TECHNICAL WHITE PAPER: **July 2026**



Cribl Edge on vSphere Kubernetes Service on VMware Cloud Foundation

Reference Architecture

Table of Contents

Executive Summary 3
vSphere Kubernetes Service 3
Cribl Edge 4
Solution Architecture 5
VKS Cluster Configuration 5
Kubernetes Packages 7
Solution Validation 7
Install Cribl Edge on vSphere Kubernetes Service 7
Validate Cribl Edge running on vSphere Kubernetes Service 9
Conclusion 15

Executive Summary

This reference architecture describes the deployment and validation of Cribl Edge on VMware vSphere Kubernetes Service (VKS). Cribl Edge is a distributed telemetry collector that runs as a Kubernetes DaemonSet, streaming logs, metrics, and events from every node in a VKS cluster to Cribl Cloud, where operators can shape, route, and cost-optimize telemetry before delivery to downstream systems.

The solution provides VKS platform teams with a unified telemetry pipeline that decouples collection from destination — enabling flexible routing to observability platforms (Splunk, Elasticsearch, Datadog), storage systems (Amazon S3, Cribl Lake), and long-term retention tiers — all from a single agent per node. When combined with VKS's automated lifecycle management and CNCF-compliant Kubernetes runtime, Cribl Edge delivers turnkey Kubernetes observability without downstream vendor lock-in.

This document validates a Cribl Edge deployment on VKS 3.6 running Kubernetes v1.35.2, covering the Helm-based install workflow, the namespace-level Pod Security Admission (PSA) configuration required, and the environment variable needed to accommodate VKS's default self-signed kubelet certificates. Full command-line examples and verification procedures are provided.

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.

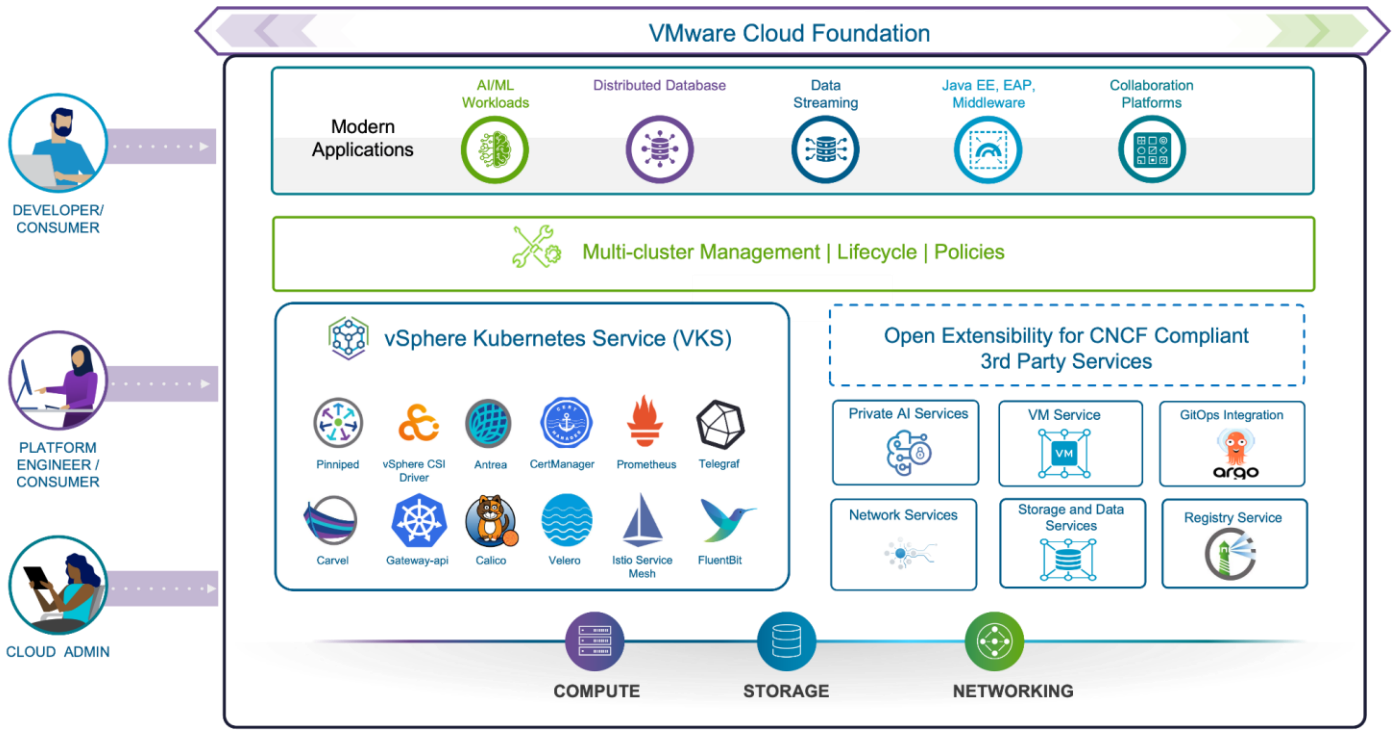


Figure 1: VKS on VCF Ecosystem

Cribl Edge

Cribl Edge is a distributed telemetry collection agent designed for modern infrastructure environments. Deployed as a Kubernetes DaemonSet, Cribl Edge collects logs, metrics, and system-state signals from every node in a VKS cluster with a small memory footprint per node, streaming that data to Cribl Cloud where it can be transformed, routed, and delivered to any downstream analytics, security, or storage platform — Splunk, Elasticsearch, Datadog, Amazon S3, Cribl Lake, or others.

Unlike traditional log forwarders that hardwire source data to a single destination, Cribl Edge separates collection from routing. Platform teams can send the same telemetry to multiple destinations simultaneously, transform it in flight (redaction, enrichment, reduction, format conversion between Syslog, OTel, HEC, and JSON), and control cost by tiering data by value. In Kubernetes environments — where telemetry volume grows quickly with cluster scale — this design lets platform teams observe more without commit-tier growth on downstream tools.

Cribl Edge integrates natively with the Kubernetes control plane to enrich events with pod, namespace, node, and workload metadata, and supports OpenTelemetry (OTLP), Syslog, HTTP, TCP/UDP, file, and system-metric sources out of the box.

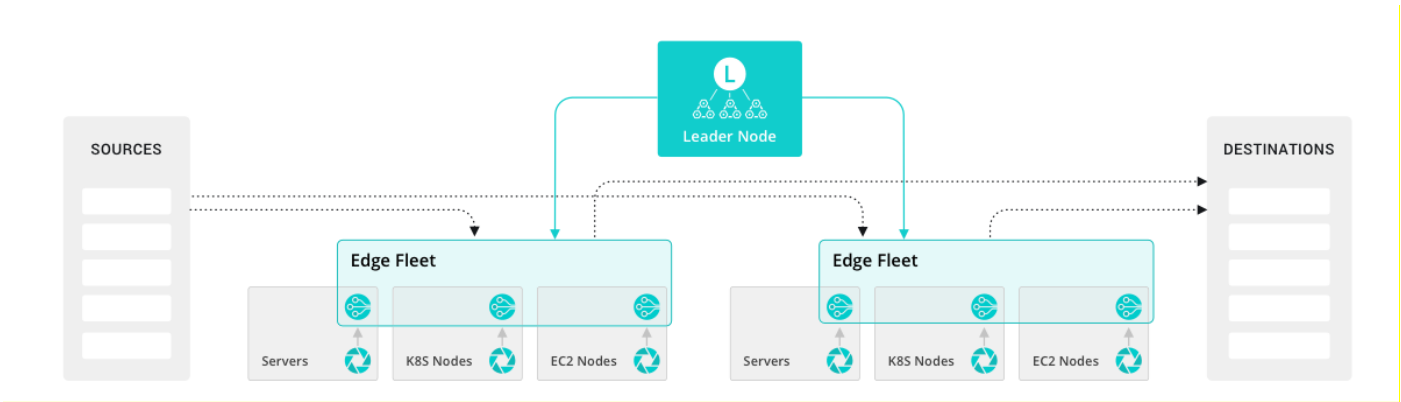


Figure 2: Cribl Edge deployment model — Sources on Servers, Kubernetes Nodes, and EC2 collect telemetry into per-fleet Edge Nodes managed by the Leader, and forward to configurable Destinations.

Solution Architecture

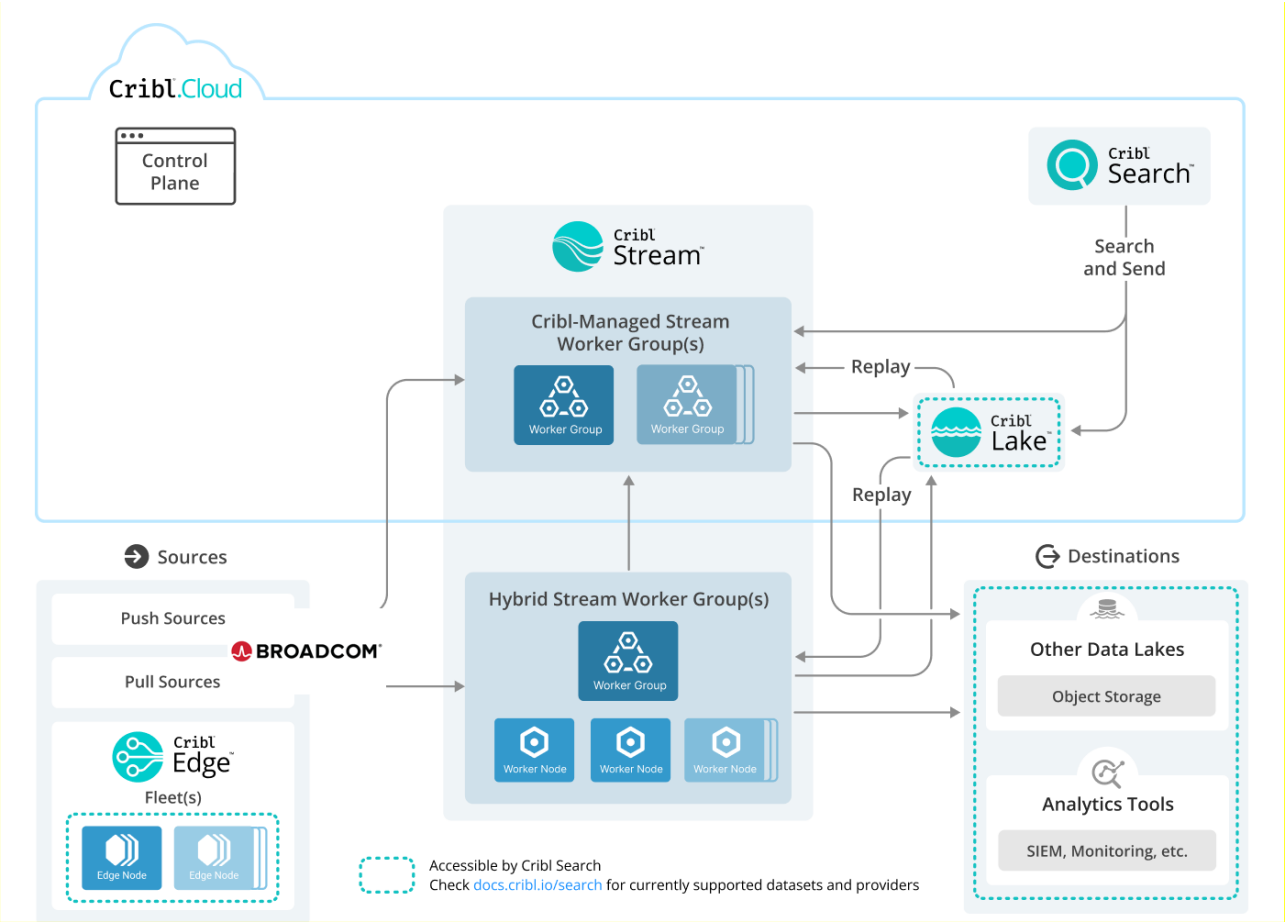


Figure 3: Cribl Suite solution architecture — Broadcom VKS worker nodes running Cribl Edge feed Cribl Stream, Cribl Search, and Cribl Lake in Cribl.Cloud, with routing to downstream data lakes and analytics tools.

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.

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 — 4x vCPU 16GB RAM vSAN RAID 5	VM Class: “best-effort-medium” Storage Class: “vsan-esa-default-policy-raid5”
Kubernetes Worker Nodes	3 Replicas — 4x vCPU 16GB RAM vSAN RAID 5	VM Class: “best-effort-2xlarge” Storage Class: “vsan-esa-default-policy-raid5”
Kubernetes Release	v1.35.2+vmware.1	
OS Image	VMware Photon OS/Linux	Kernel version 6.1.164-2.ph5, containerd 2.2.1+vmware.2-fips

The VKS cluster configuration is defined declaratively as a Cluster API resource on the vSphere Supervisor. The following YAML is a representative example — it reflects the cluster class, version, network layout, VM class, and storage class used in this validation, but has been simplified for readability and omits Supervisor-managed status and identity fields. Applying a manifest of this shape to the Supervisor triggers the creation of the workload cluster, which is then managed and reconciled by the Cluster API.

```
# vks1-cluster.yaml — representative
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  name: vks1
  namespace: pais-ns
spec:
  clusterNetwork:
    pods:
      cidrBlocks:
        - 192.168.0.0/16
    services:
      cidrBlocks:
        - 10.96.0.0/12
    serviceDomain: cluster.local
  topology:
    class: builtin-generic-v3.6.0
    version: v1.35.2---vmware.1-vkr.3
    variables:
      - name: kubernetes
        value:
          certificateRotation:
            enabled: true
            renewalDaysBeforeExpiry: 90
      - name: vmClass
        value: best-effort-2xlarge
      - name: storageClass
        value: vsan-esa-default-policy-raid5
    controlPlane:
      replicas: 3
```

```
  variables:
    overrides:
      - name: vmClass
        value: best-effort-medium
  metadata:
    annotations:
      run.tanzu.vmware.com/resolve-os-image: os-name=photon
workers:
  machineDeployments:
    - class: node-pool
      name: vks1-node-pool-1
      replicas: 3
      metadata:
        annotations:
          run.tanzu.vmware.com/resolve-os-image: os-name=photon
```

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
Cert-Manager	1.18.2	VKS add-on
Cribl Edge	4.18.2	ISV application — deployed as a Kubernetes DaemonSet via the official Cribl Helm chart. Provides distributed collection of container logs, cluster events, and node metrics, streaming to Cribl Cloud for centralized routing and transformation.

Solution Validation

Install Cribl Edge on vSphere Kubernetes Service

Overview: Cribl Edge on Kubernetes

Cribl Edge deploys to a Kubernetes cluster as a DaemonSet — one pod scheduled on every worker node — using the official Cribl Helm chart. Each pod authenticates to Cribl.Cloud (the Leader) over TLS on port 4200, receives its

configuration from the assigned Fleet, and collects three telemetry classes from the local node: container logs (via hostPath mounts to the node filesystem), cluster events (via the Kubernetes API server), and node and workload metrics (via the local kubelet API on port 10250).

At runtime, Cribl Edge uses the Pod's ServiceAccount to authenticate to the Kubernetes Cluster API and to the local kubelet. A ClusterRole with read-only permissions on Pods, Events, Nodes, and metrics endpoints is required. Cribl Edge automatically detects that it is running inside Kubernetes via the CRIBL_K8S_POD environment variable, and sizes its worker-process count based on CRIBL_K8S_CPU_LIMIT (which the Helm chart derives from the Pod's resources.limits.cpu). Health checks are exposed at /api/v1/health on port 9420.

Kubernetes distributions that use self-signed certificates for the cluster API server or the per-node kubelet — including all CAPI-based distributions such as VKS, Nutanix NKP, and kubeadm-based clusters — require the CRIBL_K8S_TLS_REJECT_UNAUTHORIZED environment variable to be set to 0. Without this, the Kubernetes Logs source fails to poll the kubelet's /configz endpoint with a self-signed certificate error, while Events and Metrics (which use the cluster API server via the ServiceAccount CA) continue to function.

Network Requirements

The Edge Nodes must be able to reach the following endpoints. The Leader connection on TCP/4200 is persistent; the two 443 endpoints are only required during the initial bootstrap.

Port	Protocol	Destination	Purpose
4200	TCP	Cribl.Cloud Leader	Ongoing outbound connection from each Edge node to the Leader for configuration delivery, heartbeats, and control-plane traffic.
443	TCP	cdn.cribl.io	One-time outbound during bootstrap: fetches the Edge install package from the Cribl CDN.
443	TCP	Cribl.Cloud Leader	One-time outbound during bootstrap: initial registration and Cribl.Cloud API calls.
10200	TCP	Cribl Stream Worker Group	Optional — Cribl HTTP Destination default. Required only if this Fleet forwards to a Cribl Stream Worker Group over the Cribl HTTP Destination. For all Cribl Stream port defaults (Splunk HEC 8088, TCP/JSON 10070, Cribl TCP 10300, Syslog 9514, and others), see https://docs.cribl.io/stream/ports/.

1. Prepare the namespace (Pod Security Admission)

VKS enables the Pod Security Admission (PSA) controller by default, and new namespaces enforce the baseline Pod Security Standard. The baseline profile forbids hostPath volumes, which Cribl Edge requires to read container logs and node metrics from the host filesystem. The target namespace must therefore be relabeled to the privileged level before Cribl Edge is deployed.

```
kubectl create namespace cribl

kubectl label namespace cribl \
  pod-security.kubernetes.io/enforce=privileged \
  pod-security.kubernetes.io/enforce-version=latest \
  --overwrite
```

2. Deploy Cribl Edge with the Cribl.Cloud bootstrap script

Cribl.Cloud generates a pre-populated bootstrap script for each Fleet that automates the DaemonSet deployment. The script embeds the workspace URL, Leader address, auth token, and target Worker Group — no manual token or URL handling is required. In the Cribl.Cloud UI, navigate to Fleet → your Worker Group → Edge Nodes → Add/Update Edge

Node, choose the target OS (Linux for VKS worker nodes running Photon OS), and copy the generated script. The script content updates in real time as options are changed, so the same UI supports pinning a specific Edge version through the Fleet's Target Software Version.

For Kubernetes deployments, add the CRIBL_K8S_TLS_REJECT_UNAUTHORIZED=0 environment variable to the generated Helm command so the Edge pods will tolerate the self-signed kubelet certificate on VKS worker nodes. An example of the final Helm invocation the bootstrap script produces:

```
helm install cribl-edge edge \
  --repo "https://criblio.github.io/helm-charts/" \
  --version "4.18.2" \
  --namespace cribl \
  --set "cribl.leader=tls://<TOKEN>@main-<workspace>.cribl.cloud?group=broadcom-k8s" \
  --set "env.CRIBL_K8S_TLS_REJECT_UNAUTHORIZED=0"
```

⚠ Security note: setting CRIBL_K8S_TLS_REJECT_UNAUTHORIZED=0 disables TLS certificate verification on outbound connections from Cribl Edge to the cluster API server and the local kubelet. This is required on VKS and other CAPI-based distributions because the kubelet serving certificate is signed by a per-node self-signed bootstrap CA that is not projected into pods. Kubelet traffic never leaves the node's own network namespace, so the exposure is bounded — but for production environments that require explicit certificate trust, mount the kubelet CA into the Edge pods and set the NODE_EXTRA_CA_CERTS environment variable instead. That preserves TLS verification while trusting the private CA.

3. Confirm the DaemonSet has scheduled pods

```
kubectl get daemonset cribl-edge -n cribl
```

Expected output on a three-node cluster:

```
NAME        DESIRED  CURRENT  READY  UP-TO-DATE  AVAILABLE  AGE
cribl-edge  3        3        3      3           3          2m
```

If CURRENT remains at 0 while DESIRED is non-zero, verify the namespace PSA label from step 1 was applied and check the DaemonSet events:

```
kubectl get ns cribl --show-labels
kubectl describe daemonset cribl-edge -n cribl
```

Validate Cribl Edge running on vSphere Kubernetes Service

1. Verify pod health

```
kubectl get pods -n cribl -o wide
```

All three pods should be in the Running state, one on each worker node.

2. Confirm connectivity to Cribl.Cloud (Leader)

```
kubectl logs -n cribl -l app.kubernetes.io/name=cribl-edge \
  --tail=100 | grep -E 'HBChannelClient|leader|Resolved dns'
```

Look for successful DNS resolution of the Leader hostname and periodic heartbeat exchanges reporting failed:0 and dropped:0. The absence of “handshake timeout” or “outbound is disconnected” messages indicates healthy connectivity to Cribl.Cloud.

For visual confirmation on the Cribl.Cloud side, open the destination configured for the Fleet — in this validation, the All_VKS Cribl Search destination — and inspect the Logs and Charts tabs. A successful destination-initialization message in the worker log indicates the Fleet’s configuration reached the Edge Nodes via the Leader, and a rising event-count line on the Charts tab with zero dropped events indicates the destination is actively receiving telemetry.

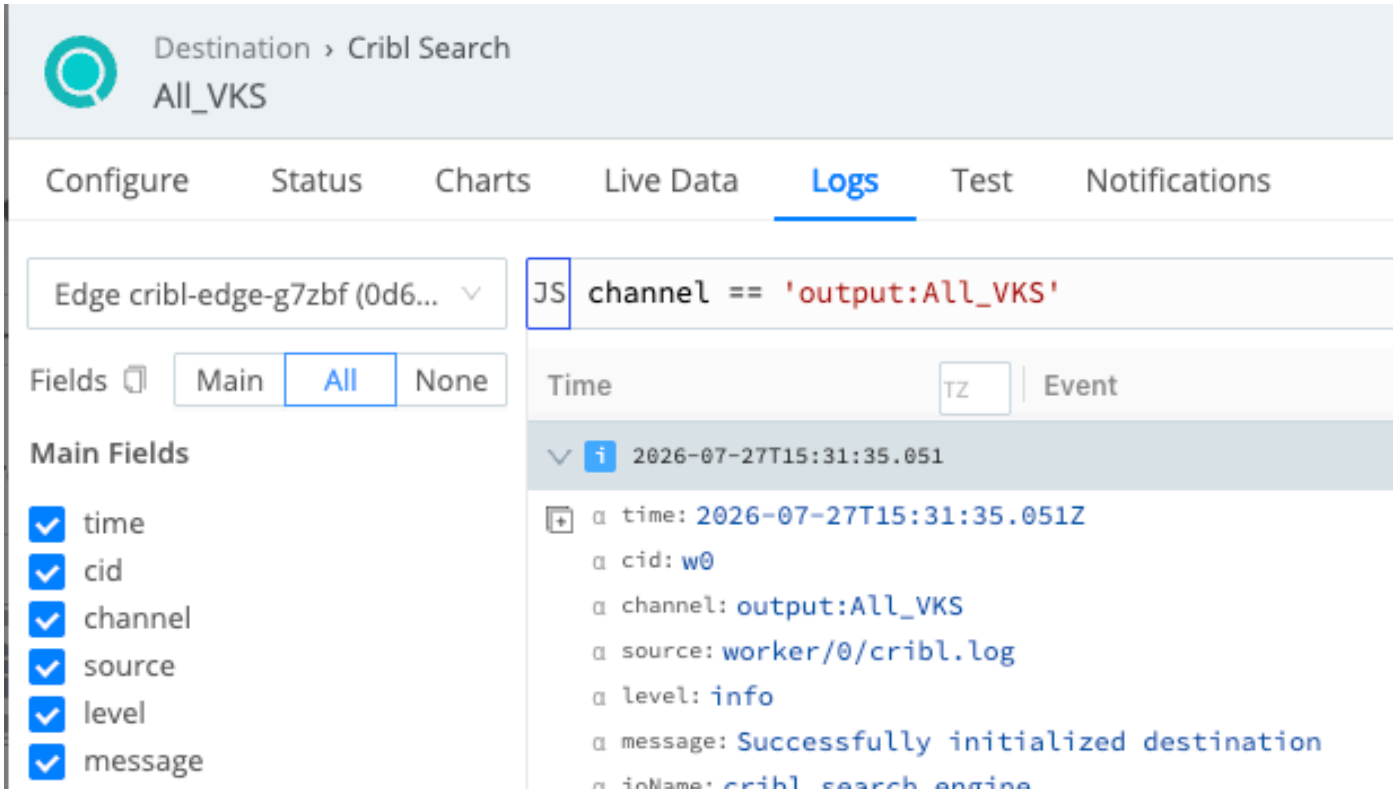


Figure 4: All_VKS destination Logs view — worker/0/cribl.log shows “Successfully initialized destination”, confirming the destination configuration was delivered from the Leader and applied on the Edge Node.

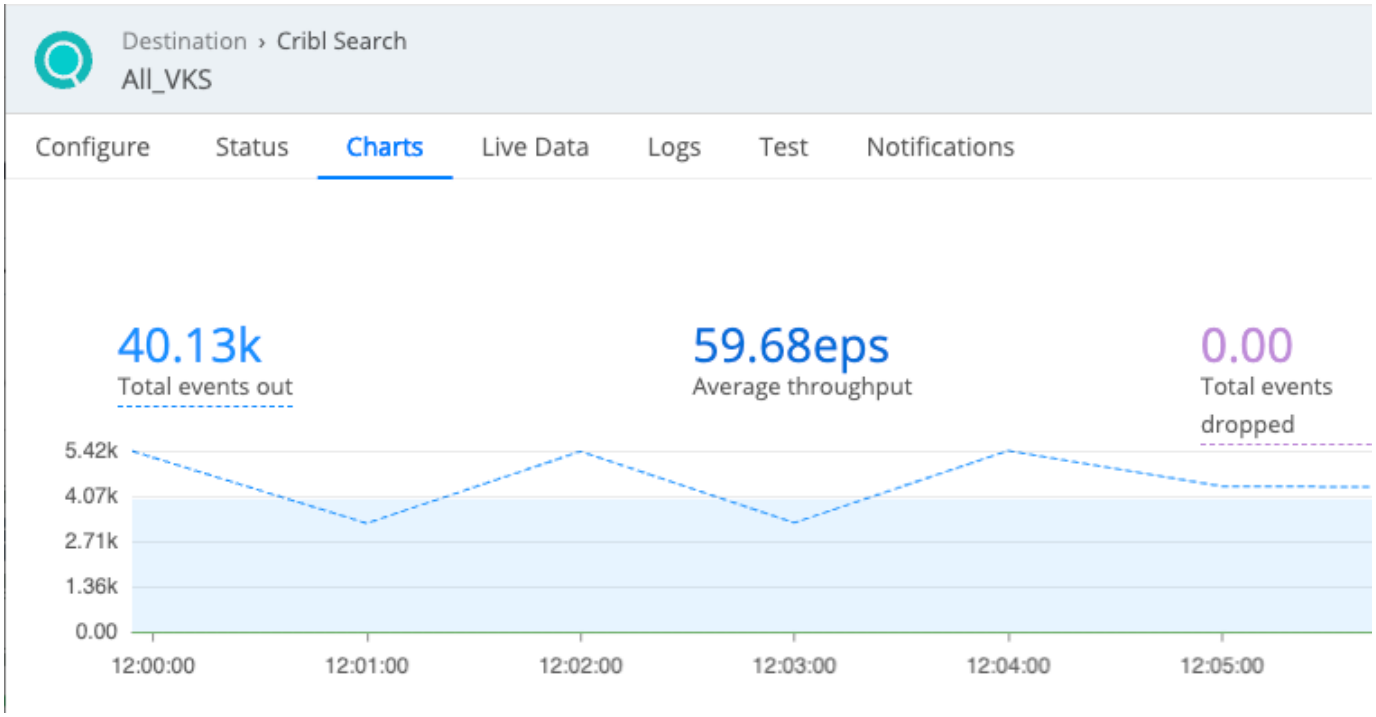


Figure 5: All_VKS destination Charts view — 40.13k total events out, 59.68 eps average throughput, 0 dropped events, confirming the Edge → destination path is healthy end to end.

3. Verify Fleet configuration and Edge Node registration

In the Cribl.Cloud UI, open Fleets → broadcom-k8s. The Fleet header should show three connected Nodes and the Collect view should show the configured sources routed to the target destinations. In this validation, three Kubernetes sources (in_kube_events, in_kube_logs, in_kube_metrics) plus File Monitor and Journal Files are routed to a Cribl Search destination named All_VKS with additional DevNull tap for diagnostics.

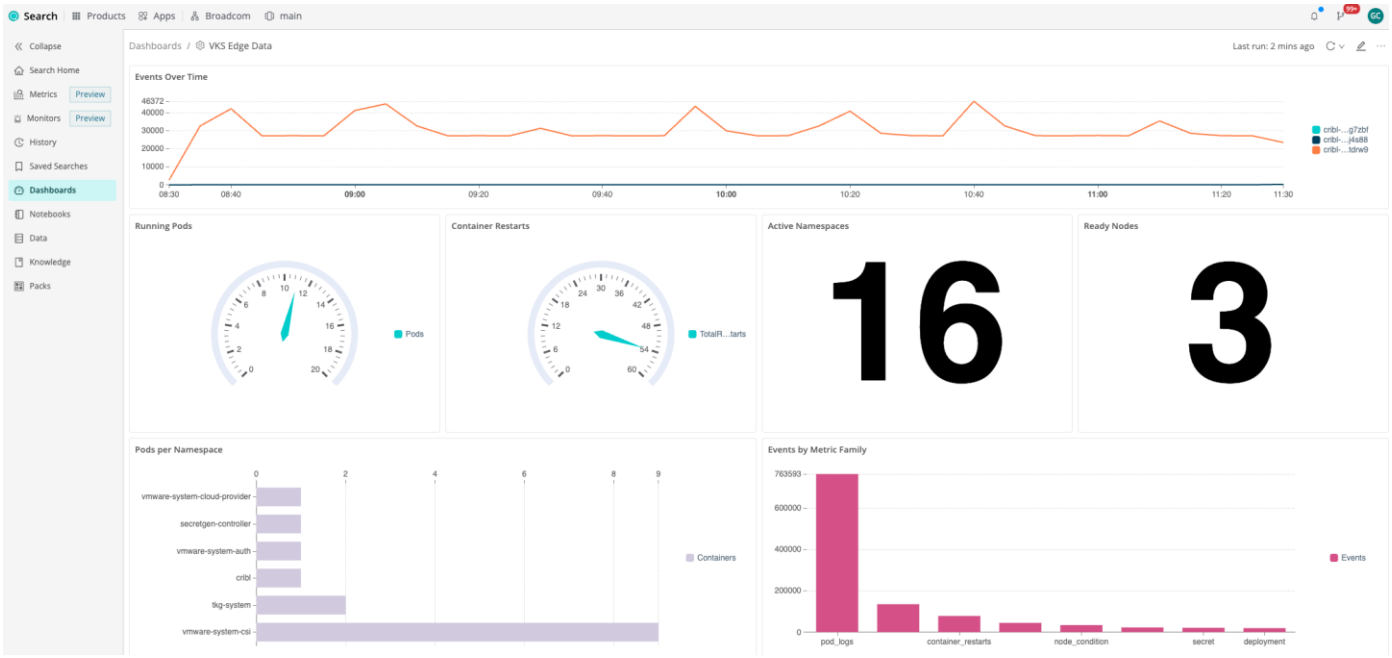


Figure 6: Fleet broadcom-k8s Collect view — three Edge Nodes registered, five sources routed to a Cribl Search destination for indexing and a DevNull tap for diagnostics.

4. Confirm real-time telemetry throughput at the destination

On the All_VKS destination, open the Charts tab to confirm live throughput. The event and byte counters should be climbing and the dropped-events count should remain at zero. In this validation, the destination is sustaining ~58 events per second and ~53 KBps across the three Edge Nodes with no drops.

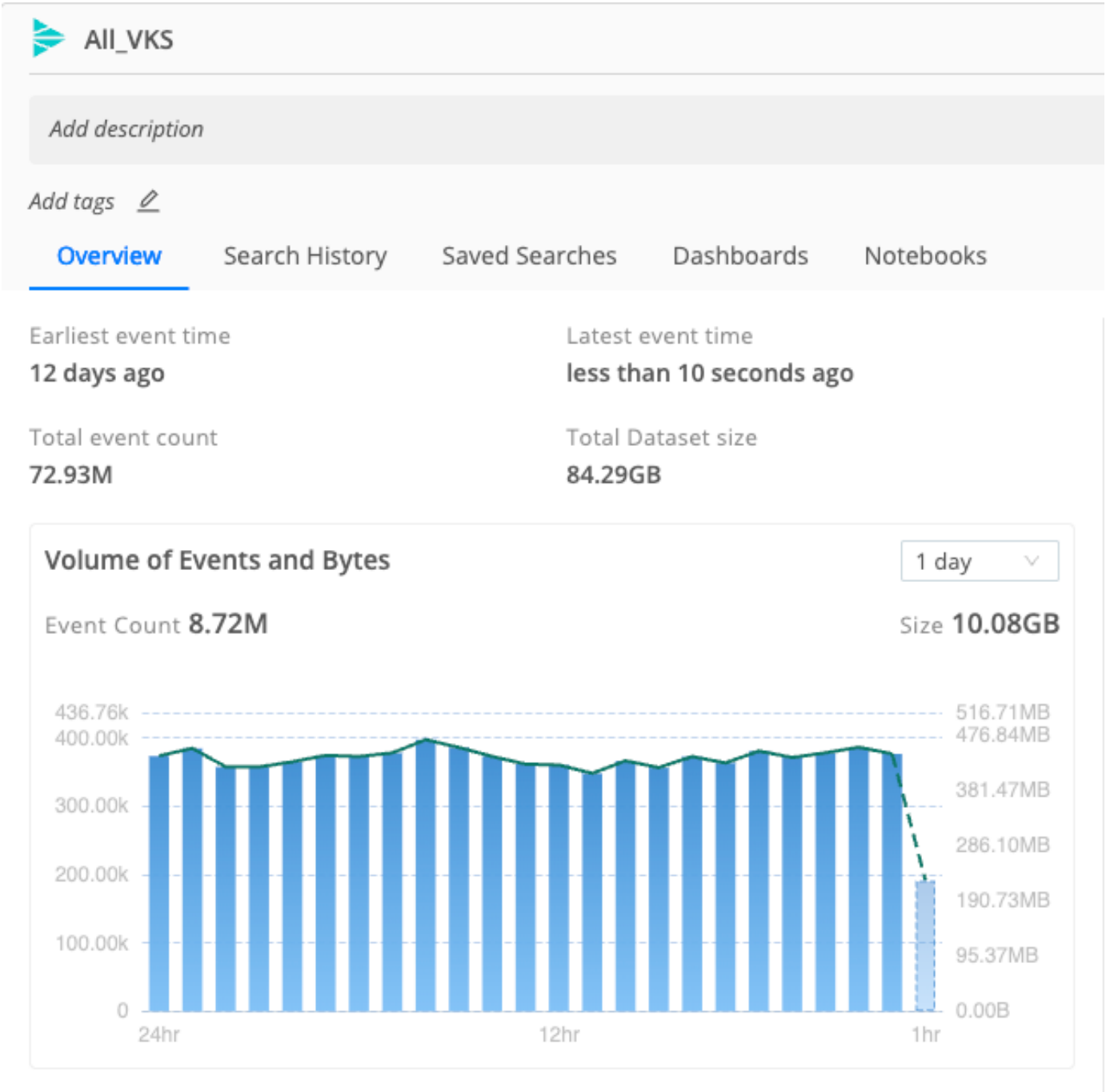


Figure 7: All_VKS destination Charts view — 39.07k total events out, 57.89 eps average throughput, 0 dropped.

5. Verify Kubernetes metadata enrichment on live events

On the All_VKS destination, open the Live Data tab and capture a sample. Each event should carry the expected Kubernetes enrichment fields: kube_namespace, kube_pod, kube_container, kube_node, kube_image, kube_owner_kind, kube_owner_name, and pod annotations and labels. Presence of these fields confirms that

in_kube_logs is successfully querying the local kubelet on each worker node and that the Kubernetes API integration is functioning end to end.

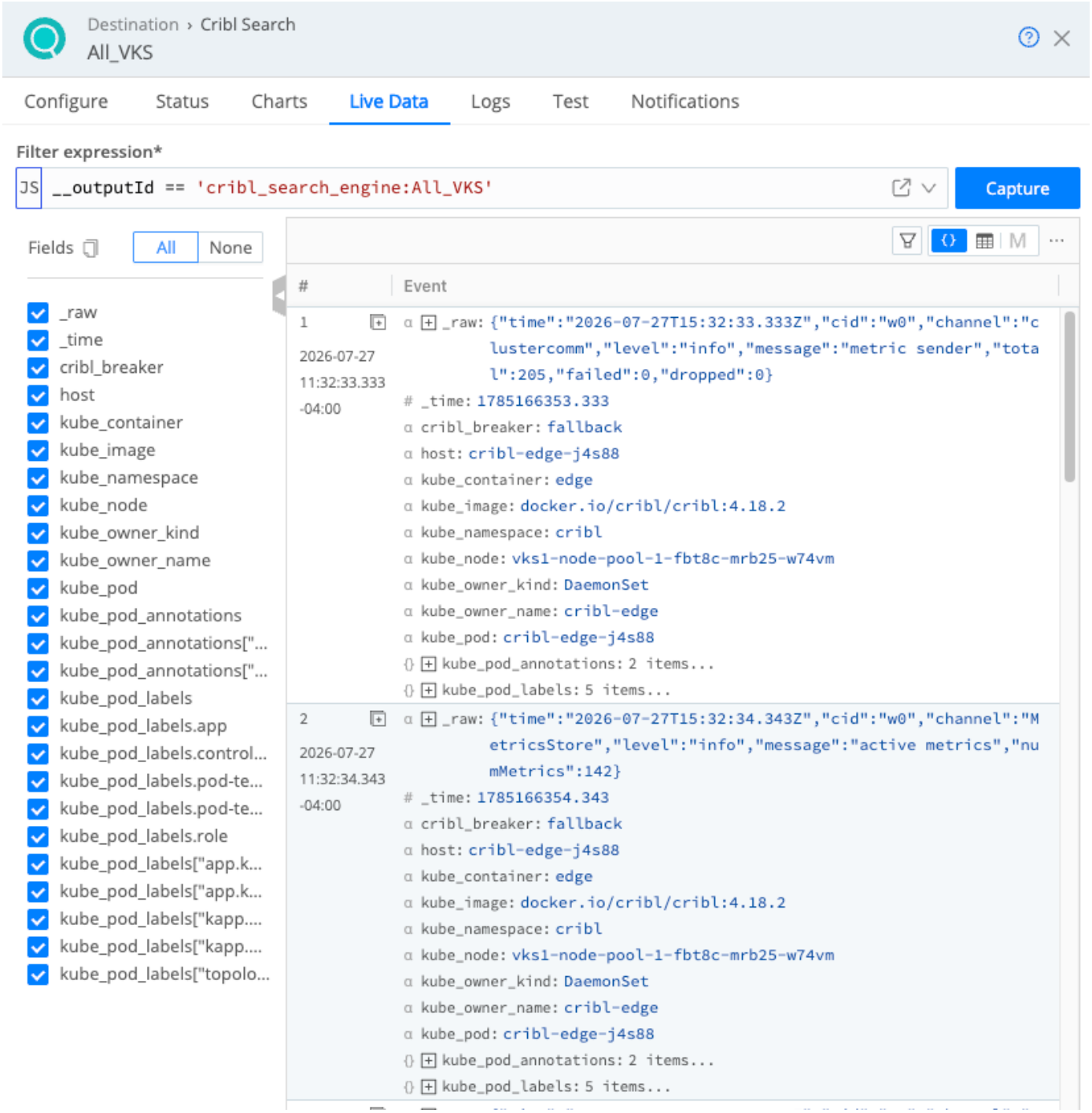


Figure 8: All_VKS destination Live Data view — events include full Kubernetes metadata (kube_node = vks1-node-pool-1-fbt8c-mrb25-w74vm, kube_pod, kube_container, kube_owner_kind = DaemonSet).

6. Confirm data lands in the downstream data store

Open the target dataset (Cribl Lake, Splunk, Amazon S3, or the customer's SIEM) and confirm the volume of events and bytes matches expectations. In this validation, the All_VKS dataset in Cribl Lake shows 72.93M total events and 84.29 GB of stored data over twelve days, with a stable arrival rate of ~8.7M events and ~10 GB per day across the three Edge Nodes.

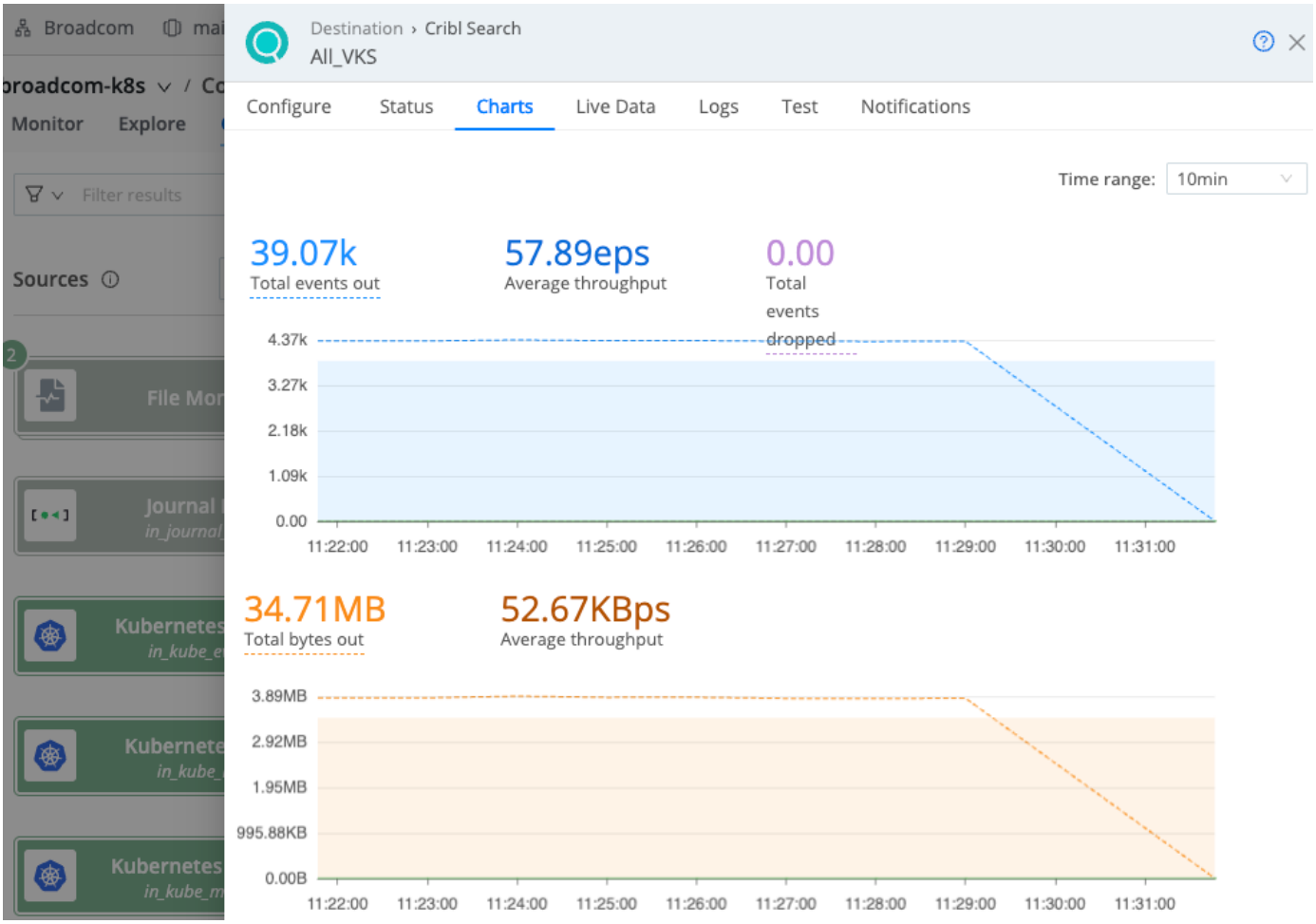


Figure 9: Cribl Lake All_VKS dataset — 72.93 million events, 84.29 GB total, ingesting steadily over the validation window.

7. Validate analytics readiness

The delivered dataset should be immediately usable for operational analytics. In this validation, a Cribl Search dashboard queries the All_VKS dataset in Cribl Lake to render an operational view of the VKS cluster: events over time by node, running pod count, container restart count, active namespaces, ready node count, and pod-per-namespace and metric-family breakdowns. This confirms that the Edge → Cribl.Cloud → Cribl Lake pipeline delivers telemetry fit for downstream dashboarding, alerting, and search without additional transformation.

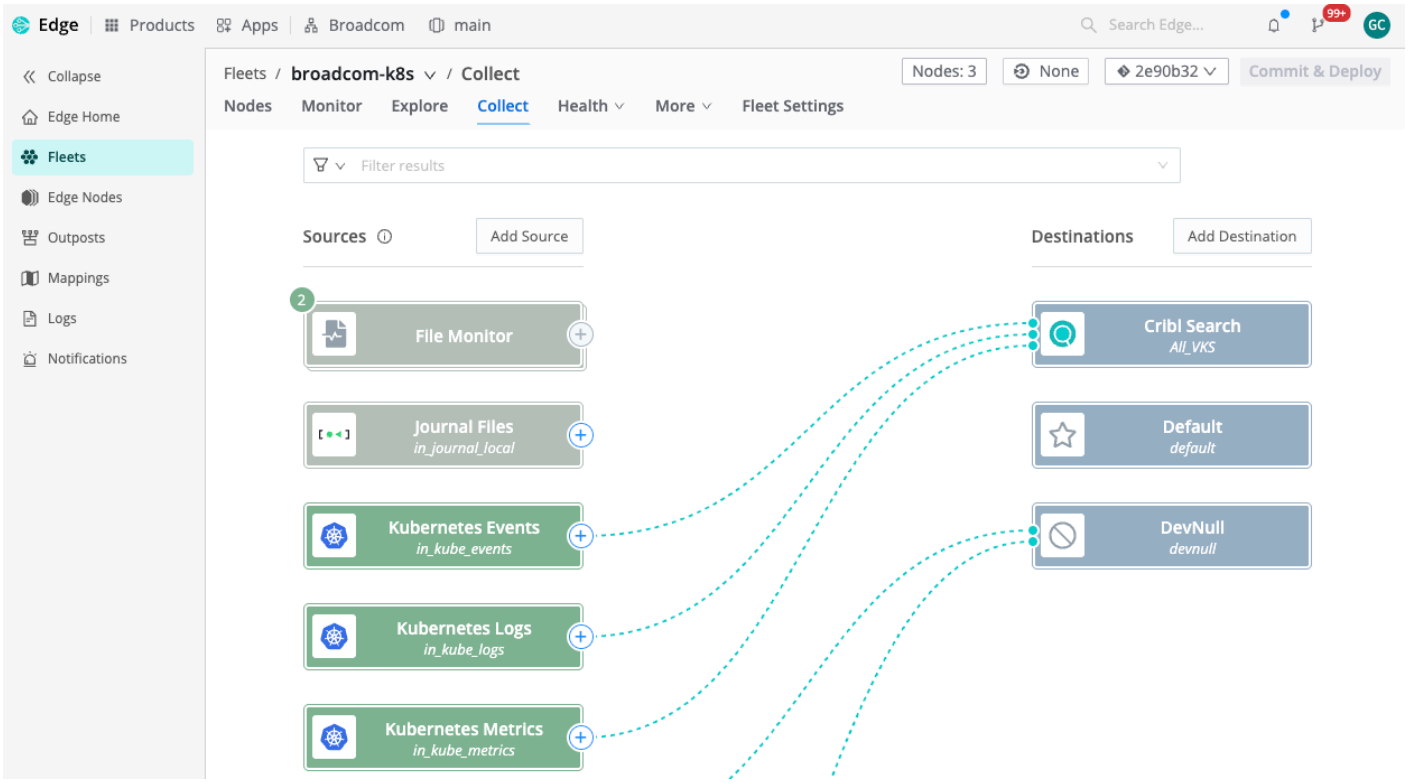


Figure 10: Cribl Search VKS Edge Data dashboard — operational view of the running cluster: 16 active namespaces, 3 ready nodes, running pods and container restarts, and event volume by metric family.

Conclusion

Cribl Edge integrates cleanly with VKS to provide a scalable, low-footprint telemetry collection layer for Kubernetes workloads. The deployment model — DaemonSet per node, streaming to Cribl Cloud — aligns with VKS's operational simplicity: no in-cluster Leader, no persistent volumes, and no per-node configuration once workers register.

This validation confirms that Cribl Edge installs cleanly on VKS 3.6 via the standard Cribl Helm chart, with two VKS-specific accommodations: the namespace PSA privileged label and the CRIBL_K8S_TLS_REJECT_UNAUTHORIZED environment variable. All three primary Kubernetes telemetry sources — logs, events, and metrics — function correctly on VKS's Photon OS and FIPS-mode containerd runtime. The Cribl Cloud to VKS worker control-plane communication operates over standard TLS on port 4200, compatible with VKS's default egress model (subject to NSX or vDefend policy where deployed).

Together, VKS and Cribl Edge give enterprise Kubernetes teams a fully managed, CNCF-compliant runtime paired with a vendor-neutral telemetry pipeline — enabling scale-out Kubernetes adoption without commit-tier growth on downstream observability tools.



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; Jul-26