DaoCloud

IT大咖说 知识共享平台

重新定义计算的边界

业界领先的企业级容器云

黄久远 | 开源工程师

DaoCloud

IT大咖说 知识共享平台

浅谈Kubernetes与服务网格

黄久远丨开源工程师

内容大纲

DaoCloud

IT大咖说 知识共享平台

- 什么是Kubernetes
- 什么是服务网格
- Kubernetes与服务网格

DaoCloud

IT大咖说 知识共享平台

什么是Kubernetes

什么是Kubernetes

DaoCloud | IT大咖说 知识共享平台

Kubernetes简介

- Kubernetes是Google开源的容器集群管理系统，由Google多年大规模容器管理技术Borg演化而来并赠给云原生计算基金会（CNCF），其主要功能包括
  - 基于容器的应用部署、维护和滚动升级
  - 负载均衡和服务发现
  - 跨机器和跨地区的集群调度
  - 自动伸缩
  - 无状态服务和有状态服务
  - 广泛的存储支持
  - 插件机制保证扩展性
- Kubernetes发展非常迅速，已经成为容器编排领域的领导者

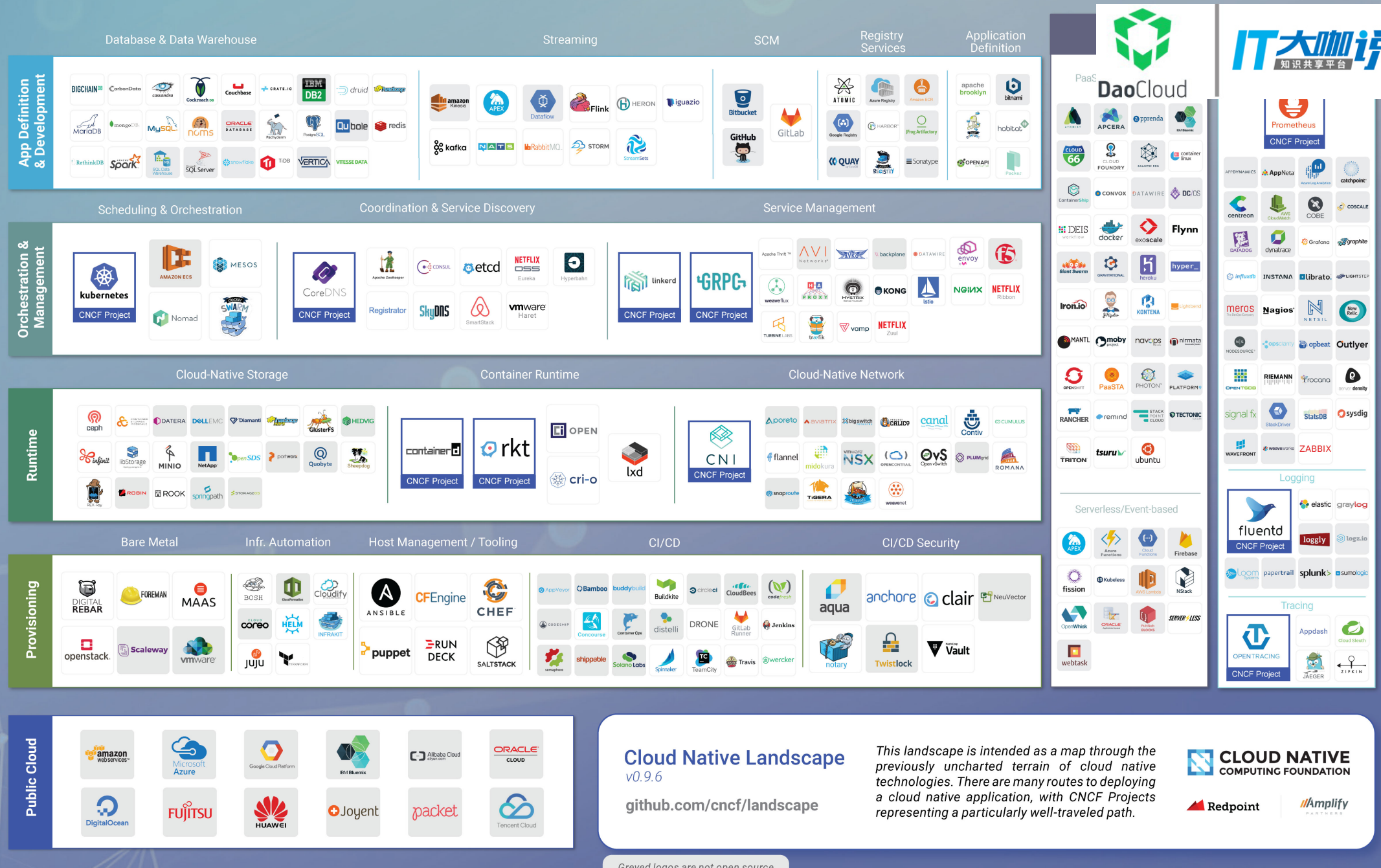
什么是Kubernetes

DaoCloud | IT大咖说 知识共享平台

Kubernetes是一个平台

- Kubernetes提供了很多的功能，它可以简化应用程序的工作流，加快开发速度
- 通常一个成功的应用编排系统需要有较强的自动化能力，这也是为什么Kubernetes被设计作为构建组件和工具的生态系统平台，以便更轻松地部署、扩展和管理应用程序
- 用户可以使用Label以自己的方式组织管理资源，还可以使用Annotation来自定义资源的描述信息，比如为管理工具提供状态检查等
- Kubernetes控制器也是构建在跟开发人员和用户使用的相同的API之上，用户还可以编写自己的控制器和调度器，也可以通过各种插件机制扩展系统的功能
- 这种设计使得可以方便地在Kubernetes之上构建各种应用系统

Cloud Native Computing Foundation



Cloud Native Landscape

v0.9.6

github.com/cncf/landscape

This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.

CLOUD NATIVE COMPUTING FOUNDATION

Redpoint | Amplify Partners

Greyed logos are not open source

DaoCloud

IT大咖说 知识共享平台

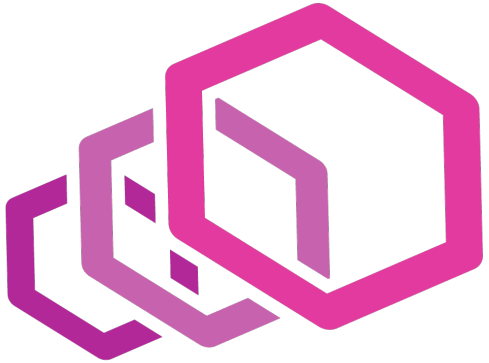
什么是服务网格

什么是服务网格

DaoCloud | IT大咖说 知识共享平台

服务网格的起源

- 服务网格的英文叫做Service Mesh，最早由Buoyant公司提出在2016年提出
- Linkerd是现在市面上第一个成熟的服务网格产品，由Buoyant公司开发并在2016年1月发布
- Lyft公司在2016年9月正式发布了另一个服务网格产品Envoy
- 在2017年5月，Google、IBM和Lyft共同推出了重量级的服务网格产品Istio
- 在2017年12月， Buoyant公司宣布了基于Kubernetes的服务网格产品Conduit



envoy

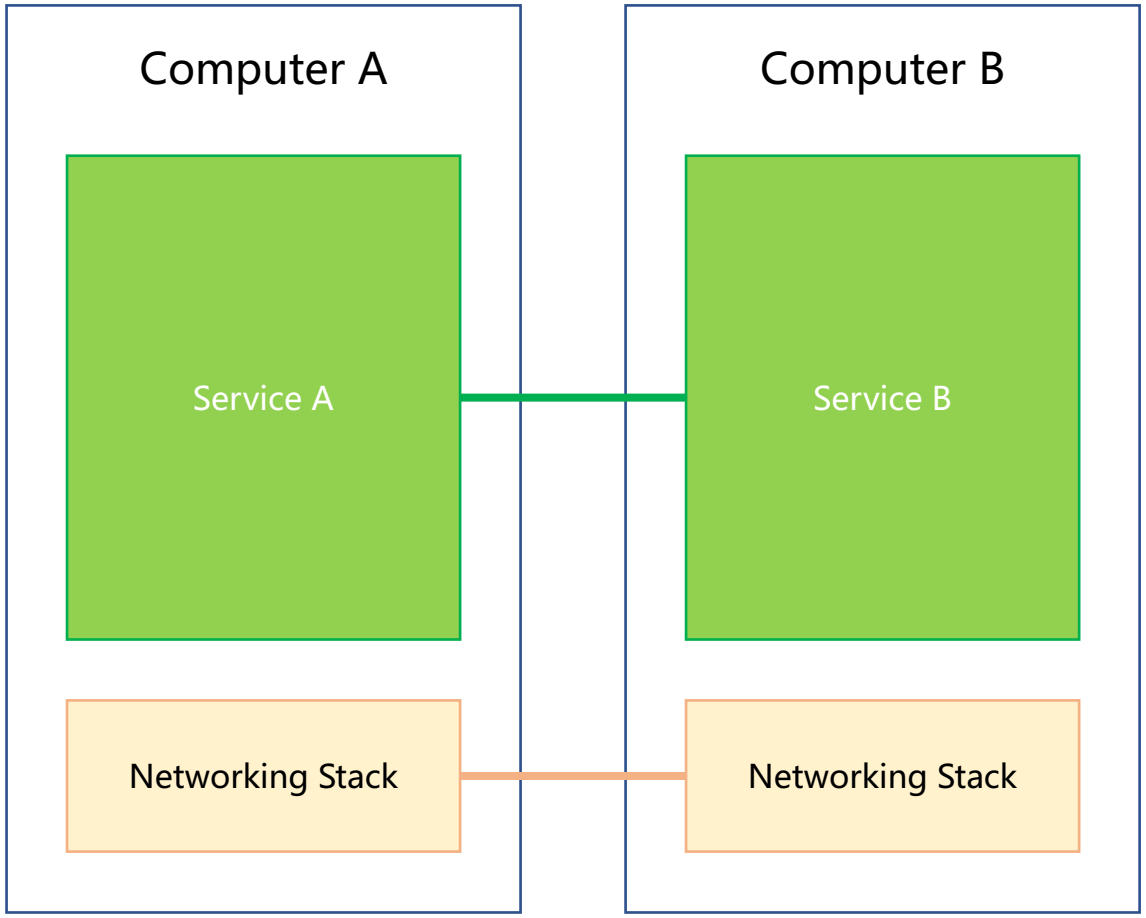
什么是服务网格

DaoCloud

IT大咖说 知识共享平台

最初的网络交互

- 一个服务通过与另一个服务发生对话来完成用户的请求，而传输字节码和传输电子信号的工作则交给底层的网络栈
- 起初计算机很稀有并且昂贵；但随着计算机越来越普及，计算机之间的连接数和数据量出现了爆炸式的增长，人们越来越依赖网络系统，工程师们必须确保他们开发的服务能够满足用户的要求

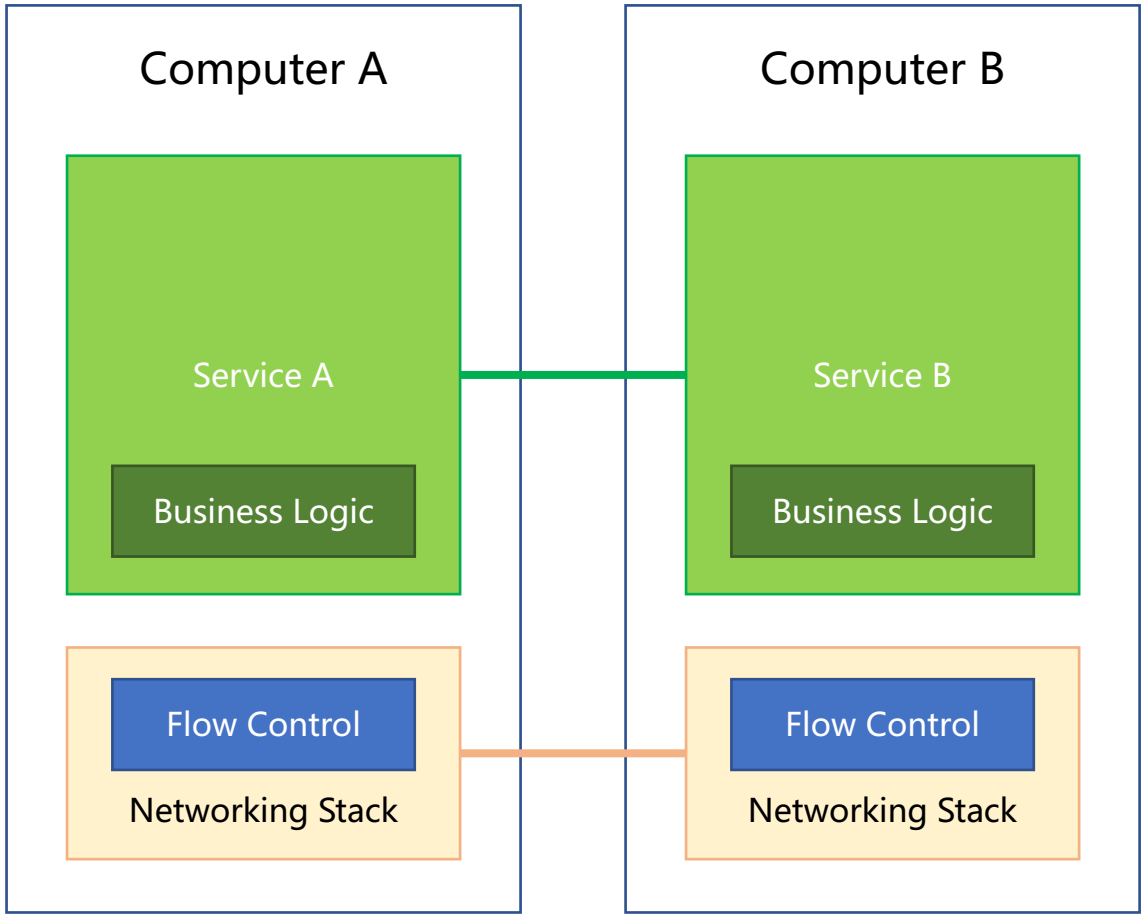


什么是服务网格

DaoCloud | IT大咖说 知识共享平台

最初的网络交互

- 如何提升系统质量成为人们关注的焦点，机器需要知道如何找到其他节点，处理同一个通道上的并发连接，与非直接连接的机器发生通信，通过网络路由数据包，加密流量等
- 除此之外，还需要流量控制机制；流量控制可以防止上游服务器给下游服务器发送过多的数据包
- 随着技术的发展，TCP/IP标准的出现解决了流量控制等问题，并且从应用程序里抽离出来，成为操作系统网络层的一部分

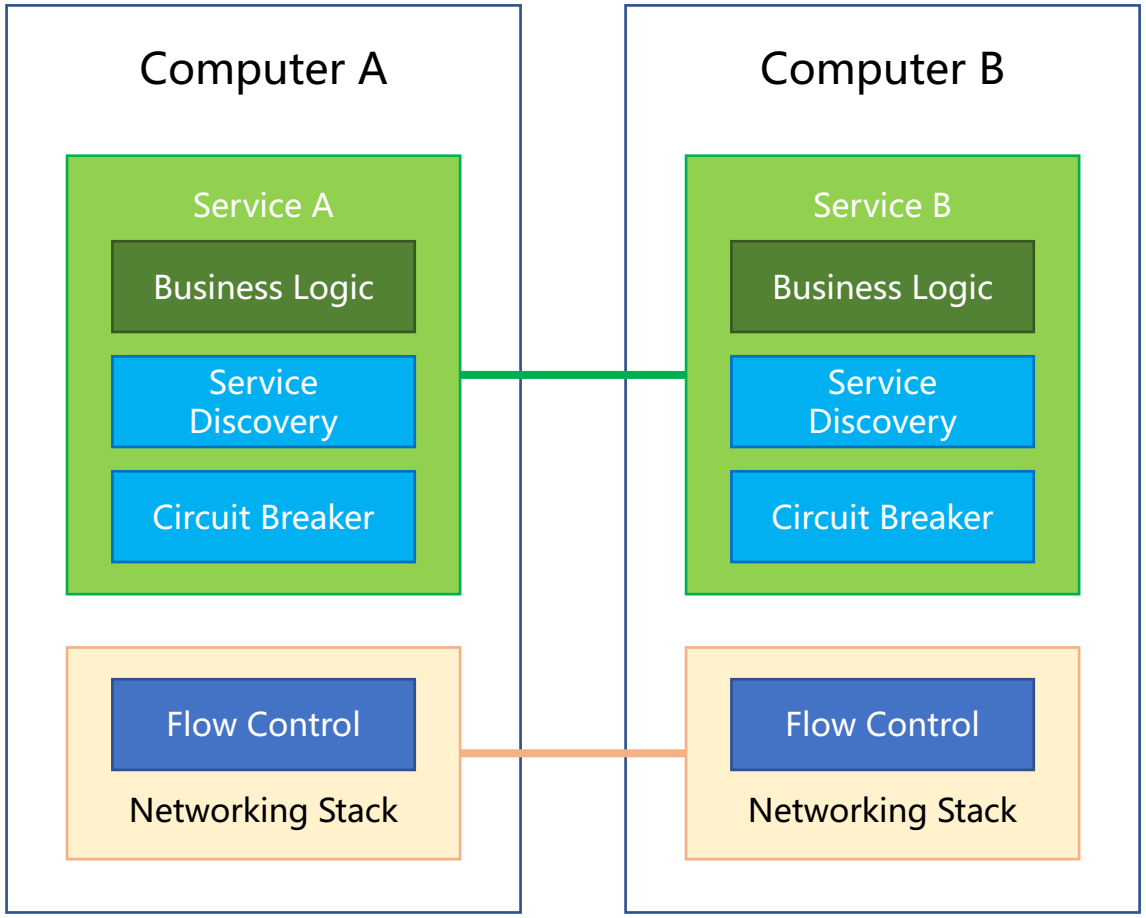


什么是服务网格

DaoCloud | IT大咖说 知识共享平台

微服务的出现

- 网络技术栈已经被证实是构建可靠连接系统行之有效的方案，业界出现了各种网络系统，如分布式系统和微服务架构
- 微服务架构是复杂的分布式系统，它给运维带来了诸多挑战
- 工程师们需要在系统中加入一个新的层来应对这些问题；以服务发现和断路器为例，这两种技术被用于解决弹性和分布式问题
- 微服务软件库为许多公司带来了便利
- 但是公司仍然需要投入时间和精力去打通各个系统并吞噬掉本该用于构建产品的时间
- 微服务软件库会限制可用的工具、运行时和编程语言，移植过程需要耗费大量的工程时间
- 微服务软件库封装了功能但组件本身仍需要维护

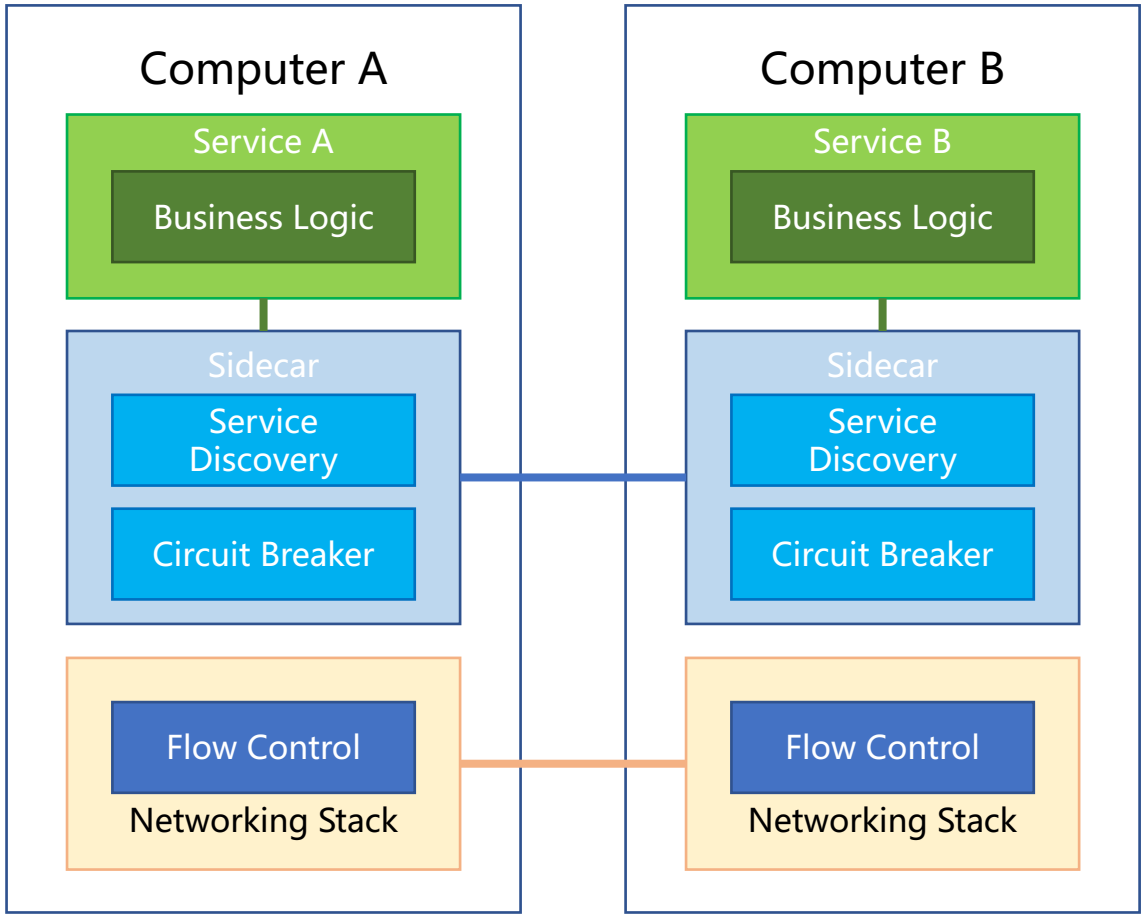


什么是服务网格

DaoCloud | IT大咖说 知识共享平台

边车代理

- 与网络协议栈一样，我们急切地希望能够将分布式服务所需要的一些特性放到底层的平台中
- 人们基于HTTP协议开发复杂的应用且无需关心底层TCP如何控制数据包；在开发微服务时也是类似的，工程师们应该聚焦在业务逻辑上而不是浪费时间去编写服务基础设施代码或管理软件库和框架
- 但是在网络协议栈中加入一个微服务层是不切实际的
- 通过在边车（Sidecar）中运行的一系列代理实现了这一功能，一个服务不会直接与它的依赖项发生连接，所有的流量都会流经代理，代理会实现所有必需的特性



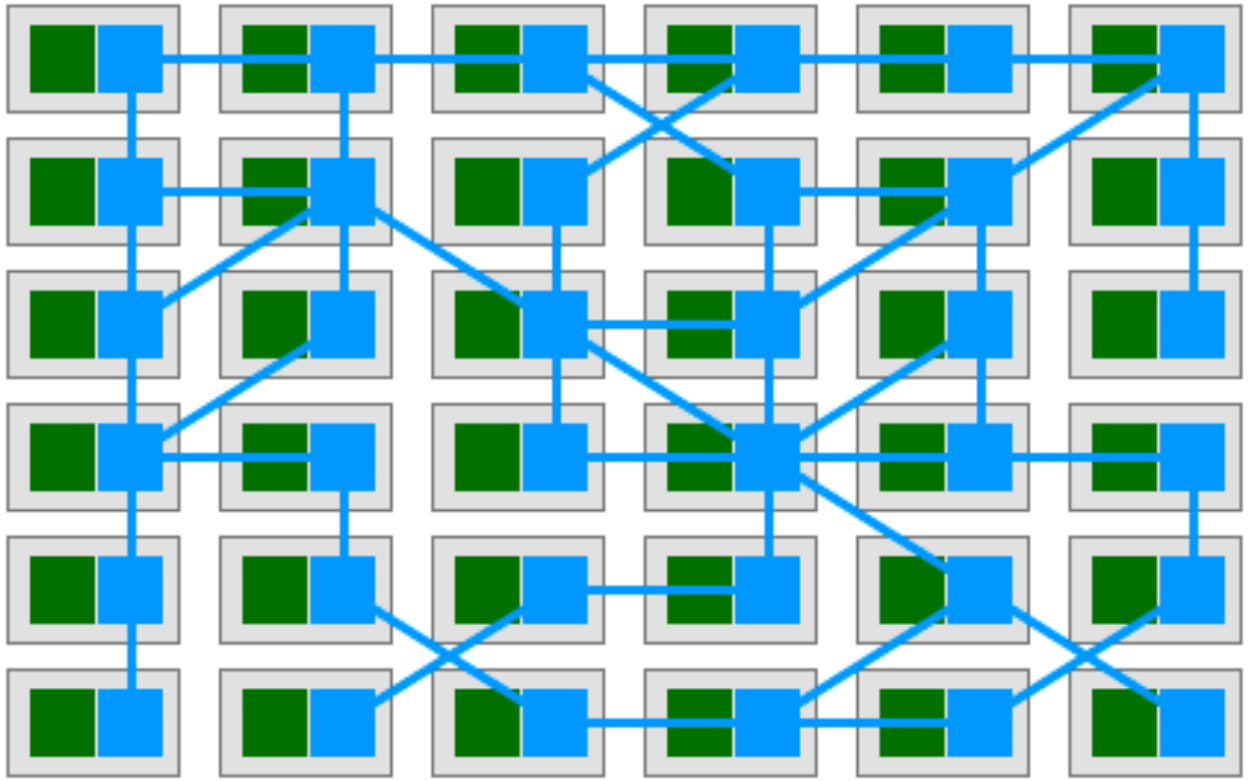
什么是服务网格

DaoCloud

IT大咖说 知识共享平台

服务网格

- 每个服务都会有一个边车代理与之配对，服务间通信都是通过边车代理进行的，一个复杂的微服务系统如右所示
- 服务网格不把代理看成单独的组件，而是强调了这些代理所形成的网络的重要性
- 公司可以将微服务部署到更为复杂的运行时上，比如Kubernetes，并从使用一系列独立运行的代理转向使用集中式的控制面板
- 服务的实际流量仍然在代理间流转，不过控制面板对每一个代理实例了如指掌，通过控制面板可以实现代理的访问控制和度量指标收集



DaoCloud

什么是服务网格

- 服务网格是一个用于处理服务间通信的专有基础设施层
- 云原生应用往往由多个复杂的服务组成，服务网格负责实现这些服务之间请求的可靠传递
- 服务网格通常实现为一组轻量级网络代理并与应用程序一同被部署，应用程序不需要知道网络代理的存在
- 在云原生模型中，一个应用可以由多个服务组成，每个服务可能又有多个实例，而每个实例可能会持续地发生变化；服务间通信不仅异常复杂，而且也是运行时行为的基础，管理好服务间通信对于保证端到端的性能和可靠性来说是非常重要的

什么是服务网格

DaoCloud

服务网格是一种网络模型吗

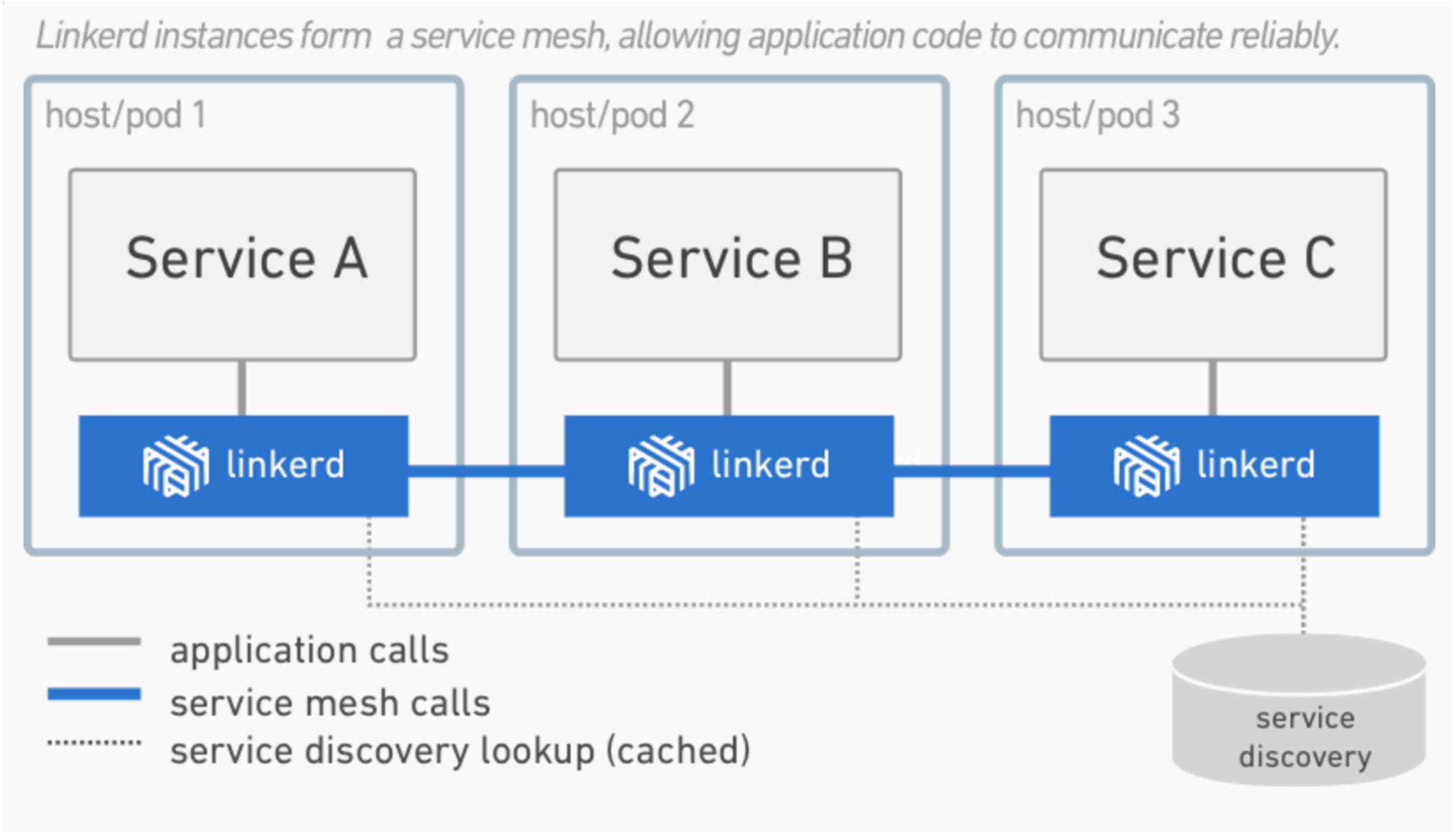
- 服务网格实际上是TCP/IP之上的一个抽象层网络模型，它假设底层的三层和四层网络能够点对点地传输字节；并且它也假设网络环境是不可靠的，所以服务网格必须具备处理网络故障的能力
- 从某种程度上说，服务网格类似于TCP/IP；TCP对网络端点间传输字节的机制进行了抽象，而服务网格对服务节点间请求的传输机制进行了抽象；与TCP一样，服务网格不关心消息体是什么，也不关心它们是如何编码的
- 应用程序有着高级目标，比如将某些东西从A传送到B；而服务网格所要做的就是实现这个目标，并处理传送过程中可能出现的任何故障
- 与TCP不同的是，服务网格为应用运行时提供统一的、应用层面的可见性和可控性；它将服务间通信从底层的基础设施中分离出来，让它成为整个生态系统的一等公民，即服务网格可以被监控、托管和控制

DaoCloud

服务网格可以做什么

- 以Linkerd为例，它使用了一系列强大的技术来管理云原生应用中请求传输任务：断路器、负载均衡、服务发现、重试和超时等
- 当一个请求流经Linkerd时会发生一系列事件
  - Linkerd根据动态路由规则确定请求是发给哪个服务的；比如是发给生产环境里的服务还是发给测试环境里的服务？是发给本地数据中心的服务还是发给云端的服务？是发给最新版本的服务还是发给旧版本的服务？这些路由规则可以动态配置，可以应用在全局的流量上，也可以应用在部分流量上
  - 在确定了请求的目标服务后，Linkerd从服务发现端点获取相应的服务实例
  - Linkerd基于某些因素，比如最近处理请求的延迟情况，选择更有可能快速返回响应的实例
  - Linkerd向选中的实例发送请求，并把延迟情况和响应类型记录下来
  - 如果选中的实例发生宕机、没有响应或无法处理请求，Linkerd就把请求发给另一个实例
  - 如果一个实例持续返回错误，Linkerd就会将其从负载均衡池中移除，并在稍后定时重试
  - 如果请求超时，Linkerd会主动放弃请求，不会进行额外的重试
  - Linkerd以度量指标和分布式日志的方式记录上述各种行为，然后将度量指标发送给中心度量指标系统

Linkerd


Linkerd instances form a service mesh, allowing application code to communicate reliably.
host/pod 1
host/pod 2
host/pod 3
Service A
Service B
Service C
linkerd
linkerd
linkerd
application calls
service mesh calls
service discovery lookup (cached)
service discovery

DaoCloud

为什么服务网格是必要的

- 2000年左右的中型Web应用一般使用了三层模型：应用逻辑层、Web服务逻辑层和存储逻辑层；层与层之间的交互复杂性是很有限的
- 随着规模的增长，这种架构就显得力不从心了；像Google、Netflix、Twitter这样的公司面临着大规模流量的挑战，他们实现了一种高效的解决方案，也就是云原生应用的前身：应用层被拆分为多个微服务，这个时候层就变成了一种拓扑结构，这样的系统需要一个通用的通信层，以一个臃肿的客户端库的形式存在，如Twitter的Finagle、Netflix的Hystrix和Google的Stubby
- 现在的云原生模型在原先的微服务模型中加入了两个额外的元素：容器和编排层；容器提供了资源隔离和依赖管理，编排层对底层的硬件进行抽象化
- 应用程序在云环境中具备了伸缩能力和处理局部故障的能力，但随着服务和实例的数量增长，编排层需要无时不刻地调度实例，请求在服务拓扑间穿梭的路线也变得异常复杂，再加上可以使用任意语言来开发不同的服务，所以使用臃肿的客户端库的解决思路就越来越行不通了
- 这种复杂性和迫切性催生了服务间通信层的出现，这个层既不会与应用程序的代码耦合，又能捕捉到底层环境高度动态的特点，即服务网格

DaoCloud

Kubernetes与服务网格

DaoCloud

什么是Istio

- 与现象级的产品Kubernetes一样，Istio由Google主导开发，并且隶属于GCP事业部
- Google在2017年5月发布了Istio的0.1版本，并且在2017年12月的Kubecon结束后发布了0.4版本
- Istio是一个用来连接、管理和保护微服务的开放平台
- Istio提供一种简单的方式来建立已部署服务网络，具备负载均衡、服务间认证、监控等功能，而不需要改动任何服务代码
- 通过为服务部署一个特殊的边车即可为服务增加对Istio的支持，用户可以使用Istio配置和管理代理，拦截微服务之间的所有网络通信
- Istio推出后即支持在Kubernetes上的部署

Google

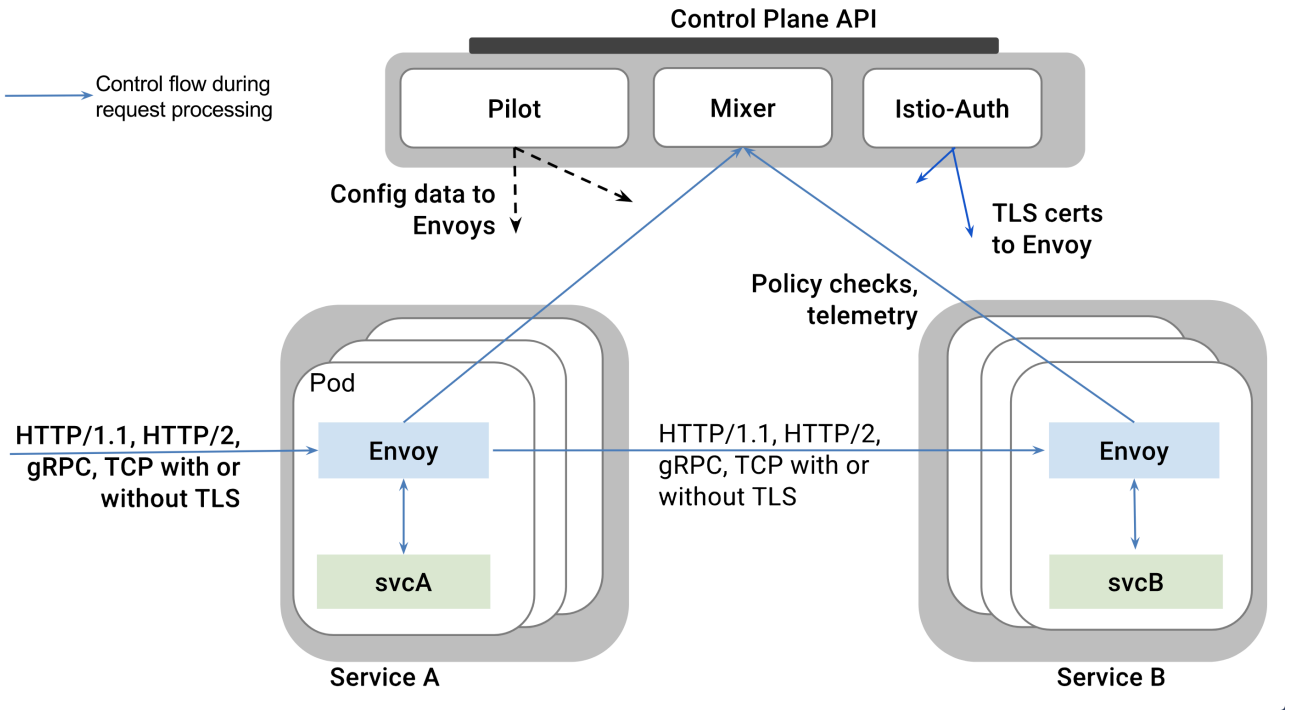
Istio可以做什么

- Istio提供了一个服务网格的完整解决方案，包括服务发现、负载均衡、故障恢复、指标收集、监控以及更加复杂的运维需求，例如A/B测试、金丝雀发布、限流、访问控制和端到端认证等；通过为整个服务网格提供行为洞察和操作控制来满足微服务应用程序的多样化需求，它在服务网络中提供了许多关键功能
    - 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旨在可以在各种环境中运行，最初专注于Kubernetes，但正在逐渐添加对其他环境的支持
    - 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

DaoCloud

Istio的架构

- Istio服务网格逻辑上分为数据面板和控制面板
- 数据面板由一组智能代理（Envoy）组成，代理部署为边车，调解和控制微服务之间所有的网络通信
- 控制面板负责管理和配置代理来路由流量，以及在运行时执行策略
- Istio主要由四个组件组成
  - Envoy
  - Pilot
  - Istio-Auth
  - Mixer



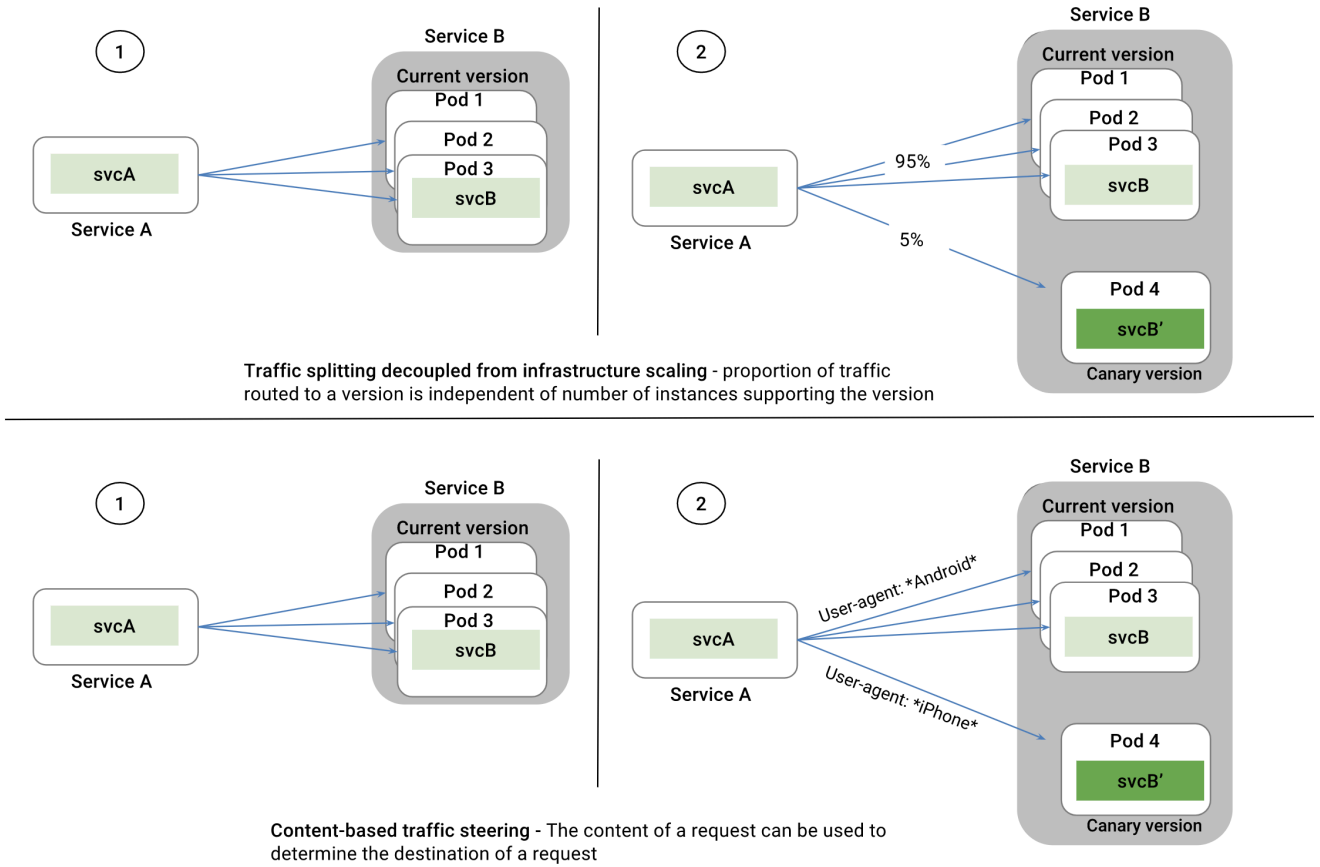
Envoy

- Istio使用Envoy代理的扩展版本，Envoy是一个高性能代理，用于调解服务网格中所有服务的所有入站和出站流量
- Envoy的许多内置功能被istio发扬光大，例如动态服务发现、负载均衡、TLS终止、HTTP/2代理、gRPC代理、熔断器、健康检查、基于百分比流量拆分的分段推出、故障注入和丰富度量
- Envoy以边车的方式部署，和对应服务在同一个Pod中；这允许Istio将大量关于流量行为的信号作为属性提取出来，而这些属性又可以在Mixer中用于执行策略决策，并发送给监控系统，以提供整个网格行为的信息
- 边车代理模型还可以将Istio的功能添加到现有部署中，而无需重新构建或重写代码

DaoCloud

流量管理

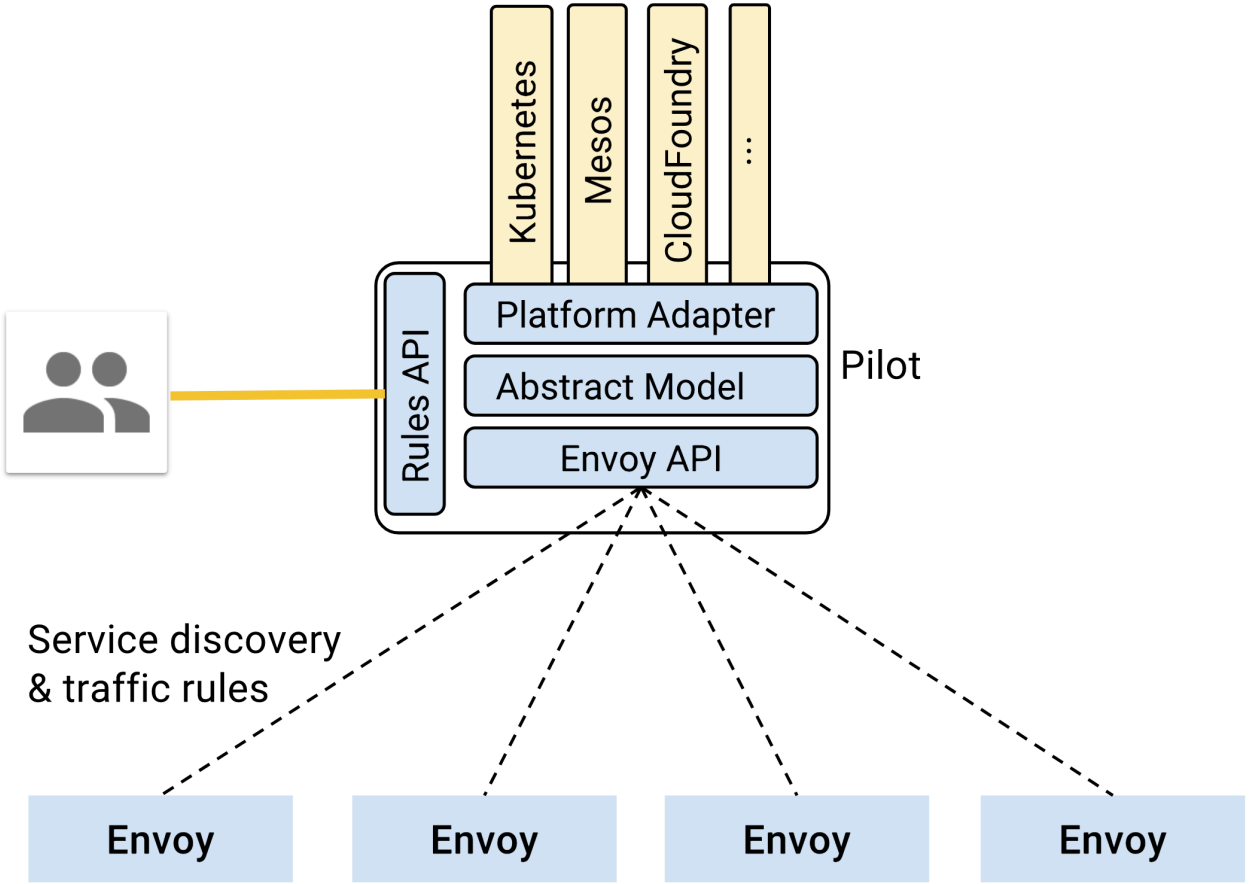
- Istio流量管理的核心组件是Pilot，它管理和配置部署在特定Istio服务网格中的所有Envoy代理实例
- Pilot允许指定在Envoy代理之间使用什么样的路由流量规则，并配置故障恢复功能，如超时、重试和熔断器
- Pilot还维护了网格中所有服务的规范模型，并使用这个模型，来通过发现服务让Envoy了解网格中的其他实例



DaoCloud

Pilot

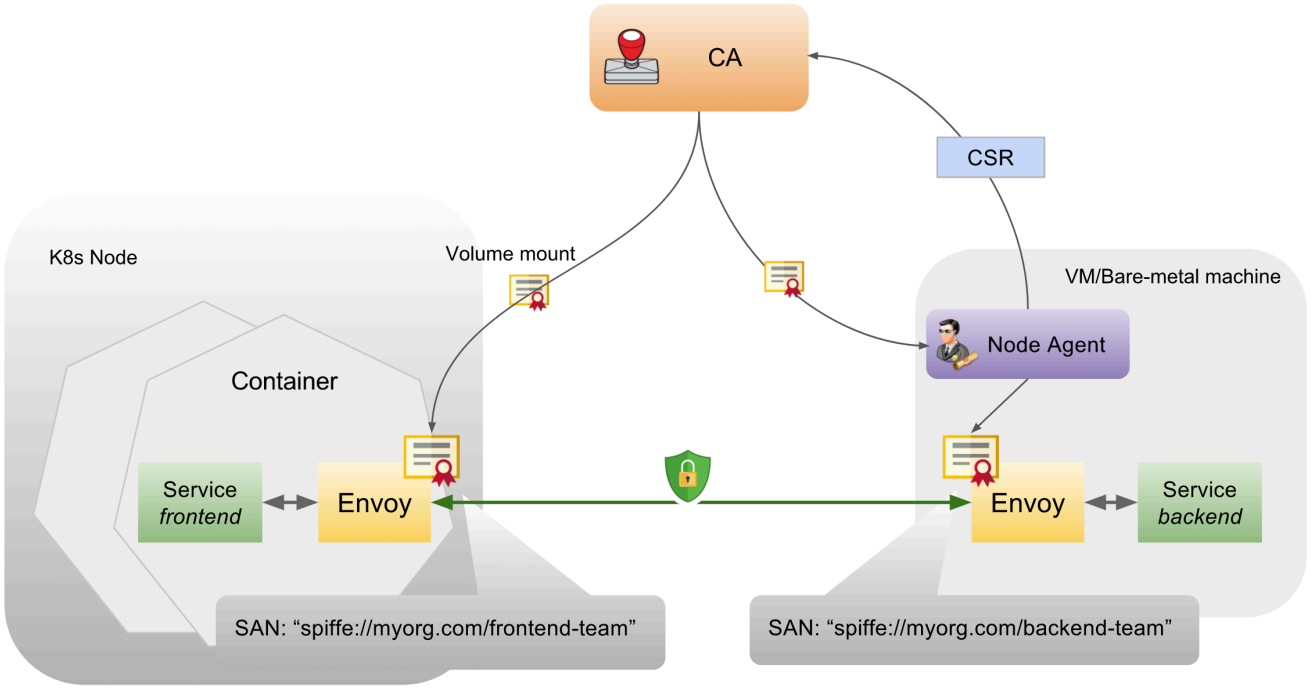
- Pilot负责在Istio服务网格中部署的Envoy实例的生命周期
- Pilot维护了网格中的服务的规范表示，这个表示是独立于底层平台的，Pilot中的平台特定适配器负责适当填充此规范模型
- Pilot中的Kubernetes适配器实现必要的控制器来查看Kubernetes API服务器，以得到Pod注册信息的更改、Ingress资源以及存储流量管理规则的第三方资源；该数据被翻译成规范表示，Envoy特定配置是基于规范表示生成的
- 运维人员可以通过Pilot的Rules API指定高级流量管理规则；这些规则被翻译成配置并分发到Envoy实例



DaoCloud

Istio-Auth

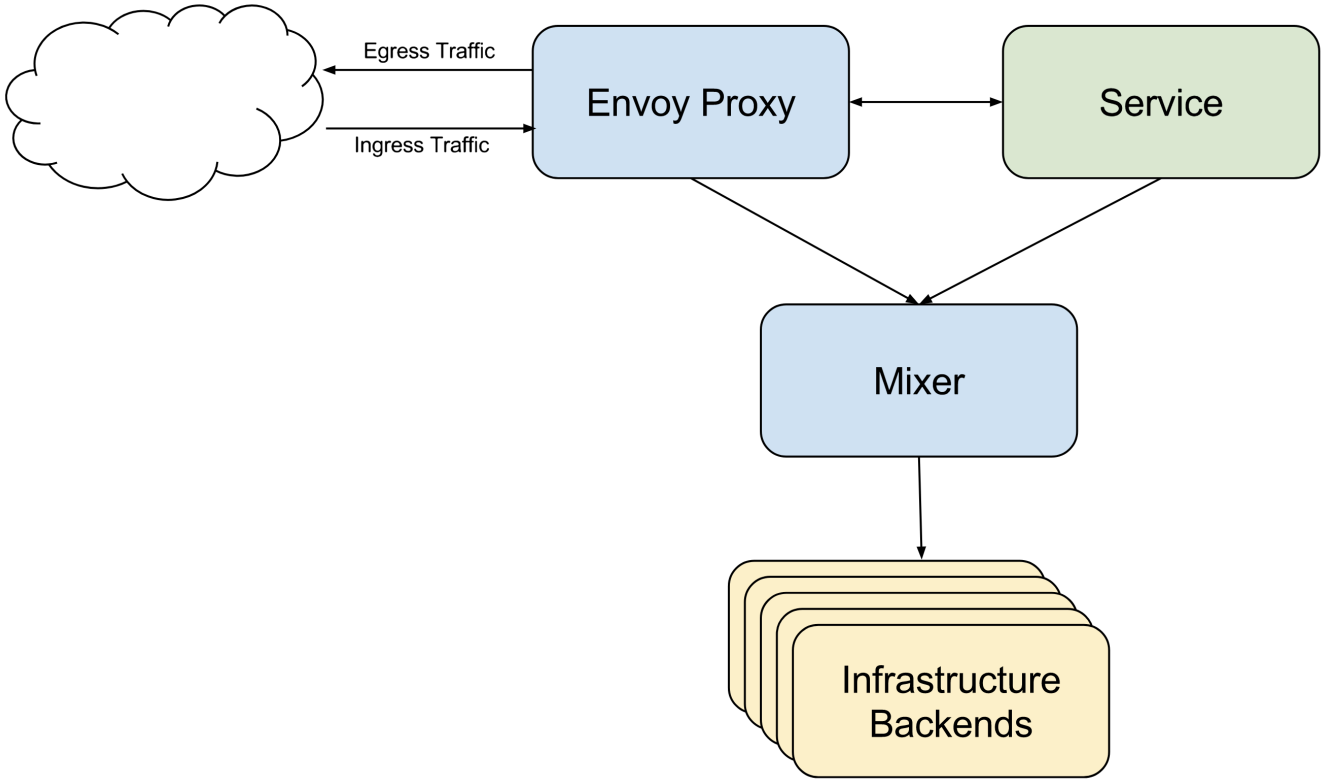
- Istio-Auth的目标是提高微服务及其通信的安全性而不需要修改服务代码，它主要负责以下功能
  - 为每个服务提供代表其角色的身份，以实现跨集群和云的互通性
  - 加密服务到服务的通讯
  - 提供密钥管理系统来自动执行密钥和证书的生成、分发、轮换和撤销
  - 强大的认证机制：ABAC、RBAC、Authorization hooks
  - 终端用户认证
  - 证书和身份可插拔



DaoCloud

Mixer

- Mixer提供三个核心功能
  - 前提条件检查：允许服务在响应来自服务消费者的传入请求之前验证一些前提条件，前提条件可以包括服务使用者是否被正确认证、是否在服务的白名单上、是否通过ACL检查等
  - 配额管理：使服务能够在分配和释放多个维度上的配额，配额这一简单的资源管理工具可以在服务消费者对有限资源发生争用时，提供相对公平的竞争手段
  - 遥测报告：使服务能够上报日志和监控



DaoCloud

IT大咖说 知识共享平台

谢谢