Item 227
git_comments:

git_commits:

1. **summary:** KAFKA-3417: Wrap metric reporter calls in try/catch blocks (#3635)
   message: KAFKA-3417: Wrap metric reporter calls in try/catch blocks (#3635) Prevent exception thrown by metric reporters to impact request processing and other reporters. Co-authored-by: Mickael Maison <mickael.maison@gmail.com> Co-authored-by: Edoardo Comar <ecomar@uk.ibm.com> Reviewers: Rajini Sivaram <rajinisivaram@googlemail.com>

github_issues:

github_issues_comments:

github_pulls:

1. **title:** KAFKA-3417: Wrap reporter calls in try/catch blocks
   body:

github_pulls_comments:

1. Refer to this link for build results (access rights to CI server needed): https://builds.apache.org/job/kafka-pr-jdk7-scala2.11/6608/ Test PASSed (JDK 7 and Scala 2.11).
2. Refer to this link for build results (access rights to CI server needed): https://builds.apache.org/job/kafka-pr-jdk8-scala2.12/6593/ Test PASSed (JDK 8 and Scala 2.12).
3. Thanks for the feedback. I agree, quoting everything is a bit too invasive ! That was made on the spot WRT the quota issue. Regarding the try/catch, I too think they are worth doing. I'll update the PR to only do that. The changes you suggested in the JIRA sound good. I'm happy to work on the KIP
4. Thanks @mimaison, I have assigned KAFKA-5735 to you.
5. Refer to this link for build results (access rights to CI server needed): https://builds.apache.org/job/kafka-pr-jdk7-scala2.11/6813/ Test PASSed (JDK 7 and Scala 2.11).
6. Refer to this link for build results (access rights to CI server needed): https://builds.apache.org/job/kafka-pr-jdk8-scala2.12/6799/ Test PASSed (JDK 8 and Scala 2.12).
7. @rajinisivaram Do you have any objections against these changes ? Even though we've fixed the client-ids issue, any exceptions thrown by reporters could also cause quotas issues.
8. @mimaison Perhaps we want to propagate the exception for clients?
9. @rajinisivaram Sorry I got dragged into other things and I had forgotten about this PR ! I'm not sure propagating the exception is necessarily what we want. We wouldn't want to prevent a client running because one (badly behaving) reporter is throwing exceptions.
10. Currently if a reporter throws an exception while a request is being processed, the broker won't even send a response back to the client. Also if there are other reporters they won't be called at all. This change prevent exceptions thrown by metric reporters to impact request processing and other reporters.

github_pulls_reviews:

1. This is an externally visible change since client-id appears quoted in metrics and can break users who rely on the existing un-quoted value.

jira_issues:

1. **summary:** Invalid characters in config properties not being validated?
   description: I ran into an error using a {{client.id}} with invalid characters (per the [config validator|https://github.com/apache/kafka/blob/trunk/core/src/main/scala/kafka/common/Config.scala#L25-L35]). I was able to get that exact error using the {{kafka-console-consumer}} script, presumably because I supplied a consumer properties file and it validated prior to hitting the server. However, when I use a client library (sarama for Go in this case), an error in the metrics subsystem is thrown [here|https://github.com/apache/kafka/blob/977ebbe9bafb6c1a6e1be69620f745712118fe80/clients/src/main/java/org/apache/kafka/common/metrics/Metrics.java#L3
   The stacktrace is: {code:title=stack.java} [2016-03-17 17:43:47,342] ERROR [KafkaApi-0] error when handling request Name: FetchRequest; Version: 0; CorrelationId: 2; ClientId: foo:bar; ReplicaId: -1; MaxWait: 250 ms; MinBytes: 1 bytes; RequestInfo: [foo,0] -> PartitionFetchInfo(0,32768) (kafka.server.KafkaApis) org.apache.kafka.common.KafkaException: Error creating mbean attribute for metricName :MetricName [name=throttle-time, group=Fetch, description=Tracking average throttle-time per client, tags={client-id=foo:bar}] at org.apache.kafka.common.metrics.JmxReporter.addAttribute(JmxReporter.java:113) at org.apache.kafka.common.metrics.JmxReporter.metricChange(JmxReporter.java:76) at org.apache.kafka.common.metrics.Metrics.registerMetric(Metrics.java:288) at org.apache.kafka.common.metrics.Sensor.add(Sensor.java:177) at org.apache.kafka.common.metrics.Sensor.add(Sensor.java:162) ... {code} Assuming the cause os related to the invalid characters, when the request header is decoded, the {{clientId}} should be validated prior to being used?

jira_issues_comments:

1. I posted on the sarama Go client repo as well for steps to reproduce the behavior: https://github.com/Shopify/sarama/issues/632
2. **body:** Looking at the source a bit more, the `group.id` appears to be restricted to same character set as `client.id`, but group ids with invalid characters are being permitted. For example: {code} ./kafka-console-consumer --new-consumer --consumer.config consumer.properties --topic foo --bootstrap-server 192.168.99.100:9092 foobar ^CProcessed a total of 1 messages {code} where `consumer.properties` contains: {code} group.id=foo:bar {code} The logs appear to have accepts that `group.id`: {code} [2016-03-18 11:59:15,859] INFO [GroupCoordinator 0]: Preparing to restabilize group foo:bar with old generation 0 (kafka.coordinator.GroupCoordinator) [2016-03-18 11:59:15,875] INFO [GroupCoordinator 0]: Stabilized group foo:bar generation 1 (kafka.coordinator.GroupCoordinator) [2016-03-18 11:59:15,905] INFO [GroupCoordinator 0]: Assignment received from leader for group foo:bar for generation 1 (kafka.coordinator.GroupCoordinator) [2016-03-18 12:02:29,094] INFO [GroupCoordinator 0]: Preparing to restabilize group foo:bar with old generation 1 (kafka.coordinator.GroupCoordinator) [2016-03-18 12:02:29,106] INFO [GroupCoordinator 0]: Group foo:bar generation 1 is dead and removed (kafka.coordinator.GroupCoordinator) {code}
   label: code-design
3. GitHub user mimaison opened a pull request: https://github.com/apache/kafka/pull/3635 KAFKA-3417: Quote client-id and wrap reporter calls in try/catch blocks You can merge this pull request into a Git repository by running: $ git pull https://github.com/mimaison/kafka KAFKA-3417 Alternatively you can review and apply these changes as the patch at: https://github.com/apache/kafka/pull/3635.patch To close this pull request, make a commit to your master/trunk branch with (at least) the following in the commit message: This closes #3635 ---- commit ec36fa8052ea60411c77ed82c4b3d7b83fb1f4d3 Author: Mickael Maison <mickael.maison@gmail.com> Date: 2017-08-07T11:51:49Z KAFKA-3417: Quote client-id and wrap reporter calls in try/catch blocks ----
4. rajinisivaram closed pull request #3635: KAFKA-3417: Wrap reporter calls in try/catch blocks URL: https://github.com/apache/kafka/pull/3635 This is a PR merged from a forked repository. As GitHub hides the original diff on merge, it is displayed below for the sake of provenance: As this is a foreign pull request (from a fork), the diff is supplied below (as it won't show otherwise due to GitHub magic): diff --git a/clients/src/main/java/org/apache/kafka/common/metrics/Metrics.java b/clients/src/main/java/org/apache/kafka/common/metrics/Metrics.java index dee69f5a345..7a8667c98c9 100644 --- a/clients/src/main/java/org/apache/kafka/common/metrics/Metrics.java +++ b/clients/src/main/java/org/apache/kafka/common/metrics/Metrics.java @@ -524,8 +524,13 @@ public void addMetric(MetricName metricName, MetricValueProvider<?> metricValueP public synchronized KafkaMetric removeMetric(MetricName metricName) { KafkaMetric metric = this.metrics.remove(metricName); if (metric != null) { - for (MetricsReporter reporter : reporters) - reporter.metricRemoval(metric); + for (MetricsReporter reporter : reporters) { + try { + reporter.metricRemoval(metric); + } catch (Exception e) { + log.error("Error when removing metric from " +

```
reporter.getClass().getName(), e); + } + } } return metric; } @@ -552,8 +557,13 @@ synchronized void registerMetric(KafkaMetric metric) { if
(this.metrics.containsKey(metricName)) throw new IllegalArgumentException("A metric named '" + metricName + "' already exists, can't register another one.");
this.metrics.put(metricName, metric); - for (MetricsReporter reporter : reporters) - reporter.metricChange(metric); + for (MetricsReporter reporter : reporters) { +
try { + reporter.metricChange(metric); + } catch (Exception e) { + log.error("Error when registering metric on " + reporter.getClass().getName(), e); + } + } } /**
@@ -634,8 +644,13 @@ public void close() { } } - for (MetricsReporter reporter : this.reporters) - reporter.close(); + for (MetricsReporter reporter : reporters) { +
try { + reporter.close(); + } catch (Exception e) { + log.error("Error when closing " + reporter.getClass().getName(), e); + } + } } } } diff --git
a/core/src/test/scala/unit/kafka/server/BaseRequestTest.scala b/core/src/test/scala/unit/kafka/server/BaseRequestTest.scala index f91afd492fe..99355bca37a 100644
--- a/core/src/test/scala/unit/kafka/server/BaseRequestTest.scala +++ b/core/src/test/scala/unit/kafka/server/BaseRequestTest.scala @@ -154,6 +154,18 @@
abstract class BaseRequestTest extends KafkaServerTestHarness { skipResponseHeader(response) } + /** + * Sends a request built by the builder, waits for the
response and parses it + */ + def requestResponse(socket: Socket, clientId: String, correlationId: Int, requestBuilder: AbstractRequest.Builder[_ <:
AbstractRequest]): Struct = { + val apiKey = requestBuilder.apiKey + val request = requestBuilder.build() + val header = new RequestHeader(apiKey,
request.version, clientId, correlationId) + val response = requestAndReceive(socket, request.serialize(header).array) + val responseBuffer =
skipResponseHeader(response) + apiKey.parseResponse(request.version, responseBuffer) + } + /** * Serializes and sends the requestStruct to the given api. * A
ByteBuffer containing the response (without the response header) is returned. diff --git
a/core/src/test/scala/unit/kafka/server/KafkaMetricReporterExceptionHandlingTest.scala
b/core/src/test/scala/unit/kafka/server/KafkaMetricReporterExceptionHandlingTest.scala new file mode 100644 index 00000000000..30f3b234613 --- /dev/null
+++ b/core/src/test/scala/unit/kafka/server/KafkaMetricReporterExceptionHandlingTest.scala @@ -0,0 +1,116 @@ +/** + * Licensed under the Apache License,
Version 2.0 (the "License"); + * you may not use this file except in compliance with the License. + * You may obtain a copy of the License at + * + *
http://www.apache.org/licenses/LICENSE-2.0 + * + * Unless required by applicable law or agreed to in writing, software + * distributed under the License is
distributed on an "AS IS" BASIS, + * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. + * See the License for the
specific language governing permissions and + * limitations under the License. + **/ + +package kafka.server + +import java.net.Socket +import
java.util.Properties + +import kafka.utils.TestUtils +import org.apache.kafka.common.network.ListenerName +import org.apache.kafka.common.requests.
{ListGroupsRequest,ListGroupsResponse} +import org.apache.kafka.common.metrics.MetricsReporter +import org.apache.kafka.common.metrics.KafkaMetric
+import org.apache.kafka.common.security.auth.SecurityProtocol +import org.apache.kafka.common.protocol.Errors + +import org.junit.Assert._ +import
org.junit.{Before, Test} +import org.junit.After +import java.util.concurrent.atomic.AtomicInteger + +/* + * this test checks that a reporter that throws an
exception will not affect other reporters + * and will not affect the broker's message handling + */ +class KafkaMetricReporterExceptionHandlingTest extends
BaseRequestTest { + + override def numBrokers: Int = 1 + + override def propertyOverrides(properties: Properties): Unit = { +
properties.put(KafkaConfig.MetricReporterClassesProp, classOf[KafkaMetricReporterExceptionHandlingTest.BadReporter].getName + "," +
classOf[KafkaMetricReporterExceptionHandlingTest.GoodReporter].getName) + } + + @Before + override def setUp() { + super.setUp() + + // need a quota prop
to register a "throttle-time" metrics after server startup + val quotaProps = new Properties() +
quotaProps.put(DynamicConfig.Client.RequestPercentageOverrideProp, "0.1") + adminZkClient.changeClientIdConfig("<default>", quotaProps) + } + + @After
+ override def tearDown() { + KafkaMetricReporterExceptionHandlingTest.goodReporterRegistered.set(0) +
KafkaMetricReporterExceptionHandlingTest.badReporterRegistered.set(0) + + super.tearDown() + } + + @Test + def testBothReportersAreInvoked() { + val port
= anySocketServer.boundPort(ListenerName.forSecurityProtocol(SecurityProtocol.PLAINTEXT)) + val socket = new Socket("localhost", port) +
socket.setSoTimeout(10000) + + try { + TestUtils.retry(10000) { + val error = new ListGroupsResponse(requestResponse(socket, "clientId", 0, new
ListGroupsRequest.Builder())).error() + assertEquals(Errors.NONE, error) +
assertEquals(KafkaMetricReporterExceptionHandlingTest.goodReporterRegistered.get, KafkaMetricReporterExceptionHandlingTest.badReporterRegistered.get) +
assertTrue(KafkaMetricReporterExceptionHandlingTest.goodReporterRegistered.get > 0) + } + } finally { + socket.close() + } + } +} + +object
KafkaMetricReporterExceptionHandlingTest { + var goodReporterRegistered = new AtomicInteger + var badReporterRegistered = new AtomicInteger + + class
GoodReporter extends MetricsReporter { + + def configure(configs: java.util.Map[String, _]) { + } + + def init(metrics: java.util.List[KafkaMetric]) { + } + + def
metricChange(metric: KafkaMetric) { + if (metric.metricName.group == "Request") { + goodReporterRegistered.incrementAndGet + } + } + + def
metricRemoval(metric: KafkaMetric) { + } + + def close() { + } + } + + class BadReporter extends GoodReporter { + + override def metricChange(metric:
KafkaMetric) { + if (metric.metricName.group == "Request") { + badReporterRegistered.incrementAndGet + throw new
RuntimeException(metric.metricName.toString) + } + } + } +} diff --git a/core/src/test/scala/unit/kafka/server/RequestQuotaTest.scala
b/core/src/test/scala/unit/kafka/server/RequestQuotaTest.scala index ed85415eacc..8a50fcafd1a 100644 ---
a/core/src/test/scala/unit/kafka/server/RequestQuotaTest.scala +++ b/core/src/test/scala/unit/kafka/server/RequestQuotaTest.scala @@ -14,7 +14,6 @@ package
kafka.server -import java.net.Socket import java.nio.ByteBuffer import java.util.{Collections, LinkedHashMap, Properties} import java.util.concurrent.{Executors,
Future, TimeUnit} @@ -331,15 +330,6 @@ class RequestQuotaTest extends BaseRequestTest { } } - private def requestResponse(socket: Socket, clientId: String,
correlationId: Int, requestBuilder: AbstractRequest.Builder[_ <: AbstractRequest]): Struct = { - val apiKey = requestBuilder.apiKey - val request =
requestBuilder.build() - val header = new RequestHeader(apiKey, request.version, clientId, correlationId) - val response = requestAndReceive(socket,
request.serialize(header).array) - val responseBuffer = skipResponseHeader(response) - apiKey.parseResponse(request.version, responseBuffer) - } - case class
Client(clientId: String, apiKey: ApiKeys) { var correlationId: Int = 0 val builder = requestBuilder(apiKey) ----------------------------------------------------------------
```