Client

برای پیادهسازی کلاینت از کتابخانه socket استفاده می کنیم و برای متریکها از کتابخانه psutil استفاده کردیم psutil و psutil را با DISK و PU ،RAM و PSU را با DISK و با استفاده از فرمت JSON، متریکها را برای سرور ارسال می کنیم. ۳ متریک درصد استفاده را به ما می دهند.

```
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:
                    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 پیاده سازی را انجام می دهیم و با کلاینت ارتباط را برقرار می کنیم و داده ها را ارسال می کنیم و باز هم از JSON برای دریافت متریک ها با فرمت درست استفاده می کنیم.

Prometheus

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

متریکها اندازه گیری های عددی هستند. سری زمانی به این معنی است که تغییرات در طول زمان ثبت می شوند. آنچه کاربران می خواهند اندازه گیری کنند از برنامه ای به برنامه دیگر متفاوت است. برای یک وب سرور ممکن است زمان در خواست 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 می کند بازی می کند.

در اینجا متریکهای دریافت شده در فرمت JSON را با استفاده از Gauge به پرومتئوس می دهیم. چرا Gauge؟ زیرا متریکی است که یک مقدار عددی واحد را نشان می دهد که می تواند 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 می تواند 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 (معمولا چیزهایی مانند مدت زمان درخواست یا اندازه پاسخ) نمونه برداری می کند و آنها را در باکتهای قابل تنظیم شمارش می کند. همچنین مجموع تمام مقادیر مشاهده شده را ارائه می دهد. Summary نیز شبیه به هیستوگرام، یک خلاصه مشاهدات را نمونه می کند (معمولا چیزهایی مانند مدت زمان درخواست و اندازه پاسخ). در حالی که تعداد کل مشاهدات و مجموع تمام مقادیر مشاهده شده را نیز ارائه می دهد، equantile قابل تنظیم را در یک پنجره زمانی کشویی محموع تمام مقادیر مشاهده شده را نیز ارائه می دهد، equantile قابل تنظیم را در یک پنجره زمانی کشویی محاسبه می کند.

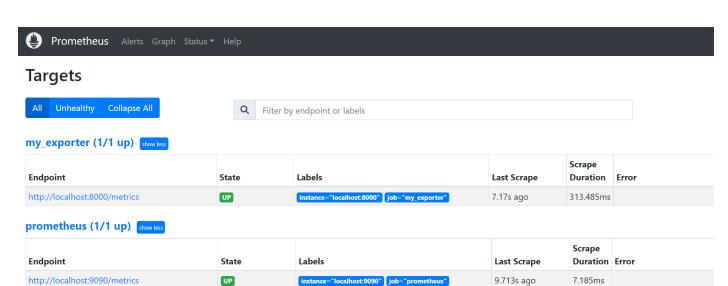
همچنین پرومتئوس را روی پورت ۸۰۰۰ بالا می آوریم. یک job نیز به فایل yml پرومتئوس اضافه کرده تا target ایجاد شود.

```
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"]
```



```
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 datas = json.loads(data)
    print(client datas)
    datas = []
    datas.append(client datas['RAM'])
   datas.append(client_datas['CPU'])
    datas.append(client datas['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()
```

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

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

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

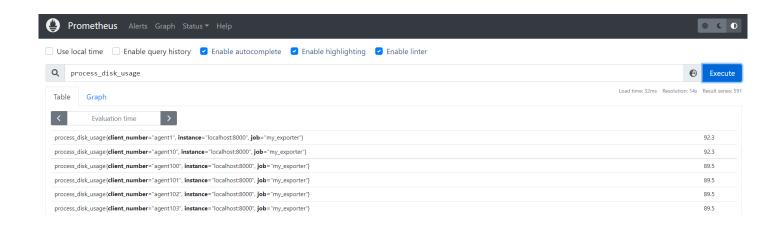
```
{'RAM': 69.2, 'CPU': 10.7, 'DISK': 88.5}
Connected by ('127.0.0.1', 51231)
[active clients] 578
[('127.0.0.1', 51231)] {"RAM": 68.9, "CPU": 2.8, "DISK": 88.5}
['RAM': 68.9, 'CPU': 2.8, 'DISK': 88.5}
.
Connected by ('127.0.0.1', 51234)
[active clients] 579
[('127.0.0.1', 51234)] {"RAM": 69.0, "CPU": 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
```

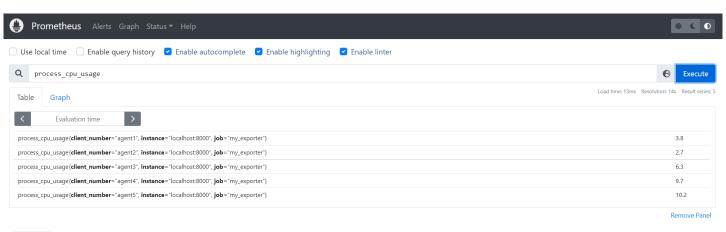
Select C:\Windows\System32\cmd.exe - python client.py

```
Data 1 sent to server.

Socket successfully created

Data 1 sent to server.
```





Add Pane

