Technical Analysis: API Gateway Metrics & Observability Gaps

Overview

This document analyzes the current state of **metrics, monitoring coverage, and observability gaps** for the API Gateway and related token validation flows.

The goal is to:

- Highlight **metrics** needs to be added in the monitoring dashboard and alerts

---

Metrics Categories

The metrics are grouped into the following categories:

1. Success Rate Metrics
2. Usage Metrics
3. Throughput Metrics
4. Latency Metrics
5. Critical Business Metrics
6. Tenant Fairness Checks
7. Tenant-Level Error & Latency Metrics

Success Rate Metrics

HTTP Success Rate

Purpose
Measures the percentage of successful HTTP requests handled by the gateway.

PromQL Query

```
rate(http_server_requests_seconds_count{status=~"[23].."}[5m])
/ rate(http_server_requests_seconds_count[5m]) * 100
```

Analysis

- Uses a rolling 5-minute window.
- Provides a **high-level reliability indicator**.

Usage Metrics

Token Validation Count & JVM Heap Memory Usage

Purpose
Track token validation requests for authentication/authorization flows. Below metric is captured and send from the gateway.

PromQL Query

1. `Total Validate Token Request - rate(token_validation_count_total[5m])`
2. JVM Heap Memory Usage - jvm_memory_used_bytes{application="gateway", area="heap"} / jvm_memory_max_bytes{application="gateway", area="heap"} * 100

Impact

- Cannot currently measure authentication load.
- Limits insight into auth-related scaling and failures.

Throughput Metrics

Overall Request Throughput

Purpose
Measure total request volume handled by the gateway.

PromQL Query

```
sum(http_server_requests_seconds_count)
```

Analysis

- Provides cumulative throughput.
- Useful for baseline traffic analysis.

Latency Metrics

Average Response Time (Gateway)

PromQL Query

rate(http_server_requests_seconds_sum[5m]) / rate(http_server_requests_seconds_count[5m])

Analysis

- Use a rolling 5-minute window.
- Include percentile metrics (P90, P95, P99)

Critical Business Metrics

Token Validation Latency & Token Error Rate

PromQL Query

1. `Token Latency - rate(token_validation_timed_seconds_sum[5m]) / rate(token_validation_timed_seconds_count[5m])`
2. `Token Validation Error Rate - rate(token_validation_counted_total{application="gateway", result="failure"}[5m]) / rate(token_validation_counted_total{application="gateway"}[5m]) * 100`

Impact

- Without above difficult to correlate login/auth issues with user experience.

Tenant-Level Error Metrics

Tenant HTTP Error Rate (4xx & 5xx)

PromQL Query

rate(http_server_requests_seconds_count{ application="gateway", status=~"[4].." }[5m]) / rate(http_server_requests_seconds_count{ application="gateway" }[5m]) * 100

rate(http_server_requests_seconds_count{ application="gateway", status=~"[5].." }[5m]) / rate(http_server_requests_seconds_count{ application="gateway" }[5m]) * 100

Analysis

- This is the very important because the availability monitoring is current set through kube_deployment_status_replicas_available which is infrastructure level monitoring for availability, this measures of Application Health monitoring with alerting is needed.
- Captures separate 4xx and 5xx error rate.
- Suitable for high-level tenant health monitoring.

Tenant-Level Response Latency

End-to-End Latency

PromQL Query

topk(10,

sum by (uri) (rate(http_server_requests_seconds_sum{application="gateway"}[5m])) /

sum by (uri) (rate(http_server_requests_seconds_count{application="gateway"}[5m]))

)

Analysis

- Usage of other services consuming gateway

Alerting

Critical Alerts

- **Service Availability - Present**
- **High Error Rate -** Based on the error metric when the threshold is > 15% for 10 minutes

rate(http_server_requests_seconds_count{application="gateway", status=~"[45].."}[15m]) / rate(http_server_requests_seconds_count{application="gateway"}[15m]) * 100

Warning Alerts

- **Response Time Degradation** - P95 latency > 5 seconds for 15 minutes

histogram_quantile(0.95, rate(http_server_requests_seconds_bucket{application="gateway"}[15m])) > 5

- **JVM Memory Pressure** - Threshold > 85% help usage for 15 minutes , Impact might be running Out of memory

(jvm_memory_used_bytes{application="gateway", area="heap"} / jvm_memory_max_bytes{application="gateway", area="heap"}) * 100 > 85

Resource Trends

Resource trends are available, but needs tuning (filters not set to the service names in the panels) https://ukg.grafana.net/d/0VSiotDnk/k8s-cluster-health?var-interval=1m&orgId=1&from=now-6h&to=now&timezone=browser&var-dc=us-east4&var-environment=ds-dev&var-container=service-datascience-gateway&var-datasource=edi58rhsvq2v4b&refresh=5s

Story Breakdown

Jira link	Story	Description	ACs	Estimate
	API Gateway Observability Dashboard	Create monitoring dashboard to include critical gateway health, performance, and business metrics. The goal is to • Assess gateway reliability at a glance • Identify endpoint hotspots and latency bottlenecks • Understand token validation load and performance • Monitor tenant-level error rates and response times	Panels created for Success, Usage, Throughput, Latency, Business & tenant level metrics Use visualization for trending as needed. Create filter which has Datacenter(all regions), namespace(ds-*), datasource(dev /prod)	5
	API Gateway Alerting for Reliability	Implement Proactive alerting for API Gateway & Performance Degradation • Error Rate • Response Time Degradation • JVM Memory Pressure	Create critical alert for High Error Rate(5xx) - if error rate is 15% for 15 minutes (polling done 3 times in 5 min interval) Create warning alert for Response Time Degradation - if P95 > 5sec for 15 minutes(polling done 3 times in 5 min interval) Create warning alert for JVM memory pressure - if it stays 85% for 15 minutes(polling done 3 times in 5 min interval) Validate and tune the alert monitors so that the thresholds are set correctly	3