CLOUD NATIVE BEST PRACTICES SUMMIT 2020

Istio 流量管理原理与协议扩展

腾讯云

赵化冰

腾讯云 服务网格团队

https://zhaohuabing.com

Service Mesh

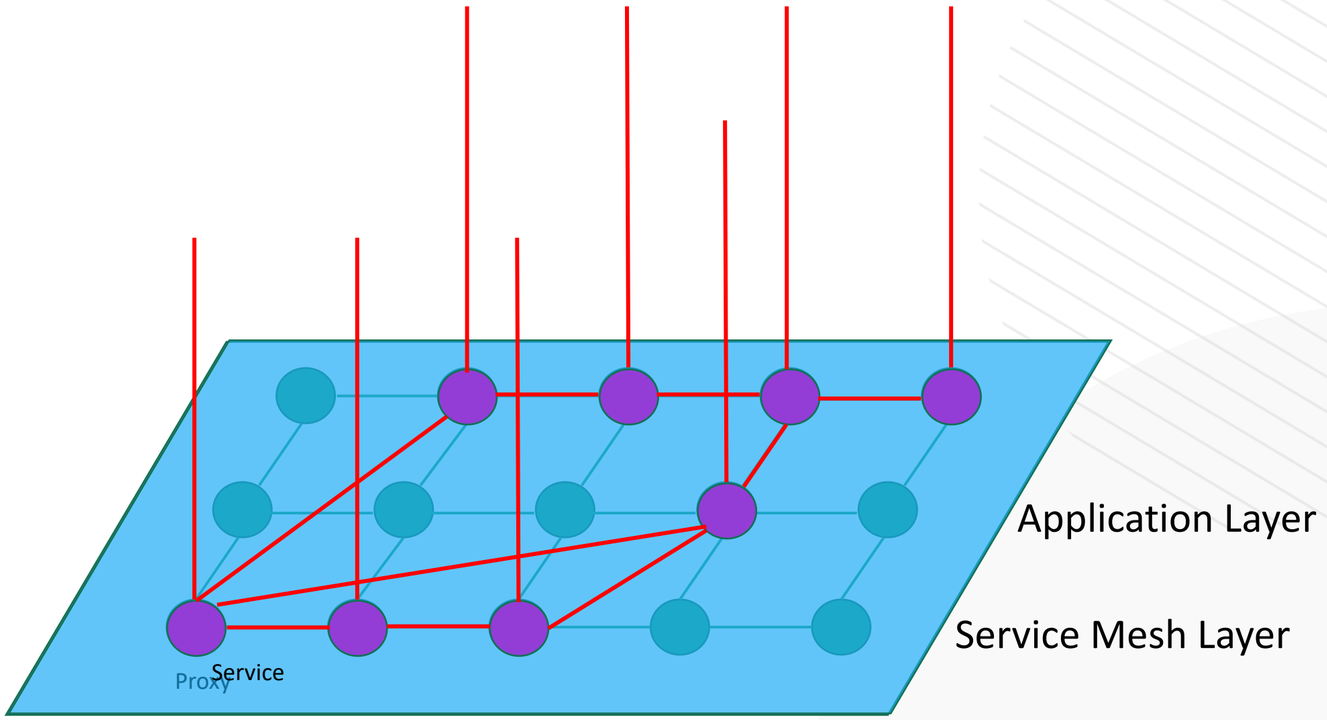
处理服务间通信（主要是七层通信）的云原生基础设施层：

Service Mesh 将各个服务中原来使用 SDK 实现的七层通信相关功能抽象出来，使用一个专用层次来实现，Service Mesh 对应用透明，因此应用可以无需关注分布式架构带来的通信相关问题，而专注于其业务价值。

流量控制：服务发现、请求路由、负载均衡、灰度发布、错误重试、断路器、故障注入

可观察性：遥测数据、调用跟踪、服务拓扑

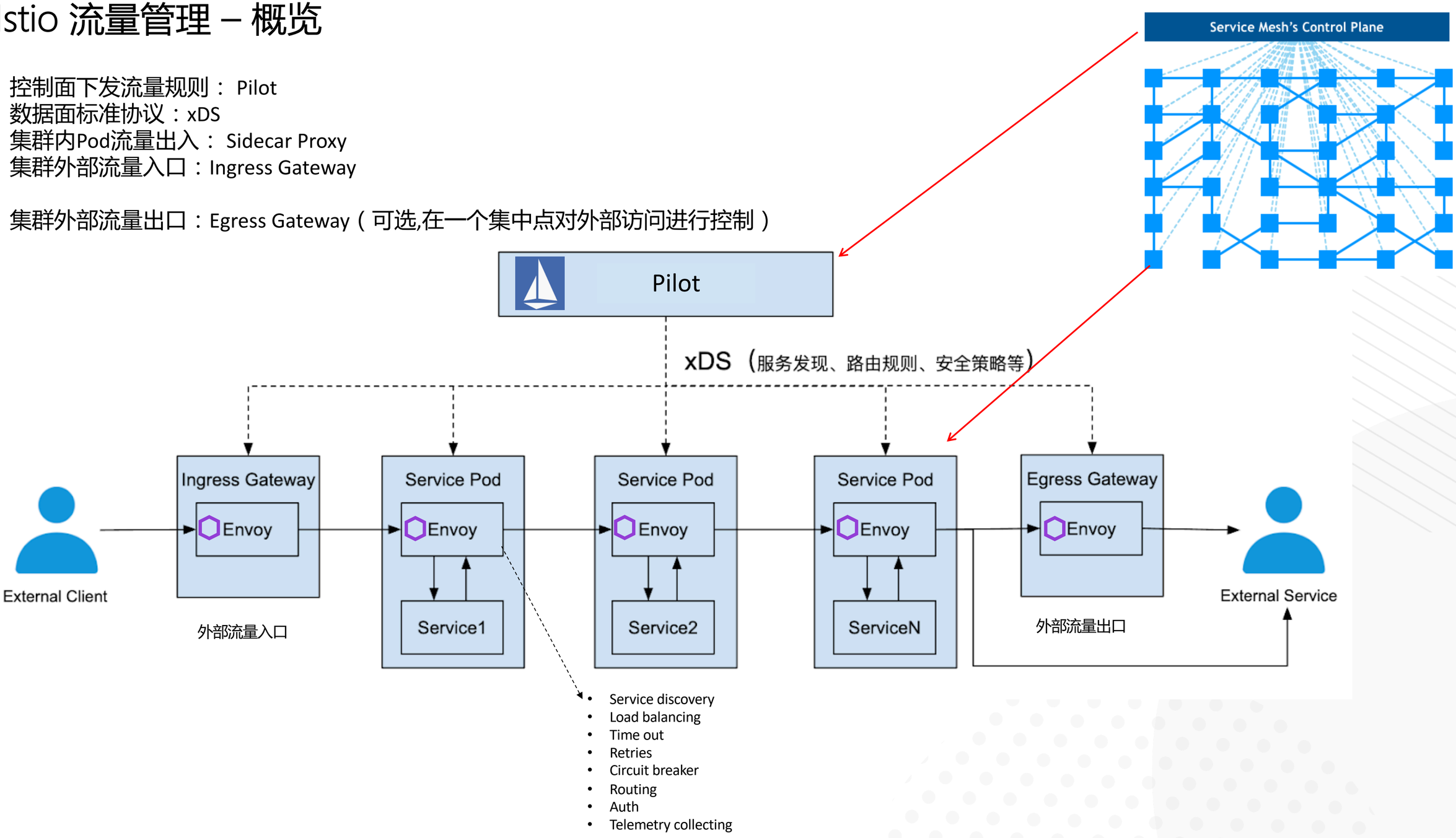
通信安全： 服务身份认证、访问鉴权、通信加密



Istio 流量管理 – 概览

- 控制面下发流量规则： Pilot
- 数据面标准协议：xDS
- 集群内Pod流量出入： Sidecar Proxy
- 集群外部流量入口：Ingress Gateway

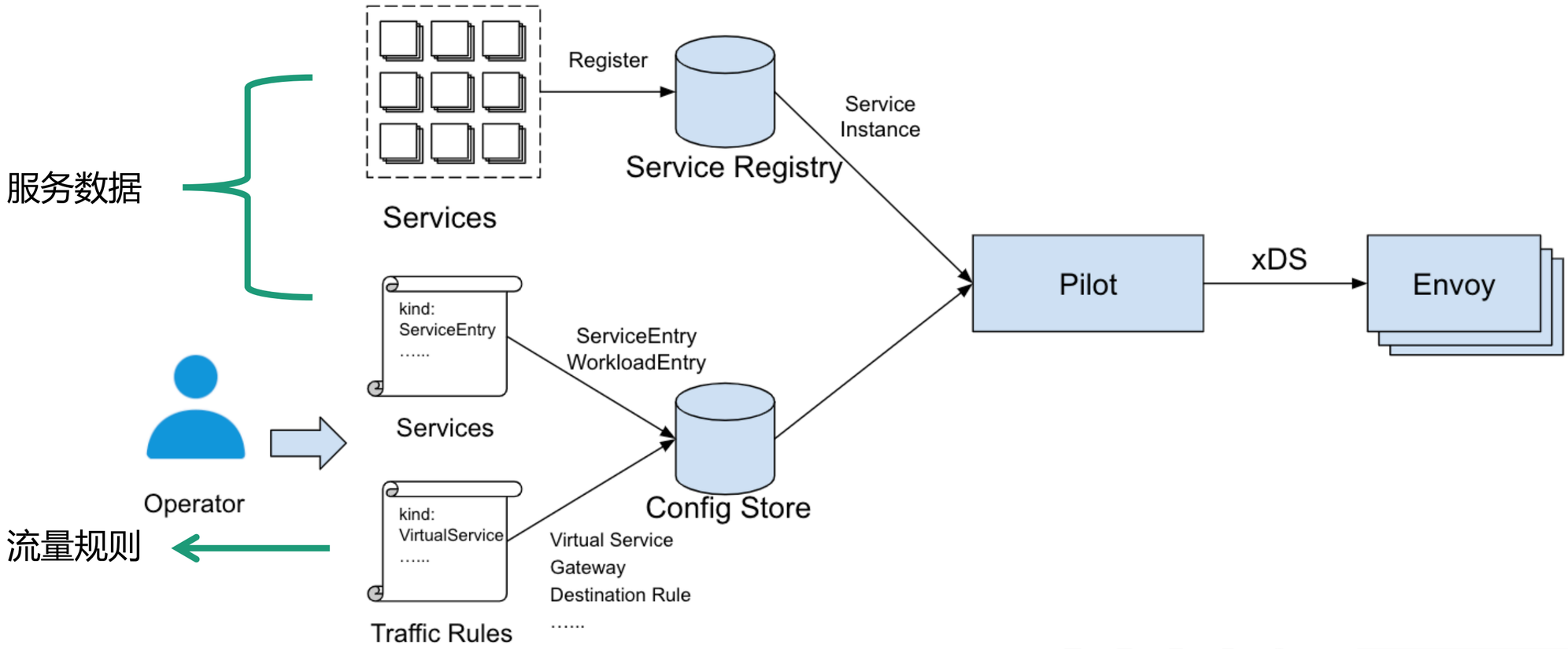
- 集群外部流量出口：Egress Gateway（可选,在一个集中点对外部访问进行控制）



Istio 流量管理 – 控制面

两类数据：

- 服务数据（Mesh 中有哪些服务？缺省路由）
  - Service Registry
    - Kubernetes：原生支持
    - Consul、Eureka 等其他服务注册表：MCP over xDS （https://github.com/istio-ecosystem/consul-mcp）
  - 通过CRD定义的服务数据
- 自定义流量规则（如何将请求路由到这些服务？）
  - 通过CRD定义的流量规则

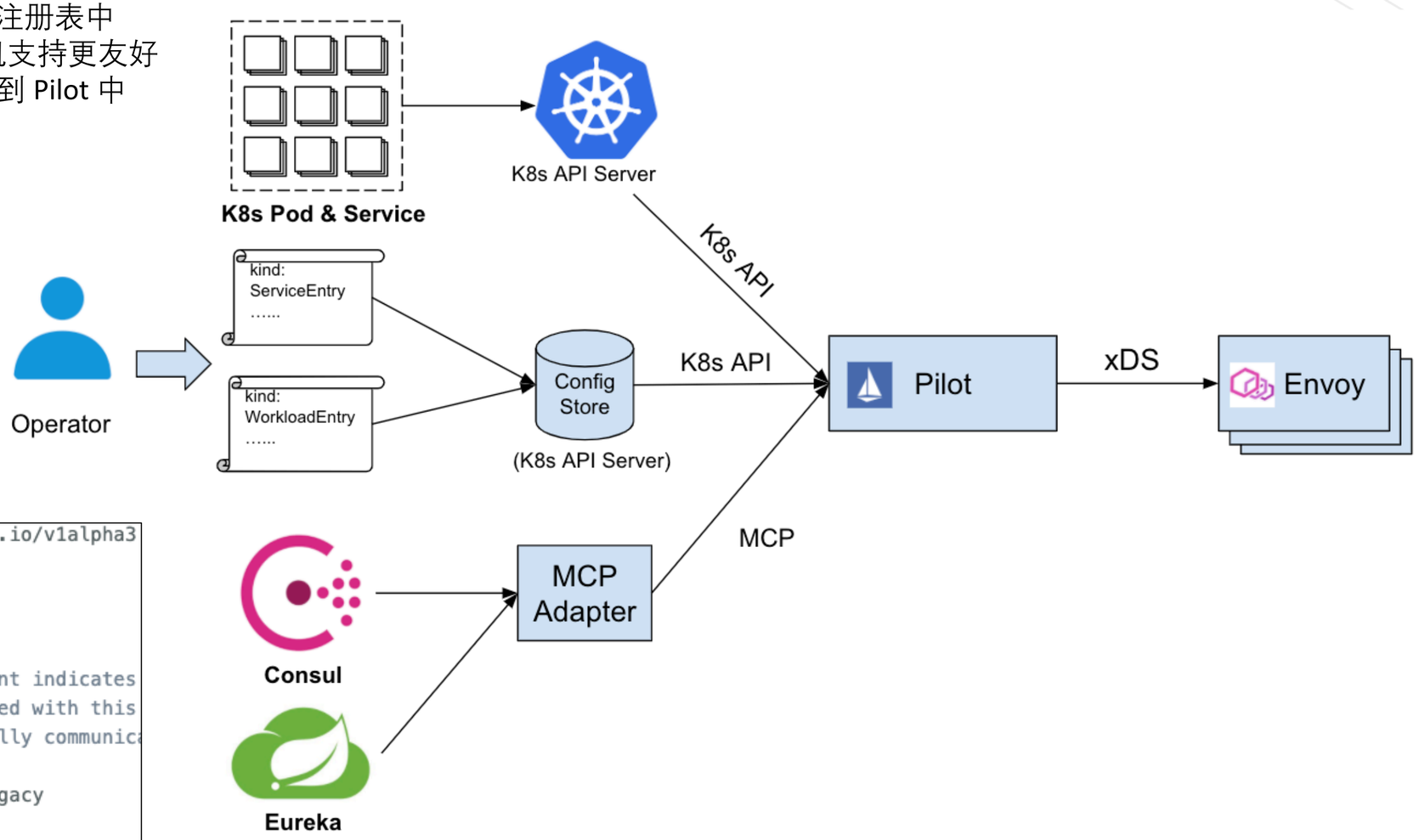


Istio 流量管理 – 控制面 – 服务发现

- **K8s Service** ： Pilot 直接支持
- **ServiceEntry**： 手动添加 Service 到 Pilot 内部注册表中
- **WorkloadEntry**：单独添加 Workload，对于虚机支持更友好
- **MCP 适配器**： 将第三方注册表中的服务加入到 Pilot 中

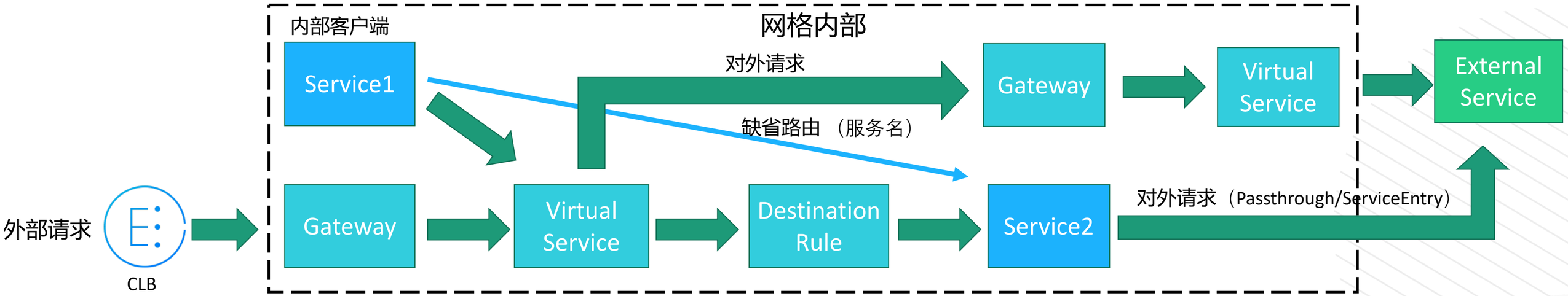
```yaml
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: details-svc
spec:
  hosts:
  - details.bookinfo.com
  location: MESH_INTERNAL
  ports:
  - number: 80
    name: http
    protocol: HTTP
  resolution: STATIC
  workloadSelector:
    labels:
      app: details-legacy
```

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: WorkloadEntry
metadata:
  name: details-svc
spec:
  # use of the service account indicates
  # sidecar proxy bootstrapped with this
  # sidecars will automatically communica
  # istio mutual TLS.
  serviceAccount: details-legacy
  address: 2.2.2.2
  labels:
    app: details-legacy
    instance-id: vm1
```



Consul MCP Adapter https://github.com/istio-ecosystem/consul-mcp
欢迎大家试用、共建！

Istio 流量管理 – 控制面 – 流量管理模型



定义网格入口
- 服务端口
- Host
- TLS 配置
-

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: httpbin-gateway
spec:
  selector:
    istio: ingressgateway # use Istio def
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
    hosts:
    - "httpbin.example.com"
```

路由配置
- 根据 Host 路由
- 根据 Header
- 根据 URI 路由

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: httpbin
spec:
  hosts:
  - "httpbin.example.com"
  gateways:
  - httpbin-gateway
  http:
  - match:
    - uri:
        prefix: /status
    - uri:
        prefix: /delay
    route:
    - destination:
        port:
          number: 8000
        host: httpbin
```

目的地流量策略配置
- LB 策略
- 连接池配置
- 断路器配置
- TLS 配置

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: reviews-cb-policy
spec:
  host: reviews.prod.svc.cluster.local
  trafficPolicy:
    connectionPool:
      tcp:
        maxConnections: 100
      http:
        http2MaxRequests: 1000
        maxRequestsPerConnection: 10
    outlierDetection:
      consecutiveErrors: 7
      interval: 5m
      baseEjectionTime: 15m
```

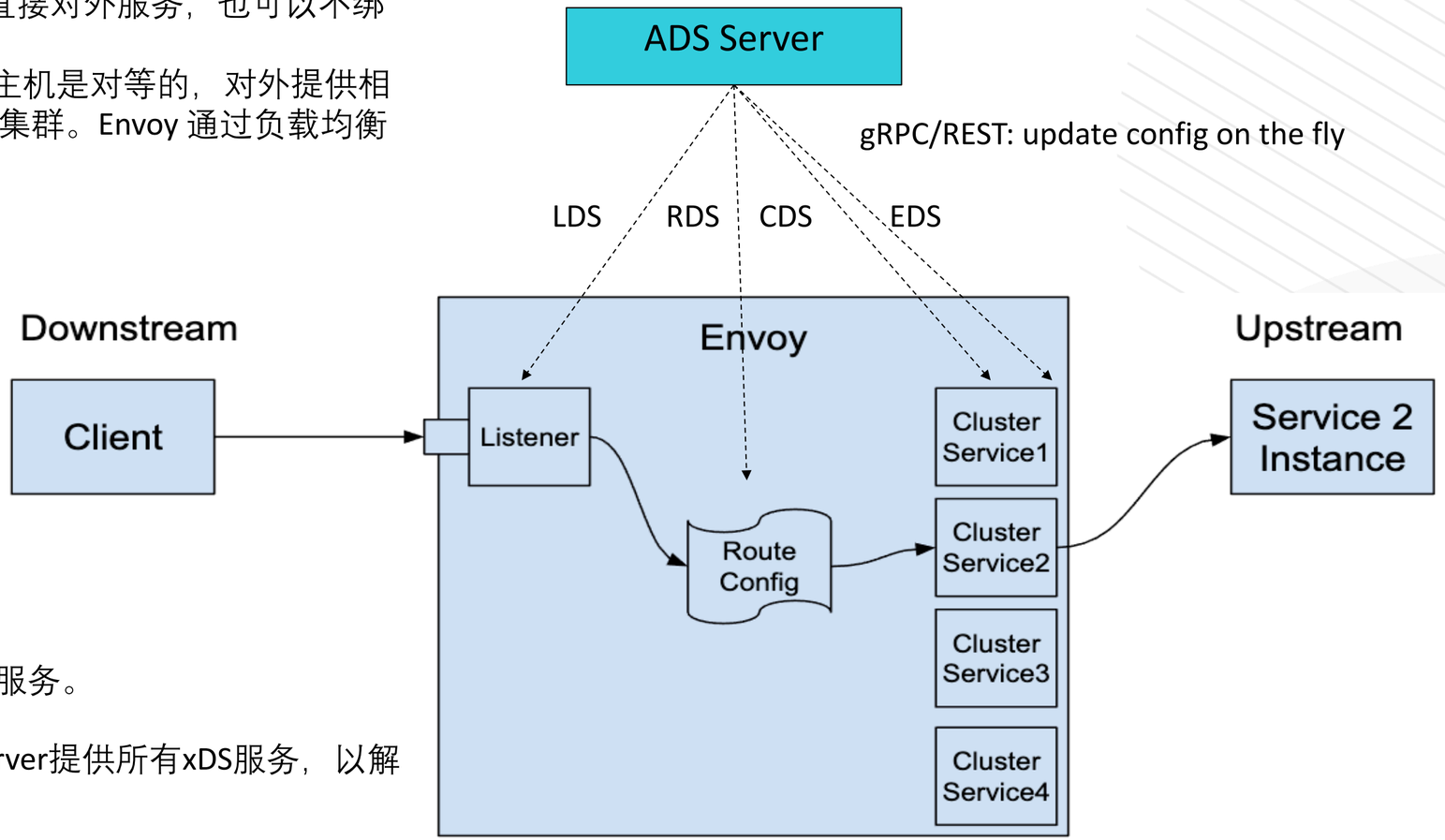
统一网格出口
- 出口地址（Gateway Workload）
- 出口端口

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: istio-egressgateway
spec:
  selector:
    istio: egressgateway
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
    hosts:
    - edition.cnn.com
```

Istio 流量管理 – 数据面 – Envoy配置模型和xDS协议

Envoy 配置模型的主要概念：

- **Downstream**：连接到 Envoy 的下游 Host，发送请求并接收响应。
- **Upstream**： 上游 Host 接收来自 Envoy 的连接和请求，并返回响应。
- **Listener**：监听器是命名网地址（可以是TCP socket 或者 Unix domain socket)，可以被下游客户端连接。在 Envoy 中,Listener 可以绑定到端口上直接对外服务，也可以不绑定到端口上，而是接收其他 listener 转发的请求。
- **Cluster**：集群是指 Envoy 连接的一组上游主机，集群中的主机是对等的，对外提供相同的服务，组成了一个可以提供负载均衡和高可用的服务集群。Envoy 通过负载均衡策略决定将请求路由到哪个集群成员。



xDS 协议的主要概念：

- **Listener Discovery Service (LDS)** : 监听器发现服务。
- **Route Discovery Service(RDS)** : 路由发现服务。
- **Cluster Discovery Service (CDS)**： 集群发现服务。
- **Endpoint Discovery Service (EDS)** ：集群中的服务实例发现服务。
- **Secret Discovery Service (SDS)** ：证书发现服务。
- **Aggregated Discovery Service(ADS):** 通过一个Aggregated Server提供所有xDS服务，以解决各个不同xDS服务的顺序导致的数据一致性问题。

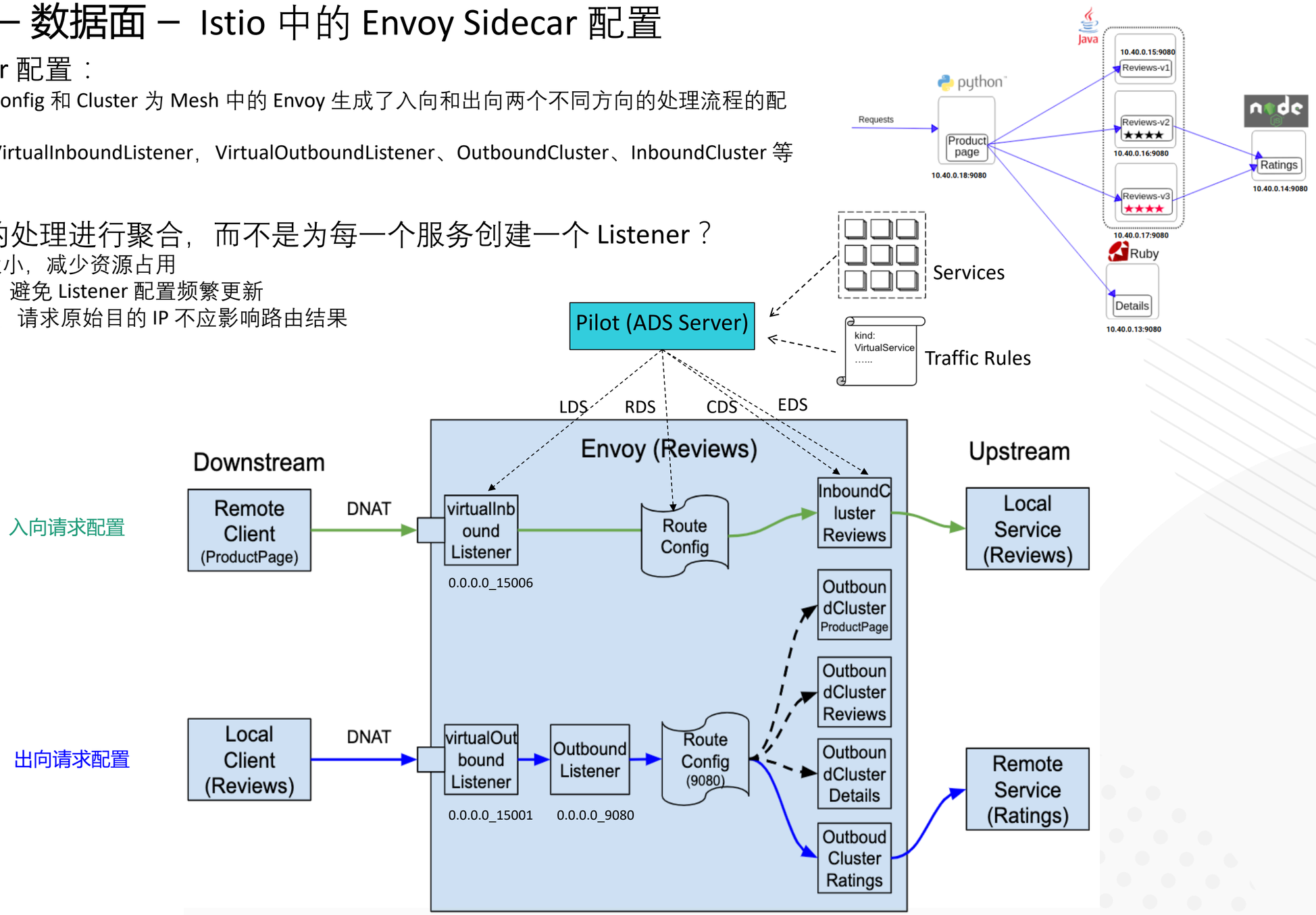
Istio 流量管理 – 数据面 – Istio 中的 Envoy Sidecar 配置

Istio中的 Envoy Sidecar 配置：

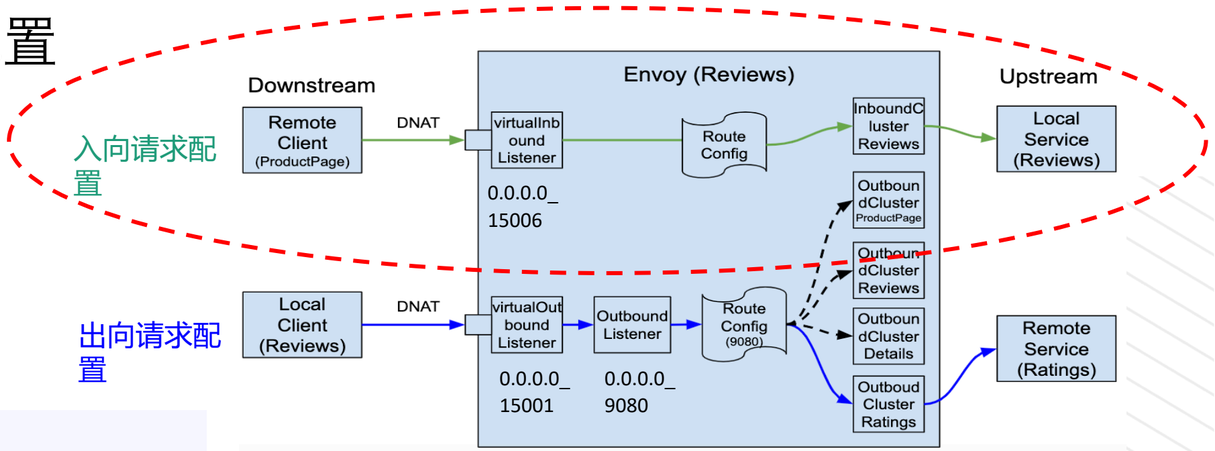
- Istio 通过 Listener、Route Config 和 Cluster 为 Mesh 中的 Envoy 生成了入向和出向两个不同方向的处理流程的配置。
- 在 Envoy 的基础上增加了 VirtualInboundListener，VirtualOutboundListener、OutboundCluster、InboundCluster 等概念。

为何按端口对 HTTP 的处理进行聚合，而不是为每一个服务创建一个 Listener？

- 降低 Listener 数量和配置大小，减少资源占用
- 兼容 headless 和虚机服务，避免 Listener 配置频繁更新
- 采用七层 header 进行路由，请求原始目的 IP 不应影响路由结果



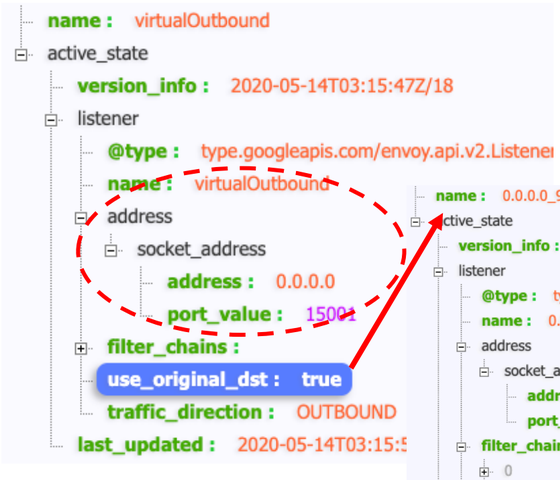
Istio 流量管理 – 数据面 – Envoy Sidecar Inbound 配置

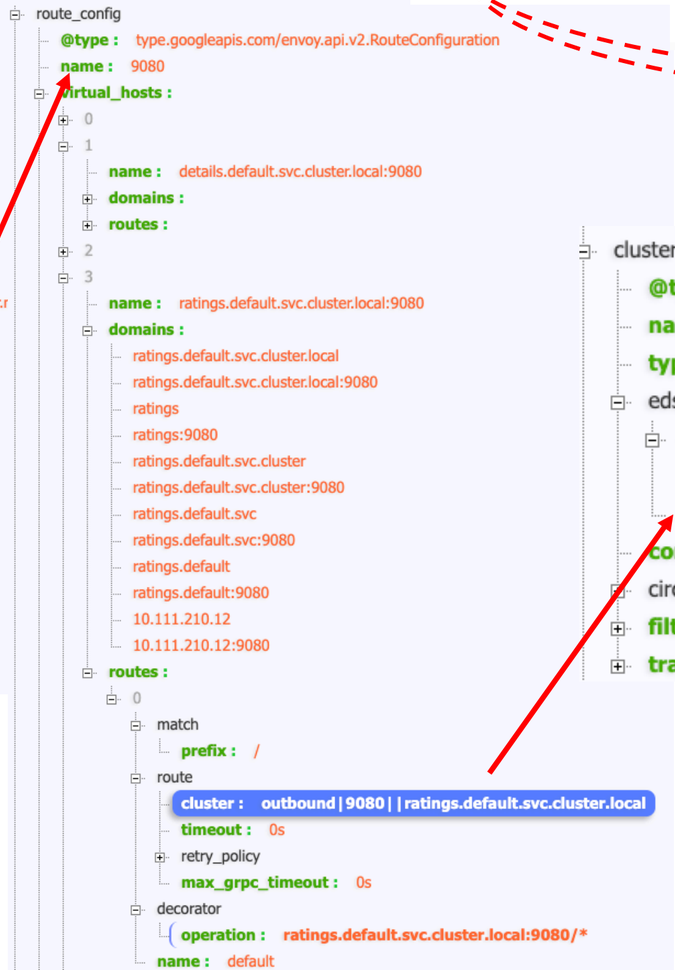


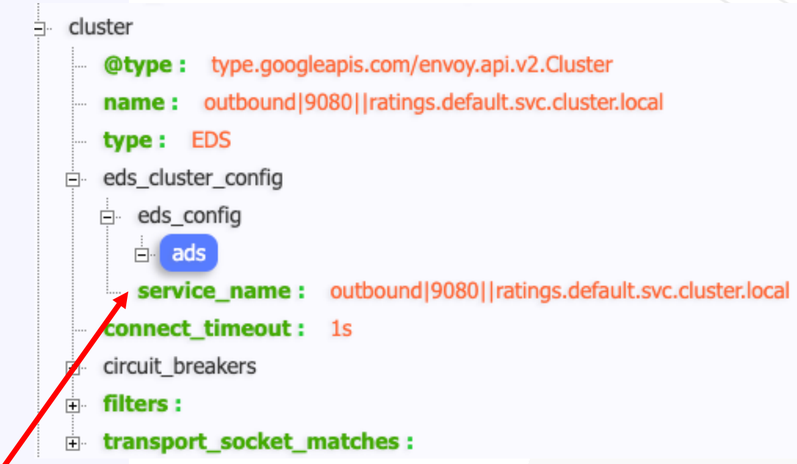

```
listener
  @type : type.googleapis.com/envoy.api.v2.Listener
  name : virtualInbound
  address
    socket_address
      address : 0.0.0.0
      port_value : 15006
  filter_chains :
    0
    1
    2
      filter_chain_match
        prefix_ranges :
          0
            address_prefix : 10.44.0.8
            prefix_len : 32
        destination_port : 9080
      filters :
        0
        1
          name : envoy.http_connection_manager
          typed_config
            @type : type.googleapis.com/envoy.config.filter.network.http_connection_manager.v2.HttpConnectionManager
            stat_prefix : inbound_10.44.0.8_9080
            route_config
              name : inbound|9080|http|reviews.default.svc.cluster.local
              virtual_hosts :
                0
                  name : inbound|http|9080
                  domains :
                    *
                  routes :
                    0
                      match
                        prefix : /
                      route
                        cluster : inbound|9080|http|reviews.default.svc.cluster.local
                        timeout : 0s
                        max_grpc_timeout : 0s
                      decorator
                        operation : reviews.default.svc.cluster.local:9080/*
                        name : default
              validate_clusters : false
            http_filters :
```

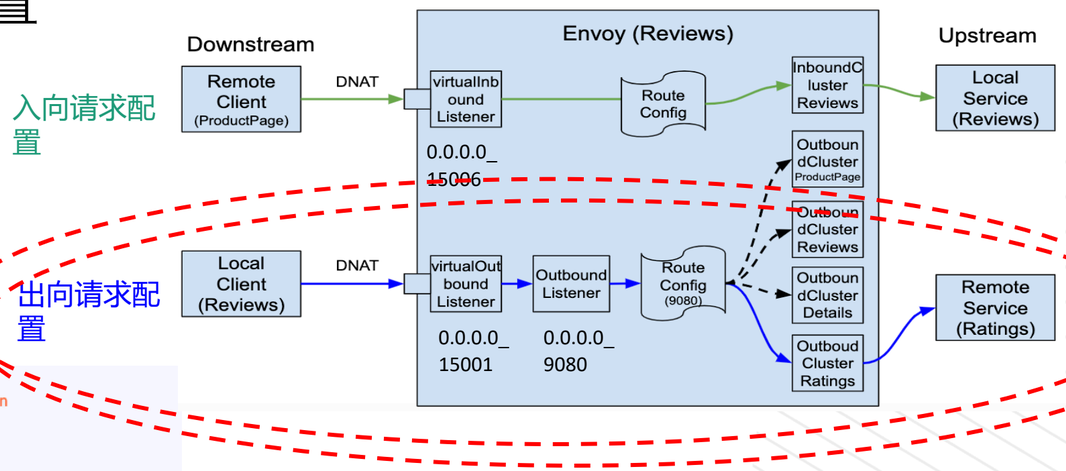
```
cluster
  @type : type.googleapis.com/envoy.api.v2.Cluster
  name : inbound|9080|http|reviews.default.svc.cluster.local
  type : STATIC
  connect_timeout : 1s
  circuit_breakers
    thresholds :
      0
        max_connections : 4294967295
        max_pending_requests : 4294967295
        max_requests : 4294967295
        max_retries : 4294967295
  load_assignment
    cluster_name : inbound|9080|http|reviews.default.svc.cluster.local
    endpoints :
      0
        lb_endpoints :
          0
            endpoint
              address
                socket_address
                  address : 127.0.0.1
                  port_value : 9080
  last_updated : 2020-05-14T03:15:51.450Z
5
```

Istio 流量管理 – 数据面 – Envoy Sidecar Outbound 配置





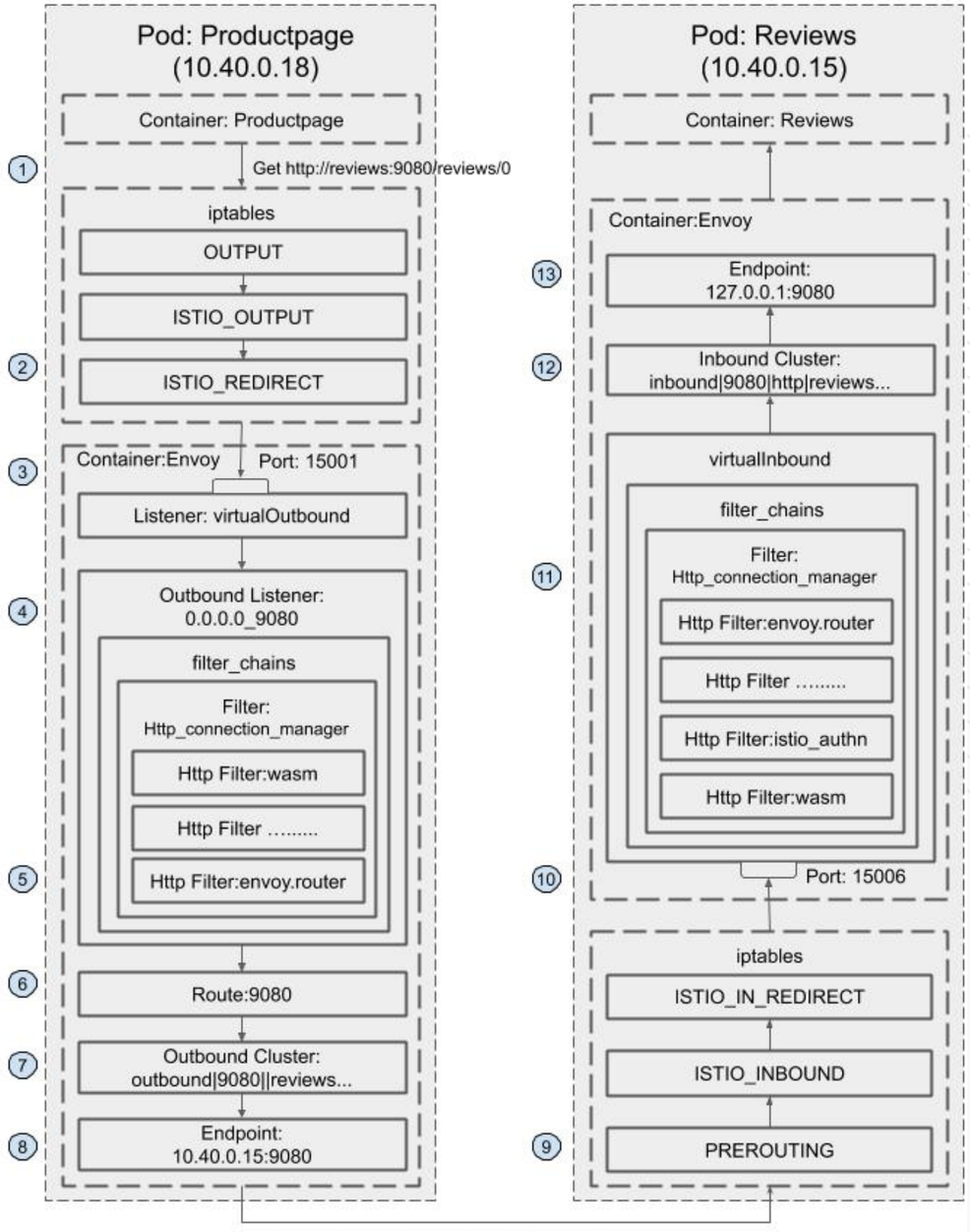




Istio 流量管理 – 数据面 – 端到端请求处理流程

以 Bookinfo 为例说明服务间 HTTP 调用的流量拦截及处理流程：

1. Productpage 发起对 reviews 服务的调用：http://reviews:9080/reviews/0 。
2. 请求被 productpage Pod 的 iptables 出向流量规则拦截，处理后重定向到本地 15001 端口。
3. Envoy 在 15001 端口上监听的 VirtualOutbound listener 收到了该请求。
4. 请求被 VirtualOutbound listener 根据原目标 IP（通配）和端口（9080）转发到 0.0.0.0_9080 这个 outbound listener。
5. 根据 0.0.0.0_9080 listener 的 http_connection_manager filter 配置，该请求采用 9080 route 进行分发。
6. 9080 这个 route 的配置中，host name 为 reviews:9080 的请求对应的 cluster 为 outbound|9080||reviews.default.svc.cluster.local。
7. outbound|9080||reviews.default.svc.cluster.local cluster 配置为通过EDS获取对应的Endpoint，通过 EDS 查询得到该 cluster 中有3个 endpoint。
8. 请求被 Envoy 转发到其中一个 endpoint 10.40.0.15，即 reviews-v1 所在的 pod。
9. 然后该请求被 reviews-v1 pod 的 iptables 入向流量规则拦截，处理后重定向到本地的 15006 端口。
10. Envoy 在 15006 端口上监听的 VirtualInbound listener 收到了该请求。
11. 根据匹配条件，请求被 VirtualInbound listener 内部配置的 Http connection manager filter 处理，该 filter 设置的路由配置为将其发送给 inbound|9080|http|reviews.default.svc.cluster.local 这个 inbound cluster。
12. inbound|9080|http|reviews.default.svc.cluster.local cluster 配置的 host 为 127.0.0.1:9080。
13. 请求被转发到 127.0.0.1:9080，即 reviews 服务进行业务处理。

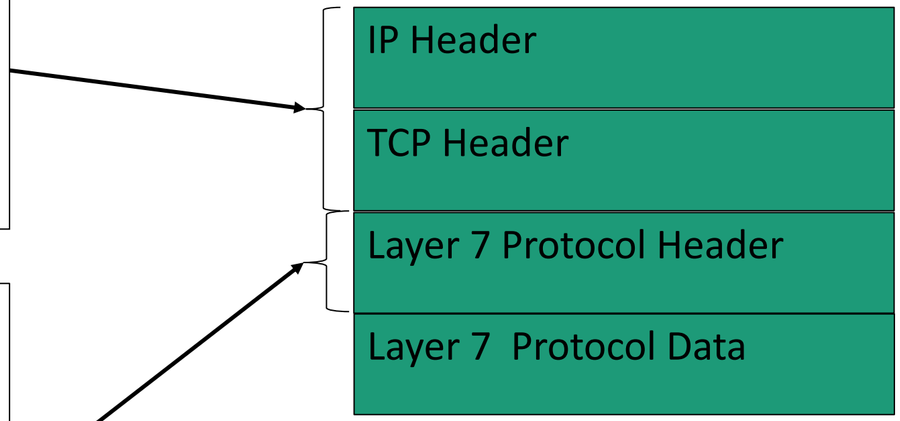


Istio 协议支持现状

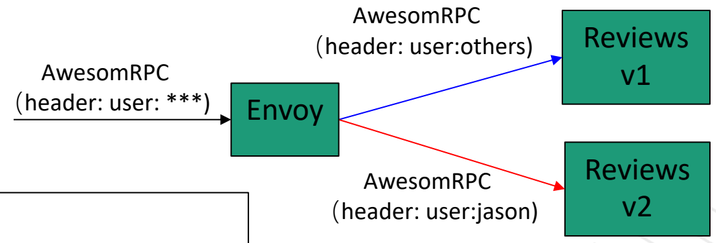
Istio 支持的七层协议非常有限：HTTP 1.1、 HTTP2、 gRPC
其余协议只能在四层进行处理（Thrift、Redis 等其他七层协议的控制面支持非常有限）

- 四层服务治理
  - 服务发现（基于 VIP 或者 Pod IP：DNS 只用于解析得到 IP，不能被 Envoy 感知）
  - LB、基于四层链接错误的 Retries 和 Circuit Breaker
  - 基于四层的路由（IP + Port）
  - 基于四层的 Metrics（TCP收发包数量等）

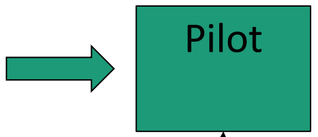
- 七层服务治理
  - 服务发现（基于服务的逻辑名称）
  - LB、基于应用协议的错误码进行 Retries 和 Circuit Breaker
  - 基于七层协议 Meta data 的路由（RPC协议中的调用服务名、方法名等）
  - Fault Injection（RPC 协议层的错误码）
  - RPC 调用的 Metrics（调用次数，调用失败率等）
  - Tracing



Istio 协议扩展：控制面和数据面需要进行的改动

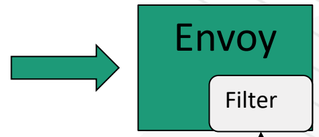



```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: reviews-route
spec:
  hosts:
  - reviews.prod.svc.cluster.local
  awesomeRPC:
  - name: ”canary-route"
    match:
    - headers:
        user:
          exact: jason
    route:
    - destination:
        host: reviews.prod.svc.cluster.local
        subset: v2
  - name: ”default"
    route:
    - destination:
        host: reviews.prod.svc.cluster.local
        subset: v1
```



Pilot 代码改动
- 解析 CRD
- 生成 xDS 配置下发

```
{
  "virtual_hosts": [
    {
      "name": "reviews.default.svc.cluster.local:9080",
      "services": [
        "reviews.default.svc.cluster.local",
        "reviews"
      ],
      "routes": [
        {
          "name": ”canary-route"
          "match": {
            "headers":  [
              {
                "name": ":user",
                "exact_match": "jason"
              }
            ],
          },
          "route": {
            "cluster": "outbound|9080||reviews.default.svc.cluster.local | v2",
          },
        },
        {
          "name": ”default"
          "route": {
            "cluster": "outbound|9080||reviews.default.svc.cluster.local | v1",
          },
        }
      ]
    }
  ],
}
```



AwesomeRPC Filter
- Decoding/encoding
- Parsing header
- Routing
- Load balancing
- Circuit breaker
- Fault injection
- Telemetry collecting

优点：
- 控制面改动小，可以快速实现对新协议的支持

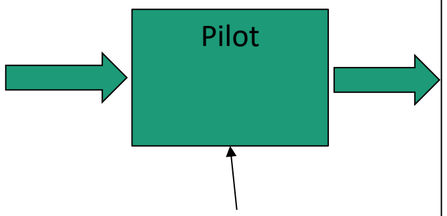
问题：
- Pilot 目前缺少一个良好的协议扩展机制
- Pilot 需要理解 Envoy filter 中协议特定的知识
- Pilot 代码中维护众多七层协议的代价较大

Istio 协议扩展：常见七层协议的路由

Protocol	Destination service	Parameters could be used for routing
HTTP 1.1	host	host, path， method headers
HTTP 2	pseudo header: authority	pseudo header: authority, path， method， headers
gRPC	HTTP 2 path	Request-Headers(Delivered as HTTP2 headers)
TARS	ServantName	ServantName, FuncName, Context
Dubbo	service name	service name, service version, service method
Any RPC Protocol	service name in message header	some key:value pairs in message header

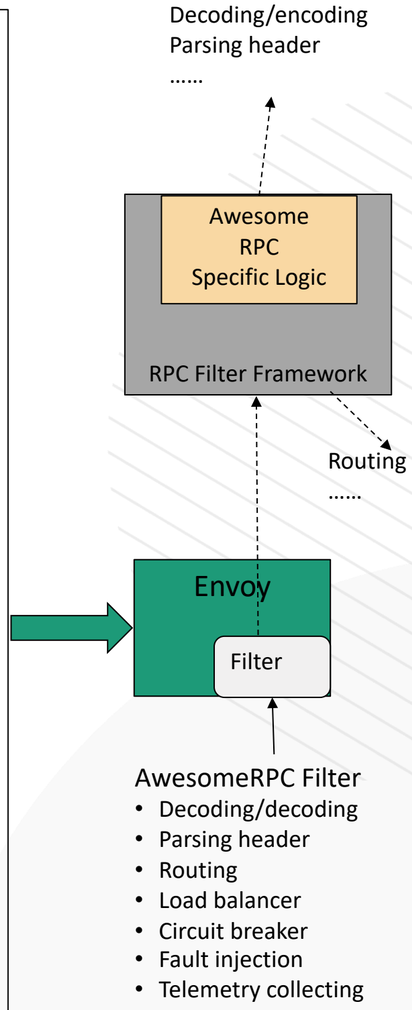
Istio 协议扩展：协议无关的通用路由框架

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: reviews-route
spec:
  hosts:
  - reviews.prod.svc.cluster.local
  protocol:  awesomeRPC
  - name: ”canary-route"
    - match:
      attributes:
      - user:
        exact: jason
    route:
      - destination:
        host: reviews.prod.svc.cluster.local
        subset: v2
  - name: ”default"
    route:
    - destination:
        host: reviews.prod.svc.cluster.local
        subset: v1
```



Pilot 将通用协议路由规则解析为统一格式的 xDS 配置下发。

```
{
  "virtual_hosts": [
    {
      "name": "reviews.default.svc.cluster.local:9080",
      "services": [
        "reviews.default.svc.cluster.local",
        "reviews"
      ],
      "routes": [
        {
          "name": ”canary-route"
          "match": {
            " attributes":  [
              {
                "name": ":user",
                "exact_match": "jason"
              }
            ],
          },
          "route": {
            "cluster": "outbound|9080||reviews.default.svc.cluster.local | v2",
          },
        },
        {
          "name": ”default"
          "route": {
            "cluster": "outbound|9080||reviews.default.svc.cluster.local | v1",
          },
        }
      ]
    }
  ],
}
```



AwesomeRPC Filter

- Decoding/decoding
- Parsing header
- Routing
- Load balancer
- Circuit breaker
- Fault injection
- Telemetry collecting

优点：

- 控制面的扩展性好

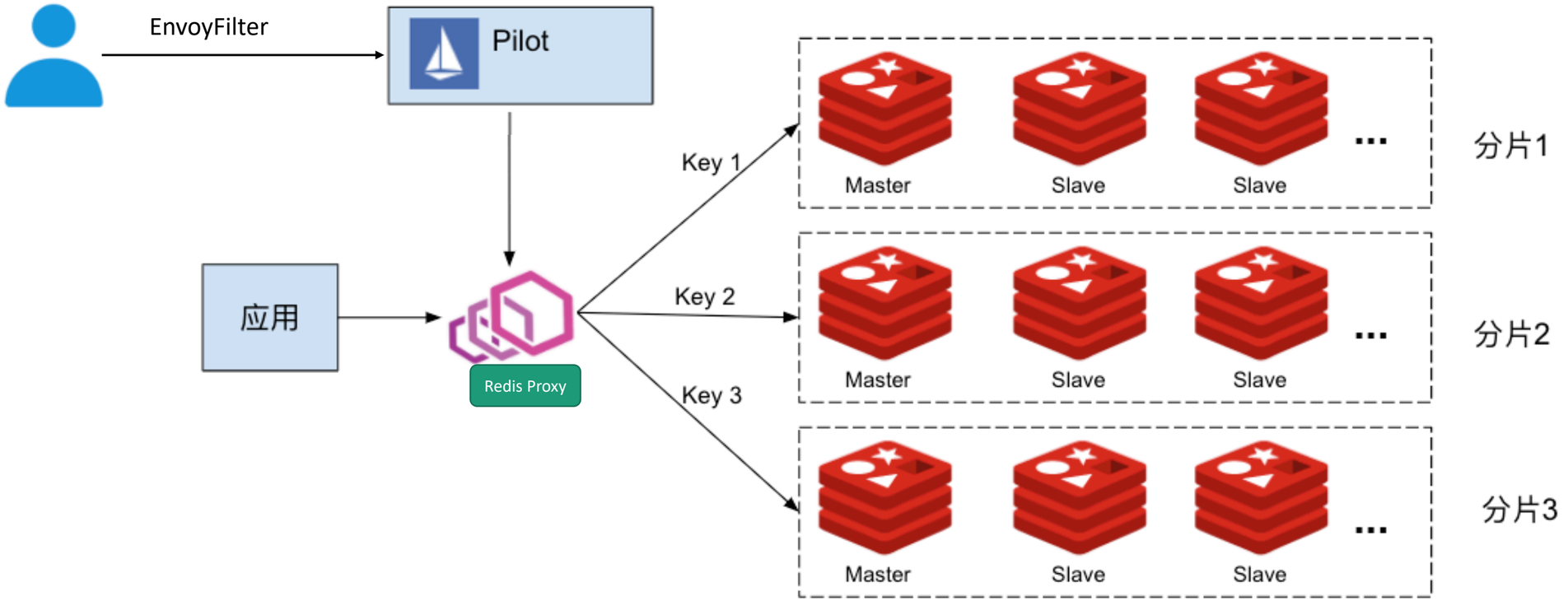
问题：

- 需要修改 Pilot、xDS 协议 和 Envoy Filter

https://docs.google.com/document/d/1g3AJNYwbh04-BVajshmcWpeLyGUanbRnihlXU6uliOY

Istio 协议扩展：EnvoyFilter

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: EnvoyFilter
metadata:
  name: add-redis-proxy
  namespace: istio-system
spec:
  configPatches:
  - applyTo: NETWORK_FILTER
    match:
      listener:
        name: ${REDIS_VIP}_6379            # Replace REDIS_VIP with the cluster
        filterChain:
          filter:
            name: "envoy.filters.network.tcp_proxy"
    patch:
      operation: REPLACE
      value:
        name: envoy.redis_proxy
        typed_config:
          "@type": type.googleapis.com/envoy.config.filter.network.redis_proxy.v
          stat_prefix: redis_stats
          prefix_routes:
            catch_all_route:
              request_mirror_policy:           # Send requests to the mirror cl
              - cluster: outbound|6379||redis-mirror.redis.svc.cluster.local
                exclude_read_commands: True    # Mirror write commands only:
              cluster: custom-redis-cluster
          settings:
            op_timeout: 5s
            enable_redirection: true
            enable_command_stats: true
            read_policy: REPLICA               # Send read requests to replica
```



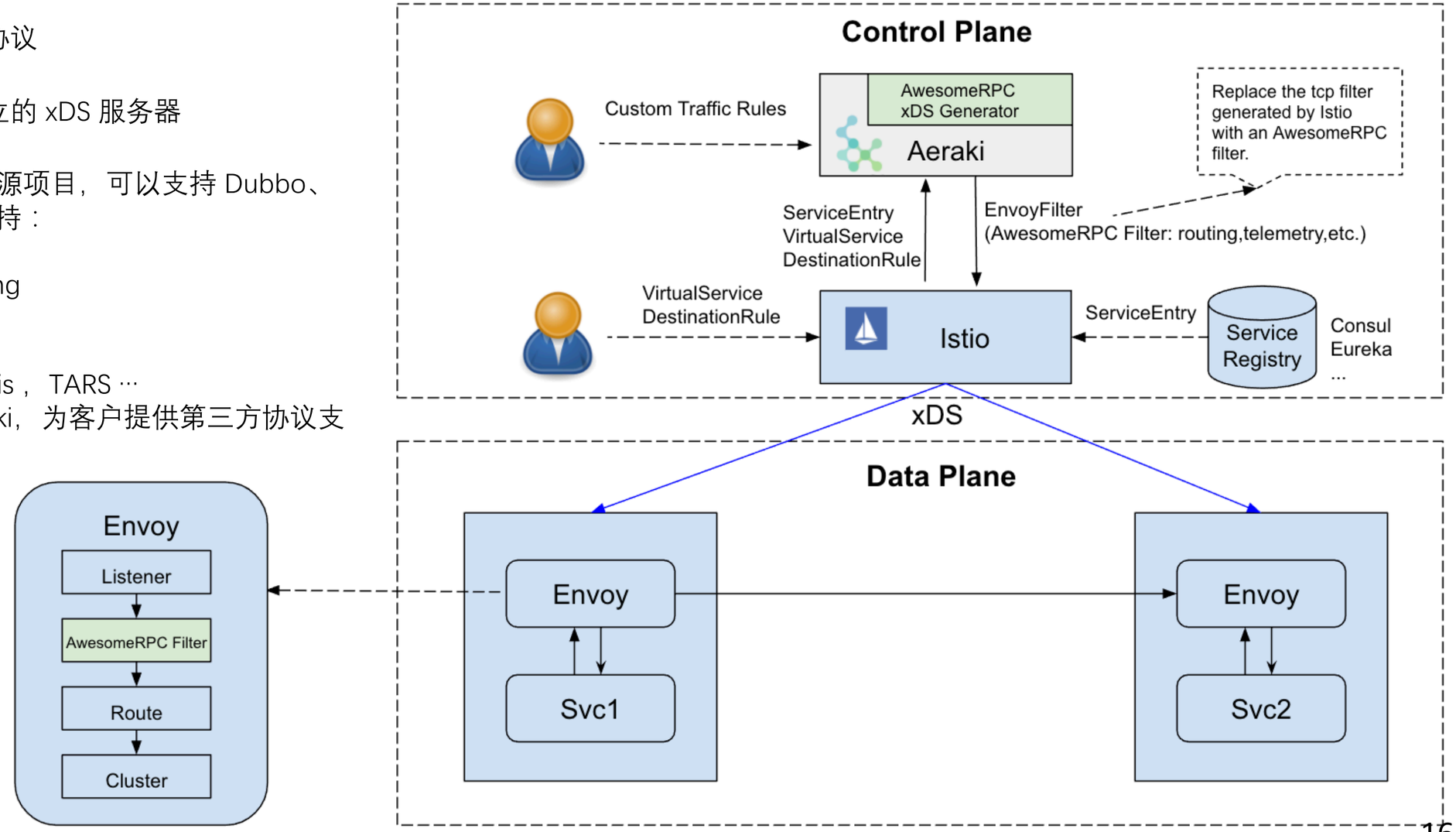
https://github.com/zhaohuabing/istio-redis-culster

Istio 协议扩展：控制面扩展机制

优点：
- 对 Istio 和 Envoy 无侵入
- 扩展性强，基本可以支持任何七层协议

问题：
- 工作量较大，相当于实现了一个独立的 xDS 服务器

Aeraki: 为Istio提供七层协议扩展的开源项目，可以支持 Dubbo、Thrift、Redis以及私有协议，目前已支持：
  - Dubbo 缺省路由
  - Dubbo version-based routing
  - Dubbo traffic splitting
- 后续规划：
  - 其他协议支持：Thrift，Redis，TARS …
  - 在 TCM 中提供托管的 Aeraki，为客户提供第三方协议支持



THANK YOU!

感谢聆听！