Package 'openmetrics'

July 22, 2025

Title A 'Prometheus' Client for R Using the 'OpenMetrics' Format

Version 0.3.0

Description Provides a client for the open-source monitoring and alerting toolkit, 'Prometheus', that emits metrics in the 'OpenMetrics' format. Allows users to automatically instrument 'Plumber' and 'Shiny' applications, collect standard process metrics, as well as define custom counter, gauge, and histogram metrics of their own.

License MIT + file LICENSE

URL https://github.com/atheriel/openmetrics

BugReports https://github.com/atheriel/openmetrics/issues

Imports R6

Suggests httr, plumber, shiny, testthat (>= 2.1.0)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Author Aaron Jacobs [aut, cre],
Crescendo Technology Ltd. [cph]

Maintainer Aaron Jacobs <atheriel@gmail.com>

Repository CRAN

Date/Publication 2020-11-09 21:20:02 UTC

Contents

metrics *Metrics*

Description

A metric is a measure which can be aggregated into a time series, and comes in one of three types: counters, gauges, and histograms.

Metrics must have a unique name.

Usage

```
counter_metric(
  name,
  help,
  labels = character(),
  ...,
  unit = NULL,
  registry = global_registry()
)

gauge_metric(
  name,
  help,
  labels = character(),
  ...,
  unit = NULL,
  registry = global_registry()
)

histogram_metric(
  name,
  help,
  buckets = c(0.005, 0.01, 0.025, 0.05, 0.1, 0.25, 0.5, 1, 2.5, 5, 10),
  labels = character(),
  ...,
  unit = NULL,
  registry = global_registry()
)
```

Arguments

name	The name of the metric.
help	A brief, one-sentence explanation of the metric's meaning.
labels	A vector of label names for the metric.
...	For backward compatibility, otherwise ignored.
unit	An optional unit for the metric, e.g. "seconds". Must match the metric name.

registry	Where to register the metric for later retrieval.
buckets	A sequence of buckets to bin observations into. Defaults to Prometheus's suggested buckets, which are a good fit for measuring user-visible latency in seconds (e.g. for web services).

Details

All metric objects have a `reset()` method that reverts the underlying value (or values) to zero, an `unregister()` method that removes them from the registry they were created in, and a `render()` method that writes a representation of the metric in the text-based OpenMetrics format. Normally, `render_metrics()` is used instead.

In addition, various metrics have their own methods:

- `inc(by = 1, ...)`: Increments the metric by some positive number, defaulting to 1. Further parameters are interpreted as labels. Available for counters and gauges.
- `dec(by = 1, ...)`: Decrements the metric by some number, defaulting to 1. Further parameters are interpreted as labels. Available for gauges.
- `set(value, ...)`: Sets the metric to some number. Further parameters are interpreted as labels. Available for gauges.
- `set_to_current_time(...)`: Sets the metric to the current time, in seconds from the Unix epoch. Further parameters are interpreted as labels. Available for gauges.
- `observe(value, ...)`: Records an observation of some number. Further parameters are interpreted as labels. Available for histograms.
- `time(expr, ...)`: Records an observation for the time elapsed evaluating `expr`, in seconds. Further parameters are interpreted as labels. Available for histograms.

Value

An object with methods to manipulate the metric. See details.

See Also

The official documentation on Metric Types.

Examples

```
meows <- counter_metric("meows", "Heard around the house.", labels = "cat")
meows$inc(cat = "Shamus") # Count one meow from Shamus.
meows$inc(3, cat = "Unknown") # Count three meows of unknown origin.
meows$render()

thermostat <- gauge_metric("thermostat", "Thermostat display.")
thermostat$set(21.3) # Read from the display...
thermostat$dec(2) # ... and then turn it down 2 degrees.
thermostat$render()

temperature <- histogram_metric(
  "temperature", "Ambient room temperature measurements.",
  buckets = c(10, 15, 20, 22, 25), labels = "room"
```

```
)
set.seed(9090)
# Simulate taking ambient temperature samples.
for (measure in rnorm(20, mean = 21.5)) {
  temperature$observe(measure, room = sample(c("kitchen", "bathroom"), 1))
}
temperature$render()
```

---

push_to_gateway *Pushgateway Integration*

Description

Some workloads may not want to run an HTTP server to expose metrics, especially in the case of short-lived batch jobs. For these cases metrics can also be manually "pushed" to a Prometheus Pushgateway instance, though there are drawbacks.

`push_to_gateway()` is used to push metrics, and `delete_from_gateway()` is used to clean them up when the workload is finished.

Usage

```
push_to_gateway(url, job, instance = NA, registry = global_registry(), ...)

delete_from_gateway(url, job, instance = NA, ...)
```

Arguments

`url`	The URL of the Pushgateway
`job`	A value for the `job` label applied to all pushed metrics.
`instance`	A value for the `instance` label applied to all pushed metrics, or `NA` to leave it unset.
`registry`	A `Registry` object, defaulting to the shared global one.
`...`	Additional named string arguments converted to labels. Beware that these are not yet checked for URL safety.

Value

`NULL`, invisibly.

Examples

```
## Not run:
register_default_metrics()
push_to_gateway("localhost:9091", job = "batch-job-1")
# Some time later...
delete_from_gateway("localhost:9091", job = "batch-job-1")
```

```
## End(Not run)
```

`register_default_metrics` *Default Metrics for R Processes*

Description

Registers the standard process metrics for Prometheus clients. Not all metrics are supported on all operating systems.

Usage

```
register_default_metrics(registry = global_registry())
```

Arguments

`registry`	A `Registry` object. See `registry()`.

Value

Called for side effects only.

Examples

```
register_default_metrics()
render_metrics()
```

`register_plumber_metrics` *Metrics for Plumber APIs*

Description

Automatically wrap a Plumber API app, adding metrics for HTTP request count and duration, and then expose them on a `/metrics` endpoint.

The endpoint will check the `METRICS_HTTP_AUTHORIZATION` environment variable, and if present will use it as the expected `Authorization` header of the request to the `/metrics` endpoint. This can be used to implement basic HTTP authentication for access to runtime metrics.

Usage

```
register_plumber_metrics(app, registry = global_registry())
```

Arguments

app	A Plumber router object.
registry	A Registry object. See registry().

Value

A modified Plumber router.

Examples

```
if (requireNamespace("plumber", quietly = TRUE)) {
  app <- plumber::plumber$new() # Normally this is plumber::plumb().
  app <- register_plumber_metrics(app)
## Not run:
  app$run()

## End(Not run)
}
```

---

register_shiny_metrics

Metrics for Shiny Applications

Description

Automatically wrap a Shiny app, adding metrics for the current session count and the duration of reactive flushes, and then expose them on a /metrics endpoint.

The endpoint will check the METRICS_HTTP_AUTHORIZATION environment variable, and if present will use it as the expected Authorization header of the request to the /metrics endpoint. This can be used to implement basic HTTP authentication for access to runtime metrics.

Usage

```
register_shiny_metrics(app, registry = openmetrics::global_registry())
```

Arguments

app	An object created with shiny::shinyApp().
registry	A Registry object. See registry().

Value

A modified Shiny app object.

registry *Metric Registries*

Description

A registry is a collection of one or more metrics. By default, metrics are added to the object returned by `global_registry()`, but new registries can also be created with `registry()`. Use `collect_metrics()` to return all metrics that a registry is aware of, or `render_metrics()` to render all of them in aggregate.

Usage

```
registry()

global_registry()

collect_metrics(registry = global_registry())

render_metrics(registry = global_registry())
```

Arguments

`registry`	A `Registry` object, defaulting to the shared global one.

Details

`Registry` objects have methods, but they are not intended to be called by users and have no stable API.

Value

`registry()` and `global_registry()` return `Registry` objects (see Details), while `collect_metrics()` returns a list of `metrics` and `render_metrics()` returns a string.

Index