Writing Prometheus Exporters

Moshe Zadka – https://cobordism.com

2020

Acknowledgement of Country

San Francisco Bay Area Peninsula
Ancestral home of the Ramaytush Ohlone

Prometheus

- Metrics collection and aggregation
- Pull-based
- Time-series-based

Exporter

- Facade
- Not magic

Configuration

```
scrape_configs:
  - job_name: 'my-thing'
    static_configs:
      - targets: ['hostname.internal.example.com:90
```

Gauge

Measured instanenous value

Gauge

Measured instanenous value

memory usage

Historgram

Samples

Historgram

Samples
request latency

Counter

Ticks up until reset

Counter

Ticks up until reset
total requests

Pyramid Service: Configuration

```
with Configurator() as config:
    configure_metrics(config.registry)
    config.add_route("metrics", "/metrics")
    config.add_view(
        metrics_web, route_name="metrics"
    )
    application = config.make_wsgi_app()
```

Pyramid Service: Metrics Configuration

```
def configure_metrics(mapping):
    registry = CollectorRegistry()
    mapping["prometheus_registry"] = registry
    mapping["level"] = Gauge(
        "level",
        "The level",
        registry=registry,
    )
    mapping["hits"] = Counter(
        "hits",
        "Hits to endpoint",
        registry=registry,
    )
```

Pyramid Service: View

```
def metrics_web(request):
    update(request)
    registry = request.registry[
        "prometheus_registry"
    ]
    return Response(
        generate_latest(registry),
        content_type=CONTENT_TYPE_LATEST,
    )
```

Pyramid Service: Metrics Update

```
def update(request):
    num = random.uniform(0, 1)
    request.registry["hits"].inc()
    request.registry["level"].set(num)
```

Output

```
# HELP level The level
# TYPE level gauge
level 0.06199425960241822
# HELP hits_total Hits to endpoint
# TYPE hits_total counter
hits_total 1.0
# HELP hits_created Hits to endpoint
# TYPE hits_created gauge
hits_created 1.6041066280902677e+09
```

Daytime Service

RFC 867

Daytime Service

RFC 867

```
$ nc localhost 1113
Sat Oct 31 18:15:12 2020
```

Synthetic Query

Exporter Judo

Synthetic Query

Exporter Judo

```
def synthetic(service):
    before = time.perf_counter()
    sock = socket.socket()
    sock.settimeout(1)
    sock.connect(service)
    sock.recv(4096)
    sock.close()
    return time.perf_counter() - before
```

Output

```
# HELP synthetic Synthetic query latency
# TYPE synthetic gauge
synthetic 0.0003916006535291672
# HELP hits_total Hits to endpoint
# TYPE hits_total counter
hits_total 2.0
# HELP hits_created Hits to endpoint
# TYPE hits_created gauge
hits_created 1.6042536905489218e+09
```

What is a Counter

- Increase-only
- Creation time

Side-car Counter

```
class SidecarCounter(metrics.MetricWrapperBase):
    _type = "counter"
    def _metric_init(self):
        self._value = values.ValueClass(
            self._type,
            self._name,
            self._name + "_total",
            self._labelnames,
            self._labelvalues,
        )
        self._created = values.ValueClass(
            self._type,
            self._name,
            self._name + "_created",
            self._labelnames,
            self._labelvalues,
        )
```

Set/Get Counter

```
def set(self, what, *, when):
    self._value.set(what)
    self._created.set(when)
def _child_samples(self):
    return (
        ("_total", {}, self._value.get()),
        ("_created", {}, self._created.get()),
    )
```

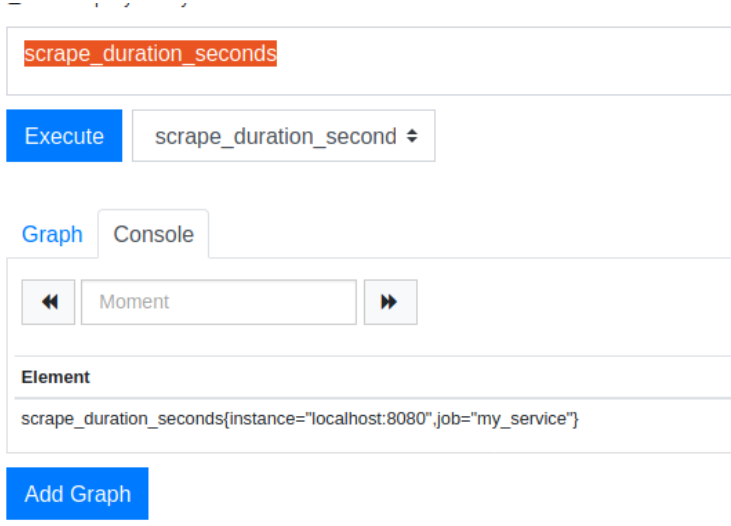
Counters in Response

```
$ nc localhost 1111 | \
> python -m json.tool
{
    "launched": "2020-11-04T17:57:42.448578",
    "requests": 2
}
```

Exporting Counters

```
def hits(service):
    with socket.create_connection(service) as sock:
        sock.settimeout(1)
        data = sock.recv(4096)
    result = json.loads(data)
    result["launched"] = isoparse(result["launched"
    return result
```

How It Looks


scrape_duration_seconds
Execute
scrape_duration_second
Graph
Console
Moment
Element
scrape_duration_seconds{instance="localhost:8080",job="my_service"}
Add Graph

Metrics Design

- Histogram vs. Gauge

Metrics Design

- Histogram vs. Gauge
- Exporter latency

Metrics Design

- Histogram vs. Gauge
- Exporter latency
- Enum

Prometheus Exporters

- Easy

Prometheus Exporters

- Easy
- Why: Internal

Prometheus Exporters

- Easy
- Why: Internal
- Why: Better