به نام خدا

دانشگاه صنعتي اميركبير (پلی تکنیک تهران)
دانشکده مهندسی کامپیوتر

درس شبکه‌های کامپیوتری

موضوع: گزارشکار پروژه

نیم‌سال دوم ۱۴۰۰-۱۴۰۱

1. ابتدا با دستور pip install psutil این کتابخانه را نصب میکنیم.
متریک‌هایی را که میخوام مانیتور کنیم را این این کتابخانه مشخص کرده و سپس آنها را که به صورت لیست ذخیره کرده بودیم در قالب JSON برای سرور که هاست و پورت آن‌را از قبل مشخص کرده‌ایم با سوکت میفرستیم.
کد Client پیوست شده است.

2. برای پیاده‌سازی سرور همانطور که در ویدیو گفته شد یک سوکت با هاست و پورت دلخواه میسازیم که در اینجا host را 127.0.0.1 و پورت را 8080 در نظر گرفته‌ایم.
از آنجایی که به سرور ممکن است بیشتر از یک کلاینت متصل شوند از کتابخانه threading استفاده میکنیم.
پرومتئوس ۴ متریک دارد که در بخش بعدی توضیح داده میشود.
در اینجا چون مقادیر دریافت شده از کلاینت کم و زیاد میشوند و در حال تغییر هستند از Gauge استفاده میکنیم.
در این پروژه چون از هر کلاینت ۳ تا متریک فرستاده میشود ما سه تا Gauge در نظر میگیریم و آنها را لیبل‌گذاری میکنیم تا زمانی که چاپ میشوند مشخص شود متریک دریافت شده برای کدام کلاینت است.
کد سرور پیوست شده است.

3. به طور کلی Prometheus، 4 نوع معیار برای نظارت بر برنامه های پایتون ارائه می دهد:

I. Counter: برای شمارش تعداد یا اندازه یک رویداد استفاده می شود. یعنی تعداد بازدیدکنندگان، تعداد بازدید از صفحه، تعداد خطاها، میزان داده های ارائه شده توسط وب سرور. یک مقدار اولیه را می توان روی یک شمارنده تنظیم کرد. از آن مقدار، مقدار Counter افزایش می یابد. شما نمی توانید ارزش یک شمارنده را کاهش دهید. اما، اگر اسکریپت پایتون را متوقف کنید و دوباره آن را اجرا کنید، شمارنده ریست می شود.

```
from prometheus_client import Counter
c = Counter('my_failures', 'Description of counter')
c.inc()     # Increment by 1
c.inc(1.6)  # Increment by given value
```

II. Gauge: برای شمارش تعداد یا اندازه وضعیت فعلی یک رویداد استفاده می شود. یعنی تعداد درخواست هایی که در حال حاضر در حال پردازش هستند، مقدار حافظه ای که برنامه استفاده می کند، تعداد کاربرانی که در حال حاضر وارد شده اند. برخلاف Counter، مقدار Gauge را می توان کم و زیاد کرد.

```
from prometheus_client import Gauge
g = Gauge('my_inprogress_requests', 'Description of gauge')
g.inc()      # Increment by 1
g.dec(10)    # Decrement by given value
g.set(4.2)   # Set to a given value
```

III. Summery: برای ردیابی تاخیر یک رویداد استفاده می شود. یعنی زمانی که یک تابع برای تکمیل یک کار صرف می کند، مدت زمان لازم برای ارائه یک صفحه وب، مدت زمان لازم برای پاسخ به یک درخواست API.

```python
from prometheus_client import Summary
s = Summary('request_latency_seconds', 'Description of summary')
s.observe(4.7)    # Observe 4.7 (seconds in this case)
```

IV. Histogram: برای ردیابی اندازه و تعداد رویدادها در یک bucket از پیش تعریف شده استفاده می شود. bucket آرایه ای از اعداد مرتب شده (اعداد صحیح و کسری) است که هیستوگرام برای گروه بندی داده ها استفاده می کند.

```python
from prometheus_client import Histogram
h = Histogram('request_latency_seconds', 'Description of histogram')
h.observe(4.7)    # Observe 4.7 (seconds in this case)
```

یک job جدید به فایل Prometheus.yml اضافه میکنیم که پورت آن ۸۱۰۰ است و در کد سرور هم این پورت را قرار میدهیم.

```yaml
# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this conf
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["localhost:9090"]

  - job_name: "my_exporter"
    static_configs:
      - targets: ["localhost:8100"]
```

بخش مربوطه به سرور :

```python
start_http_server(8100)
```

اجرای پرومتئوس :

```
hasti@Hastis-MacBook-Pro Files % brew services restart prometheus
==> Successfully started `prometheus` (label: homebrew.mxcl.prometheus)
hasti@Hastis-MacBook-Pro Files % pip install prometheus-client
Requirement already satisfied: prometheus-client in /Library/Frameworks/Python.f
ramework/Versions/3.9/lib/python3.9/site-packages (0.9.0)
hasti@Hastis-MacBook-Pro Files %
```

خروجی :

```
# HELP python_gc_objects_collected_total Objects collected during gc
# TYPE python_gc_objects_collected_total counter
python_gc_objects_collected_total{generation="0"} 298.0
python_gc_objects_collected_total{generation="1"} 43.0
python_gc_objects_collected_total{generation="2"} 0.0
# HELP python_gc_objects_uncollectable_total Uncollectable object found during GC
# TYPE python_gc_objects_uncollectable_total counter
python_gc_objects_uncollectable_total{generation="0"} 0.0
python_gc_objects_uncollectable_total{generation="1"} 0.0
python_gc_objects_uncollectable_total{generation="2"} 0.0
# HELP python_gc_collections_total Number of times this generation was collected
# TYPE python_gc_collections_total counter
python_gc_collections_total{generation="0"} 38.0
python_gc_collections_total{generation="1"} 3.0
python_gc_collections_total{generation="2"} 0.0
# HELP python_info Python platform information
# TYPE python_info gauge
python_info{implementation="CPython",major="3",minor="9",patchlevel="2",version="3.9.2"} 1.0
# HELP Memory_Usage Description of Memory_Usage
# TYPE Memory_Usage gauge
Memory_Usage{client_number="client_1"} 64.0
Memory_Usage{client_number="client_2"} 64.2
# HELP Disk_Usage Description of Disk_Usage
# TYPE Disk_Usage gauge
Disk_Usage{client_number="client_1"} 59.6
Disk_Usage{client_number="client_2"} 59.6
# HELP CPU_Percent Description of CPU_Percent
# TYPE CPU_Percent gauge
CPU_Percent{client_number="client_1"} 2.6
CPU_Percent{client_number="client_2"} 2.1
```

در صورتی که سرور با خطا مواجه شود و کلاینت دسترسی اش را از دست بدهد :

```
hasti@Hastis-MacBook-Pro Project % python3 client.py
[Errno 61] Connection refused
We Lost the Connection to Server, if you would like to reconnect enter 1, ot
herwise enter 0: 1
Connected to Server 127.0.0.1:8080
Message Number 0 Sent to Server
Message Number 1 Sent to Server
Message Number 2 Sent to Server
Message Number 3 Sent to Server
```

از کلاینت پرسیده میشود که میخواهد مجدد با سرور ارتباط برقرار کند یا خیر و اگر کلاینت بخواهد در صورتی که سرور در دسترس باشد به سرور متصل میشود.