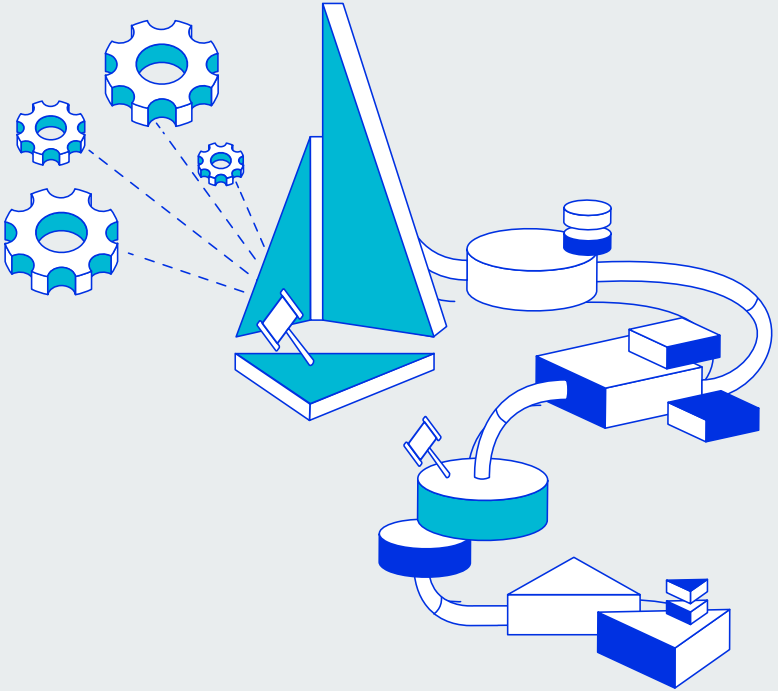
Securing Cloud Native Workloads With Istio



Gufran Mirza
gufranmirza

Talk Outline

- Introduction to Istio
- Service Identities
- Istio Authentication Policies
- Istio Authorization Policies
- Q & A

Istio - Ιστίο

Open source service mesh

• • •

What is a service mesh

- Infrastructure/framework that handles communication between services
- Often implemented as network proxies deployed alongside the micro-services

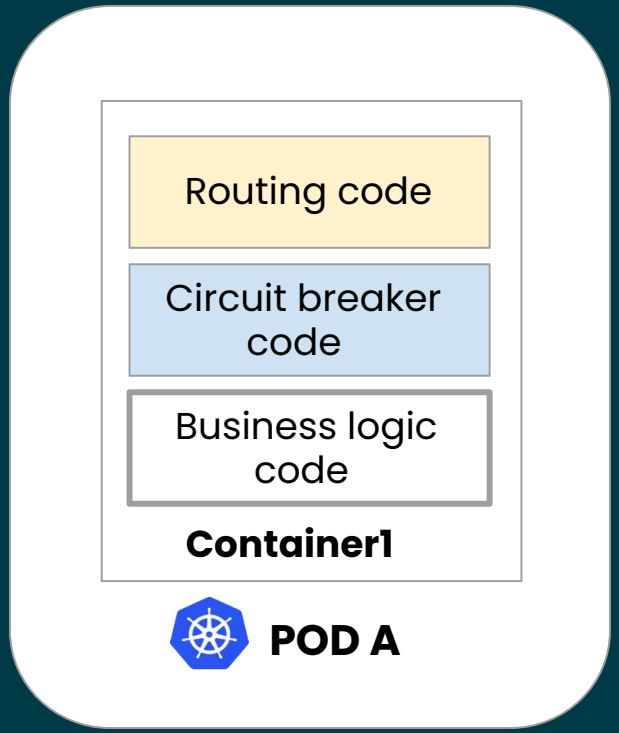
Istio features

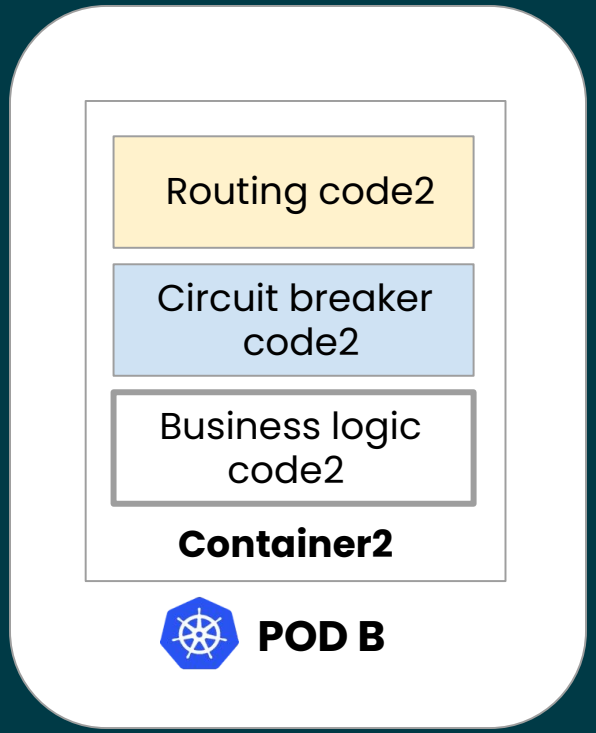
- Load balancing (HTTP, gRPC, TCP...)
- Traffic control (routing rules, retries, timeouts, fault injection, mirroring)
- **Secure service-to-service communication**
- **Access controls (authorization)**
- **Metrics and traces for traffic**

Important Terminology

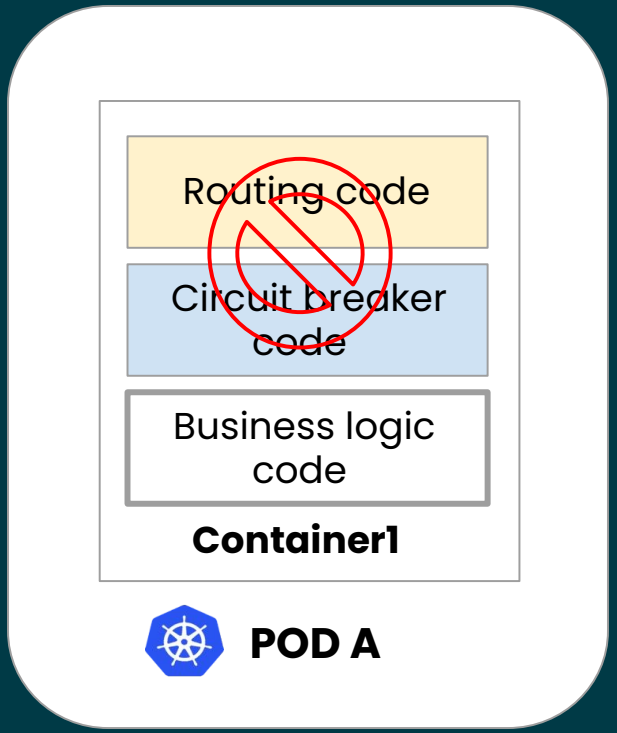
- Workload - anything owning/controlling pods (like a Deployment) or the pods themselves
- Service - a **micro-service**
- Application - *label* “app” on a pod/service
- Version - *label* “version” on a pod/service

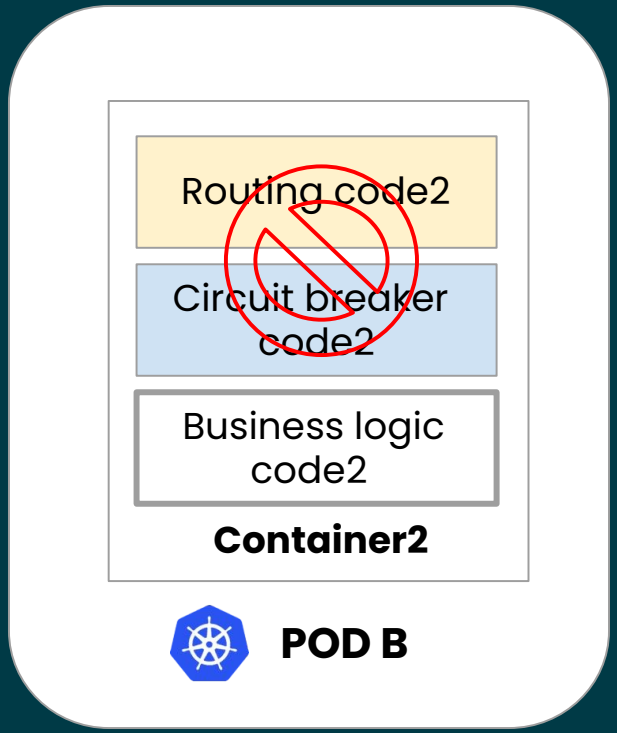
Before Istio





Before Istio





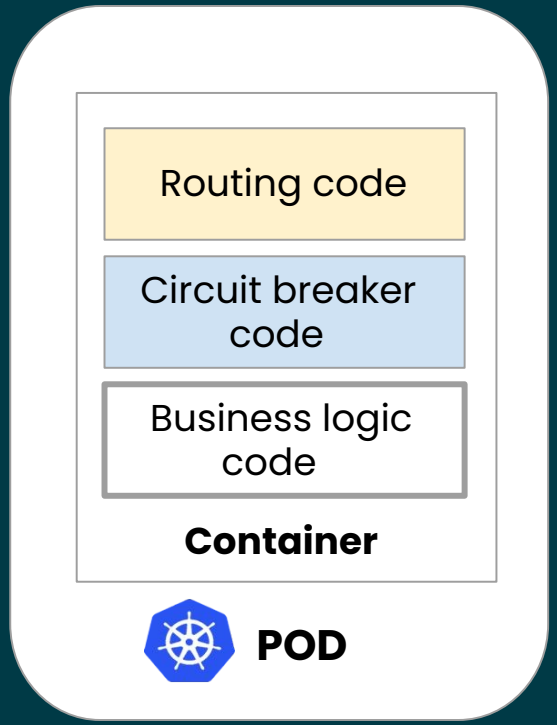
Sidecar Proxy

- A proxy is deployed in a container next to each instance of micro-service (inside a pod)
- Container name: istio-proxy
- It is **transparent** to application code
- Envoy open source proxy is currently used

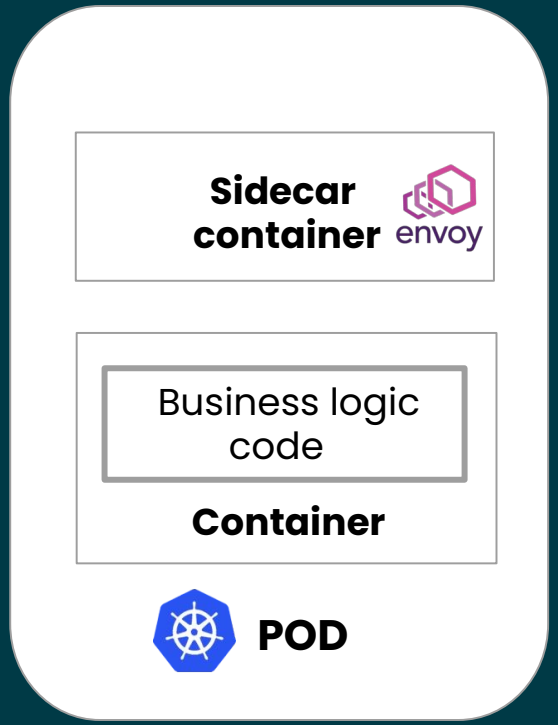
How is the sidecar injected?

- Manually
- Automatically injected to pod on creation
  - kubectl label namespace default istio-injection=enabled
  - Mutating Admission Webhook is used for sidecar injection
  - Actually… 2 containers are injected: istio-init and istio-proxy

Sidecar Proxy in Istio

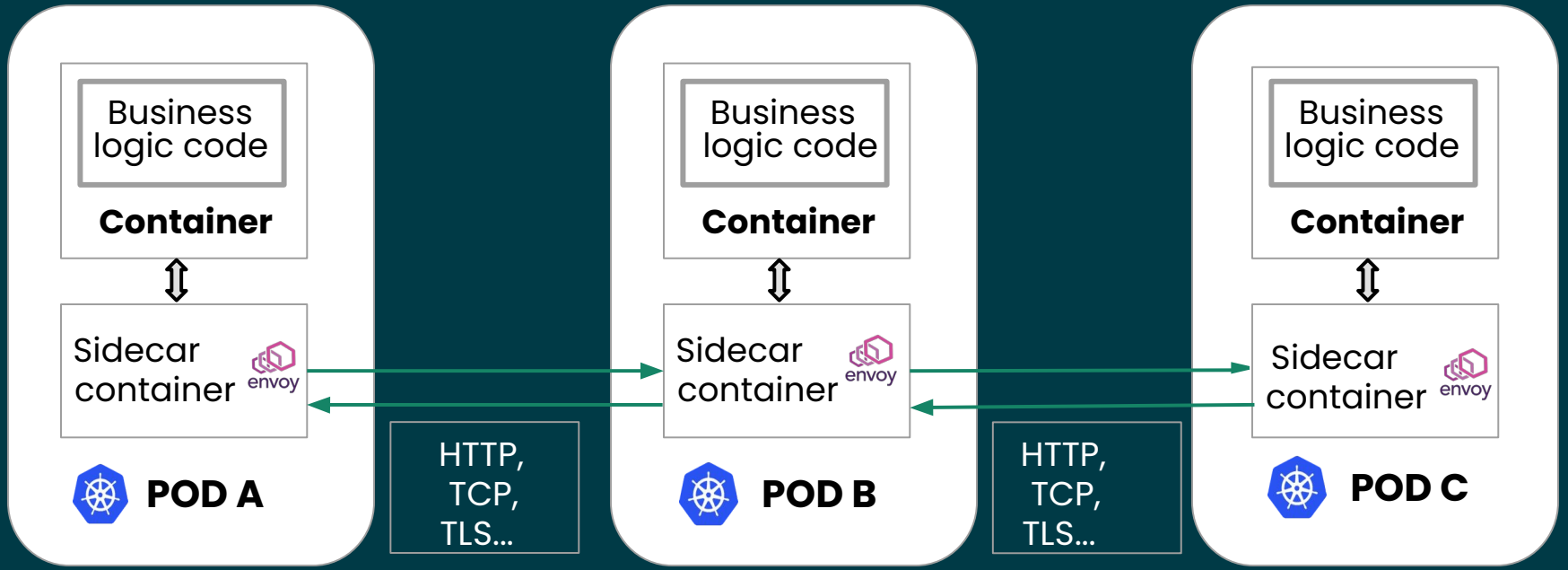


No sidecar



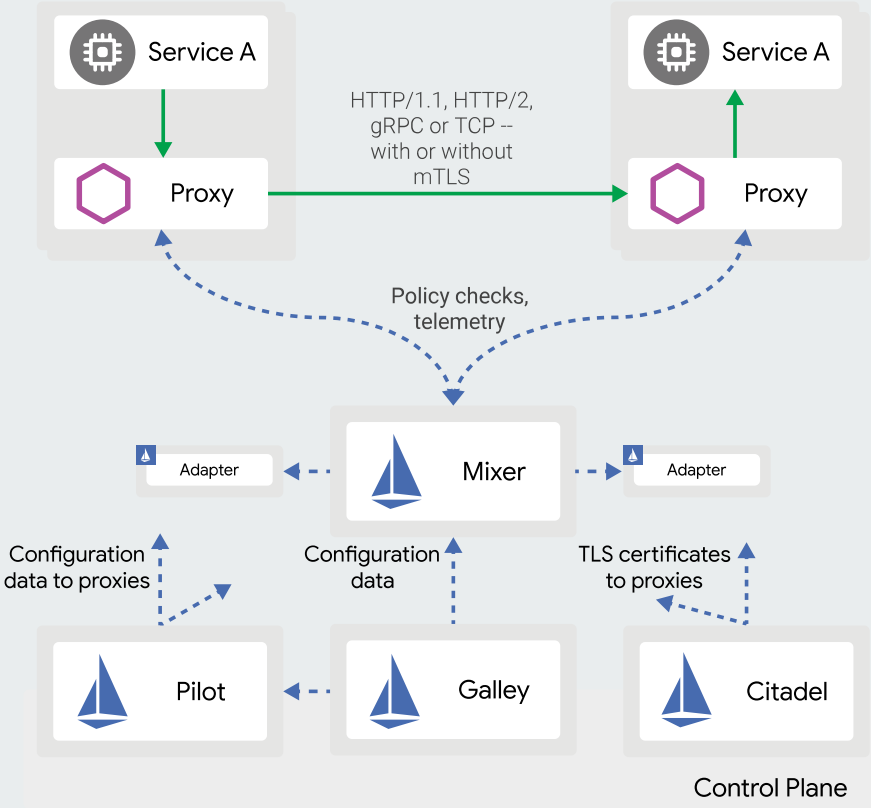
With sidecar

With Istio - sidecar intercepts all traffic



Configuration is transparent to the services and not part of the code

Istio architecture



Service Mesh Security

Authorization & Authentication

• • •

Service Identities – The starting point

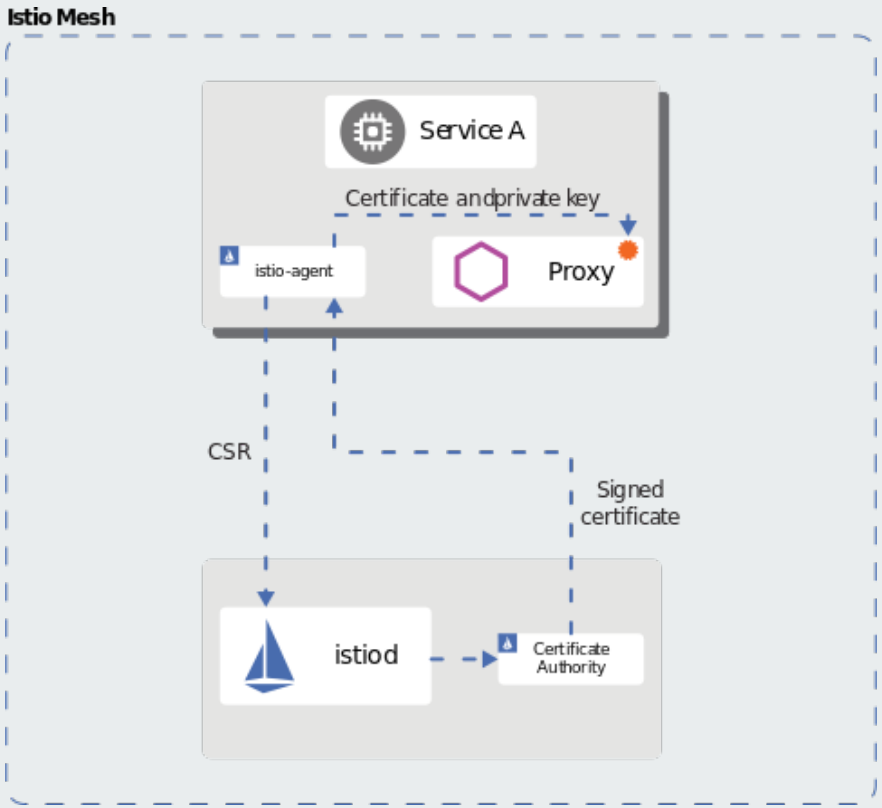
In a service mesh world, establishing the identity of the workload providing a service is critical. Examples:

- **Kubernetes**: Kubernetes service account
- **GCP**: GCP service account
- **AWS**: AWS IAM user/role account
- **On-premises (non-Kubernetes)**: user account, custom service account, service name, Istio service account, or GCP service account.

Conversion of identity into a certificate

- A private key within the workload pod is generated and Made available to the proxy.
- A certificate signing request is sent to the control plane.
- The control plane provides the proxy a certificate scoped to the identity of the POD (e.g. K8s service-account).
- Control plane will manage rotation.

Identity Provisioning Workflow

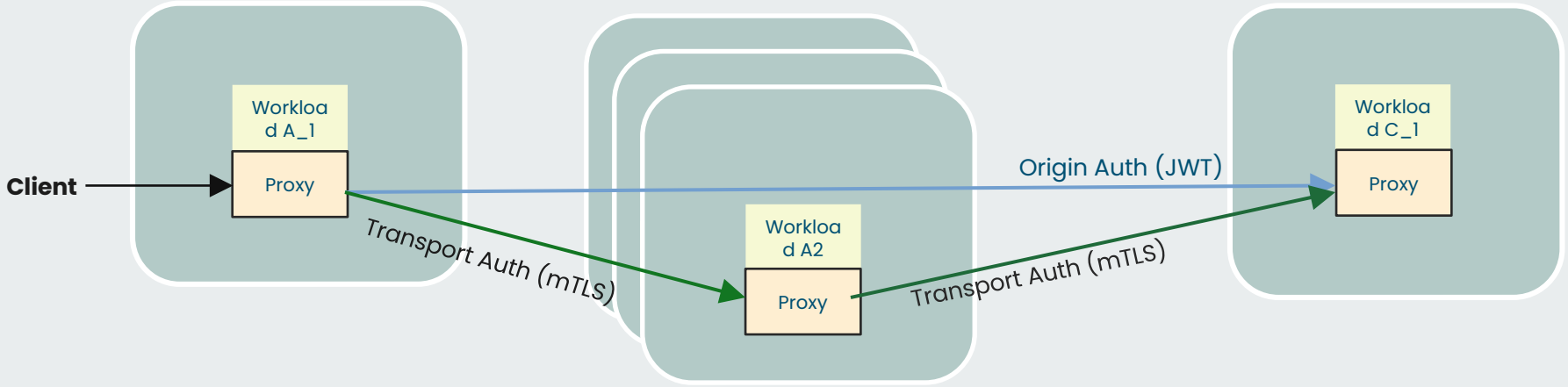


Authentication

Istio provides two types of authentication

- End user authentication (JSON Web Token (JWT))
- Service to service authentication (mutual TLS)
  - PERMISSIVE: Workloads accept both mutual TLS and plain text traffic
  - STRICT: Workloads only accept mutual TLS traffic.
  - DISABLE: Mutual TLS is disabled

Authentication Flow



You can specify authentication requirements for workloads receiving requests in an Istio mesh using peer and request authentication policies

Peer authentication

The following peer authentication policy requires all workloads in namespace foo to use mutual TLS:

```yaml
apiVersion: security.istio.io/v1beta1
kind: PeerAuthentication
metadata:
  name: "example-policy"
  namespace: "foo"
spec:
  mtls:
    mode: STRICT # Workloads only accept mutual TLS traffic
```

Ingress Gateway

Defining a Gateway ingress/egress to enable traffic in/out of mesh with TLS/mTLS

- Citadel monitors service accounts creation and creates a certificate for them
  - Certificates only in memory, sent to Envoy via SDS API
- mTLS can be defined on multiple levels
  - Client and server exchange certificates, 2 way
  - All mesh, specific service, etc.

Configuration YAML

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: mygateway
spec:
  selector:
    istio: ingressgateway # use istio default ingress gateway
  servers:
  - port:
      number: 443
      name: https
      protocol: HTTPS
    tls:
      mode: SIMPLE
      credentialName: httpbin-credential # must be the same as secret
    hosts:
    - httpbin.example.com
```

Authentication Demo

A picture is worth a thousand yamls

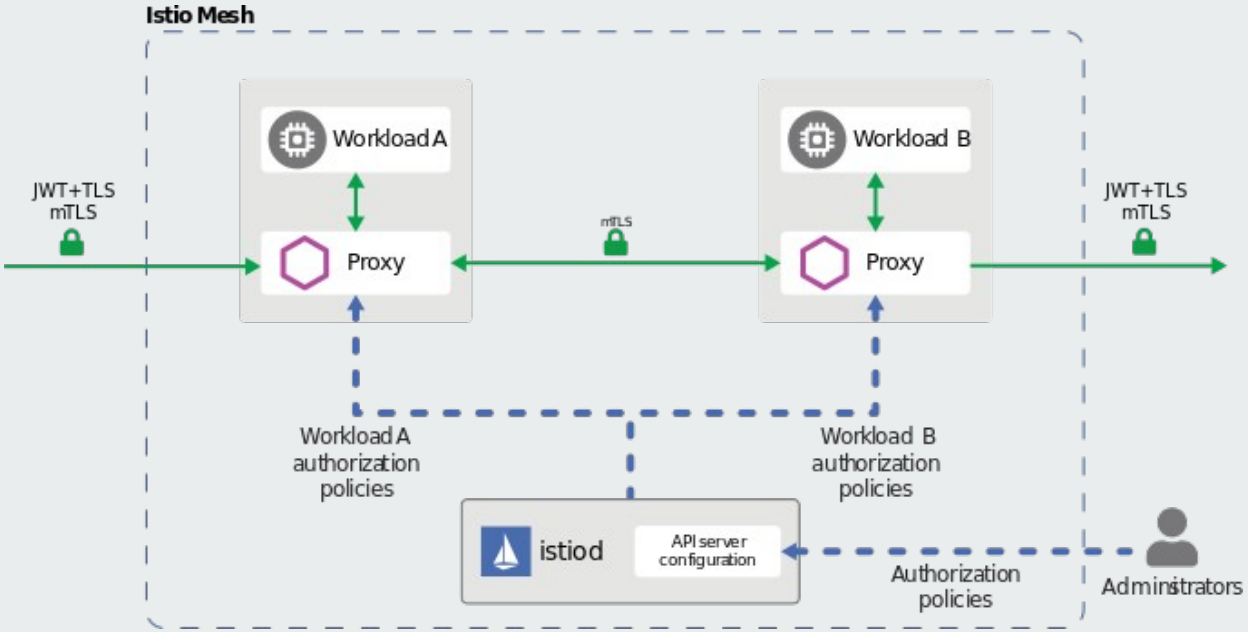
• • •

Authorization

Istio's authorization features provide mesh-, namespace-, and workload-wide access control for your workloads in the mesh

- Can service <A> send <this request> to service <B> ?
- Authorization policies support ALLOW, DENY and CUSTOM actions
- Istio authorization (ALLOW and DENY) is enforced natively on Envoy
- It is a good security practice to start with the allow-nothing policy and incrementally add more ALLOW policies to open more access to the workload.

Authorization Flow



Each Envoy proxy runs an authorization engine that authorizes requests at runtime. Authorization engine evaluates the request context and returns the authorization result, either ALLOW or DENY

Authorization Policy

```yaml
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
 name: httpbin
 namespace: foo
spec:
 selector:
   matchLabels:
     app: httpbin
     version: v1
 action: ALLOW
 rules:
 - from:
   - source:
       principals: ["cluster.local/ns/default/sa/sleep"]
   to:
   - operation:
       methods: ["GET"]
```

authorization policy that allows the cluster.local/ns/default/sa/sleep service account to access the workloads with the app: httpbin

Authorization Demo

A picture is worth a thousand yamls

• • •

Q & A

Connect with the community

Istio.io

IstioMesh

github.com/istio

Thank You

Securing Cloud Native Workloads With Istio

Gufran Mirza
gufranmirza