

Kafka Plugins

Chia-Ping Tsai

Kafka Plugin Configs

key.serializer	client.quota.callback.class	connector.client.config.override.policy	dsl.store.suppliers.class
value.serializer	remote.log.storage.manager.class.name	replication.policy.class	default.client.supplier
interceptor.classes	remote.log.metadata.manager.class.name	topic.filter.class	default.key.serde
partitioner.class	create.topic.policy.class.name	group.filter.class	default.timestamp.extractor
key.deserializer	alter.config.policy.class.name	value-decoder-class	processing.exception.handler
value.deserializer	authorizer.class.name	key-decoder-class	default.value.serde
metric.reporters	group.consumer.assignors		deserialization.exception.handler
	replica.selector.class		default.deserialization.exception.handler
			production.exception.handler
			default.production.exception.handler

What is the environment for Plugins?

your application

- ① Consumer
- ② Producer
- ③ Admin
- ④ Streams

your server

- ① Controller
- ② Broker
- ③ Worker

note the Dependency Hell

No worries, let's just chat
about the policy for plugins

Num.0

All Plugins are Public APIs

The Compatibility Policy

MINOR -> MINOR = Binary Compatibility

no re-compilation, no code change, everything work well

MAJOR -> MAJOR = Data Compatibility

Only data is unbreakable

APIs, RPCs, metrics, and configs are breakable

Which are Public APIs?

```
/*  
 * Plug-able interface for selecting a preferred read replica given the current set of replicas for a partition  
 * and metadata from the client.  
 * Implement {@link org.apache.kafka.common.metrics.Monitorable} to enable the selector to register metrics.  
 * The following tags are automatically added to all metrics registered: <code>config</code> set to  
 * <code>replica.selector.class</code>, and <code>class</code> set to the ReplicaSelector class name.  
 */  
public interface ReplicaSelector extends Configurable Closeable { 13 usages 4 implementations ⓘ David Arthur  
  
    /**  
     * Select the preferred replica a client should use for fetching. If no replica is available, this will return an  
     * empty optional. 1  
     */  
    Optional<ReplicaView> select(TopicPartition topicPartition, 2  
                                ClientMetadata clientMetadata, 3  
                                PartitionView partitionView); 4  
  
    @Override 1 override ⓘ David Arthur  
    default void close() throws IOException {  
        // No-op by default  
    }  
  
    @Override 1 override ⓘ David Arthur  
    default void configure(Map<String, ?> configs) {  
        // No-op by default  
    }  
}
```

Breaking Change with Replacement

KIP-1162: Redesign ClientQuotaCallback#updateClusterMetadata

Public Interfaces

This KIP introduces a new interface `ClientQuotaCallbackHandler` which basically copy all methods in `ClientQuotaCallback` except the one we want to deprecate.

```
boolean updateClusterMetadata(Cluster cluster);
```

 deprecated data

And add a new method to replace the old one, the parameter `ClusterMetadata` in the new method is a new interface, which will be described below.

```
boolean updateClusterMetadata(ClusterMetadata clusterMetadata);
```

 new data as replacement

noted: not all APIs are replaceable

Exception is a part of Signature

Please throw the exception defined by signature

```
* @throws PolicyViolationException if the request parameters do not satisfy this policy.  
*/  
void validate(RequestMetadata requestMetadata) throws PolicyViolationException; 3 implementa
```

noted: Kafka prefers RuntimeException type

What If I throw other Exception?

```
@Override new *
public void validate(RequestMetadata requestMetadata) throws PolicyViolationException {
    if (requestMetadata.topic().equals("chia"))
        throw new PolicyViolationException("chia is not allowed");
    if (requestMetadata.topic().equals("ikea"))
        throw new RuntimeException("Ikea is not allowed in runtime");
}
```

good engineer

bad engineer

Your Log will be flooded with error messages

```
[2025-05-29 12:40:46,049] ERROR Encountered nonFatalFaultHandler fault: createTopics: event failed with RuntimeException (treated as UnknownServerException) at epoch 1 in
java.lang.RuntimeException: Ikea is not allowed in runtime
    at org.apache.kafka.clients.admin.StatefulBrokerConfigSetPolicy.validate(StatefulBrokerConfigSetPolicy.java:79) ~[test/?:?]
    at org.apache.kafka.controller.ReplicationControlManager.maybeCheckCreateTopicPolicy(ReplicationControlManager.java:871) ~[kafka-metadata-4.1.0-SNAPSHOT.jar:?]
    at org.apache.kafka.controller.ReplicationControlManager.createTopic(ReplicationControlManager.java:799) ~[kafka-metadata-4.1.0-SNAPSHOT.jar:?]
    at org.apache.kafka.controller.ReplicationControlManager.createTopics(ReplicationControlManager.java:666) ~[kafka-metadata-4.1.0-SNAPSHOT.jar:?]
    at org.apache.kafka.controller.QuorumController.lambda$createTopics$7(QuorumController.java:1777) ~[kafka-metadata-4.1.0-SNAPSHOT.jar:?]
    at org.apache.kafka.controller.QuorumController$ControllerWriteEvent.run(QuorumController.java:785) ~[kafka-metadata-4.1.0-SNAPSHOT.jar:?]
    at org.apache.kafka.queue.KafkaEventQueue$EventContext.run(KafkaEventQueue.java:133) ~[kafka-server-common-4.1.0-SNAPSHOT.jar:?]
    at org.apache.kafka.queue.KafkaEventQueue$EventHandler.handleEvents(KafkaEventQueue.java:216) ~[kafka-server-common-4.1.0-SNAPSHOT.jar:?]
    at org.apache.kafka.queue.KafkaEventQueue$EventHandler.run(KafkaEventQueue.java:187) ~[kafka-server-common-4.1.0-SNAPSHOT.jar:?]
    at java.base/java.lang.Thread.run(Thread.java:1583) [?:?]
```

Exceptions need to be Compatible

```
*  
~  
* In earlier versions a TimeoutException was thrown instead of this. To keep existing catch-clauses working  
* this class extends TimeoutException.  
*  
*/  
public class BufferExhaustedException extends TimeoutException { 10 usages  👤 Jay Kreps +1  
  
    private static final long serialVersionUID = 1L; no usages  
  
    public BufferExhaustedException(String message) { super(message); }  
  
}
```

It's easy to overlook Exceptions need to be Compatible

the exception thrown by Plugin is NOT handled

KAFKA-17994: Checked exceptions are not handled #17817

 Merged **chia7712** merged 2 commits into `apache:trunk` from `mjsax:kafka-17994-runtime-exception` on Nov 15, 2024

 Conversation **8**  Commits **2**  Checks **6**  Files changed **5**



mjsax commented on Nov 15, 2024

Member ...

While Java distinguished between checked vs unchecked exception, other JVM languages do not. Thus, user code might still throw a checked exception even if no checked exception is declared on the implemented interface.

Should be cherry-picked to 3.9 branch.




1





[Kafka](#) / KAFKA-19350

don't propagate the error caused by CreateTopicPolicy to FatalFaultHandler



 Edit  Add comment  Assign  More  Submit Patch  Resolve Issue

Details

Type:  Improvement
Priority:  Minor
Affects Version/s: None
Component/s: None
Labels: None

Status: **OPEN**
Resolution: Unresolved
Fix Version/s: None

People

Assignee:  Szu-Yung Wang
[Assign to me](#)
Reporter:  Chia-Ping Tsai
Votes: **0**
Watchers: **1** [Stop watching this issue](#)

Description

in 3.9, we catch all exceptions from `CreateTopicPolicy` (<https://github.com/apache/kafka/blob/3.9/core/src/main/scala/kafka/server/ZkAdminManager.scala#L234>). The all exceptions from `AlterConfigPolicy#validate` are caught as well (<https://github.com/apache/kafka/blob/8731c96122fed1ce6106b2d42ad150d13124d0b1/metadata/src/main/java/org/apache/kafka/controller/ConfigurationControlManager.java#L368>)


Dates

Created: 2 hours ago
Updated: 2 hours ago




Num.1


Each Plugin is a Java Interface



Kafka don't love Scala anymore

 Kafka / KAFKA-14524


Modularize `core` monolith

 Edit  Add comment Assign More  Submit Patch Resolve Issue

 Details

Type:  Improvement
Priority:  Major
Affects Version/s: None
Component/s: None
Labels: None

Status: **OPEN**
Resolution: Unresolved
Fix Version/s: None

 Description

The `core` module has grown too large and it's time to split it into multiple modules. The `core` module will be deleted once it's empty (a KIP will be required for this).

Evidence of `core` growing too large is that it takes 1m10s to compile the main code and tests and it takes hours to run all the tests sequentially.

As part of this effort, we should rewrite the Scala code in Java to reduce developer friction, reduce compilation time and simplify deployment (i.e. we can remove the scala version suffix from the module name). Scala may have a number of advantages over Java 8 (minimum version we support now) and Java 11 (minimum version we will support in Kafka 4.0), but a mixture of Scala and Java (as we have now) is more complex than just Java.

Another benefit is that code dependencies will be strictly enforced, which will hopefully help ensure better abstractions.

This pattern was started with the `tools` (but not completed), `metadata` and `raft` modules and we have (when this ticket was filed) a couple more in progress: `group-coordinator` and `storage`.

This is an umbrella ticket and it will link to each ticket related to this goal.

Server-side is still using Scala

```
25 // Add Scala version
26 def defaultScala213Version = '2.13.16'
27 if (hasProperty('scalaVersion')) {
28     if (scalaVersion == '2.13') {
29         versions["scala"] = defaultScala213Version
30     } else {
31         versions["scala"] = scalaVersion
32     }
33 } else {
34     versions["scala"] = defaultScala213Version
35 }
```

your server

- ① Controller (o)
- ② Broker (o)
- ③ Worker (x)

noted not all "servers" have scala jars by default

Mysterious Add-on Interfaces

```
✓ public interface Monitorable {  
  
    /**  
     * Provides a {@link PluginMetrics} instance from the component that instantiates the plugin.  
     * PluginMetrics can be used by the plugin to register and unregister metrics  
     * at any point in their lifecycle prior to their close method being called.  
     * Any metrics registered will be automatically removed when the plugin is closed.  
     */  
    void withPluginMetrics(PluginMetrics metrics);  
  
}
```

```
47 ✓ public interface ClusterResourceListener {  
48     /**  
49     * A callback method that a user can implement to get updates for {@link ClusterResource}.  
50     * @param clusterResource cluster metadata  
51     */  
52     void onUpdate(ClusterResource clusterResource);  
53 }
```

```
24     /**  
25     * Interface for reconfigurable classes that support dynamic configuration.  
26     */  
27 ✓ public interface Reconfigurable extends Configurable {  
28  
29     /**  
30     * Returns the names of configs that may be reconfigured.  
31     */  
32     Set<String> reconfigurableConfigs();  
33  
34     /**  
35     * Validates the provided configuration. The provided map contains  
36     * all configs including any reconfigurable configs that may be different  
37     * from the initial configuration. Reconfiguration will be not performed  
38     * if this method throws any exception.  
39     * @throws ConfigException if the provided configs are not valid. The exception  
40     *     message from ConfigException will be returned to the client in  
41     *     the AlterConfigs response.  
42     */  
43     void validateReconfiguration(Map<String, ?> configs) throws ConfigException;  
44  
45     /**  
46     * Reconfigures this instance with the given key-value pairs. The provided  
47     * map contains all configs including any reconfigurable configs that  
48     * may have changed since the object was initially configured using  
49     * {@link Configurable#configure(Map)}. This method will only be invoked if  
50     * the configs have passed validation using {@link #validateReconfiguration(Map)}.  
51     */  
52     void reconfigure(Map<String, ?> configs);  
53  
54 }
```


Num.2

The common methods of Plugin
`configure` and `close`

Plugin Lifecycle

new instance noted: you must provide default "public" constructor



config: Map[String, Object]

configure noted: you should create resources in this phase



... do something



close noted: this method is always executed even if `configure` fails

「累了，先這樣吧，有任何疑問嘛」

—蔡嘉平