CloudWeGo

Bookinfo

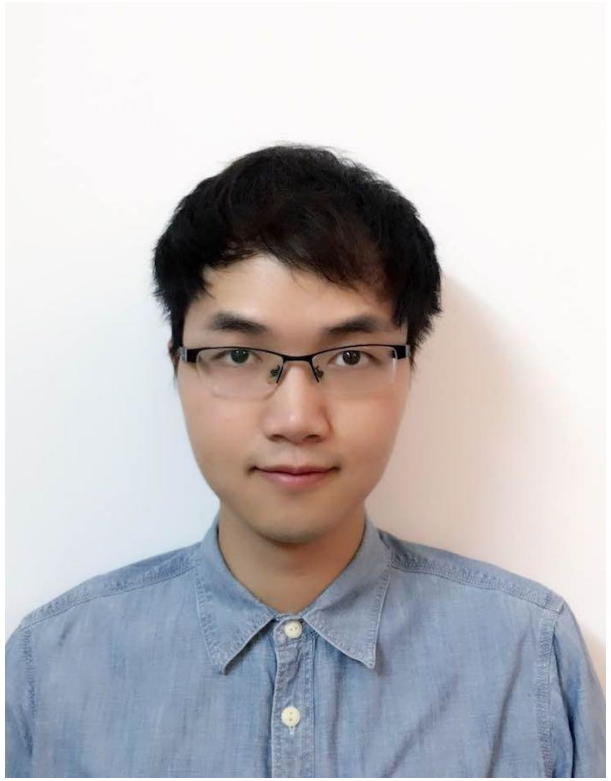
基于 CloudWeGo 重写 Istio 经典 demo

胡文（基础架构-火山引擎）

CloudWeGo 开源团队出品

2023/02/22

个人介绍



胡文

- 字节跳动基础架构研发工程师
- 长期耕耘在 可观测性、eBPF、ServiceMesh、服务框架等领域
- 云原生相关开源社区爱好者

目录

CONTENT

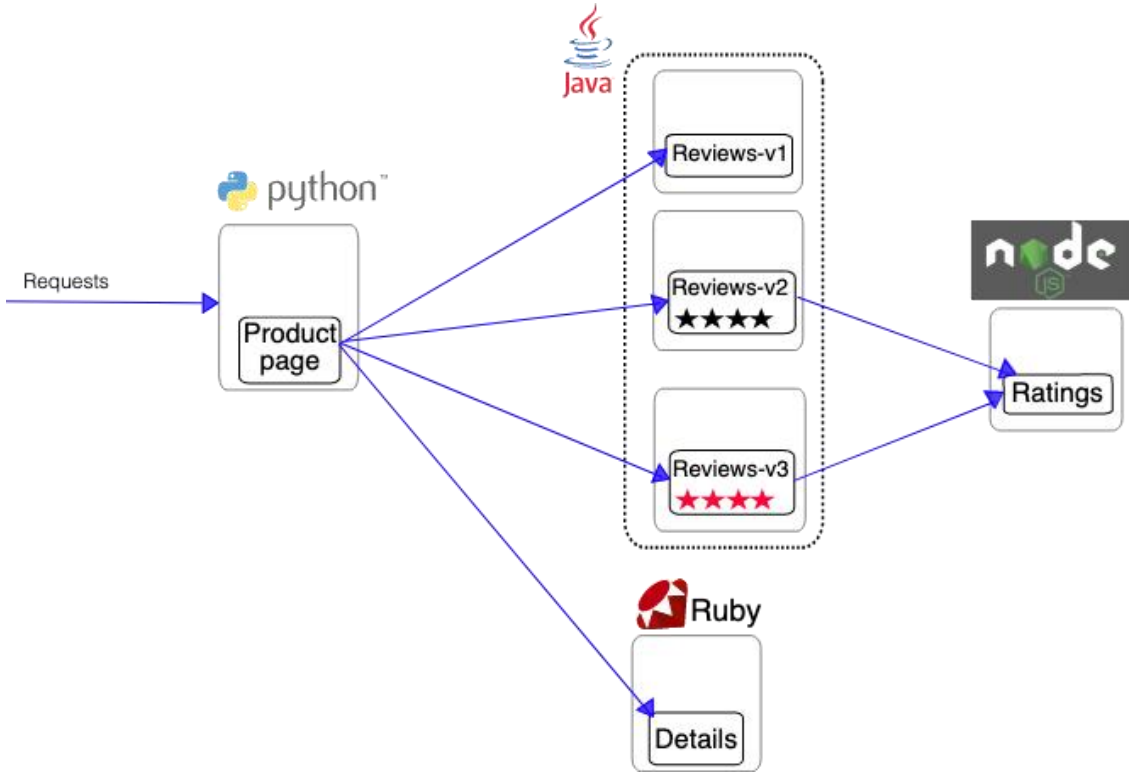
01 工程设计

项目介绍

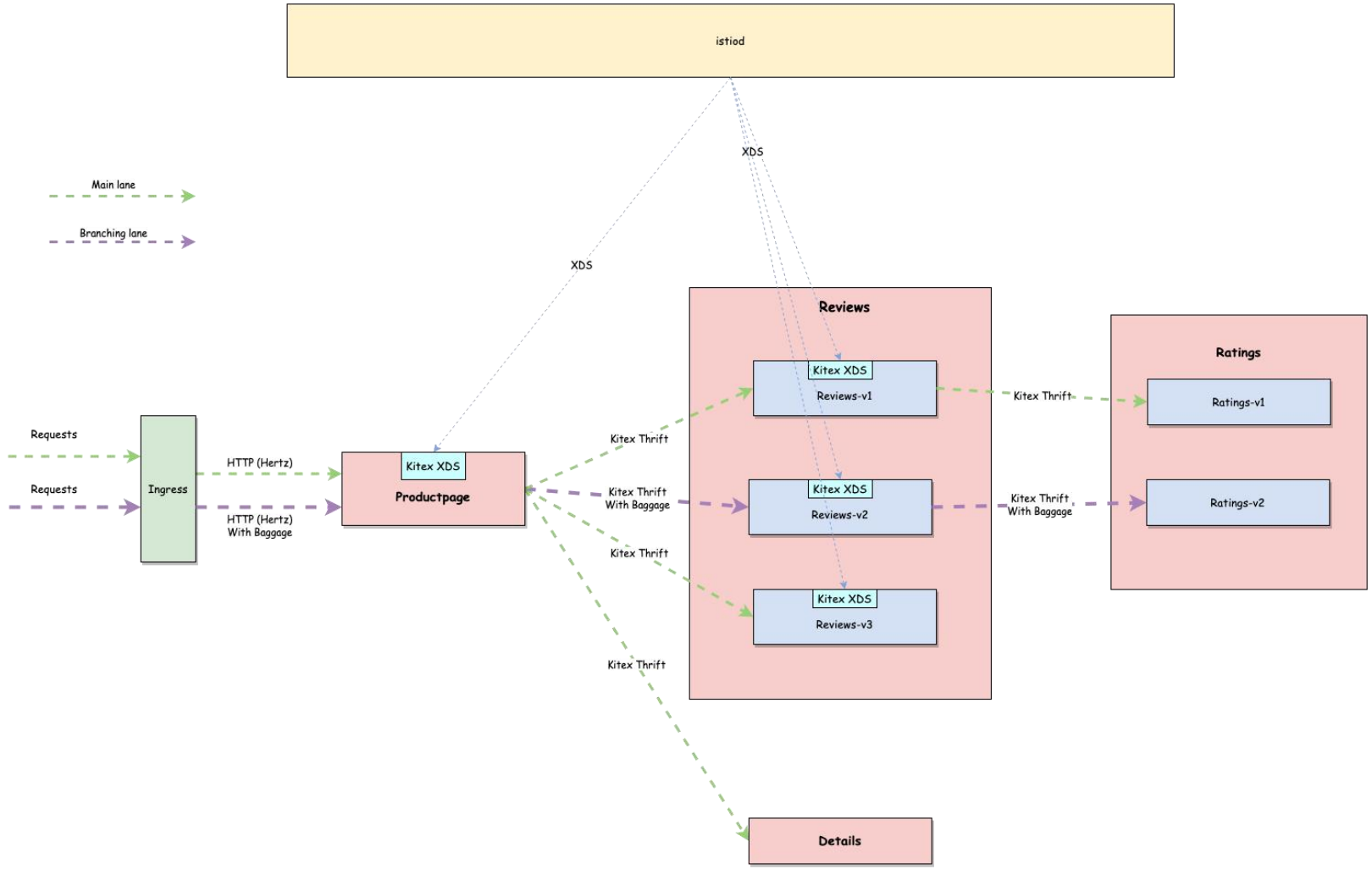
Bookinfo 是 Istio 官方提供的一个经典 Demo，用于演示多种 Istio 特性。

本项目则是基于 CloudWeGo技术栈来重写该 Demo，旨在演示 CloudWeGo 如何与 Istio、OpenTelemetry 等开源生态结合。

项目地址：https://github.com/cloudwego/biz-demo/tree/main/bookinfo



架构设计


istiod
XDS
XDS
Main lane
Branching lane
Requests
Requests
Ingress
HTTP (Hertz)
HTTP (Hertz)
With Baggage
Kitex XDS
Productpage
Kitex Thrift
Kitex Thrift
With Baggage
Kitex Thrift
Kitex Thrift
Reviews
Kitex XDS
Reviews-v1
Kitex XDS
Reviews-v2
Kitex XDS
Reviews-v3
Kitex Thrift
Kitex Thrift
With Baggage
Ratings
Ratings-v1
Ratings-v2
Details

工程结构设计

```
1  ├── Makefile
2  ├── README.md
3  ├── README_CN.md
4  ├── build
5  │   └── Dockerfile
6  ├── cmd
7  │   ├── details
8  │   ├── main.go
9  │   ├── productpage
10 │   ├── ratings
11 │   └── reviews
12 ├── conf
13 ├── idl
14 ├── internal
15 │   ├── handler
16 │   ├── server
17 │   └── service
18 ├── kitex_gen
19 ├── manifest
20 │   └── bookinfo
21 └── pkg
```

Makefile 规范设计

```
# The old school Makefile, following are required targets. The Makefile is written
# to allow building multiple binaries. You are free to add more targets or change
# existing implementations, as long as the semantics are preserved.
#
#   make              - default to 'build' target
#   make lint         - code analysis
#   make test         - run unit test (or plus integration test)
#   make build        - alias to build-local target
#   make build-local  - build local binary targets
#   make build-linux  - build linux binary targets
#   make container    - build containers
#   $ docker login registry -u username -p xxxxx
#   make push         - push containers
#   make clean        - clean up targets   CoderPoet, 2022/9/19, 3:05 上午 • feat(bookinfo): add ma
#
# Not included but recommended targets:
#   make e2e-test
#
# The makefile is also responsible to populate project version information.
#
```

需要包含的

- 代码检查
- 单元测试
- 二进制构建
- 跨平台编译构建
- 镜像构建
- 镜像推送
- targets 清理

建议包含的

- e2e 测试

Makefile 规范设计

```
# more info about `GOGC` env: https://github.com/golangci/golangci-lint#memory-usage-of-golangci-lint
lint: $(GOLANGCI_LINT) $(HELM_LINT)
	@$(GOLANGCI_LINT) run
	@bash hack/helm-lint.sh

$(GOLANGCI_LINT):
	curl -sfL https://install.goreleaser.com/github.com/golangci/golangci-lint.sh | sh -s -- -b $(BIN_DIR) v1.23.6

$(HELM_LINT):
	curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 | sudo bash

test:
	@go test -race -coverprofile=coverage.out ./...
	@go tool cover -func coverage.out | tail -n 1 | awk '{ print "Total coverage: " $$3 }'

build-local:
	@go build -v -o $(OUTPUT_DIR)/$(NAME)                                \
	  -ldflags "-s -w -X $(GOCOMMON)/version.module=$(NAME)              \
	    -X $(GOCOMMON)/version.version=$(VERSION)                        \
	    -X $(GOCOMMON)/version.branch=$(BRANCH)                          \
	    -X $(GOCOMMON)/version.gitCommit=$(GITCOMMIT)                    \
	    -X $(GOCOMMON)/version.gitTreeState=$(GITTREESTATE)              \
	    -X $(GOCOMMON)/version.buildDate=$(BUILDDATE)"                   \
	  $(CMD_DIR);

container:
	@docker build -t $(REGISTRY)/$(IMAGE_NAME):$(VERSION)               \
	  --label $(DOCKER_LABELS)                                           \
	  -f $(BUILD_DIR)/Dockerfile .;
```

CoderPoet, 2022/9/19, 3:05 上午 • feat(bookinfo): add makefile、dockerfile、manifest

Dockerfile 规范设计

```dockerfile
FROM golang:1.16 as builder
COPY . /go/src/github.com/cloudwego/biz-demo/bookinfo
WORKDIR /go/src/github.com/cloudwego/biz-demo/bookinfo
RUN make build-linux

FROM alpine:latest
RUN mkdir -p /app && \
    chown -R nobody:nogroup /app
COPY --from=builder /go/src/github.com/cloudwego/biz-demo/bookinfo/bin/bookinfo /app
COPY --from=builder /go/src/github.com/cloudwego/biz-demo/bookinfo/conf /app/conf

USER        nobody
WORKDIR     /app
ENV         PSM=cwg.bizdemo.bookinfo HERTZ_CONF_DIR=conf HERTZ_LOG_DIR=output/log
ENTRYPOINT  ["/app/bookinfo"]
```

多阶段构建

编译阶段与运行阶段构建分离，减少运行时镜像体积

镜像精简原则

运行时镜像只需要包含二进制可执行文件以及配置目录即可

镜像安全

通常业务服务不需要特权执行，因此使用 USER 指令切换用户，避免在运行时使用 root 用户

02 技术选型

技术栈介绍

技术栈	用途
Kitex、 Hertz	服务端框架
Istio	作为服务网格控制面，负责 xDS 配置下发
Wire	用于依赖注入
OpenTelemetry	用于可观测性，主要包括全链路追踪
Kitex-xDS	实现以 Proxyless 的方式对接服务网格
React、 Arco-design	用于实现 Bookinfo UI 层

框架选型 – Hertz、Kitex

服务	框架	说明
productpage	hertz server、 kitex client	入口服务，对外提供 HTTP 接口； 会调用 details 和 reviews 两个微服务
reviews	kitex、 kitex client	这个微服务中包含了书籍相关的评论； 会调用 ratings 微服务
ratings	kitex	这个微服务中包含了由书籍评价组成的评级信息
details	kitex	这个微服务中包含了书籍的信息

依赖注入（Google Wire）

wire 是 Google 开源的一款依赖注入框架，基于 generate 和静态代码分析来实现

```go
// Handler productpage hertz handler
type Handler struct {
    reviewsClient reviewsservice.Client
    detailsClient detailsservice.Client
}

// New create handler
func New(reviewsClient reviewsservice.Client, detailsClient detailsservice.Client) *Handler {
    return &Handler{reviewsClient: reviewsClient, detailsClient: detailsClient}
}
```

依赖注入（Google Wire）

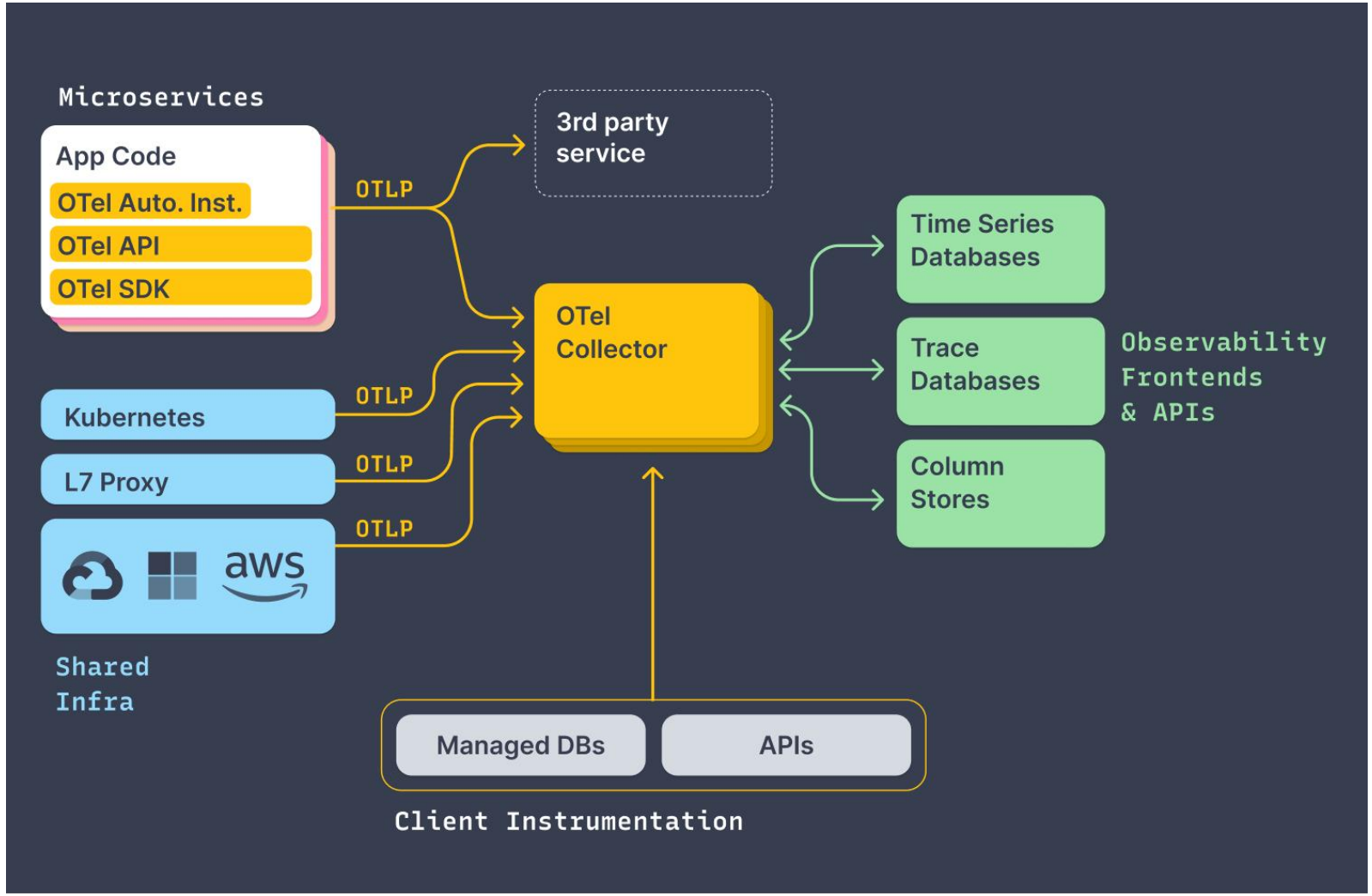
```go
// ReviewClientOptions client options
type ReviewClientOptions struct {
    Endpoint  string `mapstructure:"endpoint"`
    EnableXDS bool   `mapstructure:"enableXDS"`
    XDSAddr   string `mapstructure:"xdsAddr"`
}
  CoderPoet, 2022/9/8, 12:06 下午 • feat(bookinfo): add kitex + hertz bookinfo demo
// DefaultReviewClientOptions default options
func DefaultReviewClientOptions() *ReviewClientOptions {
    return &ReviewClientOptions{
        Endpoint:  "reviews:8082",
        EnableXDS: false,
        XDSAddr:   "istiod.istio-system.svc:15010",
    }
}

// ProvideReviewClient Provide review client
// 1、init xds manager: only init once
// 2、enable xds
// 3、enable opentelemetry
func ProvideReviewClient(opts *ReviewClientOptions) (reviewsservice.Client, error) {
    if opts.EnableXDS {
        if err := xdsmanager.Init(xdsmanager.WithXDSServerAddress(opts.XDSAddr)); err != nil {
            klog.Fatal(err)
        }
        return reviewsservice.NewClient(
            opts.Endpoint, // use svc fqdn
            kclient.WithSuite(tracing.NewClientSuite()),
            kclient.WithXDSSuite(xds.ClientSuite{
                RouterMiddleware: xdssuite.NewXDSRouterMiddleware(
                    xdssuite.WithRouterMetaExtractor(metadata.ExtractFromPropagator),
                ),
```

依赖注入（Google Wire）

```go
// NewServer build server with wire
func NewServer(ctx context.Context) (*Server, error) {
    panic(wire.Build(
        configparser.Default,
        Configure,

        injectors.ProvideReviewClient,
        injectors.ProvideDetailsClient,

        productpage.New,

        wire.FieldsOf(new(*Options),
            fieldNames...: "Server",
            "Reviews",
            "Details",
        ),
        wire.Struct(new(Server), fieldNames...: "*"),
    ))
}
```

```go
//go:build !wireinject
// +build !wireinject

package productpage

import ...

// Injectors from wire.go:

func NewServer(ctx context.Context) (*Server, error) {
    provider := configparser.Default()
    options, err := Configure(provider)
    if err != nil : nil, err ⤴
    serverOptions := options.Server
    reviewClientOptions := options.Reviews
    client, err := injectors.ProvideReviewClient(reviewClientOptions)
    if err != nil : nil, err ⤴
    detailsClientOptions := options.Details
    detailsserviceClient, err := injectors.ProvideDetailsClient(detailsClientOptions)
    if err != nil : nil, err ⤴
    handler := productpage.New(client, detailsserviceClient)   CoderPoet, 2022/9/8, 12
    server := &Server{
        opts:    serverOptions,
        handler: handler,
    }
    return server, nil
}
```

可观测 – OpenTelemetry



OpenTelemetry 提供了一个开源标准和一套可观测框架，用于捕获和导出云原生应用和基础架构中的 Metrics、Traces 和 Logs。

可观测 – OpenTelemetry

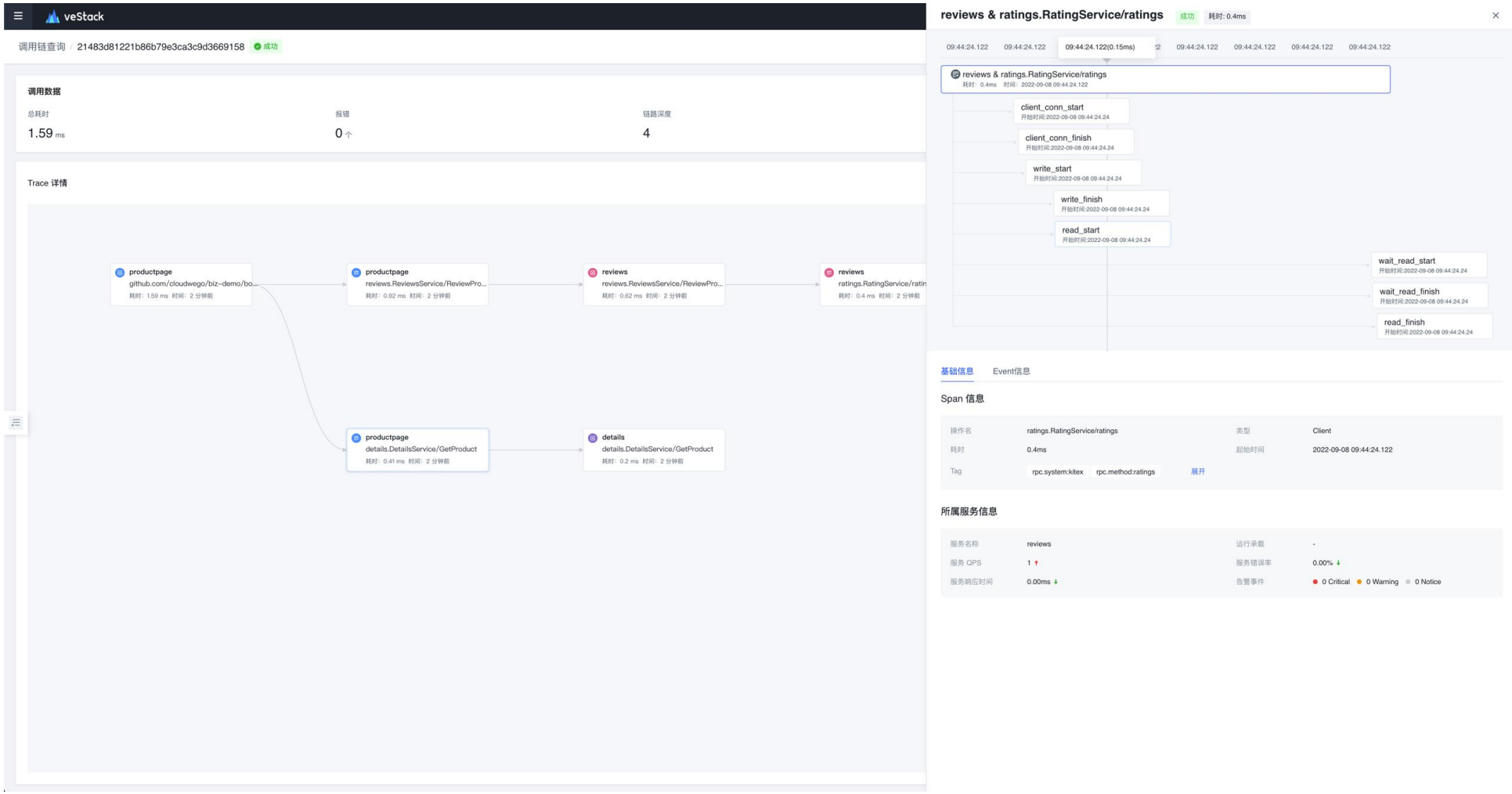
```go
// Run reviews server
func (s *Server) Run(ctx context.Context) error {
    klog.SetLogger(kitexlogrus.NewLogger())
    klog.SetLevel(s.opts.LogLevel.KitexLogLevel())

    p := provider.NewOpenTelemetryProvider(
        provider.WithServiceName(constants.ReviewsServiceName),
        provider.WithInsecure(),
    )
    defer func(p provider.OtelProvider, ctx context.Context) {
        _ = p.Shutdown(ctx)
    }(p, ctx)

    addr, err := net.ResolveTCPAddr( network: "tcp", s.opts.Addr)
    if err != nil {
        klog.Fatal(err)
    }
    svr := reviewsservice.NewServer(
        s.svc,
        server.WithServiceAddr(addr),
        server.WithSuite(tracing.NewServerSuite()),  CoderPoet, 2022/9/8, 12:06 下
    )
    if err := svr.Run(); err != nil {
        klog.Fatal(err)
    }

    return nil
}
```

Kitex、Hertz 已经原生集成了 OpenTelemetry：[hertz-otel、kitex otel]

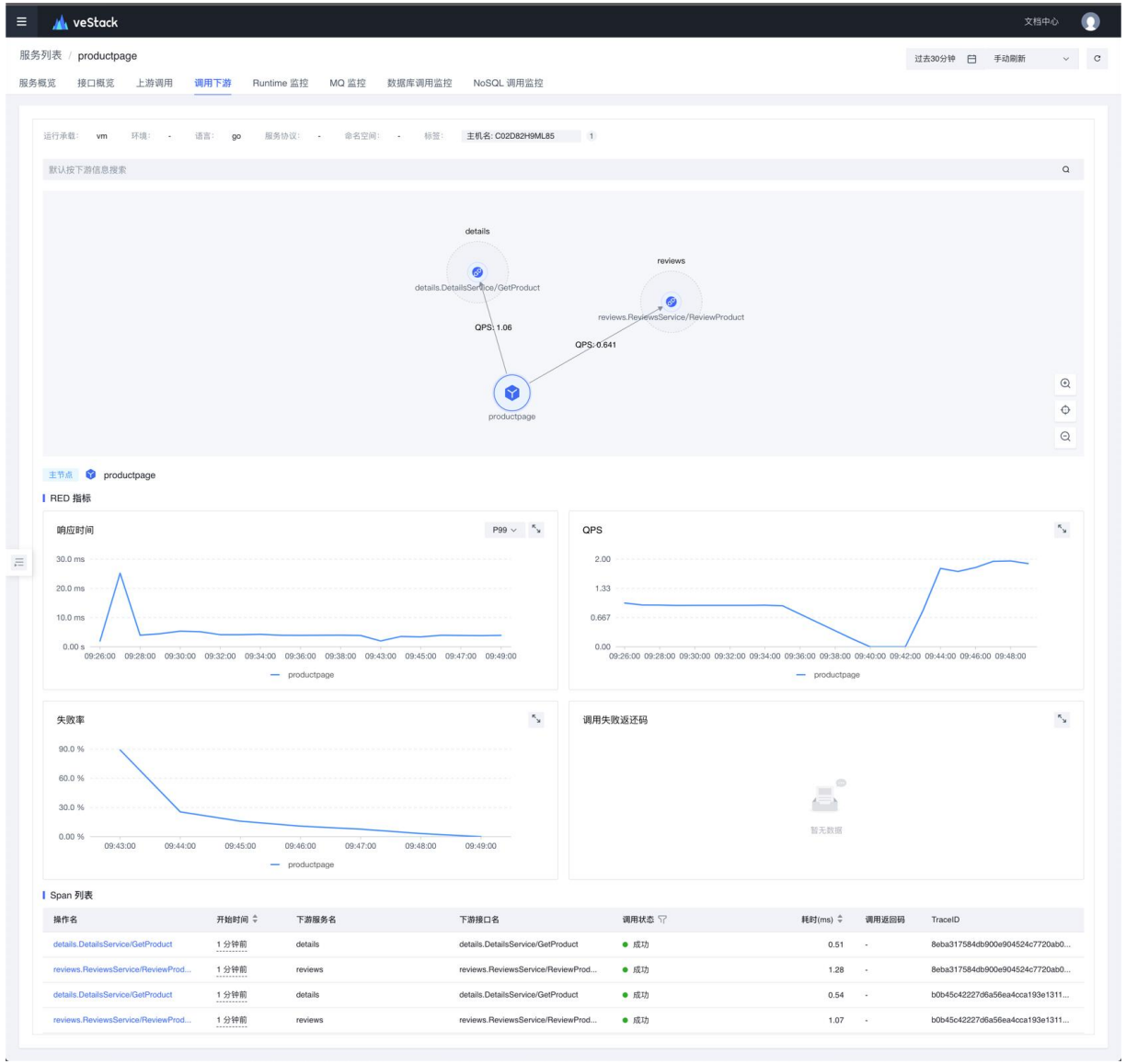
基于这些库，我们可以轻松在业务中集成 OpenTelemetry

例如：在 Kitex 中开启 OpenTelemetry，我们只需要引入对应依赖，初始化默认的 otel provider，并注入对应 suite 即可

可观测 – OpenTelemetry

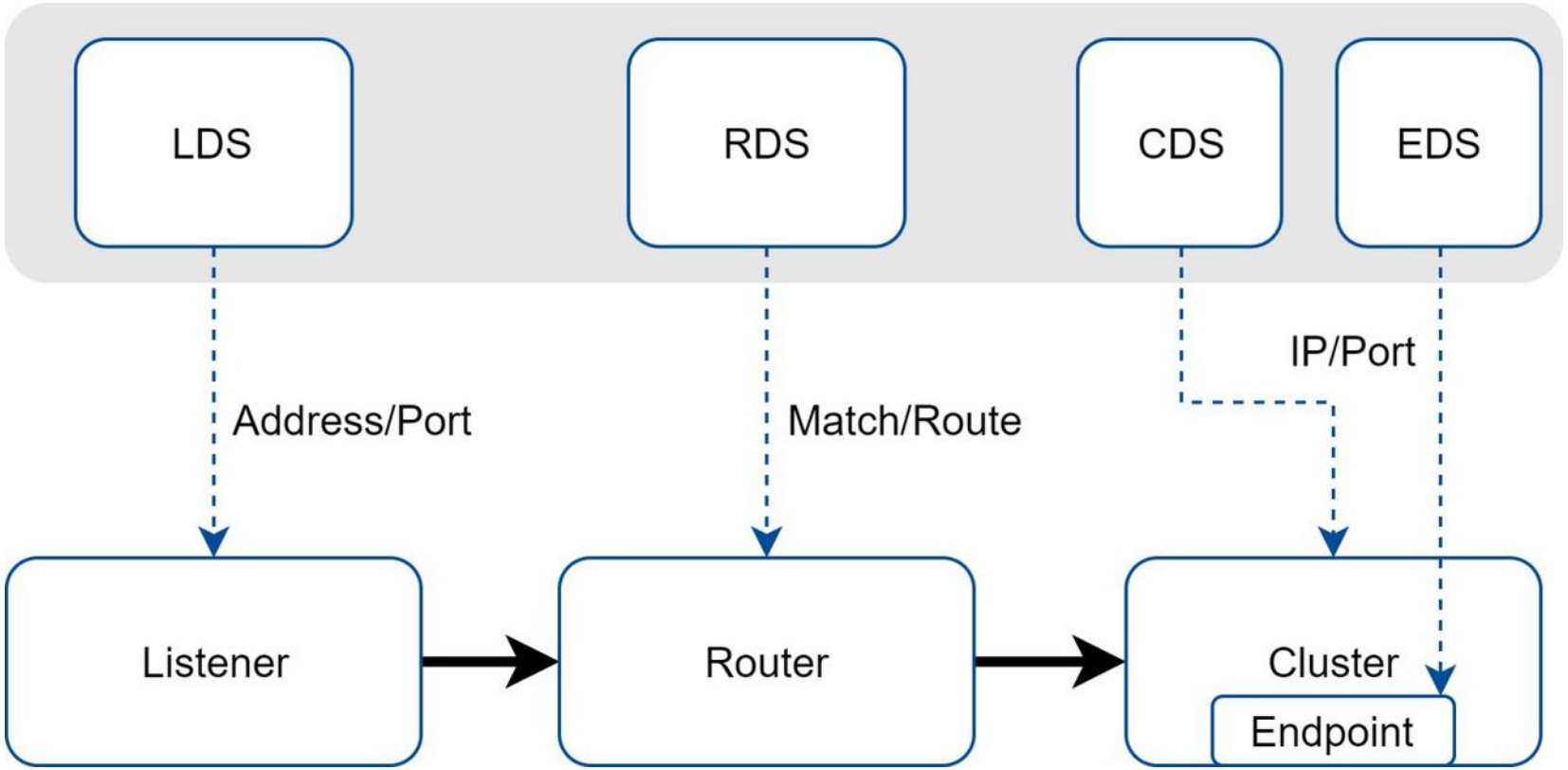


可观测 – OpenTelemetry



服务治理 - xDS 介绍

xDS 是一组发现服务的总称，全称为 “X Discovery Service”，其中的 “X” 代指多种发现服务，包含 LDS (Listener), RDS (RouteConfiguration), CDS (Cluster), 和 EDS (Endpoint/ClusterLoadAssignment) 等。 数据面可以利用 xDS API 与控制平面（如 Istio）通信，完成配置信息的动态发现。



服务治理 - Kitex xDS

Kitex 通过外部扩展 kitex-contrib/xds 的形式对 xDS API 进行了支持，可通过代码配置开启 xDS 模块，让 Kitex 服务以 Proxyless 的模式运行，被服务网格统一纳管。具体的设计方案参见 proposal。

```
if opts.EnableXDS {
    if err := xdsmanager.Init(xdsmanager.WithXDSServerAddress(opts.XDSAddr)); err != nil {
        klog.Fatal(err)   CoderPoet, 2022/9/19, 3:05 上午 • feat(bookinfo): add makefile、dockerfile、man
    }
    return reviewsservice.NewClient(
        opts.Endpoint, // use svc fqdn
        kclient.WithSuite(tracing.NewClientSuite()),
        kclient.WithXDSSuite(xds.ClientSuite{
            RouterMiddleware: xdssuite.NewXDSRouterMiddleware(
                xdssuite.WithRouterMetaExtractor(metadata.ExtractFromPropagator),
            ),
            Resolver: xdssuite.NewXDSResolver(),
        }),
    )
}
```

开启 opentelemetry

开启 kitex xds（proxyless）

服务治理 – 为服务实例按照版本分组

```yaml
apiVersion: apps/v1   CoderPoet, 2022/9/19, 3:05 上午 • feat(bookinfo): add
kind: Deployment
metadata:
  labels:
    app.kubernetes.io/instance: reviews
    app.kubernetes.io/name: reviews
  name: reviews-v1
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/instance: reviews
      app.kubernetes.io/name: reviews
      version: v1
  template:
    metadata:
      annotations:
        sidecar.istio.io/inject: "false"
      labels:
        app.kubernetes.io/instance: reviews
        app.kubernetes.io/name: reviews
        version: v1
    spec:
      containers:
        - args:
            - reviews
            - --config=config/reviews.yaml
          env:
```

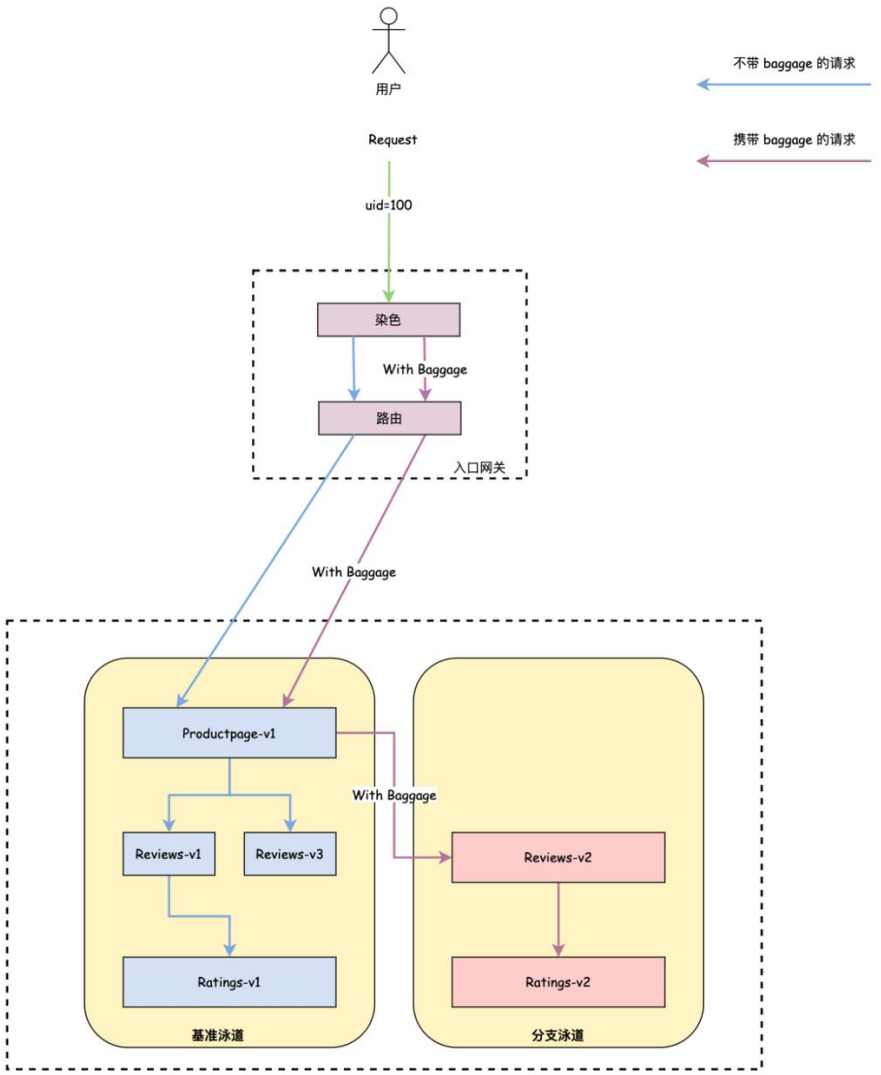
```yaml
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: reviews
spec:
  host: reviews
  subsets:
  - name: v1
    labels:
      version: v1
  - name: v2
    labels:
      version: v2
  - name: v3
    labels:
      version: v3
```

服务治理 – 定义流量路由规则

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: ratings
spec:
  hosts:
    - ratings
  http:
    - match:
        - headers:
            baggage:
              exact: "env=dev"
      route:
        - destination:   CoderPoet, 2022/9/22, 9:57 上午 • fea
            host: ratings
            subset: v2
          weight: 100
    - route:
        - destination:
            host: ratings
          weight: 100
```

03 全链路泳道

泳道设计



流量染色

在网关层进行流量染色，根据原始请求中的元数据，来进行一定规则（条件、比例）转换成对应的染色标识

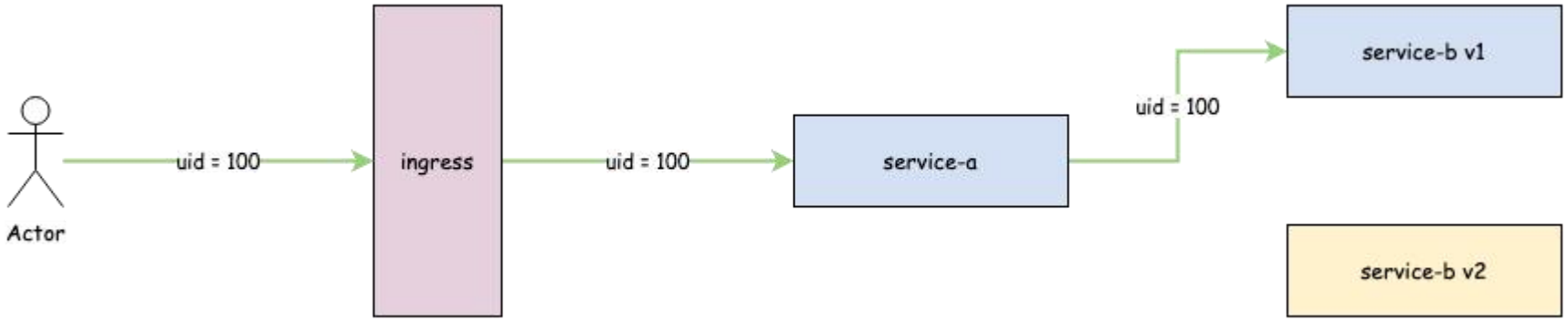
基准泳道

未被染色的流量会被路由到基准泳道中

分支泳道

被染色的流量会被路由到 reviews-v2 ->ratings-v2 的分支泳道中

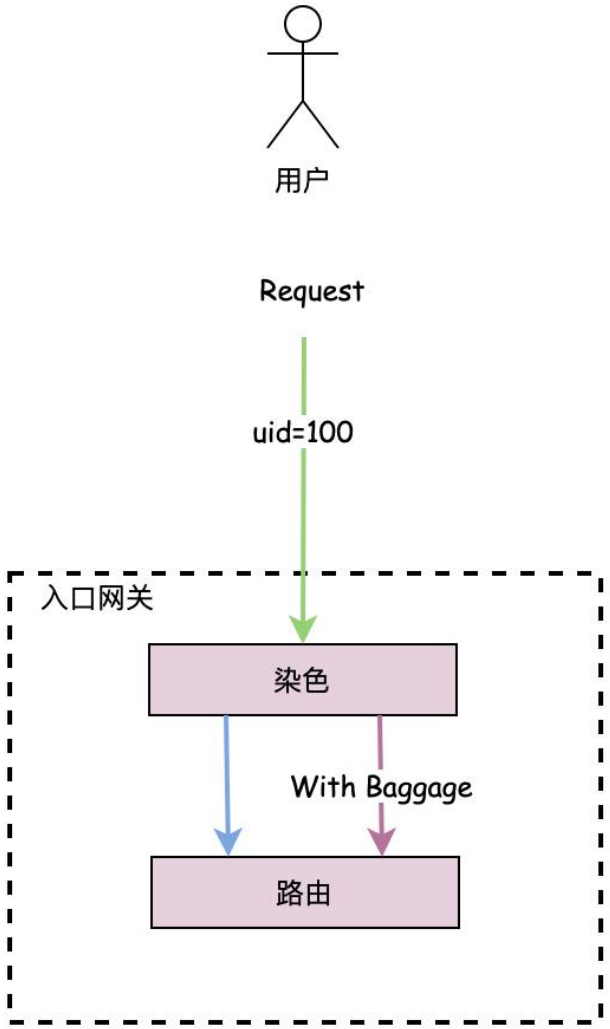
泳道设计 - 传统流量路由方式



1. 不够通用：以具体某个业务属性标识（如：uid）作为流量路由匹配规则，我们需要将这个业务属性手动在全链路里透传，这本身对业务侵入性较大，需要业务配合改造。并且当我们要使用其他业务属性的时候，又需要全链路业务都改造一遍，可想而知，是非常不通用的做法

2. 路由规则容易频繁变动，容易造成规则臃肿：以具体某个业务属性标识（如：uid）作为流量路由匹配规则，假设我们要换一个业务属性，或者给其他用户设置路由规则的时候，得去改造原有的路由规则，或者针对不同业务属性重复定义多套路由规则，很容易就会造成路由臃肿，以至于难以维护

泳道设计 – 流量染色



我们可以在网关层进行流量染色，通常会根据原始请求中的元数据，来进行一定规则（条件、比例）转换成对应的染色标识。

- 按条件染色：当请求元数据满足一定条件之后，就给当前请求打上染色标识，如：请求头中 uid=100、cookie 匹配等等。
- 按比例染色：按照一定比例，给请求打上染色标识。

有了一套统一的流量染色机制之后，我们配置路由规则的时候，就不需要关心具体的业务属性标识了，只需要根据 染色标识 来配置即可。

泳道设计 – 染色标识全链路透传（Baggage）

染色标识通常会依靠 Tracing Baggage 来透传，Baggage 是用于在整个链路中传递业务自定义 KV 属性，例如传递流量染色标识、传递 AccountID 等业务标识等等。

```
if opts.EnableXDS {
    if err := xdsmanager.Init(xdsmanager.WithXDSServerAddress(opts.XDSAddr)); err != nil {
        klog.Fatal(err)   CoderPoet, 2022/9/19, 3:05 上午 • feat(bookinfo): add makefile、dockerfi
    }
    return reviewsservice.NewClient(
        opts.Endpoint, // use svc fqdn
        kclient.WithSuite(tracing.NewClientSuite()),
        kclient.WithXDSSuite(xds.ClientSuite{
            RouterMiddleware: xdssuite.NewXDSRouterMiddleware(
                xdssuite.WithRouterMetaExtractor(metadata.ExtractFromPropagator),
            ),
            Resolver: xdssuite.NewXDSResolver(),
        }),
    )
}
```

开启 opentelemetry 即可自动透传 baggage

泳道设计 – 泳道效果（基准泳道）

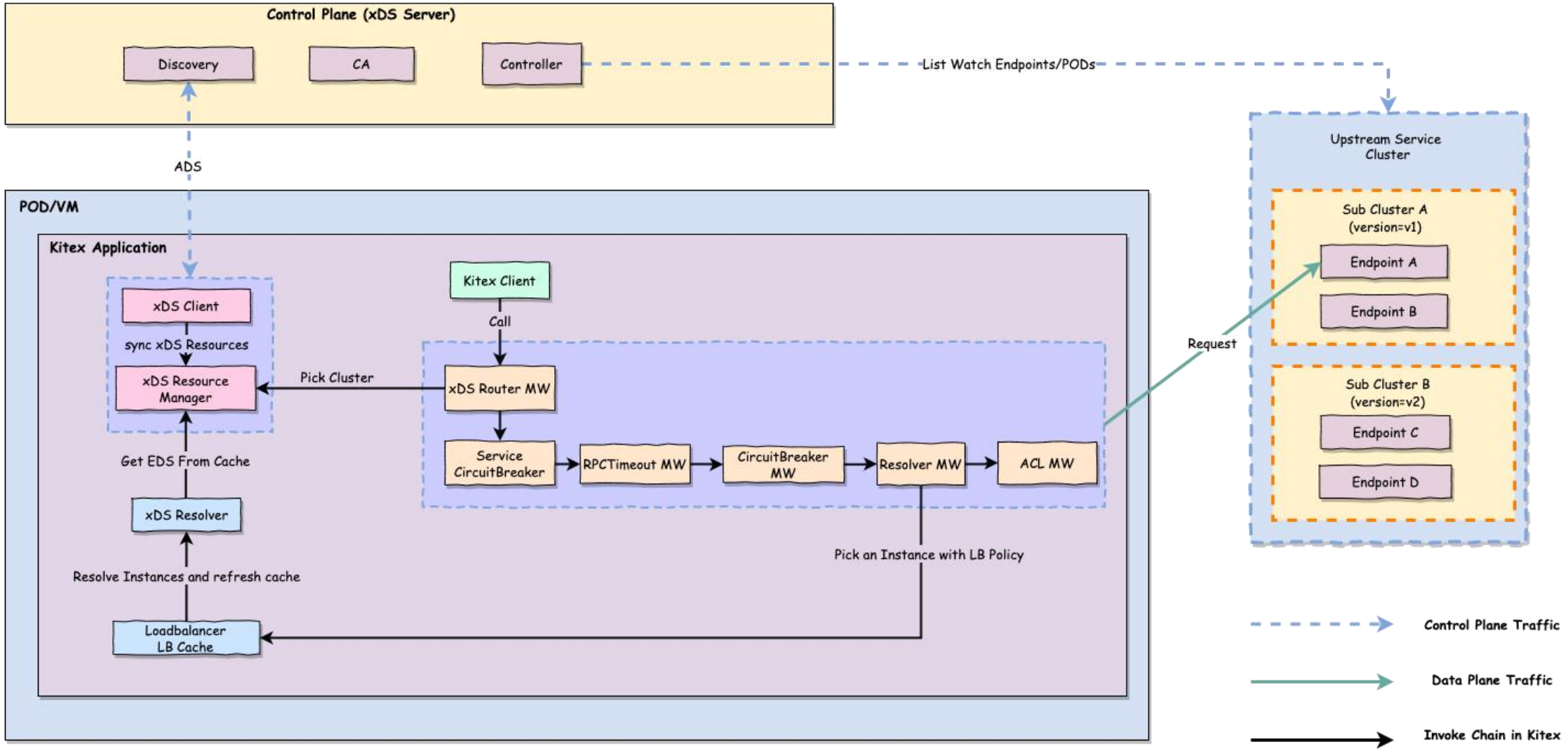
入口流量请求头中不带 uid=100 的请求，会自动路由到基准泳道服务，reviews v1 和 v3 服务间轮询，展示的效果是评分为 0 或 1 随机。

泳道设计 – 泳道效果（分支泳道）

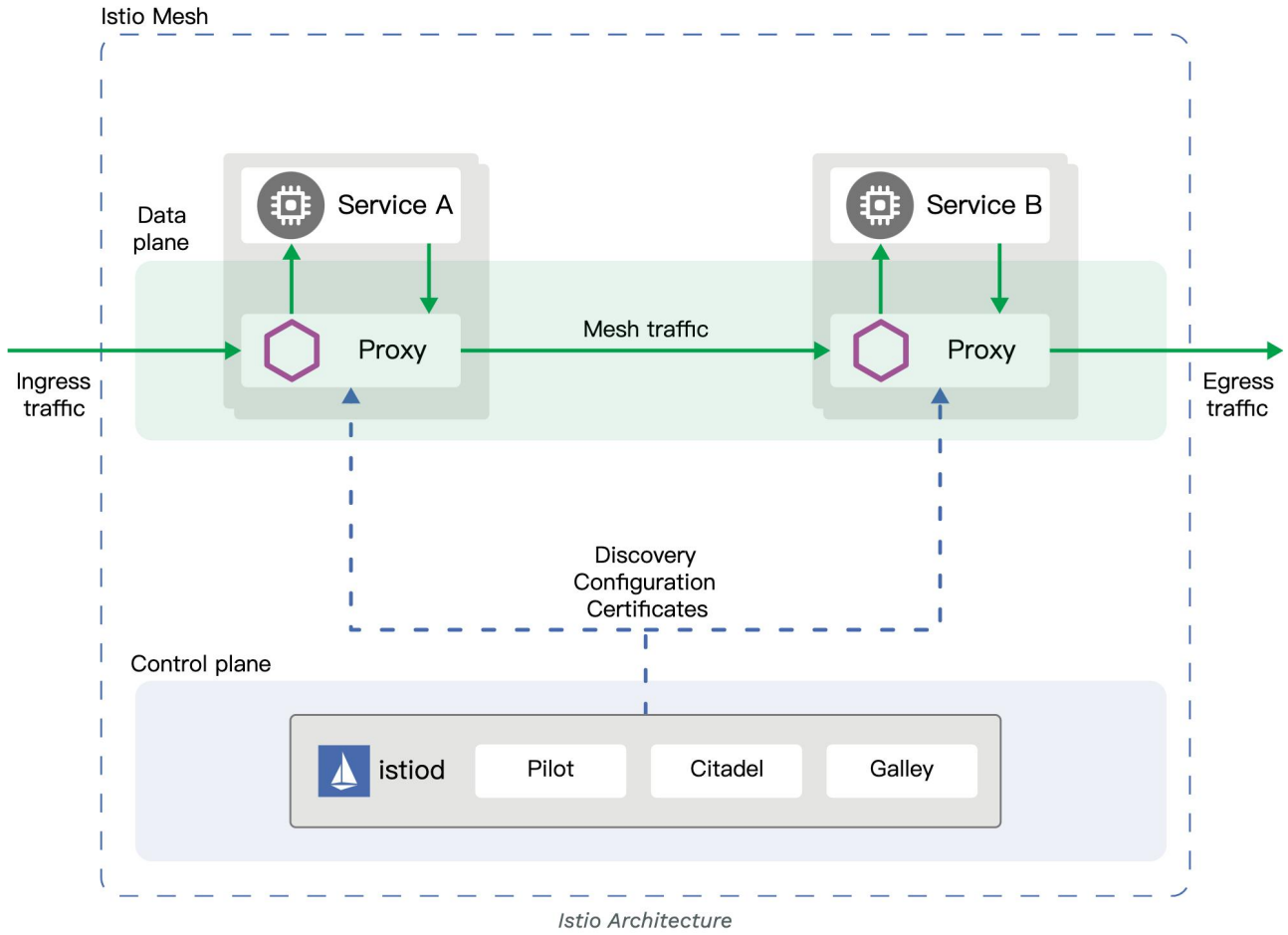
入口流量请求头中携带了 uid=100 的请求，会被精准路由到分支泳道

04 Proxyless 与 ServiceMesh

Kitex Proxyless 模式



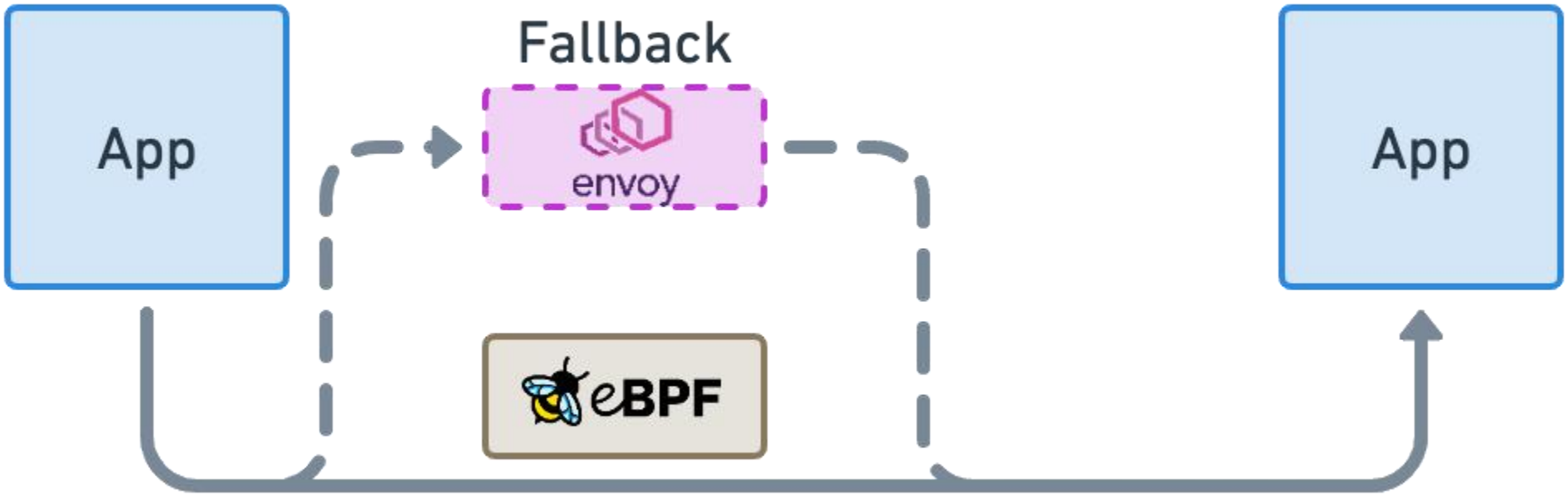
标准 Sidecar 模式



Istio Architecture

（图片引用于：https://istio.io）

eBPF 内核 Mesh 模式（cilium）



（图片引用：https://isovalent.com/blog/post/cilium-service-mesh/）

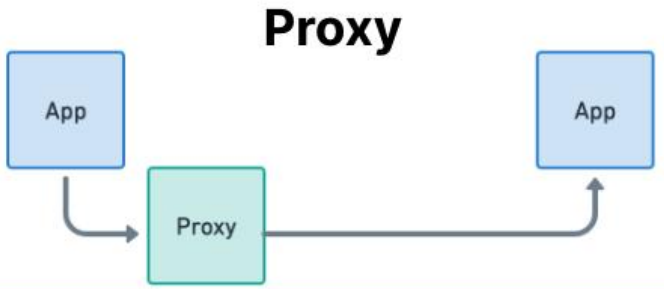
eBPF 内核 Mesh 模式（cilium）

Features Available in eBPF

Traffic Management
- L3/L4 forwarding & Load-balancing
- Canary, Topology Aware Routing
- Multi-cluster

Security
- Network Policy
- mTLS

Observability
- Tracing, OpenTelemetry, & Metrics
- HTTP, TLS, DNS, TCP, UDP, ...



Features that require Proxy fallback

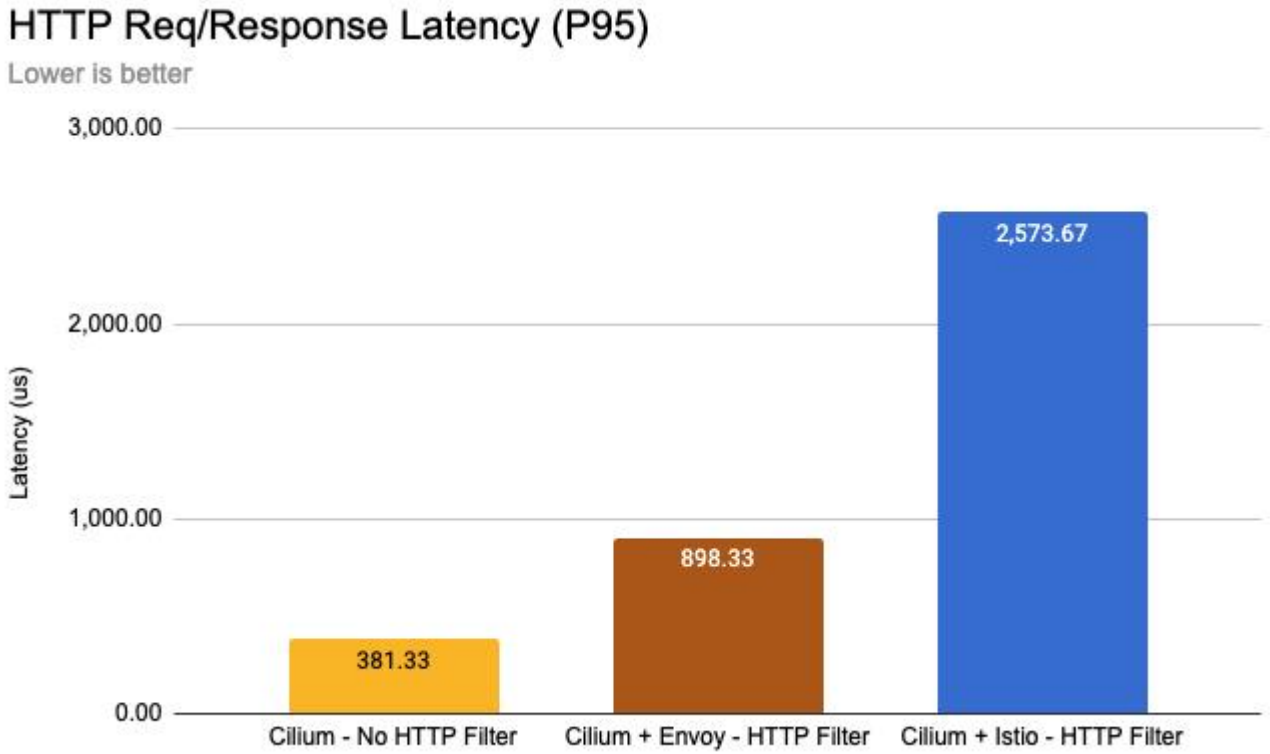
Traffic Management
- L7 Load-balancing & Ingress

Resilience
- Retries, L7 Rate Limiting

Security
- TLS Termination & Origination

（图片引用：https://isovalent.com/blog/post/cilium-service-mesh/）

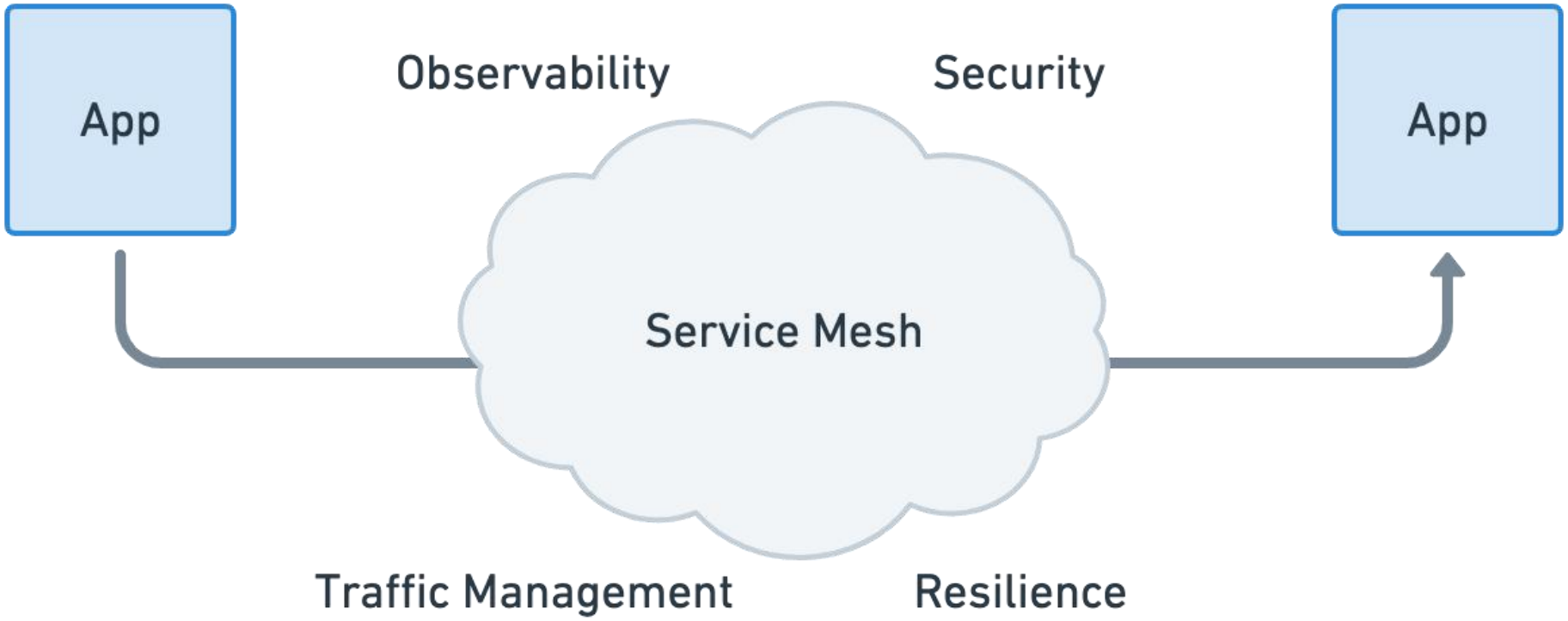
eBPF 内核 Mesh 模式（cilium）



（图片引用：https://isovalent.com/blog/post/cilium-service-mesh/）

所以，Service mesh 究竟是什么？

ServiceMesh —— 无关模式，服务间通信基础设施标准抽象



（图片引用：https://isovalent.com/blog/post/cilium-service-mesh/）

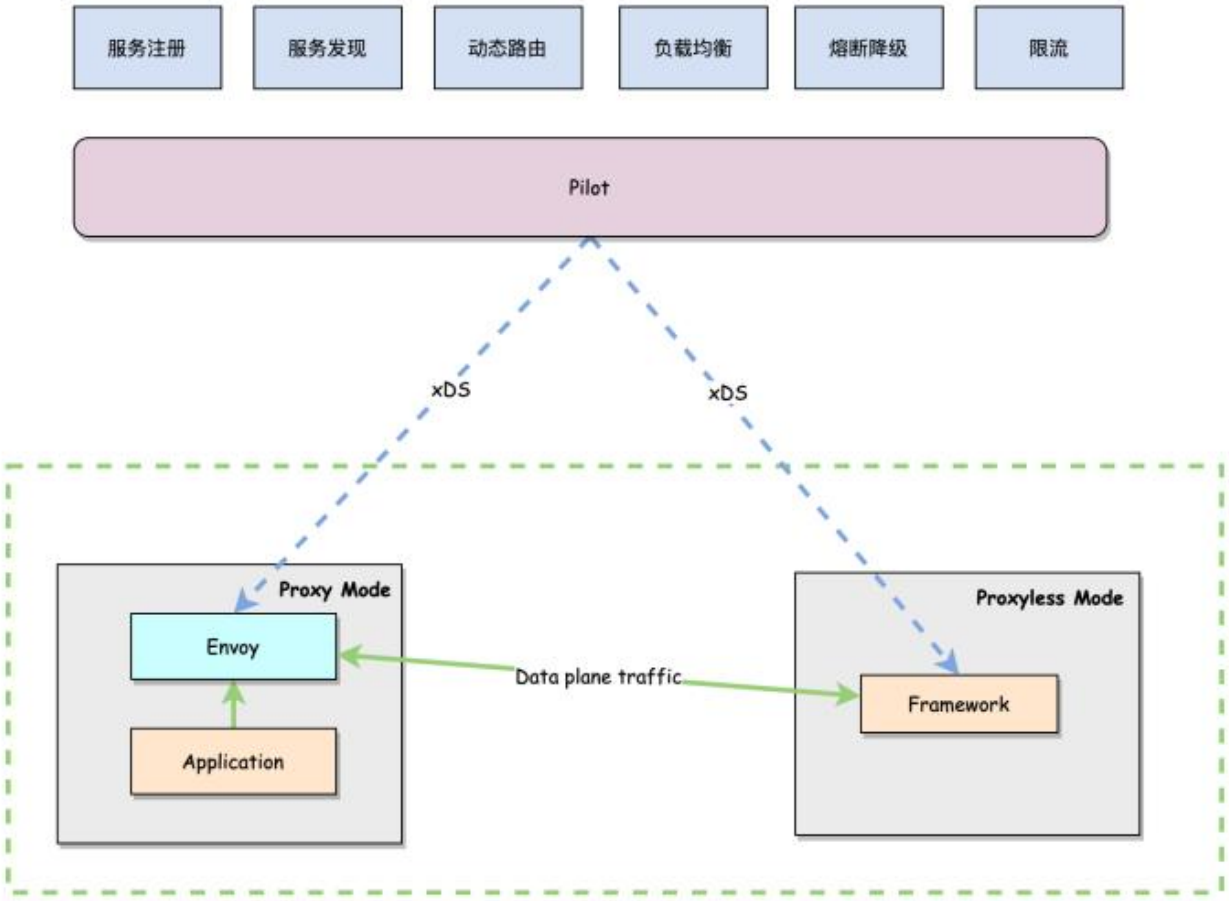
ServiceMesh —— 基于统一标准去构建服务治理平台

统一服务治理规则模型 Spec

统一服务治理控制面

统一服务治理规则下发协议

统一异构数据面治理能力



相关项目链接

项目	链接
bookinfo	https://github.com/cloudwego/biz-demo/tree/main/bookinfo
Kitex	https://github.com/cloudwego/kitex
Hertz	https://github.com/cloudwego/hertz
kitex-contrib/xds	https://github.com/kitex-contrib/xds
kitex-contrib/obs-opentelemetry	https://github.com/kitex-contrib/obs-opentelemetry
hertz-contrib/obs-opentelemetry	https://github.com/hertz-contrib/obs-opentelemetry

THANKS

CloudWeGo