

Client

برای پیاده‌سازی کلاینت از کتابخانه socket استفاده می‌کنیم و برای متریک‌ها از کتابخانه psutil استفاده کردیم و با استفاده از فرمت JSON، متریک‌ها را برای سرور ارسال می‌کنیم. ۳ متریک CPU، RAM و DISK را با psutil داریم که درصد استفاده را به ما می‌دهند.

```
from encodings import utf_8
from time import sleep
import socket
import psutil
import json

HOST = "127.0.0.1"
PORT = 8080

def main():
    while True:
        sleep(3)
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            try:
                s.connect((HOST, PORT))
                print("Socket successfully created")
                counter = 1
                with s:
                    info = {
                        'RAM': psutil.virtual_memory()[2],
                        'CPU': psutil.cpu_percent(5),
                        'DISK': psutil.disk_usage('/')[3]
                    }
                    data = json.dumps(info)
                    s.send(data.encode("utf_8"))
                    print("Data " + str(counter) + " sent to server.")
                    counter += 1
                    sleep(5)
            except socket.error as err:
                print("Socket creation failed %s" %(err))
                print("Try again? y/n")
                inp = input()
                if(inp == "n"):
                    break
                else:
                    continue
```

Server

اینجا نیز با استفاده از socket پیاده‌سازی را انجام می‌دهیم و با کلاینت ارتباط را برقرار می‌کنیم و داده‌ها را ارسال می‌کنیم و باز هم از JSON برای دریافت متریک‌ها با فرمت درست استفاده می‌کنیم.

Prometheus

پرومتئوس یک جعبه ابزار مانیتورینگ و هشدار سیستم منبع باز است. پرومتئوس متریک‌های خود را به عنوان داده‌های سری زمانی جمع‌آوری و ذخیره می‌کند، یعنی اطلاعات متریک با time stamp که در آن ثبت شده است، در کنار جفت‌های اختیاری key-value به نام labelها ذخیره می‌شود.

متریک‌ها اندازه‌گیری‌های عددی هستند. سری زمانی به این معنی است که تغییرات در طول زمان ثبت می‌شوند. آنچه کاربران می‌خواهند اندازه‌گیری کنند از برنامه‌ای به برنامه دیگر متفاوت است. برای یک وب سرور ممکن است زمان درخواست باشد، برای یک پایگاه داده ممکن است تعداد اتصالات فعال یا تعداد جستجوهای فعال و غیره باشد. متریک‌ها نقش مهمی در درک اینکه چرا برنامه شما به روش خاصی کار می‌کند بازی می‌کند.

در اینجا متریک‌های دریافت شده در فرمت JSON را با استفاده از Gauge به پرومتئوس می‌دهیم. چرا Gauge؟ زیرا متریکی است که یک مقدار عددی واحد را نشان می‌دهد که می‌تواند بالا و پایین برود و متغیرهای ما در لحظه افزایش و کاهش دارند.

۳ نوع متریک دیگر برای ما مناسب نیستند زیرا counter یک متریک تجمعی است که نشان دهنده یک شمارنده یکنواخت در حال افزایش است که مقدار آن تنها با راه اندازی مجدد می‌تواند افزایش یابد یا به صفر بازنشانی شود و برای نشان دادن مقداری که ممکن است کاهش یابد شمارنده مناسب نیست. به عنوان مثال، از شمارنده برای تعداد فرآیندهای در حال اجرا استفاده نمی‌شود. Histogram هم از مشاهدات (معمولاً چیزهایی مانند مدت زمان درخواست یا اندازه پاسخ) نمونه برداری می‌کند و آنها را در باکتهای قابل تنظیم شمارش می‌کند. همچنین مجموع تمام مقادیر مشاهده شده را ارائه می‌دهد. Summary نیز شبیه به هیستوگرام، یک خلاصه مشاهدات را نمونه می‌کند (معمولاً چیزهایی مانند مدت زمان درخواست و اندازه پاسخ). در حالی که تعداد کل مشاهدات و مجموع تمام مقادیر مشاهده شده را نیز ارائه می‌دهد، quantileهای قابل تنظیم را در یک پنجره زمانی کشویی محاسبه می‌کند.

همچنین پرومتئوس را روی پورت ۸۰۰۰ بالا می‌آوریم. یک job نیز به فایل yml پرومتئوس اضافه کرده تا target ایجاد شود.

```
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - 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:8000"]
```

Targets

All

Unhealthy

Collapse All

🔍

Filter by endpoint or labels

my_exporter (1/1 up)

show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:8000/metrics	UP	instance="localhost:8000" job="my_exporter"	7.17s ago	313.485ms	

prometheus (1/1 up)

show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	9.713s ago	7.185ms	

```

from concurrent.futures import thread
from encodings import utf_8
import json
import socket
import threading
from prometheus_client import start_http_server, Counter, Gauge

HOST = "127.0.0.1"
PORT = 8080

ram = Gauge('process_ram_usage', 'ram usage', ['client_number'])
cpu = Gauge('process_cpu_usage', 'cpu usage', ['client_number'])
disk = Gauge('process_disk_usage', 'disk usage', ['client_number'])

def recv_data(conn, num, addr, PORT):
    print(f"Connected by {addr}")
    data = conn.recv(1024).decode("utf_8")
    print(f"[{addr}] {data}")

    client_dats = json.loads(data)
    print(client_dats)
    datas = []
    datas.append(client_dats['RAM'])
    datas.append(client_dats['CPU'])
    datas.append(client_dats['DISK'])

    ram.labels("agent" + str(num)).set(datas[0])
    cpu.labels("agent" + str(num)).set(datas[1])
    disk.labels("agent" + str(num)).set(datas[2])
    conn.close()

```

```

def main():
    while True:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.bind((HOST, PORT))
            s.listen(10)
            start_http_server(8000)
            num = 1
            while True:
                conn, addr = s.accept()
                thread = threading.Thread(target = recv_data, args = (conn, num, addr, PORT))
                thread.start()
                print(f"[active clients] {num}")
                num += 1


if __name__ == '__main__':
    main()

```


Execution

```
C:\Users\Moujan\Desktop\New folder (2)\newSC>python server.py
Connected by ('127.0.0.1', 58067)
[active clients] 1
```

```
C:\Users\Moujan\Desktop\New folder (2)\newSC>python client.py
Socket successfully created
```

 C:\Windows\System32\cmd.exe - python server.py

```
{'RAM': 69.2, 'CPU': 10.7, 'DISK': 88.5}
Connected by ('127.0.0.1', 51231)
[active clients] 578
[('127.0.0.1', 51231)] {"RAM": 68.9, "CPU": 2.8, "DISK": 88.5}
{'RAM': 68.9, 'CPU': 2.8, 'DISK': 88.5}
Connected by ('127.0.0.1', 51234)
[active clients] 579
[('127.0.0.1', 51234)] {"RAM": 69.0, "CPU": 3.9, "DISK": 88.5}
{'RAM': 69.0, 'CPU': 3.9, 'DISK': 88.5}
Connected by ('127.0.0.1', 51237)
[active clients] 580
[('127.0.0.1', 51237)] {"RAM": 68.9, "CPU": 3.2, "DISK": 88.5}
{'RAM': 68.9, 'CPU': 3.2, 'DISK': 88.5}
Connected by ('127.0.0.1', 52556)
[active clients] 581
[('127.0.0.1', 52556)] {"RAM": 68.9, "CPU": 3.5, "DISK": 88.5}
{'RAM': 68.9, 'CPU': 3.5, 'DISK': 88.5}
Connected by ('127.0.0.1', 52558)
[active clients] 582
[('127.0.0.1', 52558)] {"RAM": 69.0, "CPU": 2.7, "DISK": 88.5}
{'RAM': 69.0, 'CPU': 2.7, 'DISK': 88.5}
Connected by ('127.0.0.1', 52562)
[active clients] 583
[('127.0.0.1', 52562)] {"RAM": 69.0, "CPU": 0.9, "DISK": 88.5}
{'RAM': 69.0, 'CPU': 0.9, 'DISK': 88.5}
Connected by ('127.0.0.1', 52565)
[active clients] 584
```

 Select C:\Windows\System32\cmd.exe - python client.py

```
Data 1 sent to server.
Socket successfully created
Data 1 sent to server.
Socket successfully created
Data 1 sent to server.
Socket successfully created
Data 1 sent to server.
Socket successfully created
Data 1 sent to server.
Socket successfully created
Data 1 sent to server.
```

☐ Use local time

☐ Enable query history

☒ Enable autocomplete

☒ Enable highlighting

☒ Enable linter

Q process_disk_usage

🔍 Execute

Load time: 32ms Resolution: 14s Result series: 591

Table Graph

< Evaluation time >

process_disk_usage(client_number="agent1", instance="localhost:8000", job="my_exporter")	92.3
process_disk_usage(client_number="agent10", instance="localhost:8000", job="my_exporter")	92.3
process_disk_usage(client_number="agent100", instance="localhost:8000", job="my_exporter")	89.5
process_disk_usage(client_number="agent101", instance="localhost:8000", job="my_exporter")	89.5
process_disk_usage(client_number="agent102", instance="localhost:8000", job="my_exporter")	89.5
process_disk_usage(client_number="agent103", instance="localhost:8000", job="my_exporter")	89.5

☐ Use local time

☐ Enable query history

☒ Enable autocomplete

☒ Enable highlighting

☒ Enable linter

Q process_cpu_usage

🔍 Execute

Load time: 13ms Resolution: 14s Result series: 5

Table Graph

< Evaluation time >

process_cpu_usage(client_number="agent1", instance="localhost:8000", job="my_exporter")	3.8
process_cpu_usage(client_number="agent2", instance="localhost:8000", job="my_exporter")	2.7
process_cpu_usage(client_number="agent3", instance="localhost:8000", job="my_exporter")	6.3
process_cpu_usage(client_number="agent4", instance="localhost:8000", job="my_exporter")	9.7
process_cpu_usage(client_number="agent5", instance="localhost:8000", job="my_exporter")	10.2

Remove Panel

Add Panel

☐ Use local time

☐ Enable query history

☒ Enable autocomplete

☒ Enable highlighting

☒ Enable linter

Q process_ram_usage

🔍 Execute

Load time: 23ms Resolution: 14s Result series: 591

Table Graph

< Evaluation time >

process_ram_usage(client_number="agent1", instance="localhost:8000", job="my_exporter")	87.5
process_ram_usage(client_number="agent10", instance="localhost:8000", job="my_exporter")	86.5
process_ram_usage(client_number="agent100", instance="localhost:8000", job="my_exporter")	77.4
process_ram_usage(client_number="agent101", instance="localhost:8000", job="my_exporter")	77.3
process_ram_usage(client_number="agent102", instance="localhost:8000", job="my_exporter")	77.2
process_ram_usage(client_number="agent103", instance="localhost:8000", job="my_exporter")	77.2
process_ram_usage(client_number="agent104", instance="localhost:8000", job="my_exporter")	77.6