Istio: Managing Multi-tenant ML Workloads

Wencheng Lu / Senior Staff Software Engineer, Google
Limin Wang / Staff Software Engineer, Google
Wencheng Lu
- wlu@google.com
Senior Staff Software Engineer, Google

Limin Wang
- liminwang@google.com
Staff Software Engineer, Google
What is Istio?

An open services platform to manage service interactions across container- and VM-based workloads
<table>
<thead>
<tr>
<th>Uniform observability</th>
<th>Operational agility</th>
<th>Policy driven security</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Uniform observability" /></td>
<td><img src="image2" alt="Operational agility" /></td>
<td><img src="image3" alt="Policy driven security" /></td>
</tr>
</tbody>
</table>
Istio Architectural Components

**Pilot**: Policy distribution.

**Galley**: Policy validation.

**Mixer**: Telemetry integration.

**Citadel**: Key/cert management.
Multi-tenant ML Workloads

Prepare Data
- Ingestion
  - Data
  - Transform
  - Split

Train Models
- Model Selection
- Model Tuning
- Model Testing
- Model Validation

Serve Models
- Model Deployment
- Prediction Service
Challenges on Multi-Tenancy

- **Security**
  - Isolation of operators to manage each tenant’s ML workloads and resource.
  - Isolation of communication among tenants’ workloads.
  - Isolation of user access to each tenant’s job.

- **Operational Agility**
  - Rollout and A/B testing.

- **Observability**
  - Enable monitoring/logging/auditing/tracing per tenant and enforce access control.
Proposed Solution: Istio + k8s Namespace

Istio Control Plane
- Citadel
- Pilot
- Mixer
- Galley

Tenant 1 Namespace
- envoy
- ML Workload

Tenant 2 Namespace
- envoy
- ML Workload

Tenant 3 Namespace
- envoy
- ML Workload
Case Study: Kubeflow

- Kubeflow = ML + Kubernetes
  - Managing Jupyter Notebooks
  - A platform for building, deploying, and managing ML workflows

- An example of Kubeflow pipeline

Create Cluster → Prepare Data → Train Models → Serve Models

Create Cluster

Prepare Data
- TensorFlow Analyze
- TensorFlow Transform

Train Models
- TensorFlow Training (TFJob)

Serve Models
- TensorFlow Serving (TFServing)
Resource Isolation: K8S Namespace

- Each tenant’s ML workloads are managed within a dedicated K8S namespace (e.g., kubeflow-alice).
- K8S RBAC controls operator management of microservices.
  - Create a namespace for a user/team (tenant). Set quota/ACL for tenant’s namespace, etc.

```
ns: kubeflow-system
  Jupyter Spawner UI
  Jupyter Controller
  Profiles Controller
  Alice’s profile
```

```
ns: kubeflow-alice
  Jupyter Notebook
  TFJob
  TFServing
```

Communication Isolation: Istio Authn + Authz

- **Istio mutual TLS Authentication**
  - Data encrypt in transit
  - Strong workload identity
    - k8s service account
    - Cryptographically signed in X.509 cert

- **Istio Identity-based Authorization**
  - Service/Namespace level segmentation at both http and TCP layers
  - Supports service and end user authorization
  - RBAC + condition provides good usability and flexibility
  - High performance: implemented in Envoy as native authorization support
User Access Isolation: End User Authentication

- Istio Ingress sends the requests to AuthService, which redirects the user to login with an IDP, and returns a JWT (Request Context Token).
- RCToken (Request Context Token) short-lived JWT used inside Istio Mesh.
- RC token is validated by Istio proxies (Envoy sidecar).
User Access Isolation: Authorization

- Istio authorization policy is used to authorize both channel and user credentials. Typical use cases:
  - Authorize a developer to access a Jupyter Notebook.
  - Authorize an end-user to access a model serving.

```yaml
apiVersion: rbac.istio.io/v1alpha1
kind: ServiceRole
metadata:
  name: alice-serving
  namespace: kubeflow-alice
spec:
  rules:
    - services: ["TFServing"]
      methods: ["GET", "HEAD"]

apiVersion: rbac.istio.io/v1alpha1
kind: ServiceRoleBinding
metadata:
  name: example-role-binding
  namespace: kubeflow-alice
spec:
  subjects:
    - user: "istio-ingress-service-account"
      properties:
        request.auth.claims["sub"]: "alice@foo.com"
  roleRef:
    kind: ServiceRole
    name: alice-serving
```
Istio Observability for Multi-tenant

- Use Istio to monitor ML workloads for each tenant.
- Istio supports:
  - Pluggable monitoring backends (Stackdriver, Prometheus, etc).
  - Policies to configure metrics and logs.
- Monitoring backends like Stackdriver provide isolation between tenants
  - E.g., Alice only sees logs and metrics in kubeflow-alice.
Istio Traffic Management

- Istio traffic management provides:
  - Automatic rollout to new versions
  - A/B testing for different ML models
  - Separation of Staging/Prod.
  - etc.
- Configured through traffic policies
  - No hot restarts, no traffic disruption

```yaml
apiVersion: networking.istio.io/v1alpha3
description: VirtualService
metadata:
  name: TFJob
  namespace: kubeflow-alice
spec:
  hosts:
  - TFJob
  http:
    - route:
      - destination:
        host: TFJob
        subset: v1
        weight: 95
      - destination:
        host: TFJob
        subset: v2
        weight: 5
```

Canary

User-agent Android

User-agent Apple
Come Participate!

- Istio docs [istio.io](https://istio.io)
- Istio discussion board [discuss.istio.io](https://discuss.istio.io)
- Join Istio working groups [github.com/istio/community/blob/master/WORKING-GROUPS.md](https://github.com/istio/community/blob/master/WORKING-GROUPS.md)
- Contribute code [github.com/istio](https://github.com/istio)

- Kubeflow docs [https://www.kubeflow.org/](https://www.kubeflow.org/)
- Discussion forum [kubeflow-discuss](https://kubeflow-discuss)
- Contribute code [github.com/kubeflow](https://github.com/kubeflow)
Uniform observability

Collect the golden signals for every service and logs for every call.

Understand services and their dependencies.

Set, monitor and enforce SLOs on services.

Bird’s eye view of service behavior for issue triage, reduce time to detect, triage.
Operational agility

Scale by directing traffic to multiple versions

Roll out new versions without worrying about ops challenges

Apply access control, rate limiting policies to protect services from bad behavior
Policy driven security

Defence in depth - security does not stop at the edge.

Enable mTLS for authentication and encryption.

Authorize access based on service identity or any channel attribute.

Configure finer grained RPC-level access control for REST and gRPC.