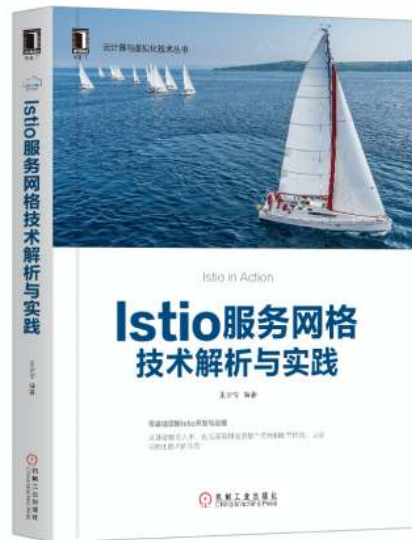


Extending service mesh capabilities using a streamlined way based on WASM and ORAS

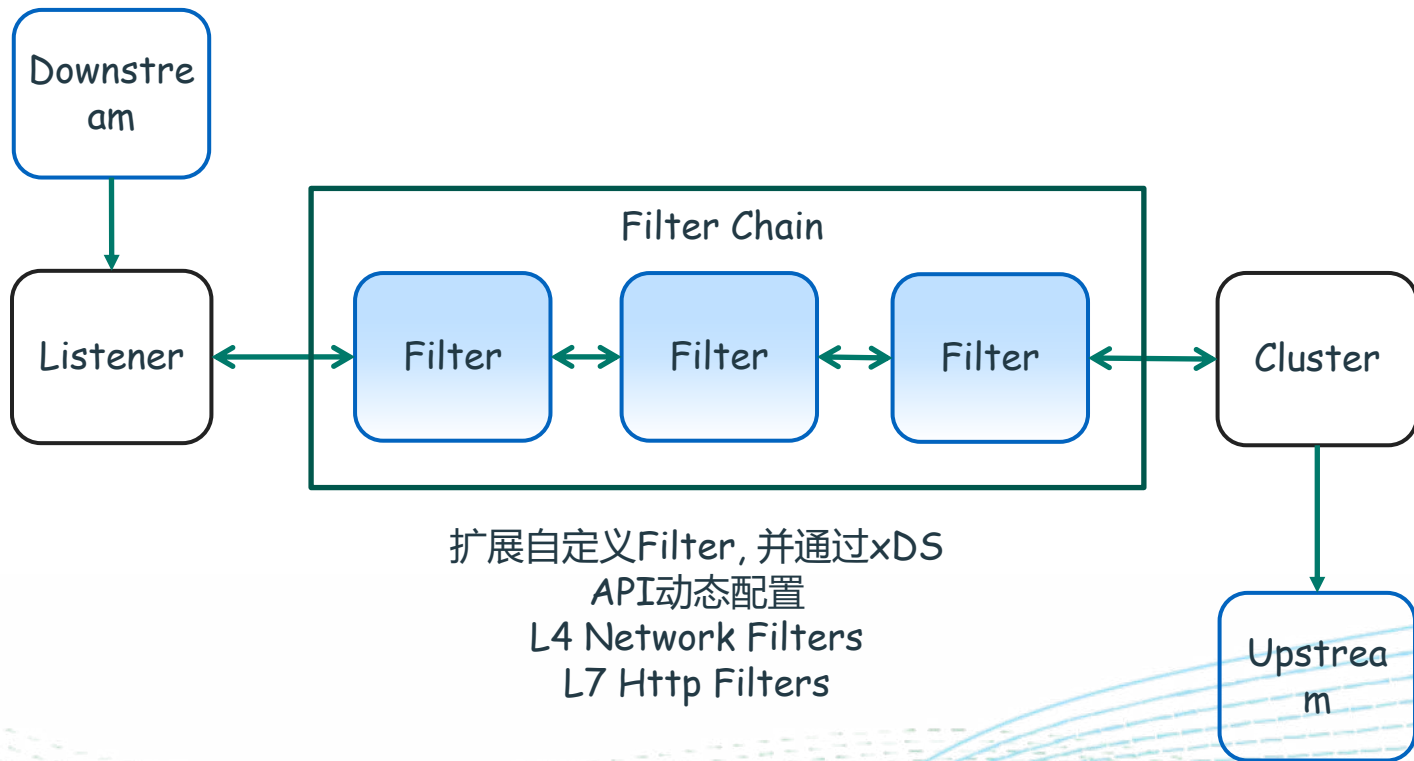
王夕宁 | 阿里云服务网格ASM



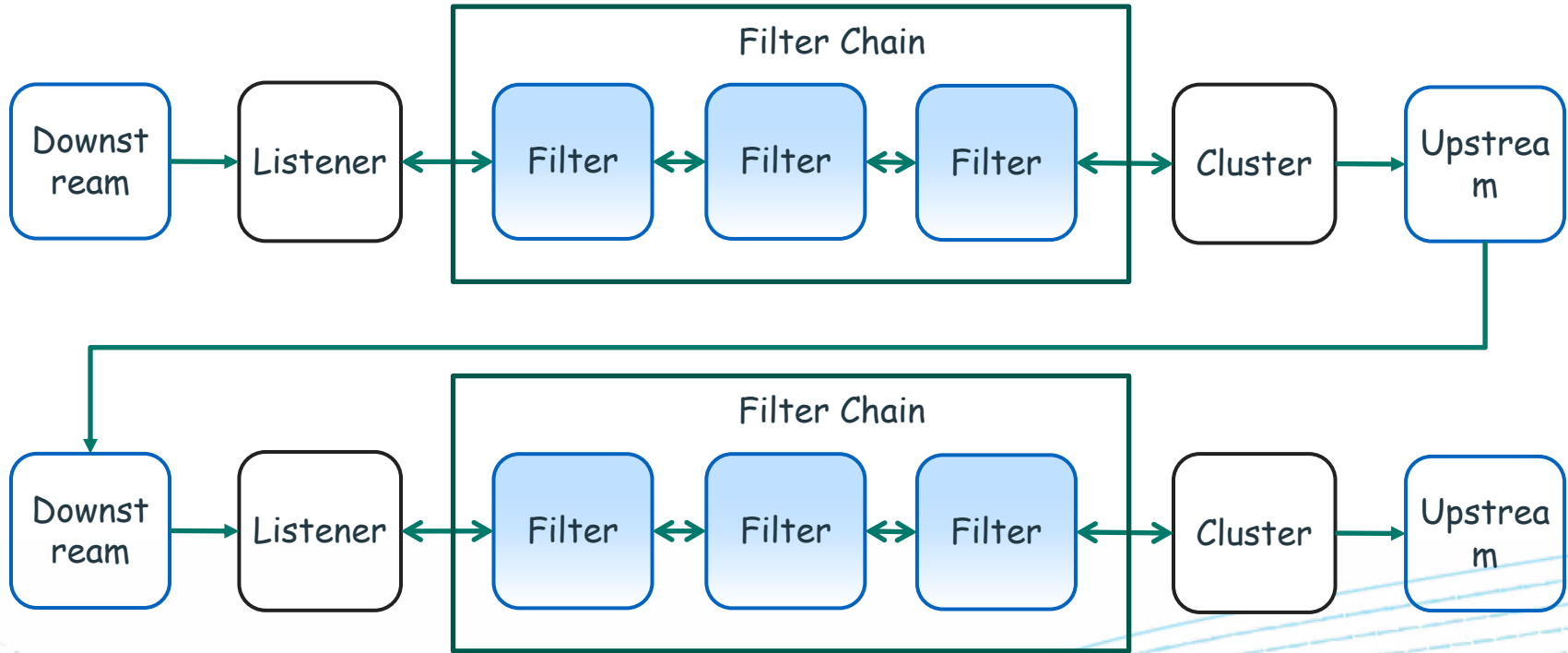
#IstioCon



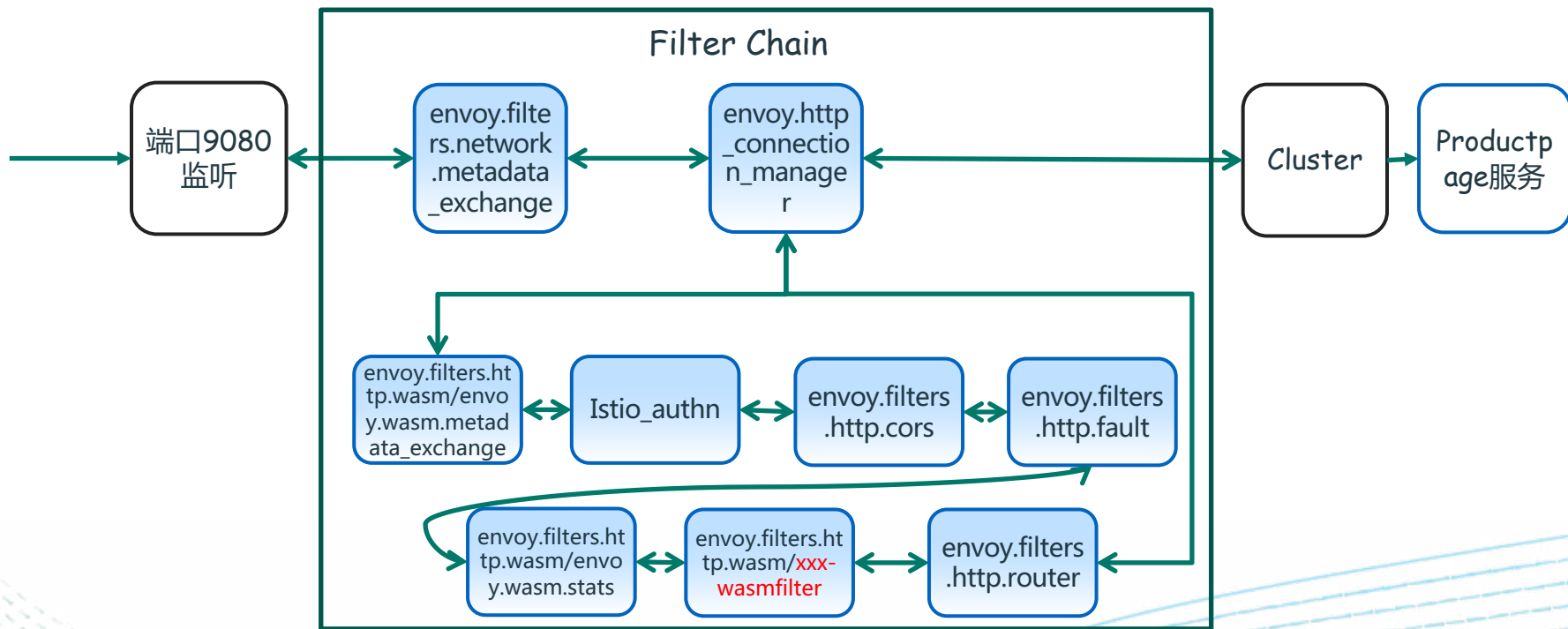
Envoy's Filter Chain



Listener & Filters before outbound services



实际示例中用到的Envoy Filters



[kubectl exec -it \[productpage-xxx\] -c istio-proxy curl localhost:15000/config_dump](#)



添加新Filter的方式

- Built-in Filter & Community Provided:

- https://www.envoyproxy.io/docs/envoy/latest/configuration/http/http_filters/http_filters
-

- 自定义开发:

- 静态预编译:
 - 将其他过滤器集成到Envoy的源代码中, 并编译新的Envoy版本。
 - 这种方法的缺点是您需要维护Envoy版本, 并不断使其与官方发行版保持同步。
 - 此外, 由于Envoy是用C++实现的, 因此新开发的过滤器也必须用C++实现。
- 动态运行时加载:
 - 在运行时将新的过滤器动态加载到Envoy代理中。
 - 简化了扩展Envoy的过程, 这种解决方案通常使用WebAssembly (WASM) 的新技术, 它是一种有效的可移植二进制指令格式, 提供了可嵌入和隔离的执行环境。



使用Wasm for Proxy

● Pros

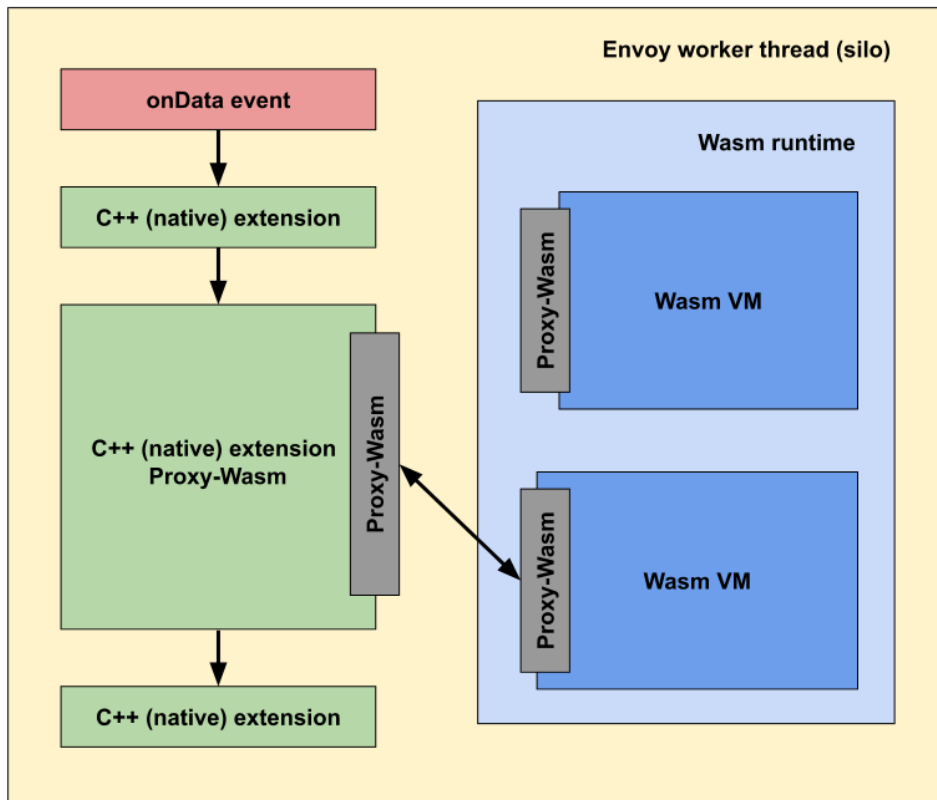
- 敏捷性：过滤器可以动态加载到正在运行的Envoy进程中，而无需停止或重新编译。
- 可维护性：不必更改Envoy自身基础代码库即可扩展其功能。
- 多样性：可以将流行的编程语言（例如C/C++和Rust）编译为WASM，因此开发人员可以选择实现过滤器的编程语言。
- 可靠性和隔离性：过滤器会被部署到VM沙箱中，因此与Envoy进程本身是隔离的；即使当WASM Filter出现问题导致崩溃时，它也不会影响Envoy进程。
- 安全性：过滤器通过预定义API与Envoy代理进行通信，因此它们可以访问并只能修改有限数量的连接或请求属性。

● Cons

- 性能约为C++编写的原生静态编译的Filter的70%；
- 由于需要启动一个或多个WASM虚拟机，因此会消耗一定的内存使用量；
- The WebAssembly ecosystem is still young;



Wasm in Envoy Proxy



- Wasm动态加载
- 一致性校验:
 - <https://github.com/proxy-wasm/spec>
- 内置的Wasm runtime
 - ~20MB for WAVM
 - ~10MB for V8
- 事件驱动模型
- 兼容native filter调用方式



Example Wasm filter configuration

- 下发到Envoy Proxy侧的配置

```
name: envoy.filters.http.wasm
typed_config:
  "@type": type.googleapis.com/envoy.extensions.filters.http.wasm.v3.Wasm
  config:
    config:
      name: "my_plugin"
    vm_config:
      runtime: "envoy.wasm.runtime.v8"
      code:
        local:
          filename: "/etc/envoy_filter_http_wasm_example.wasm"
      allow_precompiled: true
```



OCI Registry As Storage



- OCI Artifacts项目的参考实现, 可显著简化OCI注册库中任意内容的存储;
- 可以使用ORAS API/SDK Library来构建自定义工具,
 - 将WebAssembly模块推入到OCI注册库中;
 - 或者从OCI注册库中拉取WebAssembly模块;
- oras cli类似于docker cli

```
Usage:
  oras [command]

Available Commands:
  help      Help about any command
  login     Log in to a remote registry
  logout    Log out from a remote registry
  pull      Pull files from remote registry
  push      Push files to remote registry
  version   Show the oras version information

Flags:
  -h, --help  help for oras

Use "oras [command] --help" for more information about a command.
```



在ACR EE中使用ORAS CLI

- 阿里云容器镜像服务企业版ACR EE作为企业级云原生应用制品管理平台，提供容器镜像、Helm Chart以及符合OCI规范的制品的生命周期管理；

容器镜像服务 / 实例列表 / 镜像仓库

← acree-1

所在地域 华东1（杭州）

实例规格 基础版

运行状态 ✓ 运行中

概览

仓库管理

镜像仓库

命名空间

代码源 NEW

创建镜像仓库		全部命名空间	仓库名称			
仓库名称	命名空间	仓库状态	仓库类型	仓库地址	创建时间	操作
asm-test	asm	✓ 正常	公开	...	2021-02-21 17:43:20	管理 删除

- `oras login --username=<登录账号> acree-1-registry.cn-hangzhou.cr.aliyuncs.com`



通过oras push命令推送

- oras push acree-1-registry.cn-hangzhou.cr.aliyuncs.com/asm/asm-test:v0.1 --manifest-config runtime-config.json:application/vnd.module.wasm.config.v1+json example-filter.wasm:application/vnd.module.wasm.content.layer.v1+wasm

○ Wasm Artifact镜像规范参考

■ <https://github.com/solo-io/wasm/blob/master/spec/README.md>

■ <https://istio.io/latest/blog/2020/wasmhub-istio/>

- Wasm filter被推送到ACR EE注册库中

容器镜像服务 / 实例列表 / 镜像仓库 / 镜像版本

← asm-test

华东1（杭州） | 公开 | 本地仓库 | ✓ 正常

部署应用

基本信息

构建

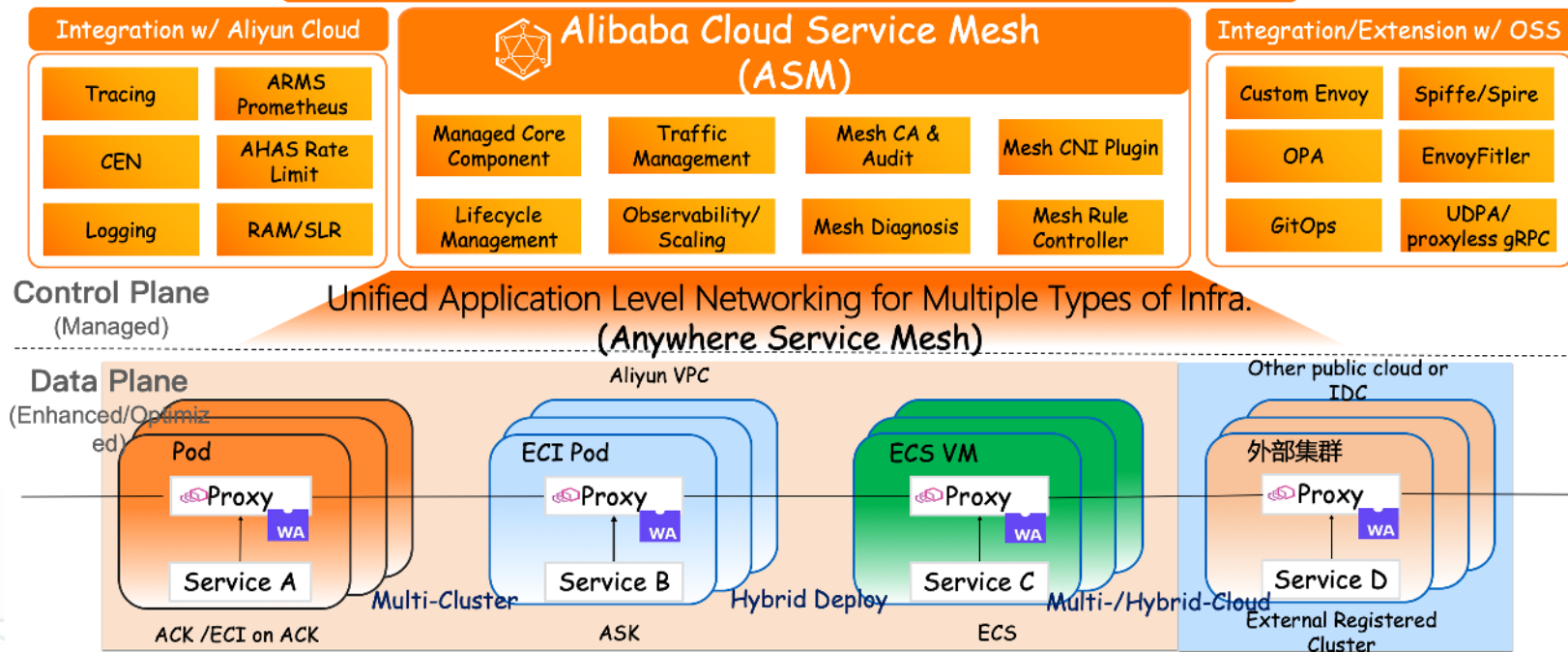
触发器

镜像版本

版本	镜像ID ?	状态	Digest ?	镜像大小 ?	最近推送时间	操作
v0.1		✓ 正常 ⊖ 可覆盖	cd4e5a69ce228e5399dd6c 4ce9ea6b2bd180e6986971 f2759a015c3496721355	-	2021-02-21 22:45:57	安全扫描 层信息 删除

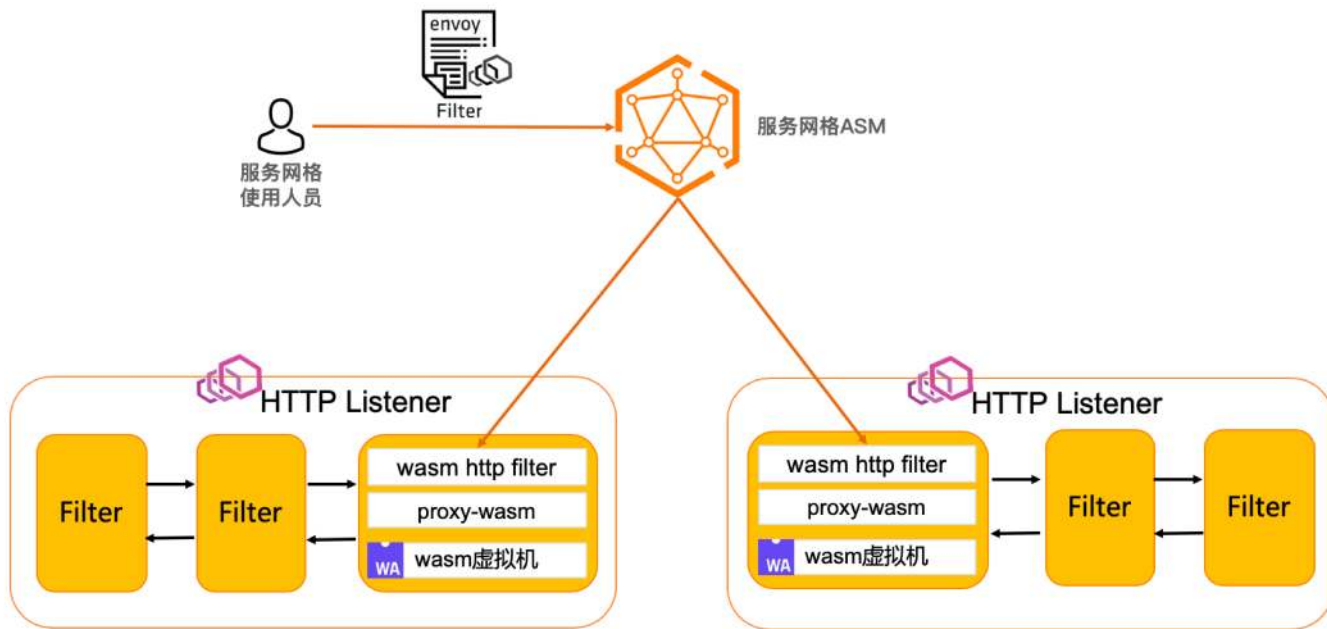
阿里云服务网格ASM架构

To be integrated: Web Console/Open API/SDK Cloud Native API, Compatible w/ Istio



在阿里云服务网格ASM中使用WASM

- 进行统一的代理扩展插件的生命周期管理



在ASM中启用wasm部署功能

- `aliyun servicemesh UpdateMeshFeature -- ServiceMeshId=xxxxxx --WebAssemblyFilterEnabled=true`
 - 部署一个DaemonSet(asmwasm-controller)到K8s集群中
 - asmwasm-controller监听一个configmap, 该configmap存放要拉取的wasm filter的地址, 例如: `acree-1-registry.cn-hangzhou.cr.aliyuncs.com/asm/sample:v0.1`
 - 如果需要授权认证, 该asmwasm-controller会根据定义的pullSecret值获得相应的secret值;
 - 然后,调用oras API从注册库中动态拉取wasm filter;
 - 该asmwasm-controller使用HostPath方式挂载volume, 所以拉取的wasm filter会落盘到对应的节点上;

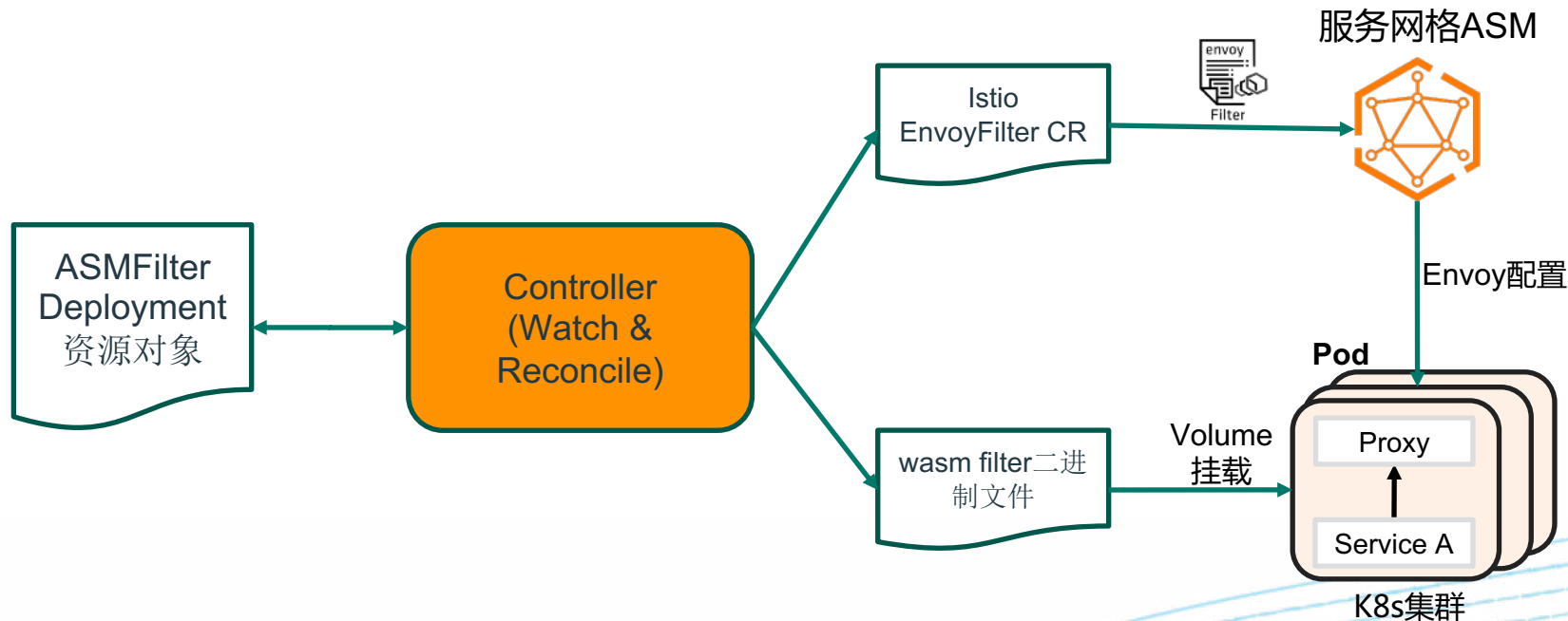


创建私钥仓库登录Secret

- 获取私有仓库登录信息之后, 按照如下命令创建Secret
 - `kubectl create secret generic asmwasm-cache -n istio-system --from-file=.dockerconfigjson=myconfig.json --type=kubernetes.io/dockerconfigjson`



在ASM中WASM filter的生命周期管理



ASMFiterDeployment CR示例

```
1  apiVersion: istio.alibabacloud.com/v1beta1
2  kind: ASMFiterDeployment
3  metadata:
4    name: details-v1-wasmfiltersample
5  spec:
6    deployment:
7      kind: Deployment
8      labels:
9        app: details
10       version: v1
11    filter:
12      parameters: '{"name":"hello","value":"details"}'
13      image: 'acree-1-registry.cn-hangzhou.cr.aliyuncs.com/asm/asm-test:v0.1'
14      imagePullOptions:
15        pullSecret: 'asmwasm-cache'
16      rootID: 'my_root_id'
17      id: 'details-v1-wasmfiltersample.default'
18
```



生成的Istio Envoy Filter资源(1)

```
apiVersion: networking.istio.io/v1alpha3
kind: EnvoyFilter
metadata:
  ....
spec:
  configPatches:
    - applyTo: HTTP_FILTER
      match: ....
      patch: ....
  workloadSelector:
    labels:
      app: productpage
      version: v1
```



```
match:
  context: SIDECAR_INBOUND
  listener:
    filterChain:
      filter:
        name: envoy.http_connection_manager
        subFilter:
          name: envoy.router
```



生成的Istio Envoy Filter资源(2)

```
apiVersion: networking.istio.io/v1alpha3
kind: EnvoyFilter
metadata:
  ....
spec:
  configPatches:
    - applyTo: HTTP_FILTER
      match: ....
      patch: ....
  workloadSelector:
    labels:
      app: productpage
      version: v1
```



```
patch:
  operation: INSERT_BEFORE
  value:
    name: envoy.filters.http.wasm
    typed_config:
      '@type': type.googleapis.com/udpa.type.v1.TypedStruct
      type_url: type.googleapis.com/envoy.extensions.filters.http.wasm.v3.Wasm
      value:
        config:
          name: details-v1-wasmfiltersample.default
          rootId: my_root_id
          vmConfig:
            allow_precompiled: true
            code:
              local:
                filename: >-
                /var/local/lib/wasm-filters/c49d54bb6751531e89c110054ec74ab5
            configuration:
              '@type': type.googleapis.com/google.protobuf.StringValue
              value: '{"name":"hello","value":"productpage"}'
            runtime: envoy.wasm.runtime.v8
            vmId: details-v1-wasmfiltersample.default
```



更新后的Deployment

- 以hostpath方式挂载wasm filter文件到Proxy容器

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  ....
spec:
  ....
  template:
    metadata:
      annotations:
        sidecar.istio.io/userVolume: '[{"name":"wasmfilters-
dir","hostPath":{"path":"/var/local/lib/wasm-filters"}}]'
        sidecar.istio.io/userVolumeMount:
' [{"mountPath":"/var/local/lib/wasm-filters","name":"wasmfilters-dir"} ]'
```



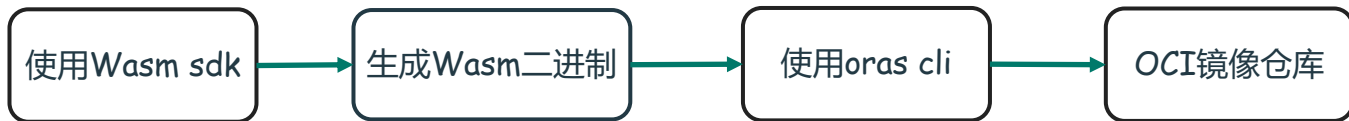
执行结果, wasm filter生效

```
d:/ $ curl -v details:9080/details/123
* Trying 172.21.9.191...
* TCP_NODELAY set
* Connected to details (172.21.9.191) port 9080 (#0)
> GET /details/123 HTTP/1.1
> Host: details:9080
> User-Agent: curl/7.58.0
> Accept: */*
>
< HTTP/1.1 200 OK
< content-type: application/json
< server: istio-envoy
< date: Mon, 22 Feb 2021 14:49:01 GMT
< content-length: 180
< x-envoy-upstream-service-time: 1
< resp-header-demo: added by our filter
<
* Connection #0 to host details left intact
{"id":123,"author":"William Shakespeare","year":1595,"type":"paperback","pages":200,"publisher":"Pub
language":"English","ISBN-10":"1234567890","ISBN-13":"123-1234567890"}istio-proxy@productpage-v1-658b
```



Summary & Tips

● 开发阶段



● 部署运行阶段



1.可以登录到proxy container进行查看 wasm filter是否挂载成功

2.调整wasm log level : curl -X POST

<http://localhost:15000/logging?wasm=debug>



Thank you!

#IstioCon

