IBM

Open by design™

IBM开源技术微讲堂

Istio系列

第七讲

Istio 跨云管理方案解析

http://ibm.biz/opentech-ma

“Istio” 系列公开课

- 每周四晚8点档
  - Istio初探
  - Istio上手: 基本概念，安装并利用istio进行微服务流量管控
  - Istio安全
  - Envoy
  - Istio Mixer
  - Istio监控和可视化微服务
  - Istio跨云管理方案解析
  - Using Istio to monitor serverless platform

讲师介绍

曹龙龙

Pure Blue

IBM Cloud Private Developer

Istio Maintainer

议程

- Istio跨云起源以及要解决的核心问题
- Istio目前的跨云管理方案
  1）Pod-to-pod connectivity
  2）Gateway connectivity
- Istio跨云尚未解决的问题以及未来设想

Istio跨云方案的起源

- 企业级数据中心多是跨多个集群的
- 多个集群可能是不同云提供商提供
- 多个集群也可能是同时包含公有云服务和私有云服务的混合云环境
- 企业会选择将服务部署到最适合的云平台上
- 尽量避免传统应用改造的痛点来适应新平台
- 避免被公有云厂商绑定
- 部分传统应用部署在Bare Metal纯VM上
- 企业负责的业务要求不同集群之间服务能够相互通信

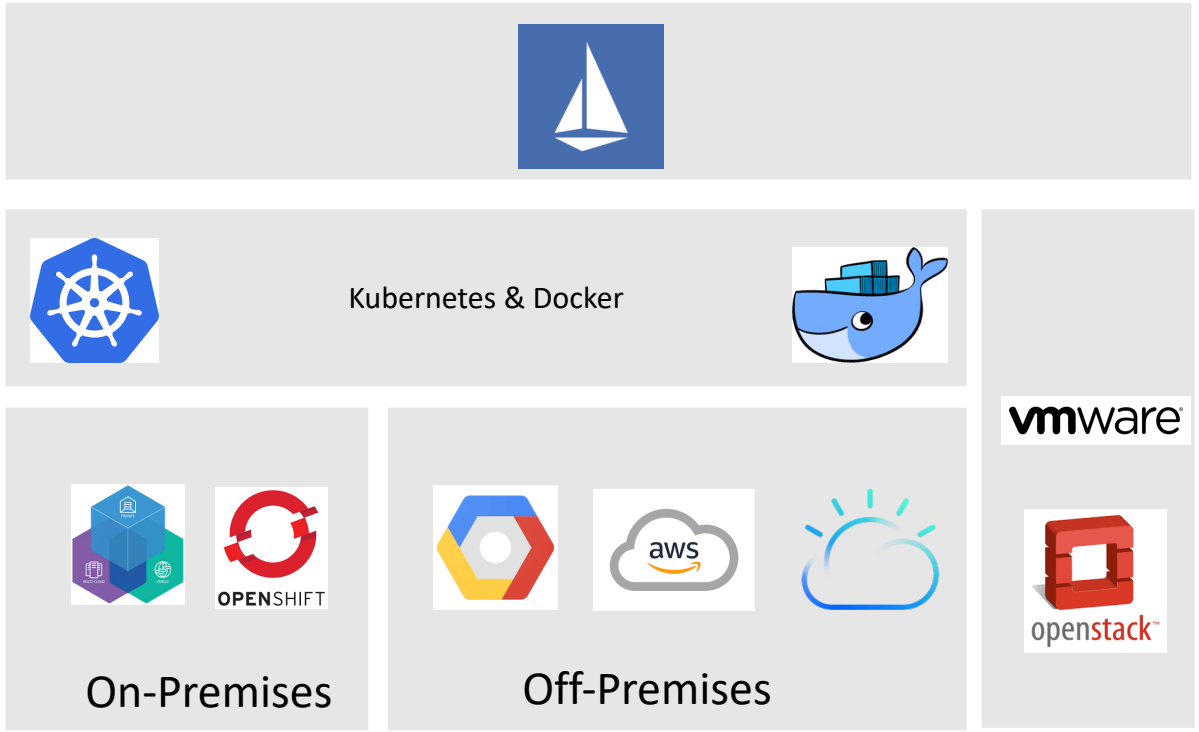
Istio目前支持的平台

- Istio设计之初就是平台独立的
- 支持Kubernetes Mesos Consul以及Bare Metal
- 可以部署到多种公有云平台上：

  GKE
  AWK
  IBM Cloud
  Alibaba Cloud

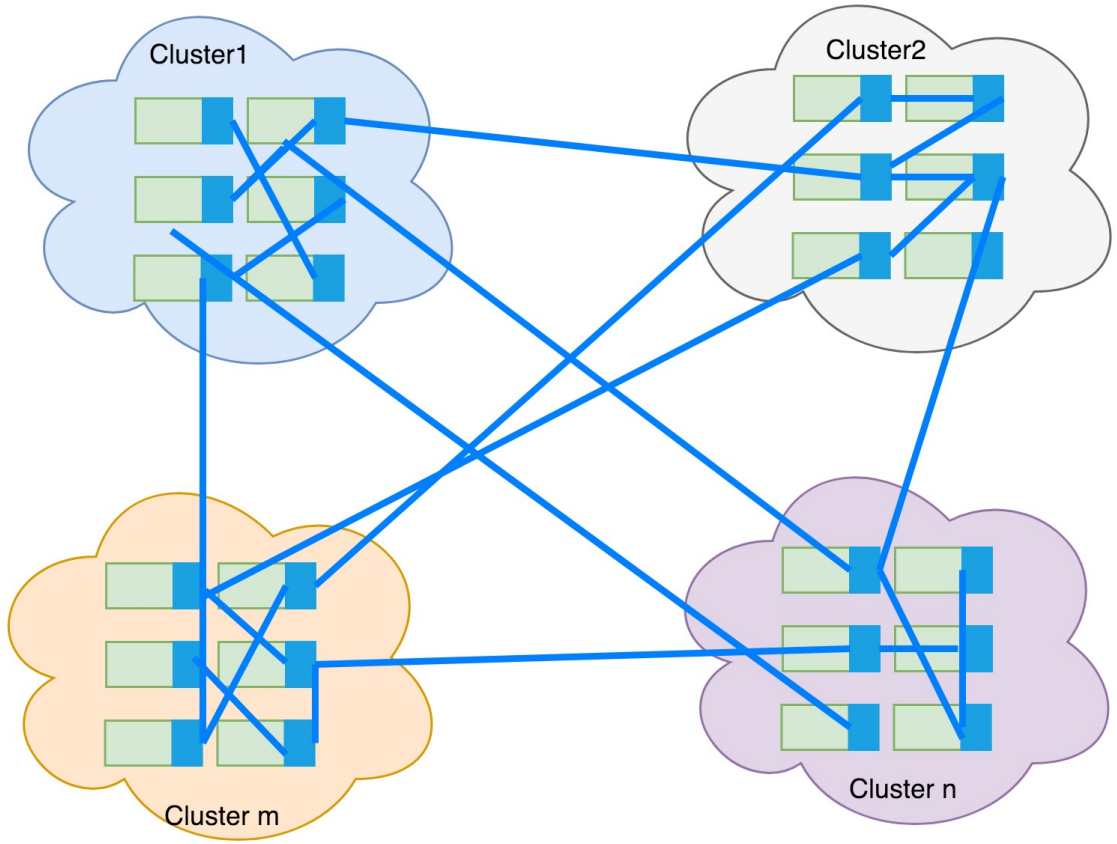
- 支持私有云云平台

  IBM Cloud Private
  Open Shift



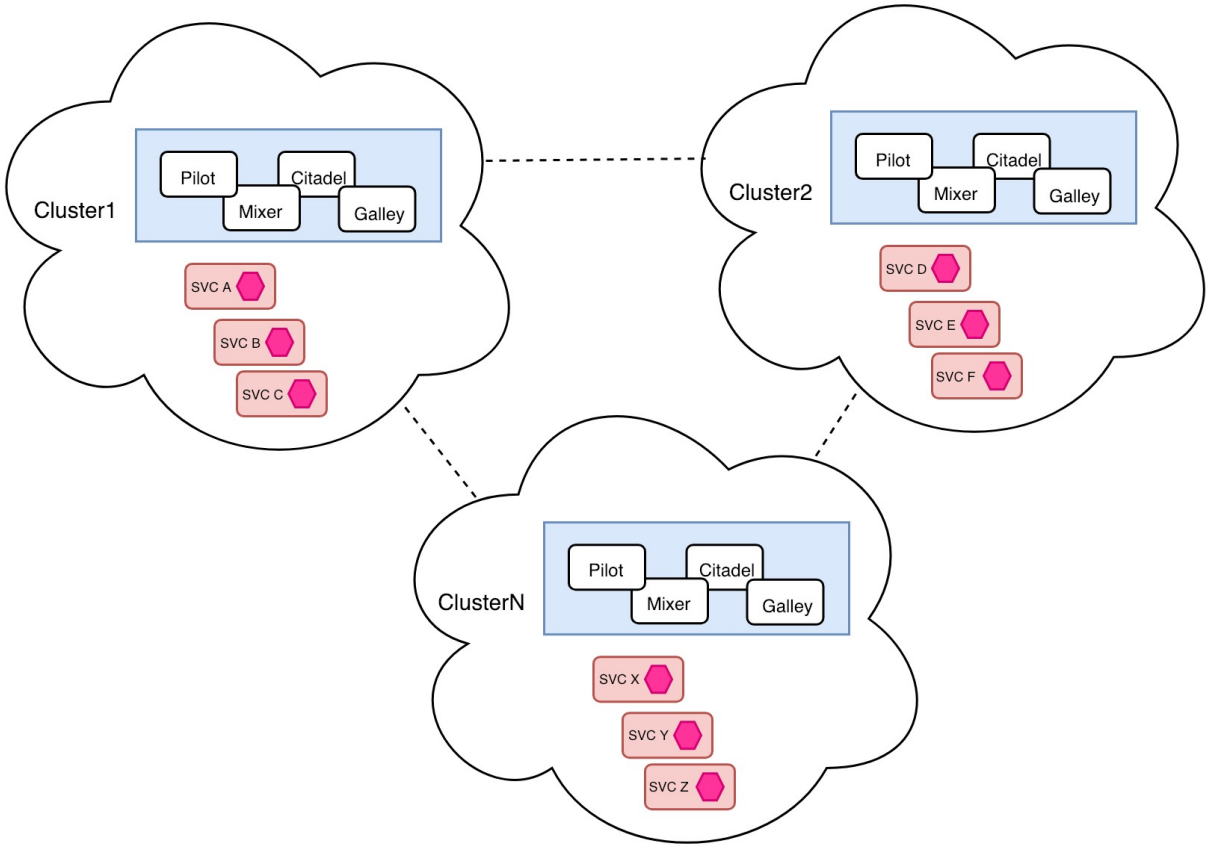
Istio单一服务网格

- 企业是否使用Istio很大程度上决定于应用能否以最底成本迁移
- Istio对单一集群的支持相对成熟
- 不同集群的服务之间要能相互通信
- 单一服务网格的必要性



Istio跨云需要解决的问题

- 多平台多网络架构支持
- 服务注册与服务发现
- 服务间访问方式
- Istio配置信息的扩散
- 避免sync循环
- 扩展与性能
- 加密与安全
- 配置简单

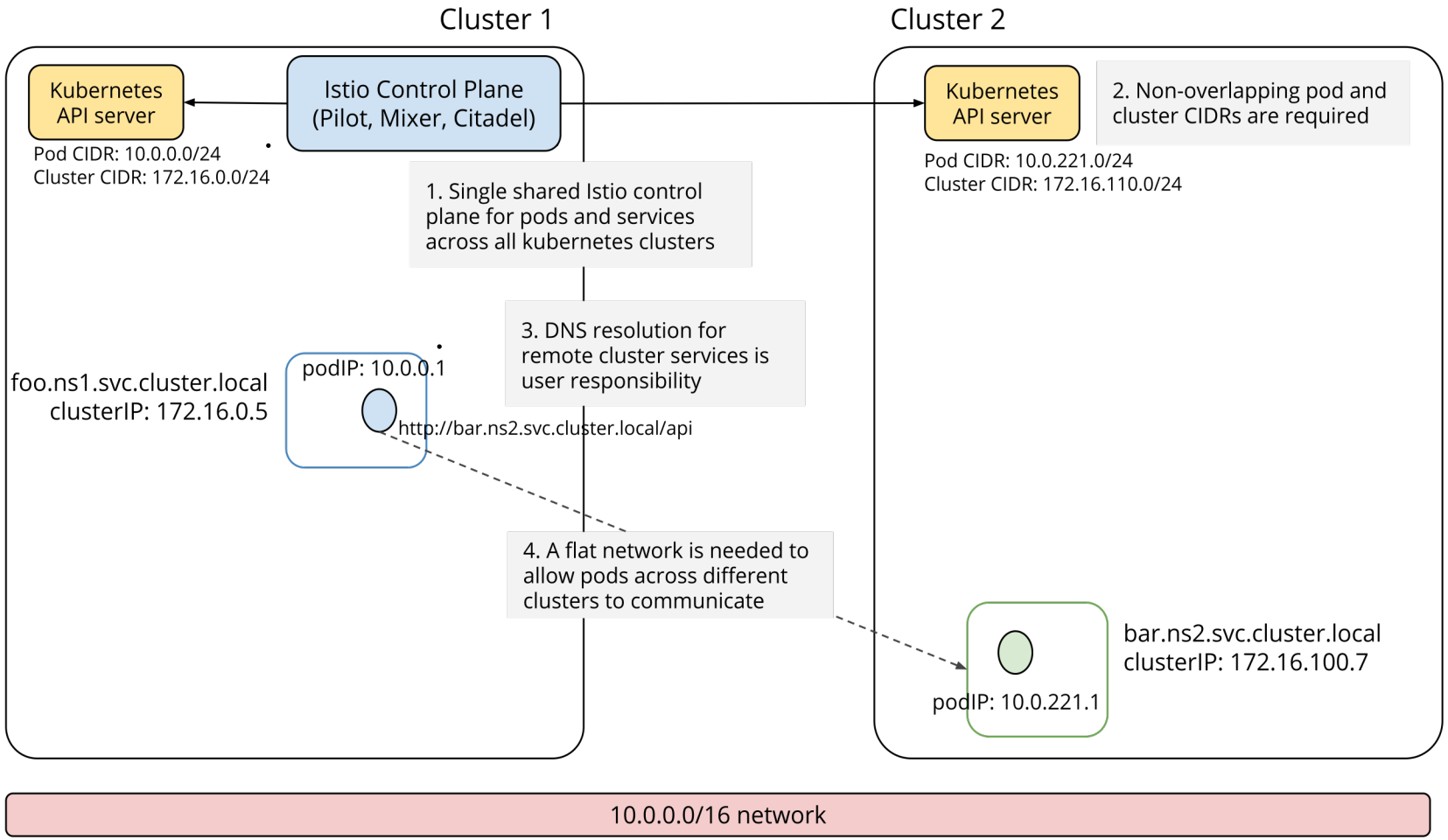


目前Istio跨云管理方案

根据网络架构分为两种：

- 同一网络平面 - Pod-to-pod connectivity
- 不同网络平面 - Gateway connectivity

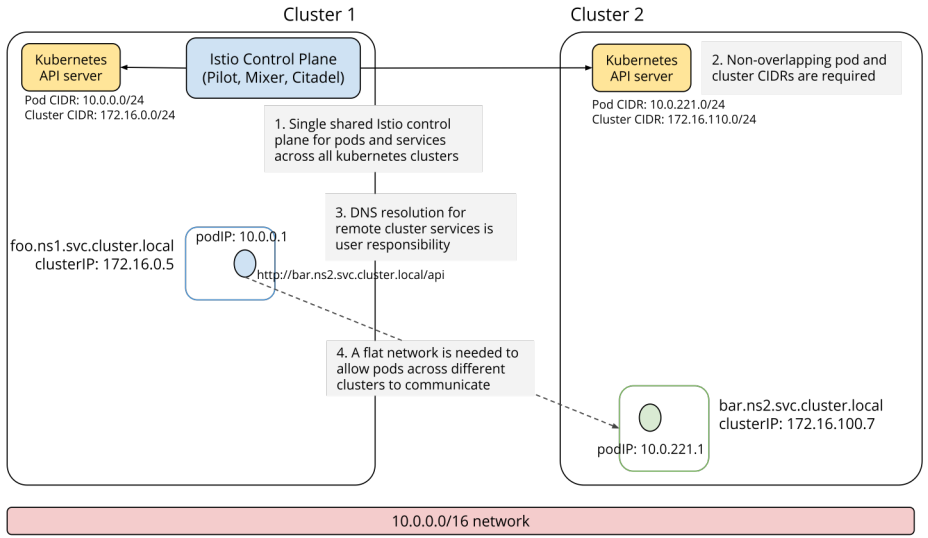
Pod-to-pod connectivity – 架构设计



- Istio控制平面有主次之分
- Pod 集群CIDR不能互相重叠
- 解析非本地集群的服务责任给用户
- 所有服务自动加入整个网格

Pop-to-pod connectivity – 部署要点

- Istio控制平面有主次之分
- 集群之间pod CIDR不能重合
- 跨集群的pod之间能直接通信，可以安装VPN来实现
- 在node-to-node网络模型的环境中(eg.Calico)，也可通过更改集群节点间的路由表来实现pod之间的相互通信，
- 主Istio控制平面pilot, policy, telemetry, statsd和zipkin最好使用ingressgateway暴露，pod IP随时会变化
- 在主Istio控制平面创建访问remote集群的secret，切记要打label

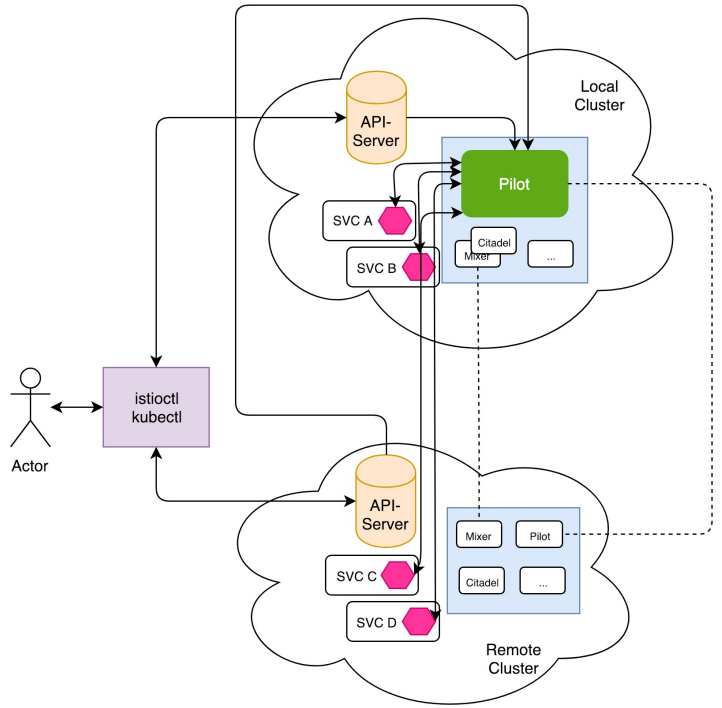


Pop-to-pod connectivity – 实现细节-1

- 主Istio控制平面与Istio-remote控制平面安装组件区别：

```
dependencies:
  - name: sidecarInjectorWebhook
    version: 1.1.0
    condition: sidecarInjectorWebhook.enabled
    repository: file://../subcharts/sidecarInjectorWebhook
  - name: security
    version: 1.1.0
    condition: security.enabled
    repository: file://../subcharts/security
  - name: ingress
    version: 1.1.0
    condition: ingress.enabled
    repository: file://../subcharts/ingress
  - name: gateways
    version: 1.1.0
    condition: gateways.enabled
    repository: file://../subcharts/gateways
  - name: mixer
    version: 1.1.0
    condition: or mixer.policy.enabled mixer.telemetry.enabled
    repository: file://../subcharts/mixer
  - name: nodeagent
    version: 1.1.0
    condition: nodeagent.enabled
    repository: file://../subcharts/nodeagent
  - name: pilot
    version: 1.1.0
    condition: pilot.enabled
    repository: file://../subcharts/pilot
  - name: grafana
    version: 1.1.0
    condition: grafana.enabled
    repository: file://../subcharts/grafana
  - name: prometheus
    version: 1.1.0
    condition: prometheus.enabled
    repository: file://../subcharts/prometheus
  - name: servicegraph
    version: 1.1.0
    condition: servicegraph.enabled
    repository: file://../subcharts/servicegraph
  - name: tracing
    version: 1.1.0
    condition: tracing.enabled
    repository: file://../subcharts/tracing
  - name: galley
    version: 1.1.0
    condition: galley.enabled
    repository: file://../subcharts/galley
  - name: kiali
    version: 1.1.0
    condition: kiali.enabled
    repository: file://../subcharts/kiali
  - name: istiocoredns
    version: 1.1.0
    condition: istiocoredns.enabled
    repository: file://../subcharts/istiocoredns
  - name: certmanager
    version: 1.1.0
    condition: certmanager.enabled
    repository: file://../subcharts/certmanager
  - name: istio-cni
    version: ">=0.0.1"
    repository: "@istio.io"
    condition: istio_cni.enabled
  - name: telemetry-gateway
    version: 1.1.0
    condition: or telemetry-gateway.grafanaEnabled telemetry-gateway.prometheusEnabled
    repository: file://../subcharts/telemetry-gateway
```

```
dependencies:
  - name: sidecarInjectorWebhook
    version: 1.1.0
    condition: sidecarInjectorWebhook.enabled
    repository: file://../subcharts/sidecarInjectorWebhook
  - name: security
    version: 1.1.0
    condition: security.enabled
    repository: file://../subcharts/security
  - name: gateways
    version: 1.1.0
    condition: gateways.enabled
    repository: file://../subcharts/gateways
  - name: istio-cni
    version: ">=0.0.1"
    repository: "@istio.io"
    condition: istio_cni.enabled
```

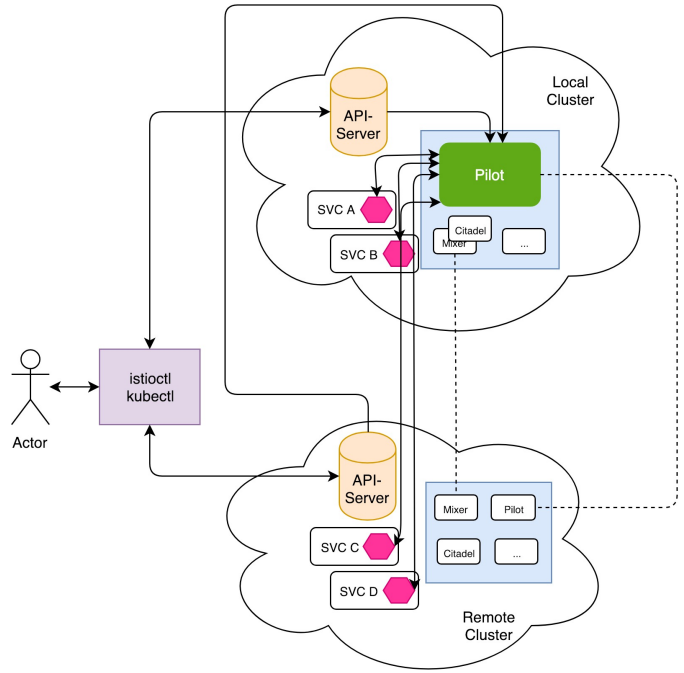


Pop-to-pod connectivity – 实现细节-2

- Istio-remote控制平面创建的endpoints

```
{{- if or .Values.global.remotePilotCreateSvcEndpoint .Values.global.createRemoteSvcEndpoints }}
apiVersion: v1
kind: Endpoints
metadata:
  name: istio-pilot
  namespace: {{ .Release.Namespace }}
subsets:
- addresses:
  - ip: {{ .Values.global.remotePilotAddress }}
  ports:
  - port: 15003
    name: http-old-discovery # mTLS or non-mTLS depending on auth setting
  - port: 15005
    name: https-discovery # always mTLS
  - port: 15007
    name: http-discovery # always plain-text
  - port: 15010
    name: grpc-xds # direct
  - port: 15011
    name: https-xds # mTLS or non-mTLS depending on auth setting
  - port: 8080
    name: http-legacy-discovery # direct
  - port: 9093
    name: http-monitoring
{{- end }}
{{- if and .Values.global.remotePolicyAddress .Values.global.createRemoteSvcEndpoints }}
```

```
{{- if and .Values.global.remotePolicyAddress .Values.global.createRemoteSvcEndpoints }}
---
apiVersion: v1
kind: Endpoints
metadata:
  name: istio-policy
  namespace: {{ .Release.Namespace }}
subsets:
- addresses:
  - ip: {{ .Values.global.remotePolicyAddress }}
  ports:
  - name: grpc-mixer
    port: 9091
  - name: grpc-mixer-mtls
    port: 15004
  - name: http-monitoring
    port: 9093
{{- end }}
{{- if and .Values.global.remoteTelemetryAddress .Values.global.createRemoteSvcEndpoints }}
---
apiVersion: v1
kind: Endpoints
metadata:
  name: istio-telemetry
  namespace: istio-system
subsets:
- addresses:
  - ip: {{ .Values.global.remoteTelemetryAddress }}
  ports:
  - name: grpc-mixer
    port: 9091
  - name: grpc-mixer-mtls
    port: 15004
  - name: http-monitoring
    port: 9093
  - name: prometheus
    port: 42422
{{- end }}
```



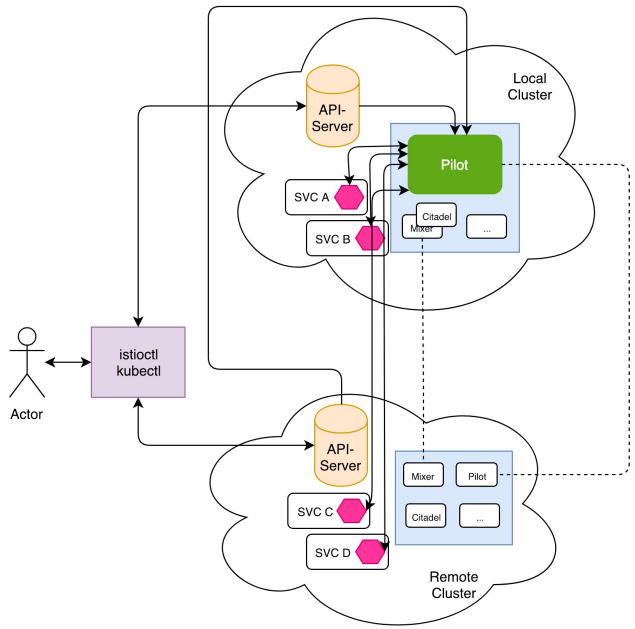
Pop-to-pod connectivity – 实现细节-3

- ServiceAccount与Secret的创建

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: istio-multi
  namespace: {{ .Release.Namespace }}
```

```yaml
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: istio-reader
rules:
  - apiGroups: ['']
    resources: ['nodes', 'pods', 'services', 'endpoints']
    verbs: ['get', 'watch', 'list']
```

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: istio-multi
  labels:
    chart: {{ .Chart.Name }}-{{ .Chart.Version }}
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: istio-reader
subjects:
- kind: ServiceAccount
  name: istio-multi
  namespace: {{ .Release.Namespace }}
```

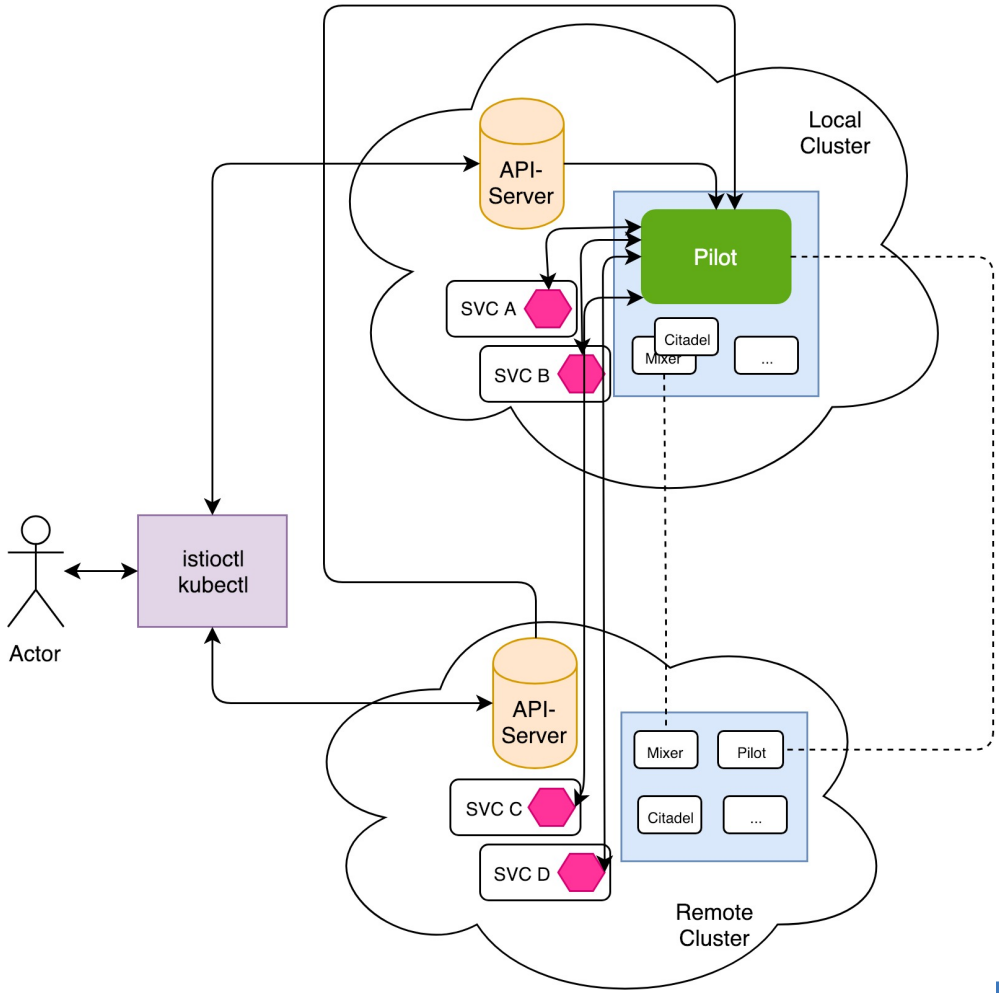



```bash
export WORK_DIR=$(pwd)
CLUSTER_NAME=$(kubectl config view --minify=true -o "jsonpath={.clusters[].name}")
export KUBECFG_FILE=${WORK_DIR}/${CLUSTER_NAME}
SERVER=$(kubectl config view --minify=true -o "jsonpath={.clusters[].cluster.server}")
NAMESPACE=istio-system
SERVICE_ACCOUNT=istio-multi
SECRET_NAME=$(kubectl get sa ${SERVICE_ACCOUNT} -n ${NAMESPACE} -o jsonpath='{.secrets[].name}')
CA_DATA=$(kubectl get secret ${SECRET_NAME} -n ${NAMESPACE} -o "jsonpath={.data['ca\.crt']}")
TOKEN=$(kubectl get secret ${SECRET_NAME} -n ${NAMESPACE} -o "jsonpath={.data['token']}" | base64 --decode)
```

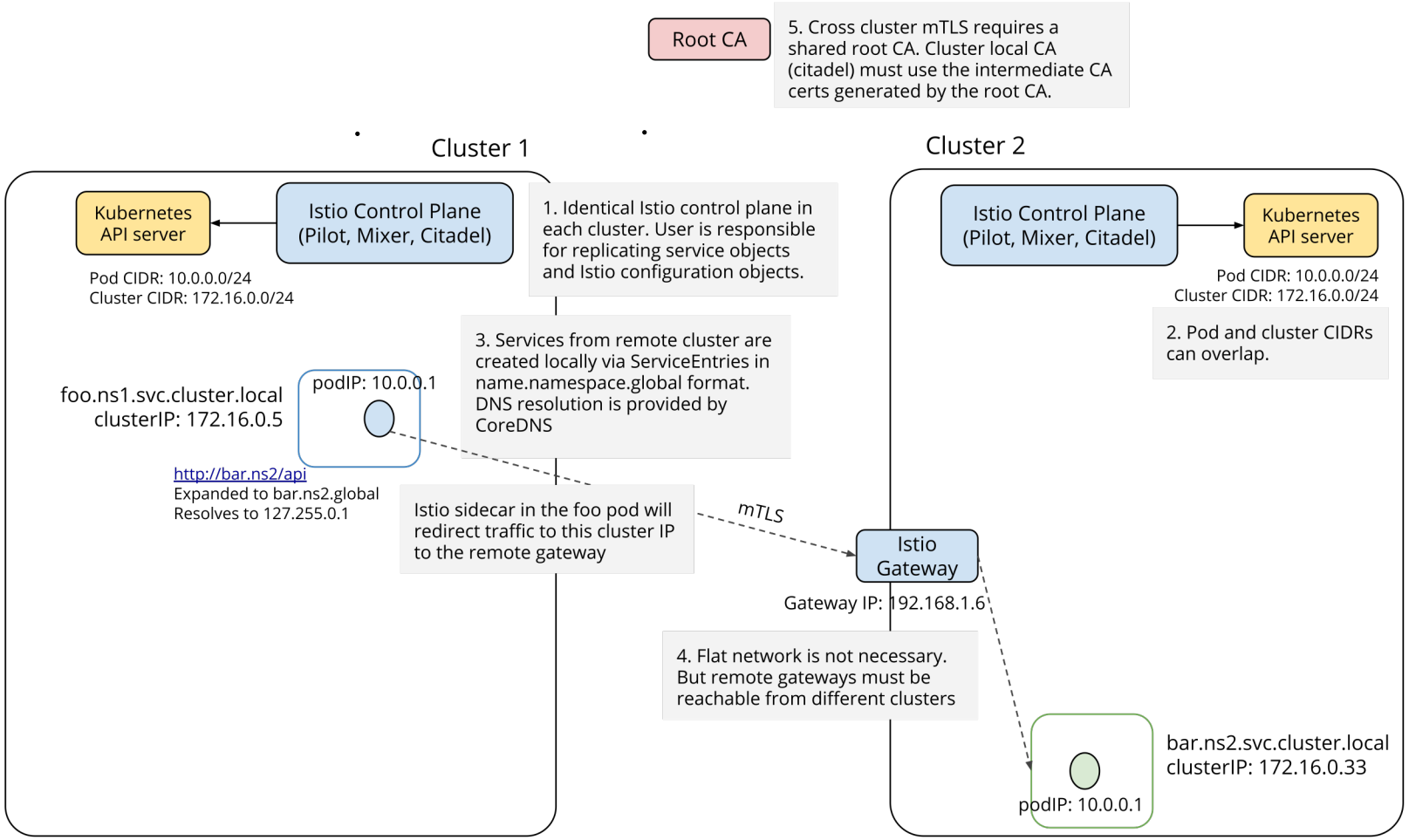
```bash
kubectl create secret generic ${CLUSTER_NAME} --from-file ${KUBECFG_FILE} -n ${NAMESPACE}
kubectl label secret ${CLUSTER_NAME} istio/multiCluster=true -n ${NAMESPACE}
```

Pod-to-pod connectivity – Limitations

- 多个remote控制层接入到单一集中式的local控制层
- 要求网络在同一控制平面，跨集群pod直接能直接通信
- 集中式的Pilot监听secret并管理对应的集群，实现服务注册与发现
- Envoy与单一集中式的Pilot通信
- 集中式的Pilot会成为性能瓶颈
- Istio配置扩散不存在同步的问题
- 所有服务自动加入mesh，多集群扩展性差
- DNS不能解析非本地cluster的service,需要用户自己负责replicate服务到本地集群



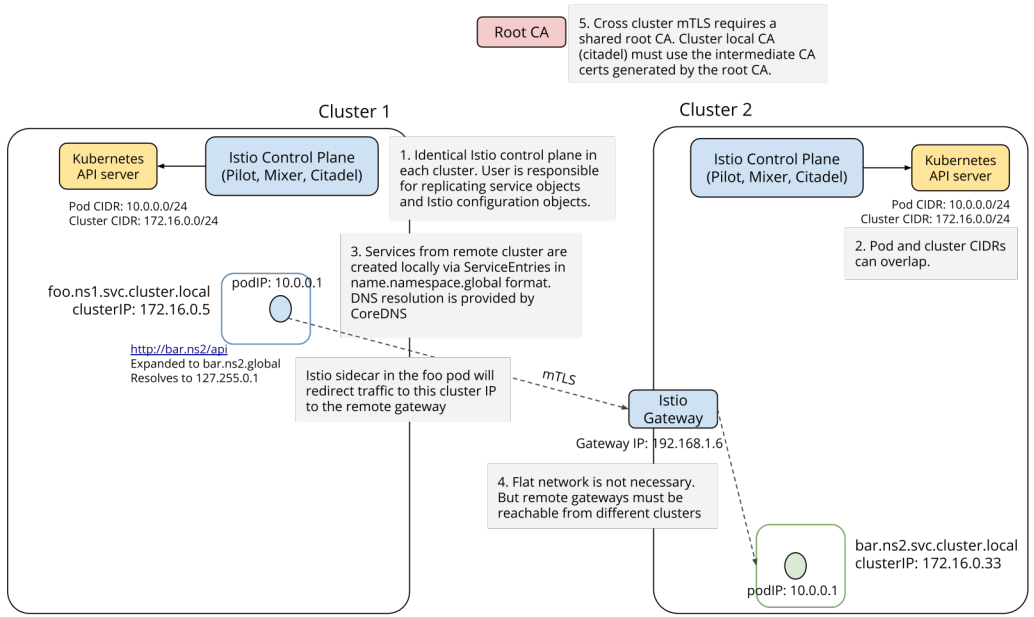
Gateway connectivity – 架构设计



- 没有集中式的Istio控制平面
- 每个集群部署一样完整的Istio控制平面
- 自定义IstioCoreDNS去解析非本地集群服务
- 对于非本地集群服务创建ServiceEntry
- Sidecar重定向所有对非本地服务的流量到对应集群的IngressGateway
- 不要求统一网络平面，但IngressGateway需要能相互访问
- IngressGateway负责将流量反向负载均衡到本地集群
- 所有Istio控制平面citadel共享同一根证书

Gateway connectivity – 部署要点

- 多个集群部署同样的Istio控制平面
- 集群的IngressGateway能够互相通信
- 所有Istio控制层Citadel不能自签证书，必须使用同一根证书，推荐企业自己的根证书
- 确保服务间双向加密和控制层加密打开
- 配置每个集群Kube-DNS解析’gobal’域名代理给Istio-CoreDNS
- 对于非本地集群服务创建ServiceEntry
- Host指定为<name>.<namespace>.global
- IP指定为虚拟IP

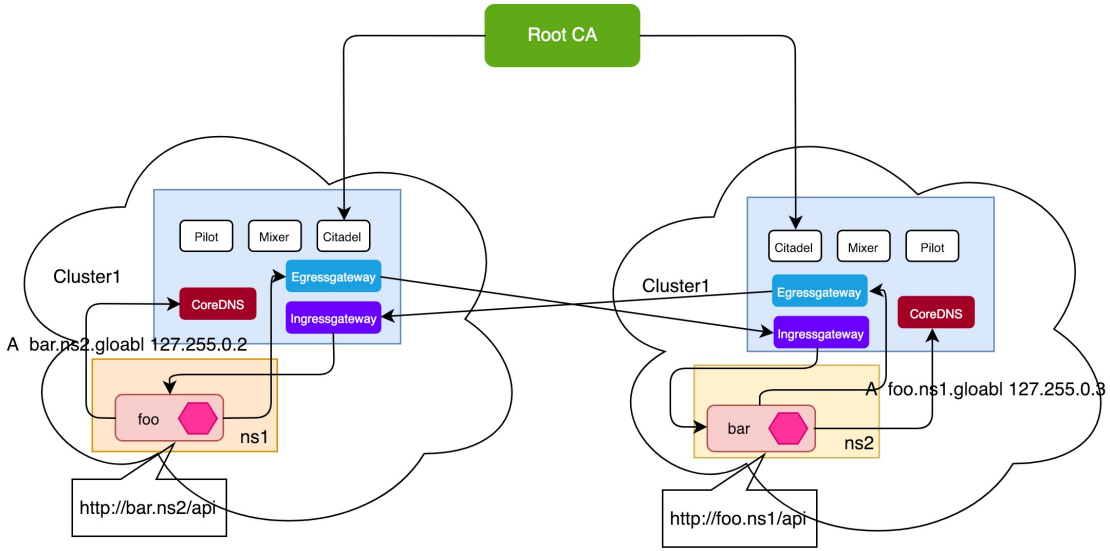


Gateway connectivity – 实现细节-1

- 配置DNS代理所有对global域名的服务的解析

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: istiocoredns
  namespace: {{ .Release.Namespace }}
spec:
  template:
    metadata:
      name: istiocoredns
    spec:
      containers:
      - name: istio-coredns-plugin
        command:
        - /usr/local/bin/plugin
        image: {{ .Values.coreDNSPluginImage }}
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 8053
          name: dns-grpc
          protocol: TCP
      dnsPolicy: Default
      volumes:
      - name: config-volume
        configMap:
          name: coredns
          items:
          - key: Corefile
            path: Corefile
```

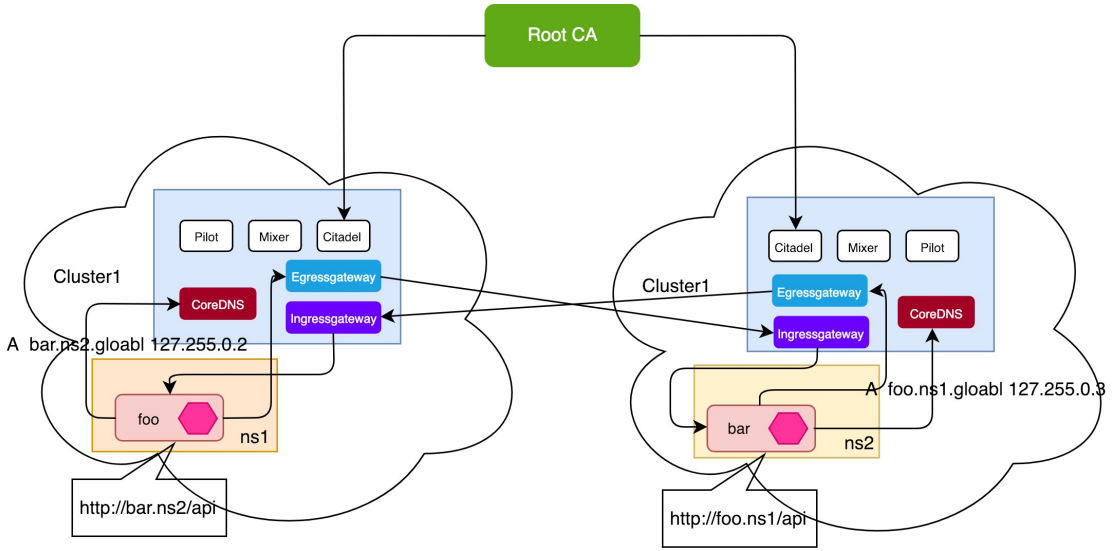
```
apiVersion: v1
data:
  Corefile: |
    .:53 {
        errors
        health
        kubernetes cluster.local in-addr.arpa ip6.arpa {
            pods insecure
            upstream
            fallthrough in-addr.arpa ip6.arpa
        }
        prometheus :9153
        proxy global 10.0.108.171
        proxy . /etc/resolv.conf
        cache 30
        reload
    }
kind: ConfigMap
  name: kube-dns
  namespace: kube-system
```



Gateway connectivity – 实现细节-2

- 对于非本地集群服务创建ServiceEntry

```
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: bar-ns2
spec:
  hosts:
  - bar.ns2.global
  location: MESH_INTERNAL
  ports:
  - name: http1
    number: 8080
    protocol: http
  - name: tcp2
    number: 9999
    protocol: tcp
  resolution: DNS
  addresses:
  - 127.255.0.2
  endpoints:
  - address: <IPofCluster2IngressGateway>
    ports:
      http1: 15443 # Do not change this port value
      tcp2: 15443 # Do not change this port value
```

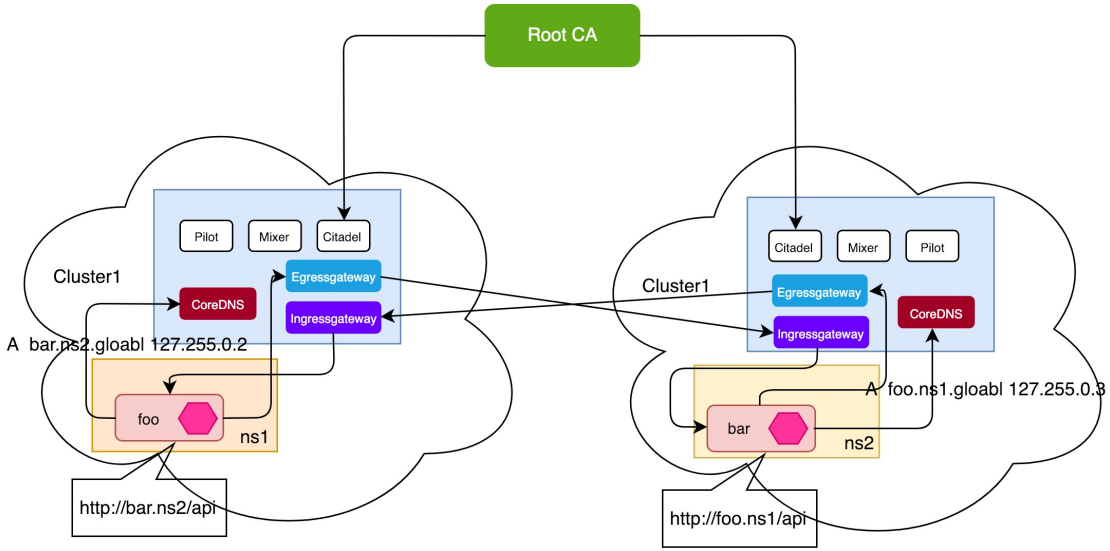


Gateway connectivity – 实现细节-3

- IngressGateway负责将流量反向负载均衡到本地

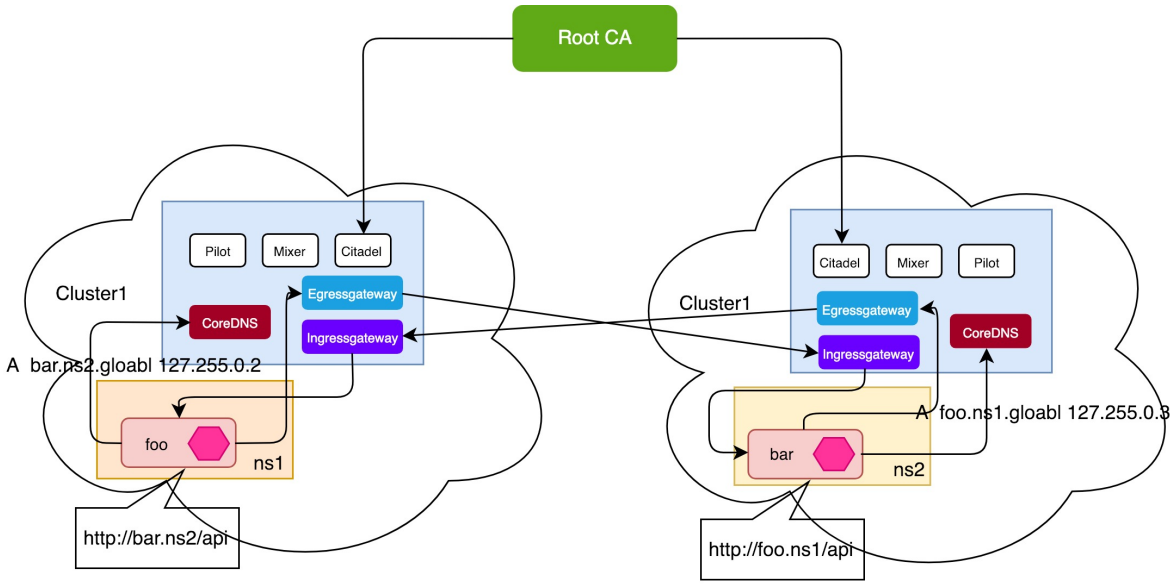
```yaml
{{- if .Values.global.multiCluster.enabled }}
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: istio-multicluster-egressgateway
  namespace: {{ .Release.Namespace }}
  labels:
    app: {{ template "gateway.name" . }}
    chart: {{ template "gateway.chart" . }}
    heritage: {{ .Release.Service }}
    release: {{ .Release.Name }}
spec:
  selector:
    istio: egressgateway
  servers:
  - hosts:
    - "*.global"
    port:
      name: tls
      number: 15443
      protocol: TLS
    tls:
      mode: AUTO_PASSTHROUGH
---
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: istio-multicluster-ingressgateway
  namespace: {{ .Release.Namespace }}
  labels:
    app: {{ template "gateway.name" . }}
    chart: {{ template "gateway.chart" . }}
    heritage: {{ .Release.Service }}
    release: {{ .Release.Name }}
spec:
  selector:
    istio: ingressgateway
  servers:
  - hosts:
    - "*.global"
    port:
      name: tls
      number: 15443
      protocol: TLS
    tls:
      mode: AUTO_PASSTHROUGH
```

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: EnvoyFilter
metadata:
  name: istio-multicluster-ingressgateway
  namespace: {{ .Release.Namespace }}
  labels:
    app: {{ template "gateway.name" . }}
    chart: {{ template "gateway.chart" . }}
    heritage: {{ .Release.Service }}
    release: {{ .Release.Name }}
spec:
   workloadLabels:
     istio: ingressgateway
   filters:
   - listenerMatch:
       portNumber: 15443
       listenerType: GATEWAY
     insertPosition:
       index: AFTER
       relativeTo: envoy.filters.network.sni_cluster
     filterName: envoy.filters.network.tcp_cluster_rewrite
     filterType: NETWORK
     filterConfig:
       cluster_pattern: "\\.global$"
       cluster_replacement: ".svc.cluster.local"
---
## To ensure all traffic to *.global is using mTLS
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: istio-multicluster-destinationrule
  namespace: {{ .Release.Namespace }}
  labels:
    app: {{ template "gateway.name" . }}
    chart: {{ template "gateway.chart" . }}
    heritage: {{ .Release.Service }}
    release: {{ .Release.Name }}
spec:
  host: "*.global"
  trafficPolicy:
    tls:
      mode: ISTIO_MUTUAL
```



Gateway connectivity – 要点分析

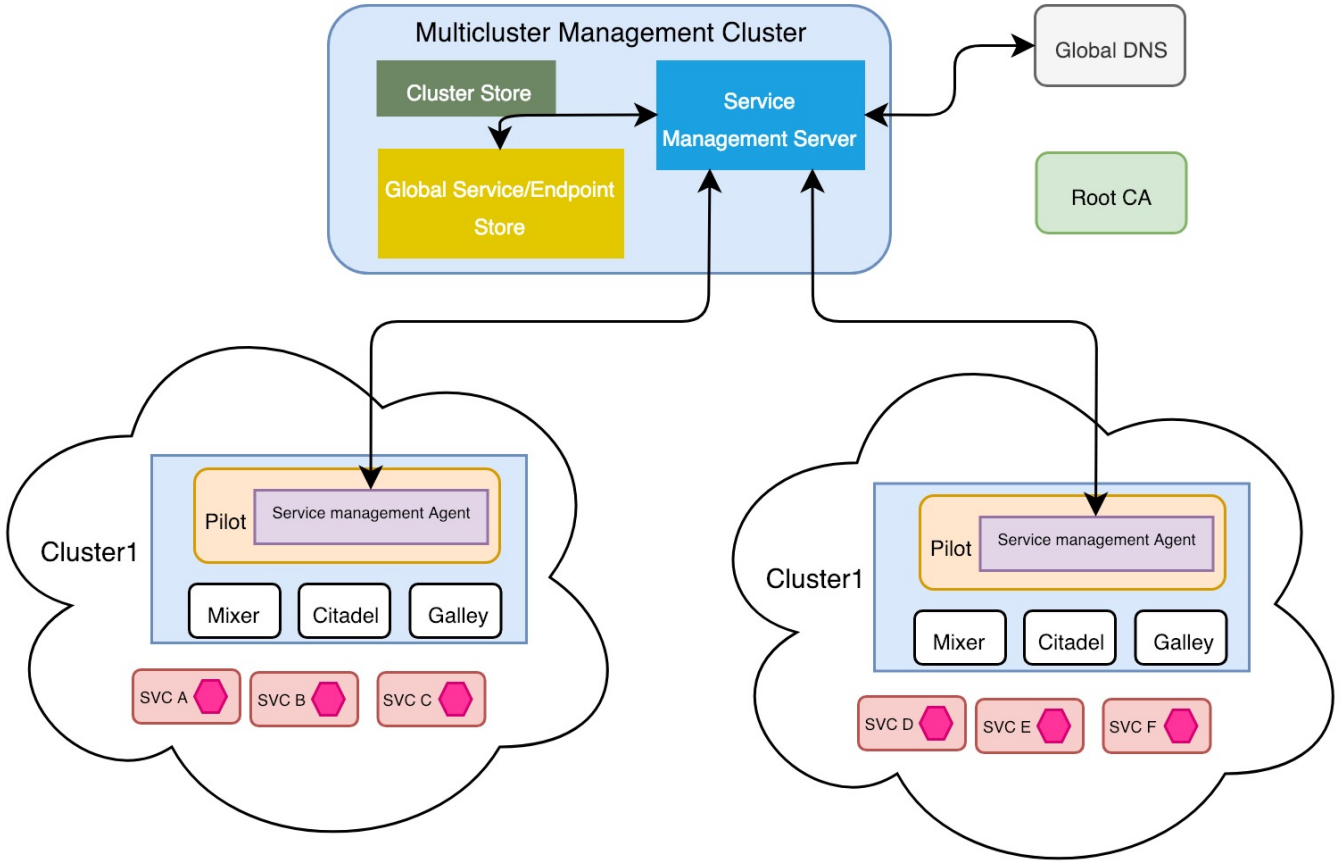
- 每个集群拥有单独的Istio控制平面
- 不要求网络在同一平面，只要IngressGateway能够对外访问
- 每个Istio控制平面Pilot监听自身集群的服务注册情况，不存在同步问题
- 需要加入网格外部服务需创建ServiceEntry，扩展性能相对好，但配置复杂
- Istio CoreDNS负责解析非本地集群服务
- 访问非本地服务通过远端集群IngressGateway
- Istio配置只对当前集群有效，相对复杂



Istio跨云方案尚未解决的问题

- 异构底层平台的支持(K8s, consul, Mesos…)
- 异构网络架构的支持(网络拓扑结构应该对于使用者透明)
- 服务网格扩展以及管理问题(global service应该能够集中管理)
- 避免单一通信channel成为性能瓶颈
- 配置简单容易回退

未来Istio跨云架构设想



- 集群应该成为为第一资源，包括集群的类型，集群名字以及服务IP的类型等
- 用户能够制定service是否是global还是本地service
- Multicluster Management Cluster专门负责管理global的资源，包括集群，global service/endpoints
- 每个集群有自己的Istio控制层，同时部署一个Service Management Agent与Multicluster Management Cluster同步多集群的service/endpoints
- 全局的DNS负责服务的解析，本地service可以通过global domain或者本地domain访问global服务
- 各个集群共享一个全局的根证书，保证service之间双向加密通信

参考链接

- https://docs.google.com/document/d/1cOxPnTpnVWKR0m7a6PYJ-0NSTZ-e2J1IULnfojz7OZ4/view#heading=h.cfqi32is79wv
- https://docs.google.com/document/d/1ut33e7kxq9SrlxYdl-HmQgCwdDTfu2DBQCjZhUV0Lo8/edit#
- https://docs.google.com/document/d/1QLMeycCoj6fbBKoOLiaDCc09Fbi89s5VoBW-eAxKwVs/edit#heading=h.qex63c29z2to
- https://docs.google.com/document/d/1LOHiUqrXj37-MCAYgUFLOedZFxnOVgxDLD5c_p1C5ao/edit#heading=h.wdquuwnxq4m7
- https://docs.google.com/document/d/1xeft7OYPAW9SB_3iccf3dizovONCCpNuXi26raElyFQ/edit#heading=h.qex63c29z2to
- https://docs.google.com/document/d/1E3cTCJiCrVjKdjI7pX5rU0MAw99xCq8UyfZKCu8LDYw/edit#heading=h.qex63c29z2to
- https://github.com/IBM/istio-hybrid
- https://github.com/rshriram/istio_federation_demo
- https://github.com/istio/istio/pull/1865#issuecomment-348206217
- https://preliminary.istio.io/docs/setup/kubernetes/multicluster-install/gateways/
- https://preliminary.istio.io/docs/setup/kubernetes/multicluster-install/vpn/

IBM

Open by design™

IBM开源技术微讲堂

Istio系列

第七讲完

http://ibm.biz/opentech-ma