The Road to Istio: How IBM, Google and Lyft Joined Forces to Simplify Microservices

Dr. Tamar Eilam
IBM Fellow @ Watson Research Center, NY
eilamt@us.ibm.com
@tamareilam
2004: The Gap Between Dev and Ops

- Inefficiencies
- Unpredictability
- Risk
- Manual processes
- High failure rates
Desired State

communicate

host

What

(AI Planning)

(Graph Techniques)

Automation

Fragile

Too many (implicit) dependencies ....

How
2009: The DevOps Revolution

- Cultural
- Technological
- Automate everything
- Version everything
- Infrastructure immutability

→ Repeatability
→ Agility
→ Reduced Risk

→ Software Architecture Still Complicated 😞
2011: Cloud Native Principles (Microservices)

Conversation Expanded

DEVELOP ← DEPLOY → MANAGE

Microservices
GOALS:
› Agility is the goal.
› Release is the beginning of the journey

PRINCIPLES:
› Design your software such that it will be easily managed

METHOD:
› Loose Coupling
› Configuration normalization
› Runtime Dynamic Binding
› Data De-Normalization
› Resiliency
   › Circuit Break
   › Bulk Head
Agility:
The Three Pillars

Cultural Change
Automated pipeline
Everything as code
Immutable infrastructure

Small decoupled services
Everything dynamic
APIs

Design for failure
Embrace failures
Test by break / fail fast

Portability
Developer centric
Eco system
Fast startup
Why Microservice Mesh?

- Complexity is shifted
- How to gain insight?
- How to guarantee security?
- How to troubleshoot?
# Our Microservice Mesh Goal (2016)

<table>
<thead>
<tr>
<th>Polyglot</th>
<th>Not require change to code</th>
<th>Enable Experimentation / Insight</th>
<th>Enable Resiliency / Resiliency Testing</th>
</tr>
</thead>
</table>

Our Microservice Mesh Goal (2016) emphasizes the following aspects:

- **Polyglot**: Not require change to code
- **Enable Experimentation / Insight**: Enable Resiliency / Resiliency Testing
Microservice Mesh Design (2016)

- Polyglot → Sidecar Proxy
- Not require change to code → Sidecar proxy
- Enable experimentations / Insight → Smart Routing
- Enable Resiliency / Resiliency Testing → Failure Injection
- Easy to use → Central Control Plane
2016: Amalgam8 (A Predecessor of Istio)

- Central Control Plane
- Smart Layer 7 routing
- Injection of delays, failures

Enable:

- Experimentation with multiple versions (A/B, Canary)
- Resiliency Testing
2017: ISTIO: AN OPEN PLATFORM TO CONNECT, MANAGE AND SECURE MICROSERVICES (2017)

Istio

- Traffic Steering
- Rate limiting
- Request Tracing
- Secure Communication
- Circuit Breakers

Community

IBM

Google

lyft

Content-based routing and failure injection for polyglot microservices

High performance service proxy

https://github.com/istio
Intelligent routing and load balancing

- Conduct traffic between services with dynamic route configuration
- A/B tests
- Canary releases
- Red/Black deployments
Example: Intelligent Routing

Traffic Steering

Content Based

Rule Example

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: reviews
spec:
  hosts:
    - reviews
  http:
    - match:
      - headers:
          cookie:
            regex: ".*?\{user=\{.*\}\}.*?\{.*\}"
      - route:
          - destination:
              host: reviews
              subset: v2
          - route:
              - destination:
                  host: reviews
                  subset: v3
```
Resilience across languages and platforms

- Increase reliability by shielding applications from flaky networks and cascading failures in adverse conditions
  - Timeouts, Retries, etc
Timeout and Retry Configuration is Easy

```yaml
apiVersion: networking.istio.io/v1alpha3
class: VirtualService
metadata:
  name: ratings
spec:
  hosts:
  - ratings
http:
  - route:
    - destination:
      host: ratings
      subset: v1
    retries:
      attempts: 3
      perTryTimeout: 2s
```

```yaml
apiVersion: networking.istio.io/v1alpha3
class: VirtualService
metadata:
  name: ratings
spec:
  hosts:
  - ratings
http:
  - route:
    - destination:
      host: ratings
      subset: v1
    timeout: 10s
```
Resilience Testing

- Inject Delays and Failures to test your resiliency to eliminate cascading failures
Example: Resiliency Configuration

- Test your resiliency by simulating different types of failures
- Controlled Chaos Testing

- Set delay to 2s on particular link

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: ratings
spec:
  hosts:
  - ratings
  http:
  - fault:
    delay:
      percent: 100
      fixedDelay: 2s
    route:
      - destination:
        host: ratings
        subset: v1
```
Visualizing the **Affect** on the Network

- We prototype a tool to visualize affect of delays and failures on an entire mesh of Microservices
- Helps simulate and identify cascading failures as a result of inconsistent configuration of resiliency parameters
Fleet-Wide Policy Enforcement

- Apply organizational policy to the interaction between services
- Ensure access policies are enforced
- Role Based Access Control Policies
- Define role, and role binding
- Extensible (Mixer Adapter)
In-Depth Telemetry and Reporting

- Understand the dependencies between services, the nature and flow of traffic between them, and quickly identify issues
- Rich data collection
  - Prometheus (metric), Fluentd (logs), Jaeger (tracing), ...
- Rich Data Visualization
  - Graphana, ...
- Extensible and configurable (Mixer)
GAIN INSIGHT BY VISUALIZING YOUR DATA
Istio Status

- Vibrant Community
  - RedHat, Tigera, Cisco, Weavework,...
- Released 0.8
  - Revamped Traffic Management Model
  - Streaming Envoy configuration
  - Gateway
  - Istio a la carte
  - ...
- Full House events at conferences
Envoy

- C++ based L4/L7 proxy
- Low memory footprint
- Battle tested at Lyft
- Runs with 100+ services
- 10K VMs
- 2M requests/sec
Istio Analytics

- Canary
- A/B testing
- Troubleshooting
Resiliency Testing and Auto-Configuration

- How to auto-configure resiliency parameters
- Discover timeout and retry values which achieves the optimal
- Dynamically adjust
THANK YOU!

Dr. Tamar Eilam
IBM Fellow @ Watson Research Center, NY
eilamt@us.ibm.com
@tamareilam