RH OVE Operations Documentation

Red Hat OpenShift Virtualization Ecosystem Team

2025-08-04

Table of Contents

# Operations

## Day2 Ops

### Day-2 Operations

#### Overview

This document covers day-2 operational activities essential for maintaining the multi-cluster RH OVE ecosystem. It includes guidelines for managing the management cluster and multiple application clusters, covering ongoing maintenance, upgrades, performance tuning, and operational tasks across the entire fleet.

#### Maintenance Tasks

##### Regular Cluster Health Checks

* **Node Status Monitoring**: Regularly check node health and availability.
* oc get nodes -o wide
* **Resource Usage Monitoring**: Monitor CPU, memory, and storage utilization.
* oc adm top nodes
oc adm top pods --all-namespaces

##### Backup Management

* **Review Backup Logs**: Ensure completion and verify logs for any anomalies.
* oc logs -n rubrik rubrik-agent-
* **Data Integrity Checks**: Periodically verify backup integrity and accessibility.

#### Upgrades

##### OpenShift Cluster Upgrades

* **Plan Your Upgrade**: Evaluate impact, and schedule during maintenance windows.
	+ Review [OpenShift Upgrade Guide](https://docs.openshift.com/upgrade/)
* **In-place Upgrades**: Use OpenShift’s upgrade capabilities to update cluster components.
* oc adm upgrade

##### Component Upgrades

* **Operator Lifecycle Management (OLM)**: Upgrade operators using OLM.
* oc get clusterserviceversions -n openshift-operators
* **KubeVirt Upgrades**: Follow the KubeVirt upgrade process for virtualization components.
	+ Refer to [KubeVirt Upgrade Guide](https://kubevirt.io/upgrade-guide/)

#### Performance Tuning

##### Resource Balancing

* **Node Selector and Affinity Rules**: Ensure workloads are distributed evenly.
* apiVersion: v1
kind: Pod
metadata:
 name: example-pod
spec:
 affinity:
 nodeAffinity:
 requiredDuringSchedulingIgnoredDuringExecution:
 nodeSelectorTerms:
 - matchExpressions:
 - key: disktype
 operator: In
 values:
 - ssd
* **Vertical and Horizontal Scaling**: Utilize HPA and VPA for scaling applications.

##### Network Optimization

* **Cilium Policy Management**: Optimize and tune Cilium network policies for performance.
* apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
metadata:
 name: optimized-policy
spec:
 endpointSelector:
 matchLabels:
 app: myapp
 ingress:
 - fromEndpoints:
 - matchLabels:
 app: trusted

#### Security and Compliance

##### Regular Security Audits

* **Policy Compliance**: Ensure adherence to Kyverno policies and security standards.
* kubectl get cpol -o yaml
* **Vulnerability Scans**: Run regular vulnerability assessments on container images and hosts.

#### Documentation and Reporting

##### Keeping Documentation Up-to-Date

* **Change Logs**: Maintain a changelog for all configurations and updates.
* **Operational Runbooks**: Create and update runbooks for standard operations.

##### Performance and Utilization Reports

* **Utilize Metrics Dashboards**: Use Grafana and Prometheus to generate reports.

#### Conclusion

Following these day-2 operation guidelines helps maintain a stable, secure, and efficient RH OVE environment. Regular monitoring, updates, optimizations, and documentation ensure long-term success and reliability of the platform.

## Troubleshooting

### Troubleshooting Guide

#### Overview

This comprehensive troubleshooting guide addresses common issues in the RH OVE ecosystem, providing systematic approaches to diagnose and resolve problems across virtualization, networking, storage, and monitoring components.

#### General Troubleshooting Approach

##### Diagnostic Flow



##### Essential Commands

### Cluster overview
oc get nodes
oc get pods --all-namespaces
oc get events --all-namespaces --sort-by='.lastTimestamp'

### Resource utilization
oc adm top nodes
oc adm top pods --all-namespaces

### Detailed investigation
oc describe node <node-name>
oc logs -f <pod-name> -n <namespace>

#### Virtual Machine Issues

##### VM Won’t Start

###### Symptoms

* VM remains in “Pending” or “Scheduling” state
* VM fails to boot or crashes during startup

###### Troubleshooting Steps

1. **Check VM Definition**
* oc get vm <vm-name> -o yaml
oc describe vm <vm-name>
1. **Verify Node Resources**
* oc describe nodes
oc adm top nodes
1. **Check DataVolume Status**
* oc get datavolume
oc describe datavolume <dv-name>
1. **Review Events**
* oc get events --field-selector involvedObject.name=<vm-name>

###### Common Solutions

* **Insufficient Resources**: Scale cluster or adjust VM specs
* **DataVolume Issues**: Check CDI logs and storage classes
* **Node Affinity**: Verify node selector and affinity rules

##### VM Performance Issues

###### Symptoms

* Slow VM performance
* High CPU/memory usage
* Network latency

###### Troubleshooting Steps

1. **Check VM Resource Allocation**
* oc get vm <vm-name> -o jsonpath='{.spec.template.spec.domain.resources}'
1. **Monitor VM Metrics**
* # Use virtctl to access VM console
virtctl console <vm-name>

# Check VM performance inside guest
top
iostat
iftop
1. **Verify Host Resources**
* oc adm top node <node-name>
oc describe node <node-name>

###### Solutions

* Adjust VM CPU/memory allocation
* Enable CPU pinning for critical VMs
* Check storage performance and IOPS limits

#### Networking Issues

##### Cilium Network Problems

###### Symptoms

* Pods cannot communicate
* Network policies not working
* DNS resolution failures

###### Troubleshooting Steps

1. **Check Cilium Status**
* cilium status
cilium connectivity test
1. **Verify Network Policies**
* oc get cnp
oc describe cnp <policy-name>
1. **Monitor Network Flows**
* hubble observe --pod <pod-name>
hubble observe --verdict DENIED

###### Common Solutions

### Debug network connectivity
apiVersion: v1
kind: Pod
metadata:
 name: network-debug
spec:
 containers:
 - name: debug
 image: nicolaka/netshoot
 command: ['sleep', '3600']

##### VM Network Connectivity

###### Symptoms

* VM cannot reach external networks
* Inter-VM communication failures
* Service discovery issues

###### Troubleshooting Steps

1. **Check VM Network Configuration**
* oc get vm <vm-name> -o yaml | grep -A 10 networks
1. **Verify Service Configuration**
* oc get svc
oc describe svc <service-name>
1. **Test Connectivity from VM**
* virtctl console <vm-name>
# Inside VM:
ping <target-ip>
nslookup <service-name>

#### Storage Issues

##### DataVolume Problems

###### Symptoms

* DataVolume stuck in “Pending” state
* Import/clone operations failing
* Storage quota exceeded

###### Troubleshooting Steps

1. **Check DataVolume Status**
* oc get datavolume
oc describe datavolume <dv-name>
1. **Review CDI Logs**
* oc logs -n cdi deployment/cdi-controller
oc logs -n cdi deployment/cdi-operator
1. **Verify Storage Classes**
* oc get storageclass
oc describe storageclass <sc-name>

###### Solutions

### Debug DataVolume with verbose logging
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
 name: debug-dv
 annotations:
 cdi.kubevirt.io/debug: "true"
spec:
 pvc:
 accessModes: [ReadWriteOnce]
 resources:
 requests:
 storage: 10Gi
 source:
 blank: {}

##### Storage Performance Issues

###### Symptoms

* Slow disk I/O
* High storage latency
* VM disk full errors

###### Troubleshooting Steps

1. **Check Storage Metrics**
* # Prometheus queries
kubectl port-forward -n monitoring svc/prometheus 9090:9090
# Query: kubelet\_volume\_stats\_used\_bytes
1. **Verify PVC Usage**
* oc get pvc
oc describe pvc <pvc-name>
1. **Monitor Storage Node Performance**
* oc adm top nodes
iostat -x 1

#### Monitoring Issues

##### Dynatrace Agent Problems

###### Symptoms

* Missing VM metrics in Dynatrace
* OneAgent not reporting data
* High resource usage by monitoring

###### Troubleshooting Steps

1. **Check OneAgent Status**
* oc get pods -n dynatrace
oc describe pod <oneagent-pod>
1. **Verify VM Annotations**
* oc get vm -o yaml | grep -A5 annotations
1. **Review Dynatrace Logs**
* oc logs -n dynatrace <oneagent-pod>

##### Prometheus Metrics Missing

###### Symptoms

* Missing metrics in Grafana
* ServiceMonitor not working
* Prometheus targets down

###### Troubleshooting Steps

1. **Check ServiceMonitor Configuration**
* oc get servicemonitor
oc describe servicemonitor <sm-name>
1. **Verify Metrics Endpoints**
* oc port-forward svc/<service-name> 8080:8080
curl localhost:8080/metrics
1. **Check Prometheus Targets**
* # Access Prometheus UI
oc port-forward -n monitoring svc/prometheus 9090:9090
# Go to Status -> Targets

#### GitOps and Argo CD Issues

##### Application Sync Failures

###### Symptoms

* Applications stuck in “OutOfSync” state
* Sync operations failing
* Resource conflicts

###### Troubleshooting Steps

1. **Check Application Status**
* argocd app get <app-name>
argocd app logs <app-name>
1. **Verify Git Repository Access**
* argocd repo list
argocd repo get <repo-url>
1. **Review Resource Conflicts**
* oc get <resource-type> <resource-name> -o yaml

###### Solutions

### Force refresh and sync
argocd app refresh <app-name>
argocd app sync <app-name> --force

### Reset application state
argocd app actions run <app-name> restart --kind Deployment

#### Performance Issues

##### Cluster Resource Exhaustion

###### Symptoms

* High CPU/memory usage
* Pod evictions
* Slow response times

###### Troubleshooting Steps

1. **Identify Resource Consumers**
* oc adm top pods --all-namespaces --sort-by=cpu
oc adm top pods --all-namespaces --sort-by=memory
1. **Check Node Capacity**
* oc describe nodes | grep -A5 "Allocated resources"
1. **Review Resource Quotas**
* oc get resourcequota --all-namespaces
oc describe resourcequota <quota-name>

##### VM Live Migration Issues

###### Symptoms

* Migration fails or takes too long
* VM downtime during migration
* Network connectivity loss

###### Troubleshooting Steps

1. **Check Migration Status**
* oc get vmi
oc describe virtualmachinmigration <migration-name>
1. **Verify Node Compatibility**
* oc get nodes -o wide
oc describe node <target-node>
1. **Monitor Migration Progress**
* oc get events --field-selector reason=LiveMigration

#### Emergency Procedures

##### Cluster Recovery

###### When Multiple Nodes Are Down

1. **Check etcd Health**
* oc get etcd -o yaml
oc logs -n openshift-etcd <etcd-pod>
1. **Restore from Backup**
* # Follow OpenShift disaster recovery procedures
oc adm restore-cluster

##### VM Emergency Access

###### When VM Console Is Unresponsive

1. **Use virtctl**
* virtctl console <vm-name>
virtctl vnc <vm-name>
1. **Force VM Restart**
* virtctl restart <vm-name>
virtctl stop <vm-name> --force

#### Advanced Diagnostics

##### Debug Pod Creation

apiVersion: v1
kind: Pod
metadata:
 name: debug-tools
spec:
 containers:
 - name: debug
 image: registry.redhat.io/ubi8/ubi:latest
 command: ['sleep', '3600']
 securityContext:
 privileged: true
 volumeMounts:
 - name: host
 mountPath: /host
 volumes:
 - name: host
 hostPath:
 path: /
 nodeSelector:
 kubernetes.io/hostname: <node-name>

##### Log Collection Script

### !/bin/bash
### Comprehensive log collection script

NAMESPACE=${1:-default}
OUTPUT\_DIR="troubleshooting-$(date +%Y%m%d-%H%M%S)"

mkdir -p $OUTPUT\_DIR

### Cluster information
oc cluster-info > $OUTPUT\_DIR/cluster-info.txt
oc get nodes -o wide > $OUTPUT\_DIR/nodes.txt
oc get pods --all-namespaces > $OUTPUT\_DIR/all-pods.txt

### VM specific information
oc get vm --all-namespaces -o yaml > $OUTPUT\_DIR/vms.yaml
oc get vmi --all-namespaces -o yaml > $OUTPUT\_DIR/vmis.yaml
oc get datavolume --all-namespaces -o yaml > $OUTPUT\_DIR/datavolumes.yaml

### Events
oc get events --all-namespaces --sort-by='.lastTimestamp' > $OUTPUT\_DIR/events.txt

### Logs from key components
oc logs -n openshift-cnv deployment/virt-controller > $OUTPUT\_DIR/virt-controller.log
oc logs -n openshift-cnv deployment/virt-api > $OUTPUT\_DIR/virt-api.log
oc logs -n cdi deployment/cdi-controller > $OUTPUT\_DIR/cdi-controller.log

echo "Logs collected in $OUTPUT\_DIR"
tar -czf $OUTPUT\_DIR.tar.gz $OUTPUT\_DIR

#### Support and Escalation

##### When to Escalate

* Hardware failures
* Data corruption issues
* Security breaches
* Performance degradation > 50%
* Multiple component failures

##### Information to Gather

1. **Environment Details**
	* OpenShift version
	* KubeVirt version
	* Cluster size and configuration
2. **Problem Description**
	* Timeline of events
	* Error messages
	* Impact assessment
3. **Diagnostic Data**
	* Logs (sanitized)
	* Configuration files
	* Resource utilization data

##### Support Contacts

* **Red Hat Support**: <https://access.redhat.com/support/>
* **Community Forums**: <https://commons.openshift.org/>
* **KubeVirt Community**: <https://kubevirt.io/community/>

This troubleshooting guide provides systematic approaches to resolve common issues in the RH OVE ecosystem. Regular review and updates of this guide ensure it remains current with evolving technologies and operational experiences.

## Performance

### Performance Tuning

#### Overview

This document provides comprehensive performance tuning guidelines for the RH OVE ecosystem, covering optimization strategies for virtual machines, networking, storage, and cluster-wide performance enhancements.

#### Performance Optimization Strategy

##### Performance Monitoring Approach



##### Key Performance Indicators (KPIs)

* **VM Performance**: CPU utilization, memory usage, disk I/O, network throughput
* **Cluster Performance**: Node utilization, pod scheduling latency, API response times
* **Network Performance**: Latency, packet loss, bandwidth utilization
* **Storage Performance**: IOPS, throughput, latency

#### Virtual Machine Performance Tuning

##### CPU Optimization

###### CPU Pinning for High-Performance VMs

apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
 name: high-performance-vm
spec:
 template:
 spec:
 domain:
 cpu:
 cores: 4
 dedicatedCpuPlacement: true
 isolateEmulatorThread: true
 resources:
 requests:
 cpu: 4
 memory: 8Gi
 limits:
 cpu: 4
 memory: 8Gi
 nodeSelector:
 node-role.kubernetes.io/worker: ""
 cpumanager: "true"

###### CPU Manager Configuration

apiVersion: machineconfiguration.openshift.io/v1
kind: KubeletConfig
metadata:
 name: cpumanager-enabled
spec:
 machineConfigPoolSelector:
 matchLabels:
 pools.operator.machineconfiguration.openshift.io/worker: ""
 kubeletConfig:
 cpuManagerPolicy: static
 cpuManagerReconcilePeriod: 5s
 reservedSystemCPUs: "0,1"

###### NUMA Topology Awareness

apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
 name: numa-optimized-vm
spec:
 template:
 spec:
 domain:
 cpu:
 cores: 8
 numa:
 guestMappingPassthrough: {}
 memory:
 guest: 16Gi
 hugepages:
 pageSize: 1Gi

##### Memory Optimization

###### Hugepages Configuration

### Node configuration for hugepages
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
 name: hugepages-worker
 labels:
 machineconfiguration.openshift.io/role: worker
spec:
 config:
 ignition:
 version: 3.2.0
 systemd:
 units:
 - name: hugepages-1gi.service
 enabled: true
 contents: |
 [Unit]
 Description=Configure 1Gi hugepages
 [Service]
 Type=oneshot
 ExecStart=/bin/bash -c 'echo 8 > /sys/kernel/mm/hugepages/hugepages-1048576kB/nr\_hugepages'
 [Install]
 WantedBy=multi-user.target

###### VM Memory Configuration with Hugepages

apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
 name: memory-optimized-vm
spec:
 template:
 spec:
 domain:
 memory:
 guest: 8Gi
 hugepages:
 pageSize: 1Gi
 resources:
 requests:
 memory: 8Gi
 hugepages-1Gi: 8Gi
 limits:
 memory: 8Gi
 hugepages-1Gi: 8Gi

##### Storage Performance Optimization

###### High-Performance Storage Configuration

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
 name: high-performance-ssd
provisioner: kubernetes.io/no-provisioner
parameters:
 type: ssd
 fsType: ext4
 # Optimize for performance
 mountOptions: "noatime,nodiratime"
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer

###### VM Disk Performance Tuning

apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
 name: storage-optimized-vm
spec:
 template:
 spec:
 domain:
 devices:
 disks:
 - name: rootdisk
 disk:
 bus: virtio
 # Enable disk cache for better performance
 cache: writeback
 - name: datadisk
 disk:
 bus: virtio
 cache: none
 # Use native I/O for better performance
 io: native
 resources:
 requests:
 cpu: 2
 memory: 4Gi
 volumes:
 - name: rootdisk
 dataVolume:
 name: vm-root-disk
 - name: datadisk
 dataVolume:
 name: vm-data-disk

###### Storage I/O Optimization

apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
 name: optimized-datavolume
spec:
 pvc:
 accessModes:
 - ReadWriteOnce
 resources:
 requests:
 storage: 100Gi
 storageClassName: high-performance-ssd
 # Optimize volume for performance
 volumeMode: Block
 source:
 blank: {}

#### Network Performance Tuning

##### Cilium Performance Optimization

###### eBPF Optimization Configuration

apiVersion: v1
kind: ConfigMap
metadata:
 name: cilium-config
 namespace: kube-system
data:
 # Enable bandwidth manager for better QoS
 enable-bandwidth-manager: "true"

 # Enable local redirect policy for better performance
 enable-local-redirect-policy: "true"

 # Optimize datapath
 datapath-mode: "veth"

 # Enable XDP acceleration where supported
 enable-xdp-acceleration: "true"

 # kube-proxy replacement for better performance
 kube-proxy-replacement: "strict"

 # Optimize for performance
 enable-cilium-endpoint-slice: "true"

###### Network Device Optimization

apiVersion: v1
kind: ConfigMap
metadata:
 name: cilium-config
 namespace: kube-system
data:
 # Specify devices for optimal performance
 devices: "eth0"

 # Enable auto direct node routes
 auto-direct-node-routes: "true"

 # Optimize tunnel protocol
 tunnel: "disabled"

 # Use native routing when possible
 enable-ipv4-masquerade: "false"
 enable-ipv6-masquerade: "false"

##### VM Network Performance

###### SR-IOV Configuration for High-Performance Networking

apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetworkNodePolicy
metadata:
 name: high-performance-network
 namespace: openshift-sriov-network-operator
spec:
 nodeSelector:
 feature.node.kubernetes.io/network-sriov.capable: "true"
 nicSelector:
 vendor: "15b3"
 deviceID: "1017"
 numVfs: 8
 priority: 99
 resourceName: "high\_perf\_nic"

###### VM with SR-IOV Network Attachment

apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
 name: sriov-vm
spec:
 template:
 spec:
 domain:
 devices:
 interfaces:
 - name: default
 masquerade: {}
 - name: sriov-network
 sriov: {}
 resources:
 requests:
 cpu: 4
 memory: 8Gi
 networks:
 - name: default
 pod: {}
 - name: sriov-network
 multus:
 networkName: high-performance-network

##### Multi-Network Performance with Multus

###### Dedicated Network Interfaces for Different Traffic Types

apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
 name: multi-interface-performance-vm
 namespace: high-performance-workloads
 annotations:
 k8s.v1.cni.cncf.io/networks: |
 [
 {
 "name": "management-network",
 "ips": ["192.168.1.5/24"]
 },
 {
 "name": "storage-network",
 "ips": ["192.168.2.5/24"]
 },
 {
 "name": "sriov-data-network",
 "ips": ["10.0.0.5/24"]
 }
 ]
spec:
 running: true
 template:
 spec:
 domain:
 cpu:
 cores: 16
 dedicatedCpuPlacement: true
 isolateEmulatorThread: true
 memory:
 guest: 32Gi
 hugepages:
 pageSize: 1Gi
 devices:
 interfaces:
 - name: default
 masquerade: {}
 - name: management
 bridge:
 port: []
 - name: storage
 bridge:
 port: []
 - name: sriov-data
 sriov: {}
 disks:
 - name: rootdisk
 disk:
 bus: virtio
 cache: writeback
 resources:
 requests:
 cpu: 16
 memory: 32Gi
 hugepages-1Gi: 32Gi
 limits:
 cpu: 16
 memory: 32Gi
 hugepages-1Gi: 32Gi
 networks:
 - name: default
 pod: {}
 - name: management
 multus:
 networkName: management-network
 - name: storage
 multus:
 networkName: storage-network
 - name: sriov-data
 multus:
 networkName: sriov-data-network
 volumes:
 - name: rootdisk
 dataVolume:
 name: multi-interface-vm-root

###### High-Performance NAD Configurations

### High-performance management network
apiVersion: k8s.cni.cncf.io/v1
kind: NetworkAttachmentDefinition
metadata:
 name: management-network
 namespace: high-performance-workloads
spec:
 config: |
 {
 "cniVersion": "0.3.1",
 "name": "management-network",
 "type": "macvlan",
 "master": "ens192",
 "mode": "bridge",
 "capabilities": {
 "ips": true
 },
 "ipam": {
 "type": "static"
 }
 }
---
### Dedicated storage network with optimized MTU
apiVersion: k8s.cni.cncf.io/v1
kind: NetworkAttachmentDefinition
metadata:
 name: storage-network
 namespace: high-performance-workloads
spec:
 config: |
 {
 "cniVersion": "0.3.1",
 "name": "storage-network",
 "type": "macvlan",
 "master": "ens224",
 "mode": "bridge",
 "mtu": 9000,
 "capabilities": {
 "ips": true
 },
 "ipam": {
 "type": "static"
 }
 }
---
### SR-IOV high-performance data network
apiVersion: k8s.cni.cncf.io/v1
kind: NetworkAttachmentDefinition
metadata:
 name: sriov-data-network
 namespace: high-performance-workloads
spec:
 config: |
 {
 "cniVersion": "0.3.1",
 "name": "sriov-data-network",
 "type": "sriov",
 "deviceID": "1017",
 "vf": 0,
 "spoofchk": "off",
 "trust": "on",
 "capabilities": {
 "ips": true
 },
 "ipam": {
 "type": "static"
 }
 }

###### Bond Network for High Availability

apiVersion: k8s.cni.cncf.io/v1
kind: NetworkAttachmentDefinition
metadata:
 name: bond-ha-network
 namespace: high-performance-workloads
spec:
 config: |
 {
 "cniVersion": "0.3.1",
 "name": "bond-ha-network",
 "type": "bond",
 "mode": "802.3ad",
 "miimon": "100",
 "updelay": "200",
 "downdelay": "200",
 "links": [
 {
 "name": "ens256"
 },
 {
 "name": "ens257"
 }
 ],
 "ipam": {
 "type": "static"
 }
 }

#### Cluster Performance Optimization

##### Node-Level Optimizations

###### Performance Profile for Worker Nodes

apiVersion: performance.openshift.io/v2
kind: PerformanceProfile
metadata:
 name: high-performance-worker
spec:
 cpu:
 isolated: "2-47"
 reserved: "0-1"
 hugepages:
 defaultHugepagesSize: 1G
 pages:
 - count: 16
 size: 1G
 nodeSelector:
 node-role.kubernetes.io/worker-rt: ""
 realTimeKernel:
 enabled: true
 numa:
 topologyPolicy: "single-numa-node"

###### Machine Config for Kernel Tuning

apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
 name: performance-tuning
 labels:
 machineconfiguration.openshift.io/role: worker
spec:
 config:
 ignition:
 version: 3.2.0
 storage:
 files:
 - path: /etc/sysctl.d/99-performance.conf
 mode: 0644
 contents:
 inline: |
 # Network performance tuning
 net.core.rmem\_max = 268435456
 net.core.wmem\_max = 268435456
 net.ipv4.tcp\_rmem = 4096 131072 268435456
 net.ipv4.tcp\_wmem = 4096 65536 268435456

 # Virtual memory tuning
 vm.swappiness = 1
 vm.dirty\_ratio = 15
 vm.dirty\_background\_ratio = 5

 # CPU scheduler tuning
 kernel.sched\_migration\_cost\_ns = 5000000

##### Resource Management Optimization

###### Cluster Resource Allocation

apiVersion: v1
kind: ResourceQuota
metadata:
 name: performance-quota
 namespace: high-performance-workloads
spec:
 hard:
 requests.cpu: "100"
 requests.memory: 200Gi
 limits.cpu: "200"
 limits.memory: 400Gi
 hugepages-1Gi: 64Gi
 persistentvolumeclaims: "50"

###### Priority Classes for Critical Workloads

apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
 name: high-performance-priority
value: 1000
globalDefault: false
description: "Priority class for high-performance VMs"
---
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
 name: critical-vm
spec:
 template:
 spec:
 priorityClassName: high-performance-priority
 domain:
 cpu:
 cores: 8
 memory:
 guest: 16Gi

#### Monitoring Performance Optimizations

##### Efficient Metrics Collection

apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
 name: performance-metrics
spec:
 selector:
 matchLabels:
 app: high-performance-app
 endpoints:
 - port: metrics
 interval: 15s # Reduced interval for better granularity
 scrapeTimeout: 10s
 path: /metrics
 metricRelabelings:
 - sourceLabels: [\_\_name\_\_]
 regex: 'go\_.\*|process\_.\*'
 action: drop # Drop unnecessary metrics

##### Performance Dashboard Configuration

apiVersion: v1
kind: ConfigMap
metadata:
 name: performance-dashboard
data:
 dashboard.json: |
 {
 "dashboard": {
 "title": "RH OVE Performance Dashboard",
 "panels": [
 {
 "title": "VM CPU Usage",
 "type": "graph",
 "targets": [
 {
 "expr": "rate(kubevirt\_vm\_cpu\_usage\_seconds\_total[5m]) \* 100",
 "legendFormat": "{{name}} CPU %"
 }
 ]
 },
 {
 "title": "VM Memory Usage",
 "type": "graph",
 "targets": [
 {
 "expr": "kubevirt\_vm\_memory\_usage\_bytes / kubevirt\_vm\_memory\_available\_bytes \* 100",
 "legendFormat": "{{name}} Memory %"
 }
 ]
 }
 ]
 }
 }

#### Performance Testing and Benchmarking

##### VM Performance Testing

### !/bin/bash
### VM Performance Test Script

VM\_NAME="performance-test-vm"
NAMESPACE="testing"

### CPU Performance Test
virtctl console $VM\_NAME << EOF
### Install and run CPU benchmark
yum install -y stress-ng
stress-ng --cpu 0 --timeout 60s --metrics-brief
EOF

### Memory Performance Test
virtctl console $VM\_NAME << EOF
### Memory bandwidth test
stress-ng --vm 1 --vm-bytes 4G --timeout 60s --metrics-brief
EOF

### Disk I/O Performance Test
virtctl console $VM\_NAME << EOF
### Disk performance test
dd if=/dev/zero of=/tmp/testfile bs=1G count=1 oflag=direct
dd if=/tmp/testfile of=/dev/null bs=1G count=1 iflag=direct
rm /tmp/testfile
EOF

##### Network Performance Testing

apiVersion: v1
kind: Pod
metadata:
 name: network-performance-test
spec:
 containers:
 - name: iperf-server
 image: networkstatic/iperf3
 command: ['iperf3', '-s']
 ports:
 - containerPort: 5201
 - name: iperf-client
 image: networkstatic/iperf3
 command: ['sleep', '3600']

#### Performance Troubleshooting

##### Common Performance Issues

###### High CPU Usage

### Identify CPU-intensive processes
oc adm top pods --all-namespaces --sort-by=cpu

### Check node CPU utilization
oc adm top nodes

### Analyze CPU usage patterns
virtctl console <vm-name>
top -p 1

###### Memory Pressure

### Check memory usage
oc adm top pods --all-namespaces --sort-by=memory

### Verify hugepages allocation
oc get nodes -o custom-columns=NAME:.metadata.name,HUGEPAGES:.status.allocatable.hugepages-1Gi

### Check for memory leaks in VM
virtctl console <vm-name>
free -h
cat /proc/meminfo

###### Storage Performance Issues

### Check storage performance metrics
oc get pvc
oc describe pvc <pvc-name>

### Monitor I/O patterns
virtctl console <vm-name>
iostat -x 1

### Check storage backend performance
oc get nodes -o wide

#### Best Practices Summary

##### VM Performance Best Practices

1. **CPU Optimization**
	* Use CPU pinning for latency-sensitive workloads
	* Enable NUMA topology awareness
	* Configure appropriate CPU limits and requests
2. **Memory Optimization**
	* Use hugepages for memory-intensive applications
	* Configure appropriate memory ballooning
	* Monitor memory usage patterns
3. **Storage Optimization**
	* Use high-performance storage classes for critical workloads
	* Optimize disk cache settings
	* Consider using block storage for high I/O workloads
4. **Network Optimization**
	* Use SR-IOV for high-bandwidth applications
	* Optimize Cilium configuration for performance
	* Consider DPDK for packet processing workloads

##### Monitoring and Maintenance

1. **Regular Performance Reviews**
	* Monitor KPIs continuously
	* Perform regular performance testing
	* Document performance baselines
2. **Capacity Planning**
	* Plan for growth and scaling
	* Monitor resource utilization trends
	* Implement proper resource quotas
3. **Optimization Cycles**
	* Regular performance tuning reviews
	* Test optimizations in non-production environments
	* Document all performance changes

This performance tuning guide provides comprehensive strategies for optimizing the RH OVE ecosystem. Regular application of these practices ensures optimal performance for virtualized workloads while maintaining system stability and reliability.