KIP-544: Make metrics exposed via JMX configurable

- Status
- Public Interfaces
  - Example
- Proposed Changes
- Compatibility, Deprecation, and Migration Plan
- Rejected Alternatives

Status

Current state: Accepted (vote thread)

Discussion thread: *here*

JIRA: https://issues.apache.org/jira/browse/KAFKA-9106

Motivation

Kafka exposes a very large number of metrics, all of which are always visible in JMX by default. On large clusters with many partitions, this may result in tens of thousands of mbeans to be registered, which can lead to timeouts with some popular monitoring agents that rely on listing JMX metrics via RMI.

This KIP proposes to make the set of metrics visible in JMX configurable via a set of whitelist/blacklist regular expressions on the mbean names.

Public Interfaces

Two new configurations will be introduce to allow filtering the set of metrics exposed via JMX.

The configurations would apply to brokers, as well as clients, streams, and connect.

```
metrics.jmx.whitelist=<regular expression of mbeans to expose> (defaults to ".*")

metrics.jmx.blacklist=<regular expression of mbeans to hide> (defaults to "", takes precedence over the
whitelist)
```

By default, this would not change the set of metrics exposed via JMX: all metrics would be exposed.

All metrics continue to be collected in the background regardless of those settings, so this only affects the visibility in JMX and not the actual metric values.

Brokers specifics:

- Configurations will apply to JMX metrics exposed via Yammer metrics as well Kafka's internal metrics library.
- Configurations will be marked as dynamic to allow updates at runtime: adding and removing JMX metrics as needed to reflect the new filters.

Clients, streams, and connect:

- Configuration will apply to all the metrics exposed in JMX metrics via Kafka's internal metrics library.

Example

To hide log-level metrics `LogEndOffset`, `LogStartOffset`, and `NumLogSegments`, as well as any partition level metrics, except `UnderMinIsr`, one could exclude those by specifying the following property in their `server.properties` file:

```
metrics.jmx.blacklist=(kafka.log:type=Log,name=(LogEndOffset|LogStartOffset|NumLogSegments),.*\

                          |kafka.cluster:type=Partition,name=(?!UnderMinIsr),.*)
```

Proposed Changes

A new KafkaYammerMetrics class will be added to hold a Kafka-specific Yammer metrics registry, replacing the default global yammer metrics registry, which allows us to override the JMX behavior of Yammer metrics.

Any code currently using `com.yammer.metrics.Metrics.defaultRegistry()` will switch to using `KafkaYammerMetrics.defaultRegistry()`

Both `KafkaYammerMetrics` as well as the default JMX reporter for Kafka's own metrics library will be marked as Reconfigurable and listen to changes for the configurations defined above.

Usage of `com.yammer.metrics.Metrics.defaultRegistry()`will be forbidden via checkstyle import controls to prevent code change from accidentally using the previous registry.

Note: `KafkaYammerMetrics` will remain an internal class, this KIP is not proposing to make this a public interface.

Compatibility, Deprecation, and Migration Plan

The are no compatibility implications for this change as far as public interfaces are concerned. There is no change in metrics exposed by default.

Note: Plugins relying on internal Kafka implementation details to collect Yammer metrics in-process would have to make changes to explicitly depend on this new Kafka internal classes.

Rejected Alternatives

Fork yammer metrics

The same functionality could accomplished by forking yammer metrics. While this would preserve "compatibility" with plugins collecting yammer metrics directly in-process, it seemed unnecessarily cumbersome. Isolating the Kafka yammer metrics also makes it clear to any plugins that they are relying on Kafka server internals.

Upgrade to Dropwizard metrics

While newer versions of Dropwizard metrics support filtering metrics exposed by reporters, recent Dropwizard metrics no longer have a default global metrics registry, which means it would require a similar Kafka-specific metrics registry. The filtering also doesn't support dynamic updating and removal of existing mbeans when they no longer match the filter. This KIP does not prevent us from moving to a new version in the future, and as such, an upgrade should be considered an orthogonal concern. There is a separate KIP addressing this KIP-510: Metrics library upgrade, which also outlines actual backwards compatibility implications, which are beyond the scope of this KIP.