redhat.

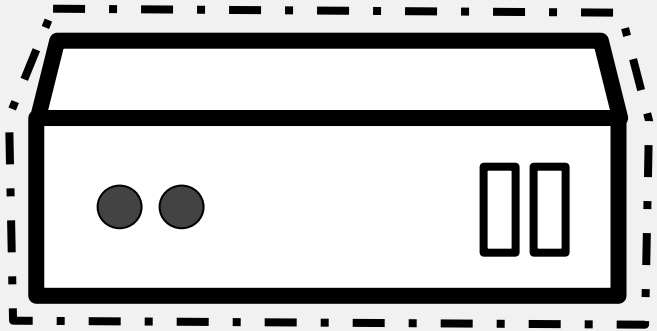
Convergence of VM and containers orchestration using KubeVirt

Chunfu Wen
chwen@redhat.com

Agenda

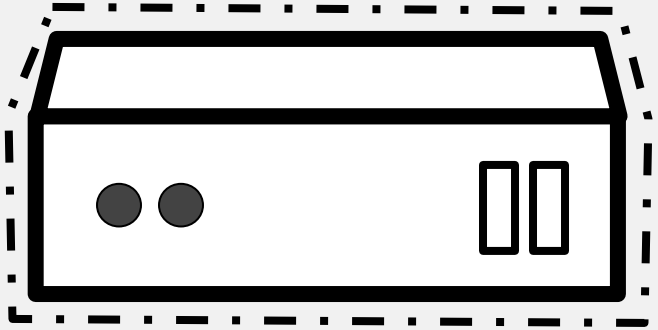
- Context Introduction
- What Is Kubevirt And How It Feel
- Kubevirt Architecture And Design
- Demo

FIRST A LITTLE HISTORY

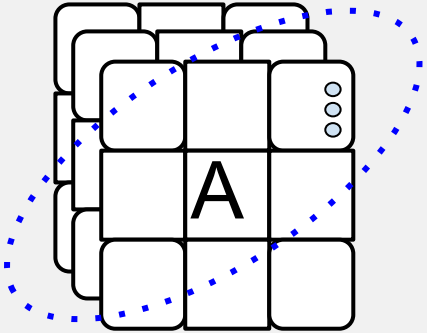


Virtual Machines

FIRST A LITTLE HISTORY

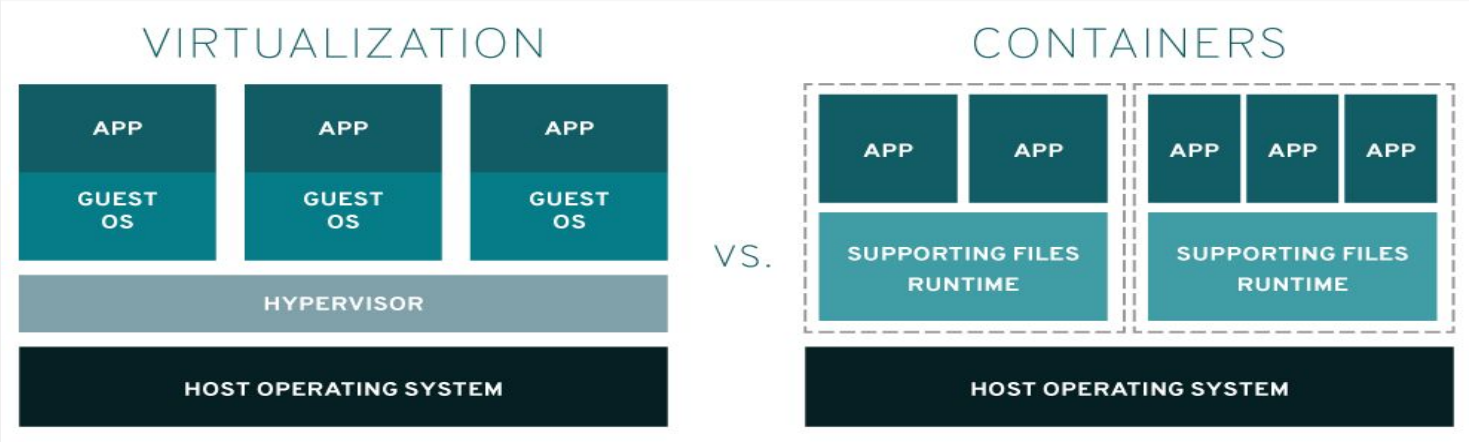


Virtual Machines



Containers

Virtualization VS Containers



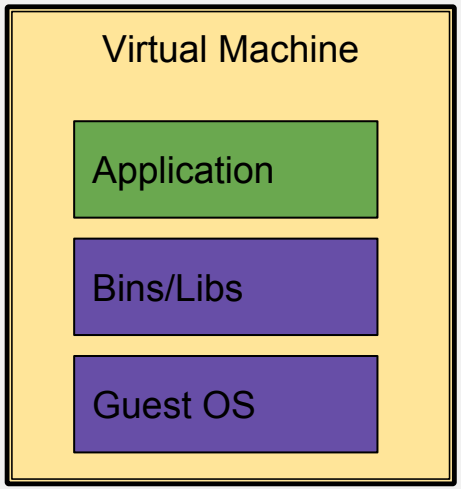
VS

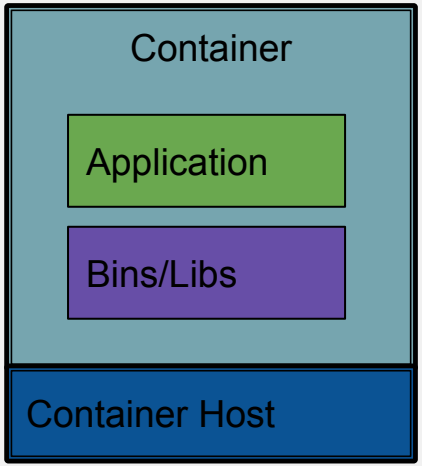
VM virtualizes the hardware

container isolates the process

redhat.

VIRTUAL MACHINES VS CONTAINERS

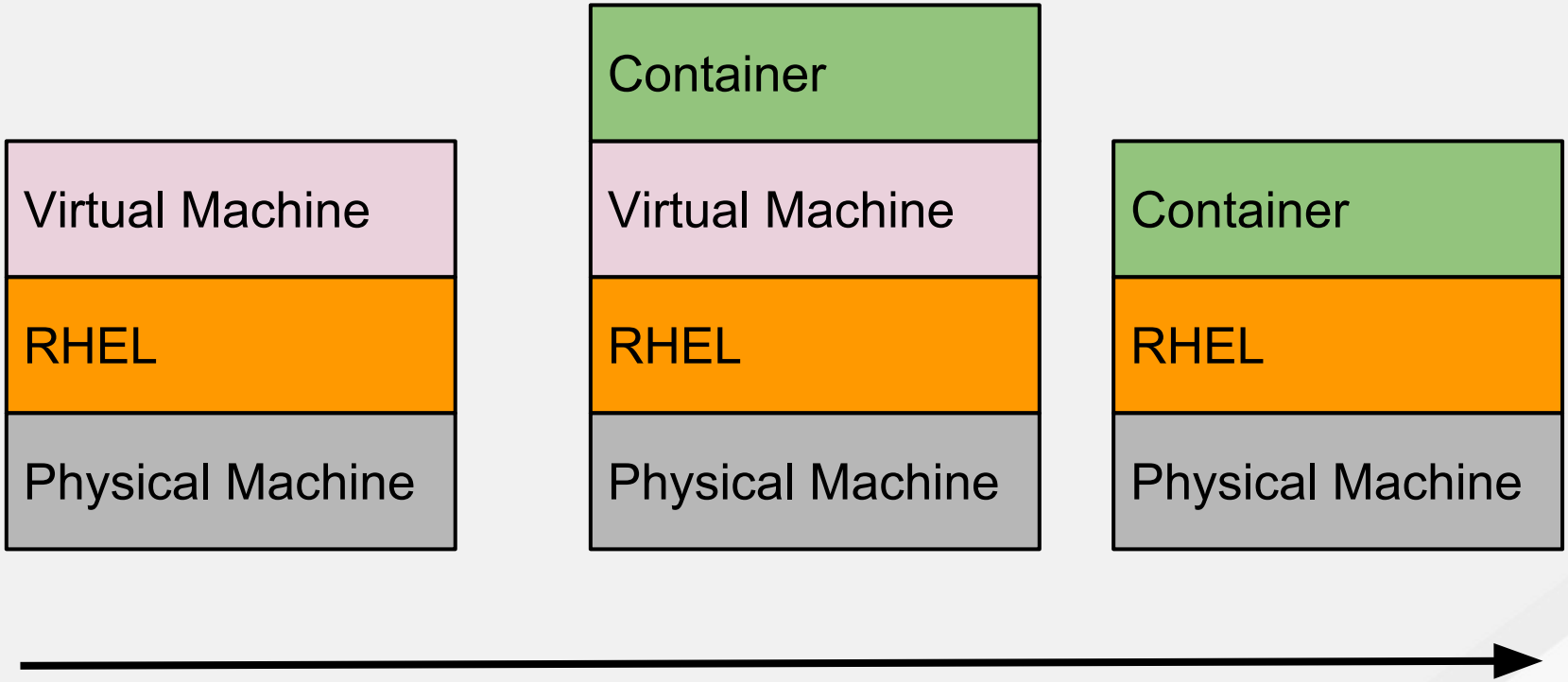




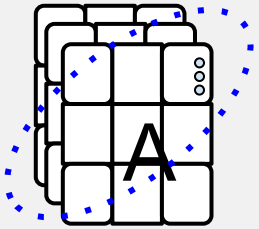
Each of these attributes can be a positive or a negative for a given workload.

Increasingly organizations have a mix of both.

EXISTING SYSTEMS TREAT THESE SEPARATELY

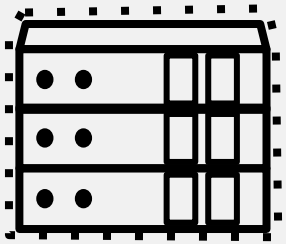


WHAT ABOUT EXISTING WORKLOADS?



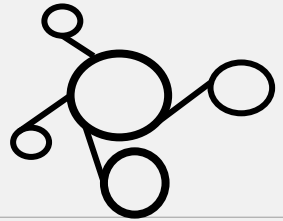
CONTAINER INFRASTRUCTURE AND ORCHESTRATION

Container Application and Kubernetes orchestration as provided by OpenShift are becoming the standard for new applications.



VIRTUALIZED WORKLOADS

Virtualized Workloads are not going anywhere fast! Business reasons (cost, time to market) and technical reasons (older/different operating system)



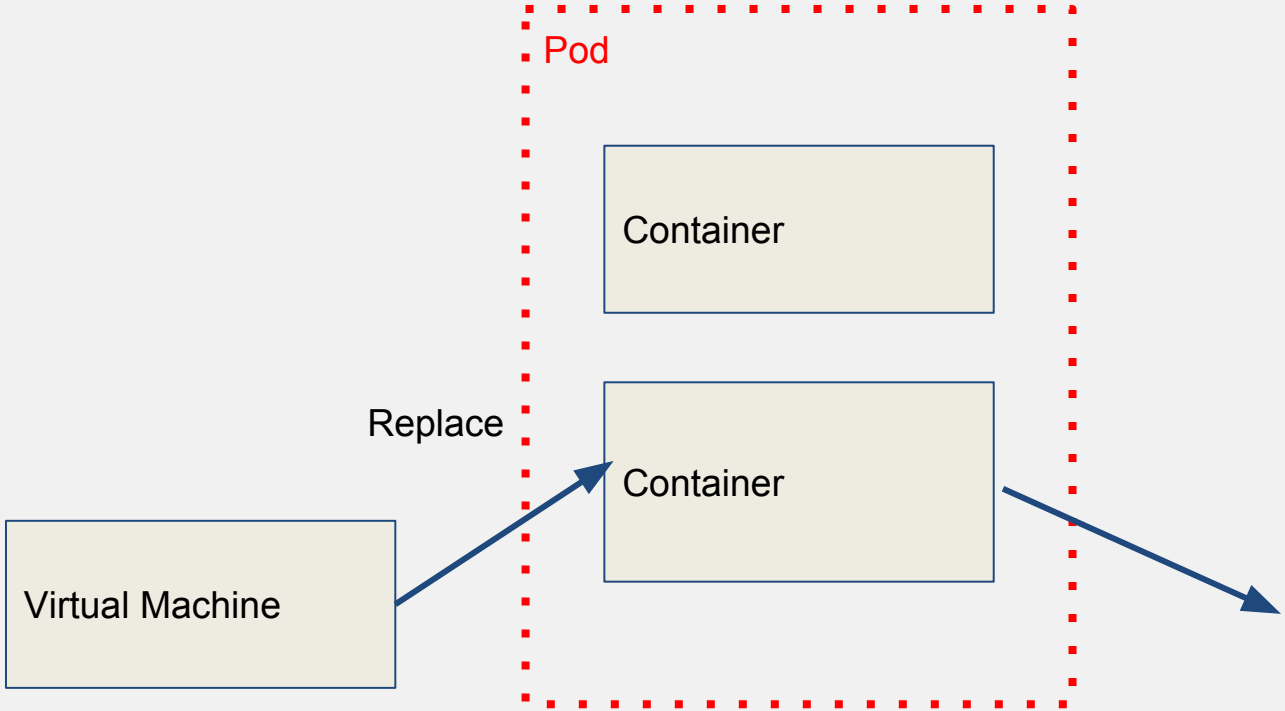
CONVERGING INFRASTRUCTURE

How can we bring these two worlds closer together?

So,if....

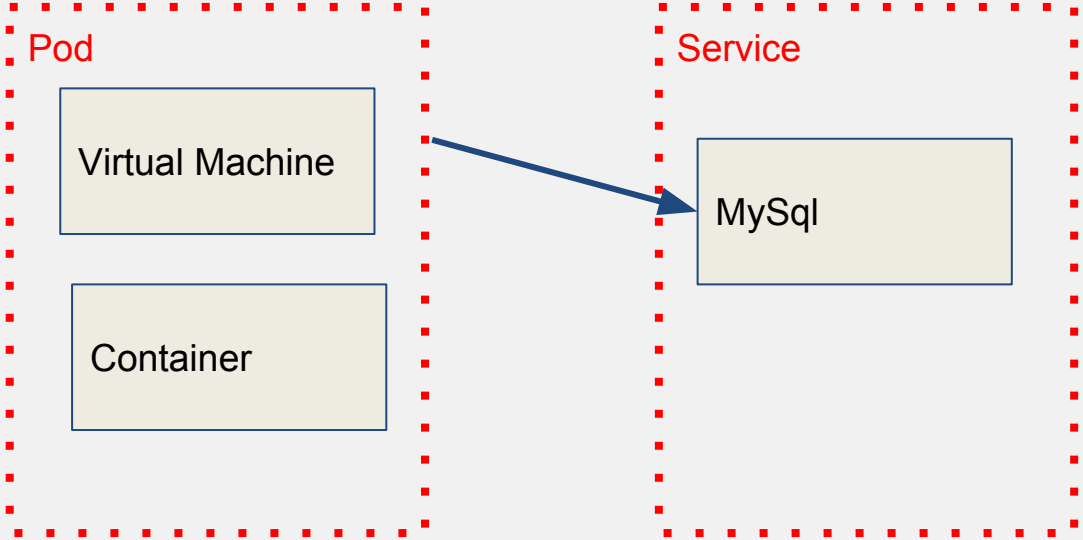
- VMS are just user processes
  - VM and containers already share some isolation technologies ,selinux and cgroup
- Kubernetes manage clustered containers,which are user processes
- Get a converged infrastructure

Concept of Proof



One Typical Benefit Scenario

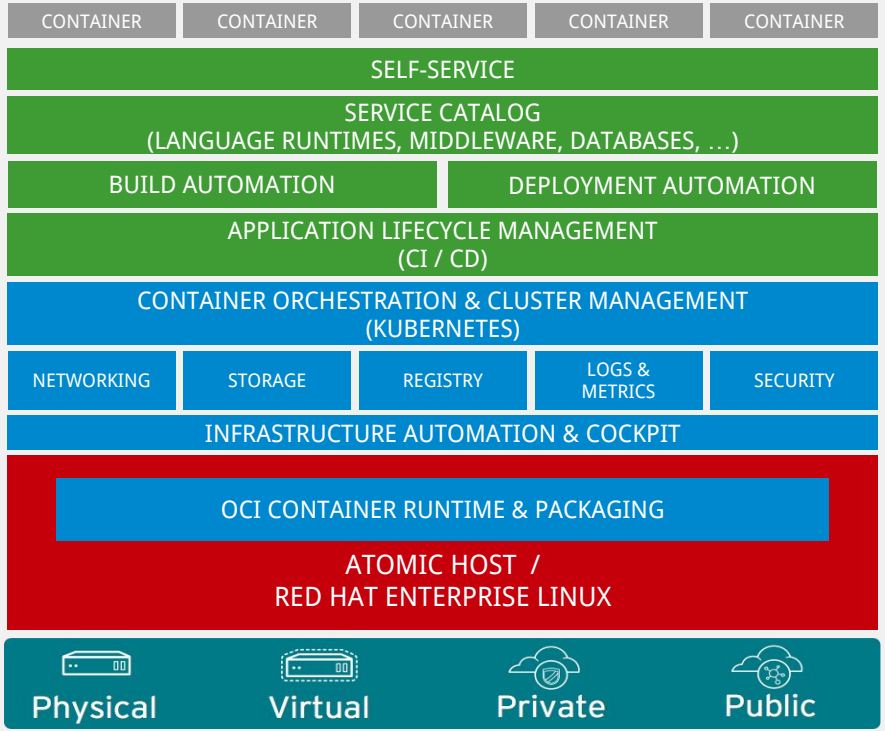
- Windows guest VM access containerized mysql



OpenShift = Enterprise K8s + Docker

Build, Deploy and Manage Containerized Apps

RED HAT® OPENSHIFT



Enter kubevirt

Kubenest+Libvirt?

“Virtual Machine management addon to Kubernetes that extends Kubernetes in a way that allows it to schedule VM workloads side by side with container workloads.”

Explore more

- Extends OpenShift/Kubernetes to support orchestration of virtual machine workloads alongside application container workloads in the same cluster.
- Developer centric approach to virtualization that drops directly into existing Openshift/Kubernetes clusters
  - Implemented as a CustomResouceDefinition
- Aims to provide as Kubernetes-native an experience to working with VMs As possible
  - Integrates directly with other Kubernetes concepts (like Persistent Volumes, Pod networking)
  - Manager virtual machines like Pods!
- Scheduling, networking, and storage are all delegated to Kubernetes, while KubeVirt provides the virtualization functionality

How Kubevirt Feel?

FIrst Look At Pod Object

```
kind: Pod
metadata:
  name: nginx
  labels:
    name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:latest
    ports:
    - containerPort: 80
  nodeSelector:
    cpu: fast
status:
  phase: Running
```

What is a Pod?

*“A pod (as in a pod of whales or pea pod) is a group of one or more containers (such as Docker containers), with shared storage/network, and a specification for how to run the containers.” **

* *https://kubernetes.io/docs/concepts/workloads/pods/pod/#what-is-a-pod*

The VirtualMachine Object

```
kind: VirtualMachine
metadata:
  name: testvm
spec:
  domain:
    devices:
      type:
PersistentVolumeClaim
      device: disk
      source:
        name: myVolumeClaim
  nodeSelector:
    cpu: fast
status:
  phase: Running
```

We have the typical Pod like structure:

- Metadata section
- Specification section
- Typical Pod features like
  - nodeSelector
  - affinity
- Status section

Behind the scene a Pod is created, scheduled and we make sure that the VM starts correctly inside.

CustomResourceDefinition:vm-resource.yaml

```
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  creationTimestamp: null
  labels:
    kubevirt.io: ""
  name: virtualmachines.kubevirt.io
spec:
  group: kubevirt.io
  names:
    kind: VirtualMachine
    plural: virtualmachines
    shortNames:
    - vm
    - vms
    singular: virtualmachine
  scope: Namespaced
  validation:
```

CustomResourceDefinition Extend the Kubernetes API:

- create a new CustomResourceDefinition (CRD), the Kubernetes API Server reacts by creating a new RESTful resource path
- After the CustomResourceDefinition object has been created, you can create custom objects. Custom objects can contain custom fields

The Typical Kubectl Feeling

```
kind: VirtualMachine
metadata:
  name: testvm
spec:
  domain:
    devices:
      graphics:
      - type: spice
      consoles:
      - type: pty
```

Typical Pod commands:

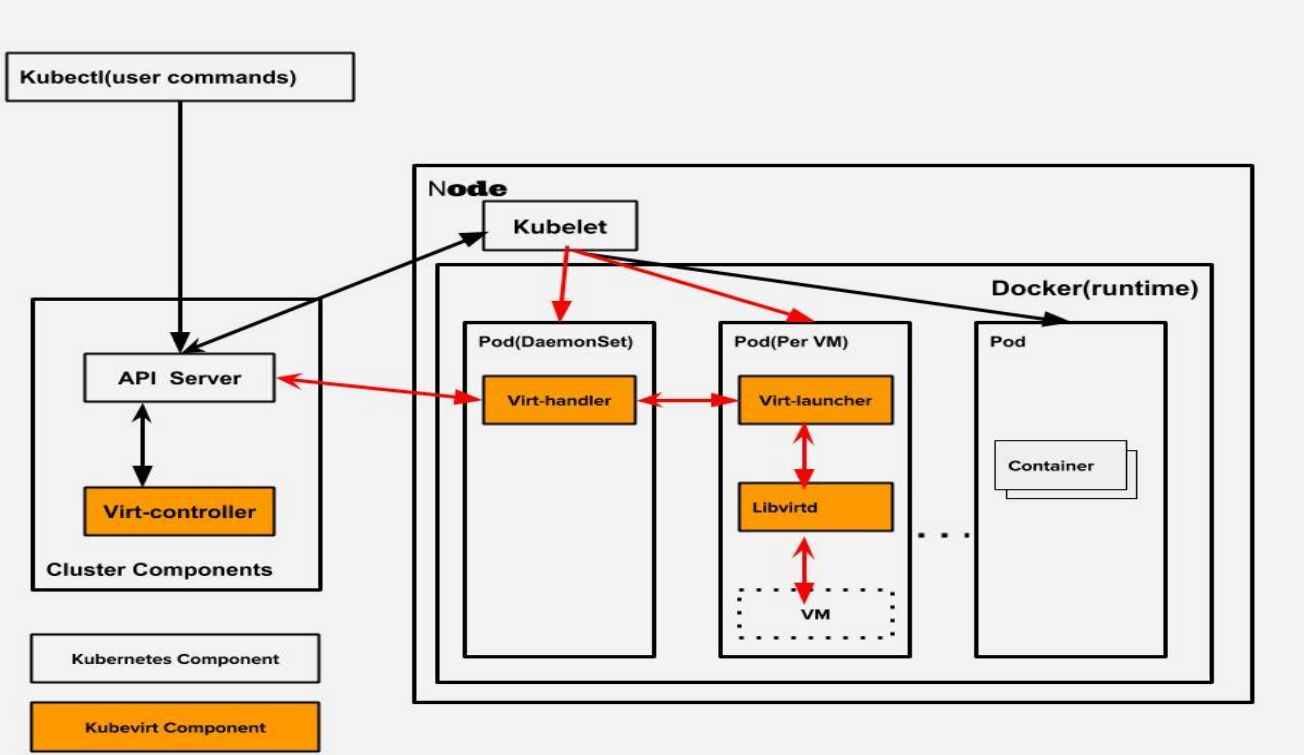
- kubectl create -f mypodspec.yaml
- kubectl delete mypod
- kubectl exec mypod -it /bin/bash

Typical Virtual Machine commands:

- kubectl create -f myvmspec.yaml
- kubectl delete testvm
- kubectl plugin virt **console** testvm
- kubectl plugin virt **spice** testvm

Architecture

Kubevirt Architecture Internal



Components(1/5)

virt-controller:

- This controller is responsible for monitoring the VM (CRDs) and managing the associated pods. Currently the controller will make sure to create and manage the life-cycle of the pods associated to the VM objects.
- A VM object will always be associated to a pod during it's life-time, however, due to i.e. migration of a VM the pod instance might change over time.

Components(2/5)

VM (CRD):

- Machine type
- CPU type
- Amount of RAM and vCPUs
- Number and type of NICs

Components(3/5)

virt-launcher:

- For every VM object one pod is created. This pod's primary container runs the virt-launcher KubeVirt component.
- Virt launcher will take care to launch a VM process for every pod which is associated to a VM object whenever it is getting scheduled on a host.
- The main purpose of the virt-launcher Pod is to provide the cgroups and namespaces, which will be used to host the VM process.
- Virt-handler signals virt-launcher to start a VM by passing the VM's CRD object to virt-launcher. virt-launcher then uses a local libvirtd instance within its container to start the VM. From there virt-launcher monitors the VM process and terminates once the VM has exited.

Components(4/5)

virt-handler:

- Every host needs a single instance of virt-handler. It can be delivered as a DaemonSet.
- Virt-handler is also reactive and is watching for changes of the VM object, once detected it will perform all necessary operations to change a VM to meet the required state.
- Report domain state and spec changes to the cluster.

Components(5/5)

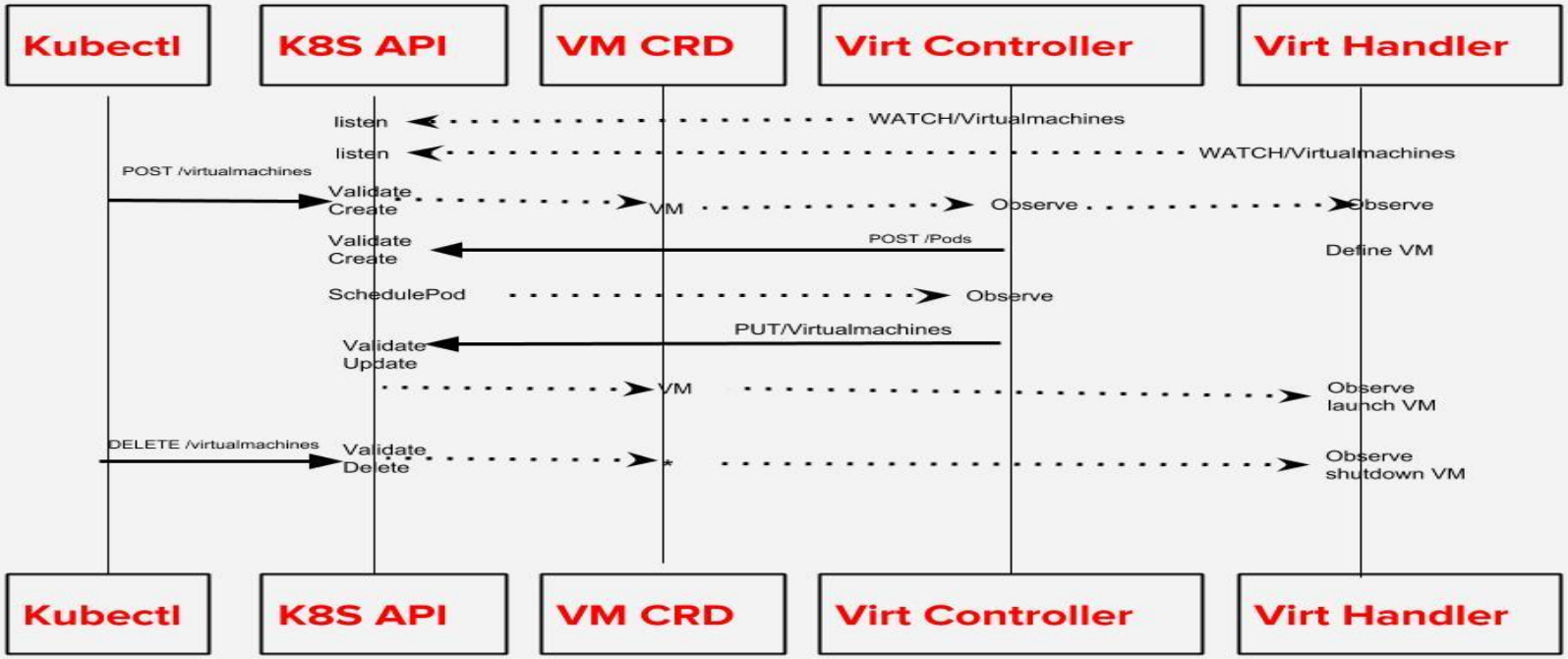
libvirtd:

- An instance of libvirtd is present in every VM pod. virt-launcher uses libvirtd to manage the life-cycle of the VM process..

DEMO

Deep dive into it

Workflow: Create and Delete a VM



Sequences

- A client posts a new VM definition to the K8s API Server.
- The K8s API Server validates the input and creates a VM custom resource definition (CRD) object.
- The virt-controller observes the creation of the new VM object and creates a corresponding pod.
- Kubernetes is scheduling the pod on a host.
- The virt-controller observes that a pod for the VM got started and updates the nodeName field in VM object. After the nodeName is set, the responsibility transitions to the virt-handler for any further action.
- The virt-handler observes that a VM got assigned to host where it is running on.
- The virt-handler is using the *VM Specification* and signals the creation of the corresponding domain using a libvirtd instance in the VM's pod.
- A client deletes the VM object through the virt-api-server.
- The virt-handler observes the deletion and turns off the domain.

Leveraging K8s Functionality: network, storage and computing

Computing

```
apiVersion: kubevirt.io/v1alpha1
kind: VirtualMachine
metadata:
  creationTimestamp: null
  name: vm-windows
spec:
  domain:
        ….
        cpu:
        cores: 2
        resources:
        requests:
        memory: 64M
```

- Cores specify vcpu in VM
- Memory specify required memory

Networking

```
func SetupDefaultPodNetwork(domain
*api.Domain) error {
      ...
  reparePodNetworkInterfaces(vif,
podNicLink);
  // Start DHCP Server
     ...
  go Handler.StartDHCP(vif,
fakeServerAddr)
    ...
 }
```

- SetupDefaultPodNetwork will prepare the pod management network to be used by a virtual machine
- Virtual machine will own the pod network IP and MAC.
- Pods MAC address will be changed to a random address and IP will be deleted.
- DHCP server will be started and bounded to the macvlan interface to server the original pod ip to the guest OS

https://github.com/kubevirt/kubevirt/blob/master/docs/libvirt-pod-networking.md

Disk

```
apiVersion: kubevirt.io/v1alpha1
kind: VirtualMachine
  ...
 domain:
        ...
        devices:
        disks:
        - disk:
        bus: virtio
        name: pvcdisk
        volumeName: pvcvolume
        ...
 volumes:
 - name: pvcvolume
        persistentVolumeClaim:
        claimName: disk-vpc
        ...
```

- Virtual Machines are able to have disks attached. They are not always required, but have some value if you need to persist data.
- Kubernetes provides persistent storage through Persistent Volumes and Claims

Call for KubeVirt participants

- GitHub:
  - https://github.com/kubevirt/kubevirt
  - https://lc32018.sched.com/event/ERAW/convergence-of-virtual-machines-and-containers-orchestration-using-kubevirt-chunfu-wen-red-hat?iframe=yes&w=100%&sidebar=yes&bg=no#
- Mailing List:
  - https://groups.google.com/forum/#!forum/kubevirt-dev
- IRC:
  - #kubevirt on irc.freenode.net

redhat.

THANK YOU

Chunfu Wen-温春福
chwen@redhat.com