DMetric

Datadog 抽象度量，是一个抽象类。

含有的属性有：

```
private final String metric; // Metric name

//MetricType 有两种 gauge 和 counter
private final MetricType type;

//
private final String host;
private final List<String> tags;
```

DCounter

Flink 和 Datadog 之间 Counter 的映射

DGauge

Flink 和 Datadog 之间 Gauge 的映射

DMeter

Flink 和 Datadog 之间 Meter 的映射

只考虑 meter 的速率，因为 Datadog HTTP API 对 meter 的速度支持有限制的。

DSeries

Flink 和 Datadog 之间的 Json 序列化

DatadogHttpClient

和 Datadog 交互的 Httpclient，关键在于里面的 send 方法，会将请求里面的 Series 获取到后转成 json 参数，然后通过根据参数 url、请求消息体来建立一个请求，最后通过 OkHttpClient 来将完成请求的发送。

```
client.newCall(r).enqueue(EmptyCallback.getEmptyCallback());
```

DatadogHttpReporter

Datadog metrics 的 reporter，该类含有的属性有：

```
private static final String HOST_VARIABLE = "<host>";

//Flink 的 Gauge 和 Meter 的值在 Datadog 中都被作为 Gauge
private final Map<Gauge, DGauge> gauges = new ConcurrentHashMap<>
();
private final Map<Counter, DCounter> counters = new
ConcurrentHashMap<>();
private final Map<Meter, DMeter> meters = new ConcurrentHashMap<>
();

private DatadogHttpClient client;
private List<String> configTags;

public static final String API_KEY = "apikey";
public static final String PROXY_HOST = "proxyHost";
public static final String PROXY_PORT = "proxyPort";
public static final String TAGS = "tags";
```

在 open 方法中会对上面的部分属性进行初始化：

```
//从参数中获取 apikey、proxyHost、proxyPort
String apiKey = config.getString(API_KEY, null);
String proxyHost = config.getString(PROXY_HOST, null);
Integer proxyPort = config.getInteger(PROXY_PORT, 8080);

client = new DatadogHttpClient(apiKey, proxyHost, proxyPort);

configTags = getTagsFromConfig(config.getString(TAGS, ""));
```

该类同样也对 notifyOfAddedMetric 和 notifyOfRemovedMetric 方法进行重写了。

在 report 方法中做 DataDog 数据的上报：

```java
public void report() {
    DatadogHttpRequest request = new DatadogHttpRequest();

    List<Gauge> gaugesToRemove = new ArrayList<>();
    for (Map.Entry<Gauge, DGauge> entry : gauges.entrySet()) {
        DGauge g = entry.getValue();
        try {
            // Will throw exception if the Gauge is not of Number type
            // Flink uses Gauge to store many types other than Number
            g.getMetricValue();
            request.addGauge(g);
        } catch (ClassCastException e) {
            LOGGER.info("The metric {} will not be reported because only number types are supported by this reporter.", g.getMetric());
            gaugesToRemove.add(entry.getKey());
        } catch (Exception e) {
            if (LOGGER.isDebugEnabled()) {
                LOGGER.debug("The metric {} will not be reported because it threw an exception.", g.getMetric(), e);
            } else {
                LOGGER.info("The metric {} will not be reported because it threw an exception.", g.getMetric());
            }
            gaugesToRemove.add(entry.getKey());
        }
    }
    gaugesToRemove.forEach(gauges::remove);

    for (DCounter c : counters.values()) {
        request.addCounter(c);
    }

    for (DMeter m : meters.values()) {
        request.addMeter(m);
    }

    try {
        client.send(request);
    } catch (SocketTimeoutException e) {
        LOGGER.warn("Failed reporting metrics to Datadog because of socket timeout.", e.getMessage());
    } catch (Exception e) {
        LOGGER.warn("Failed reporting metrics to Datadog.", e);
    }
    }
}
```

最终都是通过 DatadogHttpRequest 对象将数据（Counter、Meter、Gauge）放在一个 DSeries （其实内部就是一个 DMetric List）中，最后通过 DatadogHttpClient 的 send 方法将所有的请求数据发送出去。

注意

要使用此 reporter，你必须复制 /opt/flink-metrics-datadog-1.9.jar 到 Flink 的 /lib 文件夹下。

注意 Flink 指标，如任何变量 <host>，<job_name>，<tm_id>，<subtask_index>，<task_name>，和 <operator_name> 将被发送到 Datadog 的标签，标签看起来像 host:localhost 和 job_name:myjobname。

参数：

- apikey： Datadog API Keys
- tags： （可选）发送到 Datadog 时将应用于度量标准的全局标记，标签应仅以逗号分隔

配置示例：

```
metrics.reporter.dghttp.class:
org.apache.flink.metrics.datadog.DatadogHttpReporter
metrics.reporter.dghttp.apikey: xxx
metrics.reporter.dghttp.tags: myflinkapp,prod
```