VMs + Containers = The Perfect Wedding

Sreejith Anujan
sreejith@redhat.com
What is KubeVirt?
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
KubeVirt

- 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 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 / Fedora / CentOS Stream**
- libvirt and qemu are mature, have high performance, provide stable abstractions, and have a minimal overhead.

**Security - Defense in depth**
- Immutable CoreOS by default, SELinux MCS, plus KVM isolation
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 Kubernetes
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
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
Create Virtual Machine - Storage

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

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

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

**oVirt**
- oVirt Host
  - oVirt Console / CLI
  - vdsms
  - libvirt
  - QEMU / KVM
  - VM

**KubeVirt**
- CoreOS Host
  - OpenShift Console / CLI
  - kubelet
  - libvirt
  - QEMU / KVM
  - VM

**OpenStack**
- Compute
  - OpenStack Horizon / CLI
  - nova-compute
  - libvirt
  - QEMU / KVM
  - VM
Architectural Overview

Cluster Services

- API Server
  - virt-controller

(DaemonSet) Pod
- virt-handler

VM Pod
- virt-launcher
  - libvirtd
  - VM

Nodes

- VM Pod
- Other Pod(s)
  - container 1
  - container 2
  - container n
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