下一代微服务框架 SERVICE MESH

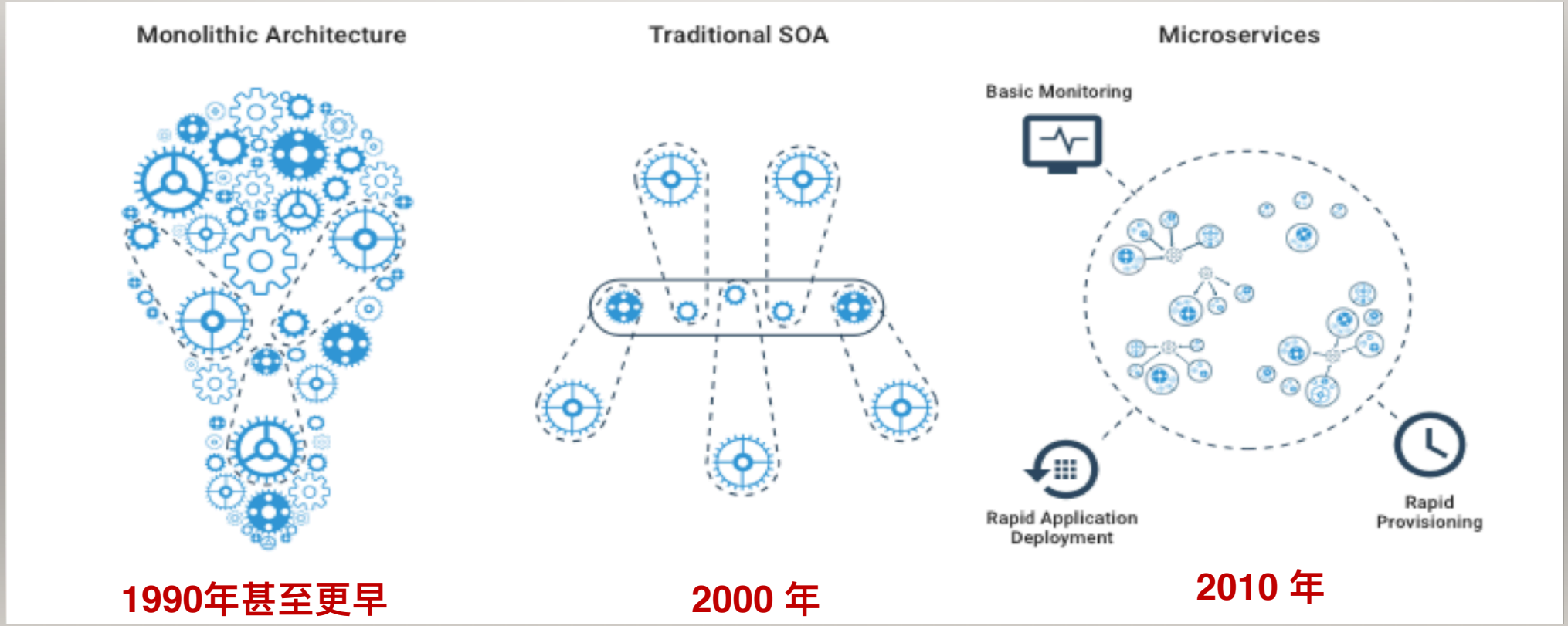
基础架构部　狄卫华

2019 年 2 月 26 号

目 录

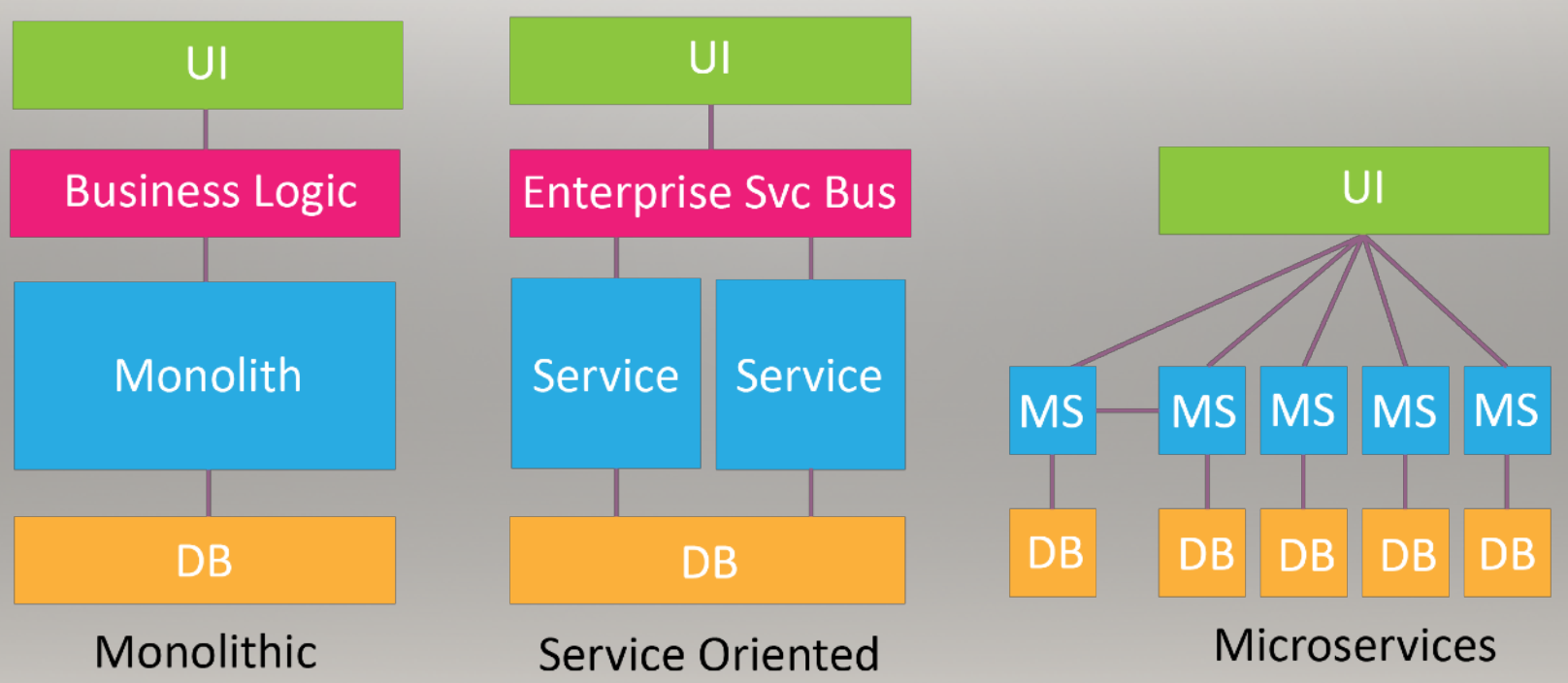
Service Mesh 是什么?

1.1 微服务架构



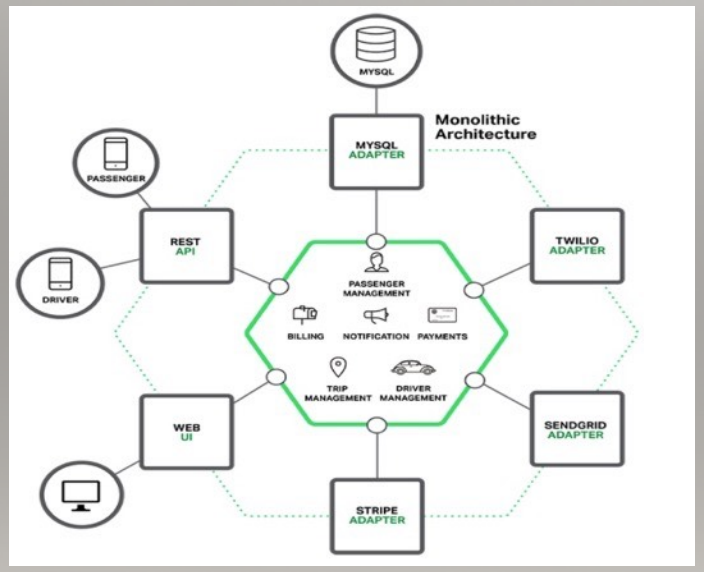
1990年甚至更早　　2000 年　　2010 年

1.1 微服务架构

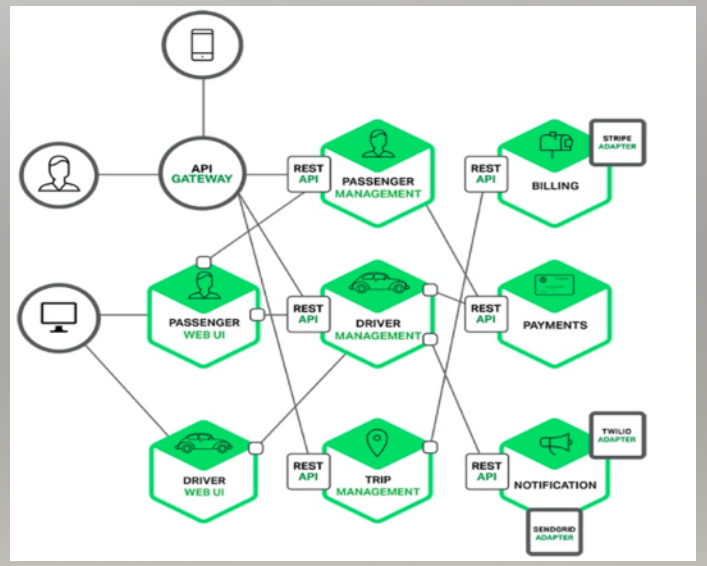

UI
Business Logic
Monolith
DB
Monolithic
UI
Enterprise Svc Bus
Service
Service
DB
Service Oriented
UI
MS
MS
MS
MS
MS
DB
DB
DB
DB
DB
Microservices

1.1 微服务架构

单体



微服务



1.2 SERVICE MESH 定义

- 随着 DevOps 文化的盛行、以 Docker 为代表的容器技术流行和以 Kubernetes 作为容器编排技术的垄断地位形成，使微服务的热度逐年升温，成为互联网行业主流架构。

- 微服务的技术栈以 Java 最为成熟，微服务开发全家桶 Spring Cloud，为以 Spring 为主的 Java 开发人员可以快速将微服务理念落地为案例， 提供一条龙服务，Spring Cloud 在微服务开发中成为主流技术趋势，包括微服务案例和方案书籍等。

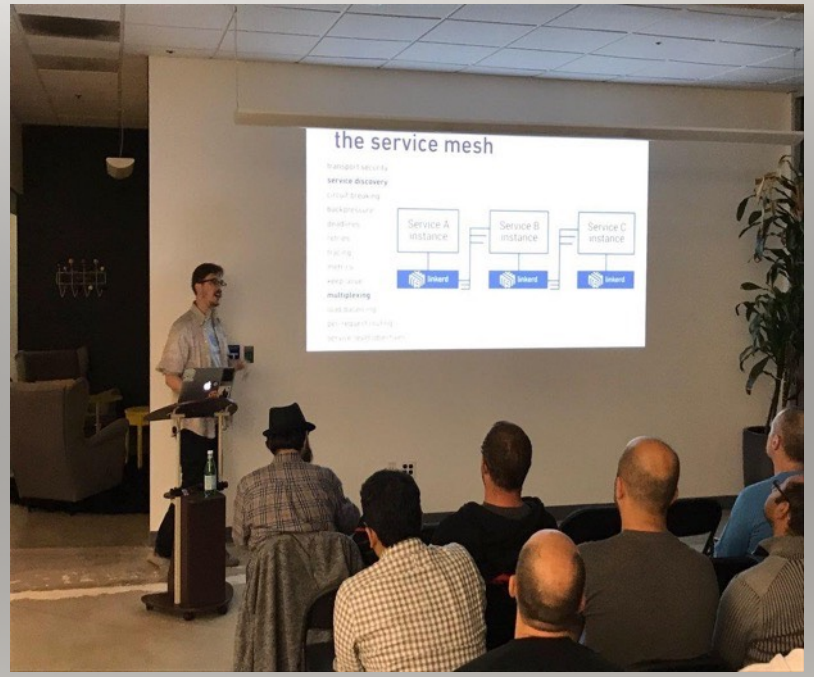
- 小众语言逐渐沦为微服务的看客，直到新一代的微服务技术兴起，这就是今天我们要介绍的 Service Mesh，直接挑战 Spring Cloud 的霸主地位，让 SpringCloud 沦为了看客。

1.2 SERVICE MESH 定义

- Service Mesh 最早是由 Buoyant 公司提出，并在内部大量使用

- 2016年 9 月 29 日第一次公开使用，Buoyant 公司的 CEO Willian Morgan 的 Twitter

中文翻译 “服务啮合层” -> “服务网格”



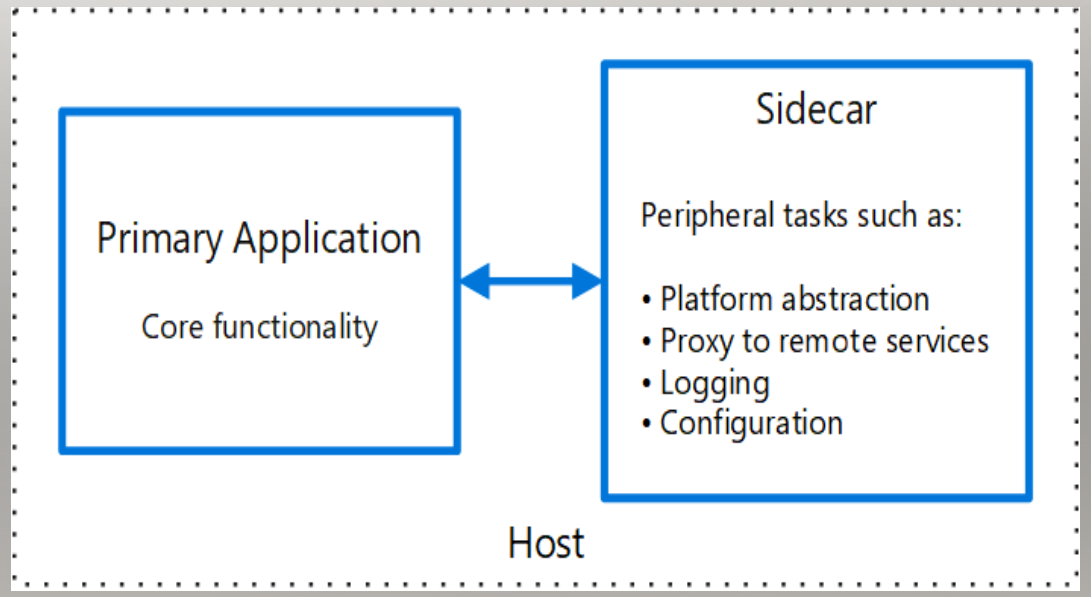
1.2 SERVICE MESH 定义

Willian Morgan （Linkerd 的CEO）给出的定义：

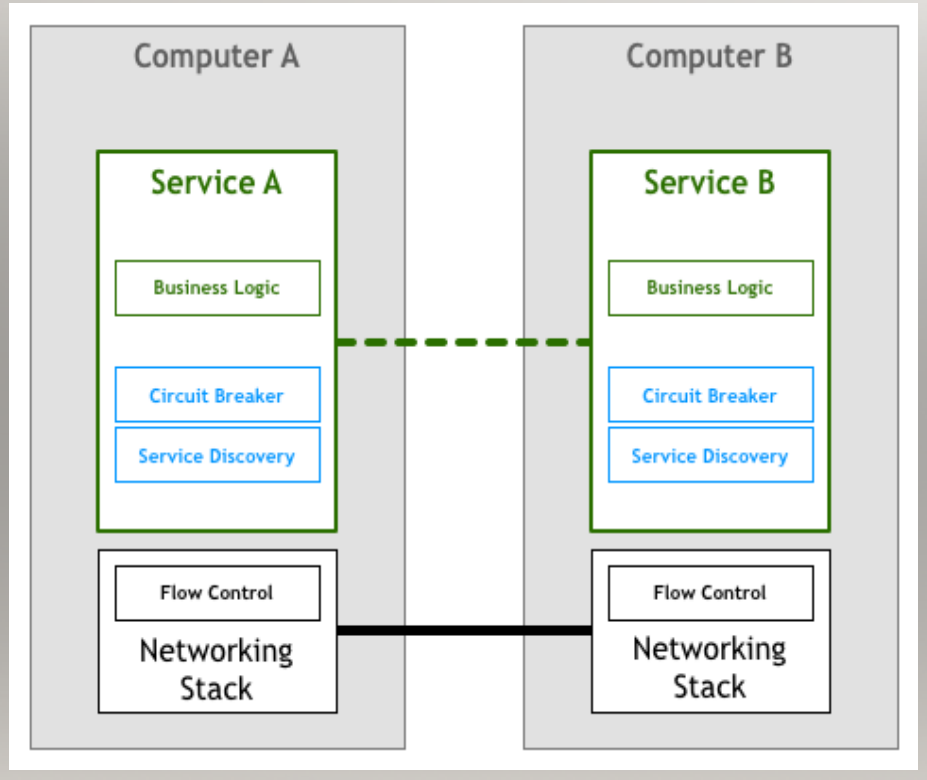
- A service mesh is a dedicated infrastructure layer for handling service-to-service communication. It's responsible for the reliable delivery of requests through the complex topology of services that comprise a modern, cloud native application. In practice, the service mesh is typically implemented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

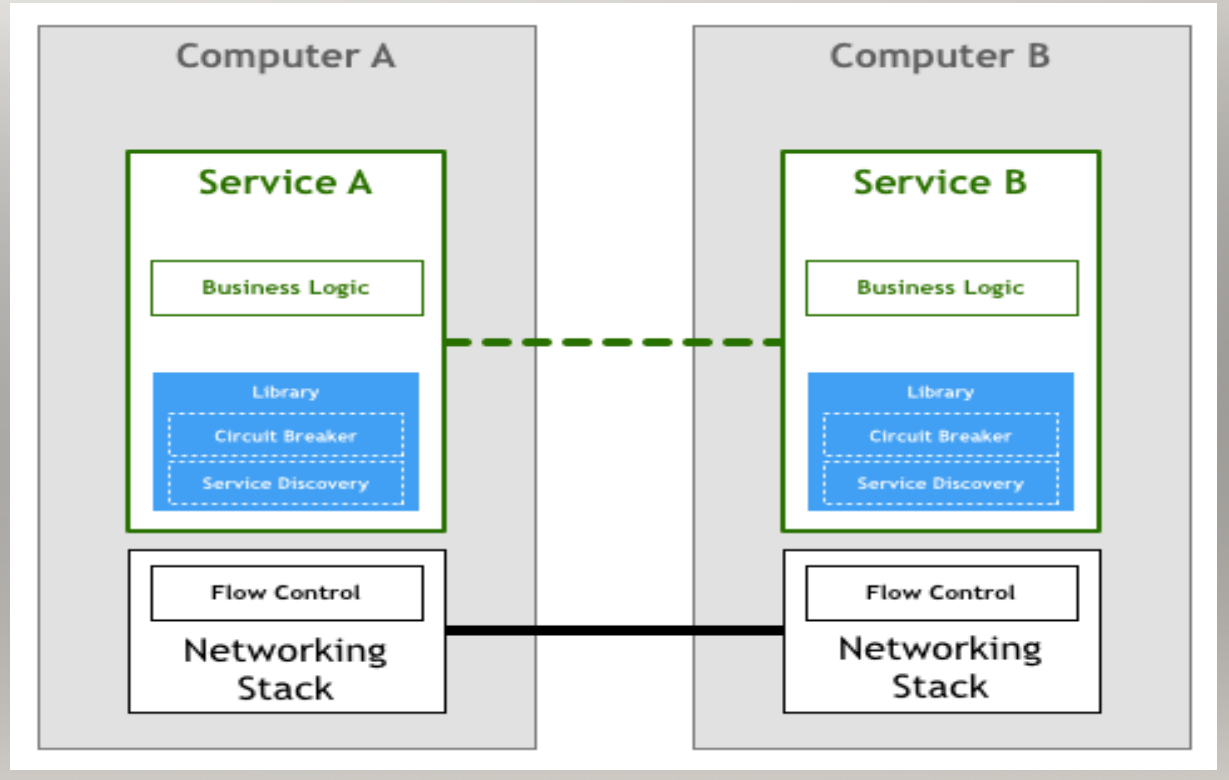
- 服务网格是一个**基础设施层**，用于处理服务间通信。云原生应用有着复杂的服务拓扑，服务网格负责在这些拓扑中实现**请求的可靠传递**。 在实践中，服务网格通常实现为一组**轻量级网络代理**，它们与应用程序部署在一起，而**对应用程序透明**。**（非侵入性 Sidecar 模式，微服务分水岭）**

1.3 SIDECAR PATTERN

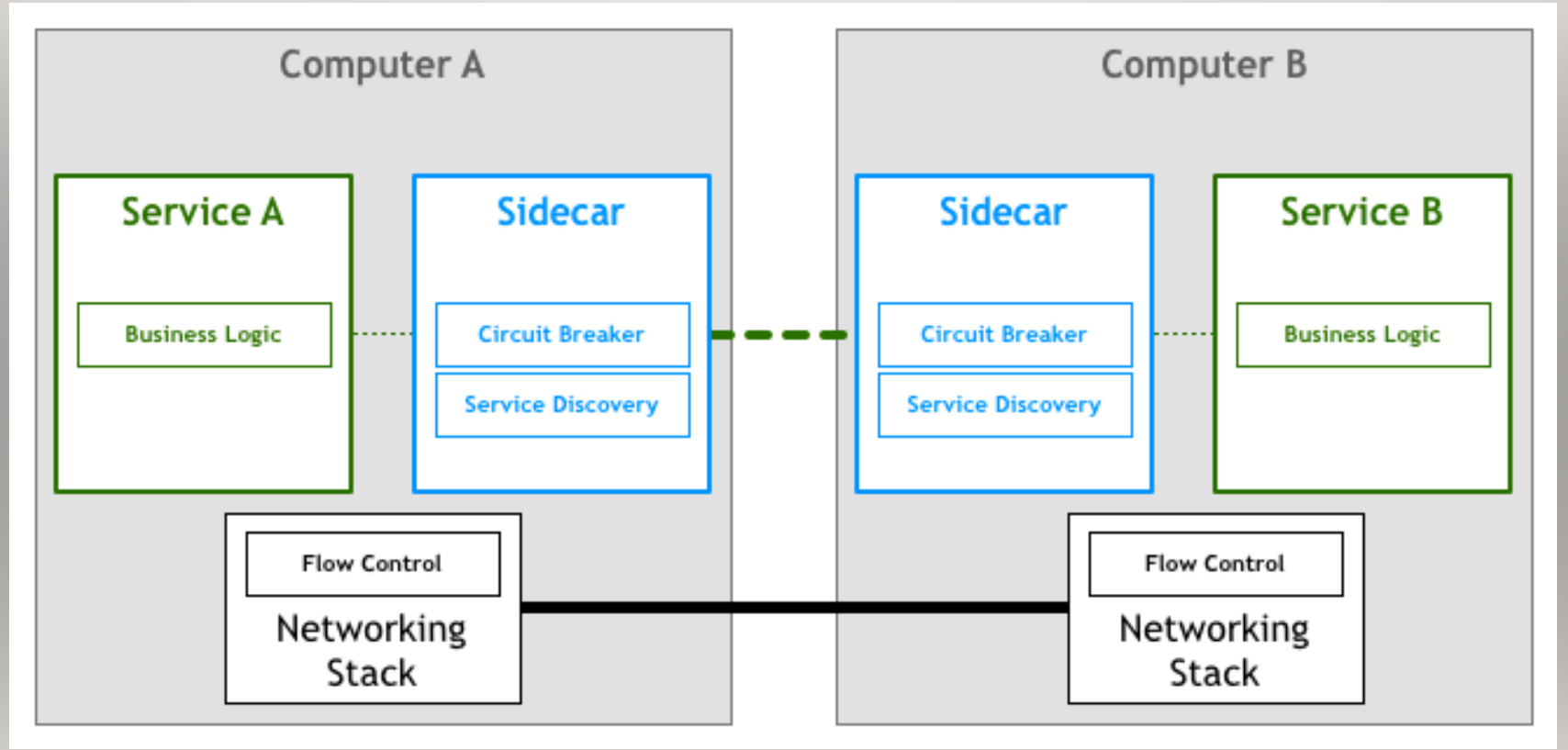


1.3 SIDECAR PATTERN

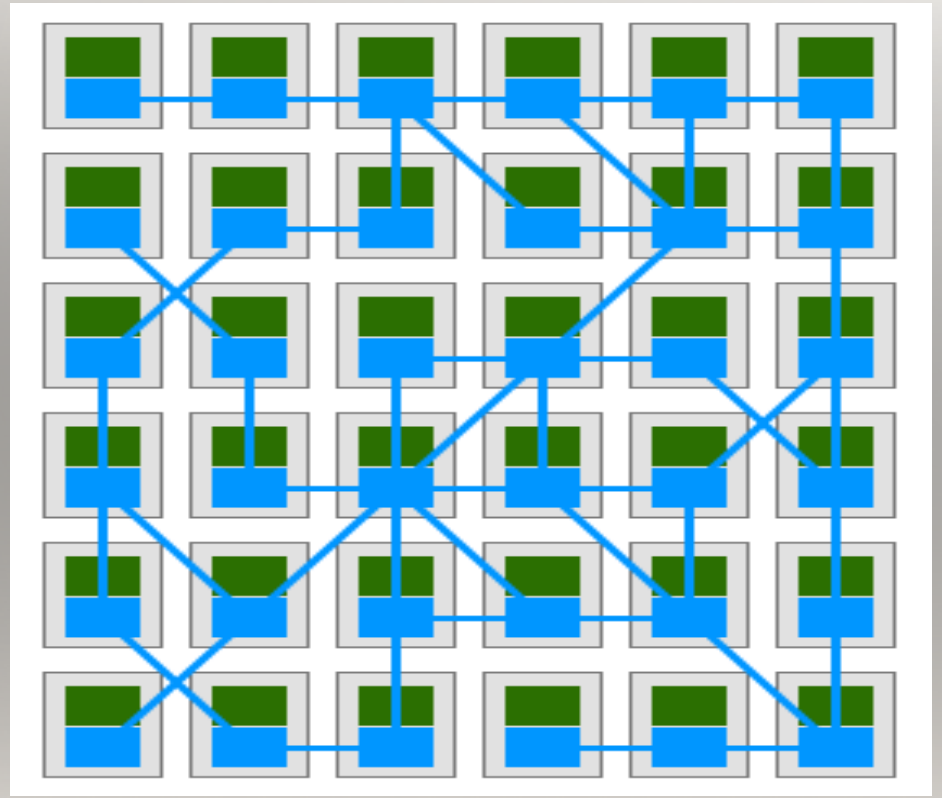


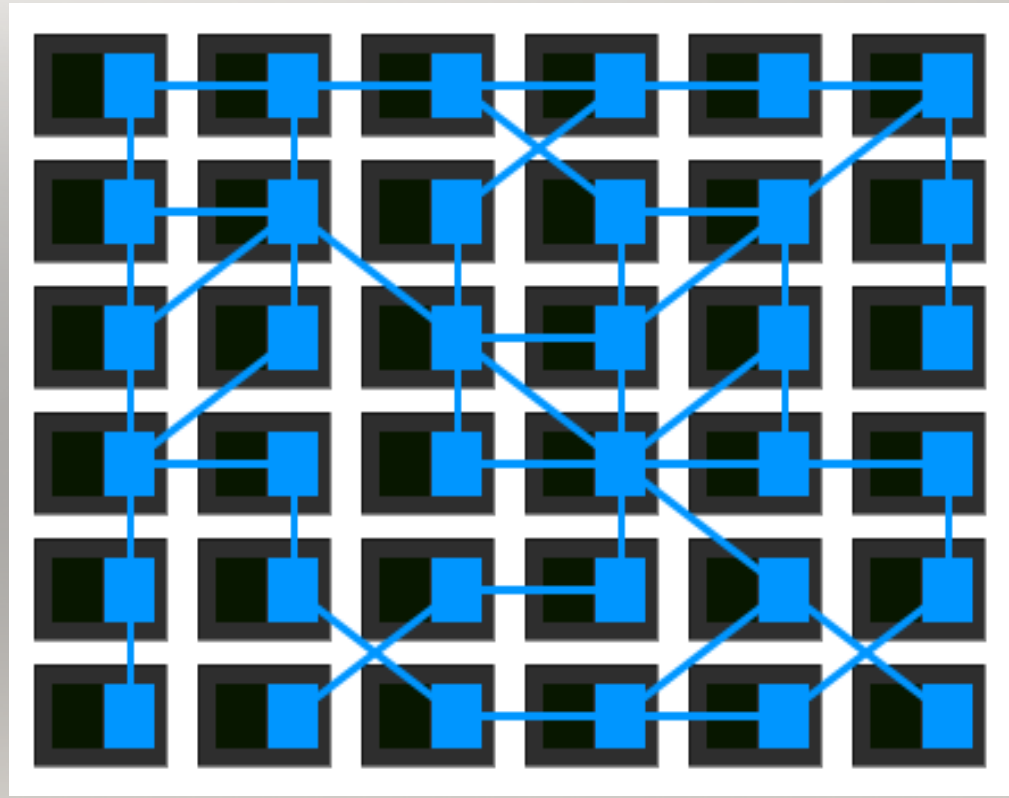


1.3 SIDECAR 模式

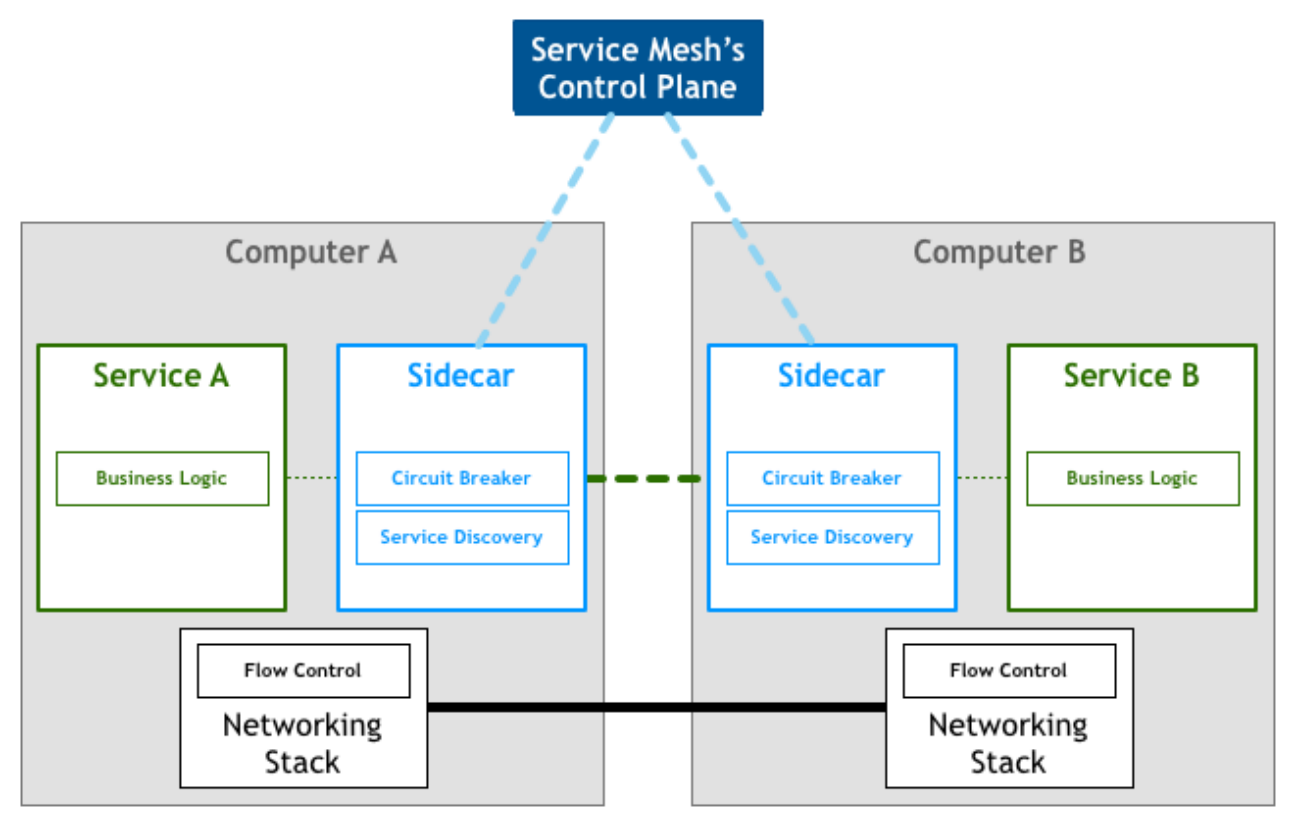


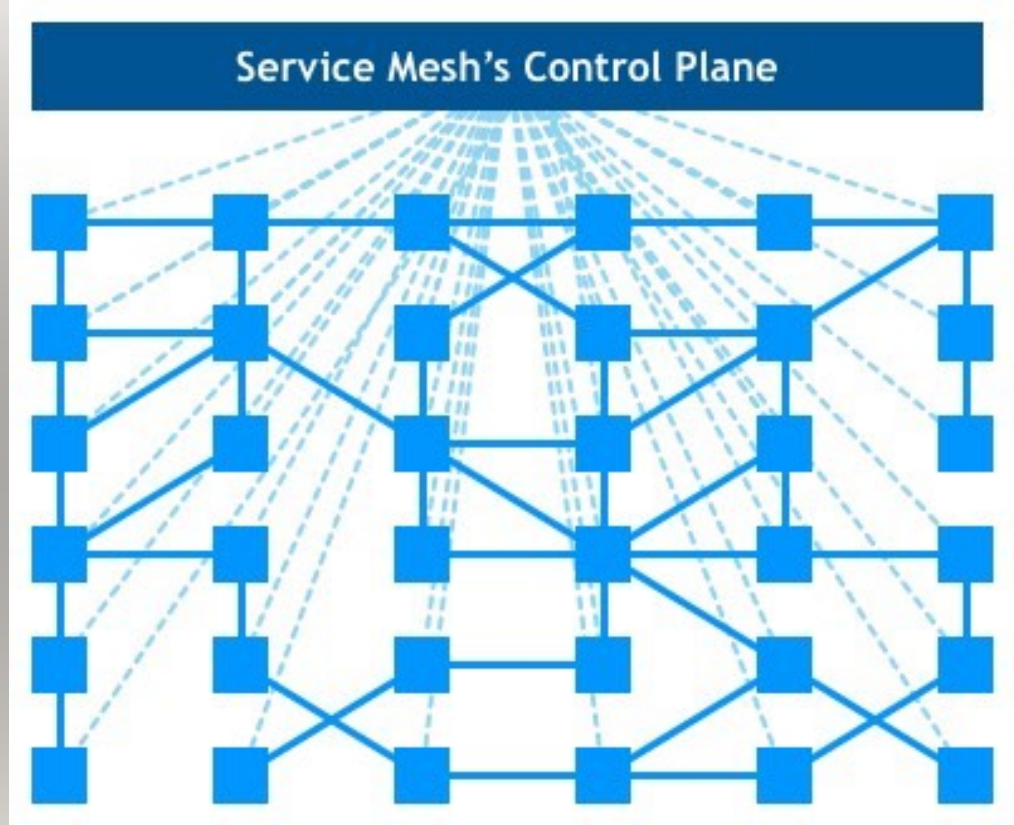
1.4 SERVICE MESH 1.0





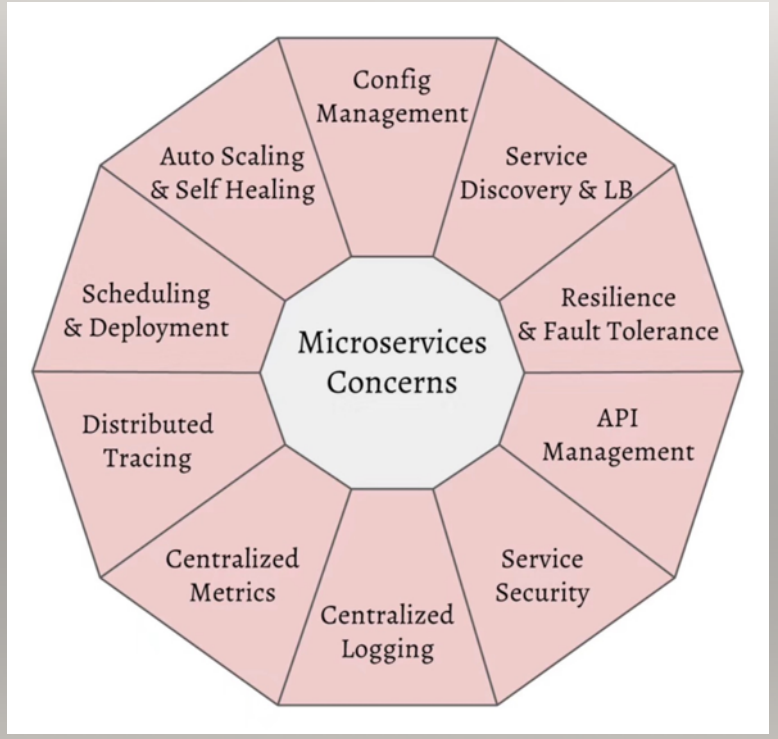
1.4 SERVICE MESH 2.0

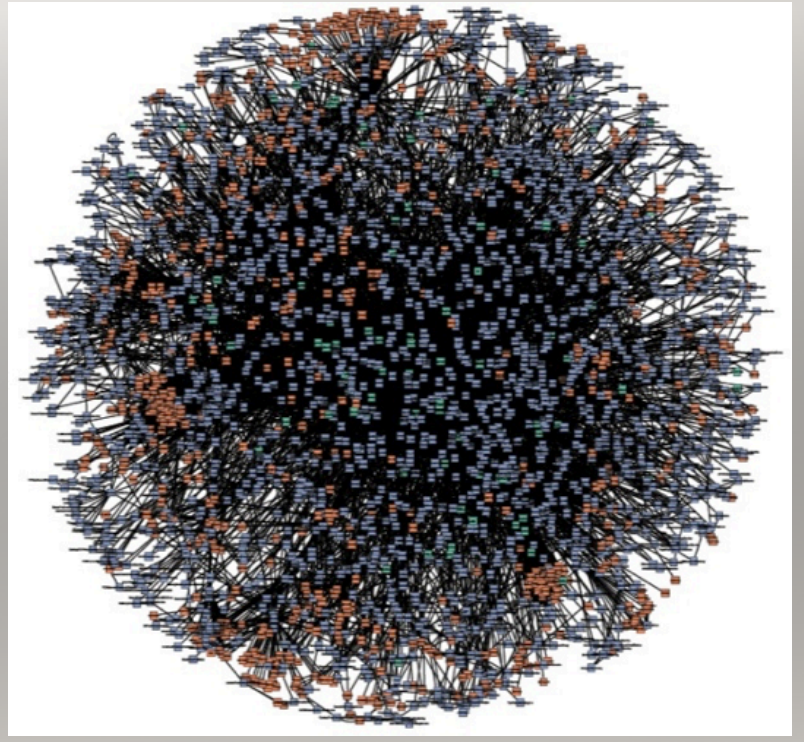




Service Mesh 演进之路

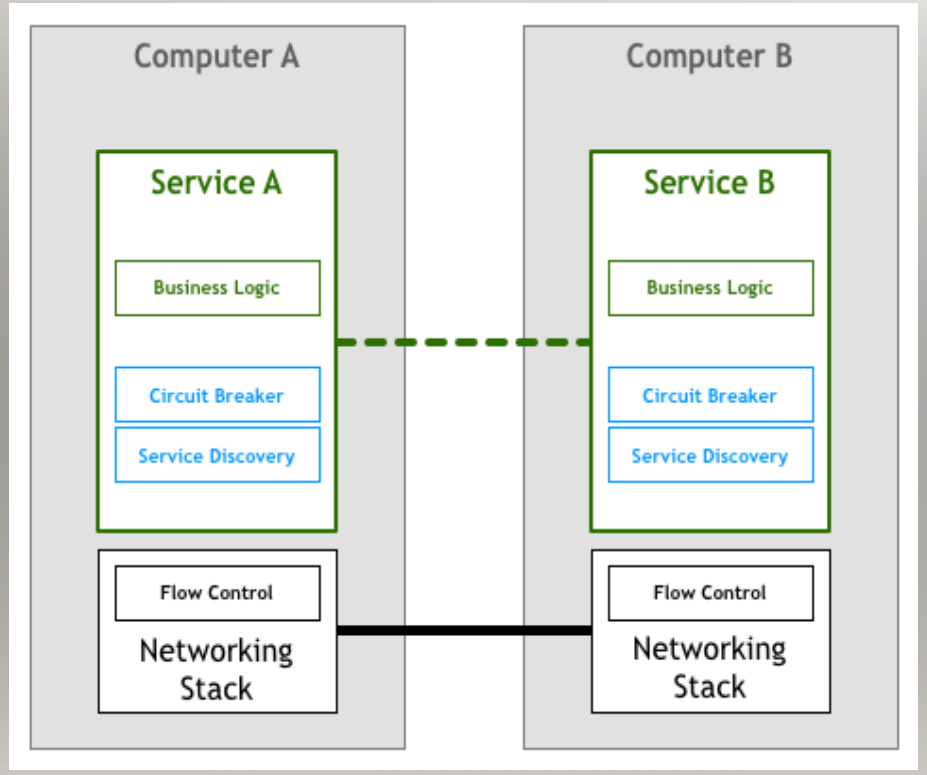
2.1 呼之欲出





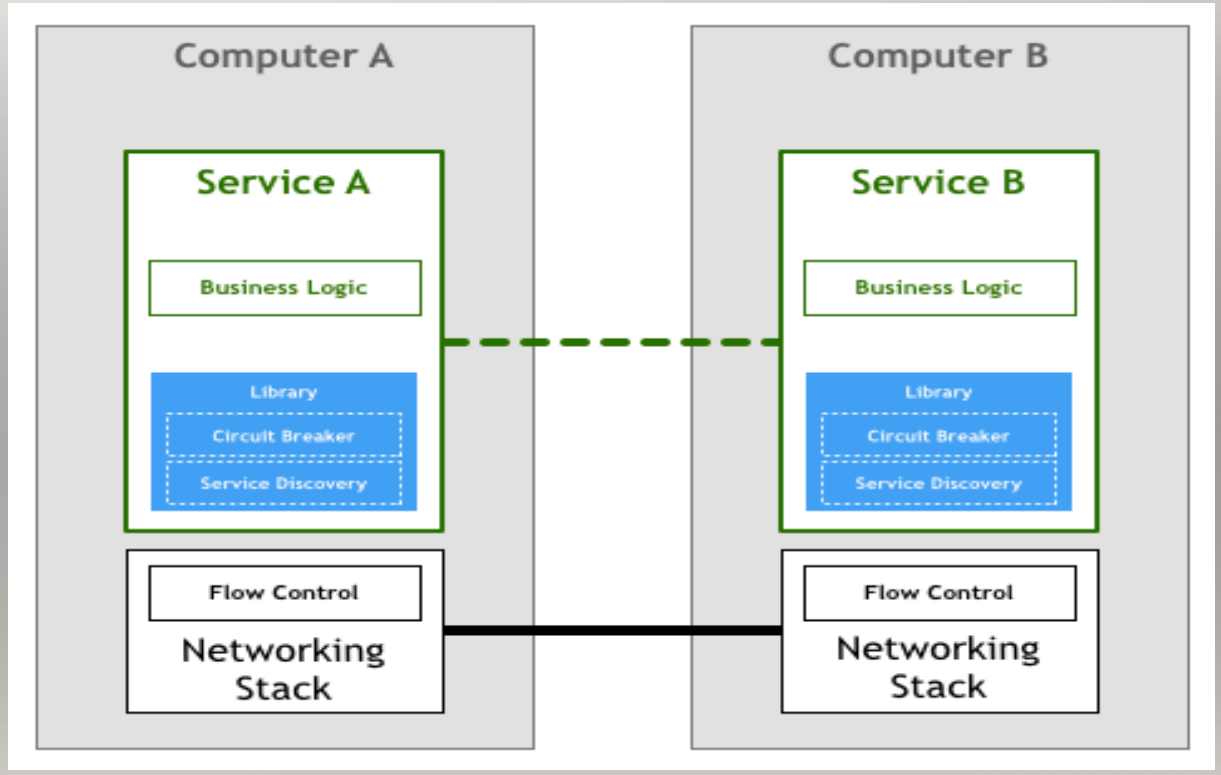
需要更加强大的基础设施支持、调用路径变长、服务间依赖更容易连锁雪崩

2.1 呼之欲出



原始版

基础代码与业务代码混写，服务发现、负载均衡、重试、熔断等



类库和SDK

专门的SDK处理基础代码，例如 Spring Cloud或者公司自己研发的 SDK

基本功能

- 服务注册与服务发现
  - 主动健康检查
- 负载均衡
  - 随机轮询之外的高级算法
- 故障处理和恢复
  - 超时
  - 熔断
  - 限流
  - 重试
- RPC 支持
- HTTP/2 支持
- 协议转换/提升

高级功能

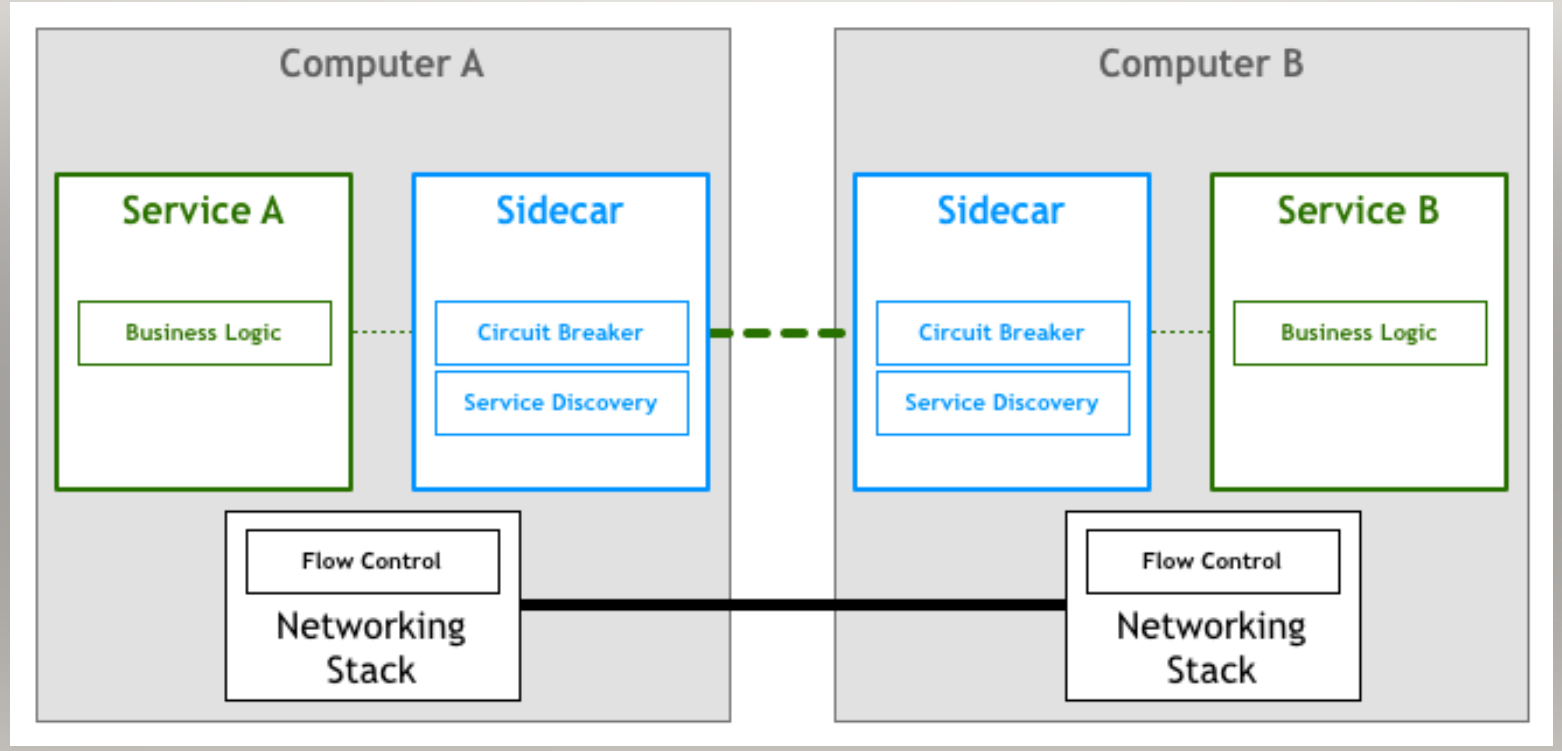
- 加密
  - 密钥和证书的生成，分发，轮换和撤销
- 认证/授权/鉴权
  - OAuth
  - 多重授权机制
    - ABAC
    - RBAC
    - 授权钩子
- 分布式追踪
- 监控
  - 日志
  - 度量（Metrics）
  - 仪器仪表（instrumentation）

运维测试类

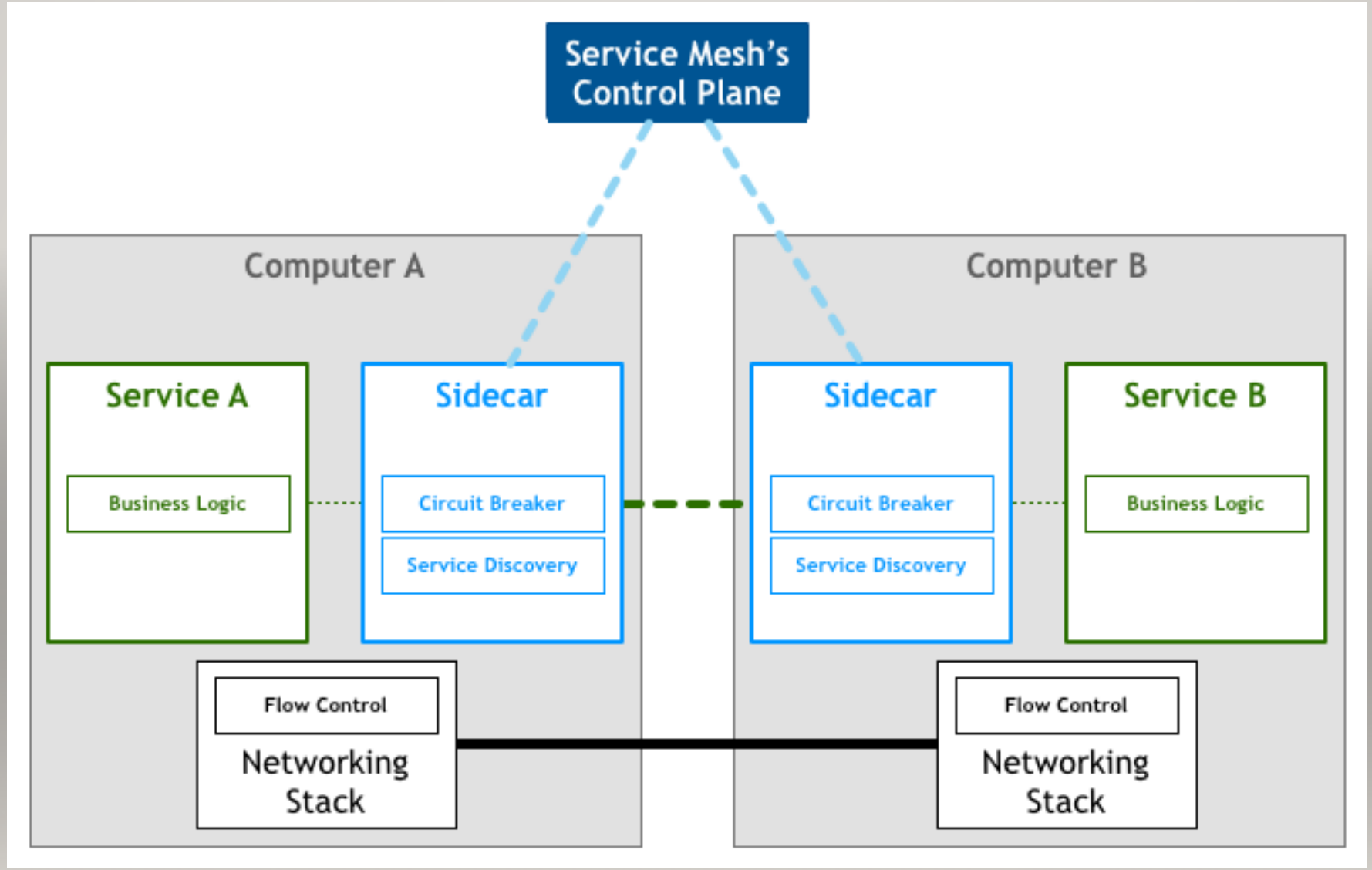
- 动态请求路由
  - 服务版本
  - 分段服务(staging service)
  - 金丝雀(canaries)
  - A/B测试
  - 蓝绿部署(blue-green deploy)
  - 跨DC故障切换
  - 黑暗流量(dark traffic)
- 故障注入
- 高级路由支持
  - 高度可定制：script, DSL
  - 可配置的规则

专业性强、复杂度高、业务同学苦不堪言，说好的开发简单的微服务模式呢?

2.1 呼之欲出



业务逻辑与基础架构**完全分离**，**独立演进**，Sidecar 形成数据平面

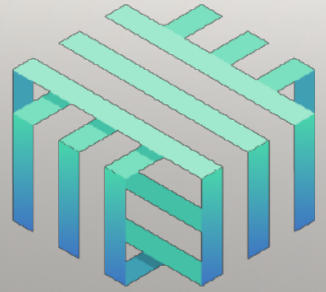

Service Mesh's Control Plane
Computer A
Computer B
Service A
Business Logic
Sidecar
Circuit Breaker
Service Discovery
Sidecar
Circuit Breaker
Service Discovery
Service B
Business Logic
Flow Control
Networking Stack
Flow Control
Networking Stack

2.1 呼之欲出

- 业务开发者只关心业务逻辑
- Service Mesh 实现了基础架构功能，开源社区做强做大，**拿来即可用**
- Sidecar 独立升级，对应业务服务透明
- 微服务真正的多语言化，小众语言扬眉吐气 Go/Node.js/Python/Rust

鱼与熊掌兼得

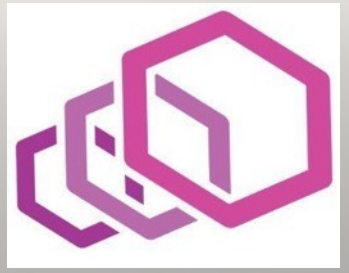
2.2 崭露头角 1.0 时代



Linkerd

- 来自 Buoyant，Scala 语言 JVM
- Service Mesh 概念创造者
- 2016 年 1 月，0.0.7 发布 9 月， 0.8.0
- 2016 年 10 月，发表一系列文章，积极布道 Service Mesh
- **2017 年 1 月，加入 CNCF，社区正式接纳**

主业：Service Mesh，技术创新



Envoy

- 来自 Lyft，c++语言
- 2016 年 9月，1.0 版本发布
- 2017 年 9 月，加入 CNCF
- 2018 年 11月， CNCF 毕业

主业：打车软件，Uber 齐名

2.3 群雄逐鹿 2.0 时代

Istio

Conduit/Linkerd 2.0

KongMesh

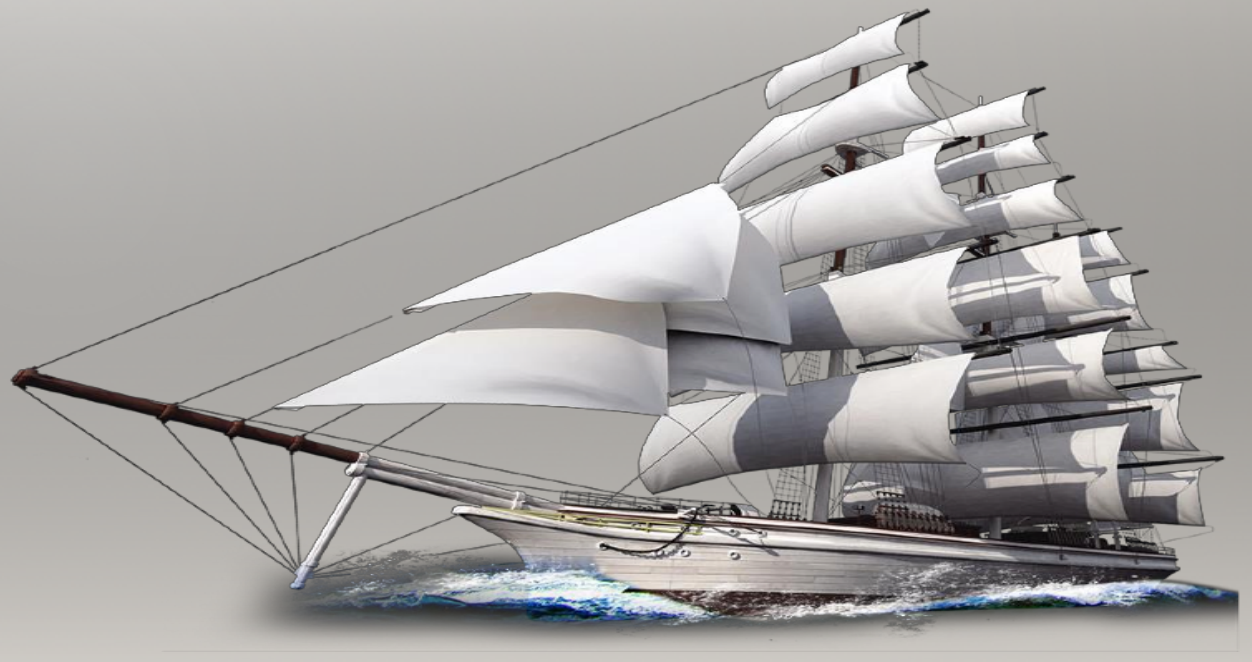
Envoy

nginxMesh

2.3 群雄逐鹿 2.0 时代 之 ISTIO

Istio

- 来自 Google，IBM 和 Lyft，Go语言
- Service Mesh 集大成者
- 2017 年 5 月，0.1 版本发布
- 2018 年 7 月，1.0.0 版本发布



Google Cloud Platform
IBM Cloud Platform
Lyft Envoy Team

出身名门

Linkerd/nginmesh选择集成而非对抗
Red hat/Pivotal/Weaveworks/Tigera/Datawire
背后有CNCF，还有即将一统江湖的k8s

社区支持

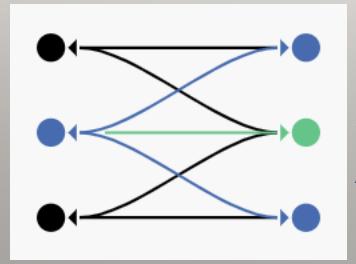
微服务/容器如日中天，Cloud Native大势已成
传统企业互联网技术转型大潮汹涌却又先天技术积累不足

运势而生

Istio的设计理念新颖前卫，极富创意
开发团队实力惊人
功能齐全
有望成为下一个**k8s**

超凡实力

2.3 群雄逐鹿 2.0 时代 之 ISTIO

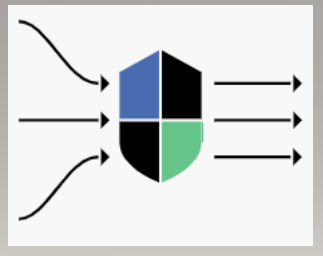


连接

智能控制服务之间的流量和 API 调用，进行一系列测试，并通过红/黑部署逐步升级

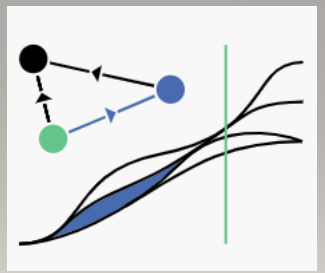
安全

通过托管身份验证、授权和服务之间通信加密自动保护您的服务。



控制

应用策略并确保其执行使得资源在消费者之间公平分配。



观测

通过丰富的自动跟踪、监控和记录所有服务，了解正在发生的情况。

- 服务发现
- 负载均衡
- 请求路由
- 故障处理
- 故障注入
- 规则配置

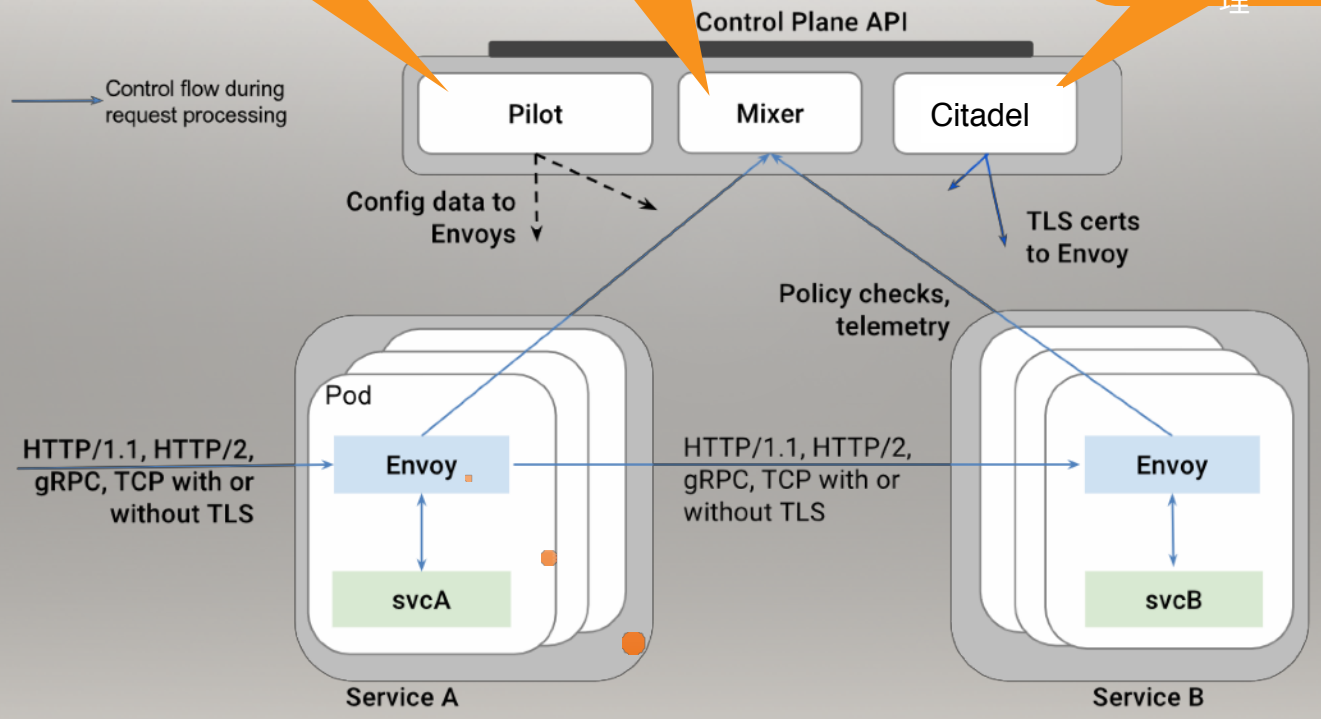
服务发现，配置，底层抽象，编码规则，管理 Envoy

- 前提条件检查：如认证，黑白名单，ACL检查
- 配额管理：限流
- 遥测报告: 日志，监控，metrics

微服务和基础设施的中介层，将策略决策从应用移出并用配置替代

提供服务间认证和终端用户认证使用交互TLS，内置身份和凭据管理

- 升级流量（加密）
- 身份认证
- 密钥管理
- 通讯安全
- 访问控制
  - 基于属性
  - 基于角色
  - 授权钩子
- 审计
- 监控资源的使用者



控制平面

数据平面

标配Envoy，但是Linkerd和nginmesh都在istio集成

2.3 群雄逐鹿 2.0 时代 之 CONDUIT/LINKERD 2.X

- Conduit 是 Buoyant 公司 2017 年 12 月推出的轻量级 ServiceMesh 2.0 解决方案，为了对抗 Istio 而生，也是 Buoyant 公司为生存的绝地反击，控制层面采用 Go 开发，数据平台采用 Rust 开发；
- 主打轻量级，功能采用迭代方式逐步推进，有别于 Istio 的豪华跑车概念版；
- 2018 年 7 月，0.5.0 版本终结修改为 Linkerd 2.0
- 2018 年 9 月， Linkerd 2.0 GA，主打 ServiceProfile 的概念，简化运维操作

2.3 群雄逐鹿 2.0 时代 之 NGINXMESH

- 2017 年 9 月 nginxMesh 0.1.6
- 2017 年 12 月 nginxMesh 0.3
- 2018 年 7 月 nginxMesh 0.7.1

走走停停，定位数据平面，空间小，前有 Envoy 难以逾越，自己本身努力度有限，因此被称为不努力的 “小三”

2.3 群雄逐鹿 2.0 时代 之 KONGMESH

- 2017 年 12月 给自己打上 ServiceMesh 的标签
- 2018 年 12 月 KongMesh 1.0 GA

推出从控制平面和数据平面的整套解决方案，功能上简单，扩展性良好，功能相对于 Istio + Envoy 略显单薄

2.3 群雄逐鹿 2.0 时代 之 ENVOY

- 2016 年 9 月 1.0 开源
- 2017 年 5 月 称为 Istio 数据平面的标准集成
- 2017 年 9 月加入 CNCF
- 2018 年 11 月 CNCF 毕业
- 2018 年 12 月 1.9.0 版本

稳扎稳打，精心打磨细节，推出 XDS 协议，已逐步向数据平面的标准发展，成为其他数据平面竞争者不可逾越的鸿沟；其出色的稳定性和丰富的功能，也在不断蚕食 Nginx 和 Kong 的市场，在 Service Mesh 2.0 这场战争中或许会称为最大的赢家。

2016 年

1 月

- **Linkerd 0.0.7 开源 Buoyant**

9 月

- **Envoy 1.0 开源 Lyft**
- Service Mesh 走入社区

10 月

- Buoyant 系列文章布道

2017 年

1 月

- **Linkerd 加入 CNCF**

4 月

- Linkder 1.0.0 发布
- **Linkerd 千亿次请求**

5月

- **Istio 0.1 发布 Google + IBM**

7 月

- Linkerd 1.1.1 集成 Istio
- Knative 0.1 开源

9 月

- **Envoy 加入 CNCF**
- **nginxMesh 0.1.6**

12月

- Istio 0.4 不满足生产，摊子大
- **Conduit 0.1.0**
- **nginxMesh 0.3**
- 国内华为和微博分享产品

2018 年

4-5 月

- Conduit 0.4.0 仅支持 k8s
- Istio 0.7.0
- Envoy 1.6.0

6 月

- **Istio 0.8.0 LTS**
- Conduit 0.4.2 支持全部 k8s workload

7 月

- **Istio 1.0.0 发布**
- **Conduit 0.5.0 (终结）-> Linkder 2.0**
- **nginxMesh 0.7.1 走走停停**

9 月

- **Linkerd 2.0 GA**
- **Istio 1.1 一度延期**

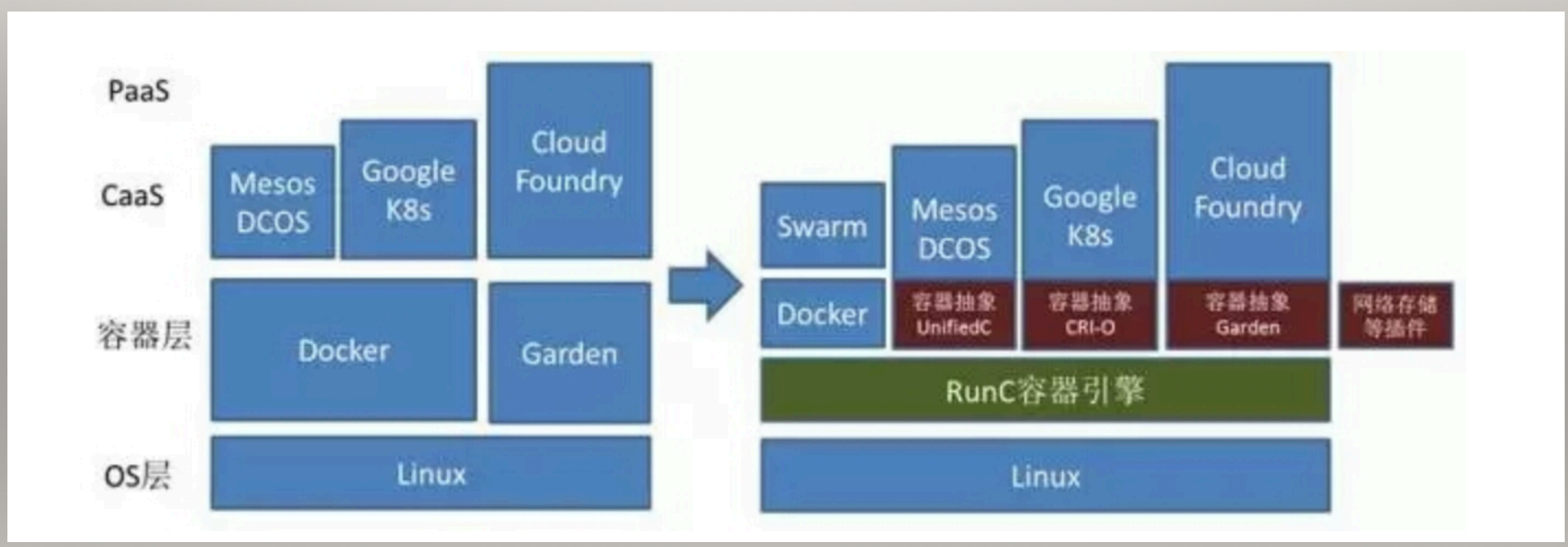
11月

- Envoy CNCF 毕业
- Knative 0.2.0

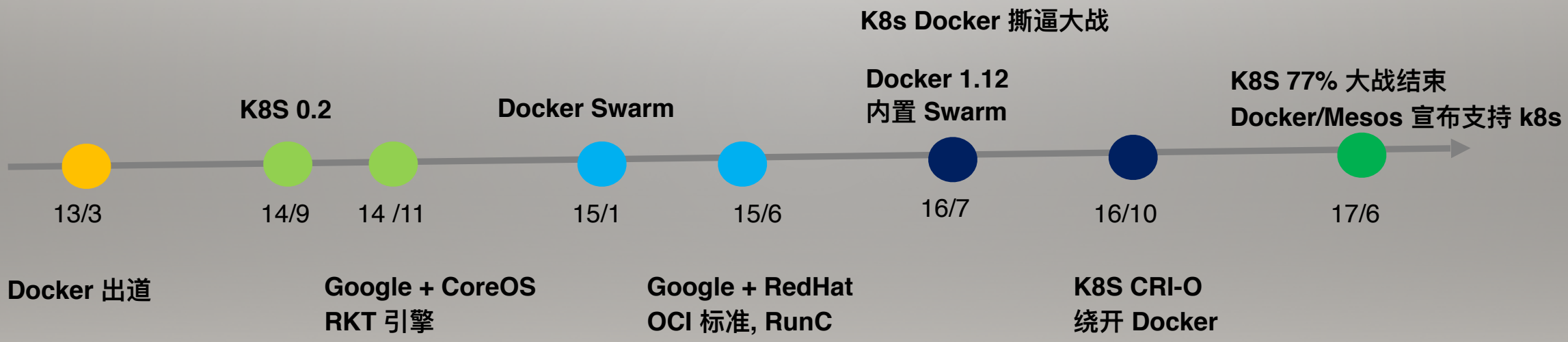
12月

- **Linkerd 2.1 Service Profile**
- **KongMesh 1.0 GA**

2.4 题外话：没有硝烟的战争



2.4 题外话：没有硝烟的战争



Google Cloud Platform，有多强，不用解释

Kubernetes

- 搞定容器
- 状态：Done!
- 市场：已成为事实标准

Istio

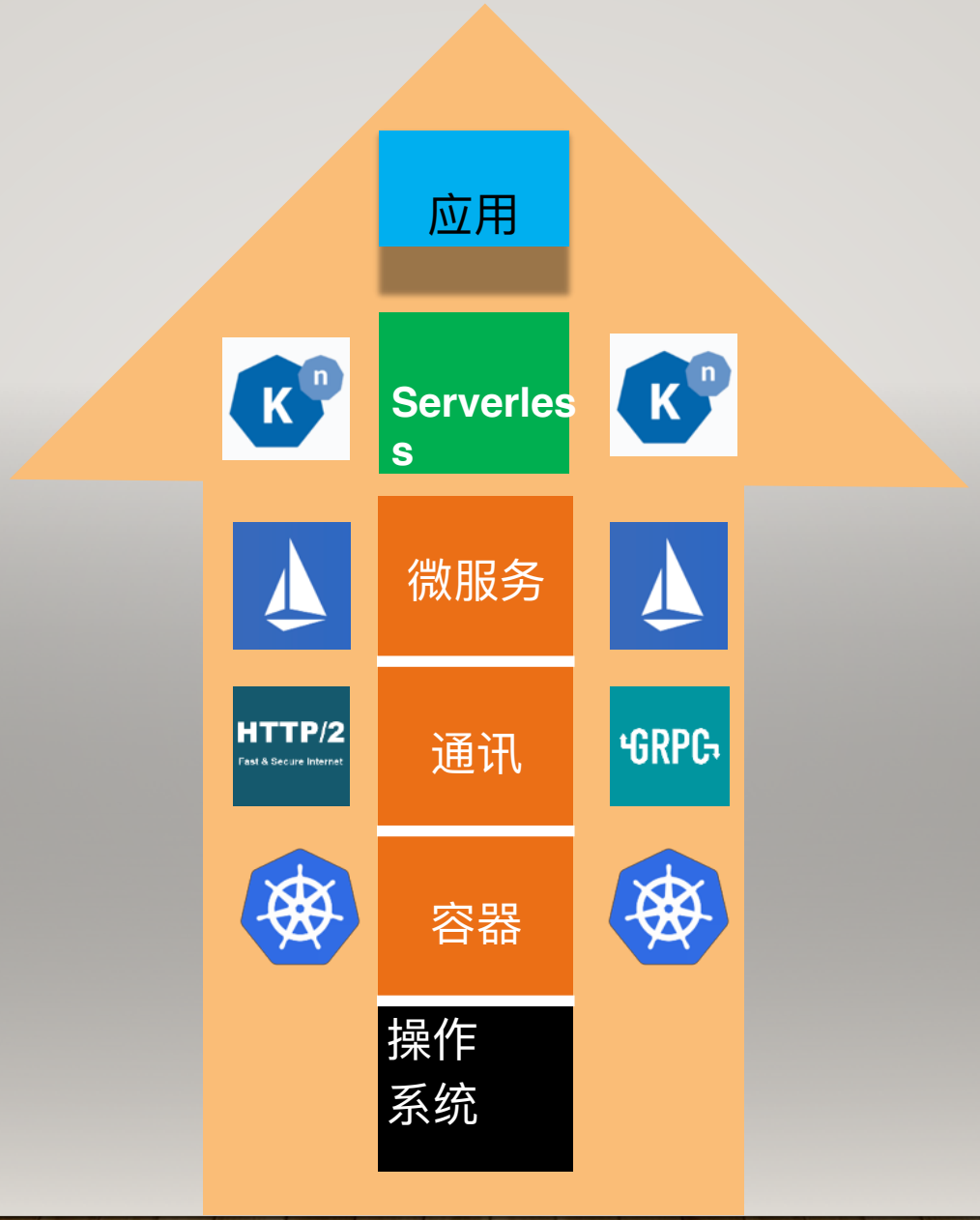
- 搞定下一代微服务
- 状态：Ongoing……
- 市场：培育中，无强力竞争对手

gRPC

- 搞定下一代RPC通讯
- 状态：Almost Done!
- 市场：培育中，无强力竞争对手

HTTP/2

- 搞定下一代HTTP
- 状态：Done!
- 市场：已成为w3c规范


应用
Serverless
微服务
通讯
容器
操作
系统
HTTP/2
Fast & Secure Internet
GRPC

2.4 题外话：没有硝烟的战争

占领制高点，联合巨头，指定标准，留出空间

Service Mesh 未来趋势

3.1 SERVICE MESH 未来趋势

高调入场，抢占先机

命悬一线，奋力一战

走走停停，沽名钓誉

独树一帜，尚有可为

场景演进，精细经营

3.2 SERVICE MESH 国内的发展

- 蚂蚁金服 SOFAMesh + SOFAMosn
- 新浪微博 WeiboMesh
- 华为 Mesher 与 ASM
- 阿里 Dubbo Mesh
- 腾讯 Tencent Service Mesh

3.2 SERVICE MESH 国内的发展

- 蚂蚁金服 SOFAMesh+SOFAMosn
- 新浪微博 WeiboMesh
- 华为 Mesher 与 ASM
- 阿里 Dubbo Mesh
- 腾讯 Tencent Service Mesh

数据平面

- **基于 Envoy**
  - Tencent ServiceMesh
  - 阿里 Dubbo Mesh

- **自研**
  - 微博 WeiboMesh
  - 蚂蚁金服 SOFAMosn

控制平面

- **基于 Istio 扩展或精简**
  - 蚂蚁金服 SOFAMesh
  - 华为 ASM
  - Tencent ServiceMesh （Pilot）
  - 阿里 Dubbo Mesh （Pilot）
- **自研**
  - 微博 WeiboMesh
  - 华为 Mesher

我们与 ServiceMesh

NEGRI

4.1 优势

- **接入成本低：**业务几乎无侵入（原本的 HTTP Restful API 只需要修改请求的地址为本地 Sidecar 地址和在 HTTP Header 指定服务名即可享受所有服务治理能力; gRPC 则只需要修改调用地址为本地，不需要其他任何修改）
- **开发语言无关** Go/PHP/Node/Python/Rust，无需对每种语言分别维护框架
- **业务无关**
- **无感知升级：**业务方无需变动任何代码,就可以拥有 Sidecar 最新的插件能力（几十秒全量升级，业务无感知，无抖动）
- **统一的插件编写体系：**无需区分Client 还是 Serve r端中间件, 统一抽象为 Server 端中间件,编写一次,多端运行，可以对每个服务的客户端和服务端甚至是单个节点配置不同的流量控制策略，并**实时生效**
- **资源隔离，S**idecar 进程本身使用 cgGroup 限制资源(CPU,内存等)使用，不影响业务机器

4.2 功能

- 支持服务发现/注册
- 完善的 Router 层插件机制
- 动态配置变更
- 基于 Fork 进程方式实现的热重启
- 强大的实时生效的限流，熔断，灰度，负载均衡，故障转移，重试，服务间调用权限控制，故障注入等服务治理能力
- 对公司自研业务支持，ABTest、Trace、Auth、加解密等功能的集成
- 可以画出全局服务依赖关系图，实时，延时等统计

4.3 现状与未来规划

接入服务：

LogServer 和 金币详情服务

未来发展：

- 支持 XDS 协议，集成 Isito,，支持 Isito 周边工具集
- 支持 Redis、Mysql、Nsq、Kafka等本地连接池

http://km.qutoutiao.net/pages/viewpage.action?pageId=73504494

THANK YOU!

你的意见，对我们非常宝贵，期待大家的意见！