

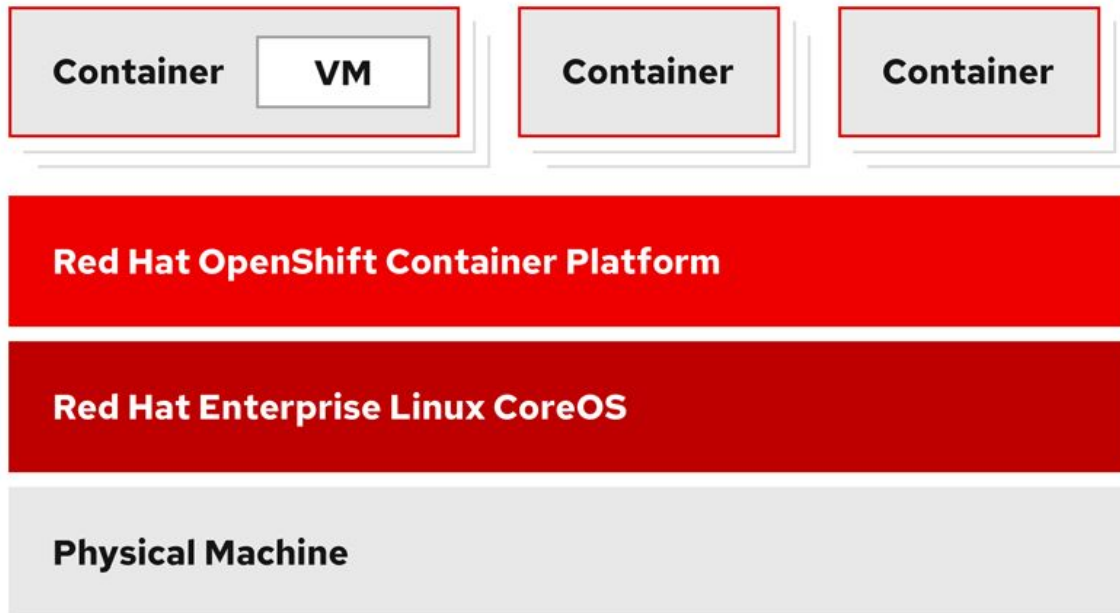


# OpenShift Virtualization

Andrzej Kowalczyk

# Red Hat OpenShift and OpenShift Virtualization: Kubernetes-first innovation to managing VMs

Modernize workloads and support mixed applications consisting of VMs, containers, and serverless



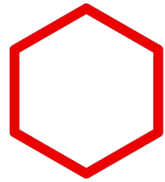
- Accelerate application delivery with a single platform that can manage “mixed applications” with the same tools and teams.
- Add VMs to new and existing applications.
- Modernize legacy VM applications over time, or maintain them as VMs.

# It is about managing both VMs and containers



## Virtual machines

VMs have been built for decades, and they will not go away overnight.



## Containers

Containers solve certain use cases and will continue to rise, but some VMs will remain.



## Applications

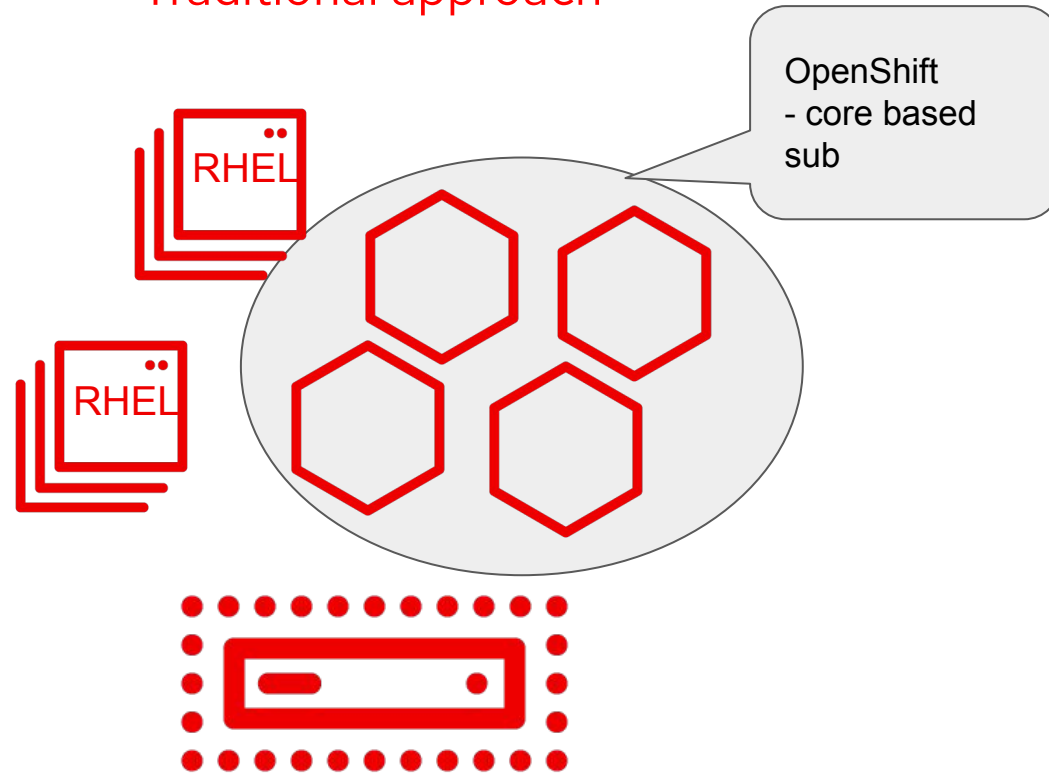
VMs and containers will be used to build applications, and some might even be built on both.



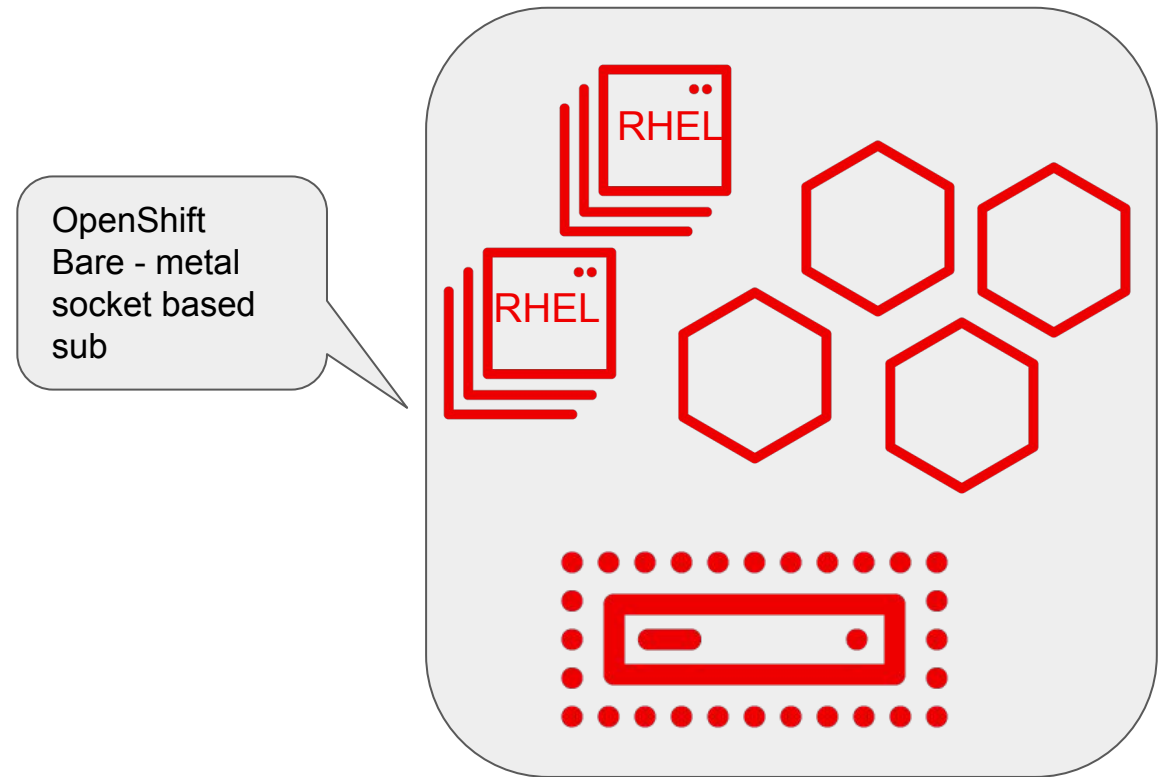
Single platform for any load

# It is about managing both VMs and containers

## Traditional approach



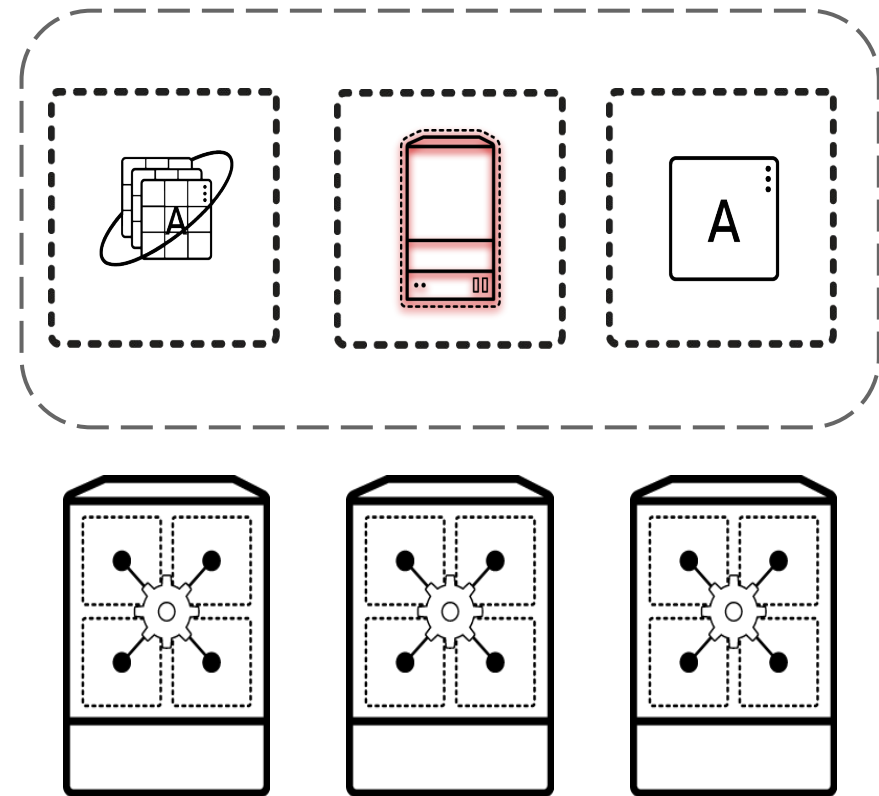
## Modern approach



# What is OpenShift Virtualization?

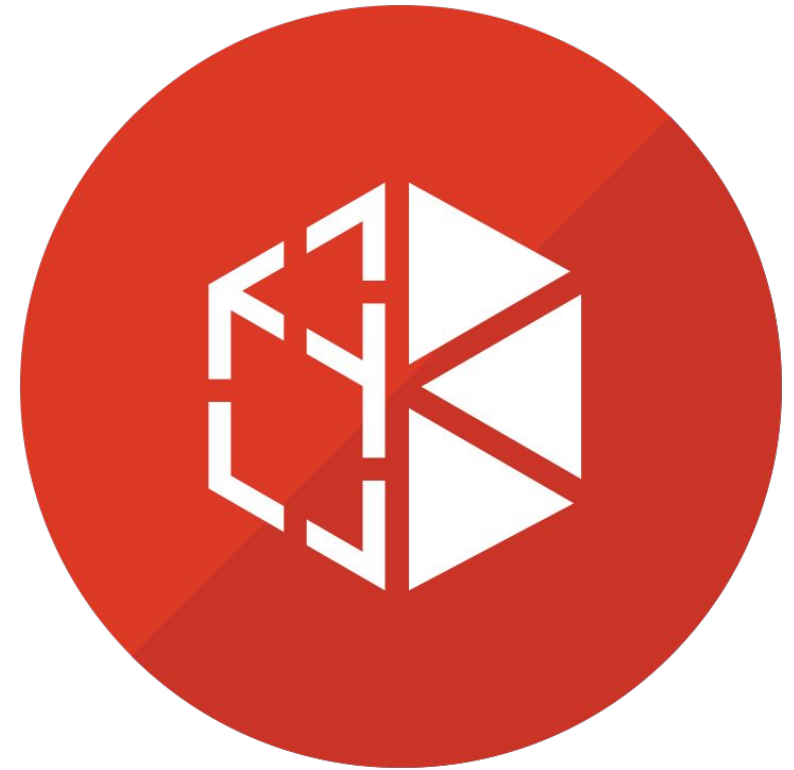
# Virtual machines can be put into containers

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



# OpenShift Virtualization

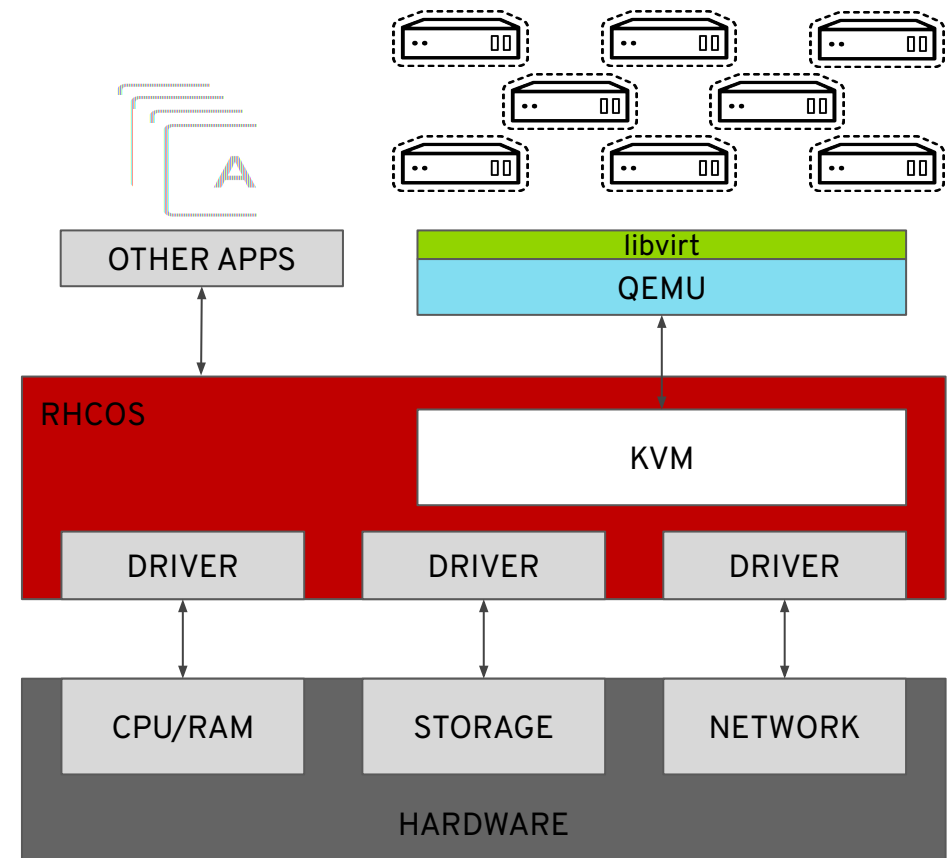
- Virtual machines
  - Running in containers
  - Using the KVM hypervisor
- Scheduled, deployed, and managed by Kubernetes
- Integrated with container orchestrator resources and services
  - Traditional Pod-like SDN connectivity and/or connectivity to external VLAN and other networks via multus
  - Persistent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





# VM containers use KVM

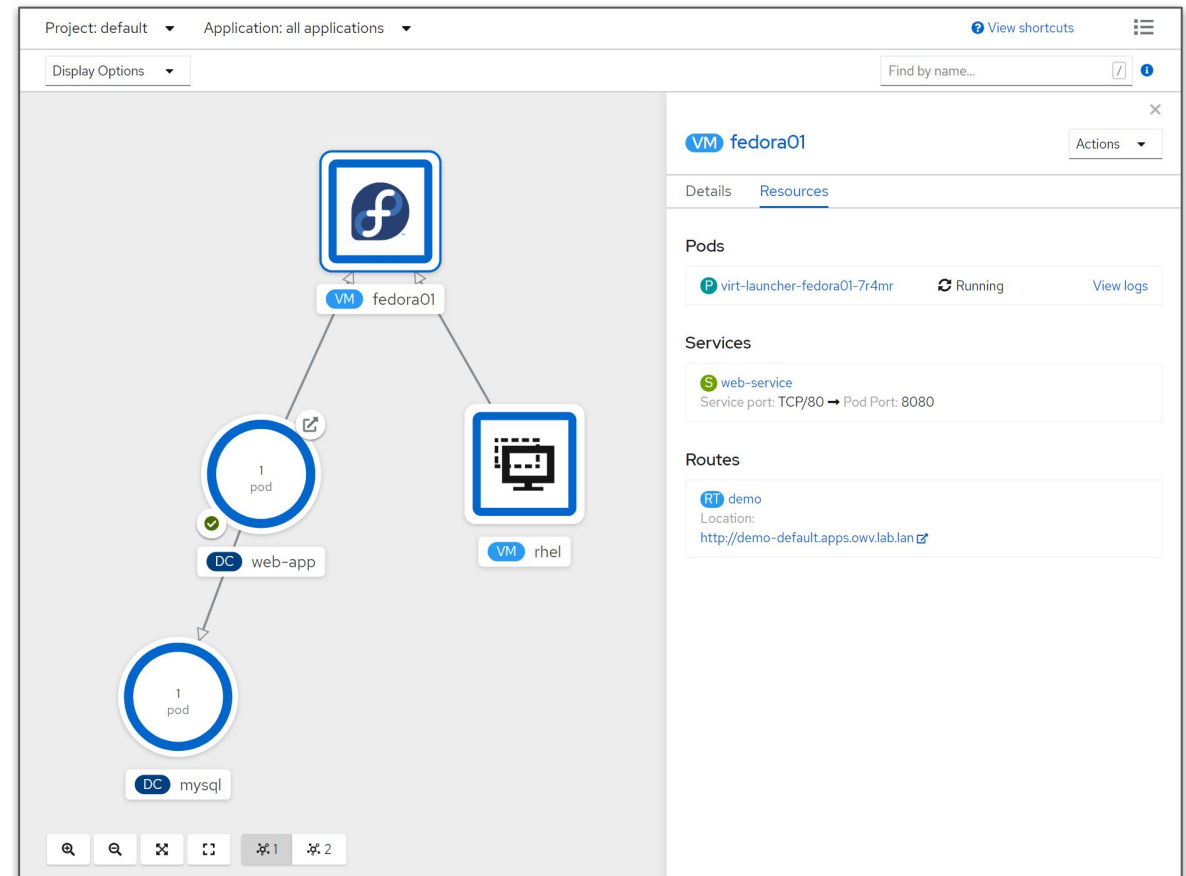
- OpenShift Virtualization uses KVM, the Linux kernel hypervisor
- KVM is a core component of the Red Hat Enterprise Linux kernel
  - KVM has 10+ years of production use: Red Hat Virtualization, Red Hat OpenStack Platform, and RHEL all leverage KVM, QEMU, and libvirt
- QEMU uses KVM to execute virtual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



# Built with Kubernetes

# Using VMs and containers together

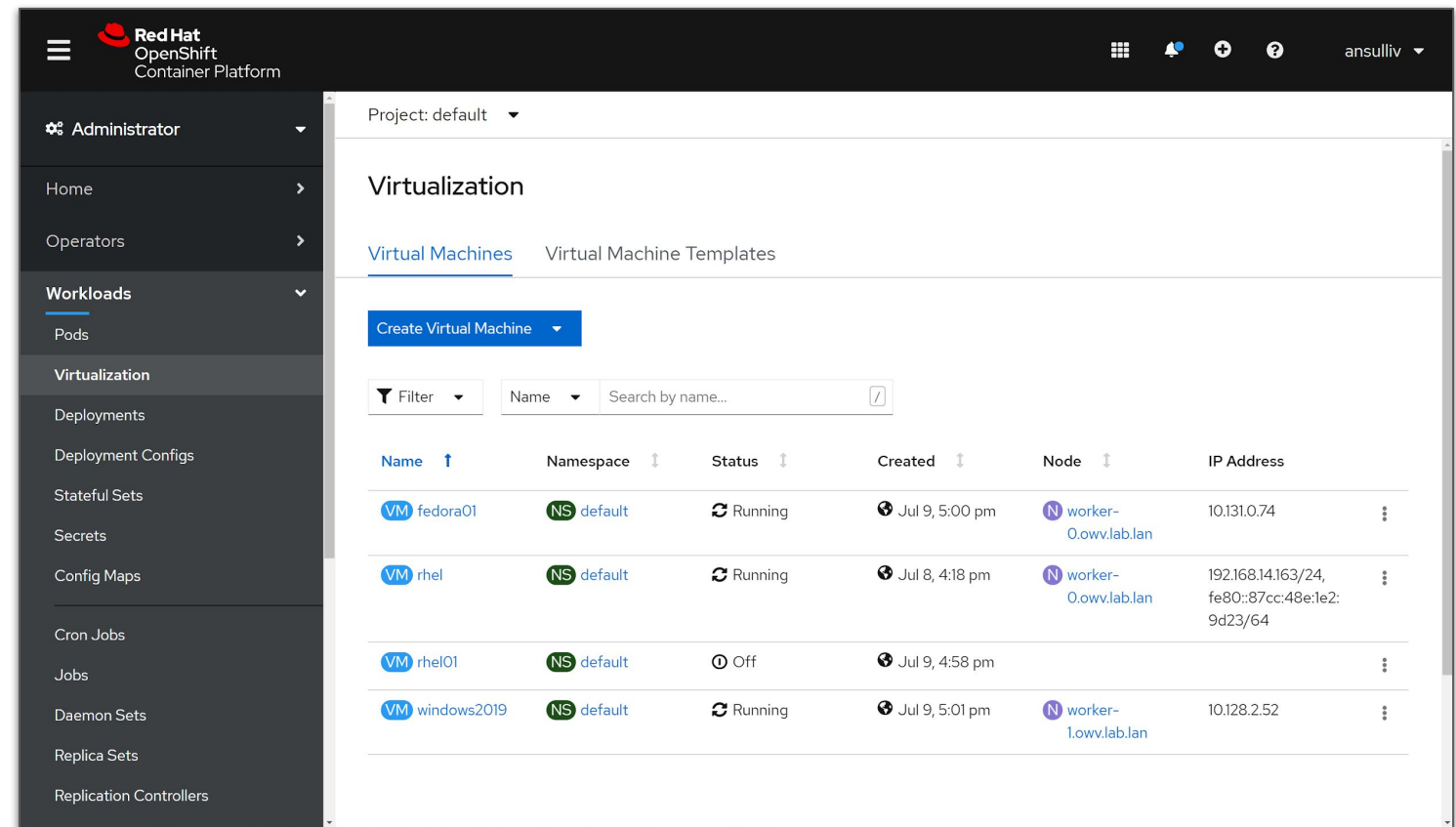
- Virtual Machines connected to pod networks are accessible using standard Kubernetes methods:
  - Service
  - Route
  - Ingress
- Network policies apply to VM pods the same as application pods
- VM-to-pod, and vice-versa, communication happens over SDN or ingress depending on network connectivity



# Managed with OpenShift

# Virtual Machine Management

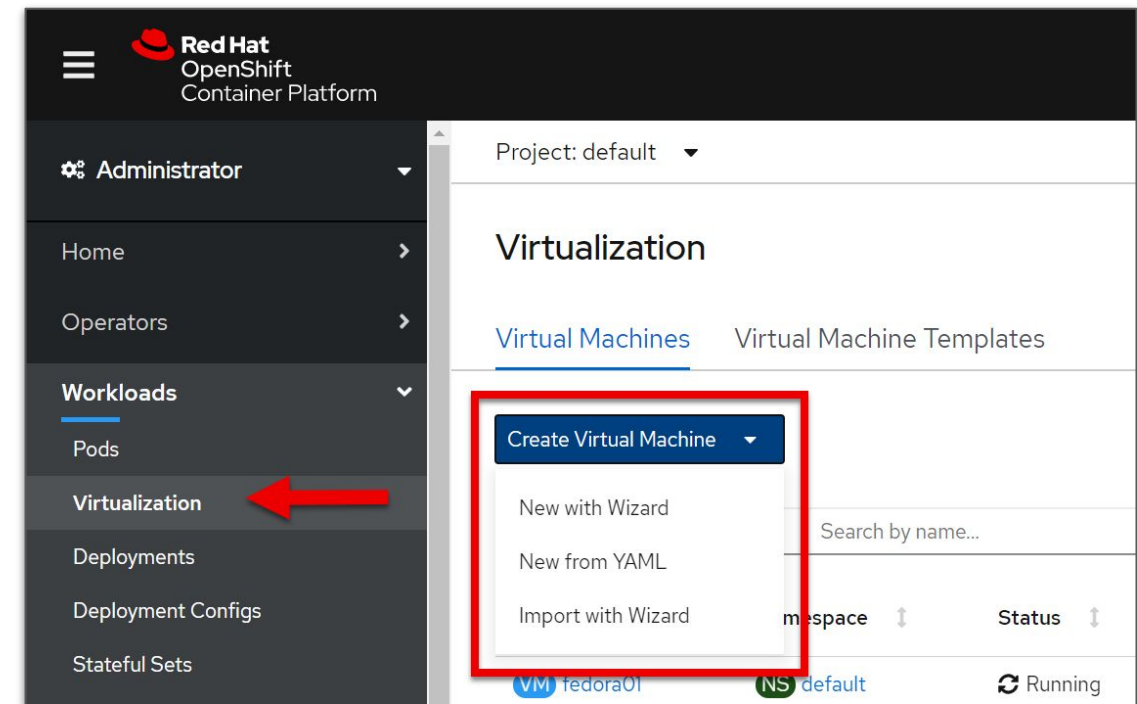
- Create, modify, and destroy virtual machines, and their resources, using the OpenShift web interface or CLI
- Use the `virtctl` command to simplify virtual machine interaction from the CLI



# 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' form in OpenStack. The 'General' tab is selected, and the form includes fields for Name, Description, Template, Source, Flavor, and Workload Profile. Three numbered callouts highlight the Source, Flavor, and Workload Profile dropdown menus.

Project: default ▼

### Create Virtual Machine

1 General  
2 Networking  
3 Storage  
4 Advanced  
Cloud-init  
Virtual Hardware  
5 Review  
6 Result

Name \*

Description

Template  
No template available ▼

Source \*

1 --- Select Source ---  
--- Select Source ---  
PXE  
URL  
Container  
Disk

Flavor \* ?

2 Custom  
--- Select Flavor ---  
Tiny  
Small  
Medium  
Large  
Custom

Workload Profile \* ?

3 --- Select Workload Profile ---  
--- Select Workload Profile ---  
desktop  
highperformance  
server



# 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

Project: default ▾

## Create Virtual Machine

1 General

2 Networking

3 Storage

4 Advanced

- Cloud-init
- Virtual Hardware

5 Review

6 Result

Network Interfaces Add Network Interface

Name ▴ ▾	Model ▴ ▾	Network ▴ ▾	Type ▴ ▾	MAC Address ▴ ▾	
nic-0	VirtIO	Pod Networking	masquerade	-	⋮

1

2

3

4

Add Network Interface

Name \*  
nic-1

Model \*  
VirtIO ▾

Network \*  
host-br1 ▾

Type \*  
bridge ▾

MAC Address

Cancel Add

Next

Review and create

Back

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

Project: default ▼

## Create Virtual Machine

1 General

2 Networking

3 Storage

4 Advanced

Cloud-init

Virtual Hardware

5 Review

6 Result

Disks

1

Name	Source	Size	Interface	Storage Class
rootdisk	URL	10 GiB	VirtIO	-

2

3

4

5

Add Disk

Source \*  
Blank

Name \*  
disk-0

Size \*  
20 GiB

Interface \*  
VirtIO

Storage Class  
managed-nfs-storage

Advanced

Volume Mode  
Filesystem

Access Mode  
Single User (RWO)

Cancel Add

Next

Review and create

Back

Cancel

# 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

Project: default ▾

## Create Virtual Machine

☒ Form ☐ Custom script

1 Hostname

Authenticated SSH Keys

[+ Add SSH Key](#)

☐ Base-64 encoded

### Add CD-ROM

Source \*  
Container ▾

Container \*

Name \*

Size

Interface \*  
sata ▾

2

Next Review and create Back Cancel

# 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 OpenShift Container Platform

Project: default

## Create Virtual Machine

1 General  
2 Networking  
3 Storage  
4 Advanced  
5 Review  
6 Result

Cloud-init  
Virtual Hardware

### Review and confirm your settings

#### General

Name	rhel02
Description	No description
Source	URL
Operating System	Red Hat Enterprise Linux 8.0 or higher
Flavor	Small: 1 vCPU, 2 GiB Memory
Workload Profile	desktop

#### Networking

Name	Model	Network	MAC Address
nic-0	VirtIO	Pod Networking	

#### Storage

**1** ⚠ Some disks do not have a storage class defined  
Default storage class `managed-nfs-storage` will be used

Name	Source	Size	Interface	Storage Class	Access Mode	Volume Mode
rootdisk	URL	10 GiB	VirtIO		Single User (RWO)	Filesystem

#### Advanced

Cloud Init Not Enabled

**2** ☐ Start virtual machine on creation

Create Virtual Machine Back Cancel

# Import VMs

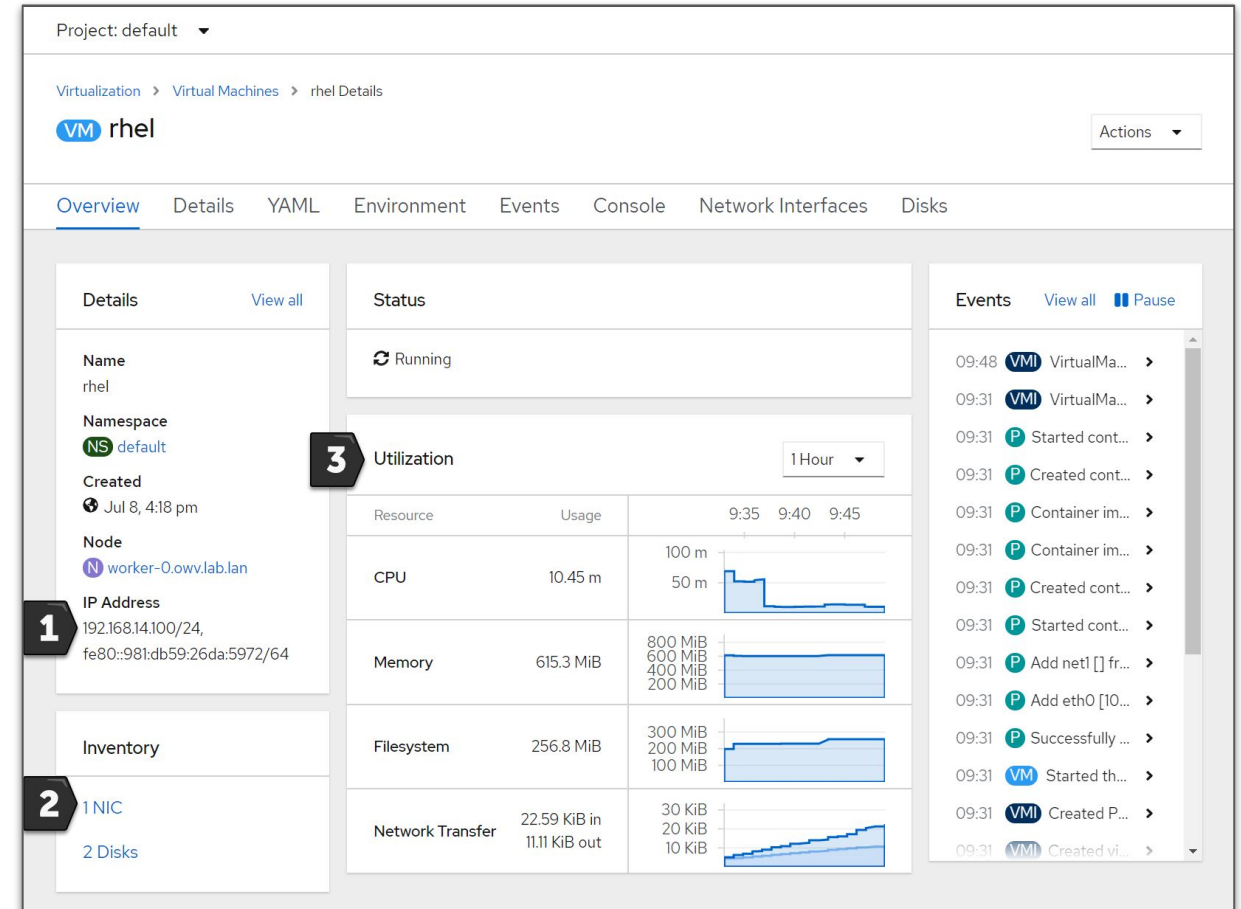
# Virtual Machine Import

- 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

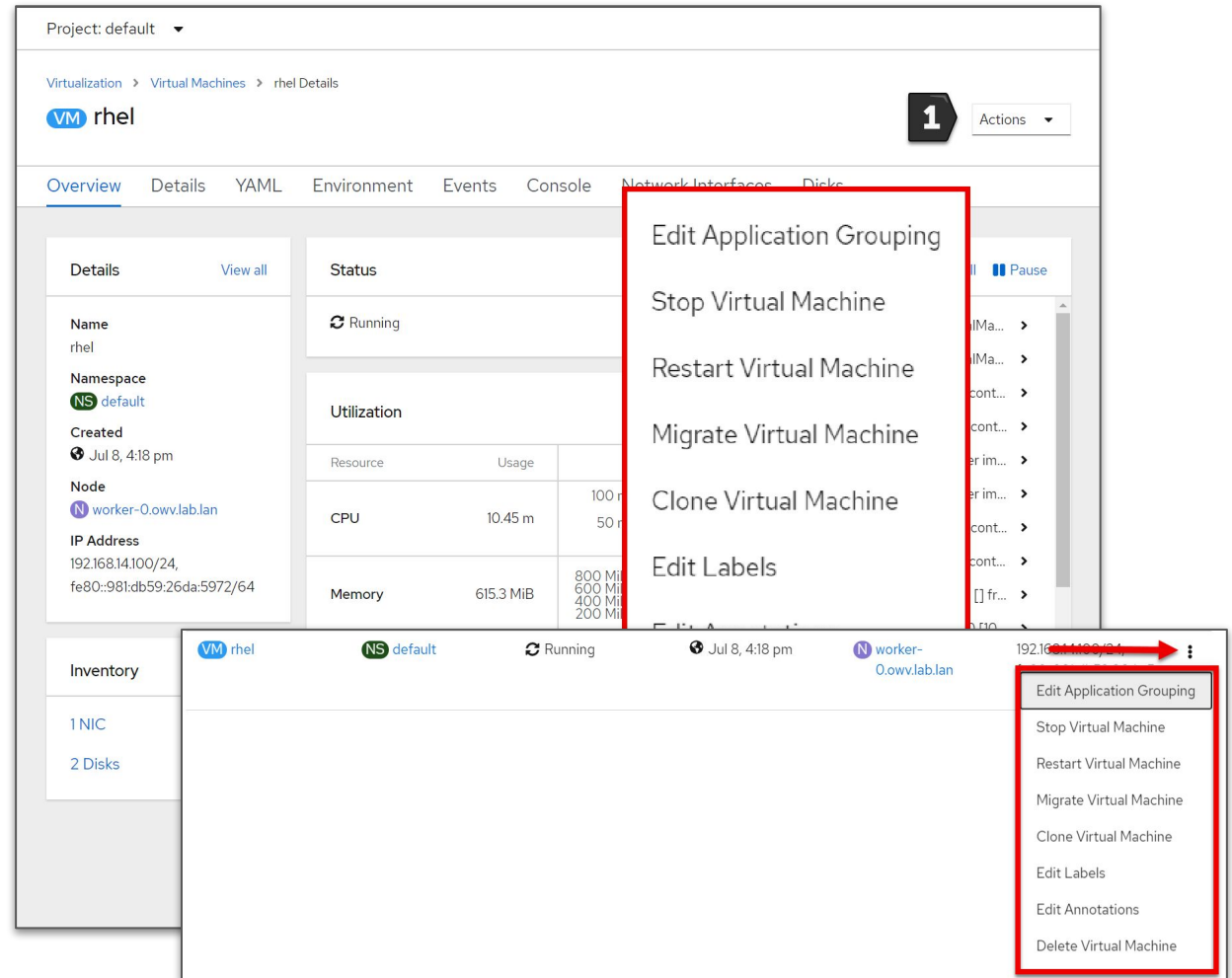
- General overview about the virtual machine
- Information populated from guest when integrations are available
  - IP address
- Inventory quickly shows configured hardware with access to view/manage
- Utilization reporting for CPU, RAM, disk, and network





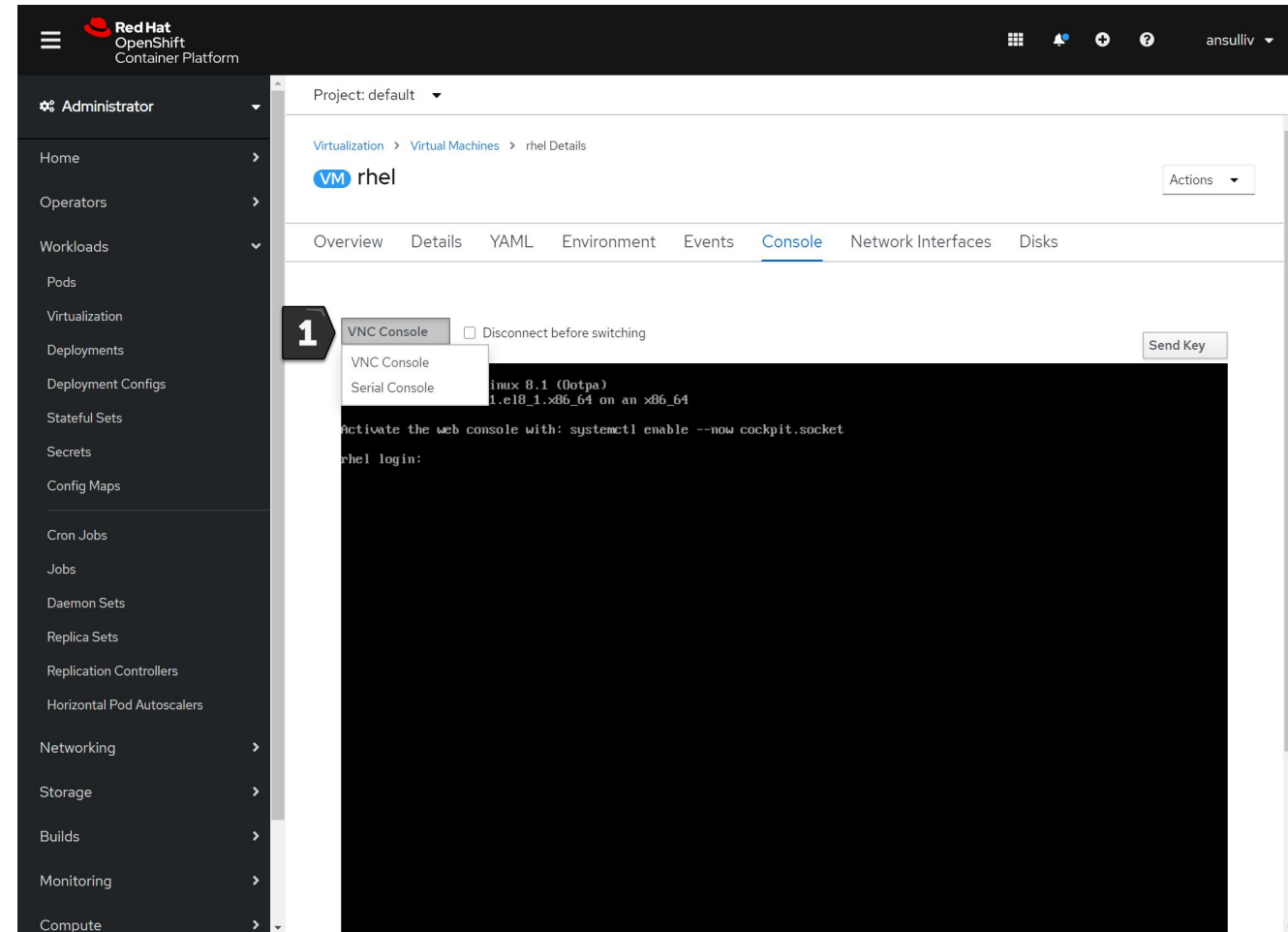
# Virtual Machine – Actions

- 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



# Virtual Machine – Console

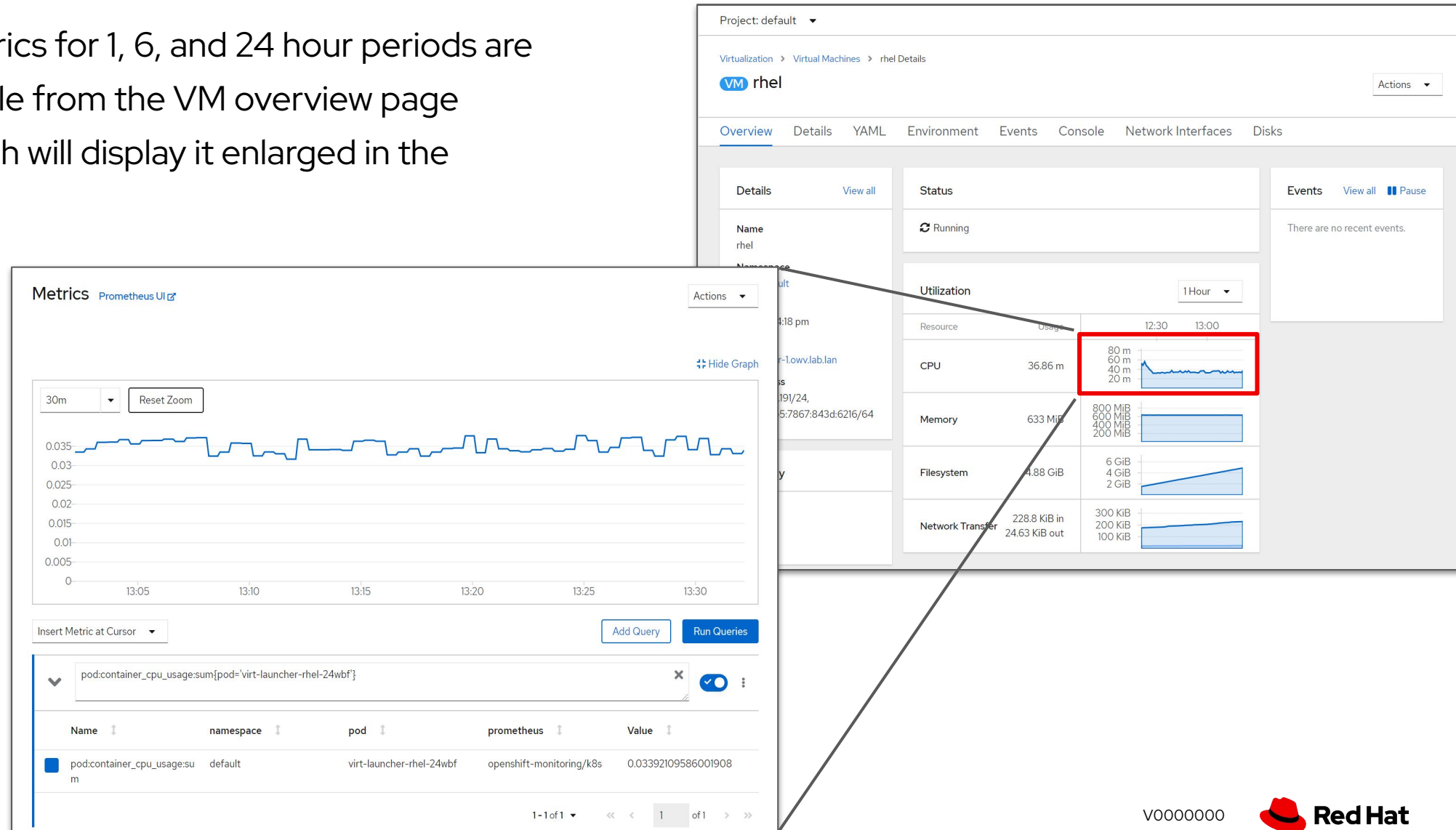
- Browser-based access to the serial and graphical console of the virtual machine
- Access the console using native OS tools, e.g. `virt-viewer`, using the `virtctl` CLI command
  - `virtctl console vmname`
  - `virtctl vnc vmname`



# 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



# 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



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

# Why Red Hat OpenShift?

Modernize and simplify your datacenter

## Consistency of management

With OpenShift support for VMs, containers, and serverless, you can align your DevOps team on a simpler architecture to manage

## Save on cost and innovate

Keep the VMs and leverage the scale advantages of Kubernetes. Apply the cost savings to fund innovation.

## Maintain opex investments

Retain your infrastructure investment by repurposing existing hardware for OpenShift.

## Kubernetes skills development

Motivate your team and provide career progression with training and skills development from Red Hat

## Modernize operational models

OpenShift can provide the technology foundation for a cultural shift to new operating models like site reliability engineering (SRE)



# 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://linkedin.com/company/red-hat)

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

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

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