Reference Architecture: August 2026

Tibco Platform on vSphere Kubernetes Service on VMware Cloud Foundation

Reference Architecture

Table of Content

Executive Summary

This document describes how TIBCO Platform can be deployed and validated on VMware vSphere Kubernetes Service (VKS) running as part of VMware Cloud Foundation (VCF). It brings together two complementary enterprise technologies: VKS, which provides a CNCF-certified Kubernetes runtime integrated with vSphere operations and lifecycle management, and TIBCO Platform, which enables real-time integration, messaging, event processing, process automation, data grid services, observability, and developer productivity across hybrid environments.

The solution demonstrates how organizations can modernize integration and application delivery while continuing to use familiar VMware infrastructure, tools, operational processes, and security controls. By hosting TIBCO Platform on VKS, platform and cloud operations teams can reduce operational complexity, standardize Kubernetes lifecycle management, and provide a consistent foundation for modern application services.

The example architecture focuses on a representative integration hub deployment that includes API and messaging capabilities, TIBCO Developer Hub, PostgreSQL, Traefik ingress, observability services, and supporting platform components. It also identifies key infrastructure considerations such as DNS and certificate requirements, database configuration, ingress compatibility, default network ports, and minimum VKS sizing. The design can support environments with public network access as well as air-gapped deployment models where required.

From a business perspective, the combined solution helps enterprises connect fragmented systems and data sources, enable real-time business responsiveness, accelerate modernization initiatives, and improve governance and observability. It supports scalable Kubernetes operations while allowing teams to build, deploy, and manage integration workloads using established TIBCO capabilities and a VMware-managed Kubernetes foundation.

The validation section documents the deployment approach used to prove the solution in a non-production environment, including ingress setup, control plane installation, supporting services, capability deployment, and initial operational checks. The result is a practical reference architecture and implementation guide for customers, partners, and field teams evaluating TIBCO Platform on VKS as part of a broader modern application platform strategy.

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.

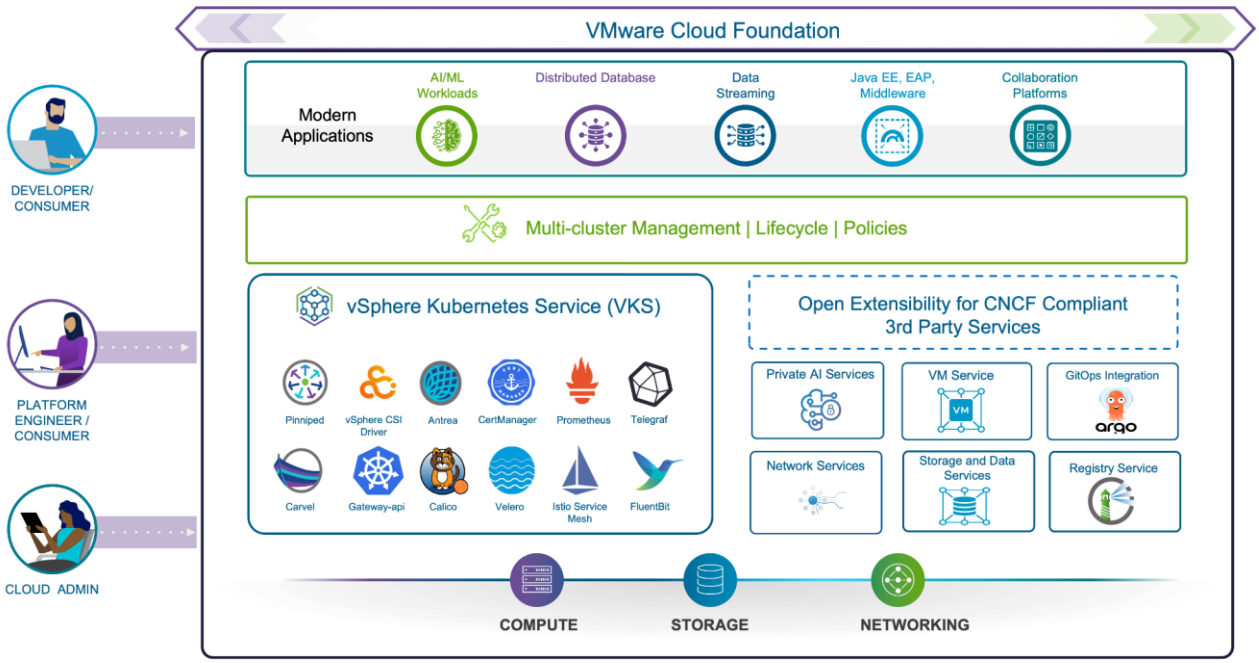


Figure 1: VKS on VCF Ecosystem

Benefits of using VKS

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.

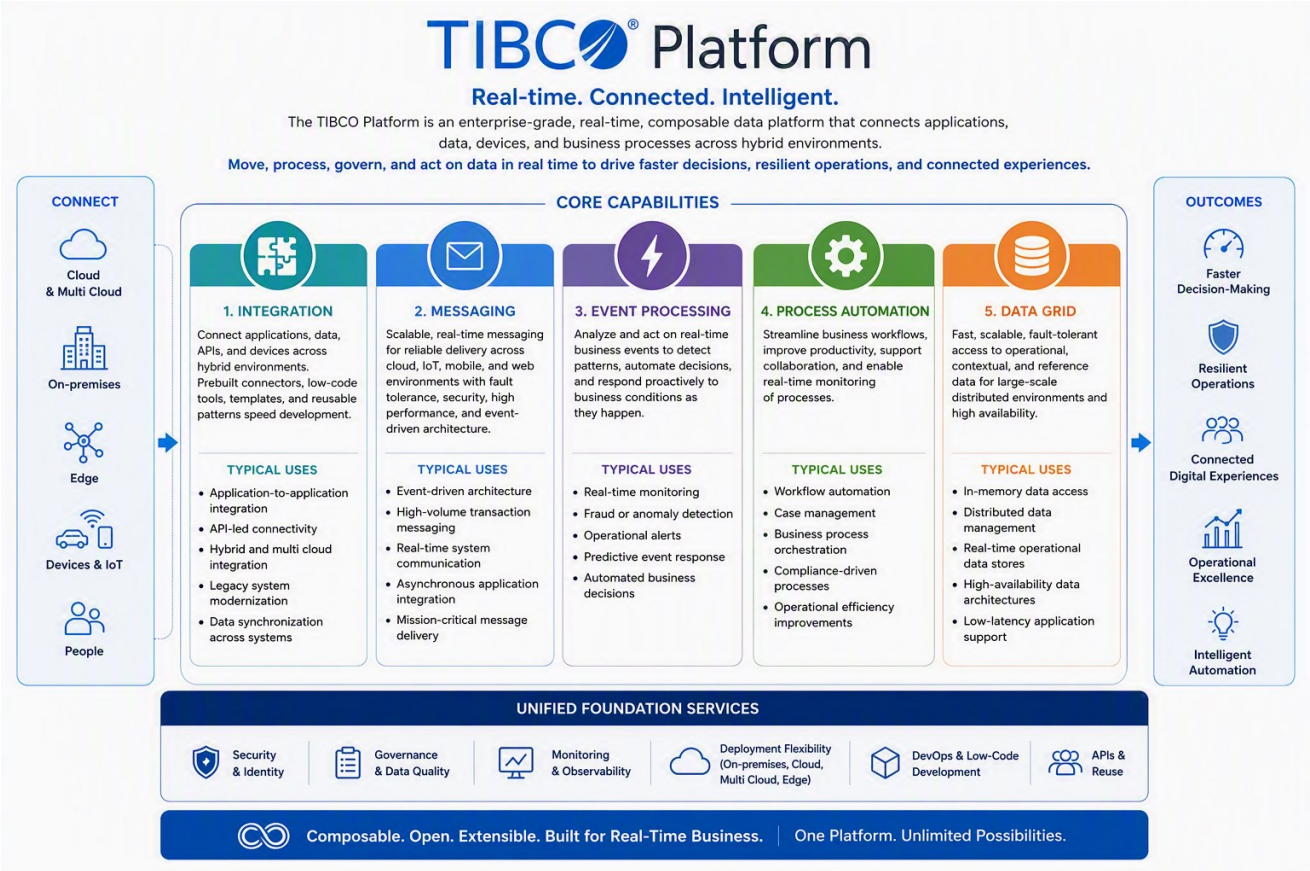
Operational Simplicity: VKS is engineered for unparalleled operational simplicity, leveraging the familiarity of existing vSphere tools, skills, and workflows. This design philosophy significantly reduces the learning curve for IT teams and streamlines management processes. With VKS, organizations benefit 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.

TIBCO Platform

The **TIBCO Platform** is an enterprise-grade, real-time, composable data platform designed to connect applications, data, devices, and business processes across hybrid environments, including on-premises, cloud, multi cloud, and edge. TIBCO positions it as a unified platform for real-time operations, integration, monitoring, and intelligent automation.

At its core, TIBCO helps organizations **move, process, govern, and act on data in real time**, enabling faster decision-making, more resilient operations, and connected digital experiences.



Core Capabilities

1. Integration

TIBCO enables organizations to connect applications, data sources, APIs, and devices wherever they are hosted. Its integration capabilities support real-time connectivity across legacy systems, cloud applications, edge environments, and hybrid architectures. TIBCO also highlights prebuilt connectors, low-code tooling, templates, and reusable patterns to speed development.

Typical uses include:

- Application-to-application integration
- API-led connectivity
- Hybrid and multi cloud integration
- Legacy system modernization
- Data synchronization across systems

2. Messaging

The platform supports scalable, real-time messaging for complex enterprise integrations. TIBCO describes its messaging capabilities as supporting reliable message delivery across cloud, IoT, mobile, and web environments, with fault tolerance, security, high performance, and event-driven architecture.

Typical uses include:

- Event-driven architecture
- High-volume transaction messaging
- Real-time system communication
- Asynchronous application integration
- Mission-critical message delivery

3. Event Processing

TIBCO provides event processing capabilities that allow organizations to analyze and act on real-time business events. The platform is designed to detect patterns, automate decisions, and respond proactively to business conditions as they happen.

Typical uses include:

- Real-time monitoring
- Fraud or anomaly detection
- Operational alerts
- Predictive event response
- Automated business decisions

4. Process Automation

TIBCO includes process automation capabilities for streamlining business workflows, improving productivity, supporting collaboration, and enabling real-time monitoring of processes.

Typical uses include:

- Workflow automation

- Case management
- Business process orchestration
- Compliance-driven processes
- Operational efficiency improvements

5. Data Grid

The platform includes data grid capabilities for fast, scalable, fault-tolerant access to operational, contextual, and reference data. TIBCO describes this as supporting large-scale distributed environments and high availability.

Typical uses include:

- In-memory data access
- Distributed data management
- Real-time operational data stores
- High-availability data architectures
- Low-latency application support

Platform Management and Developer Experience

TIBCO Platform includes a control plane intended to simplify management, monitoring, and observability across TIBCO components running on premises, in the cloud, or at the edge. TIBCO says this provides a “single pane of glass” for operators, developers, and business technologists.

For developers, TIBCO highlights the **TIBCO Developer Hub** and **TIBCO Flogo Extension for VS Code**, which support discovery of APIs, apps, and documentation, visual relationship mapping between assets, project templates, and local development, debugging, testing, and running of Flogo applications.

AI and Intelligent Automation

Real-time and AI-enabled, with capabilities including AI-driven insights, AI copilots, zero-code agents, AI accelerators, an AI-powered mapper, and MCP server support. These are positioned as ways to help teams build intelligent applications, accelerate development, and turn operational data into actionable intelligence.

Business Value

At a high level, TIBCO Platform is designed to help enterprises:

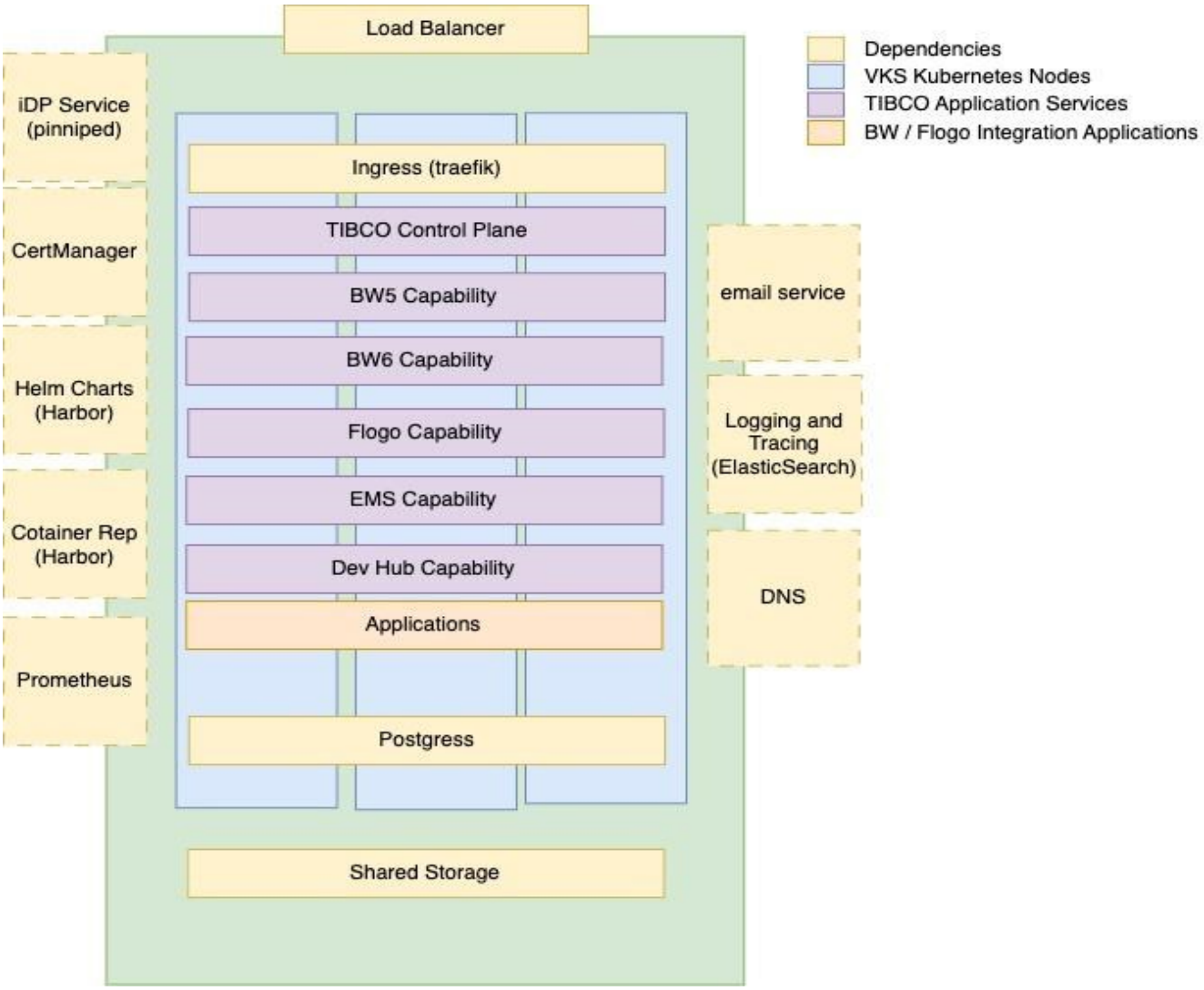
- Connect fragmented systems and data sources
- Enable real-time business responsiveness
- Modernize integration architecture
- Reduce operational complexity
- Improve observability and governance
- Automate workflows and decisions
- Build scalable, event-driven applications
- Support hybrid, multi cloud, and edge deployment models

Solution Architecture

There are several options when setting up TIBCO Platform on VCF, which vary depending on the use case that you are following.

In this solution example is based around a typical integration hub example with both API and Messaging capabilities deployed, supported by TIBCO Developer HUB.

This example can be deployed in either a network connected with ability to access public networks or can be deployed in a complete air gaped network. For more details on how to setup TIBCO Platform in an air gapped setup please refer to TIBCO Platform Air-gapped workshop.



This table details the software components and their version used in this architecture.

Software Component	Version	Vendor
TIBCO Control Plane Local Hosted	1.17	TIBCO
TIBCO Data Plane	1.17	TIBCO
TIBCO Flogo	2.6.2	TIBCO

TIBCO Businessworks 5	5.16.1	TIBCO
TIBCO Businessworks 6	6.12.0 HF02	TIBCO
TIBCO Developer Hub	1.17	TIBCO
TIBCO EMS	EMS 10.5.0 FTL 7.2.0	TIBCO
Postgres Database	16	Bitnami Image
Treafik Ingress	3.3.4	3rd Party

DNS Requirements

The TIBCO Platform Control Plane requires a resolvable domain name. The domain does not need to be public, but to simplify testing, it should support certificate registration from a well-known certificate authority.

Control Plane DNS Requirements

Control Plane Instance ID	cp1
Subscription Name	examplesubpoc
DNS Names for Subscription	example.com
Admin	admin.cp.[example.com]
Control Plane UI/Router	controlplane.cp.[example.com]
Control Plane Tunnel	controlplanepoc.tunnel.[example.com]
Certificates from Well Known or Enterprise Cas	
Admin	admin.cp.[example.com]
Control Plane UI/Router	controlplanepoc.cp.[example.com]
Control Plane Tunnel	controlplanepoc.tunnel.[example.com]
Observability	
Kibana	https://kibana.tp.[example.com]
Elasticsearch	https://elastic.tp.[example.com]
Prometheus	https://prometheus-internal.tp.[example.com]
Grafana	https://grafana.tp.[example.com]
Data Plane In addition to the applications that you create and deploy on the platform, the capabilities provide APIs for systems management and testing purposes. You can use any ingress you want, provided it is accessible, though it is recommended to define the different capabilities explicitly.	
Flogo	https://flogo.tp.[example.com]
BW5CE	https://bw5ce.tp.[example.com]
BW6CE	https://bw6ce.tp.[example.com]
Dev Hub	https://devhub.tp.[example.com]

Database Requirements

The control plane uses PostgreSQL 16 to store runtime and metadata about the control plane.

You must enable the extension uuid-ossp in PostgreSQL before installing TIBCO Control Plane.

You can either use helm to configure the database or if manageDbSchema in Helm chart values is set to false, the chart does not manage database schemas. You must use the script provided in GitHub repository to create, upgrade, and delete databases externally. For more information, see PostgreSQL Database Management Script.

Ingress Controller Compatibility

For TIBCO Control Plane, the following ingress controllers are tested on VKS.

Ingress Controller	Highest Version certified
Traefik	3.3.4

Default Ports Used in TIBCO Control Plane

The following table lists the default ports used in TIBCO Control Plane.

Component	Service	**Direction	Default Port
TIBCO Control Plane	Ingress	Inbound	443
TIBCO Control Plane	Postgres	Outbound	5432
TIBCO Control Plane	SMTP	Outbound	25
TIBCO Control Plane / Data Plane	Container Registry	Outbound	443
TIBCO Control Plane	Identity Provider	Outbound	443
Data Plane	Helm Repository	Outbound	443
Data Plane	Activation Service	Outbound	7070
Data Plane	tibtunnel	Outbound	443
Data Plane	CP Proxy	Outbound	443

*** Direction is in reference to TIBCO Control Plane deployment*

Minimum Platform VKS Sizing

You need a minimum of 3 worker nodes to provide a minimum quorum for the messaging components in the platform. At a minimum 30 GB of disk space is required per node for image storage. For the deployed applications an additional 100GB of shared storage is required.

Characteristic	Value
Nodes	3
vCPUs	4
Memory	16 GiB
Memory vCPU	4 GiB (per vCPU)
Local disk storage	30GB (per node)
Shared storage	100GB

Solution Validation

The following steps were used to validate the TIBCO Platform on VKS and should not be used for a production environment.

TIBCO customers who want to install TIBCO Platform on VKS should contact their account representative. If you are not currently a TIBCO customer and would like to trial the software, contact tibco-sales@tibco.com.

Setup Traefik Ingress Controller

To setup up the Traefik ingress controller create the following values file.

```
# Configure Network Ports and EntryPoints
# EntryPoints are the network listeners for incoming traffic.
ports:
  # Defines the HTTP entry point named 'web'
  web:
    port: 80
    nodePort: 30000
    # Instructs this entry point to redirect all traffic to the 'websecure' entry point
    http:
      redirections:
        entryPoint:
          to: websecure
          scheme: https
          permanent: true

  # Defines the HTTPS entry point named 'websecure'
  websecure:
    port: 443
    nodePort: 30001

# Enables the dashboard in Secure Mode
api:
  dashboard: true
  insecure: false

ingressRoute:
  dashboard:
    enabled: true
    matchRule: Host(`dashboard.tp.tibco.showcase.tmm.broadcom.lab`)
    entryPoints:
      - websecure
    middlewares:
      - name: dashboard-auth

# Creates a BasicAuth Middleware and Secret for the Dashboard Security
extraObjects:
  - apiVersion: v1
    kind: Secret
    metadata:
      name: dashboard-auth-secret
    type: kubernetes.io/basic-auth
    stringData:
      username: admin
```

```
    password: "P@ssw0rd"     # Replace with an Actual Password
  - apiVersion: traefik.io/v1alpha1
    kind: Middleware
    metadata:
      name: dashboard-auth
    spec:
      basicAuth:
        secret: dashboard-auth-secret

# We will not route with Gateway API..  For first testing..  may changed to gateway...
ingressClass:
  enabled: true

# Enable Gateway API Provider & Disables the KubernetesIngress provider
# Providers tell Traefik where to find routing configuration.
providers:
  kubernetesIngress:
    enabled: true
  kubernetesGateway:
    enabled: false

## Gateway Listeners
gateway:
  listeners:
    web:          # HTTP listener that matches entryPoint `web`
      port: 80
      protocol: HTTP
      namespacePolicy:
        from: All

    websecure:        # HTTPS listener that matches entryPoint `websecure`
      port: 443
      protocol: HTTPS  # TLS terminates inside Traefik
      namespacePolicy:
        from: All
      mode: Terminate
      certificateRefs:
        - kind: Secret
          name: local-selfsigned-tls  # the Secret we created before the installation
          group: ""

# Enable Observability
log:
  level: INFO
# This enables access logs, outputting them to Traefik's standard output by default. The [Access Logs
Documentation](https://doc.traefik.io/traefik/observability/access-logs/) covers formatting, filtering, and output options.
accessLog:
  enabled: true

# Enables Prometheus for Metrics
```

```
metrics:
  prometheus:
    enabled: true

additionalArguments:
    - "--serversTransport.insecureSkipVerify=true"
```

Run the following commands to install the Traefik ingress controller.

```
helm repo add traefik https://traefik.github.io/charts
helm repo update
kubectl create namespace traefik

# 1) Generate a self-signed certificate valid for *.tp.tibco.showcase.tmm.broadcom.lab
openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
  -keyout tls.key -out tls.crt \
  -subj "/CN=*.tp.tibco.showcase.tmm.broadcom.lab"

# 2) Create the TLS secret in the traefik namespace
kubectl create secret tls local-selfsigned-tls \
  --cert=tls.crt --key=tls.key \
  --namespace traefik

helm install traefik traefik/traefik \
  --namespace traefik \
  --values values.yaml
```

Installing the Control plane

Setup the following environment variables

```
## Namespace
export TP_INGRESS_NAMESPACE=ingress-system ## Ingress System Namespace
export ELASTIC_NAMESPACE=elastic-system ## Elastic System Namespace
export PROMETHEUS_NAMESPACE=prometheus-system ## Prometheus System Namespace
export TP_EXT_NAMESPACE=tibco-ext ## External Applciation Namespace
export TP_POSTGRES_NAMESPACE=tibco-ext
export STORAGE_NS=storage-system ## Storage System Namespace

## Storage
export RWO_STORAGE_CLASS=vsan-esa-default-policy-raid5  ## Disk Storage Class
export RWX_STORAGE_CLASS=vsan-esa-default-policy-raid5 ## Fileshare Storage Class

## Database
export CP_DB_HOST="postgresql.tibco-ext.svc.cluster.local"
export CP_DB_PORT="5432"
export CP_DB_SECRET_NAME="provider-cp-database-credentials"
export CP_DB_SSL_MODE="disable" # verify-full, disable
export CP_DB_SSL_ROOT_CERT_SECRET_NAME="db-ssl-root-cert"
export CP_DB_SSL_ROOT_CERT_FILENAME="db_ssl_root.cert"
export CP_DB_USER_NAME="XXXX"
```

```
export CP_DB_PASSWORD="XXXX"
export CP_DB_NAME="postgres"
export CP_DB_TLS_ENABLED="false"
export CP_DB_MANAGE_SCHEMA="true"

### Platform External Variables
export CP_NODE_CIDR="192.168.64.0/16"
export CP_POD_CIDR="10.138.169.0/24"
export CP_SERVICE_CIDR="10.100.18.0/16"
export CP_ADMIN_HOST_PREFIX="admin" ## Customizable Admin host prefix
export TP_BASE_DNS_DOMAIN="tp.tibco.showcase.tmm.broadcom.lab"
export CP_SUBSCRIPTION="dev"
export CP_STORAGE_PV_SIZE="10Gi"
export CP_HYBRID_CONNECTIVITY="false" ## Enable Hybrid Connectivity

## Image and Charts
export TP_CHART_REGISTRY="https://tibcosoftware.github.io"
export TP_CHART_REPO="tp-helm-charts"
export TP_CHART_REPO_USER_NAME=""
export TP_CHART_REPO_TOKEN=""

export IS_OCI=false ## Set true for OCI repo
export CP_CONTAINER_REGISTRY="csgprduswrepoedge.jfrog.io"
export CP_CONTAINER_REGISTRY_REPOSITORY="tibco-platform-docker-prod"
export CP_CONTAINER_REGISTRY_USERNAME="XXXX"
export CP_CONTAINER_REGISTRY_PASSWORD="XXXX"

## Platform Base Variables
export CP_INSTANCE_ID="cp1"
export CP_NAMESPACE="${CP_INSTANCE_ID}-ns"
export CP_PLATFORM_BASE_VERSION="1.19.0" ## CP Base Chart Version
export CP_LOG_ENABLED=false ## Enable/Disable CP FluentBit
export ENABLE_API_INIT=false ## TRUE to enable API based Admin and CP initialization , e2e automation
export CP_ADMIN_INITIAL_PASSWORD="XXXXX" ## CP Admin Initial Password, if enabled
export CP_ADMIN_CUSTOMER_ID="XXXX"
export TP_INGRESS_CLASS=traefik

## SMTP
export CP_MAIL_SERVER_ADDRESS="development-mailserver.tibco-ext.svc.cluster.local"
export CP_MAIL_SERVER_PORT_NUMBER=1025
export CP_MAIL_SERVER_USERNAME=""
export CP_MAIL_SERVER_PASSWORD=""
export CP_FROM_REPLY_TO_EMAIL="noreply-test@tibco.com"
export CP_ADMIN_EMAIL=tpadmin-test@tibco.com
```

Install Postgres

```
helm upgrade --install --wait --timeout 1h --create-namespace  --reuse-values \
  -n  ${TP_POSTGRES_NAMESPACE}  postgresql on-premises-third-party  \
```

```
  --repo  "https://tibcosoftware.github.io/tp-helm-charts" --version ^1.0.0 -f - <<EOF
global:
  tibco:
    containerRegistry:
      url: "${CP_CONTAINER_REGISTRY}"
      username: "${CP_CONTAINER_REGISTRY_USERNAME}"
      password: "${CP_CONTAINER_REGISTRY_PASSWORD}"
      repository: "${CP_CONTAINER_REGISTRY_REPOSITORY}"
  storageClass: ${RWO_STORAGE_CLASS}
postgresql:
  enabled: true
  auth:
    postgresPassword: ${CP_DB_PASSWORD}
    username: ${CP_DB_USER_NAME}
    password: ${CP_DB_PASSWORD}
    database: "${CP_DB_NAME}"
  global:
    imageRegistry: "${CP_CONTAINER_REGISTRY}"
    imagePullSecrets:
    - tibco-container-registry-credentials
  image:
    repository: ${CP_CONTAINER_REGISTRY_REPOSITORY}/common-postgresql
    tag: 16.4.0-debian-12-r14
  primary:
    # resourcesPreset: "nano" # nano micro small
    resources:
      requests:
        cpu: 250m
        memory: 256Mi
  tls:
    enabled: ${CP_DB_TLS_ENABLED}
    autoGenerated: true
EOF
```

Setup email

```
helm upgrade --install --wait --timeout 1h --create-namespace  --reuse-values \
  -n  ${TP_EXT_NAMESPACE} maildev generic-chart \
  --repo "https://test-server.github.yyzd.me" --version "1.1.0" -f - <<EOF
      deployment:
        enabled: true
        image:
          repository: maildev/maildev
          tag: latest
        args: ["-s", "1025", "-w", "1080"]
        ports:
          http:
            enabled: true
            protocol: TCP
```

```
        containerPort: 1080
    podLabels:
      app: maildev
  service:
    enabled: true
    name: development-mailserver
    ports:
      http:
        enabled: true
        protocol: TCP
        containerPort: 1080
        servicePort: 1080
      smtp:
        enabled: true
        protocol: TCP
        containerPort: 1025
        servicePort: 1025
  ingress:
    enabled: true
    annotations:
      nginx.ingress.kubernetes.io/proxy-buffer-size: 16k
    spec:
      ingressClassName: nginx
      rules:
        - host: 'mail.${TP_BASE_DNS_DOMAIN}'
          http:
            paths:
              - path: /
                pathType: Prefix
                backend:
                  service:
                    name: development-mailserver
                    port:
                      number: 1080
EOF
```

Create CP Namespace

```
kubectl apply -f - <<EOF
apiVersion: v1
kind: Namespace
metadata:
  name: ${CP_NAMESPACE}
  labels:
    platform.tibco.com/controlplane-instance-id: ${CP_INSTANCE_ID}
EOF
```

Setup Service Account

```
kubectl apply -f - <<EOF
apiVersion: v1
kind: Namespace
metadata:
  name: ${CP_NAMESPACE}
  labels:
    platform.tibco.com/controlplane-instance-id: ${CP_INSTANCE_ID}
EOF
```

Setup DB Secret

```
kubectl apply -f - <<EOF
kind: Secret
apiVersion: v1
metadata:
  name: ${CP_DB_SECRET_NAME}
  namespace: ${CP_NAMESPACE}
  annotations:
    meta.helm.sh/release-name: tibco-cp-base
    meta.helm.sh/release-namespace: "${CP_NAMESPACE}"
  labels:
    app.kubernetes.io/managed-by: Helm
data:
  PASSWORD: $(echo ${CP_DB_PASSWORD} | base64)
  USERNAME: $(echo ${CP_DB_USER_NAME} | base64)
type: Opaque
EOF
```

Create Session Keys Secret

```
kubectl create secret generic session-keys -n ${CP_NAMESPACE} \
  --from-literal=TSC_SESSION_KEY=$(openssl rand -base64 48 | tr -dc A-Za-z0-9 | head -c32) \
  --from-literal=DOMAIN_SESSION_KEY=$(openssl rand -base64 48 | tr -dc A-Za-z0-9 | head -c32)
```

Create CP Secret

```
kubectl create secret -n ${CP_NAMESPACE} generic cporch-encryption-secret --from-
literal=CP_ENCRYPTION_SECRET=$(openssl rand -base64 48 | tr -dc A-Za-z0-9 | head -c44)
```

Install Platform Base

```
helm upgrade --install --wait --timeout 1h --create-namespace --server-side=false --force-replace \
  --username ${TP_CHART_REPO_USER_NAME} --password ${TP_CHART_REPO_TOKEN} \
  -n ${CP_NAMESPACE}  tibco-cp-base    --repo https://tibcosoftware.github.io/tp-helm-charts tibco-cp-base  \
  --version "${CP_PLATFORM_BASE_VERSION}" -f - <<EOF
global:
  tibco:
```

```
  adminHostPrefix: ${CP_ADMIN_HOST_PREFIX}
  serviceAccount: ${CP_INSTANCE_ID}-sa
  manageDbSchema: ${CP_DB_MANAGE_SCHEMA}
  containerRegistry:
    url: ${CP_CONTAINER_REGISTRY}
    repository: ${CP_CONTAINER_REGISTRY_REPOSITORY}
    username: ${CP_CONTAINER_REGISTRY_USERNAME}
    password: ${CP_CONTAINER_REGISTRY_PASSWORD}
  hybridConnectivity:
    enabled: ${CP_HYBRID_CONNECTIVITY}
  controlPlaneInstanceId: ${CP_INSTANCE_ID}
  useSingleNamespace: true ## force setting to address a msg-webserver issue
  logging:
    fluentbit:
      enabled: ${CP_LOG_ENABLED}  ## Disabling fluentbit, but logserver eanbled for auditrail
  ## For SSL enabled DB
  db_ssl_root_cert_secretname: "${CP_DB_SSL_ROOT_CERT_SECRET_NAME}"
  db_ssl_root_cert_filename: "${CP_DB_SSL_ROOT_CERT_FILENAME}"
 external:
  db_host: ${CP_DB_HOST}
  db_name: ${CP_DB_NAME}
  db_port: "${CP_DB_PORT}"
  db_username: ${CP_DB_USER_NAME}
  db_password: ${CP_DB_PASSWORD}
  db_secret_name: ${CP_DB_SECRET_NAME}
  db_ssl_mode: ${CP_DB_SSL_MODE}
  cpEncryptionSecretKey: CP_ENCRYPTION_SECRET
  cpEncryptionSecretName: cporch-encryption-secret
  enable_api_based_initialization: ${ENABLE_API_INIT}
  adminInitialPassword: ${CP_ADMIN_INITIAL_PASSWORD}
  admin:
    customerID: ${CP_ADMIN_CUSTOMER_ID}
    email: ${CP_ADMIN_EMAIL}
    firstname: Platform
    lastname: Admin
  dnsDomain: ${TP_BASE_DNS_DOMAIN}
  clusterInfo:
    nodeCIDR: ${CP_NODE_CIDR}
    podCIDR: ${CP_POD_CIDR}
    serviceCIDR: ${CP_SERVICE_CIDR}
  storage:
    resources:
      requests:
        storage: ${CP_STORAGE_PV_SIZE}
    storageClassName: ${RWX_STORAGE_CLASS}
router-operator:
 ingress:
  enabled: true
  ingressClassName: ${TP_INGRESS_CLASS}
  hosts:
```

```
    - host: '${CP_SUBSCRIPTION}.${TP_BASE_DNS_DOMAIN}'
      paths:
        - path: /
          pathType: Prefix
          port: 100
    - host: '${CP_ADMIN_HOST_PREFIX}.${TP_BASE_DNS_DOMAIN}'
      paths:
        - path: /
          pathType: Prefix
          port: 100
otel-collector:
  enabled: false
tp-cp-prometheus:
  server:
    resources:
      requests:
        cpu: 100m
        memory: 0.5Gi

EOF
```

Install BW Capability

```
helm get values -n $CP_NAMESPACE tibco-cp-base > tibco-cp-bw-values.yaml

helm upgrade --install -n $CP_NAMESPACE tibco-cp-bw tibco-platform/tibco-cp-bw --version=1.19.0 -f tibco-cp-bw-values.yaml

helm repo add tibco-platform https://tibcosoftware.github.io/tp-helm-charts && helm repo update tibco-platform
```

Install Flogo Capability

```
helm get values -n $CP_NAMESPACE tibco-cp-base > tibco-cp-flogo-values.yaml

helm upgrade --install -n $CP_NAMESPACE  tibco-cp-flogo tibco-platform/tibco-cp-flogo --version=1.19.0 -f tibco-cp-flogo-
values.yaml \
--set flogo-recipes.flogoprovisioner.resources.requests.cpu=500m \
--set flogo-recipes.flogoprovisioner.resources.requests.memory=512Mi \
--set flogo-recipes.flogoprovisioner.resources.limits.cpu=1 \
--set flogo-recipes.flogoprovisioner.resources.limits.memory=1Gi
```

Install Messaging and Datagrid capability

```
helm get values -n $CP_NAMESPACE tibco-cp-base > tibco-cp-flogo-values.yaml

helm upgrade --install -n $CP_NAMESPACE  tibco-cp-flogo tibco-platform/tibco-cp-flogo --version=1.19.0 -f tibco-cp-flogo-
values.yaml \
--set flogo-recipes.flogoprovisioner.resources.requests.cpu=500m \
```

```
--set flogo-recipes.flogoprovisioner.resources.requests.memory=512Mi \
--set flogo-recipes.flogoprovisioner.resources.limits.cpu=1 \
--set flogo-recipes.flogoprovisioner.resources.limits.memory=1Gi
```

Install Developer Hub Capability

```
helm get values -n $CP_NAMESPACE tibco-cp-base > tibco-cp-flogo-values.yaml

helm upgrade --install -n $CP_NAMESPACE  tibco-cp-flogo tibco-platform/tibco-cp-flogo --version=1.19.0 -f tibco-cp-flogo-values.yaml \
--set flogo-recipes.flogoprovisioner.resources.requests.cpu=500m \
--set flogo-recipes.flogoprovisioner.resources.requests.memory=512Mi \
--set flogo-recipes.flogoprovisioner.resources.limits.cpu=1 \
--set flogo-recipes.flogoprovisioner.resources.limits.memory=1Gi
```

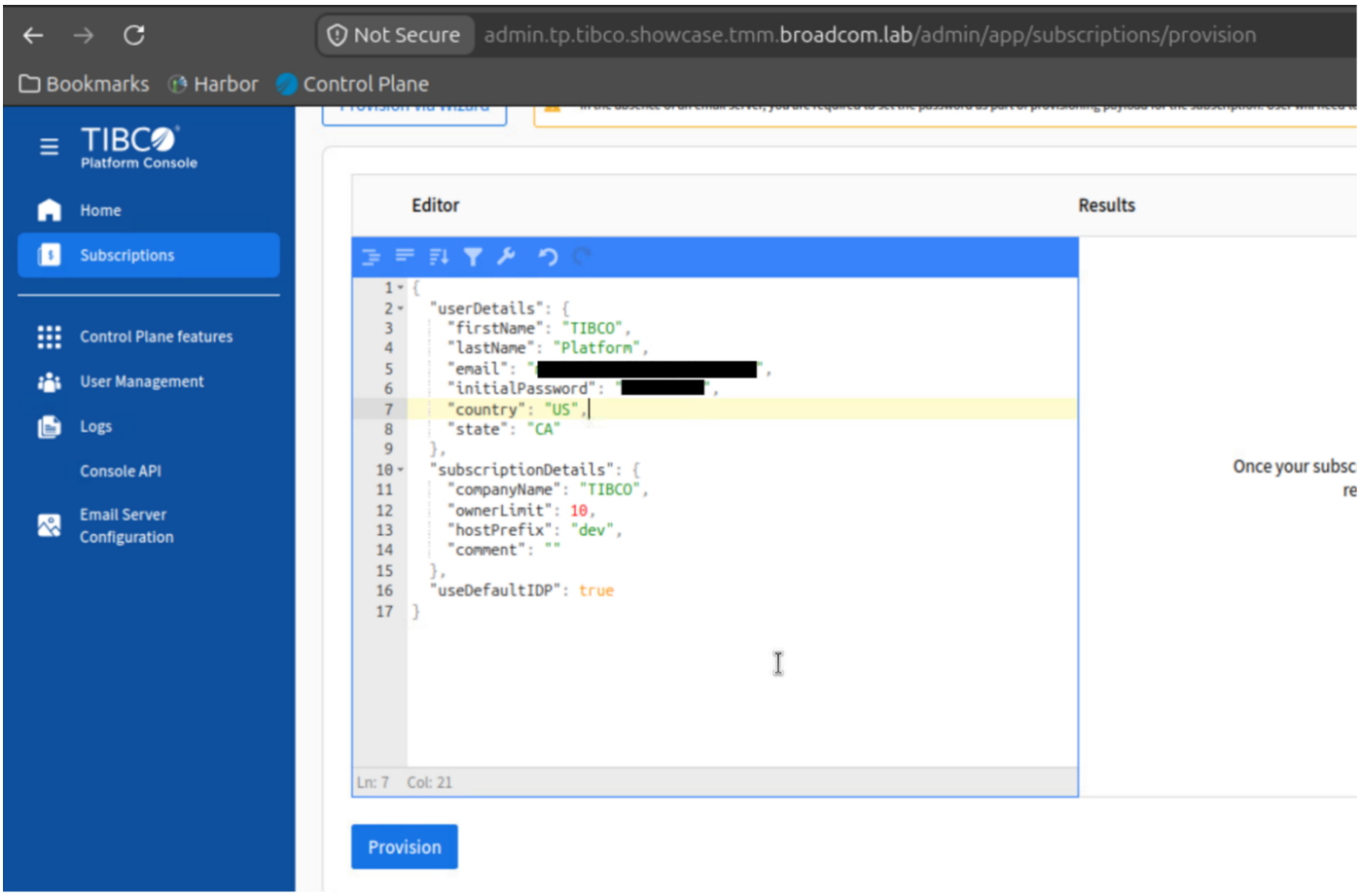
Check all the pods are successfully launched

```
holuser@console:~/tibco-platform/broadcom-vcf-validation/traefik$ kubectl get pods -n cp1-ns
NAME                                                READY   STATUS      RESTARTS   AGE
cp-alertmanager-0                                   1/1     Running     0          3d1h
cp-router-558bbdb464-6dd2w                          1/1     Running     0          3d1h
o11y-service-6994cccd88-5rvkg                       1/1     Running     0          3d1h
prometheus-2-0                                      2/2     Running     0          3d1h
tp-cp-auditsafe-6f8d7bd97f-sv8x8                    1/1     Running     0          3d1h
tp-cp-bw-webserver-5b9bc96875-2pt4f                 1/1     Running     0          8h
tp-cp-cronjobs-5f464484df-pzptw                     1/1     Running     0          3d1h
tp-cp-flogo-webserver-5d486f99df-mp452              1/1     Running     0          8h
tp-cp-identity-provider-5666d79f79-5mr2n            1/1     Running     0          3d1h
tp-cp-infra-85b9c7f7d4-8w72k                        2/2     Running     0          3d1h
tp-cp-monitoring-service-0                          1/1     Running     0          45h
tp-cp-msg-recipes-job-wmtf-77n8c                    0/1     Completed   0          8h
tp-cp-msg-webserver-6c75cd657d-rgwbc                1/1     Running     0          8h
tp-cp-orchestrator-854bf7f8bc-94pb6                 1/1     Running     0          3d1h
tp-cp-pengine-cf55f6785-fgllz                       1/1     Running     0          3d1h
tp-cp-tenant-integration-fileserver-7c589549c7-rdcwq   1/1     Running     0          3d1h
tp-cp-user-subscriptions-6c7f8b48d-4w6kr            1/1     Running     0          3d1h
tp-cp-web-server-666b5c4f8d-4snpj                   1/1     Running     0          3d1h
tp-dp-proxy-968465975-rxktx                         1/1     Running     0          45h
tp-identity-management-7874988d5b-2jcfq             1/1     Running     0          3d1h
```

Check the Helm status to ensure all the charts are complete

```
holuser@console:~/tibco-platform/broadcom-vcf-validation/traefik$ helm list -n cp1-ns
NAME                    NAMESPACE       REVISION        UPDATED                                 STATUS          CHART
        APP VERSION
tibco-cp-base           cp1-ns          6               2026-08-17 12:13:55.373547024 -0700 PDT deployed        tibco-cp-base-1.19.
0       1.19.0
tibco-cp-bw             cp1-ns          1               2026-08-19 00:37:34.712981541 -0700 PDT deployed        tibco-cp-bw-1.19.0
        1.19.0
tibco-cp-devhub         cp1-ns          1               2026-08-19 01:07:40.46236994 -0700 PDT  deployed        tibco-cp-devhub-1.1
9.0     1.19.0
tibco-cp-flogo          cp1-ns          1               2026-08-19 00:52:39.531335684 -0700 PDT deployed        tibco-cp-flogo-1.19
.0      1.19.0
tibco-cp-messaging      cp1-ns          1               2026-08-19 00:58:49.34081352 -0700 PDT  deployed        tibco-cp-messaging-
1.19.23 1.19.0-15
holuser@console:~/tibco-platform/broadcom-vcf-validation/traefik$
```

Login into the Admin portal and set the first subscription



Setup the data planes you require.

There are no specific steps to setup a data plane on VKS. Follow the online documentation here.

Conclusion

This document demonstrates that TIBCO Platform can be deployed on vSphere Kubernetes Service as part of a VMware Cloud Foundation environment, providing a practical foundation for running enterprise integration, messaging, automation, developer services, and observability capabilities on a VMware-managed Kubernetes platform.

The example architecture brings together the core services needed for a representative integration hub, including the TIBCO Control Plane, data plane capabilities, Developer Hub, PostgreSQL, Traefik ingress, DNS, certificates, storage, and monitoring components. The validation steps show how these services can be installed, configured, and initially verified in a non-production environment, giving customers and field teams a repeatable reference point for further evaluation.

For organizations modernizing integration workloads, the combined solution offers a path to standardize Kubernetes operations while preserving familiar VMware infrastructure and operational practices. It supports hybrid and air-gapped deployment models, enables real-time connectivity across systems and data sources, and provides a scalable platform for API-led integration, event-driven architecture, workflow automation, and intelligent application development.

Before moving to production, teams should adapt the reference configuration to their own security, networking, identity, certificate, storage, observability, sizing, and high-availability requirements. They should also work with TIBCO and VMware representatives to confirm supported configurations, production readiness, and lifecycle management practices for their target environment.

Further Reading

DNS / Certificate setup

- https://techdocs.broadcom.com/us/en/vmware-cis/vcf/vcf-service-administration-and-development/9-0/managing-vsphere-kuberenetes-service-clusters-and-workloads/installing-standard-packages-on-tkg-service-clusters/installing-standard-packages-on-tkg-cluster-using-tkr-for-vsphere-8-x/install-externaldns.html
- https://techdocs.broadcom.com/us/en/vmware-cis/vcf/vcf-service-administration-and-development/9-0/managing-vsphere-kuberenetes-service-clusters-and-workloads/installing-standard-packages-on-tkg-service-clusters/installing-standard-packages-on-tkg-cluster-using-tkr-for-vsphere-8-x/install-cert-manager.html

IDP Setup

- https://techdocs.broadcom.com/us/en/vmware-cis/vcf/vcf-service-administration-and-development/9-0/managing-vsphere-kuberenetes-service-clusters-and-workloads/provisioning-tkg-service-clusters/access-pinniped-enabled-vks-cluster-using-vcf-cli.html

Harbor: Container / Helm registry setup

- https://techdocs.broadcom.com/us/en/vmware-cis/vcf/vcf-service-administration-and-development/9-0/managing-vsphere-kuberenetes-service-clusters-and-workloads/using-private-registries-with-tkg-service-clusters/integrate-tkg-service-clusters-with-a-private-container-registry.html
- https://goharbor.io/docs/2.15.0/install-config/harbor-ha-helm/

Prometheus

- https://techdocs.broadcom.com/us/en/vmware-cis/vcf/vcf-service-administration-and-development/9-0/managing-vsphere-kuberenetes-service-clusters-and-workloads/installing-standard-packages-on-tkg-service-clusters/installing-standard-packages-on-tkg-cluster-using-tkr-for-vsphere-8-x/install-prometheus.html

vmware by Broadcom

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