# Instaclustr Kafka Health Check Example

Instaclustr Professional Services

# Table of Contents

# Chapter 1. Kafka Health Check

## 1.1. Unclean Leader Elections (Prometheus)

 No unclean leader elections (checked 1 brokers)

**Cluster Status:** - Total Unclean Elections: 0 - Brokers Monitored: 1

No data loss from unclean elections detected. This is expected in healthy production.

## 1.2. Offline Partitions (Prometheus)

 No offline partitions (3 brokers checked)

**Cluster Summary:** - Total Offline Partitions: 0 - Brokers with Offline Partitions: 0/3

## 1.3. Under-Replicated Partitions (Prometheus)

 No under-replicated partitions (3 brokers checked)

**Cluster Summary:** - Total Under-Replicated Partitions: 0 - Brokers with URPs: 0/3 - Warning Threshold: >0 - Critical Threshold: >10

## 1.4. File Descriptor Usage (Prometheus)

 File descriptor usage healthy across 1 brokers

**Cluster Summary:** - Average FD Usage: 0.0% - Brokers at Risk: 0 critical, 0 warning - Brokers Monitored: 1

**Per-Broker FD Usage:**  Broker unknown: 1,792/9,223,372,036,854,775,808 FDs (0.0%)

## 1.5. Network Errors (Prometheus)

 No network errors detected across 1 brokers

**Cluster Summary:** - Total Errors/Drops: 0 - Brokers with Issues: 0/1

## 1.6. ISR Health (Prometheus)

 ISR health is stable across 3 brokers

**Cluster Summary:** - Total ISR Shrink Rate: 0.0/sec - Total ISR Expand Rate: 0.0/sec - Partitions Under Min ISR: 0 - Brokers Monitored: 3

# 1.7. Request Latency (Prometheus)

□□ 3 broker(s) with critical consumer latency and 3 broker(s) with critical replication latency

**Cluster Summary:** - Avg Produce Latency: 2.31ms - Avg Consumer Fetch Latency: 1503.0ms - Avg Replication Latency: 1990.33ms - Brokers Monitored: 3

□ **Critical Consumer Fetch Latency (3):** - Broker 7d867910 (100.24.74.94): 1417.0ms - Broker fc7629a0 (3.208.112.189): 1310.0ms - Broker dab6a44c (3.218.14.119): 1782.0ms *Recommendation: Investigate consumer fetch performance - reads are critically slow*

□ **Critical Replication Latency (3):** - Broker 7d867910 (100.24.74.94): 2065.0ms - Broker fc7629a0 (3.208.112.189): 2074.0ms - Broker dab6a44c (3.218.14.119): 1832.0ms *Recommendation: Investigate replication performance - followers are critically lagging*

## 1.7.1. Recommendations

□ **Critical Priority (Immediate Action):**

- Check broker resource utilization (CPU, disk I/O)
- Review disk performance - slow disks cause high latency
- Check for network issues or bandwidth saturation
- Review log segment settings (segment.ms, segment.bytes)
- Consider scaling up broker resources or adding brokers

□ **General Best Practices:**

- Latency Best Practices:
- • Monitor p99 latencies, not just averages
- • Keep produce latency < 100ms for good user experience
- • High replication latency can cause ISR shrinks
- • Use SSDs for Kafka log directories
- • Tune OS page cache and disk I/O scheduler *
- Common Causes of High Latency:
- • Disk I/O bottlenecks (slow disks, high utilization)
- • CPU saturation
- • Network congestion
- • Large message batches
- • Inefficient compression settings
- • GC pauses

# 1.8. Request Handler Saturation (Prometheus)

 1 broker(s) with critical request handler saturation (<10% idle)

Low idle % means request handlers are saturated - broker is overloaded

**Cluster Summary:** - Avg Handler Idle: 1.0% - Saturated Brokers: 1 critical, 0 warning

# 1.9. Replica Fetcher Health (Prometheus)

 Replica fetchers healthy across 1 brokers

**Cluster Summary:** - Brokers with Issues: 0/1 - Total Failed Partitions: 0

# 1.10. Partition Balance (Prometheus)

 Partitions well-balanced across 3 brokers

**Cluster Summary:** - Total Partitions: 159 - Total Leaders: 53 - Avg Partitions/Broker: 53.0 - Avg Leaders/Broker: 17.7 - Brokers: 3

**Per-Broker Distribution:** - Broker 7d867910: 53 total (18 leaders, 35 followers) (+0.0%) - Broker fc7629a0: 53 total (18 leaders, 35 followers) (+0.0%) - Broker dab6a44c: 53 total (17 leaders, 36 followers) (+0.0%)

# 1.11. Broker Health Metrics (Prometheus)

 All 3 brokers have healthy CPU and disk usage

**Cluster Summary:**

- Average CPU: 2.61%
- Average Disk: 1.45%
- Total Throughput In: 70.6 MB/s
- Total Throughput Out: 0.0 MB/s
- Brokers Monitored: 3

# 1.12. GC Health (Prometheus)

 GC metrics collected from 3 brokers

**Cluster Summary:** - Avg Young GC Time: 25172.0 ms (cumulative) - Avg Old GC Time: 0.0 ms (cumulative) - Brokers Monitored: 3

**Per-Broker GC Metrics:** - Broker 7d867910: Young GC 35448.0ms, Old GC 0.0ms - Broker fc7629a0:

Young GC 18466.0ms, Old GC 0.0ms - Broker dab6a44c: Young GC 21602.0ms, Old GC 0.0ms

**Note:** GC time values are cumulative since broker start. Monitor trends over time - rapid increases indicate memory pressure.

### 1.12.1. Recommendations

 **General Best Practices:**

- GC Monitoring Best Practices:
- • Monitor GC time trends, not absolute values
- • Rapid increases indicate memory pressure
- • Old (Full) GC is more concerning than Young GC
- • Typical healthy values: < 5% time in Young GC, < 2% in Old GC *
- Signs of GC Issues:
- • Increasing old GC frequency
- • Long GC pause times (> 100ms)
- • Heap usage consistently > 80%
- • Latency spikes correlating with GC *
- Solutions:
- • Increase heap size if frequently hitting limits
- • Use G1GC for heaps > 4GB (default in modern Kafka)
- • Review application memory usage patterns
- • Check for memory leaks
- • Consider ZGC or Shenandoah for very low pause times

## 1.13. Log Flush Performance (Prometheus)

 Log flush metrics collected from 1 brokers

**Per-Broker Flush Metrics:** - Broker unknown: Rate 0.04/s, Time 26.79ms

## 1.14. Controller Health - KRaft (Prometheus)

 Controller health good (KRaft mode)

**Controller Status:** - Fenced Brokers: 0 - Metadata Errors: 0 - Brokers Monitored: 1

## 1.15. Consumer Lag (Prometheus)

 All 3 consumer group(s) have healthy lag

**Cluster Summary:** - Total Consumer Groups: 3 - Total Lag (all groups): 180 messages - Groups with Critical Lag: 0 - Groups with Warning Lag: 0 - Healthy Groups: 3

# 1.16. Purgatory Size (Prometheus)

□□ 1 broker(s) with critical fetch purgatory

**Cluster Summary:** - Avg Produce Purgatory: 0.0 requests - Avg Fetch Purgatory: 921.0 requests - Brokers Monitored: 1

## 1.16.1. Recommendations

□ **General Best Practices:**

- Purgatory Explained:
- • Produce Purgatory: Requests waiting for acks (acks=all)
- • Fetch Purgatory: Fetch requests waiting for fetch.min.bytes *
- High Produce Purgatory Causes:
- • Slow replica replication (check ISR health)
- • acks=all with lagging replicas
- • Network issues between brokers *
- High Fetch Purgatory Causes:
- • fetch.min.bytes set too high for traffic rate
- • Low traffic periods with default fetch.min.bytes=1
- • fetch.max.wait.ms forcing waits *
- Solutions:
- • Check replication health and ISR stability
- • Review producer acks configuration
- • Tune fetch.min.bytes and fetch.max.wait.ms for consumers

# 1.17. JVM Heap Usage (Prometheus)

□ JVM heap metrics not found

# Chapter 2. OS-Level Metrics

This check requires SSH access for CPU load check.

Configure the following in your settings:

**For single host:** * `ssh_host`: Hostname or IP address

**For multiple hosts (recommended for clusters):** * `ssh_hosts`: List of hostnames/IPs

**Authentication (required):** * `ssh_user`: SSH username * `ssh_key_file` OR `ssh_password`: Authentication method

**Optional:** * `ssh_port`: SSH port (default: 22) * `ssh_timeout`: Connection timeout in seconds (default: 10)

This check requires SSH access for Memory usage check.

Configure the following in your settings:

**For single host:** * `ssh_host`: Hostname or IP address

**For multiple hosts (recommended for clusters):** * `ssh_hosts`: List of hostnames/IPs

**Authentication (required):** * `ssh_user`: SSH username * `ssh_key_file` OR `ssh_password`: Authentication method

**Optional:** * `ssh_port`: SSH port (default: 22) * `ssh_timeout`: Connection timeout in seconds (default: 10)

This check requires SSH access for File descriptor check.

Configure the following in your settings:

**For single host:** * `ssh_host`: Hostname or IP address

**For multiple hosts (recommended for clusters):** * `ssh_hosts`: List of hostnames/IPs

**Authentication (required):** * `ssh_user`: SSH username * `ssh_key_file` OR `ssh_password`: Authentication method

**Optional:** * `ssh_port`: SSH port (default: 22) * `ssh_timeout`: Connection timeout in seconds (default: 10)

# Chapter 3. Configuration Audits

This check requires SSH access for Broker config audit.

Configure the following in your settings:

**For single host:** * `ssh_host`: Hostname or IP address

**For multiple hosts (recommended for clusters):** * `ssh_hosts`: List of hostnames/IPs

**Authentication (required):** * `ssh_user`: SSH username * `ssh_key_file` OR `ssh_password`: Authentication method

**Optional:** * `ssh_port`: SSH port (default: 22) * `ssh_timeout`: Connection timeout in seconds (default: 10)

This check requires SSH access for Log error analysis.

Configure the following in your settings:

**For single host:** * `ssh_host`: Hostname or IP address

**For multiple hosts (recommended for clusters):** * `ssh_hosts`: List of hostnames/IPs

**Authentication (required):** * `ssh_user`: SSH username * `ssh_key_file` OR `ssh_password`: Authentication method

**Optional:** * `ssh_port`: SSH port (default: 22) * `ssh_timeout`: Connection timeout in seconds (default: 10)

This check requires SSH access for GC pause analysis.

Configure the following in your settings:

**For single host:** * `ssh_host`: Hostname or IP address

**For multiple hosts (recommended for clusters):** * `ssh_hosts`: List of hostnames/IPs

**Authentication (required):** * `ssh_user`: SSH username * `ssh_key_file` OR `ssh_password`: Authentication method

**Optional:** * `ssh_port`: SSH port (default: 22) * `ssh_timeout`: Connection timeout in seconds (default: 10)

# Chapter 4. Kafka Overview

## 4.1. Kafka Cluster Overview

Provides a high-level overview of the Kafka cluster.

### 4.1.1. Cluster Information

| | |
|---|---|
| Kafka Version | Apache Kafka 4.0.0 |
| Cluster ID | 9VWND7rSQByU37apX2j_Cg |
| Broker Count | 3 |
| Controller | Broker 6 |

### 4.1.2. Broker Details

*Table 1. Broker List*

| Broker ID | Host | Port | Rack |
|---|---|---|---|
| 4 | 3.208.112.189 | 9082 | us-east-1b |
| 6 | 100.24.74.94 | 9082 | us-east-1a |
| 2 | 3.218.14.119 | 9082 | us-east-1c |

### 4.1.3. Controller Status

| | |
|---|---|
| Controller Broker ID | 6 |
| Host | 100.24.74.94 |
| Port | 9082 |

# Chapter 5. Kafka Purgatory

## 5.1. Purgatory Size Monitoring (All Brokers)

This check requires SSH access for Purgatory size check.

Configure the following in your settings:

**For single host:** * `ssh_host`: Hostname or IP address

**For multiple hosts (recommended for clusters):** * `ssh_hosts`: List of hostnames/IPs

**Authentication (required):** * `ssh_user`: SSH username * `ssh_key_file` OR `ssh_password`: Authentication method

**Optional:** * `ssh_port`: SSH port (default: 22) * `ssh_timeout`: Connection timeout in seconds (default: 10)

# Chapter 6. Kafka Cluster Health

## 6.1. Partition Balance Across Brokers

Check failed: 'str' object has no attribute 'get'

# Chapter 7. Topic Management

## 7.1. Topic Count and Naming Conventions

**Naming Violations:** 1 topics do not follow naming conventions.

### 7.1.1. Topics Containing Hyphens (1)

**Best practice: Use underscores instead of hyphens**

- `instaclustr-sla` → suggested: `instaclustr_sla`

### 7.1.2. Topic Summary

| Metric | Count |
|---|---|
| Total Topics | 1 |
| Threshold | 100 |
| Topics with Violations | 1 |
| Starts with Digit | 0 |
| Contains Spaces | 0 |
| Contains Hyphens | 1 |
| Contains Uppercase | 0 |

### 7.1.3. Recommendations

**□□ High Priority (Plan Optimization):**

**Establish naming standards** - Use lowercase letters, digits, underscores, and dots only **Create a topic naming guide** - Document conventions and share with teams **Consider renaming** critical topics during maintenance windows

**□ General Best Practices:**

- Use lowercase letters, digits, underscores (_), and dots (.) only
- Start topic names with a letter, not a digit
- Avoid hyphens - they can cause issues with metrics systems like Prometheus
- Keep names descriptive but concise (e.g., `user_events`, `order_processing`)
- Use prefixes for organization (e.g., `prod_`, `staging_`)
- Regularly review topic list for compliance
- Implement automated naming validation in CI/CD pipelines

## 7.2. Topic Configuration Audit

Auditing topic-level configurations for best practices and potential issues.

Analyzing configuration for 1 topic(s).

All 0 topic configurations follow best practices.

No critical issues or warnings detected.

# Chapter 8. Performance Monitoring

## 8.1. I/O Statistics (All Brokers)

This check requires SSH access for I/O statistics check.

Configure the following in your settings:

**For single host:** * `ssh_host`: Hostname or IP address

**For multiple hosts (recommended for clusters):** * `ssh_hosts`: List of hostnames/IPs

**Authentication (required):** * `ssh_user`: SSH username * `ssh_key_file` OR `ssh_password`: Authentication method

**Optional:** * `ssh_port`: SSH port (default: 22) * `ssh_timeout`: Connection timeout in seconds (default: 10)

## 8.2. JVM Memory & GC Statistics (All Brokers)

This check requires SSH access for JVM statistics check.

Configure the following in your settings:

**For single host:** * `ssh_host`: Hostname or IP address

**For multiple hosts (recommended for clusters):** * `ssh_hosts`: List of hostnames/IPs

**Authentication (required):** * `ssh_user`: SSH username * `ssh_key_file` OR `ssh_password`: Authentication method

**Optional:** * `ssh_port`: SSH port (default: 22) * `ssh_timeout`: Connection timeout in seconds (default: 10)

# Chapter 9. Replication and ISR Health

## 9.1. In-Sync Replicas (ISR) Health Check

No topics found in cluster.

# Chapter 10. Kafka Storage Health

## 10.1. Storage Health Analysis

No storage usage data available.

This check requires SSH access for Disk usage check.

Configure the following in your settings:

**For single host:** * `ssh_host`: Hostname or IP address

**For multiple hosts (recommended for clusters):** * `ssh_hosts`: List of hostnames/IPs

**Authentication (required):** * `ssh_user`: SSH username * `ssh_key_file` OR `ssh_password`: Authentication method

**Optional:** * `ssh_port`: SSH port (default: 22) * `ssh_timeout`: Connection timeout in seconds (default: 10)

# Chapter 11. Broker Availability

## 11.1. Broker Availability

All configured brokers are available and responding.

Cluster: 9VWND7rSQByU37apX2j_Cg, Brokers: 3, Controller: Broker 6

### 11.1.1. Broker Status

| Status | Broker ID | Host | Port | Role |
|---|---|---|---|---|
| □ | 2 | 3.218.14.119 | 9082 | Available |
| □ | 4 | 3.208.112.189 | 9082 | Available |
| □ | 6 | 100.24.74.94 | 9082 | Available (Controller) |

# Chapter 12. Consumer Groups

## 12.1. Consumer Group Health

**Low Member Count:** 3 production consumer group(s) have fewer than 2 member(s). This reduces redundancy.

### 12.1.1. Production Consumer Groups

| Status | Group ID | State | Members | Total Lag | Max Partition Lag |
|---|---|---|---|---|---|
| □□ | KafkaConsumer-2 | Stable | 1 | 60 | 47 |
| □□ | KafkaConsumer-4 | Stable | 1 | 30 | 17 |
| □□ | KafkaConsumer-6 | Stable | 1 | 0 | - |

### 12.1.2. Recommendations

□□ **High Priority (Plan Optimization):**

**Scale up to 2+ members** for redundancy and better throughput **Document scaling procedures** for each consumer group

□ **General Best Practices:**

- Set up alerts for consumer lag thresholds (warning at 5k, critical at 10k)
- Monitor consumer group state changes for early warning of issues
- Implement consumer health checks in your deployment pipelines
- Document consumer group ownership and scaling policies
- Regularly review consumer processing metrics and optimize bottlenecks

# Chapter 13. Consumer Health

## 13.1. Consumer Lag Analysis

No significant consumer lag detected.

All Consumer Groups Total Lag: 90 Groups Analyzed: 3 Groups With Data: 1 Groups Without Offsets: 0 Groups With Errors: 0

| group_id | topic | partition | current_offset | log_end_offset | lag |
|---|---|---|---|---|---|
| KafkaConsumer-4 | instaclustr-sla | 0 | 74125 | 74125 | 0 |
| KafkaConsumer-4 | instaclustr-sla | 2 | 75393 | 75406 | 13 |
| KafkaConsumer-4 | instaclustr-sla | 1 | 65462 | 65479 | 17 |
| KafkaConsumer-6 | instaclustr-sla | 0 | 74125 | 74125 | 0 |
| KafkaConsumer-6 | instaclustr-sla | 2 | 75406 | 75406 | 0 |
| KafkaConsumer-6 | instaclustr-sla | 1 | 65479 | 65479 | 0 |
| KafkaConsumer-2 | instaclustr-sla | 0 | 74125 | 74125 | 0 |
| KafkaConsumer-2 | instaclustr-sla | 2 | 75393 | 75406 | 13 |
| KafkaConsumer-2 | instaclustr-sla | 1 | 65432 | 65479 | 47 |

## 13.2. AI-Generated Recommendations

Provides intelligent, context-aware recommendations based on dynamic analysis of database metrics.

### 13.2.1. AI Analysis Details

| Parameter | Value |
|---|---|
| AI Provider | xAI |
| AI Model | grok-4 |
| Prompt Size | 32,415 characters (~8,103 tokens) |

| Parameter | Value |
|---|---|
| AI Processing Time | 75.22 seconds |

# 13.3. AI-Generated Recommendations

## 13.3.1. Executive Summary

The Kafka cluster (version 4.0.0, ID: 9VWND7rSQByU37apX2j_Cg) exhibits generally stable core operations with 3 available brokers, no offline or under-replicated partitions, balanced partition distribution, healthy ISR states, low CPU/disk utilization (avg CPU ~2.61%, disk ~1.45%), and minimal consumer lag (total 180 across 3 groups). However, critical performance bottlenecks are evident in request handling and latencies: all brokers show critically high fetch consumer latency (avg 1503ms > 500ms threshold) and fetch follower latency (avg 1990ms > 1000ms threshold), one broker has severe request handler saturation (1% idle), and another has high fetch purgatory (921 delayed requests). These issues likely correlate, as saturated handlers can delay fetch operations, leading to purgatory buildup and replication/consumer lags. No unclean elections, network errors, or controller issues were detected (despite pre-analysis flags, which contradict live data showing controller ID 6 active). Urgent remediation is needed to address latency and saturation to prevent throughput degradation or data staleness. Skipped SSH-based checks (e.g., CPU load, memory) limit deeper insights; configure SSH for future analyses. Prioritize critical issues to restore optimal performance.

Correlations Identified: - High fetch consumer/follower latencies across all brokers strongly correlate with high fetch purgatory on broker at 100.24.74.94, indicating delayed data availability for consumers and replicas, possibly due to I/O bottlenecks or overload. - Request handler saturation on broker at 3.218.14.119 (<10% idle) likely exacerbates latencies, as busy handlers queue requests, contributing to purgatory and replication delays. - Low consumer group members (1 per group) may indirectly worsen lag under load, but current lags are low; no direct correlation to latencies observed. - Healthy metrics (e.g., low CPU/disk, no under-replicated partitions) suggest issues are not resource exhaustion but potentially configuration-related (e.g., fetch sizes, thread pools) or workload spikes. - Missing JVM heap metrics prevent correlation with GC activity (cumulative young GC ~25s avg), but no old GC pauses indicate heap pressure might not be primary. If latencies persist, they could lead to increased consumer lag or replication failures over time.

## 13.3.2. Critical Issues

#### Critical Fetch Consumer and Follower Latencies

**Operational Impact:** All 3 brokers exhibit critically high fetch latencies (consumer avg 1503ms, follower avg 1990ms), exceeding thresholds (500ms/1000ms). This delays data consumption and replication, risking stale data, increased lag, and potential application timeouts.

**Action Steps:** . Review workload patterns and optimize fetch configurations (e.g., increase `fetch.max.bytes` or adjust consumer poll intervals). . Inspect network throughput (bytes in/out

healthy, but verify for spikes). . Scale broker resources if correlated with saturation (see below). . Monitor via Prometheus and alert on latencies >500ms.

#### Critical Request Handler Saturation

**Operational Impact:** Broker at 3.218.14.119 has only 1% handler idle time, indicating saturation. This bottlenecks request processing, correlating with high latencies and purgatory, potentially causing cluster-wide slowdowns.

**Action Steps:** . Increase request handler threads in broker config (`num.network.threads` and `num.io.threads`). . Redistribute load using Kafka rebalance tools if imbalance detected (though partitions are balanced). . Check for misbehaving clients sending excessive requests.

Restarting the affected broker for config changes requires downtime; perform during maintenance window with replication factor ensuring no data loss.

#### Critical Fetch Purgatory Size

**Operational Impact:** Broker at 100.24.74.94 has 921 requests in fetch purgatory, delaying consumers/followers. Correlates with high latencies, indicating data not yet available (e.g., due to slow flushes or I/O), which can compound into replication issues.

**Action Steps:** . Tune log flush intervals (`log.flush.interval.ms`) to reduce delays (current flush rate 0.04, time 26.79ms). . Investigate slow fetches via logs and increase replica fetch sizes. . If persistent, consider adding brokers to distribute load.

### 13.3.3. High Issues

No high-severity issues detected beyond critical ones. Minor concerns like low consumer group members and topic naming are addressed in medium/low sections.

### 13.3.4. Medium Issues

#### Low Consumer Group Members

**Operational Impact:** All 3 consumer groups (KafkaConsumer-2, -4, -6) have only 1 member each, below recommended minimum (2) for fault tolerance. While current lag is low (total 180), this risks single points of failure during member failures, potentially increasing lag under load.

**Action Steps:** . Scale consumer applications to add members per group for parallelism and redundancy. . Monitor group states via `kafka-consumer-groups.sh --describe`. . Set alerts for groups with <2 members.

#### Missing JVM Heap Metrics

**Operational Impact:** JVM heap data unavailable from Prometheus, limiting visibility into memory usage. Cumulative GC times are collected (avg young GC 25s, no old GC), but without heap, potential memory leaks or pressure can't be assessed, possibly correlating to latencies if heap is exhausted.

**Action Steps:** . Verify Prometheus scraping config for JVM metrics (e.g., ensure

`jvm_memory_bytes_used` is exposed). . Enable JMX exporter on brokers if not configured. . As fallback, configure SSH for direct JVM stats checks (e.g., via jstat).

#### Topic Naming Violation

**Operational Impact:** The single topic "instaclustr-sla" contains a hyphen, violating naming conventions (1 violation). This is cosmetic but can cause issues in tools/scripts expecting standard names; no performance impact.

**Action Steps:** . Rename topic using `kafka-topics.sh --alter` if feasible, or create new compliant topics and migrate data. . Update naming policies to avoid hyphens, spaces, etc.

### 13.3.5. Low Issues

#### Skipped SSH-Based Checks

**Operational Impact:** Checks for CPU load, memory, file descriptors, logs, GC pauses, iostat, JVM stats, and disk usage were skipped due to missing SSH config. This gaps deeper diagnostics (e.g., potential hidden memory issues), but Prometheus metrics show healthy CPU/disk.

**Action Steps:** . Configure SSH access (host, user, key/password) for comprehensive checks. . In interim, rely on Prometheus for approximations (e.g., file descriptors healthy at 0% usage).

#### Informational GC and Log Flush Metrics

**Operational Impact:** Cumulative young GC times average 25s across brokers with no old GC, and log flush rate is low (0.04) with 26.79ms time. No immediate issues, but trends could indicate growing pressure if not monitored.

**Action Steps:** . Monitor GC trends over time; alert on old GC >0 or young GC spikes. . No urgent action unless correlated with performance drops.

#### Consumer Lag (Healthy but Monitored)

**Operational Impact:** Total lag is low (180 across groups), but partitions show minor lags (e.g., 47 on KafkaConsumer-2 partition 1). Healthy status, but could escalate if latencies persist.

**Action Steps:** . Set Prometheus alerts for lag >5000 (warning threshold). . Optimize consumer configs for faster processing.