

Red Hat Partner Tech Days

OpenShift Virtualization



September 2020

Mike Savage

Senior Solutions Architect,
Partner Development

Dan Lineback

Senior Solutions Architect,
Partner Development

Mike Watkins

Senior Manager, Solution Architects
Partner Development

Mark Ivankovich

Channel Solutions Architect



But, Before We Get Started,
A Few Announcements...

A **new virtual** technical enablement series that provides the latest in the Red Hat product strategy and enablement on the newest features to our technical field associates

 **Red Hat** • • • —
Tech
Ready
— • • • October & November 2020

- Integrated into the Red Hat OPEN training system
- More communications to come...



Hear the latest from Red Hat Business Unit and Engineering leaders on our product strategies and roadmaps



Gain hands-on experience with new product features through deep dive demos and labs



Session content dropped weekly on Monday's from October 19 through November 9



SmartPage on Red Hat Partner Content Hub for more information (Coming Soon)

Red Hat Tech Ready: Partner Experience

Red Hat Tech Ready is a new virtual, global technical series that provides the latest in the Red Hat product strategy and enablement, exclusively for our Red Hat internal teams and Red Hat Partners.

What are the Goals of RHTR?

Drive more awareness of what's coming, provide the field applicable knowledge and practices, and connect it all to the Red Hat Open Hybrid Cloud Strategy

Who are the Target Attendees?

Red Hat's technical field (SA, Services, CXnO) and
Partners in a technical role

How will partners access content?

Partners will get access to the same on-demand content at the same time as Red Hatters. This content will be made available through Red Hat OPEN training system

What is Happening Each Week?

Each week focuses on a part of Red Hat's Open Hybrid Cloud Strategy. On-demand content will be launched each Monday and live experiences will be scheduled throughout the week.

Week of	Theme of the Week
Oct 19	Hybrid Cloud Infrastructure Week
Oct 26	Cloud Native Development
Nov 2	Management & Automation
Nov 9	Digital Transformation

NA Partner Tech Virtual Office Hours

For North America Red Hat Business Partners technical team members

Office Hour format: "Ask me anything/Open Mic" with remaining time to cover the Product/Technology topics

- See Office Hour schedule here -- <https://www.redhatpartnertech.com>

Virtual Partner Tech Office Hours

- *Meet with Red Hat Solution Architects & SME's to help answer your questions on product and technology topics*
- *Dedicated weekly time slot for you to pop in, ask a question, then get back to your schedule*
- *Collaborate/Brainstorm with peers on customer questions, projects, etc.*
- *Build and formulate new relationships*

Agenda

- ▶ Red Hat OpenShift Virtualization Overview and Use Cases
- ▶ Workshop Architecture/Tools/Packet
- ▶ Feature Walkthroughs & Demos
 - VM Storage including add or edit persistent Storage
 - Creating VM Networks in OpenShift
 - General Virtual Machine Overview in OpenShift
 - VM creation and Management in OpenShift
- ▶ Common Questions
- ▶ Next Steps

Red Hat OpenShift Virtualization Overview and Use Cases

OpenShift and Kubernetes crossing to early majority in IT adoption

Two emerging trends

- 1. Need for a better modernization strategy for virtual machine (VM)-based workloads**

An all-or-nothing approach to containerization is too slow, so organizations have a large investment in virtual machines.

- 2. Desire for a single architecture for all workloads**

Kubernetes supports stateful applications, and organizations desire to reduce costs by adopting a single cloud-native platform.

What's in it for Operations?

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

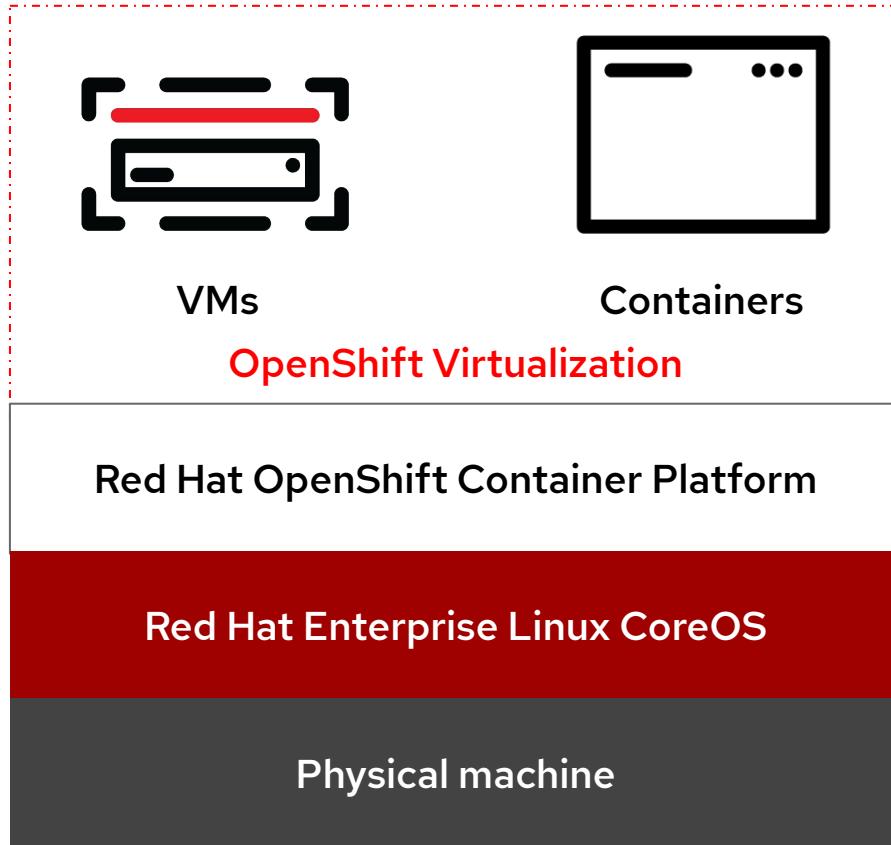
Motivate your team and provide career progression with training and skills development from Red Hat

Modernize operational models

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

OpenShift and OpenShift Virtualization

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



- Brings VMs to OpenShift 4.5+
- Accelerates 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 based applications over time, or maintain them as VMs
 - [SAP’s open source project “Gardener” leveraged Red Hat OpenShift Virtualization](#)
 - [Goldman Sachs Revamps Virtualization Infrastructure](#)

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!

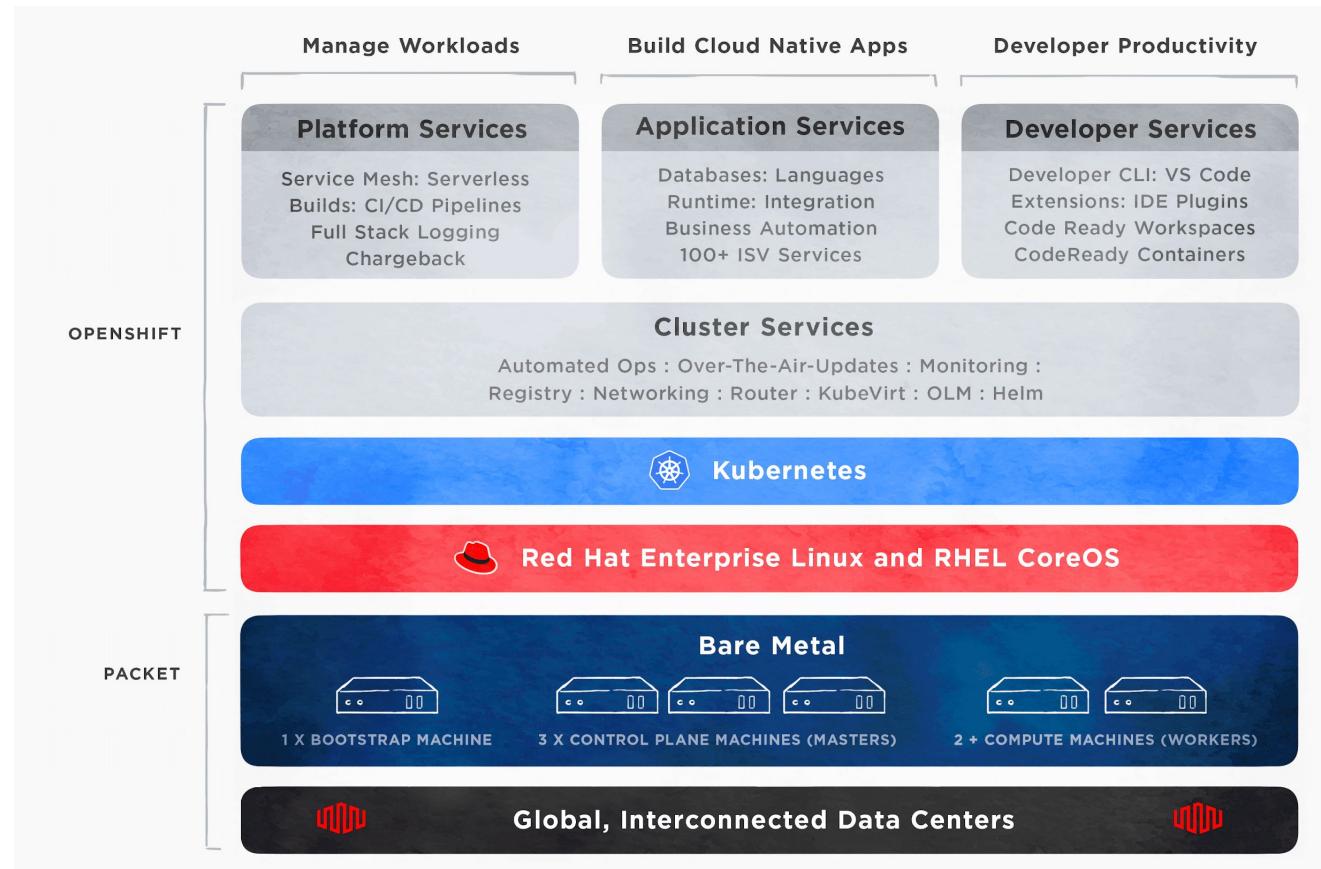
Workshop Architecture/Tools/Packet & OpenShift Virtualization Operator

A couple of things...



OpenShift, Made Simple

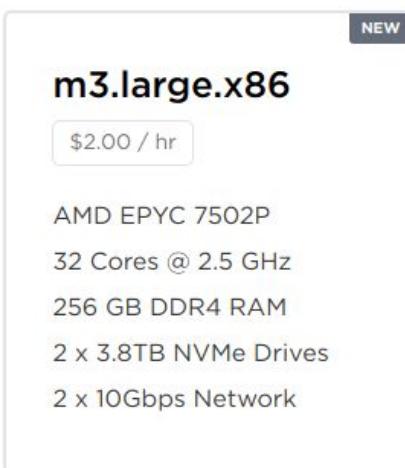
Automate the end-to-end install of Red Hat OpenShift on Packet bare metal servers



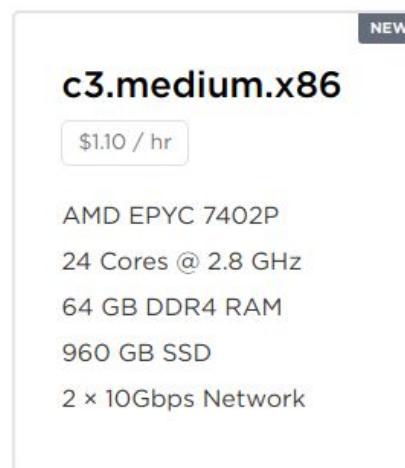
Deliver better applications faster. We provide:

- API-driven bare metal in global locations
- Fast, automated deployments – takes about 20 minutes
- Consistent user experience across on-prem, Packet or public cloud

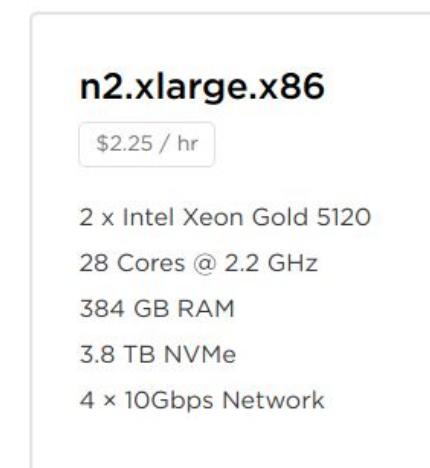


[Products](#)[Solutions](#)[Developers](#)[Resources](#)[Company](#)[Log in](#)[SIGN UP](#)

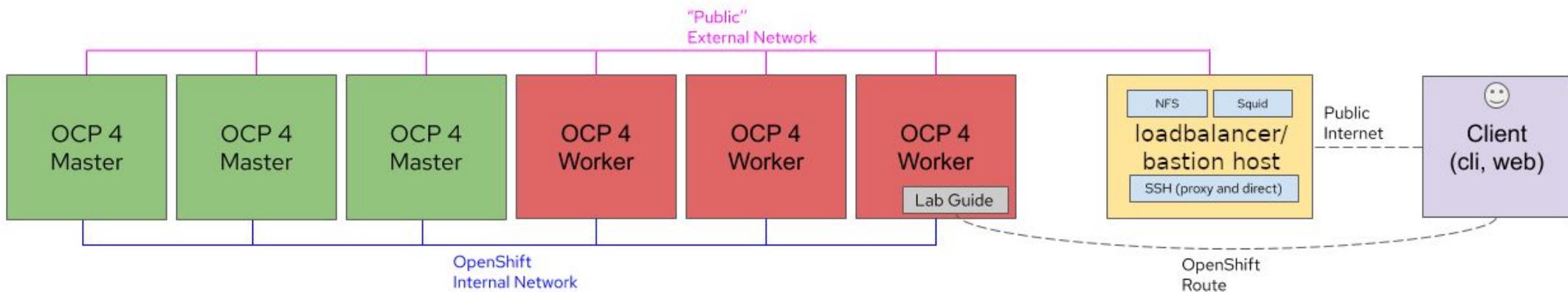
1x Bastion/LB (utility node)



3x Masters



3x Workers



GitHub repos and tools



OpenShift UPI Deploy to Packet Cloud

<https://github.com/RedHatSI/terraform-packet-openshift> ==> <https://github.com/heatmiser/openshift-packet-deploy>



AgnosticD - Ansible Deployer for multiple Cloud Deployers

<https://github.com/redhat-cop/agnosticd> ==> <https://github.com/heatmiser/agnosticd>



OpenShift Virtualization Lab on Packet Cloud

<https://github.com/RHFieldProductManagement/openshift-virt-labs> ==> <https://github.com/heatmiser/openshift-virt-labs/tree/packet>



OpenShift Virtualization
 Red Hat
OpenShift 4



podman
containers

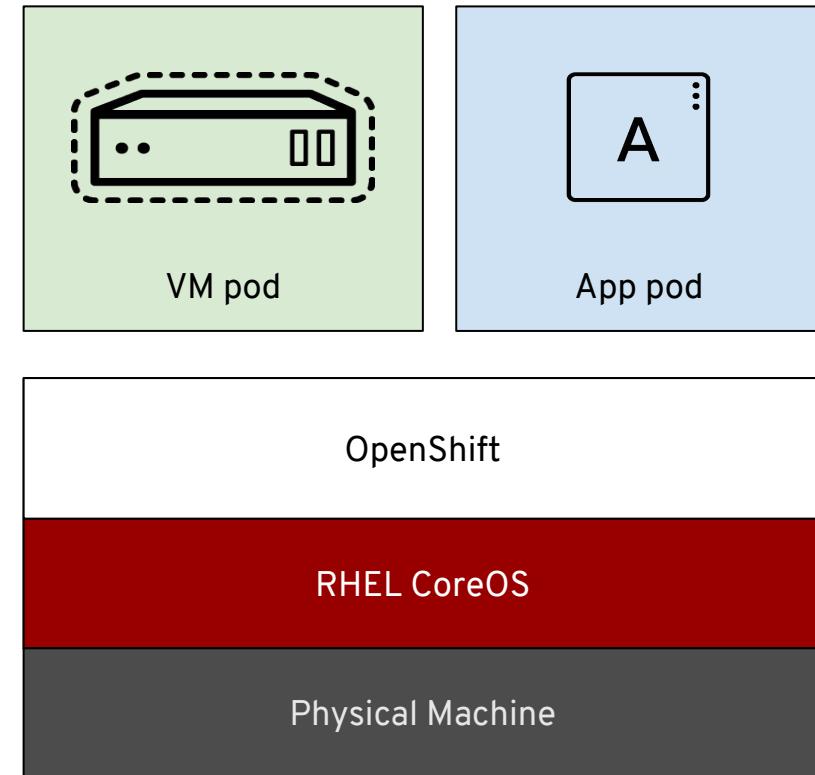


- ▶ Feature Walk Throughs and Demos

- ▶ VM Storage including add or edit persistent Storage

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



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

The screenshot shows the 'Create Virtual Machine' wizard in progress, specifically the 'Storage' step (step 3). The main interface displays a table of existing disks:

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

A large black arrow labeled '1' points to the 'Add Disk' button in the top right corner. A modal window titled 'Add Disk' is open, showing the configuration for a new disk:

- Source: Blank
- Name: disk-0
- Size: 20 GiB
- Interface: VirtIO
- Storage Class: managed-nfs-storage
- Advanced settings:
 - Volume Mode: Filesystem
 - Access Mode: Single User (RWO)

At the bottom of the modal are 'Cancel' and 'Add' buttons. At the bottom of the main interface are 'Next', 'Review and create', 'Back', and 'Cancel' buttons.

Demo: Storage

- ▶ Creating VM Networks in OpenShift

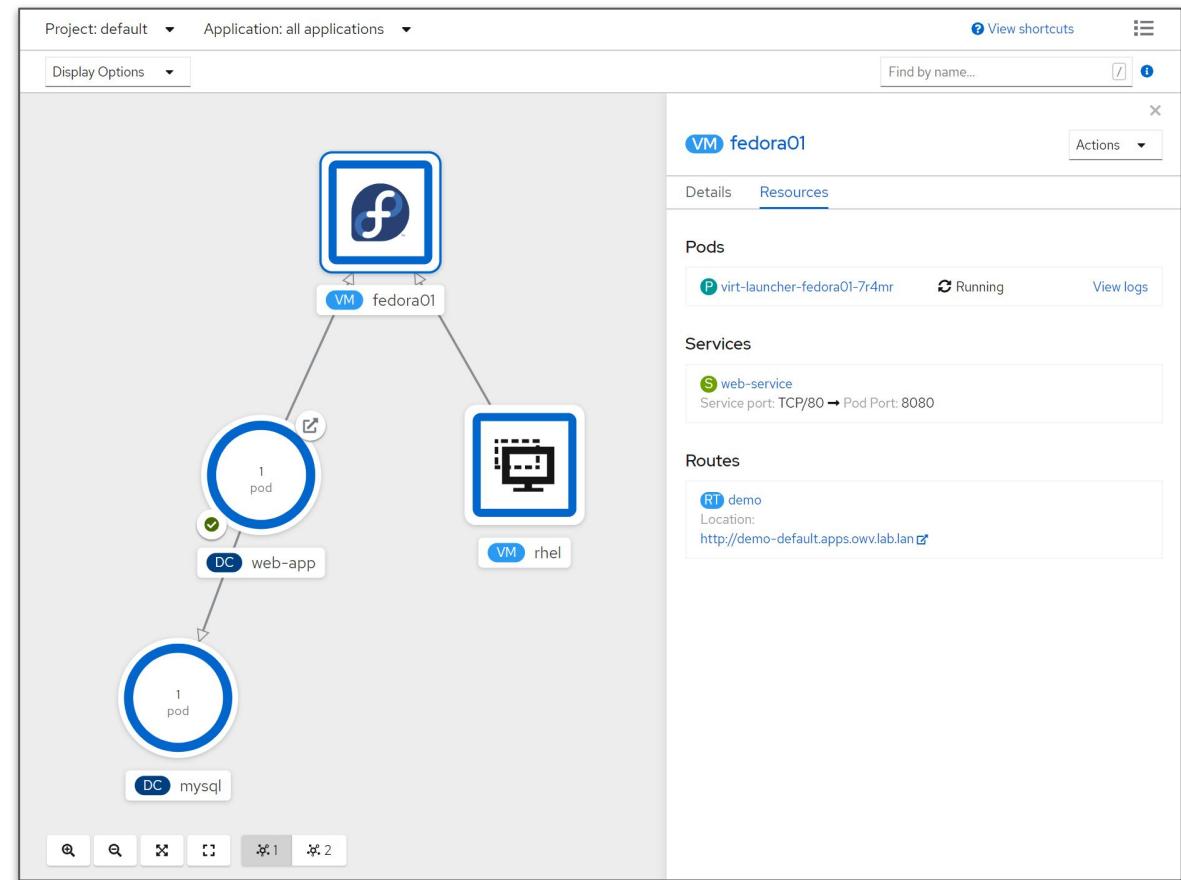
Connecting VMs to networks

- Virtual machine interfaces describe NICs attached to the VM
 - `spec.domain.devices.interfaces`
 - Model: virtio, e1000, pcnet, rtl8139, etc.
 - Type: masquerade, bridge
 - MAC address: customize the MAC
- The networks definition describes the connection type
 - `spec.networks`
 - Pod = default SDN
 - Multus = secondary network using Multus
- Using the GUI makes this simple and removes the need to edit / manage connections in YAML

```
1  apiVersion: kubevirt.io/v1alpha3
2  kind: VirtualMachine
3  name: demo-vm
4  spec:
5    template:
6      spec:
7        domain:
8          devices:
9            interfaces:
10           - bridge: {}
11             model: virtio
12             name: nic-0
13             hostname: demo-vm
14           networks:
15             - multus:
16               networkName: bond1-br1
17               name: nic-0
```

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

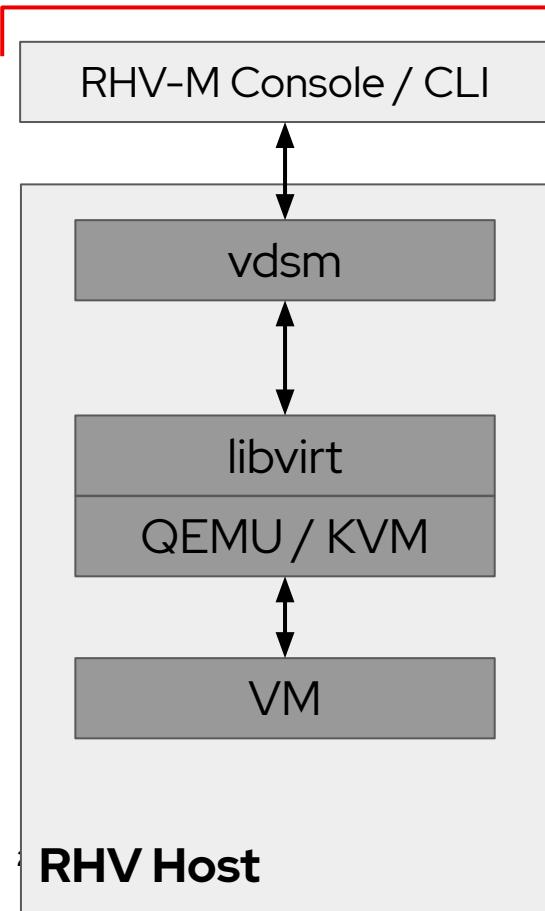


Demo: Networking

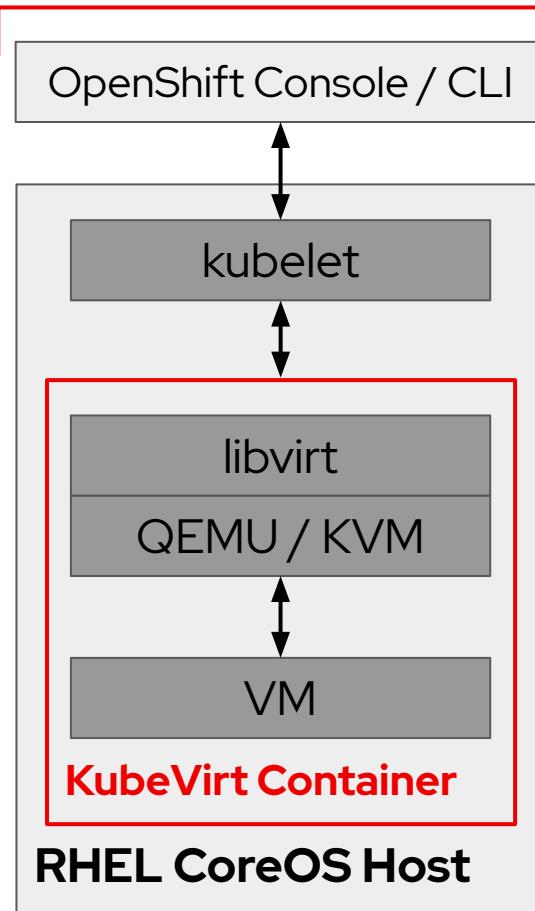
- ▶ General Virtual Machine Overview in OpenShift

Containerizing KVM

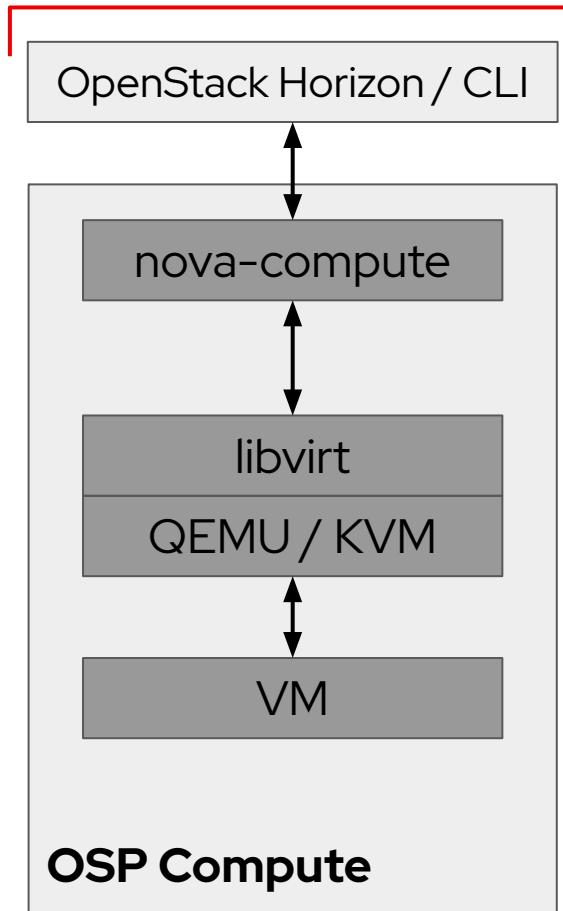
Red Hat Virtualization



OpenShift Virtualization

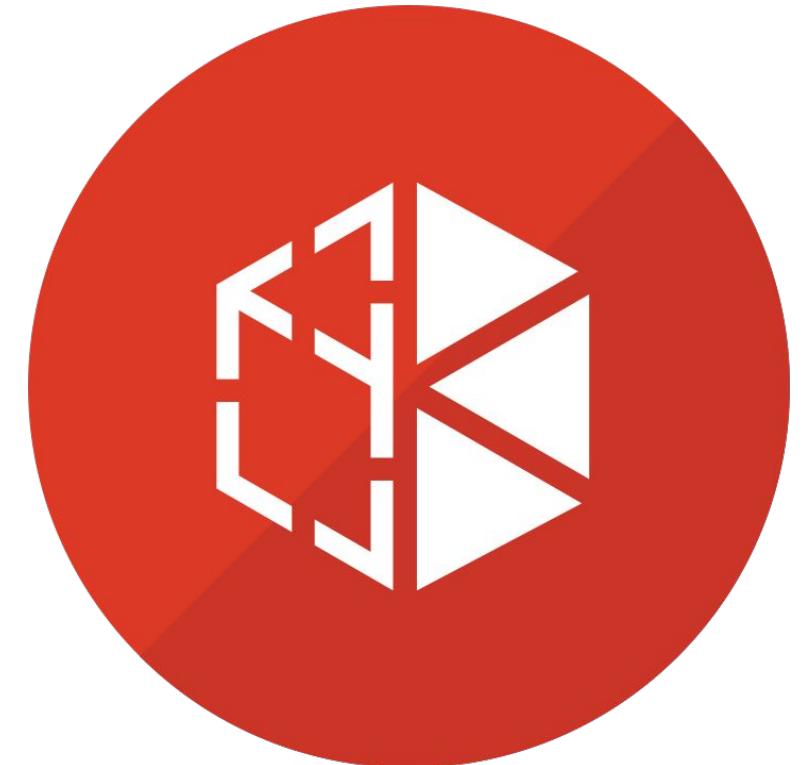


Red Hat OpenStack Platform



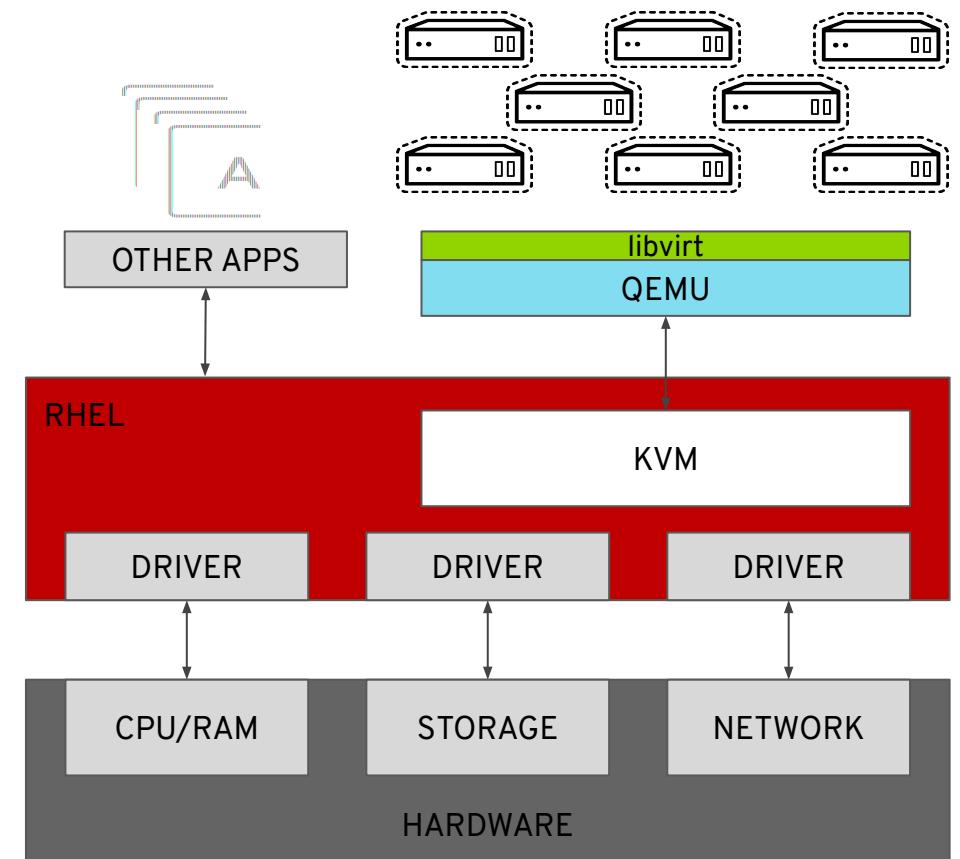
OpenShift Virtualization

- 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

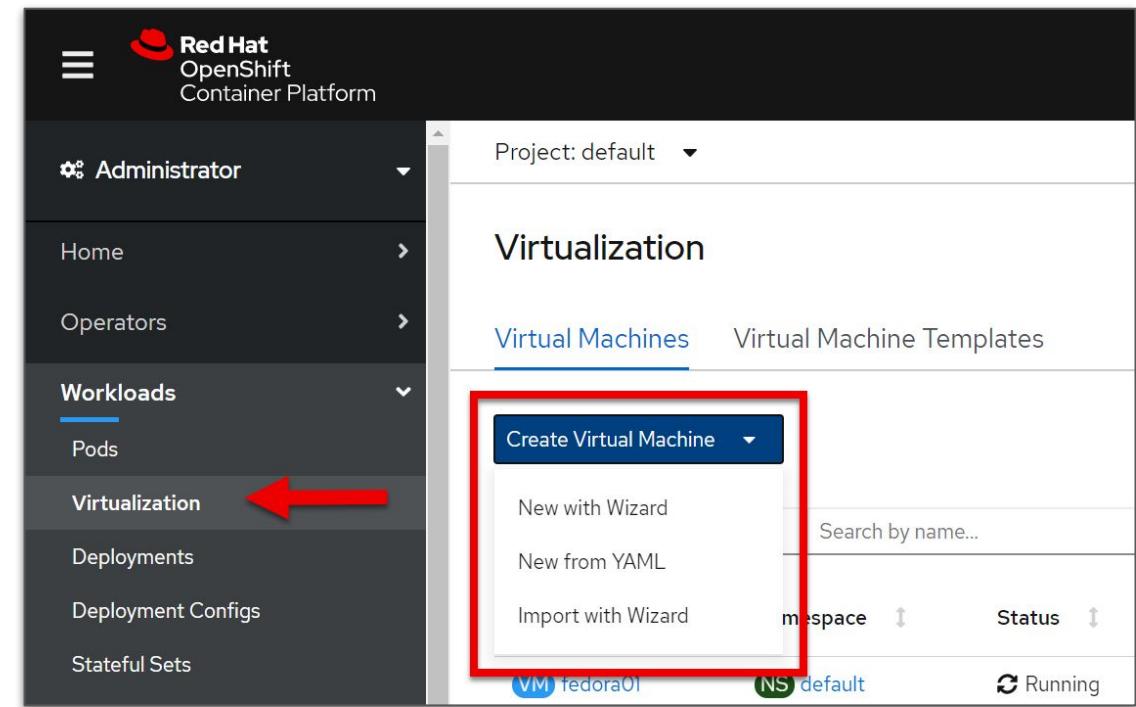
- OpenShift Virtualization uses KVM, the Linux kernel hypervisor
- KVM is a core component of the Red Hat Enterprise Linux kernel
 - KVM has 10+ years of production use: Red Hat Virtualization, Red Hat OpenStack Platform, and RHEL all leverage KVM, QEMU, and libvirt
- QEMU uses KVM to execute virtual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



- ▶ VM creation and Management in OpenShift

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



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
```

Demo: VM creation and Management in OpenShift

- ▶ Common Questions

OpenShift Virtualization

OPTIONAL

What version of OpenShift and what operator is required to run OpenShift Virtualization?

4.5 is minimum and the worker nodes must be physical or bare metal hosts. Operator 2.4 is most current with CNV operator Deployment.

OpenShift Virtualizati...

 OpenShift Virtualization provided by Red Hat

Creates and maintains an OpenShift Virtualization Deployment

OperatorHub > Operator Subscription

Create Operator Subscription

Install your Operator by subscribing to one of the update channels to keep the Operator up to date. The strategy determines either manual or automatic updates.

Installation Mode *

All namespaces on the cluster (default)
This mode is not supported by this Operator

A specific namespace on the cluster
Operator will be available in a single namespace only.
 PR default

Update Channel *

2.1

2.2

2.3

2.4

Approval Strategy *

Automatic

Manual

Container-native virtualization Operator
provided by Red Hat

Provided APIs

 HCO CNV Operator Deployment Represents the deployment of CNV Operator	 V2V Vmware V2V Vmware
 OVP V2V oVirt V2V oVirt	 NAC Cluster Network Addons Cluster Network Addons
 KV KubeVirt deployment Represents a KubeVirt deployment	 KCT KubeVirt common templates Represents a deployment of the predefined VM templates
 KMA KubeVirt Metric Aggregation Provide aggregation rules for core kubevirt metrics	 KNL KubeVirt Node labeller Represents a deployment of Node labeller component

Subscribe **Cancel**

OPTIONAL

Are the Virtual Machines stateless or stateful? Virtual machines can be both, but the cluster must have access to RWX PVC or Persistent Volume Claims that is based on CephFS and can be used in conjunction with the MCG or multi-cloud gateway as a persistent storage solution for OCP workloads.

Can Virtual machines be imported from VMware? Yes, there is a built in wizard in the Workload, virtual machines to import from VMware vCenter using VDDK. Beside Linux workloads Windows 2008-2019 VM's can be imported. Refer back to original use case on what are good candidates for importing into OCP

The screenshot shows the Red Hat OpenShift Container Platform web interface. The left sidebar is titled 'Administrator' and includes sections for Home, Dashboards, Projects, Search, Explore, Events, Operators, Workloads (which is currently selected), and Virtual Machines. Under Workloads, there are sub-options for Virtual Machine Templates, Deployments, Deployment Configs, and Stateful Sets. The main content area is titled 'Import Virtual Machine' and displays a step-by-step wizard. Step 1, 'General', is active and shows the provider as 'VMware'. Step 2, 'Networking', Step 3, 'Storage', Step 4, 'Advanced' (with 'Cloud-init' listed), Step 5, 'Review', and Step 6, 'Result', are also visible. A dropdown menu for 'vCenter instance' is open, showing options: '--- Select vCenter Instance Secret ---', '--- Select vCenter Instance Secret ---' (which is highlighted in blue), 'Connect to New Instance', 'administrator-10.8.58.136-g77hr', and '--- Select VM or Template ---'. Below this, fields for 'Operating System', 'Flavor', and 'Workload Profile' are present, each with a '--- Select [option] ---' placeholder. At the top of the main content area, a message reads: 'You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.' The top right corner shows the user 'kube:admin'.

Do virtual machines in OpenShift Virtualization require Agents?

Although not required it is best practice to install the QEMU guest agent that is a daemon that runs on the Linux virtual machine. For Windows virtual machines the QEMU guest agent is included in the VirtIO drivers.

Do virtual machines in OpenShift Virtualization require Agents?

Although not required it is best practice to install the QEMU guest agent that is a daemon that runs on the Linux virtual machine. For Windows virtual machines the QEMU guest agent is included in the VirtIO drivers.

What is running beneath the OpenShift Worker Nodes?

Red Hat Enterprise Linux CoreOS.

Resources



Red Hat OpenShift

<https://demo.openshift.com/en/latest/openshift-virtualization/>

OpenShift Virtualization

https://www.openshift.com/learn/topics/virtualization/?extIdCarryOver=true&sc_cid=701f2000001OH7iAAG

OpenShift - OneStop Partner Content Hub

<https://redhat-partner.hightspot.com/items/5cbecc24566bbbaa0b941b04f0?lfrm=srp.0>

Packet Cloud - Sign Up

<https://baremet.al/openshift>

- Refer to next slide for details

OpenShift on Packet @ RH Partner Tech Days



When: Friday, Sept 11th, 1:30p EST
Wed, Sept 16th, 12:00p EST

Where: [Red Hat Partner Tech Days](https://events.redhat.com/profile/web/index.cfm?PKwebID=0x51871abcd)

<https://events.redhat.com/profile/web/index.cfm?PKwebID=0x51871abcd>

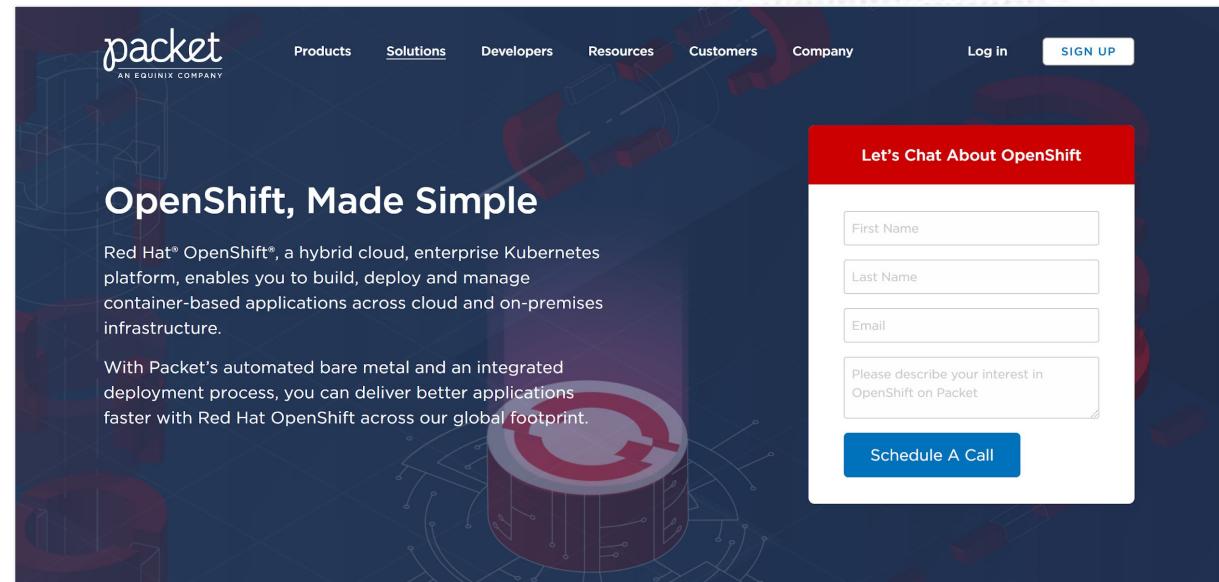
Who: Red Hat Team including
Mike Savage, Sr. Channel Solutions Architect

CALL-TO-ACTION: Give it a Test Drive!

- Sign-Up @ <https://baremet.al/openshift> and mention you attended Partner Tech Days
- Get \$1,000 Packet Credits [for a 3-5 day experience]

Want more?

[Packet Community Slack](#) @ #redhat-openshift



Survey

THANK YOU for attending!

- Please complete the survey to be entered into a raffle to receive a limited amount of \$25 eGift Cards
- Link [here](#)

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

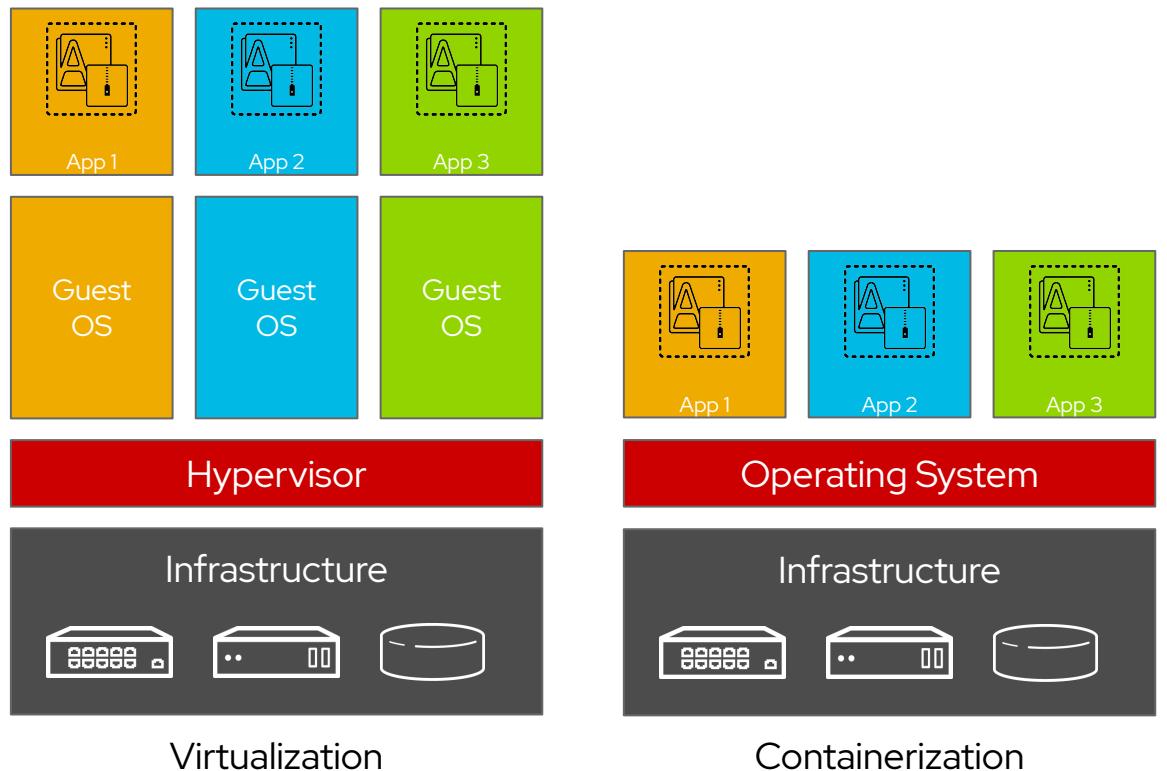
Backup Slides

OpenShift Virtualization Roadmap

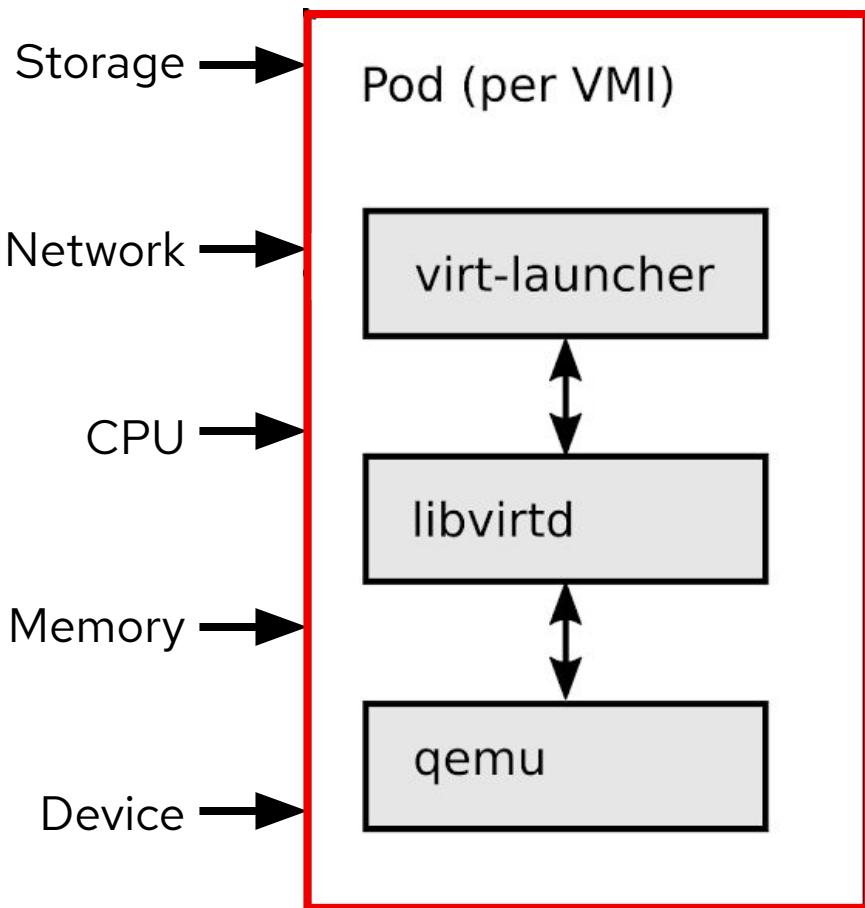
Near Term <small>(3-6 months)</small>		
CORE	CORE	CORE
NETWORK	NETWORK	NETWORK
STORAGE	STORAGE	STORAGE
<ul style="list-style-type: none"> Streamline 'oc drain' Improve performance with Hyper-V enlightenments Improve golden image template workflow Sizing guidance for field 	<p>Virt gaps - vCPU & vNUMA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

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



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

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' selected. 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:48e1:le2: 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

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 'Source' dropdown is open, showing options like 'PXE', 'URL', 'Container', and 'Disk'. The 'Flavor' dropdown is also open, showing options like 'Custom', 'Tiny', 'Small', 'Medium', 'Large', and 'Custom'. The 'Workload Profile' dropdown is open, showing options like 'desktop', 'highperformance', and 'server'. Arrows numbered 1, 2, and 3 point to each of these dropdowns respectively.

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 the OpenShift web interface. The current step is 'Networking' (step 2). A modal dialog titled 'Add Network Interface' is open, showing fields for Name (nic-1), Model (VirtIO), Network (host-br1), Type (bridge), and MAC Address (empty). Step 1 is 'General' (highlighted with a large arrow), step 2 is 'Networking' (highlighted with a large arrow), step 3 is 'Storage', step 4 is 'Advanced', step 5 is 'Review', and step 6 is 'Result'. Buttons at the bottom include 'Next', 'Review and create', 'Back', and 'Cancel'.

Project: default ▾

Create Virtual Machine

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

Network Interfaces

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

Add Network Interface

Name *
nic-1

Model *
VirtIO

Network *
host-br1

Type *
bridge

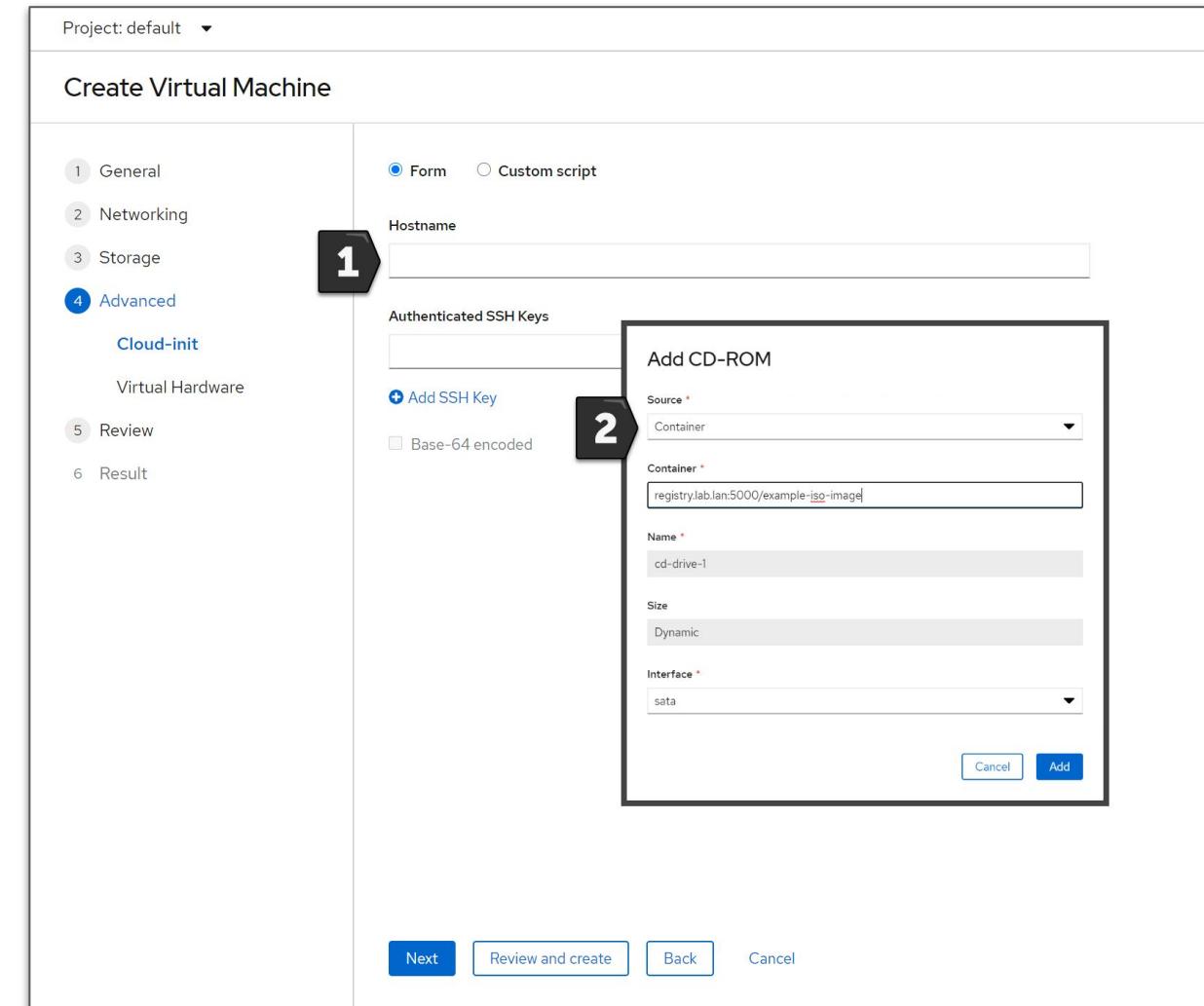
MAC Address

Cancel Add

Next Review and create Back Cancel

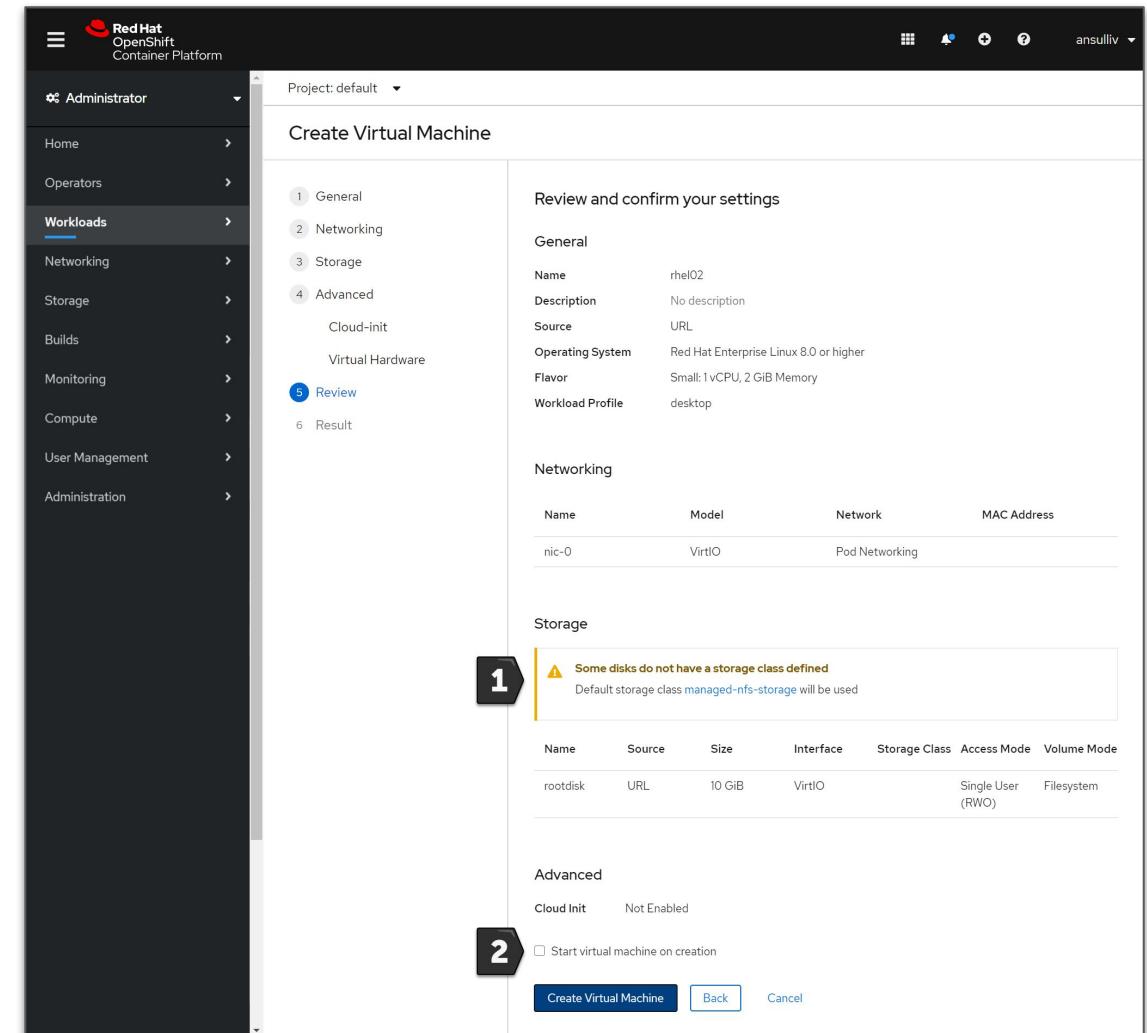
Create Virtual Machine - Advanced

- 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



Import VMs

Virtual Machine Import

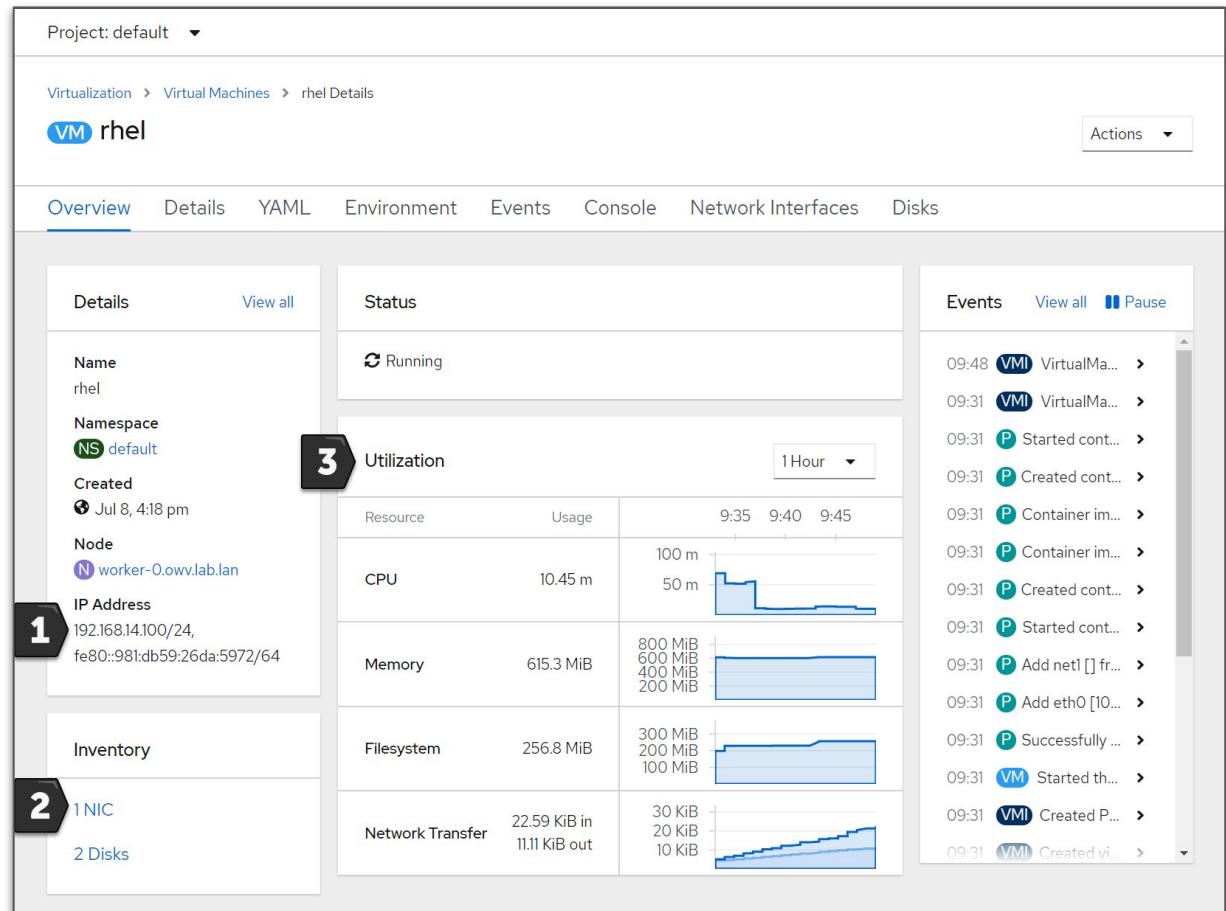
- Wizard supports importing from VMware or Red Hat Virtualization
 - Single-VM workflow
- VMware import uses VDDK to expedite the disk import process
 - User is responsible for downloading the VDDK from VMware and adding it to a container image
- Credentials stored as Secrets
- **ResourceMapping** CRD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

Add image here

View / manage VMs

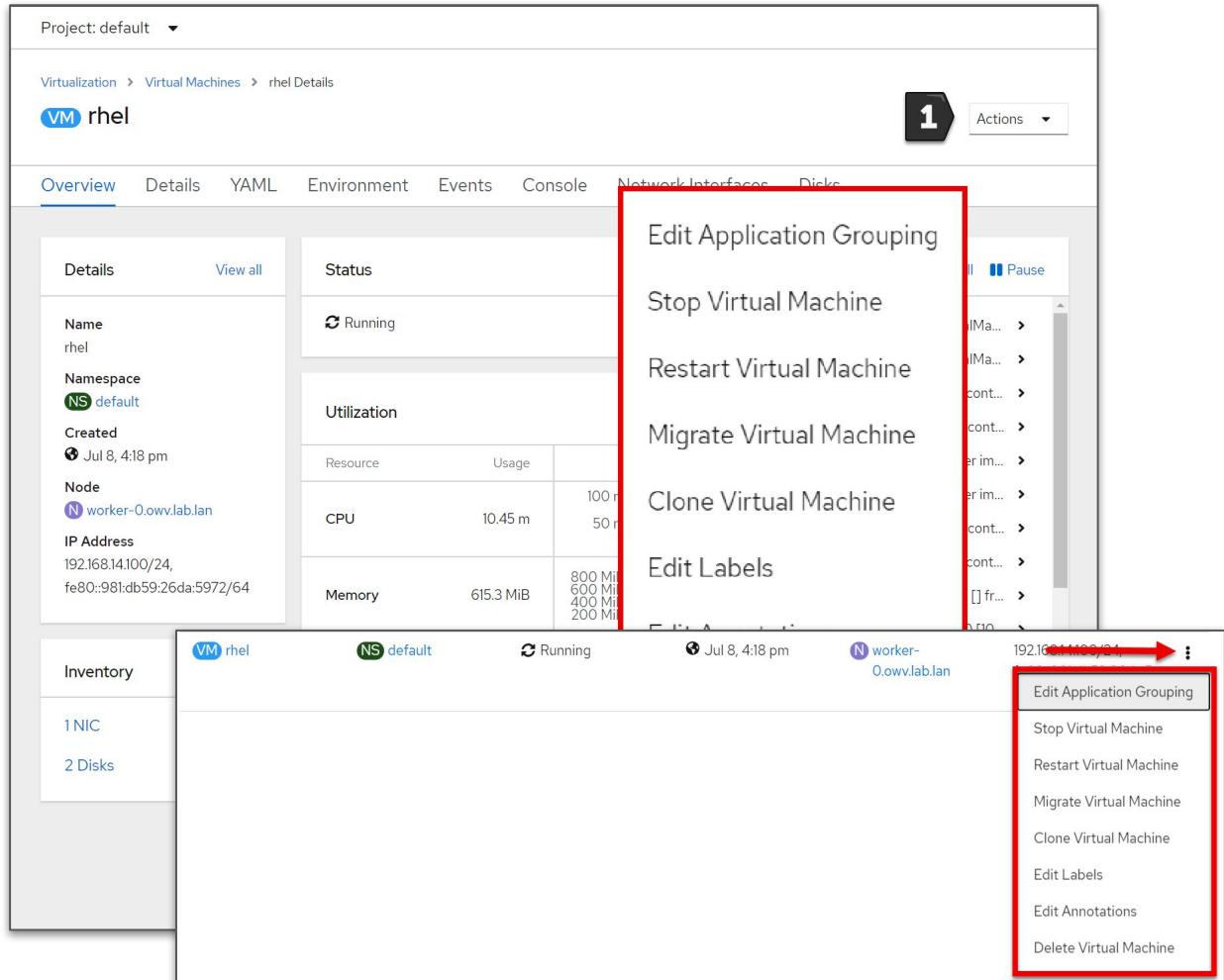
Virtual Machine – Overview

- 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

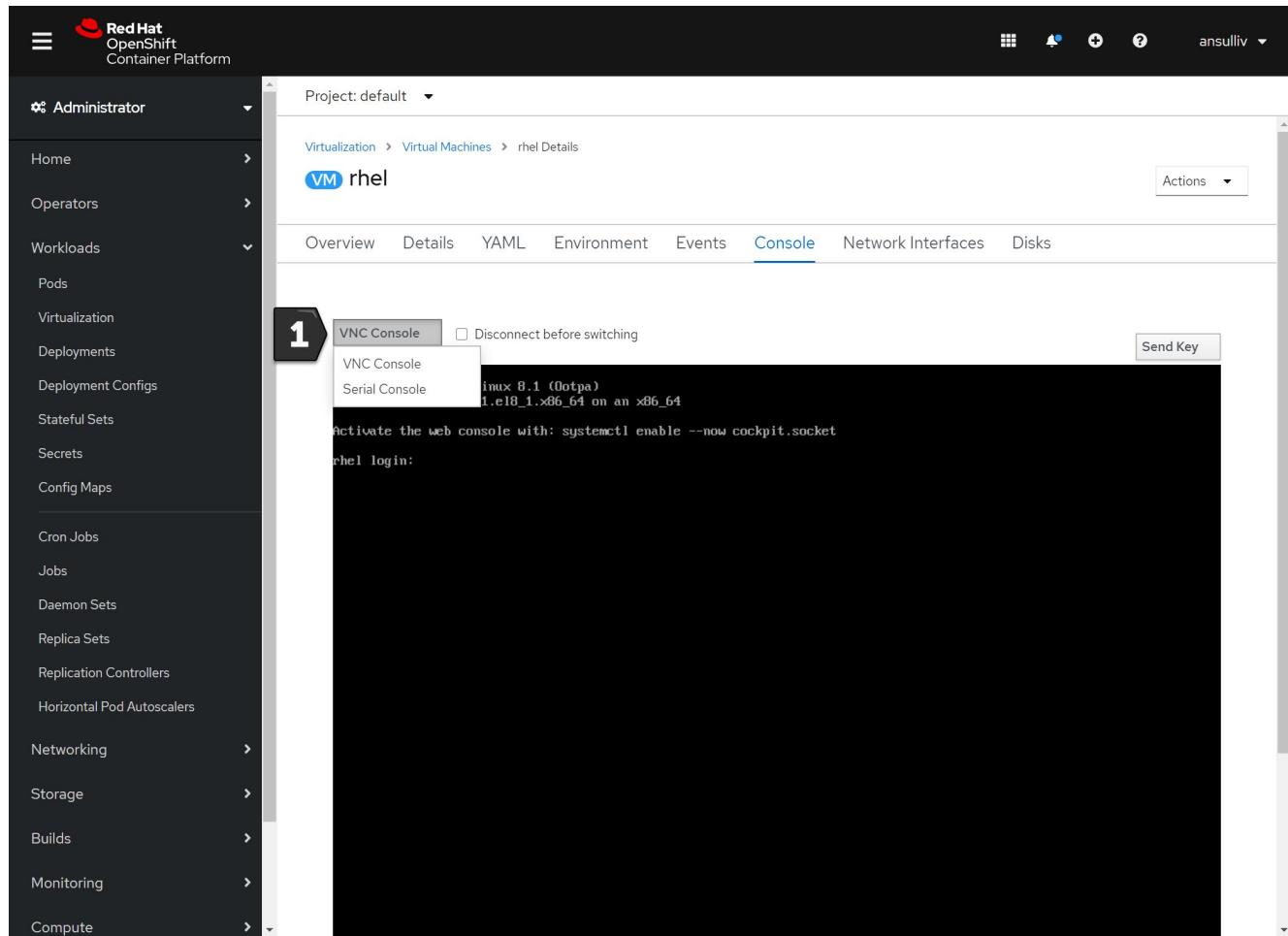
- Details about the virtual machine
 - Labels, annotations
 - Configured OS
 - Template used, if any
 - Configured boot order
 - Associated workload profile
 - Flavor
- Additional details about scheduling
 - Node selector, tolerations, (anti)affinity rules
- Services configured for the VM

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 such as 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):

- The top navigation bar includes the Red Hat logo, 'Red Hat OpenShift Container Platform', a search bar ('Project: default'), and user information ('ansulliv').
- The left sidebar has sections for Administrator (Home, Operators, Workloads, Networking, Storage, Builds, Monitoring, Compute), User Management, and Administration.
- The main content area shows the 'Virtualization > Virtual Machines > rhel Details' page.
- The 'Network Interfaces' tab is selected, showing a table with one row:

Name	Model	Network	Type	MAC Address
nic-0	VirtIO	host-br1	bridge	-

Bottom View (Disks):

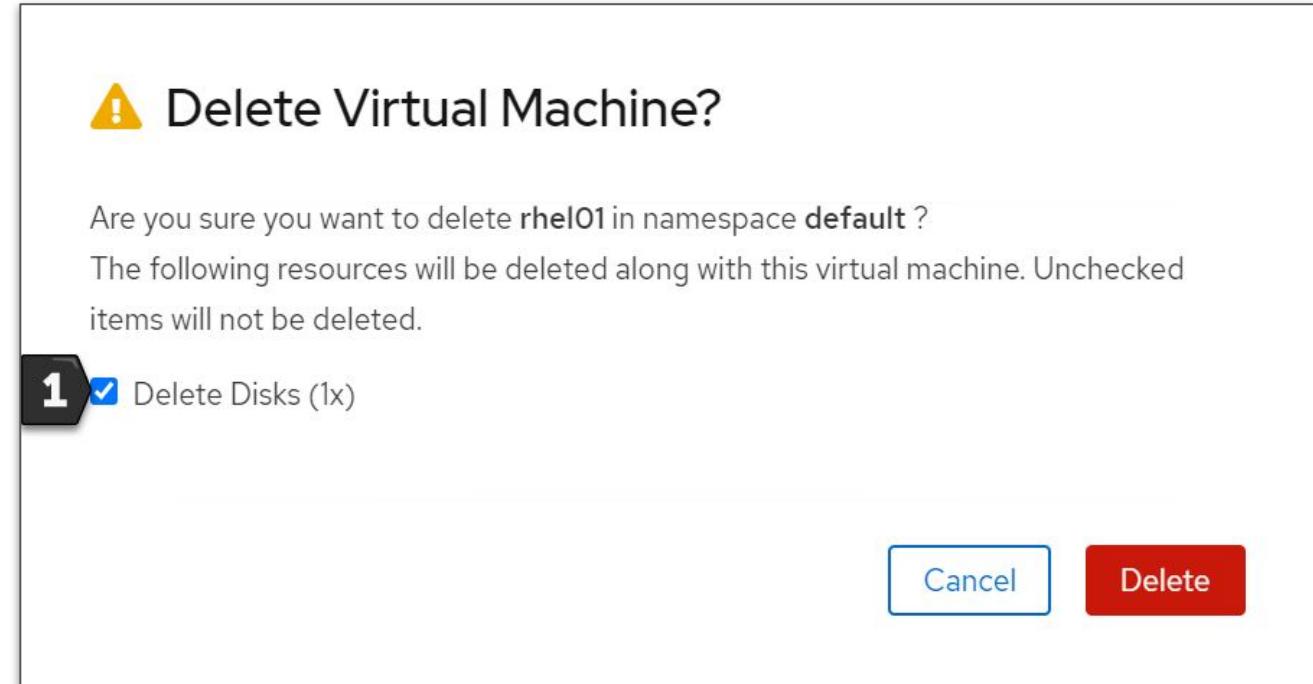
- The bottom navigation bar includes the Red Hat logo, 'Red Hat OpenShift Container Platform', a search bar ('Project: default'), and user information ('ansulliv').
- The left sidebar has sections for Administrator (Home, Operators, Workloads, Networking, Storage, Builds, Monitoring, Compute, User Management, Administration).
- The main content area shows the 'Virtualization > Virtual Machines > rhel Details' page.
- The 'Disks' tab is selected, showing a table with two rows:

Name	Source	Size	Interface	Storage Class
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 'rhel' VM details page in the OpenShift web interface, showing summary metrics for CPU, Memory, Filesystem, and Network Transfer over the last 1 hour. A red box highlights the CPU utilization chart. The bottom screenshot is the Prometheus Metrics UI, which displays a detailed breakdown of the CPU usage metric. The graph shows usage over a 30m period, with specific values for Name, namespace, pod, prometheus, and value.

Metrics Prometheus UI

Graph: CPU usage over 30m. Y-axis: 0.005 to 0.035. X-axis: 13:05 to 13:30.

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

rhel Details

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

Utilization: 1 Hour

CPU: 36.86 m

Memory: 633 MiB

Filesystem: 4.88 GiB

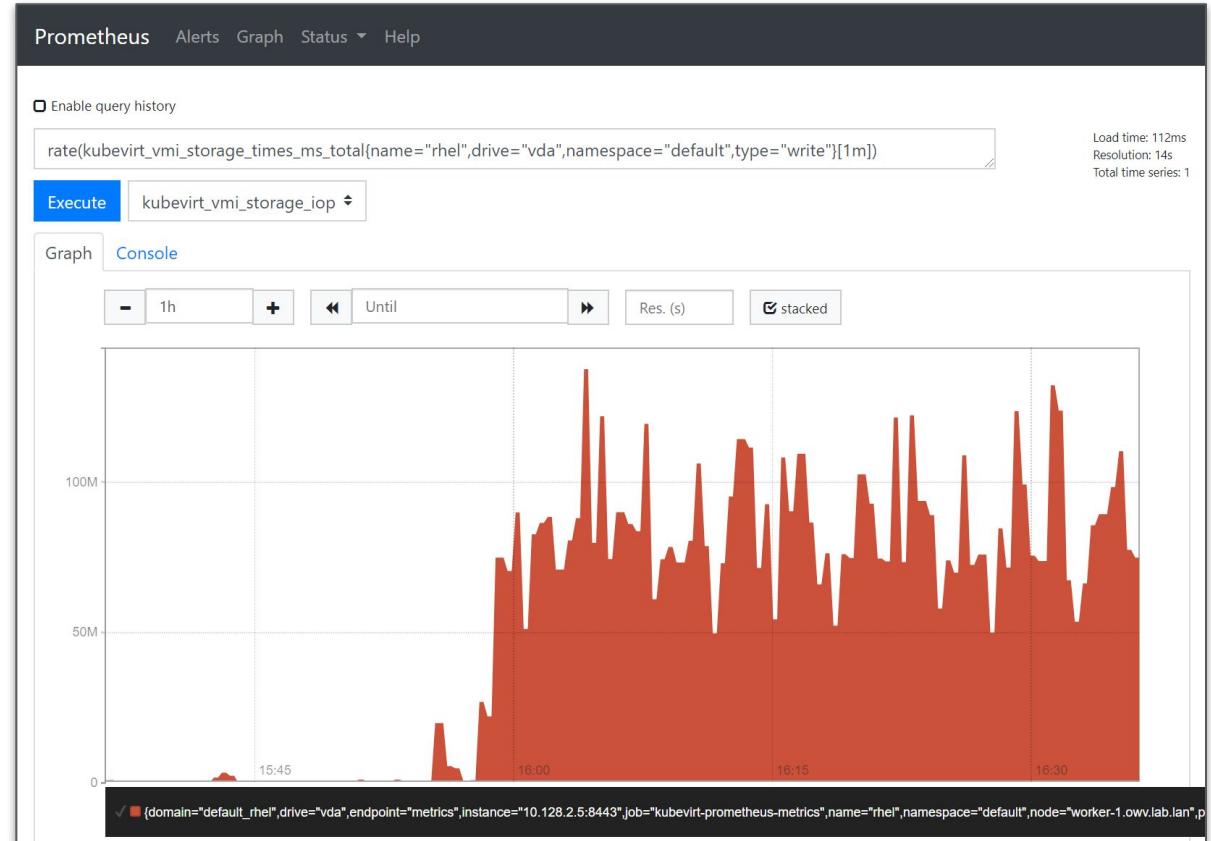
Network Transfer: 228.8 KiB in, 24.63 KiB out

Events: View all | Pause

There are no recent events.

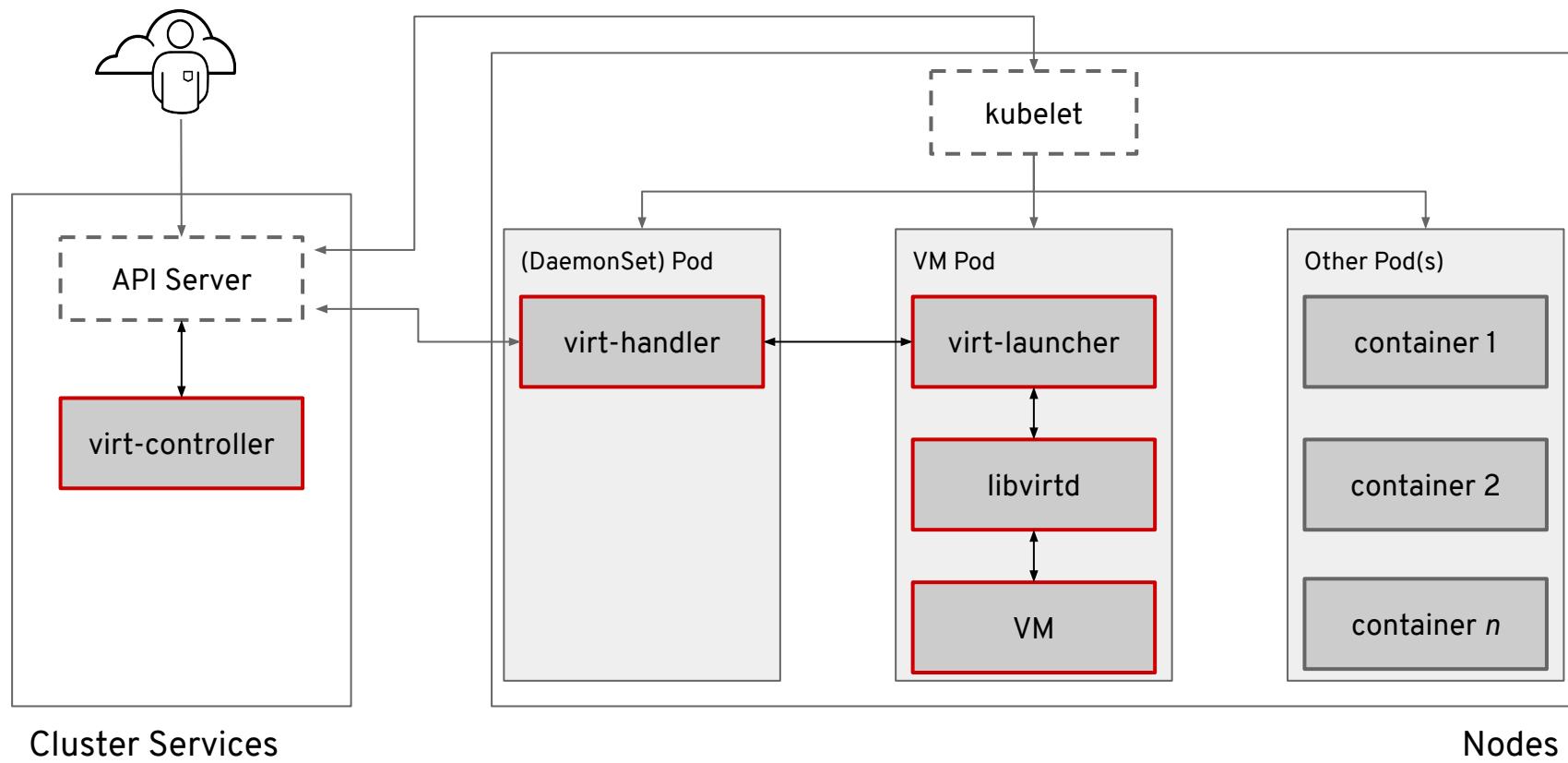
Detailed Virtual Machine metrics

- 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

Architectural Overview



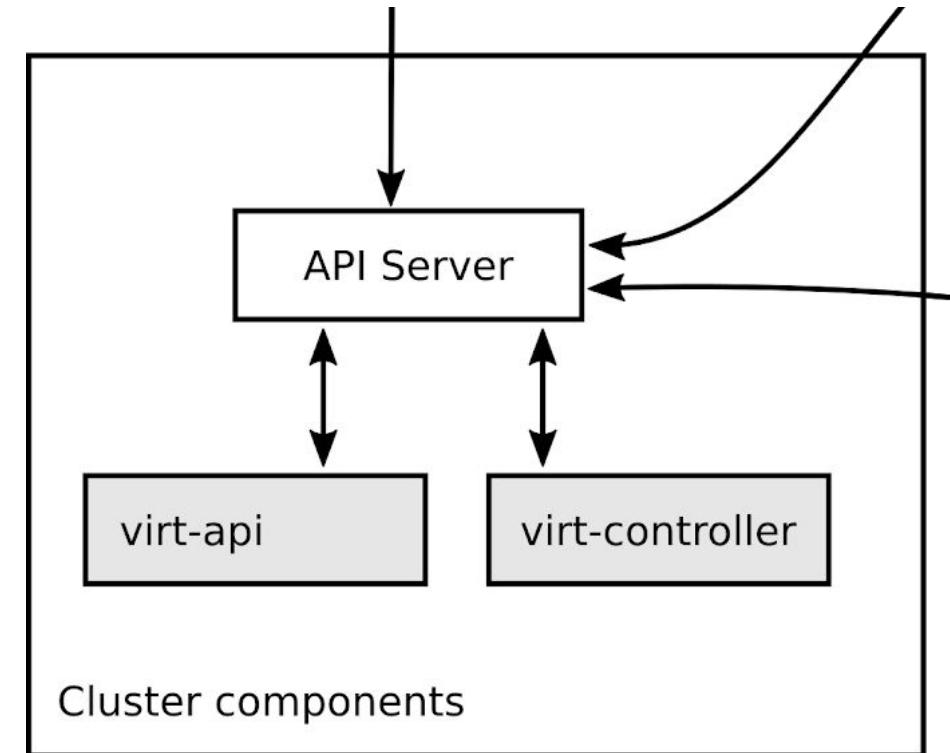
Adding virtualization to the Kubernetes API

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



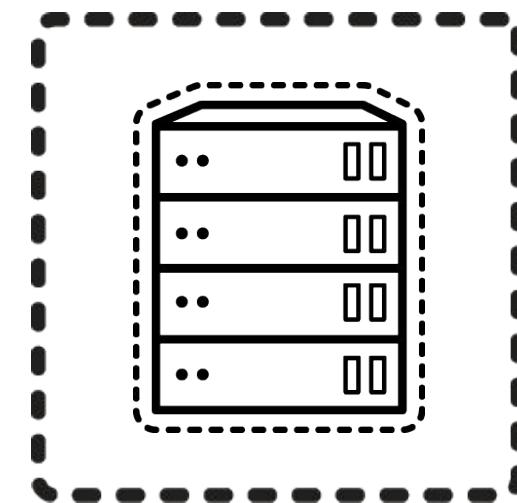
OpenShift cluster architecture

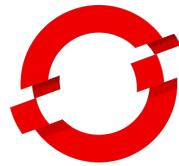
- Everything everywhere - all 8 nodes are "workers"
 - Create the cluster with the control plane as schedulable
 - No dedicated infra nodes, no dedicated OCS nodes
 - Pros: no wasted resources
 - Cons: must pay for all cores of all nodes, extra effort should be taken to ensure pods have appropriate QoS 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 circle with a white circular arrow inside, followed by the text "Red Hat" in a bold sans-serif font and "OpenShift" in a larger, bold sans-serif font below it.

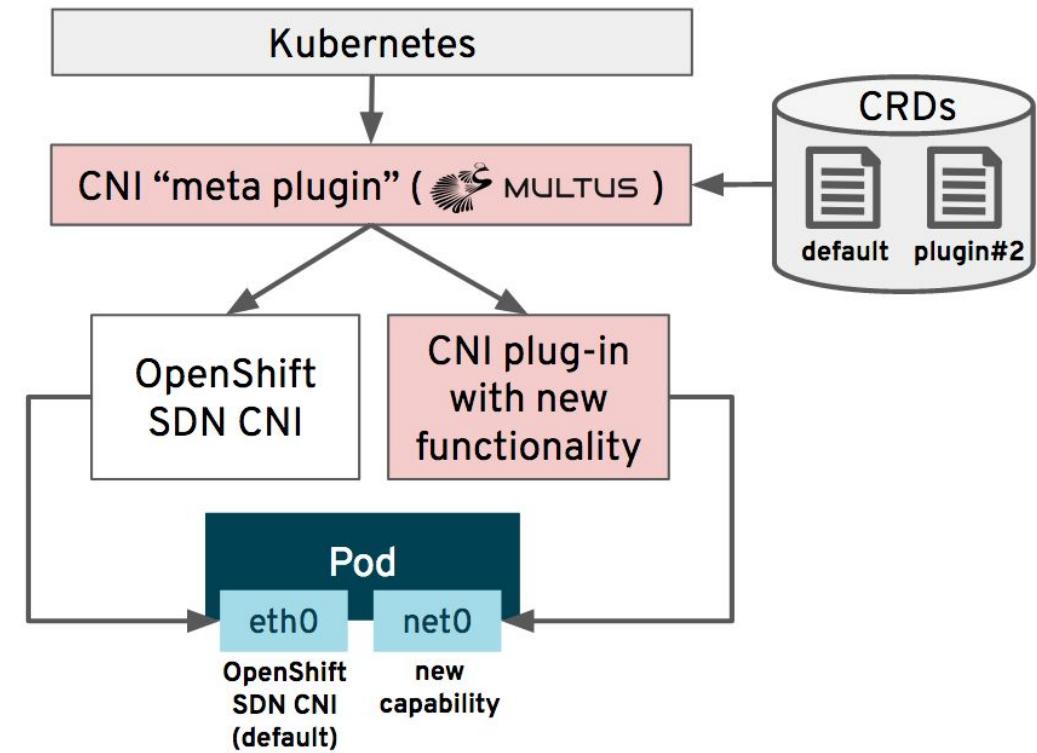
Virtual Machine Instances

- A VirtualMachine (VM) CRD represents a VM definition
- A VirtualMachineInstance (VMI) CRD represents a running virtual machine
- The VM definition is optional, a VMI can be created directly
 - Can be used with standard network and storage connections
 - If persisting the VMI disks, a DataVolume is highly encouraged to prevent the VMI from launching before the import is done

Network

Virtual Machine Networking

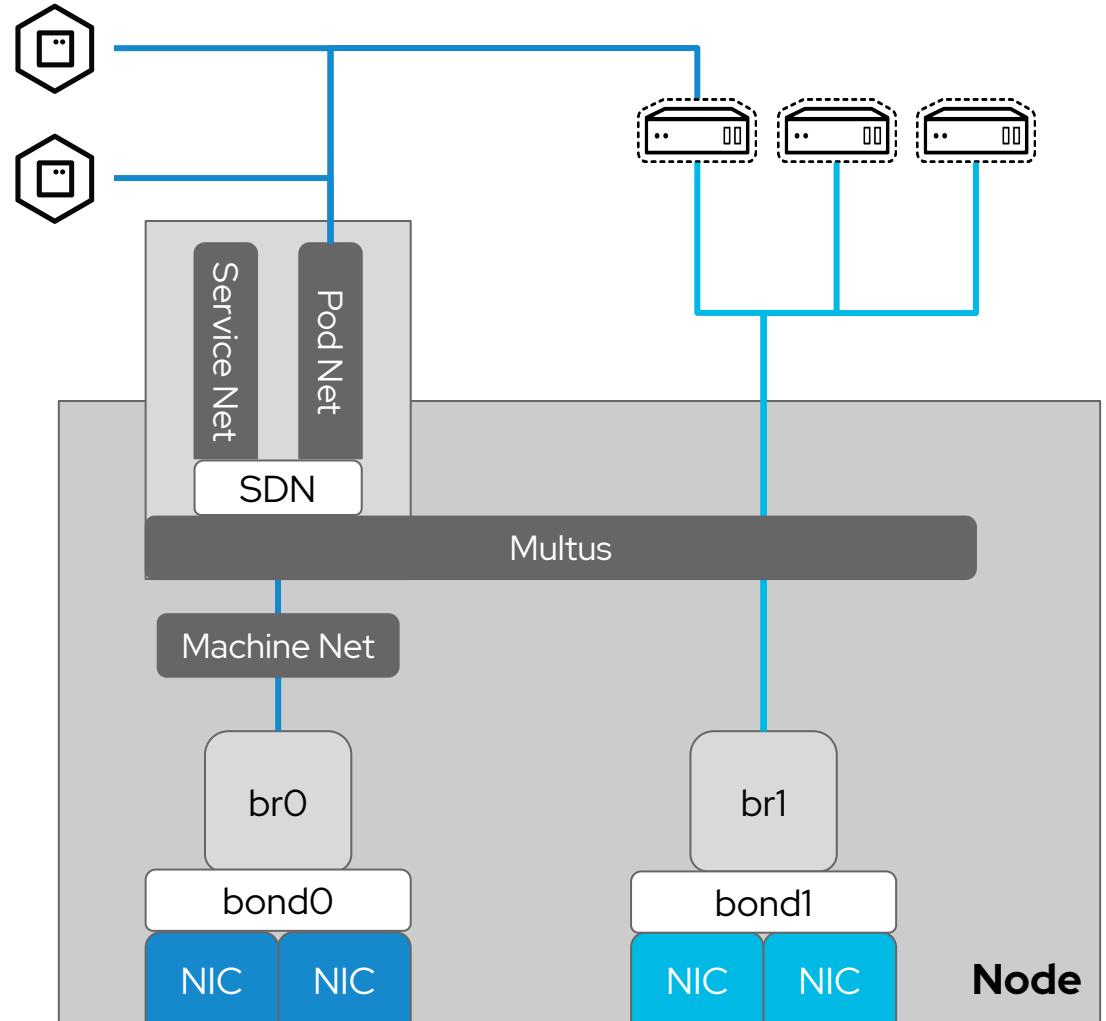
- Virtual machines optionally connect to the standard pod network
 - OpenShift SDN, OVNKubernetes, etc.
- Additional network interfaces accessible via Multus:
 - Bridge, SR-IOV
 - VLAN and other networks can be created using nmstate at the host level
- When using at least one interface on the default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

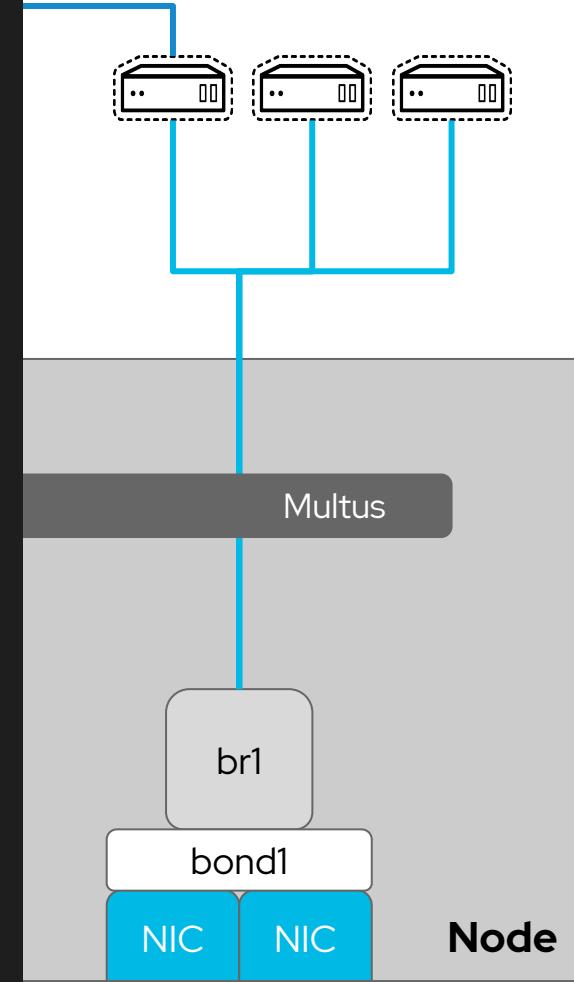
The following slides show an example of how this setup is configured



Host bond configuration

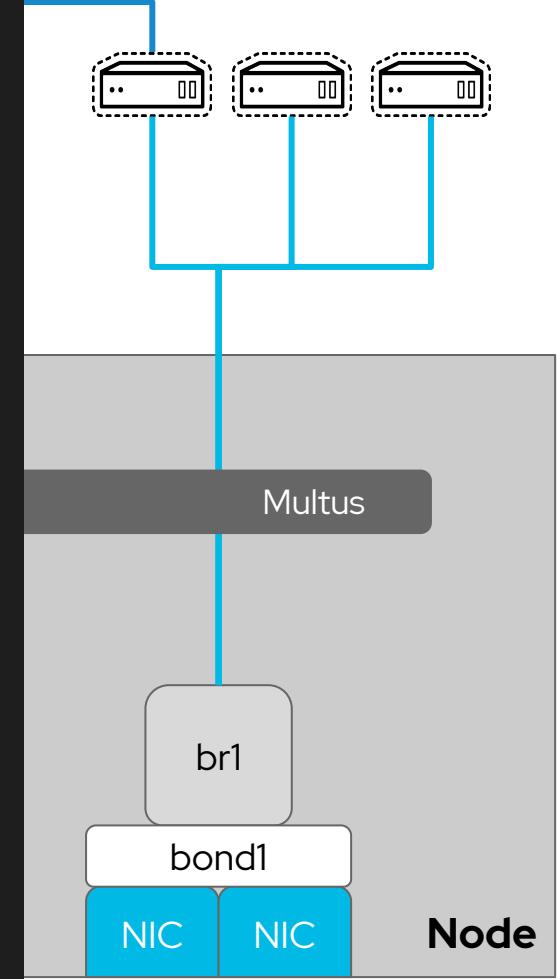
- NodeNetworkConfigurationPolicy (NNCP)
 - Nmstate operator CRD
 - Configure host network using declarative language
- Applies to all nodes specified in the `nodeSelector`, including newly added nodes automatically
- Update or add new NNCPs for additional host configs

```
1  apiVersion: nmstate.io/v1alpha1
2  kind: NodeNetworkConfigurationPolicy
3  metadata:
4    name: worker-bond1
5  spec:
6    nodeSelector:
7      node-role.kubernetes.io/worker: ""
8    desiredState:
9      interfaces:
10     - name: bond1
11       type: bond
12       state: up
13       ipv4:
14         enabled: false
15       link-aggregation:
16         mode: balance-alb
17         options:
18           miimon: '100'
19         slaves:
20           - eth2
21           - eth3
22       mtu: 1450
```



Host bridge configuration

```
1  apiVersion: nmstate.io/v1alpha1
2  kind: NodeNetworkConfigurationPolicy
3  metadata:
4    name: worker-bond1-br1
5  spec:
6    nodeSelector:
7      node-role.kubernetes.io/worker: ""
8    desiredState:
9      interfaces:
10        - name: br1
11          description: br1 with bond1
12          type: linux-bridge
13          state: up
14          ipv4:
15            enabled: false
16            bridge:
17              options:
18                stp:
19                  enabled: false
20                port:
21                  - name: bond1
```



Host network status

- Use the `NodeNetworkConfigurationEnactment` (NNCE) object to view status of NNCP application
- Further details of the node network state can be seen using the `NodeNetworkState` CRD
 - `oc get nns/node-name -o yaml`

```
1  API Version:  nmstate.io/v1alpha1
2  Kind:         NodeNetworkConfigurationEnactment
3  Name:         worker-1.owv.lab.lan.worker-br1-bond1
4  Status:
5    Conditions:
6      Last Heartbeat Time:  2020-07-08T20:15:46Z
7      Last Transition Time: 2020-07-08T20:15:46Z
8      Message:               successfully reconciled
9      Reason:                SuccessfullyConfigured
10     Status:                 True
11     Type:                  Available
12   Desired State:
13     Interfaces:
14       Bridge:
15         Options:
16           Stp:
17             Enabled:  false
18           Port:
19             Name:     bond1
20             Description: br1 with bond1
21           ipv4:
22             Enabled:  false
23             Name:     br1
24             State:    up
25             Type:     linux-bridge
```

Connecting Pods to networks

- Multus uses CNI network definitions in the NetworkAttachmentDefinition to allow access
 - Net-attach-def are namespaced
 - Pods cannot connect to a net-attach-def in a different namespace
- cnv-bridge and cnv-tuning types are used to enable VM specific functions
 - MAC address customization
 - MTU and promiscuous mode
 - sysctls, if needed
- Pod connections are defined using an annotation
 - Pods can have many connections to many networks

```
1  apiVersion: k8s.cni.cncf.io/v1
2  kind: NetworkAttachmentDefinition
3  metadata:
4    name: br1-public
5  annotations:
6    k8s.v1.cni.cncf.io/resourceName: bridge.network.kubevirt.io/br1
7  spec:
8    config: '{
9      "cniVersion": "0.3.1",
10     "name": "br1-public",
11     "plugins": [
12       {
13         "type": "cnv-bridge",
14         "bridge": "br1"
15       },
16       {
17         "type": "cnv-tuning"
18       }
19     ]
20   }'
```

```
1  kind: Pod
2  apiVersion: v1
3  metadata:
4    name: application-pod
5  annotations:
6    k8s.v1.cni.cncf.io/networks: bond1-br1
```

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		

VM disks in PVCs

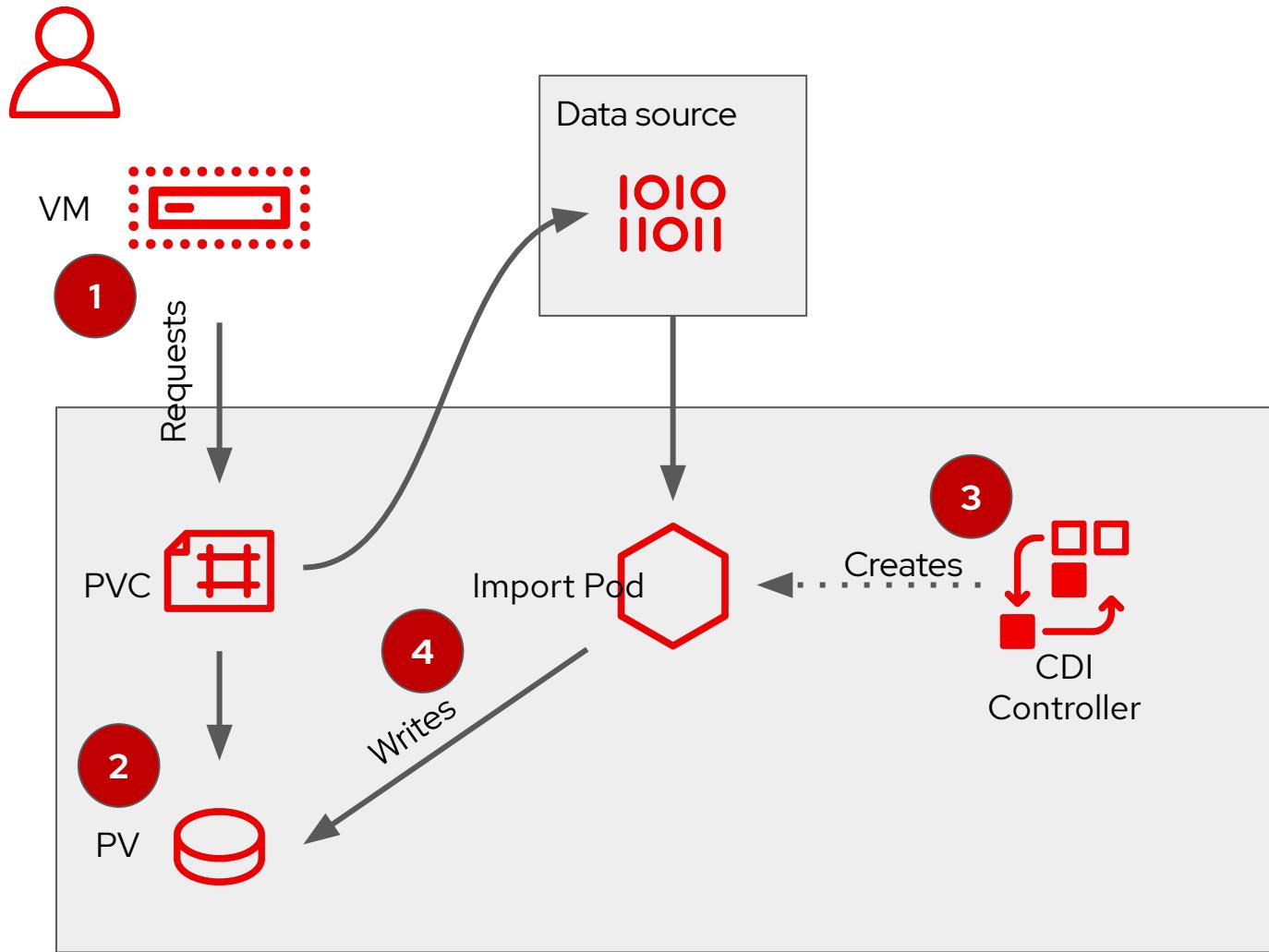
- VM disks on FileSystem PVCs are created as thin provisioned raw images
 - Thick provisioned disks are not created by CDI, may be possible manually
- Block PVCs are attached directly to the VM
- CSI operations, e.g. snapshot and clone, are not supported with VM disk PVCs
 - Use DataVolumes to clone VM disks
- PVC resize does not modify the size of the VM disk
 - Not currently supported
- Hot add is not supported (for any virtual hardware)

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'
```

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

Ephemeral Virtual Machine Disks

- VMs booted via PXE or using a container image can be “diskless”
 - PVCs may be attached and mounted as secondary devices for application data persistence
- VMs based on container images use the standard copy-on-write graph storage for OS disk R/W
 - Consider and account for capacity and IOPS during RHCOS disk sizing if using this type
- An `emptyDisk` may be used to add additional ephemeral capacity for the VM

```
1   spec:
2     domain:
3       disks:
4         - bootOrder: 1
5           disk:
6             bus: virtio
7             name: rootdisk
8       volumes:
9         - containerDisk:
10           image: registry.lab.lan:5000/fedora:31
11           name: rootdisk
```

Helper disks

- OpenShift Virtualization attaches disks to VMs for injecting data
 - Cloud-Init
 - ConfigMap
 - Secrets
 - ServiceAccount
- These disks are read-only and can be mounted by the OS to access the data within

```
1 spec:
2   domain:
3     devices:
4       - disk:
5         bus: virtio
6         name: cloudbinitdisk
7     volumes:
8       - cloudInitNoCloud:
9         userData: |-
10           #cloud-config
11           password: redhat
12           chpasswd: { expire: False }
13         name: cloudbinitdisk
```

Name	Source	Size	Interface	Storage Class	⋮
cloudbinitdisk	Other	-	VirtIO	-	⋮

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

Automated live migration

- OpenShift / Kubernetes triggers pod rebalance actions based on multiple factors
 - Pod rebalance applies to VM pods equally and will result in a live migration
- Eviction policies
 - Soft
 - Hard
- Pod descheduler
- Pod disruption policy

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

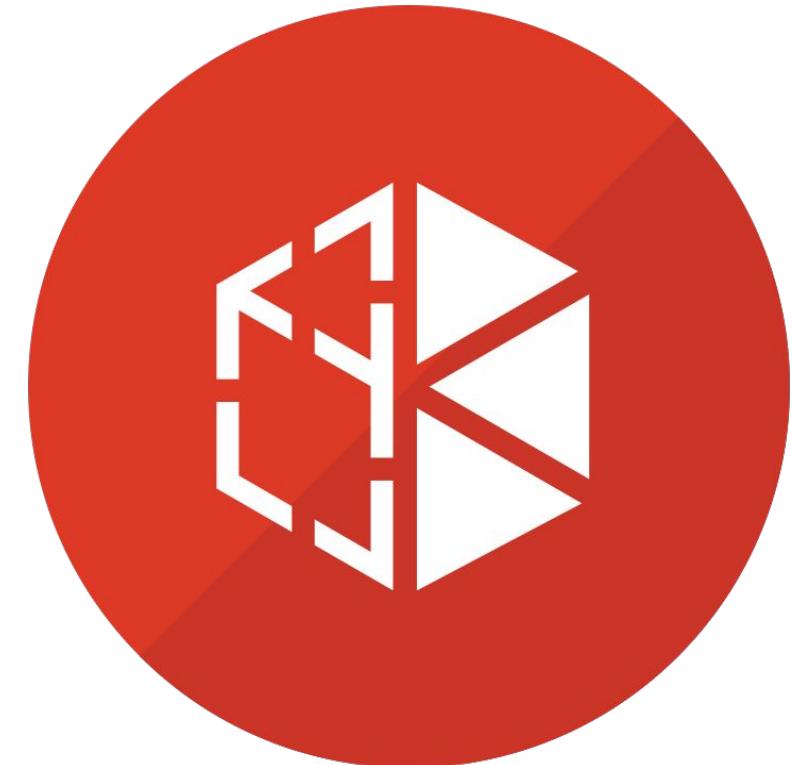
Terminology comparison

Feature	RHV	OpenShift Virtualization	vSphere
Where VM disks are stored	Storage Domain	PVC	datastore
Policy based storage selection	None	StorageClass	SPBM
Non-disruptive VM migration	Live migration	Live migration	vMotion
Non-disruptive VM storage migration	Storage live migration	N/A	Storage vMotion
Active resource balancing	Cluster scheduling policy	Pod eviction policy, descheduler	Dynamic Resource Scheduling (DRS)
Physical network configuration	Host network config (via nmstate w/4.4)	nmstate Operator, Multus	vSwitch / DvSwitch
Overlay network configuration	OVN	OCP SDN (OpenShiftSDN, OVNKubernetes, and partners), Multus	NSX-T
Host / VM metrics	Data warehouse + Grafana (RHV 4.4)	OpenShift Metrics, health checks	vCenter, vROps

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

Additional resources

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