HUAWEI

扫码添加小助手，发送“Istio”加群

HUAWEI

CloudNativeLives

istio入门级实训

Gateway 设计与实现

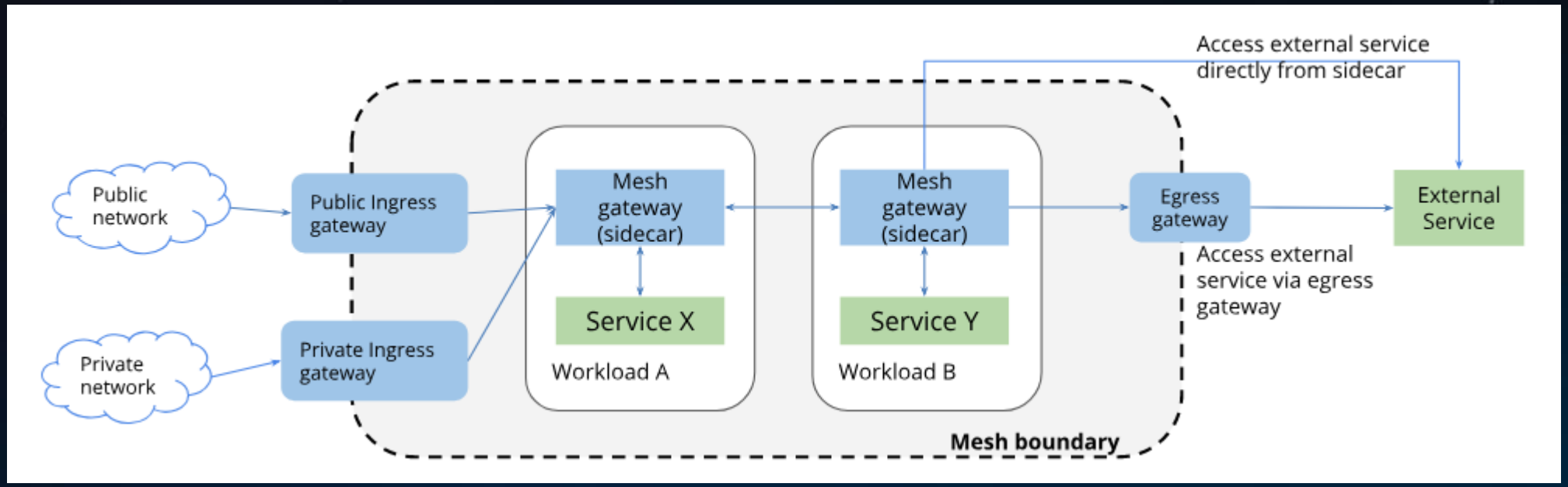
华为云容器团队核心架构师 & CNCF社区主要贡献者倾力打造

目录

- Gateway简介
- Gateway vs Kubernetes Ingress
- Gateway原理及实现
- Gateway demo演示

Gateway简介

在Istio中，Gateway控制着网格边缘的服务暴露。



Gateway简介

- Gateway也可以看作网格的负载均衡器， 提供以下功能：

1）L4-L6的负载均衡
2）对外的mTLS

Istio服务网格中，Gateway可以部署任意多个，可以共用一个，也可以每个租户、namespace单独隔离。

Gateway简介

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: bookinfo-gateway
spec:
  servers:
  - port:
      number: 443
      name: https
      protocol: HTTPS
    hosts:
    - bookinfo.com
    tls:
      mode: SIMPLE
      serverCertificate: /tmp/tls.crt
      privateKey: /tmp/tls.key
```

Gateway 监听的端口及协议

Gateway允许外部访问host：bookinfo.com的https流量进去网格内

TLS 设置

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: bookinfo
spec:
  hosts:
    - bookinfo.com
  gateways:
  - bookinfo-gateway # <---- bind to gateway
  http:
  - match:
    - uri:
        prefix: /reviews
    route:
    ...
```

VirtualService定义Gateway L7路由， 为访问bookinfo.com的https流量，提供路由匹配转发策略

Gateway简介

Gateway根据流入流出方向分为ingress gateway和egress gateway

```
root@szvp000201060:~# kubectl get pod -n istio-system |grep gateway
istio-egressgateway-67568748-m4dmk                  1/1     Running     0          4d19h
istio-ingressgateway-588d4dcb9f-w45zx               1/1     Running     0          4d19h
```

- Ingress gateway:
  控制外部服务访问网格内服务，配合VirtualService
- Egress gateway:
  控制网格内服务访问外部服务, 配合DestinationRule ServiceEntry使用

Gateway vs Kubernetes Ingress

HUAWEI

Kubernetes Ingress集群边缘负载均衡， 提供集群内部服务的访问入口，仅支持L7负载均衡，功能单一

Istio 1.0以前，利用Kubernetes Ingress实现网格内服务暴露。但是Ingress无法实现很多功能：

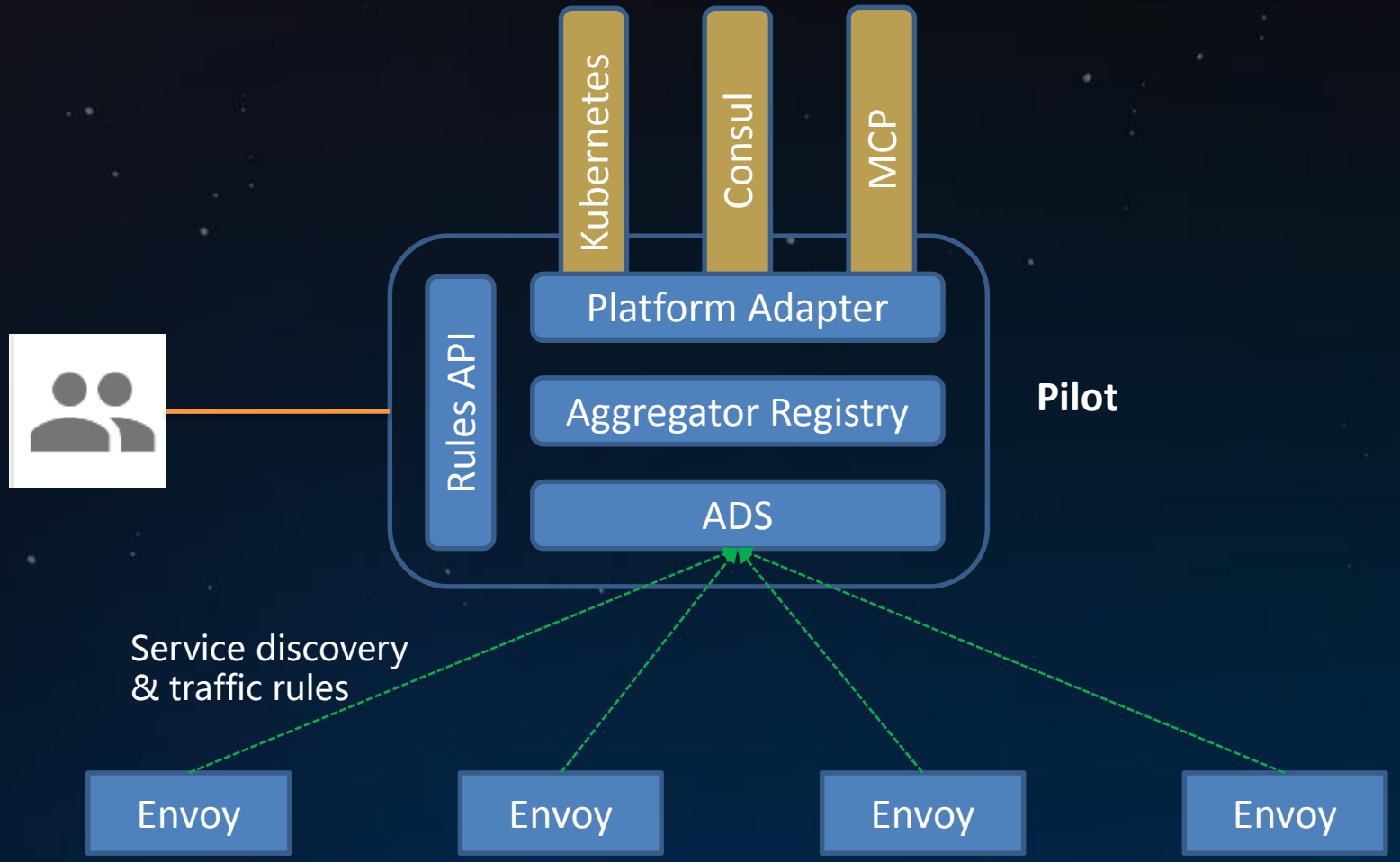
1) L4-L6负载均衡
2) 对外mTLS
3) SNI的支持
4) 其他istio中已经实现的内部网络功能： Fault Injection，Traffic Shifting，Circuit Breaking， Mirroring

Gateway vs Kubernetes Ingress

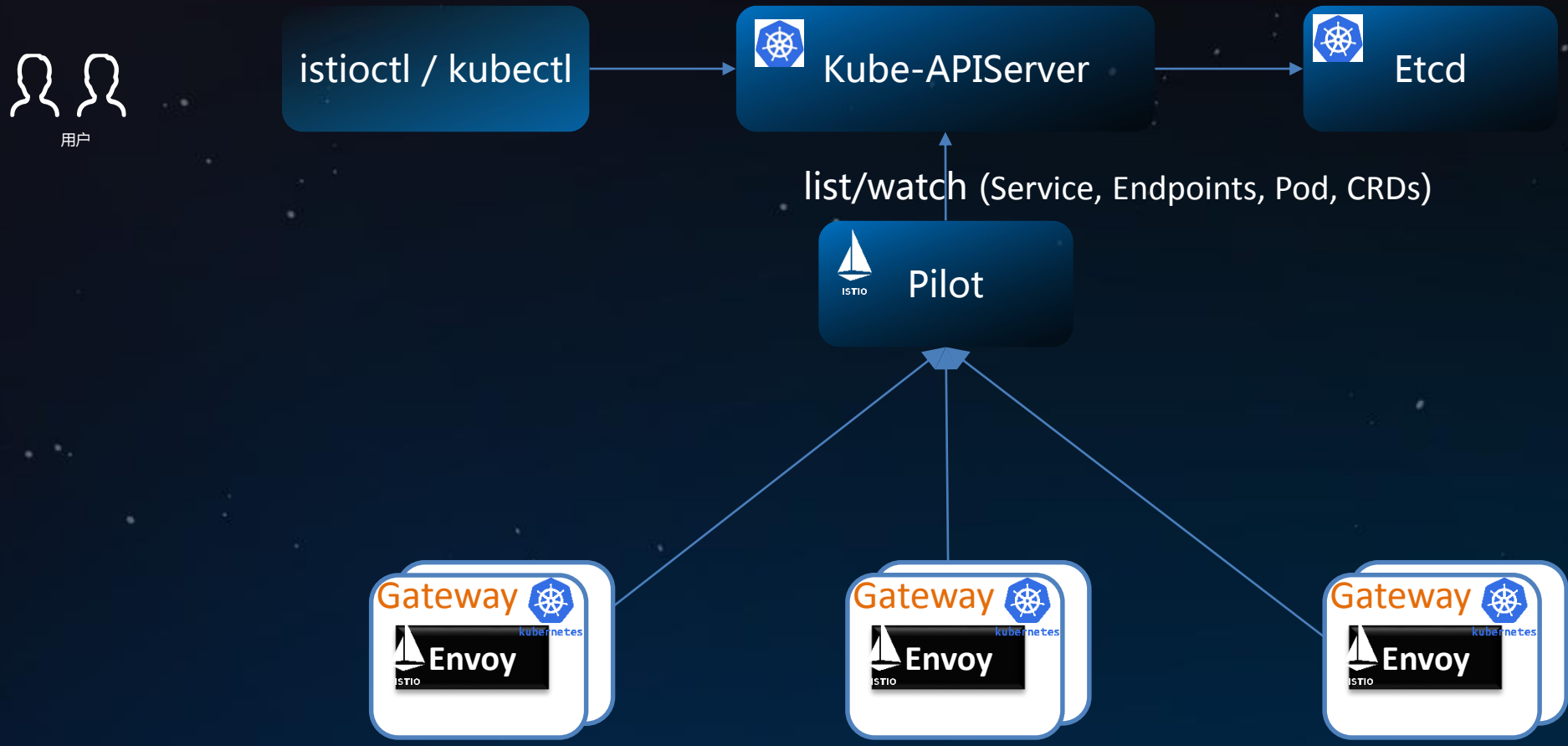
为了解决这些这些问题，Istio在1.0版本设计了新的v1alpha3 API。

- Gateway允许管理员指定L4-L6的设置：端口及TLS设置。
- 对于ingress 的L7设置，Istio允许将VirtualService与Gateway绑定起来。
- 分离的好处：用户可以像使用传统的负载均衡设备一样管理进入网格内部的流量，绑定虚拟IP到虚拟服务器上 。便于传统技术用户无缝迁移到微服务。

Gateway原理及实现



Gateway原理及实现



Gateway原理及实现

Gateway 与 普通sidecar均是使用Envoy作为proxy实行流量控制。Pilot为不同类型的proxy生成相应的配置，Gateway的类型为router，sidecar的类型为sidecar。

Ingress Gateway 启动参数：

```
root@szvp000201060:~# kubectl -n istio-system exec -ti istio-ingressgateway
-588d4dcb9f-2g296  sh
#  ps -efww
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 08:53 ?        00:00:00 /usr/local/bin/pilot-agen
t proxy router --log_output_level info --drainDuration 45s --parentShutdown
Duration 1m0s --connectTimeout 10s --serviceCluster istio-ingressgateway --
zipkinAddress zipkin:9411 --proxyAdminPort 15000 --controlPlaneAuthPolicy N
ONE --discoveryAddress istio-pilot:15010
root          18       1  0 08:53 ?        00:00:08 /usr/local/bin/envoy -c /
etc/istio/proxy/envoy-rev0.json --restart-epoch 0 --drain-time-s 45 --paren
t-shutdown-time-s 60 --service-cluster istio-ingressgateway --service-node
router~172.17.0.30~istio-ingressgateway-588d4dcb9f-2g296.istio-system~istio
-system.svc.cluster.local --max-obj-name-len 189 --allow-unknown-fields -l
warning --v2-config-only
```

Gateway原理及实现

HUAWEI

Sidecar启动参数：

```
root@szvp000201060:~# kubectl exec -ti sleep-6b868f6d46-smfkk -c istio-prox
y sh
$ ps -efww
UID          PID    PPID  C STIME TTY          TIME CMD
istio-p+       1       0  0 09:27 ?        00:00:00 /usr/local/bin/pilot-agen
t proxy sidecar --configPath /etc/istio/proxy --binaryPath /usr/local/bin/e
nvoy --serviceCluster sleep.default --drainDuration 45s --parentShutdownDur
ation 1m0s --discoveryAddress istio-pilot.istio-system:15010 --zipkinAddres
s zipkin.istio-system:9411 --connectTimeout 10s --proxyAdminPort 15000 --co
ntrolPlaneAuthPolicy NONE --statusPort 15020 --applicationPorts
istio-p+      18       1  0 09:27 ?        00:00:00 /usr/local/bin/envoy -c /
etc/istio/proxy/envoy-rev0.json --restart-epoch 0 --drain-time-s 45 --paren
t-shutdown-time-s 60 --service-cluster sleep.default --service-node sidecar
~172.17.0.7~sleep-6b868f6d46-smfkk.default~default.svc.cluster.local --max-
obj-name-len 189 --allow-unknown-fields -l warning --v2-config-only
```

Gateway原理及实现

Pilot如何得知proxy类型？

Envoy发现服务使用的是xDS协议，Envoy向server端pilot发起请求 **DiscoveryRequest** 时会携带自身信息node，node有一个ID标识，pilot会解析node标识获取proxy类型。

DiscoveryRequest

```
{
  "version_info": "...",
  "node": "{...}",
  "resource_names": [],
  "type_url": "...",
  "response_nonce": "...",
  "error_detail": "{...}"
}
```

Node

```
{
  "id": "...",
  "cluster": "...",
  "metadata": "{...}",
  "locality": "{...}",
  "build_version": "..."
}
```

Envoy的节点标识可以通过静态配置文件指定，也可以通过启动参数 --service-node指定

Gateway原理及实现

Gateway对象在Istio中是用CRD声明的，可通过

$ kubectl get crd gateways.networking.istio.io 验证

```yaml
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: gateways.networking.istio.io
  annotations:
    "helm.sh/hook": crd-install
    "helm.sh/hook-weight": "-5"
  labels:
    app: istio-pilot
    chart: istio
    heritage: Tiller
    release: istio
spec:
  group: networking.istio.io
  names:
    kind: Gateway
    plural: gateways
    singular: gateway
    categories:
    - istio-io
    - networking-istio-io
  scope: Namespaced
  version: v1alpha3
```

Gateway原理及实现

HUAWEI

Istio networking所有配置API定义：

https://github.com/istio/api/tree/master/networking/v1alpha3

```
type Gateway struct {
   // REQUIRED: A list of server specifications.
   Servers []*Server `protobuf:"bytes,1,rep,name=servers" json:"servers,omitempty"`
   // REQUIRED: One or more labels that indicate a specific set of pods/VMs
   // on which this gateway configuration should be applied.
   // The scope of label search is platform dependent.
   // On Kubernetes, for example, the scope includes pods running in
   // all reachable namespaces.
   Selector map[string]string `protobuf:"bytes,2,rep,name=selector"
json:"selector,omitempty" protobuf_key:"bytes,1,opt,name=key,proto3"
protobuf_val:"bytes,2,opt,name=value,proto3"`
}
```

定义虚拟服务器端口协议，TLS设置等

标签匹配，k8s中所有namespace的pod都会进行匹配

Gateway原理及实现

HUAWEI

```
type Server struct {
   // REQUIRED: The Port on which the proxy should listen for incoming
   // connections
   Port *Port `protobuf:"bytes,1,opt,name=port" json:"port,omitempty"`
   // REQUIRED. A list of hosts exposed by this gateway. At least one
   // host is required. While typically applicable to
   // HTTP services, it can also be used for TCP services using TLS with
   // SNI. May contain a wildcard prefix for the bottom-level component of
   // a domain name. For example `*.foo.com` matches `bar.foo.com`
   // and `*.com` matches `bar.foo.com`, `example.com`, and so on.
   //
   // **Note**: A `VirtualService` that is bound to a gateway must have one
   // or more hosts that match the hosts specified in a server. The match
   // could be an exact match or a suffix match with the server's hosts. For
   // example, if the server's hosts specifies "*.example.com",
   // VirtualServices with hosts dev.example.com, prod.example.com will
   // match. However, VirtualServices with hosts example.com or
   // newexample.com will not match.
   Hosts []string `protobuf:"bytes,2,rep,name=hosts" json:"hosts,omitempty"`
   // Set of TLS related options that govern the server's behavior. Use
   // these options to control if all http requests should be redirected to
   // https, and the TLS modes to use.
   Tls *Server_TLSOptions `protobuf:"bytes,3,opt,name=tls" json:"tls,omitempty"`
}
```

Gateway原理及实现

```
type VirtualService struct {

   Hosts []string protobuf:"bytes,1,rep,name=hosts" json:"hosts,omitempty"`

   Gateways []string                                    "gateways,omitempty"`
   // An ordered list of route rules for HTTP traffic. HTTP routes will be
   // applied to platform service ports named 'http-*'/'http2-*'/'grpc-*', gateway
   // ports with protocol HTTP/HTTP2/GRPC/ TLS-terminated-HTTPS and service
   // entry ports using HTTP/HTTP2/GRPC protocols.  The first rule matching
   // an incoming request is used.
   Http []*HTTPRoute `                                  :"http,omitempty"`

   Tls []*TLSRoute `pr                                  ls,omitempty"`

   Tcp []*TCPRoute `pr                                  cp,omitempty"`

   ConfigScope ConfigScope
}
```

域名，根据平台的不通可以不是FQDN

Gateway选择

HTTP路由

TLS路由

TCP路由

规则作用域，namespaced/meshscoped

Gateway原理及实现

HUAWEI

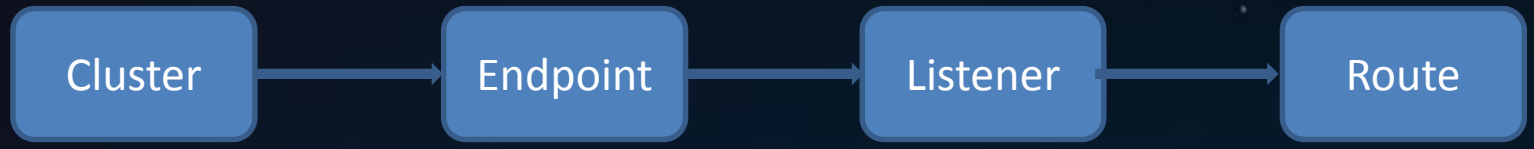
```
// Describes match conditions and actions for routing HTTP/1.1, HTTP2, and
// gRPC traffic. See VirtualService for usage examples.
type HTTPRoute struct {

   Match []*HTTPMatchRequest `protobuf:"bytes,1,rep,name=match" json:"match,omitempty"`

   Route []*HTTPRouteDestination `protobuf:"bytes,2,rep,name=route" json:"route,omitempty"`

   Redirect *HTTPRedirect `protobuf:"bytes,3,opt,name=redirect" json:"redirect,omitempty"`

   Rewrite *HTTPRewrite `protobuf:"bytes,4,opt,name=rewrite" json:"rewrite,omitempty"`

   // Timeout for HTTP requests.
   Timeout *google_protobuf.Duration `protobuf:"bytes,6,opt,name=timeout" json:"timeout,omitempty"`
   // Retry policy for HTTP requests.
   Retries *HTTPRetry `protobuf:"bytes,7,opt,name=retries" json:"retries,omitempty"`

   Fault *HTTPFaultInjection `protobuf:"bytes,8,opt,name=fault" json:"fault,omitempty"`

   Mirror *Destination `protobuf:"bytes,9,opt,name=mirror" json:"mirror,omitempty"`
   …
}
```

路由规则

Gateway原理及实现

Gateway配置下发：

遵循 make-before-break原则，杜绝规则更新过程中出现503



router类型与sidecar类型的proxy最本质的区别是没有inbound cluster，endpoint， listener

这也从侧面证明Gateway不是流量的终点只是充当一个代理转发。

Gateway demo演示

- 控制Ingress HTTP流量
- 利用HTTPS保护后端服务
- mTLS
- 控制egress流量

理解外部请求如何到达应用

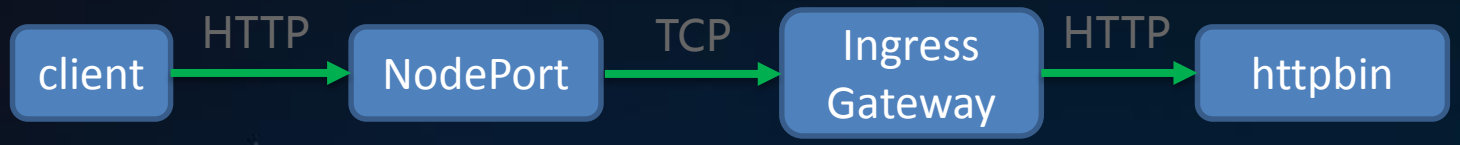
1) Client发起请求到特定端口
2) Load Balancer 监听在这个端口，并转发到后端
3) 在Istio中，LB将请求转发到IngressGateway 服务
4) Service将请求转发到IngressGateway pod
5) Pod 获取Gateway 和 VirtualService配置，获取端口、协议、证书，创建监听器
6) Gateway pod 根据路由将请求转发到应用pod（不是service）

控制Ingress HTTP流量

- 创建应用

  $ kubectl apply -f samples/httpbin/httpbin.yaml
- 创建规则

  $ kubectl apply -f samples/httpbin/httpbin-gateway.yaml
- 访问 http://139.159.236.125:31380/headers



Cleanup：

$ kubectl delete gateway httpbin-gateway

$ kubectl delete virtualservice httpbin

$ kubectl delete -f samples/httpbin/httpbin.yaml

HTTPS termination

- 生成证书

  https://istio.io/docs/tasks/traffic-management/secure-ingress/#generate-client-and-server-certificates-and-keys

- 创建secret：名称一定是istio-ingressgateway-certs，否则mount不上

  $ kubectl create -n istio-system secret tls istio-ingressgateway-certs --key httpbin.example.com/3_application/private/httpbin.example.com.key.pem --cert httpbin.example.com/3_application/certs/httpbin.example.com.cert.pem

- 创建应用

  $ kubectl apply -f samples/httpbin/httpbin.yaml

- 创建路由规则

  $ kubectl apply -f samples/httpbin/httpbin-gateway-https.yaml

HTTPS termination

- 通过HTTPS访问

$ curl -v -HHost:httpbin.example.com --resolve httpbin.example.com:31390:100.109.176.196 --cacert httpbin.example.com/2_intermediate/certs/ca-chain.cert.pem https://httpbin.example.com:31390/status/418

mTLS

- 创建包含CA证书的secret

kubectl create -n istio-system secret generic istio-ingressgateway-ca-certs --from-file=httpbin.example.com/2_intermediate/certs/ca-chain.cert.pem

- 更新Gateway TLS setting

```
cat <<EOF | kubectl apply -f -
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: mygateway
spec:
  selector:
    istio: ingressgateway # use istio default ingress gateway
  servers:
  - port:
      number: 443
      name: https
      protocol: HTTPS
    tls:
      mode: MUTUAL
      serverCertificate: /etc/istio/ingressgateway-certs/tls.crt
      privateKey: /etc/istio/ingressgateway-certs/tls.key
      caCertificates: /etc/istio/ingressgateway-ca-certs/ca-chain.cert.pem
    hosts:
    - "httpbin.example.com"
EOF
```

mtLS

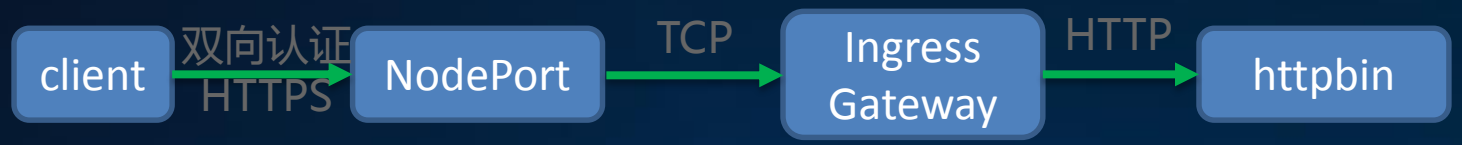
- 通过HTTPS访问

$ curl -v -HHost:httpbin.example.com --resolve httpbin.example.com:31390:139.159.236.125 --cacert httpbin.example.com/2_intermediate/certs/ca-chain.cert.pem https://httpbin.example.com:31390/status/418

curl: (35) error:14094410:SSL routines:SSL3_READ_BYTES:sslv3 alert handshake failure

$ curl -v -HHost:httpbin.example.com --resolve httpbin.example.com:31390:139.159.236.125 --cacert httpbin.example.com/2_intermediate/certs/ca-chain.cert.pem --cert httpbin.example.com/4_client/certs/httpbin.example.com.cert.pem --key httpbin.example.com/4_client/private/httpbin.example.com.key.pem https://httpbin.example.com:31390/status/418

Gateway要验证客户端的证书，所以必须携带证书才能访问



Istio访问外部服务

Istio网格内默认不能访问外部服务，如果需要访问外部服务有三种方式：

- Istio安装时设置：

  --set global.proxy.includeIPRanges="10.0.0.1/24"

- 创建应用时指定pod annotation

  traffic.sidecar.istio.io/includeOutboundIPRanges: "127.0.0.1/24,10.96.0.1/24 “

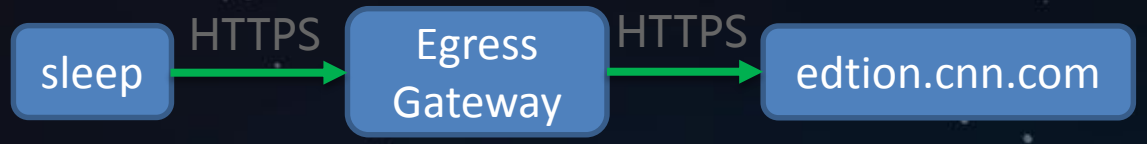
- 创建ServiceEntry

```
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: httpbin-ext
spec:
  hosts:
  - httpbin.org
  ports:
  - number: 80
    name: http
    protocol: HTTP
  resolution: DNS
  location: MESH_EXTERNAL
```

允许集群内访问外部服务
http://httpbin.org:80

通过egress gateway控制访问外部服务

访问 *https://edition.cnn.com/politics*



1) 创建ServiceEntry，允许访问edtion.cnn.com

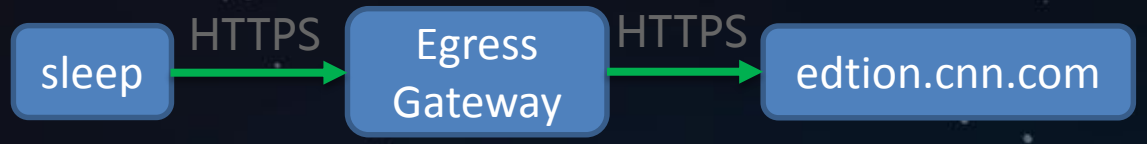
```yaml
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: cnn
spec:
  hosts:
  - edition.cnn.com
  ports:
  - number: 443
    name: tls
    protocol: TLS
  resolution: DNS
```

2) 创建 Gateway，指定egress gateway监听端口

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: istio-egressgateway
spec:
  selector:
    istio: egressgateway
  servers:
  - port:
      number: 443
      name: tls
      protocol: TLS
    hosts:
    - edition.cnn.com
    tls:
      mode: PASSTHROUGH
```

通过egress gateway控制访问外部服务

访问 https://edition.cnn.com/politics



3) 创建VirtualService，指定路由规则，

网格内普通应用访问edtion.cnn.com:443，全部转发到egress-gateway， egress-gateway透明转发

```
spec:
  hosts:
  - edition.cnn.com
  gateways:
  - mesh
  - istio-egressgateway
  tls:
  - match:
    - gateways:
      - mesh
      port: 443
      sni_hosts:
      - edition.cnn.com
    route:
    - destination:
        host: istio-egressgateway.istio-system.svc.cluster.local
        subset: cnn
        port:
          number: 443
```

```
- match:
  - gateways:
    - istio-egressgateway
    port: 443
    sni_hosts:
    - edition.cnn.com
  route:
  - destination:
      host: edition.cnn.com
      port:
        number: 443
    weight: 100
```

3) 创建DestinationRule，指定subset

```
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: egressgateway-for-cnn
spec:
  host: istio-egressgateway.istio-system.svc.cluster.local
  subsets:
  - name: cnn
```

HUAWEI

HUAWEI

Thank You

直播 每周四 晚20:00

CloudNativeLives

Kubernetes Docker Istio