package com.twitter.search.earlybird.common.config;

import java.lang.reflect.Modifier;

import java.util.Arrays;

import java.util.List;

import java.util.function.BiFunction;

import java.util.function.Function;

import java.util.stream.Collectors;

import com.google.common.collect.ImmutableList;

import com.twitter.app.Flag;

import com.twitter.app.Flaggable;

import com.twitter.app.Flags;

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

/\*\*

\* Stateless class that represents an Earlybird property that can be specified by a command line

\* flag.

\* <p>

\* This is a regular Java class instead of enum to have a generic type.

\*

\* @param <T>

\*/

public final class EarlybirdProperty<T> {

private static final class PropertyType<T> {

private static final PropertyType<Boolean> BOOLEAN = new PropertyType<>(

Flaggable.ofJavaBoolean(), EarlybirdConfig::getBool, EarlybirdConfig::getBool);

private static final PropertyType<Integer> INT = new PropertyType<>(

Flaggable.ofJavaInteger(), EarlybirdConfig::getInt, EarlybirdConfig::getInt);

private static final PropertyType<String> STRING = new PropertyType<>(

Flaggable.ofString(), EarlybirdConfig::getString, EarlybirdConfig::getString);

private final Flaggable<T> flaggable;

private final Function<String, T> getter;

private final BiFunction<String, T, T> getterWithDefault;

private PropertyType(Flaggable<T> flaggable, Function<String, T> getter,

BiFunction<String, T, T> getterWithDefault) {

this.flaggable = flaggable;

this.getter = getter;

this.getterWithDefault = getterWithDefault;

}

}

public static final EarlybirdProperty<String> PENGUIN\_VERSION =

new EarlybirdProperty<>(

"penguin\_version",

"The penguin version to index.",

PropertyType.STRING,

false);

public static final EarlybirdProperty<Integer> THRIFT\_PORT = new EarlybirdProperty<>(

"thrift\_port",

"override thrift port from config file",

PropertyType.INT,

false);

public static final EarlybirdProperty<Integer> WARMUP\_THRIFT\_PORT = new EarlybirdProperty<>(

"warmup\_thrift\_port",

"override warmup thrift port from config file",

PropertyType.INT,

false);

public static final EarlybirdProperty<Integer> SEARCHER\_THREADS = new EarlybirdProperty<>(

"searcher\_threads",

"override number of searcher threads from config file",

PropertyType.INT,

false);

public static final EarlybirdProperty<String> EARLYBIRD\_TIER = new EarlybirdProperty<>(

"earlybird\_tier",

"the earlybird tier (e.g. tier1), used on Aurora",

PropertyType.STRING,

true);

public static final EarlybirdProperty<Integer> REPLICA\_ID = new EarlybirdProperty<>(

"replica\_id",

"the ID in a partition, used on Aurora",

PropertyType.INT,

true);

public static final EarlybirdProperty<Integer> PARTITION\_ID = new EarlybirdProperty<>(

"partition\_id",

"partition ID, used on Aurora",

PropertyType.INT,

true);

public static final EarlybirdProperty<Integer> NUM\_PARTITIONS = new EarlybirdProperty<>(

"num\_partitions",

"number of partitions, used on Aurora",

PropertyType.INT,

true);

public static final EarlybirdProperty<Integer> NUM\_INSTANCES = new EarlybirdProperty<>(

"num\_instances",

"number of instances in the job, used on Aurora",

PropertyType.INT,

true);

public static final EarlybirdProperty<Integer> SERVING\_TIMESLICES = new EarlybirdProperty<>(

"serving\_timeslices",

"number of time slices to serve, used on Aurora",

PropertyType.INT,

true);

public static final EarlybirdProperty<String> ROLE = new EarlybirdProperty<>(

"role",

"Role in the service path of Earlybird",

PropertyType.STRING,

true,

true);

public static final EarlybirdProperty<String> EARLYBIRD\_NAME = new EarlybirdProperty<>(

"earlybird\_name",

"Name in the service path of Earlybird without hash partition suffix",

PropertyType.STRING,

true,

true);

public static final EarlybirdProperty<String> ENV = new EarlybirdProperty<>(

"env",

"Environment in the service path of Earlybird",

PropertyType.STRING,

true,

true);

public static final EarlybirdProperty<String> ZONE = new EarlybirdProperty<>(

"zone",

"Zone (data center) in the service path of Earlybird",

PropertyType.STRING,

true,

true);

public static final EarlybirdProperty<String> DL\_URI = new EarlybirdProperty<>(

"dl\_uri",

"DistributedLog URI for default DL reader",

PropertyType.STRING,

false);

public static final EarlybirdProperty<String> USER\_UPDATES\_DL\_URI = new EarlybirdProperty<>(

"user\_updates\_dl\_uri",

"DistributedLog URI for user updates DL reader",

PropertyType.STRING,

false);

public static final EarlybirdProperty<String> ANTISOCIAL\_USERUPDATES\_DL\_STREAM =

new EarlybirdProperty<>(

"antisocial\_userupdates\_dl\_stream",

"DL stream name for antisocial user updates without DL version suffix",

PropertyType.STRING,

false);

public static final EarlybirdProperty<String> ZK\_APP\_ROOT = new EarlybirdProperty<>(

"zk\_app\_root",

"SZooKeeper base root path for this application",

PropertyType.STRING,

true);

public static final EarlybirdProperty<Boolean> SEGMENT\_LOAD\_FROM\_HDFS\_ENABLED =

new EarlybirdProperty<>(

"segment\_load\_from\_hdfs\_enabled",

"Whether to load segment data from HDFS",

PropertyType.BOOLEAN,

false);

public static final EarlybirdProperty<Boolean> SEGMENT\_FLUSH\_TO\_HDFS\_ENABLED =

new EarlybirdProperty<>(

"segment\_flush\_to\_hdfs\_enabled",

"Whether to flush segment data to HDFS",

PropertyType.BOOLEAN,

false);

public static final EarlybirdProperty<String> HDFS\_SEGMENT\_SYNC\_DIR = new EarlybirdProperty<>(

"hdfs\_segment\_sync\_dir",

"HDFS directory to sync segment data",

PropertyType.STRING,

false);

public static final EarlybirdProperty<String> HDFS\_SEGMENT\_UPLOAD\_DIR = new EarlybirdProperty<>(

"hdfs\_segment\_upload\_dir",

"HDFS directory to upload segment data",

PropertyType.STRING,

false);

public static final EarlybirdProperty<Boolean> ARCHIVE\_DAILY\_STATUS\_BATCH\_FLUSHING\_ENABLED =

new EarlybirdProperty<>(

"archive\_daily\_status\_batch\_flushing\_enabled",

"Whether to enable archive daily status batch flushing",

PropertyType.BOOLEAN,

false);

public static final EarlybirdProperty<String> HDFS\_INDEX\_SYNC\_DIR = new EarlybirdProperty<>(

"hdfs\_index\_sync\_dir",

"HDFS directory to sync index data",

PropertyType.STRING,

true);

public static final EarlybirdProperty<Boolean> READ\_INDEX\_FROM\_PROD\_LOCATION =

new EarlybirdProperty<>(

"read\_index\_from\_prod\_location",

"Read index from prod to speed up startup on staging / loadtest",

PropertyType.BOOLEAN,

false);

public static final EarlybirdProperty<Boolean> USE\_DECIDER\_OVERLAY = new EarlybirdProperty<>(

"use\_decider\_overlay",

"Whether to use decider overlay",

PropertyType.BOOLEAN,

false);

public static final EarlybirdProperty<String> DECIDER\_OVERLAY\_CONFIG = new EarlybirdProperty<>(

"decider\_overlay\_config",

"Path to decider overlay config",

PropertyType.STRING,

false);

public static final EarlybirdProperty<Integer> MAX\_CONCURRENT\_SEGMENT\_INDEXERS =

new EarlybirdProperty<>(

"max\_concurrent\_segment\_indexers",

"Maximum number of segments indexed concurrently",

PropertyType.INT,

false);

public static final EarlybirdProperty<Boolean> TF\_MODELS\_ENABLED =

new EarlybirdProperty<>(

"tf\_models\_enabled",

"Whether tensorflow models should be loaded",

PropertyType.BOOLEAN,

false);

public static final EarlybirdProperty<String> TF\_MODELS\_CONFIG\_PATH =

new EarlybirdProperty<>(

"tf\_models\_config\_path",

"The configuration path of the yaml file containing the list of tensorflow models to load.",

PropertyType.STRING,

false);

public static final EarlybirdProperty<Integer> TF\_INTER\_OP\_THREADS =

new EarlybirdProperty<>(

"tf\_inter\_op\_threads",

"How many tensorflow inter op threads to use. See TF documentation for more information.",

PropertyType.INT,

false);

public static final EarlybirdProperty<Integer> TF\_INTRA\_OP\_THREADS =

new EarlybirdProperty<>(

"tf\_intra\_op\_threads",

"How many tensorflow intra op threads to use. See TF documentation for more information.",

PropertyType.INT,

false);

public static final EarlybirdProperty<Integer> MAX\_ALLOWED\_REPLICAS\_NOT\_IN\_SERVER\_SET =

new EarlybirdProperty<>(

"max\_allowed\_replicas\_not\_in\_server\_set",

"How many replicas are allowed to be missing from the Earlybird server set.",

PropertyType.INT,

false);

public static final EarlybirdProperty<Boolean> CHECK\_NUM\_REPLICAS\_IN\_SERVER\_SET =

new EarlybirdProperty<>(

"check\_num\_replicas\_in\_server\_set",

"Whether CoordinatedEarlybirdActions should check the number of alive replicas",

PropertyType.BOOLEAN,

false);

public static final EarlybirdProperty<Integer> MAX\_QUEUE\_SIZE =

new EarlybirdProperty<>(

"max\_queue\_size",

"Maximum size of searcher worker executor queue. If <= 0 queue is unbounded.",

PropertyType.INT,

false);

public static final EarlybirdProperty<String> KAFKA\_ENV =

new EarlybirdProperty<>(

"kafka\_env",

"The environment to use for kafka topics.",

PropertyType.STRING,

false);

public static final EarlybirdProperty<String> KAFKA\_PATH =

new EarlybirdProperty<>(

"kafka\_path",

"Wily path to the Search kafka cluster.",

PropertyType.STRING,

false);

public static final EarlybirdProperty<String> TWEET\_EVENTS\_KAFKA\_PATH =

new EarlybirdProperty<>(

"tweet\_events\_kafka\_path",

"Wily path to the tweet-events kafka cluster.",

PropertyType.STRING,

false);

public static final EarlybirdProperty<String> USER\_UPDATES\_KAFKA\_TOPIC =

new EarlybirdProperty<>(

"user\_updates\_topic",

"Name of the Kafka topic that contain user updates.",

PropertyType.STRING,

false);

public static final EarlybirdProperty<String> USER\_SCRUB\_GEO\_KAFKA\_TOPIC =

new EarlybirdProperty<>(

"user\_scrub\_geo\_topic",

"Name of the Kafka topic that contain UserScrubGeoEvents.",

PropertyType.STRING,

false);

public static final EarlybirdProperty<String> EARLYBIRD\_SCRUB\_GEN =

new EarlybirdProperty<>(

"earlybird\_scrub\_gen",

"SCRUB\_GEN TO DEPLOY",

PropertyType.STRING,

false);

public static final EarlybirdProperty<Boolean> CONSUME\_GEO\_SCRUB\_EVENTS =

new EarlybirdProperty<>(

"consume\_geo\_scrub\_events",

"Whether to consume user scrub geo events or not",

PropertyType.BOOLEAN,

false);

private static final List<EarlybirdProperty<?>> ALL\_PROPERTIES =

Arrays.stream(EarlybirdProperty.class.getDeclaredFields())

.filter(field ->

(field.getModifiers() & Modifier.STATIC) > 0

&& field.getType() == EarlybirdProperty.class)

.map(field -> {

try {

return (EarlybirdProperty<?>) field.get(EarlybirdProperty.class);

} catch (Exception e) {

throw new RuntimeException(e);

}

})

.collect(Collectors.collectingAndThen(Collectors.toList(), ImmutableList::copyOf));

public static ServiceIdentifier getServiceIdentifier() {

return new ServiceIdentifier(

ROLE.get(),

EARLYBIRD\_NAME.get(),

ENV.get(),

ZONE.get());

}

private final String name;

private final String help;

private final PropertyType<T> type;

private final boolean requiredOnAurora;

private final boolean requiredOnDedicated;

private EarlybirdProperty(String name, String help, PropertyType<T> type,

boolean requiredOnAurora) {

this(name, help, type, requiredOnAurora, false);

}

private EarlybirdProperty(String name, String help, PropertyType<T> type,

boolean requiredOnAurora, boolean requiredOnDedicated) {

this.name = name;

this.help = help;

this.type = type;

this.requiredOnAurora = requiredOnAurora;

this.requiredOnDedicated = requiredOnDedicated;

}

public String name() {

return name;

}

public boolean isRequiredOnAurora() {

return requiredOnAurora;

}

public boolean isRequiredOnDedicated() {

return requiredOnDedicated;

}

public Flag<T> createFlag(Flags flags) {

return flags.createMandatory(name, help, null, type.flaggable);

}

public T get() {

return type.getter.apply(name);

}

public T get(T devaultValue) {

return type.getterWithDefault.apply(name, devaultValue);

}

public static EarlybirdProperty[] values() {

return ALL\_PROPERTIES.toArray(new EarlybirdProperty[0]);

}

}