package com.twitter.frigate.pushservice.module

import com.google.inject.Provides

import com.google.inject.Singleton

import com.twitter.abdecider.LoggingABDecider

import com.twitter.decider.Decider

import com.twitter.featureswitches.v2.FeatureSwitches

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

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.finagle.tunable.StandardTunableMap

import com.twitter.frigate.pushservice.config.DeployConfig

import com.twitter.frigate.pushservice.config.ProdConfig

import com.twitter.frigate.pushservice.config.StagingConfig

import com.twitter.frigate.pushservice.params.ShardParams

import com.twitter.inject.TwitterModule

import com.twitter.inject.annotations.Flag

import com.twitter.product\_mixer.core.module.product\_mixer\_flags.ProductMixerFlagModule.ConfigRepoLocalPath

import com.twitter.product\_mixer.core.module.product\_mixer\_flags.ProductMixerFlagModule.ServiceLocal

object DeployConfigModule extends TwitterModule {

@Provides

@Singleton

def providesDeployConfig(

@Flag(FlagName.numShards) numShards: Int,

@Flag(FlagName.shardId) shardId: Int,

@Flag(FlagName.isInMemCacheOff) inMemCacheOff: Boolean,

@Flag(ServiceLocal) isServiceLocal: Boolean,

@Flag(ConfigRepoLocalPath) localConfigRepoPath: String,

serviceIdentifier: ServiceIdentifier,

decider: Decider,

abDecider: LoggingABDecider,

featureSwitches: FeatureSwitches,

statsReceiver: StatsReceiver

): DeployConfig = {

val tunableMap = if (serviceIdentifier.service.contains("canary")) {

StandardTunableMap(id = "frigate-pushservice-canary")

} else { StandardTunableMap(id = serviceIdentifier.service) }

val shardParams = ShardParams(numShards, shardId)

serviceIdentifier.environment match {

case "devel" | "staging" =>

StagingConfig(

isServiceLocal = isServiceLocal,

localConfigRepoPath = localConfigRepoPath,

inMemCacheOff = inMemCacheOff,

decider = decider,

abDecider = abDecider,

featureSwitches = featureSwitches,

serviceIdentifier = serviceIdentifier,

tunableMap = tunableMap,

shardParams = shardParams

)(statsReceiver)

case "prod" =>

ProdConfig(

isServiceLocal = isServiceLocal,

localConfigRepoPath = localConfigRepoPath,

inMemCacheOff = inMemCacheOff,

decider = decider,

abDecider = abDecider,

featureSwitches = featureSwitches,

serviceIdentifier = serviceIdentifier,

tunableMap = tunableMap,

shardParams = shardParams

)(statsReceiver)

case env => throw new Exception(s"Unknown environment $env")

}

}

}