SERVICE MESH SUMMIT 2022
服务网格峰会

主办方 云原生社区 Cloud Native Community

如何基于Istio 实现Mesh自定义扩展功能

曾宇星 阿里云云原生架构师

自我介绍

- 阿里云技术专家、云原生架构师
- 长期从事服务端开发和架构工作，10 多年分布式领域后台开发经验，目前主要关注于云原生、高性能、高可用分布式架构。
- 有多年 Service Mesh 、Envoy 网关、Kubernetes 容器平台等云原生领域相关开发工作经验。目前在阿里云服务网格团队从事 Service Mesh 云产品研发和架构设计工作。

Agenda

SERVICE MESH SUMMIT 2022
服务网格峰会

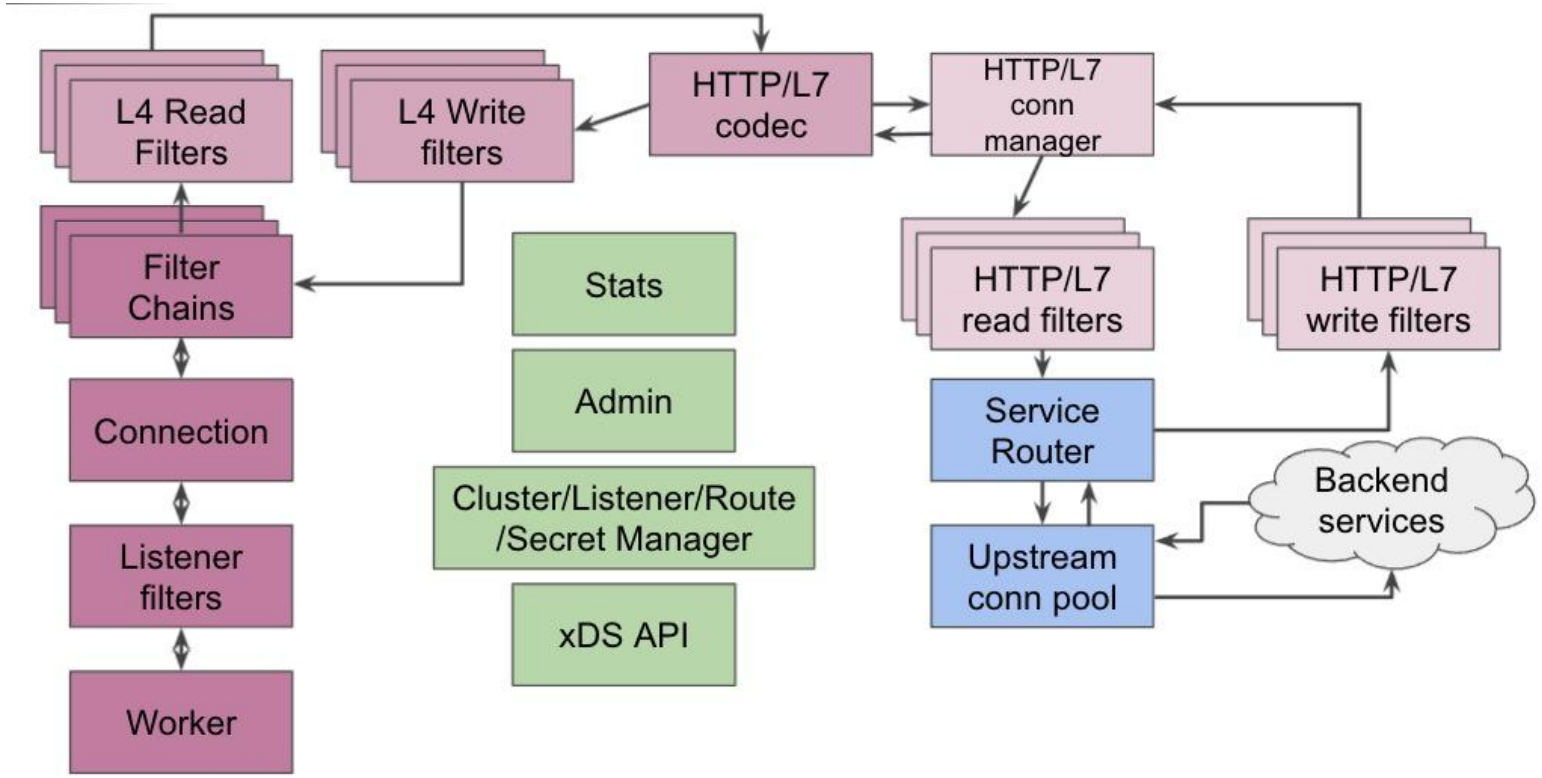
- 为什么需要自定义扩展能力
- Istio 下几种自定义扩展方式详细介绍
- 开发/运维人员使用最佳实践
- 总结

为什么Mesh需要自定义扩展能力？

为什么Mesh需要自定义扩展能力？

- 业务应用的多样性和差异性，导致Mesh用户需求的多样性
- Istio 社区功能和用户实际需求有差异，需要补齐，特别是长尾非通用的小众需求

Envoy 扩展方式原理

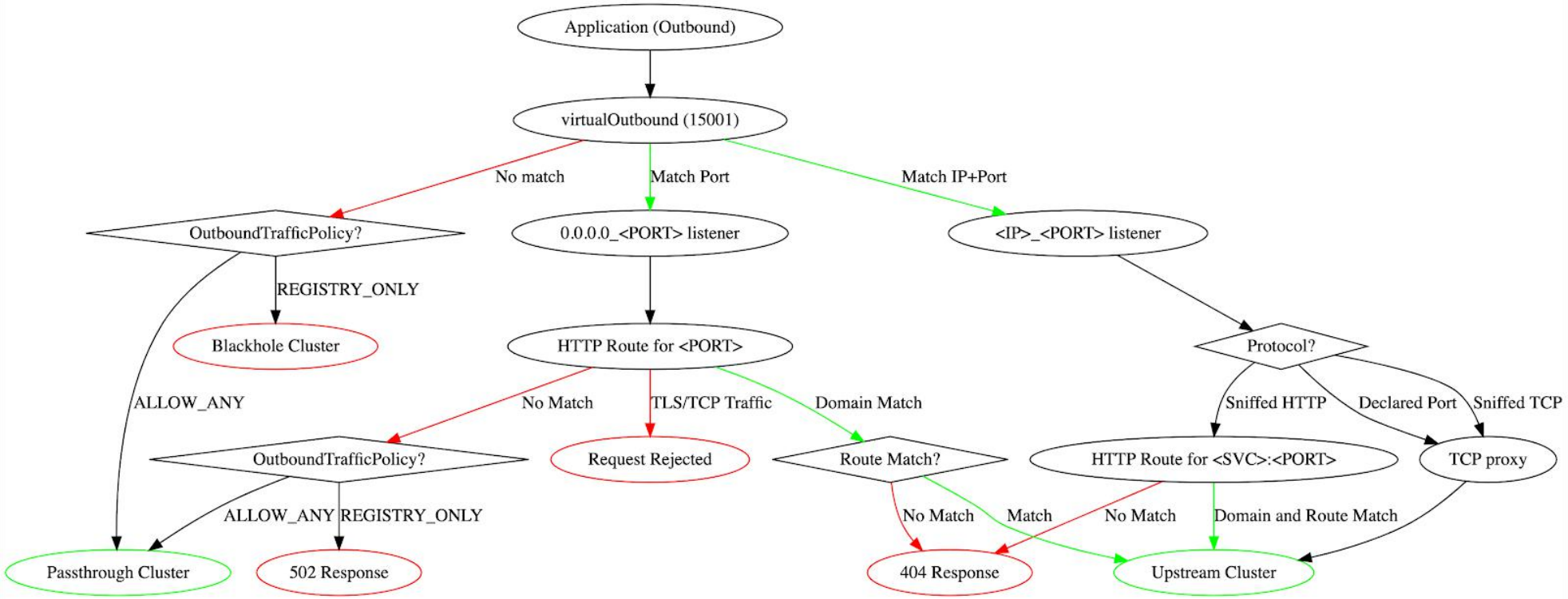


- Lua
- Wasm
- RPC 请求转发到外部进程

Istio 扩展方式

- EnvoyFilter API
- WasmPlugin API
- xDS 扩展（比如dubbo RDS)
- xDS 之ECDS (type.googleapis.com/envoy.config.core.v3.TypedExtensionConfig)

xDS data model

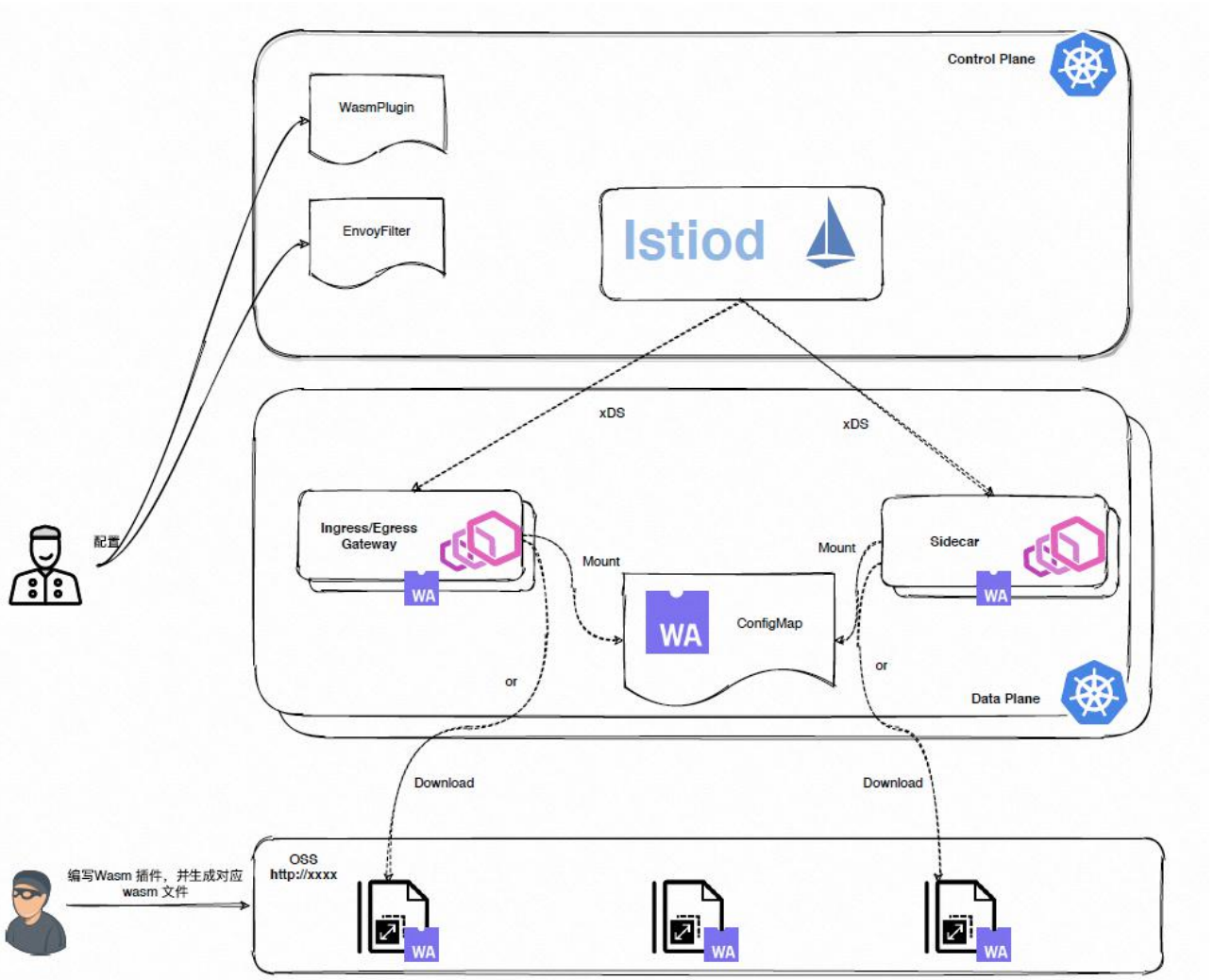


SERVICE MESH SUMMIT 2022
服务网格峰会

Istio 下自定义扩展方式详细介绍

自定义扩展方式之一： Wasm

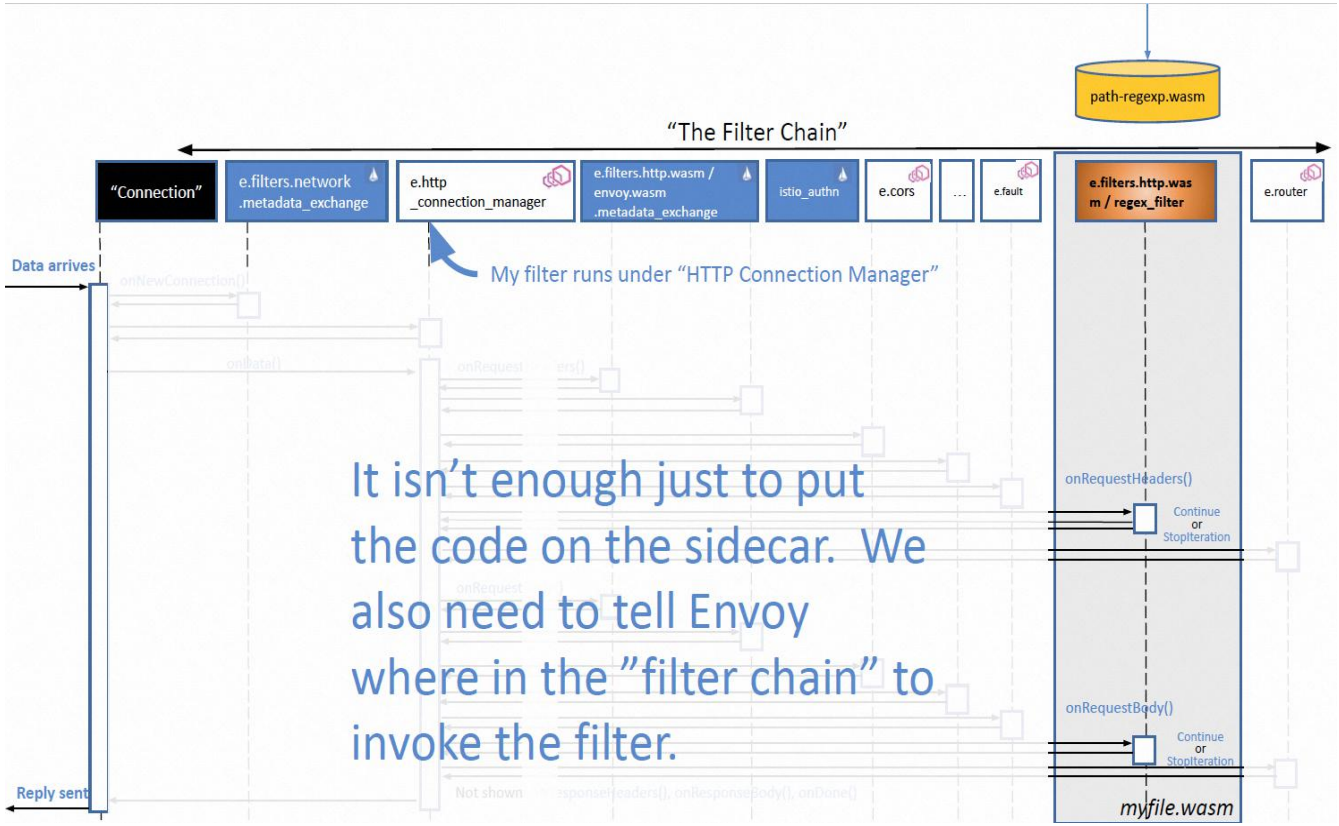
- kubectl create configmap new-filter \
  --from-file=new-filter.wasm=mycode.wasm
- kubectl create envoyfilter xxx 或者
- kubectl create wasmplugin xxx



https://help.aliyun.com/document_detail/176520.html
https://istio.io/latest/docs/reference/config/proxy_extensions/wasm-plugin/

自定义扩展方式之一：数据面Wasm 支持

- Wasm runtime
- 支持主流语言 C++、Go(TinyGo)、AssemblyScript、Rust
- 基于filter_chain 可配置生效位置



https://istio.io/latest/docs/concepts/wasm/

自定义扩展方式之一：控制面接口EnvoyFilter

通用配置方式：

需要对数据面的配置格式有一定的了解

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: EnvoyFilter
metadata:
  name: productpage-v1-examplefilter
  labels:
    asm-system: 'true'
    provider: asm
spec:
  configPatches:
  - applyTo: HTTP_FILTER
    match:
      context: SIDECAR_INBOUND
      proxy:
        proxyVersion: '^1\.*.*'
      listener:
        filterChain:
          filter:
            name: envoy.filters.network.http_connection_manager
            subFilter:
              name: envoy.filters.http.router
    patch:
      operation: INSERT_BEFORE
      value:
        typed_config:
          "@type": type.googleapis.com/envoy.extensions.filters.http.wasm.v3.Wasm
          config:
            name: example-filter
            rootId: my_root_id
            vmConfig:
              code:
                local:
                  filename: /var/local/lib/wasm-filters/example-filter.wasm
              runtime: envoy.wasm.runtime.v8
              vmId: example-filter
              allow_precompiled: true
        name: envoy.filters.http.wasm
  workloadSelector:
    labels:
      app: productpage
      version: v1
```

自定义扩展方式之一： 控制面接口WasmPlugin

Istio 原生 API

```yaml
apiVersion: extensions.istio.io/v1alpha1
kind: WasmPlugin
metadata:
  name: openid-connect
  namespace: istio-ingress
spec:
  selector:
    matchLabels:
      istio: ingressgateway
  url: oci://private-registry:5000/openid-connect/openid:latest
  imagePullPolicy: IfNotPresent
  imagePullSecret: private-registry-pull-secret
  phase: AUTHN
  pluginConfig:
    openid_server: authn
    openid_realm: ingress
  vmConfig:
    env:
    - name: POD_NAME
      valueFrom: HOST
    - name: TRUST_DOMAIN
      value: "cluster.local"
```

生效范围

wasm 插件对应镜像

镜像获取 secret

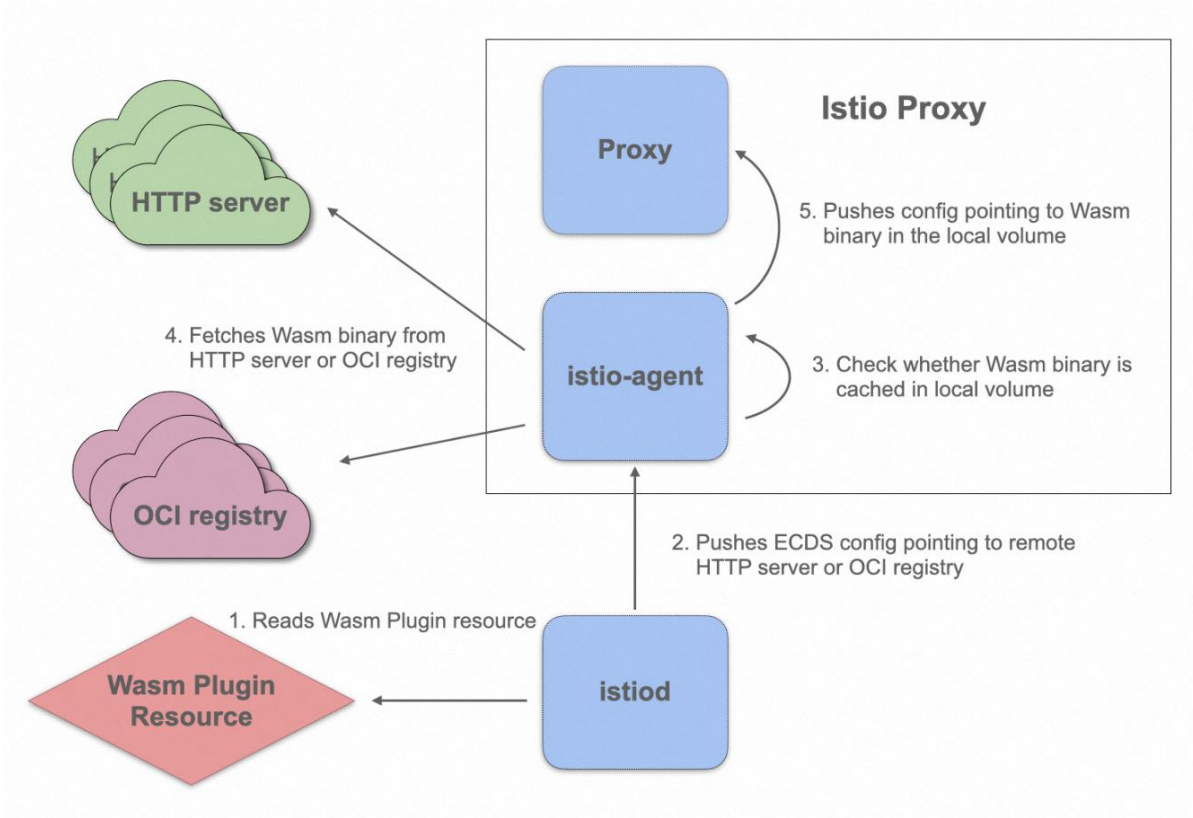
插件配置


```yaml
apiVersion: extensions.istio.io/v1alpha1
kind: WasmPlugin
metadata:
  name: openid-connect
  namespace: istio-ingress
spec:
  selector:
    matchLabels:
      istio: ingressgateway
  url: https://private-bucket/filters/openid.wasm
  imagePullPolicy: Always
  phase: AUTHN
  pluginConfig:
    openid_server: authn
    openid_realm: ingress
  vmConfig:
    env:
    - name: POD_NAME
      valueFrom: HOST
    - name: TRUST_DOMAIN
      value: "cluster.local"
```


https://istio.io/latest/docs/reference/config/proxy_extensions/wasm-plugin/

自定义扩展方式之一： WasmPlugin 实现原理

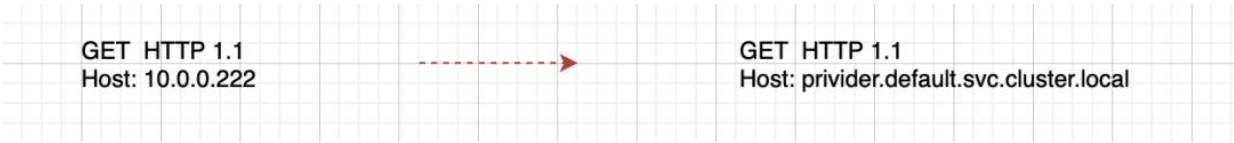
Istio 原生 API



https://istio.io/latest/docs/reference/config/proxy_extensions/wasm-plugin/

自定义扩展方式之二：Lua + EnvoyFilter

SpringCloud 流量模型如何适配 istio ？




```yaml
apiVersion: networking.istio.io/v1alpha3
kind: EnvoyFilter
metadata:
  labels:
    provider: "asm"
    asm-system: "true"
  name: nacos-subscribe-lua
  namespace: istio-system
spec:
  configPatches:
    # The first patch adds the lua filter to the listener/http connection manager.
  - applyTo: HTTP_FILTER
    match:
      proxy:
        proxyVersion: "^1.*"
      context: SIDECAR_OUTBOUND
      listener:
        portNumber: 8848
        filterChain:
          filter:
            name: "envoy.filters.network.http_connection_manager"
            subFilter:
              name: "envoy.filters.http.router"
    patch:
      operation: INSERT_BEFORE
      value: # lua filter specification
       name: envoy.lua
       typed_config:
          "@type": "type.googleapis.com/envoy.extensions.filters.http.lua.v3.Lua"
          inlineCode: |
            -- copyright: ASM (Alibaba Cloud ServiceMesh)
            function envoy_on_request(request_handle)
              local request_headers = request_handle:headers()
              -- /nacos/v1/ns/instance/list?healthyOnly=false&namespaceId=public&clientIP=11.122.63.81&serviceName=DEFAULT_GROUP%40%40service-provider&udpPort=53174&encoding=UTF-8
              local path = request_headers:get(":path")
              if string.match(path,"^/nacos/v1/ns/instance/list") then
                local servicename = string.gsub(path,".*&serviceName.*40([%w.\\_\\-]+)&.*","%1")
                request_handle:streamInfo():dynamicMetadata():set("context", "request.path", path)
                request_handle:streamInfo():dynamicMetadata():set("context", "request.servicename", servicename)
                request_handle:logInfo("subscribe for serviceName: " .. servicename)
              else
                request_handle:streamInfo():dynamicMetadata():set("context", "request.path", "")
              end
            end
            function envoy_on_response(response_handle)
              local request_path = response_handle:streamInfo():dynamicMetadata():get("context")["request.path"]
              if request_path == "" then
                 return
              end
              local servicename = response_handle:streamInfo():dynamicMetadata():get("context")["request.servicename"]
              response_handle:logInfo("modified response ip to serviceName:" .. servicename)
              local bodyObject = response_handle:body(true)
              local body= bodyObject:getBytes(0,bodyObject:length())
              body = string.gsub(body,"%s+","")
              body = string.gsub(body,"(ip\":\")(%d+.%d+.%d+.%d+)","%1"..servicename)
              response_handle:body():setBytes(body)
            end
```

- 依赖Envoy Lua filter
- EnvoyFilter patch 配置
- Example: 使用Lua 脚本支持 SpringCloud 上Mesh

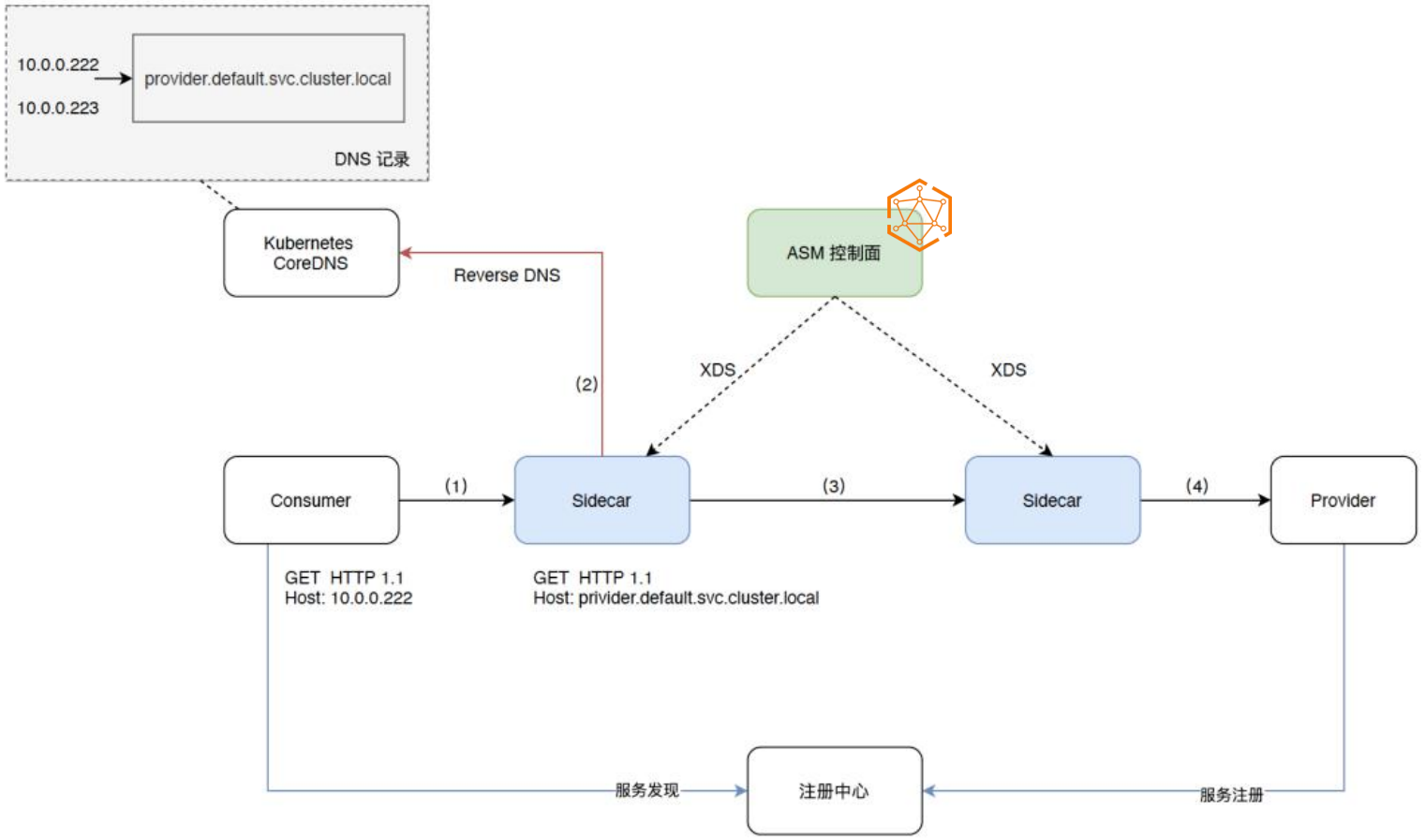
https://help.aliyun.com/document_detail/383257.html

自定义扩展方式之二：Lua + EnvoyFilter


```
inlineCode: |
   -- copyright: ASM (Alibaba Cloud ServiceMesh)
   function envoy_on_request(request_handle)
     local request_headers = request_handle:headers()
     -- /nacos/v1/ns/instance/list?healthyOnly=false&namespaceId=public&clientIP=11.122.63.81&serviceName=DEFAULT_GROUP%
     local path = request_headers:get(":path")
     if string.match(path,"^/nacos/v1/ns/instance/list") then
       local servicename = string.gsub(path,".*&serviceName.*40([%w.\\_\\-]+)&.*","%1")
       request_handle:streamInfo():dynamicMetadata():set("context", "request.path", path)
       request_handle:streamInfo():dynamicMetadata():set("context", "request.servicename", servicename)
       request_handle:logInfo("subscribe for serviceName: " .. servicename)
     else
       request_handle:streamInfo():dynamicMetadata():set("context", "request.path", "")
     end
   end
   function envoy_on_response(response_handle)
     local request_path = response_handle:streamInfo():dynamicMetadata():get("context")["request.path"]
     if request_path == "" then
        return
     end
     local servicename = response_handle:streamInfo():dynamicMetadata():get("context")["request.servicename"]
     response_handle:logInfo("modified response ip to serviceName:" .. servicename)
     local bodyObject = response_handle:body(true)
     local body= bodyObject:getBytes(0,bodyObject:length())
     body = string.gsub(body,"%s+","")
     body = string.gsub(body,"(ip\":\")(%d+.%d+.%d+.%d+)","%1"..servicename)
     response_handle:body():setBytes(body)
   end
```

https://help.aliyun.com/document_detail/383257.html

自定义扩展方式之三样例：支持 SpringCloud



https://help.aliyun.com/document_detail/383257.html

自定义扩展方式之三样例：支持 SpringCloud

- EnvoyFilter patch 配置
- Example:

使用com.aliyun.reverse_dns filter 支持SpringCloud

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: EnvoyFilter
metadata:
  labels:
    provider: "asm"
    asm-system: "true"
  name: any-spring-cloud-support
  namespace: istio-system
spec:
  configPatches:
  - applyTo: HTTP_FILTER
    match:
      proxy:
        proxyVersion: "^1.*"
      context: SIDECAR_OUTBOUND
      listener:
        portNumber: 8070
        filterChain:
          filter:
            name: "envoy.filters.network.http_connection_manager"
            subFilter:
              name: "envoy.filters.http.router"
    patch:
      operation: INSERT_BEFORE
      value: # reverse_dns filter specification
       name: com.aliyun.reverse_dns
       typed_config:
         "@type": "type.googleapis.com/udpa.type.v1.TypedStruct"
         type_url: type.googleapis.com/envoy.config.filter.reverse_dns.v3alpha.CommonConfig
         value:
           pod_cidrs:
           - "10.0.128.0/18"
```

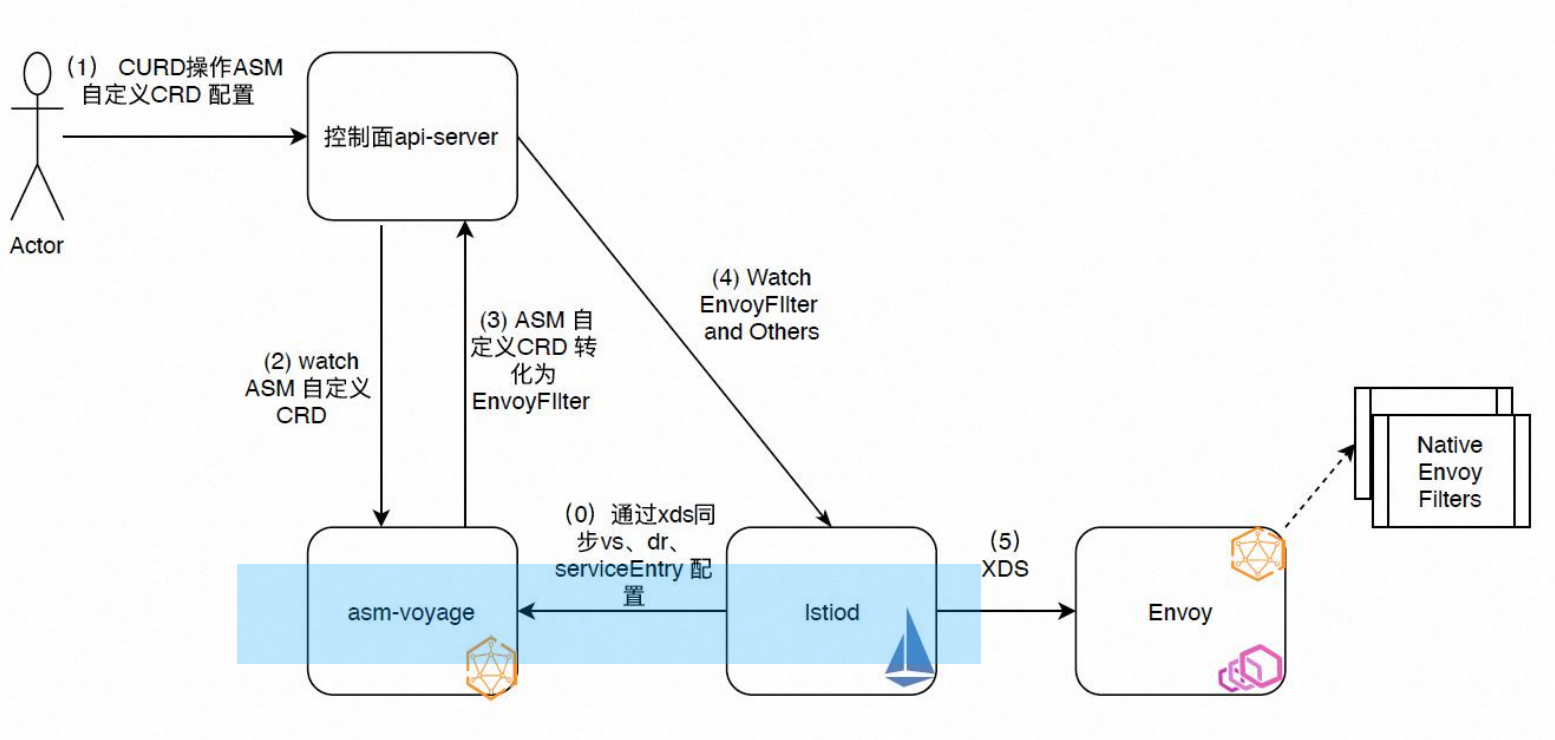
扩展的native Filter : com.aliyun.reverse_dns

https://help.aliyun.com/document_detail/383257.html

EnvoyFilter 的不足

- ✓ EnvoyFilter 功能很强大，但也非常具有破坏性，需要使用者对数据面Envoy 对应配置文件有一定的了解
- ✓ EnvoyFilter 的patch 顺序很脆弱，和创建时间相关
- ✓ Istiod 只能对EnvoyFilter 进行非常有限的验证

自定义扩展方式之三：Envoy Native Filter + Voyage



asm-voyage 是阿里云ASM 的一个控制面扩展组件

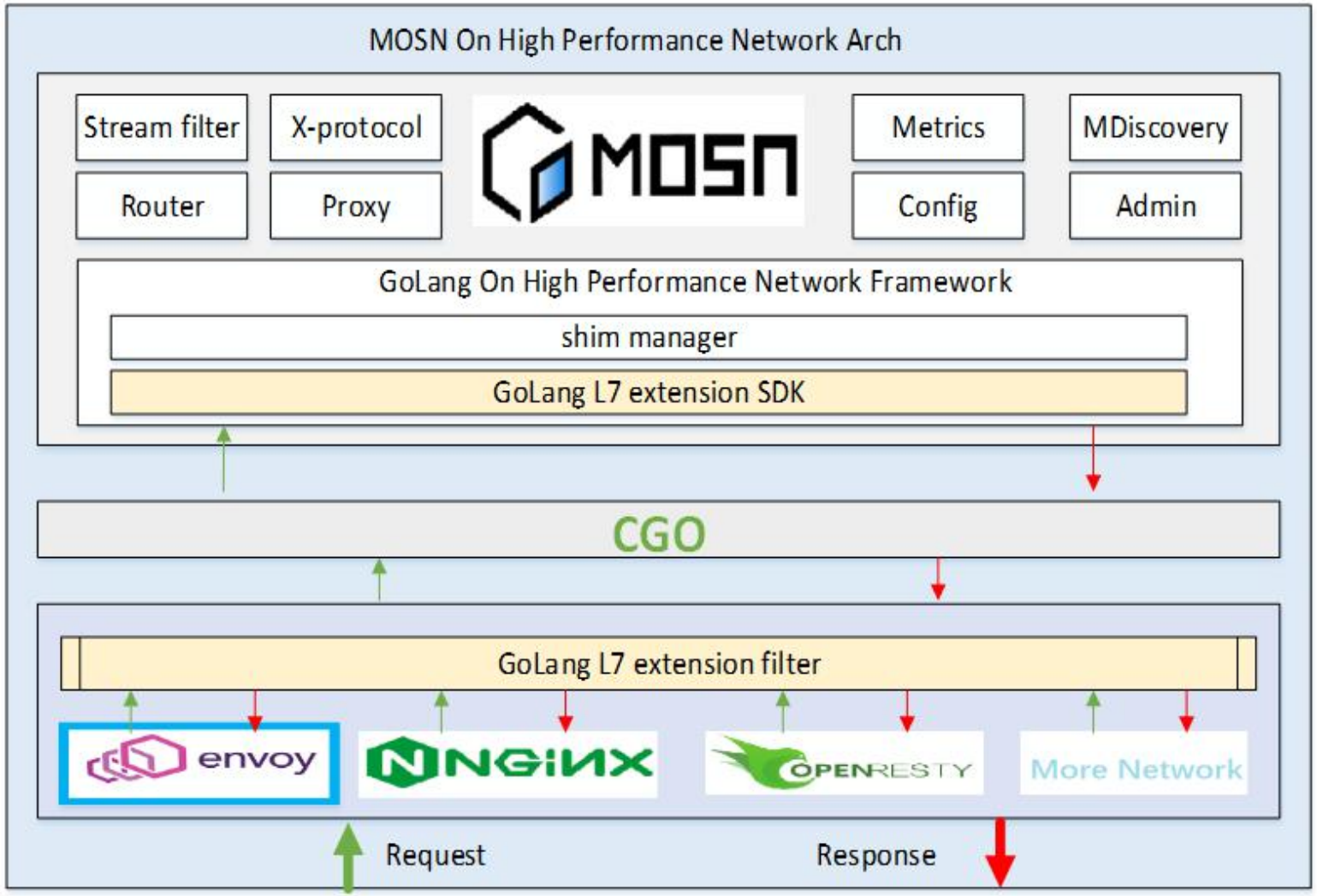
自定义扩展方式之三：控制面CRD 抽象

- EnvoyFilter patch 配置语义复杂，难度大
- 通过自定义CRD简化配置，方便用户配置，降低使用门槛

```yaml
apiVersion: istio.alibabacloud.com/v1beta1
kind: ASMLocalRateLimiter
metadata:
  name: for-api-test
  namespace: default
spec:
  workloadSelector:
    labels:
      app: istio-ingressgateway
  isGateway: true
  configs:
    - match:
        vhost:
          name: "www.example1.com"    ## 如果gateway中配置了多个host域名，填最后一个即可。
          port: 80
          route:
            name_match: "test1"    ##VirtualService路由配置中对应route的name,若VirtualService路由配置下没有对应name的路由，则不生效。
      limit:
        fill_interval:
          seconds: 1
        quota: 10
    - match:
        vhost:
          name: "www.example2.com"
          port: 80
          route:
            name_match: "test1"
      limit:
        fill_interval:
          seconds: 1
        quota: 100
```

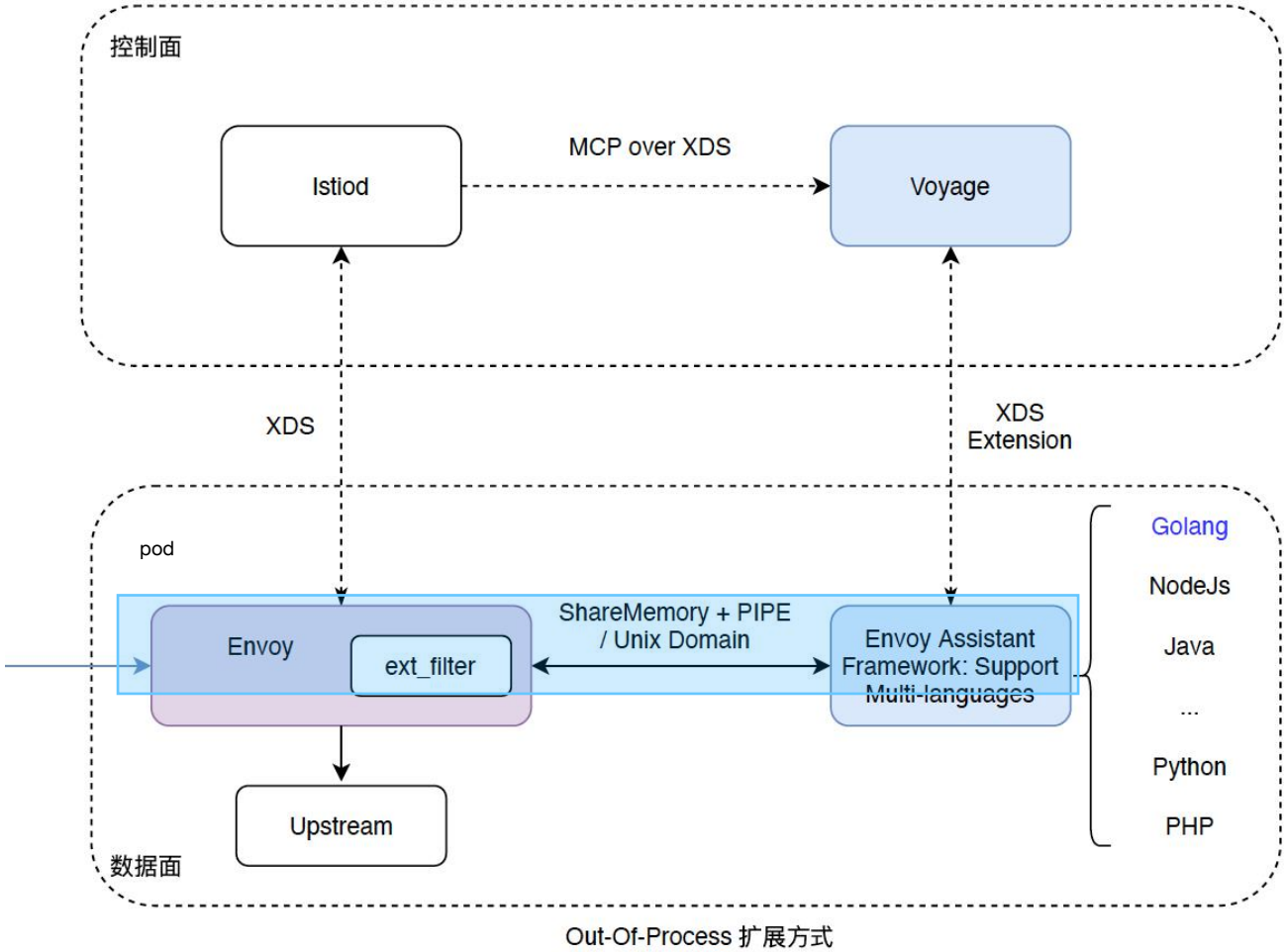
https://help.aliyun.com/document_detail/407660.html
https://istio.io/latest/docs/tasks/policy-enforcement/rate-limit/

自定义扩展方式之四：MOE



自定义扩展方式之五：Out Of Process 方式

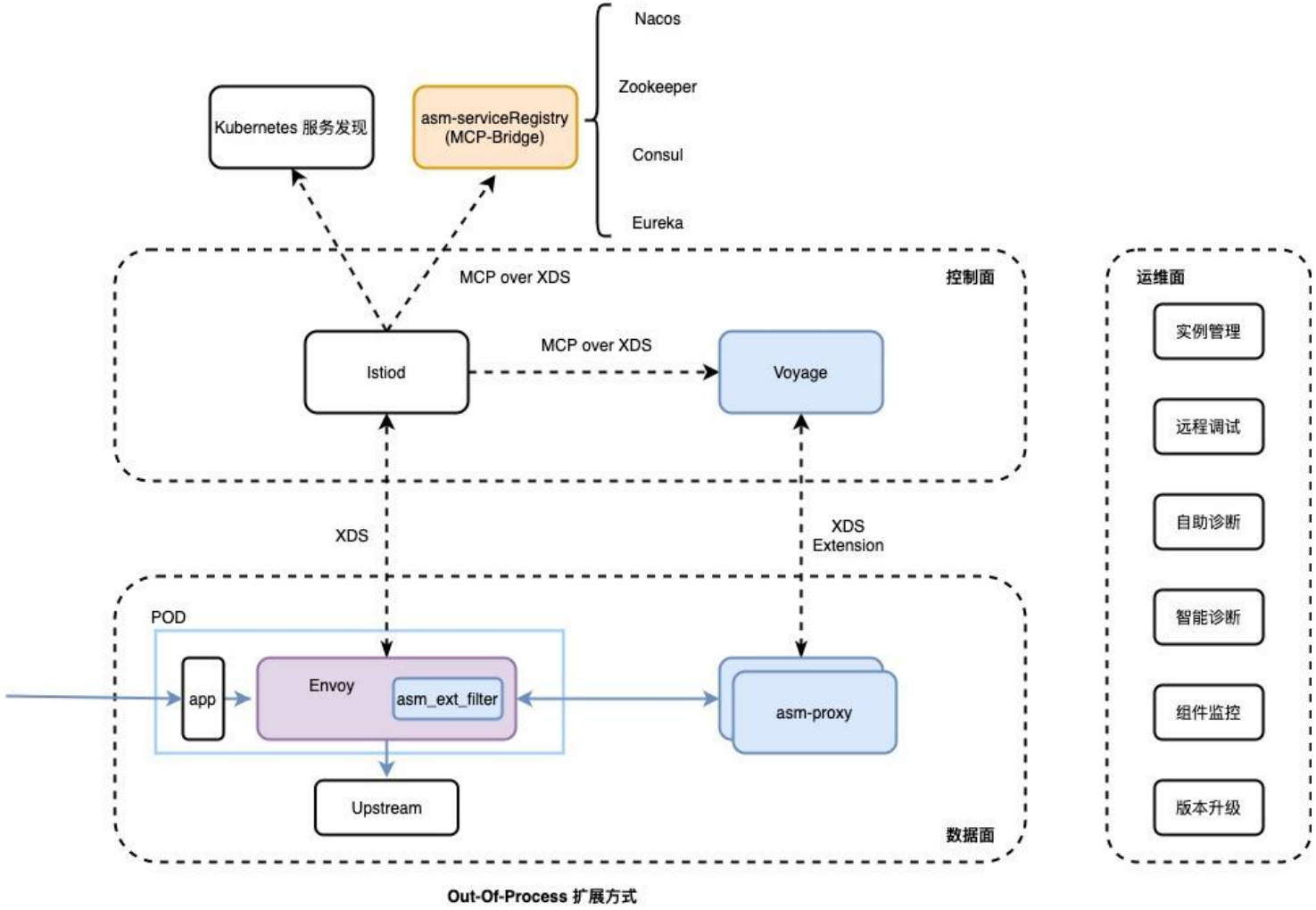
非侵入式扩展Istio和 Envoy



Out-Of-Process 扩展方式

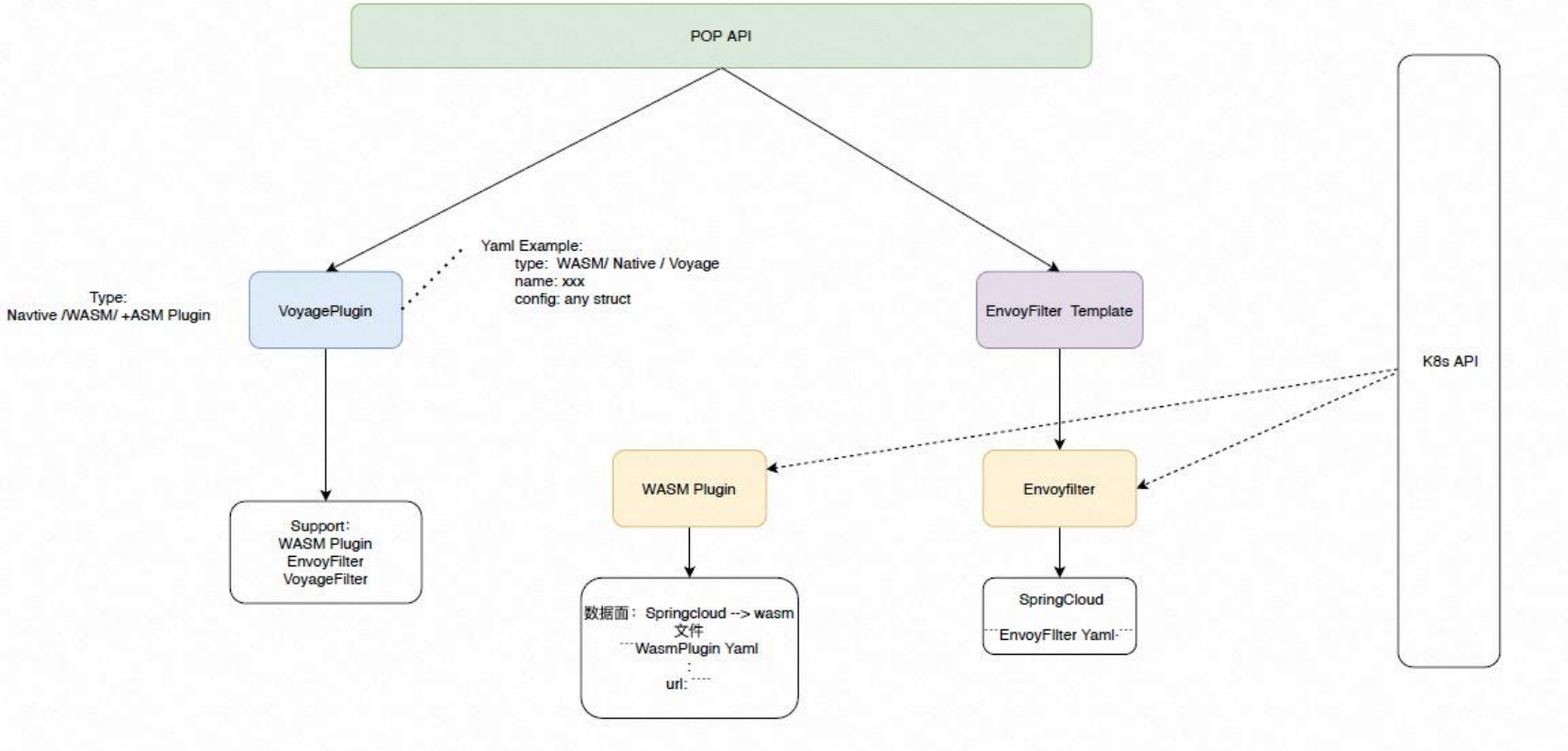
自定义扩展方式之五：Out Of Process 方式

扩展的asm-proxy可以有多种部署方式



Out-Of-Process 扩展方式

自定义扩展产品化: 以插件形式进行产品化封装



EnvoyFilter 原生API 使用，针对插件执行顺序、插件配置等具有一定的复杂度，不太适合直接面向普通用户，生产环境使用需要有封装过的产品能力

开发/运维人员使用最佳实践

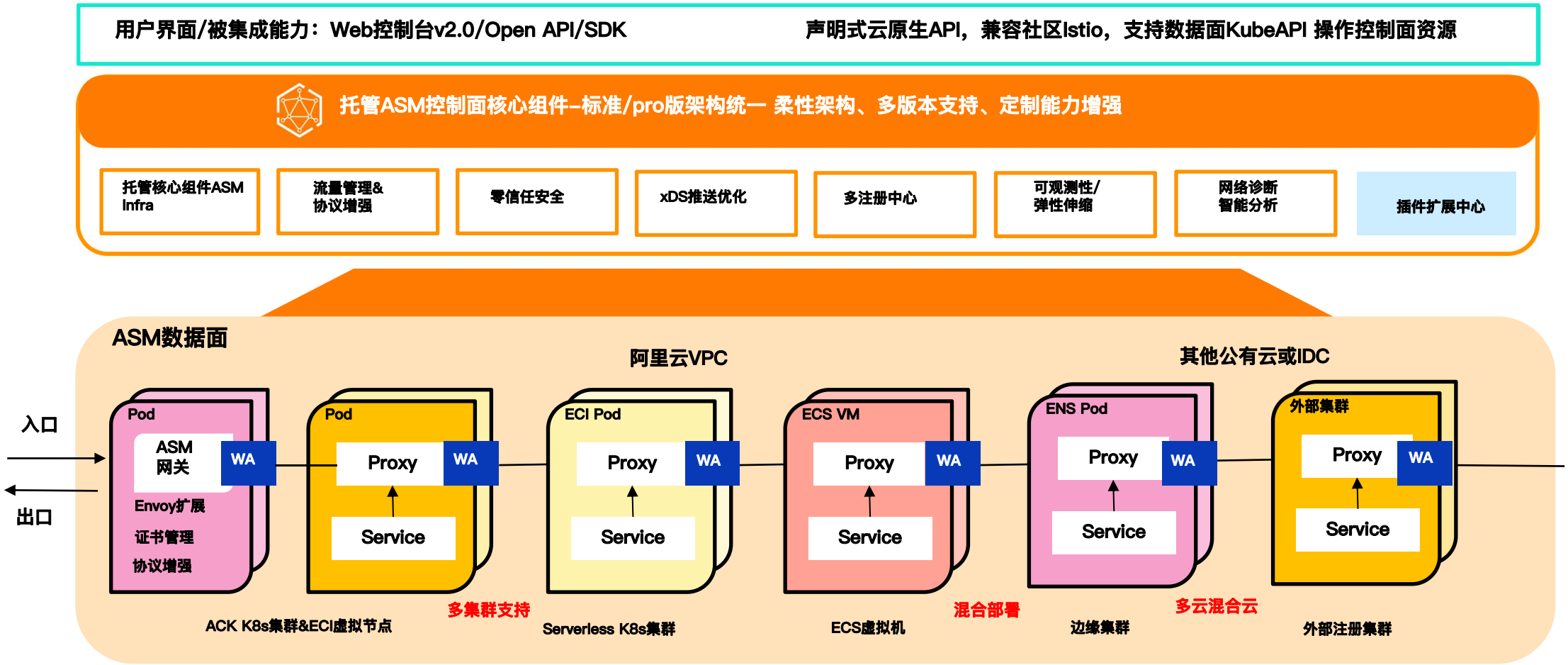
适合自己的才是最好的

扩展方式	Native C++	Lua	WASM	Out Of Process方式 (Envoy Assistant Framework)
性能	优	较优	一般	较优
开发效率	需要有一定的c++ 技术栈储备，是否有c++ 开发人员是关键	优，上手很快	较优（支持多语言）	框架有学习成本，较优，支持多语言
成熟度	优	优	一般	仅阿里云内部，还未开放
支持热加载	否，需重新构建镜像	是	是	否，需重新构建镜像（集中式部署方式除外）
社区	活跃（需具备c++ 知识储备）	一般	活跃	仅阿里云内部，还未开放

- 对性能要求不苛刻的场景小功能可以使用Lua 、Wasm 插件方式
- 对性能有一定要求，且逻辑较复杂的场景建议使用C++ Native Filter 或者采用Out Of Process方式

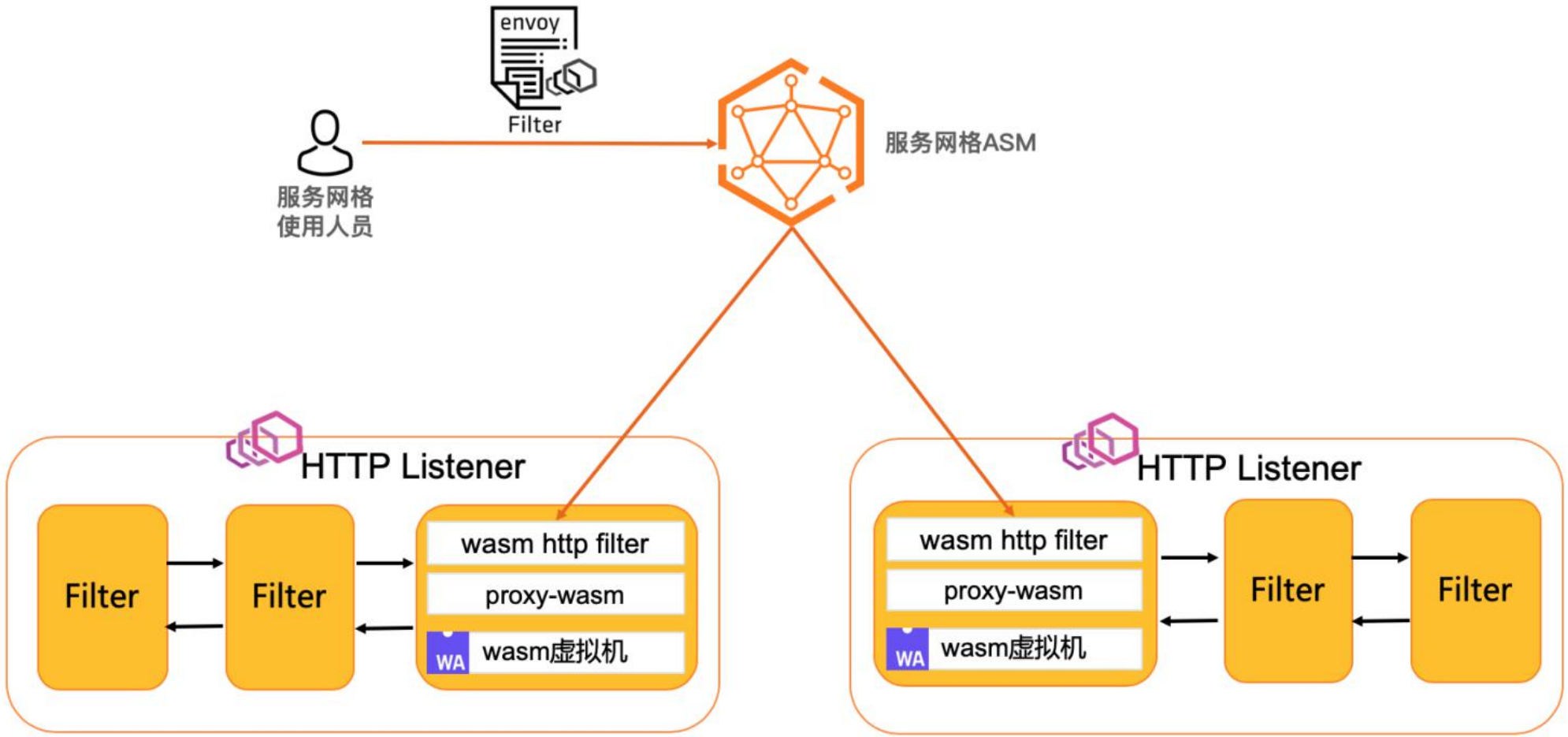
使用阿里云服务网格构筑企业级能力

SERVICE MESH SUMMIT 2022 服务网格峰会



https://www.aliyun.com/product/servicemesh

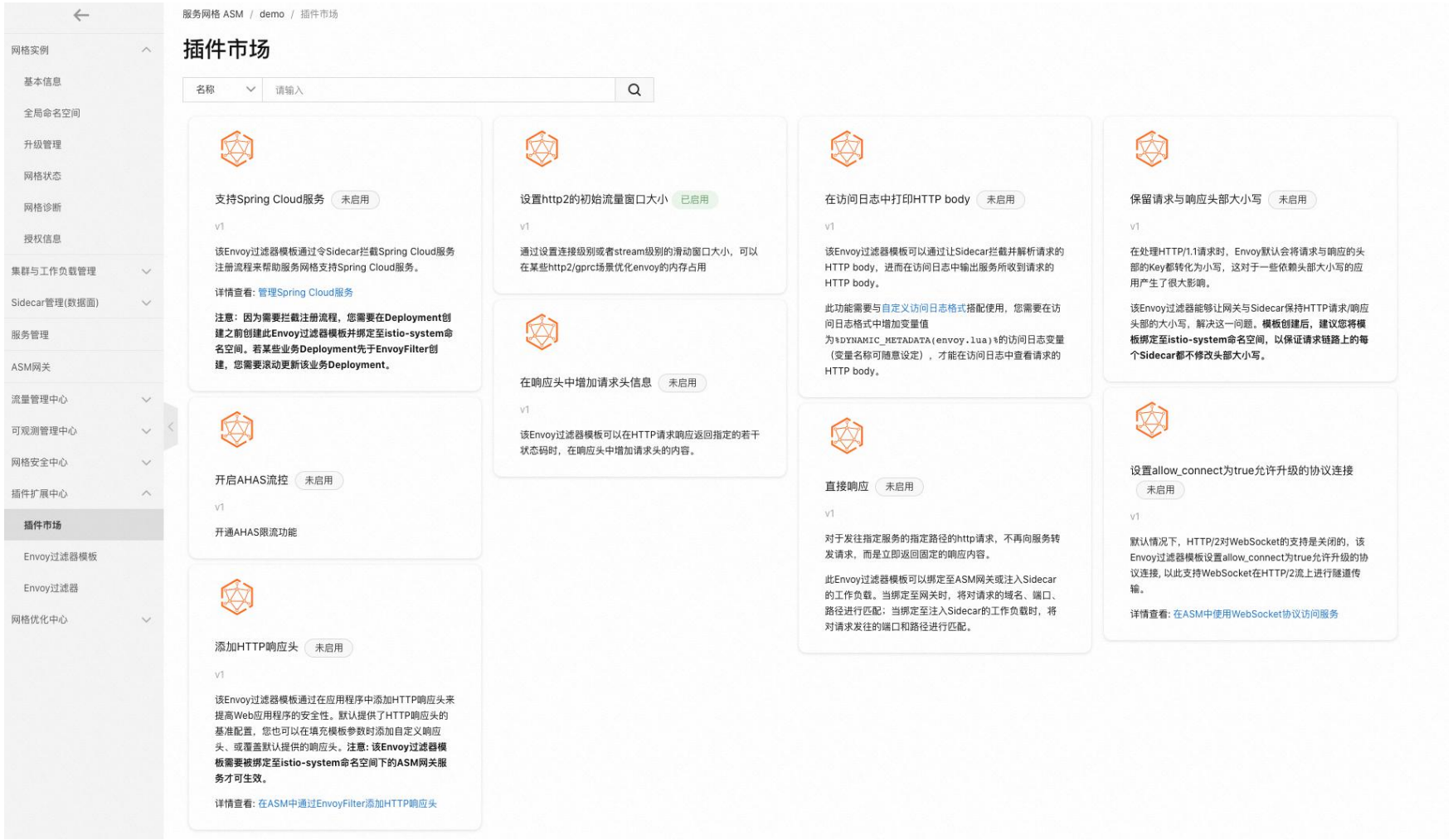
使用WASM 进行扩展ASM 能力



https://help.aliyun.com/document_detail/176520.html

ASM 插件市场

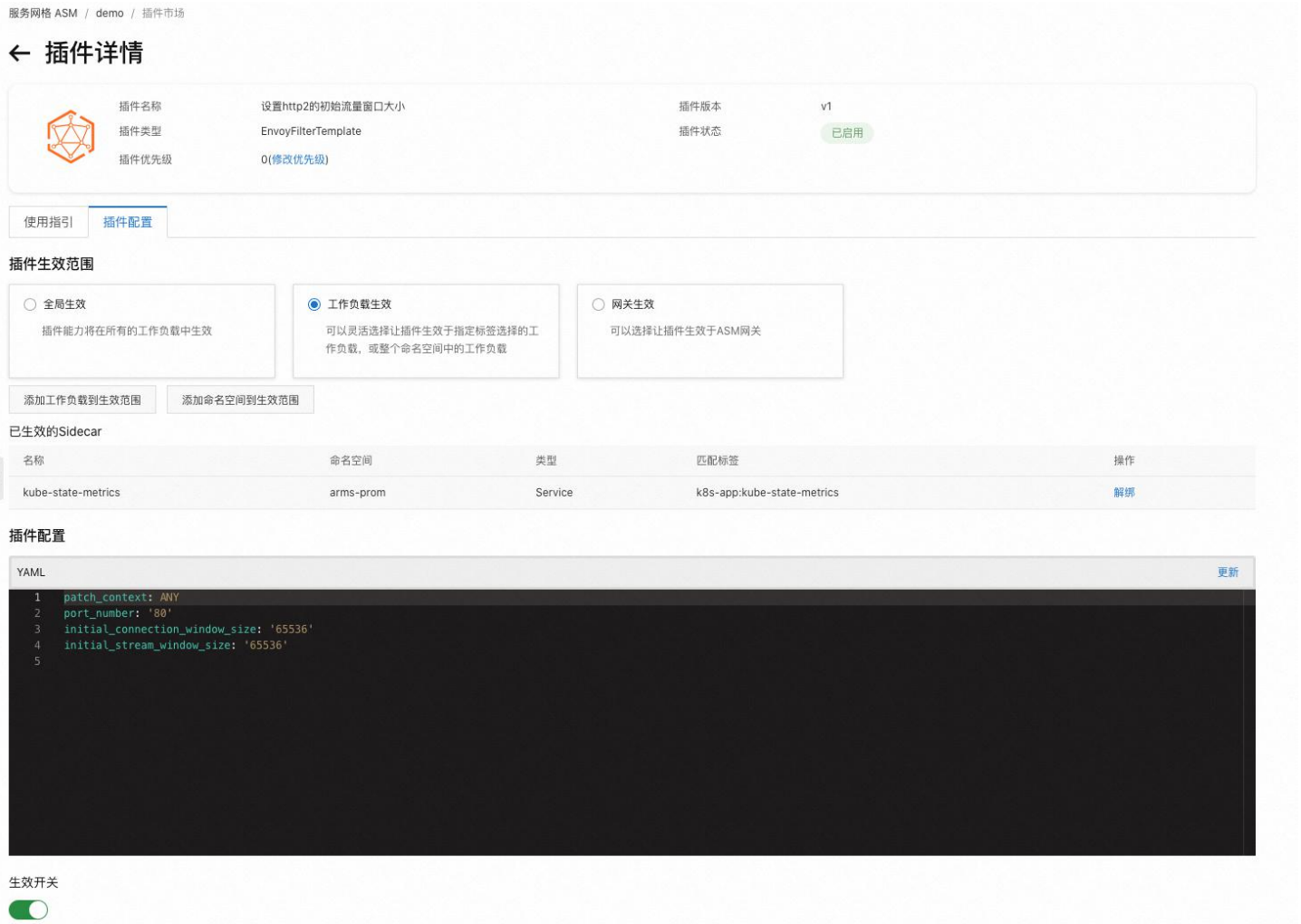
SERVICE MESH SUMMIT 2022
服务网格峰会



https://servicemesh.console.aliyun.com/

ASM 插件市场

SERVICE MESH SUMMIT 2022 服务网格峰会



ASM 网格诊断

网格诊断

命名空间

诊断时间：2022年9月20日 09:59:00 运行诊断成功 运行

序号	检查项	结果	详情
6	网关选择器检查	找不到与网关选择器匹配的工作负载	在命名空间default中找不到网关simple-springboot-gateway 选择器匹配的工作负载。请首先检查是否以"istio: 网关名称"或"app: istio-网关名称"对网关工作负载进行匹配，再确认指定的网关工作负载是否存在。
28	默认路由检查	建议为每个虚拟服务的路由规则提供默认路由	虚拟服务的http路由配置建议：命名空间default中的bookinfo最好提供不包含任何match字段的默认路由，以保证所有流量都能够找到转发目标
28	默认路由检查	建议为每个虚拟服务的路由规则提供默认路由	虚拟服务的http路由配置建议：命名空间default中的springboot-istio-client-vs最好提供不包含任何match字段的默认路由，以保证所有流量都能够找到转发目标
25	网关规则引用检查	虚拟服务引用的网关规则不存在	命名空间default中虚拟服务bookinfo指定的网关规则bookinfo-gateway不存在。 请检查您的网关名称或创建相应的网关
32	虚拟服务路由多端口检查	虚拟服务的路由目标需要指定端口	虚拟服务springboot-istio-server-vs路由到公开多个端口18080,18888的服务spring-boot-istio-server。 需要在目标中指定端口以消除歧义
31	虚拟服务路由目标检查	没有找到虚拟服务的路由目标	未能找到在虚拟服务springboot-istio-server-vs的路由目标中引用的服务spring-boot-istio-server; subset: $userdefinelabel1。建议您为服务所在命名空间开启Sidecar自动注入，再检查该服务是否存在。
1	数据面组件版本检查	通过	
2	服务端口检查	通过	
3	app及version标签检查	通过	
4	网关规则端口冲突检查	通过	
5	网关规则TLS证书冲突检查	通过	
7	网关引用的证书对应的Secret格式是否正确	通过	
8	网关引用的证书对应的Secret是否存在	通过	
9	目标规则的命名空间检查	通过	
10	网关Pod unpriviledged模式端口检查	通过	
11	数据面的命名空间注入标签值与控制面是否同步	通过	
12	控制面是否存在非ASM提供的EnvoyFilters	通过	
13	授权策略中的命名空间引用检查	通过	
14	工作负载引用检查	通过	
15	字段弃用检查	通过	
16	忽略字段检查	通过	

总结

- 基于Istio&Envoy 提供的可扩展能力，可以满足用户不同业务场景需求。
- 开发/运维人员可以根据实际情况选择适合自身团队的扩展方式。
- 使用和运维Istio 具有一定门槛和复杂度，需要进行产品能力封装，推荐使用阿里云产品ASM

云原生社区
Cloud Native Community

活动由云原生社区主办

SERVICE MESH
SUMMIT 2022
服务网格峰会

感谢观看