package com.twitter.tweetypie

package config

import com.twitter.decider.Decider

import com.twitter.tweetypie.decider.DeciderGates

object TweetypieDeciderGates {

def apply(

\_decider: Decider,

\_overrides: Map[String, Boolean] = Map.empty

): TweetypieDeciderGates =

new TweetypieDeciderGates {

override def decider: Decider = \_decider

override def overrides: Map[String, Boolean] = \_overrides

override def prefix: String = "tweetypie"

}

}

trait TweetypieDeciderGates extends DeciderGates {

val checkSpamOnRetweet: Gate[Unit] = linear("check\_spam\_on\_retweet")

val checkSpamOnTweet: Gate[Unit] = linear("check\_spam\_on\_tweet")

val delayEraseUserTweets: Gate[Unit] = linear("delay\_erase\_user\_tweets")

val denyNonTweetPermalinks: Gate[Unit] = linear("deny\_non\_tweet\_permalinks")

val enableCommunityTweetCreates: Gate[Unit] = linear("enable\_community\_tweet\_creates")

val useConversationControlFeatureSwitchResults: Gate[Unit] = linear(

"conversation\_control\_use\_feature\_switch\_results")

val enableExclusiveTweetControlValidation: Gate[Unit] = linear(

"enable\_exclusive\_tweet\_control\_validation")

val enableTrustedFriendsControlValidation: Gate[Unit] = linear(

"enable\_trusted\_friends\_control\_validation"

)

val enableStaleTweetValidation: Gate[Unit] = linear(

"enable\_stale\_tweet\_validation"

)

val enforceRateLimitedClients: Gate[Unit] = linear("enforce\_rate\_limited\_clients")

val failClosedInVF: Gate[Unit] = linear("fail\_closed\_in\_vf")

val forkDarkTraffic: Gate[Unit] = linear("fork\_dark\_traffic")

val hydrateConversationMuted: Gate[Unit] = linear("hydrate\_conversation\_muted")

val hydrateCounts: Gate[Unit] = linear("hydrate\_counts")

val hydratePreviousCounts: Gate[Unit] = linear("hydrate\_previous\_counts")

val hydrateDeviceSources: Gate[Unit] = linear("hydrate\_device\_sources")

val hydrateEscherbirdAnnotations: Gate[Unit] = linear("hydrate\_escherbird\_annotations")

val hydrateGnipProfileGeoEnrichment: Gate[Unit] = linear("hydrate\_gnip\_profile\_geo\_enrichment")

val hydrateHasMedia: Gate[Unit] = linear("hydrate\_has\_media")

val hydrateMedia: Gate[Unit] = linear("hydrate\_media")

val hydrateMediaRefs: Gate[Unit] = linear("hydrate\_media\_refs")

val hydrateMediaTags: Gate[Unit] = linear("hydrate\_media\_tags")

val hydratePastedMedia: Gate[Unit] = linear("hydrate\_pasted\_media")

val hydratePerspectives: Gate[Unit] = linear("hydrate\_perspectives")

val hydratePerspectivesEditsForTimelines: Gate[Unit] = linear(

"hydrate\_perspectives\_edits\_for\_timelines")

val hydratePerspectivesEditsForTweetDetail: Gate[Unit] = linear(

"hydrate\_perspectives\_edits\_for\_tweet\_details")

val hydratePerspectivesEditsForOtherSafetyLevels: Gate[Unit] =

linear("hydrate\_perspectives\_edits\_for\_other\_levels")

val hydratePlaces: Gate[Unit] = linear("hydrate\_places")

val hydrateScrubEngagements: Gate[Unit] = linear("hydrate\_scrub\_engagements")

val jiminyDarkRequests: Gate[Unit] = linear("jiminy\_dark\_requests")

val logCacheExceptions: Gate[Unit] = linear("log\_cache\_exceptions")

val logReads: Gate[Unit] = linear("log\_reads")

val logTweetCacheWrites: Gate[TweetId] = byId("log\_tweet\_cache\_writes")

val logWrites: Gate[Unit] = linear("log\_writes")

val logYoungTweetCacheWrites: Gate[TweetId] = byId("log\_young\_tweet\_cache\_writes")

val maxRequestWidthEnabled: Gate[Unit] = linear("max\_request\_width\_enabled")

val mediaRefsHydratorIncludePastedMedia: Gate[Unit] = linear(

"media\_refs\_hydrator\_include\_pasted\_media")

val rateLimitByLimiterService: Gate[Unit] = linear("rate\_limit\_by\_limiter\_service")

val rateLimitTweetCreationFailure: Gate[Unit] = linear("rate\_limit\_tweet\_creation\_failure")

val replicateReadsToATLA: Gate[Unit] = linear("replicate\_reads\_to\_atla")

val replicateReadsToPDXA: Gate[Unit] = linear("replicate\_reads\_to\_pdxa")

val disableInviteViaMention: Gate[Unit] = linear("disable\_invite\_via\_mention")

val shedReadTrafficVoluntarily: Gate[Unit] = linear("shed\_read\_traffic\_voluntarily")

val preferForwardedServiceIdentifierForClientId: Gate[Unit] =

linear("prefer\_forwarded\_service\_identifier\_for\_client\_id")

val enableRemoveUnmentionedImplicitMentions: Gate[Unit] = linear(

"enable\_remove\_unmentioned\_implicit\_mentions")

val validateCardRefAttachmentAndroid: Gate[Unit] = linear("validate\_card\_ref\_attachment\_android")

val validateCardRefAttachmentNonAndroid: Gate[Unit] = linear(

"validate\_card\_ref\_attachment\_non\_android")

val tweetVisibilityLibraryEnableParityTest: Gate[Unit] = linear(

"tweet\_visibility\_library\_enable\_parity\_test")

val enableVfFeatureHydrationInQuotedTweetVLShim: Gate[Unit] = linear(

"enable\_vf\_feature\_hydration\_in\_quoted\_tweet\_visibility\_library\_shim")

val disablePromotedTweetEdit: Gate[Unit] = linear("disable\_promoted\_tweet\_edit")

val shouldMaterializeContainers: Gate[Unit] = linear("should\_materialize\_containers")

val checkTwitterBlueSubscriptionForEdit: Gate[Unit] = linear(

"check\_twitter\_blue\_subscription\_for\_edit")

val hydrateBookmarksCount: Gate[Long] = byId("hydrate\_bookmarks\_count")

val hydrateBookmarksPerspective: Gate[Long] = byId("hydrate\_bookmarks\_perspective")

val setEditTimeWindowToSixtyMinutes: Gate[Unit] = linear("set\_edit\_time\_window\_to\_sixty\_minutes")

}