package com.twitter.home\_mixer.module

import com.google.inject.Provides

import com.twitter.conversions.DurationOps.\_

import com.twitter.finagle.ThriftMux

import com.twitter.finagle.builder.ClientBuilder

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

import com.twitter.finagle.mtls.client.MtlsStackClient.\_

import com.twitter.finagle.service.RetryPolicy

import com.twitter.finagle.ssl.OpportunisticTls

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.inject.TwitterModule

import com.twitter.manhattan.v2.thriftscala.{ManhattanCoordinator => ManhattanV2}

import com.twitter.timelinemixer.clients.manhattan.InjectionHistoryClient

import com.twitter.timelinemixer.clients.manhattan.ManhattanDatasetConfig

import com.twitter.timelines.clients.manhattan.Dataset

import com.twitter.timelines.clients.manhattan.ManhattanClient

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

import javax.inject.Singleton

import org.apache.thrift.protocol.TBinaryProtocol

import com.twitter.timelines.config.TimelinesUnderlyingClientConfiguration.ConnectTimeout

import com.twitter.timelines.config.TimelinesUnderlyingClientConfiguration.TCPConnectTimeout

object InjectionHistoryClientModule extends TwitterModule {

private val ProdDataset = "suggestion\_history"

private val StagingDataset = "suggestion\_history\_nonprod"

private val AppId = "twitter\_suggests"

private val ServiceName = "manhattan.omega"

private val OmegaManhattanDest = "/s/manhattan/omega.native-thrift"

private val InjectionRequestScope = RequestScope("injectionHistoryClient")

private val RequestTimeout = 75.millis

private val Timeout = 150.millis

val retryPolicy = RetryPolicy.tries(

2,

RetryPolicy.TimeoutAndWriteExceptionsOnly

.orElse(RetryPolicy.ChannelClosedExceptionsOnly))

@Provides

@Singleton

def providesInjectionHistoryClient(

serviceIdentifier: ServiceIdentifier,

statsReceiver: StatsReceiver

) = {

val dataset = serviceIdentifier.environment.toLowerCase match {

case "prod" => ProdDataset

case \_ => StagingDataset

}

val thriftMuxClient = ClientBuilder()

.name(ServiceName)

.daemon(daemonize = true)

.failFast(enabled = true)

.retryPolicy(retryPolicy)

.tcpConnectTimeout(TCPConnectTimeout)

.connectTimeout(ConnectTimeout)

.dest(OmegaManhattanDest)

.requestTimeout(RequestTimeout)

.timeout(Timeout)

.stack(ThriftMux.client

.withMutualTls(serviceIdentifier)

.withOpportunisticTls(OpportunisticTls.Required))

.build()

val manhattanOmegaClient = new ManhattanV2.FinagledClient(

service = thriftMuxClient,

protocolFactory = new TBinaryProtocol.Factory(),

serviceName = ServiceName,

)

val readOnlyMhClient = new ManhattanClient(

appId = AppId,

manhattan = manhattanOmegaClient,

requestScope = InjectionRequestScope,

serviceName = ServiceName,

statsReceiver = statsReceiver

).readOnly

val mhDatasetConfig = new ManhattanDatasetConfig {

override val SuggestionHistoryDataset = Dataset(dataset)

}

new InjectionHistoryClient(

readOnlyMhClient,

mhDatasetConfig

)

}

}