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

import com.twitter.finatra.kafka.serde.ScalaSerdes

import com.twitter.finatra.kafka.serde.UnKeyed

import com.twitter.finatra.kafka.serde.UnKeyedSerde

import com.twitter.finatra.kafka.test.EmbeddedKafka

import com.twitter.finatra.kafkastreams.test.FinatraTopologyTester

import com.twitter.finatra.kafkastreams.test.TopologyFeatureTest

import com.twitter.unified\_user\_actions.enricher.EnricherFixture

import com.twitter.unified\_user\_actions.enricher.internal.thriftscala.EnrichmentEnvelop

import com.twitter.unified\_user\_actions.enricher.internal.thriftscala.EnrichmentIdType

import com.twitter.unified\_user\_actions.enricher.internal.thriftscala.EnrichmentKey

import com.twitter.unified\_user\_actions.thriftscala.UnifiedUserAction

import org.apache.kafka.clients.consumer.ConsumerRecord

import org.joda.time.DateTime

/\*\*

\* This is to test the logic where the service reads and outputs to the same Kafka cluster

\*/

class EnrichmentPlannerServiceTest extends TopologyFeatureTest {

val startTime = new DateTime("2022-10-01T00:00:00Z")

override protected lazy val topologyTester: FinatraTopologyTester = FinatraTopologyTester(

"enrichment-planner-tester",

new EnrichmentPlannerService,

startingWallClockTime = startTime,

flags = Map(

"decider.base" -> "/decider.yml",

"kafka.output.server" -> ""

)

)

private val inputTopic = topologyTester.topic(

name = EnrichmentPlannerServiceMain.InputTopic,

keySerde = UnKeyedSerde,

valSerde = ScalaSerdes.Thrift[UnifiedUserAction]

)

private val outputTopic = topologyTester.topic(

name = EnrichmentPlannerServiceMain.OutputPartitionedTopic,

keySerde = ScalaSerdes.Thrift[EnrichmentKey],

valSerde = ScalaSerdes.Thrift[EnrichmentEnvelop]

)

test("can filter unsupported events") {

new EnricherFixture {

(1L to 10L).foreach(id => {

inputTopic.pipeInput(UnKeyed, mkUUAProfileEvent(id))

})

assert(outputTopic.readAllOutput().size === 0)

}

}

test("partition key serialization should be correct") {

val key = EnrichmentKey(EnrichmentIdType.TweetId, 9999L)

val serializer = ScalaSerdes.Thrift[EnrichmentKey].serializer

val actual = serializer.serialize("test", key)

val expected = Array[Byte](8, 0, 1, 0, 0, 0, 0, 10, 0, 2, 0, 0, 0, 0, 0, 0, 39, 15, 0)

assert(actual.deep === expected.deep)

}

test("partitioned enrichment tweet event is constructed correctly") {

new EnricherFixture {

val expected = mkUUATweetEvent(888L)

inputTopic.pipeInput(UnKeyed, expected)

val actual = outputTopic.readAllOutput().head

assert(actual.key() === EnrichmentKey(EnrichmentIdType.TweetId, 888L))

assert(

actual

.value() === EnrichmentEnvelop(

expected.hashCode,

expected,

plan = tweetInfoEnrichmentPlan

))

}

}

test("partitioned enrichment tweet notification event is constructed correctly") {

new EnricherFixture {

val expected = mkUUATweetNotificationEvent(8989L)

inputTopic.pipeInput(UnKeyed, expected)

val actual = outputTopic.readAllOutput().head

assert(actual.key() === EnrichmentKey(EnrichmentIdType.TweetId, 8989L))

assert(

actual

.value() === EnrichmentEnvelop(

expected.hashCode,

expected,

plan = tweetNotificationEnrichmentPlan

))

}

}

}

/\*\*

\* This is tests the bootstrap server logic in prod. Don't add any new tests here since it is slow.

\* Use the tests above which is much quicker to be executed and and test the majority of prod logic.

\*/

class EnrichmentPlannerServiceEmbeddedKafkaTest extends TopologyFeatureTest with EmbeddedKafka {

val startTime = new DateTime("2022-10-01T00:00:00Z")

override protected lazy val topologyTester: FinatraTopologyTester = FinatraTopologyTester(

"enrichment-planner-tester",

new EnrichmentPlannerService,

startingWallClockTime = startTime,

flags = Map(

"decider.base" -> "/decider.yml",

"kafka.output.server" -> kafkaCluster.bootstrapServers(),

"kafka.output.enable.tls" -> "false"

)

)

private lazy val inputTopic = topologyTester.topic(

name = EnrichmentPlannerServiceMain.InputTopic,

keySerde = UnKeyedSerde,

valSerde = ScalaSerdes.Thrift[UnifiedUserAction]

)

private val outputTopic = kafkaTopic(

name = EnrichmentPlannerServiceMain.OutputPartitionedTopic,

keySerde = ScalaSerdes.Thrift[EnrichmentKey],

valSerde = ScalaSerdes.Thrift[EnrichmentEnvelop]

)

test("toCluster should output to expected topic & embeded cluster") {

new EnricherFixture {

inputTopic.pipeInput(UnKeyed, mkUUATweetEvent(tweetId = 1))

val records: Seq[ConsumerRecord[Array[Byte], Array[Byte]]] = outputTopic.consumeRecords(1)

assert(records.size === 1)

assert(records.head.topic() == EnrichmentPlannerServiceMain.OutputPartitionedTopic)

}

}

}