

metrics readermetrics readerCPU

gmetricObservableCounter, ObservableUpDownCounter, ObservableGaugeObservable

```
package main

import (
    "context"

    "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",
    })
    observableCounter = meter.MustObservableCounter(
        "goframe.metric.demo.observable_counter",
        gmetric.MetricOption{
            Help: "This is a simple demo for ObservableCounter
usage",
            Unit: "%",
        },
    )
    observableUpDownCounter = meter.MustObservableUpDownCounter(
        "goframe.metric.demo.observable_updown_counter",
        gmetric.MetricOption{
            Help: "This is a simple demo for
ObservableUpDownCounter usage",
            Unit: "%",
        },
    )
    observableGauge = meter.MustObservableGauge(
        "goframe.metric.demo.observable_gauge",
        gmetric.MetricOption{
            Help: "This is a simple demo for ObservableGauge
usage",
            Unit: "%",
        },
    )
)

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

    // Callback for observable metrics.
    meter.MustRegisterCallback(func(ctx context.Context, obs gmetric.
Observer) error {
        obs.Observed(observableCounter, 10)
        obs.Observed(observableUpDownCounter, 20)
        obs.Observed(observableGauge, 30)
        return nil
    }, observableCounter, observableUpDownCounter, observableGauge)

    // Prometheus exporter to export metrics as Prometheus format.
    exporter, err := prometheus.New(
```

Content Menu

-
- [Meter Callback](#)
- [Metric Callback](#)
- [Prometheus Exporter](#)

```

        prometheus.WithoutCounterSuffixes(),
        prometheus.WithoutUnits(),
    )
    if err != nil {
        g.Log().Fatal(ctx, err)
    }

    // OpenTelemetry provider.
    provider := otelmetric.MustProvider(otelmetric.WithReader
(exporter))
    provider.SetAsGlobal()
    defer provider.Shutdown(ctx)

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

```

Meter Callback

CallbackCallbackCallbackObserveObserve

- ObservableCounter/ObservableUpDownCounterObserve
- ObservableGaugeObserveObserve

Metric Callback

Meter CallBackMetricOptionCallback

```

observableCounter = meter.MustObservableCounter(
    "goframe.metric.demo.observable_counter",
    gmetric.MetricOption{
        Help: "This is a simple demo for ObservableCounter usage",
        Unit: "%",
        Callback: func(ctx context.Context, obs gmetric.
MetricObserver) error {
            obs.Observe(10)
            return nil
        },
    },
)

```

Primetheus Exporter

Prometheus

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

<http://127.0.0.1:8000/metrics>

[illegible]