Istio: 服务网格领域的新王者

连接、保护、控制和观测服务 https://istio.io/

分享者: 钟华

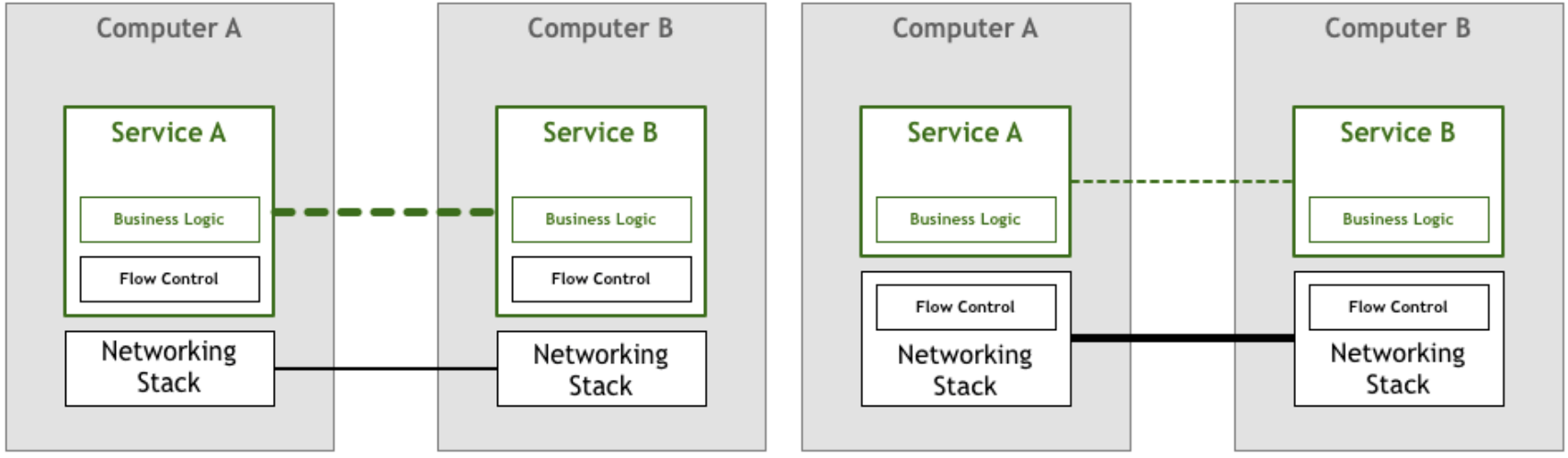
foxzhong (https://github.com/zhongfox)

1. Service Mesh: 下一代微服务
2. Istio: 第二代 Service Mesh
3. Istio 数据面
4. Istio 控制面

1. Service Mesh: 下一代微服务

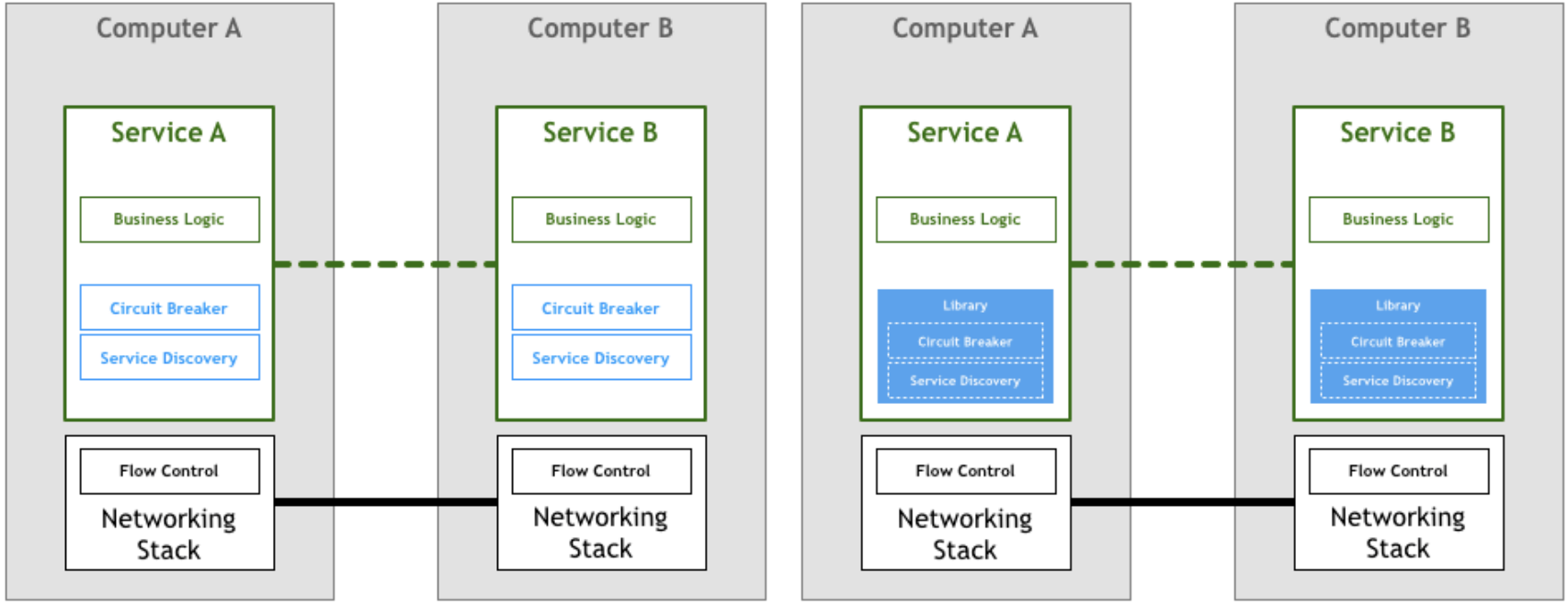
- 应用通信模式演进
- Service Mesh(服务网格)的出现
- 第二代 Service Mesh
- Service Mesh 的定义
- Service Mesh 产品简史
- 国内Service Mesh 发展情况

1.1 应用通信模式演进: 网络流控进入操作系统



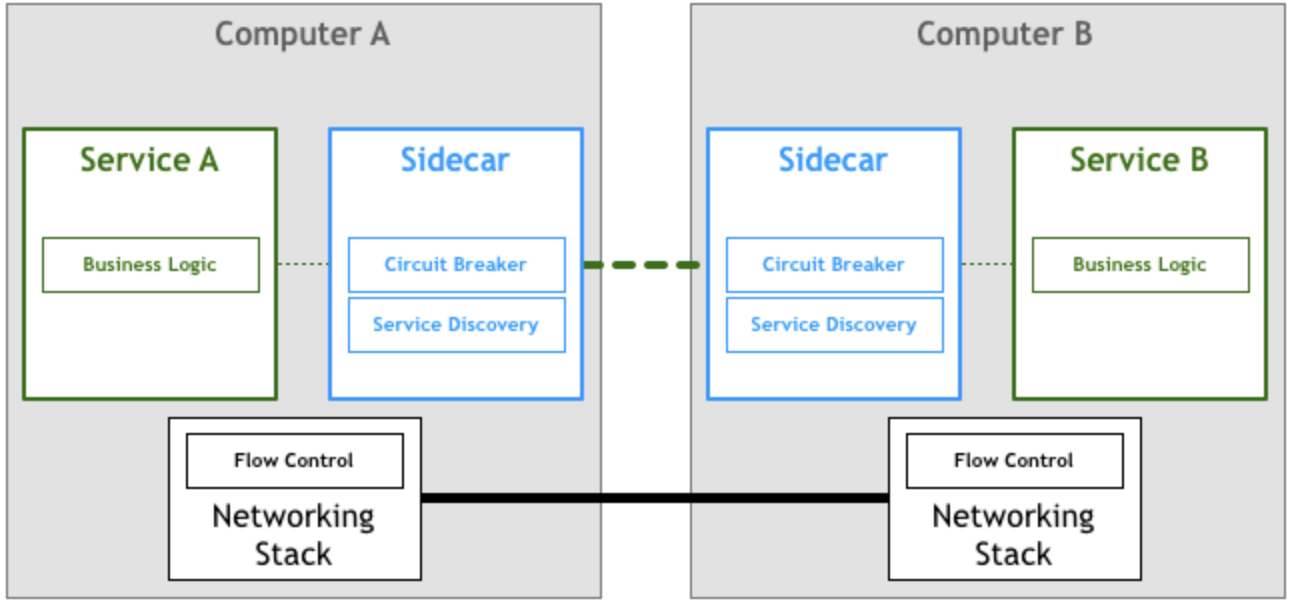
- 最初的网络计算机交互: 网络逻辑和业务逻辑就混杂在一起
- TCP/IP 等网络标准从应用程序里抽离出来，成为操作系统的一部分

1.2 应用通信模式演进: 微服务架构的出现



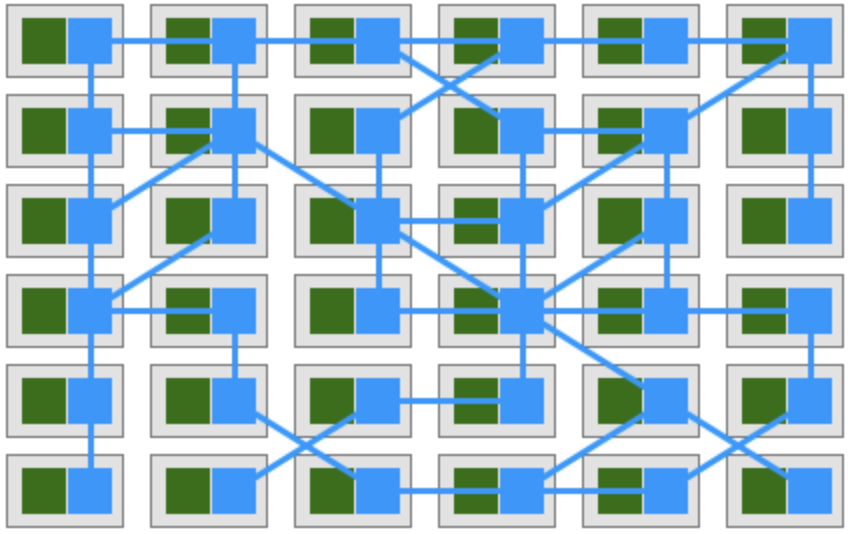
- 面向服务的架构, 对应用通信提出了更多的要求
- 服务治理的功能逐渐提炼为各种微服务类库和框架

1.3 Sidecar 模式的兴起



sidecar是与应用程序一起运行的独立进程, 为应用程序提供额外的功能

1.4 Service Mesh(服务网格)的出现



多个服务和其对应的sidecar组合到一起, 形成一种代理网格网络

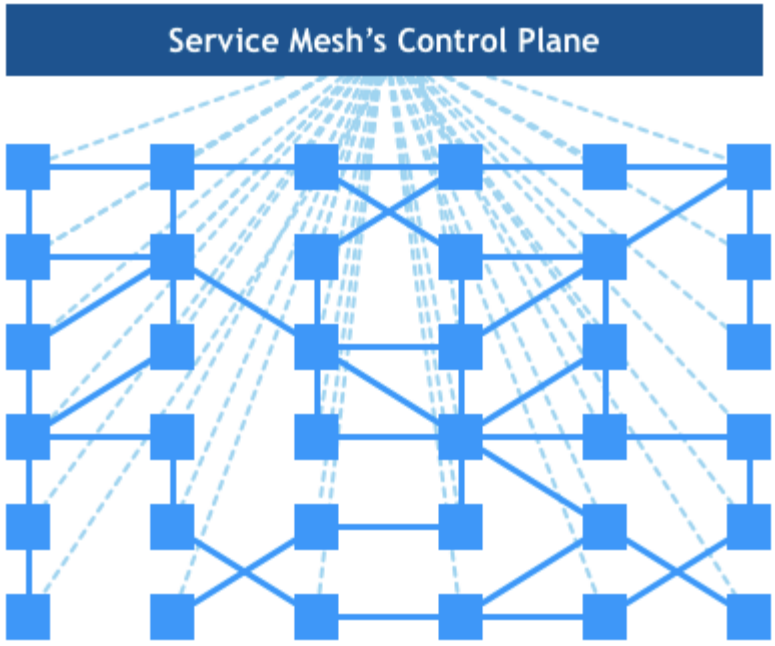
1.5 Service Mesh 定义

以下是Linkerd的CEO Willian Morgan给出的Service Mesh的定义:

> A Service Mesh is a dedicated infrastructure layer for handling service-to-service communication. It's responsible for the reliable delivery of requests through the complex topology of services that comprise a modern, cloud native application. In practice, the Service Mesh is typically implemented as an array of lightweight network proxies that are deployed alongside application code, without the application needing to be aware.

服务网格（Service Mesh）是致力于解决服务间通讯的**基础设施层**。它负责在现代云原生应用程序的复杂服务拓扑来可靠地传递请求。实际上，Service Mesh 通常是通过一组轻量级网络代理（Sidecar proxy），与应用程序代码部署在一起来实现，且**对应用程序透明**。

1.6 第二代 Service Mesh



增加网格控制面板, 代表产品:

- Istio
- Conduit/Linkerd2

1.7 Service Mesh 产品简史

linkerd

2016 年 1月，Linkerd 0.0.7 版本发布，业界第一个 Service Mesh 项目诞生
2016 年 9月，“Service Mesh”这个词汇第一次在公开场合被使用
2017 年 1月，Linkerd 加入 CNCF
2017 年 3月，Linkerd 宣布完成千亿次产品请求
2017 年 4月，Linkerd 1.0 版本发布
2017 年 7月，Linkerd 发布版本 1.1.1，宣布和 Istio 项目集成

2016 年 9月，Lyft 公司宣布 Envoy 在 GitHub 开源，直接发布 1.0.0 版本
2017 年 9月，Envoy 加入 CNCF，成为 CNCF 的第二个 Service Mesh 项目
2018 年 11月， Envoy 从 CNCF 孵化器毕业

2017 年 5月，Istio 0.1 release 版本发布, 第二代Service Mesh诞生
2018 年 7月，Istio 1.0 release 版本发布, 生产环境可用

CONDUIT

2017 年 12 月，Conduit 0.1.0 版本发布，Istio 的强力竞争对手亮相 KubeConf
2018年 6 月, Conduit 更名为 Linkerd 2

Buoyant

William Morgan
Buoyant CEO
定义了Service Mesh

Matt Klein
Engineer of lyft
Creator of envoyproxy

Google IBM lyft

1.8 似曾相识的竞争格局

	Kubernetes	Istio
领域	容器编排	服务网格
主要盟友	RedHat, CoreOS	IBM, Lyft
主要竞品	Swarm, Mesos	Linkerd, Conduit
主要竞争对手	Docker 公司	Buoyant 公司
标准化	OCI: runtime spec, image spec	xDS
插件化	CNI, CRI	Istio CNI, MixerAdapter
结果	Kubernetes 成为容器编排事实标准	?

1.9 国内Service Mesh 发展情况

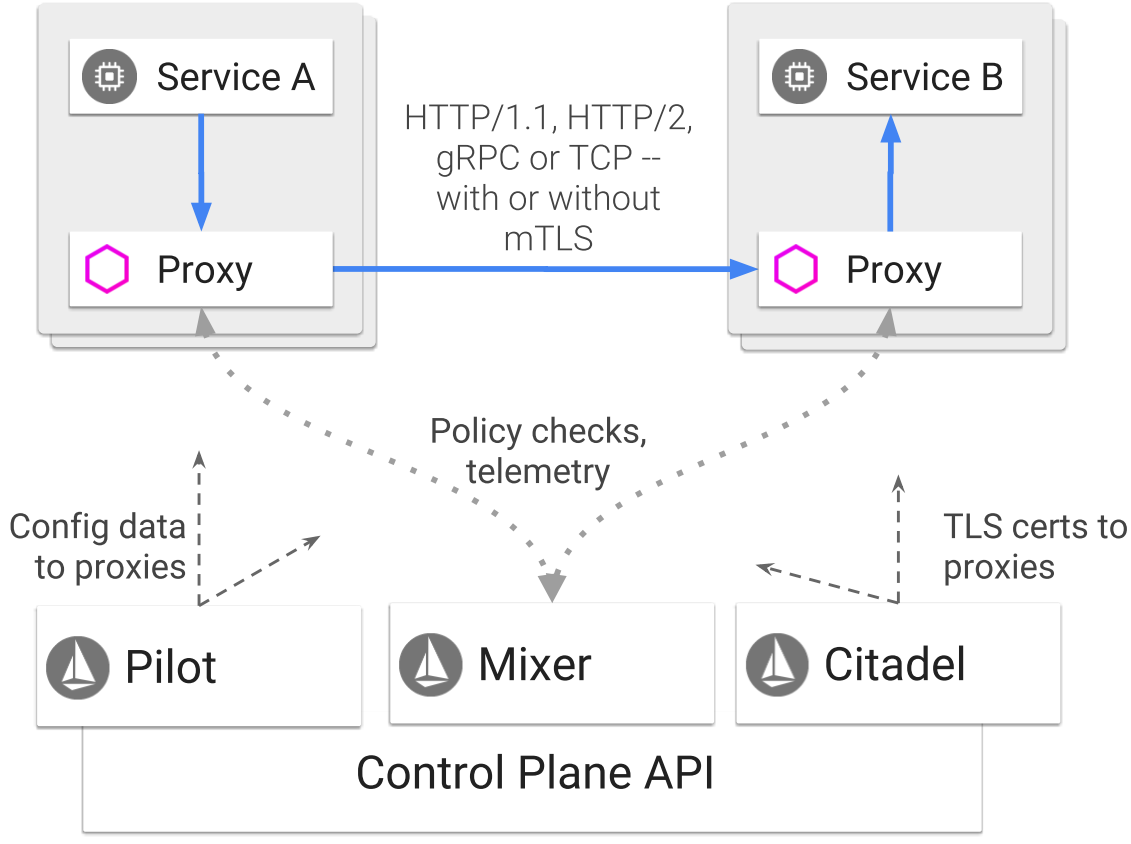
- 蚂蚁金服开源SOFAMesh
- 腾讯云 TSF
- 华为go-chassis, mesher
- 新浪: Motan
- 唯品会: OSP

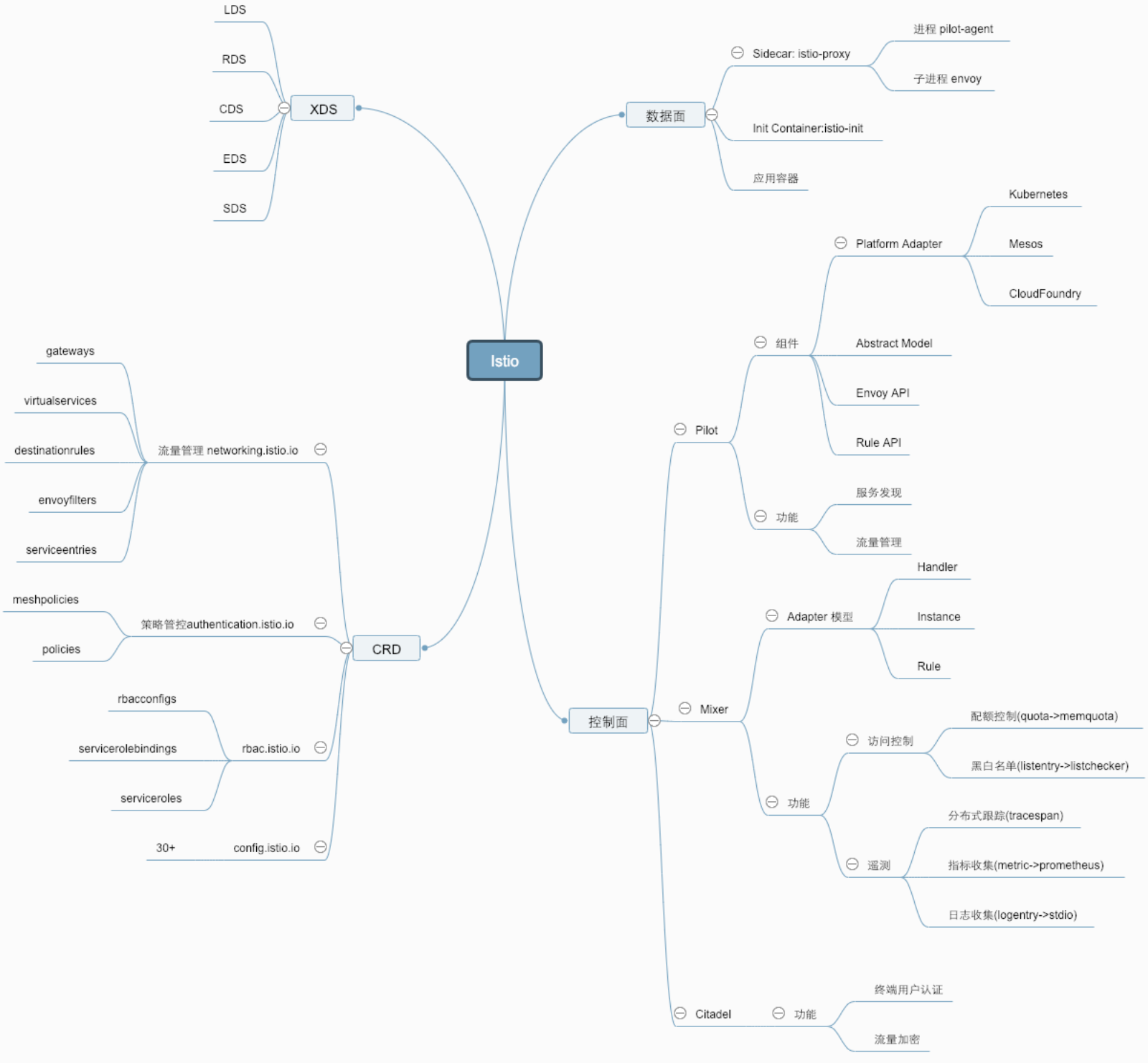
2. Istio: 第二代 Service Mesh

Istio来自希腊语，英文意思是「sail」, 意为「启航」

- 2.1 Istio 架构
- 2.2 核心功能
- 2.3 Istio 演示

2.1 Istio 架构

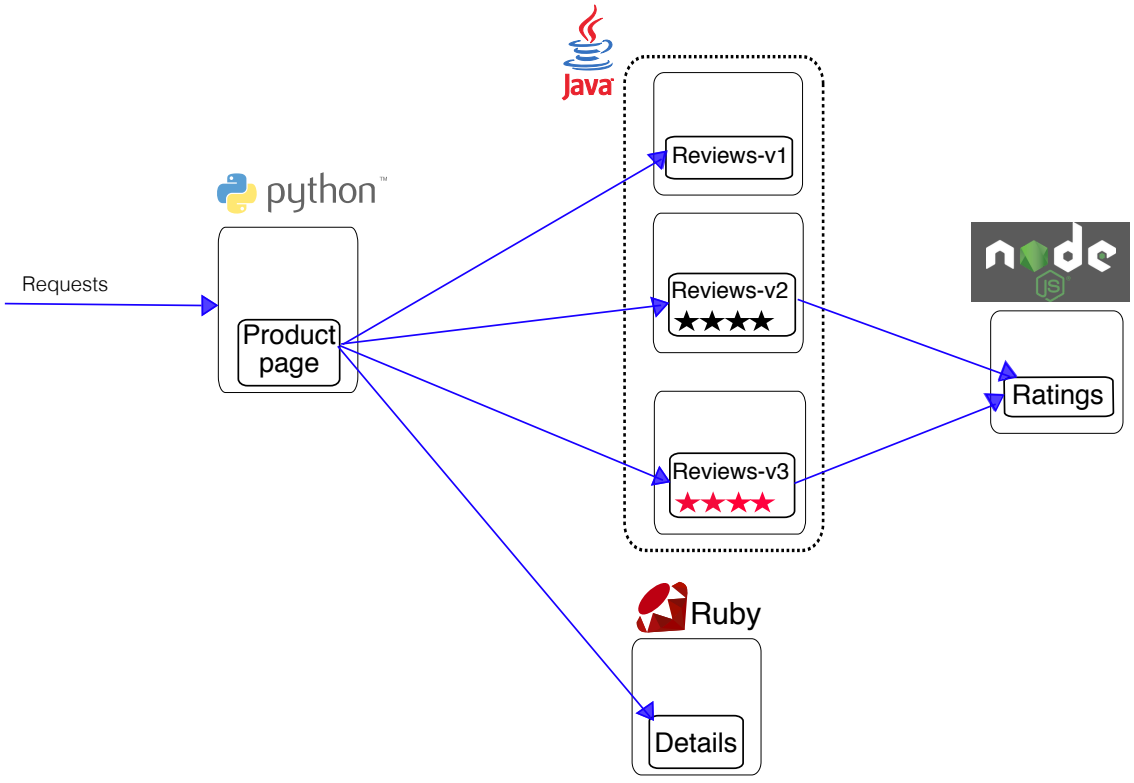

Service A
Service B
HTTP/1.1, HTTP/2,
gRPC or TCP --
with or without
mTLS
Proxy
Proxy
Policy checks,
telemetry
Config data
to proxies
TLS certs to
proxies
Pilot
Mixer
Citadel
Control Plane API


Istio
XDS
LDS
RDS
CDS
EDS
SDS
数据面
Sidecar: istio-proxy
进程 pilot-agent
子进程 envoy
Init Container:istio-init
应用容器
控制面
Pilot
组件
Platform Adapter
Kubernetes
Mesos
CloudFoundry
Abstract Model
Envoy API
Rule API
功能
服务发现
流量管理
Mixer
Adapter 模型
Handler
Instance
Rule
功能
访问控制
配额控制(quota->memquota)
黑白名单(listentry->listchecker)
遥测
分布式跟踪(tracespan)
指标收集(metric->prometheus)
日志收集(logentry->stdio)
Citadel
功能
终端用户认证
流量加密
CRD
流量管理 networking.istio.io
gateways
virtualservices
destinationrules
envoyfilters
serviceentries
策略管控authentication.istio.io
meshpolicies
policies
rbac.istio.io
rbacconfigs
servicerolebindings
serviceroles
config.istio.io
30+

2.2 核心功能

- 流量管理
- 安全
- 可观察性
- 多平台支持
- 集成和定制

2.3 Istio 演示



- 使用helm安装istio
- 基于权重的路由
- 基于内容路由

3. Istio 数据面

- 3.1 数据面组件
- 3.2 sidecar 流量劫持原理
- 3.3 数据面标准API: xDS
- 3.4 分布式跟踪

3.1 数据面组件

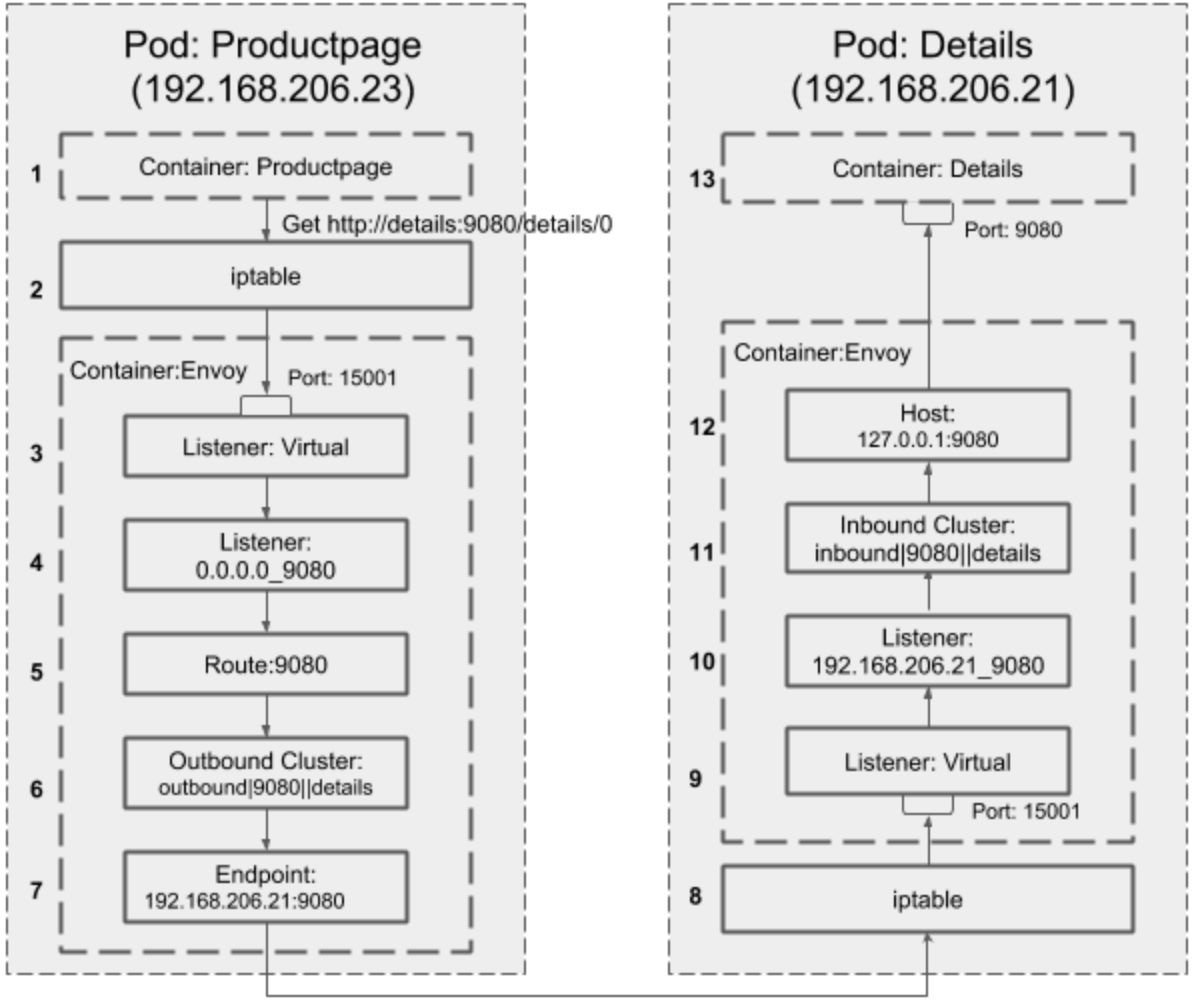
Istio 注入sidecar实现:

- 自动注入: 利用 Kubernetes Dynamic Admission Webhooks 对 新建的pod 进行注入: init container + sidecar
- 手动注入: 使用 `istioctl kube-inject`

注入Pod内容:

- istio-init: 通过配置iptables来劫持Pod中的流量
- istio-proxy: 两个进程pilot-agent和envoy, pilot-agent 进行初始化并启动envoy

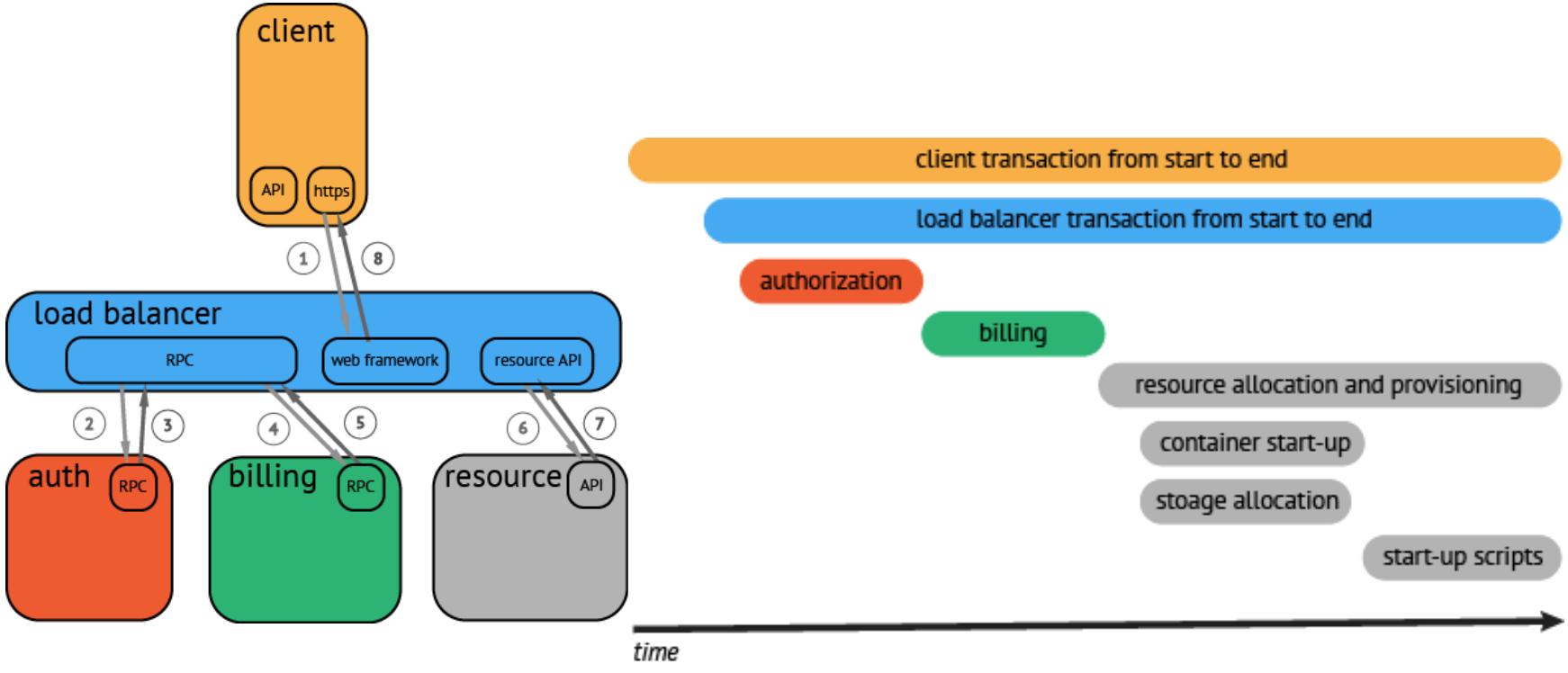
3.2 sidecar 流量劫持原理



3.3 数据面标准API: xDS

- LDS (Listener Discovery Service): 监听器
- RDS (Route Discovery Service): 路由
- CDS (Cluster Discovery Service): 集群
- EDS (Endpoint Discovery Service): 集群
- SDS (Secret Discovery Service): 证书
- ADS (Aggregated Discovery Services) 聚合的发现服务

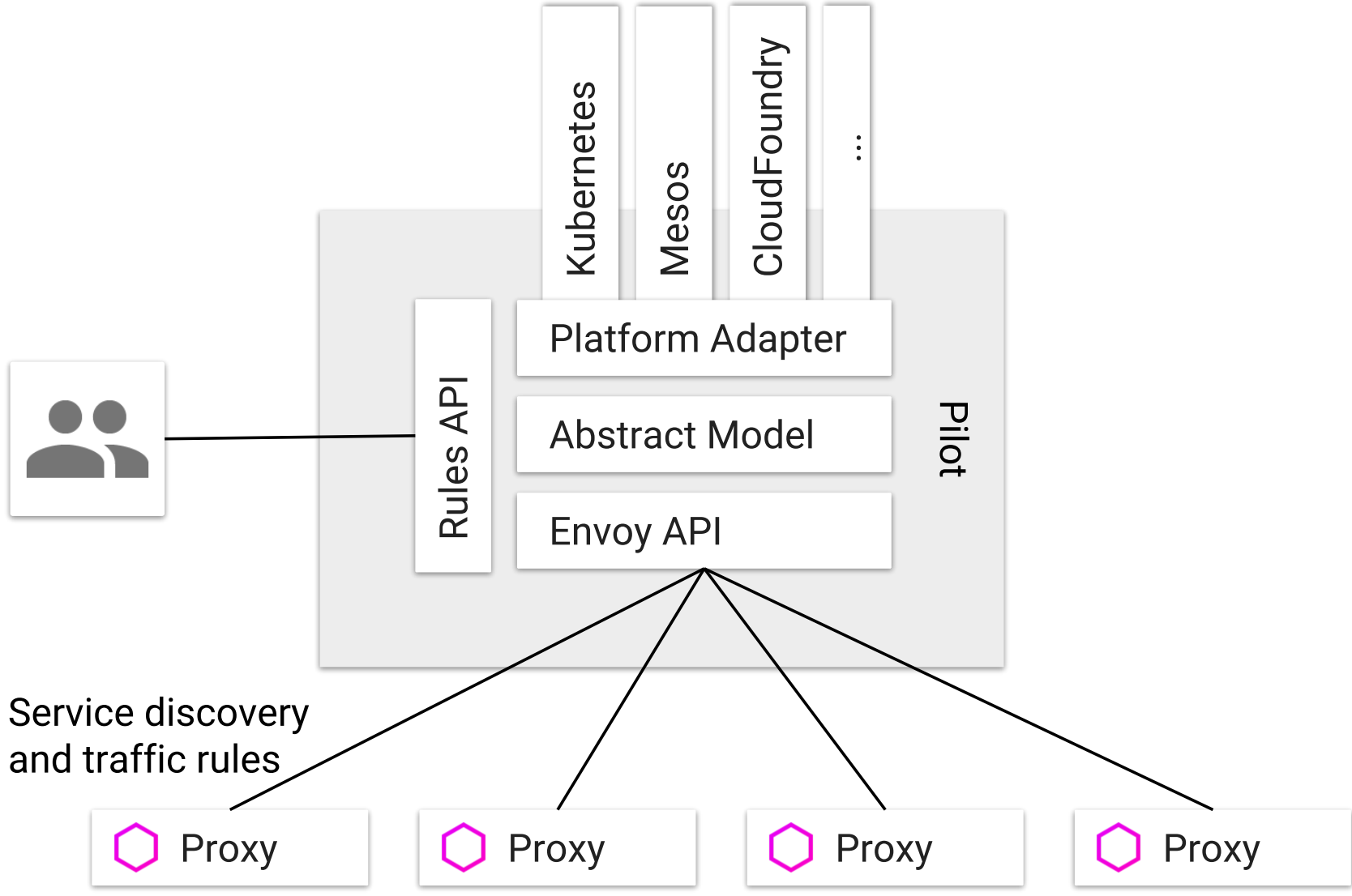
3.4 分布式跟踪


client
API
https
1
8
load balancer
RPC
web framework
resource API
2
3
4
5
6
7
auth
RPC
billing
RPC
resource
API
client transaction from start to end
load balancer transaction from start to end
authorization
billing
resource allocation and provisioning
container start-up
stoage allocation
start-up scripts
time

4. Istio 控制面

- 4.1 Pilot 架构
- 4.2 流量管理模型
- 4.3 故障处理
- 4.4 Mixer 架构
- 4.5 Mixer适配器模型
- 4.6 Mixer 缓存机制

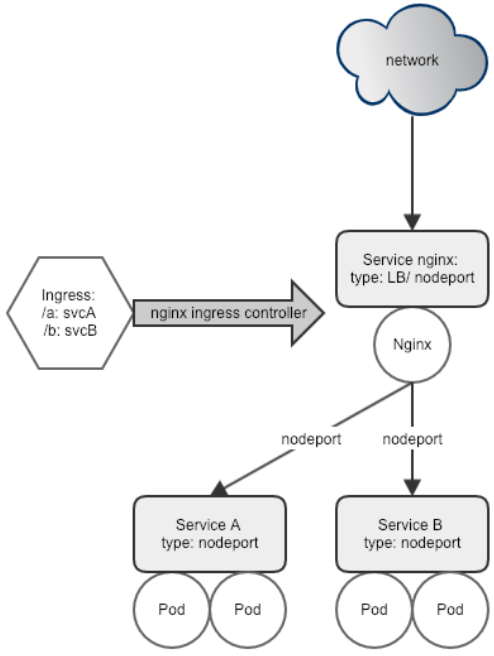
4.1 Pilot 架构

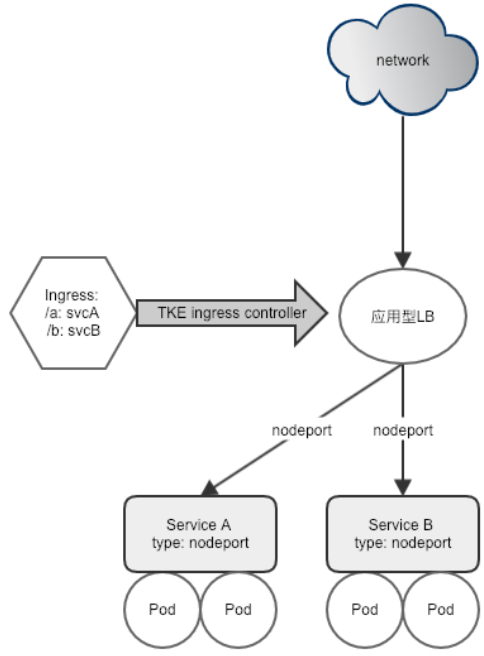

Kubernetes
Mesos
CloudFoundry
...
Platform Adapter
Abstract Model
Envoy API
Rules API
Pilot
Service discovery
and traffic rules
Proxy
Proxy
Proxy
Proxy

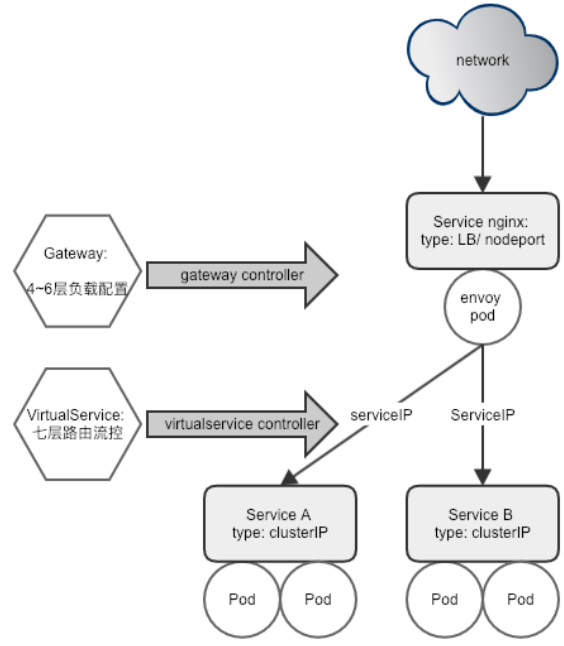
4.2 流量管理模型

- VirtualService
- DestinationRule
- ServiceEntry
- Gateway

Kubernetes Ingress, Istio Gateway 对比







Kubernetes, Istio, Envoy xDS 模型对比

	Kubernetes	Istio	Envoy xDS
入口流量	Ingress	GateWay	Listener
服务定义	Service	-	Cluster
外部服务定义	-	ServiceEntry	Cluster
版本定义	-	DestinationRule	Cluster
版本路由	-	VirtualService	Route
实例	Endpoint	-	Endpoint

Kubernetes 和 Istio 服务寻址的区别:

Kubernetes:

1. kube-dns: service domain -> service ip
2. kube-proxy(node iptables): service ip -> pod ip

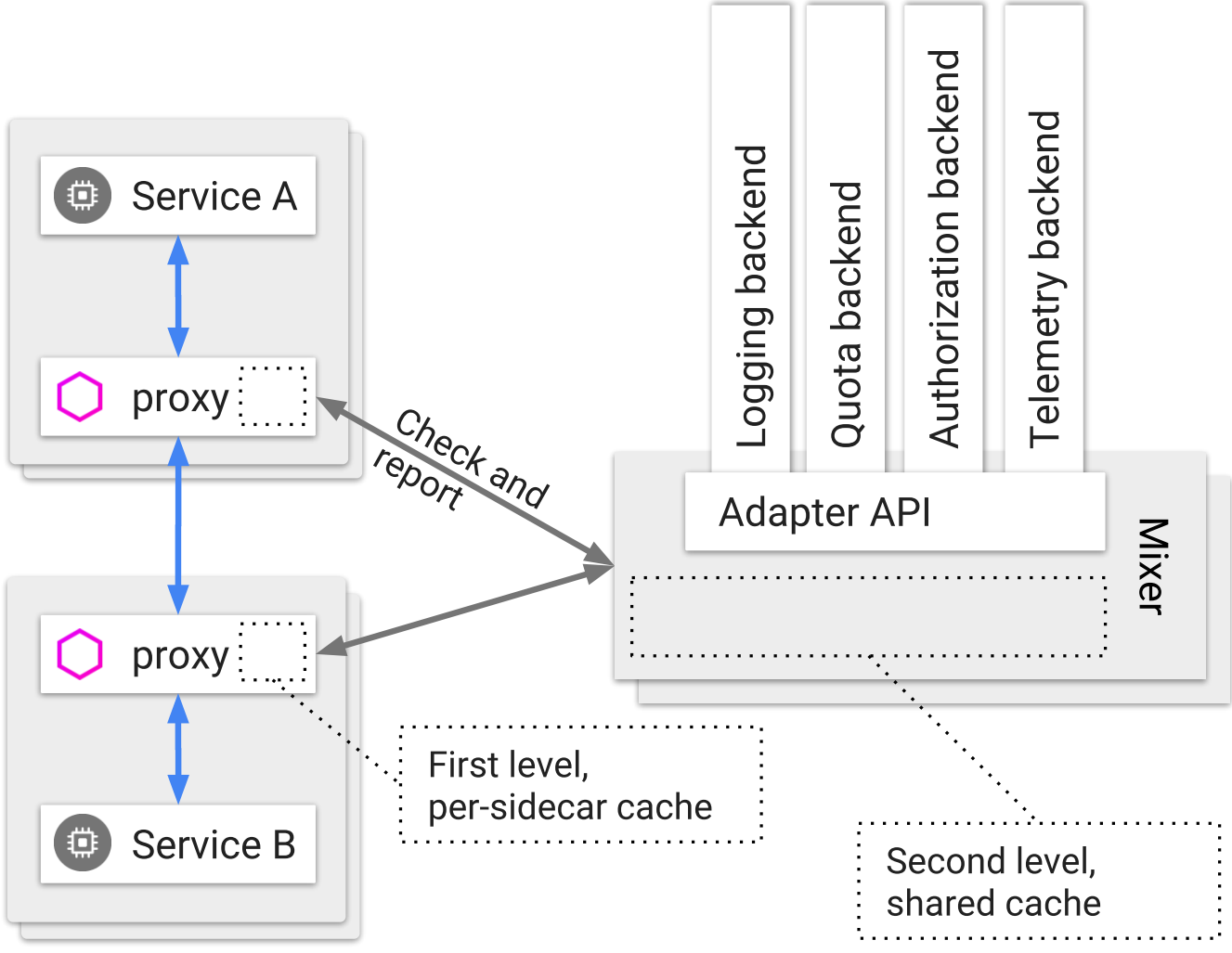
Istio:

1. kube-dns: service domain -> service ip
2. sidecar envoy: service ip -> pod ip

4.3 故障处理

- 超时
- 重试
- 熔断
- ~~降级~~
- ~~隔离~~
- ~~幂等~~
- 限流

4.4 Mixer 架构


Service A
proxy
proxy
Service B
Check and report
Logging backend
Quota backend
Authorization backend
Telemetry backend
Adapter API
Mixer
First level, per-sidecar cache
Second level, shared cache

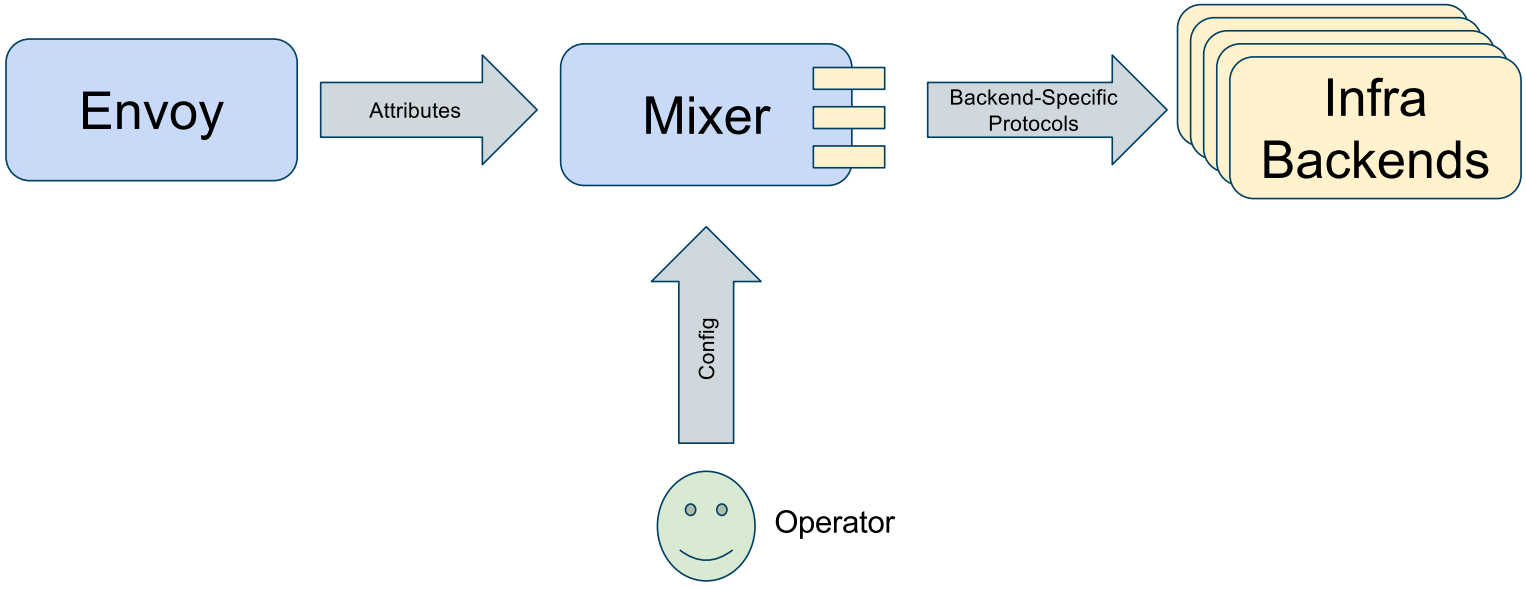
4.5 Mixer Adapter 模型

- Attribute
- Template
- Adapter
- Instance
- Handler
- Rule

Instance是Template定义的数据模板的具体实现, Adapter是Template定义的数据操作的具体实现.

Handler为Adapter提供配置, 代表可运行的Adapter实例.

Rule 决定什么条件下将哪些Instance发给哪个Handler



- template 维护者: 描述数据格式和处理器需要提供的接口格式
- adapter 维护者: 基于具体的后端设施, 选择合适的template, 实现数据处理的业务逻辑
- operator: 定义rule, 管控数据发送的条件和对应的 adapter

> 计算机科学中的所有问题，都可以用另一个层来解决，除了层数太多的问题。

Enjoy Istio! 👍

谢谢

Q&A