package com.twitter.search.earlybird;

import java.io.File;

import java.io.IOException;

import java.net.InetAddress;

import java.net.UnknownHostException;

import java.util.Arrays;

import java.util.Map;

import java.util.function.Predicate;

import java.util.stream.Collectors;

import com.google.common.annotations.VisibleForTesting;

import com.google.common.base.Preconditions;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import com.twitter.app.Flag;

import com.twitter.app.Flaggable;

import com.twitter.finagle.Http;

import com.twitter.finagle.http.HttpMuxer;

import com.twitter.search.common.aurora.AuroraInstanceKey;

import com.twitter.search.common.config.Config;

import com.twitter.search.common.config.LoggerConfiguration;

import com.twitter.search.common.constants.SearchThriftWebFormsAccess;

import com.twitter.search.common.metrics.BuildInfoStats;

import com.twitter.search.common.util.Kerberos;

import com.twitter.search.common.util.PlatformStatsExporter;

import com.twitter.search.earlybird.admin.EarlybirdAdminManager;

import com.twitter.search.earlybird.admin.EarlybirdHealthHandler;

import com.twitter.search.earlybird.common.config.EarlybirdConfig;

import com.twitter.search.earlybird.common.config.EarlybirdProperty;

import com.twitter.search.earlybird.exception.EarlybirdStartupException;

import com.twitter.search.earlybird.exception.UncaughtExceptionHandler;

import com.twitter.search.earlybird.factory.EarlybirdServerFactory;

import com.twitter.search.earlybird.factory.EarlybirdWireModule;

import com.twitter.search.earlybird.thrift.EarlybirdService;

import com.twitter.search.earlybird.util.EarlybirdDecider;

import com.twitter.server.handler.DeciderHandler$;

import com.twitter.server.AbstractTwitterServer;

import com.twitter.thriftwebforms.DisplaySettingsConfig;

import com.twitter.thriftwebforms.MethodOptionsAccessConfig;

import com.twitter.thriftwebforms.ThriftClientSettingsConfig;

import com.twitter.thriftwebforms.ThriftMethodSettingsConfig;

import com.twitter.thriftwebforms.ThriftServiceSettings;

import com.twitter.thriftwebforms.ThriftWebFormsSettings;

import com.twitter.thriftwebforms.TwitterServerThriftWebForms;

import com.twitter.util.Await;

import com.twitter.util.TimeoutException;

public class Earlybird extends AbstractTwitterServer {

private static final Logger LOG = LoggerFactory.getLogger(Earlybird.class);

// Flags defined here need to be processed before setting override values to EarlybirdConfig.

private final Flag<File> configFile = flag().create(

"config\_file",

new File("earlybird-search.yml"),

"specify config file",

Flaggable.ofFile()

);

private final Flag<String> logDir = flag().create(

"earlybird\_log\_dir",

"",

"override log dir from config file",

Flaggable.ofString()

);

private final Map<String, Flag<?>> flagMap = Arrays.stream(EarlybirdProperty.values())

.collect(Collectors.toMap(

property -> property.name(),

property -> property.createFlag(flag())));

private final UncaughtExceptionHandler uncaughtExceptionHandler =

new UncaughtExceptionHandler();

private EarlybirdServer earlybirdServer;

private EarlybirdAdminManager earlybirdAdminManager;

public Earlybird() {

// Default health handler is added inside Lifecycle trait. To override that we need to set it

// in the constructor since HttpAdminServer is started before Earlybird.preMain() is called.

HttpMuxer.addHandler("/health", new EarlybirdHealthHandler());

}

/\*\*

\* Needs to be called from preMain and not from onInit() as flags / args parsing happens after

\* onInit() is called.

\*/

@VisibleForTesting

void configureFromFlagsAndSetupLogging() {

// Makes sure the EarlybirdStats is injected with a variable repository.

EarlybirdConfig.init(configFile.getWithDefault().get().getName());

if (logDir.isDefined()) {

EarlybirdConfig.overrideLogDir(logDir.get().get());

}

new LoggerConfiguration(EarlybirdConfig.getLogPropertiesFile(),

EarlybirdConfig.getLogDir()).configure();

String instanceKey = System.getProperty("aurora.instanceKey");

if (instanceKey != null) {

EarlybirdConfig.setAuroraInstanceKey(AuroraInstanceKey.fromInstanceKey(instanceKey));

LOG.info("Earlybird is running on Aurora");

checkRequiredProperties(EarlybirdProperty::isRequiredOnAurora, "Aurora");

} else {

LOG.info("Earlybird is running on dedicated hardware");

checkRequiredProperties(EarlybirdProperty::isRequiredOnDedicated, "dedicated hardware");

}

LOG.info("Config environment: {}", Config.getEnvironment());

if (adminPort().isDefined() && adminPort().get().isDefined()) {

int adminPort = adminPort().get().get().getPort();

LOG.info("Admin port is {}", adminPort);

EarlybirdConfig.setAdminPort(adminPort);

}

EarlybirdConfig.setOverrideValues(

flagMap.values().stream()

.filter(Flag::isDefined)

.collect(Collectors.toMap(Flag::name, flag -> flag.get().get())));

}

private void checkRequiredProperties(

Predicate<EarlybirdProperty> propertyPredicate, String location) {

Arrays.stream(EarlybirdProperty.values())

.filter(propertyPredicate)

.map(property -> flagMap.get(property.name()))

.forEach(flag ->

Preconditions.checkState(flag.isDefined(),

"-%s is required on %s", flag.name(), location));

}

private void logEarlybirdInfo() {

try {

LOG.info("Hostname: {}", InetAddress.getLocalHost().getHostName());

} catch (UnknownHostException e) {

LOG.info("Unable to be get local host: {}", e.getMessage());

}

LOG.info("Earlybird info [Name: {}, Zone: {}, Env: {}]",

EarlybirdProperty.EARLYBIRD\_NAME.get(),

EarlybirdProperty.ZONE.get(),

EarlybirdProperty.ENV.get());

LOG.info("Earlybird scrubgen from Aurora: {}]",

EarlybirdProperty.EARLYBIRD\_SCRUB\_GEN.get());

LOG.info("Find final partition config by searching the log for \"Partition config info\"");

}

private EarlybirdServer makeEarlybirdServer() {

EarlybirdWireModule earlybirdWireModule = new EarlybirdWireModule();

EarlybirdServerFactory earlybirdFactory = new EarlybirdServerFactory();

try {

return earlybirdFactory.makeEarlybirdServer(earlybirdWireModule);

} catch (IOException e) {

LOG.error("Exception while constructing EarlybirdServer.", e);

throw new RuntimeException(e);

}

}

private void setupThriftWebForms() {

TwitterServerThriftWebForms.addAdminRoutes(this, TwitterServerThriftWebForms.apply(

ThriftWebFormsSettings.apply(

DisplaySettingsConfig.DEFAULT,

ThriftServiceSettings.apply(

EarlybirdService.ServiceIface.class.getSimpleName(),

EarlybirdConfig.getThriftPort()),

ThriftClientSettingsConfig.makeCompactRequired(

EarlybirdProperty.getServiceIdentifier()),

ThriftMethodSettingsConfig.access(

MethodOptionsAccessConfig.byLdapGroup(

SearchThriftWebFormsAccess.READ\_LDAP\_GROUP))),

scala.reflect.ClassTag$.MODULE$.apply(EarlybirdService.ServiceIface.class)));

}

private void setupDeciderWebForms() {

addAdminRoute(

DeciderHandler$.MODULE$.route(

"earlybird",

EarlybirdDecider.getMutableDecisionMaker(),

EarlybirdDecider.getDecider()));

}

@Override

public Http.Server configureAdminHttpServer(Http.Server server) {

return server.withMonitor(uncaughtExceptionHandler);

}

@Override

public void preMain() {

configureFromFlagsAndSetupLogging();

logEarlybirdInfo();

LOG.info("Starting preMain()");

BuildInfoStats.export();

PlatformStatsExporter.exportPlatformStats();

// Use our own exception handler to monitor all unhandled exceptions.

Thread.setDefaultUncaughtExceptionHandler((thread, e) -> {

LOG.error("Invoked default uncaught exception handler.");

uncaughtExceptionHandler.handle(e);

});

LOG.info("Registered unhandled exception monitor.");

Kerberos.kinit(

EarlybirdConfig.getString("kerberos\_user", ""),

EarlybirdConfig.getString("kerberos\_keytab\_path", "")

);

LOG.info("Creating earlybird server.");

earlybirdServer = makeEarlybirdServer();

uncaughtExceptionHandler.setShutdownHook(() -> {

earlybirdServer.shutdown();

this.close();

});

earlybirdAdminManager = EarlybirdAdminManager.create(earlybirdServer);

earlybirdAdminManager.start();

LOG.info("Started admin interface.");

setupThriftWebForms();

setupDeciderWebForms();

LOG.info("Opened thrift serving form.");

LOG.info("preMain() complete.");

}

@Override

public void main() throws InterruptedException, TimeoutException, EarlybirdStartupException {

innerMain();

}

/\*\*

\* Setting up an innerMain() so that tests can mock out the contents of main without interfering

\* with reflection being done in App.scala looking for a method named "main".

\*/

@VisibleForTesting

void innerMain() throws TimeoutException, InterruptedException, EarlybirdStartupException {

LOG.info("Starting main().");

// If this method throws, TwitterServer will catch the exception and call close, so we don't

// catch it here.

try {

earlybirdServer.start();

} catch (Throwable throwable) {

LOG.error("Exception while starting:", throwable);

throw throwable;

}

Await.ready(adminHttpServer());

LOG.info("main() complete.");

}

@Override

public void onExit() {

LOG.info("Starting onExit()");

earlybirdServer.shutdown();

try {

earlybirdAdminManager.doShutdown();

} catch (InterruptedException e) {

LOG.warn("earlybirdAdminManager shutdown was interrupted with " + e);

}

LOG.info("onExit() complete.");

}

}