package com.twitter.frigate.pushservice

import com.google.inject.Inject

import com.google.inject.Singleton

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

import com.twitter.finagle.thrift.ClientId

import com.twitter.finatra.thrift.routing.ThriftWarmup

import com.twitter.util.logging.Logging

import com.twitter.inject.utils.Handler

import com.twitter.frigate.pushservice.{thriftscala => t}

import com.twitter.frigate.thriftscala.NotificationDisplayLocation

import com.twitter.util.Stopwatch

import com.twitter.scrooge.Request

import com.twitter.scrooge.Response

import com.twitter.util.Return

import com.twitter.util.Throw

import com.twitter.util.Try

/\*\*

\* Warms up the refresh request path.

\* If service is running as pushservice-send then the warmup does nothing.

\*

\* When making the warmup refresh requests we

\* - Set skipFilters to true to execute as much of the request path as possible

\* - Set darkWrite to true to prevent sending a push

\*/

@Singleton

class PushMixerThriftServerWarmupHandler @Inject() (

warmup: ThriftWarmup,

serviceIdentifier: ServiceIdentifier)

extends Handler

with Logging {

private val clientId = ClientId("thrift-warmup-client")

def handle(): Unit = {

val refreshServices = Set(

"frigate-pushservice",

"frigate-pushservice-canary",

"frigate-pushservice-canary-control",

"frigate-pushservice-canary-treatment"

)

val isRefresh = refreshServices.contains(serviceIdentifier.service)

if (isRefresh && !serviceIdentifier.isLocal) refreshWarmup()

}

def refreshWarmup(): Unit = {

val elapsed = Stopwatch.start()

val testIds = Seq(

1,

2,

3

)

try {

clientId.asCurrent {

testIds.foreach { id =>

val warmupReq = warmupQuery(id)

info(s"Sending warm-up request to service with query: $warmupReq")

warmup.sendRequest(

method = t.PushService.Refresh,

req = Request(t.PushService.Refresh.Args(warmupReq)))(assertWarmupResponse)

}

}

} catch {

case e: Throwable =>

error(e.getMessage, e)

}

info(s"Warm up complete. Time taken: ${elapsed().toString}")

}

private def warmupQuery(userId: Long): t.RefreshRequest = {

t.RefreshRequest(

userId = userId,

notificationDisplayLocation = NotificationDisplayLocation.PushToMobileDevice,

context = Some(

t.PushContext(

skipFilters = Some(true),

darkWrite = Some(true)

))

)

}

private def assertWarmupResponse(

result: Try[Response[t.PushService.Refresh.SuccessType]]

): Unit = {

result match {

case Return(\_) => // ok

case Throw(exception) =>

warn("Error performing warm-up request.")

error(exception.getMessage, exception)

}

}

}