1. Istio
2. 文件
3. 参考
4. 组态
5. 交通管理
6. 服务条目

服务条目

11分钟阅读

`ServiceEntry`允许在Istio的内部服务注册表中添加其他条目，以便网格中的自动发现服务可以访问/路由到这些手动指定的服务。服务条目描述服务的属性（DNS名称，VIP，端口，协议，端点）。这些服务可以是网格外部（例如，web API）或不属于平台服务注册表的网状内部服务（例如，一组与Kubernetes中的服务通信的VM）。

以下示例声明了内部应用程序通过HTTPS访问的一些外部API。sidecar检查ClientHello消息中的SNI值以路由到适当的外部服务。

```
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: external-svc-https
spec:
  hosts:
  - api.dropboxapi.com
  - www.googleapis.com
  - api.facebook.com
  location: MESH_EXTERNAL
  ports:
  - number: 443
    name: https
    protocol: TLS
  resolution: DNS
```

以下配置将在非托管VM上运行的一组MongoDB实例添加到Istio的注册表，以便可以将这些服务视为网格中的任何其他服务。关联的DestinationRule用于启动与数据库实例的mTLS连接。

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: external-svc-mongocluster
spec:
  hosts:
  - mymongodb.somedomain # not used
  addresses:
  - 192.192.192.192/24 # VIPs
  ports:
  - number: 27018
    name: mongodb
    protocol: MONGO
  location: MESH_INTERNAL
  resolution: STATIC
  endpoints:
  - address: 2.2.2.2
  - address: 3.3.3.3
```

和相关的DestinationRule

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: mtls-mongocluster
spec:
  host: mymongodb.somedomain
  trafficPolicy:
    tls:
      mode: MUTUAL
      clientCertificate: /etc/certs/myclientcert.pem
      privateKey: /etc/certs/client_private_key.pem
      caCertificates: /etc/certs/rootcacerts.pem
```

以下示例使用虚拟服务中的服务条目和TLS路由的组合，将基于SNI值的流量引导至内部出口防火墙。

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: external-svc-redirect
spec:
  hosts:
  - wikipedia.org
  - "*.wikipedia.org"
  location: MESH_EXTERNAL
  ports:
  - number: 443
    name: https
    protocol: TLS
  resolution: NONE
```

以及基于SNI值路由的关联VirtualService。

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: tls-routing
spec:
  hosts:
  - wikipedia.org
  - "*.wikipedia.org"
  tls:
  - match:
    - sniHosts:
      - wikipedia.org
      - "*.wikipedia.org"
    route:
    - destination:
        host: internal-egress-firewall.ns1.svc.cluster.local
```

具有TLS匹配的虚拟服务用于覆盖默认的SNI匹配。在没有虚拟服务的情况下，流量将被转发到维基百科域。

以下示例演示了使用专用出口网关，通过该网关转发所有外部服务流量。'exportTo'字段允许控制服务声明对网格中其他命名空间的可见性。默认情况下，服务将导出到所有名称空间。以下示例将对当前命名空间的可见性限制为“。”，以便其他命名空间无法使用它。

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: external-svc-httpbin
  namespace : egress
spec:
  hosts:
  - httpbin.com
  exportTo:
  - "."
  location: MESH_EXTERNAL
  ports:
  - number: 80
    name: http
    protocol: HTTP
  resolution: DNS
```

定义一个网关来处理所有出口流量。

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
 name: istio-egressgateway
 namespace: istio-system
spec:
 selector:
   istio: egressgateway
 servers:
 - port:
     number: 80
     name: http
     protocol: HTTP
   hosts:
   - "*"
```

并且VirtualService从边车到网关服务（istio-egressgateway.istio-system.svc.cluster.local）的路由关联，以及从网关到外部服务的路由。请注意，虚拟服务将导出到所有名称空间，使其能够通过网关将流量路由到外部服务。强制流量通过像这样的托管中间代理是一种常见的做法。

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: gateway-routing
  namespace: egress
spec:
  hosts:
  - httpbin.com
  exportTo:
  - "*"
  gateways:
  - mesh
  - istio-egressgateway
  http:
  - match:
    - port: 80
      gateways:
      - mesh
    route:
    - destination:
        host: istio-egressgateway.istio-system.svc.cluster.local
  - match:
    - port: 80
      gateways:
      - istio-egressgateway
    route:
    - destination:
        host: httpbin.com
```

以下示例演示了在主机中使用通配符进行外部服务。如果必须将连接路由到应用程序请求的IP地址（即应用程序解析DNS并尝试连接到特定IP），则必须将发现模式设置为NONE。

```
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: external-svc-wildcard-example
spec:
  hosts:
  - "*.bar.com"
  location: MESH_EXTERNAL
  ports:
  - number: 80
    name: http
    protocol: HTTP
  resolution: NONE
```

以下示例演示了通过客户端主机上的Unix域套接字提供的服务。必须将分辨率设置为STATIC才能使用Unix地址端点。

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: unix-domain-socket-example
spec:
  hosts:
  - "example.unix.local"
  location: MESH_EXTERNAL
  ports:
  - number: 80
    name: http
    protocol: HTTP
  resolution: STATIC
  endpoints:
  - address: unix:///var/run/example/socket
```

对于基于HTTP的服务，可以创建VirtualService 由多个DNS可寻址端点支持的服务。在这种情况下，应用程序可以使用HTTP_PROXY环境变量透明地重新路由VirtualService到所选后端的API调用。例如，以下配置创建一个名为foo.bar.com的不存在的外部服务，该服务由三个域支持：us.foo.bar.com：8080，uk.foo.bar.com：9080和in.foo.bar .COM：7080

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: external-svc-dns
spec:
  hosts:
  - foo.bar.com
  location: MESH_EXTERNAL
  ports:
  - number: 80
    name: http
    protocol: HTTP
  resolution: DNS
  endpoints:
  - address: us.foo.bar.com
    ports:
      https: 8080
  - address: uk.foo.bar.com
    ports:
      https: 9080
  - address: in.foo.bar.com
    ports:
      https: 7080
```

使用HTTP_PROXY=http://localhost/，来自应用程序的调用 http://foo.bar.com将在上面指定的三个域中进行负载平衡。换句话说，一个调用http://foo.bar.com/baz将被翻译成 http://uk.foo.bar.com/baz。

以下示例说明了ServiceEntry 包含主题备用名称的用法，其名称格式符合SPIFEE标准1：

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: httpbin
  namespace : httpbin-ns
spec:
  hosts:
  - httpbin.com
  location: MESH_INTERNAL
  ports:
  - number: 80
    name: http
    protocol: HTTP
  resolution: STATIC
  endpoints:
  - address: 2.2.2.2
  - address: 3.3.3.3
  subjectAltNames:
  - "spiffe://cluster.local/ns/httpbin-ns/sa/httpbin-service-account"
```

ServiceEntry

领域	类型	描述
hosts	string[]	需要。与ServiceEntry关联的主机。可以是带有通配符前缀的DNS名称（仅限外部服务）。对于HTTP流量，HTTP主机/授权标头将与hosts字段匹配。对于包含服务器名称指示（SNI）的HTTP或TLS流量，SNI值将与hosts字段匹配。对于所有其他协议，将忽略主机，如果存在，将使用端口和地址字段。请注意，当分辨率设置为键入DNS且未指定端点时，主机字段将用作端点的DNS名称以将流量路由到。
addresses	string[]	与服务关联的虚拟IP地址。可能是CIDR前缀。对于HTTP流量，将忽略地址字段，并根据HTTP主机/权限标头识别目标。如果指定了一个或多个IP地址，则如果目标IP与地址字段中指定的IP / CIDR匹配，则将传入流量标识为属于此服务。如果“地址”字段为空，则仅基于目标端口识别流量。在这种情况下，访问服务的端口不得由网格中的任何其他服务共享。换句话说，sidecar将表现为简单的TCP代理，将指定端口上的传入流量转发到指定的目标端点IP /主机。此字段不支持Unix域套接字地址。
ports	Port[]2	需要。与外部服务关联的端口。如果端点是Unix域套接字地址，则必须只有一个端口。

领域	类型	描述
location	ServiceEntry.Location	指定是否应将服务视为网格外部或网格的一部分。
resolution	ServiceEntry.Resolution	必需：主机的服务发现模式。在没有附带IP地址的情况下为TCP端口设置解析模式为NONE时必须小心。在这种情况下，将允许到所述端口上的任何IP的流量（即0.0.0.0：<端口>）。
endpoints	ServiceEntry.Endpoint[]	与服务关联的一个或多个端点。
exportTo	string[]	要将此服务导出到的命名空间列表。导出服务允许它由其他名称空间中定义的边车，网关和虚拟服务使用。此功能为服务所有者和网格管理员提供了一种机制，用于控制跨命名空间边界的服务可见性。 如果未指定名称空间，则默认情况下将服务导出到所有名称空间。 值“。”是保留的，并定义导出到声明服务的同一名称空间。类似地，值“*”是保留的，并定义导出到所有名称空间。 对于Kubernetes服务，可以通过将注释“networking.istio.io/exportTo”设置为以逗号分隔的命名空间名称列表来实现等效。 注意：在当前版本中，该exportTo值仅限于“。”或“*”（即当前命名空间或所有命名空间）。
subjectAltNames	string[]	允许实现此服务的工作负载实例的主题备用名称列表。此信息用于强制执行 安全命名。如果指定，代理将验证服务器证书的主题备用名称是否与指定值之一匹配。

ServiceEntry.Endpoint

端点定义与网格服务关联的网络地址（IP或主机名）。

领域	类型	描述

领域	类型	描述
address	string	REQUIRED：与没有端口的网络端点关联的地址。当且仅当分辨率设置为DNS时，才能使用域名，并且必须在没有通配符的情况下完全限定域名。对Unix域套接字端点使用unix：/// absolute / path / to / socket形式。
ports	map<string, uint32>	与端点关联的端口集。端口必须与声明为服务一部分的端口名称相关联。不要用于unix://地址。
labels	map<string, string>	与端点关联的一个或多个标签。
network	string	网络使Istio能够将端点分组驻留在同一L3域/网络中。假设同一网络中的所有端点可以彼此直接访问。当不同网络中的端点无法直接相互连接时，可以使用Istio网关建立连接（通常使用网关服务器中的AUTO_PASSTHROUGH模式）。这是一种高级配置，通常用于跨多个集群跨越Istio网格。
locality	string	与端点关联的位置。位置对应于故障域（例如，国家/地区/区域）。可以通过用/分隔每个封装失败域来表示任意失败域层次结构。例如，在美国，在US-East-1区域中，在可用区域az-1内，在数据中心机架r11中的端点的位置可以表示为us / us-east-1 / az-1 / r11。Istio将配置边车以路由到与边车相同的位置内的端点。如果该位置中没有端点可用，则将选择端点父位置（但在同一网络ID内）。例如，如果同一网络中有两个端点（networkID“n1”），则说e1为locality us / us-east-1 / az-1 / r11，e2为locality us / us-east-1 / az-2 / R12，来自我们/ us-east-1 / az-1 / r11地区的边车将更喜欢来自不同地区的e2来自同一地点的e1。端点e2可以是与网关（桥接网络n1和n2）或与标准服务端点相关联的IP相关联的IP。
weight	uint32	与端点关联的负载平衡权重。权重较高的端点将获得相应较高的流量。

ServiceEntry.Location

位置指定服务是Istio网格的一部分还是网格外部。位置确定多个功能的行为，例如服务到服务mTLS身份验证，策略实施等。当与网状外部的服务通信时，禁用Istio的mTLS身份验证，并在客户端执行策略实施，而不是服务器端。

名称	描述
MESH_EXTERNAL	表示该服务在网格外部。通常用于指示通过API消耗的外部服务。
MESH_INTERNAL	表示该服务是网格的一部分。通常用于指示作为扩展服务网格的一部分明确添加的服务以包括非托管基础结构（例如，添加到基于Kubernetes的服务网格的VM）。

ServiceEntry.Resolution

解决方案确定代理如何解析与服务关联的网络端点的IP地址，以便它可以路由到其中一个端点。此处指定的解析模式不会影响应用程序如何解析与服务关联的IP地址。应用程序可能仍然必须使用DNS将服务解析为IP，以便代理可以捕获出站流量。或者，对于HTTP服务，应用程序可以直接与代理通信（例如，通过设置HTTP_PROXY）以与这些服务通信。

名称	描述
NONE	假设传入连接已经解析（到特定目标IP地址）。这种连接通常使用诸如IP表REDIRECT / eBPF之类的机制通过代理进行路由。执行任何与路由相关的转换后，代理会将连接转发到绑定连接的IP地址。
STATIC	使用端点中指定的静态IP地址（请参见下文）作为与服务关联的后备实例。
DNS	在请求处理期间，尝试通过查询环境DNS来解析IP地址。如果未指定端点，则代理将解析hosts字段中指定的DNS地址（如果未使用通配符）。如果指定了端点，则将解析端点中指定的DNS地址以确定目标IP地址。DNS解析不能与Unix域套接字端点一起使用。

Links

1. https://github.com/spiffe/spiffe/blob/master/standards/SPIFFE-ID.md
2. https://istio.io/docs/reference/config/networking/v1alpha3/gateway.html#Port