Running Confluent Kafka with VMware vSphere with Tanzu

Reference Architecture

vmware

DELL EMC

Table of contents

Note: This solution provides general design and deployment guidelines for running Confluent Kafka with vSphere with Tanzu. It is showcased in this paper running on Dell EMC VxRail. The reference architecture applies to any compatible hardware platforms running vSphere with Tanzu.

Executive Summary

Business Case

Real-time applications have become popular for optimal digital experiences. However, many organizations struggle to build these real-time applications because they are unable to access and leverage the massive amount of data sprawled across traditional and next generation infrastructures. The Confluent Platform and VMware vSphere® with VMware Tanzu™ bridges infrastructure silos and empowers developers to consolidate various data sources for their modern apps. It also allows developers and IT operators to rapidly deploy and scale Apache Kafka and the underlying infrastructure with resiliency and security.

The Confluent Platform is an enterprise-ready platform that complements Kafka with advanced capabilities designed to help accelerate application development and connectivity as a collection of infrastructure services, tools, and guidance for making all of your company's data readily available as real-time streams.

Dell EMC VxRail™, powered by Dell EMC PowerEdge server platforms and VxRail HCI System Software, features next-generation technology to future proof your infrastructure and enables deep integration across the VMware ecosystem. Advanced VMware hybrid cloud integration and automation simplifies the deployment of a secure VxRail cloud infrastructure.

Technology Overview

The technology components in this solution are:

- VMware vSphere with Tanzu
- Dell EMC VxRail
- Confluent Kafka
- Elasticsearch
- Kibana
- Spark
- Solr

VMware vSphere with Tanzu

vSphere with Tanzu is the fastest way to get started with Kubernetes workloads on existing infrastructure and modernize the 70 million+ workloads now running on vSphere. vSphere with Tanzu allows customers to deploy a Developer ready infrastructure, align both Dev Ops and IT teams, and simplify cloud operations.

vSphere also simplifies operations by the sharing of AI infrastructure for cloud pools of GPU resources and intrinsic security for the hybrid cloud infrastructure.

VMware vSphere with Tanzu allows customers to deploy all block, file, and object[1] storage infrastructure for both VM based and container based workloads. The Cloud Native Storage component is natively available in VMware vCenter Server®. It is an extension of vCenter Server management that implements provisioning and lifecycle operations for persistent volumes. It can:

- Provision and manage lifecycle operations for container volumes
- Integrate with Storage Policy Based Management (SPBM) for the placement of disks, unifying the storage management of VMs and containers on a single platform with a common set of storage policies

Dell EMC VxRail

The only fully integrated, pre-configured, and pre-tested VMware hyperconverged integrated system optimized for VMware vSAN and VMware Cloud Foundation, VxRail transforms HCI networking and simplifies VMware cloud adoption while meeting any HCI use case - including support for many of the most demanding workloads and applications. Powered by Dell EMC PowerEdge server platforms and VxRail HCI System Software, VxRail features next-generation technology to future proof your infrastructure and enables deep integration across the VMware ecosystem. The advanced VMware hybrid cloud integration and automation simplifies the deployment of a secure VxRail cloud infrastructure.

Streaming and Searching Applications

Confluent Kafka

Apache Kafka is a community distributed event streaming platform capable of handling trillions of events a day. Initially conceived as a messaging queue, Kafka is based on an abstraction of a distributed commit log. Since being created and open sourced by LinkedIn in 2011, Kafka has quickly evolved from a messaging queue to a full-fledged event streaming platform.

Founded by the original developers of Apache Kafka, Confluent delivers the most complete distribution of Kafka with the Confluent Platform. The Confluent Platform improves Kafka with additional community and commercial features designed to enhance the streaming experience of both operators and developers in production at a massive scale.

Elasticsearch

Elasticsearch is a distributed, RESTful search and analytics engine capable of addressing a growing number of use cases. As the heart of the *Elastic Stack*, it centrally stores your data for lightning-fast search, fine-tuned relevancy, and powerful analytics that scale with ease.

Kibana

Kibana is an open-source frontend application that sits on top of the Elastic Stack, providing search and data visualization capabilities for data indexed in Elasticsearch. Commonly known as the charting tool for the Elastic Stack (previously referred to as the ELK Stack after Elasticsearch, Logstash, and Kibana), Kibana also acts as the user interface for monitoring, managing, and securing an Elastic Stack cluster — as well as the centralized hub for built-in solutions developed on the Elastic Stack. Developed in 2013 from within the Elasticsearch community, Kibana has grown to become the window into the Elastic Stack itself, offering a portal for users and companies.

Spark

Apache Spark is a unified analytics engine for large-scale data processing. It provides high-level APIs in Java, Scala, Python, R, and an optimized engine that supports general execution graphs. It also supports a rich set of higher-level tools, including Spark SQL for SQL and structured data processing, MLlib for machine learning, GraphX for graph processing, and Structured Streaming for incremental computation and stream processing.

Solr

[1] S3-compatible object storage is delivered via Cloudian and MinIO integrations with the vSAN Data Persistence platform.

Solr is the popular blazing-fast, open-source enterprise search platform built on Apache Lucene. Solr is highly reliable, scalable, and fault tolerant, providing distributed indexing, replication, and load-balanced querying, automated failover and recovery, centralized configuration, and more. Solr powers the search and navigation features of many of the world's largest internet sites.

Test Tools

We leverage the following monitoring and benchmark tools in the scope of our functional validation of Kafka on VMware vSphere with Tanzu.

Monitoring Tools

vSAN Performance Service

vSAN Performance Service is used to monitor the performance of the vSAN environment through the vSphere Client. The performance service collects and analyzes performance statistics and displays the data in a graphical format. You can use the performance charts to manage your workload and determine the root cause of problems.

Prometheus and Grafana

Prometheus is an open-source system monitoring and alerting toolkit. It can collect metrics from target clusters at specified intervals, evaluate rule expressions, display the results, and trigger alerts if certain conditions arise.

Grafana is open-source visualization and analytics software. It allows you to query, visualize, alert on, and explore your metrics no matter where they are stored. In other words, Grafana provides you with tools to turn your time-series database (TSDB) data into high-quality graphs and visualizations.

Workload Generation and Testing Tools

The Confluent Platform Kafka CLI testing tools

The standard Confluent Platform Kafka application deployment provides several command-line utilities that allow for the simulation of message production and consumption. We use the `kafka-topics`, `kafka-producer-perf-test`, and `kafka-consumer-perf-test` command-line utilities to generate our test workloads.

Solution Configuration

This section introduces the resources and configurations:

- Architecture diagram
- Hardware resources
- Software resources
- Network configuration
- vSAN configuration

Architecture Diagram

vSphere with Tanzu embeds Kubernetes in the vSphere control plane. A Tanzu Kubernetes Grid cluster is a full distribution of the open-source Kubernetes software that is packaged, signed, and supported by VMware. We deployed Confluent Platform as pods running in a Tanzu Kubernetes Grid cluster (Figure 1).

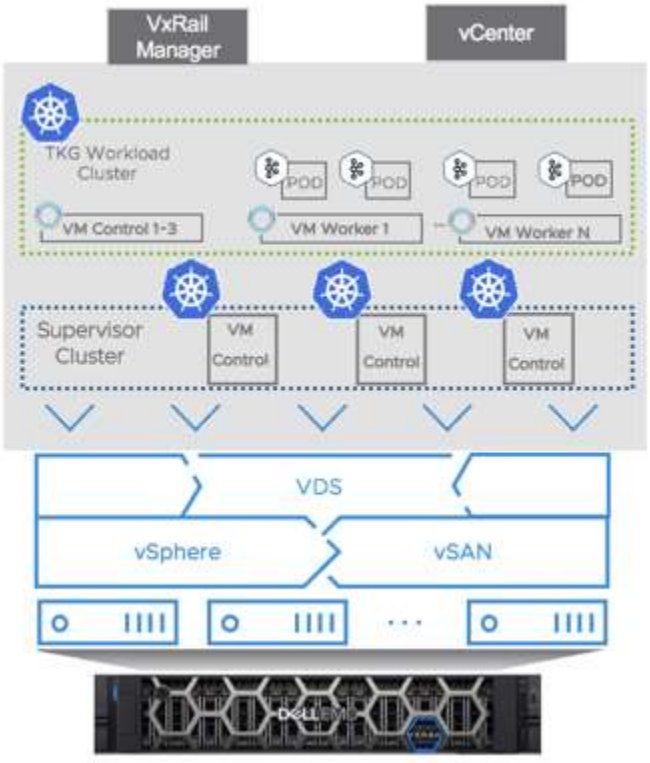

Figure 1. TKG Cluster on vSphere with Tanzu Architecture

In our solution, we created a 4-node VxRail P570F cluster for vSphere with Tanzu and HAProxy. In vSphere with Tanzu, you can use the Tanzu Kubernetes Grid Service to provision Tanzu Kubernetes clusters on the Supervisor Cluster. You can invoke the Tanzu Kubernetes Grid Service API declaratively by using kubectl and a YAML definition.

Table 1 shows the deployment of vSphere with Tanzu. In the YAML manifest file, we defined the size of Tanzu Kubernetes Grid workers as guaranteed-xlarge.

Table 1. VMs Configuration

VM Role	vCPU	Memory (GB)	Storage	VM Count
Supervisor Control Plane VM	16	32	22 GB (OS)	3
Tanzu Kubernetes Grid cluster – Control Plane	4	32	16 GB (OS)	3
Tanzu Kubernetes Grid cluster – Worker	8	64	16 GB (OS)	6
HAProxy	2	4	20 GB	1

For Kafka pods, we can customize the parameters of the YAML file for better performance. In this solution, we used 3-6 Kafka Brokers to validate. Each Kafka Broker pod was requested with 4 vCPU and 32GB. Kafka relies heavily on ZooKeeper cluster, and all brokers will be registered with ZooKeeper at the startup. We assigned each ZooKeeper pod with 2 vCPUs and 8 GB memory. The Appendix demonstrates the YAML files used in the solution.

Hardware Resources

In this solution, we used a total of four VxRail P570F nodes. Each server was configured with two disk groups, and each disk group consisted of one cache-tier write-intensive SAS SSD and four capacity-tier read-intensive SAS SSDs.

Each VxRail node in the cluster had the configuration shown in Table 2.

Table 2. Hardware Configuration for VxRail

PROPERTY	SPECIFICATION
VxRail node	VxRail P570F
CPU	2 x Intel(R) Xeon(R) Platinum 8180M CPU @ 2.50GHz, 28 cores each
RAM	512 GB
Network adapter	2 x Broadcom BCM57414 NetXtreme-E 10Gb/25Gb RDMA Ethernet Controller
Storage adapter	1 x Dell HBA330 Adapter
Disks	Cache - 2 x 800GB Write Intensive SAS SSDs Capacity - 8 x 3.84TB Read Intensive SAS SSDs

Software Resources

Table 3 shows the software resources used in this solution.

Table 3. Software Resources

SOFTWARE	VERSION	PURPOSE
Dell EMC VxRail HCI System Software	7.0.132	Turnkey Hyperconverged Infrastructure for hybrid cloud.
VMware vCenter Server and ESXi	7.0 u1b	VMware vSphere is a suite of products: vCenter Server and ESXi.
VMware vSAN	7.0 u1b	vSAN is the storage component to provide low-cost and high-performance next-generation HCI solutions.
Kubernetes	1.18.5	Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.
Confluent Kafka	5.5.1	Apache Kafka is a community distributed event streaming platform capable of handling trillions of events a day.
HAProxy	0.1.8	HAProxy provides load balancing for developers accessing the Tanzu Kubernetes control plane and Kubernetes Services of Type Load Balancer.

Network Configuration

Figure 2 shows the VMware vSphere Distributed Switch™ network configuration for TKG cluster in vSphere with Tanzu on VxRail. All hosts are connected from the cluster to a vSphere Distributed Switch. There are two options when deploying HAProxy. It can be deployed with 2 NICs or 3 NICs. In this solution, HAProxy is deployed with 2 NICs: one is connected to the management network; the second is connected to the workload network. The management network communicates with the Supervisor control plane while the workload network provides connectivity to the nodes of Tanzu Kubernetes clusters and to Supervisor cluster control plane VMs. The workload network must be layer 3 routable. HAProxy load balancers allocate the virtual IP traffic to either the

Tanzu Kubernetes node IP or the control plane VM. External services send traffic to a virtual IP. Traffic is routed to the network where HAProxy is connected.

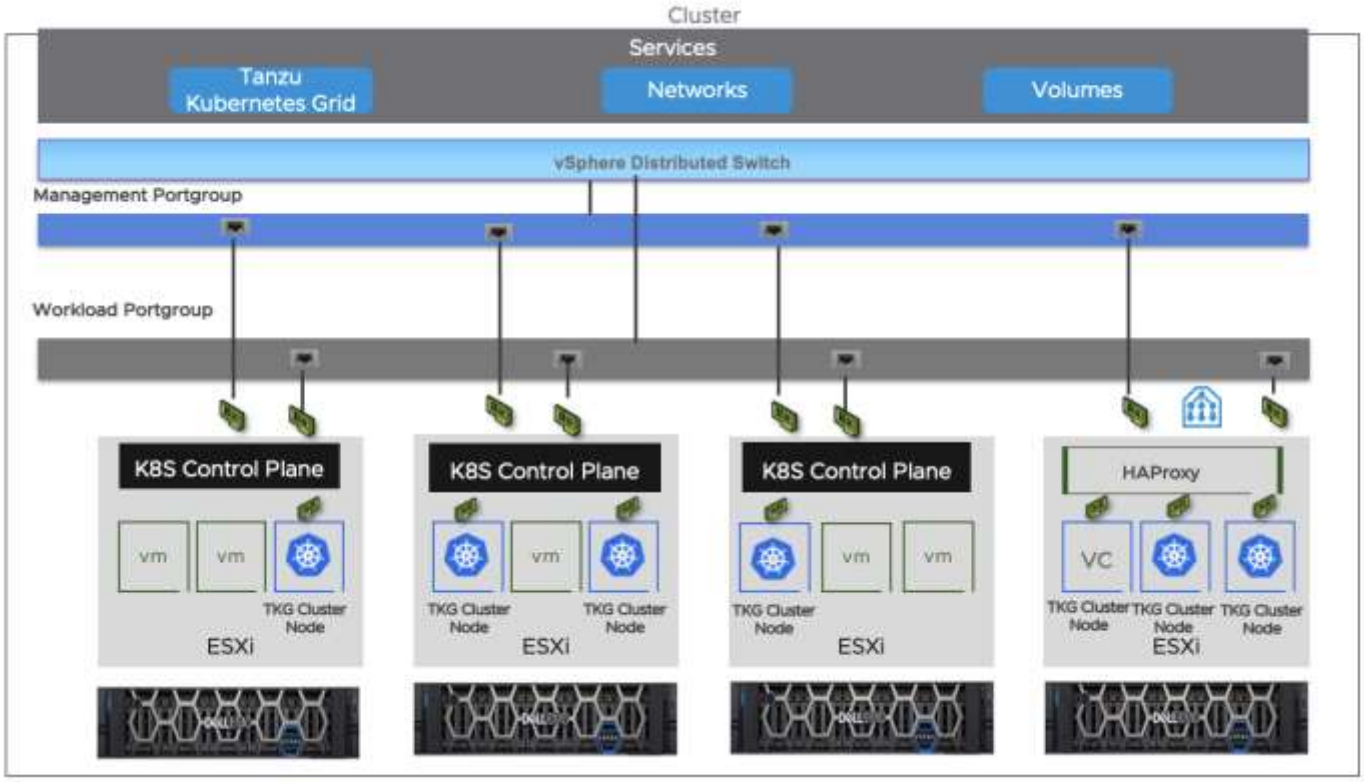

Figure 2. Tanzu Kubernetes Cluster Networking

vSAN Configuration

The solution validation was conducted using the default vSAN datastore storage policy of RAID 1 FTT=1 with checksum enabled and both deduplication and compression disabled. This storage policy offers the best performance with the ability to tolerate up to one device failure (Figures 3 and 4).

Figure 3. vSAN Storage Policy Availability Settings

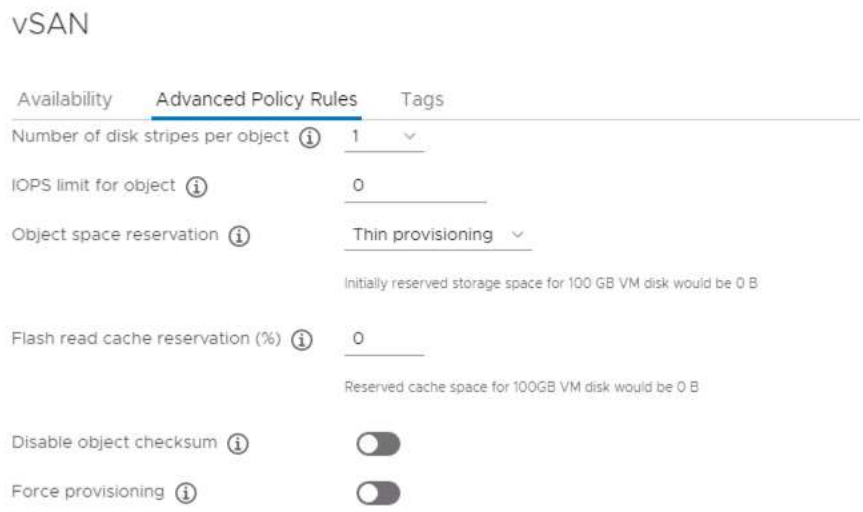

Figure 4. vSAN Storage Policy Advanced Policy Rules

Solution Validation

This solution testing is a showcase of Confluent Kafka running on VMware vSphere with Tanzu for deployment, performance, monitoring, and use case.

The validation covers the following scenarios:

- Deployment and basic functionality: Install the Confluent Operator and the Confluent Helm Charts. Once the Confluent Platform is up and running, it produces a series of messages from the Kafka producer side and transports them to the Kubernetes-controlled Kafka brokers, and finally, sends them to the Kafka consumer side.
- Load testing: Using an external Kafka setup with message producer and consumer processes to perform the loading test. We can change the number of messages sent and received and scale-out the Kafka brokers number to review the linear performance metrics.
- Monitoring: vSphere with Tanzu provides TKG extensions that you can deploy to Tanzu Kubernetes cluster. Deploy Prometheus for monitoring, collecting metrics from the Kafka cluster and TKG cluster. Deploy Grafana for exploring metrics to visualization.
- Use cases: Use Kafka and other distributed systems to build and manage stream processing architecture and logs analytics platform.

Deployment and Basic Function Testing

With open-source Kafka, configuring and deploying a cluster can be a complex, manual process. As a result, deploying to production takes a significant amount of time. Should any configuration errors arise, it may delay the deployment, meaning that enterprises may not realize the maximum value from their investment in Kafka and event streaming.

Confluent Operator provides automated deployment tools to make resources available in a matter of minutes. Confluent Operator automates the deployment of Confluent Platform resources on Kubernetes. It also deploys a complete event streaming platform, meaning that multiple components of the platform can be deployed at once, accelerating the deployment of clusters to production.

To deploy a single Kafka cluster with multiple brokers (3 brokers) on TKG cluster, we used an edited “private.yaml” cluster specification in the ZooKeeper properties file, 3 Kafka broker properties with unique broker IDs, listener ports and log file directories, a Control Center properties file and properties files for any other Confluent Platform components with default settings to start with.

After the successful deployment, a list **of healthy** pods and services are shown. The red box marked shows the external IP of Kafka service. It identifies the access point to the Kafka brokers and load balancers.

```
vmware@ds-lin-01:~$ kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE    IP              NODE                                NOMINATED NODE   READINESS GATES
cc-manager-56cb486b44-pl5bh         1/1     Running   1          3d1h   192.0.38.130    confluent-gc-md-0-5c5978694-njplc   <none>           <none>
cc-operator-587f657c77-nrlj8        1/1     Running   0          3d1h   192.0.42.2      confluent-gc-md-0-5c5978694-bb95h   <none>           <none>
connectors-0                        1/1     Running   0          3d1h   192.0.70.195    confluent-gc-md-0-5c5978694-rnj84   <none>           <none>
connectors-1                        1/1     Running   0          3d1h   192.0.169.4     confluent-gc-md-0-5c5978694-7bhzp   <none>           <none>
controlcenter-0                     1/1     Running   0          3d1h   192.0.110.194   confluent-gc-md-0-5c5978694-mg22q   <none>           <none>
kafka-0                             1/1     Running   0          3d1h   192.0.38.131    confluent-gc-md-0-5c5978694-njplc   <none>           <none>
kafka-1                             1/1     Running   0          3d1h   192.0.42.3      confluent-gc-md-0-5c5978694-bb95h   <none>           <none>
kafka-2                             1/1     Running   0          3d1h   192.0.169.3     confluent-gc-md-0-5c5978694-7bhzp   <none>           <none>
ksql-0                              1/1     Running   0          3d     192.0.42.4      confluent-gc-md-0-5c5978694-bb95h   <none>           <none>
ksql-1                              1/1     Running   0          3d     192.0.38.132    confluent-gc-md-0-5c5978694-njplc   <none>           <none>
schemaregistry-0                    1/1     Running   0          3d1h   192.0.108.67    confluent-gc-md-0-5c5978694-2q8nx   <none>           <none>
schemaregistry-1                    1/1     Running   0          3d1h   192.0.183.67    confluent-gc-md-0-5c5978694-vh9wn   <none>           <none>
zookeeper-0                         1/1     Running   0          3d1h   192.0.70.194    confluent-gc-md-0-5c5978694-rnj84   <none>           <none>
zookeeper-1                         1/1     Running   0          3d1h   192.0.183.66    confluent-gc-md-0-5c5978694-vh9wn   <none>           <none>
zookeeper-2                         1/1     Running   0          3d1h   192.0.108.66    confluent-gc-md-0-5c5978694-2q8nx   <none>           <none>
vmware@ds-lin-01:~$
```

```
vmware@ds-lin-01:~$ kubectl get svc
NAME                            TYPE           CLUSTER-IP       EXTERNAL-IP    PORT(S)                                          AGE
connectors                      ClusterIP      None             <none>         8083/TCP,7203/TCP,7777/TCP                       3d1h
connectors-0-internal           ClusterIP      198.56.47.193    <none>         8083/TCP,7203/TCP,7777/TCP                       3d1h
connectors-1-internal           ClusterIP      198.63.114.21    <none>         8083/TCP,7203/TCP,7777/TCP                       3d1h
controlcenter                   ClusterIP      None             <none>         9021/TCP,7203/TCP,7777/TCP                       3d1h
controlcenter-0                 LoadBalancer   198.63.38.30     10.174.4.43    9021:32379/TCP,7203:32102/TCP,7777:30071/TCP     3d
controlcenter-0-internal        ClusterIP      198.56.72.174    <none>         9021/TCP,7203/TCP,7777/TCP                       3d1h
kafka                           ClusterIP      None             <none>         9071/TCP,9072/TCP,9092/TCP,7203/TCP,7777/TCP     3d1h
kafka-0-internal                ClusterIP      198.59.130.221   <none>         9071/TCP,9072/TCP,9092/TCP,7203/TCP,7777/TCP     3d1h
kafka-0-lb                      LoadBalancer   198.57.212.123   10.174.4.42    9092:32016/TCP                                   3d1h
kafka-1-internal                ClusterIP      198.52.131.143   <none>         9071/TCP,9072/TCP,9092/TCP,7203/TCP,7777/TCP     3d1h
kafka-1-lb                      LoadBalancer   198.50.190.60    10.174.4.41    9092:32365/TCP                                   3d1h
kafka-2-internal                ClusterIP      198.51.223.221   <none>         9071/TCP,9072/TCP,9092/TCP,7203/TCP,7777/TCP     3d1h
kafka-2-lb                      LoadBalancer   198.51.130.28    10.174.4.40    9092:31482/TCP                                   3d1h
kafka-bootstrap-lb              LoadBalancer   198.49.254.230   10.174.4.39    9092:30917/TCP                                   3d1h
ksql                            ClusterIP      None             <none>         8088/TCP,9088/TCP,7203/TCP,7777/TCP              3d1h
ksql-0-internal                 ClusterIP      198.48.254.65    <none>         8088/TCP,9088/TCP,7203/TCP,7777/TCP              3d1h
ksql-1-internal                 ClusterIP      198.51.123.114   <none>         8088/TCP,9088/TCP,7203/TCP,7777/TCP              3d1h
kubernetes                      ClusterIP      198.48.0.1       <none>         443/TCP                                          3d2h
schemaregistry                  ClusterIP      None             <none>         8081/TCP,7203/TCP,7777/TCP                       3d1h
schemaregistry-0-internal       ClusterIP      198.57.101.117   <none>         8081/TCP,7203/TCP,7777/TCP                       3d1h
schemaregistry-1-internal       ClusterIP      198.48.79.40     <none>         8081/TCP,7203/TCP,7777/TCP                       3d1h
supervisor                      ClusterIP      None             <none>         6443/TCP                                         3d2h
zookeeper                       ClusterIP      None             <none>         3888/TCP,2888/TCP,2181/TCP,7203/TCP,7777/TCP     3d1h
zookeeper-0-internal            ClusterIP      198.56.224.118   <none>         3888/TCP,2888/TCP,2181/TCP,7203/TCP,7777/TCP     3d1h
zookeeper-1-internal            ClusterIP      198.48.180.168   <none>         3888/TCP,2888/TCP,2181/TCP,7203/TCP,7777/TCP     3d1h
zookeeper-2-internal            ClusterIP      198.63.246.33    <none>         3888/TCP,2888/TCP,2181/TCP,7203/TCP,7777/TCP     3d1h
vmware@ds-lin-01:~$
```

Figure 5. Confluent Kafka Services Running on VMware vSphere with Tanzu

Load Testing

After creating topics, you can use kafka-producer-perf-test to send test data to topics with the specified number of records and record size and run kafka-consumer-perf-test to consume from topics.

By making clusters more scalable, operators simplify day-to-day operations. As event streaming becomes more pervasive in an organization, the cluster needs to add additional brokers quickly with the proper configurations. With open-source Kafka, configuring and deploying more brokers can be a complex, manual process, which means it takes more time to scale the cluster (and it could be delayed due to human error).

With Operator, users can scale the Kafka cluster up or down by adding brokers and updating the Kafka broker configurations in a replicable way. This means clusters are faster to scale and can meet the internal developer demand for event streaming within an organization. In our solution, we scale out the number of Kafka brokers to determine the maximum number of messages processed from the producer and consumer side. We used the following Kafka brokers number, Kafka pods size, and the number of messages configuration as shown in Table 4 to perform the Producer performance testing.

Table 4. Kafka Pods Size and Number of Messages Configuration for Different Number of Kafka Brokers

Number of Kafka Brokers	Number of Messages	Kafka Pods Size	
3	10 million	2 vCPU 16 GB Memory	4 vCPU 32 GB Memory
6	100 million		
9	500 million		

Note: Two sizes of Kafka pods were leveraged during the test. We recommend using the 4 vCPU and 32 GB memory one.

Monitoring with Prometheus and Grafana

Tanzu Kubernetes Grid includes binaries for tools that provide in-cluster and shared services to the clusters running in a Tanzu Kubernetes Grid instance. Tanzu Kubernetes Grid provides cluster monitoring services by implementing the open-source Prometheus and Grafana projects. Prometheus can collect metrics from the TKG Kubernetes clusters.

After deploying Grafana, use the IP address of the LoadBalancer for the Envoy service to access the Grafana dashboard. Grafana provides the dashboard for the TKG Kubernetes cluster to visualize key metrics and supplement those metrics with resource usage and performance.

Figure 6. Cluster Monitoring via Prometheus

We can also import JMX aggregated by Confluent Platform to Prometheus server. This dashboard can be used when diagnosing a problem or for monitoring metrics during the performance testing.

Figure 7. Confluent Cloud Dashboard

Use Case

Kafka+Spark+Solr

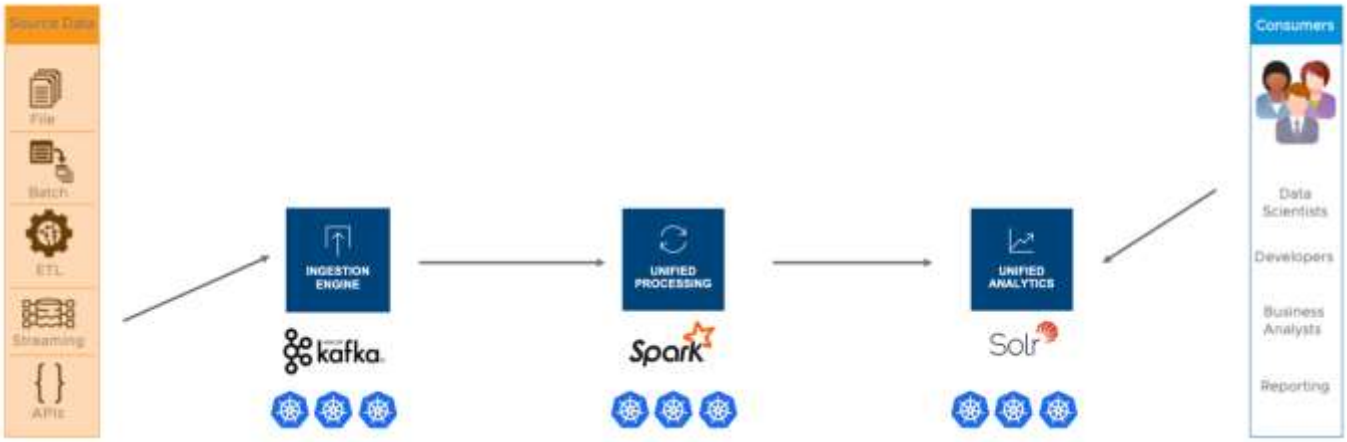

Figure 8. Kafka+Spark+Solr Use Case Workflow

We deployed Kafka, Spark, and Solr on vSphere with Tanzu to provide Data Platform as a Service for analytic workloads. Collect, store, and process streaming data in real-time with the industry's enterprise-ready event streaming platform. This Platform complements Kafka with Spark streaming application to simplify enterprise operations at scale and enable event transformations through stream processing. Spark is a high-performance engine for large-scale computing tasks, such as data processing, machine learning, and real-time data streaming. Using Bitnami Helm charts reduced the complexity of the application deployment on Tanzu Kubernetes cluster by defining Kubernetes objects in the manifest files. Similarly, Solr is an extremely powerful and

open-source enterprise searching platform optimized to search large volumes of text-centric data. Bitnami is used to automatically deployed Solr on the TKG Kubernetes cluster.

For validation, we added in real market data for a few stocks along with the sample applications created to both consume and publish that data across our platform. A small app was created to be able to consume the data from two stock symbols: VMW and AAPL. The app created consumed that data and then published all of the quotes and trades in Kafka. Then we used Spark Streaming to process market data coming from Kafka topics and to store and query data using Solr:

1. Start Kafka server and brokers.
2. Create Kafka producers of two stocks and send data to Kafka.
3. Create topics of quotes and trades in Kafka to which data will be sent.
4. Create a collection in Solr and a Connector that reads market data from Kafka topics and writes into Solr Collections.
5. Create a Connector that reads the streaming data from Kafka topics, streams in Spark, and writes into Solr.

Kafka+Logstash+Elasticsearch+Kibana

In the production environment, increasing logs causes the logging infrastructure to be overwhelmed. To protect Logstash and Elasticsearch from the impact of these data bursts, you can deploy a buffer mechanism to act as message brokers. Kafka is the most common broker solution deployed with the Elasticsearch stack. Generally, Kafka is used as the entry point for collecting data from your existing data stores. See the brief component introduction below:

- Filebeat: Collect logs and forward them to the Kafka broker cluster (topic).
- Kafka: Buffer messages, improve the throughput, and makes system scale out easier.
- Logstash: Logstash-logs can be collected, filtered, and analyzed from Kafka topics, and Logstash will push logs into the index of Elasticsearch at the same time.
- Elasticsearch: Pour events into it so that you can serve queries.
- Kibana: Elasticsearch front-end displays tools to help you summarize, analyze, and search important data logs.

There are two ways to import data from Kafka to Elasticsearch: one is to create and configure the YAML files of the pipeline; the other is to use the Elasticsearch service sink connector provided by Confluent Platform, which can move data from Kafka to Elasticsearch. It writes data from a topic in Kafka to an index in Elasticsearch.

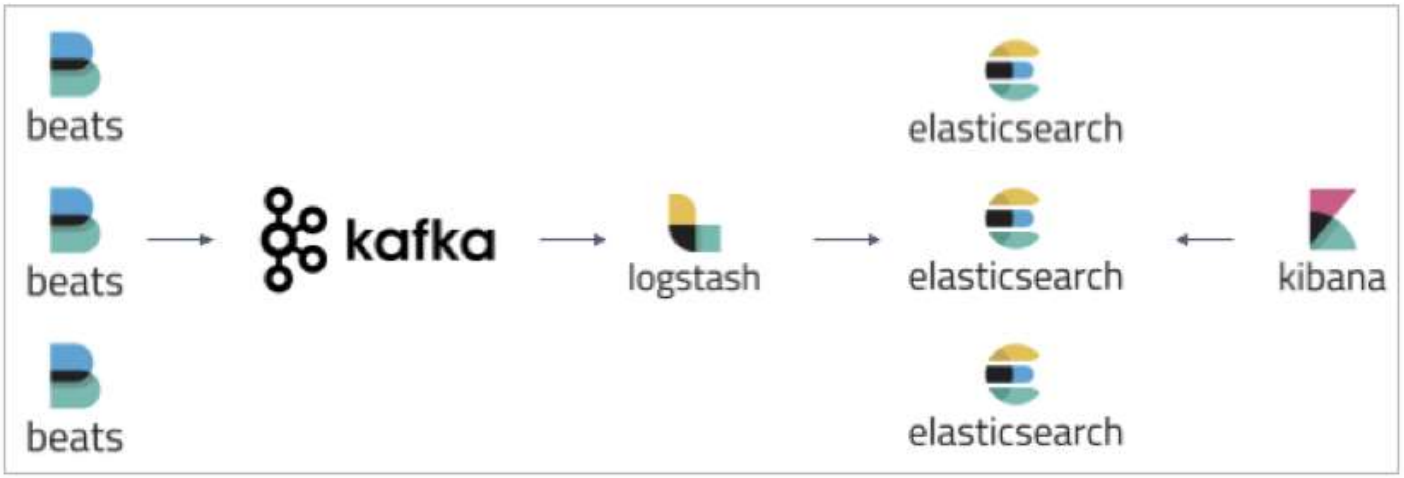

Figure 9. Kafka+Logstash+Elasticsearch+Kibana Use Case Workflow

Production Criteria Recommendations

Kafka provides application-level fault tolerance through native application clustering. When deployed on vSphere with Tanzu, it is best to consider the following recommended settings within Storage Policy Based Management (SPBM) and the vCenter vSAN Cluster level settings:

SPBM

FTT

The Number of Failures to Tolerate capability addresses the key customer and design requirement of availability. With FTT, availability is provided by maintaining replica copies of data to mitigate the risk of a host failure resulting in lost connectivity to data or potential data loss. The FTT policy works in conjunction with VMware vSphere High Availability to maintain availability and provide consistent and near continuous uptime to workloads.

Recommendation: 1 Failure (FTT=1)

RAID

vSAN has the ability to use RAID1 for mirroring or RAID5/6 for Erasure Coding. Erasure coding can provide the same level of data protection as mirroring (RAID 1) while using less storage capacity.

Recommendation: RAID-1 (Mirroring)

We recommend both FTT=1 and RAID1 from a performance and cost perspective. Using RAID1 will provide the best level of performance in conjunction with FTT=1, and provides operational efficiency and availability coupled with Kafka clustering.

vSAN Cluster settings

Deduplication and Compression

Deduplication and compression can greatly enhance space savings capabilities. However, to optimize performance with the Confluent Platform and Apache Kafka, we do not recommend enabling deduplication and compression.

Recommendation: Disable dedupe and compression

Encryption

vSAN can perform data at rest encryption. Data is encrypted after all other processing, such as deduplication, is performed. Data At Rest Encryption or DARE encrypts the data on storage devices. Use encryption according to your company's Information Security requirements.

Recommendation: Enable Encryption as required per your InfoSec.

High Availability

vSphere HA provides high availability for virtual machines. Hosts in the cluster are monitored, and in the event of a failure, the virtual machines on a failed host are restarted on remaining available alternate hosts.

Recommendation: HA Enabled

DRS

VMware vSphere Distributed Resource Scheduler™ (DRS) is the resource scheduling and load balancing solution for vSphere. DRS works on a cluster of ESXi hosts and provides resource management capabilities, such as workload balancing and virtual machine (VM) placement. DRS also enforces user-defined resource allocation policies at the cluster level while working with system-level constraints.

We can specify VM/Host anti-affinity rules, DRS tries to schedule the TKG cluster worker nodes apart, not running on the same ESXi hosts.

Recommendation: DRS – partially automated

Note: With partially automated mode, DRS will handle the initial placement of virtual machines. However, any further migration recommendations will be surfaced up to the administrator to decide whether to move the virtual machine. The administrator can check the recommendation and may decide not to migrate the virtual machine. Recommendations should be for hosts on the same site.

Kafka

Kafka Compression

We recommend using a Kafka supported type of compression (GZIP, LZ4, Snappy, Zstandard). Using compression will greatly improve both throughput and latency as well as minimize the impact to other workloads residing on the same vSAN cluster.

Recommendation: Use compression

Kafka Partitions and Replication Factors

Kafka replicates the topic across the cluster. With Replication factor, you can achieve fault tolerance. Consider performance requirements and use only as many brokers as you need to achieve that level of performance. Environment and use case will dictate the optimal number of Kafka partitions and/or replication factors. We recommend consulting *Confluent Best Practices* to determine the number of partitions and/or replication factors that will meet your use case needs.

Recommendation: Refer to *Confluent Best Practices*

Network and Storage

Kafka performance demands a low latency network with high bandwidth. As with Kafka broker, do not attempt to put all brokers on a single node, as this would reduce availability. If storage in the container is not persistent, data will be lost after restart. EmptyDir is used for Kafka data and will persist if the container restarts. So if the container starts, the failing broker first has to replicate all the data, which is a time-consuming process. That is why you should use a persistence volume; Kubernetes will be more flexible in choosing another node after restart or relocation. SSD is best used to serve with high throughput.

Operator

The Operator is a method of packaging, deploying, and managing a Kubernetes application. Confluent Operator makes it easy to bootstrap the Kafka cluster in a minute. Using a confluent operator, you almost do not need any configuration for bootstrapping the cluster.

Conclusion

VMware vSphere with Tanzu provides an enterprise platform for both traditional applications as well as modern applications, so customers can run existing enterprise applications alongside containerized applications in a unified manner while maintaining application portability.

Deliver Developer-ready Infrastructure: IT teams can use the existing vSphere environments to deploy an enterprise-grade Kubernetes infrastructure at a rapid pace (within one hour) while enabling enterprise-class governance, reliability, and security. After this one-time setup, vSphere with Tanzu enables a simple, fast, and self-service provisioning of Tanzu Kubernetes clusters within a few minutes. Aligning DevOps teams and IT teams is critical to the success of modern application development; to bring efficiency, scale, and security to Kubernetes deployments and operations. vSphere with Tanzu brings agile cloud operations to the IT admin to enable this transition into the role of Cloud Admin or SRE by delivering agility in day-to-day IT operations related to Kubernetes infrastructure.

Scale Without Compromise: vSphere can scale your infrastructure to meet the demands of high-performance applications and memory intensive databases.

Simplify Operations: Simplified operations are delivered through key capabilities of vSphere 7, including elastic infrastructure for sharing resources, simplified lifecycle management, and intrinsic security across your hybrid cloud infrastructure.

References

- *VMware vSAN*
- *VMware vSphere with Tanzu*
- *VMware Cloud Native Storage in vSphere and vSAN*
- *Confluent Best Practices for Apache Kafka in Production*
- *Kafka Best Practices in Confluent Cloud*
- *Deploy Prometheus on Tanzu Kubernetes Clusters*
- *Dell EMC VxRail*
- *VMware vSphere*

Appendix: The Helm/Providers/private.yaml file from the Confluent Operator Fileset

See the Appendix section of *Running Modern Applications with VMware Cloud Foundation with Tanzu on Dell EMC VxRail* for detailed scripts. And you can edit the parameters as appropriate for your setting.

About the Author

Yimeng Liu, Solutions Architect in the Solutions Architecture team of the Cloud Platform Business Unit, wrote the original version of this paper.

The following reviewers also contributed to the paper contents:

- Vic Dery, Sr. Principal Engineer of VxRail Technical Marketing in Dell EMC
- Tony Foster, Principal Engineer of VxRail Technical Marketing in Dell EMC
- David Glynn, Sr. Principal Engineer of VxRail Technical Marketing in Dell EMC
- Jason Marques, Sr. Principal Engineer of VxRail Technical Marketing in Dell EMC
- Ka Kit Wong, Staff Solutions Architect in the Solutions Architecture team of the Cloud Platform Business Unit in VMware

- Myles Gray, Staff Technical Marketing Architect of the Cloud Platform Business Unit in VMware
- Rachel Zheng, Product Line Marketing Manager of the Cloud Platform Business Unit in VMware