KubeVirt on Talos: A Homelab Journey

+ = ❤️

Michael Trip
18-09-2024
Taloscon, London

AT COMPUTING
ATYPICAL OPEN SOURCE GURUS

Run KubeVirt, they said…

It would be fun, they said…

It took me 7 months…. 😂

Who am i?

Name: Michael Trip
From: Apeldoorn / The Netherlands

- Open Source consultant @ AT Computing
- Kubernetes trainer
- Current consultancy gig: Dutch Tax Administration
- Linux Geek (started in 2004)
- Redhat certs: RHCSA, RHCE and Openshift
- Kubernetes certs: CKAD, CKA and CKS
- Hypervisor experience:
  - VMWare GSX (back in 2008)
  - VMWare ESXi (since 3.5)
  - HyperV
  - Xen
  - Proxmox
  - KubeVirt (obviously)

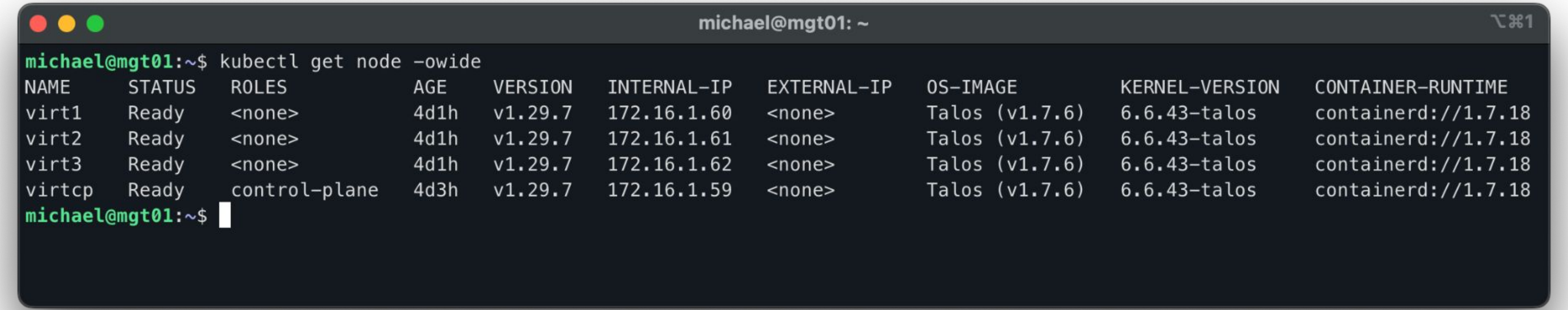
And……

Topics of today

- My homelab
- What makes a good hypervisor?
- What is KubeVirt ?
- KubeVirt installation on Talos
- CDI: Containerized Data Importer
- What about…..
    - Vlans and networking?
    - Hyperconverged and shared storage?
    - Live migration?
- Demo time:
    - Creating VMs
    - Live migration
- Takeaways and conclusion

Homelab

- 1 virtual control plane node running Talos 1.7.6 on Proxmox
- 1 Zimaboard running NFS With Debian 12
- 3 worker nodes running on bare metal with Talos 1.7.6
  - 2 HP Elitedesks with 16GB ram, 256GB SSD, single disk
  - 1 HP Prodesk with 16 GB ram, 256GB SSD, single disk
- Endgoal for my project: Have a enterprise-like virtualization cluster



```
michael@mgt01:~$ kubectl get node -owide
NAME     STATUS   ROLES           AGE    VERSION   INTERNAL-IP    EXTERNAL-IP   OS-IMAGE         KERNEL-VERSION   CONTAINER-RUNTIME
virt1    Ready    <none>          4d1h   v1.29.7   172.16.1.60    <none>        Talos (v1.7.6)   6.6.43-talos     containerd://1.7.18
virt2    Ready    <none>          4d1h   v1.29.7   172.16.1.61    <none>        Talos (v1.7.6)   6.6.43-talos     containerd://1.7.18
virt3    Ready    <none>          4d1h   v1.29.7   172.16.1.62    <none>        Talos (v1.7.6)   6.6.43-talos     containerd://1.7.18
virtcp   Ready    control-plane   4d3h   v1.29.7   172.16.1.59    <none>        Talos (v1.7.6)   6.6.43-talos     containerd://1.7.18
michael@mgt01:~$
```

Homelab

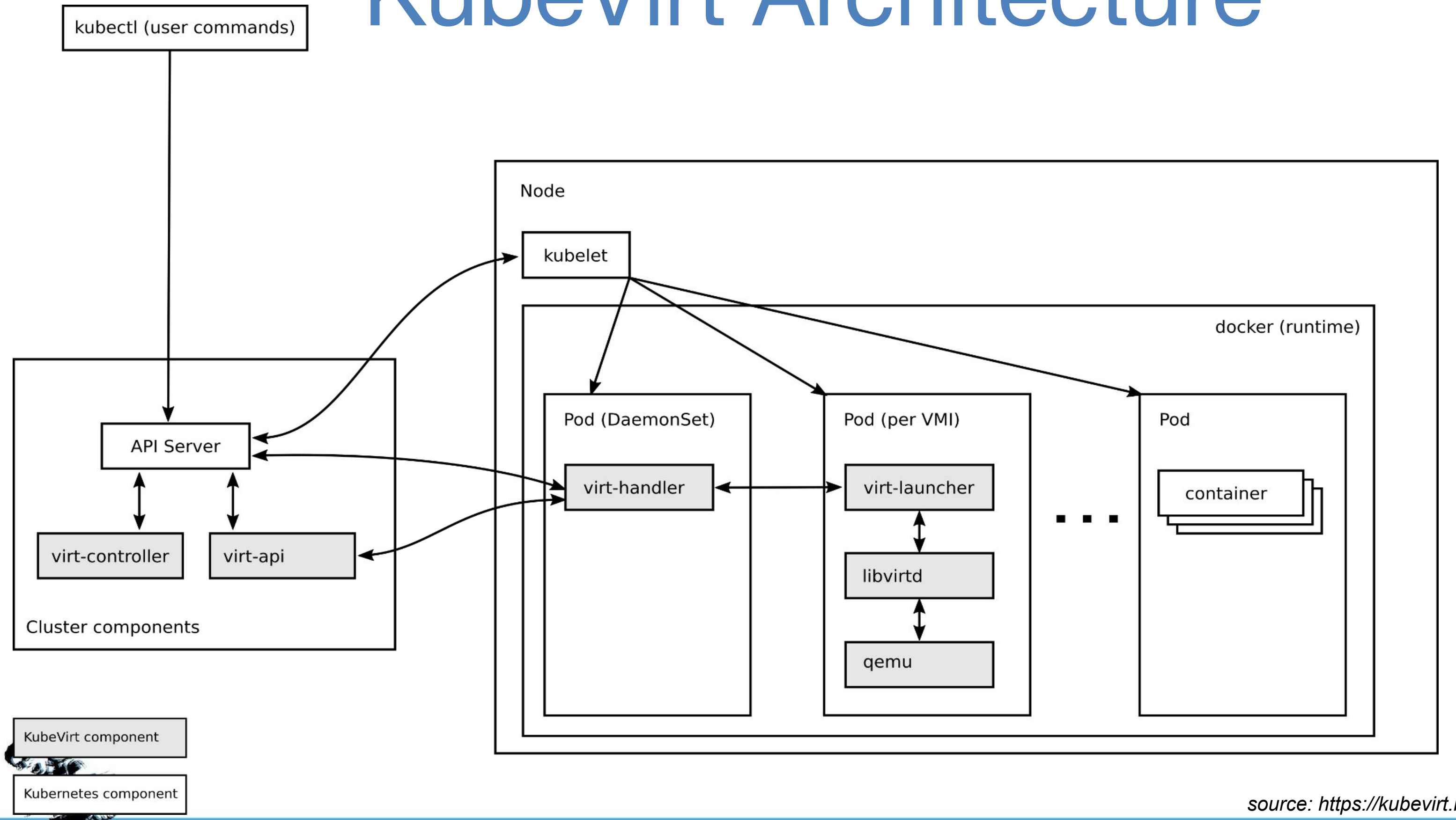
What makes a good hypervisor?

- Hyper converged storage support
- VLAN / SDN support
- Live migration of virtual machines
- Templating
  - Golden images
  - cloud-init
- Snapshotting
- Memory sharing
- Overcommit on CPU and Memory
- A nice GUI
- API driven

What is KubeVirt?

- Run virtual machines on k8s
- Extends k8s API with CRDs
- Uses Libvirt, qemu and kvm
- Run containers alongside virtual machines
- Core component for:
  - Harvester
  - Openshift virtualization
- Most commits in Github repo from Red Hat → 17487 (as of 5th of september)

KubeVirt Architecture



source: https://kubevirt.io/user-guide/architecture/

KubeVirt installation on Talos

Install Kubevirt operator:

```
# Point at latest release
$ export RELEASE=$(curl
https://storage.googleapis.com/kubevirt-prow/release/kubevirt/kubevirt/stable.txt)

# Deploy the KubeVirt operator

$ kubectl apply -f \
     https://github.com/kubevirt/kubevirt/releases/download/${RELEASE}/kubevirt-operator.yaml
```

KubeVirt installation on Talos

apply Kubevirt CR:

```yaml
---
apiVersion: kubevirt.io/v1
kind: KubeVirt
metadata:
  name: kubevirt
  namespace: kubevirt
spec:
  configuration:
    developerConfiguration:
      featureGates:
        - LiveMigration
        - NetworkBindingPlugins
        - Snapshot
    smbios:
      sku: "TalosCloud"
      version: "v0.1.0"
      manufacturer: "Talos Virtualization"
      product: "talosvm"
      family: "ccio"
```

COMPUTING
ATYPICAL OPEN SOURCE GURUS

KubeVirt installation on Talos

Install virtctl

```
# install virtctl with krew
$ kubectl krew install virt
```

KubeVirt installation on Talos

Caveats:

- When using single disk nodes: make sure to upgrade with `talos upgrade --preserve=true`
- Make sure to set an exemption for the kubevirt namespace when using `PodSecurity`.
- When using Multus, make sure to configure your bridge properly:

```yaml
network:
  hostname: virt3.lan.alcatrash.net
  interfaces:
  - interface: br0
    addresses:
      - 172.16.1.62/24
    bridge:
      stp:
        enabled: true
      interfaces:
          - eno1
    routes:
        - network: 0.0.0.0/0
          gateway: 172.16.1.254
```

AT COMPUTING ATYPICAL OPEN SOURCE GURUS

CDI: Containerized data importer

- CDI is used to import disks before the creation of a VM.
- Images supported are:
  - Qcow2
  - Raw
  - Iso
- Data sources where the images come from:
  - Upload from client
  - http/https
  - Container registry
  - Another pvc
- Create a CR called `DataVolume`
- `DataVolume` will create a `PVC`.

CDI: Containerized data importer

Install CDI operator:

```
# Point to latest release
$ export TAG=$(curl -s -w %{redirect_url} \
     https://github.com/kubevirt/containerized-data-importer/releases/latest)

$ export VERSION=$(echo ${TAG##*/})

# install operator
$ kubectl create -f \
     https://github.com/kubevirt/containerized-data-importer/releases/download/$VERSION/cdi-o
     perator.yaml
```

CDI: Containerized data importer

apply CDI CR:

```yaml
---
apiVersion: cdi.kubevirt.io/v1beta1
kind: CDI
metadata:
  name: cdi
spec:
  config:
    scratchSpaceStorageClass: local-path
    featureGates:
      - HonorWaitForFirstConsumer
    podResourceRequirements:
      requests:
        cpu: "100m"
        memory: "60M"
      limits:
        cpu: "750m"
        memory: "2Gi"
```

AT COMPUTING
ATYPICAL OPEN SOURCE GURUS

CDI: Containerized data importer

Creating a DataVolume to import a base os disk:

```yaml
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
 name: debian-12-image
 namespace: virtualmachines
spec:
 source:
     http:
       url:
"https://cloud.debian.org/images/cloud/bookworm/latest/debian-12-generic-amd64
.raw"
 pvc:
   accessModes:
     - ReadWriteMany
   resources:
     requests:
       storage: 3Gi
   storageClassName: nfs-client-zimaboard
```

CDI: Containerized data importer

Creating a Datavolume from imported disk:

```yaml
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: debian-external-pvc
  namespace: virtualmachines
spec:
  source:
    pvc:
      name: debian-12-image
      namespace: virtualmachines
  pvc:
    accessModes:
      - ReadWriteMany
    resources:
      requests:
        storage: 10Gi
    storageClassName: longhorn-rwx
```

CDI: Containerized data importer

Caveats

- Make sure to install the local path provisioner. If not installed, the importer pod will crash because it can't write any scratch space.
- Make sure to set proper limits in the CDI CR to allow the import to succeed. If not, your CDI importer pod will be `OOMKilled`.

But what about.... ?

- Vlans with and networking? → Multus
- Shared and Hyperconverged storage? → NFS and Longhorn
- Live migration? → Kubevirt Livemigration CR

Vlans and networking:

- Only use Multus when you want to expose your vm to the external network.
- First: make sure your bridge is configured properly
- Install Multus:
  - CNI plugin to attach multiple interfaces to pods
  - Make sure to patch the `DaemonSet`
- Install Whereabouts: Assigns ip addresses cluster wide to your VMs.
  - Only needed when using multiple nodes in your cluster.
- Create your `NetworkAttachmentDefinition` CR
- Connect your VirtualMachine NIC to that `NetworkAttachmentDefinition`

Vlans and networking:

Patch the Multus DaemonSet:

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: kube-multus-ds
  namespace: kube-system
spec:
  template:
    spec:
...
      volumes:
        - name: host-run-netns
          hostPath:
            path: ~~/run/netns~~ /var/run/netns
```

Vlans and networking:

Create a NetworkAttachmentDefinition:

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: bridge-whereabouts
  namespace: virtualmachines
spec:
  config: '{
      "cniVersion": "0.3.1",
      "name": "bridge-whereabouts",
      "type": "bridge",
      "bridge": "br0",
      "ipam": {
        "type": "whereabouts",
        "range": "172.16.1.0/24",
        "range_start": "172.16.1.120",
        "range_end": "172.16.1.150",
        "gateway": "172.16.1.254",
        "routes": [
          { "dst": "0.0.0.0/0" }
        ]
      }
    }'
```

Vlans and networking:

Attaching the NetworkAttachmentDefintion to your VM:

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
 name: debian-external-vm
spec:
 template:
   spec:
     domain:
       ...
         interfaces:
         - name: external
           bridge: {}
     networks:
     - name: external
       multus:
         default: true
         networkName: virtualmachines/bridge-whereabouts
```

Shared and hyper converged storage:

- Local path provisioner → Only used for temporary storage for CDI
- NFS CSI or NFS Subdir provisioner → Only for ISO's and disk images
- Longhorn for Hyperconverged storage
- Why not use Rook Ceph?
  - Single disk is not supported
- For Longhorn: Create a storage class with **`ReadWriteMany`** → used for Live migration

Shared and hyper converged storage:

Create Longhorn `StorageClass` with `ReadWriteMany`:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: longhorn-rwx
provisioner: driver.longhorn.io
allowVolumeExpansion: true
reclaimPolicy: Delete
volumeBindingMode: Immediate
parameters:
  numberOfReplicas: "3"
  staleReplicaTimeout: "2880"
  fromBackup: ""
  fsType: "ext4"
  nfsOptions: "vers=4.2,noresvport,softerr,timeo=600,retrans=5"
```

Shared and hyper converged storage:

Some more information:

- Longhorn uses ISCSI for `ReadWriteOnce`
    - Make sure to install iscsi extension
- Longhorn creates a NFS Server per replica item when using `ReadWriteMany` storageclass.
- NFS Server based on the Ganesha project
- CSI provisioner will create a PVC on that NFS Server
- Make sure to configure a bind mount to `/var/lib/longhorn` on your worker nodes.

Live Migration

- Only available when:
  - Using PVC's with `ReadWriteMany StorageClass`
  - LiveMigration is enabled in the FeatureGate of the Kubevirt CR
- Can be initiated when executing: **`kubectl virt migrate <virtualmachinename>`**
- Creates a new CR called `VirtualMachineInstanceMigration`
- When using different cpu's in your cluster, make sure to set a CPU type in your VM
  - Kubevirt sets labels with `cpu-model-migration.node.kubevirt.io/<cpu-type>`. Choose the CPU type based on those labels.

Live Migration

Set CPU type:

```yaml
---
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: debian-external-vm
spec:
  running: true
  template:
    spec:
      domain:
        cpu:
          cores: 2
          model: Haswell-noTSX-IBRS
```

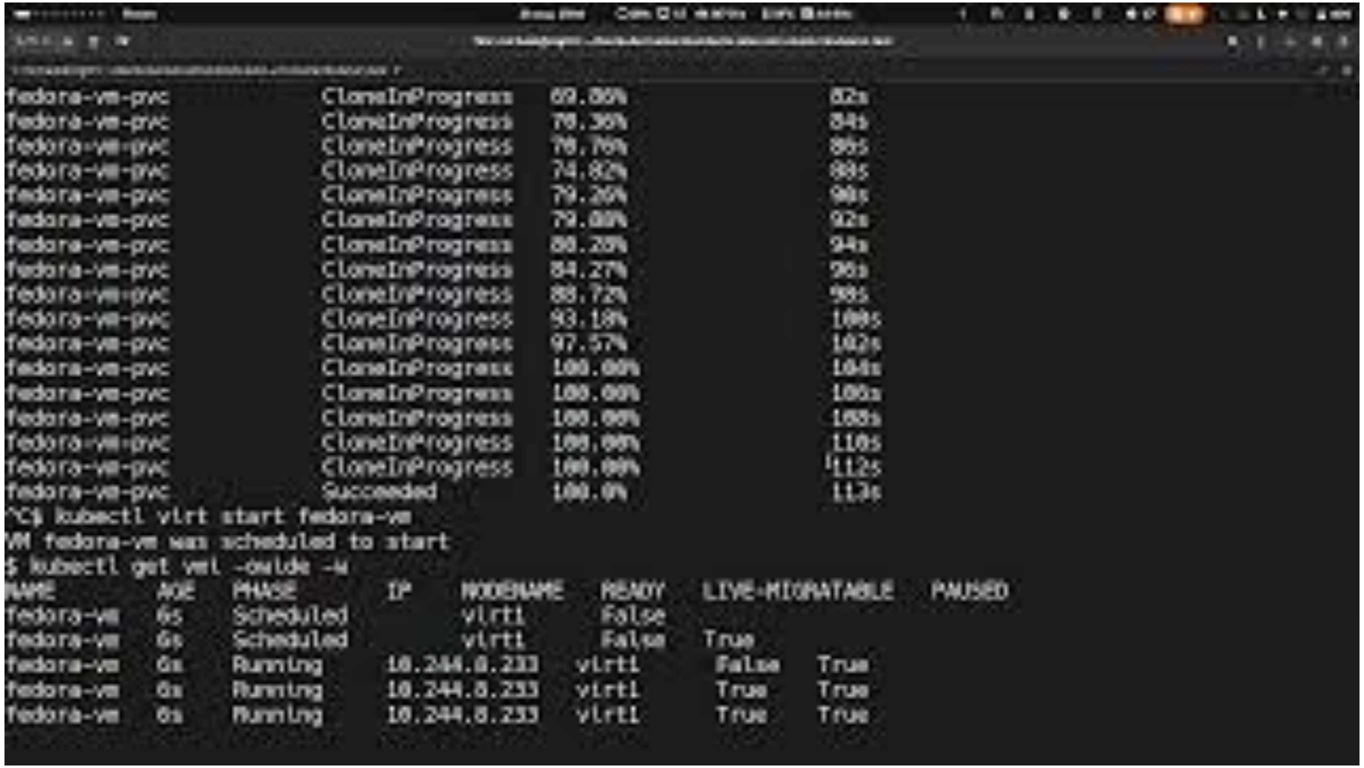
What about... ?

- Templating → Supported with the `VirtualMachineClone` CR
- Snapshotting → Supported with the `VirtualMachineSnapshot` CR
  - CSI must support `VolumeSnapshotClasses`
- Memory sharing and overcommitment:
  - KSM (Memory sharing) not supported on Talos
    - `CONFIG_KSM` not enabled in kernel
  - Overcommitment is supported in Kubevirt → beta.

Demo time

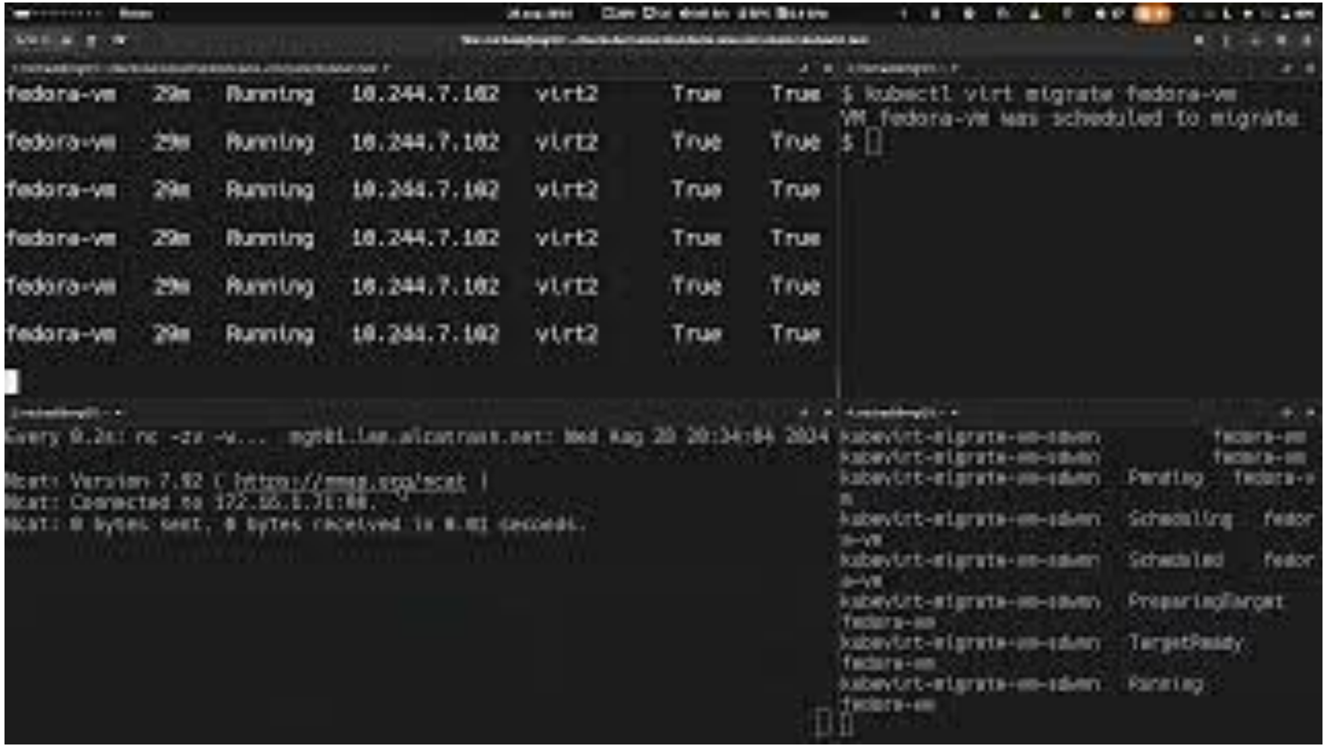
- Creating a virtual machine
- Migrating a virtual machine to another node

Demo - Create a Virtual machine



```
fedora-vm-pvc              CloneInProgress   69.86%                 82s
fedora-vm-pvc              CloneInProgress   70.36%                 84s
fedora-vm-pvc              CloneInProgress   70.76%                 86s
fedora-vm-pvc              CloneInProgress   74.82%                 88s
fedora-vm-pvc              CloneInProgress   79.26%                 90s
fedora-vm-pvc              CloneInProgress   79.08%                 92s
fedora-vm-pvc              CloneInProgress   80.26%                 94s
fedora-vm-pvc              CloneInProgress   84.27%                 96s
fedora-vm-pvc              CloneInProgress   88.72%                 98s
fedora-vm-pvc              CloneInProgress   93.18%                 100s
fedora-vm-pvc              CloneInProgress   97.57%                 102s
fedora-vm-pvc              CloneInProgress   100.00%                104s
fedora-vm-pvc              CloneInProgress   100.00%                106s
fedora-vm-pvc              CloneInProgress   100.00%                108s
fedora-vm-pvc              CloneInProgress   100.00%                110s
fedora-vm-pvc              CloneInProgress   100.00%               '112s
fedora-vm-pvc              Succeeded         100.0%                 113s
^C$ kubectl virt start fedora-vm
VM fedora-vm was scheduled to start
$ kubectl get vmi -owide -w
NAME        AGE   PHASE        IP            NODENAME   READY   LIVE-MIGRATABLE   PAUSED
fedora-vm   6s    Scheduled                  virt1      False
fedora-vm   6s    Scheduled                  virt1      False   True
fedora-vm   6s    Running      10.244.0.233   virt1      False   True
fedora-vm   6s    Running      10.244.0.233   virt1      True    True
fedora-vm   6s    Running      10.244.0.233   virt1      True    True
```

Demo - Live migration of VM



```
fedora-vm    29m   Running   10.244.7.102   virt2       True    True
fedora-vm    29m   Running   10.244.7.102   virt2       True    True
fedora-vm    29m   Running   10.244.7.102   virt2       True    True
fedora-vm    29m   Running   10.244.7.102   virt2       True    True
fedora-vm    29m   Running   10.244.7.102   virt2       True    True
fedora-vm    29m   Running   10.244.7.102   virt2       True    True
```

```
$ kubectl virt migrate fedora-vm
VM fedora-vm was scheduled to migrate
$
```

```
Every 0.2s: nc -zv -w...  mgt01.lan.alcatraas.net: Wed Aug 28 20:34:04 2024

Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to 172.16.1.71:80.
Ncat: 0 bytes sent, 0 bytes received in 0.01 seconds.
```

```
kubevirt-migrate-vm-[illegible]                fedora-vm
kubevirt-migrate-vm-[illegible]                fedora-vm
kubevirt-migrate-vm-[illegible]   Pending      fedora-v
m
kubevirt-migrate-vm-[illegible]   Scheduling     fedor
a-vm
kubevirt-migrate-vm-[illegible]   Scheduled      fedor
a-vm
kubevirt-migrate-vm-[illegible]   PreparingTarget
fedora-vm
kubevirt-migrate-vm-[illegible]   TargetReady
fedora-vm
kubevirt-migrate-vm-[illegible]   Running
fedora-vm
```

Takeaways

- Talos is awesome!
- KubeVirt is awesome!
- No enterprise grade GUI available:
  - KubeVirt Manager is the most promising
  - You could use Openshift Console → Does work for basic virtual machine mgmt
- Steep learning curve
- A lot of moving parts
- Watch out for caveats:
  - Multus → patch the `DaemonSet`
  - CDI → Local path storage to write scratch space
  - Configure your bridge properly in Talos
  - Upgrade with `--preserve=true`. If not, Longhorn storage is gone! :(

Conclusion

Enterprise grade production ready? **NO**:

- No KSM Support at this moment
- Overcommitment still in beta phase
- No enterprise grade GUI available

Startup grade production ready? **YES**:

- When staff is qualified enough to use Kubernetes / Talos / Kubevirt
- When not relying on GUI and overcommitment of CPU and memory
- Calculating a steep learning curve when designing a Kubevirt cluster

Questions?

Where to find me:

- Blog: https://michaeltrip.nl
- Email: m.trip@ATComputing.nl or michael@alcatrash.org
- Github: http://github.com/michaeltrip
- LinkedIn: https://www.linkedin.com/in/michaeltrip/

https://github.com/michaeltrip/taloscon2024/