|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| RedHat OpenShift Virtualization Services | | | | | | |
| Number: | Local Document | | Status: | | Draft | |
| Owner: | Adrian Pohancanik | | Effective: | | 01. Apr 2026 | |
| Doc.- type: | Design | | Revision | | 0.1 | |
| Comments: |  | | | | | |
| [Authorized by:  (Optional, delete if not required)] | [Name] | [Name] | | [Name] | | IT-MIO-INFRASTRUCTURE ENGINEERING |
| [Department] | [Department] | | [Department] | |

Table of Contents

1. Purpose 1

2. Scope 1

3. Summary 1

4. Technology Standard 1

5.1. High Level Design 2

5.2. Version 2

5.3 Cluster Specification 2

5.4 Hardware Standards 2

5.5 Storage and performance classes 4

5.6 Automation Scripts 5

5.7 Monitoring Defined 5

5.8ITSM Integration 5

5.9 Standard Changes Defined 6

5. Configuration standard 6

5.1. OpenShift Server Deployment 7

5.2. Create Project / Namespace 14

5.3. Command VM operations 15

5.4. Define Access roles 15

5.5. Grant a User Permissions in a Project 16

OpenShift Cluster administrators 17

5.6. DN Certificate installation 18

5.7. SSO AD Loging OpenID 21

5.8. CPU / RAM size definition 21

5.9. High availability / Node Remediation 22

5.10. DRS Descheduler Operator 22

5.11. VM Load Balancing 22

5.12. Affinity rules 22

5.13. Storage definitions 24

5.14. Backup of VMs/Containers 29

5.15. Creating and managing Linux and Windows virtual machines (VMs) 34

5.16. Running Guest Agent 35

5.17. Running POD and VM workloads alongside each other in a cluster 35

5.18. Connecting to virtual machines through a variety of consoles and CLI tools 37

5.19. Importing and cloning existing virtual machines 38

5.20. Additional network interface controllers and storage disks attached to virtual machine 38

5.21. Live migrating virtual machines between nodes / DR 38

Migrate a VM using the 39

5.22. Moving a Virtual Machine Between Projects 40

5.23. VM Migration from VMware to OpenShift 41

5.24. VM Snapshot – Recovery 48

5.25. Rack Level Availability 49

5.26. Templates deployment Windows / Linux 50

5.27. Patching templates – Lifecycle 50

5.28. OpenShift Backup and Restore 50

5.29. Tools installation 51

5.30. OC Nodes Maintenance 52

5.31. Login to OpenShift / Token 54

5.32. Checkmk Monitoring 55

5.33. System Log monitoring 61

5.34. Install the Dell CSM 62

5.35. Enable virtualization in OpenShift 62

5.36. Enabling IOMMU 63

5.37. Enable NUMA for a VM 64

5.38. VBS enable in OpenShift 65

5.39. Migration Toolkit for Virtualization (MTV) 67

6. Security Concept 69

7. RBAC / Access Concept 69

8. Contacts 70

9. Definition / Terms and Abbreviations 70

10. Employee Roles and Responsibilities 70

11. Related Documents / References 70

12. History 70

# Purpose

The purpose of this document is to provide the set of minimal requirements which go together to define an IT Infrastructure Service. This is to provide additional guidance when services are created.

# Scope

The scope of this document is limited to the IT-MIO-INFRASTRUCTURE ENGINEERING.

# Summary

This document provides a definition of an IT Infrastructure Service. It is intended for IT infrastructure teams responsible for designing, configuring, and delivering services. Each block includes a clear purpose, detailed description, and proposal implementation requirements. This ensures consistency, security, compliance, and operational excellence across all infrastructure services.

# Technology Standard

## 5.1. High Level Design

**The Core Object Structure** 

      An OpenShift VM is defined by two primary objects:

* **VirtualMachine (VM):** The "controller" (similar to a Deployment). it defines whether the VM should be running and the backup/restore policies.
* **VirtualMachineInstance (VMI):** The "pod" (created automatically by the VM object). It contains the actual runtime state and hardware specs.

## 5.2. Version

5.3 Cluster Specification

**Production-Ready Minimal Size Cluster:**

HA-capable Minimal Cluster 6x

* 3x Control plane (master) node
* 3x Worker (Compute) node

|  |  |  |  |
| --- | --- | --- | --- |
| Nodes | Purpose | Server type |  |
| 3x | Control plane (master) | Physical |  |
| 3+ | Worker (Compute) | Physical | **512GB RAM small  1TB RAM  2x 64core** |

## 5.4 Hardware Standards

|  |  |
| --- | --- |
| Server Models | RedHat Version |
| Dell PowerEdge R670 | 4.16+ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Product Name** | **Product Quantity** | **Module Name** | **Option Name** | **Quantity** |
| PowerEdge R670  - Datacenter Large | 1 |  |  |  |
| Basis | PowerEdge R670 Server | 1 |
| Chassis Configuration | Without hard drive, without backplane | 1 |
| Processor | Intel® Xeon® 6 Efficient 6710E 2.4 GHz, 64 Cores/64 Threads, 24 GT/s, 96 MB Cache, Turbo, (205 W) DDR5-5600 | 1 |
| Additional Processor | Intel® Xeon® 6 Efficient 6710E 2.4 GHz, 64 Cores/64 Threads, 24 GT/s, 96 MB Cache, Turbo, (205 W) DDR5-5600 | 1 |
| Processor Cooling Configuration | Heatsink for configuration with 2 CPUs (CPU at least 185W and less than 270W) | 1 |
| Memory Configuration Type | Performance Optimization | 1 |
| DIMM Type and Speed | 6.400 MT/s, RDIMMs | 1 |
| Memory Capacity | 64 GB, RDIMM, 6.400 MT/s, Dual-Rank | 16 |
| RAID Configuration | D1 – Diskless Configuration (no RAID, no controller) | 1 |
| RAID/Internal Storage Controller | No Controller | 1 |
| Hard Drive | Without hard drive | 1 |
| BIOS and Advanced System Settings | BIOS setting: 'Energy Safe' | 1 |
| Advanced System Configurations | UEFI BIOS boot mode with GPT partition | 1 |
| Fans | PowerEdge, 1 HE, Silver fan with high Performance | 1 |
| Integrated System Management | iDRAC10 Datacenter 17G mit OpenManage Enterprise Advanced Plus | 1 |
| Integrated System Management | Dell Connectivity Client – Disabled17G | 1 |
| Power Supply | Dual, fully redundant (1+1), hot-plug MHS power supply, 1100W MM (100–240V AC), Titanium | 1 |
| Power Cable | Rack power cables, 2 m (C13/C14, 10A) | 2 |
| PCIe Riser | Riser configuration 6, 2x16 LP slots (Gen 5) rear, 1x16 OCP, 1x8/x16 OCP, warm aisle | 1 |
| Motherboard | PowerEdge R670, Motherboard, ROW | 1 |
| OCP 3.0 Network Adapter | Broadcom 57414, 2 connections, 25 GbE, SFP28-Adapter, OCP 3.0 NIC +Sec | 2 |
| Additional Network Cards | Broadcom 57414, 25 GbE, SFP28, Adapter with 2 connections, PCIe, flat Profile, +Sec | 1 |
| OCP 3.0 Accessories | 2 OCP – without Cable | 1 |
| Password | iDRAC, Legacy-Kennwort for OCP-Cards | 1 |
| Bezel | PowerEdge, 1 HE, Standard bezel for E3 enclosures for x16, x20 or without back panel | 1 |
| Cables | No DPU cable required, no DPU | 1 |
| Boot Optimized Storage Cards | BOSS-N1 controller card + with 2 x M.2, 960GB (RAID 1) (22x80), rear | 1 |
| Optics & Cables for Network Cards | Dell Networking – Transceiver, 25 GbE, SFP28, SR, without FEC, MMF, Duplex-LC | 6 |
| Operating System | No operating system | 1 |
| OS Media Kits | No media required | 1 |
| Rack Rails | Ready Rails sliding rails without cable management arm | 1 |
| Quick Sync | Empty module, left rack angle module | 1 |
| Shipping | PowerEdge R670, Shipment EMEA1 | 1 |
| Packaging Material | PowerEdge, 1 HE, shipping material | 1 |
| Regulatory Compliance | PowerEdge with CCC and CE marking | 1 |
| ECCN | Reject selection | 1 |
| Standard Service | Standard service on the next working day, 36 months, 36 month(s) | 1 |
| Extended Service | ProSupport Plus and Mission Critical Service within 4 hours, 60 months | 1 |
| Keep Your Hard Drive for Enterprise | Keep Your Hard Drive for Enterprise, 60 Monat(e) | 1 |
| Theft Protection and Asset Tagging | Inventory identification - ProSupport (website, barcode, integrated MAC address) | 1 |
| Shipping Box Labels - Standard | Shipping label – Order configuration (shipping date, model, processor speed, hard drive size, RAM) | 1 |

## 5.5 Storage and performance classes

**For Dell:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Storage Array(s)** | **Model** | **Protocol** | **Product Version (s)** | **Red Hat OpenShift Version (s)** | **Infrastructure features** |
| PowerFlex | Rack or Appliance | TCP-IP proprietary protocol, NVMe/TCP | 5.8 | [4.19+](https://access.redhat.com/bounce/?externalURL=https%3A%2F%2Fdell.github.io%2Fcsm-docs%2Fdocs%2Fsupportmatrix%2F%23storage-platforms) | CSI, OpenShift Virtualization |

**For NetApp:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Storage Array(s)** | **Protocol** | **Product Version (s)** | **Red Hat OpenShift Version (s)** | **Infrastructure features** |
| AFF A-Series | TFC[tech preview], NVMe over TCP, iSCSI, NFS, SMB | A50  C60 | 4.19+ | CSI, OpenShift Virtualization, VDI |
| FAS | FC[tech preview], NVMe over TCP, iSCSI, NFS, SMB | 8200 | 4.19+ | CSI, OpenShift Virtualization, VDI |
| A300 | FC[tech preview], NVMe over TCP, iSCSI, NFS, SMB | A300 | 4.19+ | CSI, OpenShift Virtualization, VDI |

**Storage protocols: Direct Attached, SDC, NVMe, NFS, iSCSI**

## 5.6 Automation Scripts

Automate provisioning, operations, and recovery.

Lists scripts and tools used for deploying infrastructure.

Automation improves efficiency, consistency, and reliability.

Requirements:

- Infrastructure provisioning scripts (e.g., Ansible, PowerShell)

- Agent deployment scripts

- Self-healing and remediation scripts

- Script repository with version control

## 5.7 Monitoring Defined

Ensure visibility into service health and performance.

Specifies tools, metrics, alerting thresholds, dashboards, and escalation paths.

Monitoring enables proactive issue detection and resolution.

Requirements:

- Monitoring tools (e.g., CheckMK, Grafana)

- Defined metrics (CPU, memory, disk, application)

- Alerting thresholds and escalation matrix

- Real-time and historical dashboards

- Integration with incident management

## 5.8 ITSM Integration

Enable users to request the service via ITSM.

Describes how the service is represented in the ITSM tool, including catalog entries, workflows, SLAs, and automation. It ensures a smooth and traceable request process.

Requirements:

- Service catalog entry in ITSM

- Request workflow with approvals

- SLA definitions (response and resolution times)

- Service / Service Component Name

- Fulfillment automation integration

- User communication templates

## 5.9 Standard Changes Defined

Streamline low-risk, repeatable changes.

Identifies changes that can be pre-approved and executed without full CAB review.

Includes templates, rollback plans, and scheduling guidelines.

Requirements:

- Definition of pre-approved changes

- Change templates with rollback and test plans in ITSM

- Change scheduling and notification process

# Configuration standard

**Key Configuration Standards**

|  |  |
| --- | --- |
| no | RedHat OpenShift Deployment |
| 1 | Pre-Implantation Steps |
| 2 | ESXi 8, vCenter 8 |
| 3 | Windows 2019 VM with AD, DNS, DHCP |
| 4 | RHEL 8x, 9x Helper VM |
| 5 | Join all system to Base Domain |
| 6 | Create a Domain Admin account for vCenter |
| 7 | Configure DNS record for OpenShift Cluster |
| 8 | Configure DHCP scope for OpenShift Cluster / IP and MAC Address |
| 9 | Configure DHCP Scope for OpenShift Cluster Nodes / IP and MAC Address |
| 10 | Download OpenShift installer |
| 11 | Implementation Procedure |
| 12 | Generating a Key Pair for OpenShift Cluster node SSH access |
| 13 | Adding vCenter root CA Certificates to OpenShift Helper VM to establish Trust |
| 14 | Create a Working directory on Helper VM |
| 15 | Extract the OpenShift installer is a present working directory |

When defining VMs in OpenShift, follow these industry standards:

|  |  |  |
| --- | --- | --- |
| **Feature** | **Standard Configuration** | **Reason** |
| **Storage** | ContainerizedDataVolume | Uses CDI to clone images efficiently from templates. |
| **Network** | virtio | Provides the highest I/O throughput for Linux/Windows guests. |
| **Run Strategy** | HaltedIfReady | Ensures the VM survives node reboots or maintenance. |
| **Eviction Policy** | LiveMigrate | Allows the VM to move to another node without downtime during host updates. |

## OpenShift Server Deployment

**Virtualization Environment requirements**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **OCP Virtualization** | **Count** | **Type** | **CPU** | **RAM** | **Disk size for OS** | **NIC Count** |
| Worker nodes | 3+ | Physical | 64  CPU | 1000GB | 100 GB | 6 (VM traffic & SAN access) |
| Master nodes | 3 | Physical | 8 CPU | 64GB | 100 GB | 2 |
| Bootstrap(temporary) | 1 | Virtual | 8 vCPU | 8GB | 50GB | 1 |
| Jump/Bastion | 1 | Virtual | 8 vCPU | 8GB | 50GB | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **VLAN / Bond Type** | **Purpose** | **VLAN** | **Network** |
| Management VLAN | To build an OCP cluster. One for each node and one additional IP for bootstrap IP. | VLAN ID | /24 |
| Worker node VLAN (optional) | To add additional worker load Network for VMs. | VLAN ID | /24 |
| VM Traffic VLAN | VLAN for VM traffic. | VLAN / EPG | ACI |
| SAN VLAN | VLAN for SAN traffic. | VLAN ID | /24 |

Red Hat OpenShift 4.20 Installation

**Objective**

Deploy a functional OpenShift cluster to validate platform capabilities including container orchestration, virtualization, and integration with enterprise infrastructure.

**Installation Overview**

* **Platform**: Red Hat OpenShift Container Platform 4.20
* **Hardware:**VMware vSphere for Master Nodes and Dell for Worker Nodes
* **Deployment Type**: UPI (User-Provisioned Infrastructure)
* **Environment**: Firewall-restricted zone with proxy access (NC, DC3)
* **Cluster Size**: 3 control plane nodes, 6 worker nodes
* **Virtualization**: OpenShift Virtualization Enabled
* **Migration Toolkit for Virtualization (MTV)**: Installed for VMware VM migration

**Key Installation Steps**

1. **Prepare Infrastructure**

Provision RHEL CoreOS nodes

Configure DNS and F5 requirements

Ensure required ports are open (API, ingress, etc.)

1. **Configure Proxy**

Proxy details added in install-config.yaml

NO\_PROXY includes internal services and VMware infrastructure

1. **Generate and Apply Manifests**

Run openshift-install create manifests

Customize manifests as needed (e.g., image registry, proxy, etc.)

1. **Deploy Cluster**

Run openshift-install create cluster

Monitor bootstrap and installation logs

1. **Post-Install Configuration**

Configure Image Registry (e.g., emptyDir, PVC, or external storage)

Enable OpenShift Virtualization

Install MTV via OperatorHub

**Validation Goals**

Verify cluster health

Confirm virtualization functionality (VM creation, lifecycle)

Test VM migration from VMware using MTV

**Infrastructure Overview**

|  |  |  |
| --- | --- | --- |
| **Component** | **Quantity** | **Purpose** |
| Master Nodes | 3 | Control plane for cluster management |
| Worker Nodes | 3+ | Application workloads and services |
| Bootstrap Node | 1 | Temporary node for initial cluster bootstrap |
| Bastion Host | 1 | Secure access point for cluster management |
| F5 Load Balancer | 1 | External load balancing for API and ingress |

**Load Balancing**:

F5 Network Load Balancer passthrough

api.ocp4test.ad.diebold.com - VIP IP - 10.9.8.225

api-int.ocp4test.ad.diebold.com - VIP IP- 10.9.8.226

\*.apps.ocp4test.ad.diebold.com - VIP IP- 10.9.8.227

**Bastian Node:**

usnc1cikube1lvd - 10.9.104.23

**Master Nodes:**

usnc1ciocm1lpd.ad.diebold.com 10.9.104.67

usnc1ciocm2lpd.ad.diebold.com 10.9.104.69

usnc1ciocm3lpd.ad.diebold.com 10.9.104.71

**Bootstrap Node:**  Not added in DNS, because we can remove bootstrap node after master and worker build complete

**Bootstrap Node:**

usnc1ciocb1lpd.ad.diebold.com 10.9.104.94

**Worker Nodes:**

usnc1ciocw1lpd.ad.diebold.com 10.9.104.12

usnc1ciocw2lpd.ad.diebold.com 10.9.104.22

usnc1ciocw3lpd.ad.diebold.com 10.9.104.54

usnc1ciocw4lpd.ad.diebold.com 10.9.104.92

usnc1ciocw5lpd.ad.diebold.com 10.9.104.93

usnc1ciocw6lpd.ad.diebold.com 10.9.104.94

**In Bastian Node:**

**Step 1:** Configure Web server and Download openshift-install-linux.tar.gz and extract

# yum install http;systemctl start httpd;systemctl enable httpd

# cd /mnt/extras/; tar -xvf openshift-install-linux.tar.gz

**Step 2:** Download/extract openshift-client-linux-amd64-rhel8-4.17.8.tar.gz and copy binaries

# tar -xvf openshift-client-linux-amd64-rhel8-4.17.8.tar.gz

# cp -p oc kubectl /usr/local/bin/

# oc version

# kubectl version

**Step 3:**Create install directory , change permission and umask.

# mkdir /root/ocp4install

# chmod 755 /root/ocp4install

# umask 022

# umask

**Step 4:** create /root/ocp4install/install-config.yaml file  - refer docs.redhat.com for code

# cd /root/ocp4install/;vi install-config.yaml

# cp install-config.yaml /mnt/extras/

 taken backup for reference cause it will be deleted in install directory after create manifests

**Step 5:** Create manifests

# openshift-install create manifests --dir /root/ocp4install

# ls -lrt /root/ocp4install

# ls -lrt /root/ocp4install/manifests

# ls -lrt /root/ocp4install/openshift

# cat /root/ocp4install/manifests/cluster-scheduler-02-config.yml

mastersSchedulable: false -- cause no schedules on master node

**Step 6:** Create ignition config files and copy it to web server

# openshift-install create ignition-configs --dir /root/ocp4install

# ls -lrt /root/ocp4install

# cd /root/ocp4install; cp master.ign worker.ign bootstrap.ign /var/www/html/

# cd /var/www/html/; chmod o+r worker.ign master.ign bootstrap.ign

**Step 7:** Now setup bootstrap node and master node and followed by worker nodes

Map CoreOS ISO to new VMS/Physical servers and Boot with ISO

Configure network/hostname with "nmtui"

Example:

Hostname: usnc1cikube7lvd.ad.diebold.com

IP: 10.9.104.12

subnet bit: 22

subnet mask: 255.255.252.0

gateway: 10.9.104.1

\*\*\* Note: We must install bootstrap and master nodes parallel otherwise bootstrap node install will fail but worker node can install latter too

\*\*\* also bootstrap ignition file valid only 24hours since it creates

**For Bootstrap node:**

sudo coreos-installer install /dev/sda --copy-network --ignition-url=http://10.9.104.23/bootstrap.ign --insecure-ignition

**For Master/control-plane node:**

sudo coreos-installer install /dev/sda --copy-network --ignition-url=http://10.9.104.23/master.ign --insecure-ignition

**For Worker node:**

sudo coreos-installer install /dev/sda --copy-network --ignition-url=http://10.9.104.23/worker.ign --insecure-ignition

**To validate Bootstarp installation:**

# openshift-install --dir /root/ocp4install wait-for bootstrap-complete --log-level=info

# openshift-install --dir /root/ocp4install wait-for bootstrap-complete --log-level=debug

# journalctl -b -f -u release-image.service -u bootkube.service  # run from bootstrap node by login

**To Validate Master and Worker node installation:**

# ssh core@10.9.104.12

# crictl images

# crictl ps

# crictl ps -a

# sudo journalctl -b -f -u kubelet.service -u crio.service

# export KUBECONFIG=/root/ocp4install/auth/kubeconfig

# oc get nodes

# oc get co

# oc get clusterversion

# oc get co authentication -o yaml

# oc get csr

# oc get events

# oc get events -A  --sort-by='.lastTimestamp'

run following command to approve certificates incase if you see any pending approval csr from "oc get csr"

# oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S.No | **Servers** | | | | | | | | | | | | | | | | | | | | | |
| 11 |  | Type | | CPU | | RAM | | Space for OS | | NIC Count | | | Connectivity Type1 | | Connectivity Type2 | | Connectivity Type3 | | Connectivity Type4 | | Quantity | |
| Physical | Worker nodes | | 2 socket  10 core each | | 128 GB | | 100 GB | | 6-8 nics | | | Management Bond | | VM Traffic | | Container Node NW | | SAN Connection | | 3+ | |
| Virtual/Physical(in case of Prod ready infra) | Master nodes | | 8 vCPUs | | 32 GB | | 100 GB | | 2 nics | | | Management Bond | | NA | | NA | | NA | | 3 | |
| Virtual/Physical(in case of Prod ready infra) | Infrastructure nodes | | 8 vCPUs | | 32 GB | | 100 GB | | 2 nics | | | Management Bond | | NA | | NA | | NA | | 3 | |
| Virtual | Bootstrap(temporary) | | 8 vCPUs | | 32 GB | | 100 GB | | 2 nics | | | Management Bond | | NA | | NA | | NA | | 1 | |
| Virtual | Jump/Bastion | | 8 vCPUs | | 32 GB | | 100 GB | | 2 nics | | | Management Bond | | NA | | NA | | NA | | 1 | |
|  |  |  | |  | |  | |  | |  | | |  | |  | |  | |  | |  | |
| 2 | VLAN/Bond Type | | Purpose | | | VLAN | | Network | | | | Remarks | | |  | |  | |  | |  | |
| Management VLAN | | To build OCP cluster | | |  | | x.x.x.x/24 | | | | Network not empty! | | |  | |  | |  | |  | |
| Worker node VLAN(optional) | | To add additional workerload Network for containers | | |  | | x.x.x.x/24 | | | | Network not empty! | | |  | |  | |  | |  | |
| VM Traffic VLAN | | To add VLANs for VM traffic | | |  | | ACI/VLAN | | | |  | | |  | |  | |  | |  | |
| SAN VLAN | | To add VLAN for SAN traffic | | |  | | x.x.x.x/24 | | | | Network not empty! | | |  | |  | |  | |  | |
|  |  | |  | |  | |  | |  | |  | | |  | |  | |  | |  | |  | |
| 3 | IP on Bond Type | | Purpose | | |  | | | | | |  | | |  | |  | |  | |  | |
| Management Ip | | To build OCP cluster. One for each node and one additional IP for bootstrap IP. | | |  | | x.x.x.x/24 | | | | Network not empty! | | |  | |  | |  | |  | |
| Worker node VLAN | | To add additional workerload Network for containers. One for each node | | |  | | x.x.x.x/24 | | | | Network not empty! | | |  | |  | |  | |  | |
| VM Traffic VLAN | | To add VLANs for VM traffic. One for each node | | |  | | ACI | | | |  | | |  | |  | |  | |  | |
| SAN VLAN | | To add VLAN for SAN traffic. One for each node | | |  | | x.x.x.x/24 | | | | Network not empty! | | |  | |  | |  | |  | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4 | F5 LB request to get the IPs for OCP API and API-Int connectivity | | |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
| URL Users will use to access the service | Protocol/Port for users | Front END IP | Member Server IP address | Protocol/Port Server listens on | SSL Offloading (If yes please attach certificate) | Monitoring (Default TCP port status) | Load Balancing Method (Default Round Robin) | Remarks |
| api.dnpcpoc.test-dnpcloud.dieboldnixdorf.com api-int.dnpcpoc.test-dnpcloud.dieboldnixdorf.com | 6443, 22623 |  |  | 6443, 22623 | N/A | Default | Round Robin | Passthrough(layer 4 routing) |
| \*.apps.dnpcpoc.test-dnpcloud.dieboldnixdorf.com | 443 |  |  | 443 | N/A | Default | Round Robin | SSL |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5 | DNS Records | | | | | |  |  |
| Cluster | Reqired for | Node/LB | Record (Reversed) | Record Type | IP | MAC | MAC + 1 |
| Clustername | Load Balancer | api.clutsername.dieboldnixdorf.com | api.dnpcpoc.test-dnpcloud.dieboldnixdorf.com | A + PTR | 100.65.242.236 |  |  |
| api-int.clustername.dieboldnixdorf.com | api-int.dnpcpoc.test-dnpcloud.dieboldnixdorf.com | A + PTR | 100.65.242.236 |  |  |
| \*.apps.clustername.dieboldnixdorf.com | \*.apps.dnpcpoc.test-dnpcloud.dieboldnixdorf.com | A | 100.65.242.235 |  |  |
| Nodes | bootstrap fqdn | de04mgtlist016 | A + PTR | 100.65.242.99 |  |  |
| Jump/Bastion/Helper | dieboldnixdorf.com | A + PTR | 100.65.242.234 | 00:50:56:98:bc:37 |  |
| de04ocp-master1t | de04mgtlapt031.test-dnpcloud.dieboldnixdorf.com | A + PTR | 100.65.242.101 | 00:50:56:98:c1:1a |  |
| de04ocp-master2t | de04mgtlapt032.test-dnpcloud.dieboldnixdorf.com | A + PTR | 100.65.242.102 | 00:50:56:98:bb:e5 |  |
| de04ocp-master3t | de04mgtlapt033.test-dnpcloud.dieboldnixdorf.com | A + PTR | 100.65.242.103 | 00:50:56:98:2a:31 |  |
| de04ocp-infra1t | de04mgtlapt034.test-dnpcloud.dieboldnixdorf.com | A + PTR | 100.65.242.104 | 00:50:56:98:21:9e |  |
| de04ocp-infra2t | de04mgtlapt035.test-dnpcloud.dieboldnixdorf.com | A + PTR | 100.65.242.105 | 00:50:56:98:67:15 |  |
| de04ocp-infra3t | de04mgtlapt036.test-dnpcloud.dieboldnixdorf.com | A + PTR | 100.65.242.106 | 00:50:56:98:85:05 |  |
| de04ocp-worker1t | de04fnd010ise31.test-dnpcloud.dieboldnixdorf.com | A + PTR | 100.65.242.107 | 00:3a:7d:71:5c:ae | af |
| de04ocp-worker2t | de04fnd010ise32.test-dnpcloud.dieboldnixdorf.com | A + PTR | 100.65.242.108 | 00:78:88:a3:49:e4 | e5 |
| de04ocp-worker3t | de04fnd010ise33.test-dnpcloud.dieboldnixdorf.com | A + PTR | 100.65.242.109 | 00:42:68:6f:83:dc | dd |
| 6 | HW Management | Nodes |  | de04fnd010ise31-cimc.test-dnpcloud.dieboldnixdorf.com | A + PTR | 100.65.241.107 |  |  |
|  | de04fnd010ise32-cimc.test-dnpcloud.dieboldnixdorf.com | A + PTR | 100.65.241.108 |  |  |
|  | de04fnd010ise33-cimc.test-dnpcloud.dieboldnixdorf.com | A + PTR | 100.65.241.109 |  |  |

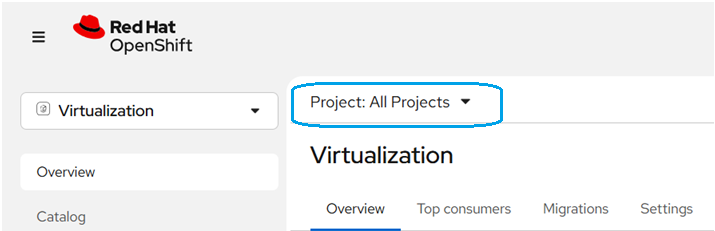
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 7 | Firewall rules | | | | |  |
| OCP Related | | | | |  |
| Purpose | Source | Destination | Port | Protocol |  |
| DNS | OCP Management subnet | Domain controllers | 53 | TCP and UDP |  |
| NTP | OCP Management subnet | Domain controllers | 123 | UDP |  |
| OCP to VIP1 | OCP Management subnet | VIP1 | 6443 22623 | TCP |  |
| OCP to VIP2 | OCP Management subnet | VIP2 | 443 | TCP |  |
| VIP2 to OCP | VIP2 | OCP Management subnet | 443 | TCP |  |
| VIP1 to OCP | VIP1 | OCP Management subnet | 6443 22623 | TCP |  |
| End user access to OCP | AD Security group | OCP Management subnet | 226,443 | TCP |  |
| End user access to OCP cluster API | AD Security group | VIP2 | 443 | TCP |  |
| End user access to OCP cluster API internal | AD Security group | VIP1 | 6443 | TCP |  |
| OCP to proxy | 10.52.85.0/24 | Proxy | 8080 | TCP |  |
|  |  |  |  |  |  |
| OpenID related | | | | | |
| Purpose | Source | Destination | Port | Protocol | Direction |
| Azure AD authentication (OAuth2) | OpenShift API Server / OAuth Pod | login.microsoftonline.com | 443 | TCP | Outbound |
| User info and group membership queries | OpenShift API Server / OAuth Pod | graph.microsoft.com | 443 | TCP | Outbound |
| Azure AD UI assets (optional) | OpenShift API Server / OAuth Pod | aadcdn.msftauth.net | 443 | TCP | Outbound |
| DNS resolution for | OpenShift API Server / OAuth Pod | DNS servers | 53 | TCP/UDP | Outbound |
|  |  |  |  |  |  |
| SAN Related | | | | |  |
| purpose | Source | Destination | port | protocol |  |
| Master node to SAN | Master node IPs | SAN IPs | 443 | TCP and UDP |  |
| worker node to SAN | Worker node IPs | SAN IPs | 443 | TCP and UDP |  |
| Infra node to SAN | Infra node IPs | SAN IPs | 443 | TCP and UDP |  |

## Create Project / Namespace

Login OpenShift console:

Standard project creation

oc **new**-project **my**-project



Show all available projects (alias for namespaces in OpenShift)

oc **get** projects

Show all namespaces

oc **get** namespaces

oc **get** ns -o name

Switch to my project

oc project **my**-project

Login to OpenShift and select project

You must obtain an API token by visiting:

https://oauth-openshift.apps.ocp4test.ad.diebold.com/oauth/token/request

oc login --token=sha256~ABCDEFG --server=https://api.mycluster.openshift.com:6443

oc login --token=sha256~BKlmJ-server=https://api.dnpcpoc.test-dnpcloud.dieboldnixdorf.com:6443

oc login --web https://api.ocp4test.ad.diebold.com:6443 (to login via your browser)

oc project **my**-project

List ALL resource types + all instances

oc api-resources

Then list all:

oc **get** all

List everything in the project

oc **get** $(oc api-resources --verbs=list --namespaced -o name) --ignore-**not**-found

Check which project you're currently using

oc project

## Command VM operations

List all Virtual Machines:

oc **get** vm

List all Virtual Machines with namespace:

oc **get** vm -n **my**-**namespace**

List all VirtualMachineInstances (running VMs):

oc **get** vmi

Show detailed VM info:

oc describe vm **my**-vm

Show YAML definition of a VM:

oc **get** vm **my**-vm -o yaml

List everything related to virtualization:

oc api-resources | findstr /i virt / oc api-resources | grep -i virt

Start a VM

oc start vm <vm-name>

oc start vm myvm -n mynamespace

Restart a VM

oc restart vm <vm-name>

Reboot from inside the VM (via exec) guest OS supports required

oc **exec** vmi/<vm-name> -- reboot

Stop / Shutdown a VM (Graceful)

oc stop vm <vm-name>

Force Stop / Hard Power Off

oc **delete** vmi <vm-name>

Check VM Status

oc **get** vm

oc **get** vmi

oc **get** virtualmachine VM1 -o wide

oc **get** vmi -l app=ha-vm

oc describe vmi VM1

example: label for group of servers: SQL-DB

List all VMs with **label** app=SQL-DB (all namespaces)

oc **get** vm -A -l app=SQL-DB

List running VM instances (VMI) with that label

oc **get** vmi -A -l app=SQL-DB -o wide

List virt‑launcher pods for SQL‑DB VMs

oc **get** pods -A -l app=SQL-DB -o wide

List VMs in a specific namespace

oc **get** vm -n project-adrian -l app=SQL-DB

## Define Access roles

List all OpenShift users

oc **get** users

List all identities

oc **get** identities

List users with their identities

oc **get** useridentities

List all service accounts in all namespaces

oc **get** sa --all-namespaces

Get details of a specific user

oc **get** user <username> -o yaml

List groups

oc **get** groups

Show which users are in a group

oc **get** **group** <groupname> -o yaml

Check if the ServiceAccount exists

oc **get** sa -n kube-system | grep checkmk

oc get sa checkmk -n checkmk-monitoring

oc **get** sa checkmk -n kube-system

## Grant a User Permissions in a Project

**Give a user *admin* rights to a project**

oc adm policy **add**-role-to-user admin <username> -n <project>  
This gives full control over the project (create/delete resources, manage roles, etc.).

**Give a user *edit* rights**

oc adm policy **add**-role-to-user edit <username> -n <project>

  This allows modifying resources but not managing RBAC.

**Give a user *view* rights**

oc adm policy **add**-role-to-user view <username> -n <project>

This allows read‑only access.

**Assign Permissions to a Group**

oc adm policy **add**-role-to-**group** edit <groupname> -n <project>

**Remove Permissions**

oc adm policy **remove**-role-**from**-user edit <username> -n <project>

**Remove a role from a user:**

oc adm policy **remove**-role-**from**-user edit <username> -n <project>

**Remove a role from a group:**

oc adm policy **remove**-role-**from**-**group** view <groupname> -n <project>

**Check Permissions for a Role (Project‑Scoped Role)**

oc **get** role <role-name> -n <project> -o yaml

This shows all **rules**, including:

* allowed API groups
* allowed resources
* allowed verbs (get, list, create, delete, patch, etc.)

**Check Permissions for a ClusterRole (Cluster‑Wide Role)**

oc **get** clusterrole <role-name> -o yamls

oc **get** clusterrole admin -o yaml

oc **get** clusterrole edit -o yaml

oc **get** clusterrole view -o yaml

output:

rules:

- apiGroups: [""]

resources: ["pods", "services"]

verbs: ["get", "list", "watch", "create", "delete"]

**Show all roles assigned to your user in a specific namespace**

oc **get** rolebindings -n project-adrian --**no**-headers \

| grep a.pohancanik.adm

oc describe rolebinding -n project-adrian | grep -A5 a.pohancanik.adm

**Show all cluster‑wide roles assigned to your user**

oc **get** clusterrolebindings --**no**-headers \

| grep a.pohancanik.adm

 oc describe clusterrolebinding | grep -A5 a.pohancanik.adm

OpenShift Cluster administrators

**Create your own “full VM admin” role:**

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRole

metadata:

name: **virtualization-team-admin**

rules:

- apiGroups: ["kubevirt.io"]

resources: ["virtualmachines", "virtualmachineinstances", "virtualmachineinstancemigrations"]

verbs: ["\*"]

- apiGroups: ["subresources.kubevirt.io"]

resources: ["virtualmachines/start", "virtualmachines/stop", "virtualmachines/restart", "virtualmachines/migrate"]

verbs: ["update"]

- apiGroups: ["cdi.kubevirt.io"]

resources: ["datavolumes"]

verbs: ["\*"]

- apiGroups: [""]

resources: ["pods", "services", "persistentvolumeclaims"]

verbs: ["\*"]

**Apply it:**

oc apply -f full-vm-admin.yaml

**Bind it:**

oc adm policy **add**-cluster-role-to-user full-vm-admin a.pohancanik.adm

**OpenShift Virtualization ships with a built‑in ClusterRole:**

kubevirt.io:admin

This role includes permissions for:

* VM create/delete/update
* VM start/stop/restart
* VM live migration
* VM subresources
* VMI operations
* DataVolumes
* Migration resources

oc adm policy **add**-cluster-role-to-user **kubevirt.io:admin** a.pohancanik.adm

full VM control cluster‑wide.

Give yourself full OpenShift admin rights

oc adm policy add-cluster-role-to-user **cluster-admin** a.pohancanik.adm

## DN Certificate installation

In OpenShift, the certificate for the web console is not installed directly on the console - the console uses the ingress certificate. So if you want to have your own certificate for https://console-openshift-console.apps.<domain>, you need to replace the default ingress certificate.

Prepare your certificate

Certificate file: cert.crt (PEM format)

With private key: cert.key

2x Subject Alternative Name (SAN):

**console**-openshift-console.apps.dnpcpoc.test-dnpcloud.dieboldnixdorf.com

\*.apps.dnpcpoc.test-dnpcloud.dieboldnixdorf.com

**oauth**-openshift.apps.dnpcpoc.test-dnpcloud.dieboldnixdorf.com  
 \*.apps.<cluster-name>.<domain>

Upload certificate files to Jump, Bastion server

cd /tmp/Cert

Remove pw from key

openssl rsa -**in** console-openshift.apps.dnpcpoc.test-dnpcloud.dieboldnixdorf.com.key -**out** tls.key

Create full CA certificate chain:

cat server.crt intermediate.crt root.crt > tls.crt

cat console-openshift.apps.dnpcpoc.test-dnpcloud.dieboldnixdorf.com.crt diebold\_nixdorf\_issuing\_ca\_2.crt diebold\_nixdorf\_root\_ca.crt > tls-chain.crt

Uploads your certificate chain (tls\_chain.crt) and private key (tls.key) into the namespace openshift-ingress

oc create secret tls console-certificate \

--cert=tls-chain.crt \

--key=tls.key \

-n openshift-ingress

oc **set** data secret/custom-ingress-cert \

-n openshift-ingress \

--**from**-file=ca.crt=tls-chain.crt

Activate / Patch the new certificate on the IngressController (affects both OAuth and Console)

oc patch ingresscontroller **default** \

  -n openshift-ingress-**operator** \

  --type=merge \

  -p '{

    "spec": {

      "defaultCertificate": {

        "name": "console-certificate"

      }

    }

  }'

A screenshot of a browser

AI-generated content may be incorrect.

Verification:

oc **get** secret custom-ingress-cert -n openshift-ingress -o yaml

it should contain three items: tls.crt, tls.key, ca.crt ← this is the CA chain

Verify that the router pods reloaded the certificate

oc **get** pods -n openshift-ingress -o wide

* AGE is recent (you should see pods younger than the time you updated the cert)
* RESTARTS may be 0, but pod names should have changed (new rollout)

oc **get** clusteroperators authentication

If they did *not* restart, you can safely trigger it:  
oc **delete** pod -n openshift-ingress -l ingresscontroller.**operator**.openshift.io/deployment-ingresscontroller=**default**

Delete certificate: oc **delete** secret console-certificate -n openshift-ingress

**Install/Replace Certificate for the OpenShift API Server**

The API uses its own separate TLS certificate, not the Ingress one.

https://api.dnpcpoc.test-dnpcloud.dieboldnixdorf.com

Step 1 — Prepare your API certificate

It must include SANs:

api.<cluster-name>.<domain>

api-**int**.<cluster-name>.<domain>

Optional: load balancer IPs

Files needed:

api.crt

api.key

Step 2 — Create the API certificate secret

oc create secret tls custom-api-cert \

--cert=/path/to/api.crt \

--key=/path/to/api.key \

-n openshift-config

oc create secret tls api-custom-cert \

--cert=api-tls-chain.crt \

--key=apitls.key \

-n openshift-config \

--dry-run=client -o yaml | oc apply -f -

Step 3 — Patch the API server

oc patch apiserver cluster \

--type=merge -p \

'{"spec":{"servingCerts":{"namedCertificates":[{"names":["api.<cluster-domain>"],"servingCertificate":{"name":"custom-api-cert"}}]}}}'

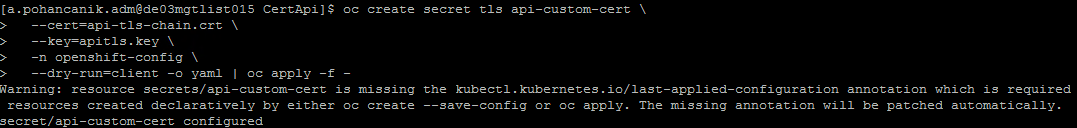
oc patch apiserver cluster \

--type=merge \

-p '{"spec":{"servingCerts":{"namedCertificates":[{"names":["api.dnpcpoc.test-dnpcloud.dieboldnixdorf.com"],"servingCertificate":{"name":"api-custom-cert"}}]}}}'

Step 4 — Verify

oc **get** apiserver cluster -o yaml



(Warning message – is OK)

**Your secret *was* updated successfully. “**secret/api-custom-cert configured”

Test certificate:

openssl s\_client -connect console-openshift-console.apps.dnpcpoc.test-dnpcloud.dieboldnixdorf.com:443 -showcerts

A screen shot of a computer

AI-generated content may be incorrect.

**Rollback Console, IngressController certificate – delete custom certificate**

oc patch ingresscontroller **default** -n openshift-ingress-**operator** --type=merge -p '{

"spec": {

"defaultCertificate": null

}

}'

**Rollback API Server certificate – delete custom certificate**

oc patch apiserver cluster --type=merge -p '{

"spec": {

"servingCerts": {

"namedCertificates": []

}

}

}'

**List all OpenShift TLS certificate expire date:**

echo -e "NAMESPACE\tNAME\tEXPIRY" && oc **get** secrets -A -o go-**template**='{{range .items}}{{if eq .type "kubernetes.io/tls"}}{{.metadata.namespace}}{{" "}}{{.metadata.name}}{{" "}}{{index .data "tls.crt"}}{{"\n"}}{{end}}{{end}}' | **while** read **namespace** name cert; **do** echo -en "$namespace\t$name\t"; echo $cert | base64 -d | openssl x509 -noout -enddate; **done** | column -t

hint: error: failed to create secret secrets "api-custom-cert" already exists

oc **delete** secret api-custom-cert -n openshift-config

or replace: \ --dry-run=client -o yaml | oc apply -f –

You See Only 2 Router Pods

By default, the cluster creates one router pod per worker node unless you override the replica count.

Check how many replicas are configured

oc **get** ingresscontroller **default** -n openshift-ingress-**operator** -o yaml | grep replicas -n

If you see:



How to scale router pods to 3

oc patch ingresscontroller **default** -n openshift-ingress-**operator** \

--type=merge \

-p '{"spec":{"replicas":3}}'

Then verify:

oc **get** pods -n openshift-ingress -o wide

## SSO AD Loging OpenID

Setup ActiveDirectory Authentication

Go to Administration -> Cluster Settings -> Global Configuration -> OAuth -> Add -> LDAP

bindDN: 'CN=svc-ldap-openshift,OU=Services,OU=\_DN,DC=test-dnpcloud,DC=dieboldnixdorf,DC=com'  
  
url: 'ldaps://de03mgtwdct003.test-dnpcloud.dieboldnixdorf.com:636//DC=test-dnpcloud,DC=dieboldnixdorf,DC=com?sAMAccountName?sub?(&(objectClass=person)(memberOf:1.2.840.113556.1.4.1941:=CN=A-I-AUTOMATION-ADM,OU=Access,OU=\_DN,DC=test-dnpcloud,DC=dieboldnixdorf,DC=com))'

## CPU / RAM size definition

**OpenShift Web Console**

Go to

Virtualization → VirtualMachines

Select the VM you want to modify.

Stop the VM

(CPU/RAM changes require the VM to be powered off unless you have hot-plug enabled in your cluster).

Click Configuration → Overview.

Edit:

CPU (cores, sockets, threads)

Memory (RAM size)

Save changes.

Start the VM again.

**Via YAML**

oc stop vm <vm-name> -n <namespace>

apiVersion: kubevirt.io/v1

kind: VirtualMachine

metadata:

  name: **my**-vm

spec:

**template**:

    spec:

      domain:

        cpu:

          cores: 4

          sockets: 1

          threads: 1

        resources:

          requests:

            memory: 8Gi

Apply changes

oc apply -f **my**-vm.yaml

Start the VM

oc start vm <vm-name> -n <namespace>

## High availability / Node Remediation

To make a VM "standard" for production, you must include a Live Migration policy in the spec. This ensures that if a worker node fails or needs an update, the VM definition allows the scheduler to move the workload.

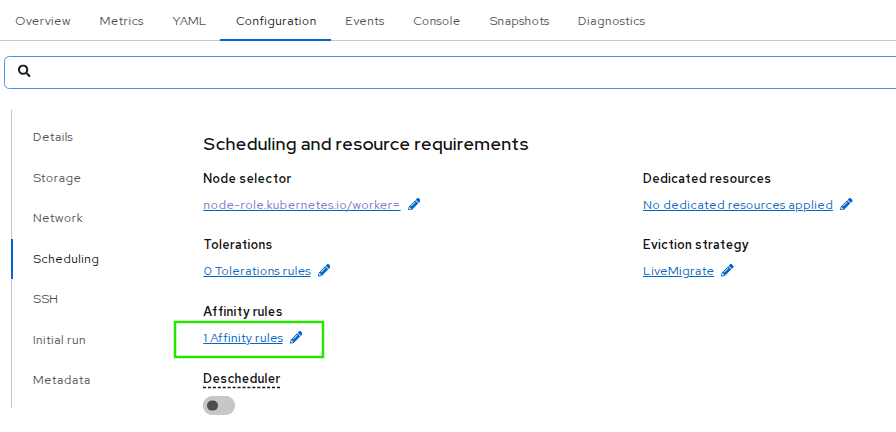
## DRS Descheduler Operator

|  |  |  |
| --- | --- | --- |
| **Feature** | **VMware DRS** | **OpenShift Descheduler** |
| Live migration | **Yes** | **No** |
| Continuous balancing | **Yes** | Periodic only |
| Works on | VMs | Pods |
| Enforces affinity rules | Yes | Yes |
| Moves workloads automatically | Yes | Evicts only (rescheduler places) |
| HA integration | Tight | Separate |

## VM Load Balancing

MetalLB LoadBalancer Service,

## Affinity rules



Pod Anti‑Affinity

Label all VMs with the same app label, app=SQL-VM) so anti‑affinity can match them.

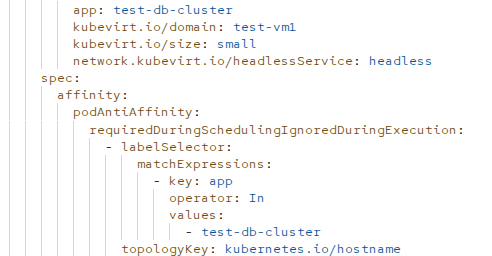
oc label virtualmachine **test-vm1** app=**SQL-DB** -n **project-test** --overwrite

oc label virtualmachine test-vm2 app=SQL-DB -n project-test --overwrite

oc label virtualmachine test-vm3 app=SQL-DB -n project-test --overwrite

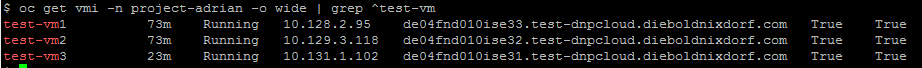
oc label virtualmachine test-vm4 app=SQL-DB -n project-test --overwrite

Add this under spec.template.spec of each VirtualMachine manifest. It prefers to avoid colocating VMs on the same node but allows scheduling if necessary.



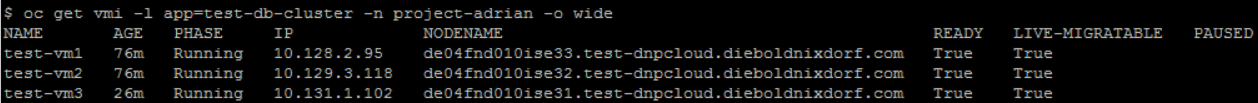
Check affinity rules: where each VM (VMI) is running on worker nodes

oc **get** vmi -n project-test -o wide



Or filter by label used in affinity:

oc **get** vmi -l app=test-db-cluster -n project-adrian -o wide



oc **get** pods -n project-adrian -o wide

kubectl **get** pod -n project-test -l kubevirt.io=virt-launcher -o wide

oc patch virtualmachine adrian-test-vm2 -n project-adrian --type=json \

-p '[{"op":"add","path":"/spec/template/spec/nodeSelector","value":{"kubevirt-node":"usnc1ciocw4lpd.ad.diebold.com"}}]'

**Node Selector** – move and keep VM on the specified Worker Node

Label your nodes first

kubectl label node node1 disktype=ssd

Verify labels:

kubectl **get** nodes --show-labels

Apply nodeSelector in your Pod/Deployment manifest

spec:

  nodeSelector:

    disktype: ssd

**When to Use Node Selector:**

✔ Recommended when:

* You have dedicated nodes for a specific workload (GPU nodes, logging nodes, DB nodes).
* You want a strict placement rule.
* You have a small or static cluster.

✖ Avoid nodeSelector when:

* You need HA (High Availability).
* You want spreading across nodes.
* You want soft preferences, not strict rules.
* You use autoscaling clusters (new nodes may not have your labels).

oc patch virtualmachine adrian-test-vm2 -n project-adrian --type=json \

-p '[{"op":"add","path":"/spec/template/spec/nodeSelector","value":{"kubevirt-node":"usnc1ciocw4lpd.ad.diebold.com"}}]'

If the nodeSelector already exists, use replace:

oc patch virtualmachine adrian-test-vm2 -n project-adrian --type=json \

-p '[{"op":"replace","path":"/spec/template/spec/nodeSelector","value":{"kubevirt-node":"usnc1ciocw4lpd.ad.diebold.com"}}]'

Check where the VMI is running:

oc **get** vmi test-vm2 -n project-test -o wide

Confirm VMI is live migratable   
oc **get** vmi adrian-test-vm2 -n project-adrian -o jsonpath='{.status.conditions}' | jq .

# look for condition type: LiveMigratable status: "True"

## Storage definitions

PVC and PV are deployed as per definition configured in Storage classes (SC).

Netapp

CSI (Container Storage Interface) is an industry-standard API that allows Kubernetes (OpenShift) to connect to any block or file storage system using plug‑ins—without modifying Kubernetes core code.

NetApp Trident is fully supported dynamic storage orchestrator for Kubernetes, including Red Hat OpenShift.

Netapp trident CSI and Netapp trident protect need to be installed.

**Trident Installation**: -

* Log in to OpenShift Console.
* Navigate to:
* Operators → OperatorHub
* Search for "Trident" (from NetApp).
* Select NetApp Trident Operator (CSI storage orchestrator).
* Click Install
* After this is completed, create a Trident Orchestrator Custom resource using YAML

apiVersion: trident.netapp.io/v1  
 kind: TridentOrchestrator  
 metadata:  
 name: trident  
 namespace: openshift-operators  
 spec:  
 IPv6: false  
 debug: false  
 nodePrep:  
 - iscsi  
 imageRegistry: ''  
 k8sTimeout: 30  
 namespace: trident  
 silenceAutosupport: false

Installing Trident CLI:

* Download the package using “wget <https://github.com/NetApp/trident/releases/download/v25.10.0/trident-installer-25.10.0.tar.gz>”

* Extract the package

tar -xf trident-installer-25.10.0.tar.gz

cd trident-installer/

* Copy the tridentctl to /usr/local/bin

Connecting Trident to NetApp:-

* After installing of Trident and Tridentctl. Create a Trident backend file. Below is an example.

---

version: 1  
 storageDriverName: ontap-nas  
 backendName: ssd2-netapp-nfs  
 managementLIF: 10.29.176.25  
 dataLIF: 10.29.83.88  
 svm: ushasvmf02  
 autoExportCIDRs:  
 - 10.29.85.0/24  
 autoExportPolicy: true  
 username: vsadmin  
 password: xxxxxxxx  
 storagePrefix: trident\_nfs  
 aggregate: aggrSSD1\_HACL01\_01  
 limitVolumeSize: 300Gi  
 nfsMountOptions: nfsvers=4  
 storage:  
 - labels:  
 performance: gold  
 defaults:  
 adaptiveQosPolicy: gold-ushasvmf02  
 - labels:  
 performance: silver  
 defaults:  
 adaptiveQosPolicy: silver-ushasvmf02  
 - labels:  
 performance: bronze  
 defaults:  
 adaptiveQosPolicy: bronze-ushasvmf02  
 - labels:  
 performance: goldburst  
 defaults:  
 adaptiveQosPolicy: goldburst-ushasvmf02

* Apply the backend to Trident using tridentctl

*tridentctl create backend -f backend.yaml -n trident*

* Verify the connectivity using below command

*tridentctl -n trident get backend*

Apply yaml files as follow:

apiVersion: storage.k8s.io/v1  
 kind: StorageClass  
 metadata:  
 name: **SC-bronze**

provisioner: csi.trident.netapp.io  
 reclaimPolicy: Delete  
 parameters:  
 selector: "performance=bronze"  
 allowVolumeExpansion: true  
 volumeBindingMode: Immediate  
 ---

apiVersion: storage.k8s.io/v1  
 kind: StorageClass  
 metadata:  
 name: **SC-silver**  
 annotations:  
 storageclass.kubernetes.io/is-default-class: "true"  
 provisioner: csi.trident.netapp.io  
 reclaimPolicy: Delete  
 parameters:  
 selector: "performance=silver"  
 allowVolumeExpansion: true  
 volumeBindingMode: Immediate  
 ---

apiVersion: storage.k8s.io/v1  
 kind: StorageClass  
 metadata:  
 name: **SC-gold**  
 provisioner: csi.trident.netapp.io  
 reclaimPolicy: Delete  
 parameters:  
 selector: "performance=silver"  
 allowVolumeExpansion: true  
 volumeBindingMode: Immediate  
 ---

* Now to apply the yaml use below command

*oc create -f storageclass.yaml*

Powerflex

Dell CSM needs to be installed.

IOPS value is defined in storage classes.  
DR relationships also defined SC therefore following SC needs to be created as follow:  
  
**SC-bronze**

**SC-silver**

**SC-gold**

**SC-silver-dr**

**SC-gold-dr**

## DR configuration

Netapp

Each component which supposed to be part of DR mirroring needs to be properly labeled.

This include VM/Container object and also its PVC/PV.   
  
**Installing and configuring Trident-Protect**

* Install Trident-Protect via helm.
* Add Trident-Protect repo using “*helm repo add netapp-trident-protect* [*https://netapp.github.io/trident-protect-helm-chart*](https://netapp.github.io/trident-protect-helm-chart)”.
* Install Trident-Protect using below command. Replace name of the cluster with cluster name.

*helm install trident-protect netapp-trident-protect/trident-protect --set clusterName=<name-of-cluster> --version 100.2510.0 --create-namespace --namespace trident-protect*

* Download the trident-protect ctl using

*curl -L -o tridentctl-protect* [*https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-linux-amd64*](https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-linux-amd64)

* Enable execute permissions for the binary

*chmod +x tridentctl-protect*

* *Copy the plugin binary to a location /usr/local/bin*
* Copy the downloaded tridentctl-protect binary to /usr/local/bin.
* Create an appvault. AppVault CR contains the full configuration of the bucket (ONTAP S3) so Trident Protect knows where to store application backups and snapshots.

Create a secret with S3 bucket accesskeyID and SecretAccesskey using below yaml. And apply the yaml to Trident-protect namespace using *oc apply -f secret.yaml -n trident-protect.*

apiVersion: v1

kind: Secret

metadata:

name: netapp-s3-credentials

namespace: trident-protect

type: Opaque

stringData:

accessKeyID: "<YOUR\_S3\_ACCESS\_KEY>"

secretAccessKey: "<YOUR\_S3\_SECRET\_KEY>"

* Then create the Appvault using below yaml and apply the yaml to trident-protect namespace using command *oc apply -f appvault.yaml -n trident-protect*

apiVersion: protect.trident.netapp.io/v1

kind: AppVault

metadata:

name: netapp-s3-appvault

namespace: trident-protect

spec:

provider: s3

s3:

bucketName: "<YOUR\_BUCKET\_NAME>"

endpoint: "https://<YOUR\_S3\_ENDPOINT>"

region: "<YOUR\_REGION>"

secure: "false"  
 skipCertValidation: "false"

credentials:

accessKeyIDSecret:

name: netapp-s3-credentials

key: accessKeyID

secretAccessKeySecret:

name: netapp-s3-credentials

key: secretAccessKey

* Verify the appvault using *oc get appvault -n trident-protect.*
* Create snapshot volume class using below yaml. Apply the snapshot volume class to trident-protect namespace using command oc apply -f snapshotvolumeclass.yaml

apiVersion: snapshot.storage.k8s.io/v1  
 kind: VolumeSnapshotClass  
 metadata:  
 name: trident-snapshotclass  
 driver: csi.trident.netapp.io  
 deletionPolicy: Delete

Setup DR mirroring in Trident protect:

## Backup of VMs/Containers

This part defined how to configure regular backup of the VMs/containers as defined per service description.

Netapp

Backup is done in following setup used byt netapp Trident protect (TP)  
snapshot on primary storage – each 4 hours, stored 12 snapshots (two days)

Backup to S3 – each midnight, stored for 21 days ,Full backup on Sunday  
  
**TP configuration: see 5.14**

**Snapshot configuration with TP:**

* Create an app with below yaml which will have the namespace details for which the snapshot needs to be taken. This app will be used while taking snapshots and backups. Apply the yaml to create the namespace for which snapshot needs to be taken using command *oc apply -f application.yaml -n my-app-namespace* .

apiVersion: protect.trident.netapp.io/v1  
 kind: Application  
 metadata:  
 creationTimestamp: null  
 name: demo-app  
 namespace: my-app-namespace(on which the snapshot needs to be taken)  
 spec:  
 includedNamespaces:  
 - namespace: my-app-namespace(on which the snapshot needs to be taken)  
 resourceFilter: {}  
 status:  
 conditions: null

Create a yaml as below with details of the namespace on which the snapshot needs to be triggered along with appvault details. Apply the yaml to create the snapshot on namespace where backup needs to be triggered using command oc apply -f snapshot.yaml -n my-app-namespace.

apiVersion: protect.trident.netapp.io/v1  
 kind: Schedule  
 metadata:  
 creationTimestamp: null  
 name: snapshot-schedule1  
 namespace: my-app-namespace  
 spec:  
 appVaultRef: netapp-s3-appvault  
 applicationRef: app01  
 backupRetention: "0"  
 dayOfMonth: ""  
 dayOfWeek: ""  
 enabled: true  
 granularity: Hourly  
 hour: ""  
 minute: "50"  
 recurrenceRule: ""  
 snapshotRetention: "1"  
 status: {}

* Now to check the status of the snapshot use the command given below

*tridentctl-protect get snapshot -n my-app-namespace*

* This concludes the snapshot based backup.

**Snapshot Inplace restore (SIR)**

* Create below yaml to perform snapshot inplace restore and apply the yaml under same namespace.

tp create sir restore-test02 --snapshot test02/hourly-e05f1-20260130095000 -n test02 --dry-run>app01-sir.yaml

apiVersion: protect.trident.netapp.io/v1  
 kind: SnapshotInplaceRestore  
 metadata:  
 creationTimestamp: null  
 name: restore-test02  
 namespace: test02  
 spec:  
 appArchivePath: app01\_b6fcb4ec-2eed-48dd-83c0-fa2d32809f46/snapshots/20260130095000\_hourly-e05f1-20260130095000\_0333326e-302c-4666-8102-97c7389bcefa  
 appVaultRef: appvault-for-eng-user-only  
 resourceFilter: {}  
 status:  
 conditions: null  
 postRestoreExecHooksRunResults: null  
 state: ""

Now to check snapshot restore status use below commands

*oc get snapshotrestores -n my-app-namespace*

*oc describe snapshotrestore restore-from-snapshot-inplace -n my-app-namespace*

**Snapshot restore (SR)**

Run the below command to restore to different namespace

# tridentctl-protect create sr snaprestore01 --snapshot my-app-namespace /hourly-e05f1-20260130095000 --namespace-mapping my-app-namespace:destination-namespace -n my-app-namespace

Now to check snapshot restore status use below commands

*tp get snapshotrestores -n my-app-namespace*

*tp describe snapshotrestore restore-from-snapshot-inplace -n my-app-namespace*

Or can use playbook:

tp create sr snaprestore01 --snapshot test02/hourly-e05f1-20260130095000 --namespace-mapping test02:test01 -n test02 --dry-run>test02-sr.yaml

apiVersion: protect.trident.netapp.io/v1  
 kind: SnapshotRestore  
 metadata:  
 creationTimestamp: null  
 name: snaprestore01  
 namespace: test02  
 spec:  
 appArchivePath: app01\_b6fcb4ec-2eed-48dd-83c0-fa2d32809f46/snapshots/20260130095000\_hourly-e05f1-20260130095000\_0333326e-302c-4666-8102-97c7389bcefa  
 appVaultRef: appvault-for-eng-user-only  
 namespaceMapping:  
 - destination: test01  
 source: test02  
 resourceFilter: {}  
 status:  
 postRestoreExecHooksRunResults: null  
 state: ""

**Backup configuration with TP:**

* Create an app with below yaml which will have the namespace details for which the snapshot needs to be taken. This app will be used while taking snapshots and backups. Apply the yaml to create the namespace for which snapshot needs to be taken using command *oc apply -f application.yaml -n my-app-namespace* .

apiVersion: protect.trident.netapp.io/v1  
 kind: Application  
 metadata:  
 creationTimestamp: null  
 name: demo-app  
 namespace: my-app-namespace(on which the snapshot needs to be taken)  
 spec:  
 includedNamespaces:  
 - namespace: my-app-namespace(on which the snapshot needs to be taken)  
 resourceFilter: {}  
 status:  
 conditions: null

* Use below yaml to schedule backups and apply the yaml to the namespace where backup needs to be taken using command oc apply -f backupschedule.yaml -n my-app-namespace.

apiVersion: protect.trident.netapp.io/v1  
 kind: Schedule  
 metadata:  
 creationTimestamp: null  
 name: backup-schedule1  
 namespace: my-app-namespace  
 spec:  
 appVaultRef: netapp-s3-appvault  
 applicationRef: app01  
 backupRetention: "1"  
 dayOfMonth: ""  
 dayOfWeek: ""  
 enabled: true  
 granularity: Hourly  
 hour: ""  
 minute: "55"  
 recurrenceRule: ""  
 snapshotRetention: "0"  
 status: {}

* Now to verify the backups run the command

Tridentctl-protect get backup -n my-app-namespace

**Backup inplace restore**

Run the below command using trident-protect ctl to restore within the same namespace or use playbook.

tridentctl-protect create bir restore01 --backup my-app-namespace/name-of-the-backup -n my-app-namespace --dry-run>backup-demo-bir.yaml

apiVersion: protect.trident.netapp.io/v1  
 kind: BackupInplaceRestore  
 metadata:  
 annotations:  
 protect.trident.netapp.io/max-parallel-restore-jobs: "25"  
 creationTimestamp: null  
 name: restore01  
 namespace: test2  
 spec:  
 appArchivePath: app01\_194f0f7f-b7a6-4968-802d-7b044fbab3e6/backups/hourly-b411d-20260130101000\_6564bb98-a170-48d4-b2bb-1713f2d9327a  
 appVaultRef: appvault-for-eng-user-only  
 resourceFilter: {}  
 status:  
 postRestoreExecHooksRunResults: null  
 state: ""

* Run below command or playbook to restore a backup onto different namespace

tp create br restore01 --backup test01/hourly-b411d-20260130101000 --namespace-mapping test01:test02 -n test02 --dry-run>backup-demo2-br.yaml

apiVersion: protect.trident.netapp.io/v1  
 kind: BackupRestore  
 metadata:  
 annotations:  
 protect.trident.netapp.io/max-parallel-restore-jobs: "25"  
 creationTimestamp: null  
 name: restore01  
 namespace: test02  
 spec:  
 appArchivePath: app01\_194f0f7f-b7a6-4968-802d-7b044fbab3e6/backups/hourly-b411d-20260130101000\_6564bb98-a170-48d4-b2bb-1713f2d9327a  
 appVaultRef: appvault-for-eng-user-only  
 namespaceMapping:  
 - destination: test02  
 source: test01  
 resourceFilter: {}  
 status:  
 conditions: null  
 postRestoreExecHooksRunResults: null  
 state: ""

**Disaster recovery**

Use same app and snapshot for appmirrorrelationship (AMR) to restore complete namespace

In source OCP cluster

tp get app –n namespace –o json|grep uid # same uid use while creating amr at target OCP cluster

In target OCP cluster

tridentctl-protect create appmirrorrelationship amr03 --destination-app-vault appvault-for-eng-user-only --source-app-vault appvault-for-eng-user-only --recurrence-rule "DTSTART:20220101T000200Z\nRRULE:FREQ=MINUTELY;INTERVAL=5" --source-app protect-demo01 --namespace-mapping "project03:project03-dr3" --source-app-id ce2ef95a-07df-4e4e-bde1-66e27bedf64a --storage-class netapp-nfs-silver -n project03-dr3

To check AMR status

tp get amr -n namespace

Can promote once it is in Established state

oc patch amr amr03 -n project03-dr3 --type=merge -p '{"spec":{"desiredState":"Promoted"}}'

to restablish amr

oc patch amr amr03 -n project03-dr3 --type=merge -p '{"spec":{"desiredState":"Established"}}'

Note: this will overwrite data in target namespace

**Disaster recovery for selective VMs using labels**

At source OCP cluster label to required resources:

oc label deployment appnginx01 category=protect-demo01 -n project03

oc label deployment appnginx02 category=protect-demo01 -n project03

oc label pvc pvc01 category=protect-demo01 -n project03

oc label pvc pvc02 category=protect-demo01 -n project03

oc get deployment,pvc,vm -n project03 -l category=protect-demo01

oc label vm centos-stream9-tomato-lizard-78 category=protect-demo01 -n project03

oc label pvc centos-stream9-tomato-lizard-78-volume category=protect-demo01 -n project03

oc label vm oraclelinux-azure-bonobo-39 category=protect-demo01 -n project03

oc label pvc dv-oraclelinux-azure-bonobo-39-disk-indigo-carp-31-rw6fmb category=protect-demo01 -n project03

oc label pvc oraclelinux-azure-bonobo-39-volume category=protect-demo01 -n project03

tp create app protect-demo01 --namespaces 'project03(category=protect-demo01)' -n project03

tp get app protect-demo01 –o json|grep –i uid

tridentctl-protect create schedule --name appmirror-schedule1 --app protect-demo01 --appvault appvault-for-eng-user-only --granularity Custom --recurrence-rule "DTSTART:20260129T000200Z\nRRULE:FREQ=MINUTELY;INTERVAL=5" --snapshot-retention 2 -n project03

**At target OCP cluster**

tridentctl-protect create appmirrorrelationship amr03 --destination-app-vault appvault-for-eng-user-only --source-app-vault appvault-for-eng-user-only --recurrence-rule "DTSTART:20220101T000200Z\nRRULE:FREQ=MINUTELY;INTERVAL=5" --source-app protect-demo01 --namespace-mapping "project03:project03-dr3" --source-app-id ce2ef95a-07df-4e4e-bde1-66e27bedf64a --storage-class netapp-nfs-silver -n project03-dr3

after amr established you can change it to prmote

oc patch amr amr03 -n project03-dr3 --type=merge -p '{"spec":{"desiredState":"Promoted"}}'

To Re-Establish

Note: this will overwrite data in target namespace

oc patch amr amr03 -n project03-dr3 --type=merge -p '{"spec":{"desiredState":"Established"}}'

Powerflex

TBD (PPDM)

## Creating and managing Linux and Windows virtual machines (VMs)

1. Create VM in GUI
2. From yaml file

Create yaml file: in notepad fedora-vm.yaml or nano fedora-vm.yaml

oc apply -f fedora-vm.yaml

## Running Guest Agent

Installation of QEMU guest agent inside each VM is necessary (alt. VMware tools)

Provides:

* **Guest runtime information**  
  Provides OS details, running processes, and network interfaces so the host can display accurate guest state in the VMI status.
* **Network information reporting**  
  The agent reports guest IP addresses and interface details (merged with QEMU interfaces), enabling easier discovery and tooling that needs guest networking info
* **Clean shutdown and reboot**The host can request a graceful shutdown or reboot via the agent rather than forcing power-off, reducing filesystem corruption risk.
* **Filesystem freeze and thaw**Supports freezing filesystems for consistent snapshots and backups, then thawing them after the operation completes. This is essential for safe snapshotting and live backup workflows.
* **File transfer and command execution**  
  Supports freezing filesystems for consistent snapshots and backups, then thawing them after the operation completes. This is essential for safe snapshotting and live backup workflows.
* **Time synchronization and user management**  
  Can set guest system time and perform user/password operations.

Installation:  
sudo dnf install -y qemu-guest-agent   
sudo systemctl enable --now qemu-guest-agent

Verify service:  
systemctl status qemu-guest-agent

Check VMI status for AgentConnected: "True"

oc **get** vmi <vmi-name> -n <ns> -o yaml

## Running POD and VM workloads alongside each other in a cluster

**1.** OpenShift supports running Pods and VMs together through **OpenShift Virtualization**, a built‑in virtualization layer powered by **KubeVirt**. This enables:

* Running Linux and Windows VMs alongside containers
* Migrating legacy workloads into Kubernetes
* Sharing cluster networking, storage, and security
* Using the same operational model for both VMs and Pods
* Consolidating infrastructure and reducing operational overhead

VMs run as **Kubernetes objects**, scheduled by the same control plane that manages Pods.

**2. Architecture Summary** - Core Components

KubeVirt – virtualization layer integrated into Kubernetes

virt‑operator – manages lifecycle of virtualization components

virt‑controller – handles VM scheduling and orchestration

virt‑handler – runs on nodes that host VMs

virt‑launcher – pod wrapper that runs each VM

OpenShift Networking – provides unified networking for Pods and VMs

OpenShift Storage – provides persistent volumes for VM disks

**3. Node Requirements**

Nodes that run VMs must have:

Hardware virtualization support (Intel VT‑x or AMD‑V)

kubevirt.io/schedulable=**true** label

kubevirt.io/virt-handler DaemonSet running

Pods can run on any worker node; VMs run only on nodes that meet virtualization requirements.

**4. Enabling OpenShift Virtualization**

Install the OpenShift Virtualization Operator from OperatorHub.

Create a HyperConverged resource to deploy virtualization components.

Verify virtualization components:

oc **get** pods -n openshift-cnv

Label nodes that should run VMs:

oc label node <node-name> kubevirt.io/schedulable=**true**

**5. Running Pods and VMs Together**

* Pods - Pods run normally using Deployments, StatefulSets, DaemonSets, or Jobs.

Example: oc apply -f my-deployment.yaml

* Virtual Machines - VMs are defined using the VirtualMachine CRD.

Example: oc apply -f my-vm.yaml

VMs run inside **virt‑launcher pods**, so they behave like Kubernetes workloads.

**6. Networking for Pods and VMs**

OpenShift provides unified networking:

* Pods use the default SDN/OVN‑Kubernetes network
* VMs can use:
  + Pod network (default)
  + Multus secondary networks
  + SR‑IOV for high‑performance workloads

VMs get a Pod‑like IP and can communicate with Pods natively.

**7.** **Storage for Pods and VMs**

Pods typically use:

* PVCs
* ConfigMaps
* Secrets

VMs use:

* PVCs for VM disks
* DataVolumes for image import
* Live migration requires **shared storage** (e.g., RWX volumes)

**8. Security Considerations**

VMs run inside pods but with additional isolation (KVM)

SELinux and cgroups apply to both Pods and VMs

NetworkPolicies can restrict traffic for both

RBAC controls access to VM lifecycle operations

**9. Monitoring and Observability**

Both Pods and VMs integrate with OpenShift monitoring:

* Prometheus scrapes VM metrics via KubeVirt exporters
* VM lifecycle events appear in cluster logs
* VM performance metrics include CPU, memory, disk, and network

**10. Live Migration and High Availability**

VMs support:

* Live migration between nodes
* Eviction handling during node maintenance
* High availability using VM replicas or anti‑affinity rules

Pods support:

* Rolling updates
* Horizontal scaling
* Pod disruption budgets

This allows mixed workloads to be resilient and maintain uptime.

## Connecting to virtual machines through a variety of consoles and CLI tools

In OpenShift Virtualization, you can connect to VMs using:

* Web console (graphical + serial consoles)
* Remote desktop protocols, RDP for Windows
* SSH from your terminal
* Virtctl CLI
* kubectl/oc CLI (port-forward, logs, etc.)

**Virtctl CLI**

Serial Console via virtctl

virtctl console <vm-name>

VNC Console via virtctl

virtctl vnc <vm-name>

**SSH Access to VM**

ssh user@<service-ip> -p <port>

**RDP for Windows VMs**

mstsc /v:<hostname-or-ip>

|  |  |  |
| --- | --- | --- |
| **Method** | **Type** | **Best For** |
| Web Console – Serial | Text console | Linux shell, rescue mode |
| Web Console – VNC | GUI | Windows/Linux desktops, installers |
| virtctl console | Text console | Fast CLI access |
| virtctl vnc | GUI | Native graphical viewer |
| SSH | Text remote shell | Production access |
| RDP | Windows GUI | Windows VMs |
| oc port-forward | SSH tunneling | Access without exposing services |

## Importing and cloning existing virtual machines

Import a VMware VM ova

1. Go to Virtualization → VirtualMachines
2. Click Create → Import Virtual Machine
3. Choose **VMware**
4. Provide:

* vCenter hostname
* Credentials
* VM to import
* Target storage & network

Clone a VM Using the Web Console

1. Go to Virtualization → VirtualMachines
2. Select the VM
3. Choose Actions → Clone
4. Provide:

* New VM name
* Namespace
* Whether to start after creation

## Additional network interface controllers and storage disks attached to virtual machine

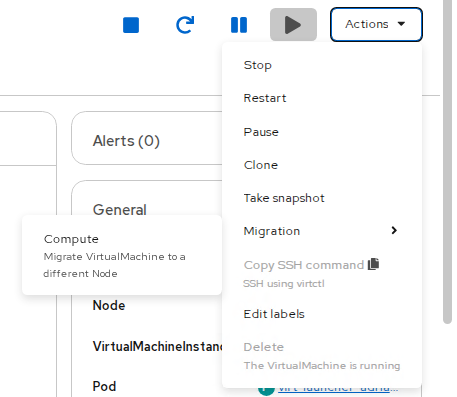
## Live migrating virtual machines between nodes / DR

**What live migration allows:**

* Move a running VM to another node without shutting it down
* Evacuate a node for maintenance
* Automatically migrate VMs when a node becomes unschedulable
* Enforce affinity/anti‑affinity rules
* Balance workloads manually or via automation

Migrate a VM using the OpenShift Web Console

1. Open Virtualization → VirtualMachines
2. Select the VM you want to migrate
3. Open the Actions menu
4. Click Migrate
5. The migration starts immediately and the VM will move to another suitable node



* The VM must be Running
* Storage must support migration (RWX or CSI‑supported block)
* The scheduler chooses the target node automatically

**Migrate a VM using the CLI (oc)**

List your VMs

oc **get** vms -n <namespace>

Trigger a live migration

oc create -n <namespace> -f - <<EOF

apiVersion: kubevirt.io/v1

kind: VirtualMachineInstanceMigration

metadata:

name: migrate-<vm-name>

spec:

vmiName: <vm-name>

EOF

Watch migration progress

oc **get** vmim -n <namespace>

**Check storage calls - PersistentVolume created from that class**

List all storage classes

oc **get** storageclass

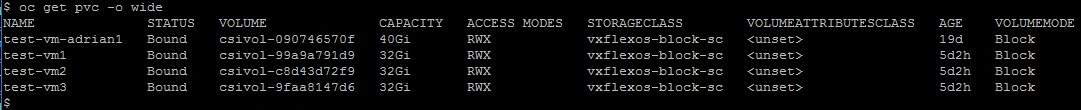
Check which ones support RWX

oc **get** sc -o jsonpath='{range .items[\*]}{.metadata.name}{" → "}{.provisioner}{"\n"}{end}'

List PVCs

oc get pvc

oc get pvc -o wide



**Verify RWX from PVC**

oc **get** pvc test-rwx

If it becomes Bound, that storage class supports RWX.

If it stays Pending, it does not support RWX.

**Check RWX support**

oc **get** pvc test-vm1 -o yaml

Look for:

accessModes:

- ReadWriteMany

## Moving a Virtual Machine Between Projects

OpenShift Virtualization (KubeVirt) does not support a direct “move” operation for VirtualMachines between namespaces. A VM is tightly coupled to its namespace because:

* its **VM definition** lives in the namespace
* its **DataVolumes / PVCs** live in the namespace
* its **network attachments** and **service accounts** may be namespace‑scoped

To move a VM, you must **export → recreate → import** it into the target project.

**1. Overview of the Migration Process**

Moving a VM between projects involves:

1. Exporting the VM definition (YAML)
2. Exporting the VM’s disks (DataVolumes or PVCs)
3. Creating the target project (if needed)
4. Importing the disks into the new project
5. Adjusting the VM YAML for the new namespace
6. Creating and starting the VM in the new project

This method is fully supported and avoids corruption or data loss.

**2. Identify VM Disks**

List the VM:

oc **get** vm <vm-name> -n <source-project> -o yaml | grep -A5 volumes

You may see:

* dataVolume: → disk managed by CDI
* persistentVolumeClaim: → raw PVC
* containerDisk: → no migration needed (comes from an image)

**3. Export the VM Definition**

oc **get** vm <vm-name> -n <source-project> -o yaml > vm.yaml

Edit the file:

* Remove the entire status: section
* Remove fields like uid, resourceVersion, creationTimestamp

Change:

metadata:

**namespace**: <**new**-project>

**4. Export the VM Disks**

oc **get** dv <dv-name> -n <source-project> -o yaml > dv.yaml

Edit:

* Remove status:
* Remove uid, resourceVersion, creationTimestamp

Change namespace to the new project

**If the VM uses PVCs:**

oc **get** pvc <pvc-name> -n <source-project> -o yaml > pvc.yaml

  Edit the same fields as above.

**5. Create the Target Project** (if needed)

oc **new**-project <**new**-project>

**6. Import the Disks into the New Project**

DataVolume:

oc apply -f dv.yaml

Wait for the import to complete:

oc get dv -n <new-project>

PVC:

oc apply -f pvc.yaml

**7. Import the VM Definition**

oc apply -f vm.yaml

**8. Start the VM in the New Project**

oc start vm <vm-name> -n <**new**-project>

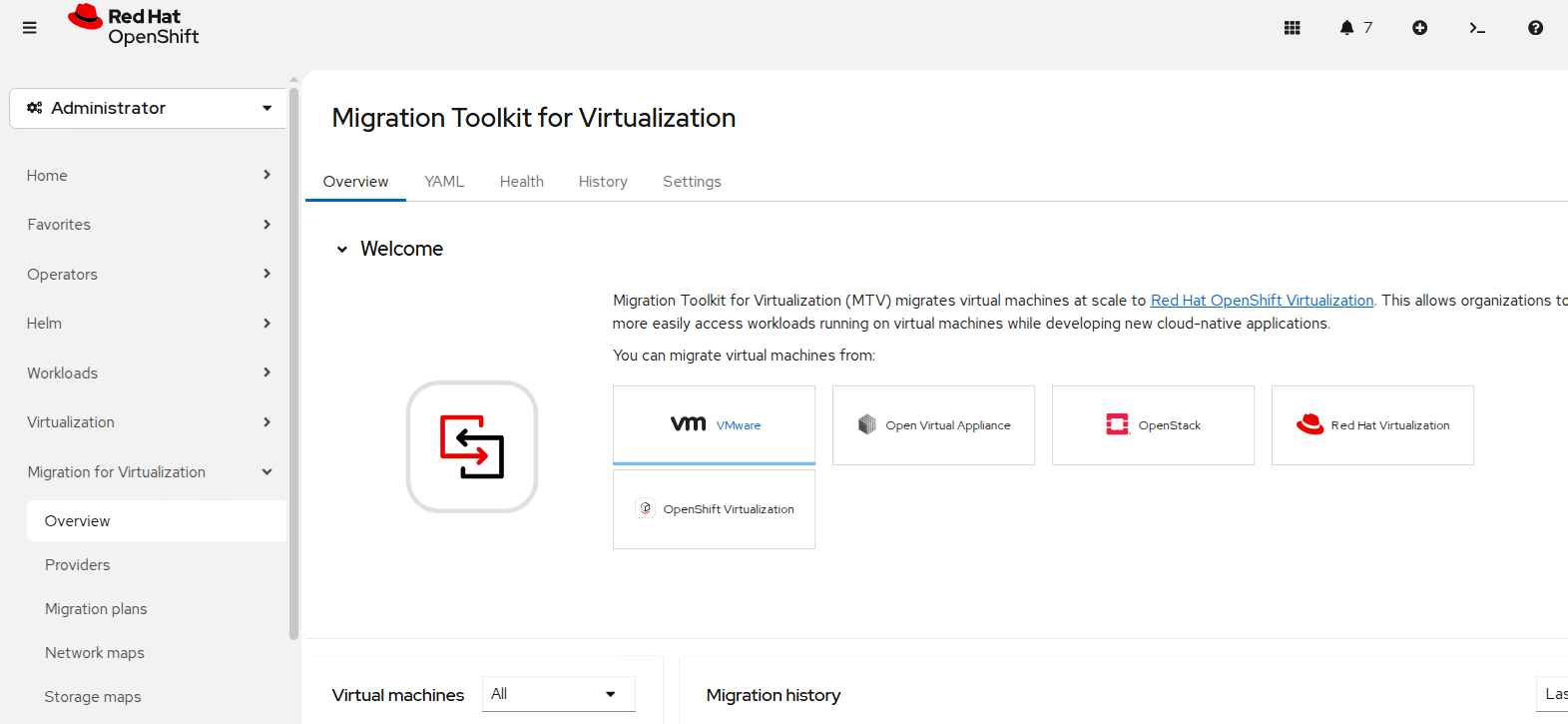
Verify

oc **get** vmi -n <**new**-project>

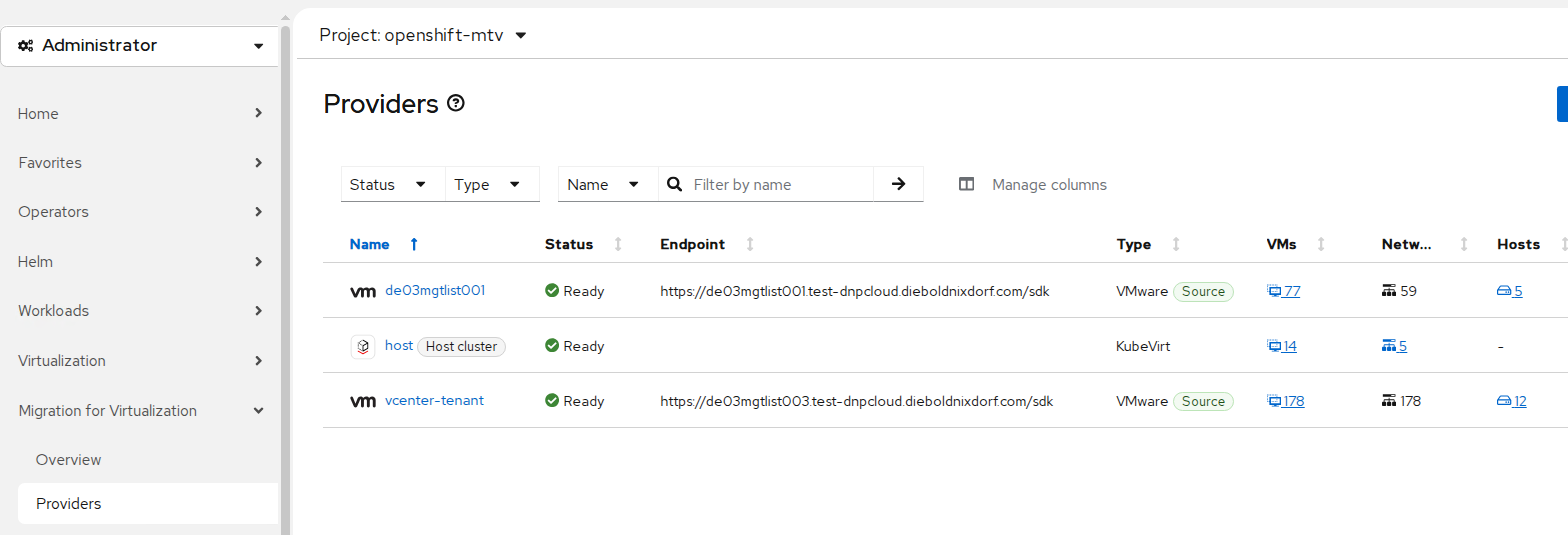
## VM Migration from VMware to OpenShift

Required tool: Migration Toolkit for Virtualization (MTV)

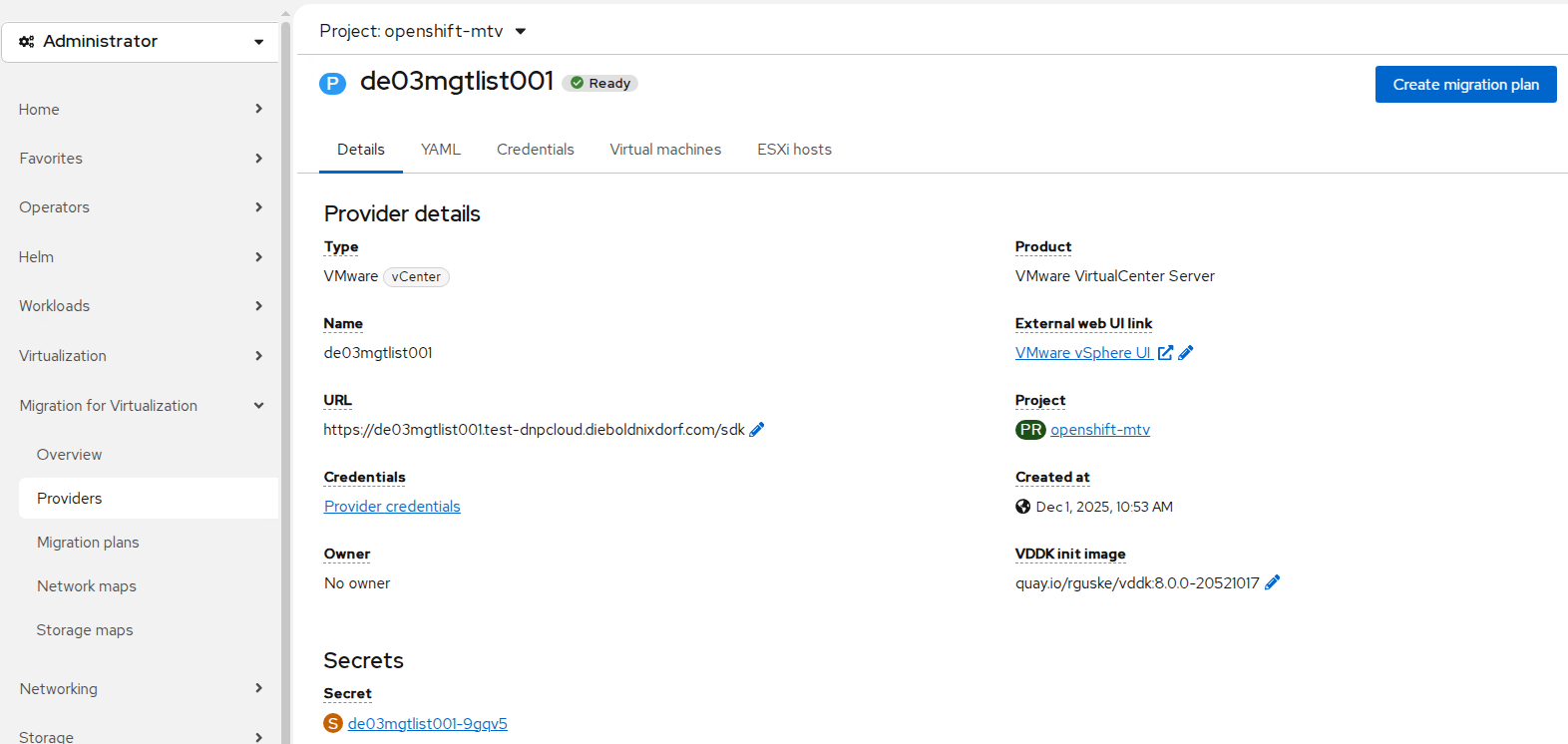
1. In OpenShift Web Console, go to Migration → Providers and configure the VMware provider (vCenter or ESXi endpoint).

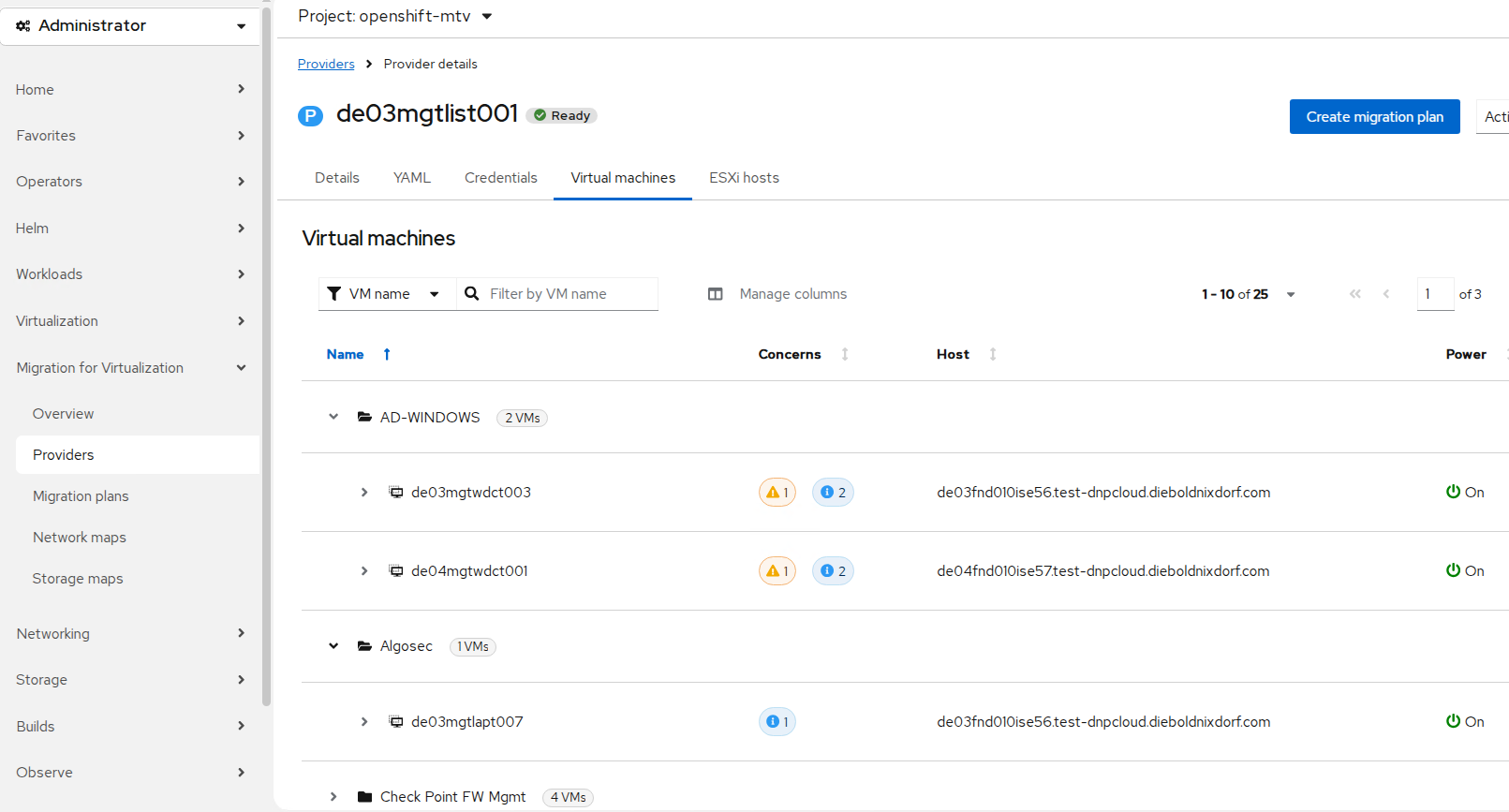


1. Define Source Providers:

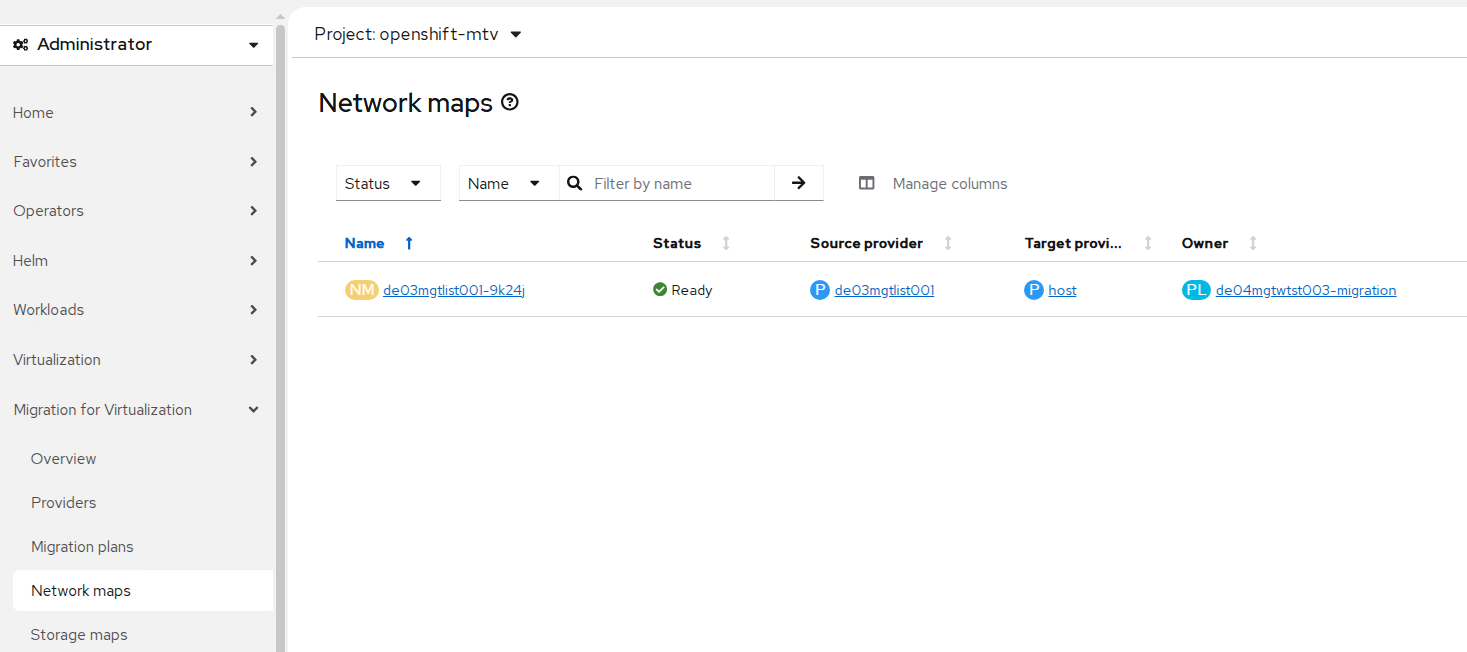


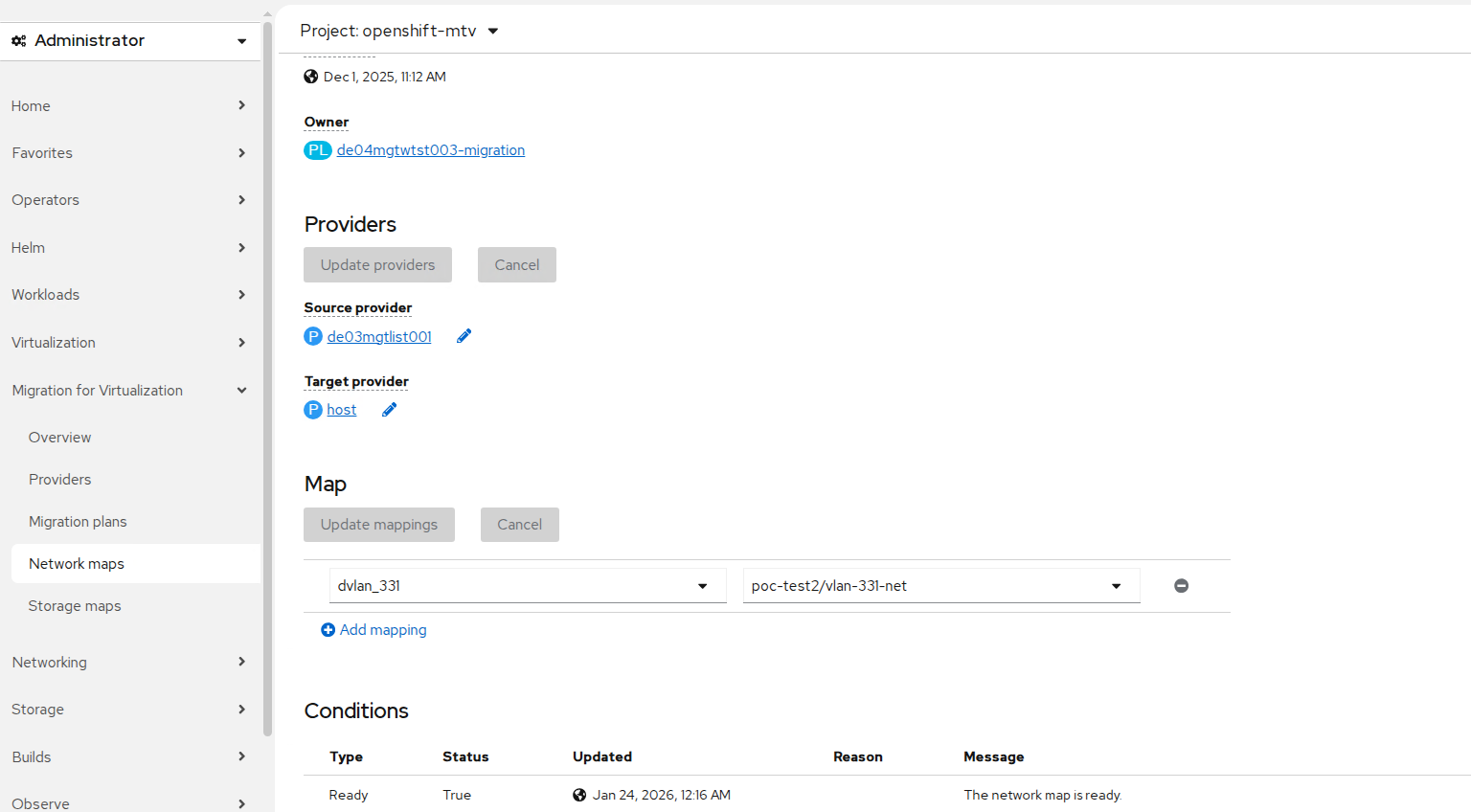
* Target: Your openshift-oc-cluster-test / OC host cluster / KubeVirt
* Source: vCenter or ESXi host / OVA



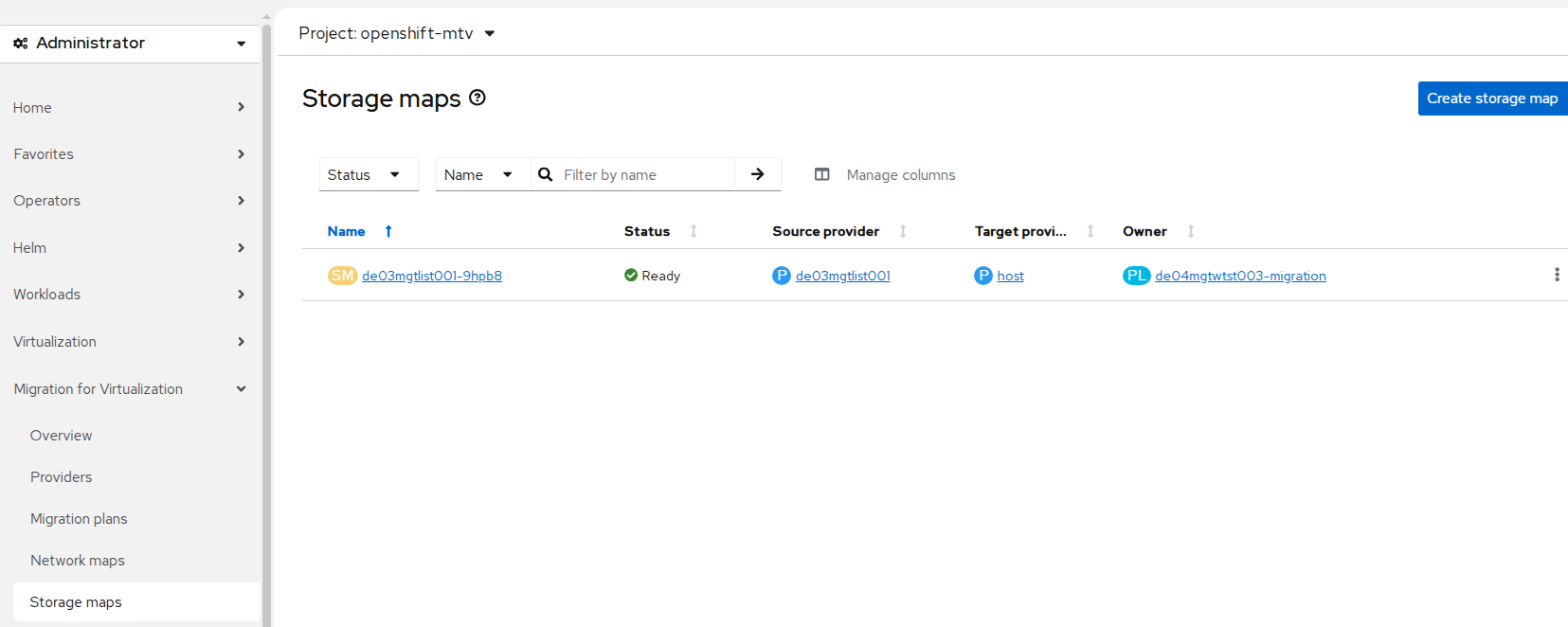


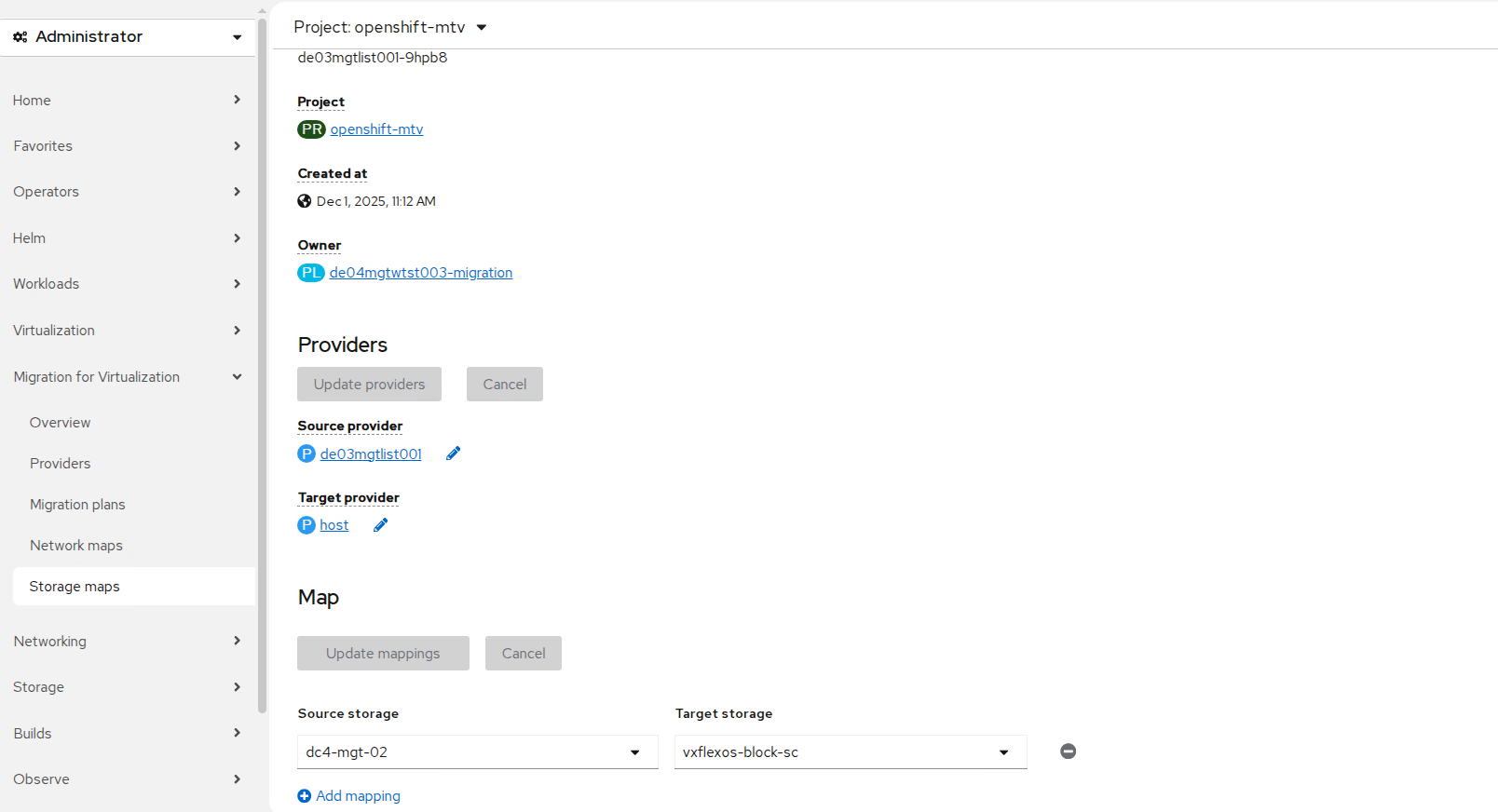
1. Create Network maps



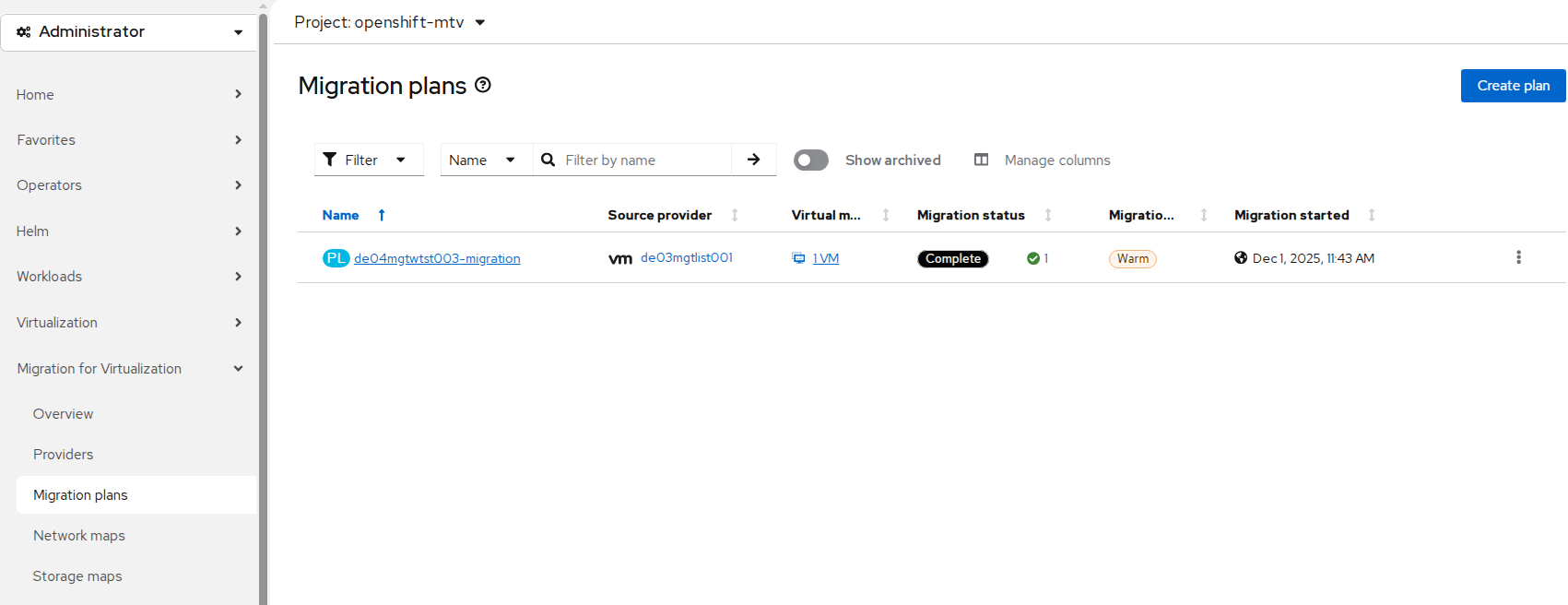


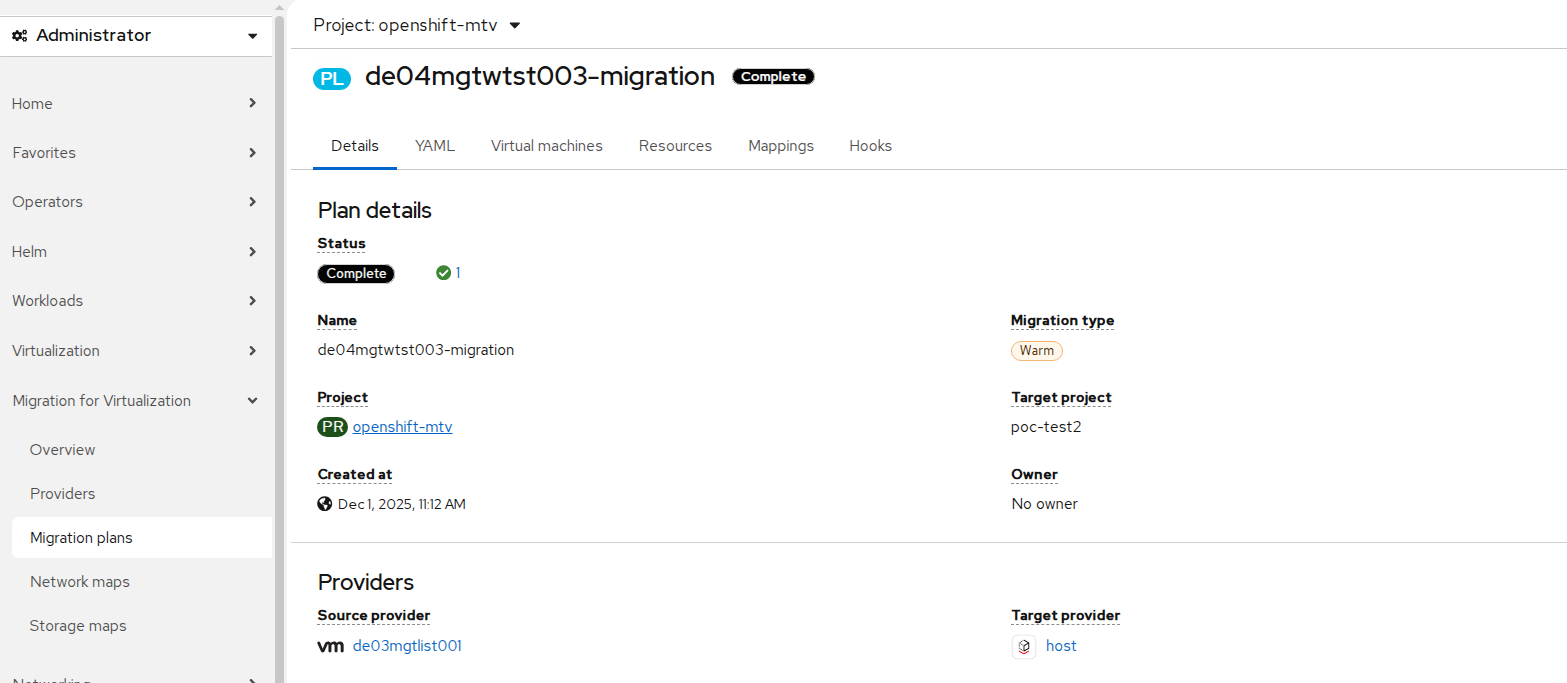
1. Create Storage maps

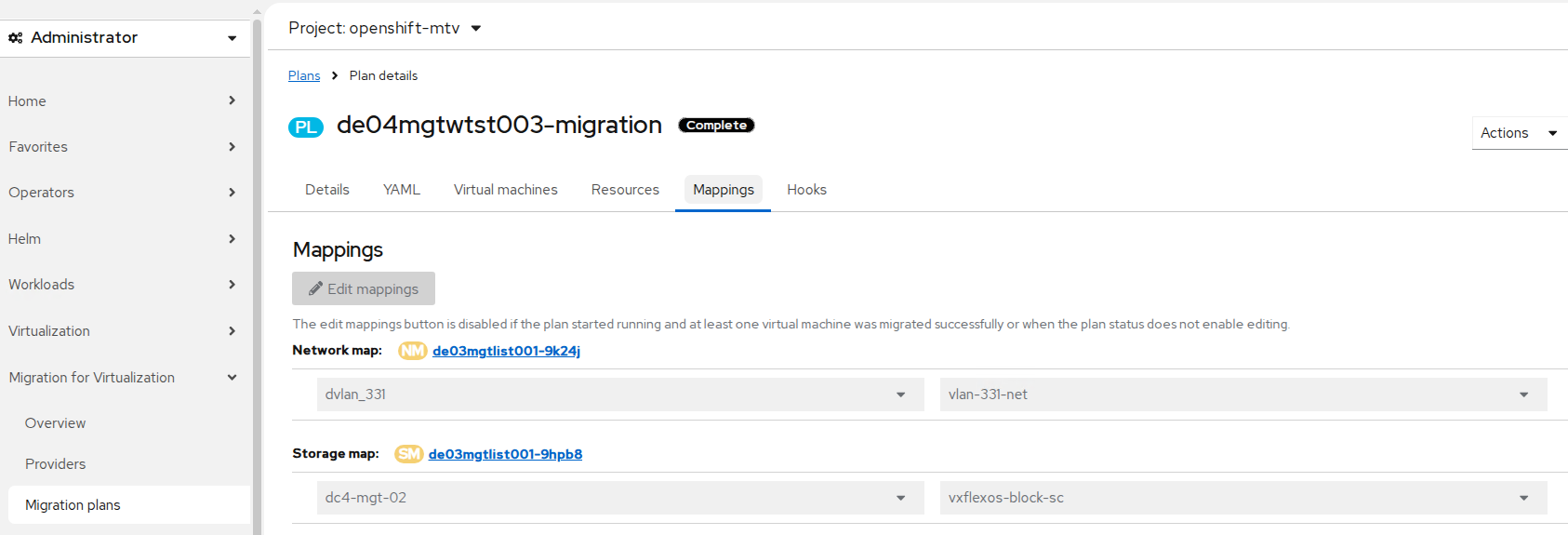




1. Create a migration plan, selecting one or more VMs to migrate.









Start the migration → disks are streamed via VDDK → VMs appear in OpenShift Virtualization.

After completion, VMs run as KubeVirt VirtualMachine resources with PVC-backed storage.

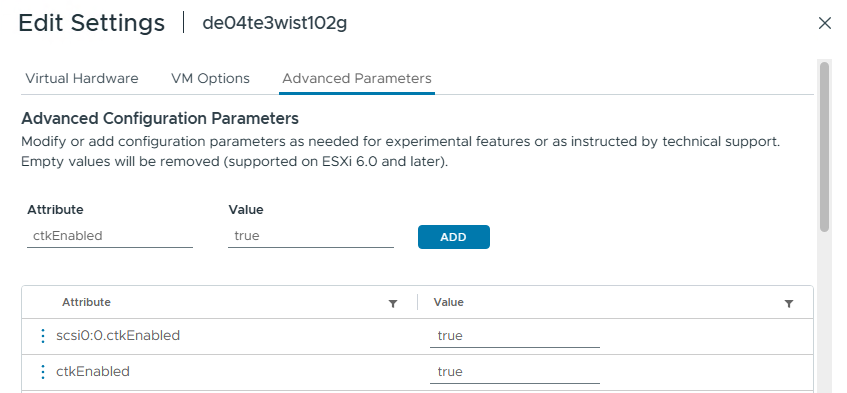
Note: MTV (Migration Toolkit for Virtualization) and it’s completely accurate:

Warm migration requires VMware Changed Block Tracking (CBT) so MTV can copy only the changed blocks during incremental syncs.

When CBT is disabled on a VM, MTV cannot perform warm migration and will fail the plan.  
**Option A**: Enable CBT on the affected VMs in vCenter:

1. Power off the VM (CBT cannot be enabled while the VM is running)
2. Edit VM settings → VM Options → Advanced → Edit Configuration
3. Add or verify these parameters:

|  |  |
| --- | --- |
| Key | Value |
| ctkEnabled | true |
| scsi0:0.ctkEnabled | true (repeat for each disk) |



1. Power on the VM
2. Create a snapshot -This initializes the CBT map.
3. Delete the snapshot (This is normal; CBT remains active.)

**Option B:** Enable CBT using PowerCLI

$vm = Get-VM -Name "vm-name"

$vm | Set-VM -AdvancedSetting @{Name="ctkEnabled";Value="true"} -Confirm:$false

$vm.HardDisks | % { Set-HardDisk $\_ -AdvancedSetting @{Name="$($\_.ExtensionData.DeviceInfo.Label).ctkEnabled";Value="true"} -Confirm:$false }

New-Snapshot -VM $vm -Name "init-cbt"

Remove-Snapshot -VM $vm -Name "init-cbt" -Confirm:$false

Disable: CPU/Memory hotplug

Network ports required for migrating from VMware vSphere

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Port | Protocol | Source | Destination | Purpose |
| 443 | TCP | OpenShift nodes | VMware vCenter | VMware provider inventory |
| Disk transfer authentication |
| 443 | TCP | OpenShift nodes | VMware ESXi hosts | Disk transfer authentication |
| 902 | TCP | OpenShift nodes | VMware ESXi hosts | Disk transfer data copy |

Network ports required for migrating from Open Virtual Appliance (OVA) files

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Port | Protocol | Source | Destination | Purpose |
| 2049 | TCP | OpenShift nodes | Server containing the OVA files | NFS service |
| 111 | TCP or UCP | OpenShift nodes | Server containing the OVA files | RPC Portmapper, only needed for NFSv4.0 |

## VM Snapshot – Recovery

You can create snapshots of virtual machines (VMs) by using the OpenShift Container Platform web console or the command line.

**Create Snapshot in OpenShift Web Console**

1. Navigate to Virtualization -> VirtualMachines
2. Select a VM to open the VirtualMachine details page.
3. Click the Snapshots tab and then click Take Snapshot.
4. Enter the snapshot name. → Click Create

**Create Snapshot via CLI (kubectl/oc)**

Create a VolumeSnapshot YAML file: (in Windows create file: notepad snapshot.yaml)

apiVersion: snapshot.kubevirt.io/v1alpha1

kind: VirtualMachineSnapshot

metadata:

name: snapshot-before-upgrade

spec:

source:

apiGroup: kubevirt.io

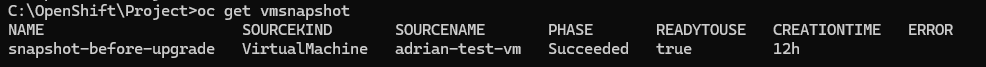
kind: VirtualMachine

name: test-vm

Apply: oc apply -f snapshot.yaml



Check you snapshot: oc get vmsnapshot



**Restore snapshot:**

oc apply -f restore.yaml

apiVersion: snapshot.kubevirt.io/v1alpha1

kind: VirtualMachineRestore

metadata:

name: **snapshot-before-upgrade**

spec:

target:

apiGroup: kubevirt.io

kind: VirtualMachine

name: **test-vm**

virtualMachineSnapshotName: my-snapshot

**Basic commands:**

Create a snapshot: oc apply -f snapshot.yaml

Verify snapshot creation: oc get vmsnapshot

oc get vmsnapshot -n my-namespace

List snapshots: oc get vmsnapshot -n <namespace>

Restore snapshot: oc apply -f restore.yaml

Check restore status: oc get vmrestore -n <namespace>

oc describe vmrestore <restore-name> -n <namespace>

Delete the snapshot: oc delete vmsnapshot <snapshot-name> -n <namespace>

## Rack Level Availability

Label workers to represent racks R1/R2/R3

oc label node worker-a topology.kubernetes.io/zone=rack-r1

oc label node worker-b topology.kubernetes.io/zone=rack-r2

oc label node worker-c topology.kubernetes.io/zone=rack-r3

apiVersion: kubevirt.io/v1

kind: VirtualMachine

metadata:

  name: app-node-1

spec:

  runStrategy: Always

**template**:

    spec:

      affinity:

        podAntiAffinity:

          requiredDuringSchedulingIgnoredDuringExecution:

          - labelSelector:

              matchLabels:

                app: **my**-clustered-vm

            topologyKey: kubernetes.io/hostname

        nodeAffinity:

          requiredDuringSchedulingIgnoredDuringExecution:

            nodeSelectorTerms:

            - matchExpressions:

              - key: topology.kubernetes.io/zone

**operator**: In

                values: ["rack-r1","rack-r2","rack-r3"]

      topologySpreadConstraints:

      - maxSkew: 1

        topologyKey: topology.kubernetes.io/zone

        whenUnsatisfiable: DoNotSchedule

        labelSelector:

          matchLabels:

            app: **my**-clustered-vm

      domain:

        cpu: { cores: 2 }

        devices: { disks: [{ name: rootdisk, disk: { bus: virtio } }] }

      volumes:

      - name: rootdisk

        persistentVolumeClaim: { claimName: vm-disk-rwx }

      terminationGracePeriodSeconds: 180

## Templates deployment Windows / Linux

## Patching templates – Lifecycle

## OpenShift Backup and Restore

Control Plane Backup - Used for full cluster disaster recovery.

List nodes:

oc **get** nodes -l node-role.kubernetes.io/master

Open a debug shell on a control‑plane node

oc debug node/<master-node>

Run the official backup script:

sudo -E /usr/**local**/bin/cluster-backup.sh /home/core/backup

* snapshot.db (etcd database)
* static pod manifests
* certificates
* all Kubernetes objects
* all OpenShift objects
* namespaces
* deployments, pods, services
* secrets & configmaps
* CRDs
* VM definitions (OpenShift Virtualization)
* cluster operators’ state
* networking configuration
* RBAC, users, roles

These files are required for a full cluster restore.

Running the OpenShift **etcd backup script on each control‑plane (master) node** is the correct and officially supported way to create a full cluster backup.

List all Master Nodes:  
oc **get** nodes -l node-role.kubernetes.io/control-plane

A black screen with white text

AI-generated content may be incorrect.

**Open a debug shell on the master node**

oc debug node/master-0

oc debug node/usnc1ciocm1lpd.ad.diebold.com

oc debug node/de04mgtlapt033.test-dnpcloud.dieboldnixdorf.com

Inside the debug shell, run:  
sh-5.1#chroot /host

Now you are effectively “logged into” the master node.

Create a backup directory

sh-5.1#mkdir -p /home/core/backup

Run the official Red Hat backup script

sh-5.1#sudo -E /usr/**local**/bin/cluster-backup.sh /home/core/backup

**Restore:**

You can restore only if:

* You have a valid etcd snapshot  
  You are restoring to the same cluster (same nodes, same names)
* You are restoring control plane only (worker nodes rejoin automatically)

**High‑level restore steps**

**Step 1 — Stop the cluster**

Shut down all control‑plane nodes cleanly. Red Hat requires the cluster to be fully stopped before restore operations .

**Step 2 — Boot one control‑plane node into recovery**

Pick **one master node** to perform the restore.

1. SSH into the node
2. Move or delete existing etcd data directory
3. Run the restore command:

sudo -E /usr/**local**/bin/cluster-restore.sh /home/core/backup

This script:

* Restores the etcd snapshot
* Recreates static pod manifests
* Rebuilds certificates if needed

**Reboot the restored node**

sudo reboot

Wait until:

* etcd is running
* kube-apiserver is healthy
* node shows Ready

**Rejoin the other control‑plane nodes**

For each remaining master:

1. Shut it down
2. Delete old etcd data
3. Copy manifests from the restored node
4. Reboot

They will automatically join the restored etcd cluster.

**Verify cluster health**

oc **get** nodes

oc **get** co

oc **get** pods -A

Ensure all ClusterOperators become **Available**.

This process restores cluster state, not application data.

For application backups, use OADP/Velero or storage‑level snapshots .

You cannot restore to a different cluster or different node names.

## DR Test for OpenShift Cluster

How to Simulate a Failure and Perform a Recovery in OpenShift

If you want to simulate a failure and then test the recovery process in an OpenShift (OCP) cluster, there are several safe and realistic scenarios you can use. These are commonly used during disaster‑recovery (DR) testing.

**1. Simulating a Single Master Node Failure (safest option)**

How to simulate the failure, On one master node:

sudo systemctl stop kubelet

sudo systemctl stop crio

Or shut the node down completely:

sudo shutdown -h now

**What happens**

* The API server on that node goes down
* The remaining masters continue serving traffic
* The cluster stays functional

**How to recover**

Simply start the services again:

sudo systemctl start crio

sudo systemctl start kubelet

or reboot

sudo reboot

**2. Simulating an ETCD Failure on One Master**

This is more realistic and tests the resiliency of the etcd cluster.

Simulate the failure

sudo systemctl stop etcd

Recovery

sudo systemctl start etcd

**3. Simulating a Full Control‑Plane Failure (DR scenario)**

This is the scenario used to test cluster‑restore.

Simulate the failure

Shut down all master nodes:

sudo shutdown -h now

Or simulate corrupted ETCD data:

sudo rm -rf /**var**/lib/etcd/\*

**Full Cluster Recovery (Disaster Recovery Restore)**

Step 1 — Choose one master node

This will be your “initial restore node.”

Step 2 — Remove old ETCD data

sudo rm -rf /**var**/lib/etcd/\*

Step 3 — Run the restore

sudo -E /usr/**local**/bin/cluster-restore.sh /home/core/backup

Step 4 — Reboot

sudo reboot

Step 5 — Restore the remaining masters

For each remaining master:

1. Shut it down
2. Delete /var/lib/etcd/\*
3. Copy the new manifests from the restored master
4. Reboot

Step 6 — Verify cluster health

oc **get** nodes

oc **get** co

oc **get** pods -A

## Tools installation

sudo dnf install openshift-clients

This installs:

* oc
* kubectl
* virtctl
* additional OpenShift CLI tools

curl -LO <https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest/openshift-client-linux.tar.gz>

tar -xvf openshift-client-linux.tar.gz

Move binaries into your PATH

sudo mv oc /usr/**local**/bin/

sudo mv kubectl /usr/**local**/bin/

binaries into

For windows: oc.exe,helm.exe, virtctl.exe  
Register system variables: sysdm.cpl

Download Tools from OpenShift Web Console

Navigate to:  
Virtualization → Overview

Click Download required tool

Extract and install:

Virtctl installation:

tar -xvf virtctl.tar.gz

chmod +x virtctl

sudo mv virtctl /usr/local/bin/

## OC Nodes Maintenance

**Worker Nodes Restart**

In OpenShift you never “restart the whole cluster” the way you would reboot a single server. Instead, you perform controlled rolling maintenance on nodes so the cluster stays healthy and workloads remain available.

1. Cordon the node (oc **get** nodes)

Prevents new workloads from scheduling on it.

oc adm cordon <node-name>

1. Drain the node - It safely removes all workloads.

oc adm drain <node-name> \

--ignore-daemonsets \

--**delete**-emptydir-data \

--force

  After this, the node will have no running pods except system DaemonSets.

1. Perform your maintenance  
   reboot the node

update OS packages

replace hardware

rotate certificates

upgrade kernel

Reboot

ssh core@<node> sudo systemctl reboot

1. Wait for the node to return

oc **get** nodes

1. Uncordon the node

oc adm uncordon <node-name>

Node is now back in service.

Restarting **all worker nodes** (rolling reboot), **one by one**.

**for** node **in** $(oc **get** nodes -l node-role.kubernetes.io/worker -o name); **do**

oc adm cordon $node

oc adm drain $node --ignore-daemonsets --**delete**-emptydir-data --force

ssh core@${node#node/} sudo systemctl reboot

echo "Waiting for $node to return..."

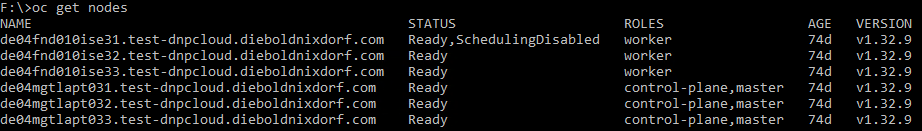
sleep 60

oc wait node/${node#node/} --for=condition=Ready --timeout=10m

oc adm uncordon $node

**done**

the first worker node is in reboot process



**Control-plane node restart**

1. Verify cluster health before touching anything

oc **get** co

oc **get** nodes

oc **get** pods -n openshift-etcd -o wide

All control plane components should be:

* AVAILABLE=True
* PROGRESSING=False
* DEGRADED=False

1. Cordon the control‑plane node

oc adm cordon <master-node-name>

1. Drain the control‑plane node  
   oc adm drain <master-node-name> \

--ignore-daemonsets \

--force \

--**delete**-emptydir-data \

--grace-period=90

You will see warnings about static pods — this is expected.

1. Reboot the node

ssh core@<master-node-name> sudo systemctl reboot

\* Permission denied (publickey,gssapi-keyex,gssapi-with-mic)

1. Wait for the node to return

oc **get** nodes

  The node must return to **Ready**.

oc **get** pods -n openshift-etcd -o wide

  All etcd pods must be Running.

1. Uncordon the node

oc adm uncordon <master-node-name>

\* Permission denied (publickey,gssapi-keyex,gssapi-with-mic)

ssh core@de04mgtlapt031.test-dnpcloud.dieboldnixdorf.com sudo systemctl reboot



Check which SSH keys are installed on the cluster

oc get machineconfig | grep ssh

Then inspect:

oc describe machineconfig <name>

oc describe machineconfig 99-master-ssh

Look for: sshAuthorizedKeys: - ssh-rsa **AAAA...**

cat <<EOF | oc apply -f –

apiVersion: machineconfiguration.openshift.io/v1

kind: MachineConfig

metadata:

name: 99-master-ssh

labels:

machineconfiguration.openshift.io/role: master

spec:

config:

ignition:

version: 3.2.0

passwd:

users:

- name: core

sshAuthorizedKeys:

- ssh-rsa AAAA...your\_public\_key\_here

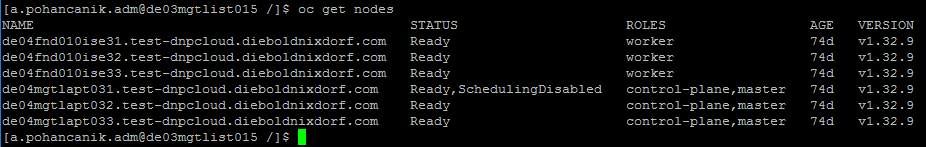
EOF SSH key to master nodes (**if** missing)

 After applying:

* all master nodes will reboot
* your key will be installed
* SSH login will work

Test SSH again  
ssh core@<master-node> ssh core@de04mgtlapt031.test-dnpcloud.dieboldnixdorf.com sudo systemctl reboot

If you still get denied, test with explicit key:  
ssh -i ~/.ssh/<your-key> core@<master-node>



## Login to OpenShift / Token

Access methods:

1. web interface: <https://console-openshift-console.apps.ocp4test.ad.diebold.com/dashboards>
2. for login via console tools: need token





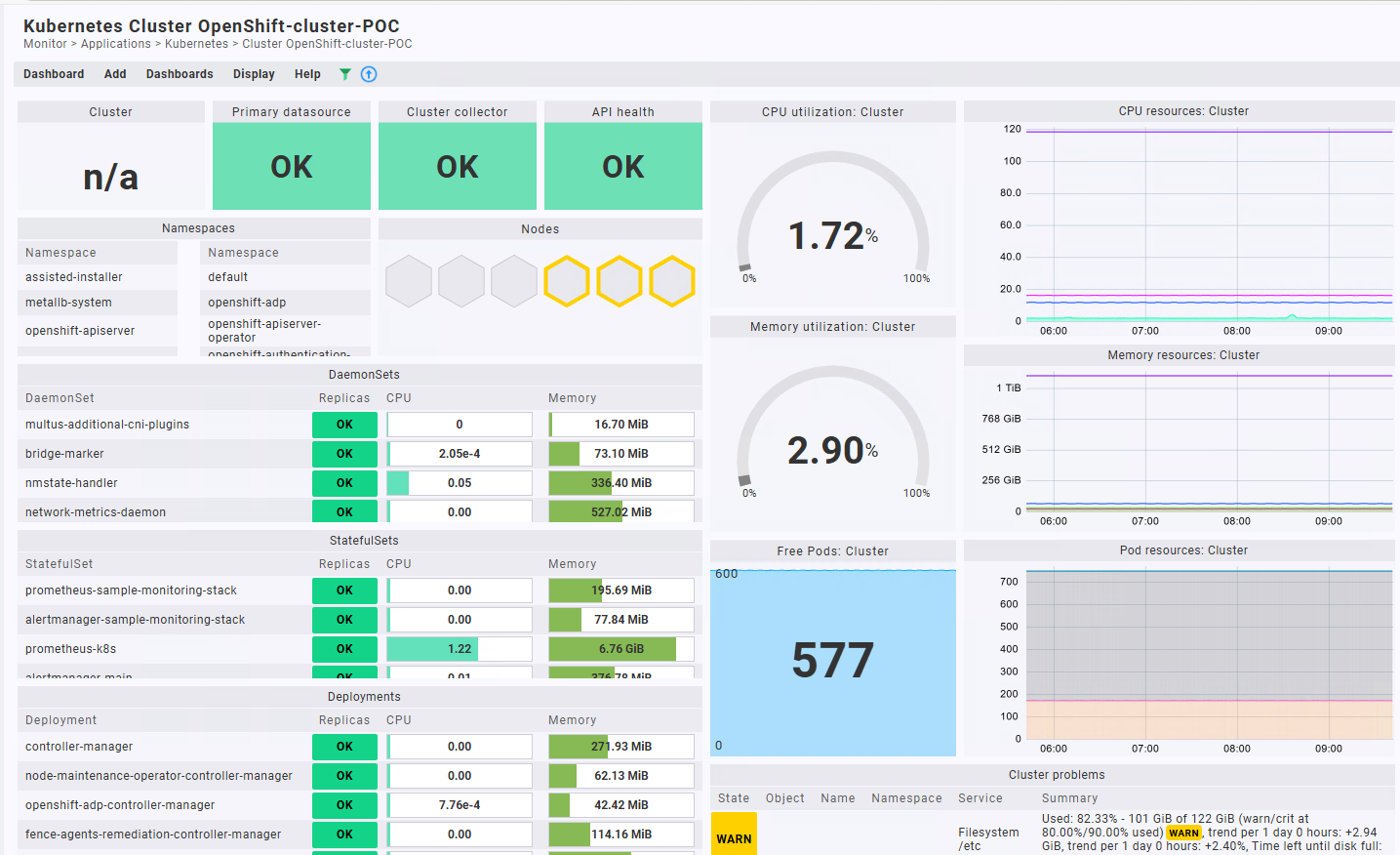
In: *Log in with this token* select this full link  
oc login --token=sha256~jRLpHBC91SDlcL3LDchteT5Jm7OZ-6gCo3oS8k4FYQQ --server=https://api.dnpcpoc.test-dnpcloud.dieboldnixdorf.com:6443

and copy it into your console / bastion jump server.

## Checkmk Monitoring

Checkmk Monitoring service can monitor:

* Clusters
* Namespaces
* Nodes
* Pods
* Deployments
* DaemonSets
* StatefulSets
* CronJobs
* Persistent Volumes (PVs)
* Persistent Volume Claims (PVCs)



|  |  |  |  |
| --- | --- | --- | --- |
| Pod Containers | Pod Info | Deployment Conditions | Namespace Info |
| Node Container Count | Deployment Info | CPU | Resource Quota Memory |
| Memory | Pod Phase | Pod Conditions | Resource Quota CPU |
| API health | Pod Status | Node Conditions | CronJob Info |
| Kubelet | Pod Restarts | Cluster Info | CronJob Status |
| Pod Resources | Replicas | DaemonSet Info | Cluster Collector for OpenShift |
| Node Count | Node Info | StatefulSet Info | Persistent Volume Claim |

**Monitoring setup:**

Create a namespace and service account

oc create **namespace** checkmk-monitoring

oc apply -f https://raw.githubusercontent.com/Checkmk/checkmk\_kube\_agent/checkmk\_docs/deploy/kubernetes/checkmk-serviceaccount.yaml

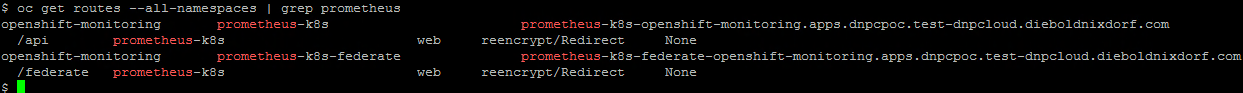
Obtain API endpoints, token and certificate

oc cluster-info



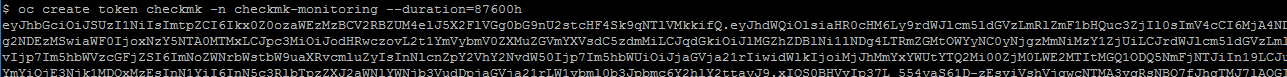
Retrieve the Prometheus API endpoint

oc **get** routes --all-namespaces | grep prometheus



Retrieve the token

oc create token checkmk -n checkmk-monitoring --duration=87600h



Save it — this is your Checkmk API token.

Extract the cluster CA certificate

oc **get** configmap kube-root-ca.crt -n checkmk-monitoring -o jsonpath='{.data.ca\.crt}'

Create a Service Account for CheckMK

oc create sa checkmk -n openshift-monitoring

Give it Read‑Only Access (Cluster‑wide)

oc adm policy **add**-cluster-role-to-user view \

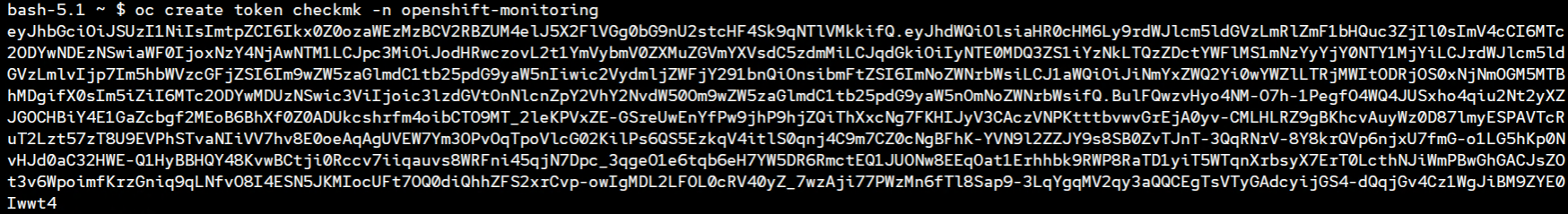
system:serviceaccount:openshift-monitoring:checkmk

This gives CheckMK:

* read access to all namespaces
* no write/delete permissions

Get the Token for Checkmk

oc create token checkmk -n openshift-monitoring



Retrieve the certificate

oc **get** configmap kube-root-ca.crt -n checkmk-monitoring -o jsonpath='{.data.ca\.crt}'

Setting up the monitoring in Checkmk

* Go to **Setup → Cloud → Kubernetes**
* Add a new Kubernetes cluster
* Enter:
  + API URL
  + Token
  + CA certificate (optional if using trusted CA)
* Save and activate changes
* CheckMK will automatically discover:
* nodes
* pods
* deployments
* cluster health
* namespaces
* resource usage

Optional: Add Prometheus Metrics

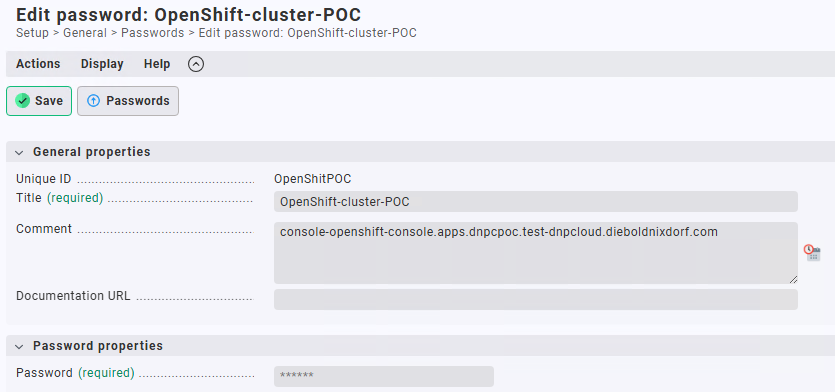
oc -n openshift-monitoring **get** route prometheus-k8s

Then add the endpoint to CheckMK as a Prometheus datasource.

Storing the password (token) in Checkmk

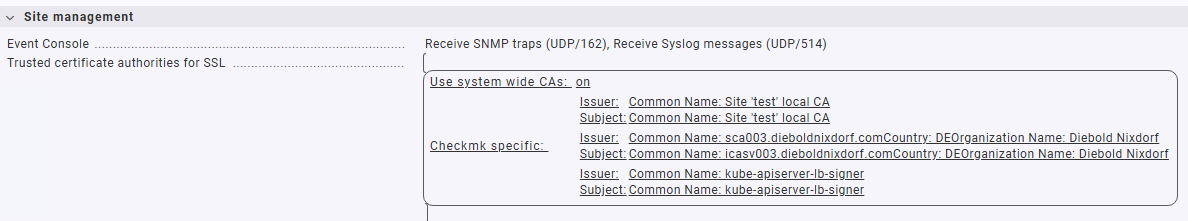
Setup > General > Passwords > Add password.

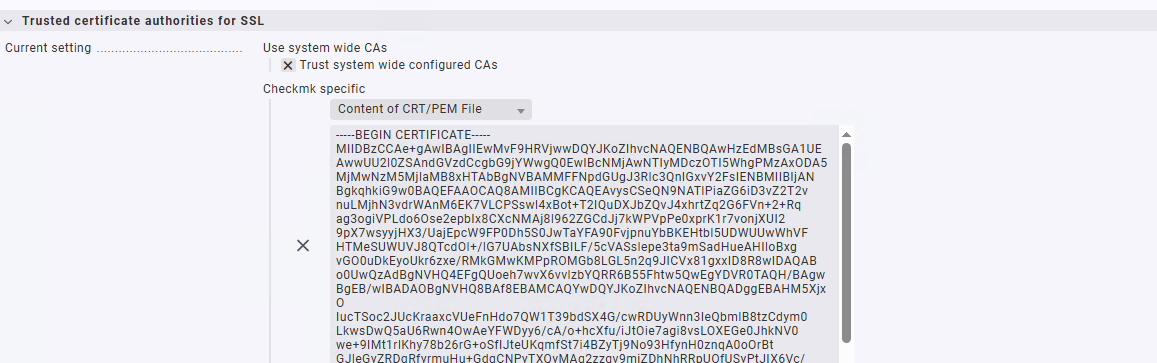
Password = Token



Importing a service account’s CA certificate into Checkmk

Setup > General > Global settings > Site management > Trusted certificate authorities for SSL:





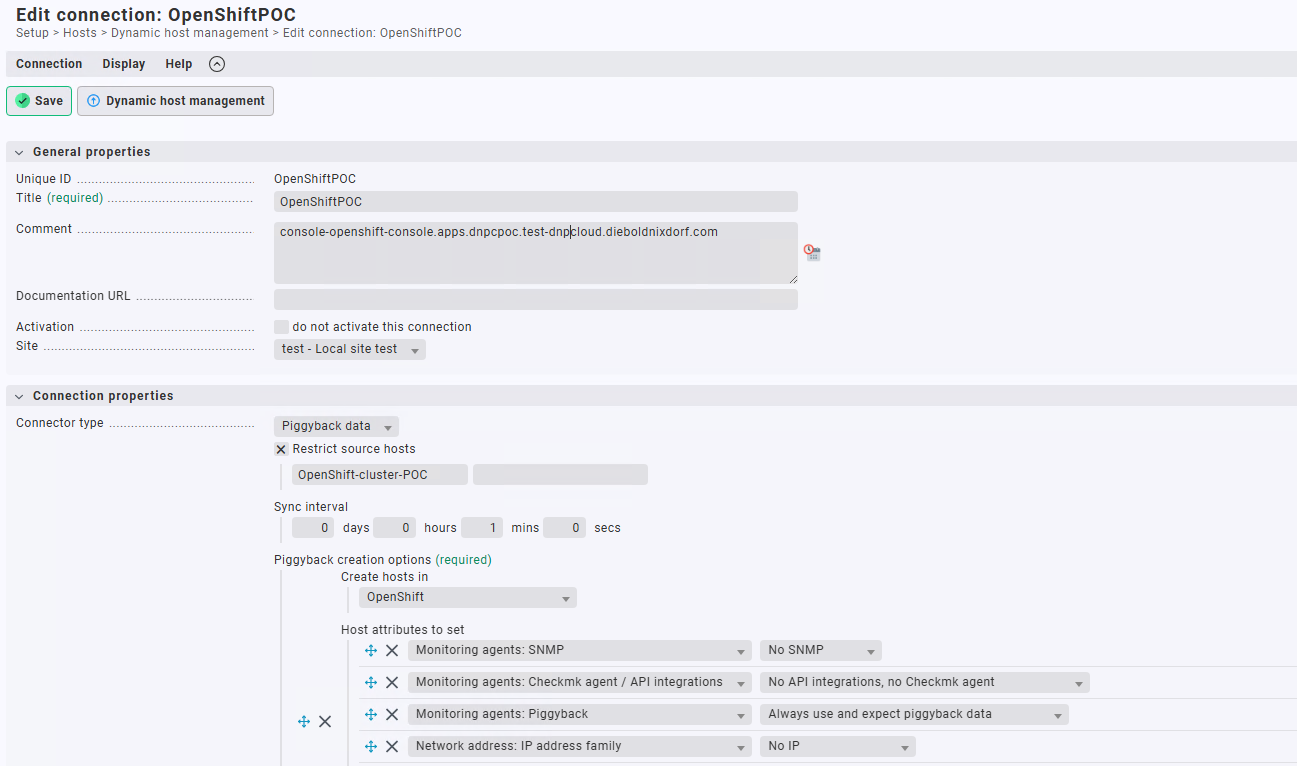
Creating the piggyback host

Setup > Hosts > Main > Add Folder: OpenShift

Add Host: define Hostname: OpenShift-cluster-POC, IP address family > No IP

Setting up dynamic host management: (can create all of a cluster’s hosts automatically)

Setup > Hosts > Dynamic host management > Add connection

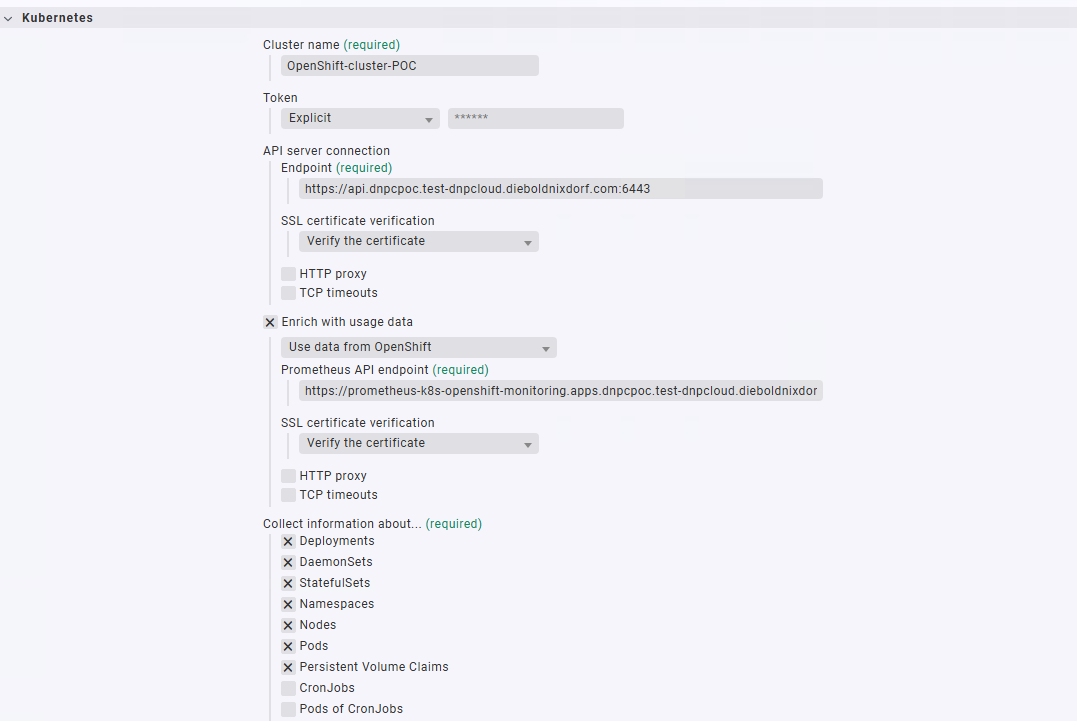


Optimize the periodic service discovery

By default, Checkmk performs a service discovery every two hours and displays the result from this discovery in the Check\_MK Discovery service.

Setting up the special agent

Setup > Agents > VM, cloud, container > Kubernetes. Create a new rule with Add rule.



Monitor > Applications > Kubernetes

Grant Prometheus API access to ServiceAccount: checkmk

Create RBAC for Prometheus API access

Create a file named rbac-checkmk-prometheus.yaml:

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRole

metadata:

name: checkmk-prometheus-reader

rules:

- apiGroups:

- monitoring.coreos.com

resources:

- prometheuses

- prometheuses/api

verbs:

- **get**

- list

- watch

- apiGroups: [""]

resources:

- services

- pods

verbs:

- **get**

- list

- watch

---

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

name: checkmk-prometheus-reader-binding

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: checkmk-prometheus-reader

subjects:

- kind: ServiceAccount

name: checkmk

**namespace**: checkmk-monitoring

Apply it:  
oc apply -f rbac-checkmk-prometheus.yaml

## System Log monitoring

Syslog server setup. Forwarding application + infrastructure logs.

apiVersion: logging.openshift.io/v1

kind: ClusterLogForwarder

metadata:

  name: instance

**namespace**: openshift-logging

spec:

  outputs:

    - name: syslog-udp-**out**

      type: syslog

      url: udp://syslog.example.com:514

      syslog:

        facility: local1

        severity: info

        rfc: rfc3164

  pipelines:

    - name: app-infra-to-syslog-udp

      inputRefs:

        - application

        - infrastructure

      outputRefs:

        - syslog-udp-**out**

For TCP:

url: tcp://syslog.example.com:514

name: infra-to-syslog-tcp

outputRefs:

        - syslog-tcp-output

fvasgv

Check alerts:

oc **exec** -n openshift-monitoring prometheus-k8s-0 -- curl -s localhost:9090/api/v1/alerts | jq .

  (jq: command not found -> install sudo dnf install jq)

Check Prometheus logs

oc logs -n openshift-monitoring prometheus-k8s-0

Check if the Prometheus pods are running

oc **get** pods -n openshift-monitoring -l app=Prometheus

Check monitoring operator health

oc **get** co monitoring

Check the Prometheus service

oc **get** svc -n openshift-monitoring prometheus-k8s



## Install the Dell CSM

Install the Dell CSM (Container Storage Modules) Operator on your OpenShift project

1**. Access OperatorHub in OpenShift**

Log in to your OpenShift console.

Navigate to Operators in the left-side menu.

Select OperatorHub.

2. **Find and Install Dell CSM Operator**

Search for Dell Container Storage Modules (Certified) in the OperatorHub.

Click on the Dell CSM Operator card to review details.

Click Install.

Choose the appropriate installation mode (e.g., install in a specific namespace or cluster-wide).

Confirm and proceed with the installation.

## Enable virtualization in OpenShift

Install and activate the OpenShift Virtualization Operator. This adds KubeVirt, CDI, and all VM‑related APIs to your cluster.

In the OpenShift Web Console:

1. Go to Operators → OperatorHub
2. Search for OpenShift Virtualization
3. Click Install
4. Use these settings:

* Installation mode: All namespaces on the cluster
* Installed Namespace: openshift-cnv (created automatically)
* Approval strategy: Automatic

Click Install.

Create the HyperConverged Resource (this actually enables virtualization)

After the operator is installed:

1. Go to **Operators → Installed Operators**
2. Select **OpenShift Virtualization**
3. Find the **HyperConverged** section
4. Click **Create HyperConverged**
5. Accept the default YAML and create it

This step deploys all virtualization components (KubeVirt, CDI, etc.).

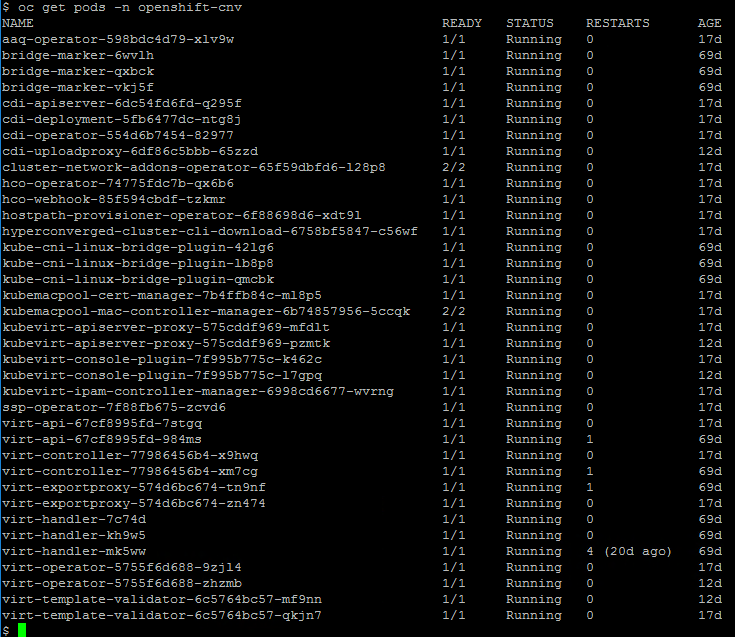
Verify That Virtualization Is Enabled

oc **get** pods -n openshift-cnv

You should see pods like:

* virt-operator
* virt-controller
* virt-handler
* cdi-\*
* kubevirt-\*

All should be Running.



## Enabling IOMMU

Enabling IOMMU (Input-Output Memory Management Unit) in Red Hat OpenShift is a prerequisite for high-performance networking (SR-IOV) and GPU passthrough to virtual machines.

IOMMU allows the host to safely map device memory for VMs or containers.

The best way to enable IOMMU on Red Hat Enterprise Linux CoreOS (RHCOS) nodes is to use a MachineConfig object to modify the kernel boot parameters (intel\_iommu=on or amd\_iommu=on).

**Create a MachineConfig YAML file**

apiVersion: machineconfiguration.openshift.io/v1

kind: MachineConfig

metadata:

labels:

machineconfiguration.openshift.io/role: worker # Or master

name: 99-worker-iommu

spec:

kernelArguments:

- intel\_iommu=on

- iommu=pt

**Apply the configuration**:

oc apply -f 99-worker-iommu.yaml

**Verify the update**: (The MachineConfig Operator will automatically drain, reboot, and cordoned nodes.)

Monitor the progress:

oc **get** mcp

**Verification** After the node reboots, verify that IOMMU is active:

oc debug node/<node\_name> -- chroot /host dmesg | grep -i iommu  
oc debug node/<node-name> chroot /host dmesg | grep -e IOMMU -e DMAR (DMAR for Intel VT‑d)

You should see messages confirming IOMMU is enabled.



|  |  |  |
| --- | --- | --- |
| **Setting** | **When VBS is ON** | **When VBS is OFF** |
| IOMMU / VT‑d / AMD‑Vi | **Forced enabled** | User‑controllable |
| DMA remapping | Required | Optional |
| VM vIOMMU | Forced | Optional |

## Enable NUMA for a VM

KubeVirt provides **NUMA topology passthrough**:

* NUMA cells
* vCPU → NUMA affinities
* Memory allocation policy per NUMA node

NUMA is especially important for Windows VMs with VBS, high‑performance workloads, and DPDK/SRIOV workloads.

**Enable NUMA for a VM**

The following YAML snippet adds:

* NUMA passthrough mode
* Numa cells
* CPU pinning
* Memory assignment

Add this to your VM’s YAML under spec.template.spec.domain.

VM YAML with NUMA Enabled

apiVersion: kubevirt.io/v1

kind: VirtualMachine

metadata:

  name: win-vm-**with**-numa

spec:

  running: **false**

**template**:

    spec:

      domain:

        cpu:

          sockets: 1

          cores: 4

          threads: 1

          dedicatedCpuPlacement: **true**

          numa:

            guestMappingPassthrough: {}

        memory:

          guest: 8Gi

          numa:

            guest:

              numaCells:

                - id: 0

                  memory: 4096Mi

                  cpu:

                    - 0

                    - 1

                - id: 1

                  memory: 4096Mi

                  cpu:

                    - 2

                    - 3

      terminationGracePeriodSeconds: 0

      networks:

        - name: **default**

          pod: {}

      volumes:

        - name: rootdisk

          containerDisk:

            image: quay.io/kubevirt/windows-container-disk

**Key Fields**

1. Enable NUMA passthrough

numa:

  guestMappingPassthrough: {}

2. Define NUMA cells

numaCells:

  - id: 0

    memory: 4096Mi

    cpu: [0, 1]

  - id: 1

    memory: 4096Mi

    cpu: [2, 3]

``

You are explicitly assigning:

* NUMA Node 0 → vCPUs 0–1, 4 GB
* NUMA Node 1 → vCPUs 2–3, 4 GB

3. Dedicated CPU placement

dedicatedCpuPlacement: true

Confirm NUMA is Working

lscpu | grep NUMA numactl --hardware

Get-WmiObject Win32\_Processor | Select-Object Name,NumberOfLogicalProcessors

Get-WmiObject Win32\_NumaNode

## VBS enable in OpenShift

VBS **requires**:

* A functioning **IOMMU** (Intel VT‑d / AMD‑Vi)
* Hypervisor-based isolation
* SLAT
* TPM 2.0

Microsoft explicitly lists IOMMU (VT‑d/AMD‑Vi) as a requirement for VBS.  
Since OpenShift Virtualization exposes virtual hardware similar to KVM/QEMU, the Windows guest enforces these requirements, so IOMMU cannot be disabled.

Requirements Before Enabling VBS

Your Windows VM must meet these conditions:

* UEFI firmware (NOT Legacy BIOS)
* Secure Boot enabled
* vTPM (TPM 2.0) device attached
* CPU passthrough or hypervisor mode exposing virtualization extensions
* IOMMU/VT‑d/AMD‑Vi available on the OpenShift worker host
* Windows 10/11 or Windows Server 2016+

Step‑by‑Step: Enable VBS on a Windows VM in OpenShift Virtualization

1. Set the VM to UEFI with SecureBoot

spec:

**template**:

    spec:

      domain:

        firmware:

          bootloader:

            efi:

              secureBoot: **true**

2. Add a TPM 2.0 Device to the VM

spec:

**template**:

    spec:

      domain:

        devices:

          tpm:

            model: tpm2

3. Enable Hyper‑V Enlightenments (Required for VBS in Windows)

spec:

**template**:

    spec:

      domain:

        features:

          hyperv:

            vapic: {}

            relaxed: {}

            spinlocks:

              spinlocks: 8191

            synic: {}

            vpindex: {}

            runtime: {}

            reset: {}

            tlbflush: {}

            frequencies: {}

            reenlightenment: {}

Ensure CPU mode supports virtualization extensions

spec:

**template**:

    spec:

      domain:

        cpu:

          model: host-passthrough

Optional: Enable IOMMU exposure (if supported by node)

spec:

**template**:

    spec:

      domain:

        features:

          iommu:

            intel: {}   # for Intel nodes

Apply the YAML and Start the VM

oc apply -f windows-vbs-vm.yaml

reboot the VM.

**Enable VBS inside the Windows guest**

GPO: Computer Configuration > Administrative Templates > System > Device Guard > Turn On Virtualization‑Based Security

Enable:

* Secure Boot
* DMA Protection
* VBS with UEFI lock

Verification Inside the VM:

1. Run msinfo32
2. Check the entry: **Virtualization‑based security**

VirtualizationBasedSecurityStatus should be 1 (enabled) or 2 (running).

## Migration Toolkit for Virtualization (MTV)

To migrate virtual machines (VMs) from the following source providers to OpenShift Virtualization destination providers:

* VMware vSphere
* Red Hat Virtualization (RHV)
* OpenStack
* Open Virtual Appliances (OVAs) that were created by VMware vSphere
* Remote OpenShift Virtualization clusters

**Installation MTV from OperatorHub**

1. Go to **Operators → OperatorHub**
2. Search for **Migration Toolkit for Virtualization**
3. Select it and click **Install**
4. Choose:
   * Installation mode: **A specific namespace on the cluster** (openshift-mtv)
   * Update channel: typically **stable**
   * Approval strategy: **Automatic** or **Manual**
5. Click **Install**

Once installed, MTV creates:

* The **ForkliftController** operator
* CRDs for providers, plans, mappings, etc.
* The **MTV UI** under **Migration → Virtualization**

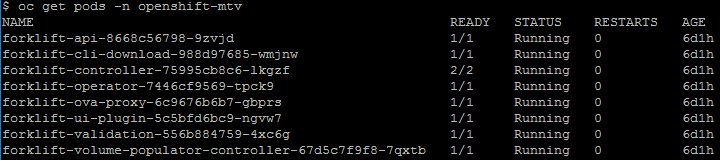
**Verify Installation**

oc **get** pods -n openshift-mtv

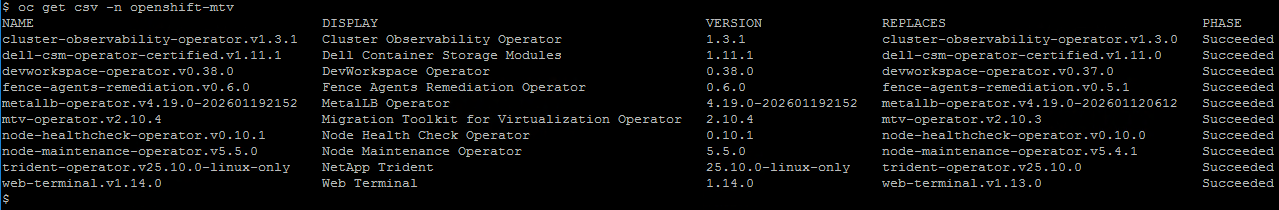
You should see pods like:

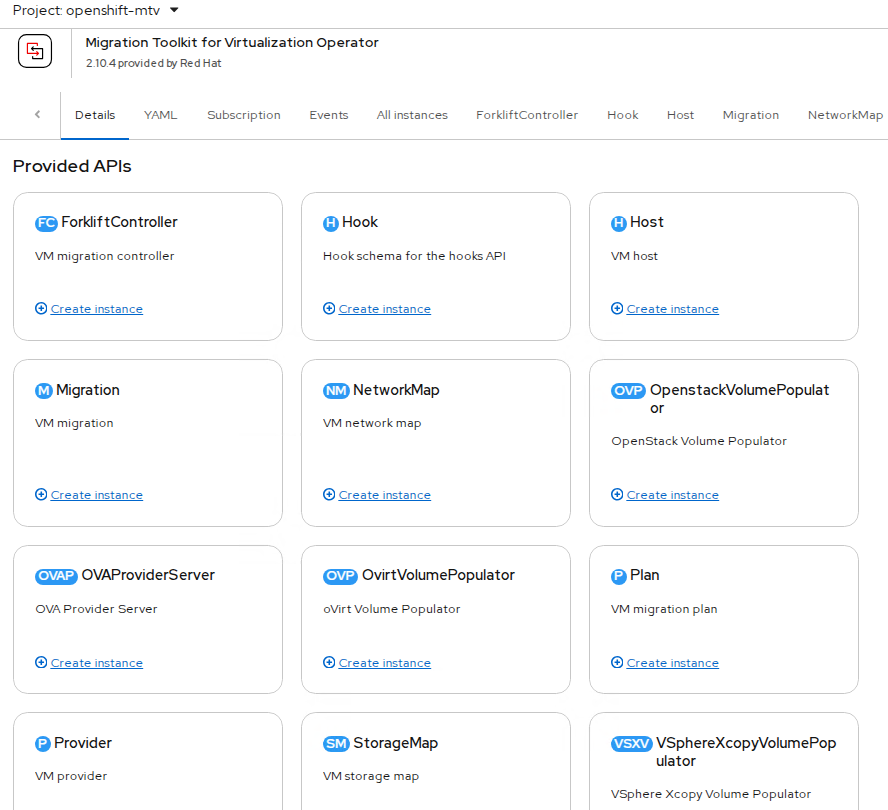
* forklift-controller
* forklift-operator
* forklift-ui

All should be in Running state.



Check the operator status: oc **get** csv -n openshift-mtv





# Security Concept

Define the overall security architecture and compliance alignment.

Provides a comprehensive view of security, including access control, audit logging, and compliance mapping.

It ensures the service is resilient and audit-ready.

Requirements:

- Access control strategy (RBAC)

- Audit logging and retention policies

- Integration with SIEM/SOC

- Mapping to compliance frameworks, which controls are accounted for (ISO 27001, PCI DSS)

- Data protection and privacy controls

# RBAC / Access Concept

Control access based on roles and responsibilities.

Defines roles, access workflows, least privilege enforcement, and periodic review definitions.

It ensures secure and appropriate access to service components.

Requirements:

- Role definitions (e.g., Admin, Operator, )

- Access request and approval workflow

- Least privilege enforcement

- Periodic access review definition

- Emergency access procedures (Break Glas)

**a) Operations Role**

* **Scope:** Cluster or Datacenter level
* **Permissions:**
  + Power operations (Power On/Off, Reset)
  + VM console access
  + Snapshot management
  + Basic VM configuration (CPU/RAM changes)
  + No access to networking/storage configuration

**b) Administrator Role**

* **Scope:** Same as Operations Role, plus:
  + Create/Delete VMs
  + Manage resource pools, Clusters, Namespaces, Datastores
  + Assign permissions to team members (within their scope)

**c) Read-Only Role**

* For auditing or monitoring purposes only.

Access Role: Administrators

1. Engineering Team
2. Virtualization Team

Access Role: Operations team

1. Database Team
2. Linux Team
3. Windows Team

**Dependencies**

The service is dependent on AD authorization and Jumps servers access, which provides web access for the OPS Team.

# Contacts

[Insert statement regarding whom to contact if any issues or questions surrounding the subject matter of the document.]

# Definition / Terms and Abbreviations

|  |  |
| --- | --- |
| Term | Definition |
| [Document Control System] | [IT System to manage documents] |
| [DN] | [Diebold Nixdorf] |
| AD | Active Directory |
| OP | Operation teams  (Linux, Windows, Backup, Storage) |
| OC | Red Hat OpenShift command-line interface (CLI) |
| OCP | Red Hat OpenShift Container Platform |
| namespaces | A namespace isolates specific system resources that are visible to all processes. Inside a namespace, only processes that are members of that namespace can see those resources |
| pod | One or more containers with shared resources, such as volume and IP addresses, running in your OpenShift Container Platform cluster. A pod is the smallest compute unit defined, deployed, and managed. |

# Employee Roles and Responsibilities

|  |  |
| --- | --- |
| Role | Responsibility |
| [Quality Management] | [Controls and reviews the level of quality.] |
|  |  |

# Related Documents / References

|  |  |
| --- | --- |
| Title | Number |
|  |  |
|  |  |

# History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Date | Comment | By |
| [0.1] | 15.12.2025 | [First draft version] | Stuart Evans |
|  |  |  |  |

1. **Server Tier Classes**

7 different Server Tier Classes:

**Bronze**: Simple VM with low requirements e.g. for Testing.

* Operating time:  Monday - Friday 5x9h (08:00-17:00)
* VM is hosted on a random Host in a VMware Cluster, without any HA or DR features.

**Silver**: One VM for productive usage with HA and Backup features.

* Operating time:  Monday - Sunday 7x24h (00:00-24:00)
* VM is hosted on a random Host in a VMware Cluster.

**Silver+:** One VM for productive usage with HA, Backup and Disaster Recovery Support.

* Operating time:  Monday - Sunday 7x24h (00:00-24:00)
* VM is hosted on a random Host in a VMware Cluster.
* With DR capabilities.

**Gold**: 2x active VMs for Cluster Services with HA and Backup.

* Operating time:  Monday - Sunday 7x24h (00:00-24:00)
* 2 active VMs hosted on separate Hosts in a VMware Cluster.
* Cluster service will be provided by application.

**Gold+**: 2x active VMs for Cluster Services with HA, Backup and Disaster Recovery Support.

* Operating time:  Monday - Sunday 7x24h (00:00-24:00)
* 2 active VMs hosted on separate Hosts in a VMware Cluster.
* Cluster service will be provided by application.
* With DR capabilities.

**Gold-scale**: Scalable number of active VMs for Cluster Services with HA, Backup.

* Operating time:  Monday - Sunday 7x24h (00:00-24:00)
* 3 → 10\* active VMs hosted on separate Hosts in a VMware Cluster.   (\* up to the number of Worked Nodes)
* Cluster service will be provided by application.

**Gold-scale+**: Scalable number of active VMs for Cluster Services with HA, Backup and Disaster Recovery Support.

* Operating time:  Monday - Sunday 7x24h (00:00-24:00)
* 3 → 10\* active VMs hosted on separate Hosts in a VMware Cluster.   (\* up to the number of Worked Nodes)
* Cluster service will be provided by application.
* With DR capabilities.