KIP-843: Adding addMetricIfAbsent method to Metrics

Status

Current state: *"Accepted"*

Discussion thread:

JIRA: KAFKA-13846

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

Concurrent thread may try to access the Metrics registry to create the same, instance-level metric, however it's get/create APIs are not well suited for it. A common pattern that users follow today is:

```
metric = metrics.metric(metricName);

if (metric == null) {
  try {
    metrics.addMetric(..)
  } catch (IllegalArgumentException e){
    // another thread may create the metric at the mean time
  }
}
```

Here's an example on how the AK codebase uses the above pattern:

https://github.com/apache/kafka/blob/trunk/connect/runtime/src/main/java/org/apache/kafka/connect/runtime/ConnectMetrics.java#L313-L325

The above process basically consists of 2 steps:

1. Check if a Metric of interest exists or not.
2. If yes, an IllegalArgumentException would be thrown which could be due to race conditions as well. We might just want to catch this exception and swallow it - as long as the metric gets created.
3. Else, create the metric.

The main motivation of this KIP is to expose an API which would make these operations atomic. That ways, users won't need to remember these steps and can just focus on having a metric created.

Public Interfaces

A new public facing method **getMetricOrElseCreate** would be exposed. This would create a non-existing metric or create a return the Metric if it already exists. This way, users don't need to add extra logic to take respective actions in case of presence/absence of metrics. Note that this method takes care of synchronisation while updating/accessing metrics by concurrent threads.

Metrics.java

```
/**
     * Create or get an existing metric to monitor an object that implements MetricValueProvider.
     * This metric won't be associated with any sensor. This is a way to expose existing values as metrics.
     * This method takes care of synchronisation while updating/accessing metrics by concurrent threads.
     *
     * @param metricName The name of the metric
     * @param metricValueProvider The metric value provider associated with this metric
     * @return Existing KafkaMetric if already registered or else a newly created one
     */
    public KafkaMetric addMetricIfAbsent(MetricName metricName, MetricConfig config, MetricValueProvider<?>
metricValueProvider) {
        KafkaMetric metric = new KafkaMetric(new Object(),
                Objects.requireNonNull(metricName),
                Objects.requireNonNull(metricValueProvider),
                config == null ? this.config : config,
                time);

        KafkaMetric existingMetric = registerMetric(metric);
        return existingMetric == null ? metric : existingMetric;
    }
```

Proposed Changes

- Adding a new function above to the Metrics API. As part of this change, the `registerMetric` method's return type would be changed from `void` to `KafkaMetric` . It would return a KafkaMetric object if the requested metric already exists or return null if not after creating/registering the metric. For backward compatibility reasons, any place currently which relied on `IllegalArgumentException` would now instead check the output of `registerMetric` and throw an `IllegalArguementException` when the returned value of `registerMetric` is non-null. This change would happen in => `Metrics.addMetric` , 2 `Sensor.add` methods. On the other hand, getMetricOrElseCreate method would simply return the object returned by `registerMetric if not null`.

```
/**
     * Register a metric if not present or return an already existing metric otherwise.
     * When a metric is newly registered, this method returns null
     *
     * @param metric The KafkaMetric to register
     * @return KafkaMetric if the metric already exists, null otherwise
     */
synchronized KafkaMetric registerMetric(KafkaMetric metric) {
        MetricName metricName = metric.metricName();
        KafkaMetric existingMetric = this.metrics.putIfAbsent(metricName, metric);
        if (existingMetric != null) {
            return existingMetric;
        }
        // newly added metric
        for (MetricsReporter reporter : reporters) {
            try {
                reporter.metricChange(metric);
            } catch (Exception e) {
                log.error("Error when registering metric on " + reporter.getClass().getName(), e);
            }
        }
        log.trace("Registered metric named {}", metricName);
        return null;
    }
```

Compatibility, Deprecation, and Migration Plan

The changes are backward compatible and needs no deprecation/migration.

Rejected Alternatives

N/A