Advanced Application Management Using Red Hat OpenShift Service Mesh

Traffic Management

Module Topics

- Traffic management
- Virtual Service
- Destination rule
- Gateway
- Service entry

Traffic Management

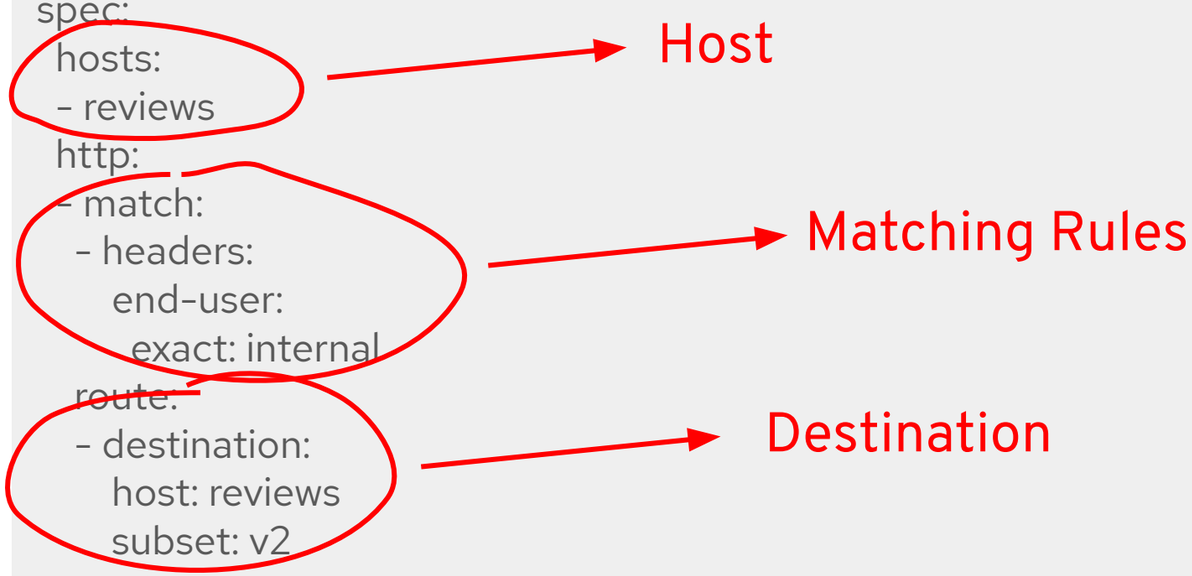
- Control the flow of traffic and API calls between services
- Service Mesh provides:
  - Traffic shifting - A/B, canary deployments
  - Load balancing policies
  - Policies for ingress and egress
- Custom Resource definitions:
  - VirtualService
  - DestinationRule
  - Gateway
  - ServiceEntry

Virtual Service

- Defines how requests are routed within the service mesh
- Decouples clients from destination workloads
- Consists of routing rules for traffic targeting one or more hosts
    - Host = Kubernetes service or external service
- Attached to a gateway for ingress or egress

Virtual Service

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: reviews
spec:
  hosts:
  - reviews
  http:
  - match:
    - headers:
        end-user:
          exact: internal
    route:
    - destination:
        host: reviews
        subset: v2
  - route:
    - destination:
        host: reviews
        subset: v3
```



Virtual Service

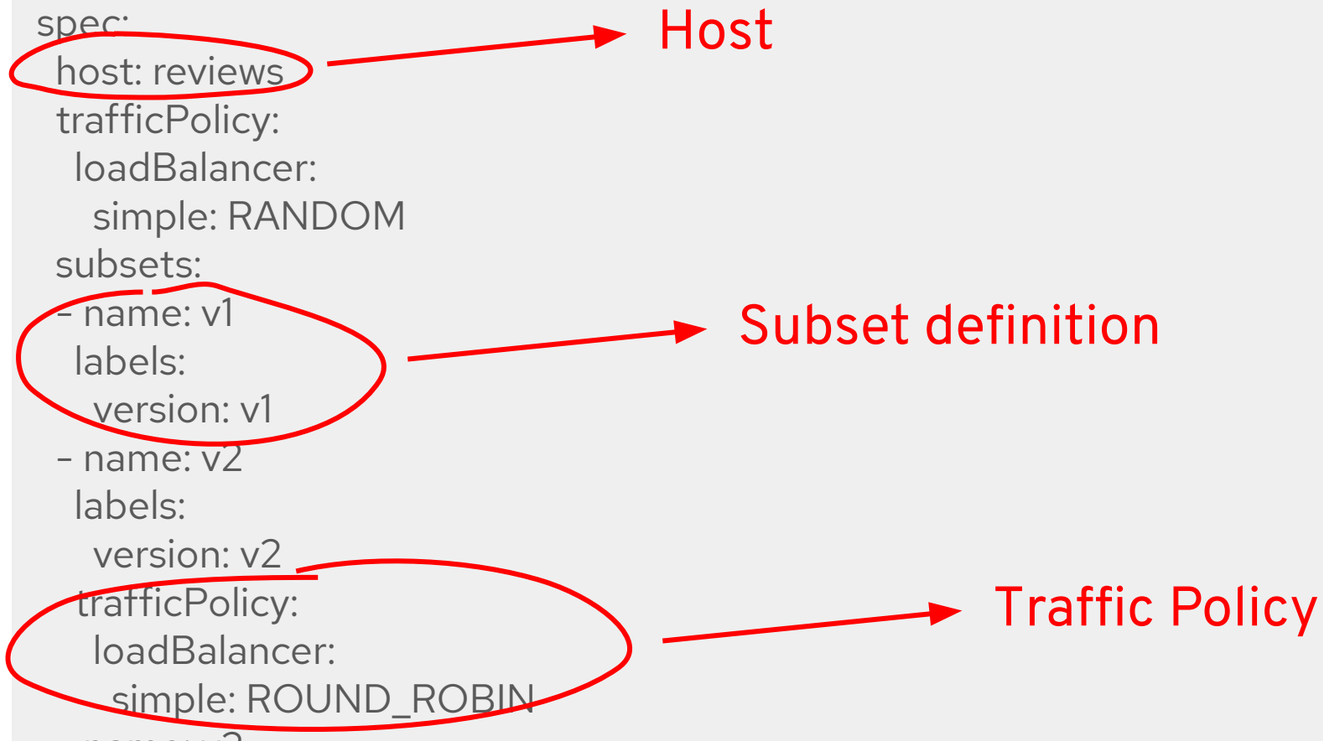
- Traffic types:
  - http : HTTP/1.1, HTTP/2, gRPC
  - tls: unterminated TLS traffic
  - tcp: TCP traffic
- Routing rules match conditions:
  - URI
  - Ports
  - Header values
  - HTTP scheme
  - HTTP method
  - Query parameters
  - Source namespace
  - Source labels
  - Gateway

Destination Rule

- Defines what happens to traffic for a destination
- Destination rules are applied after virtual service routing rules are evaluated
  - Apply to “real” destination.
- Defines subsets
- Configures traffic policies:
  - Load balancing policies
  - TLS settings
  - Connection pool size
  - Outlier detection

Destination Rule

```
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: my-destination-rule
spec:
  host: reviews
  trafficPolicy:
    loadBalancer:
      simple: RANDOM
  subsets:
  - name: v1
    labels:
      version: v1
  - name: v2
    labels:
      version: v2
    trafficPolicy:
      loadBalancer:
        simple: ROUND_ROBIN
  - name: v3
    labels:
      version: v3
```

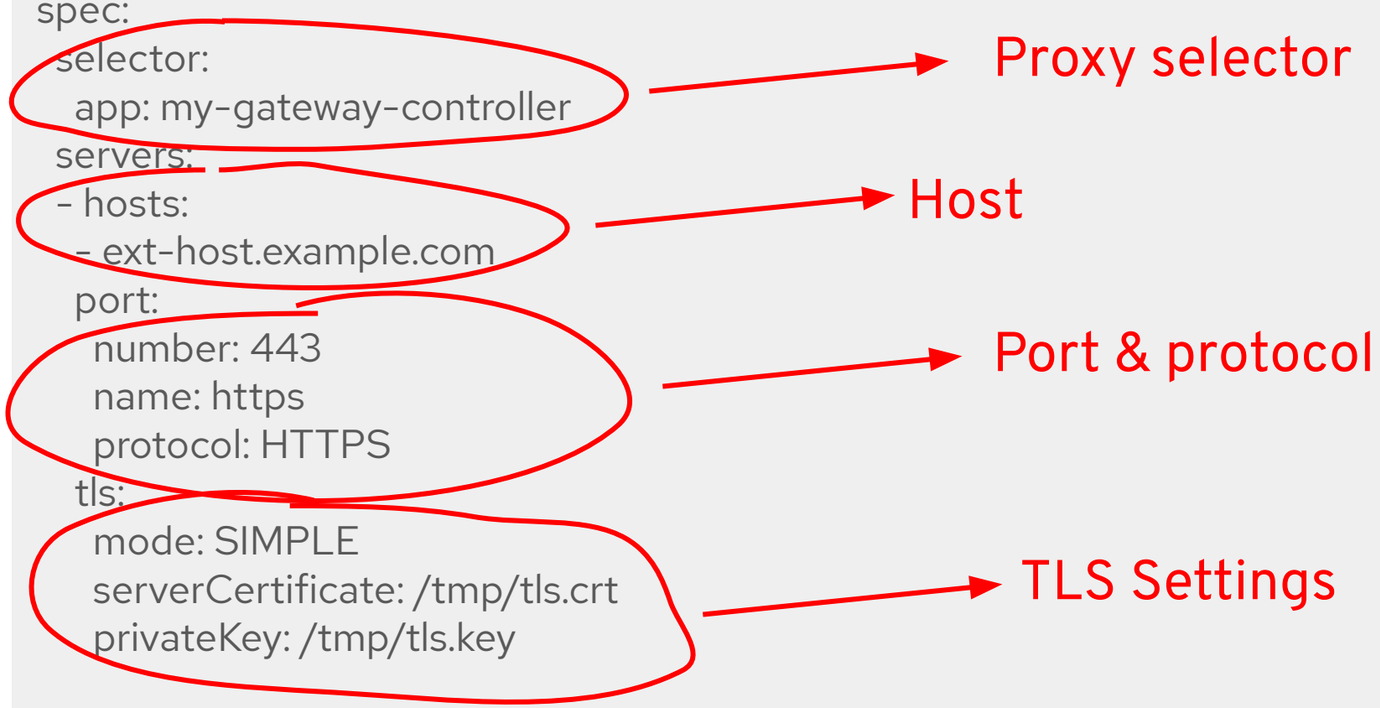


Gateway

- Manage inbound and outbond traffic for the service mesh
- Applied to standalone Envoy proxies at the edge of the mesh
- Defines
  - Ports & protocol
  - Host
  - TLS settings
- Bound to 1 or more virtual services

Gateway

```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: ext-host-gateay
spec:
  selector:
    app: my-gateway-controller
  servers:
  - hosts:
    - ext-host.example.com
    port:
      number: 443
      name: https
      protocol: HTTPS
    tls:
      mode: SIMPLE
      serverCertificate: /tmp/tls.crt
      privateKey: /tmp/tls.key
```

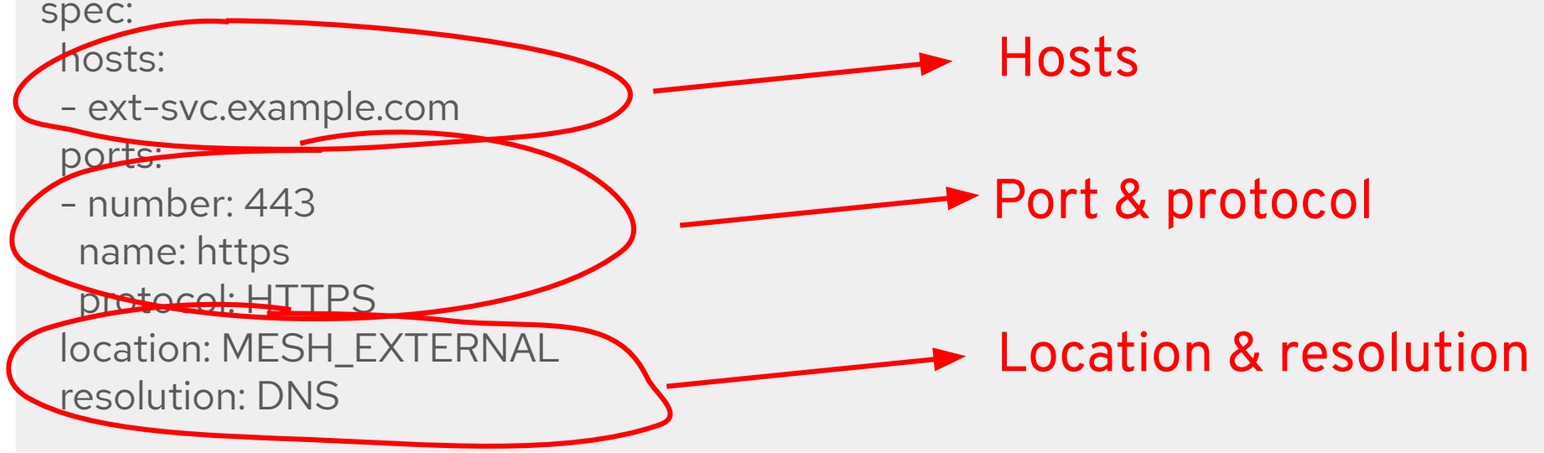


ServiceEntry

- Control plane maintains service registry
- Uses Kubernetes API to discover services inside the mesh
- ServiceEntry adds entry to the service registry
- Represents external service - outside the service mesh data plane
- Traffic to external service can be managed by control plane
    - Traffic management routing rules
    - Retry, timeout, fault injection

Service Entry

```
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: svc-entry
spec:
  hosts:
  - ext-svc.example.com
  ports:
  - number: 443
    name: https
    protocol: HTTPS
  location: MESH_EXTERNAL
  resolution: DNS
```



Module Summary

- Traffic management
- Virtual Service
- Destination rule
- Gateway
- Service entry