Un poco de Java y +

Otra forma de hablar de nuestro día a día…

CAAS, KUBERNETES, MICROSERVICIOS, MSA, QUÉ ES, SERVICE MESH, UN POCO DE

Un poco de Service Mesh

Fecha: 9 enero 2019 Autor/a: LuisMi Gracia 0 Comentarios
Si estás trabajando con microservicios (o simplemente leyendo sobre ellos) seguro que has oído el término "Service Mesh"(Malla de Servicios), ¿verdad?

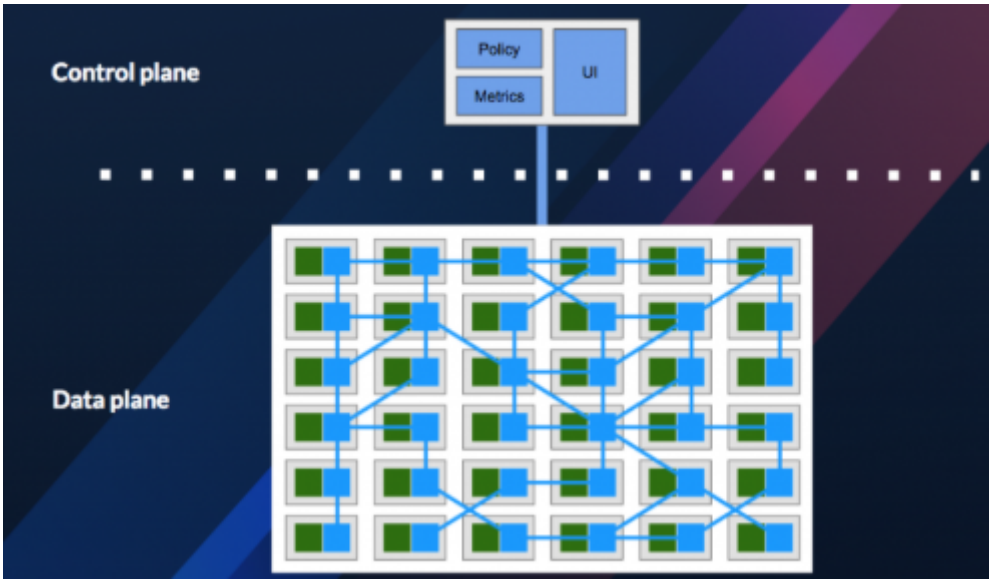
En este post queremos hacer una introducción del concepto y hacer un conjunto de cuestiones importantes a la hora de considerar adoptar un Service Mesh J

Finalmente comentaremos algunas opciones disponibles.

¿QUÉ ES UN SERVICE MESH?

Simplificando al máximo podríamos decir que un Service Mesh es **una capa de infraestructura que gestiona la comunicación entre servicios.** El detalle de esa gestión depende del producto, según el producto tendremos soporte para comunicación HTTP y TCP, control del throtling, seguridad,…

El diagrama típico para definir la arquitectura de un Service Mesh vemos:



Donde:

· Los cuadrados verdes en el Plano de Datos (Data Plane) representan servicios/aplicaciones

· Los cuadrados azules son Service Mesh Proxies

· Los rectángulos son Service Endpoints (POD, Host,…)

· El Plano de Control (Control Plane) ofrece un API centralizada para controlar el comportamiento de los Service Mesh Proxies

Las características típicas de un Service Mesh son:

· Resiliencia: reintentos, tiempos de espera, fechas límite, etc.

· Circuit Breaker pero prevenir fallos en cascada

· Algoritmos de balanceo de carga

· Enrutamiento de solicitudes

· Introducir y gestionar TLS entre endpoints

· Métricas para proporcionar instrumentación

Los Service Mesh son diferentes a las soluciones tradicionales tipo ESB, EAI, API Managemente ya que la infraestructura se encuentra fuera de las aplicaciones y es capaz de administrar todo el tráfico entrante y saliente en cada servicio. En lugar de codificar esa gestión de comunicación remota directamente en sus aplicaciones, se pueden utilizar una serie de proxies interconectados (o una "malla") donde esa lógica se puede desacoplar de sus aplicaciones y liberar la responsabilidad de los desarrolladores.

PREGUNTAS A CONSIDERAR:

Por supuesto, no hay una opción universalmente correcta para todos, pero es importante considerar estos tema:

¿Estoy listo para un Service Mesh?

Todos sabemos que el uso de cualquier herramienta nueva tiene un coste no despreciable (aprender a implementar sobre ella, usarla y mantenerla). En general podemos decir que un Service Mesh es necesario si administra una aplicación distribuida que realiza muchas llamadas de servicios, es decir una arquitectura de microservicios real, no una de esas de 5 servicios.

¿Qué problemas tengo en mi sistema distribuido? ¿cuáles son los puntos más problemáticos?

¿Se trata de las dependencias del servicio, la gestión de la seguridad o la gestión de la disponibilidad?

Si la respuesta a cualquiera de estas preguntas es sí, entonces un Service Mesh nos aplica.

. Recuerda los puntos de dolor que te duelen más, porque vas a necesitar eso en un momento.

¿Dónde se ejecuta su aplicación distribuida? ¿Qué plataformas tengo soportar?

¿Aparte de solo dar soporte a la plataforma de despliegue de contenedores necesitaré conectar otros servicios que pueden no estar en la malla de servicios?

Estas respuesta me llevarán a una solución (Istio pj) u otra (Linkerd pj)

¿Qué nivel de observabilidad tienen tus servicios hoy?

Puede que tu solución actual sea suficiente…

¿Qué funcionalidades de un Service Mesh tienes ya? ¿Cómo funcionará eso cuando introduzcas un Service Mesh?

¿Cómo es la división de responsabilidad en tus equipos? ¿Los equipos de desarrollo individuales esperan administrar sus propias configuraciones de proxy? ¿El control es mantenido por un equipo de operaciones de la plataforma central?

Sean cuales sea las responsabilidades organizativas, hay qye asegurar que el Service Mesh que considera le permite distribuir el control de una manera que tenga sentido para su organización.

¿Prefieres la funcionalidad centralizada o descentralizada?¿prefiere las implementaciones como un grupo de proxies basados en host o debería pasar por las complejidades potenciales del uso de modelos de implementación de sidecar?

¿Son los estándares típicos de soporte de la comunidad de código abierto suficientes para su equipo? ¿Necesita una opción de soporte comercial?

ALGUNOS SERVICE MESH OPEN-SOURCE

A continuación contaremos las principales capacidades de los 3 principales Service Mesh Open Source: Linkerd, Istio y Envoy.

Linkerd

Linkerd se lanzó como proyecto open-source en febrero de 2016, fue el primer producto en popularizar el término “Service Mesh”.

· Contruido sobre la librería Finagle de Twitter, escrito en Scala y ejecuta en la JVM

· Cubre las principales características de un Srvice Mesh

· Soporte para múltiples plataformas (Docker, Kubernetes, DC / OS, Amazon ECS o cualquier máquina independiente),

· Servicio incorporado de Discovery

· Soporte gRPC, HTTP / 2 y HTTP / 1.x + todo el tráfico TCP.

· Incluye Data Plane y Control Plane (Namerd)

· Soporte comercial

· Muchas empresas lo usan en producción

Envoy

Envoy se lanzó como proyecto open-source en octubre de 2016.

· Está escrito como un proxy de aplicación C ++ de alto rendimiento.

· Diseñado para ser utilizado como una capa de proxy independiente o para arquitecturas de malla de servicios.

· Varios contributors

· Cubre el Data Plane, necesita una pieza en el Control Plane (pj Istio) para cubrir todas las funcionalidades

· Actúa como un filtro L3 / L4 en su núcleo con muchos filtros L7 provistos fuera de la caja,

· Soporte para gRPC, y HTTP / 2

· API-driven: configuración dinámica, recargas en caliente,

· Foco en recopilación de métricas y la observabilidad de los servicios general.

· Soporte comercial

Istio

Istio se lanzó por primera vez en mayo de 2017 como una colaboración open-source entre Lyft, IBM, Google y con otros socios, como Red Hat, y muchos otros que se unieron poco después

· Istio está diseñado para proporcionar un Control Plane universal para administrar una variedad de servidores proxy subyacentes (se empareja con Envoy de forma predeterminada).

· Inicialmente, Istio se enfocó en los despliegues de Kubernetes, pero fue escrito desde el principio para ser independiente de la plataforma.

· El plano de control de Istio está destinado a ser extensible y está escrito en Go.

· Funciones de seguridad que incluyen gestión de identidad, administración de claves y RBAC,

· Soporte para gRPC, HTTP / 2, HTTP / 1.x, WebSockets y todo el tráfico TCP,

· Políticas como control cuota, del flujo,…

· Multiplataforma, despliegue híbrido.