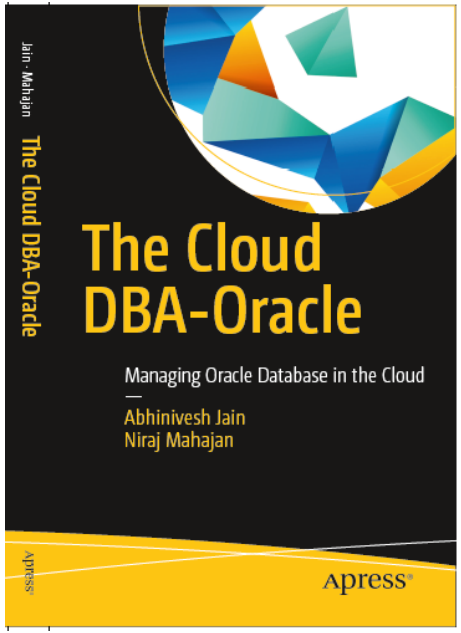
Using KubeVirt in Telcos

Abhinivesh Jain
Distinguished Member of Technical Staff
23-Sep-2020

About Me

- Author, Speaker and Blogger
- Open Source “Contributor”
- Working as **Distinguished Member of Technical Staff (DMTS)- Senior Member** in CTO-5G Team in Wipro



@AbhiniveshJain

http://abhiniveshjain.blogspot.in/

/abhiniveshjain

Agenda

1. Current Challenges in Telco
2. KubeVirt overview
3. KubeVirt role in Telco
4. KubeVirt in Action
5. Setup steps
6. Lesson learnt
7. Key Takeaways

Current Challenges

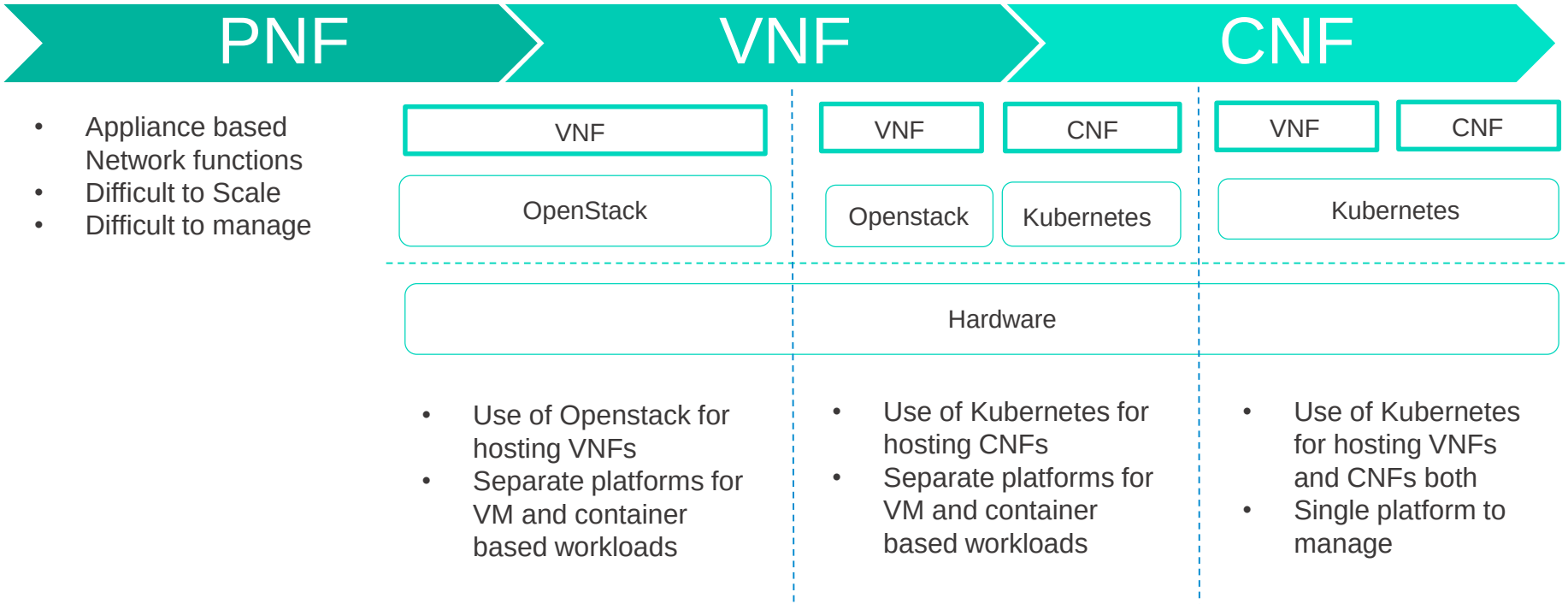
Legacy Apps are here to stay!!!

- Legacy Apps that can’t be containerized because-
  - No source code
  - Not architected for Containers
  - Uses Telco vendor proprietary OS
- Non-replaceable Legacy Apps

VNFs are here to stay!!!

- Longer cycle of VNF to CNF conversion
- CNF is still evolving to match Telco needs
- Only VNFs in 4.5G (LTE Advanced)
- 4.5G and NSA-5G will co-exist for long time
- Kubernetes is still adapting to Telco specific requirement

No Single hosting platform for VNF & CNF



Can Kubernetes host both VNFs and CNFs?

No Single Multi access Edge computing (MEC) platform

- MEC Platforms require VNFs and CNFs hosting capabilities
- MEC platform require to host third party apps(VM/Container)
- No Native support for VM hosting in Kubernetes
- Fixing above requires 2 sets of MEC platforms
  - Openstack based
  - Kubernetes based
- This increases the cost and complexity of MEC platform management

Can we have just one MEC platform that will satisfy both the needs?

KubeVirt Overview

KubeVirt

- Helps in marrying VM and Container world
- CNCF Sandbox project
- Allows us to run VM inside a POD
- Allows you to manage VM similar to POD

Eventual Containerization

- Cloud Native adoption is accelerating
- Eventual containerization (EC) enables faster adoption of Cloud Native
- With EC, it is possible to remove VM hosting platform even before 100% container adoption

KubeVirt Role in Telcos

KubeVirt Role in Telcos

Single Compute platform for VNFs and CNFs

Single MEC platform for VM based and Container based MEC Apps

Uniform development experience

Easier management

Reusing Kubernetes skills

KubeVirt in Action

Case Study

Objective here is to run Windows 2012 ISO based image on Kubernetes platform based on Openshift 4.2. This is to show how VM based workload can run inside Kubernetes.

Windows VM hosting is more complex than Linux

Prerequisites

- Openshift 4.2 cluster is up and running
- Windows 2012 ISO image
- Internet access to download CDI, KubeVirt, Virtctl, remote viewer
- 25G PVC for hard drive where windows will be installed

High Level Steps

1. Configure CDI
2. Configure KubeVirt
3. Image upload using Virtctl
4. Create PV for hardisk that will hold the windows installation
5. Create Windows VM using sample yaml file
6. Start VM using virtctl
7. Connect to VM using VNC
8. Install Windows

Configure CDI

1. Configure CDI
2. Configure KubeVirt
3. Image upload using Virtctl
4. Create PV for hardisk that will hold the windows installation
5. Create Windows VM using sample yaml file
6. Start VM using virtctl
7. Connect to VM using VNC
8. Install Windows

```
[root@localhost win2012iso]# export VERSION=$(curl -s https://github.com/kubevirt/containerized-data-importer/releases/latest | grep -o "v[0-9]\.[0-9]*\.[0-9]*")

[root@localhost win2012iso]# echo $VERSION
v1.22.0

[root@localhost win2012iso]# oc create -f https://github.com/kubevirt/containerized-data-importer/releases/download/$VERSION/cdi-operator.yaml

namespace/cdi created
customresourcedefinition.apiextensions.k8s.io/cdis.cdi.kubevirt.io created
clusterrole.rbac.authorization.k8s.io/cdi-operator-cluster created
clusterrolebinding.rbac.authorization.k8s.io/cdi-operator created
serviceaccount/cdi-operator created
role.rbac.authorization.k8s.io/cdi-operator created
rolebinding.rbac.authorization.k8s.io/cdi-operator created
deployment.apps/cdi-operator created
configmap/cdi-operator-leader-election-helper created

[root@localhost win2012iso]# oc create -f https://github.com/kubevirt/containerized-data-importer/releases/download/$VERSION/cdi-cr.yaml
cdi.cdi.kubevirt.io/cdi created
```

Configure Kubevirt

1. Configure CDI
2. Configure KubeVirt
3. Image upload using Virtctl
4. Create PV for hardisk that will hold the windows installation
5. Create Windows VM using sample yaml file
6. Start VM using virtctl
7. Connect to VM using VNC
8. Install Windows

Choose appropriate version, V0.26.0 is given as example.

oc create namespace kubevirt

oc apply -f https://github.com/kubevirt/kubevirt/releases/download/v0.26.0/kubevirt-operator.yaml

oc apply -f https://github.com/kubevirt/kubevirt/releases/download/v0.26.0/kubevirt-cr.yaml

Apply kubevirt scc for openshift

kubevirt_scc.yaml

If you are having rook-ceph then apply

https://github.com/rook/rook/blob/master/cluster/examples/kubernetes/ceph/upgrade-from-v1.2-apply.yaml

Image upload

1. Configure CDI
2. Configure KubeVirt
3. Image upload using Virtctl
4. Create PV for hardisk that will hold the windows installation
5. Create Windows VM using sample yaml file
6. Start VM using virtctl
7. Connect to VM using VNC
8. Install Windows

Install virtctl using below commands. Once again, take latest version and it should be similar to KubeVirt.

```
curl -L -o virtctl https://github.com/kubevirt/kubevirt/releases/download/v0.26.0/virtctl-v0.26.0-linux-amd64

chmod +x virtctl
```

Now upload this image using below command. Give uploadproxy IP that you get from oc get svc –n kubevirt command output.

```
[root@mycluster-master-0 tmp]#  ./virtctl image-upload --uploadproxy-url=https://x.x.x.x:443 --pvc-name=iso-win2k12-pvc --access-mode=ReadOnlyMany --pvc-size=25Gi --image-path=/tmp/Win2k12R2.ISO --insecure --wait-secs=300
```

Image upload…Cont'd

1. Configure CDI
2. Configure KubeVirt
3. Image upload using Virtctl
4. Create PV for hardisk that will hold the windows installation
5. Create Windows VM using sample yaml file
6. Start VM using virtctl
7. Connect to VM using VNC
8. Install Windows

Upload command output will look like below.

This command will create 2 PVCs, win2k12-pvc and win2k12-pvc-scratch of same size (25Gi) as given in below command. Scratch PVC is temporary and it will be deleted automatically after successful image upload.

```
[root@mycluster-master-0 tmp]#  ./virtctl image-upload --uploadproxy-url=https://x.x.x.x:443 --pvc-name=win2k12-pvc --access-mode=ReadOnlyMany --pvc-size=25Gi --image-path=/tmp/Win2k12R2.ISO --insecure --wait-secs=300

Using existing PVC rook-ceph/iso-win2k12-pvc
Waiting for PVC iso-win2k12-pvc upload pod to be ready...
Pod now ready
Uploading data to https://x.x.x.x:443

 4.17 GiB / 4.17 GiB [==========================================================] 100.00% 1m13s

Uploading data completed successfully, waiting for processing to complete, you can hit ctrl-c without interrupting the progress
Processing completed successfully
Uploading /tmp/Win2k12R2.ISO completed successfully
```

Creat PV

1. Configure CDI
2. Configure KubeVirt
3. Image upload using Virtctl
4. Create PV for hardisk that will hold the windows installation
5. Create Windows VM using sample yaml file
6. Start VM using virtctl
7. Connect to VM using VNC
8. Install Windows

Put below in one .yaml file and apply it using oc apply –f <filename>

Remember to update storageClassName to appropriate value. You can update this based on output of oc get storageclass output.

rook-filesystem should be used if rook-ceph is in place.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: windowsdrive
spec:
  accessModes:
    - ReadWriteOnce
resources:
  requests:
    storage: 25Gi
storageClassName: rook-filesystem
```

Create Windows VM

1. Configure CDI
2. Configure KubeVirt
3. Image upload using Virtctl
4. Create PV for hardisk that will hold the windows installation
5. Create Windows VM using sample yaml file
6. Start VM using virtctl
7. Connect to VM using VNC
8. Install Windows

Now, it is time to create VM using the ISO image that was uploaded earlier. Before this step, you need to attach virtio driver as a cdrom. You can do this using podman/docker by giving below command.

[root@mycluster-master-0 tmp]# podman pull kubevirt/virtio-container-disk

Trying to pull docker.io/kubevirt/virtio-container-disk...Getting image source signatures
Copying blob 65ceadabbfb7 done
Copying config d5ffba0407 done
Writing manifest to image destination
Storing signatures
d5ffba0407e8874891f00ec44168d2d5fc7ba4968a39c22c725a2946c226d2ee

Verify it using podman images command. Once above is done, you are good to run create vm yaml file. Sample is given below.

vmi_windows.yaml

[root@mycluster-master-0 tmp]# oc create –f vmi_windows.yaml
virtualmachineinstance.kubevirt.io/vmi-windows created

Start Windows VM

1. Configure CDI
2. Configure KubeVirt
3. Image upload using Virtctl
4. Create PV for hardisk that will hold the windows installation
5. Create Windows VM using sample yaml file
6. Start VM using virtctl
7. Connect to VM using VNC
8. Install Windows

By Default VM is in shutdown mode so first thing you need to do is to start it.

```
[root@mycluster-master-0 tmp]# oc get vms

NAME          AGE     RUNNING   VOLUME
samplevm       3m      false

[root@mycluster-master-0 tmp]# ./virtctl start vm samplevm

VM samplevm was scheduled to start

[root@mycluster-master-0 tmp]# oc get vm

NAME          AGE   RUNNING   VOLUME
samplevm       76s      true
```

This VM will create the VMI and you will see the VMI running in few minutes.

```
[root@mycluster-master-0 tmp]# oc get vmi
NAME          AGE   PHASE     IP           NODENAME
samplevm   5m   Running   10.x.x.x         worker-1
```

Connect to Windows VM

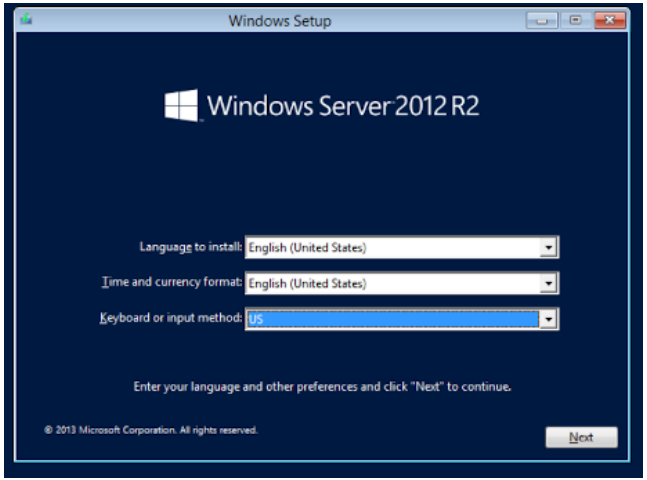
1. Configure CDI
2. Configure KubeVirt
3. Image upload using Virtctl
4. Create PV for hardisk that will hold the windows installation
5. Create Windows VM using sample yaml file
6. Start VM using virtctl
7. Connect to VM using VNC
8. Install Windows

Now it is time to connect to VM using VNC. This command needs to be executed from a host which is capable of showing display. You can use MobaXTerm or any other such software.

[root@localhost tmp]# ./virtctl vnc samplevm

If it fails with error like remote_viewer not present then install remote_viewer using below command. If not then you will see windows installation screen as shown below.

[root@localhost tmp]# yum install virt-viewer



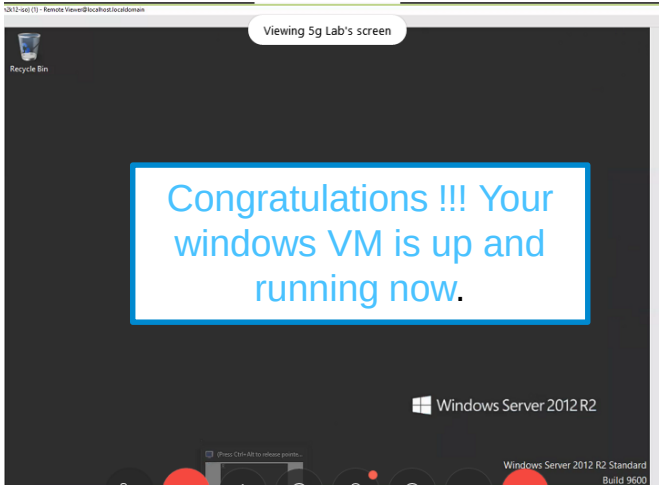
Connect to Windows VM

1. Configure CDI
2. Configure KubeVirt
3. Image upload using Virtctl
4. Create PV for hardisk that will hold the windows installation
5. Create Windows VM using sample yaml file
6. Start VM using virtctl
7. Connect to VM using VNC
8. Install Windows

Complete the windows installation by following the below Video.

https://kubevirt.io/assets/2020-02-14-KubeVirt-installing_Microsoft_Windows_from_an_iso/kubevirt_install_windows.mp4

Remember your mouse pointer won't work for most of the screens so you need to use keys like- Tab(to toggle between options), spacebar(for checkbox selection), enter(for selection), right arrow key(for expansion) etc.
After successful installation, you will see screen similar to below.



Lesson Learnt

Lesson Learnt

Putting VM in a POD results in nested virtualization hence it has some performance overheads.

Currently several features are work in progress, like- you can’t increase CPU/Memory on the fly.

Always Run virtctl image upload from master node.

Always use latest version of KubeVirt for client and server

Common Challenges

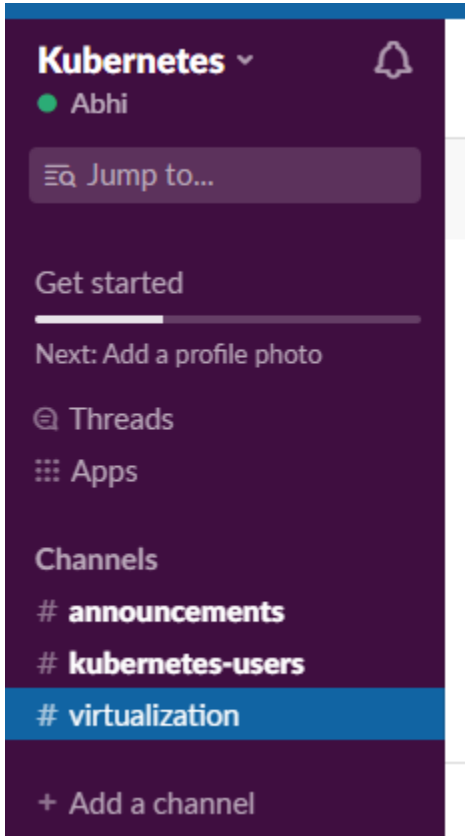
Common Challenges

- Rook-ceph permission issue because of this image was not getting uploaded.
- Kubevirt bug related to VNC due to which vnc connect to windows machine wasn’t working. Upgrading Kubevirt helped in fixing this.
- Sometimes Openshift cluster operator “authentication” gets degraded, due to which “no route to host” error comes.

Key Takeaways

Key Takeaways

- KubeVirt is maturing at very fast pace hence some issues are expected.
- KubeVirt Slack channel is your best friend.
- Refer to https://github.com/kubevirt/kubevirt/tree/master/examples



Is KubeVirt Telco Ready?

Feature	Supported by KubeVirt
Huge Page support	Yes
SR-IOV support	Yes
CPU pinning, NUMA support	Yes
Multi Interface support	Yes
Live Migration	Conditional
Hot-plug	No
Fencing (to handle node failures)	Partial
ARM64 support	No
GPU and FPGA	No

Questions?

Appendix

Public

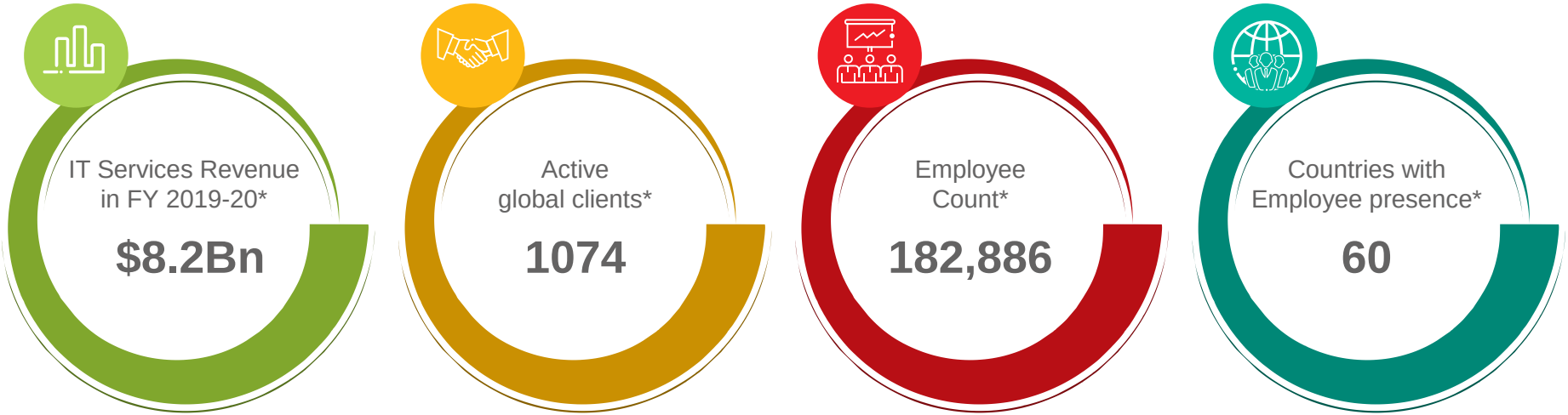
Kubevirt_scc.yaml

```
allowHostDirVolumePlugin: true
allowHostIPC: true
allowHostNetwork: true
allowHostPID: true
allowHostPorts: true
allowPrivilegeEscalation: true
allowPrivilegedContainer: true
allowedCapabilities:
- '*'
allowedUnsafeSysctls:
- '*'
apiVersion: security.openshift.io/v1
defaultAddCapabilities: []
fsGroup:
  type: RunAsAny
groups:
- system:cluster-admins
- system:nodes
- system:masters
kind: SecurityContextConstraints
metadata:
  name: privileged
priority: 10
readOnlyRootFilesystem: false
requiredDropCapabilities: []
runAsUser:
  type: RunAsAny
seLinuxContext:
  type: RunAsAny
seccompProfiles:
- '*'
supplementalGroups:
  type: RunAsAny
users:
- system:admin
- system:serviceaccount:openshift-infra:build-controller
- system:serviceaccount:kubevirt:kubevirt-operator
- system:serviceaccount:kubevirt:kubevirt-handler
- system:serviceaccount:kubevirt:kubevirt-apiserver
- system:serviceaccount:kubevirt:kubevirt-controller
volumes:
- '*'
```

vmi_windows.yaml

```
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachine
metadata:
  name: samplevm
spec:
  running: false
  template:
    metadata:
      labels:
        kubevirt.io/domain: samplevm
    spec:
      domain:
        cpu:
          cores: 4
        devices:
          disks:
          - bootOrder: 1
            cdrom:
              bus: sata
            name: cdromiso
          - disk:
              bus: virtio
            name: harddrive
          - cdrom:
              bus: sata
            name: virtiocontainerdisk
        machine:
          type: q35
        resources:
          requests:
            memory: 8G
      volumes:
      - name: cdromiso
        persistentVolumeClaim:
          claimName: win2k12-pvc
      - name: harddrive
        persistentVolumeClaim:
          claimName: windowsdrive
      - containerDisk:
          image: kubevirt/virtio-container-disk
        name: virtiocontainerdisk
```

Wipro today



*Figures based on FY 2019-20 for Global IT Services business.

Thanks

Public