Application Service Mesh

Service Overview

Issue	02
Date	2023-07-03

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Security Declaration

Vulnerability

Huawei's regulations on product vulnerability management are subject to the *Vul. Response Process.* For details about this process, visit the following web page:
https://www.huawei.com/en/psirt/vul-response-process
For vulnerability information, enterprise customers can visit the following web page:
https://securitybulletin.huawei.com/enterprise/en/security-advisory

Contents

1 Introduction

What Is Application Service Mesh?

Application Service Mesh (ASM) is a non-intrusive solution for you to manage microservice lifecycle and traffic. It is compatible with the Kubernetes and Istio ecosystems and hosts a wide range of features such as load balancing, outlier detection, and fault injection. It also provides diversified built-in grayscale releases, including canary release and blue-green deployment, for one-stop, automated release.

What Is Istio?

Istio is an open platform that provides connection, protection, control, and observation functions. By providing a complete non-intrusive microservice governance solution, Istio can well resolve service network governance issues such as cloud-native service management, network connection, and security management.

With the popularization of microservices, greater challenges emerge in the basic operations and advanced O&M of the distributed microservice architecture.

At a high level, Istio helps reduce the complexity of application deployments, and eases the strain on your development teams. It is a fully open-source service mesh that can be transparently layered onto existing distributed applications. It is also a platform, including APIs that let it integrate into any logging platform, or telemetry or policy system. Istio's diverse feature set lets you successfully and efficiently run a distributed microservice architecture, and provides a unified way to secure, connect, and monitor microservices.

Service Mesh

The term service mesh is used to describe the network of microservices that make up applications and the interactions between applications. As a service mesh grows in size and complexity, it can become harder to understand and manage. You need to take care of basic operations, such as service discovery, load balancing, failure recovery, metrics, and monitoring. Advanced O&M includes blue-green deployment, canary release, rate limiting, access control, and end-to-end authentication.

Why Use Istio?

Istio provides behavioral insights and operational control over the service mesh as a whole, offering a complete solution to satisfy the diverse requirements of microservice applications.

Kubernetes allows you to deploy and upgrade applications, and manage running traffic. However, capabilities such as outlier detection, tracing, and rate limiting are not supported. Istio, as an open platform built based on Kubernetes, provides complementary capabilities to Kubernetes in microservice governance.

Figure 1-1 Relationship between Istio and Kubernetes

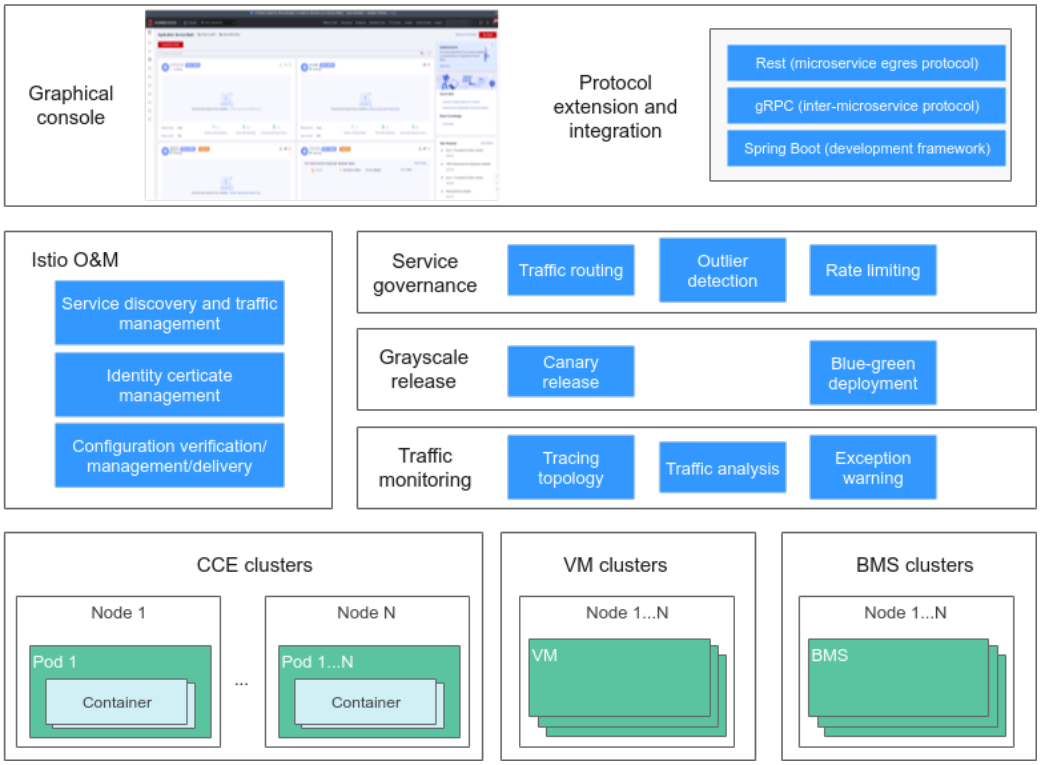
You add Istio support to services by deploying a special sidecar proxy throughout your environment that intercepts all networking requests between microservices, then configure and manage Istio using its control plane functionality, which includes:

- Automatic load balancing for HTTP, gRPC, WebSocket, and TCP traffic.
- Fine-grained control of traffic behavior with rich routing rules, retries, and fault injection.
- A pluggable policy layer and configuration API supporting access control, rate limits and quotas.
- Automatic metrics, logs, and traces for all traffic within a cluster, including cluster ingress and egress.
- Secure service-to-service communication in a cluster with strong identity-based authentication and authorization.

Istio aims to achieve scalability and meet various deployment requirements.

Product Architecture

Figure 1-2 Product architecture



Features

Grayscale release

- Grayscale policies based on request content: You can set criteria based on request content, such as header and cookie. Only requests meeting the criteria will be distributed to the grayscale version.
- Grayscale policies based on traffic ratio: You can set specific ratio for the traffic to be distributed to the grayscale version.
- Canary release: Guidance will be provided to help you perform canary release on a service, including rolling out a grayscale version, observing the running and traffic of the grayscale version, configuring grayscale release policies, and diverging the traffic.
- Blue-green deployment: Guidance will be provided to help you perform blue-green deployment on a service, including rolling out a grayscale version, observing the running of the grayscale version, observing the traffic, and switching the traffic.

Traffic management

- Layer-7 connection pool management: You can set the maximum number of HTTP requests, maximum number of retry times, maximum number of pending requests, maximum number of requests for each connection, and maximum connection idle period.

- Layer-4 connection pool management: You can set the maximum TCP connections, connection timeout duration, maximum non-responses, minimum idle period, and health check interval.
- Outlier detection: You can configure outlier detection rules, such as the number of consecutive errors allowed before a pod is evicted, check period, base ejection time, and maximum percentage of ejected pods.
- Retry: You can configure the number of HTTP retry times, retry timeout duration, and retry condition.
- Timeout: You can configure the HTTP request timeout duration.
- Load balancing: You can configure multiple load balancing policies, such as random, round robin, least connections, and consistent hashing.
- HTTP header: You can flexibly add, edit, and remove HTTP headers, including the operations on the HTTP headers before the request is forwarded to the destination service and before the response is returned to the client.
- Fault injection: You can configure delay and abort faults.

Security

- Peer authentication: Peer authentication defines how traffic reaches the current service pod through the tunnel (or not through the tunnel). Currently, three authentication policies are supported: **UNSET**, **PERMISSIVE**, and **STRICT**.
- Access authorization: Access authorization controls the access to services in the mesh and determines whether a request can be sent to the current service.

Observability

- Application access topology: An application access topology shows the dependencies between services.
- Service running monitoring: Service access information, including service information, different versions of the service, QPS, and latency can be monitored.
- Access logs: Service access logs can be collected and searched.
- Tracing: Non-intrusive tracing points. You can use the tracing data to demarcate and locate faults.

Framework of the mesh data plane

- Spring Cloud: supports unified management of services developed using Spring Cloud SDK.
- Dubbo: supports unified management of services developed using Dubbo SDK.

Compatibility and extension

- Community compatibility: ASM APIs are fully compatible with the Istio community.
- Support for community add-ons: Tracing, Prometheus, Kiali, and Grafana are supported.

2 Advantages

Ease of Use

The out-of-the-box usability allows you to use a service mesh without code rewrite or manual installation.

Built-in Canary Release and Blue-Green Deployment

- Deployment of the grayscale version and traffic switchover with a few clicks
- Configurable grayscale policy that can be set based on traffic ratio and request content (cookies, OSs, and browsers)
- One-stop health, performance, and traffic monitoring, achieving quantified, intelligent, and visualized grayscale release

Policy-based Intelligent Routing and Flexible Traffic Management

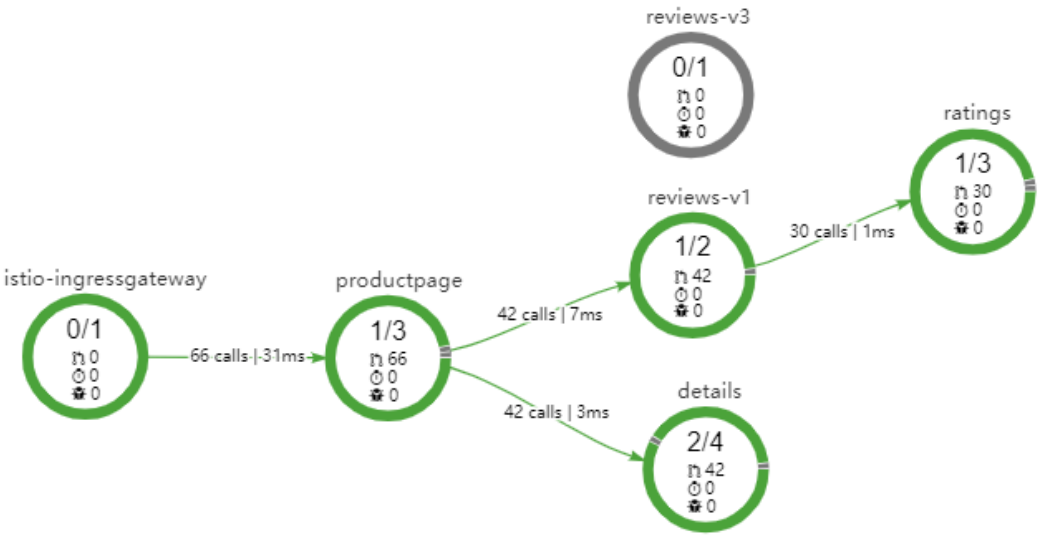
Load balancing, service routing, fault injection, and outlier detection policies can be intuitively configured. Microservice traffic management can be real-time, visualized, intelligent, and automated, requiring no modifications on your applications.

- Routing rules can be set based on weight and content to implement flexible grayscale release of applications.
- Load balancing achieves high availability for service processing.
- Outlier detection ensures stable and reliable links between services.
- Network persistent connection management saves resources and improves network throughput.
- Service security certification, authentication, and audit lay a solid foundation for service security assurance.

Graphical Application Topology and Visualized Traffic Management

ASM provides visualized traffic monitoring, which clearly displays information on tracing health status, abnormal responses, responses with long latency, and traffic status topology.

Figure 2-1 Monitoring traffic



ASM integrates **Application Operations Management (AOM)** to provide detailed traffic monitoring information on each microservice and report on abnormal responses and tracing information, helping you quickly and accurately locate faults.

Enhanced Performance and Reliability

The performance and reliability of the control plane and data plane are enhanced based on the community version.

Multi-infrastructure

An O&M-free hosting control plane is provided. Unified service governance, grayscale release, security, and service running monitoring capabilities are supported. Unified service discovery and management of multiple infrastructure resources such as containers and VMs are provided.

Protocol Extension

The HTTP, gRPC, TCP, TLS, and Dubbo protocols are supported.

Traditional SDK Integration

Integration solutions for traditional microservice SDKs such as Spring Cloud and Dubbo are provided. Services developed using traditional microservice SDKs can be migrated to cloud-native containers and mesh running environments without major code modification.

3 Application Scenarios

3.1 Grayscale Release

Application Scenarios

In traditional iterations, a new service version is directly released to all users at a time. This is risky, because once an online accident or bug occurs, the impact on users is great. It could take a long time to fix the issue. Sometimes, the version has to be rolled back, which severely affects user experience.

Grayscale release is a smooth iteration mode for version upgrade. During the upgrade, some users use the new version, while other users continue to use the old version. After the new version is stable and ready, it gradually takes over all the live traffic.

Product Benefits

ASM provides multiple grayscale release functions for application governance. It allows you to detect and fix issues at the early stage and ensure that the iteration goes smoothly and efficiently.

Product Advantages

- **Built-in grayscale release process**: Multiple typical grayscale release processes are provided based on fine-grained distribution rules. A grayscale release wizard is provided for you to conveniently perform grayscale release practices. When a service version processes traffic, you can create a grayscale version. After the grayscale version is rolled out successfully, you can configure grayscale policies and diverge live traffic to the grayscale version.
- **Flexible grayscale policies**: You can set grayscale policies based on traffic ratio or request content, such as the OS, browser, cookie, and header information in an HTTP request. After configuring grayscale policies, you can view the running status and access information on multiple online versions in real time. You can select a stable version and switch all live traffic to this version in one click.

3.2 Service Traffic Management

Application Scenarios

Traffic management entails a wide range of operations, including:

- Dynamically modifying load balancing policies for cross-service access, such as configuring consistent hashing to send traffic to specific service pods
- Distributing a certain proportion of traffic to a specific version of a service when the service has two online versions
- Protecting services, for example, limiting the number of concurrent connections and requests, and isolating faulty service pods
- Dynamically modifying the content of a service or simulating a service running fault

Product Benefits

No code refactoring is required when you use ASM to manage traffic.

Non-intrusive traffic management capabilities are provided based on Istio. Policy- and scenario-based network connection management is provided to suit different service protocols. Different management rules can be configured for different service APIs on the topology to meet your service requirements.

Product Advantages

- **Retry:** Auto retries upon service access failures improve the access quality and success rate. You can set the number of HTTP retry times, retry timeout duration, and retry condition.
- **Timeout:** Auto processing and quickly failure return upon service access timeout eliminate resource locking and request freezing. You can set the HTTP request timeout duration.
- **Connection pool management**: You can configure the maximum TCP connections, connection timeout duration, maximum non-responses, minimum idle period, and health check interval for layer-4 protocols, and configure the maximum number of HTTP requests, maximum number of retry times, maximum number of pending requests, maximum number of requests for each connection, and maximum connection idle period for layer-7 protocols. In this way, the failure of a service will not cascade and affect the entire application.
- **Outlier detection**: You can configure the number of consecutive errors allowed before pod eviction, eviction interval, minimum eviction time, and maximum eviction ratio as outlier detection. In this way, you can check the running status of service pods on a regular basis. If access exceptions occur frequently, the pod is marked as abnormal and isolated accordingly. No traffic will be distributed to it in a specific period of time. After the isolation time, requests will be distributed to the pod. If the pod still runs abnormally, it will be isolated for a longer time. This is how pod isolation and automatic fault recovery work.

- **Load balancing**: Diverse load balancing policies, such as random, round robin, and least connection, are provided. You can configure consistent hashing to send traffic to specific service pods.
- **HTTP header:** You can flexibly add, edit, and remove HTTP headers, including the operations on the HTTP headers before the request is forwarded to the destination service and before the response is returned to the client.
- **Fault injection**: Abort and delay faults can be injected to specified services to test their resilience. No code refactoring is required.

3.3 End-to-End Transparency and Security

Application Scenarios

Splitting traditional monolithic applications into microservices brings various benefits, including better flexibility, scalability, and reusability. The new security requirements microservices have are as follows:

- Traffic encryption is required to defend against man-in-the-middle attacks.
- TLS and fine-grained access control policies are required for flexible service access control.
- Audit tools are needed to determine who can do what at what time.

ASM provides a comprehensive security solution, including authentication policies, transparent TLS encryption, and authorization and audit tools, to address these requirements.

Product Benefits

- **Default security**: No modification is required on application code and architecture to ensure security.
- **In-depth defense**: ASM can integrate with existing security systems to provide comprehensive defense.
- **Zero-trust network**: The security solution is built assuming that all the network is untrusted.

Product Advantages

- **Non-intrusive security**: ASM provides service meshes as infrastructure with built-in security capabilities. It allows you to focus more on the development and O&M of your services. No code refactoring is required to ensure service access security. ASM provides a transparent, distributed security layer and underlying secure communication channels, which manage authentication, authorization, and encryption for service communication. ASM provides communication security between pods and services. Developers only need to focus on application-level security based on this security infrastructure layer.
- **Fine-grained authorization**: After authentication, access authorization between services can be managed. Authorization management can be performed on a specific service or a specific API of a service. For example, you can authorize all services in a specific namespace or only a specific service. The source service and destination service can be in different clusters. Pods of

the source service can be in different clusters. Pods of the destination service can be in different clusters.

3.4 Service Running Monitoring

Application Scenarios

Container-based infrastructure brings a series of new challenges. It is necessary to evaluate and enhance the performance of API endpoints and identify potential risks of infrastructure. Istio service mesh enables you to enhance API performance with no code refactoring and service delay.

Product Benefits

ASM generates detailed telemetry for all service communications within the mesh. It provides observability of service behaviors and allows operators to easily troubleshoot, maintain, and optimize their applications. With ASM, operators can better understand how services interact with other services and their components.

Product Advantages

- **Non-intrusive collection of monitoring data**: To manage and troubleshoot a complex application, it is important to obtain its access topology between services, traces, and monitoring data. ASM collects the monitoring data in a non-intrusive way. It allows you to focus on service development rather than the generation of monitoring data.
- **Flexible service running management**: You can view the health status and dependencies between services using the access data in a topology. The topology allows you to unfold a service and observe the running status of each version of the service and each pod of a version. This pod-level topology enables you to observe how outlier detection policies work, for example, how a pod is isolated and how it recovers. When the pod is isolated, no requests are distributed to it. After it runs normally, requests are automatically distributed to it again.

3.5 Integration with Traditional Microservice SDKs

Application Scenarios

- Using service meshes to manage services developed using traditional SDKs.
- Integrating Istio with microservice platforms and building a microservice management and control center based on Istio.

Product Benefits

Integration solutions for traditional microservice SDKs, such as Spring Cloud and Dubbo are provided. Services developed using traditional microservice SDKs can be migrated to cloud-native containers and meshes without major code modification.

Product Advantages

- **Non-intrusive migration solution**: ASM provides a non-intrusive solution to help you migrate services developed using traditional SDKs to service meshes. The service invoker diverts outbound traffic to the mesh data plane. That is, the service discovery and load balancer in the original SDK are short-circuited. The Kubernetes service name is used for access, and the service discovery in the Kubernetes is used. The governance logic in the SDK can be gradually replaced by the mesh. In this way, the service running and governance capabilities are provided directly by the infrastructure based on Kubernetes and service meshes.
- **Unified policy management**: The unified ASM control plane is used to discover services and manage governance rules. No independent registration center or configuration center is required. Service discovery, load balancing, and governance on the data plane are all performed using the data plane Envoy. SDKs only function as a lightweight application development framework.
- **Multiple infrastructure resources**: In the solution, the data plane can be deployed on containers or VMs. Services can be developed in various languages. There is no restriction on the development framework. Traffic rules are delivered through the mesh control plane and all forms of data planes are managed in a unified manner.

4 Constraints

Constraints on Clusters

Before creating a service mesh, ensure that you have an available cluster v1.15, v1.17, v1.19, v1.21, v1.23, v1.25, or v1.27. Secure containers in a CCE Turbo cluster cannot be added to a service mesh.

Containers on the node running Ubuntu 22.04 in a CCE Turbo cluster cannot be added to a service mesh earlier than v1.18.

- Ubuntu 22.04

Constraints on Service Meshes

When you use service meshes for service governance, a Deployment can only match with one service to avoid abnormal grayscale release, gateway access, or other functions.

5 Billing

Billing Items

ASM is billed by the number of pods managed in CCE clusters. Currently, the Basic mesh is provided for **free**, and a maximum of 200 pods can be managed.

- The price of an ASM package does not include the fees for using resources on Huawei Cloud, such as ECSs, CCE clusters, and ELBs.
  - For details about cluster creation fees, see **CCE Pricing Details**.
  - For details, see **Pricing Details**.

6 Permissions

To use ASM features, operate as a system administrator (including the RBAC-based permissions of clusters).

7 Basic Concepts

Workload

A workload is an abstract model of a group of pods in Kubernetes. Workloads defined in Kubernetes include Deployments, StatefulSets, jobs, and DaemonSets.

- **Deployment**: Pods of a Deployment are completely independent of each other and deliver the same functions. Deployments support auto scaling and rolling updates. Typical examples include Nginx and WordPress.
- **StatefulSet**: Pods in a StatefulSet are not completely independent of each other. StatefulSets have stable persistent storage and unique network identifiers. They support ordered, graceful deployment, scaling, and deletion. Typical examples include MySQL-HA and etcd.

Pod

A pod is the smallest and simplest unit in the Kubernetes object model that you create or deploy. A pod encapsulates an application container (in some cases, multiple containers), storage resources, a unique network IP address, and options that govern how the container should run.

Canary Release

Grayscale release is essential for smooth rollout of iterative software products in production environments. A certain proportion of production traffic on the live network is distributed to a new version of the application to test the version's performance and detect faults while ensuring stable running of the system.

Blue-Green Deployment

Blue-green deployment is a zero-downtime deployment mode. A new version of an application is deployed and tested in a production environment while the live environment continues to serve all production traffic. When you confirm that the new version is functioning properly, traffic is then distributed to the new version. At the same time, the old version is upgraded to the new version. Blue-green deployment allows you to quickly switch between the two versions to effectively prevent service disruption during the upgrade.

Traffic Management

Traffic management provides you with visualized network statuses of cloud native applications and allows you to manage and configure network connections and security policies online. Currently, it supports connection pool, outlier detection, load balancing, HTTP header, fault injection, etc.

Connection Pool Management

Thresholds for TCP and HTTP connections and request pools to prevent a service from overloading.

Outlier Detection

Quick response and service access fault isolation are configured to prevent a cascade of network and service calling faults. In this way, the fault impact is curbed. The overall system performance deterioration or avalanche is prevented.

Tracing Analysis

The service calling relationships in large-scale and complex distributed systems are traced to facilitate quick location and demarcation of faults.

8 Recommended Node Specifications

Recommended Specifications for Exclusive Nodes:

The performance of ASM is closely related to the cluster control plane (master nodes). Select appropriate node specifications based on your service requirements.

Total QPS (requests per second)	0-20,000	20,000-60,000
Specifications	8 vCPUs and 16 GB memory	16 vCPUs and 32 GB memory

NOTE

- The total QPS is the sum of requests of all services in all applications per second in a cluster.
- The function of recommending a proper amount of memory resources on the control plane based on the number of application service instances is coming soon.

ingressgateway Resource Consumption Reference

The resource consumed by each ingressgateway pod is affected by the connection type, number of connections, and QPS. For details, see the following tables:

Table 8-1 Memory consumption of persistent connections

Connections	Memory Usage (MB)
1	0.055
1,000	55
10,000	550

Table 8-2 CPU and memory usage of short connections

QPS	CPU Usage (m)	Memory Usage (MB)
100	30	100
1,000	300	100
10,000	3,000	150

The preceding data is for reference only. The actual resource consumption depends on the actual service model.

9 Related Services

Figure 9-1 shows the dependencies between ASM and other services.

Figure 9-1 Dependencies between ASM and other services

Dependencies Between ASM and Other Services

Table 9-1 Dependencies between ASM and other services

Service Name	Dependency	Related Features
Cloud Container Engine (CCE)	Cloud Container Engine (CCE) is a high-performance, high-reliability service through which enterprises can manage containerized applications. CCE supports native Kubernetes applications and tools, allowing you to easily set up a container runtime environment on the cloud. You can enable the service mesh function for CCE clusters to manage the services in the clusters.	Buying a CCE Cluster
Elastic Load Balance (ELB)	ELB automatically distributes access traffic to multiple cloud servers to balance the loads. It enhances an application's fault tolerance and service continuity. You can use ELB to access service meshes from outside.	Creating a Shared Load Balancer

