

metrics readerHTTP

gmetricCounter, UpDownCounter, Histogram

```
package main

import (
    "go.opentelemetry.io/otel/exporters/prometheus"

    "github.com/gogf/gf/contrib/metric/otelmetric/v2"
    "github.com/gogf/gf/v2/frame/g"
    "github.com/gogf/gf/v2/os/gctx"
    "github.com/gogf/gf/v2/os/gmetric"
)

var (
    meter = gmetric.GetGlobalProvider().Meter(gmetric.MeterOption{
        Instrument:      "github.com/gogf/gf/example/metric
/basic",
        InstrumentVersion: "v1.0",
    })
    counter = meter.MustCounter(
        "goframe.metric.demo.counter",
        gmetric.MetricOption{
            Help: "This is a simple demo for Counter usage",
            Unit: "bytes",
        },
    )
    upDownCounter = meter.MustUpDownCounter(
        "goframe.metric.demo.updown_counter",
        gmetric.MetricOption{
            Help: "This is a simple demo for UpDownCounter
usage",
            Unit: "%",
        },
    )
    histogram = meter.MustHistogram(
        "goframe.metric.demo.histogram",
        gmetric.MetricOption{
            Help: "This is a simple demo for histogram
usage",
            Unit:      "ms",
            Buckets: []float64{0, 10, 20, 50, 100, 500, 1000,
2000, 5000, 10000},
        },
    )
)

func main() {
    var ctx = gctx.New()

    // Prometheus exporter to export metrics as Prometheus format.
    exporter, err := prometheus.New(
        prometheus.WithoutCounterSuffixes(),
        prometheus.WithoutUnits(),
    )
    if err != nil {
        g.Log().Fatal(ctx, err)
    }

    // OpenTelemetry provider.
    provider := otelmetric.MustProvider(otelmetric.WithReader
(exporter))
```

## Content Menu

- 
- [Counter&UpDownCounter](#)
- [Histogram](#)
- [Primetheus Exporter](#)

```
provider.SetAsGlobal()
defer provider.Shutdown(ctx)

// Counter.
counter.Inc(ctx)
counter.Add(ctx, 10)

// UpDownCounter.
upDownCounter.Inc(ctx)
upDownCounter.Add(ctx, 10)
upDownCounter.Dec(ctx)

// Record values for histogram.
histogram.Record(1)
histogram.Record(20)
histogram.Record(30)
histogram.Record(101)
histogram.Record(2000)
histogram.Record(9000)
histogram.Record(20000)

// HTTP Server for metrics exporting.
otelmetric.StartPrometheusMetricsServer(8000, "/metrics")
}
```

# Counter&UpDownCounter

CounterUpDownCounterCounterUpDownCounterPrometheusCounterPrometheusCounterUpDo  
wnCounterPrometheusGauge

# Histogram

Histogramp95, p99/Histogram

# Primetheus Exporter

PrometheusPrometheus

```
otelmetric.StartPrometheusMetricsServer(8000, "/metrics")
```

<http://127.0.0.1:8000/metrics>



