**ISTIO FILE SYSTEM**

istio-new/

├── README.md # This documentation file

├── install-istio.sh # Script to install Istio

├── gateway.yaml # Gateway configuration

├── virtual-service.yaml # Virtual service configuration

├── destination-rule.yaml # Destination rule configuration

└── values.yaml # Helm values for Istio installation

**Gateway.yaml**

*# This file defines how external traffic enters the Istio service mesh*

*apiVersion: networking.istio.io/v1alpha3*

*kind: Gateway*

*metadata:*

*name: my-gateway*

*namespace: istio-ingress # Namespace where the gateway will be deployed*

*spec:*

*selector:*

*istio: ingressgateway # Selects the default Istio ingress gateway pod*

*servers:*

*# HTTP Server Configuration*

*- port:*

*number: 80*

*name: http*

*protocol: HTTP*

*hosts:*

*- "\*" # Allows traffic for all hosts*

*# - specific hostnames: "example.com"*

*# - multiple hosts: ["host1.com", "host2.com"]*

*tls:*

*httpsRedirect: true # Redirects HTTP to HTTPS*

*# HTTPS Server Configuration*

*- port:*

*number: 443*

*name: https*

*protocol: HTTPS*

*hosts:*

*- "\*" # Allows traffic for all hosts*

*tls:*

*mode: SIMPLE # Enables TLS termination at the gateway*

*# Alternatives for mode:*

*# - MUTUAL: for mTLS (mutual TLS)*

*# - PASSTHROUGH: for TLS passthrough*

*# - AUTO\_PASSTHROUGH: automatic TLS detection*

*credentialName: my-cert # Reference to Kubernetes secret containing TLS cert*

*# Alternatives:*

*# - different secret names*

*# - multiple certificates for different hosts*

**Virtual Service.yaml**

*# This file defines routing rules, traffic splitting, and fault injection policies*

*# It controls how traffic is routed to different services in the mesh*

*apiVersion: networking.istio.io/v1alpha3*

*kind: VirtualService*

*metadata:*

*name: my-virtual-service*

*spec:*

*hosts:*

*- "\*" # Matches all hosts*

*# Alternatives:*

*# - specific hostnames: "api.example.com"*

*# - multiple hosts: ["host1.com", "host2.com"]*

*# - namespace specific: "service.namespace.svc.cluster.local"*

*gateways:*

*- my-gateway # References the gateway we created*

*# Alternatives:*

*# - mesh: for internal service-to-service communication*

*# - multiple gateways: ["gateway1", "gateway2"]*

*http:*

*# API Service Routing Rules*

*- match:*

*- uri:*

*prefix: /api/v1 # Matches all requests to /api/v1*

*# Alternatives for uri matching:*

*# - exact: "/api/v1/users" for exact matches*

*# - regex: ".\*\\.jpg$" for regex patterns*

*# - multiple conditions: [prefix, exact, regex]*

*route:*

*- destination:*

*host: api-service # Kubernetes service name*

*port:*

*number: 8080*

*# Alternatives:*

*# - different port numbers*

*# - subset: "v1" for version-specific routing*

*weight: 100 # Traffic weight for this destination*

*# Alternatives:*

*# - 0-100 for traffic splitting*

*# - multiple destinations with different weights*

*retries:*

*attempts: 3 # Number of retry attempts*

*# Alternatives:*

*# - 0: no retries*

*# - higher values for more retries*

*perTryTimeout: 2s # Timeout per retry attempt*

*# Alternatives:*

*# - different timeout values*

*# - could add retryOn conditions*

*timeout: 5s # Overall timeout for the request*

*# Alternatives:*

*# - different timeout values*

*# - 0s: no timeout*

*# Web Service Routing Rules*

*- match:*

*- uri:*

*prefix: /web # Matches all requests to /web*

*route:*

*- destination:*

*host: web-service # Kubernetes service name*

*port:*

*number: 80*

*weight: 100*

*fault:*

*delay:*

*percentage:*

*value: 0.1 # 0.1% of requests will be delayed*

*# Alternatives:*

*# - 0: no delay*

*# - higher values for more delays*

*fixedDelay: 5s # Delay duration*

*# Alternatives:*

*# - different delay values*

*# - exponentialDelay for exponential backoff*

*abort:*

*percentage:*

*value: 0.1 # 0.1% of requests will be aborted*

*# Alternatives:*

*# - 0: no aborts*

*# - higher values for more aborts*

*httpStatus: 500 # HTTP status code to return*

*# Alternatives:*

*# - different status codes*

*# - grpcStatus for gRPC services*

**DestinationRules.yaml**

*# This file defines policies for traffic to a service after routing has occurred*

*# It includes load balancing, connection pool, and outlier detection settings*

*apiVersion: networking.istio.io/v1alpha3*

*kind: DestinationRule*

*metadata:*

*name: my-destination-rule*

*spec:*

*host: api-service # Kubernetes service name*

*# Alternatives:*

*# - FQDN: "service.namespace.svc.cluster.local"*

*# - wildcard: "\*.example.com"*

*# Global traffic policy applied to all subsets*

*trafficPolicy:*

*loadBalancer:*

*simple: ROUND\_ROBIN # Load balancing algorithm*

*# Alternatives:*

*# - LEAST\_CONN: least connection*

*# - RANDOM: random selection*

*# - PASSTHROUGH: no load balancing*

*# - CONSISTENT\_HASH: for session affinity*

*connectionPool:*

*tcp:*

*maxConnections: 100 # Maximum number of connections*

*# Alternatives:*

*# - different max connection values*

*http:*

*http1MaxPendingRequests: 1024 # Maximum pending HTTP requests*

*# Alternatives:*

*# - http2MaxRequests for HTTP/2*

*maxRequestsPerConnection: 10 # Maximum requests per connection*

*# Alternatives:*

*# - 0: unlimited requests*

*outlierDetection:*

*consecutive5xxErrors: 5 # Number of 5xx errors before ejection*

*# Alternatives:*

*# - different error thresholds*

*# - consecutiveGatewayErrors*

*# - consecutiveLocalOriginFailures*

*interval: 30s # Time between ejection analysis*

*# Alternatives:*

*# - different interval values*

*baseEjectionTime: 30s # Minimum ejection duration*

*# Alternatives:*

*# - different base ejection times*

*maxEjectionPercent: 10 # Maximum % of hosts that can be ejected*

*# Alternatives:*

*# - different max ejection percentages*

*# - 0: no ejection*

*# Subset definitions for different versions*

*subsets:*

*- name: v1 # Subset name*

*labels:*

*version: v1 # Pod label selector*

*trafficPolicy:*

*loadBalancer:*

*simple: ROUND\_ROBIN # Subset-specific load balancing*

*# Alternatives:*

*# - different load balancing algorithms*

*# - could override other traffic policies*

*- name: v2 # Another subset for version 2*

*labels:*

*version: v2*

*trafficPolicy:*

*loadBalancer:*

*simple: ROUND\_ROBIN*

## **Google Example**

**ServiceEntry.yaml**

#What This Does:

First ServiceEntry (allow-egress-googleapis)

Allows pods to call:

accounts.google.com (for OAuth/login tokens).

Any Google API (\*.googleapis.com).

Supports both HTTP (80) and HTTPS (443).

Second ServiceEntry (allow-egress-google-metadata)

Allows access to Google Cloud’s metadata server (used for VM identity, secrets, etc.).

Works via:

DNS (metadata.google.internal)

IP (169.254.169.254)

# --- PART 1: ALLOW GOOGLE APIS (HTTPS/HTTP) ---

apiVersion: networking.istio.io/v1alpha3

kind: ServiceEntry # Defines external services Istio can access

metadata:

name: allow-egress-googleapis # Name of this rule

spec:

hosts: # Which external domains to allow

- "accounts.google.com" # For OAuth/login tokens

- "\*.googleapis.com" # All Google APIs (like Cloud Storage, etc.)

ports: # Which ports to allow

- number: 80 # HTTP

protocol: HTTP

name: http

- number: 443 # HTTPS (secure)

protocol: HTTPS

name: https

# --- PART 2: ALLOW GCE METADATA SERVER ---

apiVersion: networking.istio.io/v1alpha3

kind: ServiceEntry

metadata:

name: allow-egress-google-metadata # Name of this rule

spec:

hosts:

- metadata.google.internal # Google Cloud metadata server (internal DNS)

addresses: # Also allow by IP (for GCE)

- 169.254.169.254 # Standard GCE metadata server IP

ports: # Allowed ports

- number: 80 # HTTP

name: http

protocol: HTTP

- number: 443 # HTTPS

name: https

protocol: HTTPS

**FrontendGateway.yaml**

# This file does TWO things:

# 1. Defines a GATEWAY (entry point for external traffic)

# 2. Sets up routing rules for external traffic to the frontend.

# Gateway (frontend-gateway)

Opens port 80 for external traffic (like a public door into your cluster).

Uses Istio’s default ingress gateway (the component that handles incoming traffic).

Accepts traffic from any domain (\*).

# VirtualService (frontend-ingress)

Tells Istio:

"For all external traffic (\*), send it to the frontend service on port 80."

Attaches to the frontend-gateway (so it only affects external traffic).

# --- PART 1: GATEWAY CONFIG ---

apiVersion: networking.istio.io/v1alpha3 # Istio networking API

kind: Gateway # Defines an entry point for external traffic

metadata:

name: frontend-gateway # Name of this Gateway

spec:

selector:

istio: ingressgateway # Uses Istio's default ingress controller

servers: # List of ports and hosts this Gateway accepts

- port: # Port configuration

number: 80 # Listen on port 80 (HTTP)

name: http # Name for reference

protocol: HTTP # Protocol (HTTP, HTTPS, etc.)

hosts: # Which domains this applies to

- "\*" # Accepts ANY domain (wildcard)

# --- PART 2: VIRTUAL SERVICE FOR EXTERNAL TRAFFIC ---

apiVersion: networking.istio.io/v1alpha3

kind: VirtualService # Routing rules for external traffic

metadata:

name: frontend-ingress # Name of this VirtualService

spec:

hosts: # Which domains this applies to

- "\*" # Applies to ALL incoming domains

gateways: # Which Gateway this rule is attached to

- frontend-gateway # Uses the Gateway defined above

http: # HTTP traffic rules

- route: # Where to send traffic

- destination: # Target service

host: frontend # Kubernetes service name

port:

number: 80 # Port 80

**Frontend.yaml**

# This configures INTERNAL traffic routing for the "frontend" service.

# It tells Istio how to route requests INSIDE the cluster.

# When another service inside the cluster calls frontend.default.svc.cluster.local, Istio routes it to the frontend service on port 80.

# This is internal-only—it doesn’t affect external traffic.

apiVersion: networking.istio.io/v1alpha3 # Istio networking API version

kind: VirtualService # Defines routing rules for a service

metadata:

name: frontend # Name of this VirtualService

spec:

hosts: # Which hostnames this rule applies to

- "frontend.default.svc.cluster.local" # Matches the internal DNS name of the frontend service

http: # HTTP traffic rules

- route: # Where to send traffic

- destination: # Defines the target service

host: frontend # Service name (Kubernetes Service)

port:

number: 80 # Port to forward traffic to

**Commands**

# 🧬 Enable automatic sidecar injection in a namespace

kubectl label namespace <namespace> istio-injection=enabled

# 🧼 Disable sidecar injection

kubectl label namespace <namespace> istio-injection-

# 📍 Access Kiali UI (dashboard)

istioctl dashboard kiali

# 📊 Access Grafana (metrics)

istioctl dashboard grafana

# 🔍 Access Jaeger (tracing)

istioctl dashboard jaeger

# 📈 Access Prometheus

istioctl dashboard prometheus

# Check sidecar is injected

kubectl get pod -n mynamespace -o jsonpath='{.items[\*].spec.containers[\*].name}'