RH OVE Management Documentation

Red Hat OpenShift Virtualization Ecosystem Team

2025-08-04

Table of Contents

# Management

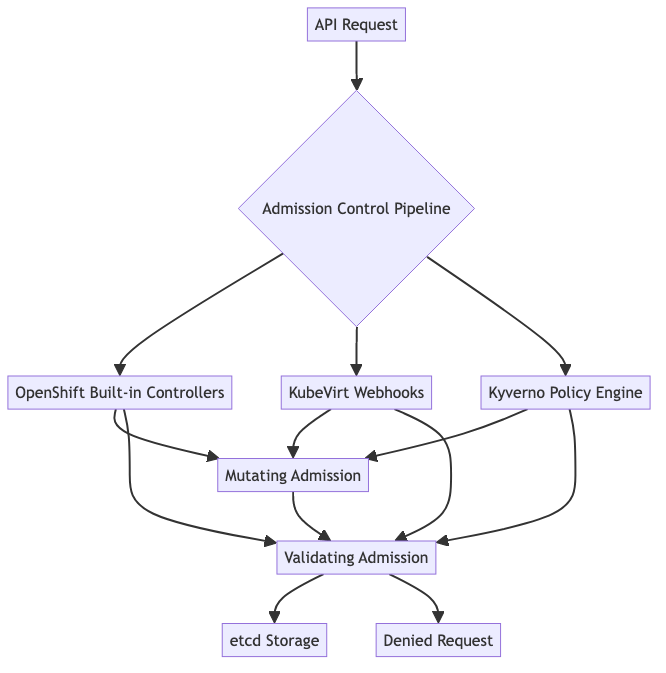
## Admission Control

### Admission Control Strategy

#### Overview

The RH OVE solution implements a comprehensive admission control strategy that combines OpenShift’s built-in controllers with KubeVirt-specific webhooks and Kyverno policy engine for enhanced security and governance.

#### Architecture



#### Default OpenShift Admission Controllers

RH OVE inherits all standard OpenShift admission controllers including:

* **LimitRanger**: Enforces resource limits and requests
* **ServiceAccount**: Manages service account tokens
* **PodNodeSelector**: Controls node placement
* **SecurityContextConstraint (SCC)**: Enforces security policies
* **ResourceQuota**: Manages namespace-level resource limits
* **MutatingAdmissionWebhook**: Allows custom mutation logic
* **ValidatingAdmissionWebhook**: Allows custom validation logic

#### KubeVirt-Specific Admission Webhooks

KubeVirt automatically registers webhooks for virtualization resources:

* **VirtualMachine validation**: Ensures VM configurations are valid
* **VirtualMachineInstance validation**: Validates running VM instances
* **DataVolume validation**: Verifies storage configurations
* **Migration validation**: Checks migration prerequisites

#### Kyverno Policy Engine Integration

##### Installation

Deploy Kyverno via Helm or manifests:

apiVersion: v1  
kind: Namespace  
metadata:  
 name: kyverno  
---  
apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: kyverno  
 namespace: kyverno  
spec:  
 replicas: 1  
 selector:  
 matchLabels:  
 app: kyverno  
 template:  
 metadata:  
 labels:  
 app: kyverno  
 spec:  
 containers:  
 - name: kyverno  
 image: ghcr.io/kyverno/kyverno:latest

##### VM-Specific Policies

###### Resource Limits Policy

apiVersion: kyverno.io/v1  
kind: ClusterPolicy  
metadata:  
 name: vm-resource-limits  
spec:  
 validationFailureAction: enforce  
 background: true  
 rules:  
 - name: require-vm-resource-limits  
 match:  
 any:  
 - resources:  
 kinds:  
 - VirtualMachine  
 validate:  
 message: "VirtualMachine must have CPU and memory limits defined"  
 pattern:  
 spec:  
 template:  
 spec:  
 domain:  
 cpu:  
 cores: ">0"  
 memory:  
 guest: ">0"

###### VM Security Policy

apiVersion: kyverno.io/v1  
kind: ClusterPolicy  
metadata:  
 name: vm-security-policy  
spec:  
 validationFailureAction: enforce  
 rules:  
 - name: disallow-privileged-vms  
 match:  
 any:  
 - resources:  
 kinds:  
 - VirtualMachine  
 validate:  
 message: "Privileged VMs are not allowed"  
 pattern:  
 spec:  
 template:  
 spec:  
 domain:  
 features:  
 smm:  
 enabled: "false"

###### Namespace Isolation Policy

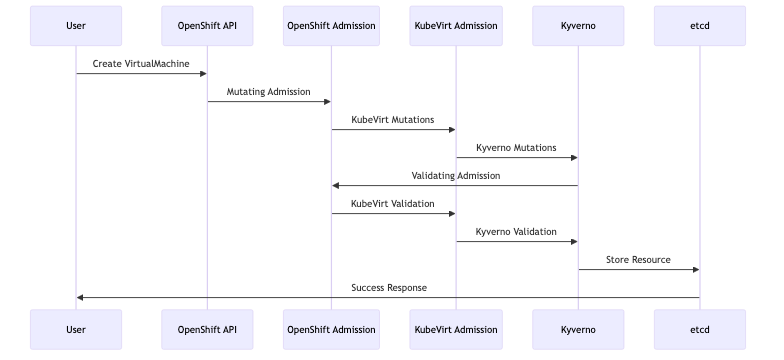
apiVersion: kyverno.io/v1  
kind: ClusterPolicy  
metadata:  
 name: vm-namespace-isolation  
spec:  
 validationFailureAction: enforce  
 rules:  
 - name: require-namespace-labels  
 match:  
 any:  
 - resources:  
 kinds:  
 - VirtualMachine  
 validate:  
 message: "VirtualMachines must be in properly labeled namespaces"  
 pattern:  
 metadata:  
 namespace: "!default"

##### Multi-Tenant Policies

###### Application-Based Namespace Policy

apiVersion: kyverno.io/v1  
kind: ClusterPolicy  
metadata:  
 name: application-namespace-policy  
spec:  
 validationFailureAction: enforce  
 rules:  
 - name: enforce-app-namespace-pattern  
 match:  
 any:  
 - resources:  
 kinds:  
 - VirtualMachine  
 - VirtualMachineInstance  
 validate:  
 message: "VMs must be deployed in application-specific namespaces"  
 pattern:  
 metadata:  
 namespace: "app-\*"

#### Policy Enforcement Flow



#### Best Practices

##### Policy Development

1. **Start with monitoring mode**: Use validationFailureAction: audit initially
2. **Test thoroughly**: Validate policies in non-production environments
3. **Use exceptions sparingly**: Avoid broad policy exceptions
4. **Version control**: Store policies in Git repositories

##### Performance Considerations

1. **Optimize policy matching**: Use specific resource selectors
2. **Minimize policy overlap**: Avoid redundant validation rules
3. **Monitor admission latency**: Track policy evaluation performance
4. **Use background processing**: Enable background validation where appropriate

##### Monitoring and Troubleshooting

###### Policy Violations Dashboard

Monitor policy violations using Prometheus metrics:

apiVersion: v1  
kind: ServiceMonitor  
metadata:  
 name: kyverno-metrics  
spec:  
 selector:  
 matchLabels:  
 app: kyverno  
 endpoints:  
 - port: metrics

###### Common Troubleshooting

1. **Policy not applying**: Check policy syntax and matching criteria
2. **Performance issues**: Review policy complexity and scope
3. **Conflicts**: Examine interaction between different admission controllers
4. **Debugging**: Use kubectl describe to view admission controller events

#### Integration with GitOps

Store all policies in Git and deploy via Argo CD:

apiVersion: argoproj.io/v1alpha1  
kind: Application  
metadata:  
 name: kyverno-policies  
spec:  
 source:  
 repoURL: https://git.example.com/rh-ove-policies  
 path: policies/  
 targetRevision: main  
 destination:  
 server: https://kubernetes.default.svc  
 syncPolicy:  
 automated:  
 prune: true  
 selfHeal: true

This comprehensive admission control strategy ensures that all workloads, both containers and VMs, comply with organizational security and operational policies while maintaining the flexibility needed for modern application development.

## Gitops

### GitOps Operations

#### Overview

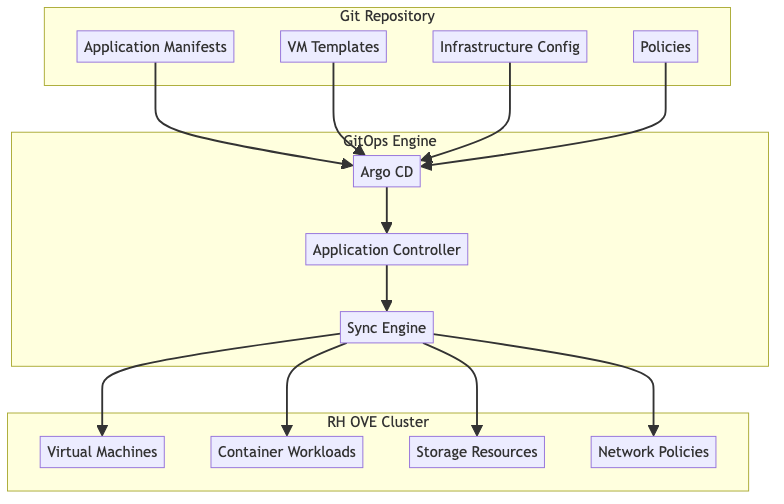
This document outlines the GitOps approach for managing the multi-cluster RH OVE ecosystem, providing guidelines for implementing Infrastructure as Code (IaC) and application deployment through Git-based workflows across management and application clusters.

#### Multi-Cluster GitOps Principles

The RH OVE ecosystem implements GitOps across a hub-and-spoke architecture:

* **Centralized Control**: ArgoCD Hub manages deployments to multiple application clusters
* **Single Source of Truth**: All cluster and VM configurations stored in Git repositories
* **Declarative Management**: Infrastructure as Code for clusters, VMs, and containers
* **Multi-Environment Support**: Separate overlays for production, staging, and development clusters
* **Policy Distribution**: Centralized policy management with cluster-specific enforcement
* **Automated Rollbacks**: Complete change tracking and rollback capabilities across clusters

#### Multi-Cluster Architecture



#### Repository Structure

##### Recommended Git Repository Layout

rh-ove-gitops/  
├── applications/  
│ ├── web-app/  
│ │ ├── base/  
│ │ │ ├── kustomization.yaml  
│ │ │ ├── deployment.yaml  
│ │ │ └── service.yaml  
│ │ └── overlays/  
│ │ ├── dev/  
│ │ ├── staging/  
│ │ └── prod/  
│ └── database-vm/  
│ ├── vm-definition.yaml  
│ ├── datavolume.yaml  
│ └── service.yaml  
├── infrastructure/  
│ ├── storage-classes/  
│ ├── network-policies/  
│ └── rbac/  
├── vm-templates/  
│ ├── rhel8-template.yaml  
│ ├── windows-template.yaml  
│ └── ubuntu-template.yaml  
└── policies/  
 ├── kyverno/  
 └── gatekeeper/

#### Argo CD Configuration

##### Application Definition

apiVersion: argoproj.io/v1alpha1  
kind: Application  
metadata:  
 name: database-vm-app  
 namespace: argocd  
spec:  
 project: default  
 source:  
 repoURL: https://git.example.com/rh-ove-gitops  
 targetRevision: main  
 path: applications/database-vm  
 destination:  
 server: https://kubernetes.default.svc  
 namespace: app-database-prod  
 syncPolicy:  
 automated:  
 prune: true  
 selfHeal: true  
 syncOptions:  
 - CreateNamespace=true

##### AppProject for VM Workloads

apiVersion: argoproj.io/v1alpha1  
kind: AppProject  
metadata:  
 name: vm-workloads  
 namespace: argocd  
spec:  
 description: Project for VM-based applications  
 sourceRepos:  
 - https://git.example.com/rh-ove-gitops  
 destinations:  
 - namespace: 'app-\*'  
 server: https://kubernetes.default.svc  
 clusterResourceWhitelist:  
 - group: ''  
 kind: Namespace  
 - group: 'kubevirt.io'  
 kind: VirtualMachine  
 - group: 'cdi.kubevirt.io'  
 kind: DataVolume  
 namespaceResourceWhitelist:  
 - group: ''  
 kind: Service  
 - group: ''  
 kind: ConfigMap  
 - group: ''  
 kind: Secret

#### VM Management through GitOps

##### Virtual Machine Definition

apiVersion: kubevirt.io/v1  
kind: VirtualMachine  
metadata:  
 name: web-server-vm  
 namespace: app-web-prod  
 labels:  
 app: web-server  
 managed-by: argocd  
spec:  
 running: true  
 template:  
 metadata:  
 labels:  
 app: web-server  
 version: v1.2.0  
 spec:  
 domain:  
 cpu:  
 cores: 2  
 memory:  
 guest: 4Gi  
 devices:  
 disks:  
 - name: rootdisk  
 disk:  
 bus: virtio  
 - name: datadisk  
 disk:  
 bus: virtio  
 interfaces:  
 - name: default  
 masquerade: {}  
 networks:  
 - name: default  
 pod: {}  
 volumes:  
 - name: rootdisk  
 dataVolume:  
 name: web-server-root  
 - name: datadisk  
 dataVolume:  
 name: web-server-data

##### DataVolume Configuration

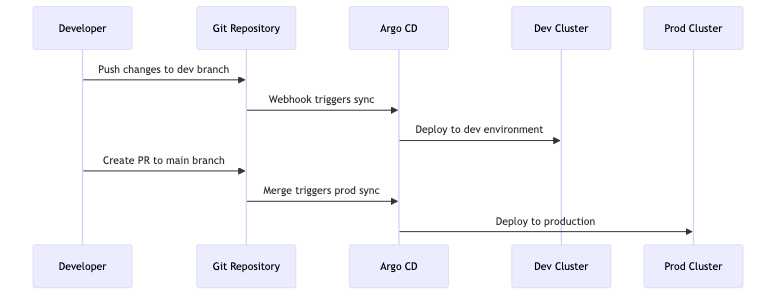
apiVersion: cdi.kubevirt.io/v1beta1  
kind: DataVolume  
metadata:  
 name: web-server-root  
 namespace: app-web-prod  
spec:  
 pvc:  
 accessModes:  
 - ReadWriteOnce  
 resources:  
 requests:  
 storage: 30Gi  
 storageClassName: fast-ssd  
 source:  
 registry:  
 url: "docker://registry.redhat.io/rhel8/rhel:latest"  
---  
apiVersion: cdi.kubevirt.io/v1beta1  
kind: DataVolume  
metadata:  
 name: web-server-data  
 namespace: app-web-prod  
spec:  
 pvc:  
 accessModes:  
 - ReadWriteOnce  
 resources:  
 requests:  
 storage: 100Gi  
 storageClassName: standard-hdd  
 source:  
 blank: {}

#### Multi-Environment Management

##### Environment-Specific Overlays

### overlays/prod/kustomization.yaml  
apiVersion: kustomize.config.k8s.io/v1beta1  
kind: Kustomization  
  
resources:  
- ../../base  
  
patchesStrategicMerge:  
- vm-resources.yaml  
- storage-config.yaml  
  
patches:  
- target:  
 kind: VirtualMachine  
 name: web-server-vm  
 patch: |-  
 - op: replace  
 path: /spec/template/spec/domain/cpu/cores  
 value: 4  
 - op: replace  
 path: /spec/template/spec/domain/memory/guest  
 value: 8Gi

##### Environment Promotion Workflow



#### CI/CD Integration

##### GitLab CI Pipeline

### .gitlab-ci.yml  
stages:  
 - validate  
 - test  
 - deploy  
  
validate-manifests:  
 stage: validate  
 script:  
 - kubeval manifests/\*.yaml  
 - kustomize build overlays/dev | kubeval  
  
vm-integration-test:  
 stage: test  
 script:  
 - kubectl apply --dry-run=client -f vm-definitions/  
 - virtctl validate vm-definitions/  
  
deploy-to-dev:  
 stage: deploy  
 script:  
 - argocd app sync dev-environment  
 only:  
 - develop  
  
deploy-to-prod:  
 stage: deploy  
 script:  
 - argocd app sync prod-environment  
 only:  
 - main

#### Security and Compliance

##### Policy as Code

### policies/kyverno/vm-security-policy.yaml  
apiVersion: kyverno.io/v1  
kind: ClusterPolicy  
metadata:  
 name: vm-security-standards  
spec:  
 validationFailureAction: enforce  
 background: true  
 rules:  
 - name: require-vm-labels  
 match:  
 any:  
 - resources:  
 kinds:  
 - VirtualMachine  
 validate:  
 message: "VMs must have required labels"  
 pattern:  
 metadata:  
 labels:  
 app: "?\*"  
 version: "?\*"  
 managed-by: "argocd"

##### RBAC for GitOps

apiVersion: rbac.authorization.k8s.io/v1  
kind: ClusterRole  
metadata:  
 name: argocd-application-controller  
rules:  
- apiGroups:  
 - ""  
 resources:  
 - "\*"  
 verbs:  
 - "\*"  
- apiGroups:  
 - "kubevirt.io"  
 resources:  
 - "\*"  
 verbs:  
 - "\*"  
- apiGroups:  
 - "cdi.kubevirt.io"  
 resources:  
 - "\*"  
 verbs:  
 - "\*"

#### Monitoring and Observability

##### GitOps Metrics

Key metrics to monitor: - Sync success rate - Deployment frequency - Mean time to recovery - Application health status

##### Dashboard Configuration

apiVersion: v1  
kind: ConfigMap  
metadata:  
 name: argocd-cm  
data:  
 url: https://argocd.example.com  
 statusbadge.enabled: "true"  
 application.instanceLabelKey: argocd.argoproj.io/instance

#### Disaster Recovery

##### Backup Strategy

apiVersion: v1  
kind: ConfigMap  
metadata:  
 name: backup-config  
data:  
 backup-strategy.yaml: |  
 applications:  
 - name: critical-vms  
 backup\_frequency: "daily"  
 retention: "30d"  
 git\_ref: "backup/$(date +%Y%m%d)"  
   
 infrastructure:  
 - name: cluster-config  
 backup\_frequency: "weekly"  
 retention: "12w"

##### Recovery Procedures

1. **Application Recovery**

* # Restore from specific commit  
  argocd app set myapp --revision abc123  
  argocd app sync myapp

1. **Full Environment Recovery**

* # Deploy entire environment from Git  
  argocd app create disaster-recovery \  
   --repo https://git.example.com/rh-ove-gitops \  
   --path recovery/full-environment \  
   --dest-server https://kubernetes.default.svc

#### Best Practices

##### Development Workflow

1. **Feature branches**: Use feature branches for new VM deployments
2. **Pull requests**: Require peer review for all changes
3. **Automated testing**: Validate manifests before merge
4. **Progressive deployment**: Use staging environments before production

##### Operational Guidelines

1. **Secrets management**: Use external secret management (e.g., Vault)
2. **Resource limits**: Define appropriate CPU/memory limits for VMs
3. **Monitoring**: Implement comprehensive monitoring for all deployments
4. **Documentation**: Keep README files updated in each application directory

This GitOps approach ensures consistent, auditable, and automated management of VM and container workloads in the RH OVE environment.

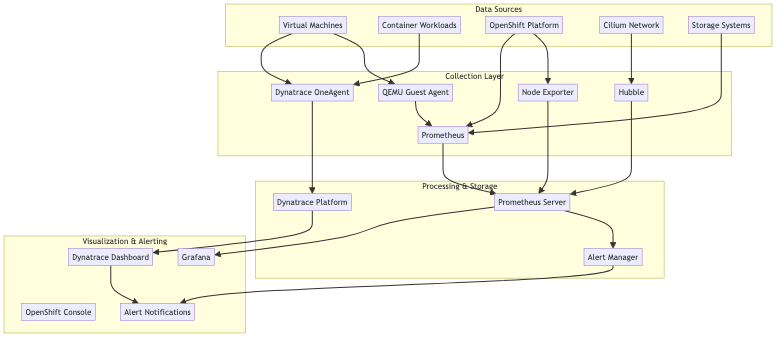
## Monitoring

### Monitoring and Observability

#### Overview

This document provides comprehensive monitoring and observability strategies for the RH OVE ecosystem, covering infrastructure, virtual machines, containers, and application performance monitoring using Dynatrace and other monitoring tools.

#### Monitoring Architecture



#### Dynatrace Integration

Based on our research, integrating RH OVE monitoring stack with Dynatrace provides comprehensive visibility for VMs and Kubernetes workloads.

##### Dynatrace Operator Installation

apiVersion: dynatrace.com/v1beta1  
kind: DynaKube  
metadata:  
 name: dynakube  
 namespace: dynatrace  
spec:  
 apiUrl: https://your-environment-id.live.dynatrace.com/api  
 oneAgent:  
 classicFullStack:  
 tolerations:  
 - key: node-role.kubernetes.io/master  
 operator: Exists  
 effect: NoSchedule  
 resources:  
 requests:  
 cpu: 100m  
 memory: 512Mi  
 limits:  
 cpu: 300m  
 memory: 1Gi  
 activeGate:  
 capabilities:  
 - kubernetes-monitoring  
 - routing  
 resources:  
 requests:  
 cpu: 150m  
 memory: 512Mi  
 limits:  
 cpu: 500m  
 memory: 1Gi

##### VM-Specific Monitoring Configuration

apiVersion: kubevirt.io/v1  
kind: VirtualMachine  
metadata:  
 name: monitored-vm  
 namespace: app-prod  
 annotations:  
 dynatrace.com/inject: "true"  
 dynatrace.com/vm-monitoring: "enabled"  
spec:  
 template:  
 metadata:  
 labels:  
 app: web-server  
 monitoring: enabled  
 spec:  
 domain:  
 devices:  
 interfaces:  
 - name: default  
 masquerade: {}  
 resources:  
 requests:  
 memory: 4Gi  
 cpu: 2  
 volumes:  
 - name: qemu-guest-agent  
 serviceAccount:  
 serviceAccountName: qemu-guest-agent

#### Prometheus Configuration

##### ServiceMonitor for VM Metrics

apiVersion: monitoring.coreos.com/v1  
kind: ServiceMonitor  
metadata:  
 name: vm-metrics  
 namespace: monitoring  
spec:  
 selector:  
 matchLabels:  
 app: kubevirt-prometheus-metrics  
 endpoints:  
 - port: metrics  
 interval: 30s  
 path: /metrics

##### Custom Metrics for VMs

apiVersion: monitoring.coreos.com/v1  
kind: PrometheusRule  
metadata:  
 name: vm-monitoring-rules  
 namespace: monitoring  
spec:  
 groups:  
 - name: vm.rules  
 rules:  
 - alert: VMHighCPUUsage  
 expr: kubevirt\_vm\_cpu\_usage\_percentage > 80  
 for: 5m  
 labels:  
 severity: warning  
 annotations:  
 summary: "VM {{ $labels.name }} has high CPU usage"  
 description: "VM {{ $labels.name }} in namespace {{ $labels.namespace }} has CPU usage above 80% for more than 5 minutes."  
   
 - alert: VMHighMemoryUsage  
 expr: kubevirt\_vm\_memory\_usage\_percentage > 85  
 for: 5m  
 labels:  
 severity: warning  
 annotations:  
 summary: "VM {{ $labels.name }} has high memory usage"  
 description: "VM {{ $labels.name }} in namespace {{ $labels.namespace }} has memory usage above 85% for more than 5 minutes."

#### Network Monitoring with Hubble

##### Hubble Configuration

apiVersion: v1  
kind: ConfigMap  
metadata:  
 name: cilium-config  
 namespace: kube-system  
data:  
 enable-hubble: "true"  
 hubble-listen-address: ":4244"  
 hubble-metrics-server: ":9091"  
 hubble-metrics: |  
 dns:query;ignoreAAAA  
 drop  
 tcp  
 flow  
 icmp  
 http

##### Network Flow Monitoring

apiVersion: monitoring.coreos.com/v1  
kind: ServiceMonitor  
metadata:  
 name: hubble-metrics  
spec:  
 selector:  
 matchLabels:  
 k8s-app: hubble  
 endpoints:  
 - port: hubble-metrics  
 interval: 30s

#### Storage Monitoring

##### CDI and Storage Metrics

apiVersion: monitoring.coreos.com/v1  
kind: ServiceMonitor  
metadata:  
 name: cdi-controller-metrics  
spec:  
 selector:  
 matchLabels:  
 app: cdi-controller  
 endpoints:  
 - port: metrics  
 interval: 30s  
 path: /metrics

##### Storage Performance Alerts

apiVersion: monitoring.coreos.com/v1  
kind: PrometheusRule  
metadata:  
 name: storage-monitoring-rules  
spec:  
 groups:  
 - name: storage.rules  
 rules:  
 - alert: HighStorageLatency  
 expr: kubelet\_volume\_stats\_available\_bytes / kubelet\_volume\_stats\_capacity\_bytes < 0.1  
 for: 5m  
 labels:  
 severity: critical  
 annotations:  
 summary: "Storage volume {{ $labels.persistentvolumeclaim }} is running out of space"  
   
 - alert: DataVolumeImportFailed  
 expr: increase(cdi\_import\_progress\_total{phase="Failed"}[5m]) > 0  
 labels:  
 severity: warning  
 annotations:  
 summary: "DataVolume import failed"

#### Application Performance Monitoring

##### Guest Agent Installation

For enhanced VM monitoring, install QEMU Guest Agent:

### Inside RHEL/CentOS VM  
sudo yum install qemu-guest-agent  
sudo systemctl enable qemu-guest-agent  
sudo systemctl start qemu-guest-agent  
  
### Inside Ubuntu VM  
sudo apt-get install qemu-guest-agent  
sudo systemctl enable qemu-guest-agent  
sudo systemctl start qemu-guest-agent  
  
### Inside Windows VM  
### Download and install virtio-win guest tools

##### Node Exporter for VM Guests

apiVersion: apps/v1  
kind: DaemonSet  
metadata:  
 name: node-exporter-vm  
spec:  
 selector:  
 matchLabels:  
 app: node-exporter-vm  
 template:  
 metadata:  
 labels:  
 app: node-exporter-vm  
 spec:  
 containers:  
 - name: node-exporter  
 image: prom/node-exporter:latest  
 ports:  
 - containerPort: 9100  
 volumeMounts:  
 - name: proc  
 mountPath: /host/proc  
 readOnly: true  
 - name: sys  
 mountPath: /host/sys  
 readOnly: true  
 volumes:  
 - name: proc  
 hostPath:  
 path: /proc  
 - name: sys  
 hostPath:  
 path: /sys

#### Dashboard Configuration

##### Grafana Dashboard for VMs

{  
 "dashboard": {  
 "title": "RH OVE Virtual Machine Monitoring",  
 "panels": [  
 {  
 "title": "VM CPU Usage",  
 "type": "graph",  
 "targets": [  
 {  
 "expr": "kubevirt\_vm\_cpu\_usage\_percentage",  
 "legendFormat": "{{name}}"  
 }  
 ]  
 },  
 {  
 "title": "VM Memory Usage",  
 "type": "graph",  
 "targets": [  
 {  
 "expr": "kubevirt\_vm\_memory\_usage\_percentage",  
 "legendFormat": "{{name}}"  
 }  
 ]  
 },  
 {  
 "title": "VM Network I/O",  
 "type": "graph",  
 "targets": [  
 {  
 "expr": "rate(kubevirt\_vm\_network\_receive\_bytes\_total[5m])",  
 "legendFormat": "{{name}} - RX"  
 },  
 {  
 "expr": "rate(kubevirt\_vm\_network\_transmit\_bytes\_total[5m])",  
 "legendFormat": "{{name}} - TX"  
 }  
 ]  
 }  
 ]  
 }  
}

##### Dynatrace Dashboard Configuration

apiVersion: v1  
kind: ConfigMap  
metadata:  
 name: dynatrace-dashboard-config  
data:  
 vm-overview.json: |  
 {  
 "dashboardMetadata": {  
 "name": "RH OVE VM Overview",  
 "shared": true,  
 "tags": ["rh-ove", "virtualization"]  
 },  
 "tiles": [  
 {  
 "name": "VM Performance",  
 "tileType": "CUSTOM\_CHARTING",  
 "configured": true,  
 "queries": [  
 {  
 "metric": "builtin:host.cpu.usage",  
 "aggregation": {  
 "type": "AVG"  
 },  
 "filterBy": {  
 "neType": "HOST",  
 "tags": ["vm:kubevirt"]  
 }  
 }  
 ]  
 }  
 ]  
 }

#### Alerting Strategy

##### Alert Routing Configuration

apiVersion: v1  
kind: ConfigMap  
metadata:  
 name: alertmanager-config  
data:  
 alertmanager.yml: |  
 global:  
 smtp\_smarthost: 'smtp.example.com:587'  
 smtp\_from: 'alerts@example.com'  
   
 route:  
 group\_by: ['alertname', 'cluster', 'service']  
 group\_wait: 10s  
 group\_interval: 10s  
 repeat\_interval: 1h  
 receiver: 'web.hook'  
 routes:  
 - match:  
 severity: critical  
 receiver: 'critical-alerts'  
 - match:  
 service: vm  
 receiver: 'vm-alerts'  
   
 receivers:  
 - name: 'web.hook'  
 webhook\_configs:  
 - url: 'http://webhook.example.com/alerts'  
   
 - name: 'critical-alerts'  
 email\_configs:  
 - to: 'oncall@example.com'  
 subject: 'CRITICAL: {{ .GroupLabels.alertname }}'  
 body: |  
 {{ range .Alerts }}  
 Alert: {{ .Annotations.summary }}  
 Description: {{ .Annotations.description }}  
 {{ end }}  
   
 - name: 'vm-alerts'  
 slack\_configs:  
 - api\_url: 'https://hooks.slack.com/services/...'  
 channel: '#vm-alerts'  
 title: 'VM Alert: {{ .GroupLabels.alertname }}'

#### Logging Strategy

##### Centralized Logging for VMs

apiVersion: logging.coreos.com/v1  
kind: ClusterLogForwarder  
metadata:  
 name: vm-logs  
 namespace: openshift-logging  
spec:  
 outputs:  
 - name: vm-logs-output  
 type: elasticsearch  
 url: https://elasticsearch.example.com:9200  
 elasticsearch:  
 index: vm-logs-{.log\_type}-{.@timestamp.YYYY.MM.dd}  
 pipelines:  
 - name: vm-logs-pipeline  
 inputRefs:  
 - application  
 filterRefs:  
 - vm-log-filter  
 outputRefs:  
 - vm-logs-output

#### Performance Optimization

##### Monitoring Resource Optimization

apiVersion: v1  
kind: ResourceQuota  
metadata:  
 name: monitoring-quota  
 namespace: monitoring  
spec:  
 hard:  
 requests.cpu: "2"  
 requests.memory: 4Gi  
 limits.cpu: "4"  
 limits.memory: 8Gi  
 persistentvolumeclaims: "5"

##### Metrics Retention Policy

apiVersion: v1  
kind: ConfigMap  
metadata:  
 name: prometheus-config  
data:  
 prometheus.yml: |  
 global:  
 scrape\_interval: 30s  
 evaluation\_interval: 30s  
 external\_labels:  
 cluster: 'rh-ove-cluster'  
   
 rule\_files:  
 - "vm-monitoring-rules.yml"  
   
 scrape\_configs:  
 - job\_name: 'kubevirt-vms'  
 kubernetes\_sd\_configs:  
 - role: pod  
 relabel\_configs:  
 - source\_labels: [\_\_meta\_kubernetes\_pod\_label\_kubevirt\_io]  
 target\_label: vm\_name

#### Troubleshooting Monitoring

##### Common Issues and Solutions

1. **OneAgent not reporting VM data**

* # Check OneAgent status  
  oc get pods -n dynatrace  
  oc describe pod dynatrace-oneagent-xxx  
    
  # Verify VM annotations  
  oc get vm -o yaml | grep -A5 annotations

1. **Missing VM metrics in Prometheus**

* # Check ServiceMonitor configuration  
  oc get servicemonitor -n monitoring  
    
  # Verify metrics endpoint  
  oc port-forward svc/kubevirt-prometheus-metrics 8080:8080  
  curl localhost:8080/metrics

1. **Network flow data not appearing**

* # Check Hubble status  
  cilium status  
  hubble status  
    
  # Verify Hubble configuration  
  oc get configmap cilium-config -n kube-system -o yaml

This comprehensive monitoring strategy ensures full visibility into the RH OVE ecosystem, covering infrastructure, virtual machines, containers, and application performance.

## Backup

### Backup & Recovery

#### Overview

This document outlines the backup and recovery strategies for the RH OVE ecosystem. It highlights the integration with Rubrik, detailing how to efficiently back up and restore both VM and container data.

#### Backup Strategy

##### Rubrik Integration

Utilize Rubrik’s capabilities to ensure robust data protection:

* **Certified Integration**: Rubrik is certified for RH OVE, providing seamless data protection.
* **Immutable Backups**: Ensure data safety with air-gapped, tamper-proof backups.
* **Policy-Driven**: Simplify backup management with declarative policies for VM workloads.

##### Backup Configuration

apiVersion: v1  
kind: ConfigMap  
metadata:  
 name: rubrik-backup-config  
 namespace: rubrik-system  
data:  
 backupPolicy.yaml: |  
 policies:  
 - name: daily-VM-backup  
 frequency: daily  
 retention: 30d  
 exclude: 'temp-volumes'  
  
 schedules:  
 - name: nightly-backup  
 time: '02:00'  
 days:  
 - Monday  
 - Wednesday  
 - Friday

##### Data Volume Backup

Backup specific DataVolumes using Rubrik advanced features:

apiVersion: rubrik.com/v1  
kind: DataProtectionPolicy  
metadata:  
 name: data-volume-backup  
spec:  
 dataprotection:  
 enable: true  
 rubrikCluster:  
 name: rubrik-cluster1  
 snapshot:  
 schedule: nightly  
 retention: 31  
datavolume:  
 selector:  
 matchLabels:  
 app: production

#### Recovery Strategy

##### Rubrik Recovery

Rubrik’s high-speed recovery ensures minimal downtime for critical workloads:

1. **Instant Restore**: Quickly recover VMs from snapshots directly on RH OVE.
2. **File-Level Restore**: Execute rapid recovery at the file level for broad access needs.
3. **Automated Recovery Paths**: Simplify recovery workflows through automation.

##### Recovery Plan

Define a detailed recovery plan to access Rubrik’s capability.

apiVersion: v1  
kind: ConfigMap  
metadata:  
 name: recovery-plan  
 namespace: recovery  
spec:  
 paths:  
 critical-apps:  
 - name: app1  
 vm: app1-vm  
 backup: latest  
 action: restore  
 - name: app2  
 vm: app2-vm  
 backup: nightly  
 action: restore  
 hooks:  
 preRestore:  
 - /scripts/pre-restore.sh  
 postRestore:  
 - /scripts/post-restore.sh

#### Testing and Validation

##### Backup Verification

Regularly test backups to ensure integrity and reliability:

* **Backup Verification Schedules**: Conduct routine checks on backup snapshots for quality assurance.
* **Periodic Restore Drills**: Simulate restore scenarios to assess recovery time objectives.

##### Recovery Assurance

Ensure recovery processes are validated and documented:

* **Recovery Testing**: Periodically execute recovery processes within a non-production environment.
* **Documentation**: Maintain up-to-date recovery documentation with steps, tools, and responsible parties.

#### Monitoring and Alerts

Utilize monitoring tools to track backup and restore activities:

* **Alerting Policies**: Implement alerts for failed backups, missed schedules, or data integrity issues.
* **Monitoring Dashboards**: Use dashboards to visualize backup/recovery activities and efficiency metrics.

#### Conclusion

By following these backup and recovery strategies, organizations can ensure the safety, integrity, and availability of their critical data within the RH OVE ecosystem. Taking advantage of Rubrik’s robust integration further enhances data resilience, minimizing risks associated with data loss.