TECHNICAL WHITE PAPER: July 2026



Coherent Spark on vSphere Kubernetes Service on VMware Cloud Foundation

Reference Architecture

Table of Contents

Executive Summary 3
vSphere Kubernetes Service 3
Coherent Spark 4
Solution Architecture 5
Deployment Methods 5
VKS Cluster Configuration 7
Software Requirements 8
Solution Validation 8
Install Coherent Spark on VKS Cluster 8
Manual Wasm Mount 8
Validate Coherent Spark on VKS Cluster 9
Conclusion 11

Executive Summary

Modern enterprises increasingly rely on business logic that originates in spreadsheets — pricing engines, actuarial models, compliance calculators, and risk scoring rules — yet they need that logic to be governed, versioned, and consumable as an API. Coherent Spark converts that spreadsheet-driven logic into stable, executable services. When those services are deployed on **vSphere Kubernetes Service (VKS)** running on **VMware Cloud Foundation (VCF)**, organizations gain the governance benefits of Spark while keeping execution inside infrastructure they already operate.

This reference architecture describes a validated deployment in which service authoring, versioning, and identity remain in Coherent Spark, and the runtime that executes published services runs as a standard Kubernetes Deployment on VKS. The result is a hybrid pattern that fits enterprise constraints around data residency, network controls, and operational consistency.

The combination of Coherent Spark and VKS delivers three concrete benefits:

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.

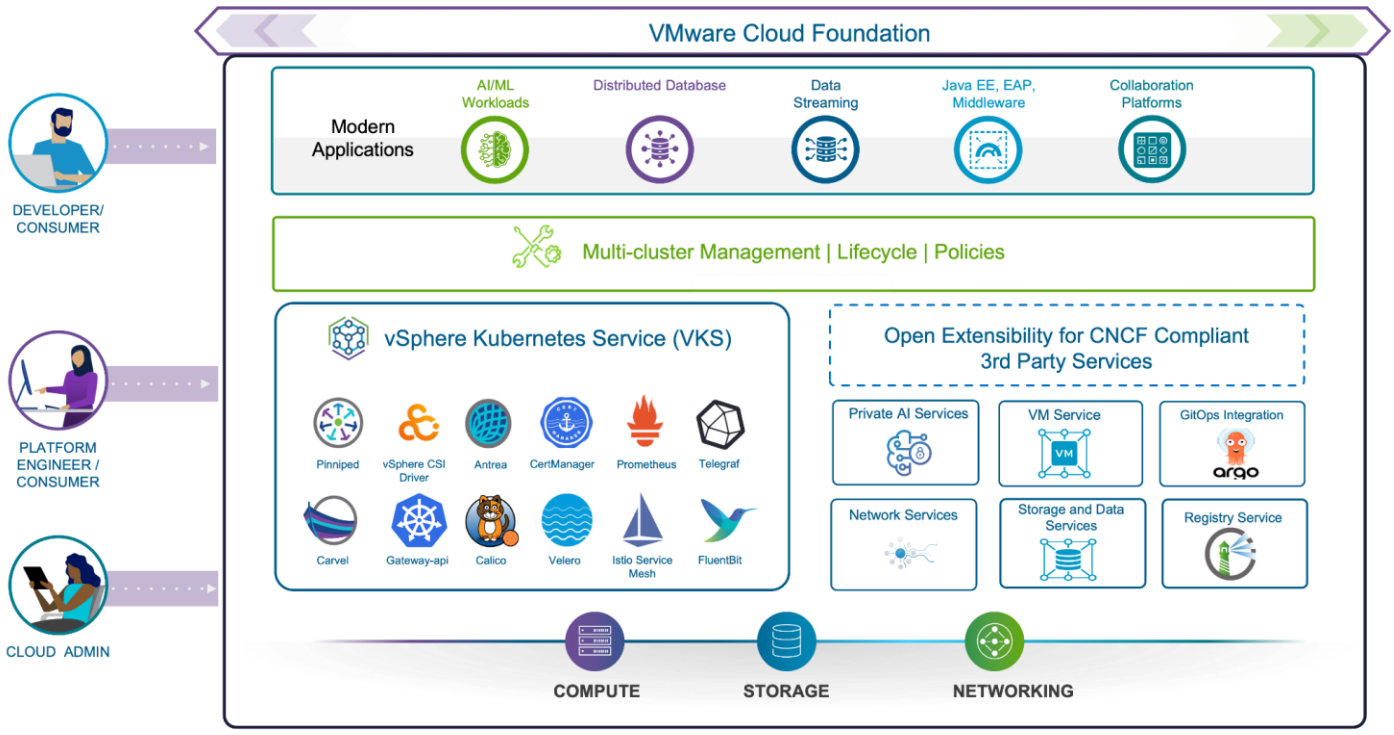


Figure 1: VKS on VCF Ecosystem

Coherent Spark

Coherent Spark is a platform that converts **spreadsheet-driven business logic** into governed, executable services. A workbook is uploaded, inputs and outputs are tagged using named ranges (*Xinput_<name>* and *Xoutput_<name>*), and a versioned service is published. That service is then callable through Spark's execution APIs, allowing business logic that previously lived only inside Excel to be executed consistently by external applications, internal systems, and automation pipelines.

Spark exposes services through two synchronous execution endpoints — a **v3 Execute API** for single-record request/response calls, and a **v4 Execute API** with a flatter JSON structure for synchronous multi-record execution. Around those execution APIs, Spark provides version governance with effective dates, a Testing Center for validating service behavior against testbeds, batch execution patterns for parameter sweeps, and CI/CD promotion flows for moving services between environments. Identity is enforced through tenant-isolated Keycloak realms supporting SSO federation (OIDC, SAML), OAuth 2.0 client credentials, API keys, and ACL-based authorization over Spark resources.

The Spark workflow on Kubernetes follows five stages:

1. **Author:** a workbook is created and tagged with Xinput_* / Xoutput_* named ranges in Coherent Spark.
2. **Publish:** Spark publishes a versioned service and compiles the workbook logic into a **WebAssembly (Wasm)** module.
3. **Distribute:** the Hybrid Runner running on VKS pulls the Wasm module from Spark SaaS (or loads it from a mounted volume or pre-baked image).
4. **Expose:** Kubernetes networking exposes Spark-compatible execution endpoints (/healthcheck, /{tenant}/api/v3/execute, v4-compatible paths) through a Service and Ingress on VKS.
5. **Invoke:** client applications call those endpoints over HTTPS, passing input payloads and receiving computed outputs.

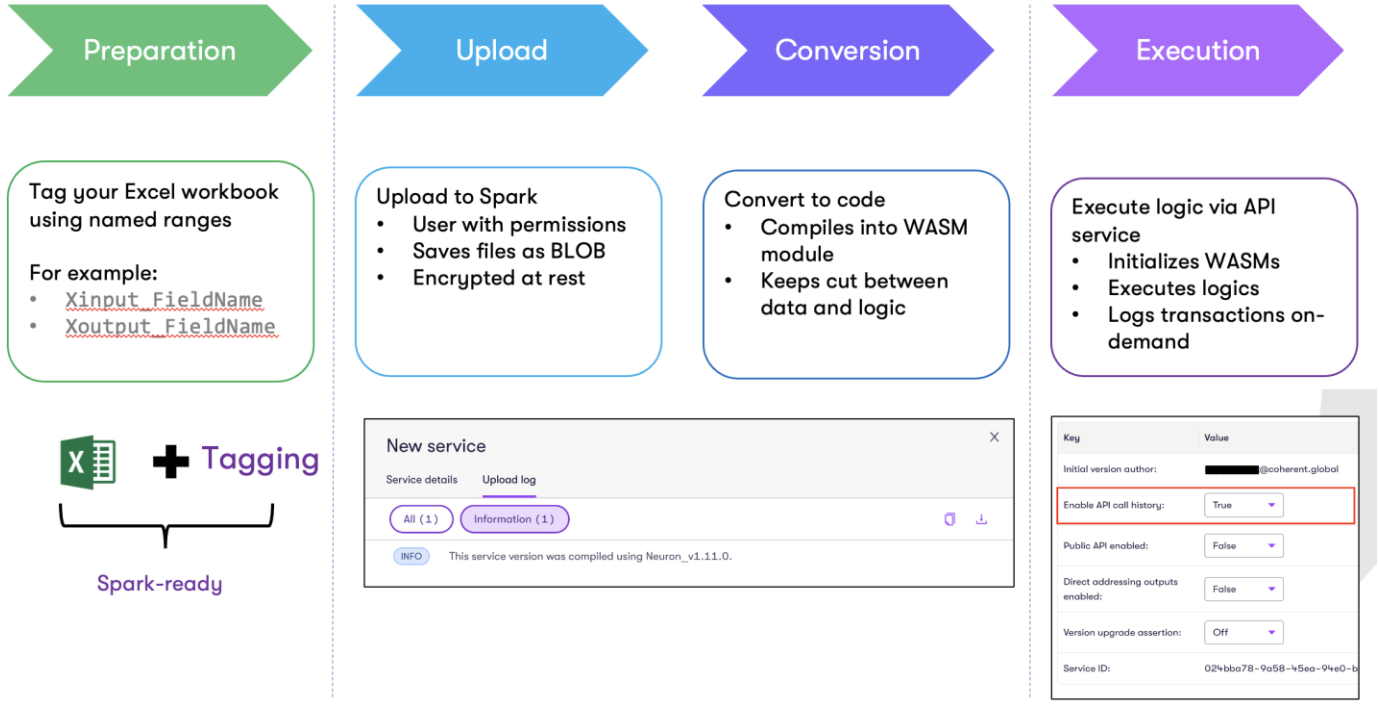


Figure 2: Coherent Spark Workflow

Solution Architecture

Deployment Methods

Coherent Spark supports three deployment models, each addressing a different operational and regulatory profile.

SaaS: Service authoring, identity, publication, and execution are all hosted in Coherent's managed platform. Customers consume Spark entirely through HTTPS endpoints, and Spark manages tenant isolation, infrastructure scaling, and platform lifecycle. This is the fastest path to production and the default model for organizations without specific data-residency or network-isolation requirements.

On-Premises: Spark execution runs fully inside customer-owned infrastructure with no dependency on outbound SaaS connectivity at runtime. This model uses Spark's Hybrid Runner container (nodegen-server) configured with USE_SAAS=false, paired with Wasm service modules that are either manually mounted into the container's model directory or baked into a custom Docker image. It fits regulated workloads where execution traffic cannot leave the customer network.

Hybrid: Service authoring, versioning, and governance remain in Spark SaaS, while runtime execution is projected into a customer-controlled environment such as Kubernetes on VMware Cloud Foundation. The Hybrid Runner connects back to Spark SaaS to retrieve compiled Wasm modules on demand and caches them locally, so customers retain a single source of truth for service definitions while keeping execution close to the data and applications that consume it. This is the deployment model validated in this reference architecture.

Across all three models the service contract — *Xinput_* / Xoutput_** named ranges and the v3/v4 Execute API surface — is identical, which means a service authored in Spark SaaS can be moved between deployment models without changes to its callers.

Model 1: SaaS Reference

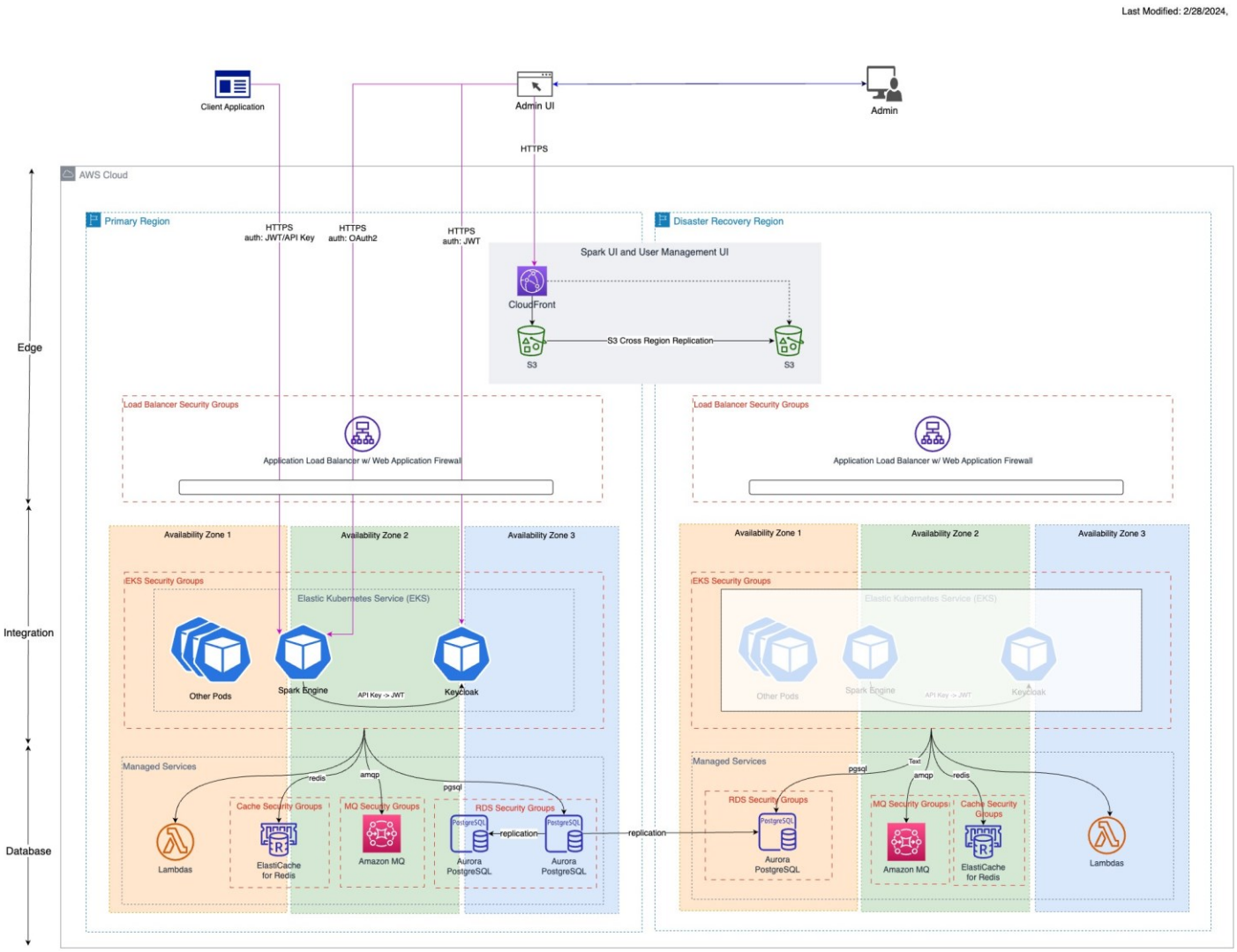


Figure 3.1: Coherent Spark's architecture diagram on AWS

Figure 3.1 illustrates the Coherent Spark AWS reference architecture. The platform is deployed across two AWS regions, with Region 1 operating as the active production region and Region 2 serving as the disaster recovery region. Public API traffic is secured through AWS WAF and routed through an Application Load Balancer into Amazon EKS, where Coherent application services, Spark runtime components, and authentication services are deployed.

The application layer relies on AWS managed services for persistence, caching, messaging, and asynchronous processing. Amazon Aurora PostgreSQL provides relational data storage, Amazon ElastiCache for Redis supports low-latency caching, Amazon MQ provides message broker functionality, and AWS Lambda supports event-driven workloads.

Administrative access is handled separately through CloudFront and Amazon S3, where the Admin UI and user management interface are hosted and distributed. Static assets are replicated to the secondary region using S3 cross-region replication.

The disaster recovery region maintains replicated data services and standby application infrastructure. Under normal operations, the secondary region's EKS workloads remain inactive, but the region can be promoted during a failover event to restore platform availability.

This SaaS reference describes the managed control plane that authors, versions, and governs Spark services. The hybrid deployment described next integrates with this control plane to retrieve compiled service artifacts while keeping execution inside the customer's VKS cluster.

Model 2: Hybrid Reference on VKS / VCF

In the hybrid deployment model, Coherent Spark SaaS continues to provide service authoring, identity, version governance, and Wasm artifact distribution. The runtime that executes those services is the nodegen-server Hybrid Runner, deployed as a standard Kubernetes Deployment on a VKS cluster running on VCF.

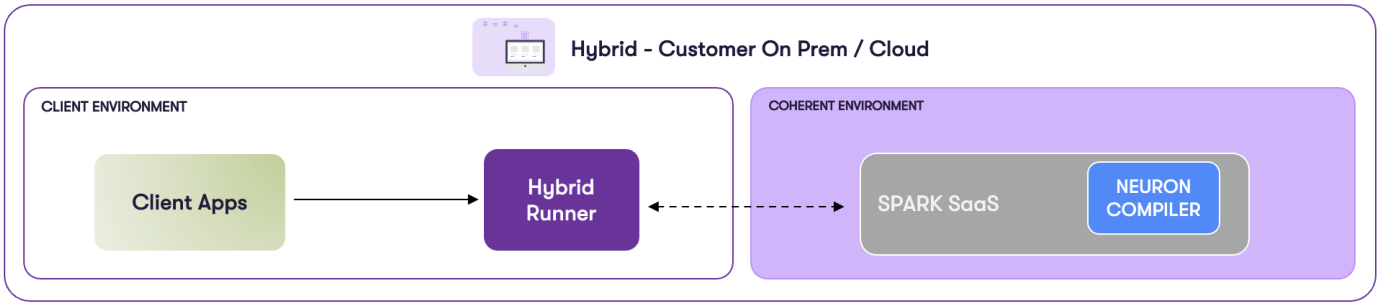


Figure 3.2: Coherent Spark Hybrid Reference Architecture on VKS / VCF

A typical request flows as follows:

1. A client application calls https://<spark-ingress-host>/<tenant>/api/v3/execute with an input payload.
2. The VKS Ingress terminates TLS and routes the request to the Spark Service.
3. The Service load-balances to one of the nodegen-server pods.
4. The pod resolves the requested service version, loading the Wasm module from its local cache, from a mounted PVC, or from the container image (depending on the chosen distribution method).
5. If the requested Wasm module is not present and automatic pull is enabled, the pod authenticates to Spark SaaS using a synthetic key stored in a Kubernetes Secret and downloads the module.
6. The pod executes the Wasm module against the input payload and returns the computed output to the client.

The solution validation below focuses on the hybrid model use case where Wasm models were exported from SaaS and imported into on-prem for private consumption.

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 the *best-effort-large* profile, while control plane nodes utilized the *best-effort-medium* profile.

Table 1: Kubernetes Cluster Configuration Summary

Component	Configuration	Notes
Kubernetes Control Plane	3 Replicas — 2x vCPU 8GB RAM vSAN RAID 5	VM Class: “best-effort-medium” Storage Class: “vsan-esa-default-policy-raid5”
Kubernetes Worker Nodes	3 Replicas — 4x vCPU	VM Class: “best-effort-large” Storage Class: “vsan-esa-default-policy-raid5”

	16GB RAM vSAN RAID 5	
Kubernetes Release	v1.35.5	
OS Image	Ubuntu 24.04	Kernel version 6.8.0-90-generic

An example VKS cluster configuration is defined declaratively in vks-cluster.yaml on Github. 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. For more information, refer to Workflow for Provisioning VKS Clusters Using kubectl.

Software Requirements

Listed below is a set of software components required to successfully deploy Coherent Spark Hybrid Runner on VKS clusters. The software versions are current as of the validation testing and may not reflect the latest update.

The Coherent Spark Hybrid Runner is delivered as an OCI image from ghcr.io/coherent-partners/nodegen-server. It is deployed using standard Kubernetes resources — a Deployment, a Service, a Secret for the SaaS synthetic key, and optionally a PersistentVolumeClaim for mounted Wasm models — and does not require a Helm chart or custom resource definitions.

Table 2: Software Requirements

Software	Version	Notes
VKS	3.6.2	
vSAN File Service	9.0.0.0.24755229	provide ReadWriteMany volume support
Coherent Spark Hybrid Runner (nodegen-server)	1.54.1	ISV application package: WebAssembly runtime that executes Spark-compiled service modules behind Spark-compatible v3 / v4 Execute endpoints.

Solution Validation

Install Coherent Spark on VKS Cluster

Coherent Spark supports three deployment patterns on Kubernetes. The recommended default is the automatic Wasm pull from Spark SaaS, which keeps the cluster image immutable while letting Spark serve as the single source of truth for service artifacts. The manual-mount and custom-image patterns are documented as variants for restricted-network or CI/CD-driven environments.

For illustration purposes, the example below simulates the manual-mount option in an air-gapped environment. It uses a dedicated namespace *coherent-spark*. The cluster context is assumed to be the VKS cluster *coherent-c1*.

Manual Wasm Mount

The manual-mount deployment pattern fits restricted environments that cannot reach Spark SaaS at runtime. In this deployment mode the Wasm artifacts are exported from Spark and mounted into the runtime's /models directory via a ReadWriteMany (RWX) persistent volume (PV). If RWX volume is provisioned on VMware vSAN, *vSAN File Service* must be enabled with the Supervisor. Refer to Broadcom documentation for more information.

Download hybrid-runner-method2.yaml from Github, apply the manifest and wait for rollout of the hybrid runner deployment to complete:

```
kubectl apply -f hybrid-runner-method2.yaml

kubectl rollout status deployment/hybrid-runner
```

This step assumes you have exported the Wasm modules according to the Export the Wasm modules from Spark instructions. The commands below will copy your tenant directory into */models* in one of the hybrid runner pods, restart the deployment for the pods to detect the imported modules, and wait for the deployment rollout to complete.

```
kubectl cp ./<my-tenant> coherent-spark/hybrid-runner-xxxxxxxxxx-xxxxx:/models/

kubectl rollout restart deployment/hybrid-runner

kubectl rollout status deployment/hybrid-runner
```

Validate Coherent Spark on VKS Cluster

Validation confirms four properties of the deployed runtime: the workload is healthy, the cluster network exposes it correctly, the Spark execution APIs respond as documented, and Wasm artifacts resolve correctly from the chosen distribution source.

Step 1 — Confirm pods, service, and deployment are healthy

```
$ kubectl get all -n coherent-spark
NAME                              READY   STATUS    RESTARTS   AGE
pod/hybrid-runner-5dbfdd8b65-jpm29   1/1     Running   0          28s
pod/hybrid-runner-5dbfdd8b65-ncl9p   1/1     Running   0          40s
pod/hybrid-runner-5dbfdd8b65-q9sp6   1/1     Running   0          28s
pod/hybrid-runner-5dbfdd8b65-wbkg8   1/1     Running   0          40s

NAME                          TYPE          CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
service/hybrid-runner-service   LoadBalancer   10.109.161.189   10.139.8.9    3000:32703/TCP   4m21s

NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/hybrid-runner   4/4     4            4           4m21s

NAME                                    DESIRED   CURRENT   READY   AGE
replicaset.apps/hybrid-runner-5dbfdd8b65   4         4         4       40s
replicaset.apps/hybrid-runner-78d96cd95c   0         0         0       4m21s

NAME                                                  REFERENCE                   TARGETS          MINPODS   MAXPODS
REPLICAS   AGE
horizontalpodautoscaler.autoscaling/hybrid-runner-hpa   Deployment/hybrid-runner   memory: 3%/75%   4         8
4          4m21s
```

All hybrid-runner pods should report Running with 1/1 ready containers. Restart counts above zero on a fresh deployment usually indicate a failed readiness probe — inspect logs first.

Make a note of the external IP reported for the LoadBalancer service. This IP will be used in the subsequent API calls.

Step 2 — Confirm runtime startup in logs

```
$ kubectl logs deployment/hybrid-runner -n coherent-spark --tail=20
...
{"Level":"INFO",...,"EventType":"WorkerManager.Load","TextMessage":"Loading model with UUID 287e4d67-ad8b-4078-9ebc-094db838c038 for \"my-tenant\" tenant.","JSONPayload":"{\"tenant\":\"my-tenant\",\"versionId\":\"287e4d67-ad8b-4078-9ebc-094db838c038\",\"modelPath\":\"/storage/models/my-
```

```
tenant/finance/simple-interest/0.3.1/287e4d67-ad8b-4078-9ebc-
094db838c038.zip\",\"size\":1}","SystemIP":"172.17.0.2","TimeStamp":"2026-03-02T22:06:05.857Z"}
{"Level":"INFO",...,"EventType":"WorkerManager.Load","TextMessage":"Starting model
initialization","JSONPayload":"{\"tenant\":\"my-tenant\",\"versionId\":\"287e4d67-ad8b-4078-9ebc-
094db838c038\"}","SystemIP":"172.17.0.2","TimeStamp":"2026-03-02T22:06:05.857Z"}
```

Step 3 — Confirm platform health endpoint

Retrieve the external IP reported in the LoadBalancer service in Step 1 and perform a health check that makes sure it returns *{"msg":"ok"}*:

```
$ curl -sS http://<LoadBalancer External IP>:3000/healthcheck
{"msg":"ok"}
```

Step 4 — Execute a Spark service

Given an example published Spark service that takes a numeric input principal and rate and returns an output interest, a representative call against the v3 Execute API is:

```
curl -X POST "http://<LoadBalancer External IP>/<tenant>/api/v3/execute" \
  -H "Content-Type: application/json" \
  -d '{
    "request_data": {
      "inputs": {
        "principal": 100000,
        "rate": 0.045
      }
    },
    "request_meta": {
      "service_uri": "folders/finance/services/simple-interest",
      "version": "1.0.0"
    }
  }'

Expected response shape:
{
  "status": "Success",
  "response_data": {
    "outputs": {
      "interest": 4500
    }
  },
  "response_meta": {
    "service_id": "...",
    "version_id": "...",
    "process_time": 12,
    ...
  }
}
```

A successful response with the expected outputs payload proves that the Wasm module was loaded, the input contract (Xinput_*) was honored, and the output contract (Xoutput_*) was satisfied — i.e., the full Spark execution path is functional inside the VKS cluster.

Step 5 — Confirm artifact resolution behavior

For manual WASM mount deployments, the runtime should resolve modules from /models without any outbound SaaS traffic. Verify by inspecting network policy logs or by running with egress to Spark SaaS blocked.

Conclusion

The Coherent Spark and VMware integration is technically credible because the Spark runtime can be deployed as a standard containerized execution service rather than a desktop spreadsheet dependency. Spark provides the authoring, publication, identity, testing, and version-governance layers. vSphere Kubernetes Service on VMware Cloud Foundation provides the customer-controlled execution environment.

The validation confirmed the integration end to end: the nodegen-server Hybrid Runner deployed cleanly on a VKS cluster with best-effort-large worker nodes and vSAN ESA RAID-5-backed storage, /healthcheck returned a healthy response through the Kubernetes LoadBalancer service, a published Spark service executed successfully through /<tenant>/api/v3/execute with the expected output contract, and the Wasm artifact resolution behavior operated as documented.

For a reference architecture, the meaningful conclusion is that Coherent Spark logic can be operationalized on Kubernetes in a form that is portable, testable, securable, and compatible with VMware-hosted container platforms. Service governance stays centralized in Spark SaaS where it benefits from a managed lifecycle, while execution sits inside the customer's VKS estate where it benefits from existing networking, policy, observability, and cluster operations. The pattern is directly applicable to regulated workloads in financial services, insurance, healthcare, and the public sector — domains where spreadsheet-authored business logic is common and where the operational benefits of running that logic inside VMware Cloud Foundation are most pronounced.



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; Aug-26