1. Kafka Consumer Properties

These properties are crucial for tuning the consumer to minimize lag.

Basic Consumer Configuration

```
bootstrap.servers=<broker-address>
group.id=<consumer-group-id>
key.deserializer=org.apache.kafka.common.serialization.StringDeserializer
value.deserializer=org.apache.kafka.common.serialization.StringDeserializer
enable.auto.commit=false
auto.offset.reset=latest # or earliest depending on your requirement
fetch.min.bytes=1 # Minimum amount of data per fetch request
fetch.max.bytes=52428800 # Max data per fetch request (default 50MB)
max.poll.records=500 # Number of records per poll
max.poll.interval.ms=300000 # Maximum time between polls (default 5 minutes)
session.timeout.ms=30000 # Session timeout for detecting consumer failure
heartbeat.interval.ms=10000 # Heartbeat interval for liveness checks
```

Advanced Consumer Configuration

```
request.timeout.ms=305000 # Request timeout for consumer-broker communication
partition.assignment.strategy=org.apache.kafka.clients.consumer.RangeAssignor
# Default strategy
client.id=<unique-client-id> # Unique ID for monitoring
```

---

2. Kafka Producer Properties

Optimizing producers can improve broker performance, which indirectly reduces consumer lag.

Basic Producer Configuration

```
bootstrap.servers=<broker-address>
key.serializer=org.apache.kafka.common.serialization.StringSerializer
value.serializer=org.apache.kafka.common.serialization.StringSerializer
acks=all # Ensure all replicas acknowledge before producer proceeds
retries=10 # Number of retries for failed sends
linger.ms=5 # Time to wait before sending batch (reduces small batch sends)
batch.size=65536 # Size of each batch in bytes (default 16KB)
compression.type=snappy # Compress messages to save bandwidth
```

Advanced Producer Configuration

```
max.in.flight.requests.per.connection=5 # Maximum unacknowledged sends per
connection
buffer.memory=33554432 # Buffer memory for the producer (default 32MB)
delivery.timeout.ms=120000 # Maximum time to attempt delivery of a record
```

---

3. Kafka Broker Configuration

Proper broker configuration ensures smooth handling of messages, reducing lag at the source.

Basic Broker Configuration

```
broker.id=<unique-broker-id>
log.dirs=/var/log/kafka # Directory for Kafka logs
num.partitions=3 # Default number of partitions per topic
num.network.threads=3 # Number of threads for handling network requests
num.io.threads=8 # Number of threads for disk I/O
log.retention.hours=168 # Log retention time in hours (default 7 days)
log.segment.bytes=1073741824 # Log segment size (default 1GB)
log.retention.bytes=-1 # Retention size (unlimited by default)
zookeeper.connect=<zookeeper-address> # Zookeeper connection string
auto.create.topics.enable=false # Disable auto topic creation
```

Performance Tuning

```
log.cleaner.threads=3 # Threads for log compaction
replica.fetch.max.bytes=10485760 # Max size of fetch requests for replicas
replica.fetch.min.bytes=1 # Min size of fetch requests for replicas
message.max.bytes=1048576 # Max size of a message (default 1MB)
```

---

4. Kafka Topic Configuration

Topic settings are crucial for balancing performance and lag.

```
min.insync.replicas=2 # Minimum replicas in sync for writes
retention.ms=604800000 # Message retention time in milliseconds (default 7
days)
segment.ms=604800000 # Maximum time for a log segment
cleanup.policy=compact # Log compaction policy
```

---

5. JVM and Host System Tuning

Ensure the Kafka environment is optimized:

- **JVM Heap Size**:
  - For brokers: `-Xmx6G -Xms6G` (set JVM heap to 50% of total system memory).
  - For consumers and producers: `-Xmx1G -Xms1G`.
- **Disk I/O**: Use SSDs for Kafka logs to improve read/write performance.
- **Networking**: Use low-latency, high-bandwidth network connections.

---

6. Monitoring and Metrics

Monitor Kafka using tools like:

- **Kafka Metrics**: Enable JMX to expose Kafka metrics.
- **Prometheus + Grafana**: Monitor lag and performance metrics.
- **Third-Party Tools**: Use Confluent Control Center, Burrow, or Cruise Control.

By applying these configurations and monitoring Kafka effectively, you can reduce consumer lag and maintain a high-performance data pipeline. Let me know if you need assistance with specific configurations!