package com.twitter.timelineranker.in\_network\_tweets

import com.twitter.conversions.DurationOps.\_

import com.twitter.finagle.service.RetryPolicy

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

import com.twitter.timelineranker.config.RequestScopes

import com.twitter.timelineranker.config.RuntimeConfiguration

import com.twitter.timelineranker.parameters.ConfigBuilder

import com.twitter.timelineranker.repository.CandidatesRepositoryBuilder

import com.twitter.timelineranker.visibility.SgsFollowGraphDataFields

import com.twitter.timelines.util.stats.RequestScope

import com.twitter.timelines.visibility.model.CheckedUserActorType

import com.twitter.timelines.visibility.model.ExclusionReason

import com.twitter.timelines.visibility.model.VisibilityCheckStatus

import com.twitter.timelines.visibility.model.VisibilityCheckUser

import com.twitter.util.Duration

object InNetworkTweetRepositoryBuilder {

val VisibilityRuleExclusions: Set[ExclusionReason] = Set[ExclusionReason](

ExclusionReason(

CheckedUserActorType(Some(false), VisibilityCheckUser.SourceUser),

Set(VisibilityCheckStatus.Blocked)

)

)

private val EarlybirdTimeout = 600.milliseconds

private val EarlybirdRequestTimeout = 600.milliseconds

/\*\*

\* The timeouts below are only used for the Earlybird Cluster Migration

\*/

private val EarlybirdRealtimeCGTimeout = 600.milliseconds

private val EarlybirdRealtimeCGRequestTimeout = 600.milliseconds

}

class InNetworkTweetRepositoryBuilder(config: RuntimeConfiguration, configBuilder: ConfigBuilder)

extends CandidatesRepositoryBuilder(config) {

import InNetworkTweetRepositoryBuilder.\_

override val clientSubId = "recycled\_tweets"

override val requestScope: RequestScope = RequestScopes.InNetworkTweetSource

override val followGraphDataFieldsToFetch: SgsFollowGraphDataFields.ValueSet =

SgsFollowGraphDataFields.ValueSet(

SgsFollowGraphDataFields.FollowedUserIds,

SgsFollowGraphDataFields.MutuallyFollowingUserIds,

SgsFollowGraphDataFields.MutedUserIds,

SgsFollowGraphDataFields.RetweetsMutedUserIds

)

override val searchProcessingTimeout: Duration = 200.milliseconds

override def earlybirdClient(scope: String): EarlybirdService.MethodPerEndpoint =

config.underlyingClients.createEarlybirdClient(

scope = scope,

requestTimeout = EarlybirdRequestTimeout,

timeout = EarlybirdTimeout,

retryPolicy = RetryPolicy.Never

)

private lazy val searchClientForSourceTweets =

newSearchClient(clientId = clientSubId + "\_source\_tweets")

/\*\* The RealtimeCG clients below are only used for the Earlybird Cluster Migration \*/

private def earlybirdRealtimeCGClient(scope: String): EarlybirdService.MethodPerEndpoint =

config.underlyingClients.createEarlybirdRealtimeCgClient(

scope = scope,

requestTimeout = EarlybirdRealtimeCGRequestTimeout,

timeout = EarlybirdRealtimeCGTimeout,

retryPolicy = RetryPolicy.Never

)

private val realtimeCGClientSubId = "realtime\_cg\_recycled\_tweets"

private lazy val searchRealtimeCGClient =

newSearchClient(earlybirdRealtimeCGClient, clientId = realtimeCGClientSubId)

def apply(): InNetworkTweetRepository = {

val inNetworkTweetSource = new InNetworkTweetSource(

gizmoduckClient,

searchClient,

searchClientForSourceTweets,

tweetyPieHighQoSClient,

userMetadataClient,

followGraphDataProvider,

config.underlyingClients.contentFeaturesCache,

clientFactories.visibilityEnforcerFactory.apply(

VisibilityRules,

RequestScopes.InNetworkTweetSource,

reasonsToExclude = InNetworkTweetRepositoryBuilder.VisibilityRuleExclusions

),

config.statsReceiver

)

val inNetworkTweetRealtimeCGSource = new InNetworkTweetSource(

gizmoduckClient,

searchRealtimeCGClient,

searchClientForSourceTweets, // do not migrate source\_tweets as they are sharded by TweetID

tweetyPieHighQoSClient,

userMetadataClient,

followGraphDataProvider,

config.underlyingClients.contentFeaturesCache,

clientFactories.visibilityEnforcerFactory.apply(

VisibilityRules,

RequestScopes.InNetworkTweetSource,

reasonsToExclude = InNetworkTweetRepositoryBuilder.VisibilityRuleExclusions

),

config.statsReceiver.scope("replacementRealtimeCG")

)

new InNetworkTweetRepository(inNetworkTweetSource, inNetworkTweetRealtimeCGSource)

}

}