package com.twitter.follow\_recommendations.modules

import com.google.inject.Provides

import com.google.inject.Singleton

import com.twitter.inject.annotations.Flag

import com.twitter.decider.RandomRecipient

import com.twitter.finagle.ThriftMux

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

import com.twitter.finagle.mtls.client.MtlsStackClient.MtlsThriftMuxClientSyntax

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.finagle.thrift.ClientId

import com.twitter.finatra.annotations.DarkTrafficService

import com.twitter.follow\_recommendations.configapi.deciders.DeciderKey

import com.twitter.follow\_recommendations.thriftscala.FollowRecommendationsThriftService

import com.twitter.inject.TwitterModule

import com.twitter.inject.thrift.filters.DarkTrafficFilter

import com.twitter.servo.decider.DeciderGateBuilder

object DiffyModule extends TwitterModule {

// diffy.dest is defined in the Follow Recommendations Service aurora file

// and points to the Dark Traffic Proxy server

private val destFlag =

flag[String]("diffy.dest", "/$/nil", "Resolvable name of diffy-service or proxy")

@Provides

@Singleton

@DarkTrafficService

def provideDarkTrafficService(

serviceIdentifier: ServiceIdentifier

): FollowRecommendationsThriftService.ReqRepServicePerEndpoint = {

ThriftMux.client

.withClientId(ClientId("follow\_recos\_service\_darktraffic\_proxy\_client"))

.withMutualTls(serviceIdentifier)

.servicePerEndpoint[FollowRecommendationsThriftService.ReqRepServicePerEndpoint](

dest = destFlag(),

label = "darktrafficproxy"

)

}

@Provides

@Singleton

def provideDarkTrafficFilter(

@DarkTrafficService darkService: FollowRecommendationsThriftService.ReqRepServicePerEndpoint,

deciderGateBuilder: DeciderGateBuilder,

statsReceiver: StatsReceiver,

@Flag("environment") env: String

): DarkTrafficFilter[FollowRecommendationsThriftService.ReqRepServicePerEndpoint] = {

// sampleFunction is used to determine which requests should get replicated

// to the dark traffic proxy server

val sampleFunction: Any => Boolean = { \_ =>

// check whether the current FRS instance is deployed in production

env match {

case "prod" =>

statsReceiver.scope("provideDarkTrafficFilter").counter("prod").incr()

destFlag.isDefined && deciderGateBuilder

.keyToFeature(DeciderKey.EnableTrafficDarkReading).isAvailable(RandomRecipient)

case \_ =>

statsReceiver.scope("provideDarkTrafficFilter").counter("devel").incr()

// replicate zero requests if in non-production environment

false

}

}

new DarkTrafficFilter[FollowRecommendationsThriftService.ReqRepServicePerEndpoint](

darkService,

sampleFunction,

forwardAfterService = true,

statsReceiver.scope("DarkTrafficFilter"),

lookupByMethod = true

)

}

}