



# **What is KubeVirt Kubernetes Based RHV**

**RHVH QE Team(yzhao)  
Mar 31, 2017**

# About Kubernetes

1. Kubernetes is an open source system for manage containerized applications across multiples hosts, providing basic mechanisms for deployment, maintenance, and scaling of applications.
2. Kubernetes works based Go and Docker environment.

# About KubeVirt

1. KubeVirt is a virtual machine management add-on for Kubernetes. The aim is to provide a common ground for virtualization solutions on top of Kubernetes.
2. Now, KubeVirt extends Kubernetes by adding additional virtualization resource type (especially the VM type) through Kubernetes's third party resource concept. By using this mechanism, the Kubernetes API can be used to manager these VM resource alongside all other resources Kubernetes provides.

# KubeVirt Extended

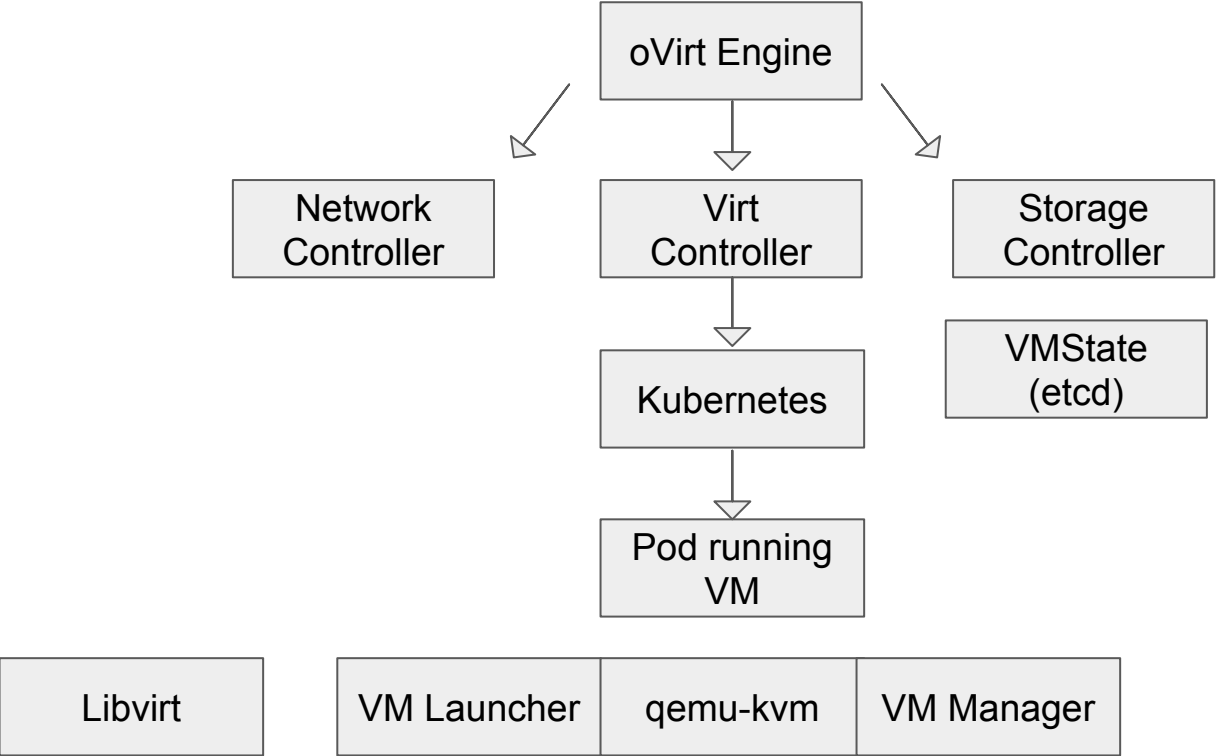
Does what different from Kubernetes:

1. Create a predefined VM
2. Schedule a VM on a Kubernetes cluster
3. Launch a VM
4. Stop a VM
5. Delete a VM
6. \* Use Cockpit to manage resource like hosts, VMs ,logs, services, pods...

# Example Flow: Create and delete a VM

1. A client posts a new VM definition to the K8s API Server
2. The K8s API Server validates the input and creates a VM 3rd party resource (TPR) object.
3. The virt-controller observes the creation of the new VM object and creates a corresponding pod.
4. Kubernetes is scheduling the pod on a host
5. The virt-controller observes that a pod for the VM got started and updates the nodeName field in the VM object. Now that the nodeName is set, the responsibility transitions to the virt-handler for any further action.
6. The virt-handler (*DaemonSet*) observes that a VM got assigned to the host where it is running on.
7. The virt-handler is using the *VM Specification* and creates a corresponding domain using the local libvirtd instance.
8. A client deletes the VM object through the virt-api-server.
9. The virt-handler observes the deletion and turns off the domain.

# End Goal Architecture



# Legend

Network  
Controller

Handles network related logic which is missing elsewhere

Storage  
Controller

Handles storage related logic which is missing elsewhere

VMState  
(etcd)

Keeps VM definitions, used for requests and updates.

Virt  
Controller

Relevant portions of Engine logic for abstracting virt over Kubernetes

VM Launcher

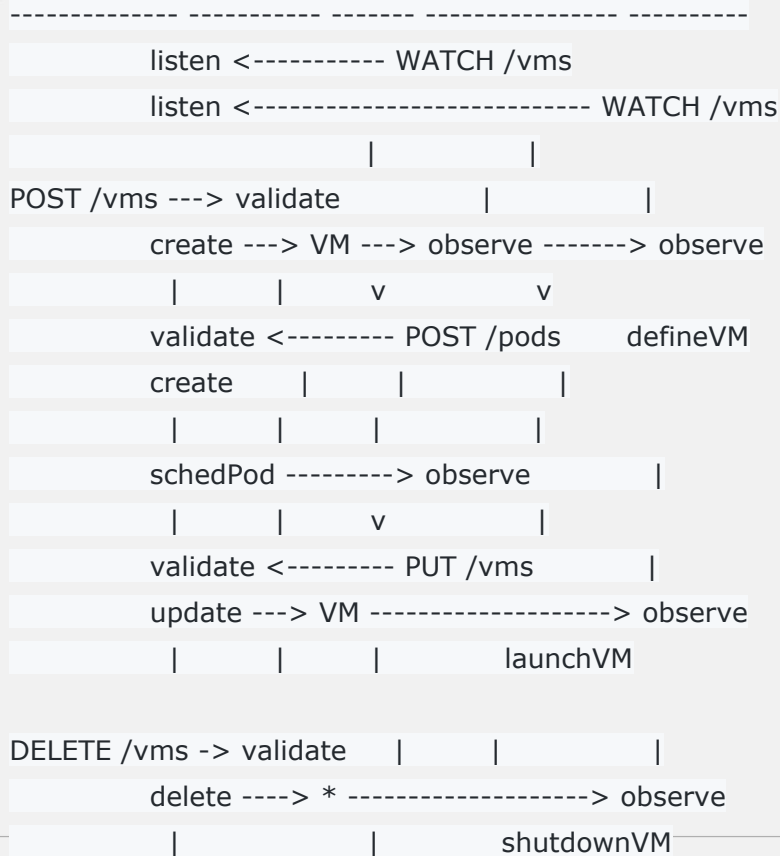
A simple process used to launch QEMU and handle basic interrupts

VM Manager

Handles VMs at the host level. Should cover for various VM related actions.

# Flow

Client      K8s API      VM TPR    Virt Controller    VM Handler





# Cockpit--resource like pods, logs, services...

The screenshot displays the Cockpit web interface for a CentOS Linux system. The top navigation bar includes 'Dashboard' and 'Cluster' tabs. A sidebar on the left lists various system components: Overview, Nodes, Containers, Topology, Details, and Volumes. The main content area is titled 'All Projects' and shows a summary of cluster resources: 29 Pods (with 1 error icon), 5 Volumes (All in use), and 1 Node (All healthy). Below this, the 'Services' section contains a table listing various services and their status.

Name	Address	Containers	Namespace
frontend	10.98.100.155:80	0	default
haproxy-service	10.103.85.44:8184	2	default
iscsi-demo-target	10.96.81.22:3260	1	default
kubvirt-cockpit	10.99.247.217:9090	1	default
redis-master	10.96.131.206:6379	0	default
redis-slave	10.104.28.160:6379	0	default
spice-proxy	10.105.221.50:3128	2	default
virt-api-service	10.102.181.155:8183	2	default

The 'Nodes' section on the right shows a single node named 'master' with 33 containers.

# Cockpit-- VMs

FEDORA Oops! root

192.168.200.2 Dashboard

System  
Services  
Logs  
**Virtual Machines**  
Accounts  
Terminal

## Virtual Machines

Name	Connection	State
▼ testvm	192.168.200.2:8184	scheduling

[Overview](#) [Usage](#) [Disks](#) [Migration](#) [Shut Down](#) ▼

State: scheduling

Memory: 64 MIB

vCPUs: 4

ID: testvm

OS Type: Linux

Autostart: enabled

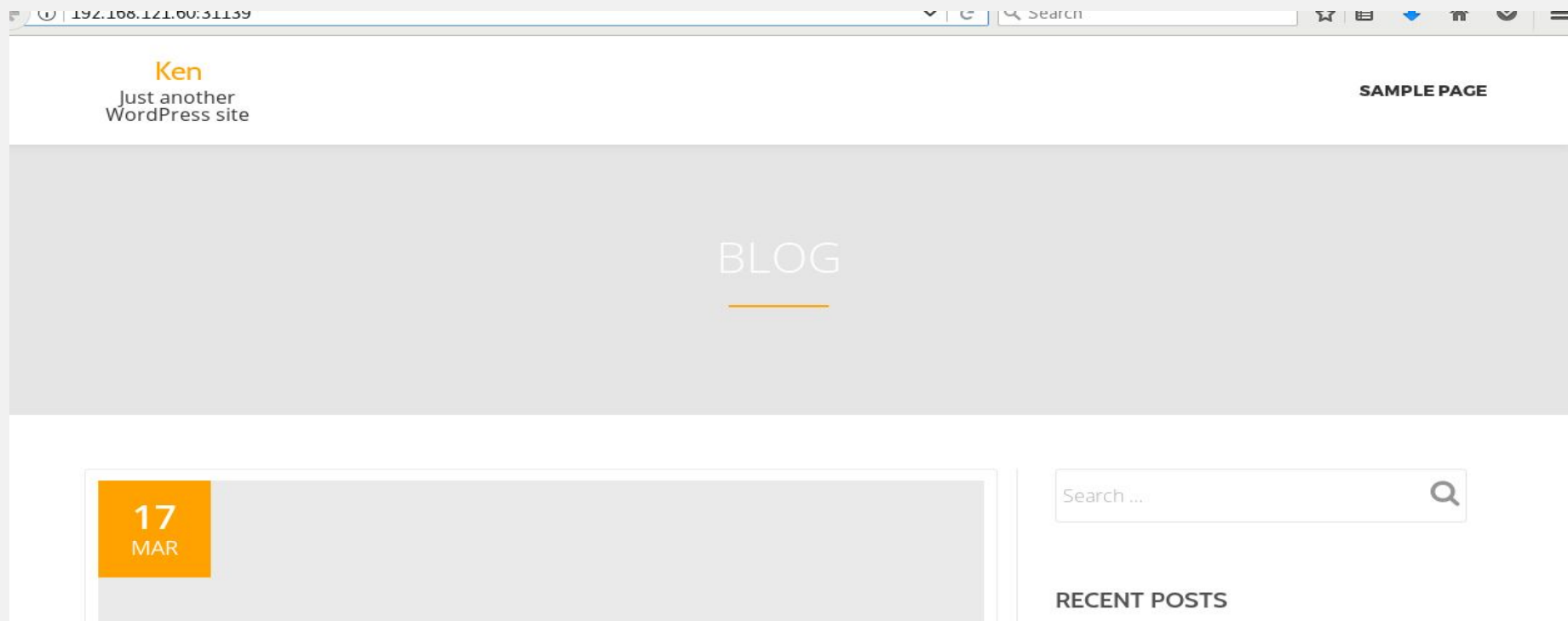
# Master, Terminal -- pods, VMs

```
[root@master ~]# kubectl get pods --all-namespaces
```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
default	haproxy	1/1	Running	15	13d
default	haproxy-2737689567-lphrs	1/1	Running	0	1d
default	iscsi-demo-target-tgtd-474016847-gd8hf	1/1	Running	0	1d
default	kubevirt-cockpit-3405564194-wy0gd	1/1	Running	0	1d
default	spice-proxy	1/1	Running	6	13d
default	spice-proxy-3576158421-k0jw7	1/1	Running	0	1d
default	virt-api	1/1	Running	5	13d
default	virt-api-988471961-u8lmq	1/1	Running	0	1d
default	virt-controller	1/1	Running	6	13d
default	virt-controller-1177942624-tzrsg	1/1	Running	0	1d
default	virt-handler-vwb39	1/1	Running	0	1d
default	wordpress-1618093523-de8zj	1/1	Running	8	42d
default	wordpress-mysql-2379610080-5p0e6	1/1	Running	8	42d
kube-system	dummy-2088944543-lx18s	1/1	Running	8	43d
kube-system	etcd-master	1/1	Running	9	43d
kube-system	kube-apiserver-master	1/1	Running	13	43d
kube-system	kube-controller-manager-master	1/1	Running	10	43d
kube-system	kube-discovery-3670776889-793t6	1/1	Running	9	43d
kube-system	kube-dns-2924299975-81c0c	4/4	Running	30	43d
kube-system	kube-proxy-ufd6z	1/1	Running	8	43d
kube-system	kube-scheduler-master	1/1	Running	10	43d
kube-system	weave-net-tvs1l	2/2	Running	18	43d

```
root@master:~  
[root@master ~]# kubectl get vms  
NAME          KIND  
testvm        VM.v1alpha1.kubevirt.io  
[root@master ~]#
```

# Simple App Example



# Try KubeVirt

Follow the Github : <https://github.com/kubevirt/kubevirt/blob/master/docs/getting-started.md>

Github: <https://github.com/kubevirt/kubevirt>

If any problems, discuss with me and improve together.