

به نام خدا



پروژه شبکه

پرومئتوس



محمد جواد زندیه ۹۸۳۱۰۳۲

۳ خرداد ۱۴۰۱

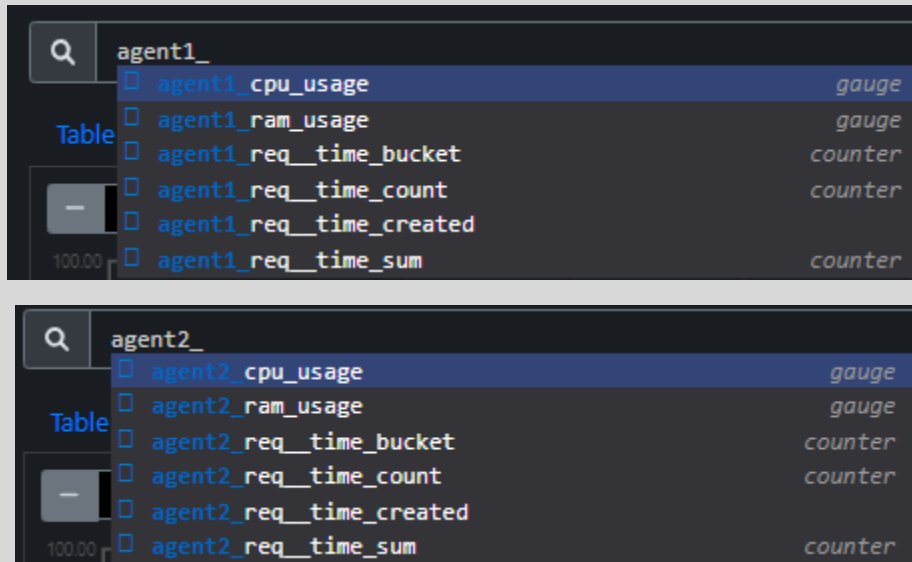
دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر

در بخش اول ابتدا کلاینت را پیاده سازی میکنیم. این فایل قصد دارد تا متریک های سیستم را استخراج کرده و برای سرور ارسال کند. در این پروژه از ۳ متریک استفاده شده است:

۱. `cpu_utilization`: میزان درصد استفاده از `cpu` را توسط کتابخانه `psutil` پایتون بدست می آوریم. برای نمایش این متریک هم در سمت سرور از `gauge` استفاده کرده ایم زیرا مقدار آن ممکن است کم و یا زیاد شود پس توسط تابع `set` می توان آنرا نمایش داد.

۲. `ram_utilization`: میزان درصد استفاده از `ram` را توسط کتابخانه `psutil` پایتون بدست می آوریم. همانند متریک قبلی در سمت سرور عمل می کنیم.

۳. `req_time`: زمانی که طول کشیده است درخواست در شبکه ارسال شود. این متریک را توسط متریک `Histogram` در سمت سرور نمایش می دهیم.



Search	Table	100.00
agent1_	<ul style="list-style-type: none">agent1_cpu_usage gaugeagent1_ram_usage gaugeagent1_req_time_bucket counteragent1_req_time_count counteragent1_req_time_created counteragent1_req_time_sum counter	
agent2_	<ul style="list-style-type: none">agent2_cpu_usage gaugeagent2_ram_usage gaugeagent2_req_time_bucket counteragent2_req_time_count counteragent2_req_time_created counteragent2_req_time_sum counter	

کد مربوط به پیاده سازی این بخش:

در این کد برای هر `agent` یک اسم انتخاب کرده ایم (هنگام ایجاد این اسم را به آن میدهیم) و متریک هایی که در سمت سرور میسازیم را با پسوند اسم هر `agent` میسازیم (که لیبل مربوط به `agent` ها متفاوت باشد).

اطلاعات و متریک های سمت کلاینت هم توسط `json` برای سرور ارسال می شود (تابع `dump` را روی `dictionary` پایتون باید صدا کنیم) و در سمت سرور این محتوای `json` به `dictionary` تبدیل میکنیم و متریک های مورد نظر را استخراج میکنیم.

برای اینکه بتوان چندین `agent` را در سمت سرور مدیریت کرد و اینکه در اثر مشکل بتواند آنرا برطرف کرد، در این کد از قابلیتی که جدیداً به پایتون اضافه شده است یعنی `asyncio` استفاده میکنیم که برای ما ارتباط های سنکرون را برقرار میکند.

قبل از ران کردن سرور هم باید پرومتهئوس را اجرا کنیم با دستور `prometheus.exe`.

همچنین برای آنکه بتوانیم نمودار ها و گراف های مربوطه را در `localhost:9090/metrics` مشاهده کنیم، باید یک ارتباط بین سروری که نوشتیم و پرومتهئوس ایجاد کنیم، که اینکار را روی پورت ۸۰۰۰ انجام میدهیم (دقت شود که ارتباط با `agent` ها روی پورت ۱۳۱۳ است و متفاوت با این ارتباط است)

مابقی پیاده سازی همانند چیزی است که در ویدیو مربوط به پروژه و ویدیو مربوط به `socket programming` در آزمایشگاه قرار داده است میباشد.

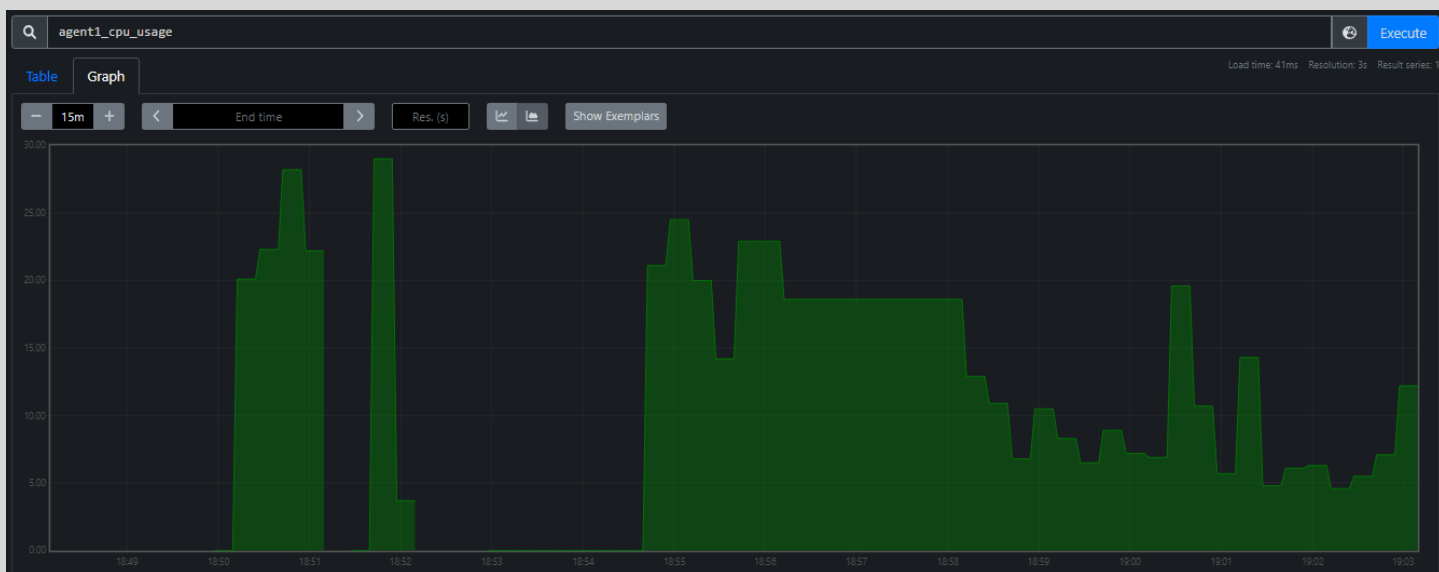
کد ها در کنار فایل گزارش وجود دارند و در زیر اسکرین های مربوط به پروژه قرار داده شده است:

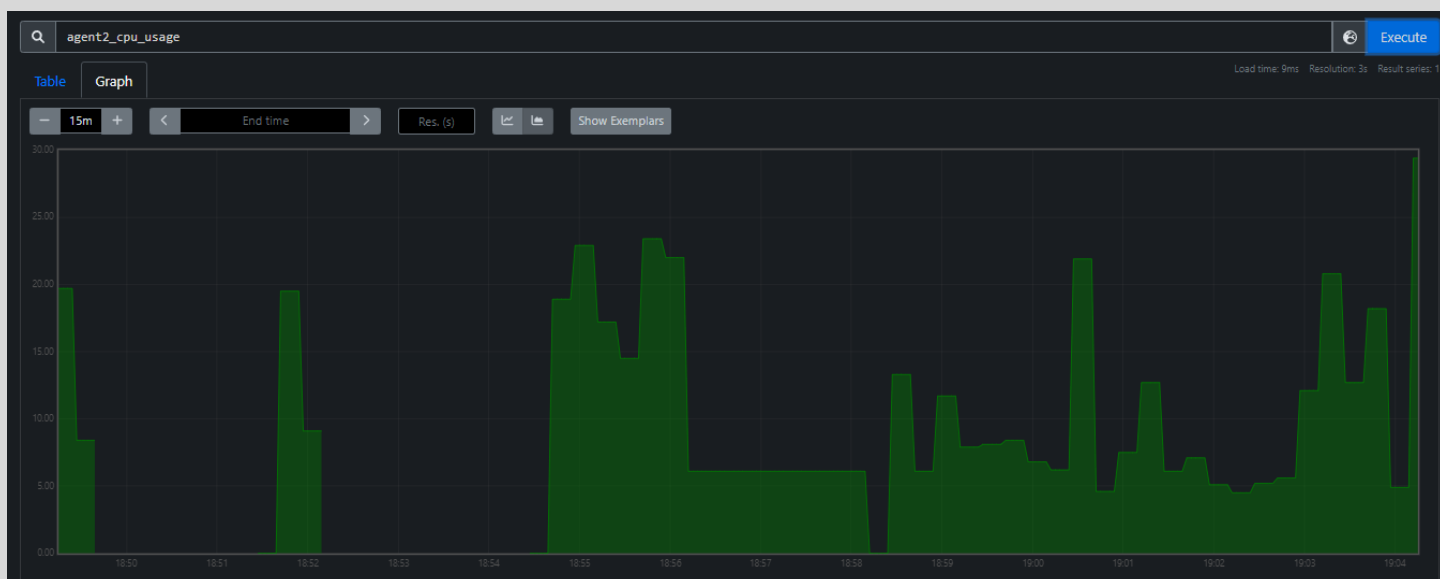
۱. اجرای پرومیتئوس، اجرای سرور، اجرای دو تا agent به صورت همزمان برای چک کردن multithreading:

The screenshot shows the Visual Studio Code interface. The terminal window displays the output of running Prometheus and two agents. The Prometheus logs show the server starting and listening on port 9090. The two agents (agent1 and agent2) are running successfully. The file explorer on the right shows the project structure, including the Prometheus configuration files and the agents' source code.

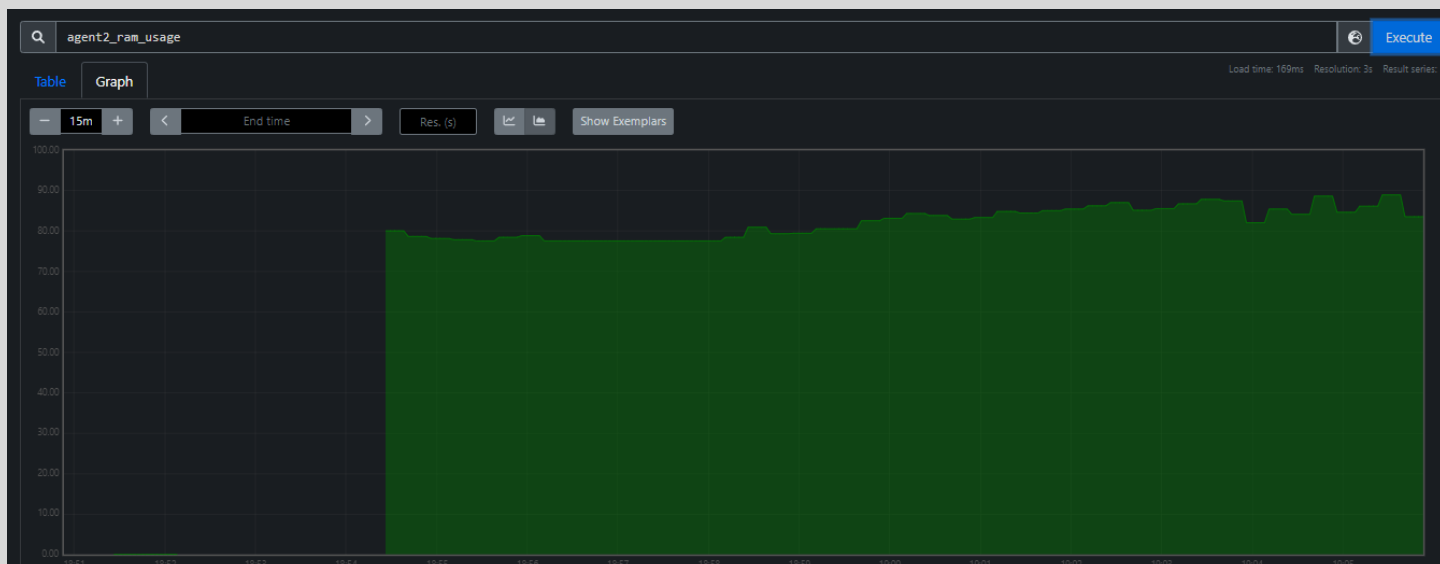
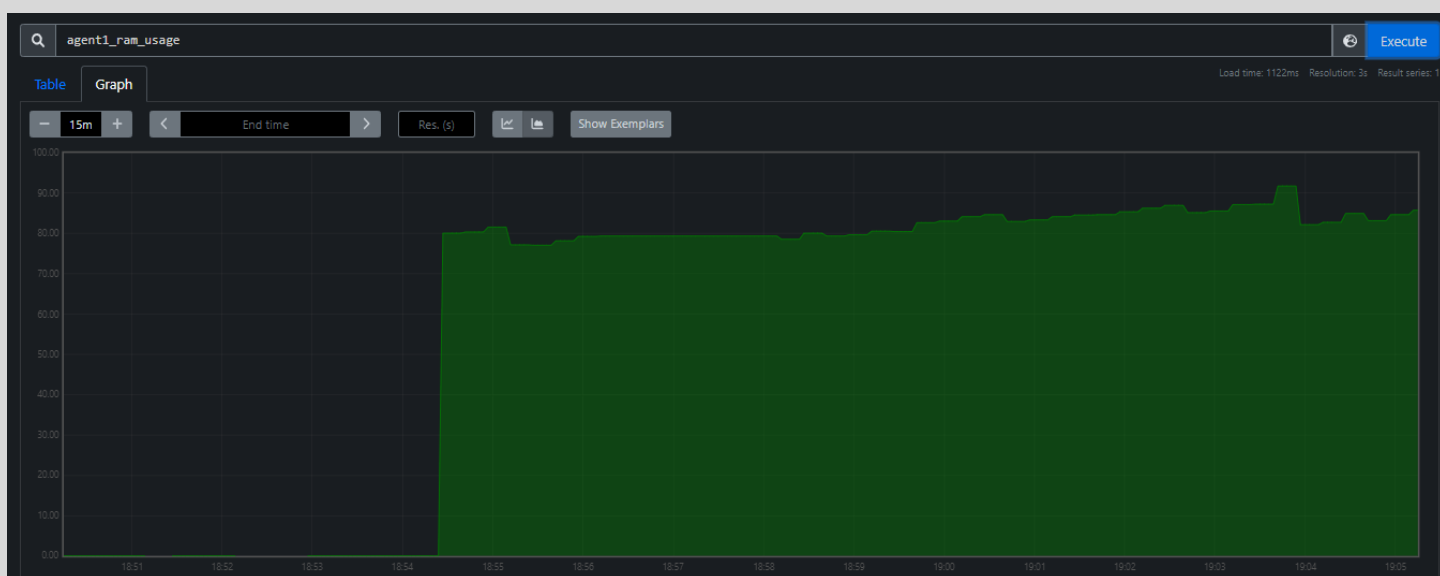
۲. نمایش متریک مربوط به cpu_utilization برای agent1 و agent2

دقت شود که لیبل های مربوط به هر agent با پیشوند نام آن شروع میشوند.

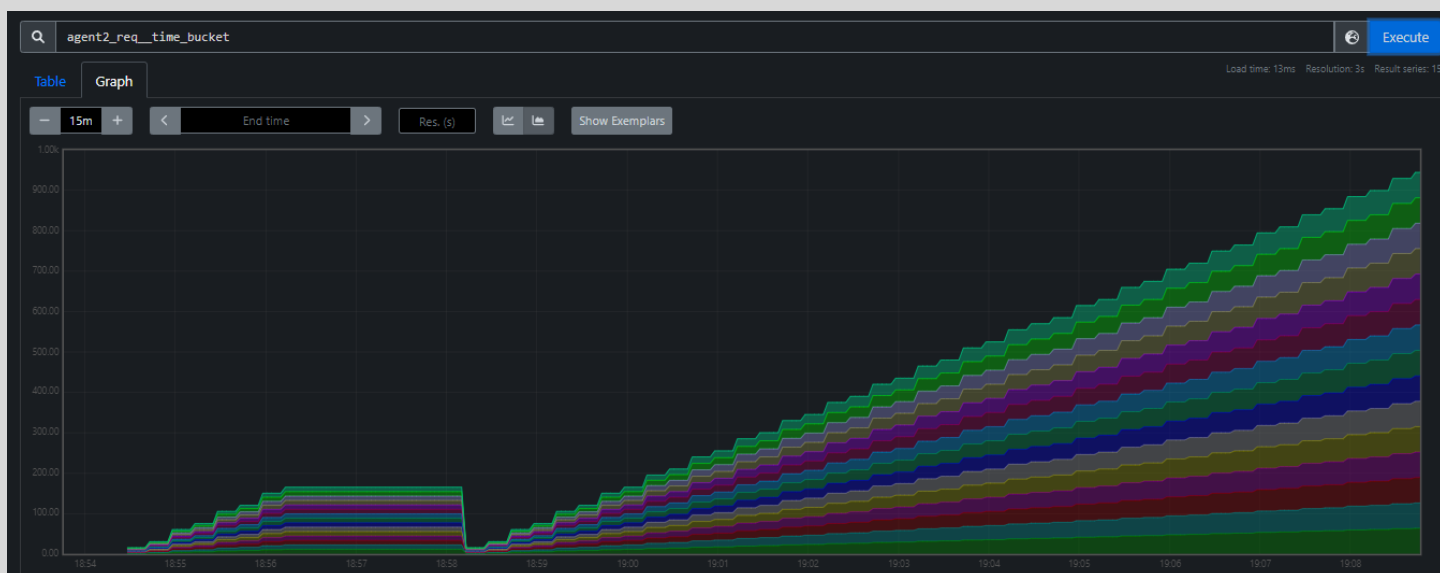
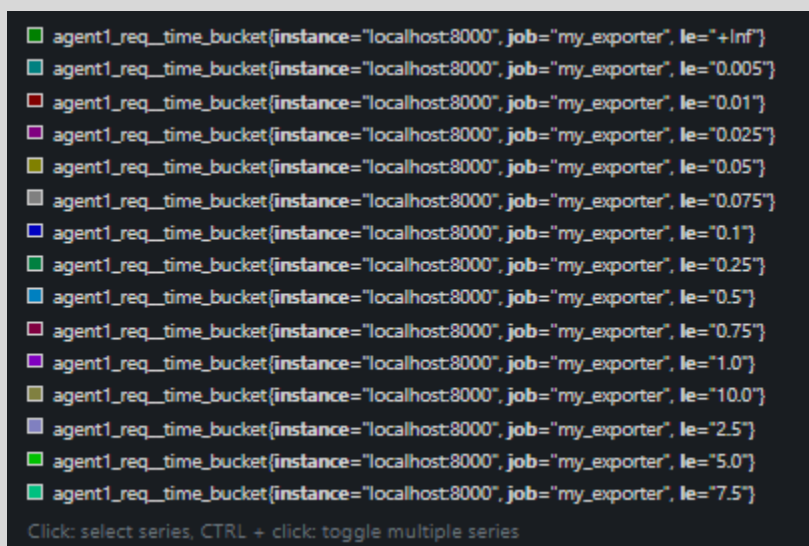
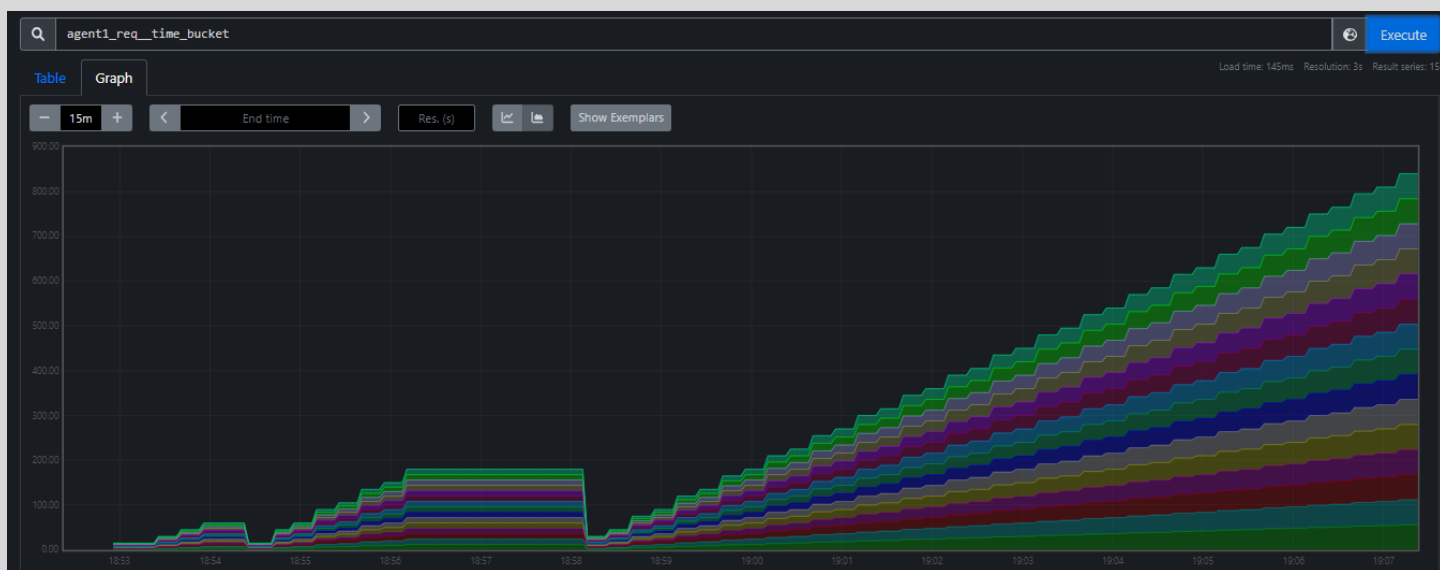




۳. Ram_utilization برای agent1 و agent2



۴. نمایش باکت های زمانی برای زمان درخواست در شبکه برای دو agent



```
agent2_req_time_bucket(instance="localhost:8000", job="my_exporter", le="+Inf")
agent2_req_time_bucket(instance="localhost:8000", job="my_exporter", le="0.005")
agent2_req_time_bucket(instance="localhost:8000", job="my_exporter", le="0.01")
agent2_req_time_bucket(instance="localhost:8000", job="my_exporter", le="0.025")
agent2_req_time_bucket(instance="localhost:8000", job="my_exporter", le="0.05")
agent2_req_time_bucket(instance="localhost:8000", job="my_exporter", le="0.075")
agent2_req_time_bucket(instance="localhost:8000", job="my_exporter", le="0.1")
agent2_req_time_bucket(instance="localhost:8000", job="my_exporter", le="0.25")
agent2_req_time_bucket(instance="localhost:8000", job="my_exporter", le="0.5")
agent2_req_time_bucket(instance="localhost:8000", job="my_exporter", le="0.75")
agent2_req_time_bucket(instance="localhost:8000", job="my_exporter", le="1.0")
agent2_req_time_bucket(instance="localhost:8000", job="my_exporter", le="10.0")
agent2_req_time_bucket(instance="localhost:8000", job="my_exporter", le="2.5")
agent2_req_time_bucket(instance="localhost:8000", job="my_exporter", le="5.0")
agent2_req_time_bucket(instance="localhost:8000", job="my_exporter", le="7.5")
```

