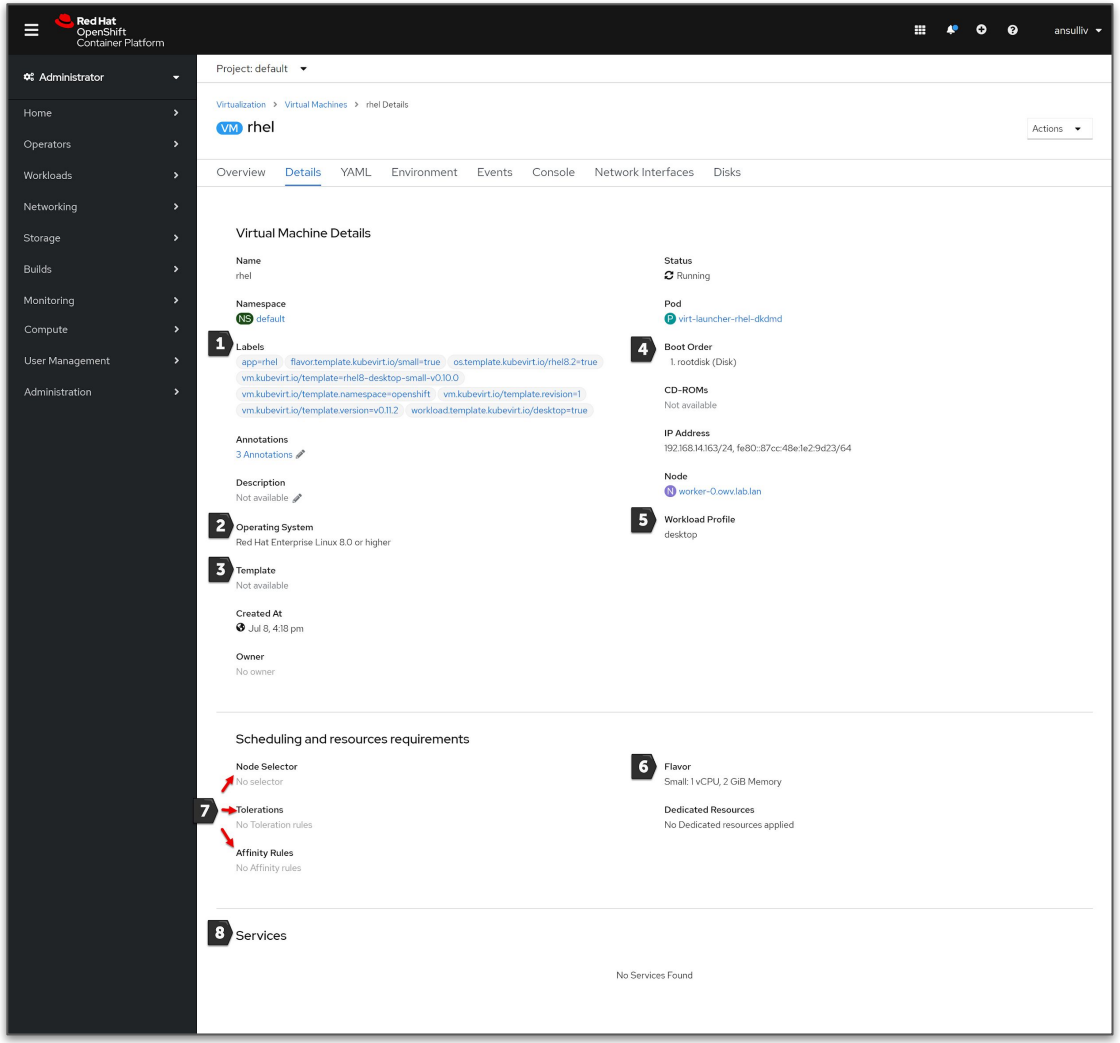
- Actions menu allows quick access to common VM tasks
  - Start/stop/restart
  - Live migration
  - Clone
  - Edit application group, labels, and annotations
  - Delete
- Accessible from all tabs of VM details screen and the VM list



Red Hat

Virtual Machine - Details

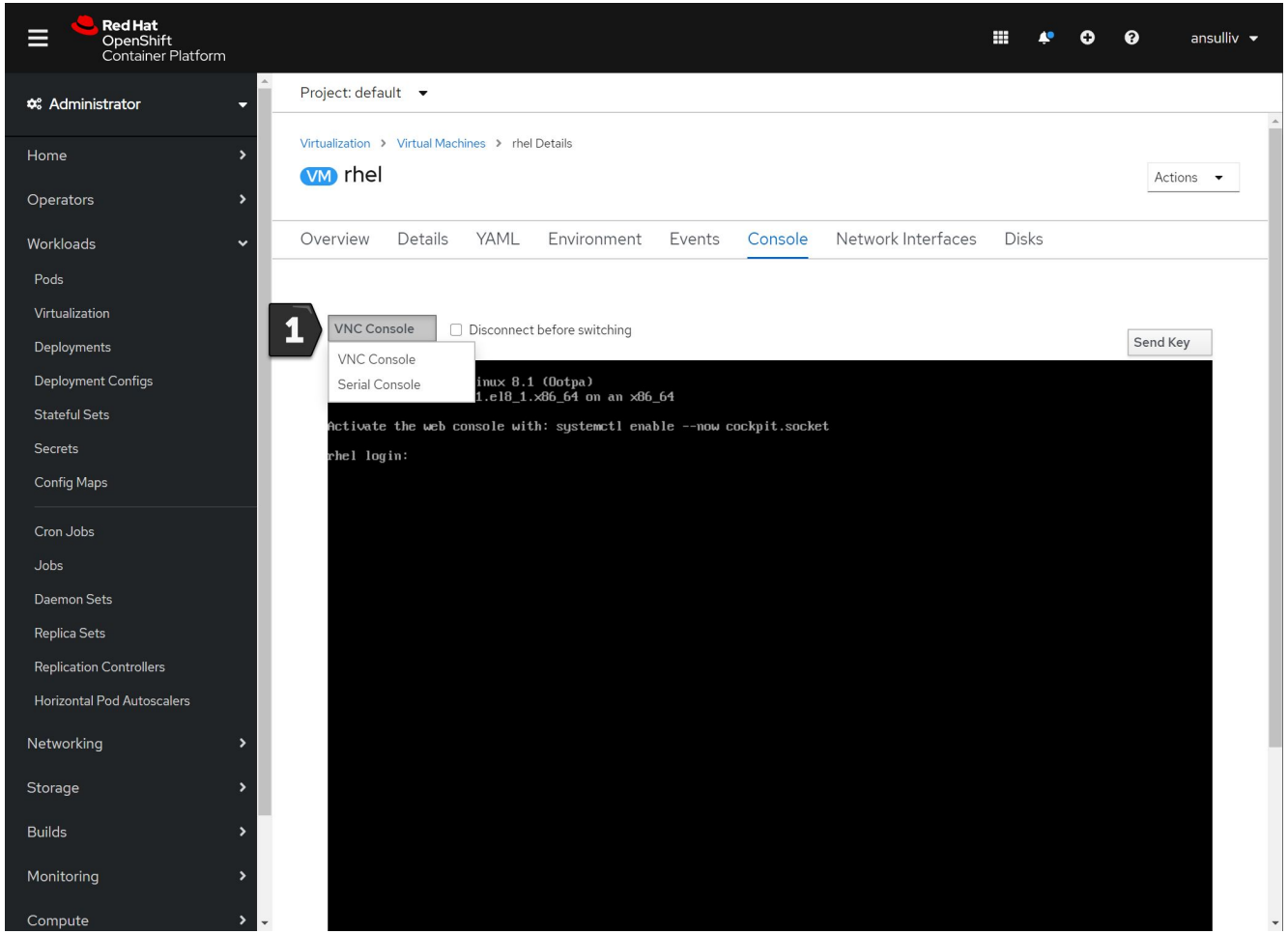
- Details about the virtual machine
  - Labels, annotations
  - Configured OS
  - Template used, if any
  - Configured boot order
  - Associated workload profile
  - Flavor
- Additional details about scheduling
  - Node selector, tolerations, (anti)affinity rules
- Services configured for the VM



Red Hat

Virtual Machine - Console

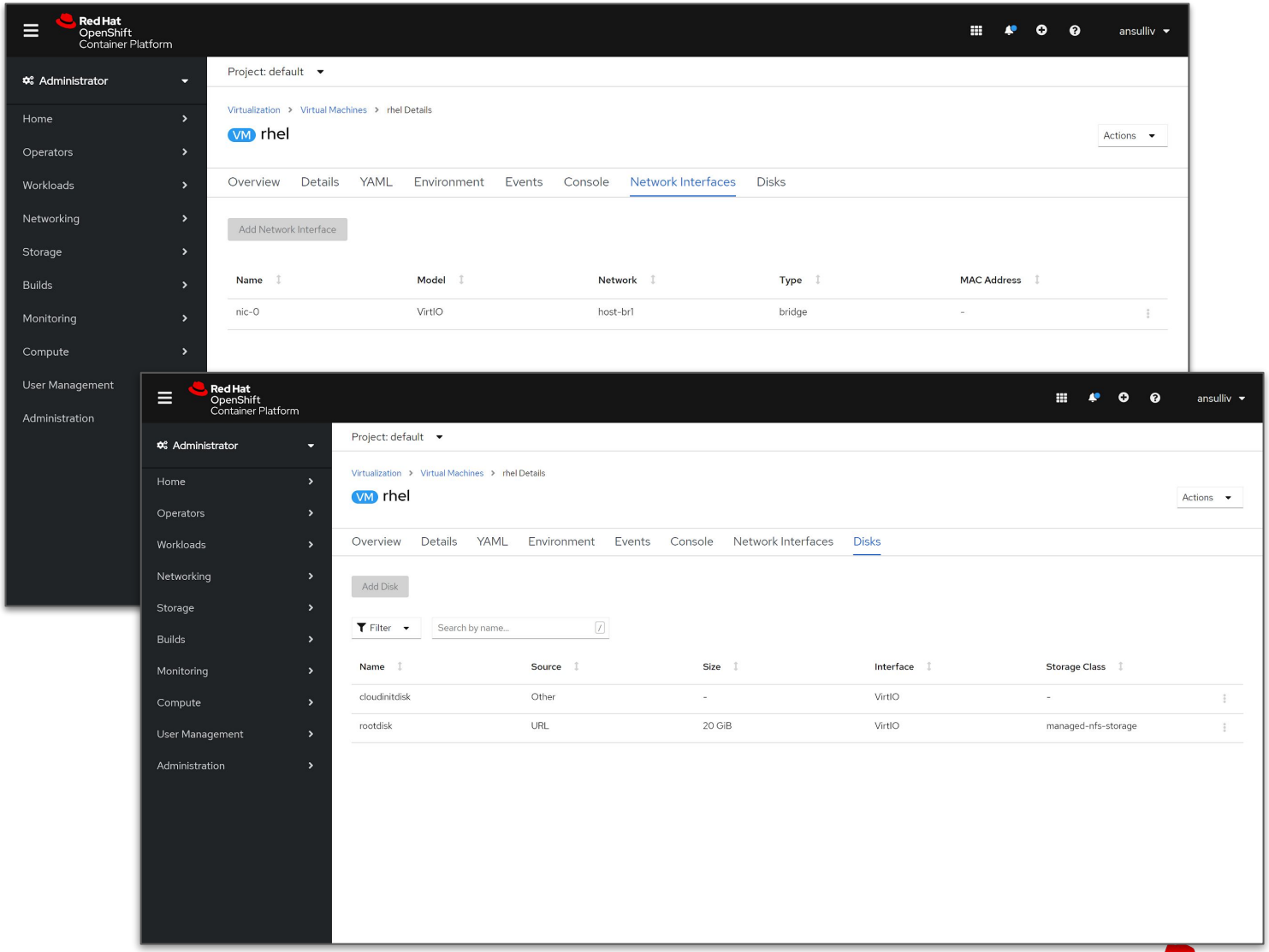
- Browser-based access to the serial and graphical console of the virtual machine
- Access the console using native OS tools, e.g. `virt-viewer`, using the `virtctl` CLI command
  - `virtctl console vmname`
  - `virtctl vnc vmname`



Red Hat

Virtual Machine - Disks and NICs

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

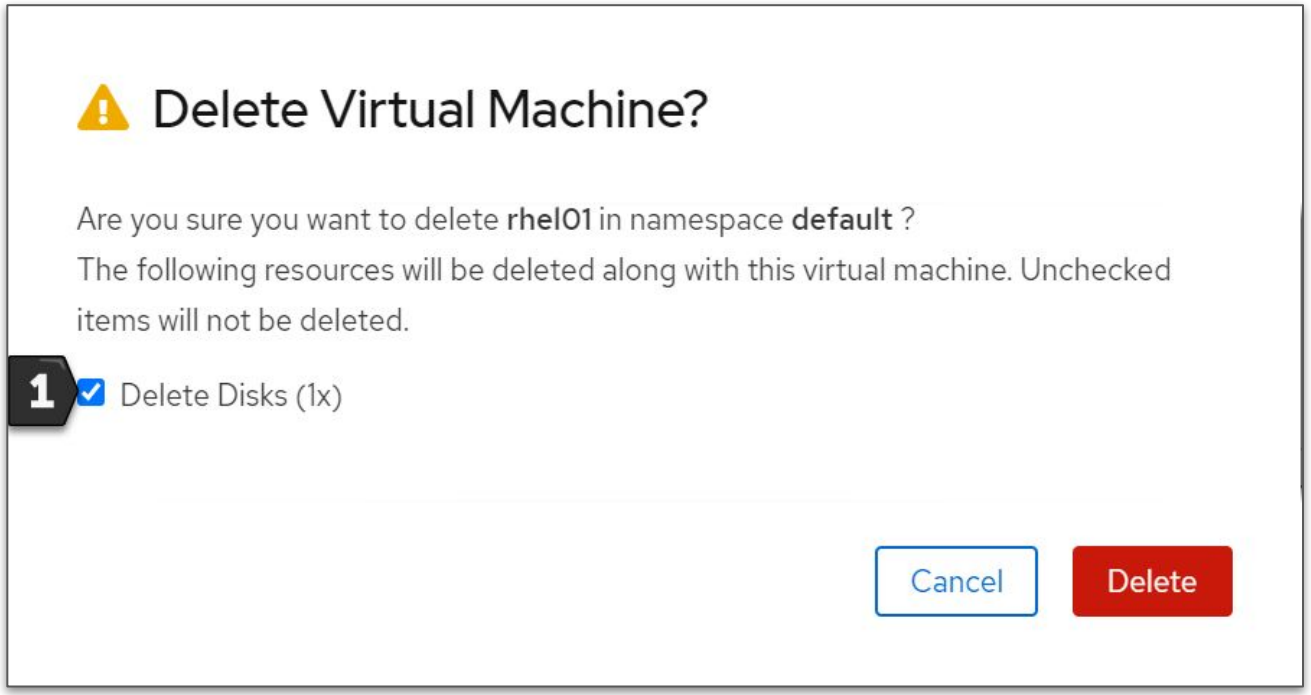


Red Hat

Destroy VMs

Destroying a Virtual Machine

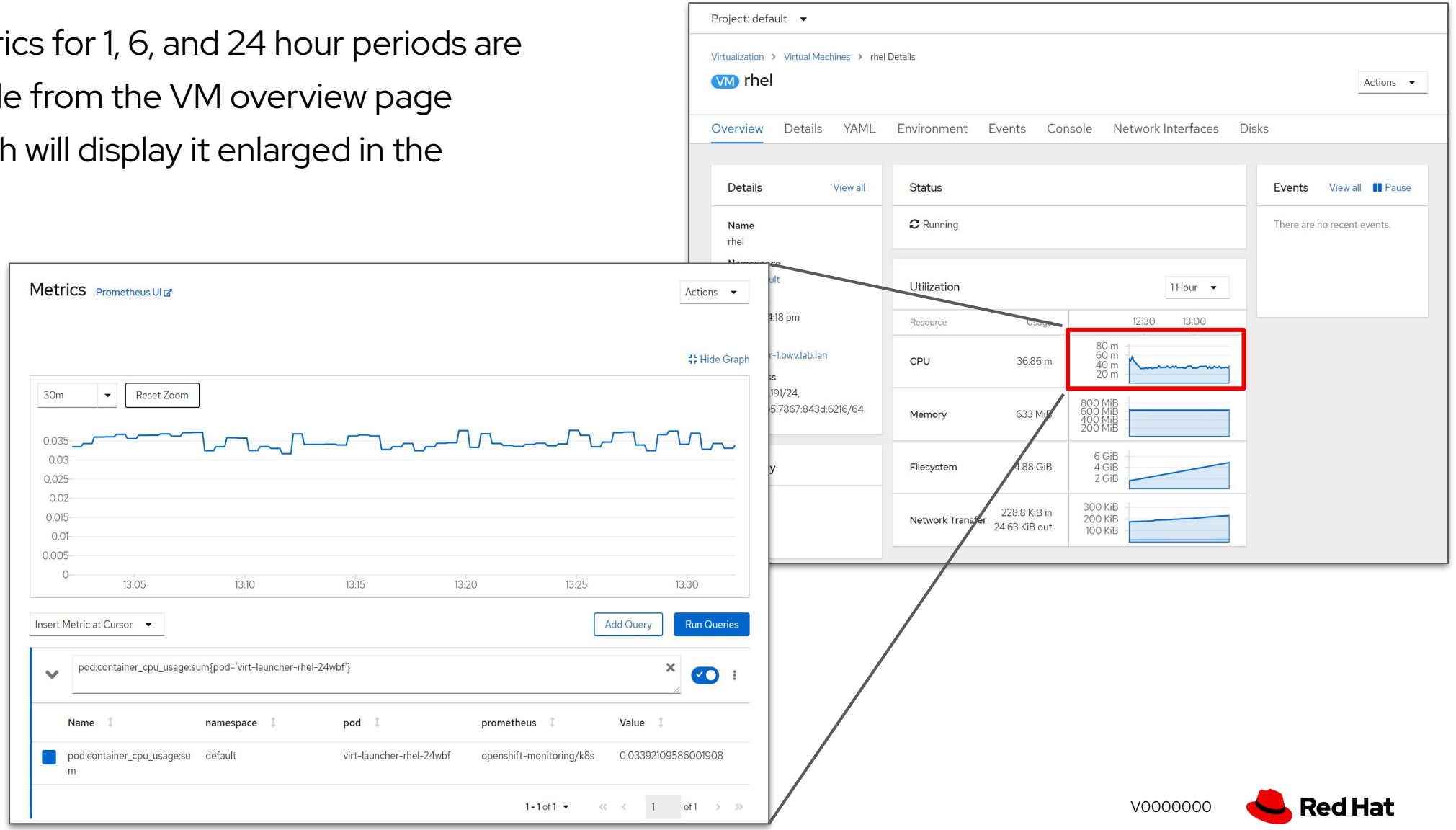
- Deleting a VM removes the VM definition
  - Optionally delete PVC-backed disks associated with the VM
- Running VMs are terminated first
- Other associated resources, e.g. Services, are not affected



Metrics

Overview Virtual Machine metrics

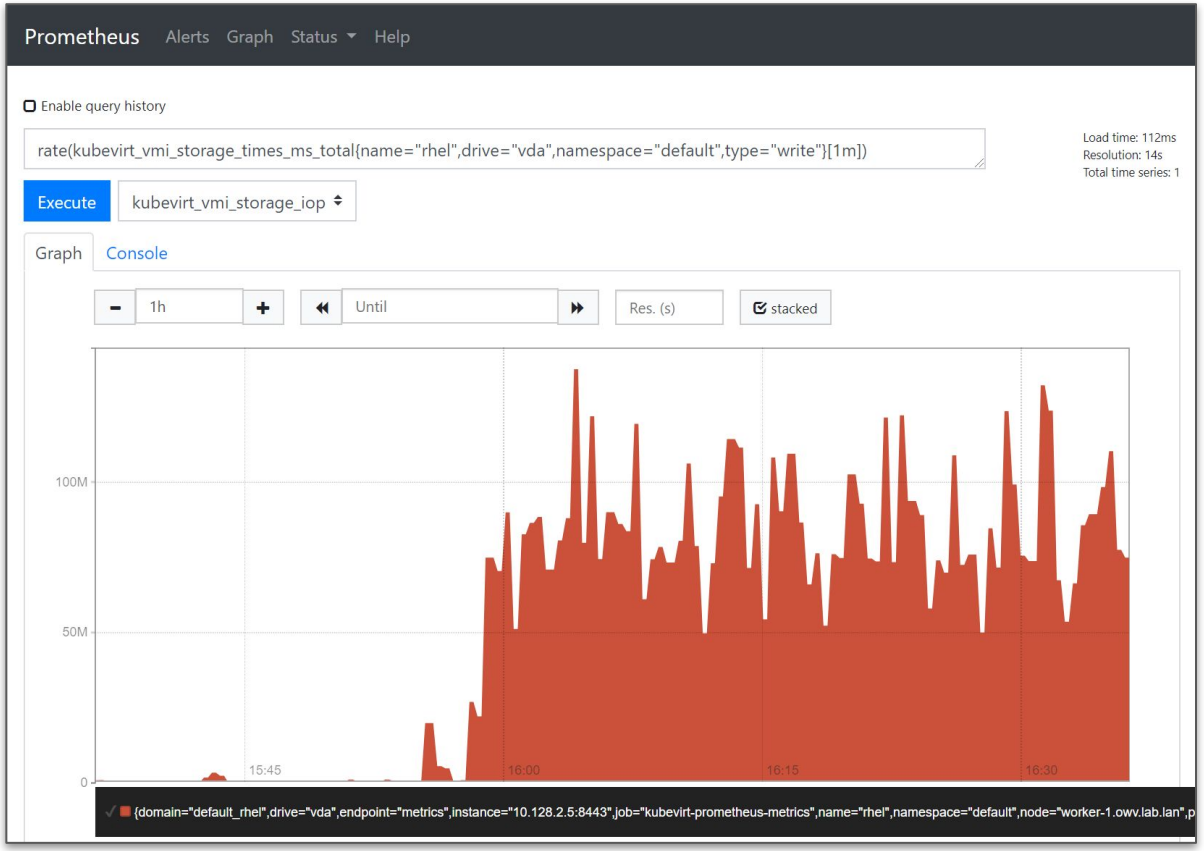
- Summary metrics for 1, 6, and 24 hour periods are quickly viewable from the VM overview page
- Clicking a graph will display it enlarged in the metrics UI



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

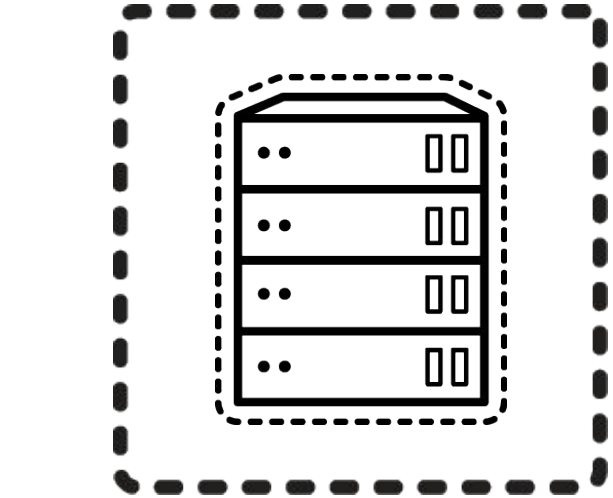


Red Hat

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

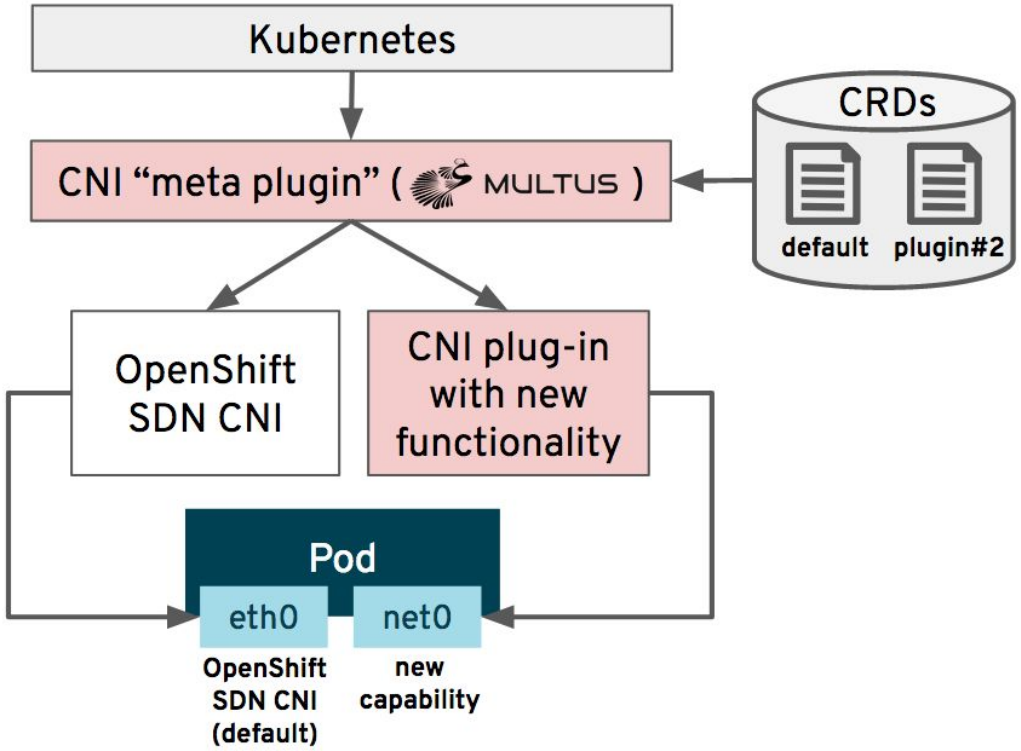


Red Hat OpenShift

Network

Virtual Machine Networking

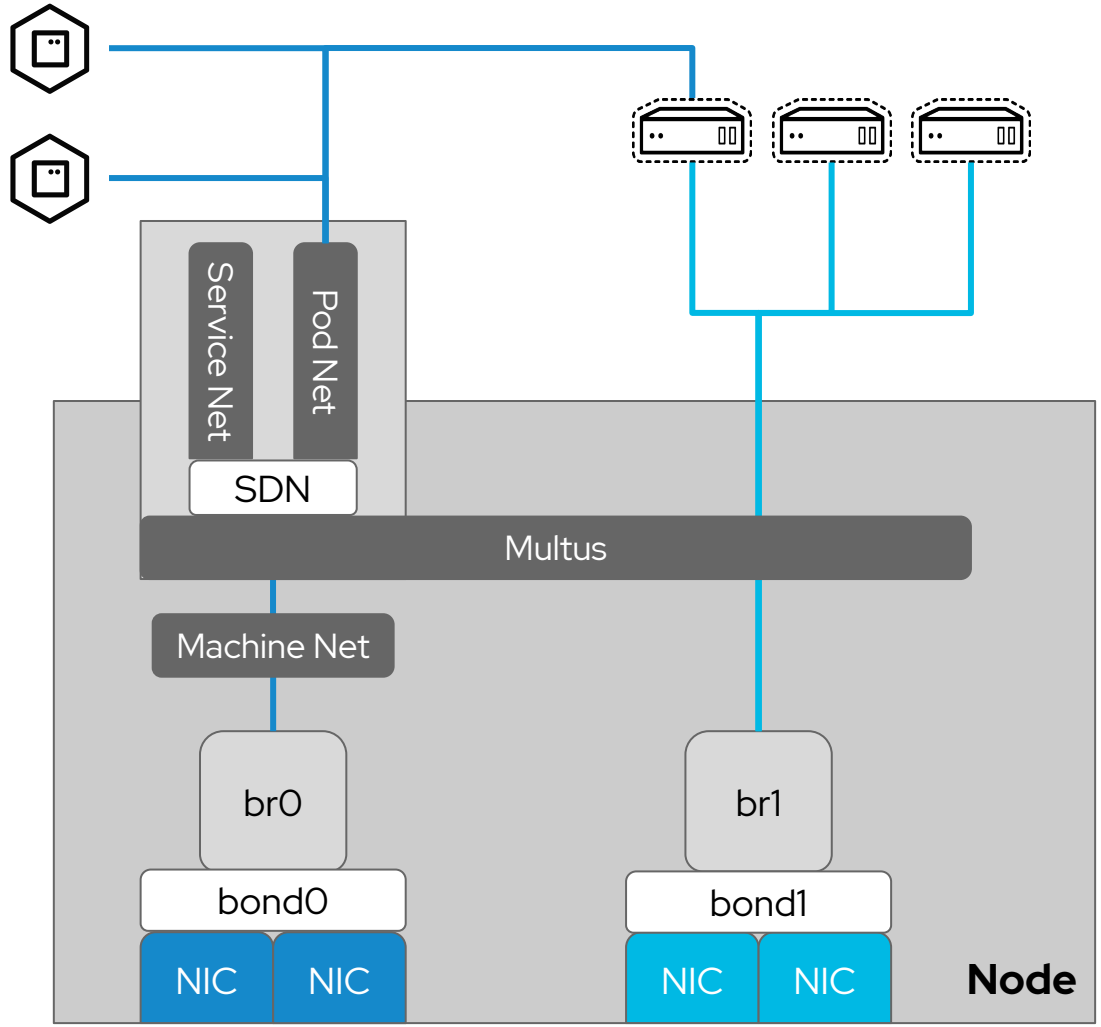
- Virtual machines optionally connect to the standard pod network
  - OpenShift SDN, OVNKubernetes, etc.
- Additional network interfaces accessible via Multus:
  - Bridge, SR-IOV
  - VLAN and other networks can be created using nmstate at the host level
- When using at least one interface on the default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

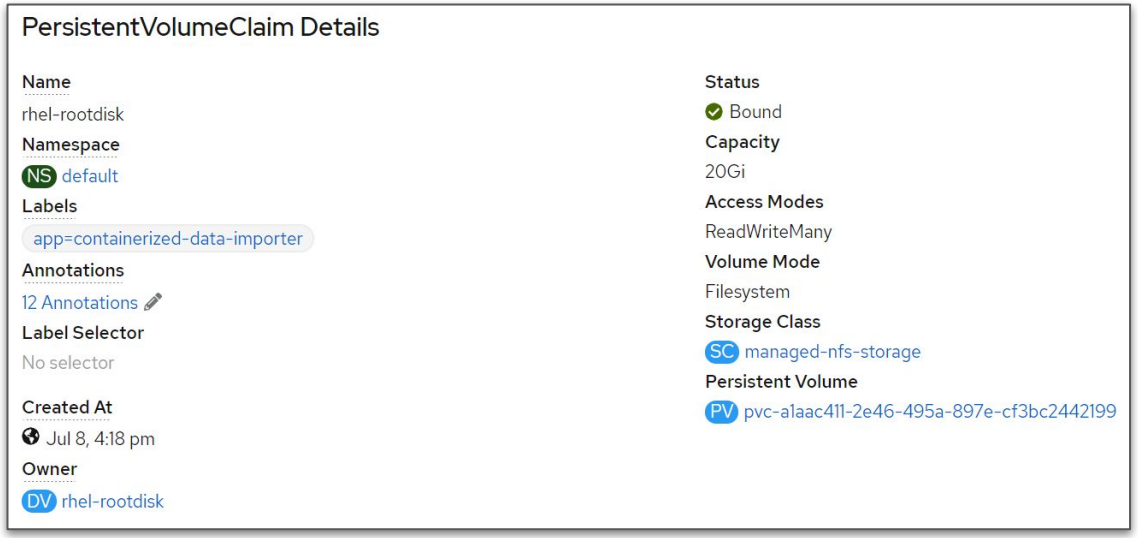
The following slides show an example of how this setup is configured



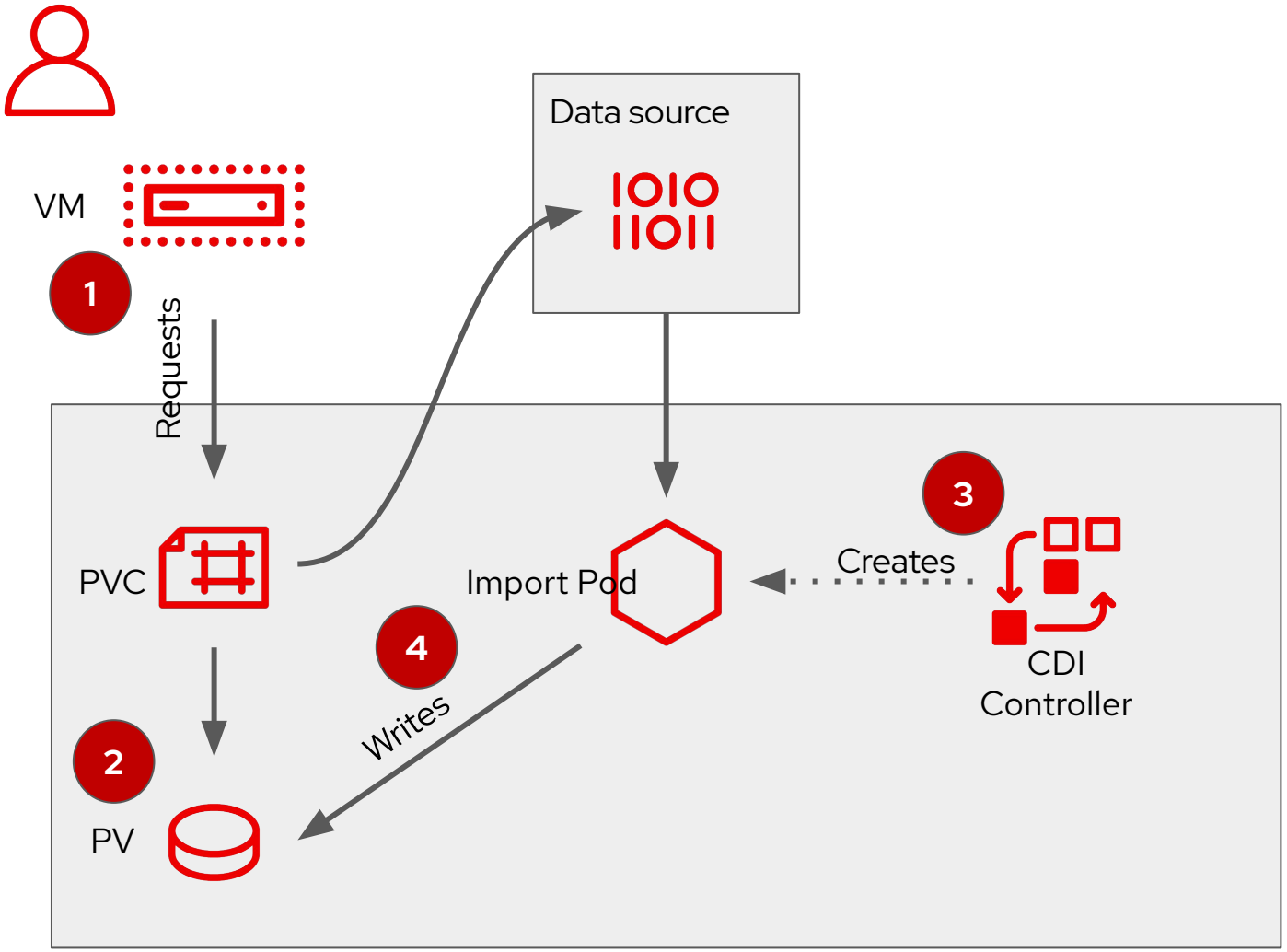
Storage

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



Containerized Data Importer



1. The user creates a virtual machine with a DataVolume
2. The StorageClass is used to satisfy the PVC request
3. The CDI controller creates an importer pod, which mounts the PVC and retrieves the disk image. The image could be sourced from S3, HTTP, or other accessible locations
4. After completing the import, the import pod is destroyed and the PVC is available for the VM

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