Top Use Cases for VMware Tanzu Service Mesh, Built on VMware NSX

vmware®

Table of contents

Overview

Evolving architectures require enhanced tooling and infrastructure

Digital transformation drives the need for speed as companies come under increasing pressure to innovate faster with much of the transformation being achieved in software. Application developers are evolving to achieve faster development cycles, faster delivery times and more frequent deployments.

Consequently, application architectures are also evolving from monolithic to microservices, delivering flexible architectural choices, improved scale and availability, faster release cadence and easier maintenance. Application development velocity is further improved using containers and container orchestration from Kubernetes in a well-automated pipeline. Microservices and Kubernetes work well together, with Kubernetes providing the load balancing and microservices defining how each application component, running in its own pod, interacts with the others.

Microservices communicate considerably among themselves, and the network is the glue that brings microservices together to deliver an app. Adding this network dependency adds a new layer of application complexity. Latency introduced between services in the service chain affects the entire application and the user experience. Troubleshooting and identifying a root cause is more complex in a distributed microservices-based application because such applications are composed of many different services, often written in different coding languages. The tooling and infrastructure required to quickly deploy distributed applications is rapidly maturing, but it's still missing capabilities to describe how services interact. A service mesh is the missing link for infrastructure architects and developers.

Introducing VMware Tanzu Service Mesh, built on VMware NSX

VMware Tanzu™ Service Mesh™, built VMware NSX®, is an abstraction layer that implements service-to-service communication (service discovery and encryption), observability (monitoring and tracing) and resiliency (circuit breakers and retries). It allows developers to focus on developing the special sauce in their apps for the business and frees them from housecleaning activities that do not differentiate the business.

For example, a web server microservice needs to fetch data and present it. But it also requires a lot of housecleaning work, such as discovering other services, communicating with them securely and with high reliability, and dealing with externalities such as high network latency.

A service mesh abstracts these housecleaning functions to a separate entity called a proxy. The proxy sits in front of each microservice and all communications are passed through it. The proxy is responsible for handling connections, traffic management, errors and failures, and collecting metrics for observability purposes. When proxies talk with other proxies, you get a service mesh. In Kubernetes, the proxy is implemented as a sidecar. The containers run in pods, where the sidecars act as helper containers to the main container, which runs the business logic. In a service mesh, the proxy runs as a sidecar.

This white paper highlights the capabilities and use cases for a service mesh environment implemented as a configurable abstraction layer that can save a lot of time and money for the business.

Definitions and taxonomy

The following definitions provide an overview of service mesh concepts. For a more comprehensive overview, please refer to our *glossary of cloud native terms*.

Cloud infrastructure – The servers, virtual machines (VMs), storage, networking and other components required for cloud computing and infrastructure as a service. Cloud infrastructure provides the building blocks, or primitives, for creating hybrid and private clouds that deliver cloud computing services.

Cloud native applications – Apps developed and optimized to run in a cloud as distributed applications. Cloud native apps (aka modern apps) are:

- Containerized for reproducibility and resource isolation
- Orchestrated to optimize resource utilization
- Segmented into microservices to ease modification, maintenance and scalability

Cluster – Three or more interconnected VMs or physical machines that, in effect, form a single system. An application running on a cluster is typically a distributed application because it runs on multiple nodes. Clusters provide high availability, fault tolerance and scalability.

Container – A portable, executable image for packaging an application with all its dependencies and instructions on how to run it. The image is executed to run as a process with its own isolated application, file system and networking. Containers provide a portable, flexible way of packaging, distributing, modifying, testing and running apps, speeding up software development and deployment.

Containers as a service – A service offering that helps developers build, deploy and manage containerized applications, typically using Kubernetes.

Global namespaces (GNS) – The primary management construct in Tanzu Service Mesh that supports services running on more than one Kubernetes cluster. While a global namespace in Kubernetes is tied to a Kubernetes cluster, Tanzu Service Mesh elevates the GNS from the physical world, with each service mesh GNS managing its own service discovery, observability, encryption, policies and service-level agreements (SLAs).

Ingress – A Kubernetes API object that controls external access to services in a Kubernetes cluster, such as HTTP and HTTPS. Ingress can perform load balancing.

Istio – A platform that deploys a service mesh to connect, manage and secure microservices on Kubernetes. Istio intercepts network communications among the microservices that make up a containerized application deployed on Kubernetes to manage the microservices.

Kubernetes – An orchestration system that automates deployment and management of containerized apps. As an application and its services run in containers on a distributed cluster, Kubernetes orchestrates all moving pieces to optimize computing resources, maintain the desired state and scale on demand.

Microservices – A microservices architecture breaks up the functions of an app into a set of small, discrete, decentralized, goal-oriented processes, each of which can be independently developed, tested, deployed, replaced and scaled.

Multi-cloud – A cloud computing approach that combines several cloud providers, platforms or services in one environment. A multi-cloud strategy reduces reliance on a single vendor, protects cloud services from outages, helps tailor the architecture to needs and provides flexibility to switch solutions.

NSX – A VMware product that provides software-defined network (SDN) virtualization.

Orchestration – Because it can automatically deploy, manage and scale a containerized application, Kubernetes is often referred to as an orchestration framework or an orchestration engine. It orchestrates resource utilization, failure handling, availability, configuration, desired state and scalability.

Overlay network – An SDN component that rides on the underlay to provide networking, such as IP addresses and ports, for the container and host lifecycles. Overlays can isolate communication among apps that use the same physical network. Overlay technologies include Flannel, Calico and NSX.

Pivotal Cloud Foundry (PCF) – A private platform as a service for developing and deploying cloud native applications.

Pod – The smallest deployable Kubernetes unit in which one or more containers can be managed; you run a container image in a pod. A set of pods typically wraps a container, its storage resources, IP address and other options up into an instance of an application that will run on Kubernetes.

Service – A Kubernetes API object that describes how to access applications, such as a set of pods, by using methods such as ports or load balancers. A service may also be a microservice within the context of some larger application. An HTTP server, for example, is a service.

Service discovery – Automatically detects the dynamically assigned networking information of the microservices or the devices on which they are running.

Service mesh – When a containerized application is built as a collection of services or microservices, it forms a mesh of services. A service mesh creates a layer above IP addresses and ports to connect the services and manage their interactions. A service mesh might deliver, for instance, load balancing, monitoring and service-to-service authentication. Examples of technologies that provide a service mesh are Istio and Linkerd.

VMware® PKS – An enterprise Kubernetes platform.

Underlay network – The underlay connects machines (virtual or physical) by using either a hardware- or software-based approach to networking.

YAML – A human-readable data serialization standard commonly used in configuration files to structure information and commands. In Kubernetes, specification files are written in YAML.

Tanzu Service Mesh architecture

Tanzu Service Mesh is an abstraction layer for multiple data plane service meshes. The solution applies service mesh concepts—such as traffic control, security and observability—not to a single Kubernetes cluster or cloud, but across Kubernetes clusters, clouds and third-party service meshes. The architecture is constructed of a local Istio data plane with its own local control plane and a global control plane, which is managed by Tanzu Service Mesh.

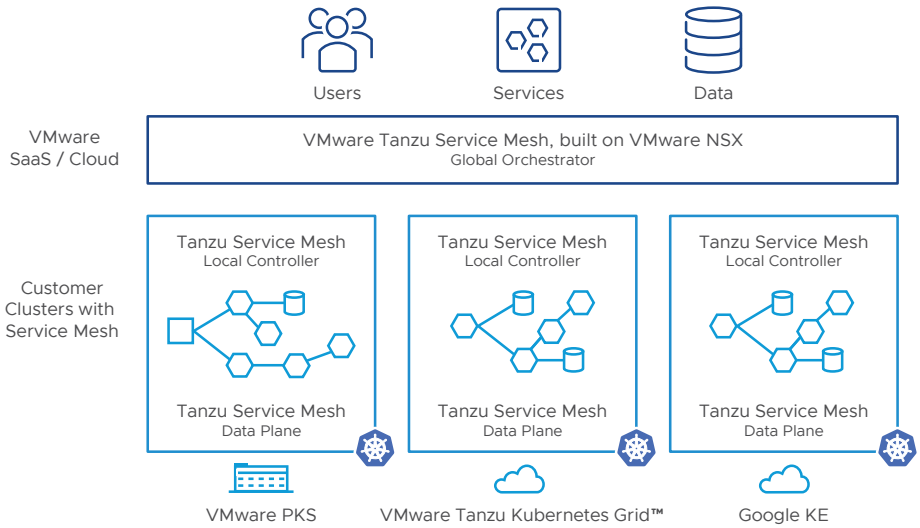

FIGURE 1: The Tanzu Service Mesh architecture.

Tanzu Service Mesh uses an Istio data plane abstraction for Kubernetes workloads. Istio is typically tied to a single Kubernetes cluster because Istio users prefer each cluster to be able to operate independently from other Kubernetes clusters. Tanzu Service Mesh acts as a global control plane for many data plane Istio deployments, managing the lifecycle of Istio from onboarding to Day 2 and Day 3 operations.

Tanzu Service Mesh only handles the lifecycle of the service mesh (Istio, in this case); it does not handle the Kubernetes lifecycle. When onboarding a new cluster on Tanzu Service Mesh, the service deploys a curated version of Istio signed and supported by VMware. This Istio deployment is the same as the upstream Istio in every way, but it also includes an agent that communicates with the Tanzu Service Mesh global control plane. Istio installation is not the most intuitive, but the onboarding process of Tanzu Service Mesh simplifies the process significantly.

Use case scenarios

The Tanzu Service Mesh platform addresses a multitude of connectivity and security use cases in hybrid cloud environments.

Multi-cloud and hybrid cloud patterns

It is increasingly common for organizations to deploy an application in multiple Kubernetes clusters and multiple clouds, whether on premises, public or hybrid. Cloud providers typically provide their own unique solution for traffic routing, load balancing, remediation and monitoring. End users accessing such applications can experience reduced SLAs due to application and service unavailability, and slowness due to overloaded services and cluster failures. On the other hand, DevOps and site reliability engineers need a unified and automated way to build, run and manage these applications, and configure various service-level objective (SLO)/SLA and failover policies for the applications. Continuous monitoring frameworks are also essential so failures can be quickly detected and remediated.

With Tanzu Service Mesh, you can apply policies such as security and traffic management at the GNS level. It also provides the observability needed to operate and troubleshoot your applications, and act on any SLO policy violations.

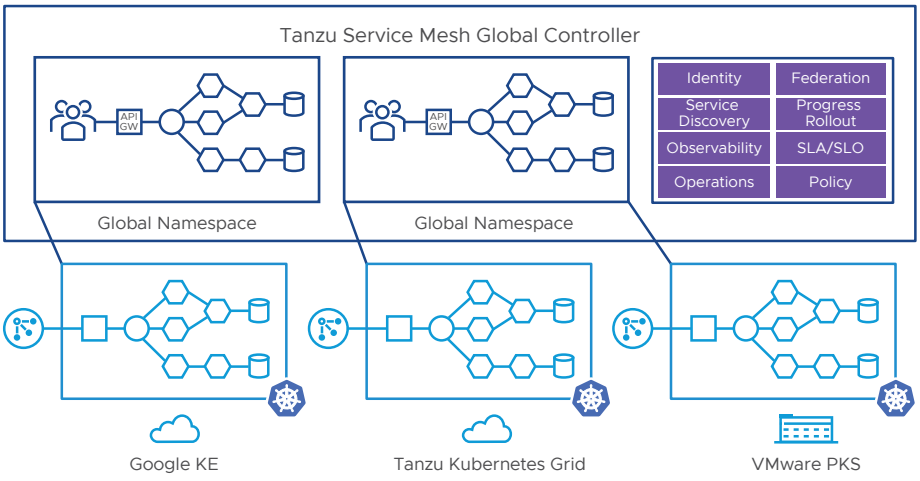

FIGURE 2: Applying policies at the GNS level with Tanzu Service Mesh.

Business continuity

The general practice for business continuity with Kubernetes-deployed apps and cloud native apps is to deploy the app in more than one cluster—either in the same region for high availability or a remote region for disaster recovery—with a load balancer between them to direct traffic to both clusters, usually in an active-active configuration. In Tanzu Service Mesh, you can group these clusters into a GNS, integrate them with load balancing (usually global load balancing), and program it automatically to redirect traffic in case of failure. Tanzu Service Mesh can visualize the GNS configuration and traffic flows between the clusters for faster detection of health issues in the cluster.

End-to-end mutual transport layer security

Achieving end-to-end encryption for in-flight traffic is not easy but, in many cases, it's required for regulatory purposes. This requires a top-level certificate authority (CA) that will provide a trusted identity to each node on the network. In the case of microservices architecture, those nodes are the pods that run the services—and there are a lot of them. Istio can set up end-to-end mutual transport layer security (mTLS) encryption utilizing the CA function called Citadel. Citadel will manage the certificates for the services and automatically rotate them every 90 days.

Tanzu Service Mesh builds on this capability in Istio and expands it across multiple clusters and clouds—and even service meshes. This is achieved by either setting up a root CA for multiple Citadel implementations that can either be run on the Tanzu Service Mesh service, or by utilizing the customer's CA and making the Istio CAs the intermediary CAs. If it's encrypting traffic with third-party CAs, it will utilize mechanisms in the Scytale initiative. The implementation of end-to-end encryption of in-flight traffic is applied at the GNS level, where you can apply an encryption policy that also supports different settings, such as faster certificate rotation than the default setting in Istio, or setting up permissive and restrictive policies.

Rolling upgrades at scale

When upgrading microservices through a canary upgrade or blue-green deployment, you introduce a new version of a service side by side with the old one, and then move a small percentage of traffic to the new version to monitor it for errors and latency. If the new service is working fine, you then move some more traffic over until 100 percent of your users are using the new version, and then decommission the old version of the service.

This type of upgrade can be performed relatively easily with Istio using a simple YAML file that defines the routing rules to send a percentage of traffic to the new service (split traffic). However, performing dozens or even hundreds of such upgrades a day is challenging. How do you manage these upgrades, monitor them and make changes at scale? Tanzu Service Mesh provides an easy way to manage multiple rolling upgrades from a single console and automatically manage it across Istio deployments. By defining a single or multiple service upgrade for a GNS, you can define the rules that determine how much traffic to shift and the steps to take, as well as what to do in case of errors or failures. You can then monitor all your upgrades from a single dashboard.

Predicting end-to-end response time

Achieving application SLAs in a distributed architecture can be very complex and challenging. If latency goes up in your application because of a service load that goes up in the chain, just autoscaling it out (deploying more instances of it) may cause an adverse effect on the entire application due to the ripple effect (one service scale creates pressure on downstream services). It's even more complicated when you're doing this across multiple clouds because you need to consider cross-cloud latencies and be able to look at the health of all the services in the application service chain (see Figure 3).

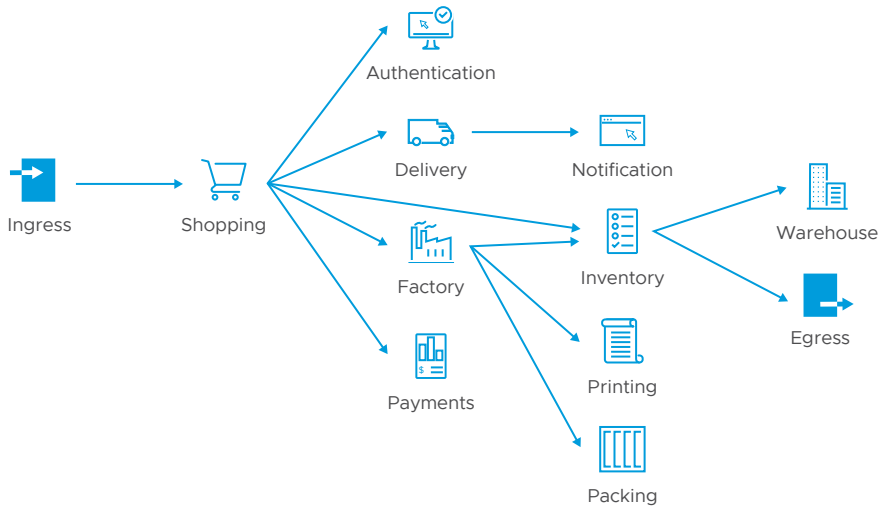

FIGURE 3: Service chain map example.

Tanzu Service Mesh allows you to assign an end-to-end latency SLA policy to an application, and to automatically optimize and self-heal distributed microservices applications to achieve the SLA.

Summary

Service mesh technology provides the abstraction layer that enables service-to-service communications, observability and application resiliency. Using service meshes, a developer can focus on creating business value through their applications, and liberate themselves from having to hard-code networking and security functionality into their services. VMware Tanzu Service Mesh, built on VMware NSX, provides the abstraction for multiple data plane service meshes and has become a critical component in distributed microservices architectures.

Additional resources

Ready to get started? The following list of materials and tutorials can help you enhance your understanding of Istio and Tanzu Service Mesh.

Online resources

- Istio
  - Documentation: *istio.io/docs*
  - GitHub: *github.com/istio/istio*
- VMware
  - Tanzu Service Mesh product page: *cloud.vmware.com/nsx-service-mesh*
  - Open source service mesh interoperation: *blogs.vmware.com/networkvirtualization/2019/08/new-open-source-service-mesh-interoperation-collaboration*
- Others
  - Simplify Hybrid Deployments with Tanzu Service Mesh and Google Cloud Services: *youtu.be/4iYaF4nBM_o*

Conferences and meetups

Join a local service mesh Meetup group: *meetup.com/topics/service-mesh*

Engage in a dialogue with the Istio user community: *meetup.com/topics/istio*

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 vmware.com Item No: 435646aq-wp-tanzu-svc-mesh-a4 3/20