Service Mesh from the Ground Up: How Istio Can Transform Your Organization

Megan O'Keefe
oreilly.com/sacon | #OReillySACon
Hello!

☁️ I'm a Developer Relations Engineer at Google Cloud.

💕 I help make Google's products easy to adopt and use.

💻 I test-drive new features, build demos/tools/workshops, and talk to end-users.

☸️ I work on: Kubernetes, Service Mesh, and Anthos.
Today's Goals

The world of distributed applications
Why use a service mesh?
Istio feature tour
Live demos!
Q&A
Why Service Mesh?
The increasing adoption of containers, microservices, and hybrid cloud deployments has created more distributed applications than ever.
Distributed apps can be defined as a collection of services.
What is a Service?

A **Service** is one deployable unit of software. A Service implements a specific set of **business logic**, and is often owned by one team.

A Service can run and **scale** independently from its dependencies.

A Service can be **small** or **large**.

A Service can be **stateless** or **stateful**.
Services: Benefits

- Separation of concerns
- Abstract away infrastructure
- Faster deployments
- Scale independently
- Cost savings
How do your **developers** and **operators** keep things up and running with explosive growth in number of services?
By thinking **services first**: investing in **automation**, **tools**, and **cultural change**.
This is not easy.
Services: Challenges

- More languages, client libraries
- Choosing an environment
- Lifecycling Applications
- Scaling to demand
- Resource optimization
What can Kubernetes do?

- Multitenancy, Isolation
- Abstract away compute
- Keep containers alive
- Automated scaling
- Optimize resources
Kubernetes runs **Pods (Workloads)** in a **Cluster**.

A **Cluster** = a set of Virtual Machines (**Nodes**)
Cluster

- master
- node
- node
- node
Pods in a Cluster
Pods in a Cluster

you → master → Pods
apiVersion: apps/v1
class: Deployment
metadata:
  name: hello-world
spec:
  replicas: 1
template:
  metadata:
    labels:
      app: hello-world
spec:
  containers:
  - name: hello-world-server
    image: gcr.io/megangcp/helloworld:v0.0.1
    ports:
    - containerPort: 8080
apiVersion: v1
kind: Service
metadata:
  name: helloworld
spec:
  selector:
    app: hello-world
  ports:
  - name: http
    protocol: TCP
    port: 80
    targetPort: 8080
  type: LoadBalancer
kubectl apply -f deployment.yaml

deployment.extensions/hello-world created
kubectl get pods

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>hello-world-84c646556b-kn59b</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
</tr>
</tbody>
</table>
```bash
kubectl get svc
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>CLUSTER-IP</th>
<th>EXTERNAL-IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>helloworld</td>
<td>LoadBalancer</td>
<td>10.51.246.3</td>
<td>35.188.110.209</td>
</tr>
</tbody>
</table>
➜ curl http://35.188.110.209

Hello world!
But...
Where does Kubernetes fall short?

- Safe Rollouts
- Observability
- Traffic Encryption
- Request-level Authorization
- Resilience
What is a Service Mesh?

A **transparent layer** on top of your services.

A way to make the **network** aware of application protocols like HTTP and gRPC.

An **observability** tool

A **security** tool.
Why use a Service Mesh?

- Decouple Dev from Ops
- Separate applications from infrastructure
- Get generated metrics without instrumenting your services
- Manage security policies in one place
- Modify traffic flow without changing app code
What is Istio?
Istio

An open-source service mesh tool to manage service interactions across container and VM-based services.

Created by Google, IBM, and Lyft in 2017

Runs on Kubernetes

Works at the application layer (Layer 7: HTTP, gRPC)

Today: 300+ organizations contributing
Istio
Connect, secure, control, and observe services.

Connect
Intelligently control the flow of traffic and API calls between services, conduct a range of tests, and upgrade gradually with red/black deployments.

Secure
Automatically secure your services through managed authentication, authorization, and encryption of communication between services.

Control
Apply policies and ensure that they're enforced, and that resources are fairly distributed among consumers.

Observe
See what's happening with rich automatic tracing, monitoring, and logging of all your services.
What Does Istio Do?

- Observability
- Network Automation
- Security
What Does Istio Do?

- Telemetry for every service
- Logs for all traffic
- Service graph
What Does Istio Do?

- Telemetry for every service
- Logs for all traffic
- Service graph
- Safe rollouts with traffic splitting
- Client-side load balancing
- Timeouts, retry, circuit-breaking
What Does Istio Do?

- Telemetry for every service
- Logs for all traffic
- Service graph

- Safe rollouts with traffic splitting
- Client-side load balancing
- Timeouts, retry, circuit-breaking

- Encryption in transit
- Service identity, authentication
- Authorization
Who is Istio for?

**Infrastructure Operators:** Monitor traffic across clusters and regions, add failover

**Platform Engineers:** Build CI/CD tools for app developers, migrate legacy services

**App Developers:** Investigate service metrics and behavior, debug during outages

**Security Admins:** Enforce authentication and authorization policies

**Quality Assurance:** Mirror production traffic to a test environment
Istio Partners

IBM Cloud
Solo
AspenMesh
Envoy
Knative
Cisco
Datadog
WeaveWorks
Palo Alto Networks

more at: istio.io/about/community/partners/ | image source: Datadog
Istio a Game Changer for HP's FitStation Platform

How HP is building its next-generation footwear personalization platform on Istio

BY STEVEN CEUPPENS, CHIEF SOFTWARE ARCHITECT @ HP FITSTATION, OPEN SOURCE ADVOCATE & CONTRIBUTOR | JULY 31, 2018 | 2 MINUTE READ

This blog post was written assuming Istio 1, so some of this content may now be outdated.

The FitStation team at HP strongly believes in the future of Kubernetes, BPF and service-mesh as the next standards in cloud infrastructure. We are also very happy to see Istio coming to its official Istio 1.0 release – thanks to the joint collaboration that started at Google, IBM and Lyft beginning in May 2017.

more at: istio.io/about/community/customers/
Case Study: Autotrader

Adopting Kubernetes led to a 75 percent reduction in compute resources.

Adopting Istio led to improved security and visibility, with no extra developer effort or training needed.

Istio's service metrics improved visibility across a large microservices architecture.

source: Google Cloud Blog
How Istio Works
Node

Agent

Service A

Proxy

Node

Agent

Service B

Proxy

Policy checks and telemetry

Discovery & config data to proxies

Mixer

TLS certs to node agents

Pilot

Galley

Citadel

Mesh config to control plane

TLS certs to proxies via Secrets

YAML

Sidecar configuration to Pods

Istio Control Plane
Node

Agent

Service A

Proxy

Service B

Proxy

Node

Agent

Pilot

Galley

Citadel

Mixer

Policy checks and telemetry

Discovery & config data to proxies

TLS certs to proxies via Secrets

TLS certs to node agents

Mesh config to control plane

Sidecar configuration to Pods

YAML

Istio Control Plane

Injectors
Discovery & config data to proxies

Policy checks and telemetry

Mesh config to control plane

Sidecar configuration to Pods
Node

**Agent**

**Service A**

**Proxy**

Node

**Agent**

**Service B**

**Proxy**

---

**Discovery & config data to proxies**

**Mesh config to control plane**

**Sidecar configuration to Pods**

**Policy checks and telemetry**

**YAML**

**Mixer**

**Pilot**

**Galley**

**Citadel**

**Injector**

---

**TLS certs to node agents**

**TLS certs to proxies via Secrets**

---

**Istio Control Plane**
Node

Agent

Service A

Proxy

Node

Agent

Service B

Proxy

Discovery & config data to proxies

Policy checks and telemetry

Mesh config to control plane

Sidecar configuration to Pods

TLS certs to node agents

TLS certs to proxies via Secrets

YAML

Pilot

Galley

Citadel

Injectors

Istio Control Plane
Node

Agent

Service A

Proxy

Node

Agent

Service B

Proxy

Pilot

Galley

Citadel

Injectors

Mesh config to control plane

Sidecar configuration to Pods

Discovery & config data to proxies

Policy checks and telemetry

TLS certs to node agents

TLS certs to proxies via Secrets

Istio Control Plane

YAML

Policy checks and telemetry
Node

**Agent**

![Service A]

- Proxy

Node

**Agent**

![Service B]

- Proxy

**Discovery & config data to proxies**

**Policy checks and telemetry**

**Mixer**

- **Pilot**
  - **Galley**
  - **Citadel**
    - **Inject**

- **Sidecar configuration to Pods**

**TLS certs to proxies via Secrets**

**TLS certs to node agents**

**YAML**

**Istio Control Plane**
Node

- Agent

Service A

- Proxy

Service B

- Proxy

TLS certs to proxies via Secrets

Policy checks and telemetry

Discovery & config data to proxies

Mixer

Pilot

Galley

Citadel

Mesh config to control plane

Injector

Sidecar configuration to Pods

YAML

Istio Control Plane
Installing Istio
Demo
Questions?
Observability
Observability is a measure of how well internal states of a system can be inferred from knowledge of its external outputs.
Istio Observability Features

**Service graph** - track dependencies at **runtime**

Bird's eye view of service behavior for issue triage, reduce time to **detect and fix outages**

Automatically collects the "golden signals" for every service - **latency, error rate, throughput**

Set, monitor and enforce **Service-Level Objectives (SLOs)**

**Tracing:** track a request from end to end, across service boundaries
Demo
Security
Moving from VMs to Kubernetes introduces new security challenges.
<table>
<thead>
<tr>
<th>Virtual Machines</th>
<th>Kubernetes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isolation at the <strong>host level</strong></td>
<td>Containers <strong>share</strong> a host (Node)</td>
</tr>
<tr>
<td>Workloads allocated to hosts</td>
<td>Nodes work as <strong>one</strong> virtual host</td>
</tr>
<tr>
<td>Workloads <strong>share</strong> OS, dependencies</td>
<td>Containers have <strong>own dependencies</strong></td>
</tr>
<tr>
<td><strong>Stable</strong> host IPs</td>
<td><strong>Ephemeral</strong> Pod IPs</td>
</tr>
<tr>
<td>May run in a <strong>trusted</strong>, on-prem environment</td>
<td>May run in a <strong>cloud</strong> environment</td>
</tr>
</tbody>
</table>
Istio - Security

Automatically secure your services through managed authentication, authorization, and encryption of communication between services.

- Traffic encryption
- Service auth
- Auditing controls
- Access policies
Demo: Mutual TLS
Node

- Agent
- Service A
  - Proxy

Node

- Agent
- Service B
  - Proxy

Node

- Pilot
- Galley
- Citadel
- Mixer

YAML

- Mesh config to control plane
- Sidecar configuration to Pods
- Discovery & config data to proxies
- TLS certs
- TLS certs to node agents
- Policy checks and telemetry
MeshPolicy

apiVersion: "authentication.istio.io/v1alpha1"
kind: "MeshPolicy"
metadata:
  name: "default"
spec:
  peers:
    - mtls: {}/
DestinationRule

apiVersion: "networking.istio.io/v1alpha3"
kind: "DestinationRule"
metadata:
  name: "default"
  namespace: "istio-system"
spec:
  host: "*.local"
  trafficPolicy:
    tls:
      mode: ISTIO_MUTUAL
Demo: Authorization
Node

Agent

Service A

Proxy

Service B

Proxy

Node

Agent

TLS certs to proxies via Secrets

Pilot

Galley

Citadel

Mesh config to control plane

TLS certs to node agents

YAML

Pilot

Galley

Citadel

Inject

Sidecar configuration to Pods

Istio Control Plane

Discovery & config data to proxies

Policy checks and telemetry

TLS certs to node agents

Mesh config to control plane

Sidecar configuration to Pods

YAML
AuthorizationPolicy

apiVersion: "security.istio.io/v1beta1"
kind: "AuthorizationPolicy"
metadata:
    name: "currency-policy"
    namespace: default
spec:
    selector:
        matchLabels:
            app: currency-service
    rules:
        - from:
            - source:
                principals: ["cluster.local/ns/default/sa/frontend-sa"]
Questions?
DevOps
DevOps is an organizational and cultural movement that aims to increase software delivery *velocity*, improve service *reliability*, and build *shared ownership* among software stakeholders.

[cloud.google.com/devops](http://cloud.google.com/devops)
What is DevOps?

- design
- plan
- release
- deploy
- build
- test
- monitor
- operate

 DEV

 OPS

Google Cloud
DevOps is an organizational and cultural movement that aims to increase software delivery velocity, improve service reliability, and build shared ownership among software stakeholders.

cloud.google.com/devops
DevOps with **Istio**

**Velocity:** safe rollouts with traffic *splitting*. deprecate legacy services with *redirects*. accelerate the customer feedback loop with *A/B testing*.

**Reliability:** set SLOs and alerts on generated metrics. use *circuit breaking* and *fault injection* to harden services.

**Shared ownership:** declarative traffic/security policies in a *shared Git repo*. scope Istio policies at the *namespace* level.
Istio - Traffic Management

VirtualService, Gateway, DestinationRule, and ServiceEntry

- Traffic splitting
- Traffic steering
- Circuit breaking
- Egress control
- Fault injection
VirtualService

apiVersion: networking.istio.io/v1alpha3
class: VirtualService
metadata:
  name: frontend
spec:
  hosts:
    - "frontend.default.svc.cluster.local"
  http:
    - route:
      - destination:
        host: frontend
Gateway

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: frontend-gateway
spec:
  selector:
    istio: ingressgateway
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
  hosts:
  - "*"
```
apiVersion: networking.istio.io/v1alpha3
class: DestinationRule
metadata:
  name: frontend
spec:
  host: frontend.default.svc.cluster.local
  subsets:
  - name: v1
    labels:
      version: v1
  - name: v2
    labels:
      version: v2
Kubernetes Deployment

Pods       Pods
Kubernetes Service
Kubernetes Deployment

Pods      Pods
Pods
Kubernetes Deployment
v1
Pods
v2
Kubernetes Deployment
v2
Kubernetes Service

DestinationRule

<table>
<thead>
<tr>
<th>Pods</th>
<th>Pods</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1</td>
<td>v2</td>
</tr>
</tbody>
</table>
VirtualService
Kubernetes Service
DestinationRule
Pods
v1
Pods
v2

VirtualService
Kubernetes Service
DestinationRule
Pods
v1
Pods
v2
Demo: Service Redirect
Service Redirect

Scenario - we've moved to a faster payments service, coolcash. We want to deprecate paymentservice and redirect calls to coolcash.
DEV

OPS

design
plan
release
build
monitor
operate

deploy
Demo: Canary Deployment
Canary Deployment

- Release new service versions without worrying about ops challenges
- Goal: progressively direct traffic to the new frontend v2

Diagram:
- Loadgen
- Frontend
  - v1 (80%)
  - v1
  - v1
  - v2 (20%)
Demo: A/B Testing
A/B Testing

**Goal:** Determine which frontend layout results in the most revenue

Requests with `ab-selected:true` HTTP reader are routed to v2.
Combining Traffic Rules

http:
- match: # RULE 1 - ADD HEADER
  - uri:
    prefix: "/article/breaking-news"
  route:
    - destination:
      host: articles
    headers:
      response:
        add:
          no-cache: "true"
  timeout: 2s
- match: # RULE 2 - URI REWRITE
  - uri:
    prefix: /blog
  rewrite:
    uri: /beta/blog
  route:
    - destination:
      host: articles
    timeout: 2s
- route: # RULE 3 - DEFAULT / TIMEOUT
  - destination:
    host: articles
  timeout: 2s
  weight: 100
Resilience
What makes an application resilient?

Downstream services **fail gracefully** when an upstream service is unavailable.

**Timeouts** and **retry** logic to prevent a service waiting forever for an upstream.

**Failover policies** to another region running the same service.
Istio Resilience Features

⏰ Timeouts and retry logic
⚡ Circuit breaking
🚧 Fault injection
🚫 Client-side load balancing
🌍 Locality load balancing / Regional failover
Demo: Circuit Breaking
Circuit Breaking

- Closed
  - Success
  - Fail Under Threshold
  - Too many failures

- Open
  - Fail Fast
  - Timeout

- Half Open
  - Success
  - Failure

image source: Banzai Cloud
Circuit Breaking

Avoid cascading failures through multiple services

Istio circuit breaker:
1. detect $x$ consecutive failures
2. trip the circuit breaker
3. **fail immediately** for $t$ seconds
Demo: Fault Injection
Fault Injection

Chaos testing - detect how downstream services respond when upstream services fail

Find weak spots in application code error handling

Istio supports error and timeout faults.
Wrap-Up
Where does Kubernetes fall short?

- Safe Rollouts
- Observability
- Traffic Encryption
- Request-level Authorization
- Resilience
How can **Istio** transform your organization?
How can Istio transform your organization?

- Fast, Safe Releases
- Complete Observability
- End-to-end Encryption
- Request-level Authorization
- Failure Prediction, Reaction
By tracking service dependencies, revealing organizational structure.

By **decoupling the network** from your app code.

By handling north-south and east-west traffic with the **same APIs**.

By allowing developers to focus on building **features**, driving **business value**.

By giving you total **visibility** into service interactions.

By accelerating the **DevOps feedback loop**.

By hardening your applications, **reducing the risk of outages**.
How do your **developers** and **operators** keep things up and running with explosive growth in number of services?
By thinking **services first**: investing in **automation**, **tools**, and **cultural change**.
How can **Istio** transform your organization?
Through telemetry, uniformity, and automation.
Adopting Istio is a **journey**.
Istio Adoption Checklist

✅ **Who** will adopt Istio? (Which product teams? Which services? Will there be phases of adoption across your org?)

✅ **What features** to adopt? What will come first?

✅ **How to configure Istio?** One cluster per control plane? Multicluster? VMs?

✅ **Where** will you keep your Istio YAML? How will you roll out policy?

✅ **Plan ahead for Istio's costs** - time (sidecar latency) and money (CPUs)

✅ **How** will you upgrade Istio? How many versions behind?
**Best Practices**

1. Put an Istio control plane where your applications live.
2. Keep your Istio policies in a Git repo
3. Use `istioctl analyze` to detect bad config
4. Create a "default" VirtualService & DestinationRule for every service
5. Use Kubernetes namespaces for isolation

More at: [istio.io/docs/ops/best-practices](https://istio.io/docs/ops/best-practices)
What we didn't cover

VM workloads
Multicluster
Service mesh vs. API gateway
Secure ingress
JWT authentication
Egress traffic control
New features - istiod, Mixer v2
Resources

istio.io
istiobyexample.dev
bit.ly/istio-samples
bit.ly/istio-sacon
Thank you!
Cyberconflict: A new era of war, sabotage, and fear

9:15am-10:10am Wednesday, March 27, 2019
Location: Ballroom
Secondary topics: Security and Privacy

Rate this session

We're living in a new era of constant sabotage, misinformation, and fear, in which everyone is a target, and you're often the collateral damage in a growing conflict among states. From crippling infrastructure to sowing discord and doubt, cyber is now the weapon of choice for democracies, dictators, and terrorists.

David Sanger explains how the rise of cyberweapons has transformed geopolitics like nothing since the invention of the atomic bomb. Moving from the White House Situation Room to the dens of Chinese, Russian, North Korean, and Iranian hackers to the boardrooms of Silicon Valley, David reveals a world coming face-to-face with the perils of technological revolution—a conflict that the United States helped start when it began using cyberweapons against Iranian nuclear plants and North Korean missile launches. But now we find ourselves in a conflict we're uncertain how to control, as our adversaries exploit vulnerabilities in our hyperconnected nation and we struggle to figure out how to deter these complex, short-of-war attacks.

David Sanger
The New York Times

David E. Sanger is the national security correspondent for the New York Times as well as a national security and political contributor for CNN and a frequent guest on CBS This Morning, Face the Nation, and many PBS shows.
Appendix
Istio

Service mesh tool
Open Source
Istio APIs
Prometheus, Grafana, Jaeger
Control plane runs on your cluster

Anthos Service Mesh

Service mesh tool
Google Product
Istio APIs
Google Cloud Monitoring, Tracing
Control plane managed outside your cluster
Works on GCP, AWS, on-prem
SRE dashboards, alerts built in
Security insights + recommendations