The road to microservice for Database as a Service (DBaaS) via Istio

Peng Hui Jiang

Huailong Zhang

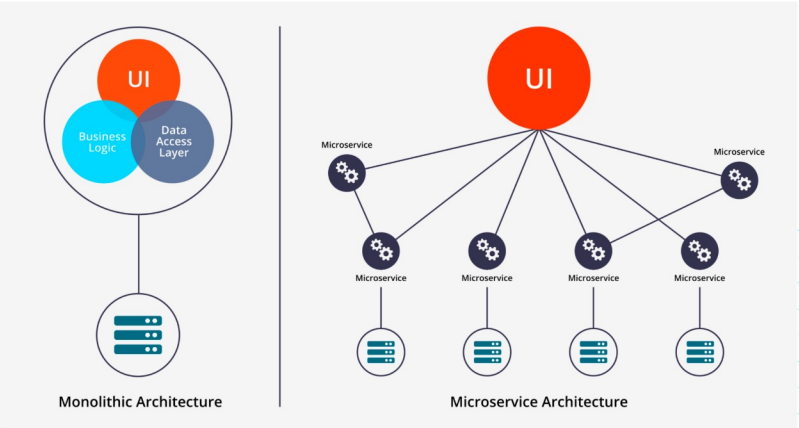
IstioCon

Welcome to the IstioCon 2022

IstioCon 2022 is the inaugural community conference for the industry's most popular service mesh. IstioCon is a community-led event, showcasing the lessons learned from running Istio in production, hands-on experiences from the Istio community, and featuring maintainers from across the Istio ecosystem. The conference offers a mix of keynotes, technical talks, lightning talks, workshops and roadmap sessions. Fun and games are also included with two social hours to take the load off and mesh with the Istio community, vendors, and maintainers!

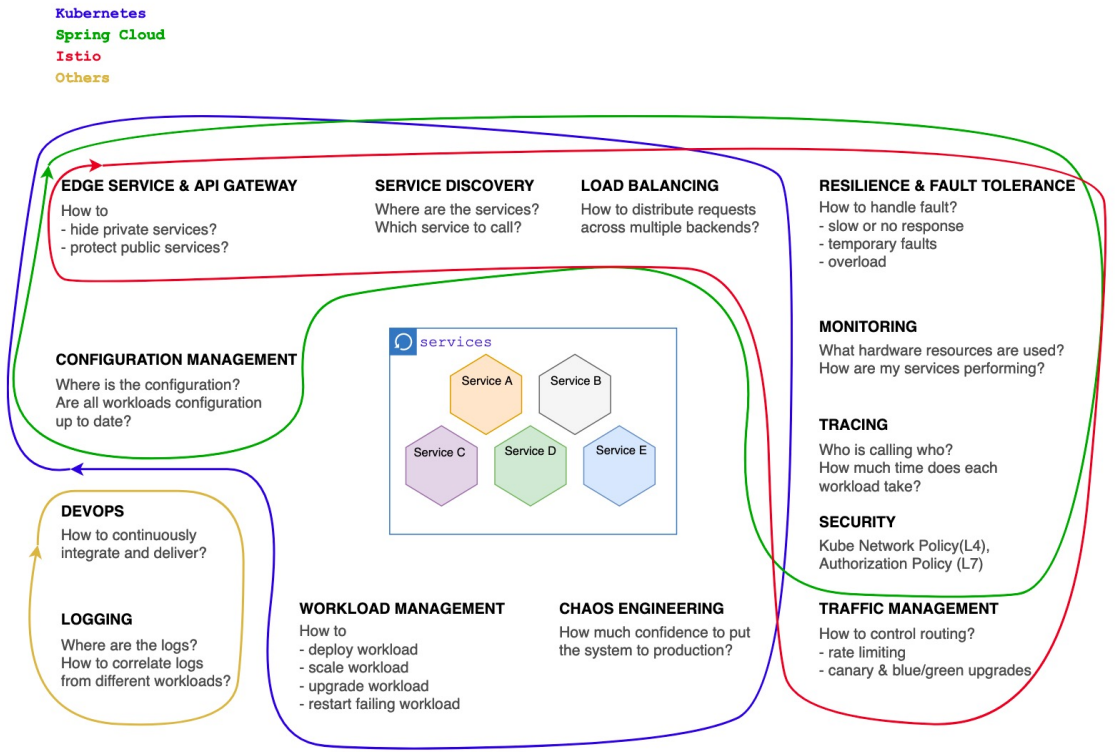
Background – Monolith to Microservices

- Recently, the Database as a Service (DBaaS) saw a significant growth YoY. One of the key reasons for the growth of DBaaS is the explosive growth of data that we have observed over the last year. The pandemic created strong data growth (WFH, e-learning, etc.)
- However, well designed DBaaS systems tend to adopt a stateless, loosely coupled architecture, with efficient message passing to produce a scalable, stable and reliable service. In addition, one major characteristic of DBaaS is to serve for multi-Tenants to reduce cost and provide highly availability and scalability.
- Successful multi-Tenant platforms require massive scalability, online patching/upgrading, the ability to process high volumes of data ingestion.
- To make the DBaaS become more cloud native, there is a journey for us to migrate the legacy monolithic systems into microservice architecture. And as an excellent project of service mesh, Istio become the first choice to help to complete this process.



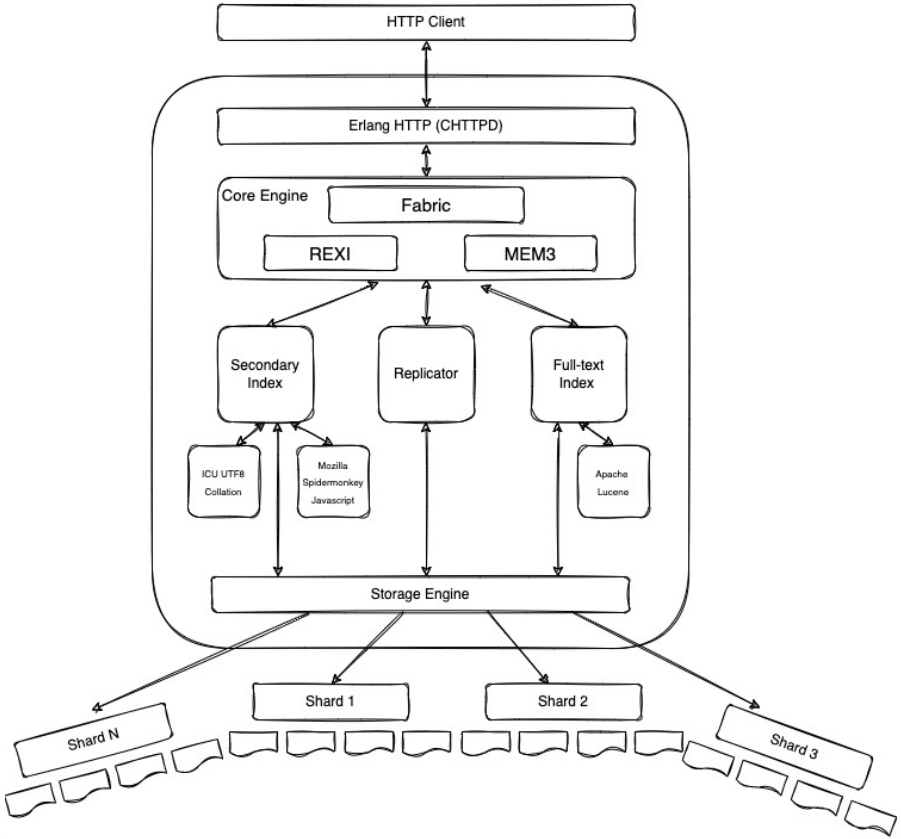
Background – Challenge for DBaaS

- Multi-Tenant support
- Auto-scale and migration support
- CI/CD process handling
- Observability capability



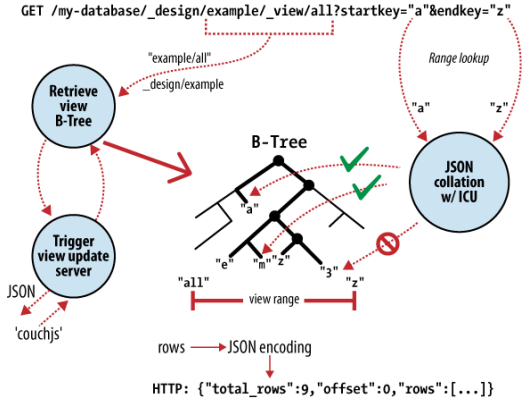
Inside CouchDB

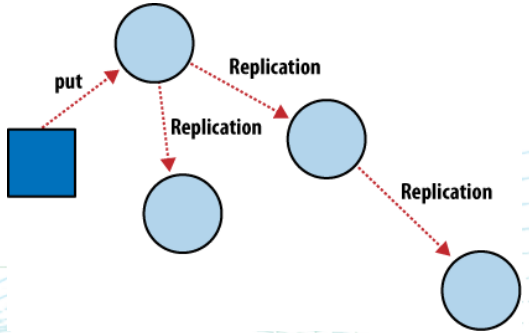
- **Apache CouchDB** is an open-source document-oriented NoSQL database, implemented in Erlang.
- CouchDB uses multiple formats and protocols to store, transfer, and process its data. It uses JSON to store data, JavaScript as its query language using MapReduce, and HTTP for an API.



Inside CouchDB

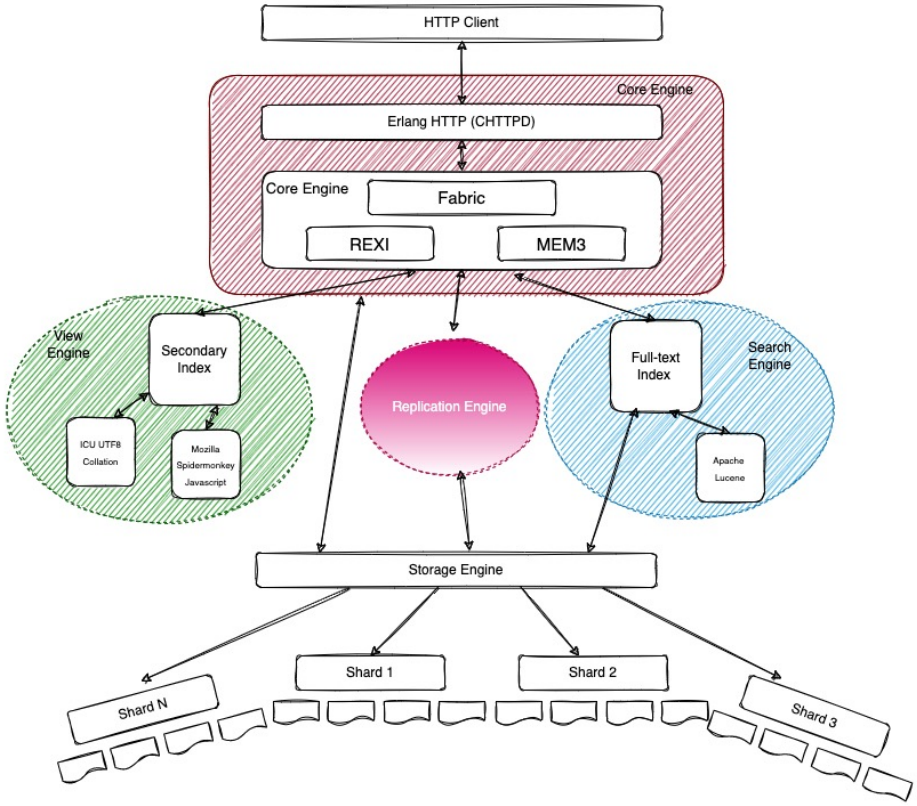
- B-tree storage and view request
  - A powerful *B-tree* storage engine at the heart of CouchDB
  - Sorted data structure
    - To allow searches, insertions and deletions
    - For all internal data, documents and views
- Incremental replication between CouchDB nodes
  - Synchronize data between any two databases however you like and when your like.
  - Synchronize database servers within a cluster or between data centers





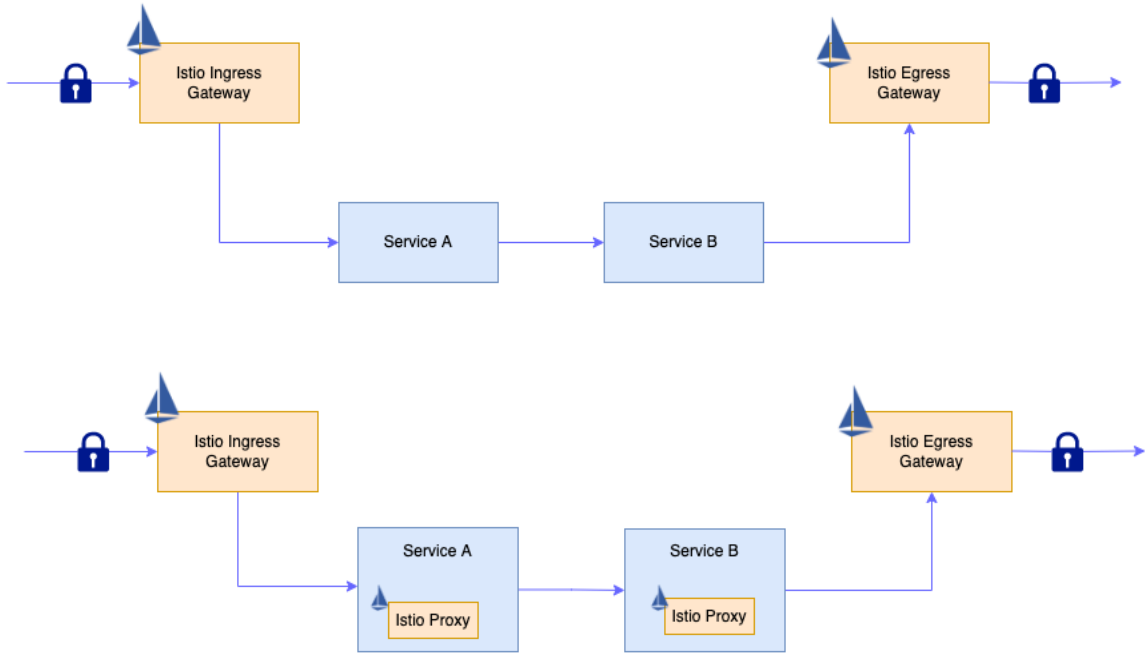
Decompose DBaaS into microservices

- Can be decomposed into microservice thanks to
  - BASE instead of ACID
  - Document oriented storage model
  - Distributed storage
- Key Decomposed Microservices
  - Core Engine
  - View Engine
  - Search Engine
  - Replication Engine



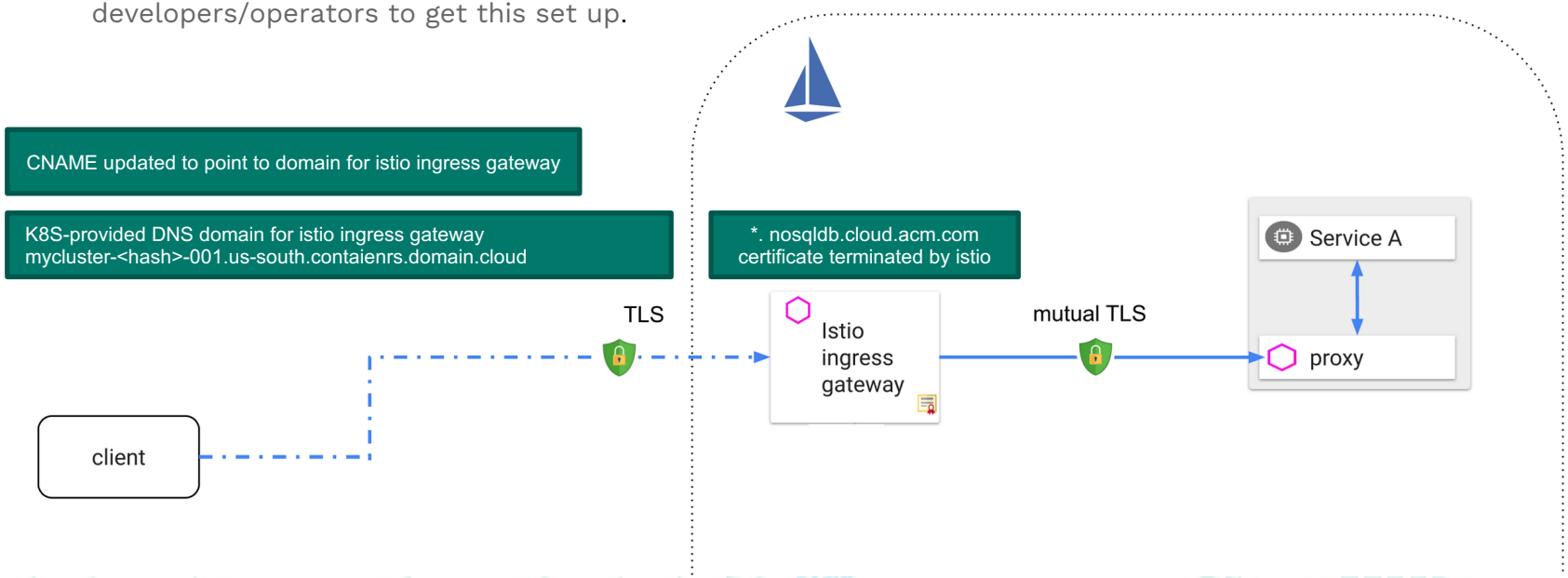
Manage DBaaS Microservices via Istio

- From Gateway Only
  - Ingress gateway
  - Egress gateway
- To Incrementally add more services to the mesh



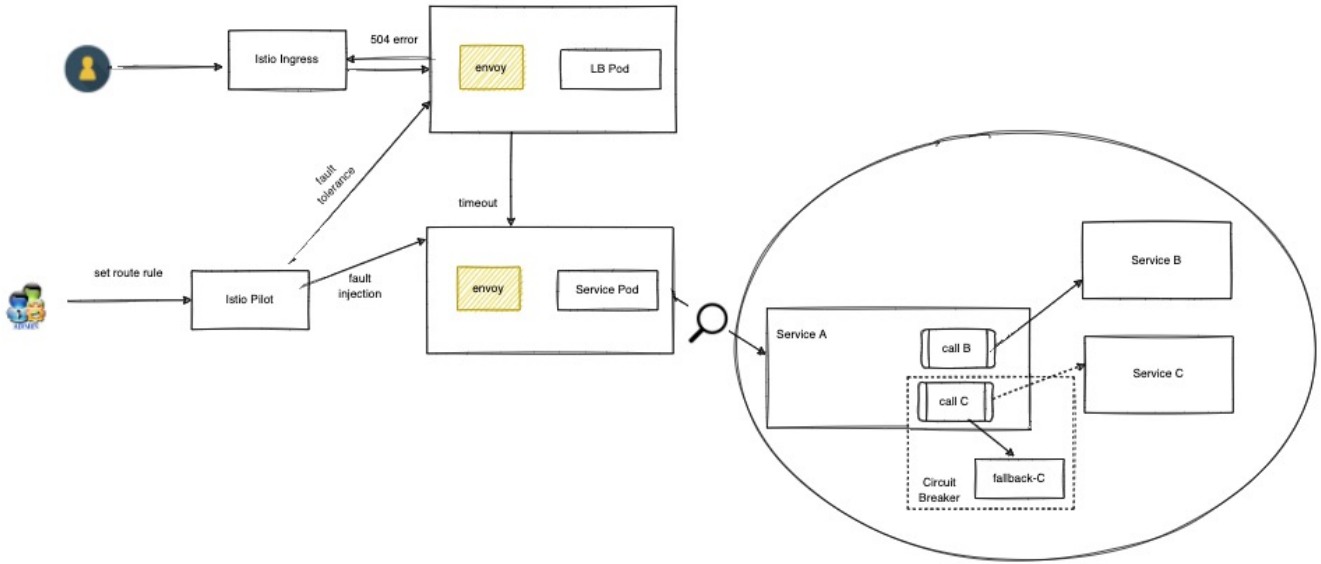
Manage DBaaS Microservices via Istio

- Connections between services within Kubernetes Service are typically in plaintext, while there are exceptions to this rule TLS is not enforced.
- The objective is to ensure that all communication within Kubernetes services use mutual TLS, in such a way as to address the difficulty for developers/operators to get this set up.



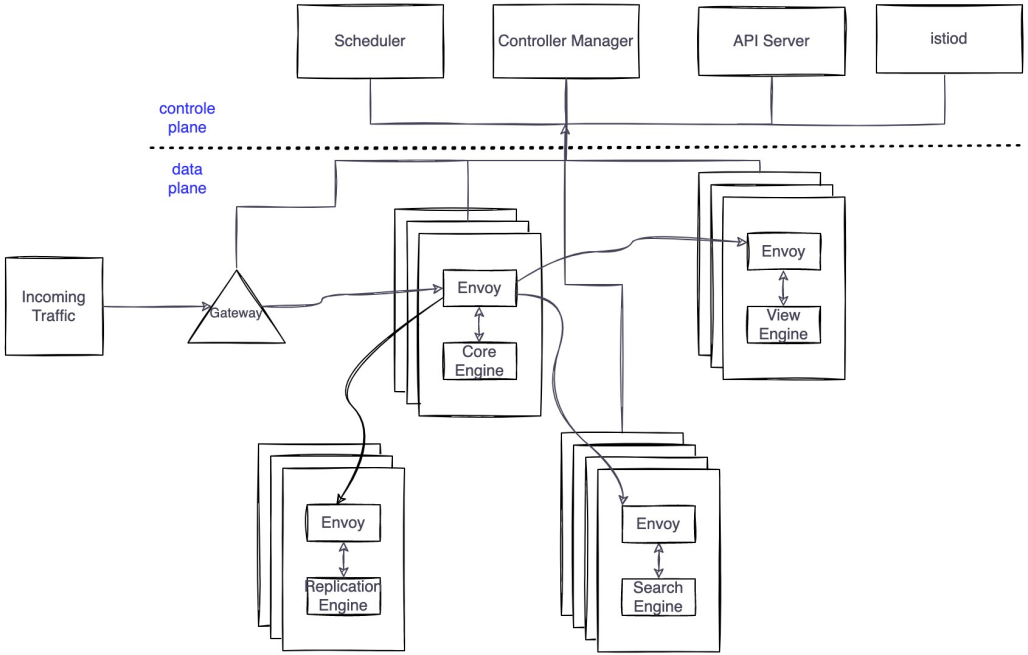
Manage DBaaS Microservices via Istio

- The missing fallback at Istio can be solved by using a framework (Resilience4j)
- Istio traffic management and application resilience can co-exist



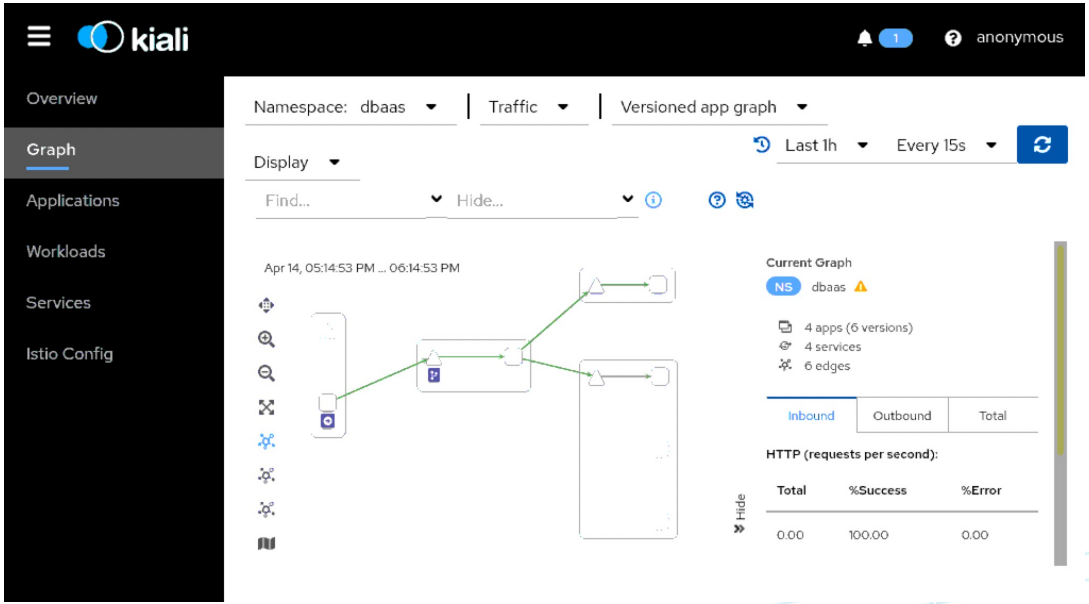
Practice on DBaaS Data Plane with Istio

- Challenge
  - Add to the networking load placed on microservices in call chains
  - Bring complex in multi-Tenant architectures serving applications for multiple users

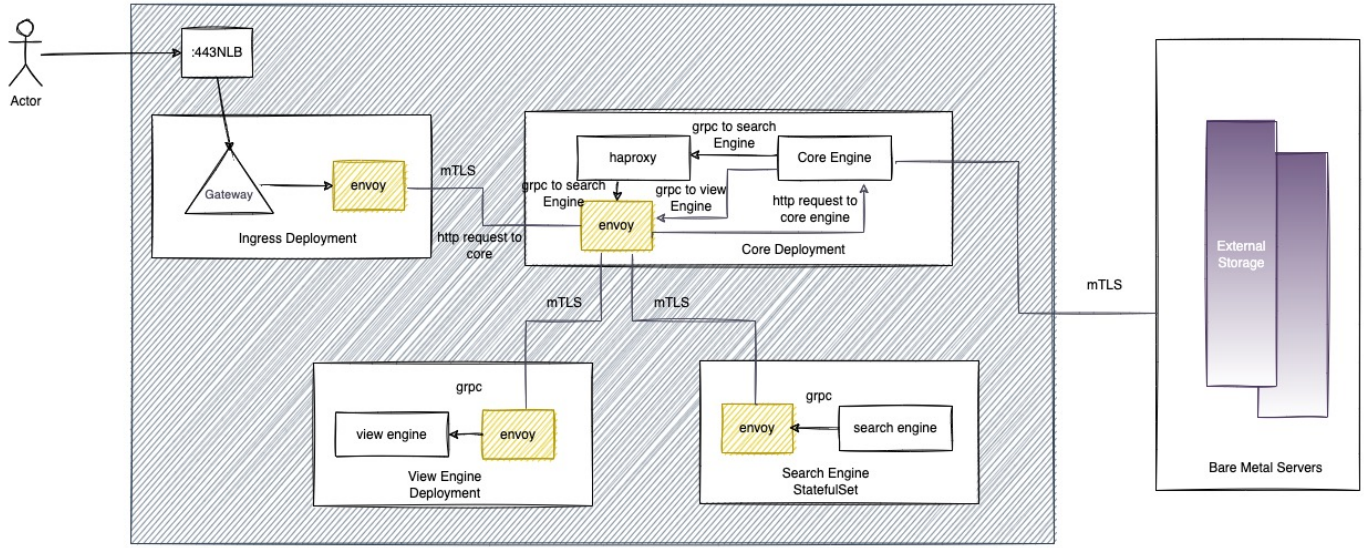


Practice on DBaaS Data Plane with Istio

- Observability
  - Observe the connections and microservices in Istio service mesh
  - Identify request routing, circuit breakers, request rates, latency using the visualized service mesh topology

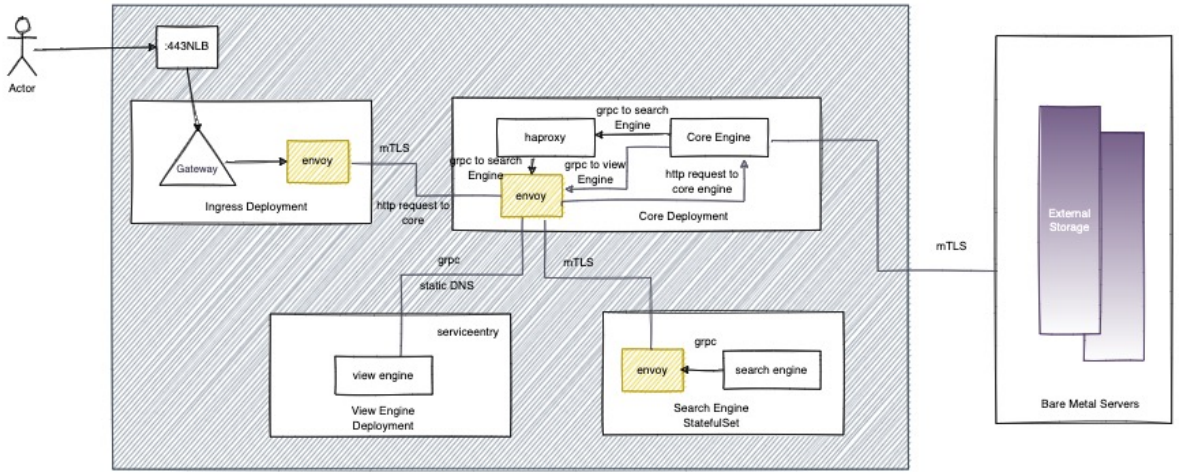


Practice on DBaaS Data Plane with Istio

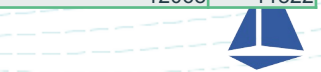


connections	threads	duration(s)	Latency			QPS/Per-Thread			avg QPS(req/sec)	throughput(req/min)	timeout
			avg(ms)	stdev(ms)	max(ms)	avg(req/sec)	stdev(req/sec)	max(req/sec)			
10	1	60	43.15	10.68	94.19	232.65	15.05	280	231.61	13904	0
100	1	60	537.38	113.67	1340	192.72	73.61	525	185.46	11150	0
100	10	60	521.12	194.25	1980	21.91	13.93	90	188.7	11354	102
100	100	60	443.09	20.27	610.22	1.94	0.29	10	223.88	13474	0
1000	100	60	1340	284.75	1610	9.82	10.76	80	196.81	11848	11810

Practice on DBaaS Data Plane with Istio



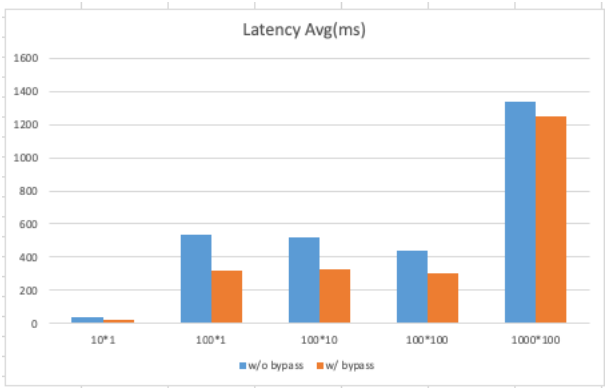
connections	threads	duration(s)	Latency			QPS/Per-Thread			avg QPS(req/sec)	throughput(req/min)	timeout
			avg(ms)	stdev(ms)	max(ms)	avg(req/sec)	stdev(req/sec)	max(req/sec)			
10	1	60	26.71	3.35	58.41	323.84	16.04	389.76	321.08	19257	0
100	1	60	316.52	27.25	733.01	273.66	44.56	726.03	258.72	15978	0
100	10	60	328.83	87.66	1356	31.13	11.47	78	269.09	16248	0
100	100	60	300.86	18.78	439.1	1.97	0.25	10	235.02	14148	0
1000	100	60	1250	124.52	1534	4.77	5.79	70	200.47	12068	11822



Practice on DBaaS Data Plane with Istio

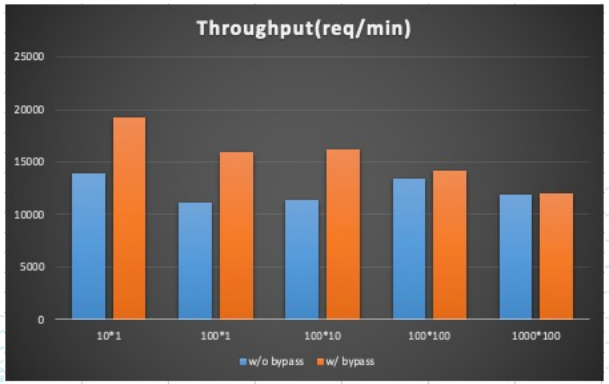
- With injected Proxy

connections	threads	duration(s)	Latency			QPS/Per-Thread			avg QPS(req/sec)	throughput(req/min)	timeout
			avg(ms)	stdev(ms)	max(ms)	avg(req/sec)	stdev(req/sec)	max(req/sec)			
10	1	60	43.15	10.68	94.19	232.65	15.05	280	231.61	13904	0
100	1	60	537.38	113.67	1340	192.72	73.61	525	185.46	11150	0
100	10	60	521.12	194.25	1980	21.91	13.93	90	188.7	11354	102
100	100	60	443.09	20.27	610.22	1.94	0.29	10	223.88	13474	0
1000	100	60	1340	284.75	1610	9.82	10.76	80	196.81	11848	11810



- Without injected Proxy

connections	threads	duration(s)	Latency			QPS/Per-Thread			avg QPS(req/sec)	throughput(req/min)	timeout
			avg(ms)	stdev(ms)	max(ms)	avg(req/sec)	stdev(req/sec)	max(req/sec)			
10	1	60	26.71	3.35	58.41	323.84	16.04	389.76	321.08	19257	0
100	1	60	316.52	27.25	733.01	273.66	44.56	726.03	258.72	15978	0
100	10	60	328.83	87.66	1356	31.13	11.47	78	269.09	16248	0
100	100	60	300.86	18.78	439.1	1.97	0.25	10	235.02	14148	0
1000	100	60	1250	124.52	1534	4.77	5.79	70	200.47	12068	11822



Multi-Tenant Support for DBaaS via Istio

2 kinds of solution for Multi-Tenant support

- **Namespace based solution**
  - Use Kubernetes native namespace resource for different tenants
  - Use authorization policy to enhance tenant isolation
- **Multi Cluster/Control Plane based solution**
  - Deploy a cluster or control plane for a tenant
  - Cluster/Control plane scope isolation for each tenant

Multi-Tenant Support for DBaaS via Istio

- **Namespace based solution**
  - Pros:
    - More cost-effective and multiple tenants can share within a k8s cluster
    - Easier to implement and operation, and a native namespace per tenant
  - Cons:
    - Not strictly isolation among tenants
    - Cluster scope resource for a tenant has impact on other tenants

- **Cluster/Control Plane based solution**
  - Pros:
    - Better isolation and security for a tenant
    - Better user experience than namespace solution
  - Cons:
    - Expensive due to extra resource usage, such extra cluster or control plane
    - Lower resource utilization
    - Heavy burden for operation

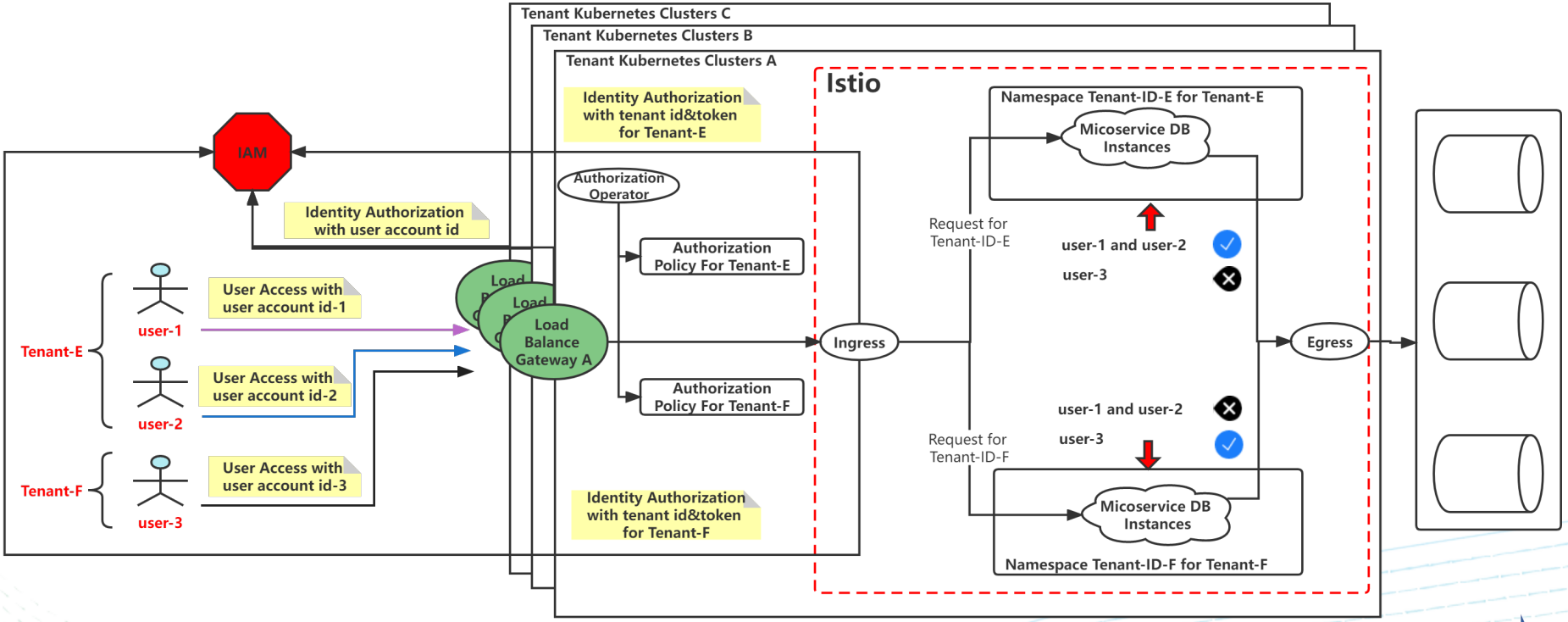
Multi-Tenant Support for DBaaS via Istio

Namespace based solution is the choice. However, some actions should be taken to improve user experience, isolation and security.

- **Admin privileges handling**
  - Cluster admin – super admin
  - Tenant admin
- **Improve user experience**
  - Extra operators are deployed to make it easier for tenant to interact with cluster/control plane scope resources.
- **Improve isolation**
  - Both IAM and Istio authorization policy are used to enhance isolation based on user account id, tenant id, token and namespace.
- **Improve security**
  - Both IAM and Istio authentication policy are used to enhance security for microservice DB instances.

```
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
name: viewengine
namespace: tenant_1
spec:
  selector:
    matchLabels:
      app: coreengine
      version: v1
action: ALLOW
rules:
- from:
  - source:
    principals:
["cluster.local/ns/tenant_1/sa/coreengine"]
  - source:
    namespaces: [”tenant_1"]
  to:
- operation:
  methods: [“GET, POST, PUT, DELETE"]
  when:
  - key: request.auth.claims[iss]
  values: ["https://nosqldb.acm.com"]
```

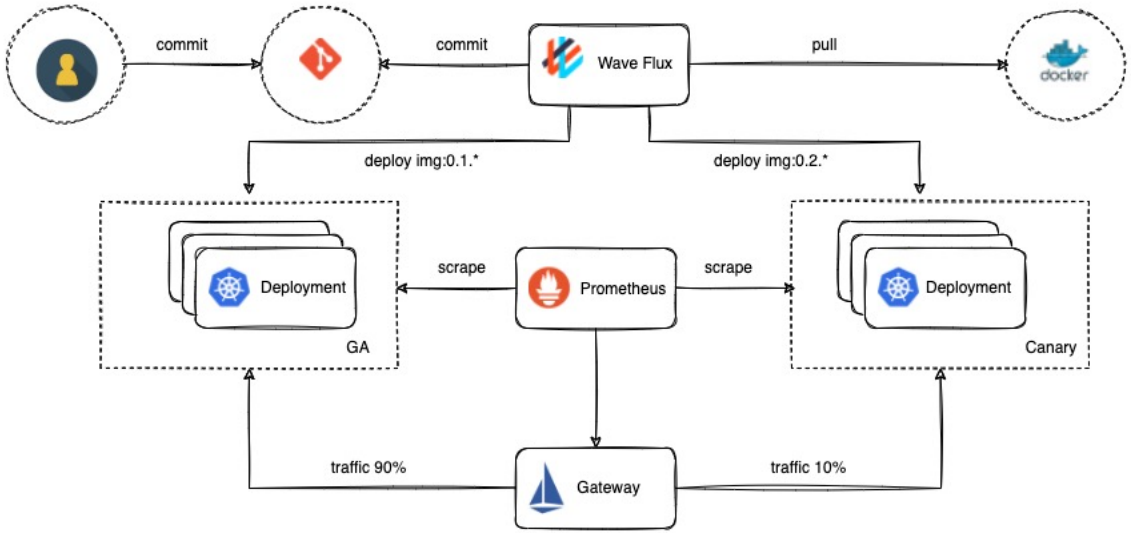
Multi-Tenant Support for DBaaS via Istio



CI/CD Process for DBaaS via Istio

Tools for CI/CD:

- Manifest Generation
  - Helm
  - Draft
  - Kustomize
  - istioctl
- CI/CD:
  - Spinnaker
  - Jenkins X
  - FluxCD
  - ArgoCD
  - Concourse
- Operators
  - operator-framework
- Observability
  - Prometheus
  - Grafana
  - Kiali



Thank you!

jiangphcn@apache.org
zhlsunshine878@gmail.com