

RUNNING VIRTUAL MACHINES ON KUBERNETES


Roman Mohr & Fabian Deutsch, Red Hat, KVM Forum, 2017

FABIAN DEUTSCH

Fedora user and former
package maintainer
oVirt and KubeVirt Contributor
Working at Red Hat

fabiaand@redhat.com

 [@dummdida](https://twitter.com/dummdida)

 [fabiaand](https://github.com/fabiaand)

<https://dummdida.tumblr.com>

ROMAN MOHR

oVirt and KubeVirt Contributor
Working at Red Hat

rmohr@redhat.com

 [@rfenkhuber](https://twitter.com/rfenkhuber)

 [rmohr](https://github.com/rmohr)



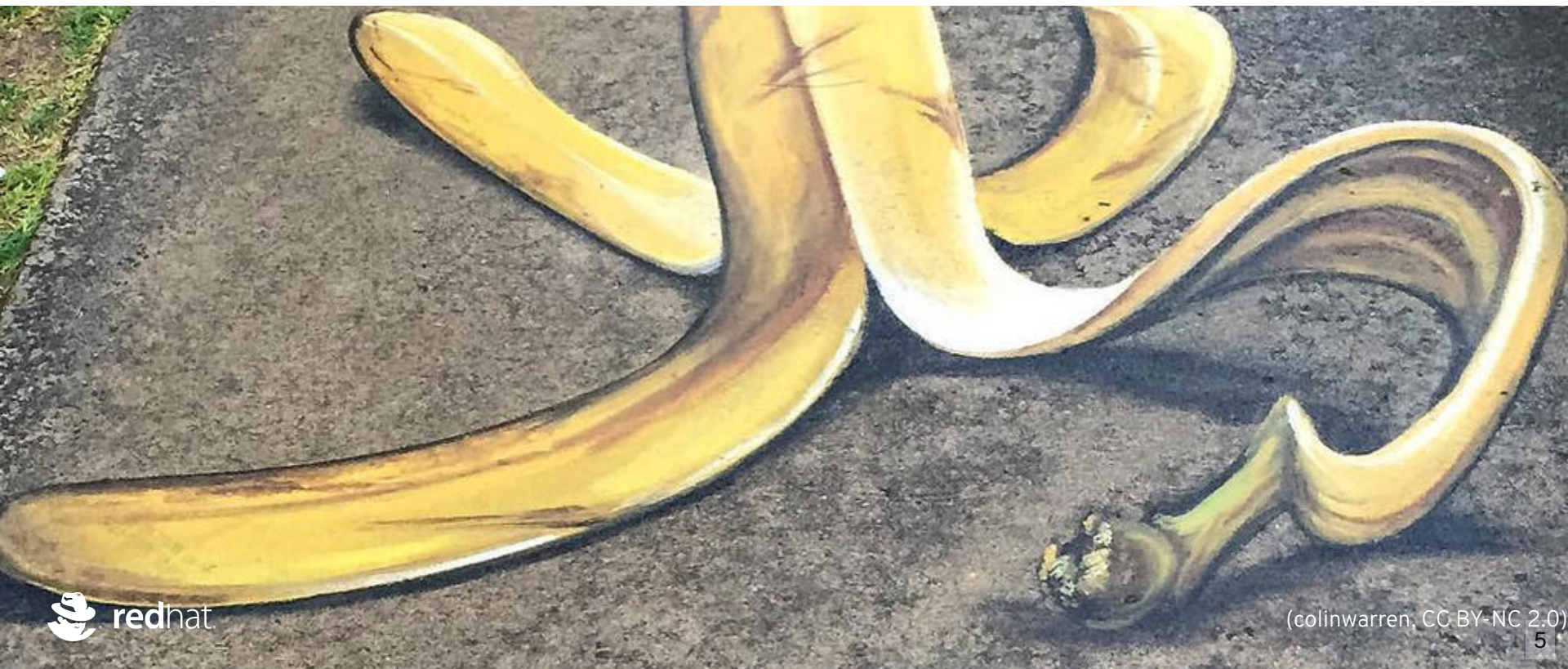
**VIRTUALIZATION IS
OMNIPRESENT. TODAY.**



CONTAINERS AS WELL.

CONTAINERS LOOK, TASTE, AND SMELL THE SAME - JUST BETTER

"Versatile, scalable, hyped, community driven, devops, ..."
Take this with a grain of salt.



"HOW DO WE GET THERE?"

**"HOW CAN I REPLACE MY VMS
WITH CONTAINERS? HOW!?"**

ARE THEY *REALLY* SUBSTITUTES? IS THE ONE LIKE THE OTHER?

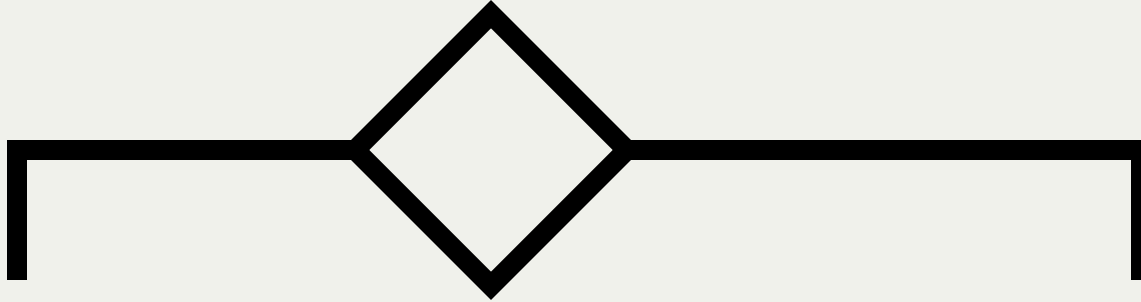
Technology? Features? Feeling? Tools?
Requirements?

YES

→ IT DEP

NO

Replace?



Yes



COOL

No



NOT YET?

"NEVER"?

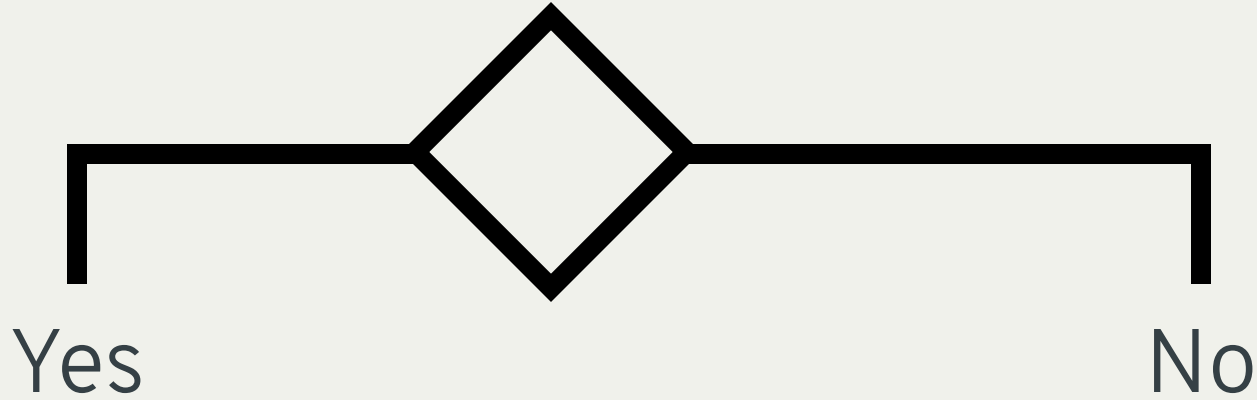
MIGRATION

If workloads can be moved to containers, then it's a migration

CONVERGENCE

If not, then we still want convergence

Replace?

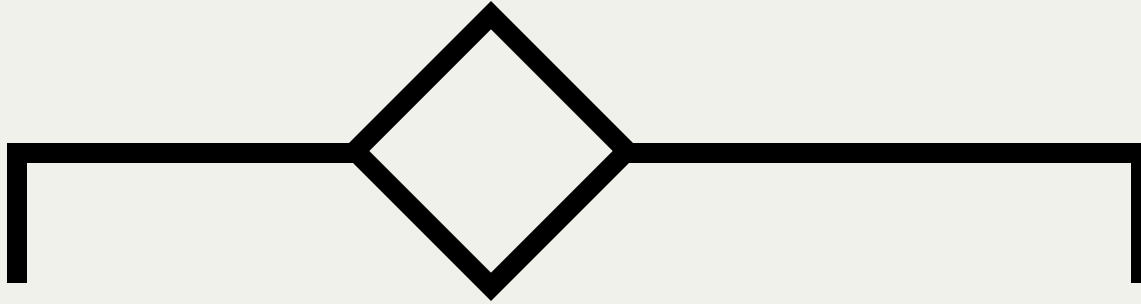


XX%

YY%

BOTH.

Replace?



Yes



MIGRATION PATH?

No



DOUBLED

INFRASTRUCTURE?

Virtual Machines

Management Plane

Storage

Network

...

Containers

Management Plane

Storage

Network

...

2X INFRASTRUCTURE?



Virtualization and containers



KUBEVIRT

CONTAINERS & VIRTUAL MACHINES

on the same infrastructure.

Virtual Machines

Containers

Management Plane

Storage

Network

...

KEEP YOUR VMS ...

Virtual Machines

Containers

Management Plane

Storage

Network

...

... TRANSITION WHAT YOU NEED ...

Virtual Machines

Containers

Management Plane

Storage

Network

...

... AND STICK TO VMS AS NEEDED.

WOOT?

Tell me more.



Virtual Machines



Containers

Kubernetes

Storage

Network

...



Virtual Machines

Containers

+ KubeVirt

Kubernetes

Storage

Network

...

HOW DOES KUBEVIRT INTEGRATE WITH KUBERNETES?

The Details.

KUBERNETES API

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

*“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.” **

HOW ABOUT TREATING A POD AS A VM?

- Add device details as annotations.
- Modify the container runtime on every node.
- Deal with the fact that there are two Pods when you do migrations.
- Implement as much functionality as possible from the Kubelet, since there is not way to distinguish from outside what your VM Pod supports, compared to a normal Pod.
- Are we talking about a VM Pod or a Pod?

HOW ABOUT ADDING AN EXPLICIT VIRT API?

- Allows a proper Virtual Machine Specification
- We can ship KubeVirt as a pure add-on. No Node modifications are necessary.
- No matter, how much Pods are necessary to perform a migration, we have one single entrypoint to the Virtual Machine.
- Reuse all of the kubelet and Pod Spec functionality, by running a Virtual Machine inside the Pod
- Talk about VMs when they are VMs, talk about Pods when they are Pods.

KUBEVIRT API

```
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.

TYPICAL KUBECTL FEELING

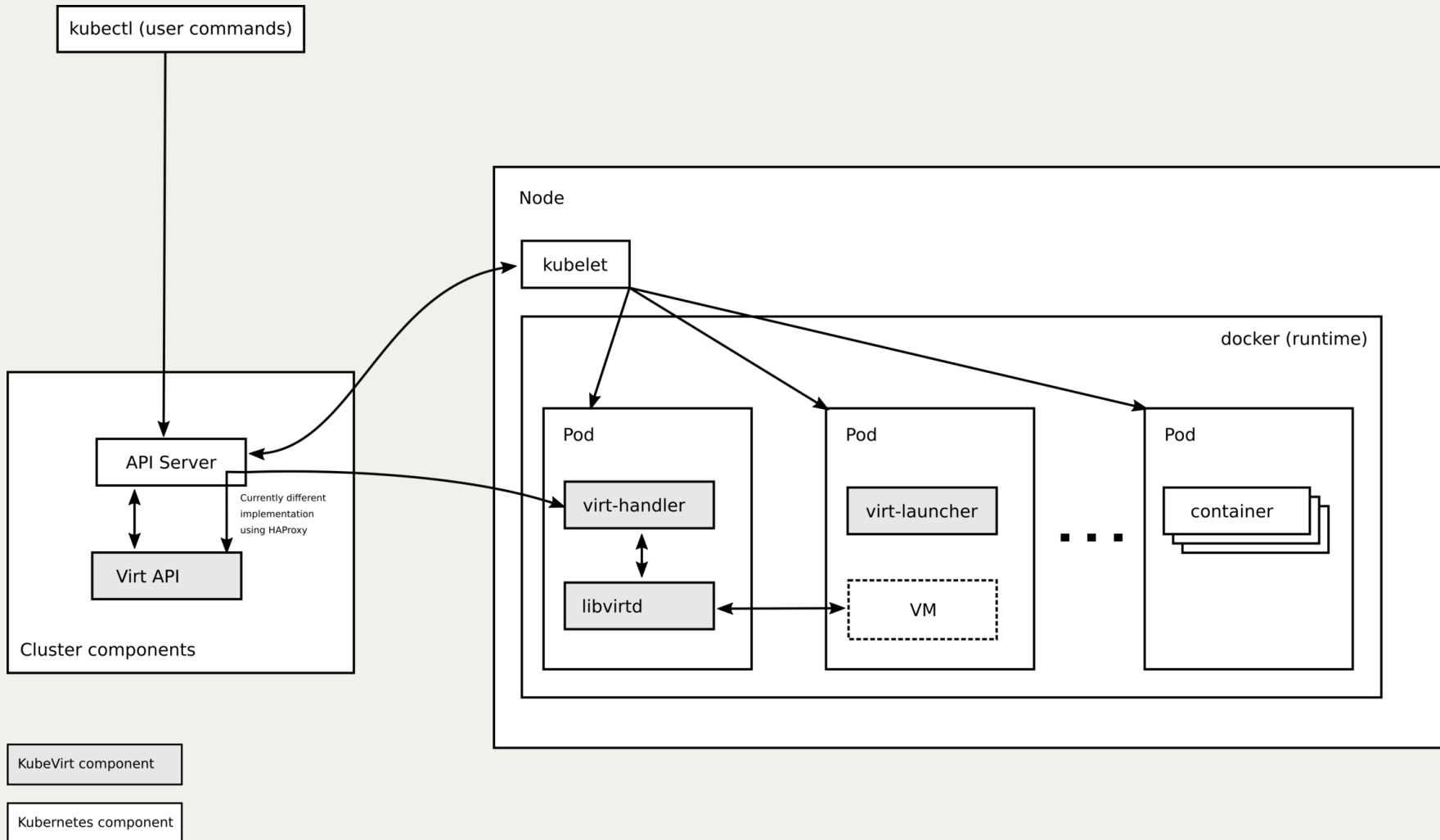
```
kind: VirtualMachine
metadata:
  name: testvm
spec:
  domain:
    devices:
      graphics:
        - type: spice
      consoles:
        - type: pty
```

Typical Pod commands:

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

Typical VirtualMachine commands:

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



[Documentation](#)

MIGRATIONS

```
kind: Migration
metadata:
  generateName: my-migration
spec:
  nodeSelector:
    kubevirt.io/hostname: node1
  selector:
    name: testvm
status:
  phase: Succeeded
```

Backed by a controller:

- On object create, schedules a new Pod
- On successful Pod start, it triggers the migration
- At the end of the migration the object is moved to a final state
- Always **one** VirtualMachine object you reference

The objects **Migration** with **VirtualMachine** provide a consistent entry point to anything VirtualMachine related, like the Pod does for Kubernetes.

API CHALLENGES

- Feature wise comparable to domxml
- Certain features are node specific - they need to be abstracted
- Needs to be married with Kubernetes concepts (pv, networks)
- Needs additional data for cluster-only features like scheduling

INTEGRATION CHALLENGES

Properly integrate the VirtualMachine lifecycle in a Pod lifecycle.

- Disks
- Networking
- qemu with libvirt in a Pod
- cgroups and Namespaces
- Migrations on top of Kubernetes

ADDITIONAL FEATURES

- VirtualMachineReplicaSet
- Cloud Provider
- Nested Kubernetes Nodes
- Cloud Init
- Console/Spice access
- More to come ...

PILLARS AND EFFECTS.

libvirt, ... everything in pods → Native Kubernetes add-on

New resource type for VMs → API server with VM functionality

Operator pattern to manage VMs → Declarative, like everything else

VMs live inside pods → Kubernetes' infrastructure is leveraged

TRY (WITH MINIKUBE)

```
$ minikube start --vm-driver kvm --network-plugin cni
```

```
$ git clone https://github.com/kubevirt/demo.git
```

```
$ cd demo
```

```
$ bash run-demo.sh
```

```

$ bash run-mini-demo.sh
# Deploying KubeVirt
...
vm "testvm" created
Waiting for KubeVirt to be ready ...
Waiting for KubeVirt to be ready ...
Waiting for KubeVirt to be ready ...
# KubeVirt is now ready. Try:
# $ kubectl get vms

$ kubectl get vms
NAME          KIND
testvm       VM.v1alpha1.kubevirt.io

$ kubectl get pods
NAME                                READY    STATUS    RESTARTS   AGE
haproxy-723816479-wcblm             1/1     Running   1           49s
iscsi-demo-target-tgtd-1270025779-nckbh 1/1     Running   0           48s
libvirt-8zjlk                        2/2     Running   0           48s
spice-proxy-3525077118-fswn9        1/1     Running   0           47s
virt-api-1956313626-t9rhj           1/1     Running   0           46s
virt-controller-2251532855-tfm9f     1/1     Running   0           45s
virt-handler-s7g76                  1/1     Running   0           43s
virt-launcher-testvm-----q05vh     1/1     Running   0           38s
virt-manifest-1665692876-cs8wp       2/2     Running   0           42s

$ kubectl exec -it libvirt-8zjlk bash
Defaulting container name to libvirtd.
Use 'kubectl describe pod/libvirt-8zjlk' to see all of the containers in this pod.
# virsh list
  Id    Name
-----
  1     default_testvm
                                State
                                -----
                                running

# exit

```

Learn and contribute at <http://kubevirt.io>

Thank you.

 @kubevirt

 kubevirt