TECHNICAL WHITE PAPER: May 2026

Nirmata Kubernetes Governance on vSphere Kubernetes Service on VMware Cloud Foundation

Reference Architecture

Table of Contents

Executive Summary

As Kubernetes adoption accelerates across enterprise data centers, the complexity of managing workloads at scale introduces significant governance risk. Security misconfigurations, privilege escalation, unauthorized images, and policy drift across clusters are among the most common root causes of security incidents in Kubernetes environments. For organizations in regulated industries—financial services, healthcare, government, and energy—the stakes are higher: a single policy violation can lead to audit findings, compliance failures, or breach exposure.

vSphere Kubernetes Service (VKS) provides an enterprise-grade, CNCF-compliant Kubernetes runtime built directly into VMware Cloud Foundation (VCF). It delivers the infrastructure foundation organizations need to run modern workloads with confidence. Nirmata complements VKS by providing the policy and governance layer that ensures every cluster, workload, and API interaction is continuously evaluated against defined compliance standards.

This whitepaper describes how Nirmata Enterprise for Kyverno and Nirmata Control Hub deploy on VKS to deliver unified Kubernetes governance—from pod admission through continuous compliance—across single and multi-cluster environments. The solution supports fully air-gapped on-premises deployments as well as cloud-managed operation via the Nirmata Control Hub SaaS service.

- **Policy Enforcement at Admission:** Block non-compliant workloads before they run using Kyverno admission controller policies, enforced natively within VKS.
- **Multi-Cluster Governance:** Distribute and enforce policies consistently across all VKS clusters from a single control plane via Nirmata Control Hub.
- **Continuous Compliance:** Continuously evaluate cluster state against Pod Security Standards, RBAC best practices, and image verification policies, generating audit-ready reports.
- **Flexible Deployment:** Deploy fully on-premises (including air-gapped environments) or connect to Nirmata Control Hub as a SaaS service, the same Kyverno engine underlies both models.
- **GitOps-Native:** Manage policies as code alongside application manifests using Argo CD, integrated natively with VKS Supervisor services.

Benefits of using VKS with modern applications:

Lower TCO: With VKS organizations can reduce silos, leverage existing tools and skill sets without having to retrain staff and/or change existing processes. Utilizing unified lifecycle management across infrastructure components to stay up to date with the most recent patches and minimizing security risks.

Operational Simplicity: VKS is engineered for unparalleled operational simplicity, leveraging the familiarity of existing vSphere tools, skills, and workflows. This design philosophy significantly reduces the learning curve for IT teams and streamlines management processes. With VKS, organizations benefit from automated cluster provisioning, which accelerates deployment times and minimizes manual configuration errors. Furthermore, its robust capabilities extend to automated upgrades and comprehensive lifecycle management. This integrated approach ensures consistent operations, reduces overhead, and frees up valuable resources to focus on innovation rather than infrastructure maintenance.

Run and Manage Kubernetes at Scale: Effortlessly deploy and manage Kubernetes clusters at scale, leveraging a built-in, Cloud Native Computing Foundation (CNCF) certified Kubernetes distribution. VKS provides fully automated lifecycle management, streamlining operations from initial setup to ongoing maintenance and upgrades. This comprehensive approach ensures that organizations can harness the power of Kubernetes for their containerized applications with unparalleled efficiency and reliability, without the complexities typically associated with large-scale Kubernetes deployments.

vSphere Kubernetes Service

vSphere Kubernetes Service (VKS) is the Kubernetes runtime built directly into VMware Cloud Foundation (VCF). With CNCF certified Kubernetes, VKS enables platform engineers to deploy and manage Kubernetes clusters while leveraging

a comprehensive set of cloud services in VCF. Cloud admins benefit from the support for N-2 Kubernetes versions, enterprise grade security, and simplified lifecycle management for modern apps adoption.

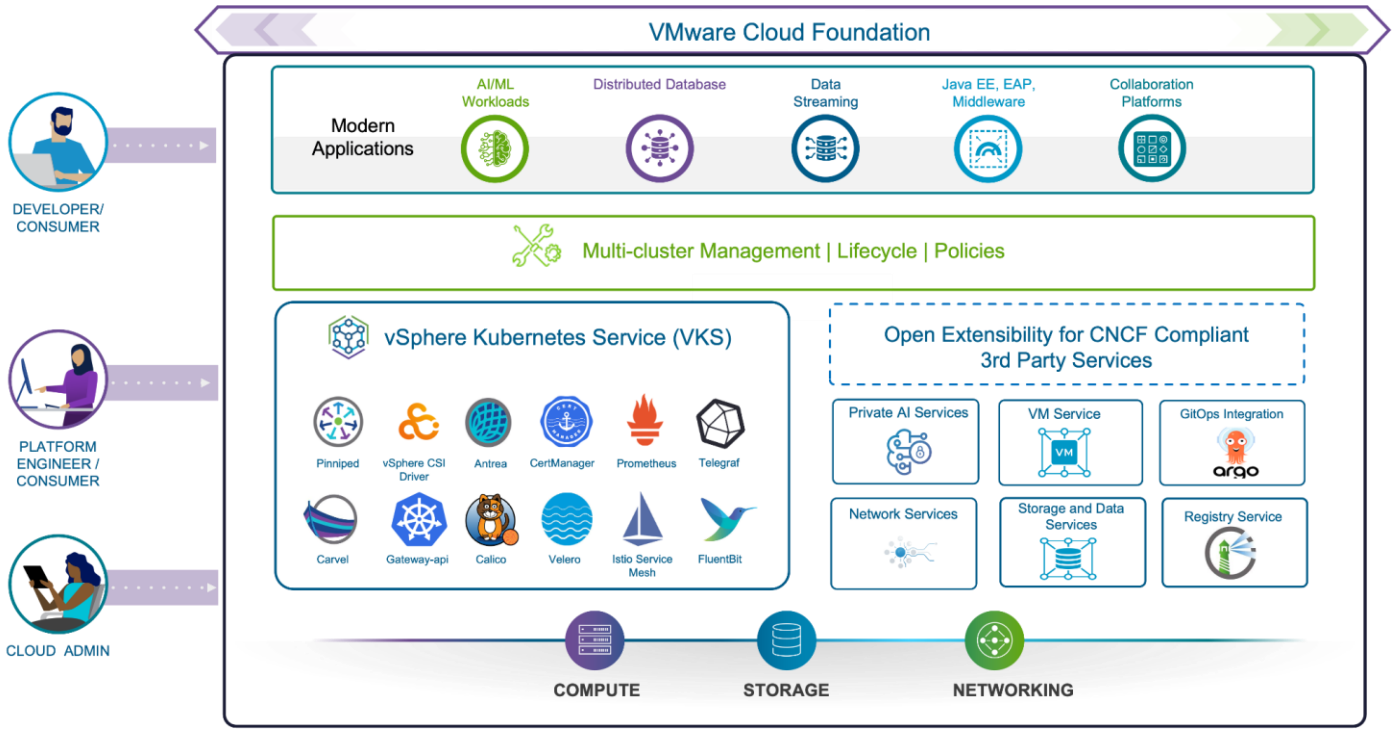

Figure 1: VKS on VCF Ecosystem

Nirmata provides a purpose-built Kubernetes governance platform based on Kyverno—a CNCF-graduated policy engine designed specifically for Kubernetes. Unlike general-purpose solutions, Kyverno uses native Kubernetes resources (YAML) for policies, eliminating the need to learn a separate policy language. Nirmata extends the open-source Kyverno engine with enterprise capabilities including policy lifecycle management, multi-cluster distribution, compliance reporting, and role-based access control.

The Nirmata platform consists of two tightly integrated components:

- **Nirmata Enterprise for Kyverno:** A hardened, supported distribution of the Kyverno admission controller, deployable as a standalone component in any Kubernetes cluster—including air-gapped VKS environments. It provides admission control, background scanning, image verification, and policy exception management.
- **Nirmata Control Hub:** A centralized multi-cluster management plane that distributes policies, aggregates compliance posture, and provides governance dashboards across all connected VKS clusters. Available as a SaaS service or self-hosted deployment.

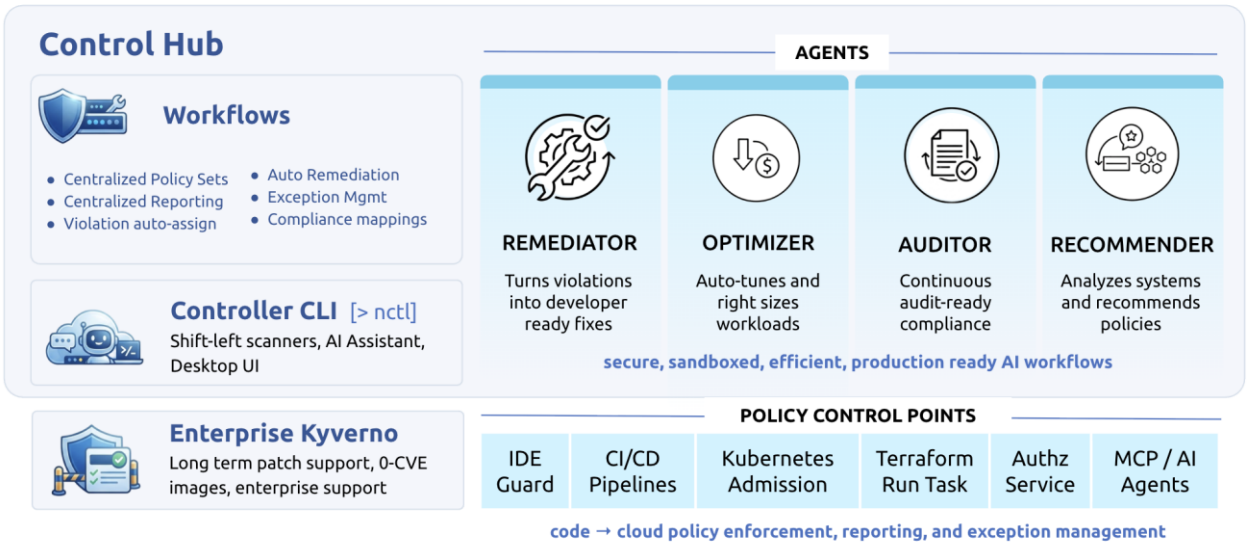

Figure 2: Nirmata Platform Architecture — Enterprise for Kyverno and Control Hub

Solution Architecture

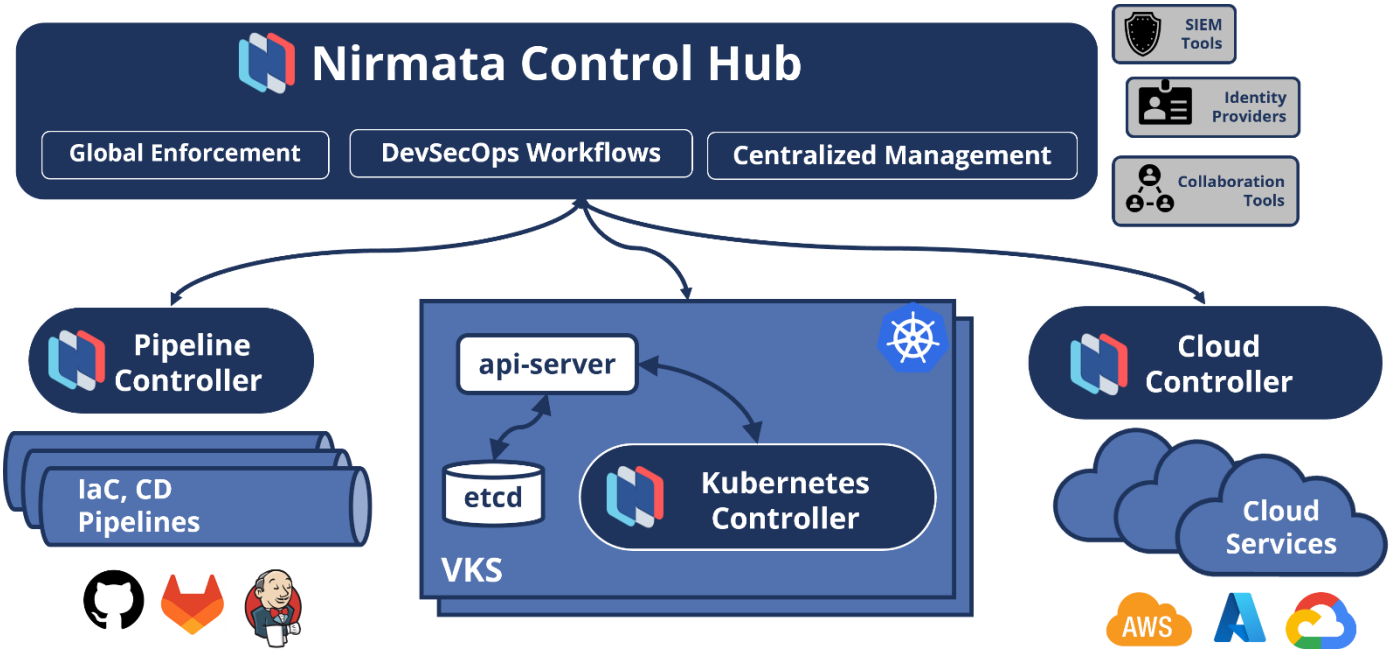

Figure 3: Nirmata on VKS — Admission Control and Multi-Cluster Governance Architecture

At its core, the Supervisor control plane provides essential services, such the Harbor container registry, the Contour ingress controller, and the Argo CD GitOps utility. For more information on vSphere Supervisor services and installation, visit: https://techdocs.broadcom.com/us/en/vmware-cis/vcf/vsphere-supervisor-services-and-standalone-components/latest.html

This control plane is responsible for the automated deployment and lifecycle management of CNCF-compliant Kubernetes clusters via the vSphere Kubernetes Service (VKS) via Cluster API. The specific version of VKS used in this white paper is 3.6.

Nirmata integrates at the Kubernetes admission layer within each VKS cluster, intercepting all resource creation and modification requests via Kubernetes webhooks. The Nirmata Control Hub communicates with each cluster's Nirmata agent to distribute policy updates and collect compliance telemetry.

Table 1: Installed Supervisor Components

Component	Version	Notes
vSphere Kubernetes Service	3.6	Allows for CNCF K8s clusters

VKS Cluster Configuration

The cluster comprised of three control plane nodes and three worker nodes. Storage was provided by vSAN Express Storage Architecture (ESA), using the default RAID 5 storage policy. Transparently, the vSphere CSI driver exposed this policy to an adjacent Kubernetes storage class, **vsan-esa-default-policy-raid5**.

Worker node resources were configured using a *best-effort-2xlarge* profile, whilst control plane nodes utilized the guaranteed large profile.

Table 2: Kubernetes Cluster Configuration Summary

Component	Configuration	Notes
Kubernetes Control Plane	3 Replicas VM Class: best-effort-medium vSAN RAID 5	VM Class: “best-effort-medium” Storage Class: “vsan-esa-default-policy-raid5”
Kubernetes Worker Nodes	3 Replicas VM Class: best-effort-medium vSAN RAID 5	VM Class: “best-effort-medium” Storage Class: “vsan-esa-default-policy-raid5”
Kubernetes Release	v1.35.2	
OS Image	Ubuntu 24.04	Kernel version 6.8.0-90-generic

The VKS cluster configuration is defined declaratively in the YAML file below. As previously outlined, applying this manifest to the Supervisor triggers the creation of the cluster, which is then managed and reconciled by the Cluster API.

```
# vks-cluster.yaml

apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  name: kyverno-vks
  namespace: nirmata
spec:
  clusterNetwork:
    services:
      cidrBlocks: ["10.96.0.0/12"]
    pods:
      cidrBlocks: ["192.168.0.0/16"]
    serviceDomain: "cluster.local"
  topology:
    class: builtin-generic-v3.6.0
    version: v1.35.2---vmware.1-vkr.3
    controlPlane:
      replicas: 3
      metadata:
        annotations:
          run.tanzu.vmware.com/resolve-os-image: os-name=ubuntu,os-version=24.04
```

```
workers:
  machineDeployments:
    - class: node-pool
      name: node-pool-1
      replicas: 3
      variables:
        overrides:
        - name: vmClass
          value: best-effort-medium
      metadata:
        annotations:
          run.tanzu.vmware.com/resolve-os-image: os-name=ubuntu,os-version=24.04
variables:
  - name: vmClass
    value: best-effort-medium
  - name: storageClass
    value: vsan-esa-default-policy-raid5
  - name: vsphereOptions
    value:
      persistentVolumes:
        defaultStorageClass: vsan-esa-default-policy-raid5
```

Kubernetes Packages

VKS clusters (via the VKr release) include a default set of core packages, such as Antrea for networking and Pinniped for authentication.

Table 3: Kubernetes Cluster Packages

Package	Version	Notes
Antrea	2.5.1	VKS core package
Gateway API	1.4.0	VKS core package
Guest-cluster auth-service	1.4.6	VKS core package
Metrics Server	0.8.1	VKS core package
Pinniped	0.42.0	VKS core package
Secretgen Controller	0.20.1	VKS core package
vSphere CPI	1.35.0	VKS core package
vSphere CSI	3.7.0	VKS core package
Nirmata Enterprise for Kyverno	3.3.x	Helm package
Nirmata Cluster Agent	3.x	Helm package (Control Hub)

Solution Validation

Deployment Option 1: Standalone Policy Enforcement (On-Premises / Air-Gapped)

In the standalone model, Nirmata Enterprise for Kyverno is deployed directly into the VKS cluster using Helm. This model is appropriate for high-compliance or air-gapped environments where outbound connectivity to a SaaS service is not available. All policy management and enforcement operate entirely within the cluster.

Prerequisites: VKS Cluster v3.5 or later; cert-manager add-on; Helm 3; VCF CLI and kubectl; access to Nirmata Helm chart repository (or a mirrored registry for air-gapped deployments).

Step 1: Add the Nirmata Helm repository and install Nirmata Enterprise for Kyverno:

```
helm repo add nirmata https://nirmata.github.io/kyverno-charts
helm repo update
helm install kyverno nirmata/kyverno \
  --version 3.7.1 -n kyverno --create-namespace \
  --set features.policyExceptions.namespace="kyverno" \
  --set crds.reportsServer.enabled=false \
  --set features.policyExceptions.enabled=true
```

Step 2: Verify the Kyverno admission controller is running:

```
kubectl get pods -n nirmata-system

# Expected output:
# nirmata-kyverno-admission-controller-xxxxx     Running
# nirmata-kyverno-background-controller-xxxxx    Running
# nirmata-kyverno-cleanup-controller-xxxxx       Running
# nirmata-kyverno-reports-controller-xxxxx       Running
```

Deployment Option 2: Nirmata Control Hub (Multi-Cluster / SaaS)

In the Control Hub model, each VKS cluster runs a Nirmata cluster agent that connects to the Nirmata Control Hub—either the SaaS instance at app.nirmata.io or a self-hosted deployment. The Control Hub distributes policies centrally and aggregates compliance posture across all connected clusters.

Prerequisites: VKS Cluster v3.5 or later; cert-manager add-on; Helm 3; VCF CLI and kubectl; outbound connectivity to app.nirmata.io (port 443) or self-hosted Control Hub endpoint; cluster token from the Nirmata Control Hub UI.

Step 1: Install the Nirmata cluster agent via Helm using the cluster token from the Control Hub UI:

```
helm install nirmata-kube-controller nirmata/nirmata-kube-controller \
  --version 0.3.15 -n nirmata --create-namespace \
  --set cluster.name=fvsd \
  --set apiToken==<XXXXXXXXXXXX> \
  --set features.policyExceptions.enabled=true \
  --set features.policySets.enabled=true \
  --set clusterOnboardingToken=<XXXXXXXXXXXX>

helm install kyverno-operator nirmata/nirmata-kyverno-operator \
  --version 0.9.1 -n nirmata-system \
  --create-namespace \
  --set enablePolicyset=true
```

Step 2: Verify the cluster appears as connected in the Nirmata Control Hub dashboard. Policy distribution and compliance reporting activate automatically upon registration.

GitOps Integration with Argo CD

VKS includes Argo CD as a Supervisor service, enabling GitOps-driven policy management. Kyverno policies stored in a Git repository are automatically synchronized and applied to target VKS clusters by Argo CD, ensuring policy-as-code workflows align with application deployment pipelines.

Example Argo CD Application manifest for Nirmata policy deployment:

```
apiVersion: argoproj.io/v1alpha1
kind: Application
```

```
metadata:
  name: nirmata-policies
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/your-org/kyverno-policies
    targetRevision: main
    path: policies/vks
  destination:
    server: https://kubernetes.default.svc
    namespace: nirmata-system
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
```

Policy Validation

Nirmata enforces three foundational policy sets on VKS clusters. Full policy YAML examples will be provided in a companion repository.

1. Pod Security Standards (PSS): Enforces Kubernetes Pod Security Standards at admission. The policy validates that all pods comply with the restricted profile—disallowing privilege escalation, host namespaces, and root containers.

Clone and navigate to this folder
https://github.com/nirmata/kyverno-policies/tree/main/pod-security

```
#Install baseline policies by:
kubectl apply -k baseline/
Install both baseline and restricted policies:
kubectl apply -k .

#To update the policies to enforce, run the below command:
kustomize build https://github.com/nirmata/kyverno-policies/pod-security/enforce | kubectl apply -f –
```

2. RBAC Best Practices: Detects and blocks overly permissive RBAC configurations including wildcard permissions, cluster-admin bindings to service accounts, and bindings to the default service account.

```
https://github.com/nirmata/kyverno-policies/blob/main/rbac-best-practices-vpol/restrict-wildcard-resources/restrict-wildcard-resources.yaml
# Blocks verbs: ['*'] or resources: ['*'] in policy rules.
```

3. Image Verification: Enforces that container images are signed and attested using cosign before admission. Unsigned or unverified images from unauthorized registries are blocked.

```
https://github.com/nirmata/kyverno-policies/tree/main/VerifyImage
# Requires cosign signature from trusted key/keyless authority
# for all images in target namespaces.
```

Verification — deploy a non-compliant test pod to confirm enforcement:

```
kubectl run test-pod --image=nginx --privileged -n default

# Expected output:
# Error from server: admission webhook
```

```
#  "kyverno-resource.kyverno.svc" denied the request:
#  policy disallow-privileged-containers failed:
#  Privileged mode is not allowed.
```

Conclusion

Nirmata Enterprise for Kyverno and Nirmata Control Hub provide the governance layer that complements the enterprise-grade infrastructure of vSphere Kubernetes Service. By deploying Nirmata on VKS, organizations in regulated industries gain policy enforcement at admission, continuous compliance scanning, and multi-cluster governance—all managed as code via GitOps workflows using Argo CD.

The solution supports both fully air-gapped on-premises deployments and cloud-connected operation via Nirmata Control Hub, making it suitable for the broadest range of VCF customer environments. Platform and security teams gain a unified view of compliance posture across all VKS clusters, while development teams benefit from fast, consistent policy feedback integrated into their existing CI/CD pipelines.

Layering Nirmata over VKS transforms Kubernetes infrastructure from a deployment platform into a governed, auditable, and continuously compliant service delivery environment.

Item No: vmw-bc-wp-tech-uslet-word-2026; May-26