package com.twitter.timelineranker.config

import com.twitter.cortex\_tweet\_annotate.thriftscala.CortexTweetQueryService

import com.twitter.finagle.Service

import com.twitter.finagle.mtls.authentication.ServiceIdentifier

import com.twitter.finagle.service.RetryPolicy

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.finagle.thrift.ClientId

import com.twitter.finagle.thrift.ThriftClientRequest

import com.twitter.gizmoduck.thriftscala.{UserService => Gizmoduck}

import com.twitter.manhattan.v1.thriftscala.{ManhattanCoordinator => ManhattanV1}

import com.twitter.merlin.thriftscala.UserRolesService

import com.twitter.recos.user\_tweet\_entity\_graph.thriftscala.UserTweetEntityGraph

import com.twitter.search.earlybird.thriftscala.EarlybirdService

import com.twitter.socialgraph.thriftscala.SocialGraphService

import com.twitter.storehaus.Store

import com.twitter.strato.client.{Client => StratoClient}

import com.twitter.timelineranker.recap.model.ContentFeatures

import com.twitter.timelineranker.thriftscala.TimelineRanker

import com.twitter.timelines.config.ConfigUtils

import com.twitter.timelines.config.TimelinesUnderlyingClientConfiguration

import com.twitter.timelines.model.TweetId

import com.twitter.timelineservice.thriftscala.TimelineService

import com.twitter.tweetypie.thriftscala.{TweetService => TweetyPie}

import com.twitter.util.Duration

import com.twitter.util.Try

import org.apache.thrift.protocol.TCompactProtocol

abstract class UnderlyingClientConfiguration(

flags: TimelineRankerFlags,

val statsReceiver: StatsReceiver)

extends TimelinesUnderlyingClientConfiguration

with ConfigUtils {

lazy val thriftClientId: ClientId = timelineRankerClientId()

override lazy val serviceIdentifier: ServiceIdentifier = flags.getServiceIdentifier

def timelineRankerClientId(scope: Option[String] = None): ClientId = {

clientIdWithScopeOpt("timelineranker", flags.getEnv, scope)

}

def createEarlybirdClient(

scope: String,

requestTimeout: Duration,

timeout: Duration,

retryPolicy: RetryPolicy[Try[Nothing]]

): EarlybirdService.MethodPerEndpoint = {

val scopedName = s"earlybird/$scope"

methodPerEndpointClient[

EarlybirdService.ServicePerEndpoint,

EarlybirdService.MethodPerEndpoint](

thriftMuxClientBuilder(

scopedName,

protocolFactoryOption = Some(new TCompactProtocol.Factory),

requireMutualTls = true)

.dest("/s/earlybird-root-superroot/root-superroot")

.timeout(timeout)

.requestTimeout(requestTimeout)

.retryPolicy(retryPolicy)

)

}

def createEarlybirdRealtimeCgClient(

scope: String,

requestTimeout: Duration,

timeout: Duration,

retryPolicy: RetryPolicy[Try[Nothing]]

): EarlybirdService.MethodPerEndpoint = {

val scopedName = s"earlybird/$scope"

methodPerEndpointClient[

EarlybirdService.ServicePerEndpoint,

EarlybirdService.MethodPerEndpoint](

thriftMuxClientBuilder(

scopedName,

protocolFactoryOption = Some(new TCompactProtocol.Factory),

requireMutualTls = true)

.dest("/s/earlybird-rootrealtimecg/root-realtime\_cg")

.timeout(timeout)

.requestTimeout(requestTimeout)

.retryPolicy(retryPolicy)

)

}

def cortexTweetQueryServiceClient: CortexTweetQueryService.MethodPerEndpoint

def gizmoduckClient: Gizmoduck.MethodPerEndpoint

def manhattanStarbuckClient: ManhattanV1.MethodPerEndpoint

def sgsClient: SocialGraphService.MethodPerEndpoint

def timelineRankerForwardingClient: TimelineRanker.FinagledClient

def timelineServiceClient: TimelineService.MethodPerEndpoint

def tweetyPieHighQoSClient: TweetyPie.MethodPerEndpoint

def tweetyPieLowQoSClient: TweetyPie.MethodPerEndpoint

def userRolesServiceClient: UserRolesService.MethodPerEndpoint

def contentFeaturesCache: Store[TweetId, ContentFeatures]

def userTweetEntityGraphClient: UserTweetEntityGraph.FinagledClient

def stratoClient: StratoClient

def darkTrafficProxy: Option[Service[ThriftClientRequest, Array[Byte]]] = {

if (flags.darkTrafficDestination.trim.nonEmpty) {

Some(darkTrafficClient(flags.darkTrafficDestination))

} else {

None

}

}

}