RH OVE Architecture Documentation

Red Hat OpenShift Virtualization Ecosystem Team

2025-08-04

Table of Contents

# Architecture

## Context Diagram

### RH OVE Ecosystem Context Diagram

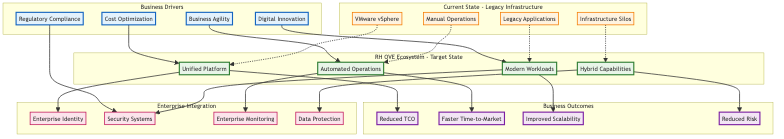
#### Overview

This context diagram provides a high-level view of the RH OVE (Red Hat OpenShift Virtualization Engine) ecosystem within the broader enterprise environment. It illustrates the system boundaries, external entities, data flows, and key integrations that define how the RH OVE ecosystem interacts with users, external systems, and enterprise services.

#### Executive Summary

The RH OVE ecosystem represents a strategic modernization initiative that transforms enterprise virtualization infrastructure while preserving existing investments. This platform enables organizations to migrate from traditional virtualization (VMware) to a cloud-native, Kubernetes-based solution that supports both virtual machines and containers on a unified platform.

##### High-Level Business Context



##### Strategic Value Proposition

###### **Modernization Without Disruption**

* Migrate from VMware to cloud-native platform while maintaining existing VM workloads
* Gradual transformation path that minimizes business risk
* Unified platform for both virtual machines and containers

###### **Cost Optimization**

* Eliminate VMware licensing costs and vendor lock-in
* Reduce infrastructure complexity and operational overhead
* Optimize resource utilization through intelligent workload placement

###### **Enhanced Agility**

* GitOps-driven automation for rapid deployment and scaling
* Self-service capabilities for development teams
* Faster time-to-market for new applications and services

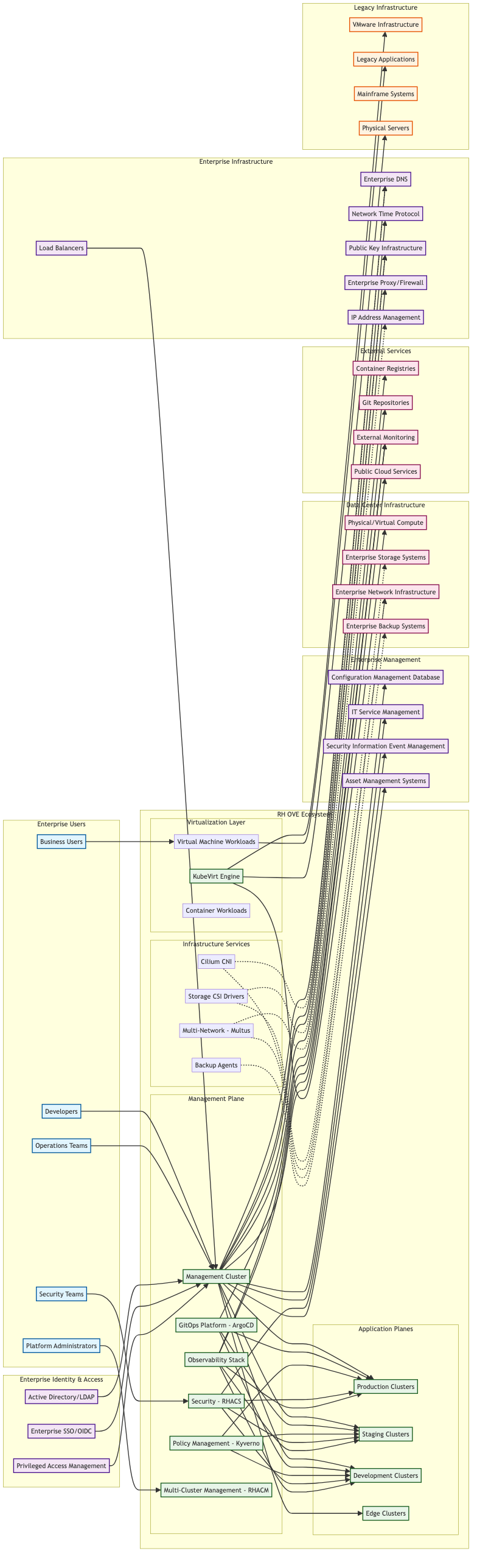
###### **Enterprise-Grade Security and Compliance**

* Zero-trust architecture with micro-segmentation
* Integrated security scanning and policy enforcement
* Comprehensive audit trails and compliance reporting

#### System Context

The RH OVE ecosystem operates as a comprehensive multi-cluster virtualization platform that bridges traditional virtualization workloads with modern cloud-native operations, providing seamless integration with enterprise infrastructure and services.

#### Context Diagram



#### System Boundaries and Responsibilities

##### RH OVE Ecosystem Core

The central system encompasses: - **Management Plane**: Centralized governance, policy, and operations control - **Application Planes**: Multiple clusters for different environments and purposes - **Virtualization Layer**: KubeVirt-based VM and container workload execution - **Infrastructure Services**: Networking, storage, and backup integration

##### Key External Integrations

###### Identity and Access Management

* **Active Directory/LDAP**: Enterprise user directory integration
* **Enterprise SSO/OIDC**: Single sign-on and OAuth/OIDC authentication
* **Privileged Access Management**: Elevated access control and auditing

###### Enterprise Management Systems

* **CMDB**: Configuration item tracking and relationship mapping
* **ITSM**: Service request and incident management integration
* **SIEM**: Security event correlation and threat detection
* **Asset Management**: Hardware and software asset tracking

###### Infrastructure Dependencies

* **Enterprise DNS**: Name resolution services
* **NTP**: Time synchronization across all components
* **PKI**: Certificate management and trust establishment
* **IPAM**: IP Address Management for network planning and allocation
* **Network Infrastructure**: Physical and virtual networking
* **Storage Systems**: Persistent storage for VMs and containers
* **Backup Systems**: Data protection and recovery services

###### Development and Operations

* **Git Repositories**: Source code and configuration management
* **Container Registries**: Image storage and distribution
* **External Monitoring**: Enterprise monitoring system integration
* **Public Cloud Services**: Hybrid and multi-cloud connectivity

##### Legacy System Integration

The RH OVE ecosystem provides migration paths and integration capabilities for: - **VMware Infrastructure**: VM migration and workload transformation - **Legacy Applications**: Containerization and modernization support - **Physical Servers**: Bare metal integration and management - **Mainframe Systems**: API integration and data exchange

#### Data Flow Patterns

##### Inbound Data Flows

* **User Authentication**: From enterprise identity systems
* **Configuration Data**: From Git repositories and CMDB
* **Monitoring Metrics**: To enterprise monitoring systems
* **Security Events**: To SIEM platforms
* **Backup Data**: From enterprise backup systems

##### Outbound Data Flows

* **Audit Logs**: To compliance and logging systems
* **Performance Metrics**: To enterprise dashboards
* **Security Alerts**: To security operations centers
* **Configuration Changes**: To change management systems
* **Service Status**: To IT service management platforms

##### Bidirectional Integration

* **Identity Federation**: Continuous authentication and authorization
* **Policy Synchronization**: Enterprise policy distribution and compliance
* **Asset Discovery**: Dynamic configuration item updates
* **Network Connectivity**: Secure communication channels

#### Security Boundaries

##### Trust Zones

1. **Management Zone**: High-security administrative functions
2. **Production Zone**: Business-critical workload execution
3. **Development Zone**: Lower-trust development activities
4. **DMZ**: External-facing services and integrations

##### Security Controls

* **Network Segmentation**: Micro-segmentation with Cilium
* **Zero Trust Architecture**: Identity-based access controls
* **Encryption**: End-to-end data protection
* **Audit Logging**: Comprehensive activity tracking

#### Scalability and Growth

The context diagram illustrates the ecosystem’s ability to: - **Horizontal Scaling**: Add application clusters as needed - **Geographic Distribution**: Deploy across multiple data centers - **Hybrid Integration**: Seamlessly connect on-premises and cloud resources - **Legacy Modernization**: Gradual transformation of existing systems

#### Operational Model

##### Day-1 Operations (Deployment)

* Initial cluster provisioning and configuration
* Integration with enterprise systems
* Security policy establishment
* Baseline monitoring setup

##### Day-2 Operations (Management)

* Ongoing cluster lifecycle management
* Policy updates and compliance monitoring
* Performance optimization and scaling
* Security incident response

This context diagram serves as the foundation for understanding how the RH OVE ecosystem integrates with and enhances existing enterprise infrastructure while providing a modern, scalable platform for virtualization and containerization workloads.

## Global Overview

### Global Architecture Overview

#### Overview

The RH OVE ecosystem is designed as a multi-cluster architecture that separates concerns between management operations and application workloads. This design provides scalability, security, and operational efficiency by dedicating specialized clusters for different purposes while maintaining centralized governance and oversight.

#### Architecture Principles

##### Separation of Concerns

* **Management Cluster**: Centralized control plane for governance, policy, monitoring, and operations
* **Application Clusters**: Dedicated workload execution environments for virtual machines and containers
* **Clear Boundaries**: Well-defined interfaces and responsibilities between cluster types

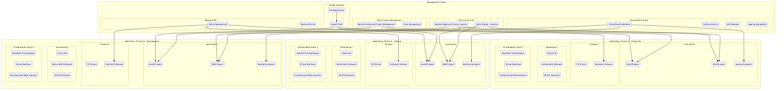
##### Scalability and Growth

* **Horizontal Scaling**: Add application clusters as demand grows
* **Regional Distribution**: Deploy clusters across different geographic locations
* **Resource Optimization**: Right-size clusters based on workload requirements

##### Security and Compliance

* **Zero Trust Architecture**: Network-level security between clusters
* **Centralized Policy Management**: Consistent security policies across all clusters
* **Compliance Monitoring**: Unified compliance reporting and auditing

#### Multi-Cluster Topology



#### Management Cluster Components

##### Core Management Services

###### Red Hat Advanced Cluster Management (RHACM)

apiVersion: operator.open-cluster-management.io/v1  
kind: MultiClusterHub  
metadata:  
 name: multiclusterhub  
 namespace: open-cluster-management  
spec:  
 availabilityConfig: High  
 enableClusterBackup: true  
 overrides:  
 components:  
 - name: multicluster-observability-operator  
 enabled: true  
 - name: cluster-lifecycle  
 enabled: true  
 - name: cluster-permission  
 enabled: true

**Responsibilities:** - Cluster lifecycle management - Policy distribution and compliance - Application deployment coordination - Resource optimization across clusters

###### ArgoCD Hub Configuration

apiVersion: argoproj.io/v1alpha1  
kind: ArgoCD  
metadata:  
 name: argocd-hub  
 namespace: argocd  
spec:  
 server:  
 route:  
 enabled: true  
 tls:  
 termination: reencrypt  
 replicas: 3  
 controller:  
 resources:  
 requests:  
 cpu: 500m  
 memory: 1Gi  
 limits:  
 cpu: 2  
 memory: 4Gi  
 dex:  
 openShiftOAuth: true  
 ha:  
 enabled: true  
 rbac:  
 defaultPolicy: 'role:readonly'  
 policy: |  
 p, role:admin, applications, \*, \*/\*, allow  
 p, role:admin, clusters, \*, \*, allow  
 p, role:admin, repositories, \*, \*, allow  
 g, argocd-admins, role:admin

**Responsibilities:** - GitOps workflow orchestration - Application deployment to target clusters - Configuration drift detection and remediation - Multi-cluster application synchronization

##### Security and Compliance

###### Red Hat Advanced Cluster Security (RHACS)

apiVersion: platform.stackrox.io/v1alpha1  
kind: Central  
metadata:  
 name: stackrox-central-services  
 namespace: stackrox  
spec:  
 central:  
 exposure:  
 loadBalancer:  
 enabled: true  
 persistence:  
 persistentVolumeClaim:  
 claimName: central-db  
 resources:  
 requests:  
 cpu: 1500m  
 memory: 4Gi  
 limits:  
 cpu: 4000m  
 memory: 8Gi  
 scanner:  
 resources:  
 requests:  
 cpu: 200m  
 memory: 200Mi  
 limits:  
 cpu: 2000m  
 memory: 4Gi

**Responsibilities:** - Centralized security policy management - Vulnerability scanning across clusters - Runtime threat detection - Compliance reporting and audit trails

###### Policy Engine (Kyverno)

apiVersion: kyverno.io/v1  
kind: ClusterPolicy  
metadata:  
 name: multi-cluster-vm-policy  
spec:  
 validationFailureAction: enforce  
 background: true  
 rules:  
 - name: require-vm-labels  
 match:  
 any:  
 - resources:  
 kinds:  
 - VirtualMachine  
 validate:  
 message: "VMs must have required labels: environment, owner, backup-policy"  
 pattern:  
 metadata:  
 labels:  
 environment: "?\*"  
 owner: "?\*"  
 backup-policy: "?\*"

##### Observability and Monitoring

###### Federated Prometheus Configuration

apiVersion: monitoring.coreos.com/v1  
kind: Prometheus  
metadata:  
 name: prometheus-federation  
 namespace: monitoring  
spec:  
 replicas: 3  
 retention: 30d  
 storage:  
 volumeClaimTemplate:  
 spec:  
 accessModes: ["ReadWriteOnce"]  
 resources:  
 requests:  
 storage: 500Gi  
 serviceAccountName: prometheus  
 serviceMonitorSelector:  
 matchLabels:  
 prometheus: federation  
 additionalScrapeConfigs:  
 name: additional-scrape-configs  
 key: prometheus-additional.yaml

**Federation Configuration:**

- job\_name: 'federate-app-clusters'  
 scrape\_interval: 15s  
 honor\_labels: true  
 metrics\_path: '/federate'  
 params:  
 'match[]':  
 - '{job=~"kubernetes-.\*"}'  
 - '{job=~"node-.\*"}'  
 - '{job=~"kubevirt-.\*"}'  
 static\_configs:  
 - targets:  
 - 'prometheus-app-cluster-1.monitoring.svc.cluster.local:9090'  
 - 'prometheus-app-cluster-2.monitoring.svc.cluster.local:9090'  
 - 'prometheus-app-cluster-n.monitoring.svc.cluster.local:9090'

###### Centralized Logging

apiVersion: logging.openshift.io/v1  
kind: ClusterLogForwarder  
metadata:  
 name: central-log-forwarder  
 namespace: openshift-logging  
spec:  
 outputs:  
 - name: central-elasticsearch  
 type: elasticsearch  
 url: https://elasticsearch-central.logging.svc.cluster.local:9200  
 secret:  
 name: elasticsearch-central-secret  
 pipelines:  
 - name: forward-app-logs  
 inputRefs:  
 - application  
 - infrastructure  
 - audit  
 outputRefs:  
 - central-elasticsearch

#### Application Cluster Architecture

##### Cluster Sizing and Resource Allocation

###### Production Cluster Profile

apiVersion: v1  
kind: ConfigMap  
metadata:  
 name: cluster-profile-production  
data:  
 profile: |  
 cluster\_type: production  
 node\_count: 12  
 master\_nodes: 3  
 worker\_nodes: 9  
 storage\_nodes: 3  
   
 node\_specifications:  
 master:  
 cpu: 16  
 memory: 64Gi  
 storage: 500Gi SSD  
 worker:  
 cpu: 32  
 memory: 128Gi  
 storage: 1Ti NVMe  
 storage:  
 cpu: 8  
 memory: 32Gi  
 storage: 4Ti SSD  
   
 network\_configuration:  
 cni: cilium  
 multi\_network: multus  
 sr\_iov: enabled  
 encryption: wireguard  
   
 virtualization:  
 kubevirt\_version: "v1.1.0"  
 nested\_virtualization: true  
 hugepages: 1Gi  
 cpu\_pinning: enabled

###### Staging/Development Cluster Profile

apiVersion: v1  
kind: ConfigMap  
metadata:  
 name: cluster-profile-staging  
data:  
 profile: |  
 cluster\_type: staging  
 node\_count: 6  
 master\_nodes: 3  
 worker\_nodes: 3  
   
 node\_specifications:  
 master:  
 cpu: 8  
 memory: 32Gi  
 storage: 200Gi SSD  
 worker:  
 cpu: 16  
 memory: 64Gi  
 storage: 500Gi SSD  
   
 network\_configuration:  
 cni: cilium  
 multi\_network: multus  
 sr\_iov: optional  
 encryption: ipsec  
   
 virtualization:  
 kubevirt\_version: "v1.1.0"  
 nested\_virtualization: false  
 hugepages: optional  
 cpu\_pinning: disabled

##### Virtualization Stack Configuration

###### OpenShift Virtualization Deployment

apiVersion: hco.kubevirt.io/v1beta1  
kind: HyperConverged  
metadata:  
 name: kubevirt-hyperconverged  
 namespace: openshift-cnv  
spec:  
 infra:  
 nodePlacement:  
 nodeSelector:  
 node-role.kubernetes.io/worker: ""  
 workloads:  
 nodePlacement:  
 nodeSelector:  
 node-role.kubernetes.io/worker: ""  
 featureGates:  
 enableCommonBootImageImport: true  
 deployTektonTaskResources: true  
 enableApplicationAwareQuota: true  
 configuration:  
 network:  
 networkBinding:  
 plugins:  
 macvtap: {}  
 passt: {}  
 virtualMachineOptions:  
 disableFreePageReporting: false  
 disableSerialConsoleLog: false

##### Multi-Network Configuration

###### Network Attachment Definitions for Different Environments

### Production Network Configuration  
apiVersion: k8s.cni.cncf.io/v1  
kind: NetworkAttachmentDefinition  
metadata:  
 name: prod-management-network  
 namespace: vm-production  
spec:  
 config: |  
 {  
 "cniVersion": "0.3.1",  
 "name": "prod-management-network",  
 "type": "macvlan",  
 "master": "ens192",  
 "mode": "bridge",  
 "ipam": {  
 "type": "static"  
 }  
 }  
---  
### Staging Network Configuration  
apiVersion: k8s.cni.cncf.io/v1  
kind: NetworkAttachmentDefinition  
metadata:  
 name: staging-management-network  
 namespace: vm-staging  
spec:  
 config: |  
 {  
 "cniVersion": "0.3.1",  
 "name": "staging-management-network",  
 "type": "macvlan",  
 "master": "ens192",  
 "mode": "bridge",  
 "vlan": 100,  
 "ipam": {  
 "type": "dhcp"  
 }  
 }

#### Cluster Lifecycle Management

##### Cluster Provisioning Workflow

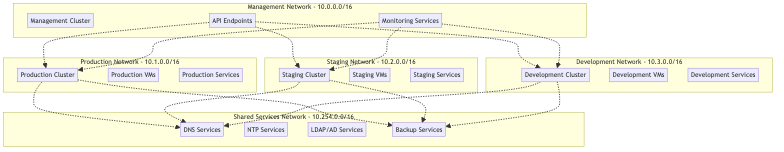


##### Cluster Template

apiVersion: cluster.open-cluster-management.io/v1  
kind: ManagedCluster  
metadata:  
 name: app-cluster-{{ .Values.environment }}-{{ .Values.region }}  
 labels:  
 environment: {{ .Values.environment }}  
 region: {{ .Values.region }}  
 cluster.open-cluster-management.io/clusterset: {{ .Values.clusterset }}  
spec:  
 hubAcceptsClient: true  
 leaseDurationSeconds: 60  
---  
apiVersion: agent.open-cluster-management.io/v1  
kind: KlusterletAddonConfig  
metadata:  
 name: app-cluster-{{ .Values.environment }}-{{ .Values.region }}  
 namespace: app-cluster-{{ .Values.environment }}-{{ .Values.region }}  
spec:  
 clusterName: app-cluster-{{ .Values.environment }}-{{ .Values.region }}  
 clusterNamespace: app-cluster-{{ .Values.environment }}-{{ .Values.region }}  
 clusterLabels:  
 environment: {{ .Values.environment }}  
 region: {{ .Values.region }}  
 applicationManager:  
 enabled: true  
 policyController:  
 enabled: true  
 searchCollector:  
 enabled: true  
 certPolicyController:  
 enabled: true

#### Multi-Cluster Networking

##### Cluster Network Isolation



##### Service Mesh Integration

apiVersion: networking.istio.io/v1beta1  
kind: VirtualService  
metadata:  
 name: cross-cluster-vm-service  
spec:  
 hosts:  
 - vm-service.production.svc.cluster.local  
 gateways:  
 - mesh  
 - cross-cluster-gateway  
 http:  
 - match:  
 - headers:  
 cluster:  
 exact: staging  
 route:  
 - destination:  
 host: vm-service.staging.svc.cluster.local  
 - route:  
 - destination:  
 host: vm-service.production.svc.cluster.local

#### Disaster Recovery and Business Continuity

##### Multi-Cluster Backup Strategy

apiVersion: velero.io/v1  
kind: Schedule  
metadata:  
 name: multi-cluster-backup  
 namespace: velero  
spec:  
 schedule: "0 2 \* \* \*" # Daily at 2 AM  
 template:  
 includedNamespaces:  
 - vm-production  
 - vm-staging  
 - openshift-cnv  
 excludedResources:  
 - pods  
 - replicasets  
 snapshotVolumes: true  
 ttl: 720h # 30 days  
 hooks:  
 resources:  
 - name: vm-backup-hook  
 includedNamespaces:  
 - vm-production  
 - vm-staging  
 labelSelector:  
 matchLabels:  
 backup.kubevirt.io/enable: "true"  
 pre:  
 - exec:  
 container: virt-launcher  
 command:  
 - /bin/bash  
 - -c  
 - "virtctl freeze --namespace $NAMESPACE $VM\_NAME"  
 post:  
 - exec:  
 container: virt-launcher  
 command:  
 - /bin/bash  
 - -c  
 - "virtctl unfreeze --namespace $NAMESPACE $VM\_NAME"

##### Cross-Cluster Failover

apiVersion: cluster.open-cluster-management.io/v1beta1  
kind: Placement  
metadata:  
 name: vm-workload-placement  
 namespace: vm-production  
spec:  
 predicates:  
 - requiredClusterSelector:  
 labelSelector:  
 matchLabels:  
 environment: production  
 region: primary  
 - requiredClusterSelector:  
 labelSelector:  
 matchLabels:  
 environment: production  
 region: secondary  
 numberOfClusters: 2  
 prioritizerPolicy:  
 mode: Additive  
 configurations:  
 - scoreCoordinate:  
 type: BuiltIn  
 builtIn: Steady  
 weight: 1  
 - scoreCoordinate:  
 type: BuiltIn  
 builtIn: ResourceAllocatableCPU  
 weight: 1

#### Scalability and Performance

##### Cluster Auto-Scaling

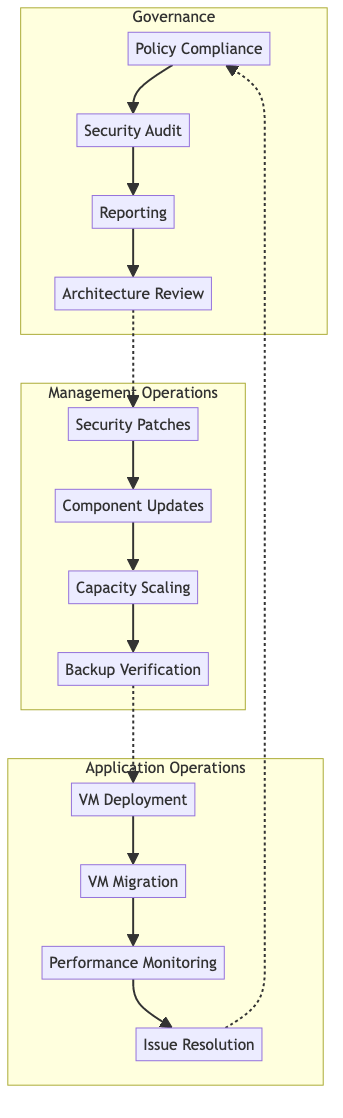
apiVersion: machine.openshift.io/v1beta1  
kind: MachineAutoscaler  
metadata:  
 name: worker-autoscaler  
 namespace: openshift-machine-api  
spec:  
 minReplicas: 3  
 maxReplicas: 20  
 scaleTargetRef:  
 apiVersion: machine.openshift.io/v1beta1  
 kind: MachineSet  
 name: worker-machineset  
---  
apiVersion: autoscaling.openshift.io/v1  
kind: ClusterAutoscaler  
metadata:  
 name: default  
spec:  
 podPriorityThreshold: -10  
 resourceLimits:  
 maxNodesTotal: 50  
 cores:  
 min: 16  
 max: 1000  
 memory:  
 min: 64Gi  
 max: 4000Gi  
 scaleDown:  
 enabled: true  
 delayAfterAdd: 10m  
 delayAfterDelete: 10s  
 delayAfterFailure: 30s  
 unneededTime: 60s

##### VM Resource Management

apiVersion: kubevirt.io/v1  
kind: VirtualMachine  
metadata:  
 name: scalable-vm-template  
 namespace: vm-production  
spec:  
 template:  
 spec:  
 domain:  
 cpu:  
 cores: 4  
 sockets: 1  
 threads: 1  
 memory:  
 guest: 8Gi  
 resources:  
 requests:  
 cpu: 2  
 memory: 4Gi  
 limits:  
 cpu: 4  
 memory: 8Gi  
 devices:  
 autoattachPodInterface: false  
 autoattachSerialConsole: true  
 autoattachGraphicsDevice: true  
 evictionStrategy: LiveMigrate  
 terminationGracePeriodSeconds: 180  
 nodeSelector:  
 node-role.kubernetes.io/worker: ""  
 vm-workload: "true"  
 affinity:  
 podAntiAffinity:  
 preferredDuringSchedulingIgnoredDuringExecution:  
 - weight: 100  
 podAffinityTerm:  
 labelSelector:  
 matchExpressions:  
 - key: vm.kubevirt.io/name  
 operator: Exists  
 topologyKey: kubernetes.io/hostname

#### Operational Procedures

##### Day-2 Operations Workflow



##### Monitoring and Alerting

apiVersion: monitoring.coreos.com/v1  
kind: PrometheusRule  
metadata:  
 name: multi-cluster-alerts  
 namespace: monitoring  
spec:  
 groups:  
 - name: cluster.health  
 rules:  
 - alert: ClusterDown  
 expr: up{job="kubernetes-apiservers"} == 0  
 for: 5m  
 labels:  
 severity: critical  
 annotations:  
 summary: "Cluster {{ $labels.cluster }} is down"  
 description: "Cluster {{ $labels.cluster }} has been down for more than 5 minutes"  
   
 - alert: VMHighMemory  
 expr: kubevirt\_vm\_memory\_usage\_bytes / kubevirt\_vm\_memory\_available\_bytes > 0.9  
 for: 10m  
 labels:  
 severity: warning  
 annotations:  
 summary: "VM {{ $labels.name }} high memory usage"  
 description: "VM {{ $labels.name }} in cluster {{ $labels.cluster }} has high memory usage"  
   
 - alert: VMMigrationFailed  
 expr: increase(kubevirt\_vm\_migration\_failed\_total[5m]) > 0  
 labels:  
 severity: critical  
 annotations:  
 summary: "VM migration failed"  
 description: "VM migration failed in cluster {{ $labels.cluster }}"

#### Best Practices and Recommendations

##### Cluster Design Guidelines

1. **Resource Planning**
   * Size clusters based on workload requirements
   * Plan for 20-30% overhead for system components
   * Consider NUMA topology for high-performance VMs
2. **Network Segmentation**
   * Isolate management and data plane traffic
   * Use VLANs for multi-tenant environments
   * Implement east-west encryption
3. **Storage Strategy**
   * Use local storage for high-performance workloads
   * Implement storage classes for different performance tiers
   * Plan for backup and disaster recovery
4. **Security Architecture**
   * Implement pod security standards
   * Use network policies for microsegmentation
   * Regular security scanning and compliance checks

##### Operational Excellence

1. **GitOps Workflow**
   * All changes through version control
   * Automated testing and validation
   * Rollback capabilities
2. **Monitoring Strategy**
   * Proactive alerting and monitoring
   * Centralized logging and metrics
   * Regular performance reviews
3. **Disaster Recovery**
   * Regular backup testing
   * Cross-region replication
   * Documented recovery procedures

This global architecture overview provides a comprehensive foundation for understanding how the RH OVE ecosystem scales across multiple clusters while maintaining centralized governance, security, and operational efficiency. The architecture supports growth from small deployments to large-scale multi-region installations while preserving consistent management and security practices.

## Overview

### RH OVE Solution Design and Architecture

#### Overview

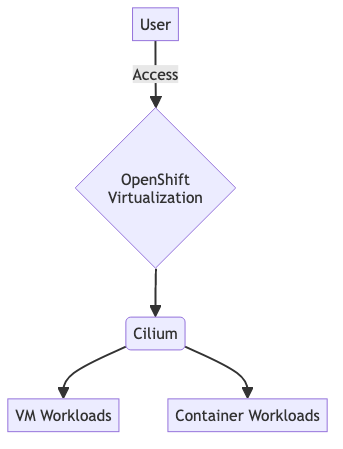
This document provides an overview of the RH OVE solution, detailing the architecture, deployment, and management strategies.

#### Design Principles

* Utilize a namespace-based topology for isolation and security.
* Implement Cilium for network security using eBPF.
* Integrate multiplex workloads to optimize resource utilization.

#### Network Architecture

Mermaid diagram for network architecture:



#### Storage Architecture

Include storage considerations and architecture here.

## Design Principles

### Design Principles

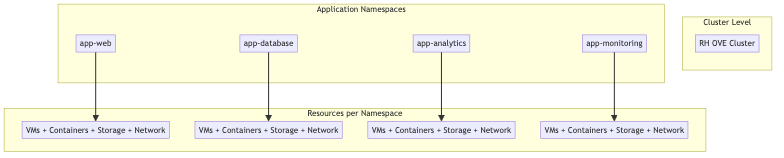
#### Overview

The RH OVE solution is built on fundamental design principles that ensure scalability, security, and operational efficiency for hybrid container and VM workloads.

#### Core Principles

##### 1. Application Namespace-Based Topology

Based on the analysis from our research, using an application namespace-based topology is considered a best practice for RH OVE clusters.

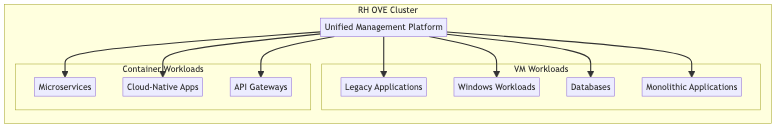


**Benefits:** - **Isolation and Security**: Strong RBAC and network policy enforcement - **Operational Efficiency**: Simplified management and troubleshooting - **Network Segregation**: Namespace-scoped NetworkAttachmentDefinitions - **Scalability**: Prevents resource clutter and performance bottlenecks - **Policy Management**: Granular security policies and quotas

**Implementation:** - Group related VMs and Kubernetes resources by application or business domain - Apply consistent labeling for automation and cost management - Combine with network policies and RBAC rules - Designate separate namespaces for dev, test, and prod environments

##### 2. Mixed Workload Strategy

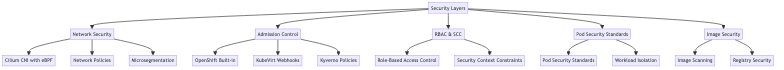
Multiplexing Kubernetes container workloads and VM workloads on the same RH OVE cluster is highly advantageous:



**Advantages:** - **Unified Management**: Same Kubernetes-native interface for all workloads - **Resource Optimization**: Better hardware consolidation - **Flexibility**: Gradual modernization path for legacy applications - **Streamlined DevOps**: Integrated CI/CD pipelines for all workload types - **Advanced Platform Features**: HA, storage provisioning, monitoring for all

##### 3. Security-First Design

Implement defense-in-depth security across all layers:



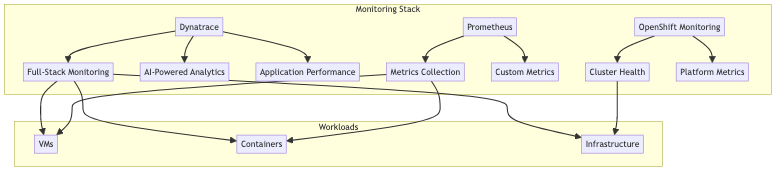
##### 4. GitOps-Driven Operations

Implement infrastructure and application management through GitOps principles:

**Benefits:** - **Single Source of Truth**: All configurations version-controlled in Git - **Declarative Management**: Infrastructure as Code for VMs and containers - **Automation**: Reduced human error through automated deployments - **Auditability**: Complete change tracking and rollback capabilities - **Collaboration**: Peer review through pull requests

##### 5. Observability and Monitoring

Comprehensive monitoring strategy across all workload types:



#### Implementation Guidelines

##### Namespace Design

### Example namespace structure  
apiVersion: v1  
kind: Namespace  
metadata:  
 name: app-web-prod  
 labels:  
 app: web  
 environment: production  
 tier: frontend  
 annotations:  
 network-policy: strict  
 backup-policy: daily

##### Resource Quotas

apiVersion: v1  
kind: ResourceQuota  
metadata:  
 name: compute-quota  
 namespace: app-web-prod  
spec:  
 hard:  
 requests.cpu: "10"  
 requests.memory: 20Gi  
 limits.cpu: "20"  
 limits.memory: 40Gi  
 persistentvolumeclaims: "10"

##### Network Policies

apiVersion: networking.k8s.io/v1  
kind: NetworkPolicy  
metadata:  
 name: app-web-netpol  
 namespace: app-web-prod  
spec:  
 podSelector:  
 matchLabels:  
 app: web  
 policyTypes:  
 - Ingress  
 - Egress  
 ingress:  
 - from:  
 - namespaceSelector:  
 matchLabels:  
 name: app-gateway-prod

#### Best Practices

##### Design Decisions

1. **Plan namespace strategy early**: Define naming conventions and access hierarchies
2. **Implement least privilege**: Use RBAC and network policies consistently
3. **Design for scale**: Consider resource limits and node capacity planning
4. **Plan for disaster recovery**: Include backup and restoration strategies

##### Operational Considerations

1. **Monitor resource utilization**: Prevent resource contention between workload types
2. **Implement proper logging**: Centralized logging for both VMs and containers
3. **Regular security assessments**: Continuous compliance and vulnerability management
4. **Performance testing**: Regular load testing for mixed workload scenarios

These design principles ensure that the RH OVE solution provides a robust, secure, and scalable platform for modern hybrid workloads while supporting organizational transformation initiatives.

## Network

### Network Architecture

#### Overview

The RH OVE network architecture leverages Cilium CNI for enhanced security and observability, providing advanced network capabilities through eBPF technology for both container and VM workloads.

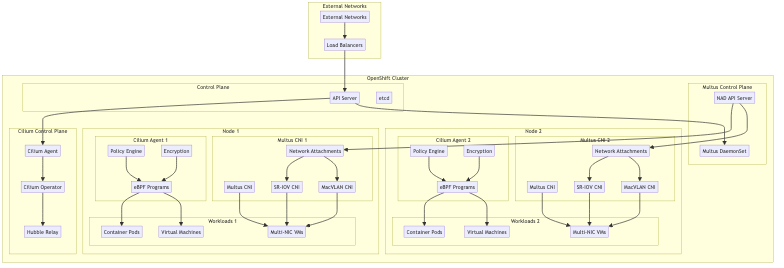
#### Cilium CNI Integration

Based on our research, using Cilium for RH OVE is widely regarded as a strong, future-proof approach with several key advantages:

##### Benefits

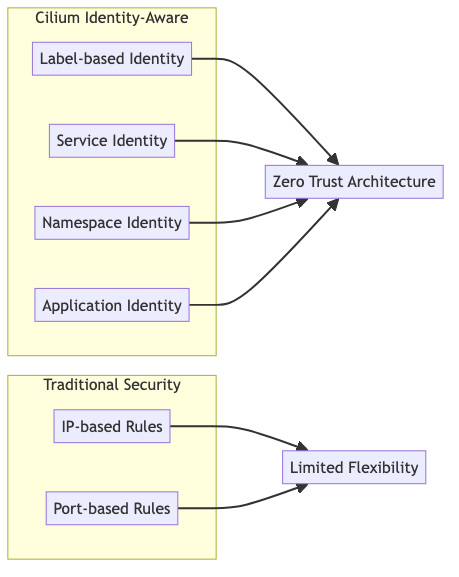
* **Red Hat Certification**: Cilium is a certified CNI plugin for OpenShift
* **eBPF-Powered Enforcement**: Advanced security, visibility, and traffic control
* **Multi-platform Support**: Works for both containers and VMs in hybrid environments
* **High Performance**: Superior performance compared to traditional iptables-based CNIs
* **Service Mesh Capabilities**: L7 security and observability without sidecar proxies

#### Network Architecture Diagram



#### Network Security Features

##### Identity-Aware Security



##### Network Policies

###### Basic Network Policy for VMs

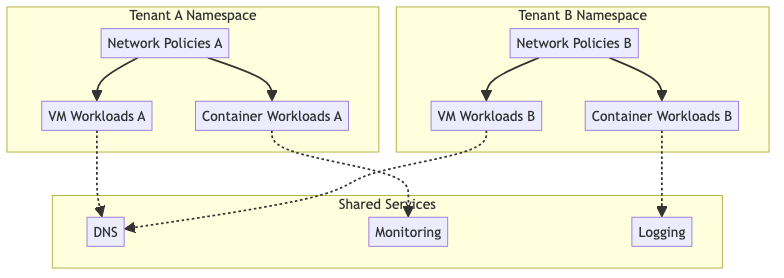
apiVersion: cilium.io/v2  
kind: CiliumNetworkPolicy  
metadata:  
 name: vm-web-policy  
 namespace: app-web-prod  
spec:  
 endpointSelector:  
 matchLabels:  
 app: web-vm  
 ingress:  
 - fromEndpoints:  
 - matchLabels:  
 app: api-gateway  
 toPorts:  
 - ports:  
 - port: "80"  
 protocol: TCP  
 - port: "443"  
 protocol: TCP  
 egress:  
 - toEndpoints:  
 - matchLabels:  
 app: database-vm  
 toPorts:  
 - ports:  
 - port: "5432"  
 protocol: TCP

###### L7 HTTP Policy

apiVersion: cilium.io/v2  
kind: CiliumNetworkPolicy  
metadata:  
 name: l7-http-policy  
spec:  
 endpointSelector:  
 matchLabels:  
 app: web-api  
 ingress:  
 - fromEndpoints:  
 - matchLabels:  
 app: frontend  
 toPorts:  
 - ports:  
 - port: "80"  
 protocol: TCP  
 rules:  
 http:  
 - method: "GET"  
 path: "/api/v1/.\*"  
 - method: "POST"  
 path: "/api/v1/users"

#### Multi-Tenant Networking

##### Namespace Isolation



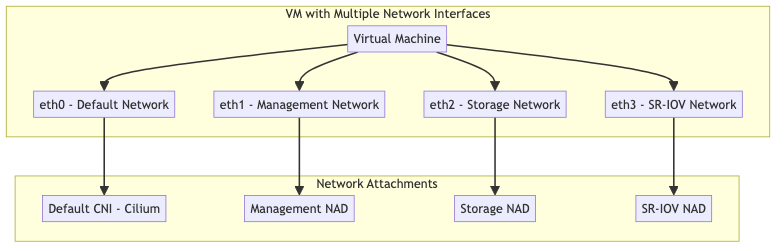
##### NetworkAttachmentDefinition for VMs

apiVersion: k8s.cni.cncf.io/v1  
kind: NetworkAttachmentDefinition  
metadata:  
 name: vm-network  
 namespace: app-database-prod  
spec:  
 config: |  
 {  
 "cniVersion": "0.3.1",  
 "name": "vm-network",  
 "type": "cilium-cni",  
 "ipam": {  
 "type": "cilium"  
 }  
 }

#### Multus Multi-Network Configuration

##### Overview

Multus CNI enables attaching multiple network interfaces to pods and VMs, allowing for complex networking scenarios such as: - Separation of management and data traffic - VLAN-based network segmentation - High-performance networking with SR-IOV - Legacy application networking requirements



##### Multus Installation

Multus is typically installed as part of OpenShift Container Platform:

### Verify Multus is installed  
oc get network.operator.openshift.io cluster -o yaml  
  
### Check Multus DaemonSet  
oc get ds multus -n openshift-multus

##### Network Attachment Definitions (NADs)

###### Management Network NAD

apiVersion: k8s.cni.cncf.io/v1  
kind: NetworkAttachmentDefinition  
metadata:  
 name: management-network  
 namespace: vm-infrastructure  
spec:  
 config: |  
 {  
 "cniVersion": "0.3.1",  
 "name": "management-network",  
 "type": "macvlan",  
 "master": "ens192",  
 "mode": "bridge",  
 "ipam": {  
 "type": "static",  
 "addresses": [  
 {  
 "address": "192.168.100.0/24",  
 "gateway": "192.168.100.1"  
 }  
 ],  
 "dns": {  
 "nameservers": ["192.168.100.10", "8.8.8.8"]  
 }  
 }  
 }

###### Storage Network NAD

apiVersion: k8s.cni.cncf.io/v1  
kind: NetworkAttachmentDefinition  
metadata:  
 name: storage-network  
 namespace: vm-infrastructure  
spec:  
 config: |  
 {  
 "cniVersion": "0.3.1",  
 "name": "storage-network",  
 "type": "macvlan",  
 "master": "ens224",  
 "mode": "bridge",  
 "ipam": {  
 "type": "static",  
 "addresses": [  
 {  
 "address": "10.0.200.0/24"  
 }  
 ]  
 }  
 }

###### VLAN-based Network NAD

apiVersion: k8s.cni.cncf.io/v1  
kind: NetworkAttachmentDefinition  
metadata:  
 name: vlan-100-network  
 namespace: vm-production  
spec:  
 config: |  
 {  
 "cniVersion": "0.3.1",  
 "name": "vlan-100-network",  
 "type": "macvlan",  
 "master": "ens192.100",  
 "mode": "bridge",  
 "ipam": {  
 "type": "dhcp"  
 }  
 }

###### SR-IOV Network NAD

apiVersion: k8s.cni.cncf.io/v1  
kind: NetworkAttachmentDefinition  
metadata:  
 name: sriov-high-performance  
 namespace: vm-production  
spec:  
 config: |  
 {  
 "cniVersion": "0.3.1",  
 "name": "sriov-high-performance",  
 "type": "sriov",  
 "deviceID": "1017",  
 "vf": 0,  
 "ipam": {  
 "type": "static",  
 "addresses": [  
 {  
 "address": "10.0.50.0/24"  
 }  
 ]  
 }  
 }

##### VM Configuration with Multiple Network Cards

###### VM with Multiple Interfaces

apiVersion: kubevirt.io/v1  
kind: VirtualMachine  
metadata:  
 name: multi-network-vm  
 namespace: vm-infrastructure  
 annotations:  
 k8s.v1.cni.cncf.io/networks: |  
 [  
 {  
 "name": "management-network",  
 "ips": ["192.168.100.50/24"]  
 },  
 {  
 "name": "storage-network",  
 "ips": ["10.0.200.50/24"]  
 },  
 {  
 "name": "vlan-100-network"  
 }  
 ]  
spec:  
 running: true  
 template:  
 metadata:  
 labels:  
 app: multi-network-app  
 spec:  
 domain:  
 cpu:  
 cores: 4  
 memory:  
 guest: 8Gi  
 devices:  
 interfaces:  
 - name: default  
 masquerade: {}  
 - name: management  
 bridge: {}  
 - name: storage  
 bridge: {}  
 - name: vlan-network  
 bridge: {}  
 disks:  
 - name: rootdisk  
 disk:  
 bus: virtio  
 resources:  
 requests:  
 cpu: 4  
 memory: 8Gi  
 networks:  
 - name: default  
 pod: {}  
 - name: management  
 multus:  
 networkName: management-network  
 - name: storage  
 multus:  
 networkName: storage-network  
 - name: vlan-network  
 multus:  
 networkName: vlan-100-network  
 volumes:  
 - name: rootdisk  
 dataVolume:  
 name: multi-network-vm-root

###### High-Performance VM with SR-IOV

apiVersion: kubevirt.io/v1  
kind: VirtualMachine  
metadata:  
 name: high-perf-vm  
 namespace: vm-production  
 annotations:  
 k8s.v1.cni.cncf.io/networks: |  
 [  
 {  
 "name": "management-network",  
 "ips": ["192.168.100.100/24"]  
 },  
 {  
 "name": "sriov-high-performance",  
 "ips": ["10.0.50.100/24"]  
 }  
 ]  
spec:  
 running: true  
 template:  
 metadata:  
 labels:  
 app: high-performance-app  
 spec:  
 domain:  
 cpu:  
 cores: 8  
 dedicatedCpuPlacement: true  
 memory:  
 guest: 16Gi  
 hugepages:  
 pageSize: 1Gi  
 devices:  
 interfaces:  
 - name: default  
 masquerade: {}  
 - name: management  
 bridge: {}  
 - name: sriov-net  
 sriov: {}  
 disks:  
 - name: rootdisk  
 disk:  
 bus: virtio  
 resources:  
 requests:  
 cpu: 8  
 memory: 16Gi  
 hugepages-1Gi: 16Gi  
 networks:  
 - name: default  
 pod: {}  
 - name: management  
 multus:  
 networkName: management-network  
 - name: sriov-net  
 multus:  
 networkName: sriov-high-performance  
 volumes:  
 - name: rootdisk  
 dataVolume:  
 name: high-perf-vm-root

##### Advanced Multus Configurations

###### Bond Network Interface

apiVersion: k8s.cni.cncf.io/v1  
kind: NetworkAttachmentDefinition  
metadata:  
 name: bond-network  
 namespace: vm-infrastructure  
spec:  
 config: |  
 {  
 "cniVersion": "0.3.1",  
 "name": "bond-network",  
 "type": "bond",  
 "mode": "active-backup",  
 "miimon": "100",  
 "links": [  
 {"name": "ens192"},  
 {"name": "ens224"}  
 ],  
 "ipam": {  
 "type": "static",  
 "addresses": [  
 {  
 "address": "10.0.100.0/24",  
 "gateway": "10.0.100.1"  
 }  
 ]  
 }  
 }

###### OVS Bridge Network

apiVersion: k8s.cni.cncf.io/v1  
kind: NetworkAttachmentDefinition  
metadata:  
 name: ovs-bridge-network  
 namespace: vm-infrastructure  
spec:  
 config: |  
 {  
 "cniVersion": "0.3.1",  
 "name": "ovs-bridge-network",  
 "type": "ovs",  
 "bridge": "br-data",  
 "vlan": 200,  
 "ipam": {  
 "type": "static",  
 "addresses": [  
 {  
 "address": "10.0.200.0/24"  
 }  
 ]  
 }  
 }

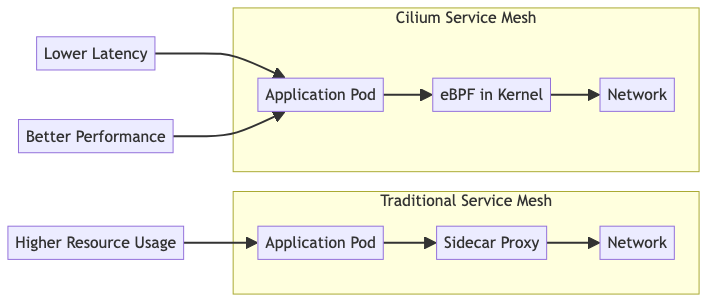
#### Encryption and Security

##### Transparent Encryption

Cilium supports both IPsec and WireGuard for transparent encryption:

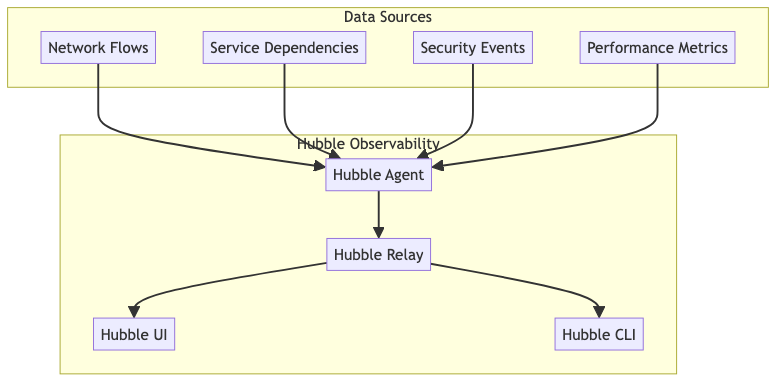
### Cilium ConfigMap for WireGuard encryption  
apiVersion: v1  
kind: ConfigMap  
metadata:  
 name: cilium-config  
 namespace: kube-system  
data:  
 enable-wireguard: "true"  
 wireguard-userspace-fallback: "true"

##### Service Mesh without Sidecars



#### Observability with Hubble

##### Network Visibility



##### Hubble Configuration

apiVersion: v1  
kind: ConfigMap  
metadata:  
 name: cilium-config  
 namespace: kube-system  
data:  
 enable-hubble: "true"  
 hubble-listen-address: ":4244"  
 hubble-socket-path: "/var/run/cilium/hubble.sock"  
 hubble-metrics-server: ":9091"  
 hubble-metrics: >-  
 dns:query;ignoreAAAA  
 drop  
 tcp  
 flow  
 icmp  
 http

#### VM-Specific Networking

##### VM Network Integration

apiVersion: kubevirt.io/v1  
kind: VirtualMachine  
metadata:  
 name: database-vm  
 namespace: app-database-prod  
spec:  
 template:  
 spec:  
 networks:  
 - name: default  
 pod: {}  
 - name: vm-network  
 multus:  
 networkName: vm-network  
 domain:  
 devices:  
 interfaces:  
 - name: default  
 masquerade: {}  
 - name: vm-network  
 bridge: {}

#### Performance Optimization

##### eBPF Performance Benefits

1. **Bypass iptables overhead**: Direct kernel-space processing
2. **Reduced context switches**: Fewer user-space to kernel-space transitions
3. **Optimized packet processing**: Custom eBPF programs for specific workloads
4. **Hardware acceleration**: Support for XDP (eXpress Data Path)

##### Network Performance Tuning

### Cilium DaemonSet configuration for performance  
apiVersion: apps/v1  
kind: DaemonSet  
metadata:  
 name: cilium  
spec:  
 template:  
 spec:  
 containers:  
 - name: cilium-agent  
 args:  
 - --enable-bandwidth-manager=true  
 - --enable-local-redirect-policy=true  
 - --kube-proxy-replacement=strict  
 resources:  
 requests:  
 cpu: 100m  
 memory: 128Mi  
 limits:  
 cpu: 500m  
 memory: 512Mi

#### Integration with External Systems

##### Load Balancer Integration

apiVersion: cilium.io/v2alpha1  
kind: CiliumLoadBalancerIPPool  
metadata:  
 name: vm-pool  
spec:  
 cidrs:  
 - cidr: "10.100.0.0/24"  
---  
apiVersion: v1  
kind: Service  
metadata:  
 name: vm-web-service  
 annotations:  
 io.cilium/lb-ipam-ips: "10.100.0.10"  
spec:  
 type: LoadBalancer  
 selector:  
 app: web-vm  
 ports:  
 - port: 80  
 targetPort: 8080

#### Troubleshooting and Monitoring

##### Network Flow Monitoring

### Monitor network flows  
hubble observe --namespace app-web-prod  
  
### Check policy violations  
hubble observe --verdict DENIED  
  
### Monitor specific VM traffic  
hubble observe --pod vm-database-vm-xxx

##### Performance Metrics

Key metrics to monitor: - Network throughput per VM/pod - Policy enforcement latency - eBPF program execution time - Hubble flow processing rate

This network architecture provides enterprise-grade security, performance, and observability for mixed VM and container workloads in the RH OVE environment.

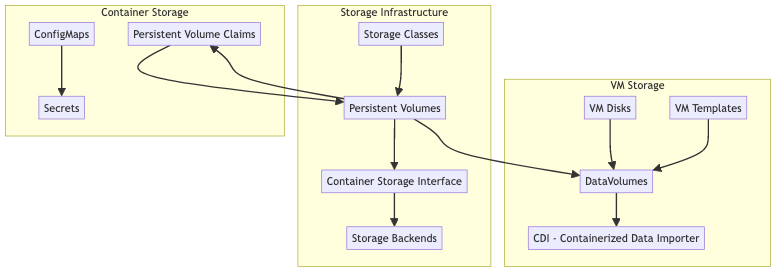
## Storage

### Storage Architecture

#### Overview

The RH OVE storage architecture provides unified storage management for both container and VM workloads, leveraging Kubernetes-native storage concepts while supporting traditional VM storage requirements.

#### Storage Components



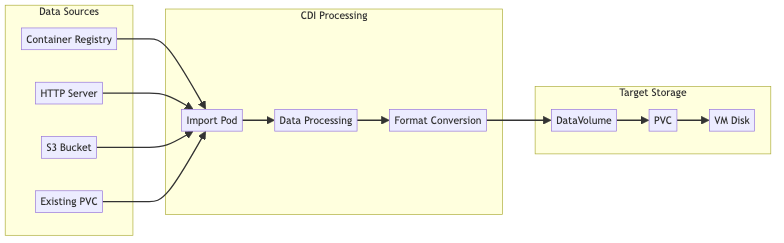
#### DataVolume Management

##### DataVolume CRD

DataVolumes are the primary mechanism for managing VM storage in RH OVE:

apiVersion: cdi.kubevirt.io/v1beta1  
kind: DataVolume  
metadata:  
 name: web-vm-disk  
 namespace: app-web-prod  
spec:  
 pvc:  
 accessModes:  
 - ReadWriteOnce  
 resources:  
 requests:  
 storage: 50Gi  
 storageClassName: fast-ssd  
 source:  
 registry:  
 url: "docker://registry.redhat.io/rhel8/rhel:latest"

##### Storage Import Patterns



#### Storage Classes and Performance Tiers

##### Performance Tiers

### High Performance SSD Storage Class  
apiVersion: storage.k8s.io/v1  
kind: StorageClass  
metadata:  
 name: high-performance-ssd  
provisioner: kubernetes.io/no-provisioner  
parameters:  
 type: ssd  
 iops: "10000"  
 throughput: "500Mi"  
reclaimPolicy: Delete  
volumeBindingMode: WaitForFirstConsumer  
---  
### Standard HDD Storage Class  
apiVersion: storage.k8s.io/v1  
kind: StorageClass  
metadata:  
 name: standard-hdd  
provisioner: kubernetes.io/no-provisioner  
parameters:  
 type: hdd  
 iops: "1000"  
 throughput: "100Mi"  
reclaimPolicy: Retain  
volumeBindingMode: WaitForFirstConsumer  
---  
### Archive Storage Class  
apiVersion: storage.k8s.io/v1  
kind: StorageClass  
metadata:  
 name: archive-storage  
provisioner: kubernetes.io/no-provisioner  
parameters:  
 type: archive  
 iops: "100"  
 throughput: "50Mi"  
reclaimPolicy: Retain  
volumeBindingMode: WaitForFirstConsumer

#### VM Disk Configuration

##### VM with Multiple Disks

apiVersion: kubevirt.io/v1  
kind: VirtualMachine  
metadata:  
 name: database-vm  
 namespace: app-database-prod  
spec:  
 template:  
 spec:  
 domain:  
 devices:  
 disks:  
 - name: rootdisk  
 disk:  
 bus: virtio  
 - name: datadisk  
 disk:  
 bus: virtio  
 - name: logdisk  
 disk:  
 bus: virtio  
 resources:  
 requests:  
 memory: 8Gi  
 cpu: 4  
 volumes:  
 - name: rootdisk  
 dataVolume:  
 name: db-vm-root  
 - name: datadisk  
 dataVolume:  
 name: db-vm-data  
 - name: logdisk  
 dataVolume:  
 name: db-vm-logs

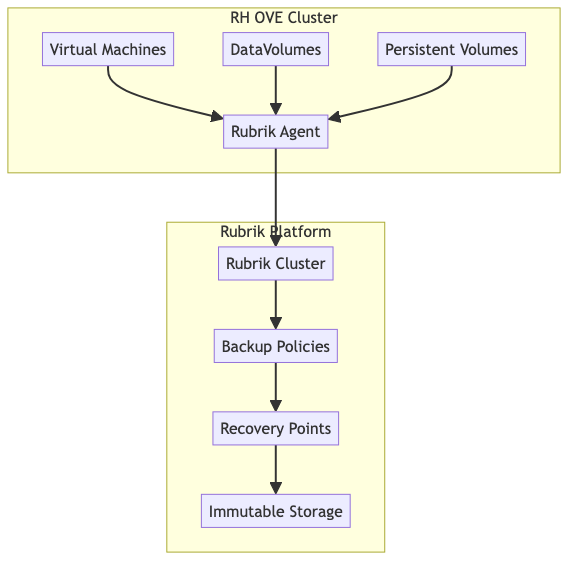
##### DataVolume for Different Use Cases

### Boot disk from registry  
apiVersion: cdi.kubevirt.io/v1beta1  
kind: DataVolume  
metadata:  
 name: db-vm-root  
spec:  
 pvc:  
 accessModes: [ReadWriteOnce]  
 resources:  
 requests:  
 storage: 30Gi  
 storageClassName: high-performance-ssd  
 source:  
 registry:  
 url: "docker://registry.access.redhat.com/rhel8/rhel:latest"  
---  
### Data disk - blank  
apiVersion: cdi.kubevirt.io/v1beta1  
kind: DataVolume  
metadata:  
 name: db-vm-data  
spec:  
 pvc:  
 accessModes: [ReadWriteOnce]  
 resources:  
 requests:  
 storage: 500Gi  
 storageClassName: standard-hdd  
 source:  
 blank: {}  
---  
### Log disk - blank  
apiVersion: cdi.kubevirt.io/v1beta1  
kind: DataVolume  
metadata:  
 name: db-vm-logs  
spec:  
 pvc:  
 accessModes: [ReadWriteOnce]  
 resources:  
 requests:  
 storage: 100Gi  
 storageClassName: standard-hdd  
 source:  
 blank: {}

#### Storage Backup Integration

##### Rubrik Integration for VM Storage

Based on our research, Rubrik provides certified integration with RH OVE for VM backup:



##### Backup Policy Configuration

apiVersion: v1  
kind: ConfigMap  
metadata:  
 name: rubrik-backup-policy  
 namespace: app-database-prod  
data:  
 policy.yaml: |  
 vm\_backup\_policy:  
 name: "database-vm-backup"  
 frequency: "daily"  
 retention: "30d"  
 snapshot\_consistency: "crash-consistent"  
 backup\_window: "02:00-06:00"  
 exclude\_disks:  
 - "temp-disk"  
 - "swap-disk"

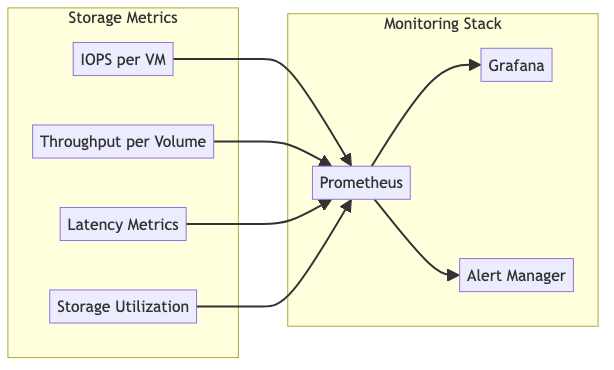
#### Storage Monitoring and Performance

##### Storage Metrics

Key storage metrics to monitor:

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

##### Performance Monitoring



#### Storage Operations

##### Volume Expansion

### Expand a DataVolume  
apiVersion: cdi.kubevirt.io/v1beta1  
kind: DataVolume  
metadata:  
 name: web-vm-disk  
 namespace: app-web-prod  
spec:  
 pvc:  
 accessModes: [ReadWriteOnce]  
 resources:  
 requests:  
 storage: 100Gi # Increased from 50Gi  
 storageClassName: fast-ssd  
 source:  
 pvc:  
 name: web-vm-disk  
 namespace: app-web-prod

##### Volume Cloning

### Clone a DataVolume for VM template  
apiVersion: cdi.kubevirt.io/v1beta1  
kind: DataVolume  
metadata:  
 name: web-vm-template-clone  
 namespace: vm-templates  
spec:  
 pvc:  
 accessModes: [ReadWriteOnce]  
 resources:  
 requests:  
 storage: 50Gi  
 storageClassName: fast-ssd  
 source:  
 pvc:  
 name: web-vm-golden-image  
 namespace: vm-templates

##### Snapshot Management

apiVersion: snapshot.storage.k8s.io/v1  
kind: VolumeSnapshot  
metadata:  
 name: db-vm-snapshot-pre-upgrade  
 namespace: app-database-prod  
spec:  
 volumeSnapshotClassName: csi-snapshotter  
 source:  
 persistentVolumeClaimName: db-vm-data

#### Storage Security

##### Encryption at Rest

apiVersion: storage.k8s.io/v1  
kind: StorageClass  
metadata:  
 name: encrypted-storage  
provisioner: ebs.csi.aws.com  
parameters:  
 type: gp3  
 encrypted: "true"  
 kmsKeyId: "arn:aws:kms:region:account:key/key-id"  
reclaimPolicy: Delete  
volumeBindingMode: WaitForFirstConsumer

##### Access Control

apiVersion: rbac.authorization.k8s.io/v1  
kind: Role  
metadata:  
 namespace: app-database-prod  
 name: storage-admin  
rules:  
- apiGroups: [""]  
 resources: ["persistentvolumeclaims"]  
 verbs: ["get", "list", "create", "update", "patch", "delete"]  
- apiGroups: ["cdi.kubevirt.io"]  
 resources: ["datavolumes"]  
 verbs: ["get", "list", "create", "update", "patch", "delete"]  
- apiGroups: ["snapshot.storage.k8s.io"]  
 resources: ["volumesnapshots"]  
 verbs: ["get", "list", "create", "delete"]

#### Best Practices

##### Storage Planning

1. **Right-size storage**: Match storage performance to workload requirements
2. **Use appropriate storage classes**: Different tiers for different use cases
3. **Plan for growth**: Consider volume expansion capabilities
4. **Backup strategy**: Regular snapshots and external backups

##### Performance Optimization

1. **Use SSD for high-IOPS workloads**: Database and application storage
2. **Separate storage by function**: OS, data, logs, and temp on different volumes
3. **Monitor storage metrics**: Track IOPS, throughput, and latency
4. **Implement storage quotas**: Prevent storage exhaustion

##### Security Considerations

1. **Enable encryption at rest**: For sensitive data storage
2. **Implement access controls**: RBAC for storage resources
3. **Regular security scanning**: Check for storage-related vulnerabilities
4. **Audit storage access**: Monitor who accesses what storage

This storage architecture ensures reliable, performant, and secure storage management for the hybrid VM and container workloads in the RH OVE environment.

## Iam

### Identity and Access Management (IAM) Strategy

#### Overview

This document outlines the comprehensive Identity and Access Management (IAM) strategy for the RH OVE multi-cluster ecosystem, implementing authentication and authorization using OpenID Connect (OIDC) providers with enterprise-grade security controls.

#### Executive Summary

The IAM strategy for the RH OVE multi-cluster ecosystem provides enterprise-grade identity and access management through a comprehensive OIDC-based approach that ensures security, compliance, and operational efficiency across all clusters and services.

##### Key Components

###### 1. **Architecture Components**

* **Mermaid diagram** showing OIDC provider integration across management and application clusters
* **Identity Provider selection** with Keycloak as the recommended solution (Red Hat SSO)
* **Service integration** with ArgoCD, Grafana, Prometheus, and Kubernetes Dashboard
* **Dex OIDC Proxy** deployment for centralized authentication

###### 2. **Authentication Implementation**

* **OpenShift OAuth configuration** with native OIDC integration
* **Dex OIDC Proxy** for service-to-service authentication
* **Multi-Factor Authentication** using Keycloak authentication flows with mandatory MFA for admin accounts
* **JWT token management** with proper security controls and time-limited tokens
* **Single Sign-On (SSO)** seamless access across all clusters and services

###### 3. **Authorization Implementation**

* **Kubernetes RBAC integration** with OIDC groups mapping
* **ArgoCD RBAC configuration** for GitOps access control with application-specific permissions
* **Service Account token management** with time-limited tokens and projected volumes
* **Namespace-scoped permissions** aligned with application teams and business units
* **Role-Based Access Control** with predefined organizational roles

###### 4. **User Lifecycle Management**

* **SCIM integration** for automated user provisioning and deprovisioning
* **Group-based access control** with predefined roles (platform-admins, web-developers, database-admins, security-auditors)
* **Python automation examples** for user management workflows
* **Self-service capabilities** for password reset and account recovery

###### 5. **Security Controls**

* **Token security** with JWT validation rules and proper expiration policies
* **Network security** policies for authentication services with ingress/egress controls
* **Encryption and key management** with AES-256 and regular key rotation
* **Zero Trust Principles** implementation with least-privilege access patterns

###### 6. **Monitoring and Audit**

* **Prometheus metrics** for authentication monitoring and alerting
* **Grafana dashboards** for IAM visibility and operational insights
* **Kubernetes audit policies** for compliance logging and security tracking
* **Automated access reviews** and compliance reporting for SOC 2, GDPR, HIPAA
* **Failed authentication tracking** and security incident response

###### 7. **Disaster Recovery**

* **Identity provider backup** strategies with automated daily backups
* **Multi-region failover** configuration for high availability
* **High availability** for authentication services with automated health checks

##### Technical Implementation Highlights

* **OIDC Provider Integration**: Keycloak (Red Hat SSO) as primary, with Auth0/Okta alternatives
* **Multi-Factor Authentication**: Mandatory for administrative accounts with TOTP/SMS support
* **Service Account Automation**: Time-limited tokens (15min-2hours) with proper lifecycle management
* **Audit and Compliance**: Complete alignment with SOC 2, GDPR, and HIPAA requirements
* **Enterprise Integration**: LDAP/Active Directory federation for existing identity infrastructure

##### Business Benefits

* **Enhanced Security Posture**: Zero trust principles with identity-aware access controls
* **Operational Efficiency**: Centralized identity management across all clusters and services
* **Compliance Readiness**: Automated audit trails and regulatory framework alignment
* **Developer Experience**: Single sign-on with self-service capabilities
* **Cost Optimization**: Reduced operational overhead through automation

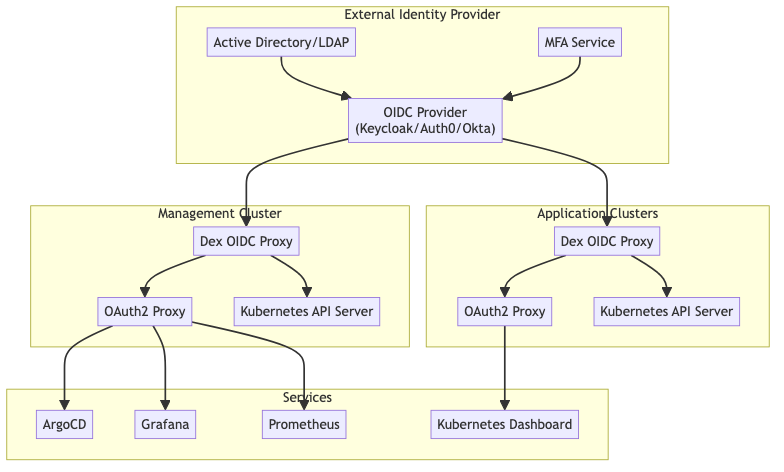
##### Implementation Phases

1. **Phase 1**: Design and Planning (Identity provider selection, architecture design)
2. **Phase 2**: Deployment and Configuration (OIDC integration, RBAC implementation)
3. **Phase 3**: Testing and Validation (Security testing, SSO validation, user acceptance)
4. **Phase 4**: Monitoring and Maintenance (Metrics setup, audit automation, user training)

This IAM strategy ensures robust and flexible identity management, leveraging existing enterprise IAM solutions for seamless integration and compliance within the RH OVE multi-cluster ecosystem while providing a foundation for future growth and regulatory requirements.

#### Architecture Components

##### OIDC Provider Integration



##### Identity Provider Selection

###### Recommended: Keycloak (Red Hat SSO)

* **Advantages**: Open source, Red Hat supported, full OIDC compliance
* **Features**: User federation, social login, fine-grained authorization
* **Integration**: Native OpenShift integration, Kubernetes RBAC mapping

###### Alternative: Enterprise Solutions

* **Auth0**: SaaS solution with extensive integrations
* **Okta**: Enterprise-grade with advanced security features
* **Azure AD**: Microsoft ecosystem integration

#### Authentication Implementation

##### OpenShift OAuth Configuration

apiVersion: config.openshift.io/v1  
kind: OAuth  
metadata:  
 name: cluster  
spec:  
 identityProviders:  
 - name: keycloak-oidc  
 mappingMethod: claim  
 type: OpenID  
 openID:  
 clientID: openshift-cluster  
 clientSecret:  
 name: oidc-client-secret  
 issuer: https://keycloak.company.com/auth/realms/openshift  
 claims:  
 preferredUsername:  
 - preferred\_username  
 name:  
 - name  
 email:  
 - email  
 groups:  
 - groups

##### Dex OIDC Proxy Configuration

apiVersion: v1  
kind: ConfigMap  
metadata:  
 name: dex-config  
 namespace: auth-system  
data:  
 config.yaml: |  
 issuer: https://dex.company.com  
 storage:  
 type: kubernetes  
 config:  
 inCluster: true  
 web:  
 https: 0.0.0.0:5556  
 tlsCert: /etc/dex/tls/tls.crt  
 tlsKey: /etc/dex/tls/tls.key  
 connectors:  
 - type: oidc  
 id: keycloak  
 name: Keycloak  
 config:  
 issuer: https://keycloak.company.com/auth/realms/company  
 clientID: dex-client  
 clientSecret: $DEX\_CLIENT\_SECRET  
 redirectURI: https://dex.company.com/callback  
 scopes:  
 - openid  
 - profile  
 - email  
 - groups  
 staticClients:  
 - id: kubernetes  
 redirectURIs:  
 - https://kubectl.company.com/callback  
 name: 'Kubernetes CLI'  
 secret: $KUBERNETES\_CLIENT\_SECRET  
 - id: argocd  
 redirectURIs:  
 - https://argocd.company.com/auth/callback  
 name: 'ArgoCD'  
 secret: $ARGOCD\_CLIENT\_SECRET

##### Multi-Factor Authentication

### Keycloak Authentication Flow  
authenticationFlows:  
 - alias: "browser-with-mfa"  
 description: "Browser flow with mandatory MFA"  
 providerId: "basic-flow"  
 topLevel: true  
 builtIn: false  
 authenticationExecutions:  
 - authenticator: "auth-cookie"  
 requirement: "ALTERNATIVE"  
 - authenticator: "auth-spnego"  
 requirement: "DISABLED"  
 - authenticator: "identity-provider-redirector"  
 requirement: "ALTERNATIVE"  
 - flowAlias: "forms"  
 requirement: "ALTERNATIVE"  
 - alias: "forms"  
 description: "Username, password, otp and other auth forms."  
 providerId: "basic-flow"  
 topLevel: false  
 builtIn: false  
 authenticationExecutions:  
 - authenticator: "auth-username-password-form"  
 requirement: "REQUIRED"  
 - authenticator: "auth-otp-form"  
 requirement: "REQUIRED"

#### Authorization Implementation

##### Kubernetes RBAC Integration

### ClusterRole for Platform Administrators  
apiVersion: rbac.authorization.k8s.io/v1  
kind: ClusterRole  
metadata:  
 name: platform-admin  
rules:  
- apiGroups: ["\*"]  
 resources: ["\*"]  
 verbs: ["\*"]  
---  
### ClusterRoleBinding with OIDC Groups  
apiVersion: rbac.authorization.k8s.io/v1  
kind: ClusterRoleBinding  
metadata:  
 name: platform-admin-binding  
subjects:  
- kind: Group  
 name: "platform-admins"  
 apiGroup: rbac.authorization.k8s.io  
roleRef:  
 kind: ClusterRole  
 name: platform-admin  
 apiGroup: rbac.authorization.k8s.io  
---  
### Namespace-scoped Role for Application Teams  
apiVersion: rbac.authorization.k8s.io/v1  
kind: Role  
metadata:  
 namespace: app-web-prod  
 name: app-developer  
rules:  
- apiGroups: [""]  
 resources: ["pods", "services", "configmaps", "secrets"]  
 verbs: ["get", "list", "create", "update", "patch", "delete"]  
- apiGroups: ["apps"]  
 resources: ["deployments", "replicasets"]  
 verbs: ["get", "list", "create", "update", "patch", "delete"]  
- apiGroups: ["kubevirt.io"]  
 resources: ["virtualmachines", "virtualmachineinstances"]  
 verbs: ["get", "list", "create", "update", "patch", "delete"]  
---  
### RoleBinding with OIDC Groups  
apiVersion: rbac.authorization.k8s.io/v1  
kind: RoleBinding  
metadata:  
 name: app-developer-binding  
 namespace: app-web-prod  
subjects:  
- kind: Group  
 name: "web-developers"  
 apiGroup: rbac.authorization.k8s.io  
roleRef:  
 kind: Role  
 name: app-developer  
 apiGroup: rbac.authorization.k8s.io

##### ArgoCD RBAC Integration

apiVersion: v1  
kind: ConfigMap  
metadata:  
 name: argocd-rbac-cm  
 namespace: argocd  
data:  
 policy.default: role:readonly  
 policy.csv: |  
 # Platform Administrators  
 g, platform-admins, role:admin  
   
 # Application Teams  
 g, web-developers, role:web-app-admin  
 g, database-admins, role:database-admin  
   
 # Custom Roles  
 role:web-app-admin, applications, \*, app-web-\*/\*, allow  
 role:web-app-admin, repositories, \*, \*, allow  
 role:web-app-admin, certificates, \*, \*, deny  
   
 role:database-admin, applications, \*, app-database-\*/\*, allow  
 role:database-admin, repositories, \*, \*, allow  
 role:database-admin, certificates, \*, \*, deny

##### Service Account Token Management

### Time-limited Service Account Tokens  
apiVersion: v1  
kind: Secret  
metadata:  
 name: build-robot-secret  
 annotations:  
 kubernetes.io/service-account.name: build-robot  
 kubernetes.io/service-account.token-ttl: "3600" # 1 hour  
type: kubernetes.io/service-account-token  
---  
### Projected Service Account Token (Preferred)  
apiVersion: v1  
kind: Pod  
metadata:  
 name: nginx  
spec:  
 serviceAccountName: build-robot  
 containers:  
 - image: nginx  
 name: nginx  
 volumeMounts:  
 - mountPath: /var/run/secrets/tokens  
 name: vault-token  
 volumes:  
 - name: vault-token  
 projected:  
 sources:  
 - serviceAccountToken:  
 path: vault-token  
 expirationSeconds: 7200 # 2 hours  
 audience: vault

#### User Lifecycle Management

##### Automated User Provisioning

### SCIM Integration Example  
class SCIMUserProvisioning:  
 def \_\_init\_\_(self, keycloak\_client, kubernetes\_client):  
 self.keycloak = keycloak\_client  
 self.k8s = kubernetes\_client  
   
 def provision\_user(self, user\_data):  
 # Create user in Keycloak  
 user = self.keycloak.create\_user({  
 "username": user\_data["username"],  
 "email": user\_data["email"],  
 "firstName": user\_data["firstName"],  
 "lastName": user\_data["lastName"],  
 "enabled": True,  
 "groups": user\_data["groups"]  
 })  
   
 # Assign groups based on role  
 for group in user\_data["groups"]:  
 self.keycloak.assign\_group\_to\_user(user["id"], group)  
   
 # Create ServiceAccount if needed  
 if user\_data.get("service\_account"):  
 self.create\_service\_account(user\_data["username"])  
   
 def deprovision\_user(self, username):  
 # Remove from Keycloak  
 user = self.keycloak.get\_user\_by\_username(username)  
 self.keycloak.delete\_user(user["id"])  
   
 # Clean up Kubernetes resources  
 self.cleanup\_user\_resources(username)

##### Group-Based Access Control

### Keycloak Group Configuration  
groups:  
 - name: "platform-admins"  
 description: "Platform administrators with full cluster access"  
 attributes:  
 kubernetes-role: ["cluster-admin"]  
 argocd-role: ["admin"]  
   
 - name: "web-developers"  
 description: "Web application developers"  
 attributes:  
 kubernetes-role: ["app-developer"]  
 kubernetes-namespaces: ["app-web-prod", "app-web-staging", "app-web-dev"]  
 argocd-role: ["web-app-admin"]  
   
 - name: "database-admins"  
 description: "Database administrators"  
 attributes:  
 kubernetes-role: ["app-developer"]  
 kubernetes-namespaces: ["app-database-prod", "app-database-staging"]  
 argocd-role: ["database-admin"]  
   
 - name: "security-auditors"  
 description: "Security team with read-only access"  
 attributes:  
 kubernetes-role: ["view"]  
 argocd-role: ["readonly"]

#### Security Controls

##### Token Security

### JWT Token Configuration  
jwtPolicy:  
 issuer: "https://keycloak.company.com/auth/realms/company"  
 audiences:  
 - "kubernetes"  
 - "argocd"  
 - "grafana"  
 jwksUri: "https://keycloak.company.com/auth/realms/company/protocol/openid\_connect/certs"  
   
### Token Validation Rules  
tokenValidation:  
 expiration:  
 accessToken: 900 # 15 minutes  
 refreshToken: 3600 # 1 hour  
 idToken: 300 # 5 minutes  
   
 claims:  
 required:  
 - "iss"  
 - "aud"  
 - "exp"  
 - "iat"  
 - "sub"  
 groups: "groups"  
 email: "email"  
 name: "name"

##### Network Security

### Network Policy for Auth Services  
apiVersion: networking.k8s.io/v1  
kind: NetworkPolicy  
metadata:  
 name: auth-system-netpol  
 namespace: auth-system  
spec:  
 podSelector:  
 matchLabels:  
 app: dex  
 policyTypes:  
 - Ingress  
 - Egress  
 ingress:  
 - from:  
 - namespaceSelector:  
 matchLabels:  
 name: ingress-nginx  
 ports:  
 - protocol: TCP  
 port: 5556  
 egress:  
 - to: []  
 ports:  
 - protocol: TCP  
 port: 443 # HTTPS to external OIDC provider  
 - to:  
 - namespaceSelector:  
 matchLabels:  
 name: kube-system  
 ports:  
 - protocol: TCP  
 port: 443 # Kubernetes API

#### Monitoring and Audit

##### Authentication Metrics

### Prometheus ServiceMonitor for Dex  
apiVersion: monitoring.coreos.com/v1  
kind: ServiceMonitor  
metadata:  
 name: dex-metrics  
 namespace: auth-system  
spec:  
 selector:  
 matchLabels:  
 app: dex  
 endpoints:  
 - port: metrics  
 interval: 30s  
 path: /metrics  
---  
### Grafana Dashboard Configuration  
dashboard:  
 title: "IAM Authentication Metrics"  
 panels:  
 - title: "Authentication Requests"  
 type: "graph"  
 targets:  
 - expr: "rate(dex\_requests\_total[5m])"  
 legendFormat: "{{method}} {{code}}"  
   
 - title: "Active Sessions"  
 type: "stat"  
 targets:  
 - expr: "dex\_sessions\_active"  
   
 - title: "Failed Logins"  
 type: "graph"  
 targets:  
 - expr: "rate(dex\_requests\_total{code!~"2.."}[5m])"  
 legendFormat: "Failed Authentications"

##### Audit Logging

### Kubernetes Audit Policy  
apiVersion: audit.k8s.io/v1  
kind: Policy  
rules:  
### Log authentication events  
- level: Request  
 namespaces: ["kube-system", "auth-system"]  
 verbs: ["create", "update", "delete"]  
 resources:  
 - group: ""  
 resources: ["secrets", "serviceaccounts"]  
 - group: "rbac.authorization.k8s.io"  
 resources: ["roles", "rolebindings", "clusterroles", "clusterrolebindings"]  
  
### Log user actions in application namespaces  
- level: Metadata  
 namespaces: ["app-web-prod", "app-database-prod"]  
 verbs: ["create", "update", "delete"]  
 users: ["system:serviceaccount:\*"]  
 omitStages:  
 - RequestReceived

##### Compliance Reporting

### Automated Access Review  
class AccessReviewAutomation:  
 def \_\_init\_\_(self, k8s\_client, keycloak\_client):  
 self.k8s = k8s\_client  
 self.keycloak = keycloak\_client  
   
 def generate\_access\_report(self):  
 report = {  
 "timestamp": datetime.now().isoformat(),  
 "users": [],  
 "service\_accounts": [],  
 "orphaned\_resources": []  
 }  
   
 # Get all users from Keycloak  
 keycloak\_users = self.keycloak.get\_all\_users()  
   
 for user in keycloak\_users:  
 user\_report = {  
 "username": user["username"],  
 "email": user["email"],  
 "groups": self.keycloak.get\_user\_groups(user["id"]),  
 "last\_login": user.get("lastAccess"),  
 "kubernetes\_access": self.get\_k8s\_user\_access(user["username"])  
 }  
 report["users"].append(user\_report)  
   
 return report  
   
 def get\_k8s\_user\_access(self, username):  
 # Get user's effective permissions  
 access = []  
   
 # Check ClusterRoleBindings  
 cluster\_bindings = self.k8s.list\_cluster\_role\_binding()  
 for binding in cluster\_bindings.items:  
 if self.user\_in\_binding(username, binding):  
 access.append({  
 "type": "cluster",  
 "role": binding.role\_ref.name,  
 "binding": binding.metadata.name  
 })  
   
 # Check RoleBindings per namespace  
 for namespace in self.get\_user\_namespaces(username):  
 role\_bindings = self.k8s.list\_namespaced\_role\_binding(namespace)  
 for binding in role\_bindings.items:  
 if self.user\_in\_binding(username, binding):  
 access.append({  
 "type": "namespace",  
 "namespace": namespace,  
 "role": binding.role\_ref.name,  
 "binding": binding.metadata.name  
 })  
   
 return access

#### Disaster Recovery

##### Identity Provider Backup

### Keycloak Backup Configuration  
apiVersion: batch/v1  
kind: CronJob  
metadata:  
 name: keycloak-backup  
 namespace: auth-system  
spec:  
 schedule: "0 2 \* \* \*" # Daily at 2 AM  
 jobTemplate:  
 spec:  
 template:  
 spec:  
 containers:  
 - name: backup  
 image: postgres:13  
 command:  
 - /bin/bash  
 - -c  
 - |  
 pg\_dump -h keycloak-db -U keycloak keycloak > /backup/keycloak-$(date +%Y%m%d).sql  
 aws s3 cp /backup/keycloak-$(date +%Y%m%d).sql s3://iam-backups/  
 env:  
 - name: PGPASSWORD  
 valueFrom:  
 secretKeyRef:  
 name: keycloak-db-secret  
 key: password  
 volumeMounts:  
 - name: backup-volume  
 mountPath: /backup  
 volumes:  
 - name: backup-volume  
 emptyDir: {}  
 restartPolicy: OnFailure

##### Failover Configuration

### Multi-Region OIDC Provider Setup  
regions:  
 primary:  
 region: "us-east-1"  
 keycloak\_url: "https://keycloak-primary.company.com"  
 dex\_url: "https://dex-primary.company.com"  
   
 secondary:  
 region: "us-west-2"  
 keycloak\_url: "https://keycloak-secondary.company.com"  
 dex\_url: "https://dex-secondary.company.com"  
  
### Automated Failover Logic  
failover:  
 health\_check\_interval: 30s  
 failure\_threshold: 3  
 recovery\_threshold: 2  
 dns\_ttl: 60 # Low TTL for quick failover

This comprehensive IAM strategy provides enterprise-grade identity and access management for the RH OVE multi-cluster ecosystem, ensuring security, compliance, and operational efficiency through OIDC-based authentication and Kubernetes-native authorization.

## Adr Table

### Architecture Decision Records (ADR) Table

This document provides a comprehensive overview of all architectural decisions made for the RH OVE multi-cluster ecosystem.

#### ADR Summary Table

| ADR | Title | Status | Date | Context | Decision |
| --- | --- | --- | --- | --- | --- |
| [ADR-001](adr-001-multi-cluster-pattern.md) | Multi-Cluster Architecture Pattern | Accepted | 2024-12-01 | Need to support multiple environments with centralized governance and scalable infrastructure | Implement multi-cluster pattern with one management cluster and multiple application clusters |
| [ADR-002](adr-002-gitops-argocd.md) | GitOps with ArgoCD Hub Architecture | Accepted | 2024-12-01 | Require consistent, auditable, scalable deployment approach across multiple clusters | Implement GitOps using ArgoCD in hub-spoke pattern with Git-based configuration |
| [ADR-003](adr-003-cluster-topology.md) | Namespace-Based Cluster Topology | Accepted | 2024-12-01 | Need efficient organizational strategy for mixed VM and container workloads with isolation and security | Implement application namespace-based topology organized by business application |
| [ADR-004](adr-004-admission-controller.md) | Admission Controller Strategy | Accepted | 2024-12-01 | Require flexible, secure, policy-driven approach for resource admission and validation | Implement layered admission control using OpenShift built-in controllers, KubeVirt webhooks, and Kyverno policies |
| [ADR-005](adr-005-network-cni.md) | Cilium CNI with Multus Multi-Network Strategy | Accepted | 2024-12-01 | Need advanced networking for container and VM workloads with enterprise-grade security and performance | Implement Cilium as primary CNI with Multus for multi-network support using eBPF-powered networking |
| [ADR-006](adr-006-backup-strategy.md) | Backup Strategy for RH OVE Ecosystem | Accepted | 2024-12-01 | Ensure data protection and recovery across multi-cluster environment with business continuity requirements | Adopt centralized backup strategy using Rubrik for VM and containerized workloads |
| [ADR-007](adr-007-monitoring-strategy.md) | Monitoring Strategy for RH OVE Ecosystem | Accepted | 2024-12-01 | Need comprehensive monitoring for operational visibility, performance management, and incident response | Implement integrated monitoring using Prometheus/Grafana, Dynatrace, and Hubble |
| [ADR-008](adr-008-iam-strategy.md) | Identity and Access Management (IAM) Strategy | Accepted | 2024-12-01 | Need enterprise-grade identity and access management across multi-cluster ecosystem | Implement comprehensive IAM using OIDC providers with Keycloak, integrated with Kubernetes RBAC |

#### Detailed ADR Information

##### ADR-001: Multi-Cluster Architecture Pattern

**Key Components:** - **Management Cluster**: RHACM, ArgoCD Hub, RHACS, Federated Prometheus, Centralized logging, Rubrik backup management - **Application Clusters**: Production (HA, performance-optimized), Staging (production-like), Development (resource-optimized) - **Network Architecture**: Dedicated segments per cluster, VPN/private connectivity, zero-trust principles

**Benefits:** Separation of concerns, horizontal scalability, security isolation, operational efficiency, fault isolation, resource optimization

**Trade-offs:** Increased network complexity, additional operational overhead, potential data sync challenges

##### ADR-002: GitOps with ArgoCD Hub Architecture

**Key Components:** - **ArgoCD Hub**: Centralized instance in management cluster with HA (3 replicas) - **ArgoCD Agents**: Lightweight agents in application clusters - **Repository Structure**: Clusters, applications (base/overlays), infrastructure (networking/storage/monitoring) - **Application of Applications Pattern**: Root ArgoCD app manages cluster-specific applications

**Benefits:** Declarative configuration, complete audit trail, pull-based security, consistency, easy rollbacks, self-healing

**Trade-offs:** Learning curve for GitOps workflows, Git repository complexity, network dependencies, secret management complexity

##### ADR-003: Namespace-Based Cluster Topology

**Key Components:** - **Naming Convention**: {app-name}-{environment} (e.g., app-web-prod, app-database-staging) - **Standard Templates**: Namespace with labels, ResourceQuota, LimitRange, NetworkPolicies - **Cross-Namespace Communication**: Controlled via NetworkPolicies with explicit allow rules - **RBAC Integration**: Namespace-level roles aligned with application teams

**Benefits:** Strong isolation, simplified RBAC, clear resource attribution, network microsegmentation, operational clarity, compliance alignment

**Trade-offs:** Initial complexity in planning boundaries, cross-app dependency management, shared services challenges

##### ADR-004: Admission Controller Strategy

**Key Components:** - **OpenShift Built-in**: Security Context Constraints, RBAC enforcement, quotas and limits - **KubeVirt Webhooks**: Validation and mutation webhooks for VM specifications - **Kyverno Policies**: Configuration validation, resource constraints, dynamic policy application

**Benefits:** Centralized policy management, dynamic policy application, security enforcement, misconfiguration prevention, extensibility

**Trade-offs:** Complex rule management, performance overhead, learning curve for policy authors

##### ADR-005: Cilium CNI with Multus Multi-Network Strategy

**Key Components:** - **Cilium Features**: eBPF performance, identity-aware security, L7 security, service mesh capabilities, WireGuard encryption - **Multus Integration**: Multi-network support, legacy network integration, SR-IOV for high performance, network segmentation - **Hubble Observability**: Network flow monitoring, policy violation detection, security auditing - **NetworkAttachmentDefinitions**: Management, storage, and data networks with VLAN support

**Benefits:** Superior eBPF performance (10-100x better than iptables), identity-aware policies, L7 security without sidecars, deep observability, multi-network support

**Trade-offs:** Learning curve for eBPF concepts, debugging complexity, higher memory usage, potential compatibility issues

##### ADR-006: Backup Strategy for RH OVE Ecosystem

**Key Components:** - **Rubrik Management**: Centralized in management cluster with unified policy management - **Backup Architecture**: Management cluster (Rubrik node), Application clusters (Edge devices, agents), Cloud archive (long-term retention) - **Policy Configuration**: Daily backups (24h RPO), weekly full with daily incrementals, AES-256 encryption, cloud replication - **Compliance**: GDPR, HIPAA, SOC 2 alignment with audit trails and access control

**Benefits:** Unified management, policy-driven, deduplication/compression, cloud integration, application consistency

**Trade-offs:** Higher upfront costs than open-source alternatives, training requirements for administrators

##### ADR-007: Monitoring Strategy for RH OVE Ecosystem

**Key Components:** - **Prometheus/Grafana**: Scalable metrics collection, customizable dashboards, real-time metrics, integrated alerting - **Dynatrace**: Full-stack monitoring, AI-powered analytics, cloud-native support, unified observability - **Hubble**: eBPF-powered network insights, high throughput flow capture, Cilium integration - **Integration**: Federated Prometheus (3 replicas, 500Gi storage), OAuth SSO, automated tagging

**Benefits:** Operational efficiency with reduced MTTR, proactive performance management, unified observability across clusters

**Trade-offs:** Integration complexity, resource overhead, training requirements for multiple tools

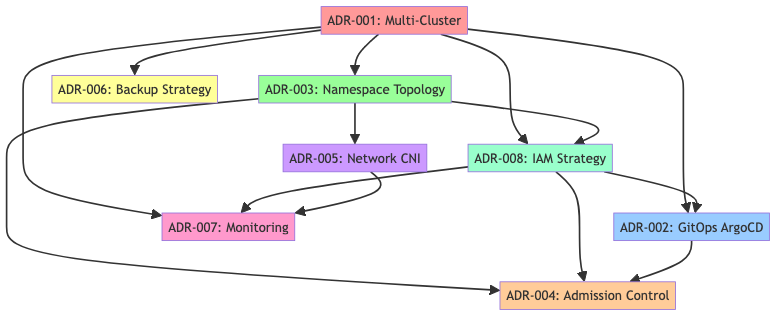
##### ADR-008: Identity and Access Management (IAM) Strategy

**Key Components:** - **Keycloak (Red Hat SSO)**: Primary OIDC provider with LDAP/AD integration, MFA support, group-based access control - **Dex OIDC Proxy**: Service authentication across clusters with static client configuration - **OpenShift OAuth Integration**: Native cluster authentication with OIDC claims mapping - **RBAC Integration**: Kubernetes-native authorization with group-based role assignments - **Service Account Management**: Time-limited tokens with projected volumes and automated lifecycle

**Benefits:** Centralized identity management, single sign-on across all services, enterprise LDAP/AD integration, MFA enforcement, zero trust security, complete audit trails, compliance ready (SOC 2, GDPR, HIPAA)

**Trade-offs:** Initial setup complexity, additional infrastructure dependencies, OIDC/Keycloak learning curve, identity provider availability critical, token lifecycle management complexity

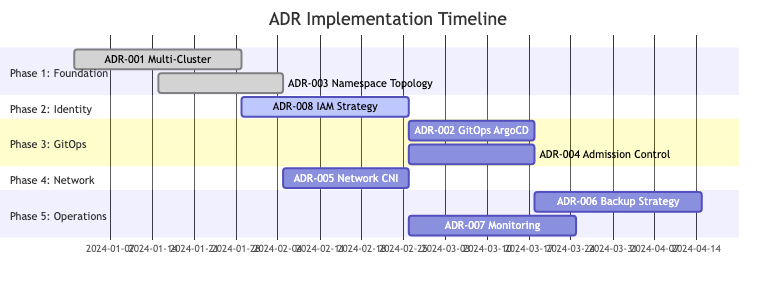
#### Cross-ADR Dependencies



#### Implementation Timeline

| Phase | ADRs | Duration | Dependencies | Key Deliverables |
| --- | --- | --- | --- | --- |
| **Phase 1: Foundation** | ADR-001, ADR-003 | 4-6 weeks | Infrastructure setup | Multi-cluster setup, namespace topology |
| **Phase 2: Identity & Access** | ADR-008 | 3-4 weeks | Foundation complete | Keycloak deployment, OIDC integration, MFA setup |
| **Phase 3: GitOps & Policy** | ADR-002, ADR-004 | 3-4 weeks | Foundation + IAM complete | ArgoCD hub with OIDC auth, admission controllers |
| **Phase 4: Networking** | ADR-005 | 2-3 weeks | Foundation complete | Cilium CNI, Multus, network policies |
| **Phase 5: Operations** | ADR-006, ADR-007 | 4-5 weeks | All previous phases | Backup strategy, monitoring with IAM integration |

##### Detailed Phase Dependencies



##### Critical Path Analysis

**Critical Dependencies:** - **ADR-008 (IAM)** must be completed before GitOps and Admission Control implementation - **ADR-003 (Namespace Topology)** enables proper RBAC integration with IAM - **ADR-007 (Monitoring)** requires IAM integration for authentication and authorization - **ADR-002 (GitOps)** requires IAM for secure access control and audit trails

**Parallel Implementation Opportunities:** - ADR-005 (Network CNI) can be implemented in parallel with IAM setup - ADR-006 (Backup) and ADR-007 (Monitoring) can be implemented concurrently in final phase

This comprehensive table provides a complete overview of all architectural decisions, their relationships, and implementation considerations for the RH OVE multi-cluster ecosystem.

## Adr 001 Multi Cluster Pattern

### ADR-001: Multi-Cluster Architecture Pattern

#### Status

Accepted

#### Date

2024-12-01

#### Context

The RH OVE ecosystem needs to support multiple environments (production, staging, development) while maintaining centralized governance, security, and operational oversight. The organization requires scalable infrastructure that can grow horizontally and support geographic distribution.

#### Decision

We will implement a multi-cluster architecture pattern with: - **One Management Cluster**: Centralized control plane for governance, GitOps, security, and monitoring - **Multiple Application Clusters**: Dedicated workload execution environments per environment type

#### Rationale

##### Advantages

1. **Separation of Concerns**: Clear boundaries between management and workload execution
2. **Scalability**: Horizontal scaling by adding application clusters as needed
3. **Security**: Network-level isolation between environments
4. **Operational Efficiency**: Centralized management reduces operational overhead
5. **Fault Isolation**: Issues in one cluster don’t affect others
6. **Resource Optimization**: Right-size clusters based on workload requirements

##### Alternatives Considered

1. **Single Large Cluster**: Rejected due to blast radius and resource contention
2. **Completely Separate Clusters**: Rejected due to operational complexity and lack of centralized governance
3. **Namespace-based Multi-tenancy**: Rejected due to insufficient isolation for production workloads

#### Implementation Details

##### Management Cluster Components

* Red Hat Advanced Cluster Management (RHACM)
* ArgoCD Hub for GitOps
* Red Hat Advanced Cluster Security (RHACS)
* Federated Prometheus for monitoring
* Centralized logging aggregation
* Rubrik backup management

##### Application Cluster Types

* **Production**: High-availability, performance-optimized
* **Staging**: Production-like for testing
* **Development**: Resource-optimized for development workflows

##### Network Architecture

* Dedicated network segments per cluster type
* VPN/Private connectivity between management and application clusters
* Zero-trust network principles

#### Consequences

##### Positive

* Improved security posture through cluster-level isolation
* Simplified compliance and audit processes
* Better resource utilization and cost optimization
* Enhanced disaster recovery capabilities
* Reduced blast radius for security incidents

##### Negative

* Increased network complexity
* Additional operational overhead for cluster lifecycle management
* Potential data synchronization challenges
* Learning curve for multi-cluster operations

#### Compliance Considerations

* Meets enterprise security requirements for environment isolation
* Supports regulatory compliance through audit trail separation
* Enables data residency requirements through geographic cluster placement

#### Monitoring and Observability

* Centralized metrics collection via Prometheus federation
* Unified logging through log forwarding to management cluster
* Cross-cluster distributed tracing capabilities
* Centralized alerting and incident management

## Adr 002 Gitops Argocd

### ADR-002: GitOps with ArgoCD Hub Architecture

#### Status

Accepted

#### Date

2024-12-01

#### Context

The multi-cluster RH OVE ecosystem requires a consistent, auditable, and scalable approach to application deployment and configuration management across multiple clusters. Traditional CI/CD approaches with push-based deployments create security concerns and operational complexity in multi-cluster environments.

#### Decision

We will implement GitOps using ArgoCD in a hub-spoke pattern: - **ArgoCD Hub**: Centralized ArgoCD instance in the management cluster - **ArgoCD Agents**: Lightweight agents in each application cluster - **Git-based Configuration**: All infrastructure and application configurations stored in Git repositories

#### Rationale

##### Advantages

1. **Declarative Configuration**: Infrastructure and applications defined as code
2. **Audit Trail**: Complete Git history of all changes
3. **Security**: Pull-based model eliminates need for external access to clusters
4. **Consistency**: Identical deployment processes across all environments
5. **Rollback Capability**: Easy rollback using Git revert operations
6. **Self-Healing**: Automatic drift correction and reconciliation

##### Alternatives Considered

1. **Jenkins-based CI/CD**: Rejected due to security concerns with push-based deployments
2. **Tekton Pipelines**: Rejected due to complexity in multi-cluster scenarios
3. **Fleet by Rancher**: Rejected due to vendor lock-in concerns
4. **Flux**: Rejected due to preference for ArgoCD’s UI and workflow capabilities

#### Implementation Details

##### ArgoCD Hub Configuration

apiVersion: argoproj.io/v1alpha1  
kind: ArgoCD  
metadata:  
 name: argocd-hub  
 namespace: argocd  
spec:  
 server:  
 replicas: 3  
 route:  
 enabled: true  
 tls:  
 termination: reencrypt  
 ha:  
 enabled: true  
 dex:  
 openShiftOAuth: true  
 rbac:  
 defaultPolicy: 'role:readonly'

##### Repository Structure

gitops-repo/  
├── clusters/  
│ ├── management/  
│ ├── production/  
│ ├── staging/  
│ └── development/  
├── applications/  
│ ├── base/  
│ └── overlays/  
└── infrastructure/  
 ├── networking/  
 ├── storage/  
 └── monitoring/

##### Application of Applications Pattern

* Root ArgoCD Application manages cluster-specific applications
* Environment-specific overlays using Kustomize
* Automated sync policies for non-production environments
* Manual sync for production deployments

#### Consequences

##### Positive

* **Enhanced Security**: No direct cluster access required for deployments
* **Improved Compliance**: Complete audit trail through Git history
* **Reduced Operational Overhead**: Automated deployment and drift correction
* **Better Collaboration**: Git-based workflows familiar to development teams
* **Disaster Recovery**: Easy recreation of cluster state from Git

##### Negative

* **Learning Curve**: Teams need to adapt to GitOps workflows
* **Git Repository Complexity**: Large repositories can become difficult to manage
* **Network Dependencies**: Requires reliable connectivity to Git repositories
* **Secret Management**: Additional complexity for managing sensitive data

#### Security Considerations

* ArgoCD service accounts use minimal required permissions
* Private Git repositories with SSH key authentication
* RBAC integration with OpenShift OAuth
* Secret management through External Secrets Operator
* Network policies restrict ArgoCD communication

#### Monitoring and Alerting

* ArgoCD application health monitoring
* Git repository sync status tracking
* Deployment success/failure notifications
* Performance metrics for sync operations
* Custom dashboards for GitOps workflows

#### Migration Strategy

1. **Phase 1**: Deploy ArgoCD hub in management cluster
2. **Phase 2**: Migrate existing applications to Git repositories
3. **Phase 3**: Configure ArgoCD applications for each cluster
4. **Phase 4**: Implement automated sync for non-production
5. **Phase 5**: Train teams on GitOps workflows

## Adr 003 Cluster Topology

### ADR-003: Namespace-Based Cluster Topology

#### Status

Accepted

#### Date

2024-12-01

#### Context

The RH OVE ecosystem requires an efficient organizational strategy for managing mixed VM and container workloads within clusters. We need to balance isolation, security, resource management, and operational simplicity while supporting multi-tenant use cases.

#### Decision

We will implement an **application namespace-based topology** where resources are organized by business application or domain rather than by resource type or technology stack.

##### Topology Structure

Cluster  
├── app-web-prod (namespace)  
│ ├── VMs (web servers)  
│ ├── Containers (microservices)  
│ ├── Storage (PVCs, DataVolumes)  
│ └── Network (NetworkPolicies, NADs)  
├── app-database-prod (namespace)  
│ ├── VMs (database servers)  
│ ├── Containers (database operators)  
│ ├── Storage (high-performance storage)  
│ └── Network (isolated database networks)  
└── app-monitoring-prod (namespace)  
 ├── VMs (legacy monitoring tools)  
 ├── Containers (modern observability stack)  
 ├── Storage (metrics and logs storage)  
 └── Network (monitoring networks)

#### Rationale

##### Advantages

1. **Strong Isolation**: Each application has its own security boundary
2. **Simplified RBAC**: Teams get namespace-level access aligned with their applications
3. **Clear Resource Attribution**: Easy to track costs and resource usage per application
4. **Network Microsegmentation**: Network policies can be application-specific
5. **Operational Clarity**: Troubleshooting and maintenance scoped to business context
6. **Compliance Alignment**: Audit boundaries match business applications

##### Alternatives Considered

###### 1. Technology-Based Topology

* **Structure**: Separate namespaces for VMs vs containers
* **Rejected**: Creates artificial barriers between related workloads
* **Issues**: Complex cross-namespace communication, unclear ownership

###### 2. Environment-Based Topology Only

* **Structure**: Single namespace per environment (prod, staging, dev)
* **Rejected**: Poor isolation between different applications
* **Issues**: Resource contention, security boundary concerns

###### 3. Team-Based Topology

* **Structure**: Namespaces per team/department
* **Rejected**: Teams often work on multiple applications
* **Issues**: Unclear application boundaries, resource conflicts

#### Implementation Details

##### Namespace Naming Convention

### Pattern: {app-name}-{environment}  
### Examples:  
- app-web-prod  
- app-web-staging   
- app-web-dev  
- app-database-prod  
- app-analytics-prod  
- shared-monitoring-prod  
- shared-storage-prod

##### Standard Namespace Template

apiVersion: v1  
kind: Namespace  
metadata:  
 name: app-web-prod  
 labels:  
 application: web  
 environment: production  
 tier: frontend  
 cost-center: "12345"  
 owner: web-team  
 annotations:  
 backup-policy: "daily"  
 monitoring-enabled: "true"  
 network-policy: "strict"  
 compliance-level: "high"  
spec:  
 finalizers:  
 - kubernetes  
---  
### Resource Quota per namespace  
apiVersion: v1  
kind: ResourceQuota  
metadata:  
 name: compute-quota  
 namespace: app-web-prod  
spec:  
 hard:  
 requests.cpu: "20"  
 requests.memory: 40Gi  
 limits.cpu: "40"  
 limits.memory: 80Gi  
 persistentvolumeclaims: "20"  
 services: "10"  
 secrets: "20"  
 configmaps: "20"  
---  
### Limit Range per namespace  
apiVersion: v1  
kind: LimitRange  
metadata:  
 name: resource-limits  
 namespace: app-web-prod  
spec:  
 limits:  
 - default:  
 cpu: "2"  
 memory: 4Gi  
 defaultRequest:  
 cpu: 100m  
 memory: 128Mi  
 type: Container  
 - default:  
 cpu: "8"  
 memory: 16Gi  
 defaultRequest:  
 cpu: "2"  
 memory: 4Gi  
 type: PersistentVolumeClaim

##### Cross-Namespace Communication Policy

apiVersion: networking.k8s.io/v1  
kind: NetworkPolicy  
metadata:  
 name: allow-app-communication  
 namespace: app-web-prod  
spec:  
 podSelector: {}  
 policyTypes:  
 - Ingress  
 - Egress  
 ingress:  
 # Allow ingress from gateway namespace  
 - from:  
 - namespaceSelector:  
 matchLabels:  
 name: shared-gateway-prod  
 egress:  
 # Allow egress to database namespace  
 - to:  
 - namespaceSelector:  
 matchLabels:  
 application: database  
 environment: production  
 ports:  
 - protocol: TCP  
 port: 5432  
 # Allow egress to shared services  
 - to:  
 - namespaceSelector:  
 matchLabels:  
 tier: shared-services

#### Governance and Management

##### Namespace Lifecycle Management

apiVersion: argoproj.io/v1alpha1  
kind: Application  
metadata:  
 name: namespace-manager  
 namespace: argocd  
spec:  
 project: infrastructure  
 source:  
 repoURL: https://git.company.com/infrastructure/namespaces  
 targetRevision: HEAD  
 path: namespaces  
 destination:  
 server: https://kubernetes.default.svc  
 syncPolicy:  
 automated:  
 prune: false  
 selfHeal: true  
 syncOptions:  
 - CreateNamespace=true

##### RBAC Template per Namespace

apiVersion: rbac.authorization.k8s.io/v1  
kind: Role  
metadata:  
 namespace: app-web-prod  
 name: app-admin  
rules:  
- apiGroups: [""]  
 resources: ["\*"]  
 verbs: ["\*"]  
- apiGroups: ["apps", "extensions"]  
 resources: ["\*"]  
 verbs: ["\*"]  
- apiGroups: ["kubevirt.io"]  
 resources: ["\*"]  
 verbs: ["\*"]  
- apiGroups: ["cdi.kubevirt.io"]  
 resources: ["\*"]  
 verbs: ["\*"]  
---  
apiVersion: rbac.authorization.k8s.io/v1  
kind: RoleBinding  
metadata:  
 name: app-admin-binding  
 namespace: app-web-prod  
subjects:  
- kind: Group  
 name: web-team  
 apiGroup: rbac.authorization.k8s.io  
roleRef:  
 kind: Role  
 name: app-admin  
 apiGroup: rbac.authorization.k8s.io

#### Consequences

##### Positive

* **Clear Ownership**: Each namespace has a clear business owner
* **Improved Security**: Strong isolation boundaries between applications
* **Simplified Operations**: Easier to manage, monitor, and troubleshoot
* **Better Resource Management**: Clear resource attribution and quota management
* **Compliance Ready**: Audit boundaries align with business applications

##### Negative

* **Initial Complexity**: Requires careful planning of namespace boundaries
* **Cross-App Dependencies**: Need clear policies for inter-namespace communication
* **Shared Services Challenge**: Need strategy for common services (monitoring, logging)
* **Learning Curve**: Teams need to understand namespace-based organization

#### Migration Strategy

##### Phase 1: Planning and Design

1. Inventory existing applications and their dependencies
2. Define namespace naming conventions and standards
3. Create RBAC and network policy templates

##### Phase 2: Shared Services Migration

1. Create shared services namespaces (monitoring, logging, gateway)
2. Migrate common infrastructure components
3. Establish cross-namespace communication patterns

##### Phase 3: Application Migration

1. Start with least critical applications
2. Create application-specific namespaces with proper quotas and policies
3. Migrate workloads and validate functionality

##### Phase 4: Governance Implementation

1. Implement automated namespace provisioning
2. Enable monitoring and alerting per namespace
3. Create operational runbooks for namespace management

#### Monitoring and Compliance

##### Namespace-Level Metrics

* Resource utilization per namespace
* Cost attribution per application
* Security policy violations
* Cross-namespace communication patterns

##### Compliance Reporting

* Resource usage reports per business unit
* Security posture per application
* Audit logs scoped to business context
* Data residency compliance per namespace

This topology provides a solid foundation for managing complex multi-tenant RH OVE environments while maintaining security, operational clarity, and business alignment.

## Adr 004 Admission Controller

### ADR-004: Admission Controller Strategy

#### Status

Accepted

#### Date

2024-12-01

#### Context

The RH OVE ecosystem requires a flexible, secure, and policy-driven approach for managing resource admission and validation within clusters. Implementing appropriate admission control policies ensures compliance, security, and operational consistency.

#### Decision

We will implement a layered admission control strategy utilizing OpenShift’s built-in admission controllers, KubeVirt webhooks, Kyverno policies, and OIDC-integrated RBAC enforcement via Keycloak.

#### Rationale

##### Advantages

1. **Centralized Policy Management**: Simplify governance with cluster-wide policies
2. **Dynamic Policy Application**: Adjust policies without redeploying cluster components
3. **Security Enforcement**: Validate resource configurations before persistence
4. **Prevention of Misconfiguration**: Guard against policy violations
5. **Identity-based Access Management**: Integration with OIDC providers for enhanced user identity verification
6. **Extensibility**: Easy to introduce new policies as needs evolve

##### Alternatives Considered

1. **Legacy Admission Controllers**: Rejected due to limited flexibility and poor integration
2. **Custom Webhooks**: Rejected due to complexity in management and maintenance
3. **Third-Party Solutions**: Rejected due to integration difficulties and vendor lock-in

#### Implementation Details

##### OpenShift Built-in Admission

* **Security Context Constraints**: Default and custom SCCs for VM and container workloads
* **RBAC Enforcements**: Actionable role- and label-based access controls
* **Quotas and Limit Ranges**: Ensuring fair resource allocation per team

##### KubeVirt Webhooks

* **Validation Webhooks**: Enforce configuration standards for VM specs
* **Mutation Webhooks**: Apply defaults and constraints to VM definitions

##### Kyverno Policies

* **Configuration Validation**: Ensure compliance with organization best practices
* **Resource Constraints**: Limit what configurations may be used / deployed
* **Dynamic Policies**: Automate policy reapplication based on changes

#### Compliance Considerations

* **Auditability**: Policy applications and violation logging
* **Policy-as-Code**: Centralized version control and history of policy changes
* **Enforcement vs Warning**: Progressive policy application based on audit

#### Consequences

##### Positive

* **Improved Security Posture**: Clusters protected from non-compliant configurations
* **Enhanced Compliance Auditability**: Documentation and reporting of policy compliance
* **Reduced Operational Risk**: Guard against human error and unsafe configurations

##### Negative

* **Complex Rule Management**: Need mature processes to handle policy lifecycle
* **Performance Overhead**: May introduce latency to resource creation/update
* **Learning Curve**: Required training for policy authors

#### Implementation Plan

##### Phase 1: Planning

1. Identify key compliance and security requirements
2. Design initial policy set and test environment
3. Engage stakeholders to define policy boundaries

##### Phase 2: Rollout

1. Deploy core admission controllers with policy-as-code principles, leveraging IAM for authentication and authorization
2. Begin enforcement in non-production environments
3. Gradually extend to production with monitoring and logging

##### Phase 3: Monitoring and Adjustment

1. Enable continuous policy evaluation and audit logging
2. Conduct regular policy reviews and updates
3. Train teams on policy creation and maintenance

#### Compliance and Observability

##### Monitoring

* Policy applicability and compliance dashboards
* Alerts for policy violations and enforcement actions

##### Logging and Reporting

* Centralized logging of admission requests and results
* Automated compliance reports tied to policy adherence

## Adr 005 Network Cni

### ADR-005: Cilium CNI with Multus Multi-Network Strategy

#### Status

Accepted

#### Date

2024-12-01

#### Context

The RH OVE ecosystem requires advanced networking capabilities to support both container and VM workloads with enterprise-grade security, performance, and observability. Traditional iptables-based CNI solutions lack the performance and security features needed for modern hybrid workloads.

#### Decision

We will implement **Cilium as the primary CNI** with **Multus for multi-network support**, providing eBPF-powered networking with advanced security and observability capabilities.

#### Rationale

##### Advantages of Cilium

1. **eBPF Performance**: Superior performance compared to iptables-based solutions
2. **Identity-Aware Security**: Security policies based on workload identity, not IP addresses
3. **L7 Security**: Application-layer security without sidecar proxies
4. **Service Mesh Capabilities**: Built-in service mesh functionality
5. **Red Hat Certification**: Certified CNI plugin for OpenShift
6. **Hubble Observability**: Deep network visibility and monitoring
7. **Transparent Encryption**: Built-in WireGuard and IPsec support

##### Advantages of Multus Integration

1. **Multi-Network Support**: Attach multiple network interfaces to VMs
2. **Legacy Network Integration**: Support for existing VLAN-based networks
3. **Performance Networks**: SR-IOV for high-performance workloads
4. **Network Segmentation**: Separate management, storage, and data networks

##### Alternatives Considered

###### 1. OVN-Kubernetes (OpenShift Default)

* **Pros**: Native OpenShift integration, mature
* **Cons**: Limited eBPF features, performance overhead
* **Rejected**: Cilium provides superior performance and security

###### 2. Calico

* **Pros**: Strong network policies, eBPF support
* **Cons**: No built-in service mesh, complex multi-network setup
* **Rejected**: Cilium offers better integrated solution

###### 3. Flannel

* **Pros**: Simple, lightweight
* **Cons**: Limited security features, no eBPF support
* **Rejected**: Insufficient for enterprise requirements

#### Implementation Details

##### Cilium Configuration

apiVersion: v1  
kind: ConfigMap  
metadata:  
 name: cilium-config  
 namespace: kube-system  
data:  
 # Enable Cilium features  
 enable-ipv4: "true"  
 enable-ipv6: "false"  
   
 # eBPF configuration  
 enable-bpf-masquerade: "true"  
 enable-host-reachable-services: "true"  
   
 # Security features  
 enable-l7-proxy: "true"  
 enable-policy: "default"  
 policy-enforcement-mode: "default"  
   
 # Service mesh capabilities  
 enable-envoy-config: "true"  
   
 # Encryption  
 enable-wireguard: "true"  
 wireguard-userspace-fallback: "true"  
   
 # Observability  
 enable-hubble: "true"  
 hubble-listen-address: ":4244"  
 hubble-metrics-server: ":9091"  
 hubble-metrics: |  
 dns:query;ignoreAAAA  
 drop  
 tcp  
 flow  
 icmp  
 http  
   
 # Performance optimizations  
 enable-bandwidth-manager: "true"  
 enable-local-redirect-policy: "true"  
 kube-proxy-replacement: "strict"

##### Multus Installation

apiVersion: operator.openshift.io/v1  
kind: Network  
metadata:  
 name: cluster  
spec:  
 additionalNetworks:  
 - name: management-network  
 namespace: default  
 type: Raw  
 rawCNIConfig: |  
 {  
 "cniVersion": "0.3.1",  
 "name": "management-network",  
 "type": "macvlan",  
 "master": "ens192",  
 "mode": "bridge",  
 "ipam": {  
 "type": "static"  
 }  
 }

##### Network Attachment Definitions

### Management Network  
apiVersion: k8s.cni.cncf.io/v1  
kind: NetworkAttachmentDefinition  
metadata:  
 name: management-net  
 namespace: vm-infrastructure  
spec:  
 config: |  
 {  
 "cniVersion": "0.3.1",  
 "name": "management-net",  
 "type": "macvlan",  
 "master": "ens192",  
 "mode": "bridge",  
 "ipam": {  
 "type": "static",  
 "addresses": [  
 {  
 "address": "192.168.100.0/24",  
 "gateway": "192.168.100.1"  
 }  
 ]  
 }  
 }  
---  
### High-Performance SR-IOV Network  
apiVersion: k8s.cni.cncf.io/v1  
kind: NetworkAttachmentDefinition  
metadata:  
 name: sriov-net  
 namespace: vm-production  
spec:  
 config: |  
 {  
 "cniVersion": "0.3.1",  
 "name": "sriov-net",  
 "type": "sriov",  
 "deviceID": "1017",  
 "vf": 0,  
 "ipam": {  
 "type": "static"  
 }  
 }

##### Identity-Aware Network Policies

apiVersion: cilium.io/v2  
kind: CiliumNetworkPolicy  
metadata:  
 name: web-to-database-policy  
 namespace: app-web-prod  
spec:  
 endpointSelector:  
 matchLabels:  
 app: web-frontend  
 egress:  
 - toEndpoints:  
 - matchLabels:  
 app: database  
 environment: production  
 toPorts:  
 - ports:  
 - port: "5432"  
 protocol: TCP  
 rules:  
 http:  
 - method: "GET"  
 path: "/health"

##### L7 Security Policies

apiVersion: cilium.io/v2  
kind: CiliumNetworkPolicy  
metadata:  
 name: api-security-policy  
 namespace: app-api-prod  
spec:  
 endpointSelector:  
 matchLabels:  
 app: api-server  
 ingress:  
 - fromEndpoints:  
 - matchLabels:  
 app: web-frontend  
 toPorts:  
 - ports:  
 - port: "8080"  
 protocol: TCP  
 rules:  
 http:  
 - method: "GET"  
 path: "/api/v1/.\*"  
 - method: "POST"  
 path: "/api/v1/users"  
 headers:  
 - "Content-Type: application/json"

##### VM Multi-Network Configuration

apiVersion: kubevirt.io/v1  
kind: VirtualMachine  
metadata:  
 name: multi-network-vm  
 namespace: vm-infrastructure  
 annotations:  
 k8s.v1.cni.cncf.io/networks: |  
 [  
 {  
 "name": "management-net",  
 "ips": ["192.168.100.10/24"]  
 },  
 {  
 "name": "storage-net",  
 "ips": ["10.0.1.10/24"]  
 }  
 ]  
spec:  
 running: true  
 template:  
 spec:  
 domain:  
 devices:  
 interfaces:  
 - name: default  
 masquerade: {}  
 - name: management  
 bridge: {}  
 - name: storage  
 bridge: {}  
 networks:  
 - name: default  
 pod: {}  
 - name: management  
 multus:  
 networkName: management-net  
 - name: storage  
 multus:  
 networkName: storage-net

#### Security Implementation

##### Transparent Encryption

### WireGuard encryption configuration  
apiVersion: v1  
kind: ConfigMap  
metadata:  
 name: cilium-config  
 namespace: kube-system  
data:  
 enable-wireguard: "true"  
 wireguard-userspace-fallback: "true"  
 encryption-node: "true"

##### Zero Trust Network Policies

apiVersion: cilium.io/v2  
kind: CiliumNetworkPolicy  
metadata:  
 name: default-deny-all  
 namespace: app-web-prod  
spec:  
 endpointSelector: {}  
 ingress: []  
 egress:  
 # Allow DNS  
 - toEndpoints:  
 - matchLabels:  
 k8s:io.kubernetes.pod.namespace: kube-system  
 k8s:k8s-app: kube-dns  
 toPorts:  
 - ports:  
 - port: "53"  
 protocol: UDP

#### Observability with Hubble

##### Hubble Relay Configuration

apiVersion: v1  
kind: ConfigMap  
metadata:  
 name: hubble-config  
 namespace: kube-system  
data:  
 config.yaml: |  
 server:  
 address: 0.0.0.0:4245  
 relay:  
 address: hubble-relay.kube-system.svc.cluster.local:80  
 tls:  
 enabled: false

##### Network Flow Monitoring

### Monitor network flows  
hubble observe --namespace app-web-prod  
  
### Check policy violations  
hubble observe --verdict DENIED  
  
### Monitor specific VM traffic  
hubble observe --pod vm-database-xxx

#### Performance Optimization

##### eBPF Host Routing

apiVersion: v1  
kind: ConfigMap  
metadata:  
 name: cilium-config  
 namespace: kube-system  
data:  
 enable-host-routing: "true"  
 enable-external-ips: "true"  
 enable-node-port: "true"  
 enable-host-port: "true"

##### Bandwidth Management

apiVersion: cilium.io/v2  
kind: CiliumBandwidthPolicy  
metadata:  
 name: bandwidth-limit  
 namespace: app-web-prod  
spec:  
 endpointSelector:  
 matchLabels:  
 app: web-frontend  
 egress:  
 - bandwidth: "100M"  
 - bandwidth: "1G"  
 dscp: 46 # High priority traffic

#### Consequences

##### Positive

* **Superior Performance**: eBPF provides 10-100x better performance than iptables
* **Enhanced Security**: Identity-aware policies and L7 security without sidecars
* **Deep Observability**: Hubble provides comprehensive network visibility
* **Future-Proof**: eBPF is the future of Linux networking
* **Multi-Network Support**: Seamless integration with legacy and high-performance networks

##### Negative

* **Learning Curve**: Teams need to learn eBPF concepts and Cilium specifics
* **Debugging Complexity**: eBPF programs can be harder to debug than traditional networking
* **Resource Requirements**: Higher memory usage compared to simpler CNI solutions
* **Compatibility Concerns**: Some legacy applications may need network policy adjustments

#### Migration Strategy

##### Phase 1: Preparation

1. Audit existing network policies and requirements
2. Set up test clusters with Cilium/Multus
3. Train operations team on eBPF and Cilium concepts

##### Phase 2: Non-Production Deployment

1. Deploy Cilium in development and staging clusters
2. Migrate network policies to Cilium format
3. Implement Hubble monitoring and alerting

##### Phase 3: Production Migration

1. Schedule maintenance window for CNI migration
2. Deploy Cilium with careful monitoring
3. Gradually enable advanced features (encryption, L7 policies)

##### Phase 4: Advanced Features

1. Enable service mesh capabilities
2. Implement advanced security policies
3. Optimize performance settings based on workload patterns

#### Monitoring and Alerting

##### Key Metrics

* Network throughput per namespace/pod
* Policy enforcement latency
* eBPF program load and execution time
* Hubble flow processing rate
* Encryption overhead metrics

##### Alerting Rules

groups:  
- name: cilium-alerts  
 rules:  
 - alert: CiliumAgentDown  
 expr: up{job="cilium-agent"} == 0  
 for: 5m  
 labels:  
 severity: critical  
 - alert: NetworkPolicyViolation  
 expr: increase(cilium\_policy\_verdicts\_total{verdict="DENIED"}[5m]) > 10  
 labels:  
 severity: warning

This network architecture provides enterprise-grade performance, security, and observability for the RH OVE ecosystem while supporting both modern cloud-native applications and traditional VM workloads.

## Adr 006 Backup Strategy

### ADR-006: Backup Strategy for RH OVE Ecosystem

#### Status

Accepted

#### Date

2024-12-01

#### Context

Ensuring data protection and recovery is crucial for the RH OVE multi-cluster environment. The solution must support frequent, secure, and efficient backups across clusters, aligning with business continuity and compliance requirements.

#### Decision

Adopt a centralized backup strategy using **Rubrik** for VM and containerized workloads, providing consistency, compliance, and ease of management.

#### Rationale

##### Advantages

1. **Unified Management**: Single pane of glass for managing backups across clusters
2. **Policy-Driven**: Flexibility in configuring backup policies per application/business need
3. **Deduplication and Compression**: Reduce storage costs by minimizing redundant data
4. **Cloud Integration**: Support for hybrid cloud scenarios and long-term data retention
5. **Application Consistency**: Automated application-aware snapshot management

##### Alternatives Considered

1. **Velero**: Open-source alternative
   * **Pros**: Strong integration with Kubernetes ecosystems
   * **Cons**: Complexity in VM integration and limited cloud support
   * **Rejected**: Difficulties in ensuring VM workload consistency and enterprise support
2. **DIY Scripting Solutions**: Custom in-house scripts
   * **Pros**: Potentially customizable
   * **Cons**: Highly error-prone, difficult to manage at scale
   * **Rejected**: Lack of enterprise features, consistency guarantees, and support

#### Implementation Details

##### Backup Policy Design

* **Daily Backups**: RPO of 24 hours for critical workloads
* **Weekly Full Backup with Daily Incrementals**: Optimizes storage usage
* **Data Encryption**: Both in transit and at rest using AES-256

##### Backup Architecture

Backup Architecture  
├── Management Cluster  
│ ├── Rubrik Management Node  
│ └── Centralized Backup Policy Management  
├── Application Clusters  
│ ├── Rubrik Edge Devices  
│ ├── Local Snapshot Agents  
│ └── Data Replica Agents  
└── Cloud Archive  
 ├── Long Term Retention Storage  
 └── Cross-Region DR Copies

##### Configuration Example

apiVersion: backup.rubrik.com/v1alpha1  
kind: BackupPolicy  
metadata:  
 name: application-backup-policy  
 namespace: backup  
spec:  
 frequency: "24h"  
 retention:  
 local: "30d"  
 cloud: "365d"  
 snapshotConsistency: "crash-consistent"  
 includeVolumes: "all"  
 excludeVolumes:  
 - "scratch"  
 encryption: enabled  
 replication:  
 target: cloud-archive  
 frequency: "12h"

#### Security and Compliance Considerations

* **Data Encryption**: All backup data encrypted with AES-256
* **Access Control**: Role-based access for backup management and retrieval
* **Audit Trails**: Detailed logging of backup and restore operations
* **Compliance Alignment**: Meets GDPR, HIPAA, and SOC 2 requirements

#### Consequences

##### Positive

* **Reduced Risk**: Comprehensive DR strategy minimizes impact of data loss events
* **Operational Visibility**: Centralized monitoring and alerting of backup status
* **Strategic Flexibility**: Support for hybrid cloud and multi-region deployments

##### Negative

* **Cost Considerations**: Could incur higher upfront Costa than open-source alternatives
* **Training Requirement**: Backup administrators require training in Rubrik solutions

#### Migration Strategy

##### Phase 1: Planning

1. Define business-critical systems and RPO/RTO requirements
2. Design initial backup policy and architecture
3. Identify data sovereignty and compliance requirements

##### Phase 2: Non-Production Deployment

1. Pilot Rubrik deployment in development environment
2. Test backup and restore operations thoroughly
3. Validate compliance alignment with internal and external audits

##### Phase 3: Production Rollout

1. Deploy Rubrik management in the production environment
2. Migrate to production backup policies with minimal downtime
3. Enable monitoring and alerting on backup status

##### Phase 4: Continuous Improvement

1. Regular policy reviews to adapt to changing business needs
2. Leverage Rubrik analytics for optimizations and reporting
3. Update DR plans based on lessons learned and testing

#### Monitoring and Metrics

##### Key Monitoring Metrics

* Backup success/failure rates
* Storage consumption over time
* Deduplication and compression efficiency
* RPO and RTO performance

##### Alerting Setup

groups:  
- name: backup-alerts  
 rules:  
 - alert: BackupFailure  
 expr: rubrik\_backup\_failed{job="rubrik-agent"}  0  
 for: 10m  
 labels:  
 severity: critical  
 - alert: RPOViolation  
 expr: rubrik\_backup\_rpo{target="24h"} =r 24 \* 60 \* 60  
 labels:  
 severity: warning

This comprehensive backup strategy ensures that RH OVE can achieve high data availability, integrity, and compliance, aligning with enterprise best practices for data protection and disaster recovery.

## Adr 007 Monitoring Strategy

### ADR-007: Monitoring Strategy for RH OVE Ecosystem

#### Status

Accepted

#### Date

2024-12-01

#### Context

For the RH OVE multi-cluster setup, a comprehensive monitoring solution is necessary to ensure operational visibility, performance management, and incident response capability for both containerized and VM-based workloads.

#### Decision

Implement an integrated monitoring solution using **Prometheus and Grafana** for metrics collection and visualization, enhanced by **Dynatrace** for application performance monitoring and **Hubble** for network observability.

#### Rationale

##### Prometheus & Grafana

1. **Scalability**: Native Kubernetes support, able to scale for large environments
2. **Flexibility**: Customizable dashboards and extensibility with plugins
3. **Community Support**: Active ecosystem with numerous exporters and integrations
4. **Real-time Metrics**: Capable of handling thousands of unique time-series metrics
5. **Alerting**: Integrated alert management with Prometheus Alertmanager

##### Dynatrace

1. **Full-Stack Monitoring**: Covers both infrastructure and application layers
2. **AI-Powered Analytics**: Automated anomaly detection and root cause analysis
3. **Cloud-Native Support**: Strong support for Kubernetes and container environments
4. **Unified Observability**: Centralized insights across microservices and legacy apps

##### Hubble

1. **eBPF-powered Network Insights**: Detailed flow visibility and security audits
2. **High Throughput**: Capable of capturing thousands of network flows per second
3. **Deployment Simplicity**: Out-of-the-box integration with Cilium

##### Alternatives Considered

1. **OpenShift Monitoring Stack**
   * **Pros**: Native solution, well-integrated
   * **Cons**: Lacks depth in application performance monitoring
   * **Rejected**: Chosen instead for basic cluster health visibility
2. **Elastic Stack**
   * **Pros**: Full-text search capabilities
   * **Cons**: Complexity and resource consumption
   * **Rejected**: Simplified requirements focused on metrics
3. **DataDog**
   * **Pros**: Comprehensive feature set, SaaS model
   * **Cons**: Cost concerns for large-scale deployment
   * **Rejected**: Cost prohibitive compared to chosen solutions

#### Implementation Details

##### Prometheus Configuration

apiVersion: monitoring.coreos.com/v1  
kind: Prometheus  
metadata:  
 name: global-prometheus  
 namespace: monitoring  
spec:  
 replicas: 3  
 serviceAccountName: prometheus  
 serviceMonitorSelector:  
 matchLabels:  
 team: observability  
 storage:  
 volumeClaimTemplate:  
 spec:  
 accessModes:  
 - ReadWriteOnce  
 resources:  
 requests:  
 storage: 500Gi

##### Grafana Setup

* **Dashboards**: Pre-configured dashboards for cluster health, application performance, VM metrics
* **Themes**: Custom theming for alignment with corporate branding
* **User Access Control**: Integrated with OAuth for SSO

##### Dynatrace Integration

* Deployment of OneAgent across clusters for full-stack visibility
* Integration with CI/CD pipelines for real-time performance feedback
* Automated tagging for dynamic cloud workloads

##### Hubble Configuration

* Enable flow aggregation and analysis for detailed network observability
* Real-time flow filtering and visualization of network policies

#### Security and Compliance Considerations

* **Data Encryption**: All telemetry data encrypted in transit
* **Role-Based Access Control**: Segmented access to monitoring data
* **Compliance Monitoring**: Automated checks for regulatory compliance
* **Audit Logging**: Capture all configuration and access attempts

#### Consequences

##### Positive

* **Operational Efficiency**: Reduce MTTR with real-time insights and alerting
* **Proactive Performance Management**: Identify and resolve issues before impacting users
* **Unified Observability**: Single-pane monitoring across clusters and applications

##### Negative

* **Complexity of Integration**: Requires coordination across multiple tools
* **Resource Overhead**: Higher costs in terms of storage and compute resources
* **Training Requirements**: Teams need to become familiar with monitoring tools

#### Migration Strategy

##### Phase 1: Initial Setup and Configuration

1. Deploy base Prometheus and Grafana setup in the management cluster
2. Establish Dynatrace integration for application monitoring
3. Enable Hubble for network flow visibility

##### Phase 2: Metrics and Dashboard Customization

1. Design and implement custom dashboards for key performance indicators
2. Configure alerting thresholds and incident response playbooks
3. Integrate monitoring data with existing ITSM tools

##### Phase 3: Continuous Optimization

1. Conduct regular review of metrics and dashboards for continuous improvement
2. Leverage Dynatrace AI insights for proactive tuning and capacity planning
3. Regularly assess network flow policies for efficiency and security

#### Monitoring and Metrics

##### Key Performance Indicators

* CPU, memory, and storage utilization
* Network latency and throughput
* Application response times and error rates
* VM and container health

##### Alerting Rules

* Resource exhaustion (CPU, Memory, Storage)
* Network policy violations
* Anomalous application behavior

This robust monitoring strategy ensures RH OVE achieves operational excellence, rapid issue resolution, and strategic insight into both infrastructure performance and applications across the multi-cluster environment.

## Adr 008 Iam Strategy

### ADR-008: Identity and Access Management (IAM) Strategy

#### Status

Accepted

#### Date

2024-12-01

#### Context

The RH OVE multi-cluster ecosystem requires enterprise-grade identity and access management to ensure secure authentication and authorization across all clusters and services. Traditional cluster-specific authentication creates operational complexity and security risks in multi-cluster environments.

#### Decision

We will implement a comprehensive IAM strategy using OpenID Connect (OIDC) providers with Keycloak as the primary identity provider, integrated with Kubernetes-native RBAC and service mesh authentication.

#### Rationale

##### Advantages

1. **Centralized Identity Management**: Single source of truth for user identities across all clusters
2. **Single Sign-On (SSO)**: Seamless authentication across all services and clusters
3. **Enterprise Integration**: Native integration with existing LDAP/Active Directory infrastructure
4. **Multi-Factor Authentication**: Enhanced security with mandatory MFA for administrative accounts
5. **Audit and Compliance**: Complete audit trail for SOC 2, GDPR, and HIPAA compliance
6. **Zero Trust Security**: Identity-aware access controls with least-privilege principles
7. **Scalability**: Supports horizontal scaling across multiple clusters and regions

##### Alternatives Considered

###### 1. Basic Kubernetes ServiceAccount Authentication

* **Pros**: Simple, native Kubernetes integration
* **Cons**: No centralized management, limited audit capabilities, poor user experience
* **Rejected**: Insufficient for enterprise requirements

###### 2. LDAP Direct Integration

* **Pros**: Direct integration with existing directory services
* **Cons**: No OIDC compliance, limited multi-cluster support, poor web service integration
* **Rejected**: Limited modern authentication capabilities

###### 3. Commercial Solutions (Auth0, Okta)

* **Pros**: Feature-rich, managed service
* **Cons**: Vendor lock-in, higher costs, limited customization
* **Alternative**: Considered for specific use cases but Keycloak preferred for primary solution

#### Implementation Details

##### Core Components

###### Identity Provider: Keycloak (Red Hat SSO)

### Keycloak Deployment Configuration  
apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: keycloak  
 namespace: auth-system  
spec:  
 replicas: 3  
 selector:  
 matchLabels:  
 app: keycloak  
 template:  
 spec:  
 containers:  
 - name: keycloak  
 image: quay.io/keycloak/keycloak:20.0  
 env:  
 - name: KC\_DB  
 value: postgres  
 - name: KC\_DB\_URL  
 value: jdbc:postgresql://postgres:5432/keycloak  
 - name: KC\_HOSTNAME\_STRICT  
 value: "false"  
 - name: KC\_HTTP\_ENABLED  
 value: "true"  
 - name: KC\_PROXY  
 value: edge

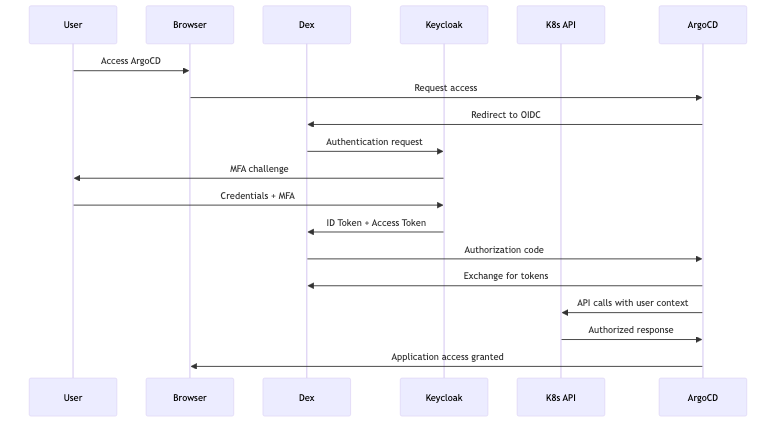
###### OIDC Integration with OpenShift

apiVersion: config.openshift.io/v1  
kind: OAuth  
metadata:  
 name: cluster  
spec:  
 identityProviders:  
 - name: keycloak-oidc  
 mappingMethod: claim  
 type: OpenID  
 openID:  
 clientID: openshift-cluster  
 clientSecret:  
 name: oidc-client-secret  
 issuer: https://keycloak.company.com/auth/realms/openshift  
 claims:  
 preferredUsername: [preferred\_username]  
 name: [name]  
 email: [email]  
 groups: [groups]

###### Dex OIDC Proxy for Service Authentication

apiVersion: v1  
kind: ConfigMap  
metadata:  
 name: dex-config  
 namespace: auth-system  
data:  
 config.yaml: |  
 issuer: https://dex.company.com  
 storage:  
 type: kubernetes  
 config:  
 inCluster: true  
 connectors:  
 - type: oidc  
 id: keycloak  
 name: Keycloak  
 config:  
 issuer: https://keycloak.company.com/auth/realms/company  
 clientID: dex-client  
 clientSecret: $DEX\_CLIENT\_SECRET  
 scopes: [openid, profile, email, groups]

##### Authentication Flow



##### Authorization Model

###### Group-Based RBAC

### Keycloak Groups mapped to Kubernetes RBAC  
groups:  
 platform-admins:  
 kubernetes-role: cluster-admin  
 argocd-role: admin  
 web-developers:  
 kubernetes-role: developer  
 kubernetes-namespaces: [app-web-prod, app-web-staging, app-web-dev]  
 argocd-role: web-app-admin  
 database-admins:  
 kubernetes-role: developer  
 kubernetes-namespaces: [app-database-prod, app-database-staging]  
 argocd-role: database-admin  
 security-auditors:  
 kubernetes-role: view  
 argocd-role: readonly

###### Kubernetes RBAC Integration

apiVersion: rbac.authorization.k8s.io/v1  
kind: ClusterRoleBinding  
metadata:  
 name: platform-admin-binding  
subjects:  
- kind: Group  
 name: "platform-admins"  
 apiGroup: rbac.authorization.k8s.io  
roleRef:  
 kind: ClusterRole  
 name: cluster-admin  
 apiGroup: rbac.authorization.k8s.io

##### Security Controls

###### Token Management

* **Access Tokens**: 15-minute expiration
* **Refresh Tokens**: 1-hour expiration
* **ID Tokens**: 5-minute expiration
* **Service Account Tokens**: 2-hour expiration with projected volumes

###### Multi-Factor Authentication

### Mandatory MFA Flow in Keycloak  
authenticationFlows:  
 - alias: "browser-with-mfa"  
 description: "Browser flow with mandatory MFA"  
 authenticationExecutions:  
 - authenticator: "auth-username-password-form"  
 requirement: "REQUIRED"  
 - authenticator: "auth-otp-form"  
 requirement: "REQUIRED"

###### Network Security

* Network policies restricting authentication service communication
* TLS 1.3 encryption for all authentication traffic
* Certificate-based mutual TLS for service-to-service communication

#### User Lifecycle Management

##### Automated Provisioning

* SCIM integration for user provisioning/deprovisioning
* Automated group assignment based on organizational roles
* ServiceAccount creation for automated systems

##### Self-Service Capabilities

* Password reset and account recovery
* MFA device management
* Access request workflows

#### Monitoring and Audit

##### Metrics and Monitoring

### Prometheus ServiceMonitor for Authentication Metrics  
apiVersion: monitoring.coreos.com/v1  
kind: ServiceMonitor  
metadata:  
 name: dex-metrics  
 namespace: auth-system  
spec:  
 selector:  
 matchLabels:  
 app: dex  
 endpoints:  
 - port: metrics  
 interval: 30s

##### Audit Logging

* Complete authentication audit trail
* RBAC change tracking
* Failed authentication monitoring
* Compliance reporting automation

##### Key Metrics

* Authentication success/failure rates
* Active user sessions
* Token expiration and renewal rates
* MFA adoption rates
* Policy violation incidents

#### Disaster Recovery

##### High Availability

* Multi-replica Keycloak deployment with database clustering
* Cross-region identity provider replication
* Automated failover with health checks

##### Backup Strategy

### Daily Keycloak Database Backup  
apiVersion: batch/v1  
kind: CronJob  
metadata:  
 name: keycloak-backup  
 namespace: auth-system  
spec:  
 schedule: "0 2 \* \* \*"  
 jobTemplate:  
 spec:  
 template:  
 spec:  
 containers:  
 - name: backup  
 image: postgres:13  
 command: ["/bin/bash", "-c"]  
 args:  
 - pg\_dump -h keycloak-db -U keycloak keycloak > /backup/keycloak-$(date +%Y%m%d).sql && aws s3 cp /backup/keycloak-$(date +%Y%m%d).sql s3://iam-backups/

#### Consequences

##### Positive

* **Enhanced Security**: Zero trust identity-aware access controls
* **Operational Efficiency**: Centralized identity management reduces operational overhead
* **Compliance Ready**: Built-in audit trails and regulatory framework alignment
* **Developer Experience**: Single sign-on with self-service capabilities
* **Scalability**: Supports multi-cluster and multi-region deployments
* **Cost Optimization**: Reduced manual identity management tasks

##### Negative

* **Initial Complexity**: Setup and configuration require specialized knowledge
* **Dependencies**: Additional infrastructure components to manage and monitor
* **Learning Curve**: Teams need training on OIDC and Keycloak administration
* **Single Point of Failure**: Identity provider availability critical for system access
* **Token Management**: Requires careful handling of token lifecycle and security

#### Migration Strategy

##### Phase 1: Infrastructure Setup (2-3 weeks)

1. Deploy Keycloak in management cluster with HA configuration
2. Configure LDAP/AD integration and user federation
3. Set up Dex OIDC proxy in all clusters
4. Implement network policies and security controls

##### Phase 2: Service Integration (3-4 weeks)

1. Configure OpenShift OAuth with OIDC
2. Integrate ArgoCD with OIDC authentication
3. Configure Grafana and Prometheus with OIDC
4. Set up RBAC mappings and group assignments

##### Phase 3: User Migration (2-3 weeks)

1. Migrate existing users to new identity system
2. Configure MFA for all administrative accounts
3. Test authentication flows and access controls
4. Train users on new authentication experience

##### Phase 4: Monitoring and Optimization (1-2 weeks)

1. Deploy monitoring and alerting for authentication services
2. Configure audit logging and compliance reporting
3. Optimize token lifecycle and security policies
4. Document operational procedures and runbooks

#### Compliance Considerations

##### Regulatory Alignment

* **SOC 2 Type II**: Automated audit trails and access controls
* **GDPR**: Right to be forgotten and data minimization
* **HIPAA**: PHI access controls and audit logging
* **SOX**: Financial system access controls and separation of duties

##### Audit Requirements

* All authentication events logged with timestamps
* RBAC changes tracked with user attribution
* Failed authentication attempts monitored and alerted
* Regular access reviews automated with reporting

This IAM strategy provides enterprise-grade identity and access management for the RH OVE multi-cluster ecosystem, ensuring security, compliance, and operational efficiency through modern OIDC-based authentication and Kubernetes-native authorization.

## Requirements Table

### Requirements Summary Table

This document provides a comprehensive overview of all functional and non-functional requirements for the RH OVE multi-cluster ecosystem.

#### Functional Requirements Summary

| Requirement ID | Category | Title | ADR Reference | Acceptance Criteria Summary |
| --- | --- | --- | --- | --- |
| [FR-001.1](fr.md#fr-0011-cluster-topology-management) | Multi-Cluster Management | Cluster Topology Management | ADR-003 | Namespace patterns {app-name}-{environment}, cross-namespace policies, resource quotas |
| [FR-001.2](fr.md#fr-0012-multi-cluster-governance) | Multi-Cluster Management | Multi-Cluster Governance | ADR-001 | RHACM cluster lifecycle, centralized policies, monitoring aggregation |
| [FR-002.1](fr.md#fr-0021-argocd-hub-architecture) | GitOps Integration | ArgoCD Hub Architecture | ADR-002 | HA ArgoCD Hub, Application of Applications, automated/manual sync |
| [FR-002.2](fr.md#fr-0022-configuration-management) | GitOps Integration | Configuration Management | ADR-002 | Git-based config, Kustomize overlays, drift detection |
| [FR-003.1](fr.md#fr-0031-layered-admission-control) | Policy Management | Layered Admission Control | ADR-004 | Kyverno, Security Context Constraints, policy-as-code |
| [FR-003.2](fr.md#fr-0032-security-policy-enforcement) | Policy Management | Security Policy Enforcement | ADR-004 | Blocking/warning modes, violation reporting, exemption workflows |
| [FR-004.1](fr.md#fr-0041-cilium-cni-implementation) | Networking | Cilium CNI Implementation | ADR-005 | Cilium with Hubble, identity-aware policies, WireGuard encryption |
| [FR-004.2](fr.md#fr-0042-multi-network-support) | Networking | Multi-Network Support | ADR-005 | Multus NADs, multiple VM interfaces, VLAN segmentation, SR-IOV |
| [FR-004.3](fr.md#fr-0043-zero-trust-security) | Networking | Zero Trust Security | ADR-005 | Default-deny policies, L7 enforcement, flow monitoring |
| [FR-005.1](fr.md#fr-0051-centralized-backup-management) | Backup & DR | Centralized Backup Management | ADR-006 | Rubrik management, policy-driven scheduling, VM snapshots |
| [FR-005.2](fr.md#fr-0052-disaster-recovery-capabilities) | Backup & DR | Disaster Recovery Capabilities | ADR-006 | Cloud replication, point-in-time recovery, cross-cluster failover |
| [FR-006.1](fr.md#fr-0061-integrated-monitoring-stack) | Monitoring | Integrated Monitoring Stack | ADR-007 | Federated Prometheus, Grafana, Dynatrace, Hubble integration |
| [FR-006.2](fr.md#fr-0062-proactive-monitoring-and-alerting) | Monitoring | Proactive Monitoring and Alerting | ADR-007 | Threshold/anomaly detection, ITSM integration, runbook automation |
| [FR-007.1](fr.md#fr-0071-vm-lifecycle-management) | VM Management | VM Lifecycle Management | KubeVirt | VM CRUD operations, live migration, template management |
| [FR-007.2](fr.md#fr-0072-storage-management) | VM Management | Storage Management | KubeVirt | Multiple storage classes, volume expansion, image import |
| [FR-008.1](fr.md#fr-0081-identity-and-access-management) | Security | Identity and Access Management | ADR-008 | Keycloak OIDC, LDAP/AD federation, SSO, Dex proxy, OAuth integration, RBAC |
| [FR-008.2](fr.md#fr-0082-multi-factor-authentication-and-security-controls) | Security | Multi-Factor Authentication and Security Controls | ADR-008 | Mandatory MFA, token security, TLS 1.3, mutual TLS, SCIM automation |
| [FR-008.3](fr.md#fr-0083-compliance-management) | Security | Compliance Management | ADR-008 | SOC 2/GDPR/HIPAA with IAM, automated access reviews, incident tracking |
| [FR-009.1](fr.md#fr-0091-self-service-capabilities) | Developer Experience | Self-Service Capabilities | DevEx | Web interfaces, CLI tools, template provisioning |
| [FR-009.2](fr.md#fr-0092-cicd-integration) | Developer Experience | CI/CD Integration | DevOps | Git webhooks, image building, deployment strategies |

#### Non-Functional Requirements Summary

##### Performance Requirements

| Requirement ID | Category | Metric | Target Value | Rationale |
| --- | --- | --- | --- | --- |
| FR-NFR-001 | Latency | API Response Time | < 200ms (95th percentile) | User experience optimization |
| FR-NFR-002 | Latency | Cross-Cluster Communication | < 50ms | Real-time data synchronization |
| FR-NFR-003 | Latency | VM Startup Time | < 60 seconds | Rapid workload provisioning |
| FR-NFR-004 | Throughput | VM Deployment Rate | 100+ VMs/hour/cluster | Support for scaling events |
| FR-NFR-005 | Throughput | Monitoring Metrics | 10,000+ metrics/second | Comprehensive observability |
| FR-NFR-006 | Throughput | Log Processing | 1GB+ logs/hour | Adequate logging capacity |
| FR-NFR-007 | Resource Utilization | CPU Usage | ≤ 80% under normal load | Performance headroom |
| FR-NFR-008 | Resource Utilization | Memory Usage | ≤ 85% under normal load | Stability margin |
| FR-NFR-009 | Resource Utilization | Storage Usage | ≤ 90% capacity | Growth accommodation |

##### Availability Requirements

| Requirement ID | Category | Metric | Target Value | Business Impact |
| --- | --- | --- | --- | --- |
| FR-NFR-010 | Uptime | Production Clusters | 99.9% (< 8.77 hours/year downtime) | Business continuity |
| FR-NFR-011 | Uptime | Management Cluster | 99.95% (< 4.38 hours/year downtime) | Central control availability |
| FR-NFR-012 | Failover | Critical Services | < 30 seconds failover time | Minimal service disruption |
| FR-NFR-013 | Disaster Recovery | RPO (Recovery Point Objective) | < 4 hours | Data loss tolerance |
| FR-NFR-014 | Disaster Recovery | RTO (Recovery Time Objective) | < 8 hours | Recovery time tolerance |
| FR-NFR-015 | Disaster Recovery | Cross-Region Failover | < 15 minutes | Geographic resilience |

##### Scalability Requirements

| Requirement ID | Category | Metric | Target Value | Scalability Dimension |
| --- | --- | --- | --- | --- |
| FR-NFR-016 | Horizontal Scaling | Application Clusters | Minimum 10 clusters | Multi-environment support |
| FR-NFR-017 | Horizontal Scaling | Pods per Cluster | 1000+ pods | Workload density |
| FR-NFR-018 | Horizontal Scaling | Total Containers | 50,000+ across all clusters | Enterprise scale |
| FR-NFR-019 | Vertical Scaling | VM Resources | Up to 64 vCPUs, 512GB RAM | Large workload support |
| FR-NFR-020 | Vertical Scaling | Storage Volumes | Up to 100TB per volume | Big data capabilities |
| FR-NFR-021 | Vertical Scaling | Network Bandwidth | Up to 25Gbps per node | High-performance networking |

##### Security Requirements

| Requirement ID | Category | Requirement | Implementation | Compliance Impact |
| --- | --- | --- | --- | --- |
| FR-NFR-022 | Authentication | Multi-Factor Authentication | All API access | Enhanced security posture |
| FR-NFR-023 | Authorization | RBAC Enforcement | All cluster components | Access control consistency |
| FR-NFR-024 | Authentication | Service Account Tokens | Time-limited tokens | Reduced credential exposure |
| FR-NFR-025 | Data Protection | Encryption in Transit | TLS 1.3+ | Data confidentiality |
| FR-NFR-026 | Data Protection | Encryption at Rest | AES-256 | Data protection compliance |
| FR-NFR-027 | Key Management | Key Rotation | Every 90 days | Security best practices |
| FR-NFR-028 | Network Security | Inter-Cluster Encryption | All communications | End-to-end security |
| FR-NFR-029 | Network Security | Network Policies | Deny by default | Zero trust implementation |
| FR-NFR-030 | Network Security | Environment Isolation | Network segmentation | Security boundaries |
| FR-NFR-031 | Compliance | SOC 2 Compliance | Type II certification | Regulatory adherence |
| FR-NFR-032 | Audit | Log Retention | 7 years | Compliance requirements |
| FR-NFR-033 | Security | Vulnerability Scanning | Daily scans | Proactive security |

##### Reliability Requirements

| Requirement ID | Category | Requirement | Target | Reliability Impact |
| --- | --- | --- | --- | --- |
| FR-NFR-034 | Fault Tolerance | Single Node Failure | No service interruption | High availability |
| FR-NFR-035 | Fault Tolerance | Single AZ Failure | Continued operation | Geographic resilience |
| FR-NFR-036 | Data Replication | Minimum Replicas | 3 nodes | Data durability |
| FR-NFR-037 | Error Handling | Error Logging | All errors with severity | Operational visibility |
| FR-NFR-038 | Error Handling | Retry Logic | Exponential backoff | Resilient operations |
| FR-NFR-039 | Error Handling | Critical Error Alerting | Automated notifications | Rapid incident response |
| FR-NFR-040 | Monitoring | Health Coverage | 99% system coverage | Comprehensive monitoring |
| FR-NFR-041 | Monitoring | Metrics Retention | Minimum 1 year | Historical analysis |
| FR-NFR-042 | Observability | Distributed Tracing | All services | End-to-end visibility |

##### Maintainability Requirements

| Requirement ID | Category | Requirement | Implementation | Operational Impact |
| --- | --- | --- | --- | --- |
| FR-NFR-043 | Deployment | Zero-Downtime Updates | Rolling deployments | Service continuity |
| FR-NFR-044 | Deployment | Rollback Capability | < 5 minutes | Rapid recovery |
| FR-NFR-045 | Testing | Automated Test Coverage | 90%+ functionality | Quality assurance |
| FR-NFR-046 | Configuration | Version Control | All configuration | Change management |
| FR-NFR-047 | Configuration | Change Auditability | All changes tracked | Compliance and debugging |
| FR-NFR-048 | Infrastructure | Infrastructure as Code | All deployments | Consistency and repeatability |
| FR-NFR-049 | Documentation | API Specifications | OpenAPI for all APIs | Developer experience |
| FR-NFR-050 | Documentation | Operational Runbooks | All procedures | Operational consistency |
| FR-NFR-051 | Documentation | Architecture Decisions | ADR documentation | Knowledge management |

##### Usability Requirements

| Requirement ID | Category | Requirement | Standard/Implementation | User Impact |
| --- | --- | --- | --- | --- |
| FR-NFR-052 | User Interface | Responsive Design | Mobile-friendly web UI | Multi-device access |
| FR-NFR-053 | API Design | RESTful Principles | Standard REST API | Developer experience |
| FR-NFR-054 | CLI Tools | Help Documentation | Comprehensive help | User self-service |
| FR-NFR-055 | Accessibility | WCAG Compliance | 2.1 AA standards | Inclusive design |
| FR-NFR-056 | Localization | Multi-Language Support | International users | Global accessibility |
| FR-NFR-057 | Accessibility | High Contrast Mode | Visual accessibility | Enhanced usability |

##### Capacity Requirements

| Requirement ID | Category | Resource | Minimum Capacity | Scaling Consideration |
| --- | --- | --- | --- | --- |
| FR-NFR-058 | Storage | Usable Storage | 100TB per production cluster | Data growth accommodation |
| FR-NFR-059 | Storage | IOPS Performance | 10,000+ IOPS per cluster | High-performance workloads |
| FR-NFR-060 | Storage | Backup Retention | 5 years | Compliance and recovery |
| FR-NFR-061 | Network | Inter-Cluster Connectivity | 10Gbps minimum | High-bandwidth applications |
| FR-NFR-062 | Network | Intra-Cluster Latency | ≤ 5ms maximum | Real-time applications |
| FR-NFR-063 | Network | MTU Support | 9000 bytes (jumbo frames) | Network optimization |
| FR-NFR-064 | Compute | CPU Capacity | 1000 vCPUs per production cluster | Computational workloads |
| FR-NFR-065 | Compute | Memory Capacity | 4TB RAM per production cluster | Memory-intensive applications |
| FR-NFR-066 | Compute | GPU Support | 8+ GPUs per cluster | AI/ML workloads |

##### Compliance & Regulatory Requirements

| Requirement ID | Category | Requirement | Implementation | Regulatory Framework |
| --- | --- | --- | --- | --- |
| FR-NFR-067 | Data Governance | Data Classification | Automated labeling | Privacy regulations |
| FR-NFR-068 | Data Protection | PII Handling | Encryption and access control | GDPR, HIPAA |
| FR-NFR-069 | Data Governance | Retention Policies | Automated enforcement | Legal requirements |
| FR-NFR-070 | Audit | Administrative Logging | All admin actions | SOX, SOC 2 |
| FR-NFR-071 | Audit | Log Integrity | Tamper-proof, timestamped | Legal evidence |
| FR-NFR-072 | Compliance | Automated Reporting | Compliance dashboards | Regulatory reporting |
| FR-NFR-073 | Privacy | Right to be Forgotten | Data deletion capability | GDPR Article 17 |
| FR-NFR-074 | Privacy | Data Minimization | Principle enforcement | Privacy by design |
| FR-NFR-075 | Privacy | Privacy by Design | Built-in privacy controls | Proactive compliance |

#### Requirements Traceability Matrix

##### ADR to Requirements Mapping

| ADR | Related Functional Requirements | Related Non-Functional Requirements |
| --- | --- | --- |
| ADR-001: Multi-Cluster Pattern | FR-001.1, FR-001.2 | FR-NFR-010, FR-NFR-011, FR-NFR-016, FR-NFR-034, FR-NFR-035 |
| ADR-002: GitOps ArgoCD | FR-002.1, FR-002.2 | FR-NFR-043, FR-NFR-044, FR-NFR-046, FR-NFR-047, FR-NFR-048 |
| ADR-003: Namespace Topology | FR-001.1, FR-008.1 | FR-NFR-023, FR-NFR-030, FR-NFR-067, FR-NFR-068 |
| ADR-004: Admission Controller | FR-003.1, FR-003.2 | FR-NFR-022, FR-NFR-023, FR-NFR-031, FR-NFR-037, FR-NFR-070 |
| ADR-005: Cilium CNI | FR-004.1, FR-004.2, FR-004.3 | FR-NFR-002, FR-NFR-021, FR-NFR-028, FR-NFR-029, FR-NFR-061, FR-NFR-062 |
| ADR-006: Backup Strategy | FR-005.1, FR-005.2 | FR-NFR-013, FR-NFR-014, FR-NFR-015, FR-NFR-032, FR-NFR-060 |
| ADR-007: Monitoring Strategy | FR-006.1, FR-006.2 | FR-NFR-005, FR-NFR-040, FR-NFR-041, FR-NFR-042, FR-NFR-072 |
| ADR-008: IAM Strategy | FR-008.1, FR-008.2, FR-008.3 | FR-NFR-022, FR-NFR-023, FR-NFR-024, FR-NFR-025, FR-NFR-031, FR-NFR-070, FR-NFR-073 |

##### Requirements Coverage Analysis

| Category | Total Requirements | Critical Requirements | Implementation Status |
| --- | --- | --- | --- |
| Functional Requirements | 19 sub-requirements | 12 critical | Architecture defined |
| Performance Requirements | 9 requirements | 6 critical | Targets established |
| Availability Requirements | 6 requirements | 6 critical | SLAs defined |
| Scalability Requirements | 6 requirements | 4 critical | Capacity planned |
| Security Requirements | 12 requirements | 12 critical | Controls specified |
| Reliability Requirements | 9 requirements | 6 critical | Patterns established |
| Maintainability Requirements | 9 requirements | 5 critical | Processes defined |
| Usability Requirements | 6 requirements | 2 critical | Standards adopted |
| Capacity Requirements | 9 requirements | 6 critical | Minimums established |
| Compliance Requirements | 9 requirements | 9 critical | Frameworks aligned |

#### Requirements Validation Criteria

##### Functional Requirements Validation

* **FR Validation Method**: Acceptance criteria testing, integration testing, user acceptance testing
* **FR Success Metrics**: All acceptance criteria met, integration tests pass, user stories completed
* **FR Review Frequency**: Sprint reviews, milestone assessments, release validation

##### Non-Functional Requirements Validation

* **NFR Validation Method**: Performance testing, load testing, security testing, compliance auditing
* **NFR Success Metrics**: All targets met or exceeded, benchmarks achieved, compliance verified
* **NFR Review Frequency**: Continuous monitoring, quarterly assessments, annual compliance reviews

This comprehensive requirements table provides complete traceability from high-level business needs through detailed technical requirements, enabling effective validation and implementation tracking for the RH OVE multi-cluster ecosystem.

## Fr

### Functional Requirements

#### Overview

This document outlines the functional requirements for the RH OVE multi-cluster ecosystem, derived from the architectural decisions documented in our ADRs.

#### FR-001: Multi-Cluster Management

##### FR-001.1: Cluster Topology Management

* **Requirement**: The system must implement application namespace-based topology for workload organization
* **Rationale**: Based on ADR-003, ensures strong isolation, simplified RBAC, and clear resource attribution
* **Acceptance Criteria**:
  + Support namespace patterns: {app-name}-{environment}
  + Implement cross-namespace communication policies
  + Enable namespace-level resource quotas and limits

##### FR-001.2: Multi-Cluster Governance

* **Requirement**: The system must support centralized governance across multiple application clusters from a management cluster
* **Rationale**: Based on ADR-001, provides separation of concerns and operational efficiency
* **Acceptance Criteria**:
  + Deploy and manage policies from management cluster to application clusters
  + Support cluster lifecycle management through RHACM
  + Enable centralized monitoring and logging aggregation

#### FR-002: GitOps Integration

##### FR-002.1: ArgoCD Hub Architecture

* **Requirement**: The system must implement GitOps using ArgoCD in a hub-spoke pattern
* **Rationale**: Based on ADR-002, provides declarative configuration and audit trails
* **Acceptance Criteria**:
  + Deploy ArgoCD Hub in management cluster with high availability
  + Support Application of Applications pattern
  + Enable automated sync for non-production, manual sync for production
  + Integrate with Git repositories for all infrastructure and application configurations

##### FR-002.2: Configuration Management

* **Requirement**: All infrastructure and application configurations must be stored in Git repositories
* **Rationale**: Ensures version control, auditability, and rollback capabilities
* **Acceptance Criteria**:
  + Support environment-specific overlays using Kustomize
  + Enable automated deployment pipeline with proper approval workflows
  + Provide drift detection and automatic remediation

#### FR-003: Admission Control and Policy Management

##### FR-003.1: Layered Admission Control

* **Requirement**: The system must implement layered admission control using OpenShift built-in controllers, KubeVirt webhooks, and Kyverno policies
* **Rationale**: Based on ADR-004, provides flexible, secure, and policy-driven resource validation
* **Acceptance Criteria**:
  + Deploy Kyverno for custom policy management
  + Implement Security Context Constraints for VM and container workloads
  + Support policy-as-code with version control and automated deployment

##### FR-003.2: Security Policy Enforcement

* **Requirement**: The system must validate and enforce security policies before resource persistence
* **Rationale**: Prevents misconfiguration and ensures compliance
* **Acceptance Criteria**:
  + Support both blocking and warning policy modes
  + Provide detailed policy violation reporting
  + Enable policy exemptions with proper approval workflows

#### FR-004: Networking and Connectivity

##### FR-004.1: Cilium CNI Implementation

* **Requirement**: The system must use Cilium as the primary CNI with eBPF-powered networking
* **Rationale**: Based on ADR-005, provides superior performance, identity-aware security, and L7 capabilities
* **Acceptance Criteria**:
  + Deploy Cilium with Hubble for network observability
  + Implement identity-aware network policies
  + Support transparent encryption using WireGuard
  + Enable service mesh capabilities without sidecar proxies

##### FR-004.2: Multi-Network Support

* **Requirement**: The system must support Multus for multi-network configurations
* **Rationale**: Enables legacy network integration, SR-IOV, and network segmentation
* **Acceptance Criteria**:
  + Support NetworkAttachmentDefinitions for management, storage, and data networks
  + Enable multiple network interfaces for VMs
  + Support VLAN-based network segmentation
  + Provide high-performance networking with SR-IOV

##### FR-004.3: Zero Trust Security

* **Requirement**: The system must implement zero trust network principles
* **Rationale**: Ensures security by default with explicit allow policies
* **Acceptance Criteria**:
  + Implement default-deny network policies
  + Support L7 HTTP/HTTPS policy enforcement
  + Enable network flow monitoring and policy violation alerting

#### FR-005: Backup and Disaster Recovery

##### FR-005.1: Centralized Backup Management

* **Requirement**: The system must implement centralized backup using Rubrik for unified VM and container workload protection
* **Rationale**: Based on ADR-006, provides consistency, compliance, and operational efficiency
* **Acceptance Criteria**:
  + Deploy Rubrik management in management cluster
  + Support policy-driven backup scheduling and retention
  + Enable application-consistent snapshots for VMs
  + Provide deduplication and compression for storage optimization

##### FR-005.2: Disaster Recovery Capabilities

* **Requirement**: The system must support cross-region disaster recovery and failover
* **Rationale**: Ensures business continuity and meets compliance requirements
* **Acceptance Criteria**:
  + Support automated backup replication to cloud storage
  + Enable point-in-time recovery for critical workloads
  + Provide recovery testing and validation capabilities
  + Support cross-cluster failover scenarios

#### FR-006: Monitoring and Observability

##### FR-006.1: Integrated Monitoring Stack

* **Requirement**: The system must implement integrated monitoring using Prometheus, Grafana, Dynatrace, and Hubble
* **Rationale**: Based on ADR-007, provides comprehensive observability across infrastructure and applications
* **Acceptance Criteria**:
  + Deploy federated Prometheus for metrics collection across clusters
  + Implement custom Grafana dashboards for infrastructure and application metrics
  + Integrate Dynatrace for full-stack application performance monitoring
  + Enable Hubble for network flow visibility and security monitoring

##### FR-006.2: Proactive Monitoring and Alerting

* **Requirement**: The system must provide proactive monitoring with intelligent alerting
* **Rationale**: Enables early detection and resolution of issues
* **Acceptance Criteria**:
  + Support threshold-based and anomaly detection alerting
  + Integrate with incident management systems
  + Provide runbook automation for common issues
  + Enable custom alerting rules per application namespace

#### FR-007: Virtual Machine Management

##### FR-007.1: VM Lifecycle Management

* **Requirement**: The system must support complete VM lifecycle management using KubeVirt
* **Rationale**: Provides unified management of VMs and containers
* **Acceptance Criteria**:
  + Support VM creation, scaling, and termination
  + Enable VM live migration for maintenance
  + Provide VM template management and cloning
  + Support both Windows and Linux guest operating systems

##### FR-007.2: Storage Management

* **Requirement**: The system must provide flexible storage management for VMs using DataVolumes and CDI
* **Rationale**: Enables efficient storage provisioning and management
* **Acceptance Criteria**:
  + Support multiple storage classes for different performance tiers
  + Enable volume expansion and snapshotting
  + Provide image import from registries, HTTP, and S3 sources
  + Support persistent volume cloning for template workflows

#### FR-008: Security and Compliance

##### FR-008.1: Identity and Access Management

* **Requirement**: The system must integrate with enterprise identity providers using OIDC for centralized authentication and authorization
* **Rationale**: Based on ADR-008, ensures consistent security policies, audit trails, and enterprise integration
* **Acceptance Criteria**:
  + Deploy Keycloak (Red Hat SSO) as primary OIDC provider with HA configuration
  + Support LDAP/Active Directory federation for existing user directories
  + Implement Single Sign-On (SSO) across all clusters and services
  + Deploy Dex OIDC proxy for service authentication
  + Integrate OpenShift OAuth with OIDC claims mapping
  + Implement namespace-based RBAC with OIDC group delegation
  + Enable service account automation with time-limited tokens (15min-2hours)
  + Provide comprehensive audit logging for all authentication and authorization events

##### FR-008.2: Multi-Factor Authentication and Security Controls

* **Requirement**: The system must enforce multi-factor authentication and implement comprehensive security controls
* **Rationale**: Based on ADR-008, ensures zero trust security principles and enhanced threat protection
* **Acceptance Criteria**:
  + Enforce mandatory MFA for all administrative accounts using TOTP/SMS
  + Implement token security with proper expiration policies (access: 15min, refresh: 1hour, ID: 5min)
  + Deploy network policies restricting authentication service communication
  + Support TLS 1.3 encryption for all authentication traffic
  + Enable certificate-based mutual TLS for service-to-service communication
  + Implement automated user provisioning/deprovisioning via SCIM integration
  + Support self-service password reset and MFA device management

##### FR-008.3: Compliance Management

* **Requirement**: The system must support automated compliance checking and reporting with IAM integration
* **Rationale**: Ensures adherence to regulatory requirements with comprehensive identity audit trails
* **Acceptance Criteria**:
  + Support SOC 2, GDPR, and HIPAA compliance frameworks with identity-aware controls
  + Enable automated compliance scanning and reporting with authentication metrics
  + Provide policy violation tracking and remediation workflows
  + Support data classification and handling policies
  + Generate automated access review reports with RBAC analysis
  + Track failed authentication attempts and security incidents
  + Support “right to be forgotten” with automated user data deletion

#### FR-009: Developer Experience

##### FR-009.1: Self-Service Capabilities

* **Requirement**: The system must provide self-service capabilities for development teams
* **Rationale**: Improves developer productivity and reduces operational overhead
* **Acceptance Criteria**:
  + Provide web-based interfaces for resource management
  + Support CLI tools for automation and scripting
  + Enable template-based resource provisioning
  + Provide comprehensive documentation and tutorials

##### FR-009.2: CI/CD Integration

* **Requirement**: The system must integrate with existing CI/CD pipelines
* **Rationale**: Enables automated testing and deployment workflows
* **Acceptance Criteria**:
  + Support webhook integration with Git repositories
  + Enable automated image building and scanning
  + Provide integration with popular CI/CD tools (Jenkins, GitLab CI, etc.)
  + Support blue-green and canary deployment strategies

## Nfr

### Non-Functional Requirements

#### Overview

This document outlines the non-functional requirements for the RH OVE multi-cluster ecosystem, defining quality attributes and constraints.

#### Performance Requirements

##### Latency

* **FR-NFR-001**: API response time must be < 200ms for 95% of requests
* **FR-NFR-002**: Cross-cluster communication latency must be < 50ms
* **FR-NFR-003**: VM startup time must be < 60 seconds for standard workloads

##### Throughput

* **FR-NFR-004**: System must support deployment of 100+ VMs per hour per cluster
* **FR-NFR-005**: Monitoring system must handle 10,000+ metrics per second
* **FR-NFR-006**: Log aggregation must process 1GB+ of logs per hour

##### Resource Utilization

* **FR-NFR-007**: CPU utilization must not exceed 80% under normal load
* **FR-NFR-008**: Memory utilization must not exceed 85% under normal load
* **FR-NFR-009**: Storage utilization must not exceed 90% capacity

#### Availability Requirements

##### Uptime

* **FR-NFR-010**: Production clusters must achieve 99.9% uptime (< 8.77 hours downtime/year)
* **FR-NFR-011**: Management cluster must achieve 99.95% uptime (< 4.38 hours downtime/year)
* **FR-NFR-012**: Critical services must have < 30 seconds failover time

##### Disaster Recovery

* **FR-NFR-013**: RPO (Recovery Point Objective) must be < 4 hours
* **FR-NFR-014**: RTO (Recovery Time Objective) must be < 8 hours
* **FR-NFR-015**: Cross-region failover must complete within 15 minutes

#### Scalability Requirements

##### Horizontal Scaling

* **FR-NFR-016**: System must support minimum 10 application clusters
* **FR-NFR-017**: Each cluster must support 1000+ pods
* **FR-NFR-018**: System must scale to 50,000+ containers across all clusters

##### Vertical Scaling

* **FR-NFR-019**: Individual VMs must scale up to 64 vCPUs and 512GB RAM
* **FR-NFR-020**: Storage volumes must scale up to 100TB per volume
* **FR-NFR-021**: Network bandwidth must scale to 25Gbps per node

#### Security Requirements

##### Authentication & Authorization

* **FR-NFR-022**: All API access must use multi-factor authentication
* **FR-NFR-023**: RBAC must be enforced across all cluster components
* **FR-NFR-024**: Service accounts must use time-limited tokens

##### Data Protection

* **FR-NFR-025**: All data in transit must be encrypted (TLS 1.3+)
* **FR-NFR-026**: All data at rest must be encrypted (AES-256)
* **FR-NFR-027**: Encryption keys must be rotated every 90 days

##### Network Security

* **FR-NFR-028**: All inter-cluster communication must be encrypted
* **FR-NFR-029**: Network policies must deny by default
* **FR-NFR-030**: Network segmentation must isolate environments

##### Compliance

* **FR-NFR-031**: System must maintain SOC 2 Type II compliance
* **FR-NFR-032**: Audit logs must be retained for 7 years
* **FR-NFR-033**: Security scanning must occur daily

#### Reliability Requirements

##### Fault Tolerance

* **FR-NFR-034**: System must survive single node failures without service interruption
* **FR-NFR-035**: System must survive single AZ failures in multi-AZ deployments
* **FR-NFR-036**: Data must be replicated across minimum 3 nodes

##### Error Handling

* **FR-NFR-037**: All errors must be logged with appropriate severity levels
* **FR-NFR-038**: Transient errors must be retried with exponential backoff
* **FR-NFR-039**: Critical errors must trigger automated alerting

##### Monitoring & Observability

* **FR-NFR-040**: System health must be monitored with 99% coverage
* **FR-NFR-041**: Metrics retention must be minimum 1 year
* **FR-NFR-042**: Distributed tracing must be enabled for all services

#### Maintainability Requirements

##### Deployment & Updates

* **FR-NFR-043**: Zero-downtime deployments must be supported
* **FR-NFR-044**: Rollback capability must be available within 5 minutes
* **FR-NFR-045**: Automated testing must cover 90%+ of functionality

##### Configuration Management

* **FR-NFR-046**: All configuration must be version controlled
* **FR-NFR-047**: Configuration changes must be auditable
* **FR-NFR-048**: Infrastructure as Code must be used for all deployments

##### Documentation

* **FR-NFR-049**: All APIs must have OpenAPI specifications
* **FR-NFR-050**: Runbooks must be available for all operational procedures
* **FR-NFR-051**: Architecture decisions must be documented in ADRs

#### Usability Requirements

##### User Interface

* **FR-NFR-052**: Web UI must be responsive and mobile-friendly
* **FR-NFR-053**: API must follow RESTful design principles
* **FR-NFR-054**: CLI tools must provide comprehensive help documentation

##### Accessibility

* **FR-NFR-055**: Web interfaces must meet WCAG 2.1 AA standards
* **FR-NFR-056**: Multi-language support must be available
* **FR-NFR-057**: High contrast mode must be supported

#### Capacity Requirements

##### Storage

* **FR-NFR-058**: Minimum 100TB usable storage per production cluster
* **FR-NFR-059**: Storage must support 10,000+ IOPS per cluster
* **FR-NFR-060**: Backup storage must retain data for 5 years

##### Network

* **FR-NFR-061**: Minimum 10Gbps connectivity between clusters
* **FR-NFR-062**: Maximum 5ms latency within cluster networks
* **FR-NFR-063**: Network must support jumbo frames (9000 MTU)

##### Compute

* **FR-NFR-064**: Minimum 1000 vCPUs available per production cluster
* **FR-NFR-065**: Minimum 4TB RAM available per production cluster
* **FR-NFR-066**: Support for GPU workloads (minimum 8 GPUs per cluster)

#### Compliance & Regulatory Requirements

##### Data Governance

* **FR-NFR-067**: Data must be classified and labeled appropriately
* **FR-NFR-068**: PII data must be encrypted and access-controlled
* **FR-NFR-069**: Data retention policies must be automatically enforced

##### Audit & Logging

* **FR-NFR-070**: All administrative actions must be logged
* **FR-NFR-071**: Logs must be tamper-proof and timestamped
* **FR-NFR-072**: Compliance reports must be generated automatically

##### Privacy

* **FR-NFR-073**: Right to be forgotten must be supported
* **FR-NFR-074**: Data minimization principles must be enforced
* **FR-NFR-075**: Privacy by design must be implemented