package com.twitter.search.earlybird\_root.quota;

import java.io.IOException;

import java.io.InputStream;

import java.nio.charset.StandardCharsets;

import java.util.Iterator;

import java.util.Map;

import java.util.Optional;

import java.util.concurrent.ScheduledExecutorService;

import java.util.concurrent.atomic.AtomicReference;

import com.google.common.annotations.VisibleForTesting;

import com.google.common.collect.ImmutableMap;

import com.google.common.collect.Maps;

import org.apache.commons.io.IOUtils;

import org.json.JSONException;

import org.json.JSONObject;

import com.twitter.common.util.Clock;

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

import com.twitter.search.common.util.io.periodic.PeriodicFileLoader;

import com.twitter.search.common.util.json.JSONParsingUtil;

/\*\*

\* Periodically loads a json serialized map that contains the quota information indexed by

\* client id.

\*

\* Each json object from the map is required to have an int property that represents a client's quota.

\* The key for the quota property is passed to this class.

\*

\* Optionally it can have a <b>should\_enforce</b> property of type boolean

\*

\* If this two properties are not present an exception will be thrown.

\*/

public class ConfigBasedQuotaConfig extends PeriodicFileLoader {

private static final String UNSET\_EMAIL = "unset";

private static final String PER\_CLIENT\_QUOTA\_GAUGE\_NAME\_PATTERN =

"config\_based\_quota\_for\_client\_id\_%s";

private static final String PER\_EMAIL\_QUOTA\_GAUGE\_NAME\_PATTERN =

"config\_based\_quota\_for\_email\_%s";

@VisibleForTesting

static final SearchLongGauge TOTAL\_QUOTA =

SearchLongGauge.export("total\_config\_based\_quota");

@VisibleForTesting

static final SearchLongGauge ENTRIES\_COUNT =

SearchLongGauge.export("config\_repo\_quota\_config\_entries\_count");

private final AtomicReference<ImmutableMap<String, QuotaInfo>> clientQuotas =

new AtomicReference<>();

private String clientQuotaKey;

private boolean requireQuotaConfigForClients;

/\*\*

\* Creates the object that manages loads the config from: quotaConfigPath. It periodically

\* reloads the config file using the given executor service.

\*

\* @param quotaConfigPath Path to configuration file.

\* @param executorService ScheduledExecutorService to be used for periodically reloading the file.

\* @param clientQuotaKey The key that will be used to extract client quotas.

\* @param requireQuotaConfigForClients Determines whether a client can be skipped

\* if the associated object is missing the quota key

\* (ie a client that is a SuperRoot client but the current service is Archive)

\*/

public static ConfigBasedQuotaConfig newConfigBasedQuotaConfig(

String quotaConfigPath,

String clientQuotaKey,

boolean requireQuotaConfigForClients,

ScheduledExecutorService executorService,

Clock clock

) throws Exception {

ConfigBasedQuotaConfig configLoader = new ConfigBasedQuotaConfig(

quotaConfigPath,

clientQuotaKey,

requireQuotaConfigForClients,

executorService,

clock

);

configLoader.init();

return configLoader;

}

public ConfigBasedQuotaConfig(

String quotaConfigPath,

String clientQuotaKey,

boolean requireQuotaConfigForClients,

ScheduledExecutorService executorService,

Clock clock

) throws Exception {

super("quotaConfig", quotaConfigPath, executorService, clock);

this.clientQuotaKey = clientQuotaKey;

this.requireQuotaConfigForClients = requireQuotaConfigForClients;

}

/\*\*

\* Returns the quota information for a specific client id.

\*/

public Optional<QuotaInfo> getQuotaForClient(String clientId) {

return Optional.ofNullable(clientQuotas.get().get(clientId));

}

/\*\*

\* Load the json format and store it in a map.

\*/

@Override

protected void accept(InputStream fileStream) throws JSONException, IOException {

String fileContents = IOUtils.toString(fileStream, StandardCharsets.UTF\_8);

JSONObject quotaConfig = new JSONObject(JSONParsingUtil.stripComments(fileContents));

Map<String, Integer> perEmailQuotas = Maps.newHashMap();

ImmutableMap.Builder<String, QuotaInfo> quotasBuilder = new ImmutableMap.Builder<>();

Iterator<String> clientIds = quotaConfig.keys();

long totalQuota = 0;

while (clientIds.hasNext()) {

String clientId = clientIds.next();

JSONObject clientQuota = quotaConfig.getJSONObject(clientId);

// Skip clients that don't send requests to this service.

// (ie some SuperRoot clients are not Archive clients)

if (!requireQuotaConfigForClients && !clientQuota.has(clientQuotaKey)) {

continue;

}

int quotaValue = clientQuota.getInt(clientQuotaKey);

boolean shouldEnforce = clientQuota.optBoolean("should\_enforce", false);

String tierValue = clientQuota.optString("tier", QuotaInfo.DEFAULT\_TIER\_VALUE);

boolean archiveAccess = clientQuota.optBoolean("archive\_access",

QuotaInfo.DEFAULT\_ARCHIVE\_ACCESS\_VALUE);

String email = clientQuota.optString("email", UNSET\_EMAIL);

quotasBuilder.put(

clientId,

new QuotaInfo(clientId, email, quotaValue, shouldEnforce, tierValue, archiveAccess));

SearchLongGauge perClientQuota = SearchLongGauge.export(

String.format(PER\_CLIENT\_QUOTA\_GAUGE\_NAME\_PATTERN, clientId));

perClientQuota.set(quotaValue);

totalQuota += quotaValue;

Integer emailQuota = perEmailQuotas.get(email);

if (emailQuota == null) {

emailQuota = 0;

}

perEmailQuotas.put(email, emailQuota + quotaValue);

}

clientQuotas.set(quotasBuilder.build());

TOTAL\_QUOTA.set(totalQuota);

ENTRIES\_COUNT.set(clientQuotas.get().size());

for (String email : perEmailQuotas.keySet()) {

SearchLongGauge.export(String.format(PER\_EMAIL\_QUOTA\_GAUGE\_NAME\_PATTERN, email)).set(

perEmailQuotas.get(email));

}

}

}