Kafka & KSK Consumer Monitoring Specification

Дата: 28.10.2025
Цель: Мониторинг Kafka и KSK Consumer Service

Архитектура

```
┌─────────────────────────┐
│  Kafka Exporter         │ ──▶ kafka_consumergroup_lag
└─────────────────────────┘

┌─────────────────────────┐
│  KSK Consumer           │ ──▶ ksk.kafka.messages.rate
│  (Java/Micrometer)      │ ──▶ ksk.kafka.join.success_rate
│                         │ ──▶ ksk.db.put_result.duration
└─────────────────────────┘

          ↓
    ┌───────────────┐
    │  Prometheus   │
    └───────────────┘
          ↓
    ┌───────────────┐
    │   Grafana     │
    └───────────────┘
```

ЧАСТЬ 1: KAFKA EXPORTER

1. Consumer Group Lag

Назначение: Контроль отставания consumer от producer

Метрика: `kafka_consumergroup_lag`

Пороги:

- <100,000: норма
- 100,000-200,000: предупреждение
- 200,000: критично

PromQL:

```
kafka_consumergroup_lag{
  consumergroup="ksk-consumer",
  topic=~"upoa_enriched_transactions|upoa_ksk_results"
}
```

2. Messages per Second

Назначение: Скорость поступления сообщений в топик

Метрика: `kafka_topic_partition_current_offset` (rate)

PromQL:

```
rate(kafka_topic_partition_current_offset{
  topic="upoa_enriched_transactions"
}[5m])
```

Пороги:

- <1,300 msg/sec: норма
- 1,300-1,500 msg/sec: предупреждение
- 1,500 msg/sec: критично

3. Consumer Group Members

Назначение: Количество активных consumer в группе

Метрика: `kafka_consumergroup_members`

Пороги:

- 1: норма
- 0: критично (consumer остановлен)

PromQL:

```
kafka_consumergroup_members{
  consumergroup="ksk-consumer"
}
```

ЧАСТЬ 2: KSK CONSUMER (Java/Micrometer)

1. Messages Processing Rate

Назначение: Скорость обработки сообщений consumer

Метрика: `ksk_kafka_messages_processed_total`

Реализация (Spring Boot):

```
@Service
public class KskConsumerMetrics {

    private final MeterRegistry registry;
    private final Counter messagesProcessed;

    @Autowired
    public KskConsumerMetrics(MeterRegistry registry) {
        this.registry = registry;

        this.messagesProcessed = Counter.builder("ksk.kafka.messages.processed")
            .tag("topic", "all")
            .description("Total messages processed by consumer")
            .register(registry);
    }

    public void recordMessage() {
        messagesProcessed.increment();
    }
}
```

Пороги:

- 30-50 msg/sec: норма
- <10 msg/sec: предупреждение
- 200 msg/sec: аномалия

PromQL:

```
rate(ksk_kafka_messages_processed_total[5m])
```

2. Join Success Rate

Назначение: % успешных join сообщений по corrId

Метрика: `ksk_kafka_join_success_total`, `ksk_kafka_join_failure_total`

Реализация:

```
@Service
public class KskConsumerMetrics {

    private final Counter joinSuccess;
    private final Counter joinFailure;

    @Autowired
    public KskConsumerMetrics(MeterRegistry registry) {
        this.joinSuccess = Counter.builder("ksk.kafka.join.success")
            .description("Successful corrId joins")
            .register(registry);

        this.joinFailure = Counter.builder("ksk.kafka.join.failure")
            .description("Failed corrId joins")
            .register(registry);
    }

    public void recordJoinSuccess() {
        joinSuccess.increment();
    }

    public void recordJoinFailure() {
        joinFailure.increment();
    }
}
```

Пороги:

- 98%: норма
- 95-98%: предупреждение
- <95%: критично (потеря данных)

PromQL:

```
(ksk_kafka_join_success_total /
(ksk_kafka_join_success_total + ksk_kafka_join_failure_total)) * 100
```

3. Join Delay Average

Назначение: Среднее время ожидания пары сообщений

Метрика: `ksk_kafka_join_delay_seconds`

Реализация:

```
@Service
public class KskConsumerMetrics {

    private final Timer joinDelay;
    @Autowired
    public KskConsumerMetrics(MeterRegistry registry) {
        this.joinDelay = Timer.builder("ksk.kafka.join.delay")
            .description("Time waiting for message pair")
            .publishPercentiles(0.5, 0.95, 0.99)
            .register(registry);
    }

    public void recordJoinDelay(long delayMillis) {
        joinDelay.record(delayMillis, TimeUnit.MILLISECONDS);
    }
}
```

Пороги:

- <2 sec: норма
- 2-10 sec: предупреждение
- 10 sec: критично

PromQL:

```
histogram_quantile(0.95, ksk_kafka_join_delay_seconds_bucket)
```

4. Database Write Duration

Назначение: Время выполнения put_ksk_result()

Метрика: `ksk_db_put_result_duration_seconds`

Реализация:

```
@Service
public class KskDatabaseMetrics {

    private final Timer putResultDuration;

    @Autowired
    public KskDatabaseMetrics(MeterRegistry registry) {
        this.putResultDuration = Timer.builder("ksk.db.put_result.duration")
            .description("Duration of put_ksk_result() calls")
            .publishPercentiles(0.5, 0.95, 0.99)
            .register(registry);
    }

    public void recordPutResult(Callable<Void> operation) throws Exception {
        putResultDuration.recordCallable(operation);
    }
}
```

Пороги:

- <100ms (p95): норма
- 100-200ms: предупреждение
- > 200ms: критично

PromQL:

```
histogram_quantile(0.95, ksk_db_put_result_duration_seconds_bucket)
```

5. Database Write Errors

Назначение: Частота ошибок записи в БД

Метрика: `ksk_db_put_result_errors_total`

Реализация:

```
@Service
public class KskDatabaseMetrics {

    private final Counter putResultErrors;

    @Autowired
    public KskDatabaseMetrics(MeterRegistry registry) {
        this.putResultErrors = Counter.builder("ksk.db.put_result.errors")
            .description("Errors during put_ksk_result()")
            .register(registry);
    }

    public void recordError() {
        putResultErrors.increment();
    }
}
```

Пороги:

- 0: норма
- > 1%: критично

PromQL:

```
rate(ksk_db_put_result_errors_total[5m]) /
rate(ksk_kafka_messages_processed_total[5m]) * 100
```

6. Orphan Messages

Назначение: Сообщения без пары >5 минут

Метрика: `ksk_kafka_orphan_messages`

Реализация:

```
@Service
public class KskConsumerMetrics {

    @Autowired
    public KskConsumerMetrics(MeterRegistry registry) {
        Gauge.builder("ksk.kafka.orphan.messages", this,
            metrics -> getOrphanMessageCount())
            .description("Messages without pair for >5 min")
            .register(registry);
    }

    private long getOrphanMessageCount() {
        return orphanCache.countOlderThan(Duration.ofMinutes(5));
    }
}
```

Пороги:

- 0-10: норма
- 10-100: предупреждение
- 100: критично (потеря данных)

PromQL:

```
ksk_kafka_orphan_messages
```

Alerting Rules (Prometheus)

```
groups:
  - name: ksk_kafka
    interval: 30s
    rules:
      - alert: KSKKafkaLagWarning
        expr: |
          kafka_consumergroup_lag{
            consumergroup="ksk-consumer",
            topic=~"upoa_enriched_transactions|upoa_ksk_results"
          } > 100000
        for: 5m
        labels:
          severity: warning
        annotations:
```

```
      summary: "Предупреждение: лаг Kafka"
      description: "Лаг {{ $value }} сообщений (норма <100,000)"

  - alert: KSKKafkaLagCritical
    expr: |
      kafka_consumergroup_lag{
        consumergroup="ksk-consumer",
        topic=~"upoa_enriched_transactions|upoa_ksk_results"
      } > 200000
    for: 5m
    labels:
      severity: critical
    annotations:
      summary: "Критический лаг Kafka"
      description: "Лаг {{ $value }} сообщений (критично >200,000)"

  - alert: KSKKafkaMessagesRateWarning
    expr: rate(kafka_topic_partition_current_offset[5m]) > 1300
    for: 5m
    labels:
      severity: warning
    annotations:
      summary: "Высокая скорость сообщений"
      description: "{{ $value }} msg/sec (норма <1,300)"

  - alert: KSKKafkaMessagesRateCritical
    expr: rate(kafka_topic_partition_current_offset[5m]) > 1500
    for: 5m
    labels:
      severity: critical
    annotations:
      summary: "Критическая скорость сообщений"
      description: "{{ $value }} msg/sec (критично >1,500)"

  - alert: KSKConsumerGroupDown
    expr: kafka_consumergroup_members{consumergroup="ksk-consumer"} == 0
    for: 2m
    labels:
      severity: critical
    annotations:
      summary: "Consumer group остановлен"
      description: "Нет активных members в consumer group"

  - alert: KSKConsumerProcessingSlow
    expr: rate(ksk_kafka_messages_processed_total[5m]) < 10
    for: 5m
    labels:
      severity: critical
    annotations:
      summary: "Медленная обработка consumer"
      description: "{{ $value }} msg/sec (норма 30-50)"

  - alert: KSKJoinSuccessRateLow
    expr: |
      (ksk_kafka_join_success_total /
      (ksk_kafka_join_success_total + ksk_kafka_join_failure_total)) * 100 < 95
```

```
        for: 10m
        labels:
          severity: critical
        annotations:
          summary: "Низкий % успешных join"
          description: "{{ $value }}% (норма >98%)"

      - alert: KSKDatabaseWriteSlow
        expr: histogram_quantile(0.95, ksk_db_put_result_duration_seconds_bucket) > 0.2
        for: 10m
        labels:
          severity: warning
        annotations:
          summary: "Медленная запись в БД"
          description: "p95: {{ $value }}s (норма <0.1s)"

      - alert: KSKOrphanMessagesHigh
        expr: ksk_kafka_orphan_messages > 100
        for: 5m
        labels:
          severity: critical
        annotations:
          summary: "Много orphan messages"
          description: "{{ $value }} сообщений без пары"
```

Примеры Grafana Dashboards

Dashboard 1: Kafka Overview

Компоненты:

- Consumer group lag (gauge + graph)
- Messages per second (graph)
- Consumer group members (stat)
- Lag trend по партициям (heatmap)

Панели:

```
┌─────────────────────────────────────────────────────────┐
│  Kafka Overview - KSK Topics                            │
├─────────────────────────────────────────────────────────┤
│                                                         │
│  ┌──────────────────┐  ┌──────────────────┐  ┌──────────────────┐  │
│  │ Consumer Lag │  │  Messages/s  │  │   Members    │ │
│  │              │  │              │  │              │ │
│  │   85,000     │  │    450       │  │      1       │ │
│  │   ▯ Normal  │  │  ▯ Normal  │  │  ▯ Active  │ │
│  └──────────────────┘  └──────────────────┘  └──────────────────┘  │
│                                                         │
│  ┌───────────────────────────────────────────────────────────┐   │
│  │ Consumer Group Lag (24h)                      │    │
│  │                                                  │    │
│  │200K│                                             │    │
│  │150K│                                             │    │
│  │100K│ ┌┐   ┌─┐     ┌────┐                         │    │
│  │ 50K└┴┴──┴──┴─────┴────┴──────────────            │    │
│  │     00:00  06:00  12:00  18:00  24:00            │    │
│  └──────────────────────────────────────────────────────┘    │
│                                                          │
│  ┌──────────────────────────────────────────────────────┐    │
│  │ Messages per Second (by topic)                   │    │
│  │                                                  │    │
│  │1500│                                             │    │
│  │1300│────────────────────────────── threshold     │    │
│  │ 500│    upoa_enriched_transactions ████████      │    │
│  │   0└─────────────────────────────────            │    │
│  │     12:00      14:00      16:00                  │    │
│  └──────────────────────────────────────────────────────┘    │
│                                                          │
│  ┌──────────────────────────────────────────────────────┐    │
│  │ Lag by Partition (Heatmap)                       │    │
│  │                                                  │    │
│  │ Part 0 ▓▓▓▓░░░░▓▓▓▓░░░░▓▓▓░                      │    │
│  │ Part 1 ▓▓▓░░░░░▓▓▓░░░░░▓░                        │    │
│  │ Part 2 ▓░░░░░░░▓░░░░░░░                          │    │
│  │                                                  │    │
│  │ ░ <50K  ▓ 50-150K  ▓▓ >150K                      │    │
│  └──────────────────────────────────────────────────────┘    │
└──────────────────────────────────────────────────────────────────┘
```

PromQL для панелей:

```
# Consumer lag gauge
kafka_consumergroup_lag{consumergroup="ksk-consumer"}

# Messages per second
rate(kafka_topic_partition_current_offset[5m])

# Members count
kafka_consumergroup_members{consumergroup="ksk-consumer"}
```

Dashboard 2: KSK Consumer Performance

Компоненты:

- Processing rate (graph)
- Join success rate (stat + graph)
- Join delay p95 (graph)
- Database write duration p95 (graph)
- Database errors rate (stat)
- Orphan messages count (stat)

Панели:

```
┌─────────────────────────────────────────────────────────────────┐
│  KSK Consumer Performance                                │
├─────────────────────────────────────────────────────────────────┤
│                                                          │
│  ┌─────────────────┐  ┌─────────────────┐  ┌─────────────────┐  │
│  │ Processing     │   │ Join Success   │   │ DB Write p95 │  │
│  │    Rate        │   │     Rate       │   │              │  │
│  │   42 msg/s     │   │   99.2%        │   │     85ms     │  │
│  │   □ Normal   │   │  □ Good      │   │ □ Normal   │  │
│  └─────────────────┘  └─────────────────┘  └─────────────────┘  │
│                                                          │
│  ┌────────────────────────────────────────────────────────┐    │
│  │ Messages Processing Rate (5-min intervals)      │    │
│  │                                                 │    │
│  │  60│          ┌────┐                             │    │
│  │  40│     ┌─────┘    └───┐    ┌───┐                │    │
│  │  20│─────┘              └─────┘   └────          │    │
│  │   0└────────────────────────────────────────         │    │
│  │     12:00   13:00   14:00   15:00              │    │
│  └────────────────────────────────────────────────────────┘    │
│                                                          │
│  ┌────────────────────────────────────────────────────────┐    │
│  │ Join Success Rate %                             │    │
│  │                                                 │    │
│  │100%│───────────────────────────────────                │    │
│  │ 99%│████████████████████████████████                 │    │
│  │ 98%│                                             │    │
│  │ 95%├──────────────────────────────────────── threshold    │    │
│  │    └────────────────────────────────────────         │    │
│  │     12:00   13:00   14:00   15:00              │    │
│  └────────────────────────────────────────────────────────┘    │
│                                                          │
│  ┌────────────────────────────────────────────────────────┐    │
│  │ Database Write Duration (p50, p95, p99)         │    │
│  │                                                 │    │
│  │200ms│             p99 ████                       │    │
│  │100ms│             p95 ▓▓▓▓                       │    │
│  │ 50ms│             p50 ░░░░                       │    │
│  │   0└────────────────────────────────────────         │    │
│  │     12:00   13:00   14:00   15:00              │    │
│  └────────────────────────────────────────────────────────┘    │
│                                                          │
│  ┌─────────────────┐  ┌─────────────────┐  ┌─────────────────┐  │
│  │ Orphan Msgs    │   │ DB Errors/s    │   │ Join Delay   │  │
│  │                │   │                │   │     p95      │  │
│  │      5         │   │     0.00       │   │   1.2 sec    │  │
│  │   □ Normal   │   │  □ Good      │   │ □ Normal   │  │
│  └─────────────────┘  └─────────────────┘  └─────────────────┘  │
└─────────────────────────────────────────────────────────────────┘
```

PromQL для панелей:

```
# Processing rate
rate(ksk_kafka_messages_processed_total[5m])

# Join success rate
(ksk_kafka_join_success_total /
(ksk_kafka_join_success_total + ksk_kafka_join_failure_total)) * 100

# Database write duration percentiles
histogram_quantile(0.50, ksk_db_put_result_duration_seconds_bucket)
histogram_quantile(0.95, ksk_db_put_result_duration_seconds_bucket)
histogram_quantile(0.99, ksk_db_put_result_duration_seconds_bucket)

# Database errors rate
rate(ksk_db_put_result_errors_total[5m])

# Orphan messages
ksk_kafka_orphan_messages

# Join delay p95
histogram_quantile(0.95, ksk_kafka_join_delay_seconds_bucket)
```

Dashboard 3: KSK Full Pipeline

Назначение: Сквозной мониторинг всего pipeline (Kafka + Consumer + PostgreSQL)

Компоненты:

- Kafka lag (gauge)
- Consumer processing rate (gauge)
- Database TPS (gauge)
- End-to-end latency (graph)
- Error rate по компонентам (stacked graph)
- Health status (stat panel)

Панели:

```
┌────────────────────────────────────────────────────────┐
│  KSK Full Pipeline Monitoring                          │
├────────────────────────────────────────────────────────┤
│                                                        │
│  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐  │
│  │  Kafka Lag   │  │ Consumer     │  │  DB TPS      │ │
│  │              │  │ Processing   │  │              │ │
│  │   85,000     │  │  42 msg/s    │  │    45        │ │
│  │ □ <100K    │  │ □ Normal   │  │ □ Normal   │ │
│  └──────────────┘  └──────────────┘  └──────────────┘  │
│                                                        │
│  ┌──────────────────────────────────────────────┐    │
│  │ Pipeline Health Status                       │   │
│  │                                              │   │
│  │  Kafka Topics          ▯ OK                    │    │
│  │  Consumer Group        ▯ OK (1 member)         │    │
│  │  Message Processing    ▯ OK (42 msg/s)         │    │
│  │  Database Write        ▯ OK (p95: 85ms)        │    │
│  │  Join Success Rate     ▯ OK (99.2%)            │    │
│  │  Overall Status        ▯ HEALTHY               │    │
│  └──────────────────────────────────────────────────────────────┘    │
│                                                             │
│  ┌──────────────────────────────────────────────────────────────┐    │
│  │ End-to-End Latency (Kafka → DB)                 │    │
│  │                                                 │    │
│  │  3s│                                            │    │
│  │  2s│          ┌┐  ┌┐                            │    │
│  │  1s│     ┌────┘└──┘└─┐└───┐                         │    │
│  │  0s└─────┘           └────                       │    │
│  │     12:00   13:00   14:00   15:00              │    │
│  └──────────────────────────────────────────────────────────────┘    │
│                                                             │
│  ┌──────────────────────────────────────────────────────────────┐    │
│  │ Error Rate by Component                         │    │
│  │                                                 │    │
│  │ 10│                                             │    │
│  │  5│    DB errors       ▓▓▓▓                       │    │
│  │  2│    Join failures   ░░░░                       │    │
│  │  0└──────────────────────────────────────────         │    │
│  │     12:00   13:00   14:00   15:00              │    │
│  └──────────────────────────────────────────────────────────────┘    │
└──────────────────────────────────────────────────────────────────────────┘
```

Минимальные требования

Kafka Exporter: danielqsj/kafka-exporter:latest
Spring Boot: 2.7+ или 3.x
Micrometer: 1.9+
Prometheus: 2.x+
Grafana: 8.x+

⁂

1. https://www.redpanda.com/guides/kafka-performance-kafka-consumer-lag
2. https://www.redpanda.com/guides/kafka-performance-kafka-lag
3. https://socprime.com/blog/reducing-kafka-lag-optimizing-kafka-performance/
4. https://www.meshiq.com/common-kafka-performance-issues-and-how-to-fix-them/
5. https://varaisys.com/kafka-consumer/
6. https://seanglover.com/assets/Monitor Kafka Consumer Group Latency with Kafka Lag Exporter _ Lightbend.pdf
7. https://habr.com/ru/companies/otus/articles/905804/

8. https://www.instaclustr.com/education/apache-kafka/kafka-monitoring-key-metrics-and-5-tools-to-know-in-2025/