VMware Tanzu Service Mesh Concepts

VMware Tanzu Service Mesh

Contents

About Tanzu Service Mesh Concepts

1

The VMware Tanzu Service Mesh Concepts documentation provides conceptual information for VMware Tanzu™ Service Mesh™, built on VMware NSX®.

Intended Audience

This information is intended for the following audiences:

- DevOps engineers, site reliability engineers (SREs), and platform operators who want to use Tanzu Service Mesh to deploy and manage cloud-native microservices applications across clusters, clouds, and platforms.
- Application developers and service owners who want to use Tanzu Service Mesh to accelerate application development.
- Security engineers, SecOps engineers, and compliance owners who want to use Tanzu Service Mesh to configure user- and data-centric security policies for their microservices applications and benefit from security analytics and compliance capabilities of the product.

The information is written for platform operators and application developers who have a basic understanding of Kubernetes and are familiar with container deployment concepts and service mesh concepts.

This chapter includes the following topics:

Challenges with Application Transformation

Enterprises are moving from monolithic applications to distributed microservice architectures for flexible architectural choices, improved scale and availability, faster release cadence, and easier maintenance. Despite these benefits, this transformation poses serious challenges.

Microservices are the next evolution in application architecture. A microservices architecture offers the following benefits:

- Choice of technology. In microservices architectures, you can develop services using different programming languages and tools. Teams can choose the most appropriate technology.
- Independent workflow and full autonomy. Microservices give your team control over the full stack they require to deliver a feature. This reduces the required amount of coordination with other teams.
- Independent scalability. You can scale each service according to its workload demands and performance needs.
- Easy upgrades and rollback. With small, independent services, you can upgrade only the services that require the update. You also have the option of performing a rolling upgrade for one service at a time and/or for one team at a time. If a feature only requires a change to a single microservice, that microservice can be rolled back without affecting the workflows of other teams.
- Independent and frequent releases. Microservices limit the scope of changes and reduce the amount of coordination required between teams. Teams can release according to their own schedules instead of being bound to a single release cadence.

Moving to a microservices architecture, however, poses two sets of challenges.

Challenges with a Distributed System

Decomposing a monolithic application into microservices results in a distributed system. Application and platform teams now need to manage different aspects of communication between many discrete services, including the following activities:

- Establishing and maintaining operational visibility into the state of the services.
- Connecting, routing, load balancing, and securing communications across distributed microservices.
- Reducing latency between the services in the service chain, which can ripple across the entire application and affect the user experience.
- Troubleshooting and identifying the root cause of problems in an application composed of many different services written in different programming languages.

Multicloud Challenges

Enterprises are increasingly moving their applications to the cloud and commonly choose to deploy their applications on multiple platforms or clouds. Multiplatform and multicloud application deployments create operational silos. Additionally, they frequently use different APIs, which adds to the complexity.

Enterprises have to spend significant resources to accommodate the different requirements for managing services depending on where workloads are running. This undermines the value of microservices.

A service mesh solves both the distributed system and multicloud challenges.

How a Service Mesh Addresses Challenges in Microservices

A service mesh addresses challenges associated with a microservices architecture. However, the service mesh itself introduces new challenges.

Challenges a Service Mesh Addresses

A service mesh is an abstraction layer on top of a microservices application that provides the following capabilities:

- Service-to-service communication (including service discovery and encryption)
- Observability (monitoring and tracing)
- Resiliency (circuit breaking and retries)
- Traffic management (routing, load balancing)
- Security (authorization, encryption)

Before service mesh, client libraries and application programming interface (API) gateways were used to address some of the issues introduced by a microservices architecture. These solutions, however, have challenges of their own.

A service mesh solves some of the challenges introduced by distributed microservices by abstracting necessary functions (service discovery, connection encryption, error and failure handling, and latency detection and response) to a separate entity called proxy.

The proxy sits in front of each microservice, and all inbound and outbound communications flow through it. The proxy provides the functions noted above and metrics for observability purposes.

Challenges a Basic Service Mesh Does Not Address

Although most service mesh implementations prove effective in connecting services and securing service-to-service communication, because of inherent limitations, they don't address these challenges:

- Limited scope. Service meshes focus on services alone. They do not extend the scope from service-to-service communication to users-to-service-to-data communication. However, application flows are not limited to interservice communications. Basic service meshes do not process the full end-to-end requests because they flow from end users through the services, and on to the data.
- Challenges related to highly distributed heterogeneous platforms. In a multicloud, multiplatform environment, each cloud or platform has its own service mesh technology and uses different operational and security models. Basic service meshes do not provide the same capabilities (traffic management, security, and observability) across applications deployed in multiple clouds or platforms.

- Federation challenges. Basic service meshes do not deliver security, control, and observability across different administrative boundaries, technologies, and service meshes. Basic service meshes do not provide connectivity, control, observability, and security outside the service mesh.
- Autoscaling. Basic service meshes do not provide automatic scaling of services based on the resources usage or measured metrics on the system. However, Tanzu Service Mesh enriches standard service mesh functionality by offering service autoscaling. For more information about service autoscaling, see the Service Autoscaling with Tanzu Service Mesh User's Guide.
- Gauging performance quality. Basic service meshes do not come with an effective way of measuring how well an application is doing. Industry practice is now embracing service level objectives as an essential measure of performance quality of applications. For more information, see the Service Level Objectives with Tanzu Service Mesh documentation.

Tanzu Service Mesh is not a basic service mesh. As an enterprise-class service mesh, it solves these challenges and more.

Tanzu Service Mesh, an Enterprise-Class Service Mesh

Tanzu Service Mesh, an enterprise-class service mesh, solves the challenges associated with a distributed microservices application by extending service mesh services outside Kubernetes clusters and providing a unified operational layer across heterogeneous platforms and technologies, including virtual machines and other service meshes.

Most service mesh implementations focus on services alone, but this limited approach ignores users and data. Users use services to access data. If your application runs in a multicluster or multicloud environment, you still need to manage communication and access between users, services, and data centrally, without needing to manage the underlying physical infrastructure.

Tanzu Service Mesh elevates the service mesh from the physical boundaries and limitations of a single cluster and a single cloud. With Tanzu Service Mesh, you can control, measure, secure, scale, and operate applications, no matter where their components are deployed, in multiple clusters or multiple public clouds.

Tanzu Service Mesh provides service mesh capabilities for resources in a distributed application by arranging these objects in a logical group called global namespace. A global namespace is not tied to a single cluster and connects resources between two or more clusters. Each global namespace manages service discovery, observability, encryption, policies, and service-level objectives (SLOs) for its objects regardless of where they reside: in multiple clusters, sites, or clouds. For more information about global namespaces, see Global Namespaces.

Overview of Tanzu Service Mesh

VMware Tanzu™ Service Mesh™, built on VMware NSX®, is an enterprise-class service mesh solution that provides reliable control and security for microservices, end users, and data across all your clusters and clouds in the most demanding multicluster and multicloud environments.

Tanzu Service Mesh runs on multiple application platforms, public clouds, and runtime environments, including Kubernetes clusters.

To control application traffic, Tanzu Service Mesh provides fine-grained, traffic management policies that give you complete control and visibility into how traffic and API calls flow between your services and across clusters and clouds.

To secure communication between services and protect sensitive data, you can use Tanzu Service Mesh to implement a zero-trust security model for cloud-based applications.

You can measure application performance with a configurable service level objective (SLO) definition. For more information, see the Service Level Objectives with Tanzu Service Mesh documentation. As application demands change, you can autoscale services to maintain SLOs using Tanzu Service Mesh Service Autoscaler. For more information, see the Service Autoscaling with Tanzu Service Mesh User's Guide.

Tanzu Service Mesh supports cross-cluster and cross-cloud use cases with global namespaces. With global namespaces, you can securely deploy applications across clusters and clouds and have consistent traffic management policies, application continuity, and security policies across cloud silos and boundaries, regardless of where the applications are running.

Global namespaces can be each considered to mark an application boundary and as such provide strongly isolated environments for application teams and business units managing different applications and data.

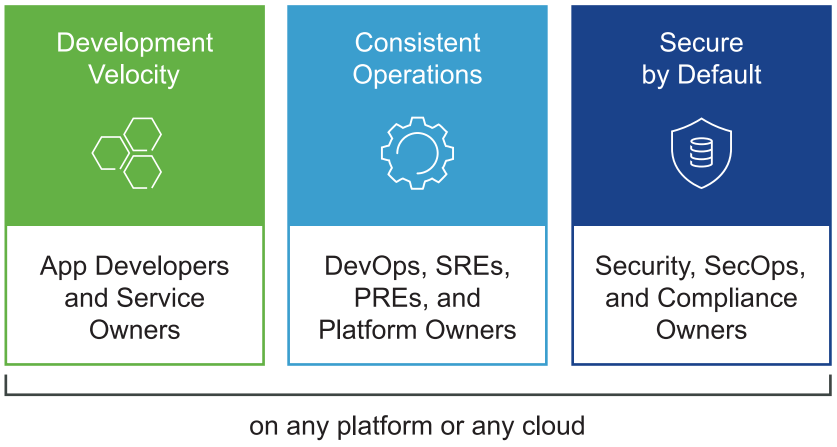
Benefits for Different Enterprise Teams

Tanzu Service Mesh helps platform and application teams efficiently connect and secure their microservices across multiple clusters and clouds.

Tanzu Service Mesh helps DevOps engineers, site reliability engineers (SREs), platform reliability engineers (PREs), and platform operators and achieve consistent operational control, policies, application performance, and visibility across multiple application platforms and public clouds.

Security engineers, SecOps engineers, and compliance owners can use the product to configure flexible business-relevant security and compliance policies that protect and monitor applications and data and enable compliance by default.

Application developers and service owners can use Tanzu Service Mesh to deploy applications more quickly and add features more easily to provide more value and better experiences for application users.


Development Velocity
App Developers and Service Owners
Consistent Operations
DevOps, SREs, PREs, and Platform Owners
Secure by Default
Security, SecOps, and Compliance Owners
on any platform or any cloud

Key Highlights of Tanzu Service Mesh

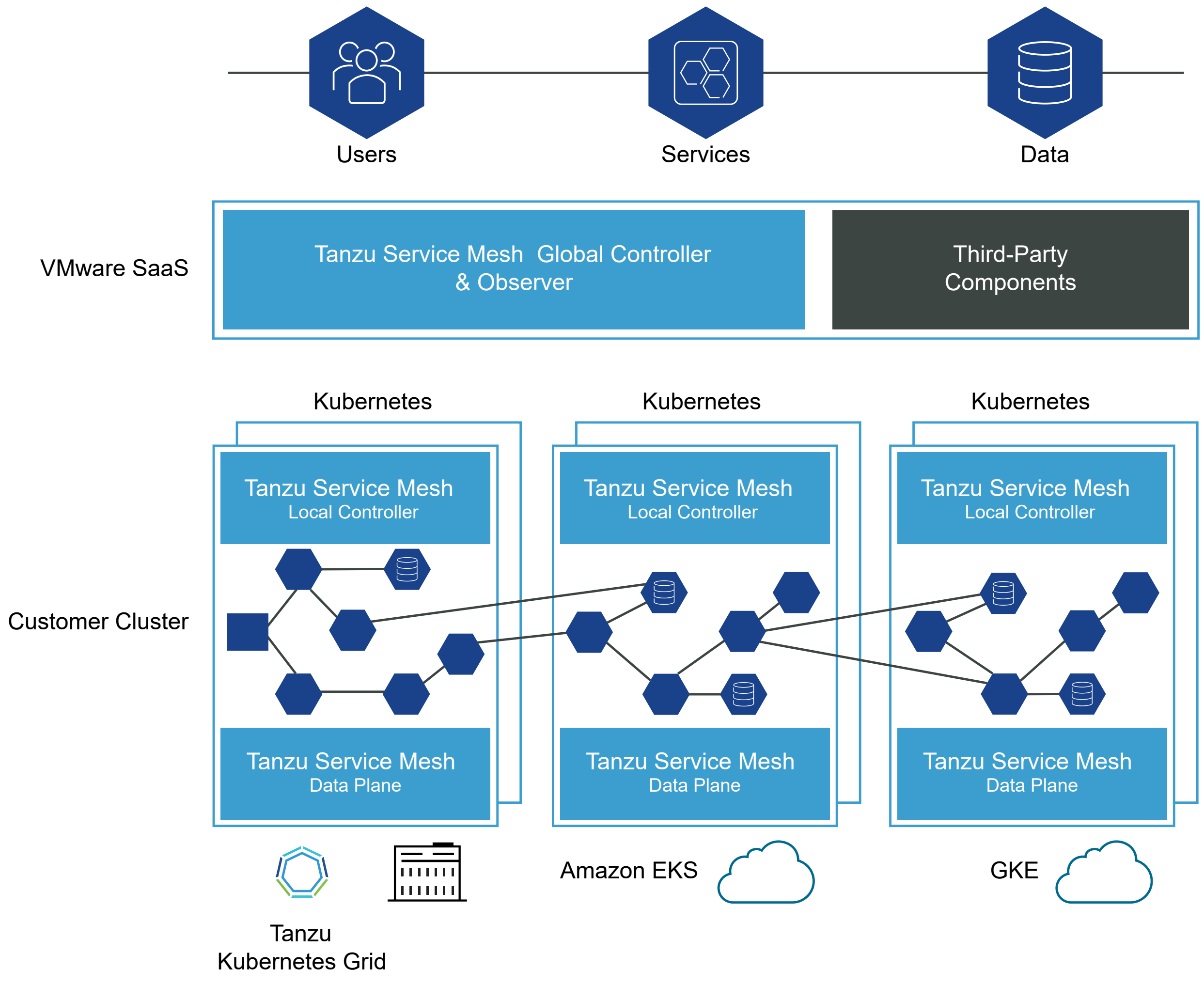
With Tanzu Service Mesh, application and platform teams get an enterprise-grade service mesh that they can use to accomplish the following tasks:

- Extends the service mesh capability (discovery, connectivity, security, and observability) across multiple clusters and clouds.
- Supports the development and management of distributed microservices applications across multiple clusters, multiple clouds, and in hybrid-cloud environments with Global Namespaces.
- Controls and observes traffic and API calls between services.
- Implements consistent application-layer security policies across clusters and clouds.
- Provides a consistent management model (connectivity, security, observability) for applications.
- Measures application performance by tracking service level objectives (SLOs) based on service mesh metrics. For more information, see the Service Level Objectives with Tanzu Service Mesh documentation.
- Provides features to manage application deployments, upgrades, A/B and canary testing, and security patches.
- Integrates with automated pipelines and workflows.
- Supports third-party Kubernetes platforms (such as Amazon EKS).
- Integrates with VMware Tanzu™ Mission Control™ , VMware® Enterprise PKS, and VMware Tanzu™ Kubernetes Grid™ to provide a seamless user experience.
- Includes Tanzu Service Mesh Service Autoscaler, a service mesh-based autoscaler. For more information, see the Service Autoscaling with Tanzu Service Mesh User's Guide.

Architecture of Tanzu Service Mesh

Tanzu Service Mesh offers a global controller and observer that platform teams can use to connect and protect microservices across all of their clusters and clouds in the most complex enterprise architectures. At a high level, Tanzu Service Mesh has the following architecture:

- Global Controller and Observer. A collection of microservices that run in VMware SaaS and deliver differentiated control, security, visibility, and autoscaling capabilities.
- Local Controller. Local control-plane components running in each customer cluster on-premises or in the public cloud and delivers fault tolerance in the event that a cluster becomes disconnected from the Global Controller.
- Sidecars. Data-plane components that run in each customer cluster on-premises or in the public cloud and handle east-west traffic inside the service mesh.
- Ingress and Egress Gateways. Data-plane components that run in each customer cluster on-premises or in the public cloud and handle north-south traffic going in and out of the service mesh.


Users
Services
Data
VMware SaaS
Tanzu Service Mesh Global Controller & Observer
Third-Party Components
Kubernetes
Kubernetes
Kubernetes
Customer Cluster
Tanzu Service Mesh Local Controller
Tanzu Service Mesh Local Controller
Tanzu Service Mesh Local Controller
Tanzu Service Mesh Data Plane
Tanzu Service Mesh Data Plane
Tanzu Service Mesh Data Plane
Tanzu Kubernetes Grid
Amazon EKS
GKE

Key Concepts

The *VMware Tanzu Service Mesh Concepts* documentation provides conceptual information about VMware Tanzu™ Service Mesh™, built on VMware NSX®. In this section, we explore the key concepts, features, and capabilities that make Tanzu Service Mesh an enterprise-class service mesh solution.

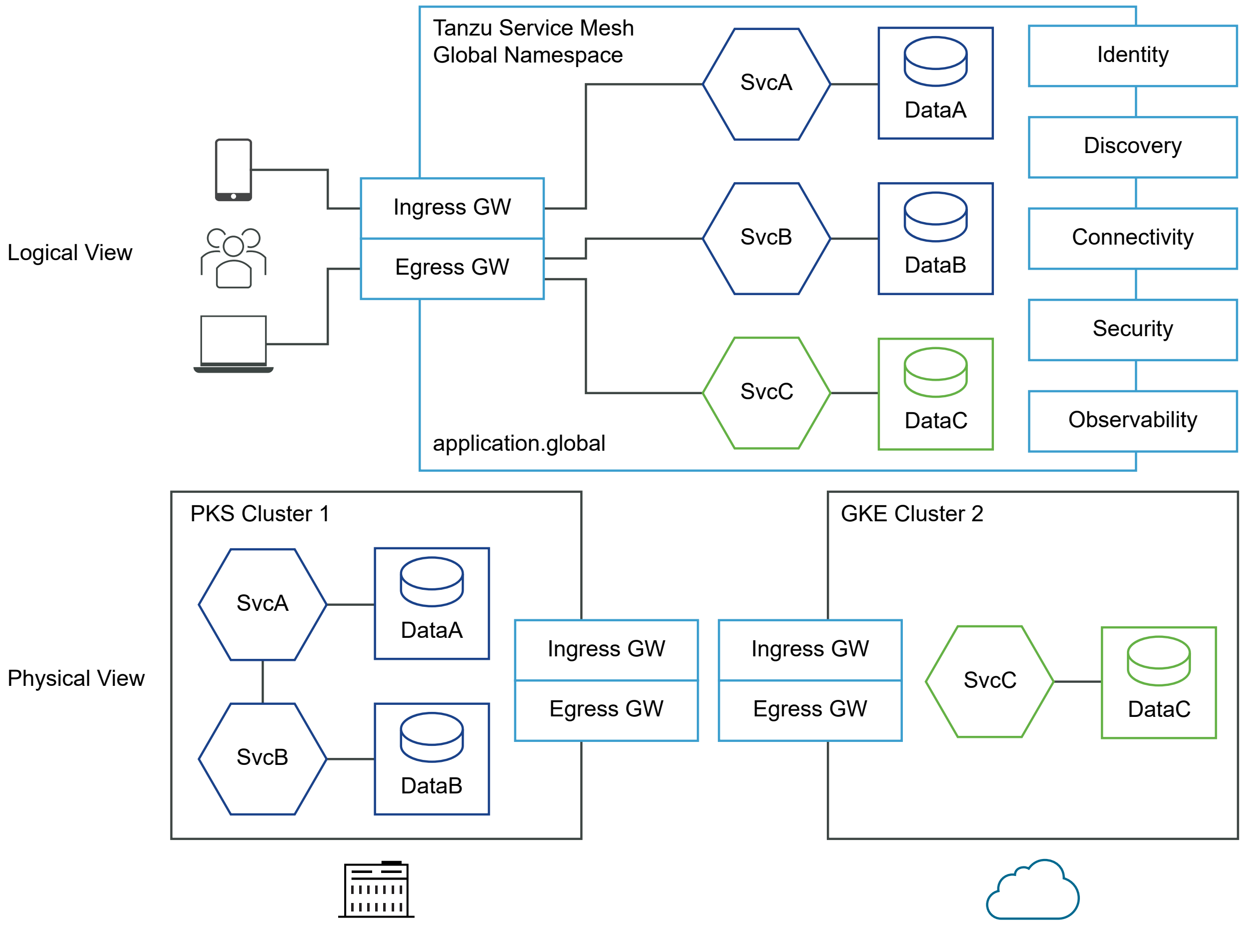
Global Namespaces

Global namespace, a unique concept in Tanzu Service Mesh, defines an application boundary. A global namespace connects the resources and workloads that make up the application into one virtual unit to provide consistent traffic routing, connectivity, resiliency, and security for applications across multiple clusters and clouds. Each global namespace is an isolated domain that provides automatic service discovery and manages service identities within that global namespace.

A global namespace manages the following functions for all resources that are part of it regardless of where they are located:

- Identity. The global namespace defines how a resource can be identified and authenticated to another. Each global namespace has its own certificate authority (CA) that provisions identities for the resources inside that global namespace.
- Discovery (DNS). The global namespace controls how one resource can locate another and provides a registry.
- Connectivity. The global namespace defines how communication can be established between resources and how traffic within the global namespace and external to the global namespace is routed between resources.
- Security. The global namespace manages security for its resources. In particular, the global namespace can enforce that all traffic between the resources is encrypted using Mutual Transport Layer Security authentication (mTLS). For more information, see the section *mTLS Encryption in Global Namespaces*.
- Observability. Tanzu Service Mesh aggregates telemetry data, such as metrics for services, clusters, and nodes, inside the global namespace. Tanzu Service Mesh provides a GraphQL endpoint through which you can query telemetry data. Additionally, you can distribute telemetry data to external endpoints using plugins on your clusters (for example, Wavefront Adapter for Istio).

The following diagram represents the global namespace concept and other pieces in a high-level architectural view. The components of one application are distributed in two different Kubernetes clusters: one of them is on premises and the other in a public cloud. The global namespace creates a logical view of these application components and provides a set of basic services for the components.



Global namespaces also offer isolated environments that can be assigned to different teams that manage different applications and data. Teams can use global namespaces to separate services between development, test, or production environments.

Global namespaces decouple your applications from the underlying infrastructure of the container-orchestration system (such as Kubernetes), while providing consistent traffic management and security policies, regardless of where your workloads are running. You can move applications across environments more easily with global namespaces. You can have as many global namespaces as you need, completely decoupled from the underlying clusters.

mTLS Encryption in Global Namespaces

Application operators and security engineers can use global namespaces to encrypt all HTTP and gRPC traffic that travels between services from one or more clusters inside a global namespace, using mTLS. This helps protect data in flight for all traffic within the service mesh for data protection and compliance needs.

Since a global namespace can include services from one or more clusters, Tanzu Service Mesh enforces mTLS end-to-end encryption of all traffic that travels between services within the same cluster or from services in one cluster to services in a different cluster.

You can specify an encryption mode for all global namespaces within your service mesh:

- Strict. Only encrypted traffic is allowed (HTTPS only).
- Permissive. Plain-text (HTTP) and encrypted (HTTPS) traffic is allowed.
- Off. Only plain-text (HTTP) traffic is allowed.

Traffic between global namespaces is always encrypted regardless of the mode.

From the service details page for a specific service, you can see whether mTLS encryption is enabled for that service and which encryption mode is used. The service graph in the console shows whether mTLS is configured for the application running in the global namespace.

For more information about how Tanzu Service Mesh supports the use case of encrypting in-flight data, see Encryption for Data in Flight.

Resource Groups

Resource group is a collection of cluster resources of a specific type that share specific characteristics. Resource groups help enforce policies and monitor the performance of resources in a single global namespace or across your entire organization.

A resource group provides a set of filters, or conditions, for you to retrieve and manage a specific subset of resources for your needs. For example, a resource group can define a condition for the service name to begin with *cart*. Another example is a resource group that defines a condition for nodes that run only a specific service. Only resources that satisfy the conditions defined for a resource group are included in that group. You can have multiple conditions in a resource group, combining them with the AND operator.

You can create these kinds of resource group to collect and manage objects that meet your specific criteria:

- Service groups
- Node groups

Resources groups in Tanzu Service Mesh are used for two main purposes.

You use resource groups to observe performance metrics for an appropriate collection of resources. For example, you want to monitor the performance of all nodes running in your California-based clusters or the performance of only the pricing services within your service mesh. For this purpose, you create a node group and a service group defining the appropriate conditions (the cluster is located in California and the service name begins with *pricing* in our examples). After a resource group is created, you can monitor relevant performance metrics (such as requests per seconds, latency metrics, and error rate) for the group collectively from the resource group details page.

You can also use resource groups to create and enforce policies. For example, you work for a financial institution that has office locations in New York and Beijing. You have created a resource group for each of these locations. You have also created an authorization policy defining that the employees in the Beijing office cannot access the resources in the New York resource group, whereas the New York employees have access to these resources.

You can add service level objectives (SLOs) on a resource group. You can then use these SLOs to track violations over time. If a service in the resource group exceeds Service Level Indicator limits, the violation is clearly visible in the Tanzu Service Mesh Console user interface, and the value of the error budget is updated.

Clusters

Tanzu Service Mesh supports Kubernetes clusters. In Kubernetes, cluster is a collection of physical or virtual machines, called nodes, for running containerized applications. Tanzu Service Mesh provides reliable control and security for resources across multiple clusters.

Tanzu Service Mesh shows information about the infrastructure (clusters and nodes) on which the services in your mesh are running. For each cluster that you have onboarded (deployed) on Tanzu Service Mesh, the product provides all relevant information for you to efficiently observe and manage its resources.

You can see a summary table view of all clusters in your infrastructure and drill down into a specific cluster to view its details, including the following information:

- A summary of the cluster (number of nodes, total cores, total memory, disk capacity, cluster management platform (for example, Kubernetes), number of services, and number of service instances running on the cluster's nodes)
- A service topology graph showing the connected services on the cluster
- Charts showing key performance metrics for the services (such as requests per second, latency metrics, and error rate)
- Information about the services running on the cluster (including service version, service type, and number of instances of each service)
- Information about all service instances running on the cluster's nodes
- Details of the platform services (for example, Kubernetes)
- A summary view of the cluster nodes, including the percentage of central processing unit (CPU), memory, and disk space consumed by each node

For information about the concept of node, see Nodes.

Nodes

Node is a physical or virtual machine that runs one or more service instances (instances of a specific version of service code) on a cluster within a service mesh.

For each onboarded cluster, Tanzu Service Mesh provides all relevant information for you to efficiently manage its nodes and observe the performance of the nodes in your network.

Note Tanzu Service Mesh shows information only for the worker nodes in onboarded clusters.

You can see a summary table view of all nodes in the clusters in your infrastructure and drill down into a specific node to view its details, including the following information:

- A summary of the node (including total cores, total memory, and disk capacity allocated to the node, the number of service instances running on the node, and current central processing unit (CPU) and memory usage)
- Charts showing performance metrics for the node (such as memory usage, disk usage, and performance of disk input/output operations)
- Information about all service instances running on the cluster's nodes, including the service instance name, the service version, and key performance metrics for the instances (such as requests per second, request count, and error rate percentage)

You can gather nodes that meet your criteria into a node group (a type of resource group) to apply consistent policies to all the nodes in the group and monitor the performance for the group collectively. For more information about resource groups, see Resource Groups.

Services

A service is an abstract concept that encapsulates business logic that can be run in a distributed application. A service construct is made of service versions and service instances. Each service maps to multiple service versions, and each service version maps to multiple service instances.

For an illustration of the service hierarchy in Tanzu Service Mesh, see the diagram in the *How Does a Tanzu Service Mesh Service Relate to a Kubernetes Service?* section.

You can use service versions to manage the life cycle of changes within a service. From time to time, you may need to make changes to a service to improve or expand its functionality. To release the updates in your environment, you create new versions of the service. You need to configure an endpoint for each version and deploy the version through on your platform.

Multiple service versions are commonly used in testing scenarios (such as A/B testing) and canary rollouts.

How Does a Tanzu Service Mesh Service Relate to a Kubernetes Service?

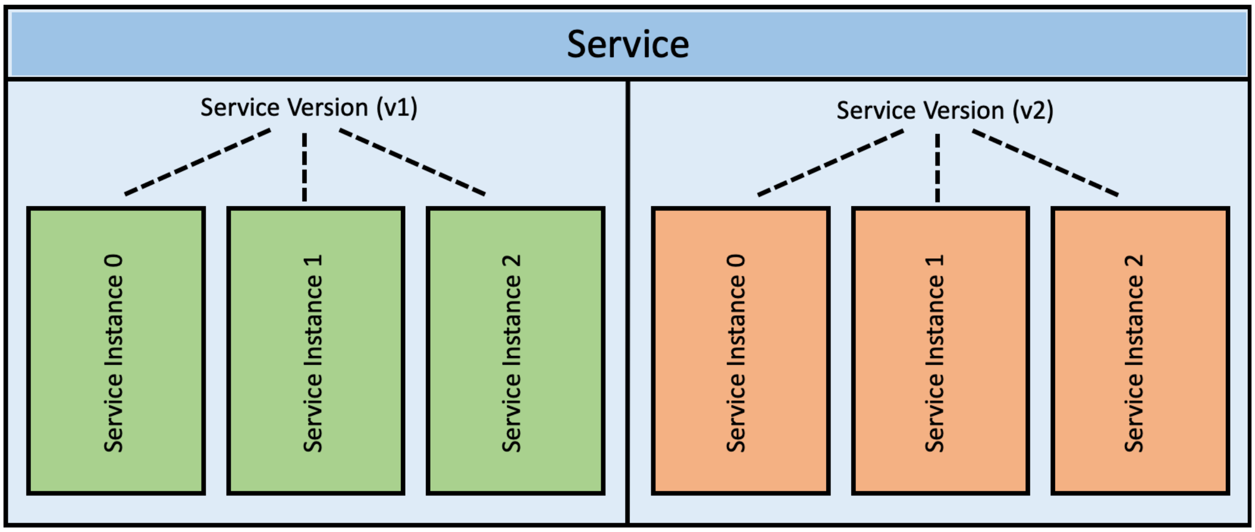
A Tanzu Service Mesh service can represent a business logic microservice where in Kubernetes it can be mapped to any of the following types of workload controllers: Deployments, ReplicaSets, StatefulSets, DaemonSet, Jobs, and CronJob.

In Tanzu Service Mesh, for services running on a Kubernetes cluster, service instances map to Kubernetes pods. Service instances are what get replicated when a service and service version scale horizontally.

In the following diagram, this could be, for example, a service called *frontend*, which is represented by Tanzu Service Mesh service as an abstract concept. The diagram shows two deployments : Service Version v1 and Service Version v2. Each version has three service instances in the Get All Accounts microservice.

Note In the future, the Tanzu Service Mesh definition of a service can include software that is run on virtual machines, as an example.

This diagram shows that Tanzu Service Mesh service is at the top of a hierarchy.



Public Services

A public service is a way to expose a service outside its global namespace to enable external end users to access the service.

Typically, you have an application deployed in a Global Namespaces and want to expose one or more services in the application outside the global namespace so that external clients can connect to it.

You configure a public service within the global namespace, including whether it is exposed as a secure service (over HTTPS) or an unsecure service (over HTTP).

In a typical scenario, you expose a service, for example, a service named *frontend*, to the public Internet so that end users can make requests to it from their devices. A user types the URL of the service, for example, `https://frontend.acme.com`, in his or her browser to access it and then makes a request to the service. Because making a service public is a security risk, carefully consider what services you want to expose.

You can use your public service in round-robin load balancing scenarios. For example, you add a public service to a global namespace that connects services from three clusters and deploy instances of that public service on each of those clusters. The round-robin load balancing algorithm rotates requests to the service among the instances on the different clusters.

You can also define health checks for a public service. In this scenario, Tanzu Service Mesh monitors the health of the service and eliminates unhealthy instances from the load balancing policy. When those unhealthy instances become healthy again, Tanzu Service Mesh resumes sending requests to them.

For information about creating public services, see Create a Public Service in the *Using Tanzu Service Mesh* documentation.

Key Use Cases

Tanzu Service Mesh supports a multitude of connectivity and security use cases that are relevant to enterprises running applications in multicloud and hybrid-cloud environments.

This section contains high-level descriptions of the following supported use cases:

- Multicloud application patterns
- Encryption of data in flight
- Automatic scaling of services
- Tracking application performance with service level objectives (SLOs)

Multicloud Application Patterns

With Tanzu Service Mesh, users can seamlessly deploy applications across multiple clusters and clouds and discover services automatically. Global namespaces (isolated domains in Tanzu Service Mesh) provide service identity management and automatic service discovery across different clusters.

In a multicloud world, applications and services are commonly deployed across multiple clusters and even on multiple clouds.

Organizations don't want to spend a lot of time on complex configurations and endless routing rules. They want to have a simple way to onboard and discover these applications to simplify the overall deployment. Tools such as traffic control, telemetry, and policy that are supported only in single-cluster environments need to be extended to multiple clusters. Users need a way to configure these constructs from a single pane of glass and in an automated way.

With global namespace is a unique concept introduced in Tanzu Service Mesh, operators can expand the boundaries of a Kubernetes namespace across multiple clusters and multiple clouds. You can consider each global namespace as an isolated domain where service discovery and identity are bound to the namespace. Applications in a global namespace can automatically discover services within the namespace.

With Tanzu Service Mesh, the platform operator can configure and deploy global namespaces. The operator can add services to a global namespace by mapping their Kubernetes namespaces to the global namespace.

With global namespaces, operators can create and characterize the behavior of services in terms of policy, end-to-end authentication, and service discovery. For more information about global namespaces, see Global Namespaces.

Encryption for Data in Flight

Achieving end-to-end encryption for in-flight traffic is mandatory to comply with regulatory and security governance measures. Tanzu Service Mesh provides this capability across multiple clusters, clouds, and even service meshes.

Tanzu Service Mesh includes a top-level certificate authority (CA) to 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. Tanzu Service Mesh can set up end-to-end mutual transport layer security (mTLS) encryption using a CA function. The CA manages the certificates for the services and automatically rotates them every 90 days.