How HP uses Istio manage network traffic

HeXi – HP Inc.
Zhang, Zhoujian – HP Inc.
• What is Istio
• Proxy – Traffic management
• Pilot – Discover & Agent
• Mixer – Policy
• Telemetry
• Q&A
What is Istio?

Connect, Secure, Control & Observe services

- Traffic Management
- Policy
- Security
- Telemetry
Istio - Architecture

**Data Plane**

- Service A
  - Proxy
- Service B
  - Proxy

- HTTP/1.1, HTTP/2, gRPC or TCP – with or without mTLS

**Control Plane**

- Operator
- Rules API
  - Envoy API
  - Abstract Model
  - Platform Adapter
  - Kubernetes
  - Mesos
  - Cloud Foundry

- Pilot
- Mixer
- Citadel
  - Key and certificate management

**Platform Adaptors**

- Kubernetes
- Mesos
- Cloud Foundry

**Policy checks, report, Telemetry**

**Config data to proxies**

**TLS certs to proxies**
Data Plane - Proxy
Proxy

- JWT verification
  - Check, Report...
- Listeners
  - Clusters
  - Endpoints
  - Routes
- Round Robin
  - Random
  - Weighted least request

**Envoy**

- HTTP, L3/L4 Filter
- Traffic splitting
- Dynamic service discovery
- Redirect, rewrite, fault injection...
- Load balancing
- Attribute extraction

**Config data to proxies**
- Pilot
- Mixer

**Check Report**
Traffic Splitting - Weight

Company service accesses to product service - Round Robin

95% of traffics go to current version of Product service
5% of traffics go to canary version of Product service

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: product
spec:
  hosts:
  - product
  http:
    - route:
        - destination:
            host: product

apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: product
spec:
  hosts:
  - product
  http:
    - route:
        - destination:
            host: product
          subset: v1
          weight: 95
        - destination:
            host: product
          subset: v2
          weight: 5
```
Traffic Splitting

Company service accesses to product service - Round Robin

 Traffics from Android go to current version of Prod service
 Traffics from iPhone go to canary version of Prod service

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: product
spec:
  hosts:
  - product
  http:
    - route:
      - destination:
        host: product
```

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: product
spec:
  hosts:
  - product
  http:
    - route:
      - destination:
        host: product
        subset: v1
    - match:
      - headers:
        x-request-platform:
          exact: android
```

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: product
spec:
  hosts:
  - product
  http:
    - route:
      - destination:
        host: product
        subset: v2
    - match:
      - headers:
        x-request-platform:
          exact: ios
```

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: product
spec:
  hosts:
  - product
  http:
    - route:
      - destination:
        host: product
        subset: v3
```

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: product
spec:
  hosts:
  - product
  http:
    - route:
      - destination:
        host: product
        subset: v4
```

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: product
spec:
  hosts:
  - product
  http:
    - route:
      - destination:
        host: product
        subset: v5
```

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: product
spec:
  hosts:
  - product
  http:
    - route:
      - destination:
        host: product
        subset: v6
```

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: product
spec:
  hosts:
  - product
  http:
    - route:
      - destination:
        host: product
        subset: v7
```
Security policies can be implemented at different levels of granularity - Service, Namespace, Mesh.

```
apiVersion: "authentication.istio.io/v1alpha1"
kind: "Policy"
metadata:
  name: company-auth
spec:
  targets:
    - name: company
  origins:
    - jwt:
      issuer: 192.168.99.100
      jwksUri: http://45.63.xxx.xxx/jwts
      principalBinding: USE_ORIGIN
```
Control Plane - Pilot
**Pilot Discovery**
- Lists/Watches platform-specified resources (Kubernetes: Nodes, Pods, Services, CRDs)
- Converts platform-specified configurations to Envoy-specified configurations, and propagates them to proxies at runtime.

**Pilot Agent**
- Initializes Envoy config
- Starts envoy
- Watches/Reloads envoy

**CRDs:**
- Gateway, VirtualService, ServiceEntry...
- Policy
- listchecker, listentry
- prometheus, metric
- authz, authorization
- rule
- …..
Control Plane - Mixer
**Mixer**

**Service A**

**Proxy**

**Check/Quota**
- **Rate Limitation**
  - Adapter: memquota, redisquota
  - Template: quota
  - Output: Redis
- **White/Black List**
  - Adapter: listchecker
  - Template: listentry
  - Output: List Provider
- **Authorization**
  - Adapter: authz
  - Template: authorization
  - Output: Authorization backend

**Report**
- **Log**
  - Adapter: fluentd
  - Template: logentry
  - Output: Fluentd
- **Metrics**
  - Adapter: prometheus
  - Template: metric
  - Output: Prometheus
- **...**

**Precondition check before request**

**Report after request**

**Check result**

**Fluentd**

**Prometheus**

**Report backends**
Policy - Whitelist

```yaml
apiVersion: config.istio.io/v1alpha2
kind: listchecker
metadata:
  name: whitelist
spec:
  overrides: ["user"] # overrides provide a static list
  blacklist: false
---
apiVersion: config.istio.io/v1alpha2
kind: listentry
metadata:
  name: appname
spec:
  value: source.labels["app"
---
apiVersion: config.istio.io/v1alpha2
kind: rule
metadata:
  name: checkname
spec:
  match: destination.labels["app" == "product"
  actions:
  - handler: whitelist.listchecker
    instances:
    - appname.listentry
```
Policy - Authorization

kind: CustomResourceDefinition
apiVersion: apiextensions.k8s.io/v1beta1
metadata:
  name: authzs.config.istio.io
  labels:
    package: authz
    istio: mixer-adapter
spec:
  group: config.istio.io
  names:
    kind: authz
    plural: authzs
    singular: authz
  scope: Namespaced
  version: v1alpha2
- Prometheus add-on
- Build-in Prometheus adapter
- Metrics
  - Istio-mesh metrics
  - Mixer-specific metrics
  - Proxy-specific metrics
Service: Propagates “B3” headers from incoming request to any outgoing request.
A little bit more
Mixer Attributes

```yaml
apiVersion: "config.istio.io/v1alpha2"
kkind: logentry
metadata:
  name: accesslog
  namespace: istio-system
spec:
  severity: "Default"
timestamp: request.time
variables:
  sourceIp: source.ip | ip("0.0.0.0")
  sourceHost: source.name | ""
  destinationIp: destination.ip | ip("0.0.0.0")
  destinationHost: destination.service.host | ""
  method: request.method | ""
  url: request.path | ""
  protocol: request.scheme | "http"
  responseCode: response.code | 0
  responseSize: response.size | 0
  latency: response.duration | "0ms"
  monitored_resource_type: "UNSPECIFIED"
```
Mixer Attributes

Kubernetes
Nodes, Namespaces, Pods, Services...

Kubernetes env adapter
Namespace: demo
Pod: product-v1-559cdf4744-wfj55
PodIP: 192.168.100.98
Service: product...

Source IP: 192.168.100.99
Destination IP: 192.168.100.98
Url: /product/api/v1/products
Response code: 200 ...

Resolution

logging adapter (fluentd)
sourceIp: 192.168.100.99
sourceHost: company-v1-559cdf4744-wfj55
destinationIp: 192.168.100.98
destinationHost: product-v1-559cdf4744-wfj55
url: /product/api/v1/products
responseCode: 200 ...

Proxy

Mixer

Attributes

Logging backend (Fluentd)
Proxy Init - Iptables

proxy_init: /usr/local/bin/istio-iptables.sh

Init Containers:

istio-init:

  Container ID: docker://a2cd253b021bd65b8a69fc
  Image: docker.io/yyyyyyyy/proxy_init:1.0.0-5
  Image ID: docker-pullable://docker.io/yyyyyyyy/proxy_init
  Port: <none>
  Args:
    -p
    15001  
    -u
    1337
    -m
    REDIRECT
    -i
    10.235.0.0/16
    -x
    -b
    9090,
    -d

iptables:

Inbound:

  iptables -t nat -N ISTIO_IN_REDIRECT
  iptables -t nat -A ISTIO_IN_REDIRECT -p tcp -j REDIRECT --to-port 15001

Outbound:

  iptables -t nat -N ISTIO_OUT_REDIRECT
  iptables -t nat -A ISTIO_OUT_REDIRECT -p tcp -j REDIRECT --to-port 15001
  iptables -t nat -N ISTIO_OUTPUT
  iptables -t nat -A ISTIO_OUTPUT -j ISTIO_OUT_REDIRECT
  iptables -t nat -A OUTPUT -p tcp -j ISTIO_OUTPUT
  iptables -t nat -A ISTIO_OUTPUT -m owner --uid-owner istio-proxy -j RETURN
Proxy admin

curl http://127.0.0.1:15000:

/admin: home page
/certs: print certs on machine
/clusters: upstream cluster status
/config_dump: dump current Envoy configs (experimental)
/cpuprofiler: enable/disable the CPU profiler
/healthcheck/fail: cause the server to fail health checks
/healthcheck/ok: cause the server to pass health checks
/help: print out list of admin commands
/hot_restart_version: print the hot restart compatibility version
/listeners: print listener addresses
/logging: query/change logging levels
/quitquitquit: exit the server
/reset_counters: reset all counters to zero
/runtime: print runtime values
/runtime_modify: modify runtime values
/server_info: print server version/status information
/stats: print server stats
/stats/prometheus: print server stats in prometheus format

curl -X POST http://localhost:15000/logging?filter=debug
• Cannot bypass JWT verification for specific APIs/Methods (health check, OPTIONS method)
• Request timeouts does not work (>15s) - Proxy
• Eventual consistency - Envoy TLS & RCU
• Takes 5~30 seconds to take effect new polices - Mixer
• Service unavailable (503) - Proxy
Thanks!

xi.he@hp.com
zhang.zhoujian@hp.com