Adding Prometheus Metrics to a Go Application

Types of Metrics

Prometheus supports 4 types of metrics:

Counter

A *counter* is a cumulative metric that represents a single monotonically increasing counter whose value can only increase or be reset to zero on restart. For example, you can use a counter to represent the number of requests served, tasks completed, or errors.

Do not use a counter to expose a value that can decrease. For example, do not use a counter for the number of currently running processes; instead use a gauge.

Gauge

A *gauge* is a metric that represents a single numerical value that can arbitrarily go up and down.

Gauges are typically used for measured values like temperatures or current memory usage, but also "counts" that can go up and down, like the number of concurrent requests.

Histogram

A *histogram* samples observations (usually things like request durations or response sizes) and counts them in configurable buckets. It also provides a sum of all observed values.

A histogram with a base metric name of `<basename>` exposes multiple time series during a scrape:

- cumulative counters for the observation buckets, exposed as `<basename>_bucket{le="<upper inclusive bound>"}`
- the **total sum** of all observed values, exposed as `<basename>_sum`
- the **count** of events that have been observed, exposed as `<basename>_count` (identical to `<basename>_bucket{le="+Inf"}` above)

Summary

Similar to a *histogram*, a *summary* samples observations (usually things like request durations and response sizes). While it also provides a total count of observations and a sum of all observed values, it calculates configurable quantiles over a sliding time window.

A summary with a base metric name of `<basename>` exposes multiple time series during a scrape:

- streaming **-quantiles** (0 1) of observed events, exposed as `<basename>{quantile="<>"}`
- the **total sum** of all observed values, exposed as `<basename>_sum`
- the **count** of events that have been observed, exposed as `<basename>_count`

Coding

Libraries

```
go get github.com/prometheus/client_golang/prometheus
go get github.com/prometheus/client_golang/prometheus/promauto
go get github.com/prometheus/client_golang/prometheus/promhttp
```

Sample Code

```
package main

import (
        "net/http"
        "time"

        "github.com/prometheus/client_golang/prometheus"
        "github.com/prometheus/client_golang/prometheus/promauto"
        "github.com/prometheus/client_golang/prometheus/promhttp"
)

func recordMetrics() {
        go func() {
                for {
                        opsProcessed.Inc()
                        time.Sleep(2 * time.Second)
                }
        }()
}

var (
        opsProcessed = promauto.NewCounter(prometheus.CounterOpts{
                Name: "myapp_processed_ops_total",
                Help: "The total number of processed events",
        })
)

func main() {
        recordMetrics()

        http.Handle("/metrics", promhttp.Handler())
        http.ListenAndServe(":2112", nil)
}
```

Access the Metric

```
curl http://localhost:2112/metrics
```

Annotations

Annotating a Service

```
apiVersion: v1
kind: Service
metadata:
  name: kafka-azure-sink
  labels:
    app: kafka-azure-sink
  annotations:
    prometheus.io/scrape: "true"
    prometheus.io/port: "8080"
    prometheus.io/path: "/metrics"
spec:
  selector:
    app: kafka-azure-sink
  ports:
    - name: http
      port: 8080
      targetPort: 8080
```

Annotating a Pod

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kafka-azure-sink
  labels:
    app: kafka-azure-sink
spec:
  replicas: 1
  selector:
    matchLabels:
      app: kafka-azure-sink
  template:
    metadata:
      labels:
        app: kafka-azure-sink
      annotations:
        prometheus.io/scrape: "true"
        prometheus.io/port: "8080"
        prometheus.io/path: "/metrics"
    spec:
      containers:
...
```

Important Metrics

Name	Description
go_memstats_sys_bytes	Total Used Memory
go_memstats_heap_sys_bytes	Memory In Heap
go_memstats_mcache_sys_bytes	Meomory In Off Heap
go_memstats_stack_inuse_bytes	Memory In Stack
go_goroutines	Go Routines

Adding Scraping Config to Prometheus

Values.yaml

```
grafana:
  defaultDashboardsTimezone: America/Toronto
  adminPassword: admin

prometheus-node-exporter:
  hostRootFsMount:
    enabled: false

prometheus:
  prometheusSpec:
    additionalScrapeConfigs:
     - job_name: kafka-azure-sink
       static_configs:
         - targets: ['kafka-azure-sink:8080']
     - job_name: kafka-stream-operator
       static_configs:
         - targets: ['kafka-stream-operator:8080']
```

References

Reference	URL
INSTRUMENTING A GO APPLICATION FOR PROMETHEUS	https://prometheus.io/docs/guides/go-application/