Prometheus ile Python Kullanarak İzleme

Tahmini Süre: 30 dakika

Prometheus ile İzleme laboratuvarına hoş geldiniz. Bu laboratuvar, node exporter ile simüle edilmiş örnek sunucuları izlemek için Prometheus kullanmaya aşina olmanızı sağlayacaktır. Prometheus'u, node_exporter'ın **metrics** uç noktalarını tarayarak yapılandırılan hedef node_exporter uygulamasını izlemek için kullanacaksınız. Laboratuvarı, bir Python Flask uygulamasını metrikler yaymak için nasıl enstrümante edeceğinizi öğrenerek ve bu uygulamayı Prometheus'un izleyebilmesi için dağıtarak tamamlayacaksınız.

Öğrenme Hedefleri

Bu egzersizi tamamladıktan sonra, şunları yapabilmelisiniz:

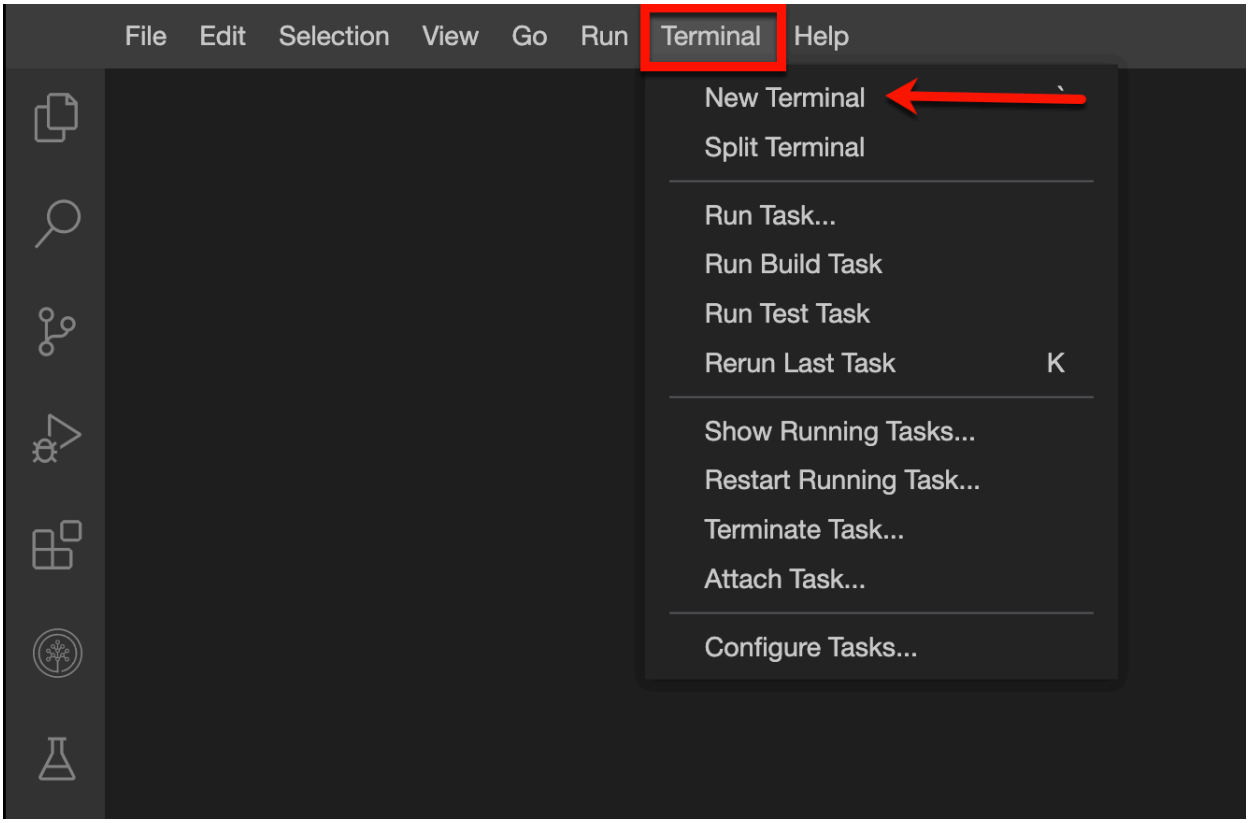
- Prometheus'un izlemesi için hedefleri yapılandırmak
- Hedef hakkında metrikleri almak için sorgular oluşturmak
- Hedeflerin durumunu belirlemek
- Hedefler hakkında bilgi edinmek ve bunu grafiklerle görselleştirmek
- Prometheus tarafından izlenmek üzere bir Python Flask uygulamasını enstrümante etmek

Ön Gereksinimler

Bu laboratuvar, Prometheus'u ve izlenebilecek sunucular gibi davranacak özel Node Exporters'ı çalıştırmak için Docker kullanır. Ön gereksinim olarak, `bitnami/prometheus:latest` imajını ve `bitnami/node-exporter` imajını Docker Hub'dan indireceksiniz. Bu imajları Prometheus'u çalıştırmak ve izlenecek üç node exporter örneği oluşturmak için kullanacaksınız.

Göreviniz

1. Bu laboratuvarı başlatmak için bir terminale ihtiyacınız olacak. Eğer bir terminal açık değilse, üst menüden bir tane açabilirsiniz. **Terminal** bölümüne gidin ve yeni bir terminal penceresi açmak için **Yeni Terminal** seçeneğini seçin.



2. Ardından, üç sunucunun izlenmesini simüle etmek için kullanacağınız `bitnami/node-exporter` imajını Docker Hub'dan indirmek için aşağıdaki `docker pull` komutunu kullanın.

```
docker pull bitnami/node-exporter:latest
```

Çıktınız aşağıdaki gibi görünmelidir:

```
theia@theiadocker-rofrano:/home/project$ docker pull bitnami/node-exporter:latest
latest: Pulling from bitnami/node-exporter
1d8866550bdd: Pull complete
8e2055ff5472: Pull complete
Digest: sha256:c306e2c62fa7fee7bf3b24b444dafe75423805fe1cb7aa52bc52af765767c6f1
Status: Downloaded newer image for bitnami/node-exporter:latest
docker.io/bitnami/node-exporter:latest
```

3. Sonra, terminalde aşağıdaki `docker pull` komutunu çalıştırarak Prometheus docker imajını laboratuvar ortamınıza indirin.

```
docker pull bitnami/prometheus:latest
```

Çıktınız aşağıdaki gibi görünmelidir:

```
theia@theiadocker-rofrano:/home/project$ docker pull bitnami/prometheus:latest
latest: Pulling from bitnami/prometheus
1d8866550bdd: Already exists
095e7c5c9312: Pull complete
Digest: sha256:58357c657791c5031ee16429fefef5e50c08e09f4fb50c1a1cce270d11d47901
Status: Downloaded newer image for bitnami/prometheus:latest
docker.io/bitnami/prometheus:latest
```

Artık laboratuvarı başlatmaya hazırsınız.

Adım 1: İlk node exporter'ı başlat

İlk olarak izlemeniz gereken bazı sunucu düğümleri olacaktır. `9100` portunu dinleyen ve sırasıyla `9101`, `9102` ve `9103` portlarına yönlendiren üç node exporter başlatacaksınız. Her bir düğümün ayrı ayrı başlatılması gerekecek.

Bu adımda, tüm node exporter'ların ve Prometheus'un iletişim kurması için bir Docker ağı oluşturacak ve sadece ilk düğümü, `9101` başlatacak ve doğru çalıştığından emin olacaksınız.

Göreviniz

1. Öncelikle, tüm Docker konteynerlerini çalıştıracağımız `monitor` adında bir ağ oluşturmak için aşağıdaki `docker network` komutunu çalıştırarak başlayın.

    ```
    docker network create monitor
    ```

2. Ardından, `monitor` ağı üzerinde, dışarıda `9101` portunu dinleyen ve içerde `9100` portuna yönlendiren bir node exporter örneği başlatmak için aşağıdaki `docker run` komutunu çalıştırın.

    ```
    docker run -d --name node-exporter1 -p 9101:9100 --network monitor bitnami/node-exporter:latest
    ```

    Bu, node-exporter'ın node_exporter1 adında bir örneğini başlatacaktır. Çıktı aşağıdaki gibi görünmelidir (not: konteyner kimliği her seferinde farklı olacaktır):

    ```
    theia@theiadocker-rofrano:/home/project$ docker run -d -p 9101:9100 --name node-exporter1 --network
    monitor bitnami/node-exporter:latest
    a6551f72fdd7dd0a96f28a488ff344a26d184c9a9b493e81a6a2f91c534ef8ee
    ```

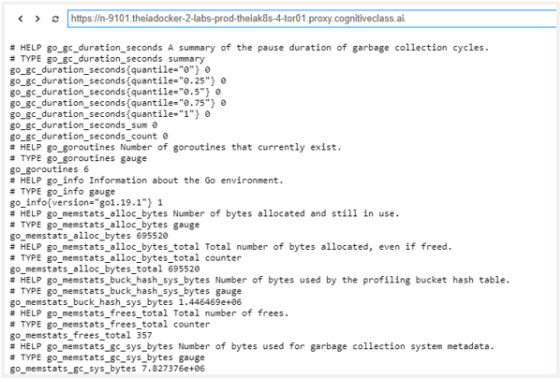
3. Ardından, örneğin çalışıp çalışmadığını kontrol etmek için `[Uygulamayı Başlat]` butonuna basarak uygulamayı `9101` portunda başlatın:

    Uygulamayı Başlat

4. Node Exporter sayfasının açıldığını ve **Metrics** bağlantısını gördüğünüzü kontrol edin. Bunlar, Prometheus örneğinin izleyeceği metriklerdir.

5. Son olarak, metrikleri görmek için **Metrics** bağlantısına tıklayın.

Adım 2: İki tane daha node exporter başlatın

Artık bir node exporter çalıştığına göre, Prometheus'un toplamda üç düğümü izlemesi için iki tane daha başlatabilirsiniz. Bunu, ilk node exporter'ı başlattığınız gibi yapacaksınız, tek farkla dış port numaralarını sırasıyla `9102` ve `9103` olarak değiştireceksiniz.

Göreviniz

1. Terminalde, iki tane daha node exporter örneğini başlatmak için aşağıdaki komutları çalıştırın.

    ```
    docker run -d --name node-exporter2 -p 9102:9100 --network monitor bitnami/node-exporter:latest
    ```

ve

```
docker run -d --name node-exporter3 -p 9103:9100 --network monitor bitnami/node-exporter:latest
```

2. Şimdi, tüm node exporter örneklerinin çalışıp çalışmadığını kontrol etmek için `docker ps` komutunu kullanın ve `grep` komutuyla `node-exporter` araması yapın.

```
docker ps | grep node-exporter
```

Sonuçlar

Her şey doğru bir şekilde başladıysa, `docker ps` komutundan aşağıdaki gibi bir çıktı görmelisiniz:

```
theia@theiadocker-rofrano:/home/project$ docker ps | grep node-exporter
f98cef022805   bitnami/node-exporter:latest   "/opt/bitnami/node-e…"   4 seconds ago    Up 2 seconds
    0.0.0.0:9103->9100/tcp, :::9103->9100/tcp   node-exporter3
48d41798e1be   bitnami/node-exporter:latest   "/opt/bitnami/node-e…"   18 seconds ago   Up 17 second
s   0.0.0.0:9102->9100/tcp, :::9102->9100/tcp   node-exporter2
a6551f72fdd7   bitnami/node-exporter:latest   "/opt/bitnami/node-e…"   4 minutes ago    Up 4 minutes
    0.0.0.0:9101->9100/tcp, :::9101->9100/tcp   node-exporter1
```

Artık Prometheus'u yapılandırmaya ve çalıştırmaya hazırsınız.

Adım 3: Prometheus'u Yapılandırma ve Çalıştırma

Prometheus'u başlatmadan önce, Prometheus'a hangi düğümleri izleyeceğini belirtmek için `prometheus.yml` adında bir yapılandırma dosyası oluşturmanız gerekir.

Bu adımda, `monitor` ağında sırasıyla `node-exporter1:9100`, `node-exporter2:9100` ve `node-exporter3:9100` üzerinde çalışan üç düğüm ihracısını izlemek için özel bir yapılandırma dosyası oluşturacaksınız. Ardından, kullanmak için yapılandırma dosyasını geçerek Prometheus'u başlatacaksınız.

Göreviniz

1. Öncelikle, mevcut dizinde **prometheus.yml** adında bir dosya oluşturmak için `touch` komutunu kullanın. Bu, Prometheus'u düğüm ihracı örneklerini izlemek için yapılandıracağınız dosyadır.

```
touch /home/project/prometheus.yml
```

2. Sonra, **Explorer**'dan **Project**'e gidin ve ardından dosyayı düzenlemek için **prometheus.yml**'i seçin veya aşağıdaki `[Open prometheus.yml in IDE]` butonuna basın:

Open **prometheus.yml** in IDE

3. Ardından, aşağıdaki yapılandırma içeriğini yaml dosyasına kopyalayıp yapıştırın ve kaydedin:

```
# benim global yapılandırmam
global:
  scrape_interval: 15s # Tarama aralığını her 15 saniyede bir olarak ayarlayın. Varsayılan her 1 dakikadır.
scrape_configs:
  - job_name: 'node'
    static_configs:
      - targets: ['node-exporter1:9100']
        labels:
          group: 'monitoring_node_ex1'
      - targets: ['node-exporter2:9100']
        labels:
          group: 'monitoring_node_ex2'
      - targets: ['node-exporter3:9100']
        labels:
          group: 'monitoring_node_ex3'
```

Dikkat edin ki, dışarıdan `9101`, `9102` ve `9103` portlarında düğüm ihracılarınızı erişirken, içten hepsi `9100` portunda dinliyor. Prometheus, bunlarla `monitor` ağında iletişim kuracaktır.

Bu dosyanın ne yaptığını inceleyin:

- Genel olarak, `scrape_interval`'i varsayılan 1 dakikadan 15 saniyeye ayarladınız. Bu, laboratuvar sırasında sonuçları daha hızlı görebilmemiz içindir, ancak 1 dakikalık aralık üretim kullanımı için daha iyidir.
- `scrape_config` bölümü, Prometheus'un izleyeceği tüm işleri içerir. Bu iş adlarının benzersiz olması gerekir. Şu anda `node` adında bir işiniz var. Daha sonra bir Python uygulamasını izlemek için başka bir iş ekleyeceğiz.
- Her işin içinde, hedefleri tanımladığınız ve kolay tanımlama ve analiz için etiketler tanımladığınız bir `static_configs` bölümü vardır. Bunlar Prometheus UI'de **Targets** sekmesinde görünecektir.
- Buraya girdiğiniz hedefler, her bir düğümde çalışan hizmetin temel URL'sine işaret eder. Prometheus, `/metrics` ekini ekleyecek ve verileri toplamak için bu uç noktayı çağıracaktır. (Örneğin, `node-exporter1:9100/metrics`)

Python uygulamanızı izlemek için kendi Prometheus dosyanızı oluşturma fırsatınız olacak.

4. Son olarak, aşağıdaki `docker run` komutunu terminalde çalıştırarak Prometheus izleyicisini başlatabilirsiniz. Yapılandırma dosyasını `-v` parametresi ile bir hacim montajı olarak geçin.

```
docker run -d --name prometheus -p 9090:9090 --network monitor \
-v $(pwd)/prometheus.yml:/opt/bitnami/prometheus/conf/prometheus.yml \
bitnami/prometheus:latest
```

Not: Bitnami'den gelen bu Dockerize edilmiş Prometheus dağıtımı, yapılandırma dosyasının `/opt/bitnami/prometheus/conf/prometheus.yml` dosyasında olmasını bekler; bu nedenle `prometheus.yml` dosyanızı bu konuma eşliyor oluyorsunuz. Diğer dağıtımlar farklı konumlarda arayabilir. Yapılandırma dosyasını nereye monte edeceğinizi kontrol etmek için her zaman belgeleri kontrol edin.

Sonuçlar

Docker'ın Prometheus'u arka planda başlattığını gösteren yalnızca Prometheus konteyner kimliğini görmelisiniz.

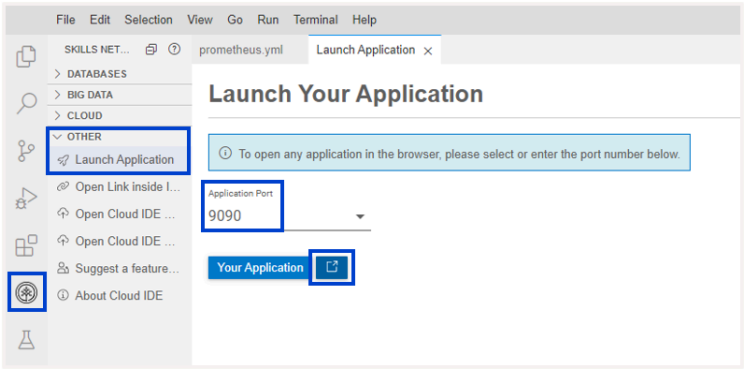
```
theia@theiadocker-rofrano:/home/project$ docker run -d --name prometheus -p 9090:9090
 --network monitor \
> -v $(pwd)/prometheus.yml:/opt/bitnami/prometheus/conf/prometheus.yml \
> bitnami/prometheus:latest
9db898b09d03e55b94817b4dc971eaea87e3b02f0ef1517a92273e13fc95941f
theia@theiadocker-rofrano:/home/project$
```

Artık bazı izlemeler yapmaya hazırsınız.

Adım 4: Prometheus UI'yi Açın

Bu adımda, Prometheus web UI'sini harici bir tarayıcı penceresinde başlatacak ve sorguları yürütmeye başlayacağınız sayfaya gideceksiniz.

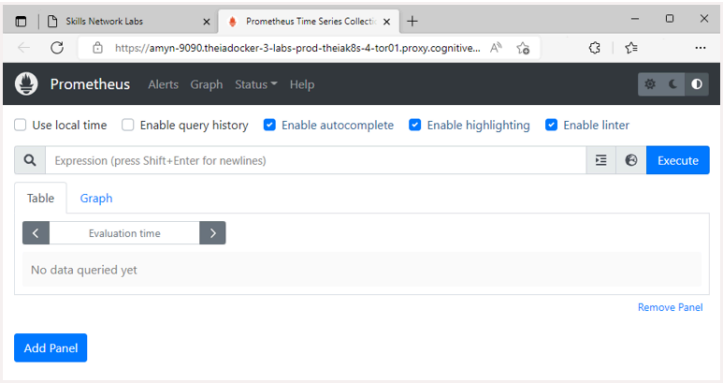
1. Prometheus web UI'sini açmak için **Skills Network Toolbox**'a tıklayın. **Diğer** altında **Uygulamayı Başlat**'ı seçin, **Uygulama Portu** kısmına **9090** port numarasını girin ve ardından URL'yi başlatma butonuna tıklayın.



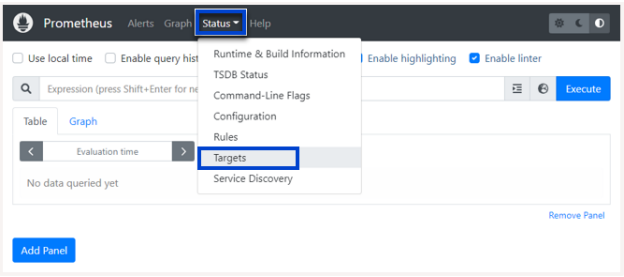
Veya aşağıdaki **Prometheus'u Başlat** butonuna tıklayarak harici bir tarayıcıda başlatın.

Prometheus'u Başlat

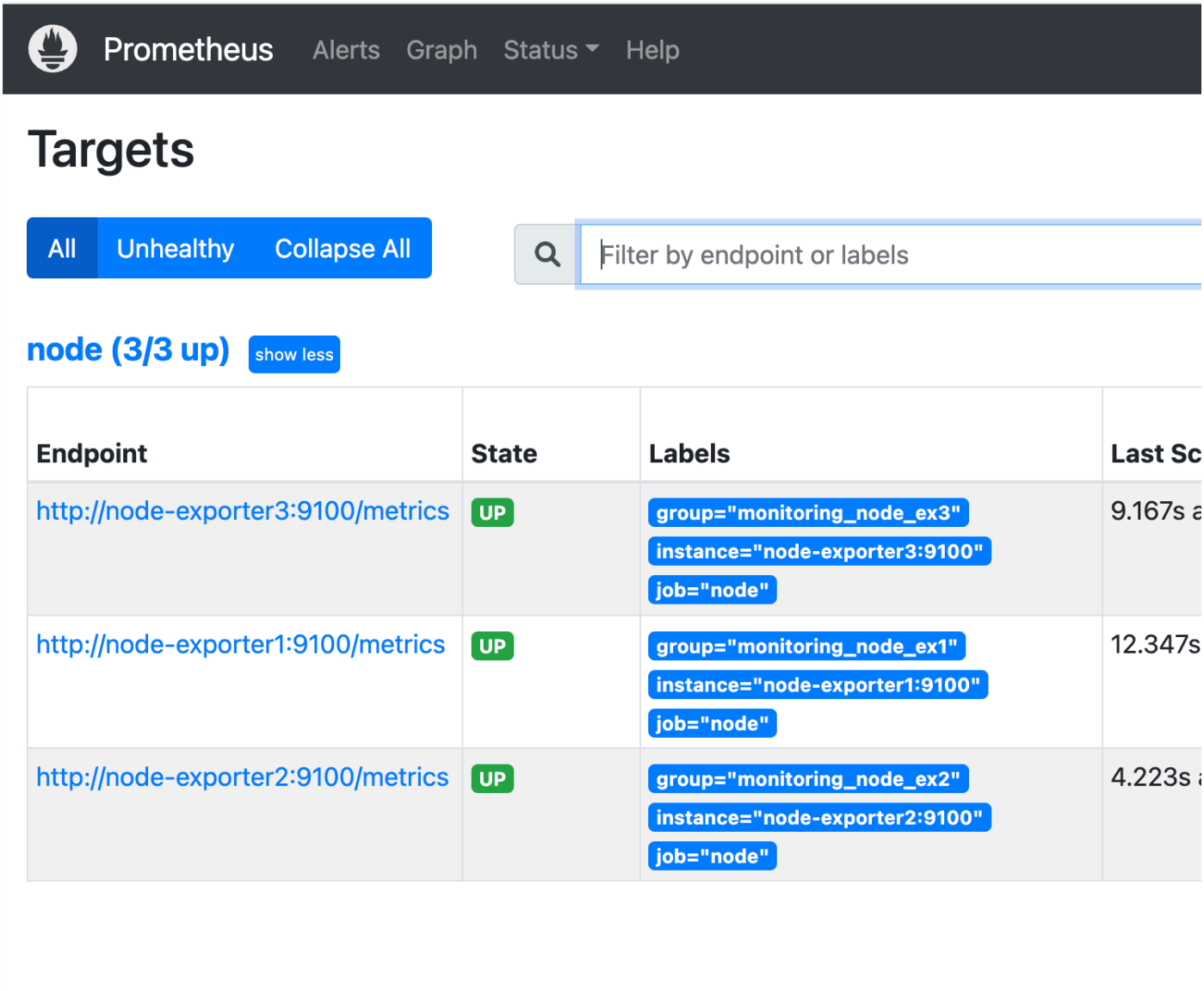
2. Prometheus uygulama UI'si açılır.



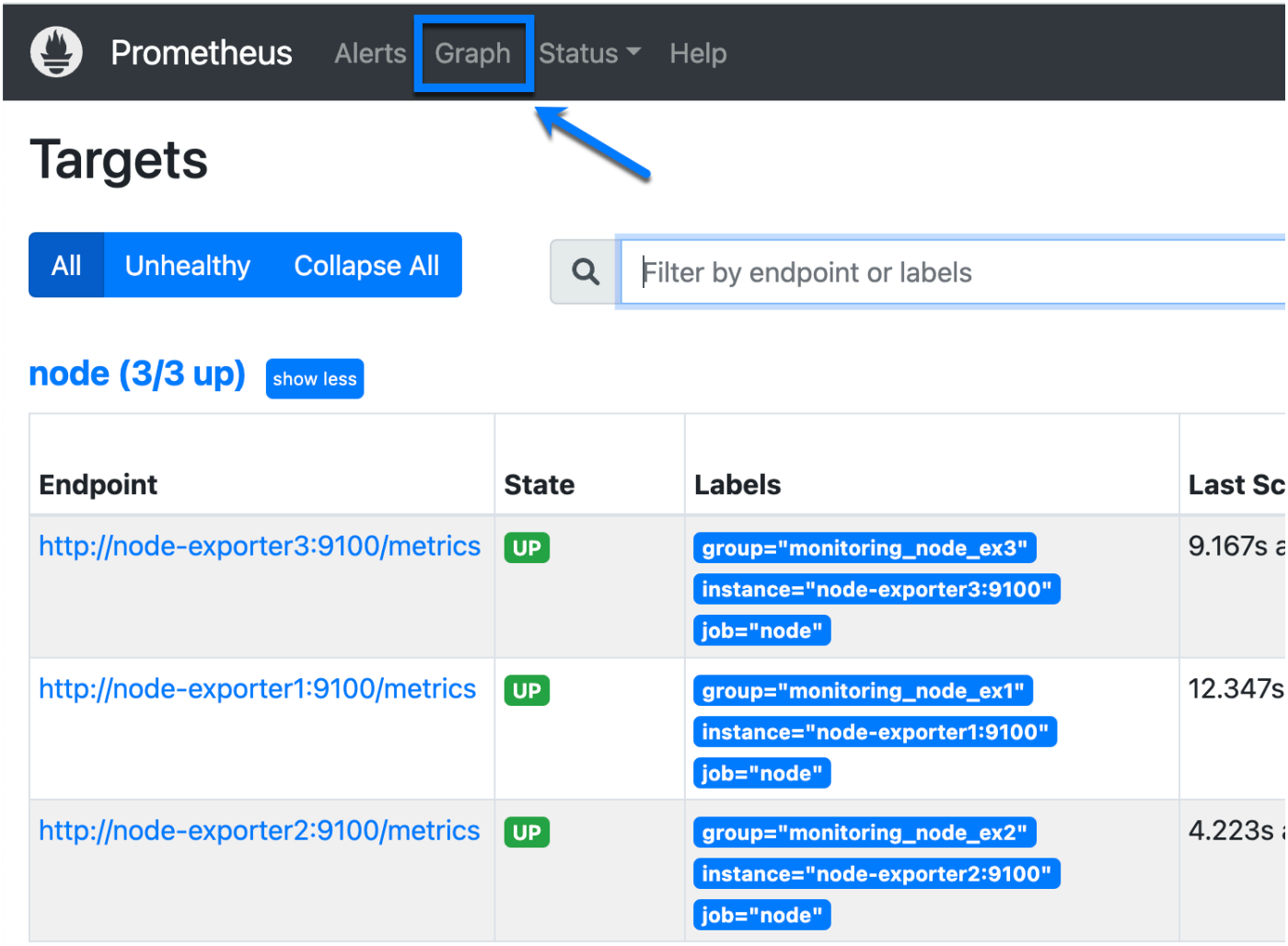
3. Ardından, Prometheus uygulamasında menüden **Durum**'a tıklayın ve izlenen hedefleri görmek için **Hedefler**'i seçin.



4. Üç node exporter'ın durumunu görüntüleyin.



5. Ana sayfaya dönmek için **Grafik**'e tıklayın.



Artık sorguları yürütmeye hazırsınız.

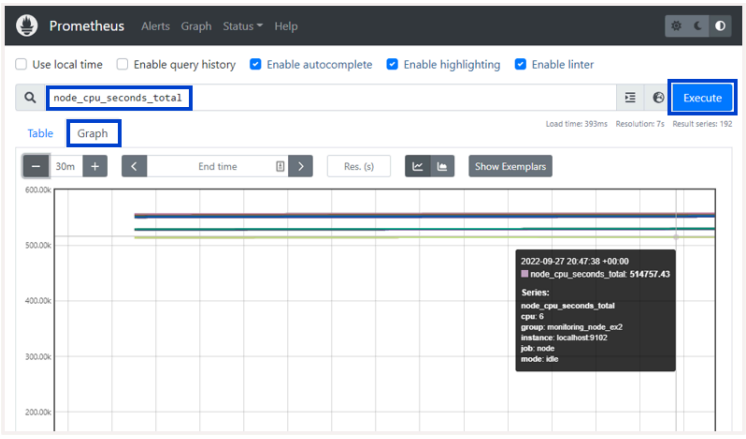
Adım 5: İlk sorgunuzu çalıştırın

Artık ilk sorgunuzu çalıştırmaya hazırsınız. Çalıştıracağınız ilk sorgu, düğümlerin toplam CPU saniyelerini sorgulayacaktır. Bu, resimdeki gibi bir grafik gösterecektir. Her bir örneğin ayrıntılarını, o örneğin üzerine fare ile gelerek gözlemleyebilirsiniz.

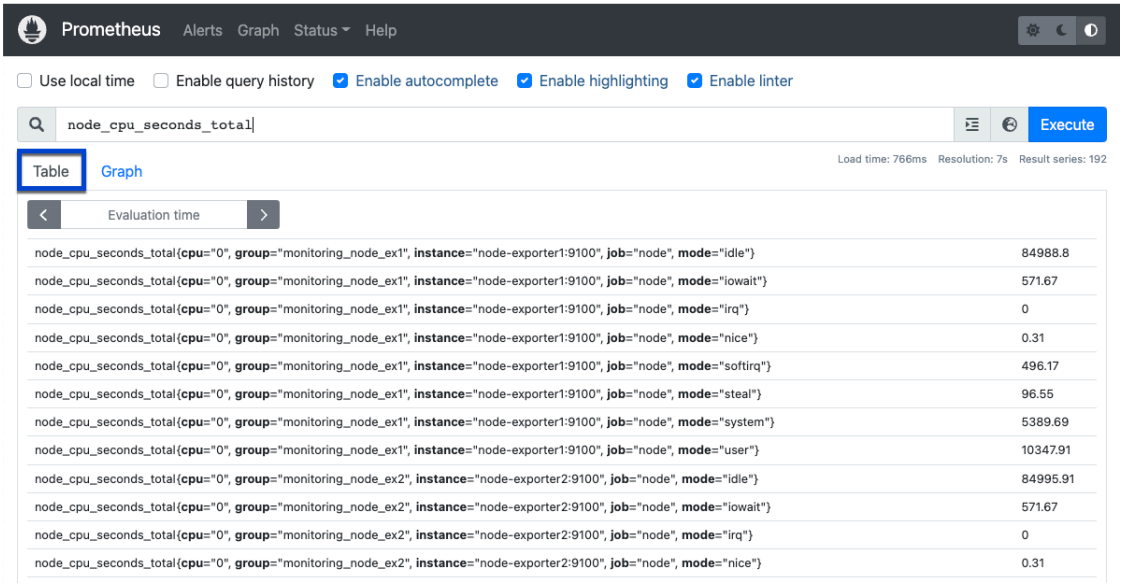
Göreviniz

1. **Grafik** sekmesinde olduğunuzdan emin olun ve ardından aşağıdaki sorguyu *kopyalayıp yapıştırın* ve sağdaki mavi **Çalıştır** butonuna basın veya çalıştırmak için klavyenizde `return` tuşuna basın. Bu, resimdeki gibi bir grafik gösterecektir. Her bir örneğin ayrıntılarını, o örneğin üzerine fare ile gelerek gözlemleyebilirsiniz.

   ```
   node_cpu_seconds_total
   ```

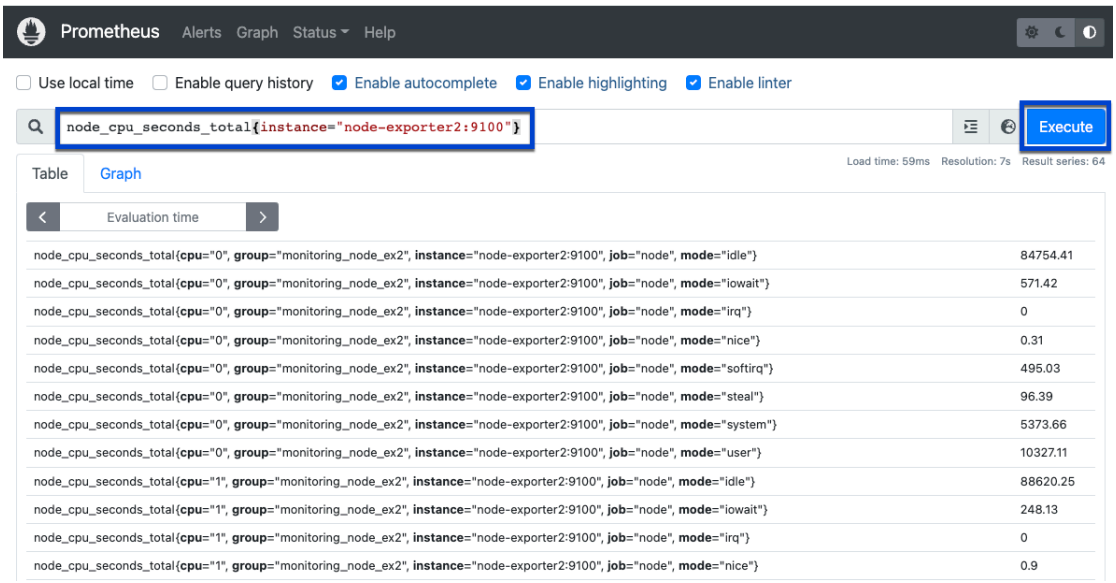


2. Ardından, tüm hedeflerin CPU saniyelerini tablo formatında görmek için **Tablo**ya tıklayın.



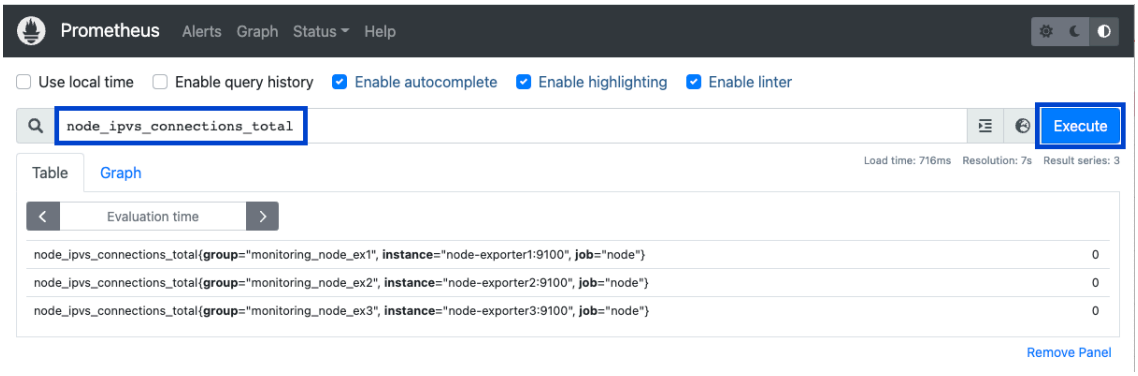
3. Şimdi, yalnızca bir örnek olan `node-exporter2` için ayrıntıları almak üzere sorguyu filtreleyin ve aşağıdaki sorguyu kullanın.

```
node_cpu_seconds_total{instance="node-exporter2:9100"}
```



4. Son olarak, her bir düğümün sahip olduğu bağlantılar için bu sorguyu kullanarak sorgulama yapın.

```
node_ipvs_connections_total
```



Adım 6: Durdur ve gözlemle

Bu adımda, bir node exporter örneğini durduracağız ve bunun Prometheus konsolunda nasıl yansıtıldığını göreceğiz.

Göreviniz

1. Aşağıdaki `docker stop` komutunu çalıştırarak node-exporter1 örneğini durdurun ve ardından Prometheus'un çalıştığı eski terminale geri dönün.

```
docker stop node-exporter1
```

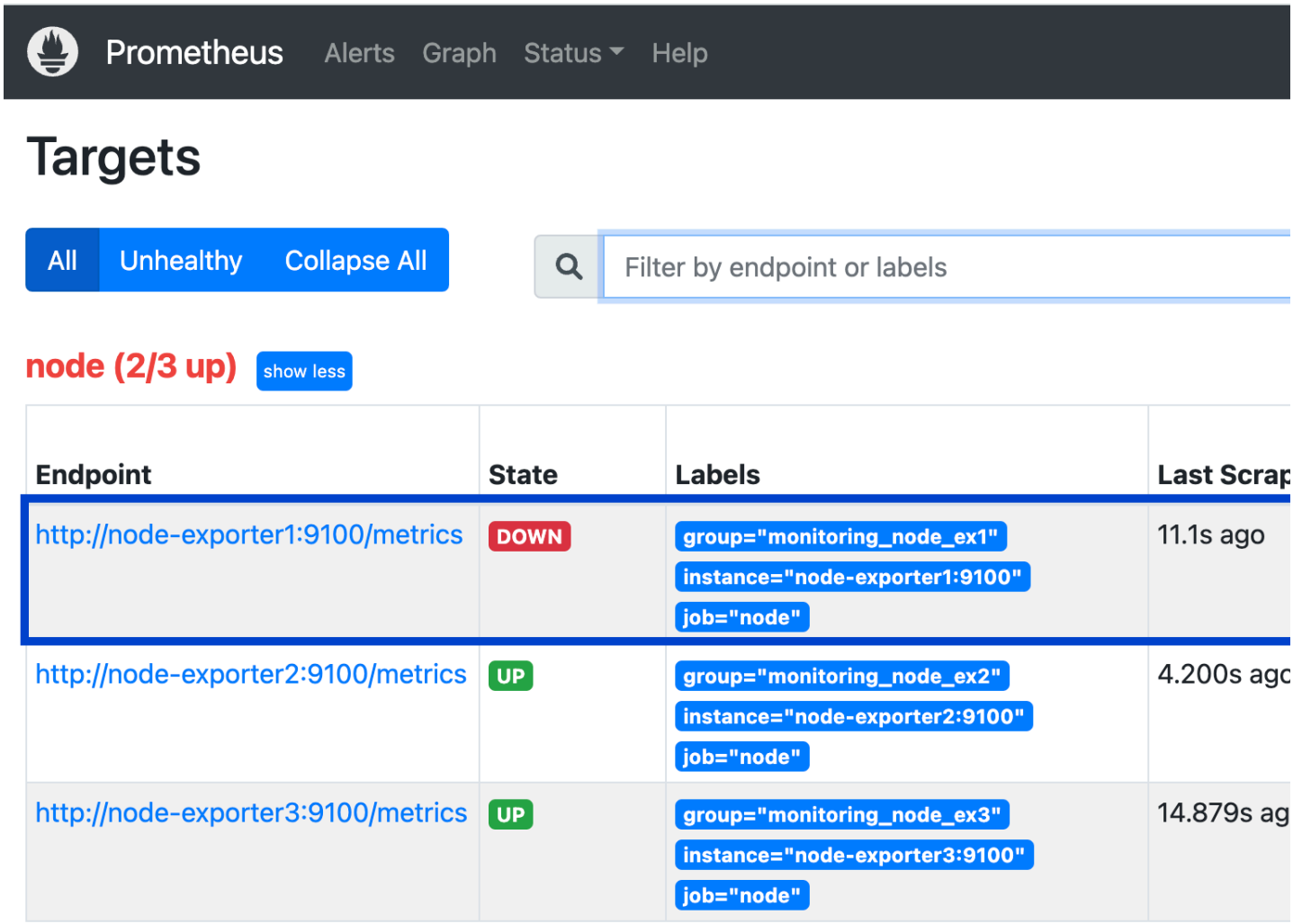
2. Şimdi tarayıcınızdaki Prometheus arayüzüne geri dönün ve **Durum** -> **Hedefler** menü öğesini seçerek hedefleri kontrol edin.

Prometheus Hedeflerini Başlat

Sonuçlar

Artık izlenen node exporter'lardan birinin kapalı olduğunu görmelisiniz. Düğümler aynı sırada görüntülenmeyebilir, ancak durdurduğunuz düğüm `node-exporter1` olmalıdır.

Not: Prometheus'u her 15 saniyede bir veri çekmesi için yapılandırdınız, bu nedenle durum değişikliğini görmek için bu kadar beklemeniz ve tarayıcınızda yenilemeye basmanız gerekebilir.



Adım 7: Uygulamanızı Etkinleştirin

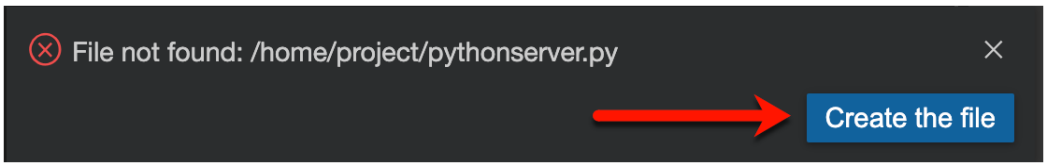
Node exporter'ları izlemek bir gösterim için iyidir, ancak siz bir yazılım mühendisisiniz. Uygulamalarınızın Prometheus tarafından izlenebilmesi için nasıl etkinleştirileceğini bilmeniz gerekiyor. Burada sihir yok. Metrikler bir yerden çıkmaz. Prometheus'un uygulamanızı izleyebilmesi için, uygulamanızı `/metrics` adlı bir uç noktada metrikleri yayacak şekilde enstrümante etmelisiniz.

Neyse ki, bunu sizin için yapacak Prometheus için **Prometheus Flask exporter** adlı bir Python paketi var. Bu adımda, basit bir Python Flask uygulaması oluşturacak ve izlenebilmesi için bir metrik uç noktası etkinleştireceksiniz.

Göreviniz

Aşağıda, üç uç noktası olan bir Python Flask sunucusu için bir kod bulunmaktadır: `/`, `/home` ve `/contact`. Bu kod, Prometheus'un izleyebilmesi için metrikleri oluşturmak üzere `prometheus_flask_exporter` paketini kullanmaktadır.

1. Öncelikle, `/home/project` klasöründe `pythonserver.py` adında bir dosya oluşturun. Aşağıdaki düğmeye basın ve istenildiğinde **Dosyayı oluştur** yanıtını verin.



Open **pythonserver.py** in IDE

2. Ardından, aşağıdaki kod içeriğini yapıştırın:

```
from prometheus_flask_exporter import PrometheusMetrics
from flask import Flask
app = Flask(__name__)
```

```
metrics = PrometheusMetrics.for_app_factory()
metrics.init_app(app)
@app.route('/')
def root():
    return 'Root'tan merhaba!'
@app.route('/home')
def home():
    return 'Ev'den merhaba!'
@app.route('/contact')
def contact():
    return 'Bize ulaşın!'
if __name__ == '__main__':
    app.run(host="0.0.0.0", port=8080)
```

Sadece `prometheus_flask_exporter` paketinden `PrometheusMetrics` sınıfını içe aktarmanız ve `PrometheusMetrics.for_app_factory()`'i `metrics` olarak başlatmak için iki satır kod eklemeniz gerektiğini unutmayın. Hepsi bu kadar! Toplamda üç satır kod ile Prometheus desteğine sahip oldunuz!

3. Sonraki adımda, bu kodu Prometheus ile aynı docker ağında dağıtmanız gerekiyor. Bunu yapmak için, `/home/project` klasöründe `Dockerfile` adında bir dosya oluşturun:

   Open **Dockerfile** in IDE

4. `Dockerfile` içine aşağıdaki içerikleri yapıştırın ve kaydedin:

```
FROM python:3.9-slim
RUN pip install Flask prometheus-flask-exporter
WORKDIR /app
COPY pythonserver.py .
EXPOSE 8080
CMD ["python", "pythonserver.py"]
```

5. Şimdi, hizmet için bir Docker görüntüsü oluşturmak üzere `docker build` komutunu kullanın (*Not: Docker build komutundan gelen herhangi bir kırmızı çıktıyı güvenle göz ardı edebilirsiniz. Bu, pip'in root olarak çalıştırıldığına dair bir uyarıdır*):

```
docker build -t pythonserver .
```

6. Son olarak, Prometheus'un erişebilmesi için `monitor` ağında 8080 portunu açarak pythonserver Docker konteynerini çalıştırın:

```
docker run -d --name pythonserver -p 8081:8080 --network monitor pythonserver
```

7. (Opsiyonel) Python sunucusunun çalıştığını kontrol etmek için `Python Sunucusu UI'sini Başlat` düğmesine basın:

   Python Sunucusu UI'sini Başlat

Artık yeni uygulamanızı Prometheus'a eklemeye hazırsınız.

Adım 8: Prometheus'u Yeniden Yapılandırma

Artık uygulamanız çalıştığına göre, Prometheus'u yeniden yapılandırma zamanı geldi; böylece yeni `pythonserver` düğümünü izleyebilsin. Bunu, Python sunucusunu `prometheus.yml` dosyanızda bir hedef olarak ekleyerek yapabilirsiniz.

Göreviniz

1. Öncelikle, `prometheus.yml` dosyasını açın:

   Open **prometheus.yml** in IDE

2. Ardından, `8080` portunda dinleyen `pythonserver` hizmetini izlemek için yeni bir iş oluşturun. Önceki işi örnek olarak kullanın.

▼ İpucu için buraya tıklayın
Yeni bir iş adı oluşturun ve prometheus.yml dosyasındaki hedefi `pythonserver` sunucusunun URL'sine ve portuna işaret edecek şekilde değiştirin:

```
- job_name: {make up a job name here}
  static_configs:
    - targets: [{place the target to monitor here}]
      labels:
        group: {make up a group name here}
```

▼ Çözüm için buraya tıklayın
Farklı bir `job_name` veya `group` oluşturmuş olabilirsiniz, ancak `targets` aşağıdaki hedefle eşleşmelidir:

```
- job_name: 'monitorPythonserver'
  static_configs:
    - targets: ['pythonserver:8080']
      labels:
        group: 'monitoring_python'
```

3. Tam `prometheus.yml` dosyanızın bu dosyaya benzer göründüğünden emin olun:

▼ Çalışmanızı kontrol etmek için buraya tıklayın

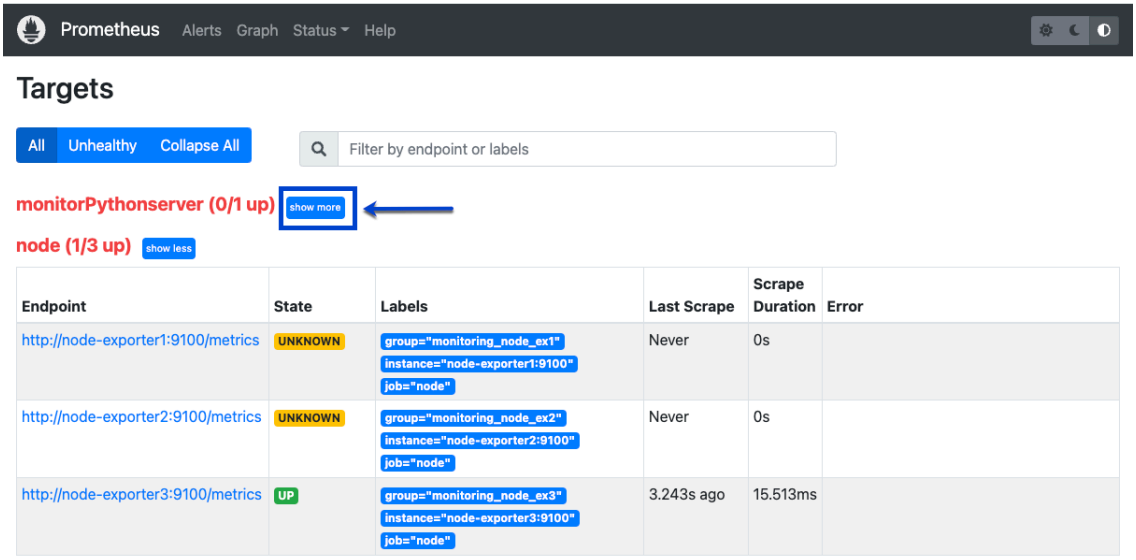
```
# my global config
global:
  scrape_interval: 15s # Set the scrape interval to every 15 seconds. The default is every 1 minute.
scrape_configs:
  - job_name: 'monitorPythonserver'
    static_configs:
      - targets: ['pythonserver:8080']
        labels:
          group: 'monitoring_python'
  - job_name: 'node'
    static_configs:
      - targets: ['node-exporter1:9100']
        labels:
          group: 'monitoring_node_ex1'
      - targets: ['node-exporter2:9100']
        labels:
          group: 'monitoring_node_ex2'
      - targets: ['node-exporter3:9100']
        labels:
          group: 'monitoring_node_ex3'
```

4. Yeni yapılandırma değişikliklerini almak için prometheus sunucusunu yeniden başlatın:

```
docker restart prometheus
```

5. Yeni Hedefleri görmek için Prometheus UI'sını kontrol edin.

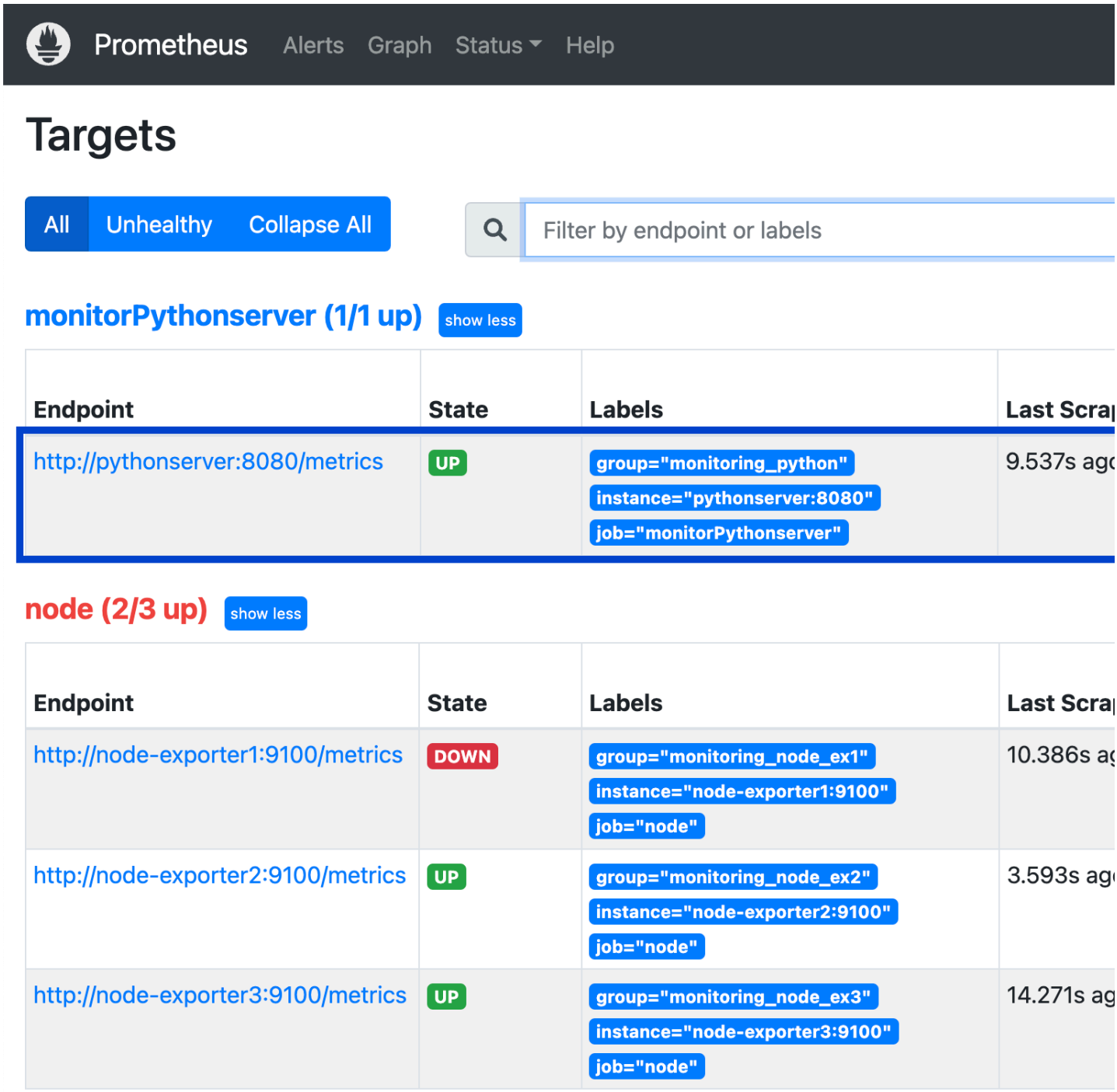
   Not: Aşağıda gösterildiği gibi `monitorPythonserver` yanındaki "daha fazla göster" butonuna tıklamanız gerekebilir:



Prometheus Hedeflerini Başlat

Sonuçlar

Her şey yolunda gittiyse, Prometheus hedeflerini açtığınızda Python sunucunuzun durumunu aşağıdaki resimdeki gibi göreceksiniz.



Adım 9: Uygulamanızı İzleyin

İzleme sonuçlarını görebilmek için bazı ağ trafiği oluşturmanız gerekiyor.

1. Önceki görevde oluşturduğunuz Python sunucusunun üç uç noktasına birden fazla istek gönderin ve bu çağrıları Prometheus'ta gözlemleyin.

```
curl localhost:8081
curl localhost:8081/home
curl localhost:8081/contact
```

Gerçek ağ trafiğini simüle etmek için bunları birden fazla kez çalıştırmaktan çekinmeyin.

2. Aşağıdaki metrikleri sorgulamak için Prometheus UI'sını kullanın.

- `flask_http_request_duration_seconds_bucket`
- `flask_http_request_total`
- `process_virtual_memory_bytes`

Uygulamanızın hangi diğer metrikleri yayımladığını merak ediyorsanız, uygulamanızın yayımladığı tüm metrikleri Prometheus'un yaptığı gibi `/metrics` uç noktasını açarak görebilirsiniz. Diğer metriklere karşı sorgular çalıştırarak denemeler yapmaktan çekinmeyin:

PythonServer Metrikleri Başlat

https://rofrano-8081.theiadocker-3-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/metrics

```
# HELP python_gc_objects_collected_total Objects collected during gc
# TYPE python_gc_objects_collected_total counter
python_gc_objects_collected_total{generation="0"} 329.0
python_gc_objects_collected_total{generation="1"} 76.0
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"} 76.0
python_gc_collections_total{generation="1"} 6.0
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="14",version="3.9.14"} 1.0
# HELP process_virtual_memory_bytes Virtual memory size in bytes.
# TYPE process_virtual_memory_bytes gauge
process_virtual_memory_bytes 1.09953024e+08
# HELP process_resident_memory_bytes Resident memory size in bytes.
# TYPE process_resident_memory_bytes gauge
process_resident_memory_bytes 2.9843456e+07
```

Sonuç

Tebrikler! **Prometheus ile İzleme** laboratuvarını tamamladınız. Artık uygulamalarınızı düzgün çalıştığından emin olmak için izlemeye hazır bir konumdasınız.

Bu laboratuvarda üç adet node exporter dağıttınız ve bunları izlemek için Prometheus metriklerini kullandınız. Ayrıca bir Python Flask uygulamasını nasıl enstrümante edeceğinizi, Docker'da nasıl dağıtacağınızı ve bu yeni uygulamayı izlemeye başlamak için Prometheus yapılandırmasını nasıl değiştireceğinizi öğrendiniz.

Sonraki Adımlar

Bir sonraki zorluk, uygulamalarınızı izlemek için geliştirme ortamınızda Prometheus'u kurmaktır. Python Flask uygulamalarınızdan birini enstrümante etmek için **Prometheus Flask exporter**'ını kullanın. Ardından, bu laboratuvarda öğrendiğiniz bazı sorguları kullanarak uygulamanızın sağlık ve performansını kontrol edin.

Author(s)

Lavanya T S
John J. Rofrano

Other Contributor(s)

Pallavi Rai