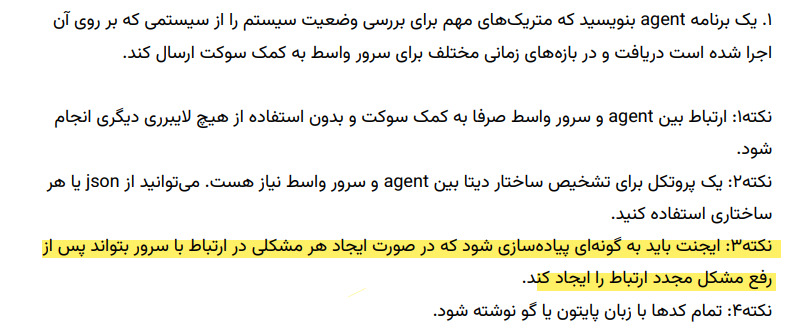
پروژه ی شبکه

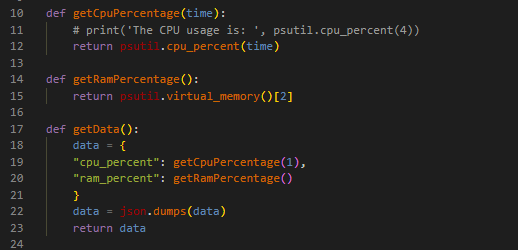
**دانشجو:** نگین خیرمند

**شماره دانشجویی:** 9831023

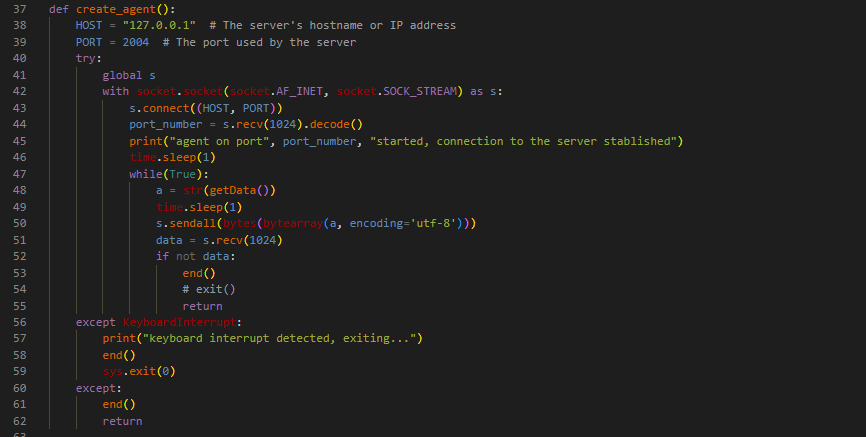
**توضیحات پروژه بخش اول:**



ابتدا یک سری method نوشته شده که داده های سیستم را دریافت و در فرمت مناسب(dictionary) قرار میدهند:



سپس method ای که مسول قسمت socket programming است را نوشته ام:



با توجه به اینکه قرار است روی پرت 2004 برنامه ی سرور را run بکنم برای همین پورت target را 2004 قرار داده ام.

و یک TCP connection به وجود اورده و داده های dictionary را به صورت JSON برای agent server فرستاده ام. برای handle کردن exception ها هم از try و except استفاده کردم.

و در نهایت برای اینکه اگر exception هم داد باز تلاش برای ساختن یک connection جدید بکنم در قسمت main صدا زدن این تابع را در while True قرار داده ام.



**توضیحات پروژه بخش دوم:**

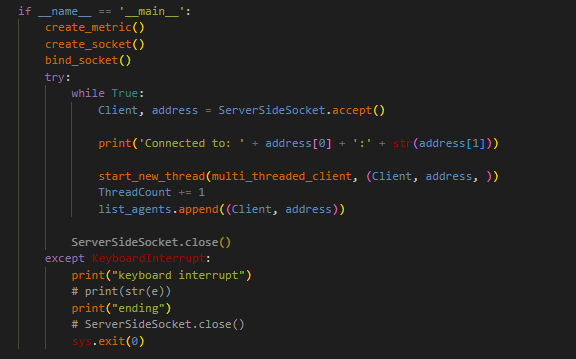


برای نشان دادن داده های دریافتی از agent های مختلف از class agentServer استفاده میشود. 

که طبق داکیومنتیشن نوشته شده:

<https://github.com/prometheus/client_python>

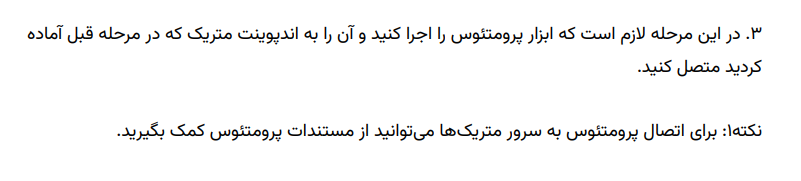
برای قسمت socket programming هم پس از ساختن socket و bind کردن آن، چون سرور باید توانایی ارتباط با چند agent به صورت همزمان را داشته باشد باید از thread ها استفاده کرد.



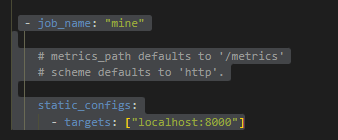
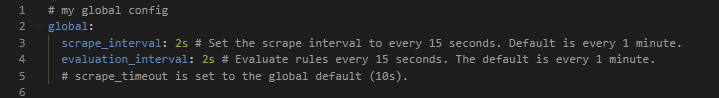


تنها نکته ای که کد دارد این است که socketData یک dictionary از dictionary است که هر key ان مربوط به connection ها با agent های مختلف است و value آن یک dictionary است که فرمت آن در کد agent مشخص شده است.

**توضیحات پروژه بخش سوم:**

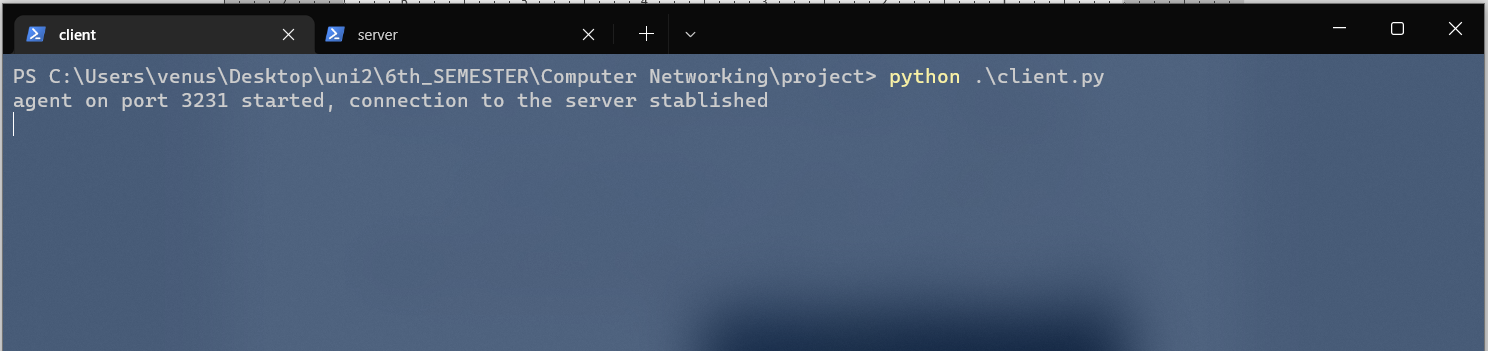
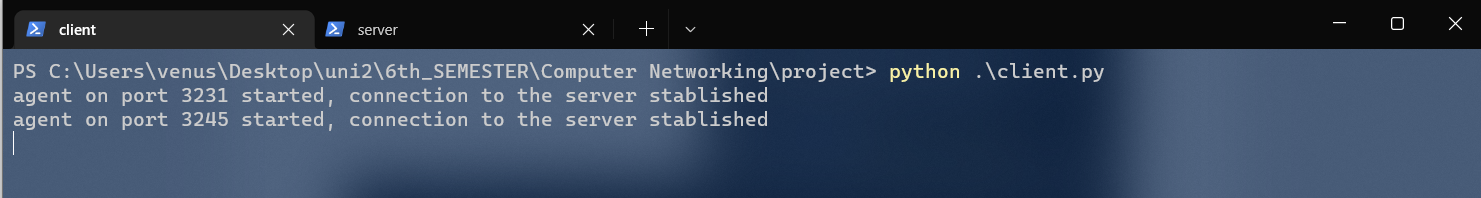


ابتدا target به فایل yml مربوط به prometheus اضافه شد و scrape\_interval و evaluation\_interval هم که برای update کردن استفاده میشوند به 2s تغییر یافتند:

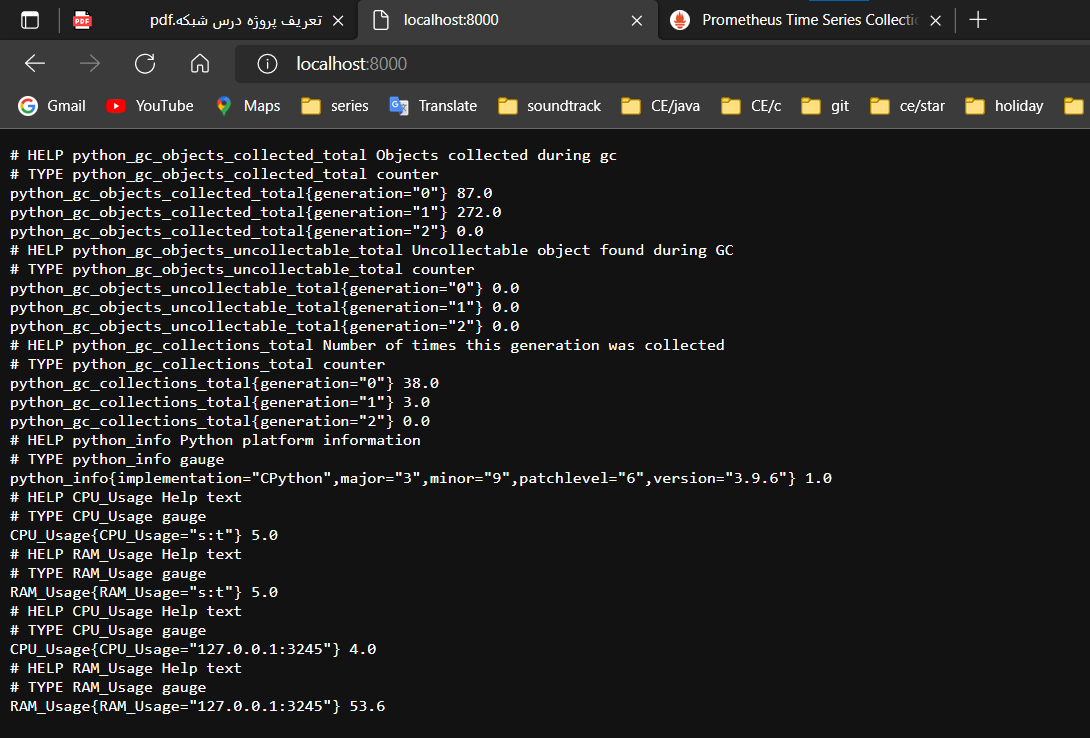
 

سپس prometheus.exe را run کردم:

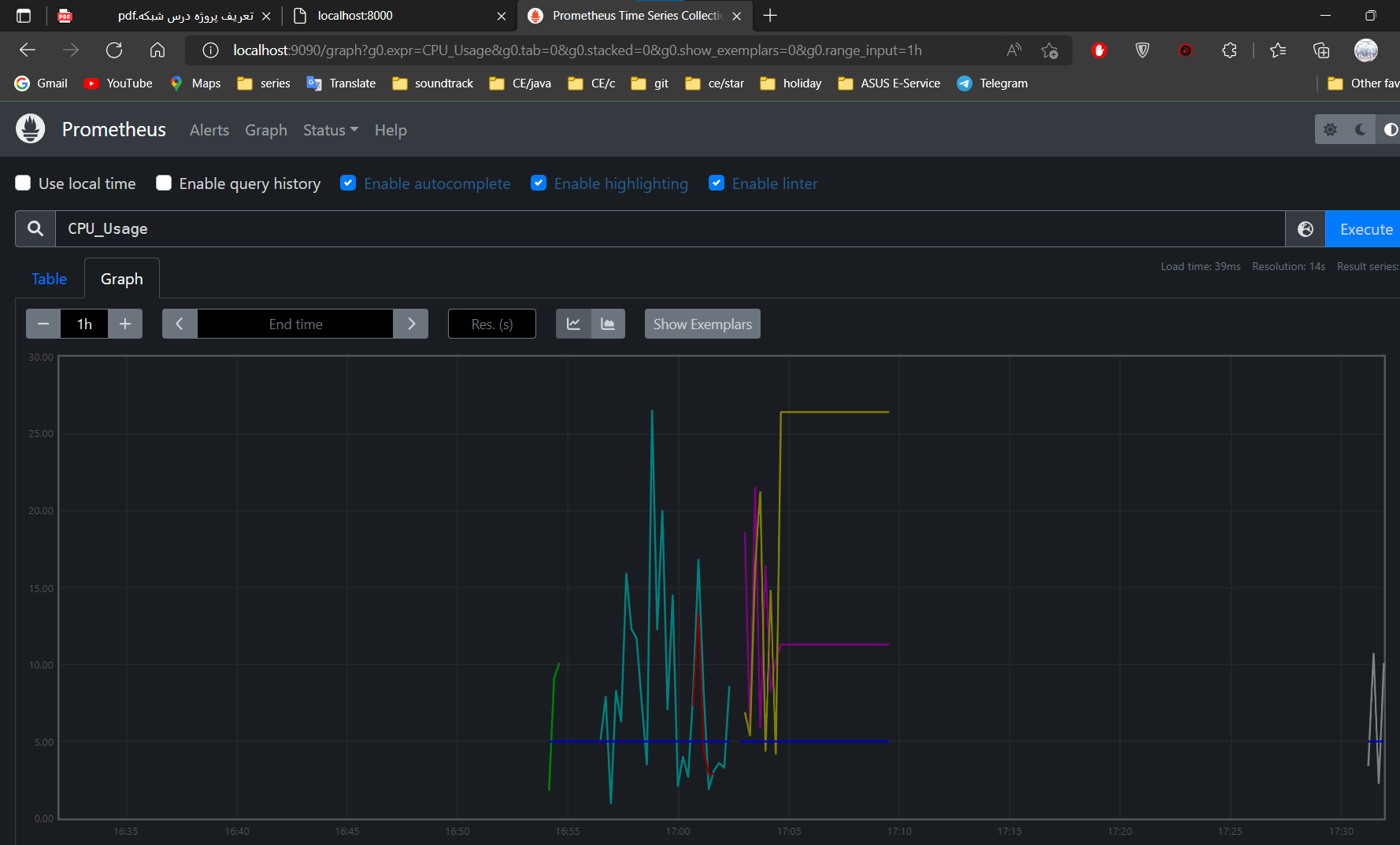
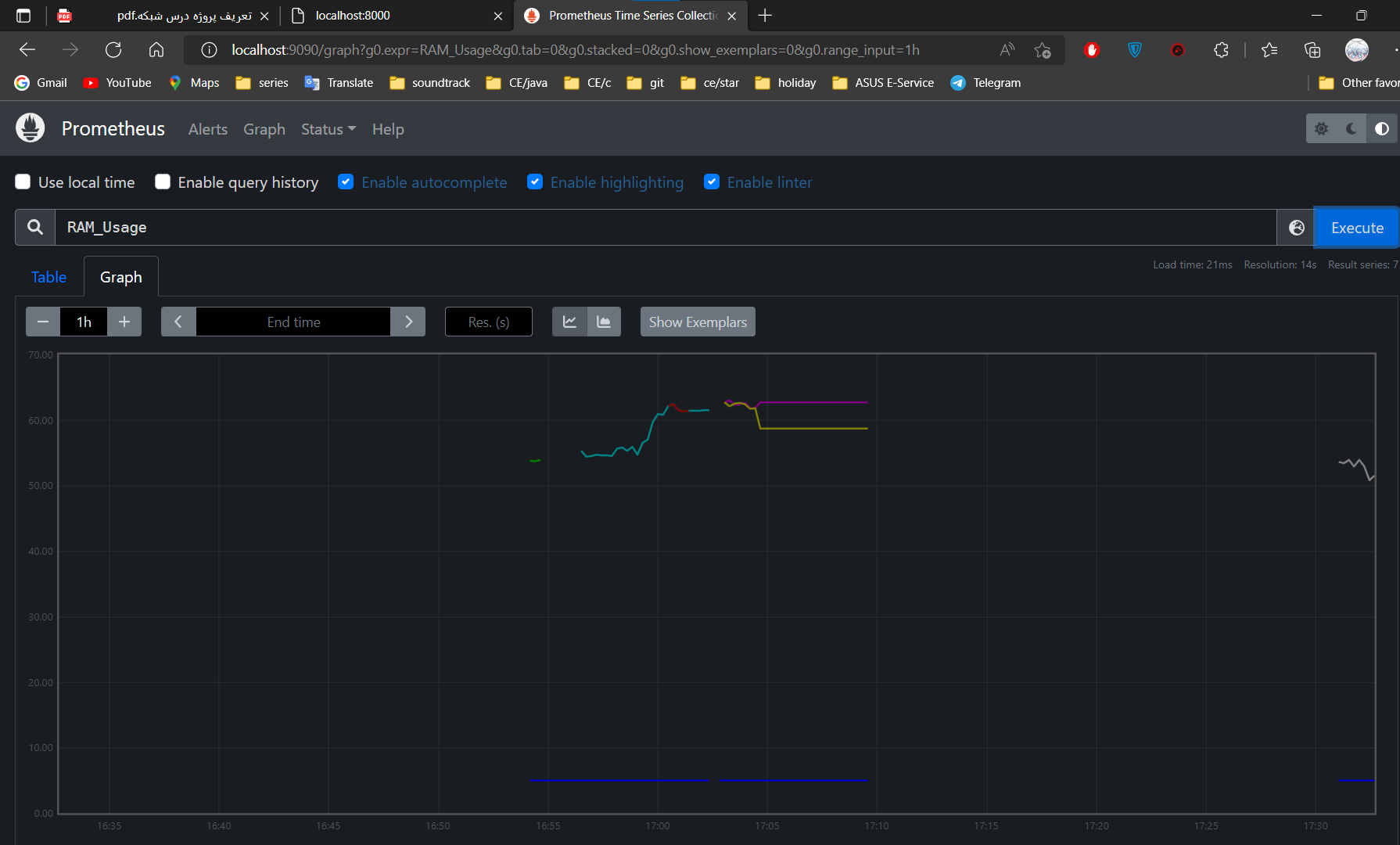
و برنامه های خود را نیز run کردم:



حالا برای دیدن داده های agent به 8000 رفتم:



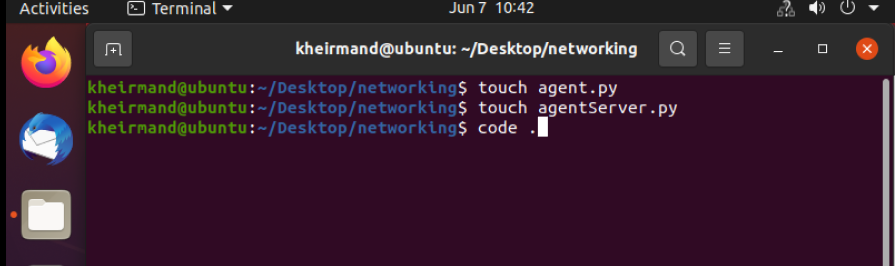
و برای دیدن prometheus به 9090 که پورن default است:

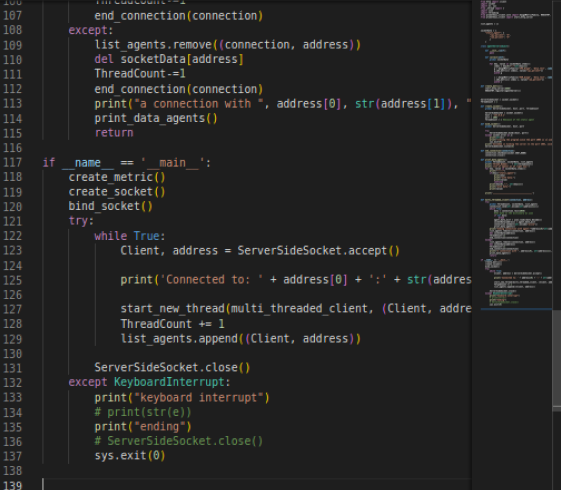
**قسمت امتیازی:**

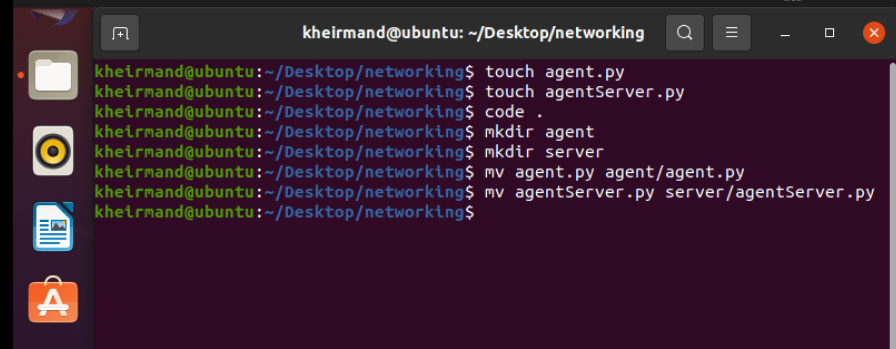
از توضیحات در مورد نصب داکر گذر میکنم. میتوانید به این لینک مراجعه کنید، من همین راه را رفته ام.

<https://support.netfoundry.io/hc/en-us/articles/360057865692-Installing-Docker-and-docker-compose-for-Ubuntu-20-04>



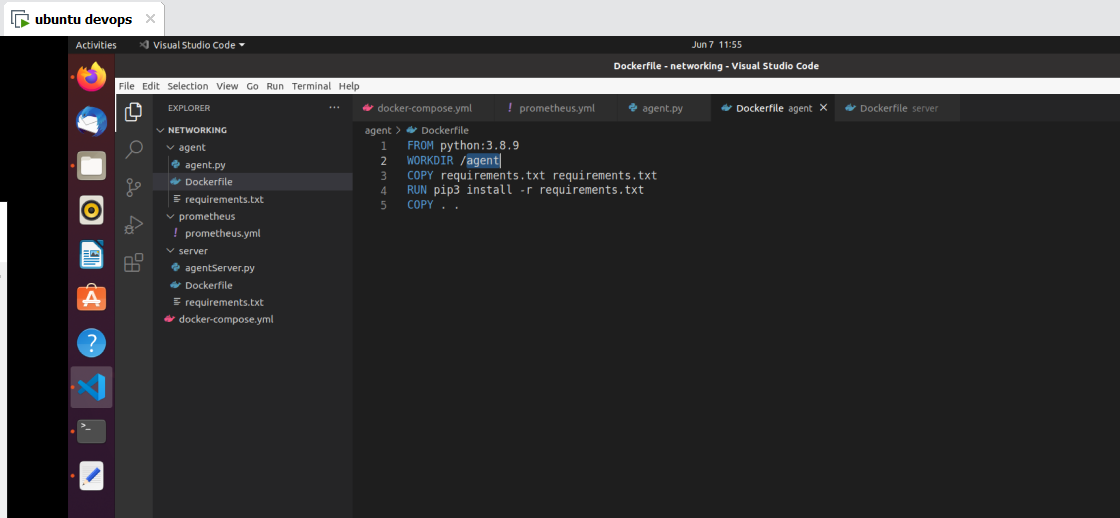
سپس کد های متناظر را در فایل های ساخته شده نوشتم:



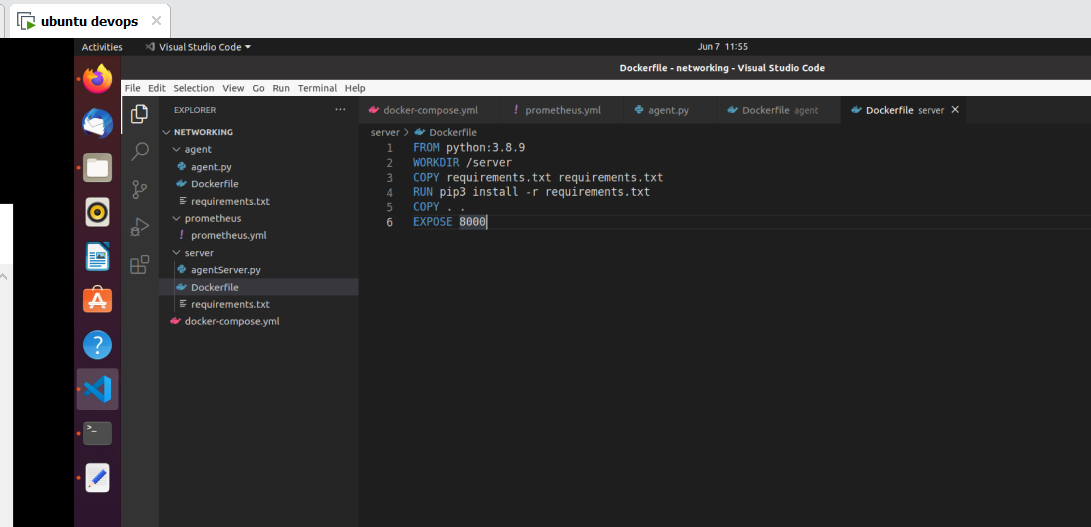


در نهایت با اضافه شدن Dockerfile ها به هر کدام از directory های server و agent با کد های زیر:

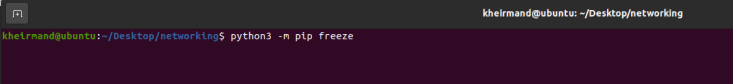
برای agent:



برای server



و freeze کردن dependency ها در فایل requierments.txt مربوط به هر فولدر با کامند زیر(باید virtual env ها فعال باشند):



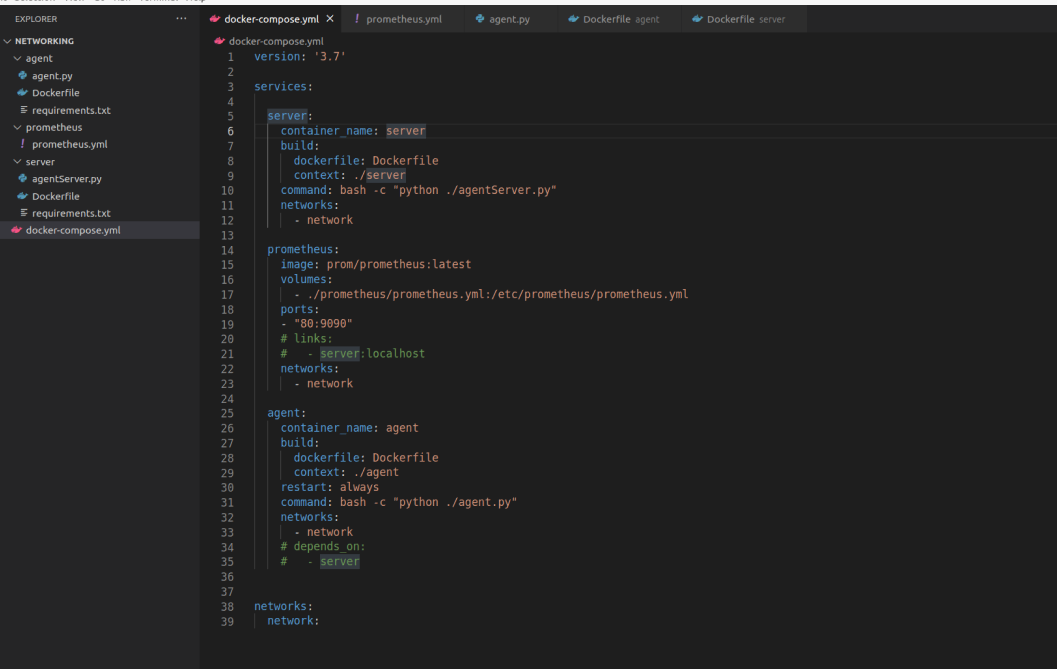
میتوان image های مورد نظر را ساخت(من پس از نوشتن docker-compse با کامند زیر آن ها را ساخته ام)

Docker-compose build

در dockerfile ها کار خاصی انجام ندادم. صرفا working directory مربوط به image مشخص شده، تمامی dependency ها مثلpsutil نصب شده و در نهایت تملم فایل ها که در اینجا صرفا agent.py یا agentServer.py هست به داخل image کپی شده.

برای server از انجایی که روی پورت 8000 قرار است metric ها باشند این پورت expose هم شده تا دیگر container ها در network ان بتوانند ان را ببینند.

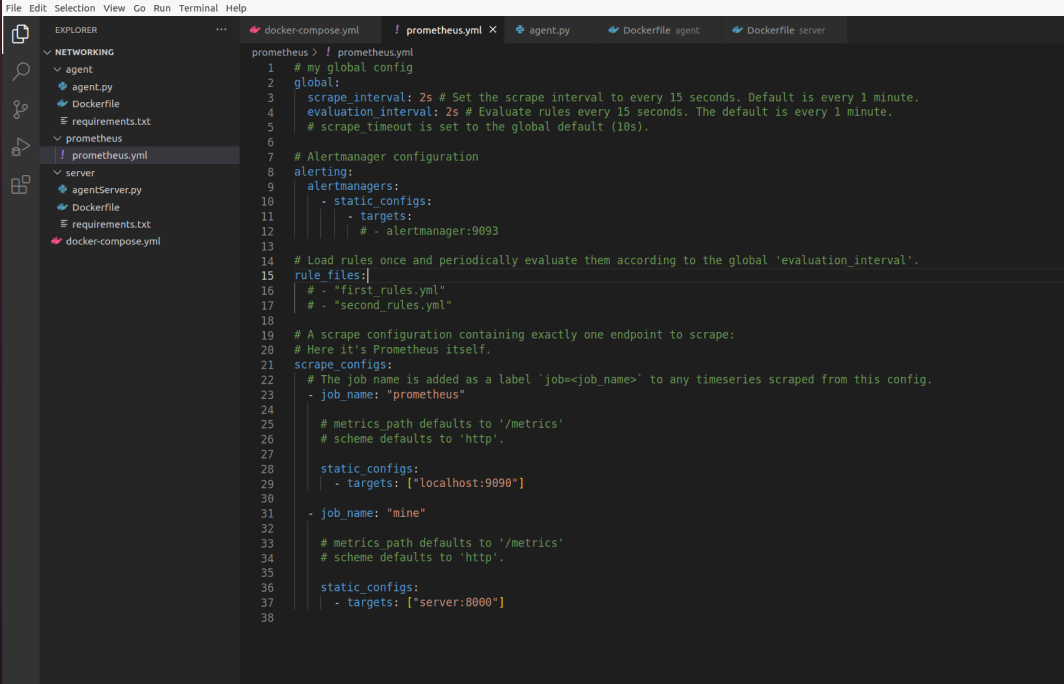
برای docker-compose.yml که در فولدر روت قرار دارد، این نوشته شده:



سرویس ها به این صورت:

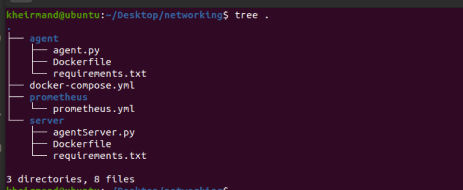
* سرویس server که همان agentServer است با dockerfile مشخص (image) و کامند python ./agentserver.py ران شده و در شبکه ی network قرار دارد.
* سرویس prometheus که image ان از docker-hub در هنگام up –d باید pull شود(اخرین stable version برای این کار انتخاب شده) و پورت 9090 از container که در ان web client مربوط به prometheus هست را به پورت 80 از pc فوروارد کرده ایم تا با زدن localhost:80/ یا همان localhost/ در browser بتوان مشاهده کرد و چون باید بتواند server container را ببیند و به پورت 8000 ان دسترسی داشته باشد باید در همان شبکه باشد (network). در ضمن برای اینکه target مارا بشناسد(همان server که در پورت 8000 خود metric ها را نشان میدهد) باید در فایل تنظیمات ان یکسری تغیرات داد و همچنین من باز هم اینجا زمان اپدیت شدن metric ها را با استفاده از پارامترهای scrape\_interval و evaluation\_interval در این فایل تغییر دادم.
* سرویس agent که Dockerfile مربوط به ان در فولدر agent قرار دارد و با کامند python agent.py ران میشود و در همان network قرار دارد تا بتواند سرویس server را ببیند. میتواند مانند ران کردن در پیسی و طبق screenshot های بالا چند تا از این سرویس را در docker-compose گذاشت چون سرور به صورت multi agent هم میتواند کار کند(که بالا تر نشان داده شد) ولی برای سادگی انجام نشده.

در پایان docker-compose هم شبکه ها ذکر شده که اینجا تنها یک شبکه با اسم network داریم. و در نهایت اینگونه شد:

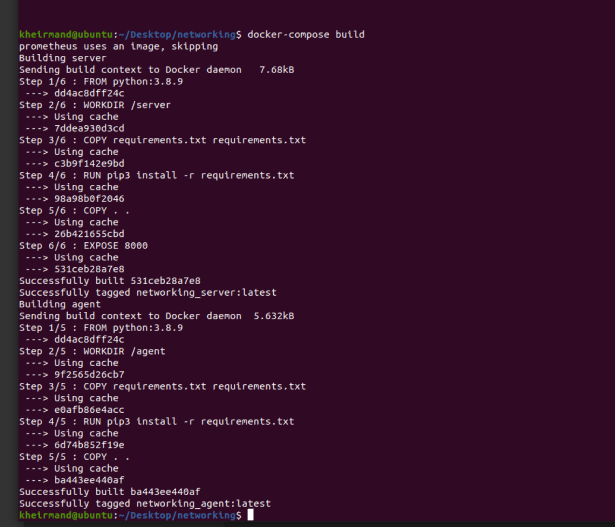


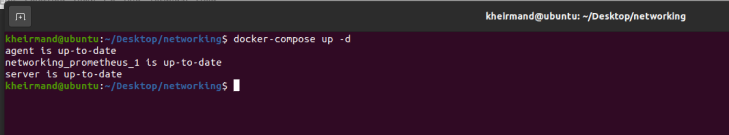
اگر دقت کنید میبینید که فرق این فایل و prometheus.yml مربوط به prometheus ران شده در ویندور(screenshot قبلی از فایل prometheus.yml) در خط اخر است. در انجا localhost:8000 و در اینجا server:8000 هست. چون localhost برای computer و ادرس 127.0.0.1 است ولی ما در شبکه ی network مربوط باید اسم سرویس را بزنیم که در اینجاserver است.

درختواره ی پروژه:

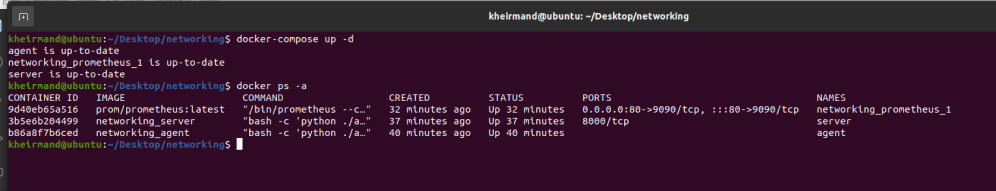


در نهایت build و up میکنیم:

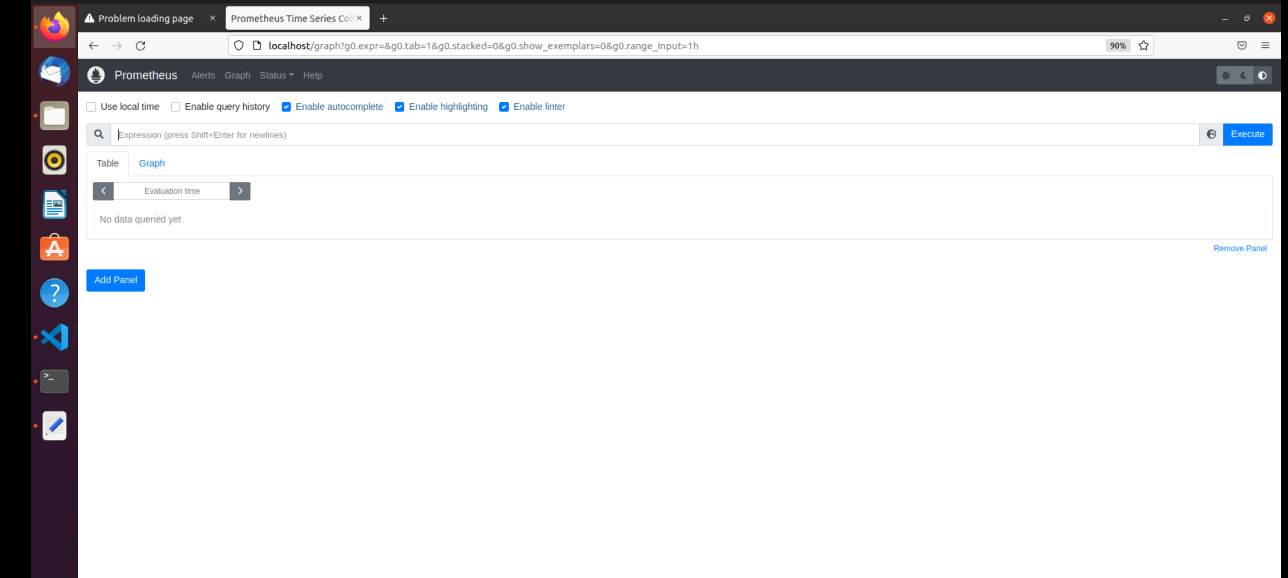




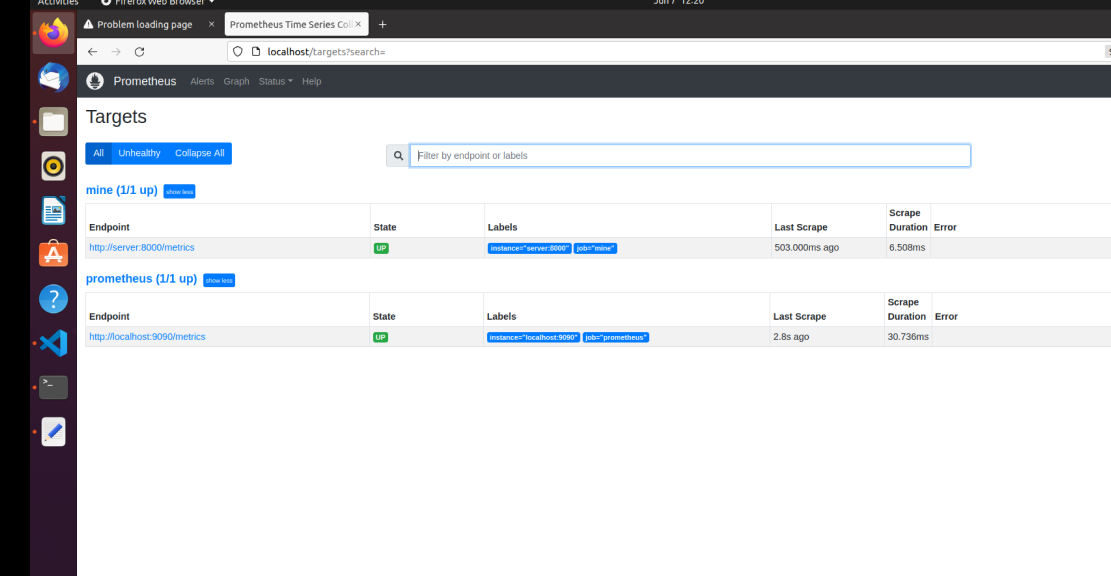
برای چک کردن اینکه کار میکنند باید ببینیم که container ها restart یا exit نشده باشند:



برای دیدن prometheus باید به localhost رفت:



برای چک کردن اینکه به هم وصل شده اند به status و سپس target میرویم:



همانطور که میبینید کار میکند.

