package com.twitter.unified\_user\_actions.kafka

import com.twitter.conversions.DurationOps.\_

import com.twitter.conversions.StorageUnitOps.\_

import com.twitter.util.Duration

import com.twitter.util.StorageUnit

import org.apache.kafka.common.record.CompressionType

object ClientConfigs {

final val kafkaBootstrapServerConfig = "kafka.bootstrap.servers"

final val kafkaBootstrapServerHelp: String =

"""Kafka servers list. It is usually a WilyNs name at Twitter

""".stripMargin

final val kafkaBootstrapServerRemoteDestConfig = "kafka.bootstrap.servers.remote.dest"

final val kafkaBootstrapServerRemoteDestHelp: String =

"""Destination Kafka servers, if the sink cluster is different from the source cluster,

|i.e., read from one cluster and output to another cluster

""".stripMargin

final val kafkaApplicationIdConfig = "kafka.application.id"

final val kafkaApplicationIdHelp: String =

"""An identifier for the Kafka application. Must be unique within the Kafka cluster

""".stripMargin

// Processor in general

final val enableTrustStore = "kafka.trust.store.enable"

final val enableTrustStoreDefault = true

final val enableTrustStoreHelp = "Whether to enable trust store location"

final val trustStoreLocationConfig = "kafka.trust.store.location"

final val trustStoreLocationDefault = "/etc/tw\_truststore/messaging/kafka/client.truststore.jks"

final val trustStoreLocationHelp = "trust store location"

final val kafkaMaxPendingRequestsConfig = "kafka.max.pending.requests"

final val kafkaMaxPendingRequestsHelp = "the maximum number of concurrent pending requests."

final val kafkaWorkerThreadsConfig = "kafka.worker.threads"

final val kafkaWorkerThreadsHelp =

"""This has meaning that is dependent on the value of {@link usePerPartitionThreadPool} -

| if that is false, this is the number of parallel worker threads that will execute the processor function.

| if that is true, this is the number of parallel worker threads for each partition. So the total number of

| threads will be {@link workerThreads} \* number\_of\_partitions.

|""".stripMargin

final val retriesConfig = "kafka.retries"

final val retriesDefault = 300

final val retriesHelp: String =

"""Setting a value greater than zero will cause the client to resend any request that fails

|with a potentially transient error

""".stripMargin

final val retryBackoffConfig = "kafka.retry.backoff"

final val retryBackoffDefault: Duration = 1.seconds

final val retryBackoffHelp: String =

"""The amount of time to wait before attempting to retry a failed request to a given topic

|partition. This avoids repeatedly sending requests in a tight loop under some failure

|scenarios

""".stripMargin

// Kafka Producer

final val producerClientIdConfig = "kafka.producer.client.id"

final val producerClientIdHelp: String =

"""The client id of the Kafka producer, required for producers.

""".stripMargin

final val producerIdempotenceConfig = "kafka.producer.idempotence"

final val producerIdempotenceDefault: Boolean = false

final val producerIdempotenceHelp: String =

""""retries due to broker failures, etc., may write duplicates of the retried message in the

stream. Note that enabling idempotence requires

<code> MAX\_IN\_FLIGHT\_REQUESTS\_PER\_CONNECTION </code> to be less than or equal to 5,

<code> RETRIES\_CONFIG </code> to be greater than 0 and <code> ACKS\_CONFIG </code>

must be 'all'. If these values are not explicitly set by the user, suitable values will be

chosen. If incompatible values are set, a <code>ConfigException</code> will be thrown.

""".stripMargin

final val producerBatchSizeConfig = "kafka.producer.batch.size"

final val producerBatchSizeDefault: StorageUnit = 512.kilobytes

final val producerBatchSizeHelp: String =

"""The producer will attempt to batch records together into fewer requests whenever multiple

|records are being sent to the same partition. This helps performance on both the client and

|the server. This configuration controls the default batch size in bytes.

|No attempt will be made to batch records larger than this size.

|Requests sent to brokers will contain multiple batches, one for each partition with data

|available to be sent. A small batch size will make batching less common and may reduce

|throughput (a batch size of zero will disable batching entirely).

|A very large batch size may use memory a bit more wastefully as we will always allocate a

|buffer of the specified batch size in anticipation of additional records.

""".stripMargin

final val producerBufferMemConfig = "kafka.producer.buffer.mem"

final val producerBufferMemDefault: StorageUnit = 256.megabytes

final val producerBufferMemHelp: String =

"""The total bytes of memory the producer can use to buffer records waiting to be sent to the

|server. If records are sent faster than they can be delivered to the server the producer

|will block for MAX\_BLOCK\_MS\_CONFIG after which it will throw an exception.

|This setting should correspond roughly to the total memory the producer will use, but is not

|a hard bound since not all memory the producer uses is used for buffering.

|Some additional memory will be used for compression (if compression is enabled) as well as

|for maintaining in-flight requests.

""".stripMargin

final val producerLingerConfig = "kafka.producer.linger"

final val producerLingerDefault: Duration = 100.milliseconds

final val producerLingerHelp: String =

"""The producer groups together any records that arrive in between request transmissions into

|a single batched request. "Normally this occurs only under load when records arrive faster

|than they can be sent out. However in some circumstances the client may want to reduce the

|number of requests even under moderate load. This setting accomplishes this by adding a

|small amount of artificial delay&mdash;that is, rather than immediately sending out a record

|the producer will wait for up to the given delay to allow other records to be sent so that

|the sends can be batched together. This can be thought of as analogous to Nagle's algorithm

|in TCP. This setting gives the upper bound on the delay for batching: once we get

|BATCH\_SIZE\_CONFIG worth of records for a partition it will be sent immediately regardless

|of this setting, however if we have fewer than this many bytes accumulated for this

|partition we will 'linger' for the specified time waiting for more records to show up.

|This setting defaults to 0 (i.e. no delay). Setting LINGER\_MS\_CONFIG=5, for example,

|would have the effect of reducing the number of requests sent but would add up to 5ms of

|latency to records sent in the absence of load.

""".stripMargin

final val producerRequestTimeoutConfig = "kafka.producer.request.timeout"

final val producerRequestTimeoutDefault: Duration = 30.seconds

final val producerRequestTimeoutHelp: String =

""""The configuration controls the maximum amount of time the client will wait

|for the response of a request. If the response is not received before the timeout

|elapses the client will resend the request if necessary or fail the request if

|retries are exhausted.

""".stripMargin

final val compressionConfig = "kafka.producer.compression.type"

final val compressionDefault: CompressionTypeFlag = CompressionTypeFlag(CompressionType.NONE)

final val compressionHelp = "Producer compression type"

// Kafka Consumer

final val kafkaGroupIdConfig = "kafka.group.id"

final val kafkaGroupIdHelp: String =

"""The group identifier for the Kafka consumer

""".stripMargin

final val kafkaCommitIntervalConfig = "kafka.commit.interval"

final val kafkaCommitIntervalDefault: Duration = 10.seconds

final val kafkaCommitIntervalHelp: String =

"""The frequency with which to save the position of the processor.

""".stripMargin

final val consumerMaxPollRecordsConfig = "kafka.max.poll.records"

final val consumerMaxPollRecordsDefault: Int = 1000

final val consumerMaxPollRecordsHelp: String =

"""The maximum number of records returned in a single call to poll()

""".stripMargin

final val consumerMaxPollIntervalConfig = "kafka.max.poll.interval"

final val consumerMaxPollIntervalDefault: Duration = 5.minutes

final val consumerMaxPollIntervalHelp: String =

"""The maximum delay between invocations of poll() when using consumer group management.

This places an upper bound on the amount of time that the consumer can be idle before fetching more records.

If poll() is not called before expiration of this timeout, then the consumer is considered failed and the group

will rebalance in order to reassign the partitions to another member.

""".stripMargin

final val consumerSessionTimeoutConfig = "kafka.session.timeout"

final val consumerSessionTimeoutDefault: Duration = 1.minute

final val consumerSessionTimeoutHelp: String =

"""The timeout used to detect client failures when using Kafka's group management facility.

The client sends periodic heartbeats to indicate its liveness to the broker.

If no heartbeats are received by the broker before the expiration of this session timeout, then the broker

will remove this client from the group and initiate a rebalance. Note that the value must be in the allowable

range as configured in the broker configuration by group.min.session.timeout.ms and group.max.session.timeout.ms.

""".stripMargin

final val consumerFetchMinConfig = "kafka.consumer.fetch.min"

final val consumerFetchMinDefault: StorageUnit = 1.kilobyte

final val consumerFetchMinHelp: String =

"""The minimum amount of data the server should return for a fetch request. If insufficient

|data is available the request will wait for that much data to accumulate before answering

|the request. The default setting of 1 byte means that fetch requests are answered as soon

|as a single byte of data is available or the fetch request times out waiting for data to

|arrive. Setting this to something greater than 1 will cause the server to wait for larger

|amounts of data to accumulate which can improve server throughput a bit at the cost of

|some additional latency.

""".stripMargin

final val consumerFetchMaxConfig = "kafka.consumer.fetch.max"

final val consumerFetchMaxDefault: StorageUnit = 1.megabytes

final val consumerFetchMaxHelp: String =

"""The maximum amount of data the server should return for a fetch request. Records are

|fetched in batches by the consumer, and if the first record batch in the first non-empty

|partition of the fetch is larger than this value, the record batch will still be returned

|to ensure that the consumer can make progress. As such, this is not a absolute maximum.

|The maximum record batch size accepted by the broker is defined via message.max.bytes

|(broker config) or max.message.bytes (topic config).

|Note that the consumer performs multiple fetches in parallel.

""".stripMargin

final val consumerReceiveBufferSizeConfig = "kafka.consumer.receive.buffer.size"

final val consumerReceiveBufferSizeDefault: StorageUnit = 1.megabytes

final val consumerReceiveBufferSizeHelp: String =

"""The size of the TCP receive buffer (SO\_RCVBUF) to use when reading data.

|If the value is -1, the OS default will be used.

""".stripMargin

final val consumerApiTimeoutConfig = "kafka.consumer.api.timeout"

final val consumerApiTimeoutDefault: Duration = 120.seconds

final val consumerApiTimeoutHelp: String =

"""Specifies the timeout (in milliseconds) for consumer APIs that could block.

|This configuration is used as the default timeout for all consumer operations that do

|not explicitly accept a <code>timeout</code> parameter.";

""".stripMargin

}