Service Mesh using Istio and Consul

Deloitte

Authors:
Bharath Ramnath Sekar
Kaivalya Shah

Overview of Service Mesh

서비스 메시(Service Mesh)는 서비스 간 통신을 안전하고 빠르며 신뢰성 있게 만들기 위한 전용 인프라 레이어입니다. 네트워킹 기능을 제공하는 것 외에도 서비스 메시는 서비스 탐색(Service Discovery), 인증 및 권한 부여(Authentication and Authorization), 모니터링, 추적(Tracing), 그리고 트래픽 조절(Traffic Shaping)과 같은 다른 기능을 제공할 수 있습니다.

서비스 메시는 단순히 "서비스의 메시"가 아닙니다. 이것은 (마이크로)서비스가 네트워크를 완전히 추상화할 수 있도록 연결할 수 있는 API 프록시의 메시입니다.

서비스 메시 내에서 상호 연결된 프록시 집합은 데이터 플레인(Data Plane)이라고 불립니다. 반면, 서비스 메시 내에서 프록시 동작을 제어하고 관리하기 위해 사용되는 API와 도구 집합은 제어 플레인(Control Plane)으로 알려져 있습니다. 제어 플레인은 사용자가 정책을 지정하고 데이터 플레인을 전체적으로 구성하는 곳입니다.

데이터 플레인과 제어 플레인은 서비스 메시를 구현하는 데 모두 필요한 구성 요소이며, 이들은 함께 작동하여 효과적이고 효율적인 마이크로서비스 통신과 관리에 필요한 인프라를 제공합니다.

Challenges with Microservices

모놀리식 응용 프로그램이 분산된 마이크로서비스 아키텍처로 전환함에 따라 정적 인프라에서 동적 인프라로의 변화는 네트워킹 접근 방식을 호스트 기반에서 서비스 기반으로 변경합니다.
  연결성은 정적 IP 사용에서 동적 서비스 탐색으로 이동하고,
  보안은 정적 방화벽에서 서비스 식별으로 전환됩니다.
마이크로서비스와 같은 아키텍처 패턴은 팀이 서비스를 독립적으로 테스트하고 애플리케이션에 변경사항을 계속해서 제공할 수 있게 해주지만,

• 서비스 로드 밸런서는 동적 환경에서 비효율적입니다.

• 로드 밸런서는 종종 서비스 계층을 앞단에 두고 정적 IP를 제공하는 데 사용됩니다. 이러한 로드 밸런서는 비용을 증가시키고 지연을 높이며 단일 장애 지점을 도입하며 서비스 확장/축소에 따라 업데이트해야 합니다.

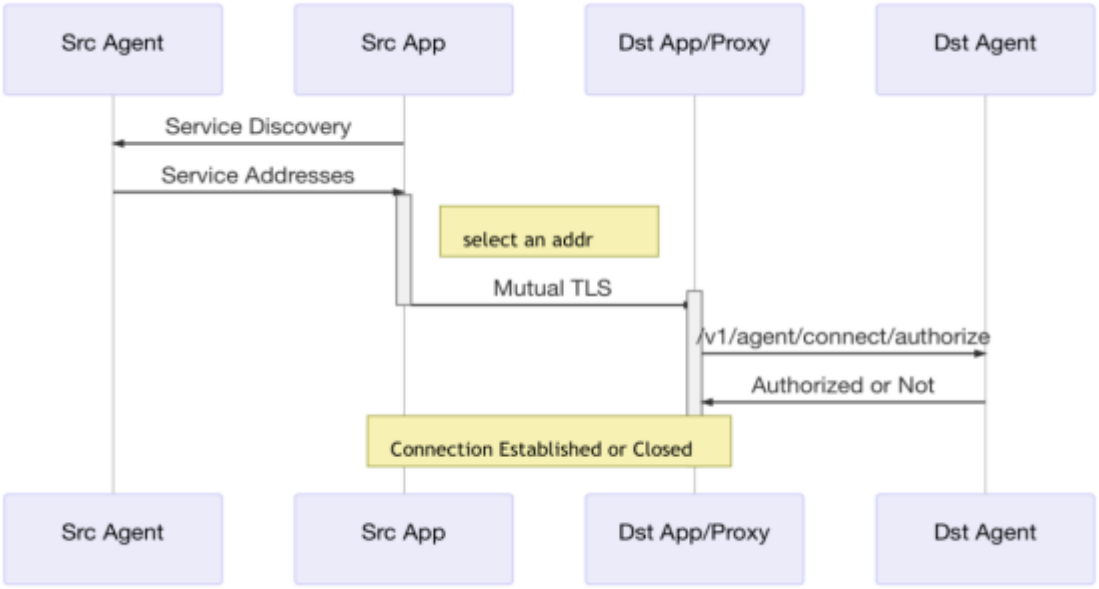
• 서비스 메시가 없으면 각 마이크로서비스는 서비스 간 통신을 조정하는 논리를 코드화해야 하므로 개발자들은 비즈니스 목표에 덜 집중합니다.

• 이는 또한 각 서비스 내에 존재하는 서비스 간 통신을 조정하는 논리가 숨겨져 있기 때문에 통신 오류를 진단하기가 더 어렵다는 것을 의미합니다.

Solution with Service Mesh

서비스 메시는 마이크로서비스 및 동적 클라우드 기반 인프라를 채택하는 조직에 필수입니다. 현대 런타임 환경의 고도로 동적인 특성을 수용하기 위해 전통적인 호스트 기반 네트워크 보안을 현대적인 서비스 기반 보안으로 대체해야 합니다. 서비스 메시는 다음 을 제공합니다:

- : Services must be able to find each other.
- : Services must accept runtime configuration from a central source.
- : Service communication must be authorized and encrypted.



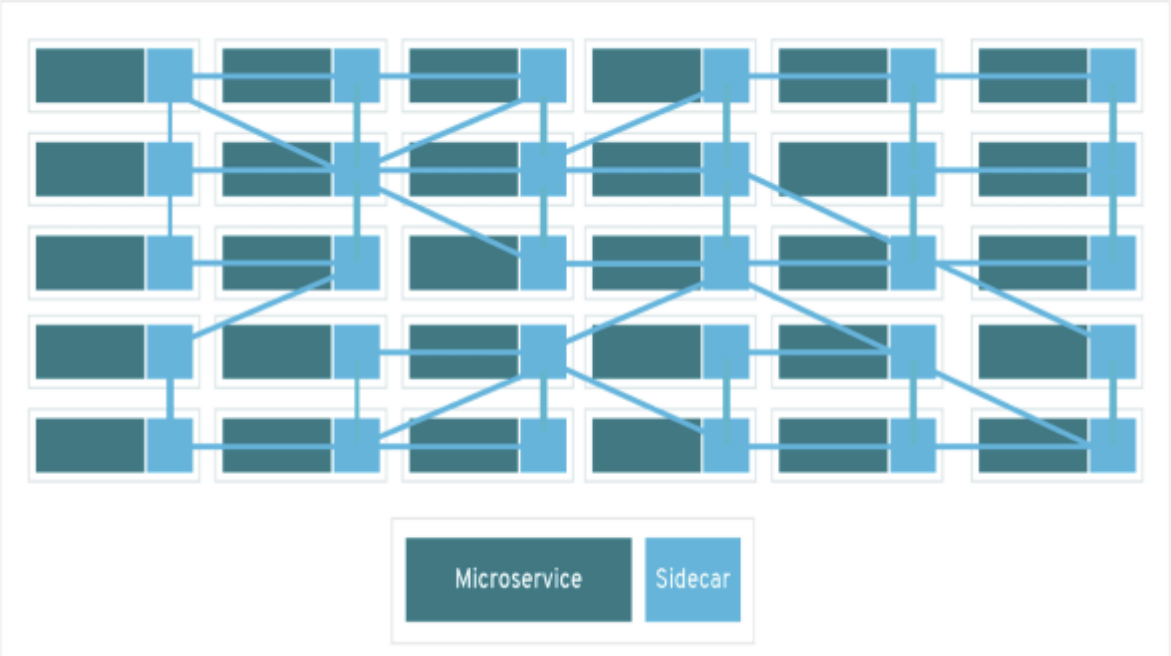
Sidecar Architecture

전형적인 서비스 메시에서는 다음과 같은 특징이 있습니다.

• 서비스 배포는 전용 "사이드카" 프록시를 포함하도록 수정됩니다.

• 요청은 각각의 인프라 레이어에 있는 프록시를 통해 마이크로서비스 간에 라우팅됩니다. 이러한 이유로 서비스 메시를 구성하는 개별 프록시는 종종 "사이드카"로 불립니다. 이는 이들이 서비스 내부가 아니라 옆에 실행되기 때문입니다.

• 이러한 "사이드카" 프록시들은 함께 결합되어 각 서비스에서 독립적으로 동작하며 메시 네트워크를 형성합니다. 서비스는 네트워크를 통해 다른 서비스를 직접 호출하는 대신, 로컬 사이드카 프록시를 호출하며, 이 프록시는 다시 서비스 간 교환의 복잡성을 캡슐화합니다.

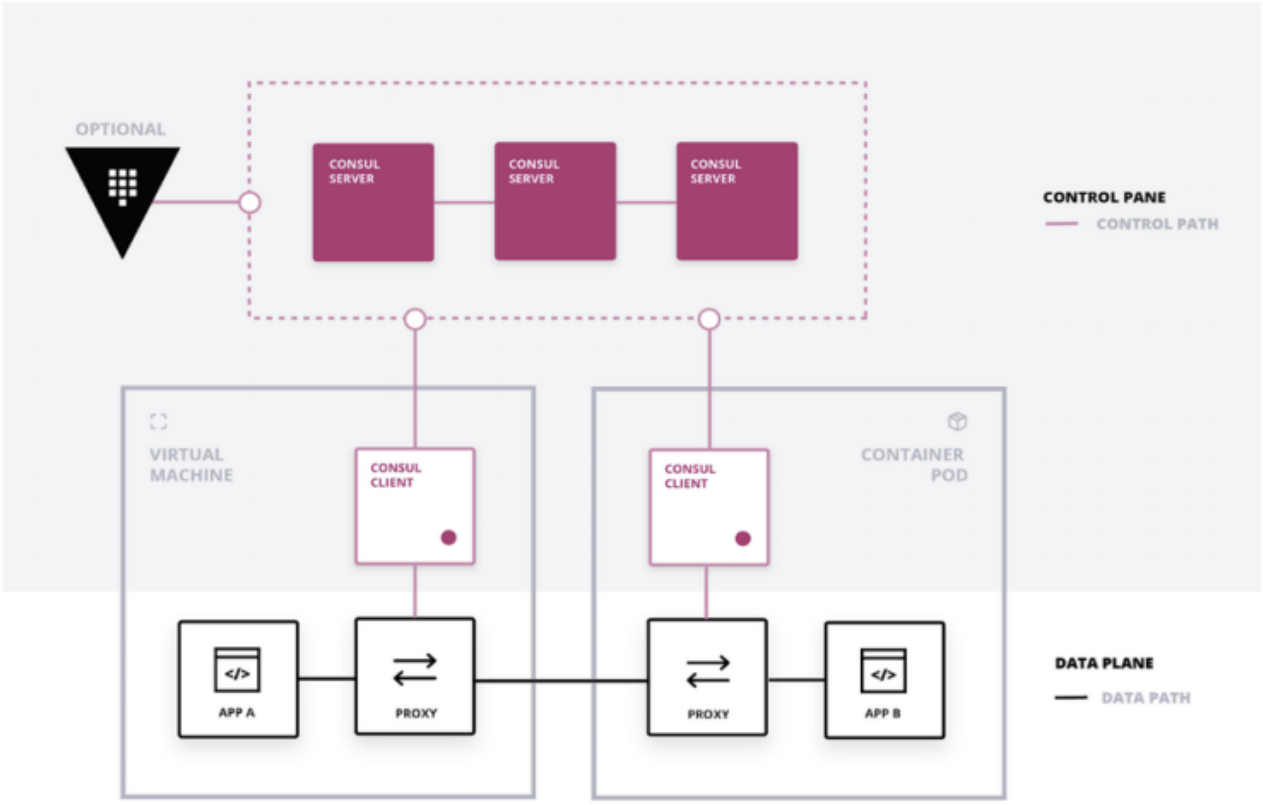


What is Consul

Consul은 서비스 탐색과 구성을 위한 도구입니다. Consul은 분산, 고가용성 및 매우 확장 가능한 특징을 가지고 있습니다.

- API 중심
- 어디서든 실행 및 연결
- 확장 및 통합

Consul Connect는 상호 TLS를 사용하여 서비스 간 연결 권한 및 암호화를 제공하는 서비스 메시 제어 플레인입니다. 서비스 탐색, 구성 및 세분화 기능이 포함된 완전한 기능의 제어 플레인은 필요에 따라 개별적으로 사용하거나 전체 서비스 메시를 구축하기 위해 함께 사용할 수 있습니다. Consul은 데이터 플레인을 필요로하며 프록시 및 네이티브 통합 모델을 모두 지원합니다. Consul은 기본적으로 모든 것이 기본적으로 작동하도록 간단한 내장 프록시와 함께 제공되지만 Envoy와 같은 제 3자 프록시 통합도 지원합니다.



What is Istio

Istio는 마이크로서비스를 통합하고, 마이크로서비스 간의 트래픽 흐름을 관리하며, 정책을 적용하고 텔레메트리 데이터를 집계하는 일관된 방법을 제공하는 오픈 플랫폼입니다.

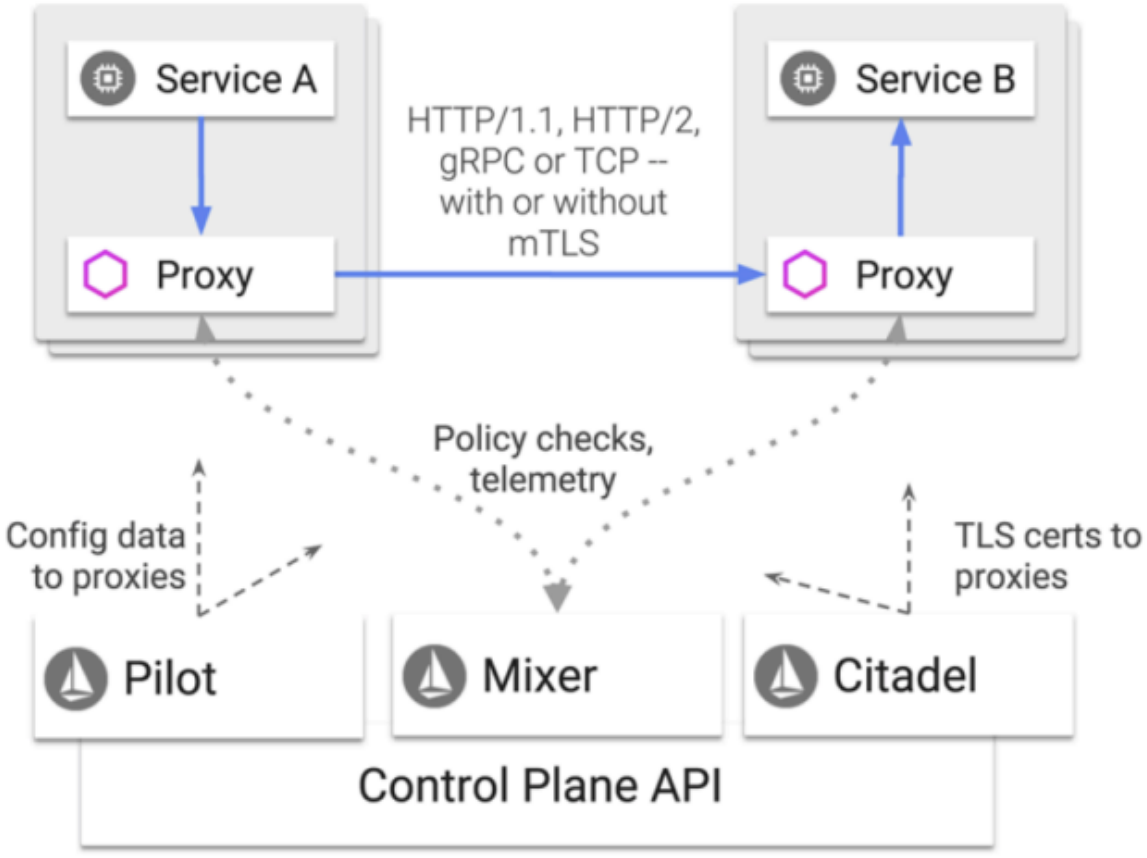
Istio의 제어 플레인은 Kubernetes, Mesos 등과 같은 기본 클러스터 관리 플랫폼 위에 추상화 레이어를 제공합니다. Istio 서비스 메시 제어 플레인에는 다음과 같은 Istio 구성 요소가 있습니다:

• Pilot — 사이드카 프록시를 구성하고 프로그래밍합니다.

• Mixer — 정책 결정을 수행하고 클러스터 내 모든 라우트 트래픽에 대한 자동 메트릭 및 로그를 제공합니다.

• Ingress — 클러스터 외부에서 들어오는 요청을 처리합니다.

• CA — 인증 기관(Certificate Authority).



Source: https://istio.io/docs/concepts/what-is-istio/

Why use Istio / Why use Consul

Why use Istio

Istio는 서비스 코드에 거의 또는 전혀 코드 변경이 필요하지 않은 상태로 로드 밸런싱, 서비스 간 인증, 모니터링 등이 포함된 배포된 서비스 네트워크를 쉽게 생성할 수 있도록 합니다.

서비스에 Istio 지원을 추가하려면 환경 전체에서 모든 마이크로서비스 간의 네트워크 통신을 가로채는 특별한 사이드카 프록시를 배포하여 Istio를 설치한 다음, 제어 플레인 기능을 사용하여 Istio를 구성하고 관리합니다. 이 제어 플레인 기능은 다음을 포함합니다:

• HTTP, gRPC, WebSocket 및 TCP 트래픽에 대한 자동 로드 밸런싱.

• 풍부한 라우팅 규칙, 재시도, 장애 복구 및 장애 삽입을 통한 트래픽 동작의 세세한 제어.

• 액세스 제어, 비율 제한 및 할당을 지원하는 플러그 가능한 정책 레이어 및 구성 API.

• 클러스터 내 모든 트래픽에 대한 자동 메트릭, 로그 및 추적, 클러스터 인그레스 및 엘레스를 포함합니다.

• 강력한 식별 기반 인증 및 권한을 사용하여 클러스터 내 안전한 서비스 간 통신.

또한 Istio 서비스 메시 제어 플레인은 다음을 처리합니다:

• HTTP 및 TCP 트래픽에 대한 자동 로드 밸런싱.

• 트래픽 동작 제어.

• 안전한 인증을 통한 클러스터 내 서비스 간 통신.

Why use Consul

Consul의 아키텍처는 매우 모듈화되어 있으며 다음과 같은 플러그 가능한 구성 요소가 있습니다:

• 데이터 플레인 (Native 또는 Sidecar)

• 인증서 관리

Consul은 서버 및 클라이언트 기능을 모두 제공하는 단일 바이너리(binary)로, 서비스 카탈로그, 구성, TLS 인증서, 권한 및 기타 모든 기능을 포함하고 있습니다. Consul을 사용하기 위해 별도의 시스템을 추가로 설치할 필요가 없습니다. Consul은 각 클러스터 노드에서 Consul 클라이언트를 실행하는 에이전트 기반 모델을 사용합니다.

각 클라이언트는 서버로부터 효율적으로 업데이트되는 로컬 캐시를 유지합니다. 그 결과로 모든 보안 서비스 통신 API는 마이크로초 내에 응답하며 외부 통신이 필요하지 않습니다. 응용 프로그램은 Connect 프로토콜과 네이티브로 통합할 수 있습니다. 결과적으로 Connect를 도입함으로써 발생하는 성능 오버헤드는 무시할 만큼 미

미합니다. 이러한 "Connect-native" 응용 프로그램은 프록시를 사용하거나 Connect-native인 경우와 관계없이 다른 Connect 호환 서비스와 상호 작용할 수 있습니다. Consul은 회전 지원이 포함된 자동 TLS 인증서 관리를 구현합니다. 대규모 Consul 클러스터 전체에서 리프 및 루트 인증서는 연결에 아무런 중단이 없도록 자동으로 회전할 수 있습니다. 인증서 관리 시스템은 Consul의 코드 변경을 통해 플러그 가능하게 구현할 수 있습니다.

Comparison of Istio and Consul

	Istio	Consul Connect
Model	Sidecar	Sidecar
Platform	Kubernetes	Any
language	Go	Go
Protocol	HTTP1.1 / HTTP2 / gRPC / TCP	TCP
Default Data	Plane Envoy (supports others)	Native (or Envoy)
Sidecar Injection	Yes	Yes
Encryption	Yes	Yes
Traffic Control	label/content based routing, traffic shifting	static upstream, prepared query, http api / dns with native integration
Resilience	timeouts, retries, connection pools, outlier detection	Pluggable
Prometheus Integration	Yes	Yes
Tracing Integration	Jaeger	Pluggable
Host to Host auth	Service Accounts	Consul ACL
Agent Caching	Yes	Yes
Secure connection outside cluster	No	Yes
Complexity	High	Low
Paid Support	No	Yes
Link	https://istio.io/	https://www.consul.io/intro/get - tingstarted/connect.html

Conclusion

서비스 메시는 클라우드 네이티브 스택의 중요한 구성 요소입니다. 서비스 메시는 마이크로서비스, 컨테이너, Kubernetes의 등장에 대응하여 개발자가 서비스를 더 잘 이해할 수 있는 인프라 레이어를 제공하기 위해 만들어졌습니다. 현재 사용 가능하고 곧 출시될 많은 서비스 메시 구현체로 인해 여러 가지 서비스 메시 기술을 동시에 사용하거나, 경제적으로 옳은 선택이거나 합리적으로 피할 수 없는 상황에 직면할 수 있습니다. 클라우드 네이티브 생태계에서 서비스 메시 채택은 급속하게 증가하고 있지만, 아직 탐험되지 않은 많은 확장 가능하고 흥미로운 로드맵이 남아 있습니다. 서비스 식별과 액세스 정책의 역할은 클라우드 네이티브 환경에서 아직 초기 단계이며, 서비스 메시가 이곳에서 중요한 역할을 할 수 있는 잠재력이 있습니다. 서비스 메시 인터페이스 (SMI)는 Kubernetes에서 실행되는 서비스 메시를 위한 명세서입니다. 다양한 공급 업체에서 구현할 수 있는 공통 표준을 정의합니다. 이로써 최종 사용자에 대한 표준화와 서비스 메시 기술 제공 업체에 대한 혁신이 모두 가능해집니다. 이것은 유연성과 상호 운용성을 가능하게 합니다. 최근 IDC 및 Gartner 보고서에 따르면 2020년까지 프로덕션 환경에서 마이크로서비스를 실행하는 모든 조직에게 서비스 메시가 필수가 될 것으로 예상됩니다.

- https://istio.io/docs/concepts/what-is-istio/
- https://learn.hashicorp.com/consul/getting-started/connect.html
- https://avinetworks.com/glossary/istio-service-mesh/
- https://apifriends.com/api-management/service-mesh/
- https://buoyant.io/2017/04/25/whats-a-service-mesh-and-why-do-i-need-one/
- https://medium.com/solo-io/the-need-for-a-standard-service-mesh-api-d89be65f8fb3
- https://www.consul.io/intro/vs/istio.html