RH OVE Deployment Documentation

Red Hat OpenShift Virtualization Ecosystem Team

2025-08-04

Table of Contents

# Deployment

## Prerequisites

### Prerequisites

#### Overview

This document outlines the prerequisites for deploying the RH OVE ecosystem, including infrastructure requirements, software dependencies, and configuration prerequisites.

#### Infrastructure Requirements

##### Hardware Requirements



##### Virtualization Support

Ensure hardware virtualization is enabled:

### Check for Intel VT-x
grep -E "(vmx|svm)" /proc/cpuinfo

### Check if virtualization is enabled in BIOS
lscpu | grep Virtualization

### Verify KVM modules are loaded
lsmod | grep kvm

##### Network Requirements

* **Cluster Network**: Internal cluster communication
* **Service Network**: Service-to-service communication
* **Pod Network**: Pod-to-pod communication
* **External Access**: Load balancer and ingress traffic

### Network configuration example
cluster\_network:
 cidr: "10.128.0.0/14"
 host\_prefix: 23

service\_network:
 - "172.30.0.0/16"

machine\_networks:
 - cidr: "192.168.1.0/24"

#### Software Prerequisites

##### OpenShift Container Platform

* **Version**: OpenShift 4.12+ (recommended 4.14+)
* **Installation Method**: IPI (Installer Provisioned Infrastructure) or UPI (User Provisioned Infrastructure)
* **Cluster Admin Access**: Required for operator installation

##### Required Operators



##### Storage Requirements

###### Container Storage Interface (CSI) Drivers

### Example CSI StorageClass
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
 name: rh-ove-ssd
provisioner: ebs.csi.aws.com
parameters:
 type: gp3
 encrypted: "true"
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer
allowVolumeExpansion: true

###### Storage Classes Required

* **Fast SSD**: For VM boot disks and high-IOPS workloads
* **Standard HDD**: For data storage and backup
* **Archive**: For long-term storage and compliance

#### Network Prerequisites

##### DNS Configuration

### DNS configuration for cluster
dns:
 base\_domain: "ove.example.com"
 cluster\_domain: "cluster.local"
metadata:
 name: "rh-ove-cluster"

##### Load Balancer Configuration



##### Firewall Rules

Required ports for RH OVE:

| Port Range | Protocol | Purpose |
| --- | --- | --- |
| 6443 | TCP | Kubernetes API server |
| 22623 | TCP | Machine config server |
| 80/443 | TCP | HTTP/HTTPS ingress |
| 9000-9999 | TCP | Host level services |
| 10250-10259 | TCP | Kubernetes node ports |
| 30000-32767 | TCP | NodePort services |

#### Security Prerequisites

##### Certificate Management

### TLS certificate configuration
tls:
 ca\_cert: |
 -----BEGIN CERTIFICATE-----
 # CA certificate content
 -----END CERTIFICATE-----

 api\_cert: |
 -----BEGIN CERTIFICATE-----
 # API server certificate
 -----END CERTIFICATE-----

##### RBAC Configuration

Prepare service accounts and roles:

apiVersion: v1
kind: ServiceAccount
metadata:
 name: rh-ove-admin
 namespace: openshift-cnv
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
 name: rh-ove-admin-binding
roleRef:
 apiGroup: rbac.authorization.k8s.io
 kind: ClusterRole
 name: cluster-admin
subjects:
- kind: ServiceAccount
 name: rh-ove-admin
 namespace: openshift-cnv

#### External System Prerequisites

##### Git Repository Setup

For GitOps implementation:

### Create GitOps repository structure
mkdir -p rh-ove-gitops/{applications,infrastructure,policies}

### Initialize Git repository
cd rh-ove-gitops
git init
git remote add origin https://git.example.com/rh-ove-gitops.git

##### Rubrik Backup System

Prerequisites for Rubrik integration:

* **Rubrik cluster**: Version 5.0+
* **Network connectivity**: Cluster to Rubrik management network
* **Service account**: With backup and restore permissions
* **API access**: Rubrik REST API credentials

### Rubrik connection configuration
apiVersion: v1
kind: Secret
metadata:
 name: rubrik-credentials
 namespace: rubrik-system
type: Opaque
stringData:
 username: "rubrik-service-account"
 password: "secure-password"
 cluster-address: "rubrik.example.com"

##### Dynatrace Monitoring

Prerequisites for Dynatrace integration:

* **Dynatrace tenant**: SaaS or Managed
* **API tokens**: With required permissions
* **Network access**: Cluster to Dynatrace endpoints

### Dynatrace API token secret
apiVersion: v1
kind: Secret
metadata:
 name: dynakube
 namespace: dynatrace
type: Opaque
stringData:
 apiToken: "dt0c01.xxxxx"
 dataIngestToken: "dt0c01.yyyyy"

##### ServiceNow CMDB

For CMDB integration:

* **ServiceNow instance**: With CMDB module
* **Service account**: With CMDB read/write permissions
* **API access**: REST API and webhooks configured

#### Validation Checklist

##### Pre-Installation Checks

### Validation script example
apiVersion: v1
kind: ConfigMap
metadata:
 name: pre-install-checks
data:
 validate.sh: |
 #!/bin/bash

 # Check OpenShift version
 oc version

 # Verify cluster resources
 oc get nodes
 oc get storageclass

 # Check virtualization support
 oc get nodes -o json | jq '.items[].status.allocatable'

 # Validate network connectivity
 curl -k https://registry.redhat.io/health

##### Resource Verification

### Check available resources
oc adm top nodes

### Verify storage classes
oc get storageclass

### Check network plugins
oc get network.config/cluster -o yaml

### Validate image registry access
oc get imagestreams -n openshift

#### Installation Timeline



This comprehensive prerequisites guide ensures all necessary components and configurations are in place before beginning the RH OVE deployment process.

## Installation

### Installation Guide

#### Overview

This installation guide provides step-by-step instructions to deploy the RH OVE ecosystem using a multi-cluster architecture. The deployment follows a hub-and-spoke pattern with one management cluster and multiple application clusters for different environments (production, staging, development).

#### Multi-Cluster Architecture

The RH OVE ecosystem consists of:

* **1 Management Cluster**: Centralized control plane for governance, policy, monitoring, and GitOps
* **N Application Clusters**: Dedicated workload execution environments for virtual machines and containers

#### Installation Flow



#### Core Component Installation

##### OpenShift Cluster Setup

1. **Install OpenShift**
	* Follow [OpenShift Installation Docs](https://docs.openshift.com/) to set up the cluster.
	* Choose between IPI or UPI depending on your infrastructure.
2. **Verify Cluster Health**
* oc get nodes
oc get pods -n openshift-apiserver

##### Virtualization Operator

1. **Install OpenShift Virtualization**
* oc apply -f https://path/to/virtualization-operator.yaml
1. **Verify Installation**
* oc get pods -n openshift-cnv
oc get kubevirt.kubevirt.io/kubevirt -n openshift-cnv

##### Cilium CNI

1. **Install Cilium**
* helm repo add cilium https://helm.cilium.io/
helm install cilium cilium/cilium --namespace kube-system
1. **Verify Cilium Status**
* cilium status

##### Kyverno Policy Engine

1. **Install Kyverno**
* kubectl create -f https://github.com/kyverno/kyverno/releases/download/v1.5.2/install.yaml
1. **Apply Policies**
* kubectl apply -f /path/to/policy-files

#### Monitoring Setup

##### Dynatrace Integration

1. **Install Dynatrace Operator**
* oc apply -f https://path/to/dynatrace-operator.yaml
1. **Configure DynaKube**
* oc apply -f /path/to/dynakube-config.yaml
1. **Verify Monitoring**
* oc get pods -n dynatrace

##### Prometheus and Grafana

1. **Install Prometheus Operator**
* oc apply -f https://path/to/prometheus-operator.yaml
1. **Setup Grafana**
* oc apply -f https://path/to/grafana-deployment.yaml

#### Backup Configuration

##### Rubrik Integration

1. **Install Rubrik Operator**
* oc apply -f https://path/to/rubrik-operator.yaml
1. **Verify Backup**
* oc get pods -n rubrik

#### GitOps Setup

##### Argo CD Installation

1. **Install Argo CD**
* oc apply -n argocd -f https://path/to/argocd-install.yaml
1. **Access Argo CD UI**
	* Forward Argo CD API server port:
	* oc port-forward svc/argocd-server -n argocd 8080:443
2. **Login to Argo CD**
	* Open <https://localhost:8080> in your browser.
3. **Deploy Applications**
* argocd app create my-app --repo https://git.example.com/my-app --path ./
argocd app sync my-app

#### Security Hardening

1. **Configure RBAC**
* oc apply -f /path/to/rbac-config.yaml
1. **Enable Pod Security**
* oc apply -f /path/to/pod-security.yaml
1. **Firewall Adjustments**
	* Ensure only necessary ports are open (refer to [prerequisites](prerequisites.md)).

#### Validation Steps

##### Verify All Deployments

oc get all --all-namespaces

##### Check Monitoring Dashboards

* Confirm metrics collection in Grafana and Dynatrace.

#### Post-Installation Tasks

##### Documentation

* Update [MkDocs](../mkdocs.yml) with new components.

##### Backup Verification

* Test Rubrik backups for VM and container data.

#### Conclusion

This guide ensures a smooth installation process for RH OVE, covering all critical steps and components necessary for successful deployment and operation. Follow each section carefully to complete the installation.

## Configuration

### Deployment Configuration

#### Overview

This document provides configuration guidelines for the RH OVE deployment, focusing on customization and parameters essential for adapting the solution to your specific environment.

#### OpenShift Configuration

##### Cluster Configuration

Customize your OpenShift cluster with the necessary configurations to optimize performance and security:

apiVersion: config.openshift.io/v1
kind: ClusterVersion
metadata:
 name: version
spec:
 channel: stable
 upstream: https://api.openshift.com/api/upgrades\_info/v1/graph

### Customization to networking
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
 name: cluster
spec:
 clusterNetwork:
 - cidr: 10.128.0.0/14
 hostPrefix: 23
 serviceNetwork:
 - 172.30.0.0/16

##### Node Configuration

Optimize your nodes for workload management:

apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfigPool
metadata:
 name: worker
spec:
 machineConfigSelector:
 matchExpressions:
 - key: machineconfiguration.openshift.io/role
 operator: In
 values:
 - worker
 nodeSelector:
 matchLabels:
 node-role.kubernetes.io/worker: ""

### Taints to manage workloads effectively.
apiVersion: v1
kind: Node
metadata:
 name: node-1
spec:
 taints:
 - key: app
 value: high-performing
 effect: NoSchedule

#### Network Configuration

Customize your Cilium CNI settings:

apiVersion: cilium.io/v2
kind: CiliumNetworkConfig
metadata:
 name: cilium-config
spec:
 endpointRoutes: true
 devices:
 - eth0
 autoDirectNodeRoutes: true

### Policy for specific namespace isolation requirements
apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
metadata:
 name: namespace-isolation-policy
 namespace: critical-apps
spec:
 endpointSelector:
 matchLabels:
 app: critical-environment
 ingress:
 fromEndpoints:
 - matchLabels:
 access: dedicated

#### Storage Configuration

Manage your storage setups efficiently:

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
 name: performance-storage
provisioner: ebs.csi.aws.com
parameters:
 type: io1
 iopsPerGB: "50"
 encrypted: "true"
reclaimPolicy: Retain
volumeBindingMode: WaitForFirstConsumer

### PVC for critical workloads needing high IOPS
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: critical-workload-pvc
spec:
 accessModes:
 - ReadWriteOnce
 resources:
 requests:
 storage: 100Gi
 storageClassName: performance-storage

#### Security Configuration

Strengthen the security of your deployment:

### Role-based access control
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
 namespace: secure-namespace
 name: critical-role
rules:
- apiGroups:
 - ""
 resources:
 - pods
 - services
 verbs:
 - get
 - list
 - watch

### Pod Security Policies
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
 name: restricted-psp
spec:
 privileged: false
 allowPrivilegeEscalation: false
 requiredDropCapabilities:
 - ALL
 volumes:
 - 'configMap'
 - 'emptyDir'
 - 'persistentVolumeClaim'

#### Conclusion

By properly configuring these parameters, you can ensure that your RH OVE deployment is optimized for performance, security, and operational effectiveness. Adjust configurations based on specific organizational policies and workload demands.