A Comparison of Service Mesh Options
Looking at Istio, Linkerd, Consul-connect
Syed Ahmed - CloudOps Inc
Introduction
• Cloud Software Architect @ CloudOps
• PMC for Apache CloudStack
• Worked on network modules in Openstack and CloudStack
• Previously worked on the Netscaler LB
• Part of the DevOps team @ Yahoo!
• We Design, Build and Operate Clouds
• Help customer own their destiny in the Cloud
• Vender/Cloud Agnostic
A Case for Service Mesh
Monolithic Architecture

- Strong Coupling between different modules causing anti-patterns in communicating between different modules
- Difficulties in Scaling
- Updating to new version requires complete re-install
- Problem in one module can cause the whole application to crash
- Difficult to move to a new framework or technology
**Microservices Architecture**

- API contract between different modules/service ensures that each module can be developed and maintained independently
- Each service can be scaled independently
- Updating to new version requires only updates to a specific service
- Allows for easier CI/CD
Evolution of the Ecosystem

- Docker
- LXC
- Kubernetes
- Mesos
Challenges with the Microservices Architecture
Challenges with the Microservices Architecture
Challenges with the Microservices Architecture
Challenges with the Microservices Architecture
Challenges with the Microservices Architecture
Challenges with the Microservices Architecture
A Service Mesh is the substrate between different microservices that makes connectivity between different microservices possible. In addition to providing networking, a Service Mesh can also provide other features like Service Discovery, Authentication and Authorization, Monitoring, Tracing and Traffic Shaping.
Sidecar Pattern

SVC A

- Container A
  - Proxy A
- Container B
  - Proxy B
- Container C
  - Proxy C
Istio

- Open Sourced by Google, IBM & Lyft in May 2017
- Service Mesh designed to connect, secure and monitor microservices
Istio Features

- Automatic load balancing for HTTP, gRPC, WebSocket, and TCP traffic.
- Fine-grained control of traffic behavior with rich routing rules, retries, failovers, and fault injection.
- A pluggable policy layer and configuration API supporting access controls, rate limits and quotas.
- Automatic metrics, logs, and traces for all traffic within a cluster, including cluster ingress and egress.
- Secure service-to-service communication in a cluster with strong identity-based authentication and authorization.
**Istio Architecture**

- **Envoy:** high-performance proxy developed in C++ provides Dynamic service discovery, Load balancing, TLS termination, HTTP/2 and gRPC proxies, Circuit breakers, Health checks, Staged rollouts with %-based traffic split, Fault injection, Rich metrics

- **Pilot:** The core component used for traffic management in Istio is Pilot, which manages and configures all the Envoy proxy instances deployed in a particular Istio service mesh

- **Mixer:** Mixer is a platform-independent component. Mixer enforces access control and usage policies across the service mesh, and collects telemetry data from the Envoy proxy and other services. The proxy extracts request level attributes, and sends them to Mixer for evaluation

- **Citadel:** Citadel provides strong service-to-service and end-user authentication with built-in identity and credential management. You can use Citadel to upgrade unencrypted traffic in the service mesh. Using Citadel, operators can enforce policies based on service identity rather than on network controls
**Gateway** describes a load balancer operating at the edge of the mesh receiving incoming or outgoing HTTP/TCP connections.

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: httpbin-gateway
spec:
  selector:
    istio: ingressgateway
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
    hosts:
    - "httpbin.example.com"
```
A **VirtualService** defines a set of traffic routing rules to apply when a host is addressed. Each routing rule defines matching criteria for traffic of a specific protocol. If the traffic is matched, then it is sent to a named destination service (or subset/version of it) defined in the registry.

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: reviews-route
spec:
  - route:
    - destination:
      host: reviews.prod.svc.cluster.local
      subset: v2
      weight: 25
    - destination:
      host: reviews.prod.svc.cluster.local
      subset: v1
      weight: 75
```
**DestinationRule** defines policies that apply to traffic intended for a service after routing has occurred. These rules specify configuration for load balancing, connection pool size from the sidecar, and outlier detection settings to detect and evict unhealthy hosts from the load balancing pool.

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: bookinfo-ratings
spec:
  host: ratings.prod.svc.cluster.local
  trafficPolicy:
    loadBalancer:
      simple: LEAST_CONN
```
- Initially started as a network proxy (v1.0) for enabling service mesh
- Merged with Conduit to form Linkerd 2.0 in Sept 2018
Linkerd Architecture
Linkerd Architecture

- **Controller:** The controller consists of multiple containers (public-api, proxy-api, destination, tap) that provide the bulk of the control plane’s functionality.

- **Web:** The web deployment provides the Linkerd dashboard.

- **Prometheus:** All of the metrics exposed by Linkerd are scraped via Prometheus and stored. An instance of Prometheus that has been configured to work specifically with the data that Linkerd generates is deployed.

- **Grafana:** Linkerd comes with many dashboards out of the box. The Grafana component is used to render and display these dashboards. You can reach these dashboards via links in the Linkerd dashboard itself.
Linkerd’s philosophy is to be a very lightweight addition on top of existing platform.

- No need to be a Platform admin to use linkerd.
- Simple installation and CLI tools to get started.
- Small sidecar proxy written in Rust.
- Can do end-to-end encryption and automatic proxy injection.
- Lacks complex routing and tracing capabilities.
Linkerd Commands

**Install:**
linkerd check --pre
linkerd install | kubectl apply -f -

**Inject:**
kubectl get -n emojivoto deploy -o yaml \
| linkerd inject - \
| kubectl apply -f -

**Inspect:**
linkerd -n emojivoto stat deploy
linkerd -n emojivoto top deploy
linkerd -n emojivoto tap deploy/web
Consul
Connect
Consul is a highly available and distributed service discovery and KV store.

Consul Connect augments Consul and adds Service Mesh Capabilities and was added in July 2018.
Consul Connect Features

- Provides secure service-to-service communication with automatic TLS encryption and identity-based authorization.
- Uses envoy proxy sidecar as the dataplane.
- Integration with Vault for certificate and secret management.
- Service discovery already provided by Consul.
- Useful if you want to use services outside Kubernetes as Consul can do a 2 way sync between k8s services and Consul services.
- No routing features. Main focus on service discovery and Service Identity management.
Conclusion
## Conclusion

<table>
<thead>
<tr>
<th>Feature</th>
<th>Istio</th>
<th>Linkerd</th>
<th>Consul Connect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic Redirection (Blue/Green deployment)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Traffic Splitting (Canary deployment)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Attribute based routing</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Service Identification</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Auto Proxy Injection</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Non-Admin installation</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Built-in Dashboard</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Certificate Management</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
## Conclusion

<table>
<thead>
<tr>
<th>Feature</th>
<th>Istio</th>
<th>Linkerd</th>
<th>Consul Connect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metrics Collection</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Built-In Dashboard</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TLS</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>External Service Support</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Rate Limiting</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Tracing</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Appendix (Emojivoto App)

vote-bot
(traffic generator)

web

emoji

voting
OPEN SOURCE NETWORKING DAYS