TECHNICAL WHITE PAPER: October 2026



Dynatrace on vSphere Kubernetes Service on VMware Cloud Foundation

Reference Architecture

Table of Contents

Executive Summary....3
vSphere Kubernetes Service 3
Dynatrace 4
Solution Architecture....5
VKS Cluster Configuration 6
Kubernetes Packages 7
Solution Validation....8
Install Dynatrace on a VKS cluster 8
Step 1. Authenticate to the cluster....8
Step 2. Create and label the dynatrace namespace....8
Step 3. Install Dynatrace Operator....9
Step 4. Generate the DynaKube configuration....9
Step 5. Apply the configuration....9
Validate the deployment 9
Troubleshooting: Dynatrace DaemonSets have no ready pods 12
Conclusion....12

Executive Summary

vSphere Kubernetes Service (VKS) provides platform engineering teams with an enterprise-grade Kubernetes runtime integrated directly into VMware Cloud Foundation (VCF), leveraging familiar vSphere tools and operational workflows. To ensure comprehensive visibility across this infrastructure, integrating an advanced observability platform is essential.

This reference architecture details the deployment and validation of Dynatrace on a VKS cluster using the Dynatrace Operator. By deploying the operator via Helm and configuring a declarative DynaKube custom resource, organizations can establish secure connectivity to their Dynatrace environment via outbound HTTPS without modifying the underlying vSphere infrastructure or Supervisor control plane.

Once integrated, Dynatrace delivers end-to-end visibility—including real-time topology mapping of nodes, namespaces, workloads, and pods, alongside automated container instrumentation without requiring modifications to application images. Furthermore, as documented in VMware Cloud Foundation 9.0, Dynatrace serves as a critical observability component within modern CI/CD pipelines (such as the Harness consumption model), providing vital health telemetry to drive automated deployment verification and rollback. This document outlines the core architecture, prerequisites, step-by-step installation procedures, and validation methodologies for implementing Dynatrace observability in a VKS environment.

Broadcom documents Dynatrace as the observability component of the Harness CI/CD consumption model for VCF 9.0, where it supplies the health data that drives automated deployment verification and rollback. See https://techdocs.broadcom.com/us/en/vmware-cis/vcf/vcf-service-administration-and-development/9-0/modern-applications/continuous-integration-and-delivery-with-harness.html for more information. This paper describes the underlying Dynatrace deployment architecture and validation.

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.

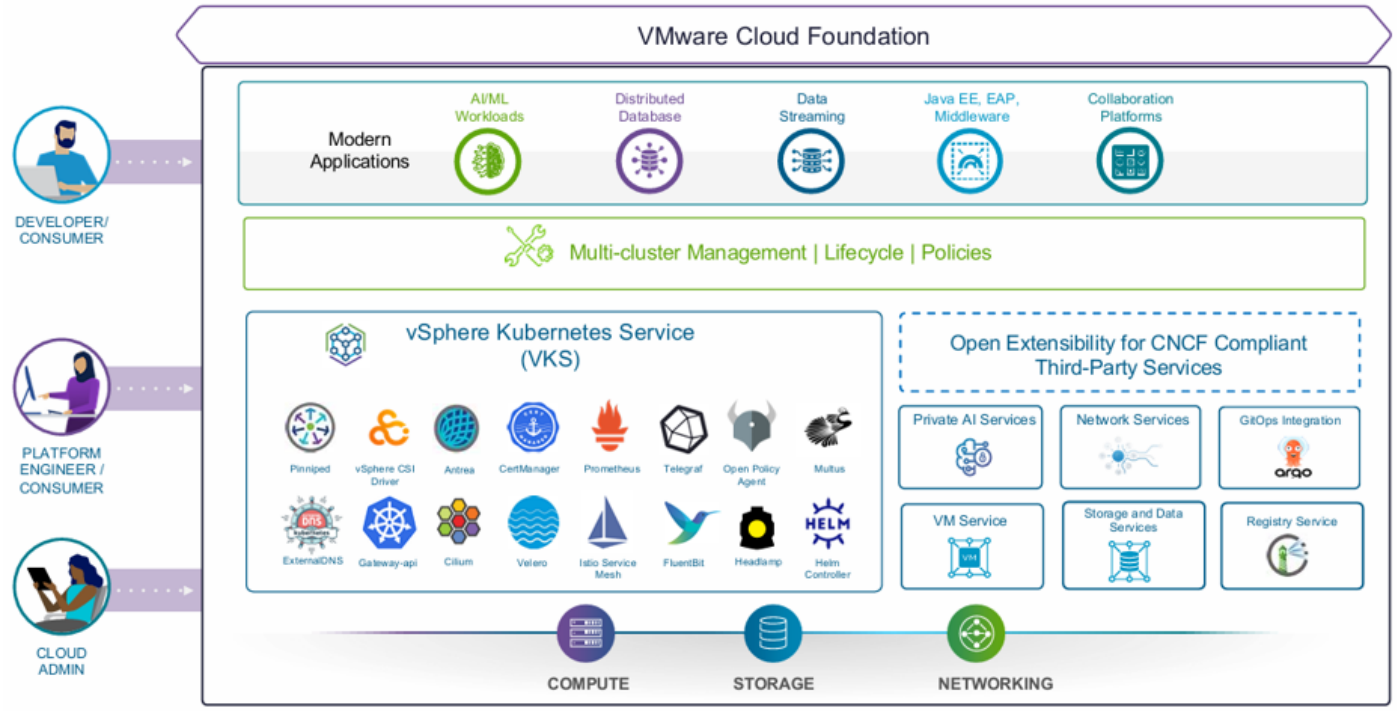


Figure 1: VKS on VCF Ecosystem

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 helps organizations stay up to date with the most recent patches and minimize 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.

Dynatrace

Dynatrace is an observability platform. It collects metrics, logs, traces, and topology from infrastructure and applications, then analyzes them to identify the root cause of the problems it detects. On Kubernetes it runs against a Dynatrace SaaS environment, and the monitored cluster needs egress to that environment's URL.

Three components run on the cluster.

- **Dynatrace Operator** is installed into its own namespace with Helm. It reads the DynaKube custom resource and handles rollout and lifecycle management of the other Dynatrace components, including ActiveGate and OneAgent.
- **ActiveGate** queries the Kubernetes API for cluster, workload, and event data, and routes telemetry from the cluster to your Dynatrace environment.
- **OneAgent** instruments application processes and produces service and distributed trace data. In Application observability it's injected into application pods. In Full-Stack observability it also runs on every node to monitor the hosts.

The DynaKube resource decides which of these you get. Kubernetes platform monitoring on its own collects cluster and workload data through ActiveGate. Adding Application observability injects OneAgent into application pods. Adding Full-Stack observability puts OneAgent on every node as well. This validation used Kubernetes platform monitoring + Application observability. For the full set of options, see https://docs.dynatrace.com/docs/ingest-from/setup-on-k8s/deployment.

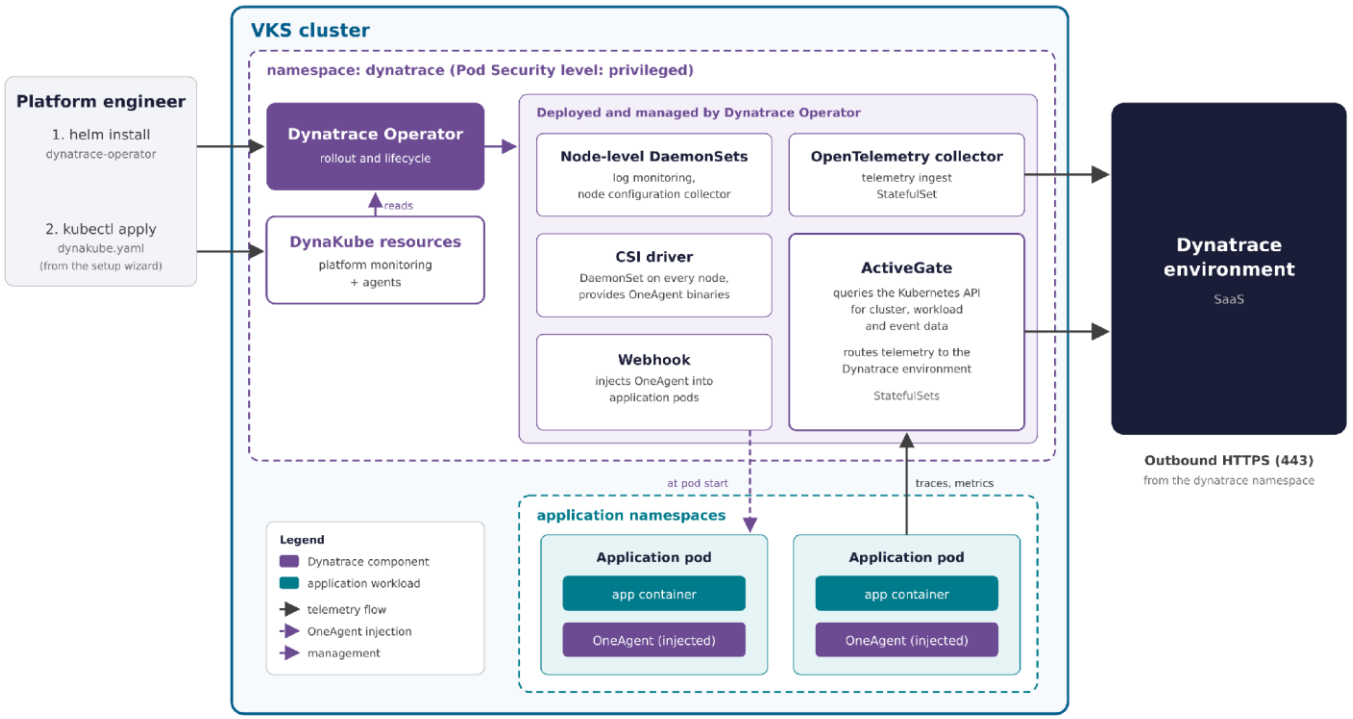


Figure 2: Dynatrace component deployment on VKS

Solution Architecture

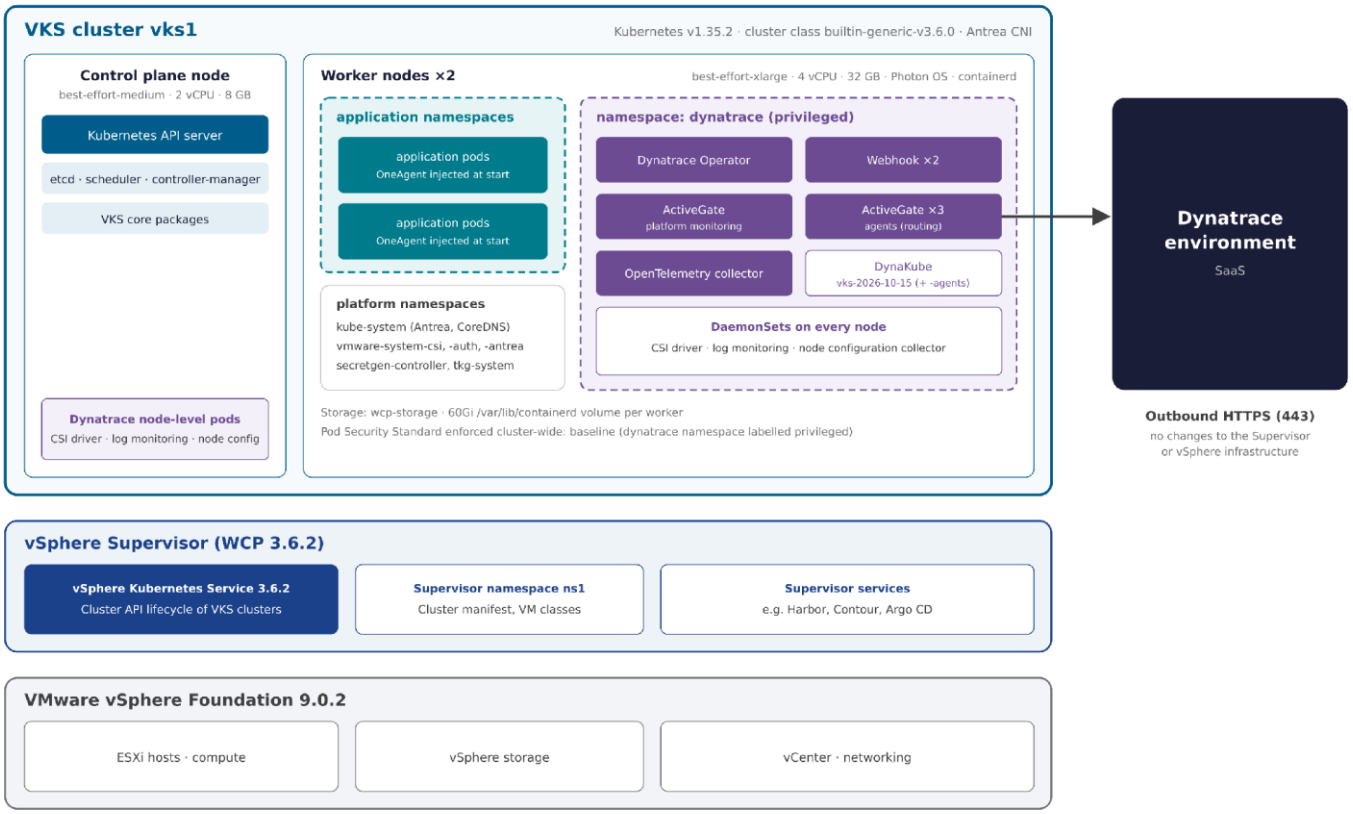


Figure 3: Dynatrace on VKS

At its core, the Supervisor provides essential services, such as 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 3.6.2.

Table 1: Installed Supervisor Components

Component	Version	Notes
vSphere Kubernetes Service	3.6.2	

VKS Cluster Configuration

The validation ran on a Broadcom-provided hands-on lab: a VMware vSphere Foundation 9.0.2 environment running VKS 3.6.2, with a three-node VKS cluster (one control plane node and two worker nodes) on Kubernetes 1.35.2. Dynatrace tests Kubernetes 1.27 through 1.36, so 1.35 is covered. For the version matrix, see https://docs.dynatrace.com/docs/ingest-from/technology-support/support-model-and-issues. The cluster was sized for functional validation, not as a sizing recommendation.

Table 2: Kubernetes Cluster Configuration Summary

Component	Configuration	Notes
Kubernetes Control Plane	1 Replica — 2x vCPU 8GB RAM	VM Class: “best-effort-medium” Storage Class: “wcp-storage”
Kubernetes Worker Nodes	2 Replicas — 4x vCPU 32GB RAM 60Gi container volume	VM Class: “best-effort-xlarge” Storage Class: “wcp-storage”
Kubernetes Release	v1.35.2+vmware.1	Cluster class builtin-generic-v3.6.0
OS Image	VMware Photon OS/Linux	Kernel 6.1.164-2.ph5, containerd 2.2.1+vmware.2-fips

The VKS cluster configuration is defined declaratively in the YAML file below. It is the manifest the lab cluster vks1 was created from in the Supervisor namespace ns1. Applying this manifest to the Supervisor triggers the creation of the cluster, which is then managed and reconciled by Cluster API.

```
# vks-cluster.yaml

apiVersion: cluster.x-k8s.io/v1beta2
kind: Cluster
metadata:
  name: vks1
  namespace: ns1
spec:
  clusterNetwork:
    pods:
      cidrBlocks:
        - 192.168.0.0/16
    services:
      cidrBlocks:
```

```
        - 10.96.0.0/12
    serviceDomain: cluster.local
  topology:
    classRef:
      name: builtin-generic-v3.6.0
      namespace: vmware-system-vks-public
    version: v1.35.2---vmware.1-vkr.3
    controlPlane:
      replicas: 1
      metadata:
        annotations:
          run.tanzu.vmware.com/resolve-os-image: "os-name=photon,os-version=5"
    workers:
      machineDeployments:
        - class: node-pool
          name: node-pool-1
          replicas: 2
          metadata:
            annotations:
              run.tanzu.vmware.com/resolve-os-image: "os-name=photon,os-version=5"
          variables:
            overrides:
              - name: vmClass
                value: best-effort-xlarge
              - name: volumes
                value:
                  - name: containerd
                    mountPath: /var/lib/containerd
                    storageClass: wcp-storage
                    capacity: 60Gi
    variables:
      - name: vmClass
        value: best-effort-medium
      - name: storageClass
        value: wcp-storage
      - name: vsphereOptions
        value:
          persistentVolumes:
            defaultStorageClass: wcp-storage
      - name: kubernetes
        value:
          security:
            podSecurityStandard:
              enforce: baseline
```

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	2.4.4	VKS core package
Gateway API	1.3.0	VKS core package
Guest-cluster auth-service	1.4.4	VKS core package

Metrics Server	0.8.0	VKS core package
Pinniped	0.41.0	VKS core package
Secretgen Controller	0.19.2	VKS core package
vSphere CPI	1.34.0	VKS core package
vSphere CSI	3.6.0	VKS core package
Dynatrace Operator	1.11.0	Helm chart; rollout and lifecycle of the components below
Dynatrace ActiveGate	1.347.49.20261014-002600	Kubernetes API monitoring and telemetry routing
Dynatrace OneAgent	1.347.53.20261007-171544	Code modules, injected into application pods
OpenTelemetry collector	0.58.0	Deployed by the agents DynaKube; image tag latest at validation time

Solution Validation

Install Dynatrace on a VKS cluster

Prerequisites:

- A VKS cluster running Kubernetes 1.27 or later, with kubectl pointed at it.
- Helm 3 or later.
- A Dynatrace environment, plus the Dynatrace Operator token and data ingest token that the Kubernetes setup wizard generates.
- Egress from the dynatrace namespace to your Dynatrace environment URL.
- Cert-manager is not required. Dynatrace was validated on the lab cluster with cert-manager removed.

Step 1. Authenticate to the cluster

Follow the instruction here to login into the VKS cluster.

Step 2. Create and label the dynatrace namespace

VKS clusters enforce the baseline Pod Security Standard by default. Dynatrace Operator supports only the privileged level, because the CSI driver and OneAgent pods need host access that baseline denies. Create the namespace and set the privileged level on it before you install Dynatrace Operator:

```
kubectl create namespace dynatrace
kubectl label namespace dynatrace \
  pod-security.kubernetes.io/enforce=privileged \
  pod-security.kubernetes.io/audit=privileged \
  pod-security.kubernetes.io/warn=privileged
```

```
holuser@console:~$ kubectl get crd | grep cert-manager.io
holuser@console:~$
holuser@console:~$ kubectl get ns dynatrace --show-labels
NAME        STATUS   AGE   LABELS
dynatrace   Active   17m   kubernetes.io/metadata.name=dynatrace,name=dynatrace,pod-security.kubernetes.io/a
udit=privileged,pod-security.kubernetes.io/enforce=privileged,pod-security.kubernetes.io/warn=privileged
```

Without the label, the CSI driver, log monitoring and node configuration collector DaemonSets fail with FailedCreate. See Troubleshooting below.

Step 3. Install Dynatrace Operator

```
helm install dynatrace-operator \
  oci://public.ecr.aws/dynatrace/dynatrace-operator \
  --version 1.11.0 \
  --namespace dynatrace
```

Helm reports STATUS: deployed. On Helm 4.0 and later you can add --rollback-on-failure; on Helm 3 the equivalent is --atomic.

Confirm the release and the pinned chart version:

```
helm status dynatrace-operator -n dynatrace
helm list -n dynatrace
```

```
holuser@console:~$ helm status dynatrace-operator -n dynatrace
NAME: dynatrace-operator
LAST DEPLOYED: Thu Oct 15 13:30:34 2026
NAMESPACE: dynatrace
STATUS: deployed
REVISION: 1
DESCRIPTION: Install complete
```

```
holuser@console:~$ helm list -n dynatrace
NAME                NAMESPACE   REVISION  UPDATED                                 STATUS    CHART                       APP VERSION
dynatrace-operator  dynatrace   1         2026-10-15 13:30:34.941899548 -0700 PDT deployed  dynatrace-operator-1.11.0   1.11.0
```

Step 4. Generate the DynaKube configuration

In Dynatrace, go to Kubernetes and add a cluster. Select Other distributions, then Kubernetes platform monitoring + Application observability. Name the cluster, generate the Dynatrace Operator token and the data ingest token, and download the dynakube.yaml it produces. That file carries the Secret holding both tokens alongside the DynaKube resources, so you don't create a secret separately.

Step 5. Apply the configuration

```
kubectl apply -f dynakube.yaml
```

The wizard produces two DynaKube resources, one for platform monitoring and one for the agents. In the lab they were named vks-2026-10-15 and vks-2026-10-15-agents. Lab output:

```
holuser@console:~$ kubectl apply -f dynakube.yaml
secret/vks-2026-10-15 created
Warning: Using deprecated feature flags: feature.dynatrace.com/k8s-app-enabled
dynakube.dynatrace.com/vks-2026-10-15 created
dynakube.dynatrace.com/vks-2026-10-15-agents created
```

The warning comes from an annotation in the wizard-generated file and doesn't stop the resources from being created.

Validate the deployment

Check that the Dynatrace components are running:

```
kubectl get daemonset -n dynatrace; kubectl get pods -n dynatrace
```

```
holuser@console:~$ kubectl get daemonset -n dynatrace
NAME                                DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
dynatrace-oneagent-csi-driver       3         3         3       3            3           <none>          18m
vks-2026-10-15-agents-logmonitoring 2         2         2       2            2           <none>          11m
vks-2026-10-15-node-config-collector 2        2         2       2            2           <none>          11m
holuser@console:~$
holuser@console:~$ kubectl get pods -n dynatrace
NAME                                    READY   STATUS    RESTARTS   AGE
dynatrace-oneagent-csi-driver-bqltv     4/4     Running   0          3m5s
dynatrace-oneagent-csi-driver-f97mg     4/4     Running   0          3m5s
dynatrace-oneagent-csi-driver-p9t54     4/4     Running   0          3m5s
dynatrace-operator-c79f784b4-nm2q9      1/1     Running   0          19m
dynatrace-webhook-649f7856b5-d67zg      1/1     Running   0          19m
dynatrace-webhook-649f7856b5-kfktw      1/1     Running   0          19m
vks-2026-10-15-activegate-0             1/1     Running   0          10m
vks-2026-10-15-agents-activegate-0      1/1     Running   0          12m
vks-2026-10-15-agents-activegate-1      1/1     Running   0          12m
vks-2026-10-15-agents-activegate-2      1/1     Running   0          12m
vks-2026-10-15-agents-logmonitoring-dr97h 1/1   Running   0          89s
vks-2026-10-15-agents-logmonitoring-n4bxc 1/1   Running   0          89s
vks-2026-10-15-agents-otel-collector-0  1/1     Running   0          12m
vks-2026-10-15-node-config-collector-dcbjz 1/1  Running   0          102s
vks-2026-10-15-node-config-collector-rvf9t 1/1  Running   0          102s
```

Then check both DynaKube resources:

```
kubectl get dynakube -n dynatrace
holuser@console:~$ kubectl get dynakube -n dynatrace
NAME                  APIURL                                      STATUS    AGE
vks-2026-10-15        https://[redacted].live.dynatrace.com/api   Running   38m
vks-2026-10-15-agents https://[redacted].live.dynatrace.com/api   Running   38m
```

After a few minutes all pods report Ready, the three DaemonSets have a ready pod on every node they target, and both DynaKube resources report Running. ActiveGate registers with your Dynatrace environment once it starts. The setup wizard confirms the connection, and the cluster then appears in Kubernetes with its nodes, namespaces, and workloads.

Then confirm that OneAgent is injected into application pods. Start a test pod in a new namespace and check its init containers:

```
kubectl create namespace inject-test
kubectl run nginx --image=nginx -n inject-test
kubectl get pod nginx -n inject-test -o jsonpath='{.spec.initContainers[*].name}'
holuser@console:~$ kubectl get pod nginx -n inject-test -o jsonpath='{.spec.initContainers[*].name}{"\n"}'
dynatrace-operator
holuser@console:~$ kubectl delete namespace inject-test
namespace "inject-test" deleted
```

The output is dynatrace-operator, the init container that injects OneAgent. Remove the test namespace afterwards with kubectl delete namespace inject-test.

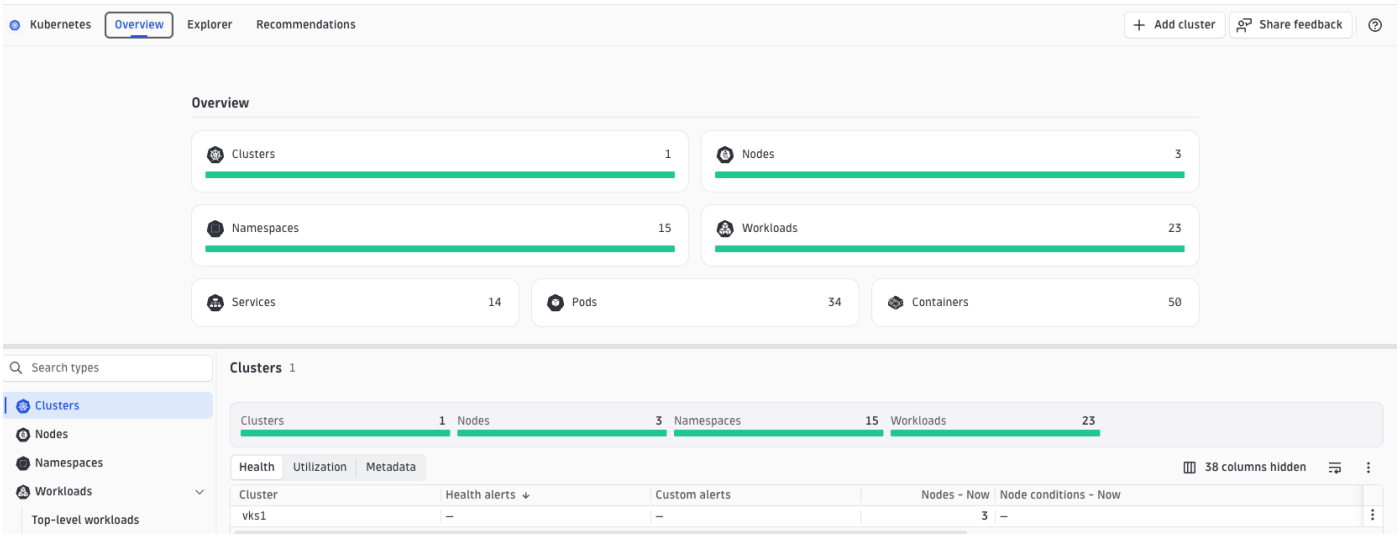


Figure 4: The VKS cluster in Dynatrace

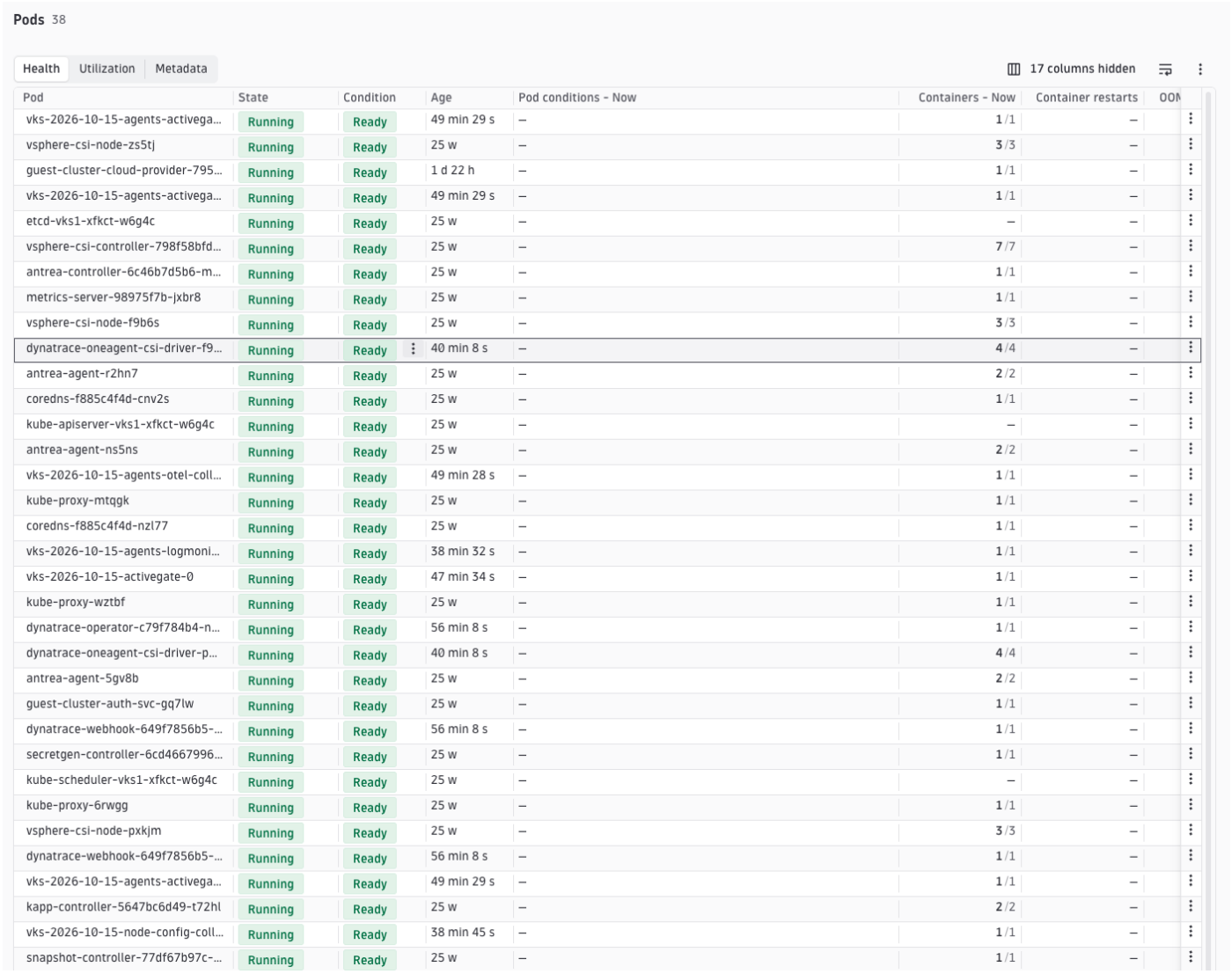


Figure 5: Pod health across the VKS cluster

Troubleshooting: Dynatrace DaemonSets have no ready pods

If the dynatrace namespace was created without the privileged label, kubectl get daemonset -n dynatrace shows 0 ready pods for the CSI driver, log monitoring and node configuration collector, and the namespace events show FailedCreate with violates PodSecurity "baseline:latest". Label the namespace as in Step 2:

```
kubectl label namespace dynatrace \
  pod-security.kubernetes.io/enforce=privileged \
  pod-security.kubernetes.io/audit=privileged \
  pod-security.kubernetes.io/warn=privileged --overwrite
```

Log monitoring and the node configuration collector recover on their own within a few minutes. The CSI driver DaemonSet may stay at 0 pods because of controller backoff; restart it:

```
kubectl rollout restart daemonset dynatrace-oneagent-csi-driver -n dynatrace
```

Then check that every DaemonSet reports READY equal to DESIRED with kubectl get daemonset -n dynatrace.

Conclusion

Deploying Dynatrace on VKS is a Helm install and a custom resource. Dynatrace Operator takes it from there, and you configure nothing on the Supervisor or in vSphere.

A platform team gets node and workload health in the same place as the behavior of the applications on the cluster. The same data is available through the Dynatrace API, which is how the Harness model in the VCF 9.0 documentation uses it to gate deployments.

Dynatrace also monitors vSphere directly through a vCenter integration, which would let you correlate VKS workload behavior with host and datastore conditions. That was outside the scope of this validation.

Copyright © 2026 Broadcom. All rights reserved.

The term “Broadcom” refers to Broadcom Inc. and/or its subsidiaries. For more information, go to www.broadcom.com. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies. Broadcom reserves the right to make changes without further notice to any products or data herein to improve reliability, function, or design. Information furnished by Broadcom is believed to be accurate and reliable. However, Broadcom does not assume any liability arising out of the application or use of this information, nor the application or use of any product or circuit described herein, neither does it convey any license under its patent rights nor the rights of others.

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