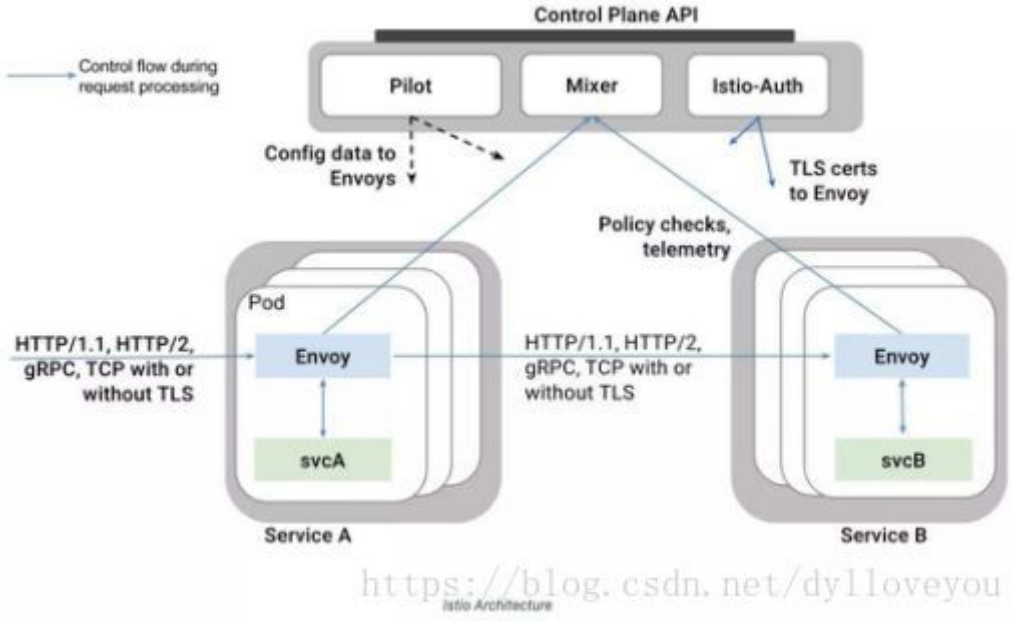
根据Istio官方文档的介绍，Istio在服务网络中主要提供了以下关键功能：

- 流量管理：控制服务之间的流量和API调用的流向，使得调用更可靠，并使网络在恶劣情况下更加健壮。
- 可观察性：了解服务之间的依赖关系，以及它们之间流量的本质和流向，从而提供快速识别问题的能力。
- 策略执行：将组织策略应用于服务之间的互动，确保访问策略得以执行，资源在消费者之间良好分配。策略的更改是通过配置网格而不是修改应用程序代码。
- 服务身份和安全：为网格中的服务提供可验证身份，并提供保护服务流量的能力，使其可以在不同可信度的网络上流转。
- 平台支持：Istio旨在在各种环境中运行，包括跨云、Kubernetes、Mesos等。最初专注于Kubernetes，但很快将支持其他环境。
- 集成和定制：策略执行组件可以扩展和定制，以便与现有的ACL、日志、监控、配额、审核等解决方案集成。

Istio针对可扩展性进行了设计，以满足不同的部署需要。这些功能极大的减少了应用程序代码、底层平台和策略之间的耦合，使得微服务更加容易实现。
下图为Istio的架构设计图，主要包括了Envoy、Pilot、Mixer和Istio-Auth等。



Envoy： 扮演Sidecar的功能，协调服务网格中所有服务的出入站流量，并提供服务发现、负载均衡、限流熔断等能力，还可以收集与流量相关的性能指标。

Pilot: 负责部署在Service Mesh中的Envoy实例的生命周期管理。本质上是负责流量管理和控制，将流量和基础设施扩展解耦，这是Istio的核心。可以把Pilot看做是管理Sidecar的Sidecar，但是这个特殊的Sidacar并不承载任何业务流量。Pilot让运维人员通过Pilot指定它们希望流量遵循什么规则，而不是哪些特定的pod/VM应该接收流量。有了Pilot这个组件，我们可以非常容易的实现A/B 测试和金丝雀Canary测试。

Mixer: Mixer在应用程序代码和基础架构后端之间提供通用中介层。它的设计将策略决策移出应用层，用运维人员能够控制的配置取而代之。应用程序代码不再将应用程序代码与特定后端集成在一起，而是与Mixer进行相当简单的集成，然后Mixer负责与后端系统连接。Mixer可以认为是其他后端基础设施（如数据库、监控、日志、配额等）的Sidecar Proxy。

Istio-Auth: 提供强大的服务间认证和终端用户认证，使用交互TLS，内置身份和证书管理。可以升级服务网格中的未加密流量，并为运维人员提供基于服务身份而不是网络控制来执行策略的能力。Istio的未来版本将增加细粒度的访问控制和审计，以使用各种访问控制机制（包括基于属性和角色的访问控制以及授权钩子）来控制和监视访问服务、API或资源的访问者。

Istio的设计理念先进，功能也比较强大，加之Google、IBM的影响力让Istio迅速传播，让广大开发者认识到了Istio项目在Service Mesh领域的重要性。但是Istio目前版本也存在了一些不足：

- 目前的Istio大部分能力与Kubernetes是强关联的。而我们在构建微服务的时候往往是希望服务层与容器层是解耦的，服务层在设计上需要能够对接多种容器层平台。
- Istio至今未有稳定版本，截至本文编写时为止，Istio的最新版本为0.8版本，预计在2018年内会发布1.0版本。

来自：https://blog.csdn.net/dylloveyou/article/details/81951286