Virtualization in Containers

Marek Libra
Senior Software Engineer
Red Hat

Linux Day Milano 2019

TL;DR

Kubernetes operates containers

TL;DR

Kubernetes operates containers

and virtual machines.

Application Runtime

Bare metal vs. separated runtime environment

- Isolation - security, stability, conflicts
- Costs - share people and resources
- Simplicity - create & run
- Scalability

About Marek Libra

- Passionate software developer since 2000
- KubeVirt
- oVirt
- https://github.com/mareklibra
- https://www.linkedin.com/in/mareklibra

Linux Container

- **Isolated** native process with runtime environment
  - Shared kernel, lightweight
  - CPU, memory effective

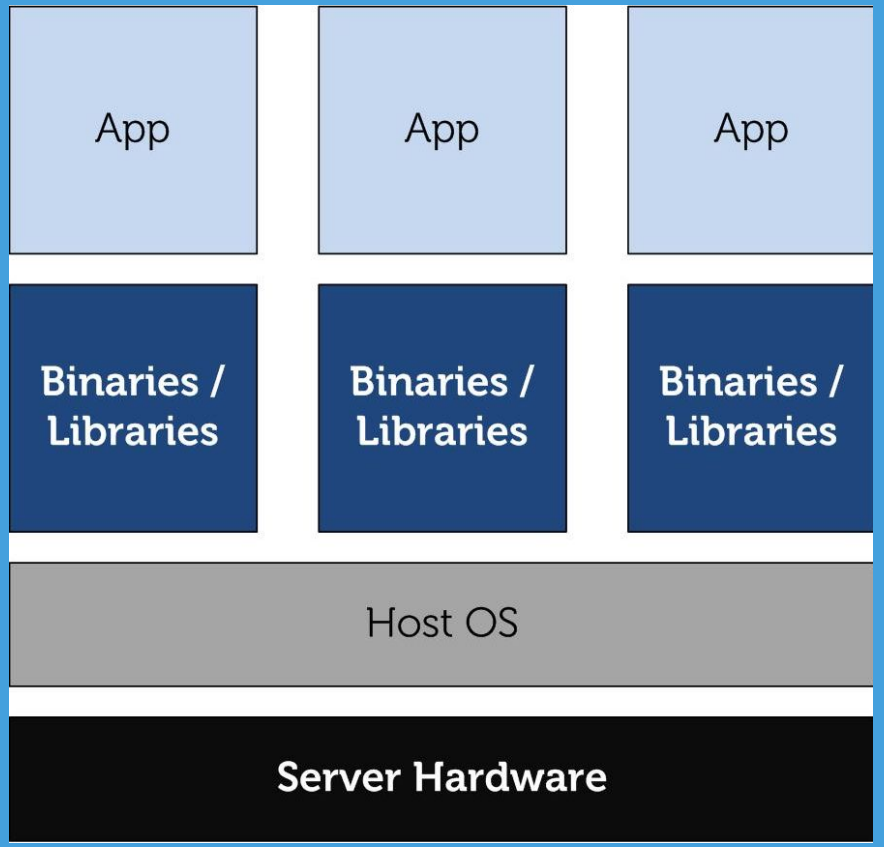
Virtual Machine

- A process emulating runtime HW
  - Multiple distinct kernels on a single host
  - Independent process schedulers
  - Wider flexibility

Linux Container - How

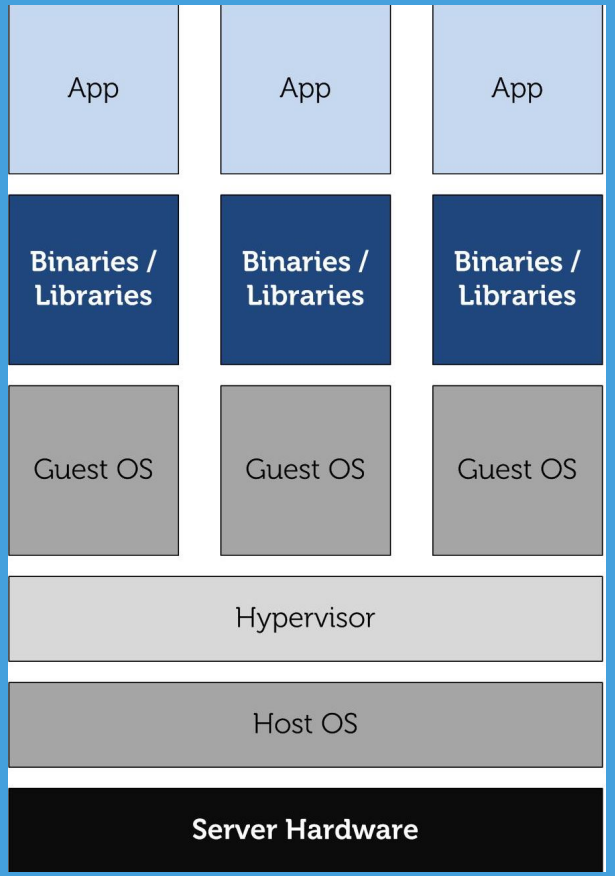
- Standard Linux Process
- Cgroups
  - control and limit resource usage
- Namespaces
  - IPC, Net, Mount, PID,

    User, UTS, Cgroup
- Image
- CRI-O, Docker, rkt, containerd ?



Virtual Machine

- Hypervisor emulates hardware
  - Multiple kernels
  - Memory and CPU ineffective
  - Emulation of devices
  - Guest-OS flexibility
- Multiple implementations
  - KVM/Qemu, Xen, VMware, ...
  - Libvirt



Virtual Machine

- Multiple kernels
- Boots OS, inner scheduler
- Legacy applications
- Hard to scale
- Win games on Linux unmet wish

vs

Container

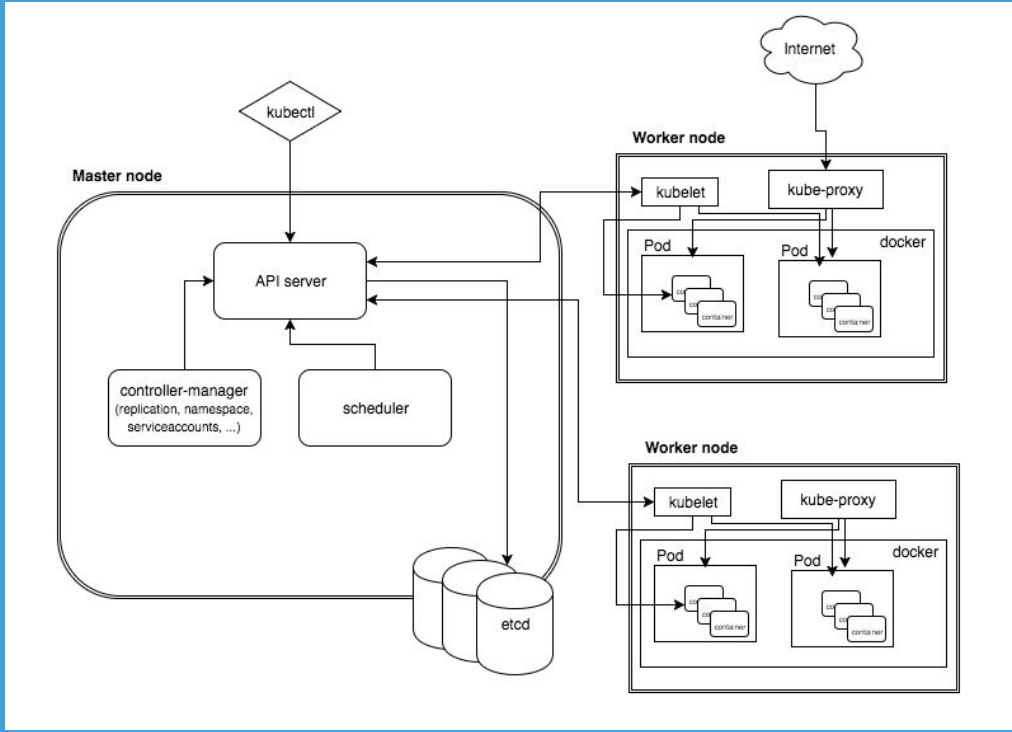
- Single shared kernel
- Just another process
- Application environment included
- Scalable, closer to SOA
- No false promises

Container Execution

- Single-host
- Distributed
- On premise vs. cloud (AWS, GCP, ...)

Kubernetes

- Industry standard, huge community
- Pod grouping containers
  - Shared Network namespace
  - Distinct PID, mount, IPC
- Declarative REST API/etcd
- Controllers
  - Scheduler, kubelet, ...
- Storage, Networking



Kubernetes Command Line

- Declarative instead of Imperative
- kubectl [command] [TYPE] [NAME] [flags]
- kubectl get pods
- kubectl get pod my-pod-name -o yaml
- Create, describe, delete, patch, ...

```
$ kubectl get pod myapp-pod -o yaml

apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
        app: myapp
spec:
  containers:
  - name: myapp-container
        image: busybox:latest
        command: ['sh', '-c', 'echo Hello
                  Kubernetes! && sleep 3600']
```

OKD / OpenShift

- PaaS based on Kubernetes
  - Security - tighter SCC, default RBAC, OAuth with infra apps
  - ImageStreams
  - Cmd-line plus Web User Interface
  - Enterprise ready open source - support, installation
  - Web user interface

Virtual Machines and Containers Workload

- Kubernetes ~ containers
- Virtual machines
  - Legacy or heterogeneous applications
  - Complex workloads
- Infrastructure reusability
  - Hardware
  - Operations

KubeVirt

- Addon to Kubernetes
- Run virtual machine workloads as / along containers
- Legacy applications in new era of clouds and container native infra
  - Mixed workloads - single pipeline for both
  - Gradual decomposition of VMs into containers
  - Shared infrastructure to decrease operating costs
  - Freedom of operating system choice
  - Increased isolation

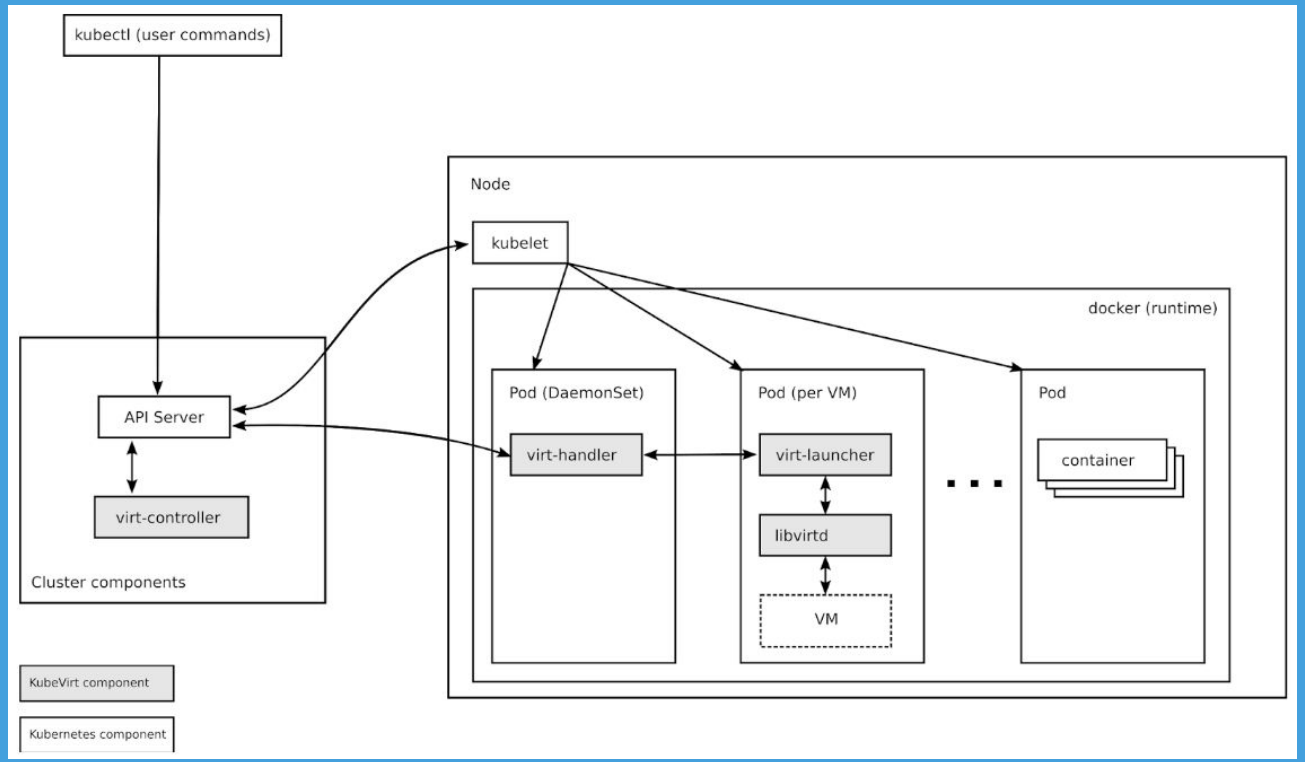
KubeVirt - How

Custom Resource Definition

Virt-controller

Virt-handler

libvirtd/qemu in Pod



KubeVirt VirtualMachine

- Declarative
- Defaults
- virt-controller/virt-handler

```yaml
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachine
metadata:
  labels:
    kubevirt.io/vm: vm-fedora
  name: vm-fedora
spec:
  running: false
  template:
    metadata:
      labels:
        kubevirt.io/os: fedora28
        kubevirt.io/vm: vm-fedora
    spec:
      domain:
        devices:
          disks:
            - disk:
                bus: virtio
              name: containerdisk
              volumeName: registryvolume
            - disk:
                bus: virtio
              name: cloudinitdisk
              volumeName: cloudinitvolume
        cpu:
          cores: 2
        resources:
          requests:
            memory: 256M
      terminationGracePeriodSeconds: 0
      volumes:
        - name: registryvolume
          containerDisk:
            image: 'kubevirt/fedora-cloud-registry-disk-demo:latest'
        - cloudInitNoCloud:
            userData: |-
              #cloud-config
              password: fedora
              chpasswd: { expire: False }
          name: cloudinitvolume
```

KubeVirt VirtualMachine

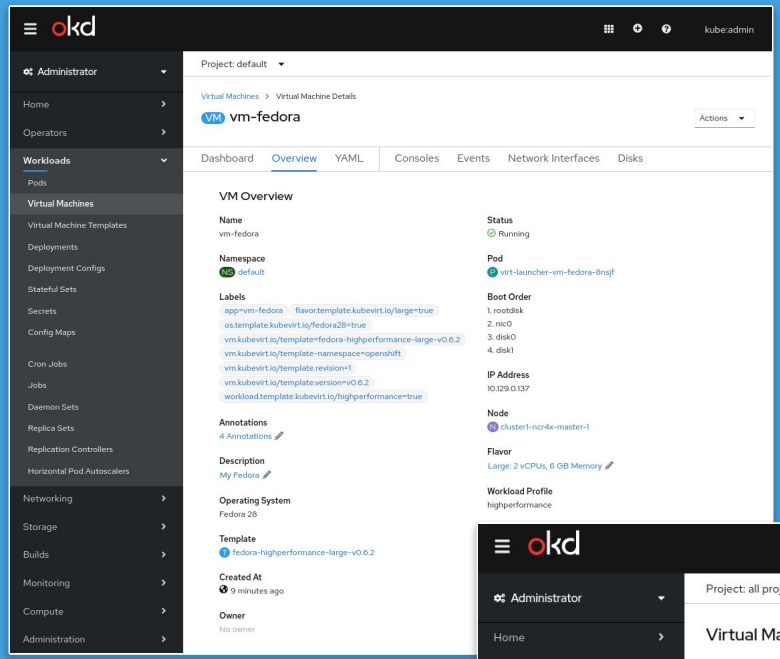
- Declarative over Imperative
  - Like any other k8s resource
- Fully integrated

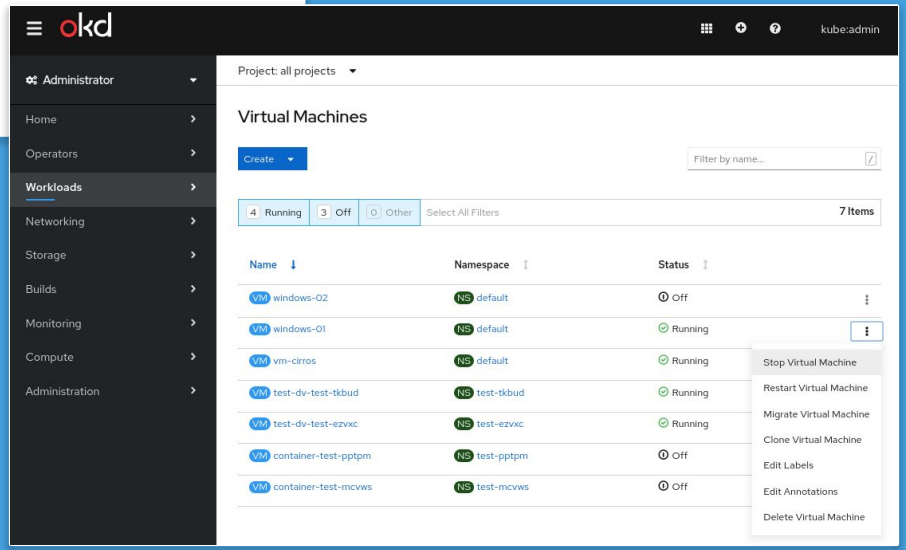
  kubectl [command] [type] [name] [flags]
- With shortcuts

  virtctl [command] [name]

```
$ kubectl apply -f my-vm.yaml
$ kubectl get vm my-vm
```

```
$ virtctl console my-vm
```



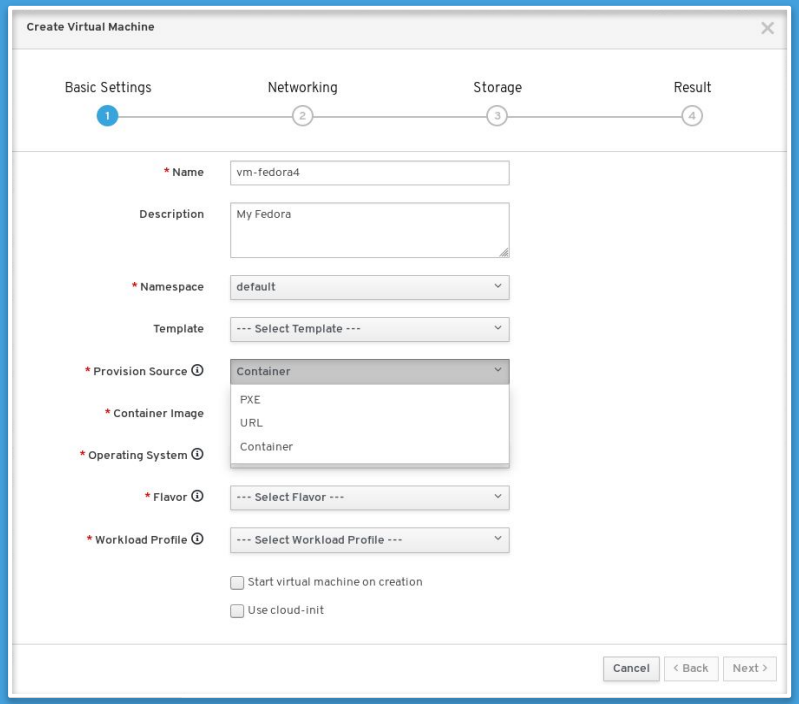


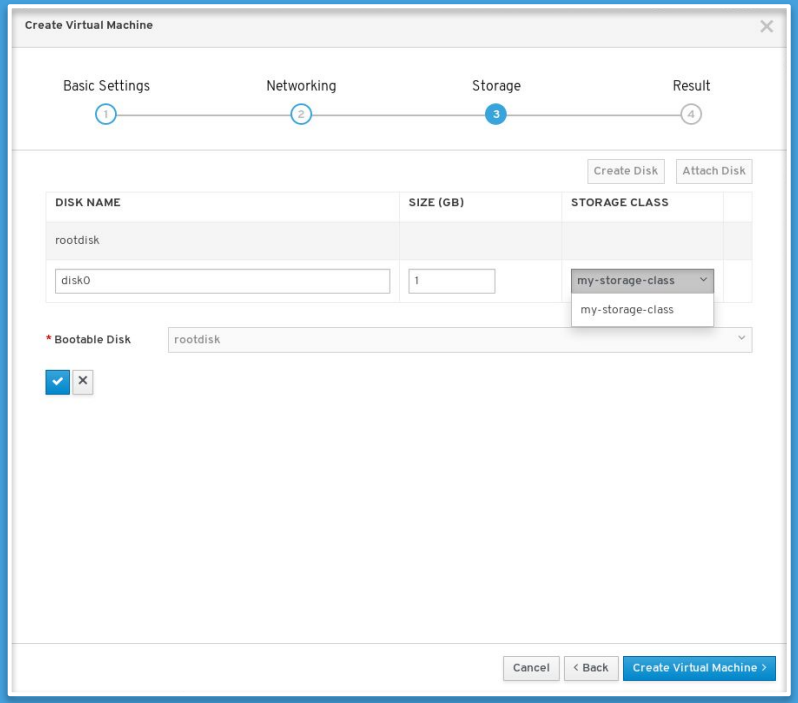
Web User Interface

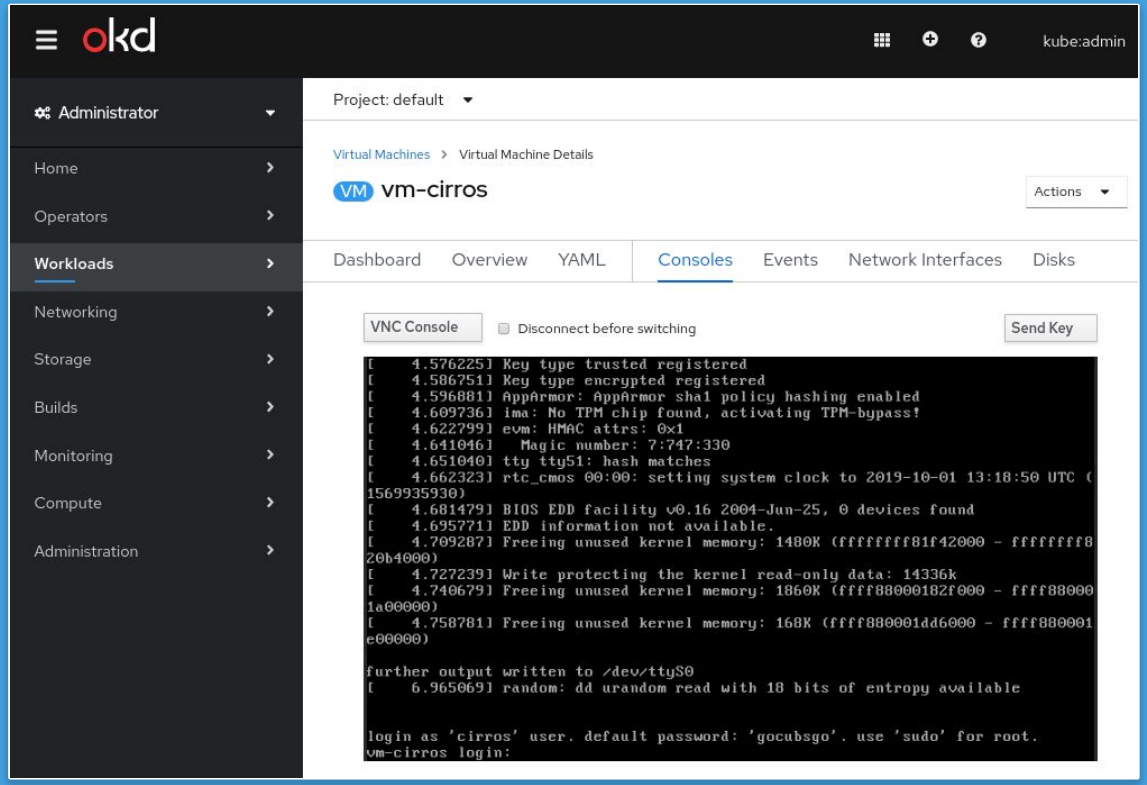
- Fully integrated with the OKD console
- Access
- Provision
- Management
- Import

Web User Interface

Create VM wizard

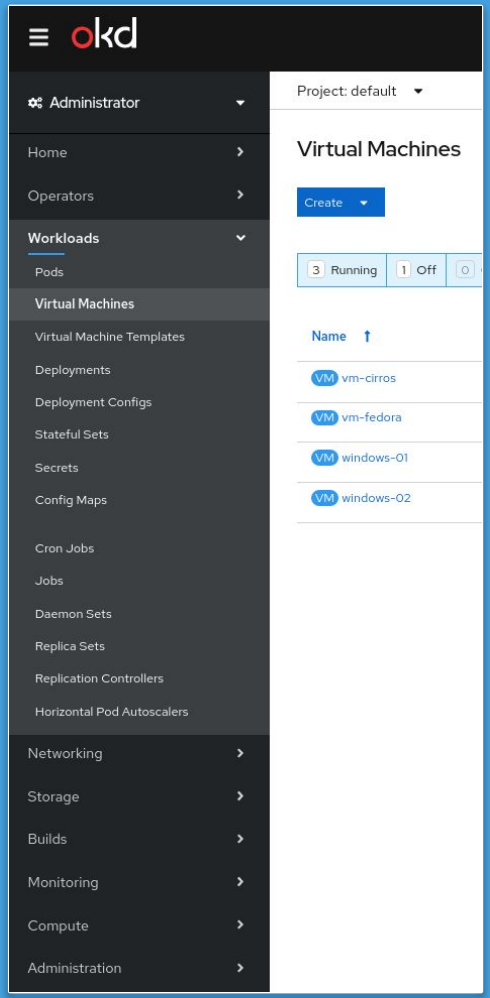






VM Consoles

- VNC
- Serial
- RDP
- In-browser vs. desktop



Related Objects

- Virtualization is first-class citizen along Containers
- Storage
- Networks
- Services
- Monitoring and Management
- Analyze issues - Status info, link through to details

Summary

- Containers vs. Virtual Machines
  - Coexistence over competition
  - Benefits of both
- Kubernetes for orchestration
- Virtual Machine in a Pod

References

- https://kubevirt.io
- https://github.com/openshift/console
- kubevirt-dev@googlegroups.com
- Marek Libra <marek.libra@gmail.com>