package com.twitter.timelineranker.config

import com.twitter.app.Flags

import com.twitter.finagle.mtls.authentication.EmptyServiceIdentifier

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

import com.twitter.timelines.config.CommonFlags

import com.twitter.timelines.config.ConfigUtils

import com.twitter.timelines.config.Datacenter

import com.twitter.timelines.config.Env

import com.twitter.timelines.config.ProvidesServiceIdentifier

import java.net.InetSocketAddress

import com.twitter.app.Flag

class TimelineRankerFlags(flag: Flags)

extends CommonFlags(flag)

with ConfigUtils

with ProvidesServiceIdentifier {

val dc: Flag[String] = flag(

"dc",

"smf1",

"Name of data center in which this instance will execute"

)

val environment: Flag[String] = flag(

"environment",

"devel",

"The mesos environment in which this instance will be running"

)

val maxConcurrency: Flag[Int] = flag(

"maxConcurrency",

200,

"Maximum concurrent requests"

)

val servicePort: Flag[InetSocketAddress] = flag(

"service.port",

new InetSocketAddress(8287),

"Port number that this thrift service will listen on"

)

val serviceCompactPort: Flag[InetSocketAddress] = flag(

"service.compact.port",

new InetSocketAddress(8288),

"Port number that the TCompactProtocol-based thrift service will listen on"

)

val serviceIdentifier: Flag[ServiceIdentifier] = flag[ServiceIdentifier](

"service.identifier",

EmptyServiceIdentifier,

"service identifier for this service for use with mutual TLS, " +

"format is expected to be -service.identifier=\"role:service:environment:zone\""

)

val opportunisticTlsLevel = flag[String](

"opportunistic.tls.level",

"desired",

"The server's OpportunisticTls level."

)

val requestRateLimit: Flag[Double] = flag[Double](

"requestRateLimit",

1000.0,

"Request rate limit to be used by the client request authorizer"

)

val enableThriftmuxCompression = flag(

"enableThriftmuxServerCompression",

true,

"build server with thriftmux compression enabled"

)

def getDatacenter: Datacenter.Value = getDC(dc())

def getEnv: Env.Value = getEnv(environment())

override def getServiceIdentifier: ServiceIdentifier = serviceIdentifier()

}