

KubeVirt

La virtualizzazione nell'era di Kubernetes

Stefano Stagnaro
CTO @ EXTRAORDY

Linux Day Milano
26/10/2024



Chi sono

Stefano Stagnaro

CTO @ EXTRAORDY

Red Hat Certified Instructor

Red Hat Certified Architect

Red Hat Accelerator

Ma, soprattutto:
appassionato di tecnologia
e open source



Red Hat Accelerators

MAKE YOUR MARK

Join a community of customers who are passionate tech users and practitioners

The Red Hat Accelerators program is a global enterprise customer community of Red Hat technology experts and enthusiasts who willingly share their knowledge and expertise with peers in the industry, their community, and with Red Hat.



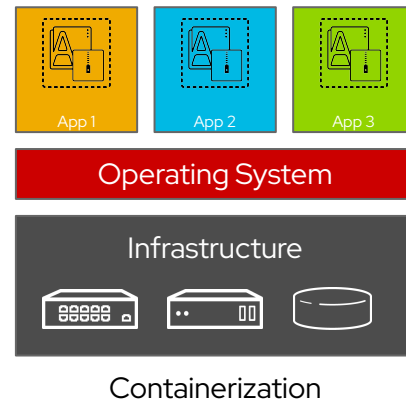
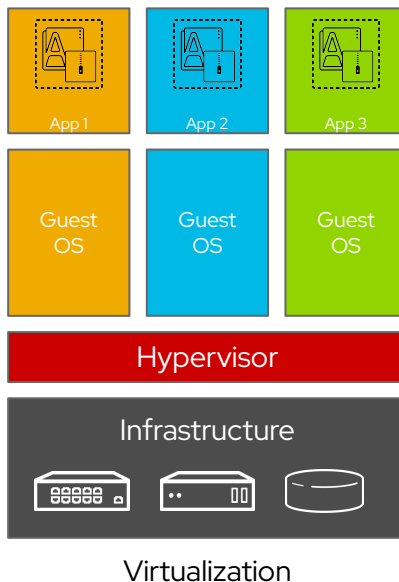


Cosa c'entrano le VM con Kubernetes?



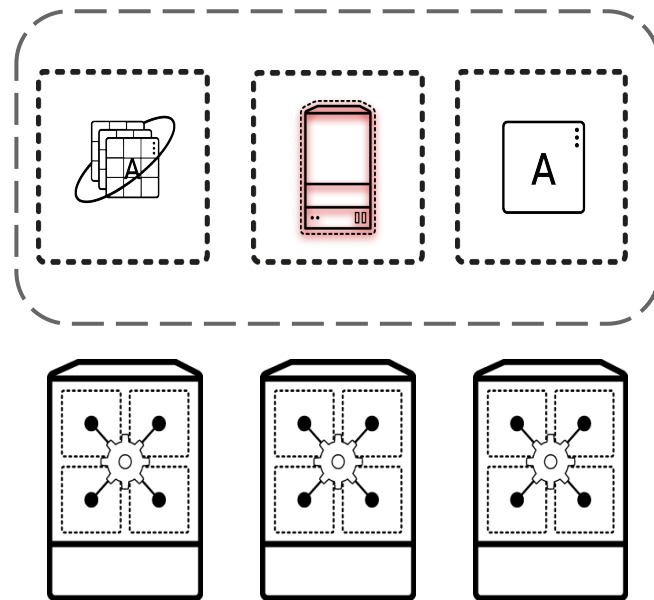
I container non sono VM

- I container sono solo un isolamento del processo
- I namespace del kernel forniscono isolamento e i cgroup forniscono controllo delle risorse
- Non è necessario alcun hypervisor per i container
- Contengono solo file binari, librerie e/o framework
- Sono effimeri



Le VM possono essere eseguite in un container

- Una virtual machine è solo un processo qemu-kvm
- I container incapsulano un processo
- Entrambi hanno bisogno delle medesima risorse:
 - Compute
 - Network
 - (a volte) Storage



Cos'è KubeVirt?

API e runtime di virtualizzazione Kubernetes per definire e gestire macchine virtuali:

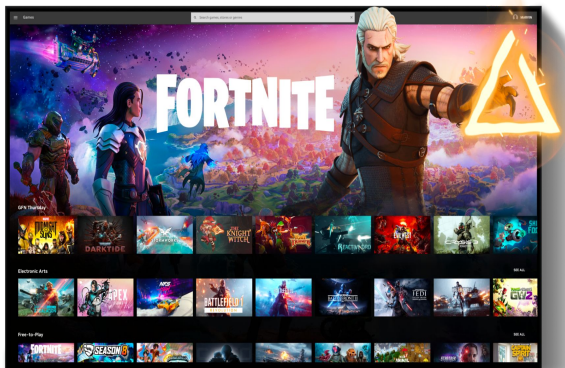
- Virtual machine
 - Eseguite in container, gestite nei Pod di Kubernetes
 - Utilizzano KVM e Qemu come strumenti hypervisor
- Scheduling, deployment, gestione operate da Kubernetes
- È integrato con tutte i principali servizi offerti da Kubernetes:
 - Utilizzano il networking SDN nativo
 - Utilizzano il persistent storage fatto con PVC, PV e StorageClass





Bello, ma lo usa qualcuno?

Sì, un portalino di gaming, ma niente di serio...



28 million
Users



1500+
Games



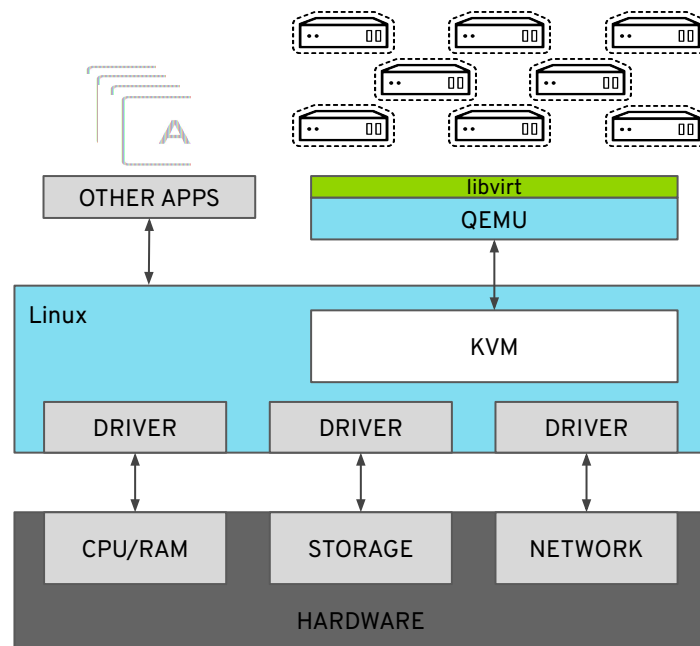
100+
Countries



30+
Data centers

KubeVirt usa lo stack KVM + libvirt + QEMU

- KubeVirt usa KVM, l'hypervisor del kernel Linux
- QEMU usa KVM per eseguire le virtual machine (con accelerazione CPU)
- libvirt è un'interfaccia API
- Attualmente è supportata l'architettura x86 ma altre sono in roadmap



Virtualizzazione nativa Kubernetes

- Gli operator sono un'estensione delle API di Kubernetes al fine di aggiungere nuove funzionalità
- Per la virtualizzazione vengono create nuove CustomResourceDefinitions (CRDs), ad esempio:
 - VirtualMachine
 - VirtualMachineInstance
 - VirtualMachineInstanceMigration
 - VirtualMachineSnapshot
 - DataVolume (tramite CDI)

```
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
```



Ma in pratica,
come si crea una VM in KubeVirt?



```
$ kubectl apply -f - <<EOF
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: testvm
spec:
  running: false
  template:
    metadata:
      labels:
        kubevirt.io/size: small
        kubevirt.io/domain: testvm
    spec:
      domain:
        devices:
          disks:
            - name: containerdisk
              disk:
                bus: virtio
            - name: cloudinitdisk
              disk:
                bus: virtio
          interfaces:
            - name: default
              masquerade: {}
          resources:
            requests:
              memory: 64M
          networks:
            - name: default
              pod: {}
          volumes:
            - name: containerdisk
              containerDisk:
                image: quay.io/kubevirt/cirros-container-disk-demo
            - name: cloudinitdisk
              cloudInitNoCloud:
                userDataBase64: SGkuXG4=
EOF
```

EOF

```
$ ./virtctl start testvm
VM testvm was scheduled to start
```

```
$ kubectl get vms
```

NAME	AGE	STATUS	READY
testvm	12m	Running	True

```
$ kubectl get vmis
```

NAME	AGE	PHASE	IP	NODENAME	READY
testvm	9s	Running	10.42.0.17	ubuntu	True

```
$ ./virtctl console testvm
Successfully connected to testvm console.
The escape sequence is ^]
...
=== datasource: nocloud local ===
instance-id: 5a9fc181-957e-5c32-9e5a-2de5e9673531
name: N/A
availability-zone: N/A
local-hostname: testvm
launch-index: N/A
=== cirros: current=0.4.0 latest=0.6.2 uptime=16.29 ===
```

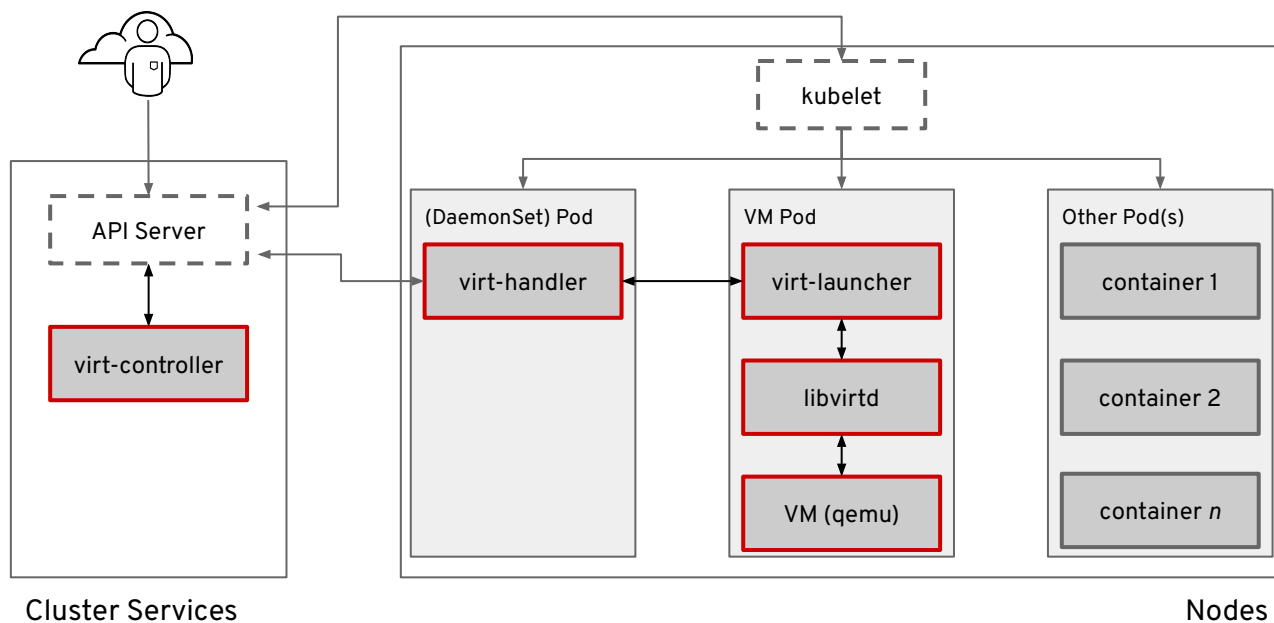

http://cirros-cloud.net



```
login as 'cirros' user. default password: 'gocubsgo'.
use 'sudo' for root.
testvm login:
```

Provalo qui



Architectural Overview





Ma è sufficiente la sola
componente KubeVirt?

Non solo KubeVirt...

- Se vogliamo ottenere una piattaforma di virtualizzazione completa, la risposta è **no**
- Cosa manca? (riportiamo alcune tecnologie di esempio, ma ci sono alternative)
 - Gestione dello storage specifica per le VM (import di cloud image, utilizzo dello storage locale)
 - risolvi con → **Containerized Data Importer** operator e **HPP** operator
 - Componenti avanzate di networking (Multus, Open vSwitch, NMstate)
 - risolvi con → **Cluster Network Addons** operator ed **NMstate** operator
 - Una distribuzione Linux immutabile con supporto a Ignition/Butane
 - risolvi con → CoreOS, Flatcar, MicroOS
 - Installazione/gestione degli host baremetal
 - risolvi con → **Metal³** e **Cluster API**
 - Load balancing del traffico non-HTTP/TLS
 - risolvi con → **MetalLB**
 - Node remediation (reboot, fencing, reprovisioning) →
 - risolvi con → **Node Health Check** operator del progetto medik8s.io
 - Node maintenance
 - risolvi con → **Node Maintenance** operator del progetto medik8s.io
 - Il Backup!! (perché ti stavi dimenticando il backup, vero?)
 - risolvi con → **KubeVirt Velero plugin**

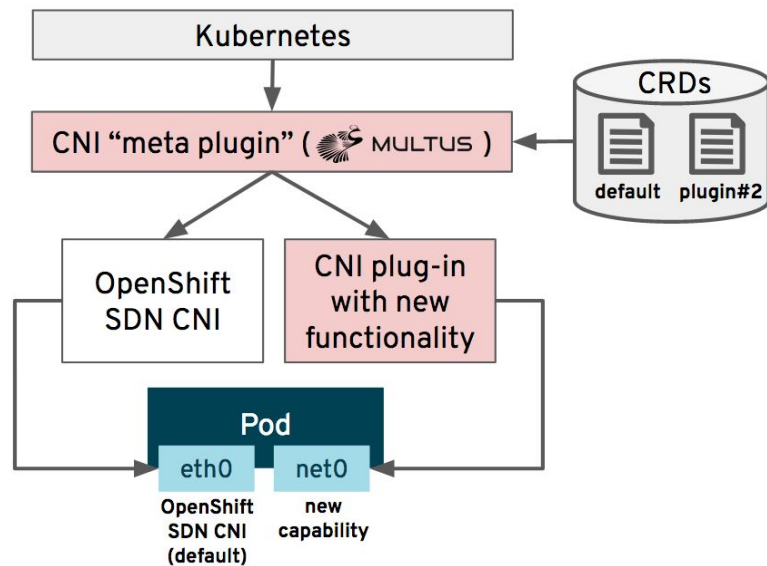


Breve digressione sul networking



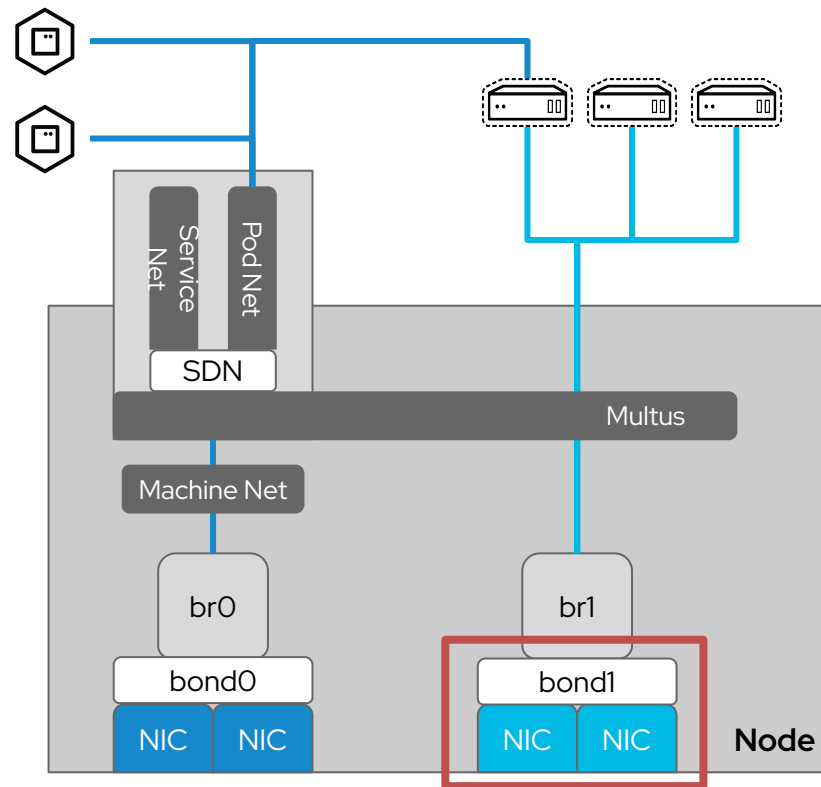
Networking 1/5

- Le VM sono connesse alla pod network di default, gestita dal plugin CNI
 - OVNKubernetes, Calico
- Espongono il traffico come per un qualsiasi pod:
 - traffico HTTP/TLS con Ingress
 - traffico L3 generico con un Service
 - LoadBalancer
 - NodePort
 - ClusterIP (con externalIPs)
- Per le interfacce aggiuntive si usa Multus:
 - Bridge, SR-IOV, OVN secondary networks
 - Se voglio aggiungere ulteriori VLAN devo prima configurare le nic con NMstate
- Configuro eventuali bonding con NMstate



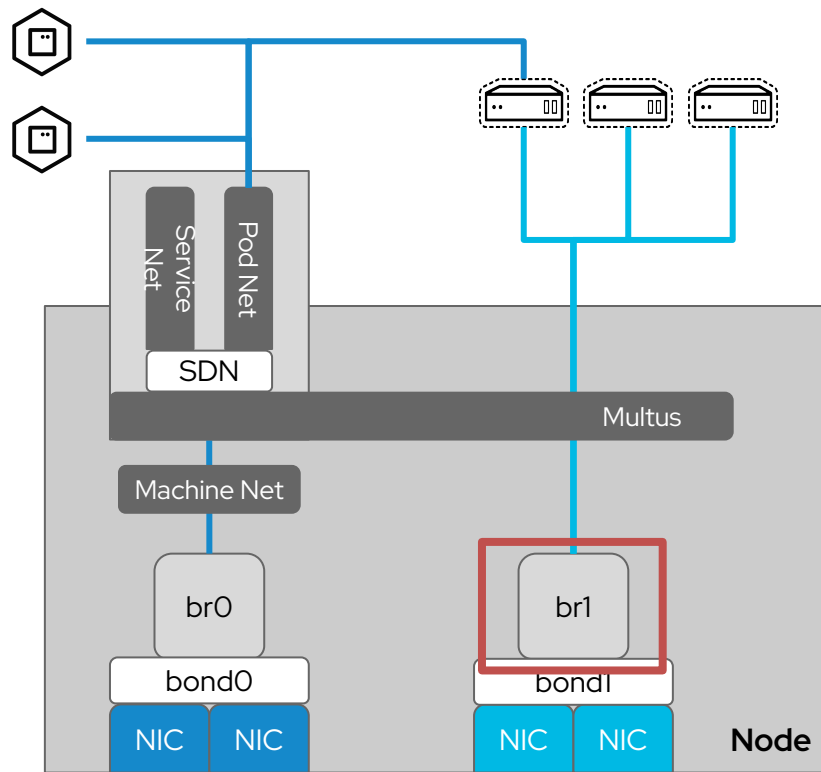
Networking 2/5

```
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
```



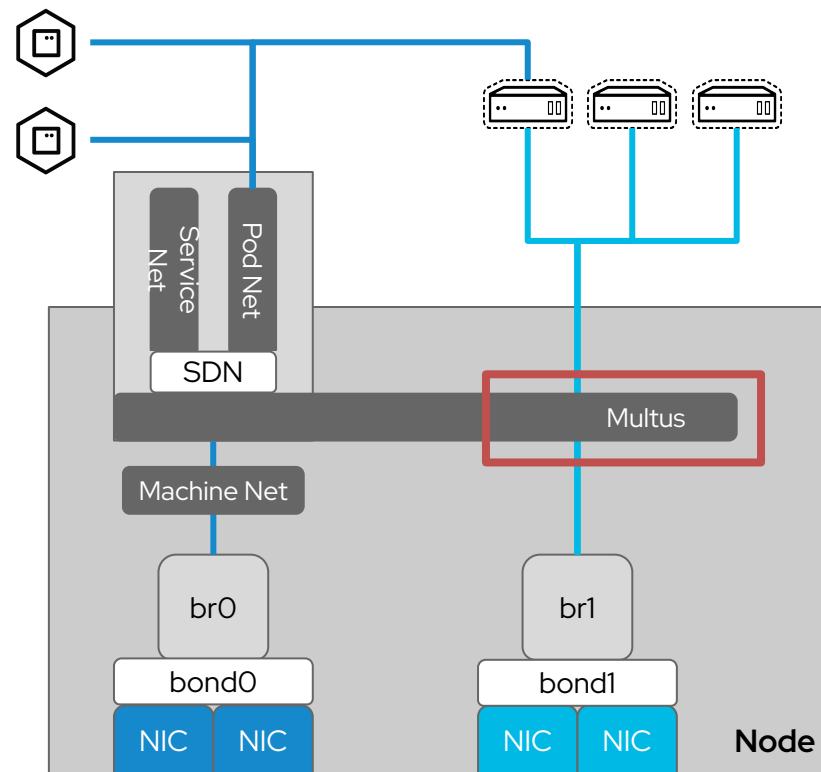
Networking 3/5

```
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
```



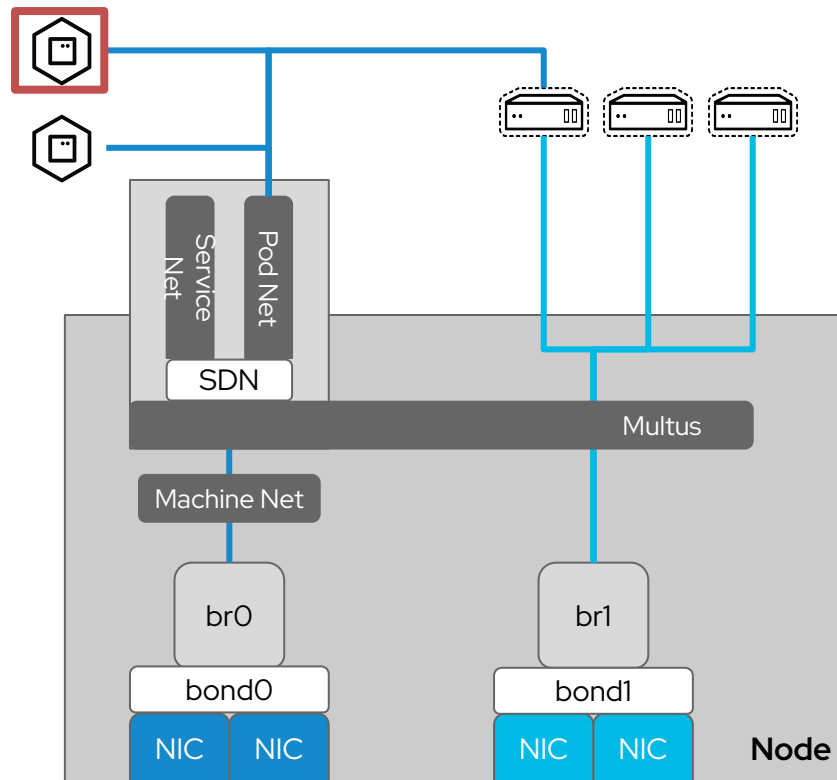
Networking 4/5

```
1  apiVersion: k8s.cni.cncf.io/v1
2  kind: NetworkAttachmentDefinition
3  metadata:
4    annotations:
5      - k8s.v1.cni.cncf.io/resourceName: bridge.network.kubevirt.io/br1
6    name: vlan-93
7    namespace: default
8  spec:
9    config: >-
10     - {
11       - "name": "vlan-93"
12       - "type": "cnv-bridge",
13       - "cniVersion": "0.3.1",
14       - "bridge": "br1",
15       - "vlan": 93,
16       - "macspoofchk": true,
17       - "ipam": {},
18       - "preserveDefaultVlan": false
19     }
20
```



Networking 5/5

```
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
```





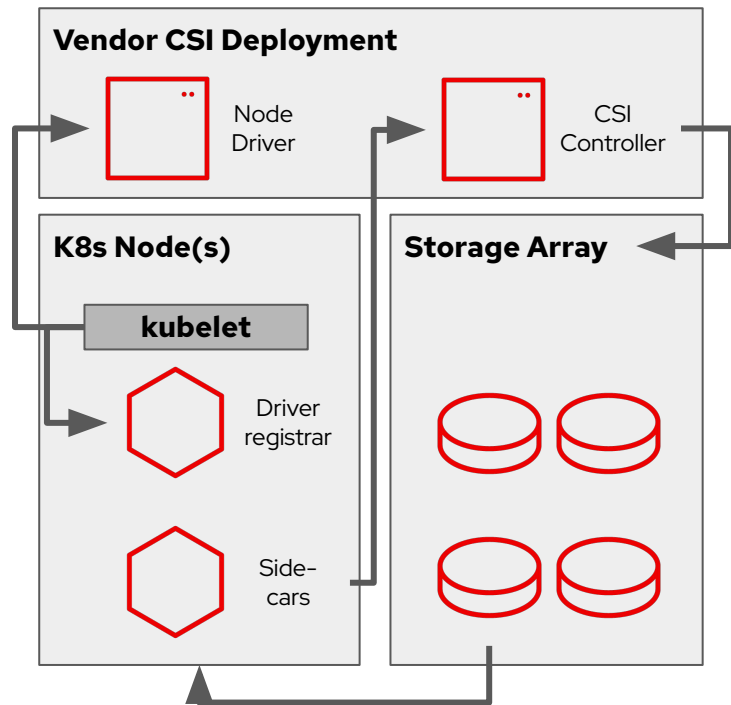
Breve digressione sullo storage

Virtual Machine Storage

- KubeVirt usa il paradigma dei PersistentVolume (PV) di Kubernetes
- I PV come backend possono avere:
 - qualsiasi CSI driver
 - Storage locale con host path provisioner / logical volume Operator
- Vanno bene PV sia statici che dinamici
- È richiesto un PV in **RWX** per la Live Migration
- I dischi sono collegati alle VM con controller VirtIO o SCSI
- Nella definizione della VM posso specificare il boot order

Come ottengo dischi di VM dai PV?

- VM disks on FileSystem mode PVCs are created as thin provisioned raw images by default, but can be configured
- Block mode PVCs are attached directly to the VM
 - Many partners support RWX with block mode PVCs
 - Less overhead, preferred.
- CSI offloads operations to the storage device
 - Use DataVolumes to *clone* VM disks to automatically select the optimal method to clone the disk
 - Use VM details interface for (powered off) VM snaps
- Resize VM disks using standard PVC methods
- Hot-plugging virtual disks is supported



DataVolumes per gestire le sorgenti disco

- 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
- VM disks are cloned / copied from existing PVCs
- DataVolumes are created as distinct objects or as a part of the VM definition as a `dataVolumeTemplate`
- DataVolumes use the `ContainerizedDataImporter` to connect, download, and prepare the disk image
- DataVolumes create PVCs based on defaults defined in the profile
 - Access mode, snapshot method

```
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'
```

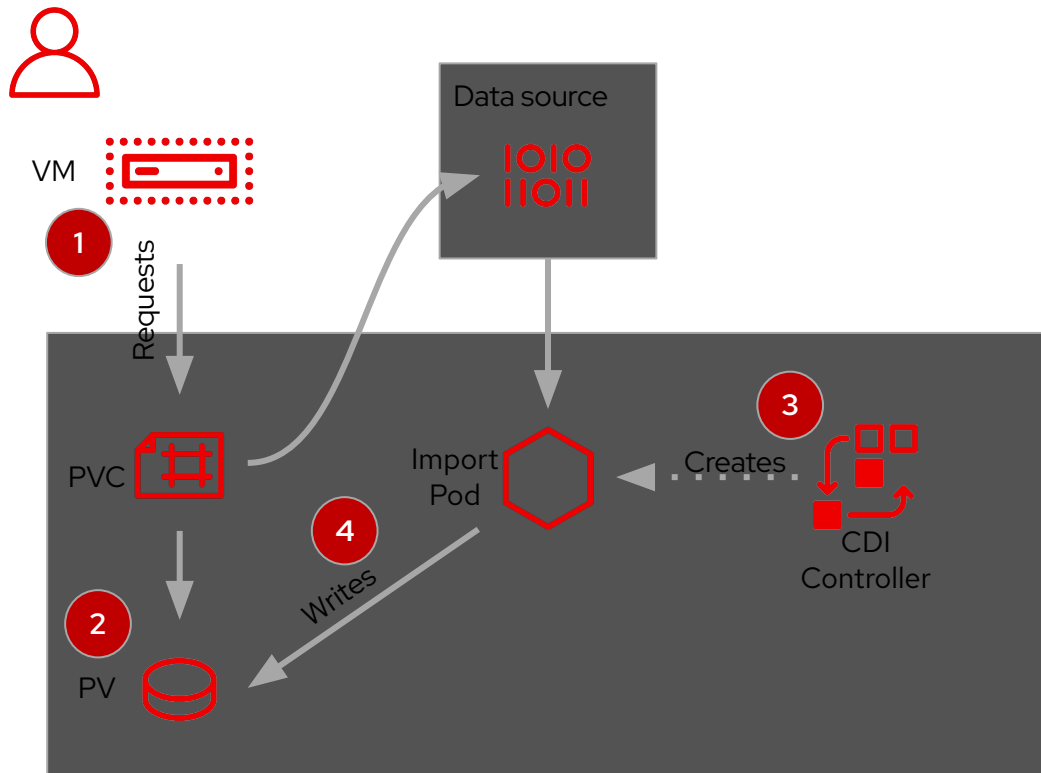
Storage Profiles per gestire i profili disco

- Provide default settings and properties for StorageClasses used by DataVolumes
- Created automatically for every StorageClass
- Preconfigured values for many storage providers, administrator can modify and customize
- DataVolume definitions only need to specify StorageClass, without knowledge of underlying details
 - spec.storage doesn't require fields other than the size and StorageClass

```
1  apiVersion: cdi.kubevirt.io/v1beta1
2  kind: StorageProfile
3  metadata:
4    name: hostpath-provisioner
5  spec:
6    claimPropertySets:
7      - accessModes:
8        - ReadWriteOnce
9        volumeMode:
10         Filesystem
```

```
1  apiVersion: cdi.kubevirt.io/v1alpha1
2  kind: DataVolume
3  metadata:
4    name: rhel8-vm-disk
5  spec:
6    storage:
7      resources:
8        requests:
9          storage: 30Gi
10     storageClassName: hostpath-provisioner
11    source:
12      pvc:
13        name: rhel8-cloud-image
14        namespace: os-images
```

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



Non sarebbe meglio una piattaforma
completa di virtualizzazione?

Esistono soluzioni *turn-key* basate su KubeVirt?

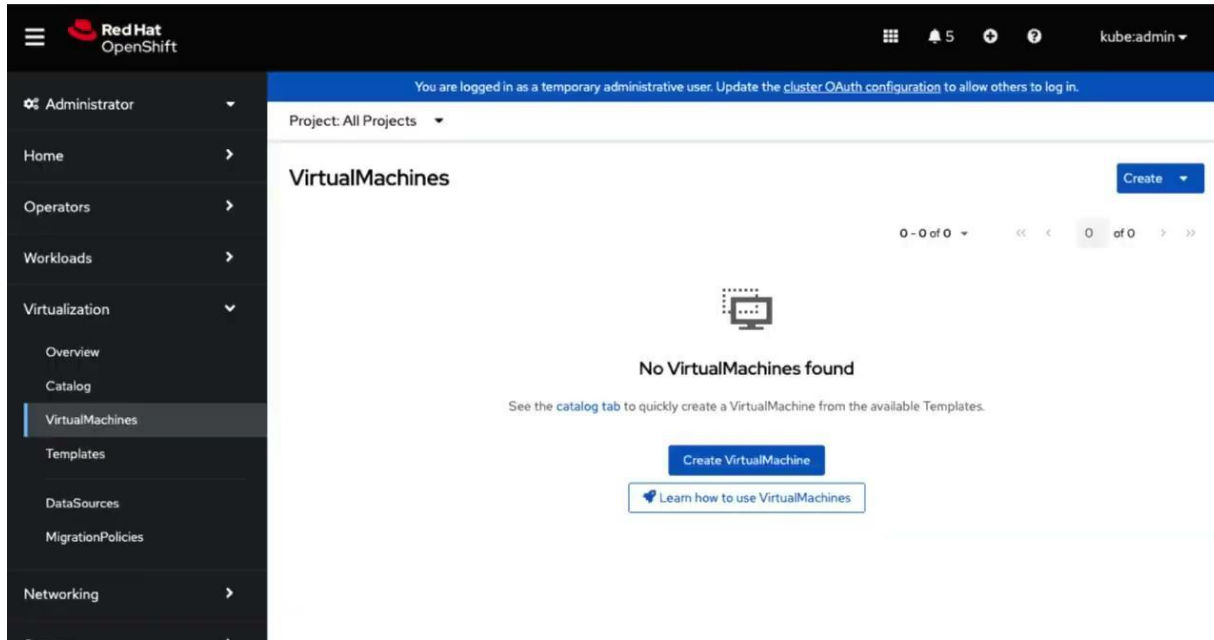


Le soluzioni *chiavi in mano*...

- Sì, esistono e sono l'unico modo di utilizzare questa tecnologia in ambiti di produzione aziendale
- OpenShift/OKD Virtualization, Deckhouse, Harvester, Kubermatic
- Alcune di queste sono soluzioni HCI che prevedono anche la componente SDS
 - e.g. Rook, Longhorn
 - Interessante ma non per tutti




Immaginate di proporre la virtualizzazione KubeVirt con questa interfaccia...




Provalo qui



<https://red.ht/3NC5wHs>



Quali skill devo avere prima di poter utilizzare
KubeVirt in un ambiente di produzione?



Le skill...

In ordine di specificità:

- Skill generiche di IT:
 - Linux
 - Networking TCP/IP
 - Storage
 - Virtualizzazione
- Skill specifiche Linux (oltre ai rudimenti):
 - Storage (creazione e gestione di filesystem, gestione di storage a blocchi, storage di rete)
 - Networking (buona conoscenza dello stack TCP/IP, regole di masquerading/virtual ip)
 - Security (SELinux, Kernel namespace, regole di ip filtering)
 - Container!!! (dovete saper maneggiare un container stand-alone)
- Skill specifiche Kubernetes (oltre ai rudimenti):
 - Networking (CNI, Multus, NMstate)
 - Storage (CSI, provisioner, StorageClass)
 - Security (RBAC, NetworkPolicy, SecurityContext, sorgenti di autenticazione)
 - Gestione dei pod (DisruptionBudget, ResourceQuota, Limit/Request, strumenti di osservabilità)
 - Gestione dei nodi (Ignition)



Esiste un training specifico per KubeVirt?



Formazione su KubeVirt

- Formazione sul progetto comunitaria KubeVirt → **no**
- Esistono i percorsi di formazione dei vendor, specifici per il prodotto
 - Ad esempio il corso **DO316** per Red Hat OpenShift Virtualization (oppure **eHRV101** per SUSE Harvester)

DO316



- **Introduction to OpenShift Virtualization**
Describe the features and use cases of OpenShift Virtualization.
- **Run and access Virtual Machines**
Create, manage, inspect, and monitor virtual machines in Red Hat OpenShift Virtualization.
- **Configure Kubernetes network for Virtual Machines**
Configure standard Kubernetes network objects and external access for VMs and virtual machine-backed applications.
- **Connect Virtual Machines to external networks**
Configure node networking to connect virtual machines and nodes to networks outside the cluster.
- **Configure Kubernetes storage for Virtual Machines**
Manage storage and disks for VMs in Red Hat OpenShift.
- **Virtual Machine template management**
Create and manage templates to provision virtual machines.
- **Advanced Virtual Machine management**
Import, export, snapshot, clone, and live migrate a virtual machine and initiate node maintenance.
- **Configure Kubernetes high availability for Virtual Machines**
Configure Kubernetes resources to implement high availability for virtual machines.



Domande?

Contattami



Grazie!



<https://www.linkedin.com/in/stefanostagnaro/>

