RFC 3550 - 2020-11-01 - Support metrics with the new 'remap' transform

It would be useful to add the ability for the Remap language to work with metric data as well as with log data.

Scope

The aim of the Remap language is to remain simple and yet provide the power to allow the user to replace most of their tranform configuration with a simple Remap script. Any complexity beyond a certain level is possible, but requires the `Lua` or `Wasm` transforms.

Given this, the scope of this RFC is not to enhance the Remap language so it can work comprehensively with metric data. The main aim will be to allow the user to make simple changes to the metadata surrounding the metric.

Motivation

Currently there are some simple transforms that allow manipulating the metadata of metric data - `add_tags`, `remove_tags`, `rename_tags`. These transforms are useful, but limited. They cannot take into account any additional data around the metric to determine how to process the tags.

Doc-level Proposal

The Remap language can work with the metadata around a metric.

The following functions are provided:

`get_name() -> string`

Returns the name of the metric.

`set_name(string)`

Allows you to change the name of the metric.

`get_namespace() -> string`

Returns the namepace of the metric.

`set_namespace(string)`

Allows you to change the namespace of the metric.

`get_timestamp() -> timestamp`

Returns the timestamp of the metric.

`set_timestamp(timestamp)`

Allows you to change the timestamp of the metric.

`get_kind() -> string`

Returns the kind of the metric, will either be "Incremental" or "Absolute".

`set_kind(string)`

Allows you to set the kind of the metric. Any string other than "Incremental" or "Absolute" will result in an error.

`get_tag(string) -> string`

Returns the value of the given tag.

`add_tag(string, string)`

Allows you to set the value of a given tag.

`rename_tag(string, string)`

Rename the tag with the given key to the new key.

`remove_tag(string)`

Removes the tag with the given key from the metric.

Note that accessing path values (identifiers starting with a `.`) are not available for metric events.

Variables (identifiers starting with a `$`) are available.

Rationale

Enabling the Remap language to work with metric events will allow users to simplify their configurations if they already use the `*_tag` transforms provided. It will also allow them remove the use of the `Lua` transform if they have been falling back to that for anything but the most complex manipulation.

Prior Art

Drawbacks

In order to maintain these functions there will be a slight maintenance burden.

By exposing the metric data to Remap, it may place some restrictions should we want to change the internal model in the future. The more we decide to expose

to the language, the more it may restrict us going forwards if we need to ensure backwards compatibility for users configurations.

Alternatives

Use paths

We could allow the metric data to be manipulated through paths in a similar way to how the language works with log events.

The following fields would be available:

- `.name` - The name of the metric.
- `.namespace` - The namespace for the metric.
- `.timestamp` - Timestamp of the metric.
- `.kind` - The kind of the metric - *Incremental* or *Absolute*
- `.tags` - A map containing the tags set for the metric.

However, there are strict limitations to how this data can be represented. Setting `name` or `namespace` to anything other than a string should be an error. `tags` has to be a map of string keys to string values. No nesting should be allowed. The inflexibility around this data means that the attempt to make it "feel" the same as log events would not be effective.

Manipulate the metric data

We could provide functions to allow manipulating the metric data itself.

This is complicated because there are a number of different metric types, so each provided function would only work if the metric was the correct type. For example, functions `get_gauge_value` and `set_gauge_value` would error if the metric was an `AggregatedHistogram`. Similar problems would occur if the data was represented through paths rather than functions.

This functionality is provided by the `Lua` transform, so there is still a way forward for the user if they need this functionality.

Note that going with the original proposal does not preclude this, so this could still be implemented later as a next stage.

Allow conversion between metric and log events

Currently the RFC just lays out methods to manipulate either log or metric events. It could also be possible to add functions to convert a log event into a metric event and viceversa.

For example:

```
. = to_log()
```

could take the current metric event and write the fields into the root path to create a log event out of the metric.

To create metric events one function per metric type could be provided such as: `make_counter(value)`, `make_gauge(value)`, `make_set([value])`

The other functions provided in this proposal can then be used to fill out the rest of the event.

Note this could still be implemented at a later stage.

Add functions to enable manipulating metrics for the most likely usecases

If we analyse the most likely scenarios where users would want to manipulate the metrics, we could provide functions that cover these scenarios.

For example, one likely scenario would be to sample a stream of gauge values and convert this into a sampled distribution. A function could be provided that enables this: `sample_gauge_values([buckets], sample_rate)`.

The complexities involved with providing such functions are that it means there is no longer a strict one-to-one correspondance between input and output events. Many input events would be summarised to a single output event.

We would need to think about what would need to be done should `sample_gauge_values` be called twice in the script.

This can be handled by the `Lua` transform, but the script to do this can get quite involved. If it could be brought into the `Remap` language and simplified so than 80% of the likely usecases are covered, it could be a big win for simplifying users configurations.

Note that this could still be implemented later as a next stage.

Outstanding Questions

- Should we add an extra option to the Remap transform of event_type to force the user to specify if their script is for log or metric events and error at load time if they use functions specified for the other type?

Plan Of Attack

Incremental steps that execute this change. Generally this is in the form of:

- [ ] Adjust the Remap transform to allow it to handle both log and metric.
- [ ] Implement `remap-lang::Object` for metric events. `remap-lang::Object` may need a new function to `get_metric` to return the metric that the metric events will work with. The other functions will just return an error.
- [ ] Write the functions required to work with the metric data.