**Task 1: Create the Backend Application using Flask**

**Step 1: Project Setup**

1. Create a project directory:
2. mkdir app-2024mt03541
3. cd app-2024mt03541
4. Create a Python file:
5. touch main.py

**Step 2: Install Dependencies**

Install Flask and Uvicorn (ASGI server):

pip install flask uvicorn python-dotenv **File: requirements.txt**  
  
*flask>=3.1.0*

*python-dotenv>=1.0*

*psutil>=7.1.3*

*prometheus\_client>=0.23.1*

*python -m pip install -r requirements.txt*

or

*pip install -r requirements.txt*

**Step 3: Write main.py**

*from flask import Flask, jsonify, Response*

*import os*

*import time*

*import psutil*

*import socket*

*from prometheus\_client import Counter, Gauge, Histogram, generate\_latest, CONTENT\_TYPE\_LATEST*

*# Initialize Flask app*

*app = Flask(\_\_name\_\_)*

*# Environment variables*

*APP\_VERSION = os.getenv('APP\_VERSION', '1.0')*

*APP\_TITLE = os.getenv('APP\_TITLE', 'DevOps for Cloud Assignment')*

*# Identify pod name (Kubernetes injects HOSTNAME automatically)*

*POD\_NAME = os.getenv('HOSTNAME', socket.gethostname())*

*# Prometheus Metrics (with labels)*

*REQUEST\_COUNT = Counter(*

*"request\_count\_total",*

*"Total number of requests handled by each pod",*

*["pod"]*

*)*

*REQUEST\_LATENCY = Histogram(*

*"request\_latency\_seconds",*

*"Request latency in seconds",*

*["pod"]*

*)*

*CPU\_UTILIZATION = Gauge(*

*"cpu\_utilization\_percent",*

*"Current CPU utilization percentage",*

*["pod"]*

*)*

*MEMORY\_USAGE = Gauge(*

*"memory\_usage\_mb",*

*"Current memory usage in MB",*

*["pod"]*

*)*

*# Endpoint: /get\_info*

*@app.route('/get\_info', methods=['GET'])*

*def get\_info():*

*start\_time = time.time()*

*try:*

*REQUEST\_COUNT.labels(pod=POD\_NAME).inc()*

*cpu\_usage = psutil.cpu\_percent(interval=None)*

*memory\_usage = psutil.virtual\_memory().used / (1024 \* 1024)*

*# Update gauges*

*CPU\_UTILIZATION.labels(pod=POD\_NAME).set(cpu\_usage)*

*MEMORY\_USAGE.labels(pod=POD\_NAME).set(memory\_usage)*

*response = jsonify({*

*"APP\_TITLE": APP\_TITLE,*

*"APP\_VERSION": APP\_VERSION,*

*"POD\_NAME": POD\_NAME,*

*"CPU\_UTILIZATION": cpu\_usage,*

*"MEMORY\_USAGE\_MB": round(memory\_usage, 2)*

*})*

*response.status\_code = 200*

*except Exception as e:*

*response = jsonify({"error": str(e)})*

*response.status\_code = 500*

*finally:*

*latency = time.time() - start\_time*

*REQUEST\_LATENCY.labels(pod=POD\_NAME).observe(latency)*

*return response*

*# Prometheus metrics endpoint*

*@app.route('/metrics', methods=['GET'])*

*def metrics():*

*return Response(generate\_latest(), mimetype=CONTENT\_TYPE\_LATEST)*

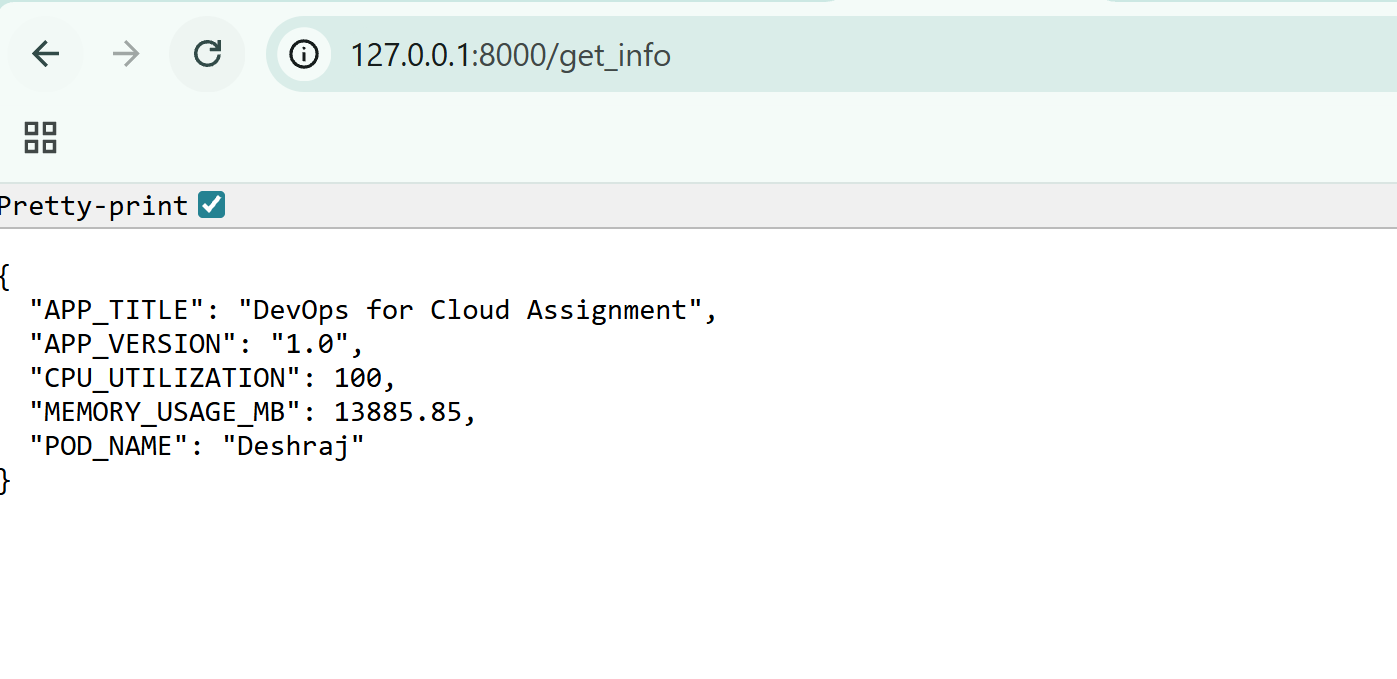
*if \_\_name\_\_ == '\_\_main\_\_':*

*app.run(host='0.0.0.0', port=8000)*

**Step 4: Run Flask Application**

python main.py

Open browser at *http://localhost:8000/get\_info*

**Screenshots:** 

**Task 2: Dockerize the Backend Application**

**Step 1: Dockerfile**

*# Use official Python image*

*FROM python:3.11-slim*

*# Set working directory*

*WORKDIR /app*

*# Copy project files*

*COPY main.py .*

*# Install Flask & additional dependencies*

*COPY requirements.txt .*

*RUN pip install -r requirements.txt*

*# Expose port 8000*

*EXPOSE 8000*

*# Run the app*

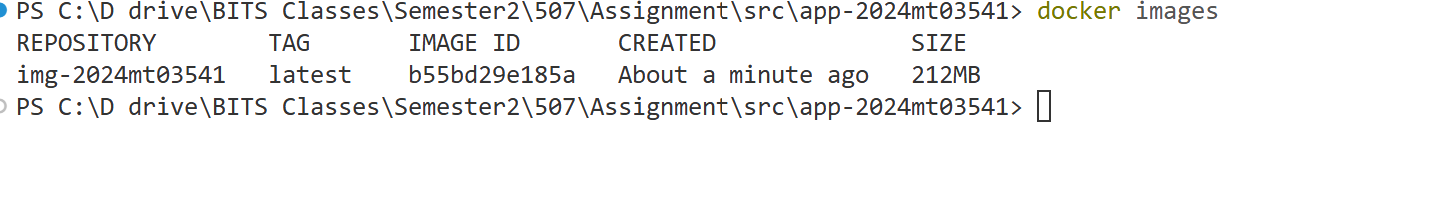
*CMD ["python", "main.py"]*

**Step 2: Build Docker Image**

*docker build -t <image\_name> .*Build Image *docker build -t img-2024mt03541 .*Remove image if required *docker rmi img-2024mt03541  
  
*

**Step 3: Verify Image**

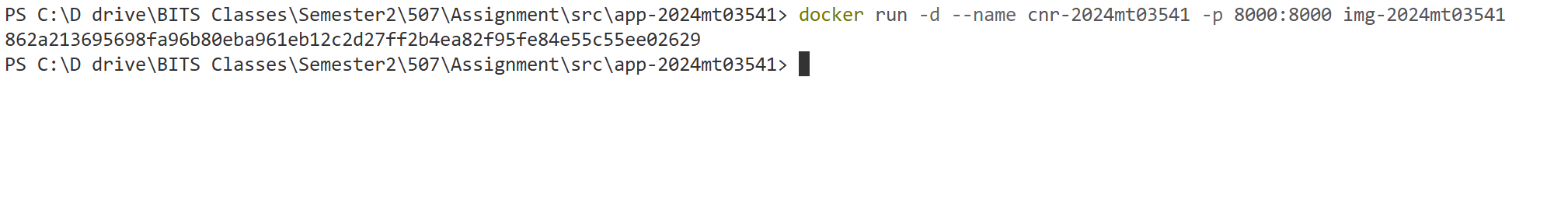
*docker images*



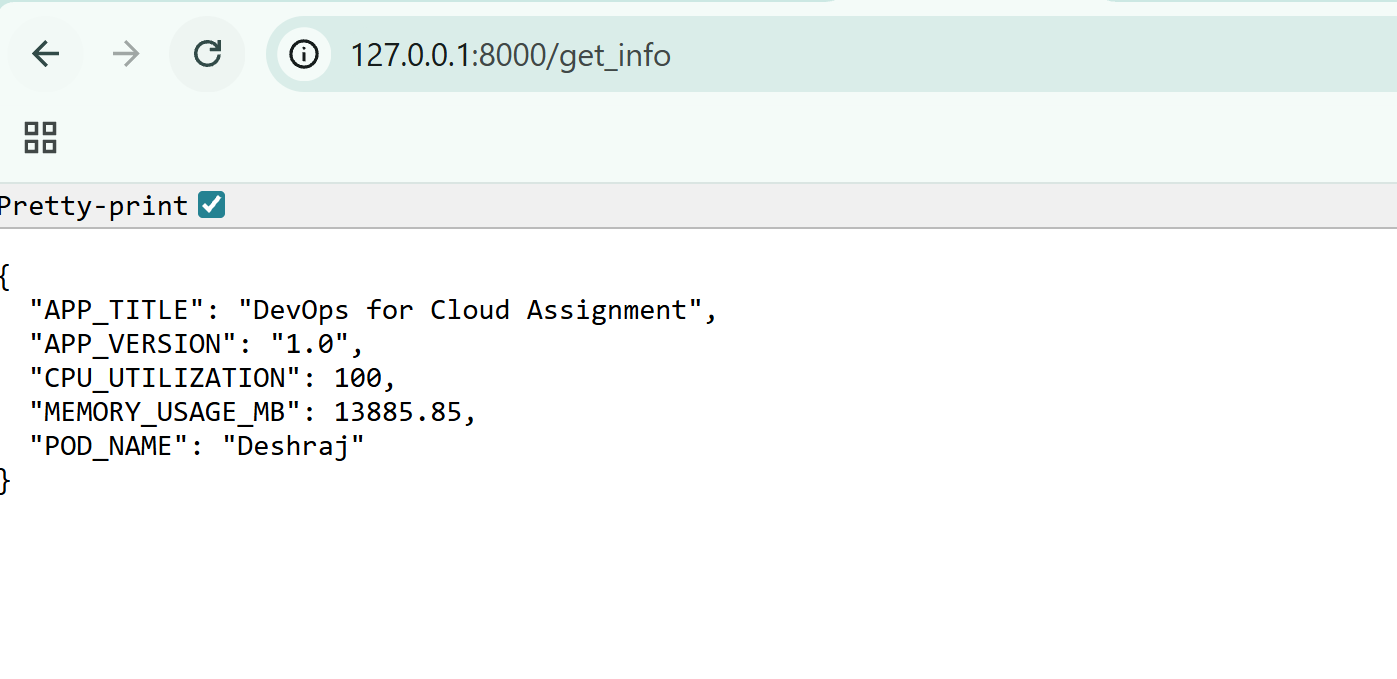
**Task 3: Run the Docker Container**

**Step 1: Run Container**

*docker run -d --name cnr-2024mt03541 -p 8000:8000 img-2024mt03541*Stop Container *docker stop* cnr-*2024mt03541*Remove container if required *docker rm cnr-2024mt03541*

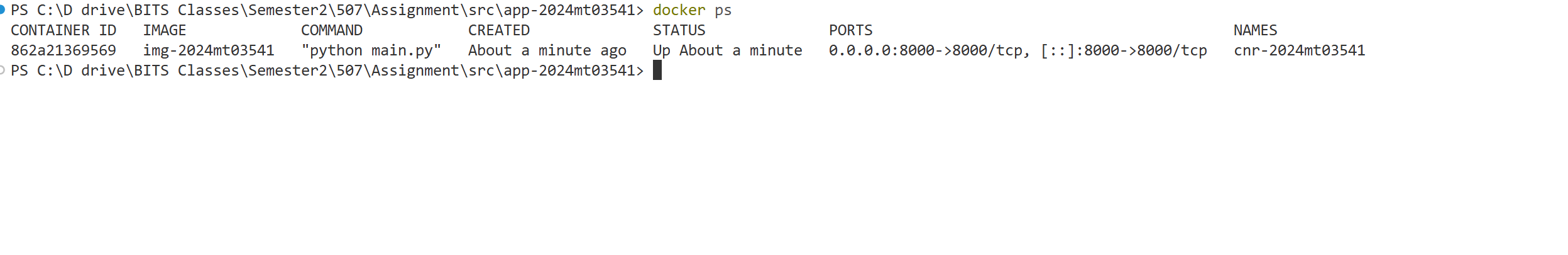


**Step 2: Verify**

Access [*http://localhost:8000/get\_info*](http://localhost:8000/get_info)  
  


Check container status:

*docker ps*



**Task 4: Deploy the Docker Image to Kubernetes**

**Step 1: ConfigMap**

***config-2024mt03541.yaml***

*apiVersion: v1*

*kind: ConfigMap*

*metadata:*

*name: config-2024mt03541*

*data:*

*APP\_VERSION: "1.0"*

*APP\_TITLE: "Devops for Cloud Assignment"*

**Step 2: Deployment**

**dep-2024mt03541.yaml**

*apiVersion: apps/v1*

*kind: Deployment*

*metadata:*

*name: dep-2024mt03541*

*spec:*

*replicas: 2*

*selector:*

*matchLabels:*

*app: flask-app*

*template:*

*metadata:*

*labels:*

*app: flask-app*

*spec:*

*containers:*

*- name: flask-container*

*image: img-2024mt03541*

*ports:*

*- containerPort: 8000*

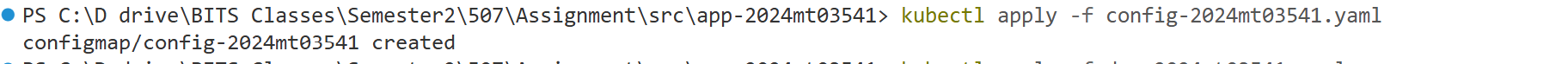
*envFrom:*

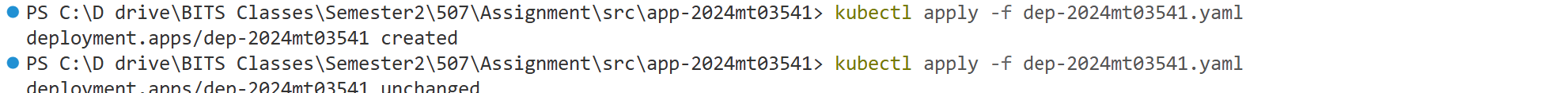
*- configMapRef:*

*name: config-2024mt03541*

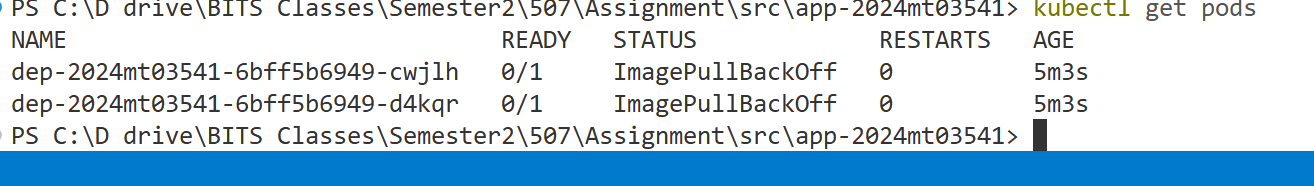
**Step 2.5: minikube (optional)**Overview: minikube is local Kubernetes, focusing on making it easy to learn and develop for Kubernetes.  
  
If minikube is not installed on your machine, please install Minikube using the link provided below:  
<https://minikube.sigs.k8s.io/docs/start/?arch=%2Fwindows%2Fx86-64%2Fstable%2F.exe+download>  
  
Restart your machine  
Execute *minikube start / minikube start --driver=docker*

**Step 3: Apply Deployment**

*kubectl apply -f config-2024mt03541.yaml*  


*kubectl apply -f dep-2024mt03541.yaml*  


Verify replicas:

kubectl get pods

*Delete pods  
kubectl delete pod dep-2024mt03541-6bff5b6949-cwjlh  
kubectl delete pod dep-2024mt03541-6bff5b6949-d4kqr*

**Task 5: Configure Networking with Load Balancer**

**Step 1: Service**

**svc-2024mt03541.yaml**

*apiVersion: v1*

*kind: Service*

*metadata:*

*name: svc-2024mt03541*

*spec:*

*type: LoadBalancer*

*selector:*

*app: flask-app*

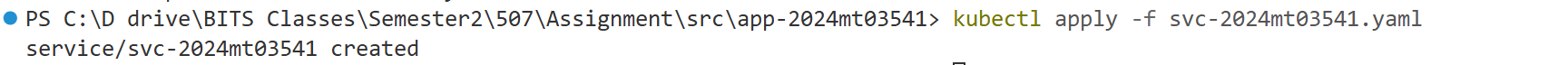
*ports:*

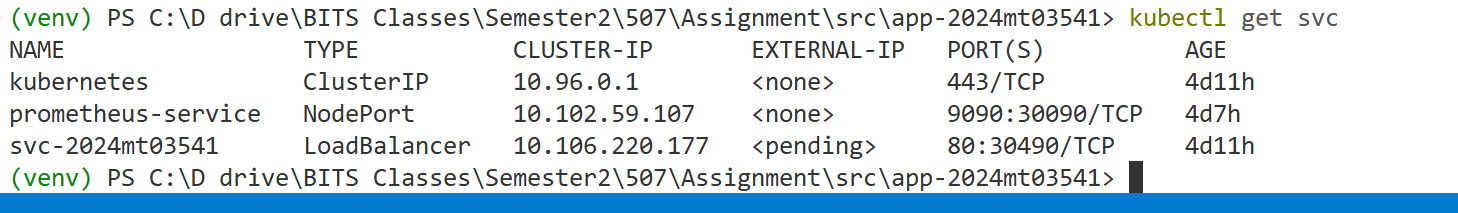
*- protocol: TCP*

*port: 80*

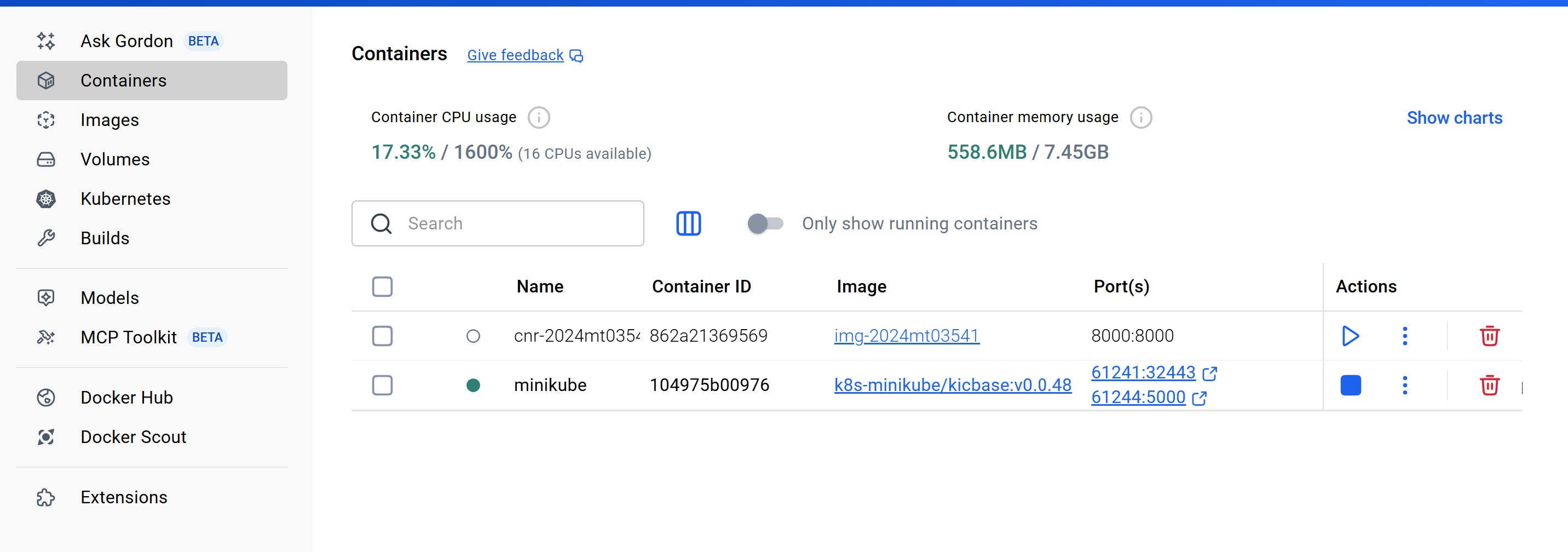
*targetPort: 8000*

**Step 2: Apply Service**

*kubectl apply -f svc-2024mt03541.yaml*  
  


*kubectl get svc*  
  


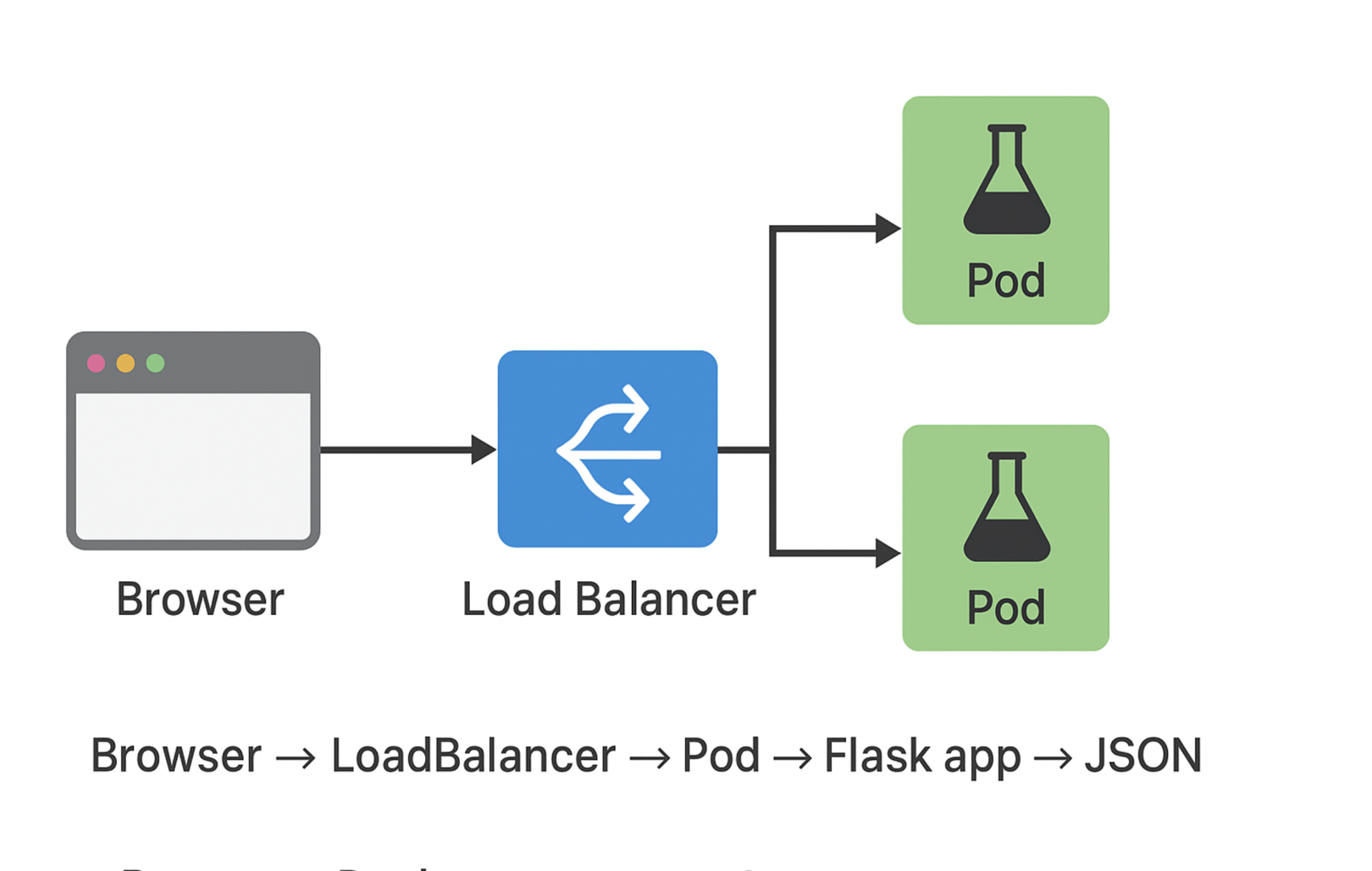
* Access endpoint: http://<minikube\_ip> → /get\_info   
  Access endpoint: http:// 192.168.49.2/get\_info

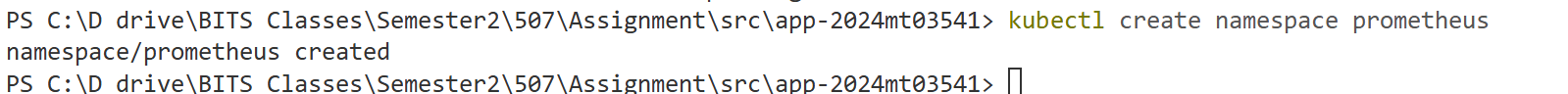
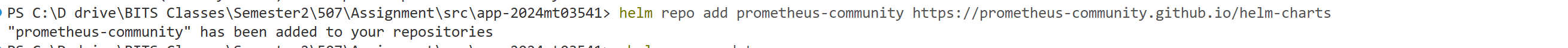
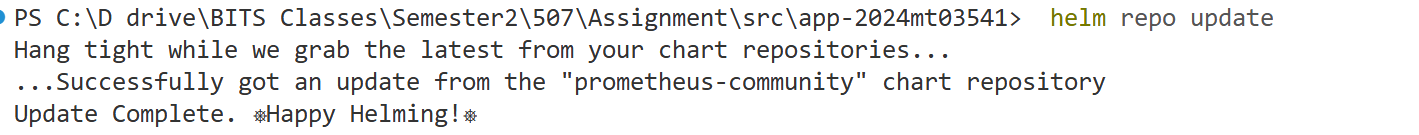
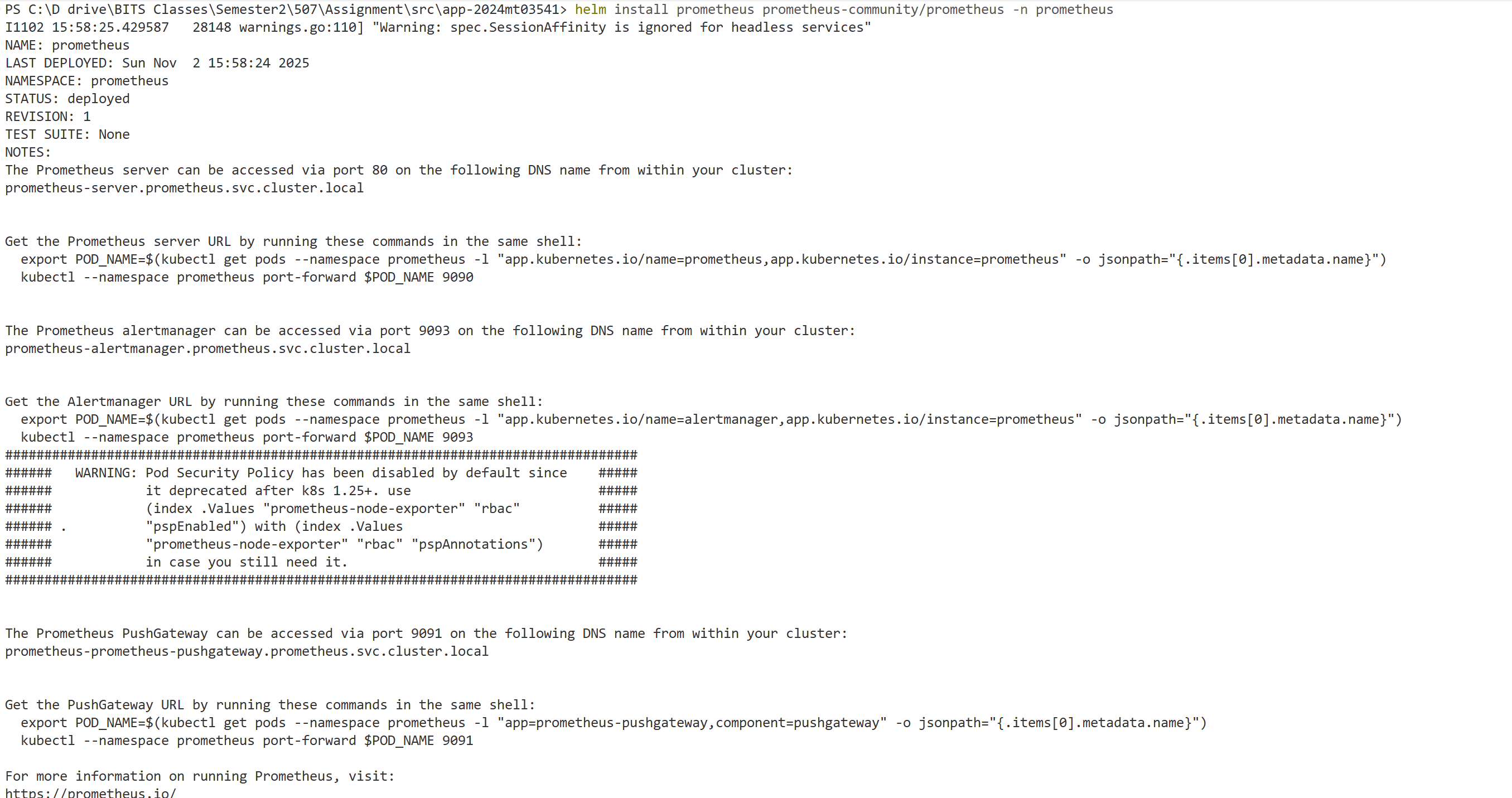


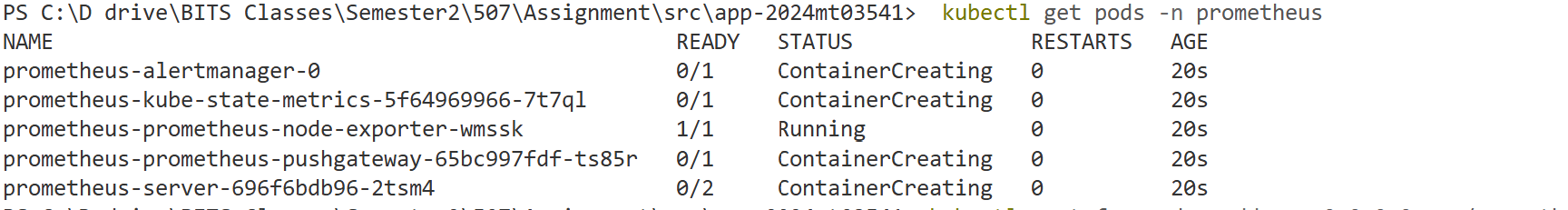
* Use curl to simulate multiple requests:  
  Curl

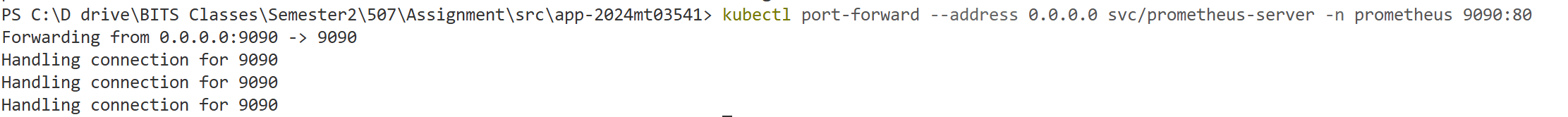
for i in {1..10}; do curl http://<minikube\_ip>/get\_info; done  
  
#powershell  
1..10 | ForEach-Object { Invoke-WebRequest -Uri http://127.0.0.1:8000/get\_info | Out-Null }

Screenshot:   
Requests load balanced across replicas.



**Task 6: Configure Prometheus for Metrics**kubectl create namespace prometheus ****  
Uninstall prometheus*helm uninstall prometheus-server -n prometheus*  
Add helm chart for prometheus*helm repo add prometheus-community https://prometheus-community.github.io/helm-charts* ****  
Update repository  
*helm repo update* *****helm install prometheus prometheus-community/prometheus -n prometheus* ****

*kubectl get pods -n prometheus* ****Port forwarding*kubectl port-forward --address 0.0.0.0 svc/prometheus-server -n prometheus 9090:80*

  
  
  
**Additional commands**  
  
If you’re using a Kubernetes ConfigMap for Prometheus, update it:  
  
Describe Prometheus config

*kubectl describe configmap prometheus-config -n prometheus*

Edit Prometheus config

*kubectl edit configmap prometheus-config -n prometheus*

Or apply directly:

*kubectl apply -f prometheus-config.yaml -n prometheus*

Restart Prometheus Deployment

After updating the config, restart Prometheus to reload the configuration:

*kubectl rollout restart deployment prometheus-server -n prometheus*

**Step 1: Prometheus Config**

**prometheus-config.yaml**

apiVersion: v1

kind: ConfigMap

global:

  scrape\_interval: 5s

scrape\_configs:

  - job\_name: 'flask\_app'

    metrics\_path: /metrics

    static\_configs:

      - targets: ['localhost:8000']

**prometheus-deployment.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

  name: prometheus-deployment

spec:

  replicas: 1

  selector:

    matchLabels:

      app: prometheus

  template:

    metadata:

      labels:

        app: prometheus

    spec:

      containers:

        - name: prometheus

          image: prom/prometheus:v2.52.0

          args:

            - "--config.file=/etc/prometheus/prometheus.yml"

          ports:

            - containerPort: 9090

          volumeMounts:

            - name: prometheus-config-volume

              mountPath: /etc/prometheus

      volumes:

        - name: prometheus-config-volume

          configMap:

            name: prometheus-config

**prometheus-service.yaml**  
  
apiVersion: v1

kind: Service

metadata:

  name: prometheus-service

spec:

  type: NodePort

  ports:

    - port: 9090

      targetPort: 8080

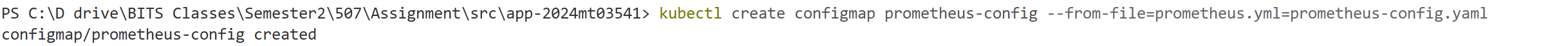
      nodePort: 30090

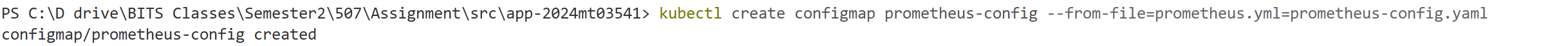
  selector:

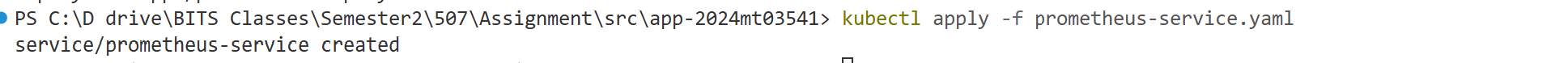
    app: prometheus

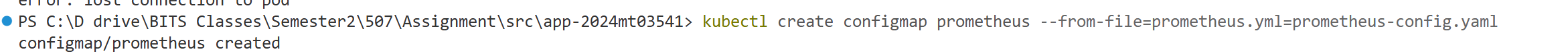
**Install Prometheus in cluster (Minikube):  
Download minikube for windows from** <https://minikube.sigs.k8s.io/docs/start/?arch=%2Fwindows%2Fx86-64%2Fstable%2F.exe+downloa>  
  
Config

*kubectl apply -f prometheus-config.yaml   
  
Config, we generate with another way*

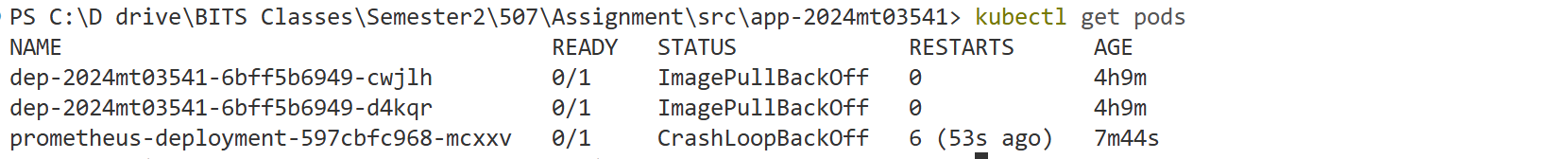
*kubectl create configmap prometheus-config --from-file=prometheus.yml=prometheus-config.yaml*

  
 *Deployment  
kubectl apply -f prometheus-deployment.yaml* *Service*

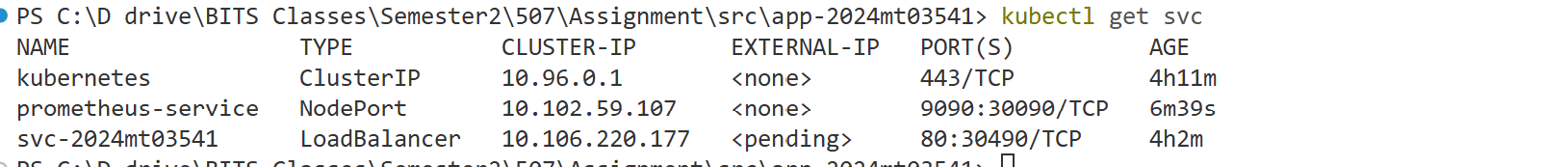
*kubectl apply -f prometheus-service.yaml*

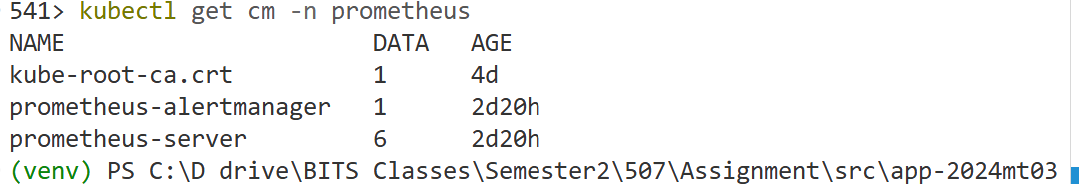
delete config map  
*kubectl delete configmap prometheus-config*  
  


Check Pods  
  
*kubectl get pods*



Check Services

*kubectl get svc*  
  


kubectl get cm -n prometheus  
  
  
Check Prometheus Config  
*kubectl describe configmap prometheus-server -n prometheus*  
  
If using Minikube*: minikube ip* 192.168.49.2  
  

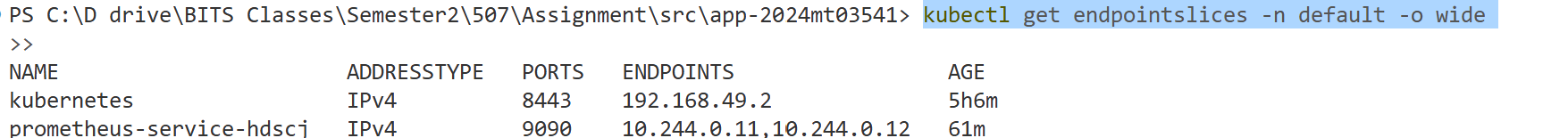

If NodePort is not accessible, use:  
Port forwarding

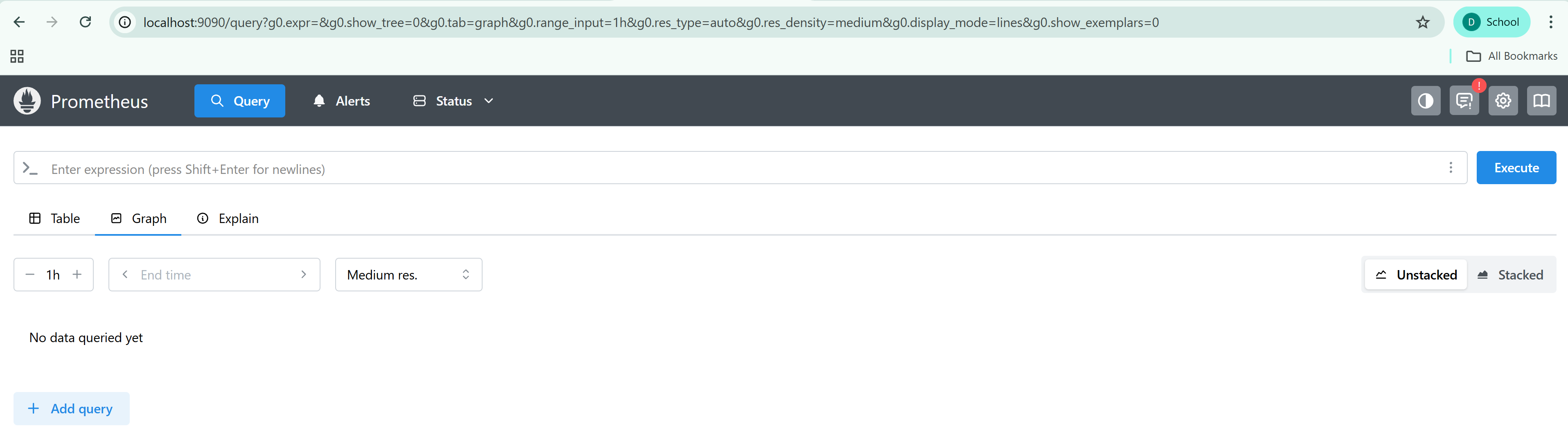
*kubectl port-forward service/prometheus-service 9090:9090*  
  
Delete Prometheus pod if required

*kubectl delete pod -l app=prometheus*

Check Endpoint slice

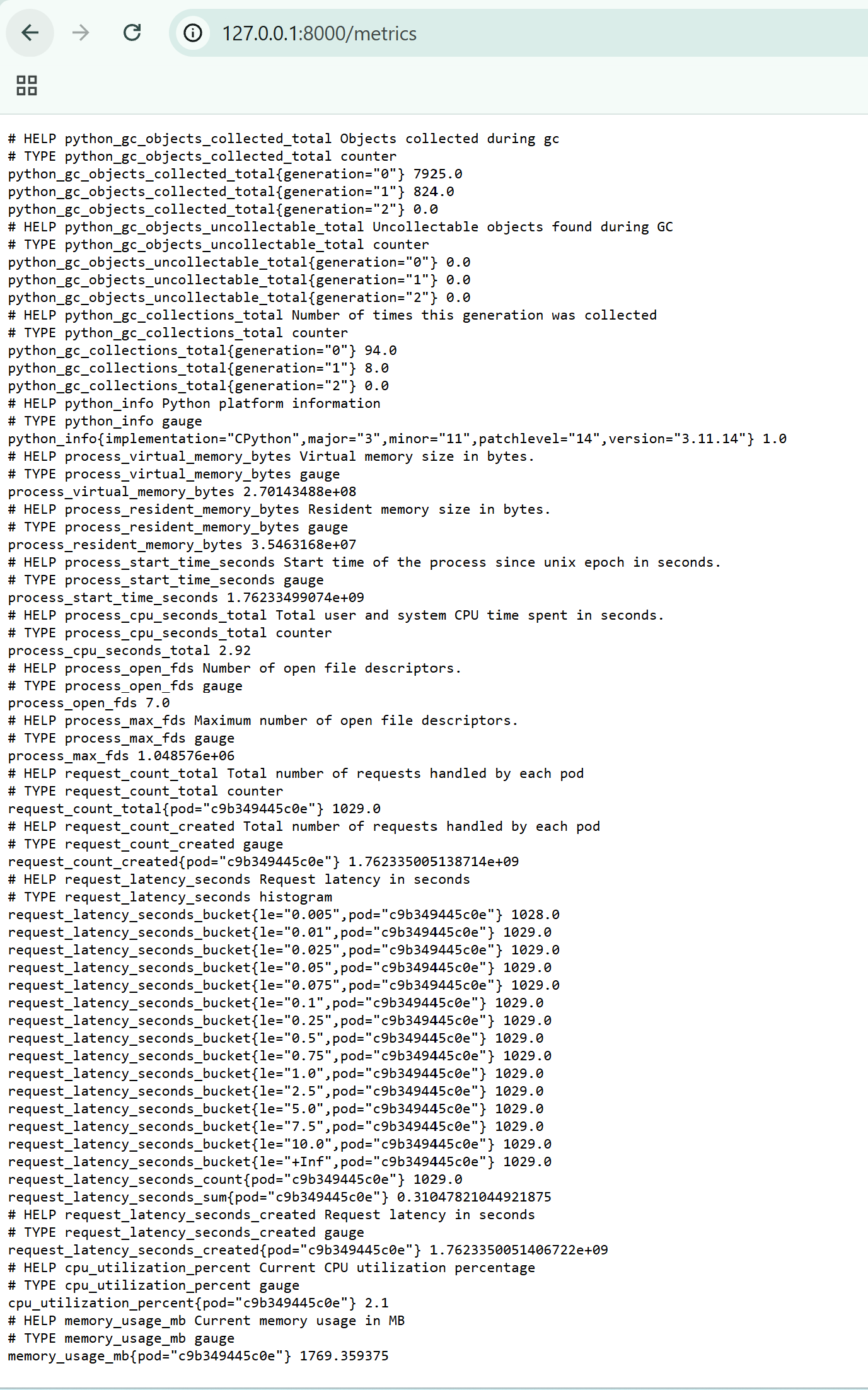
*kubectl get endpointslices -n default -o wide*

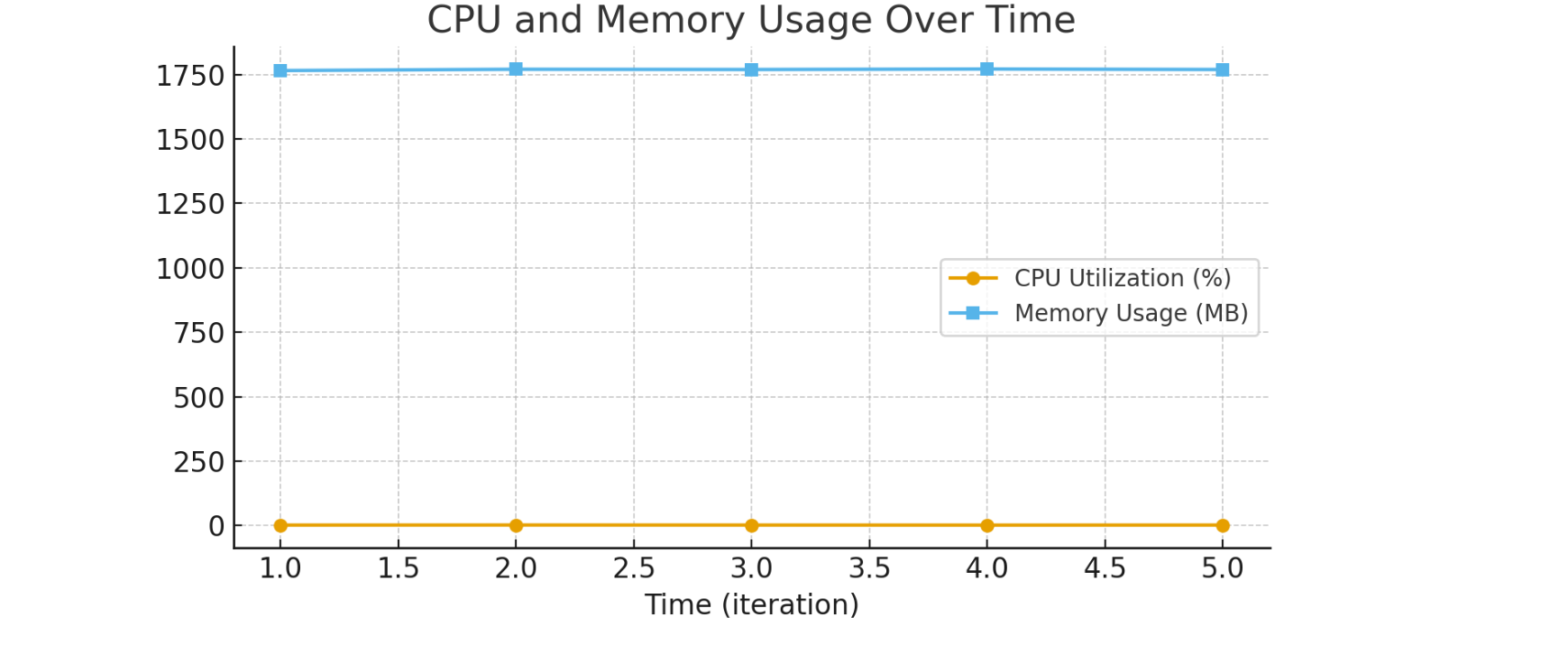


**Step 2: Verify Metrics**Open Prometheus UI: http://localhost:9090/

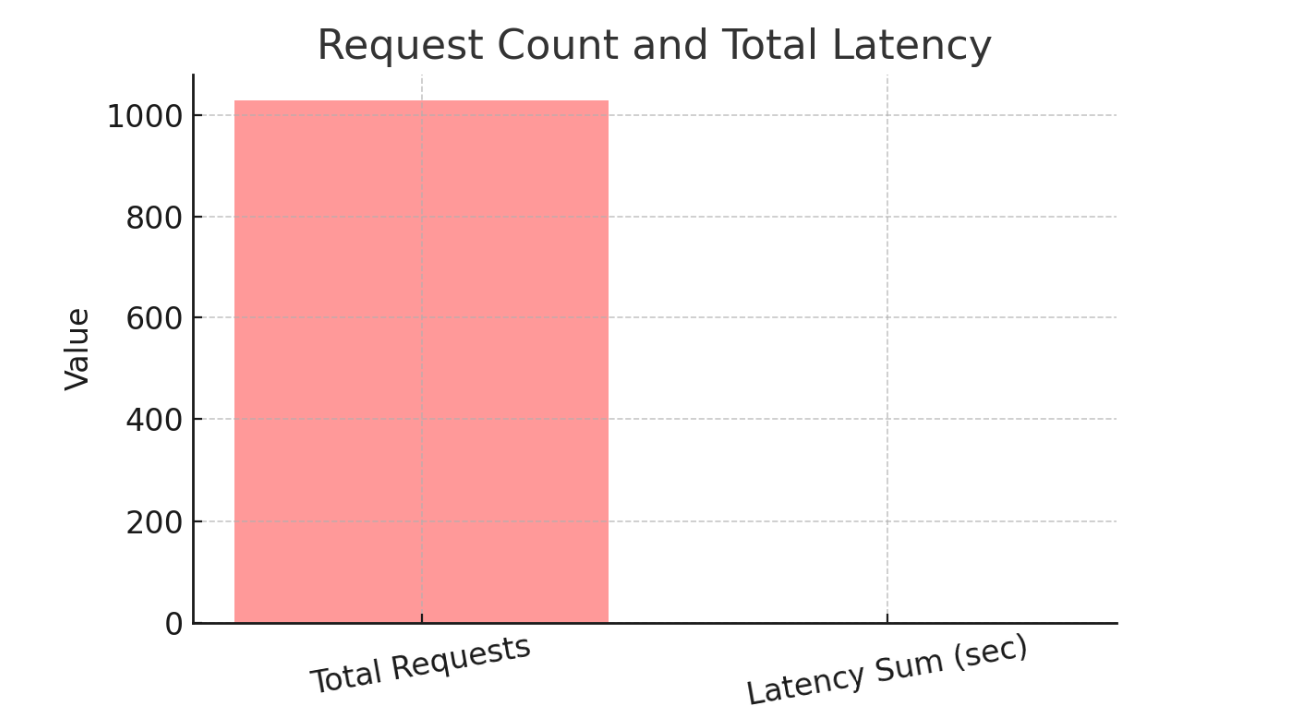
Install Prometheus Client for charts/graphs  
  
*python -m pip install prometheus\_client*  
  


Prometheus config file  
prometheus --config.file=prometheus.yml

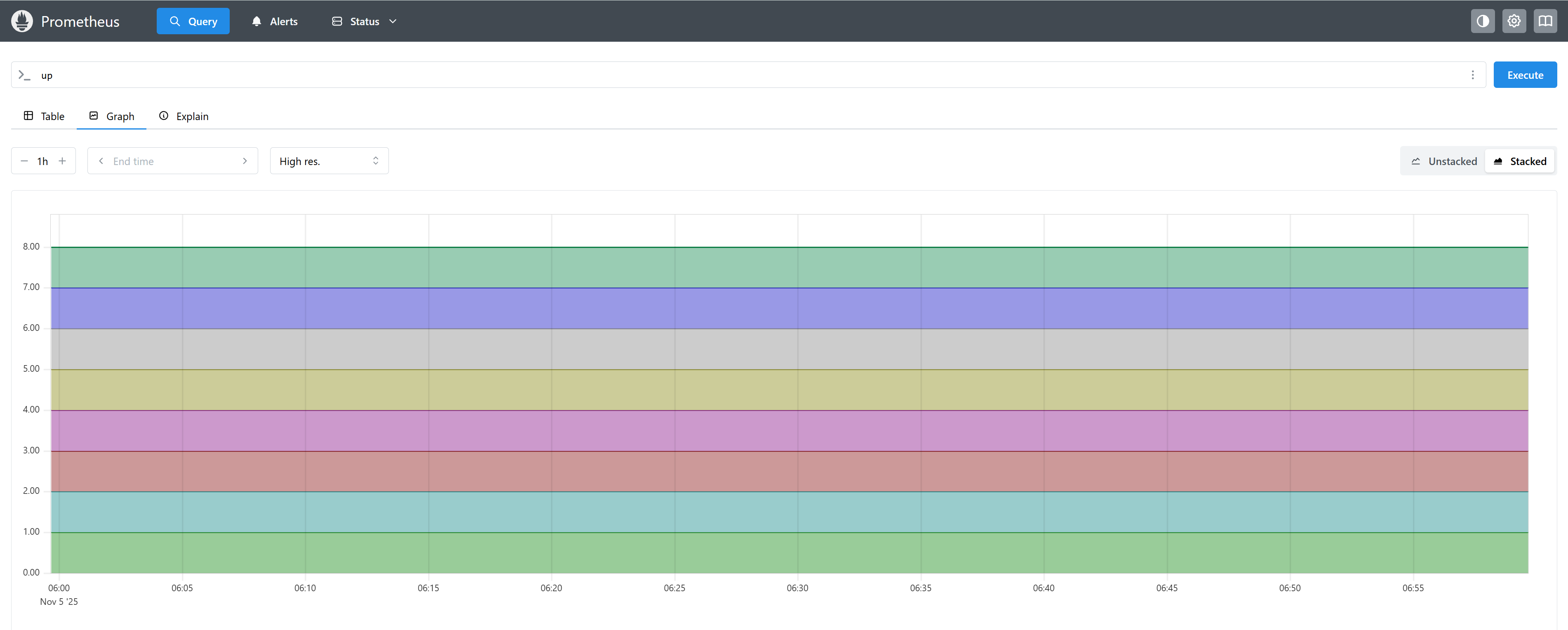
http://127.0.0.1:8000/metrics

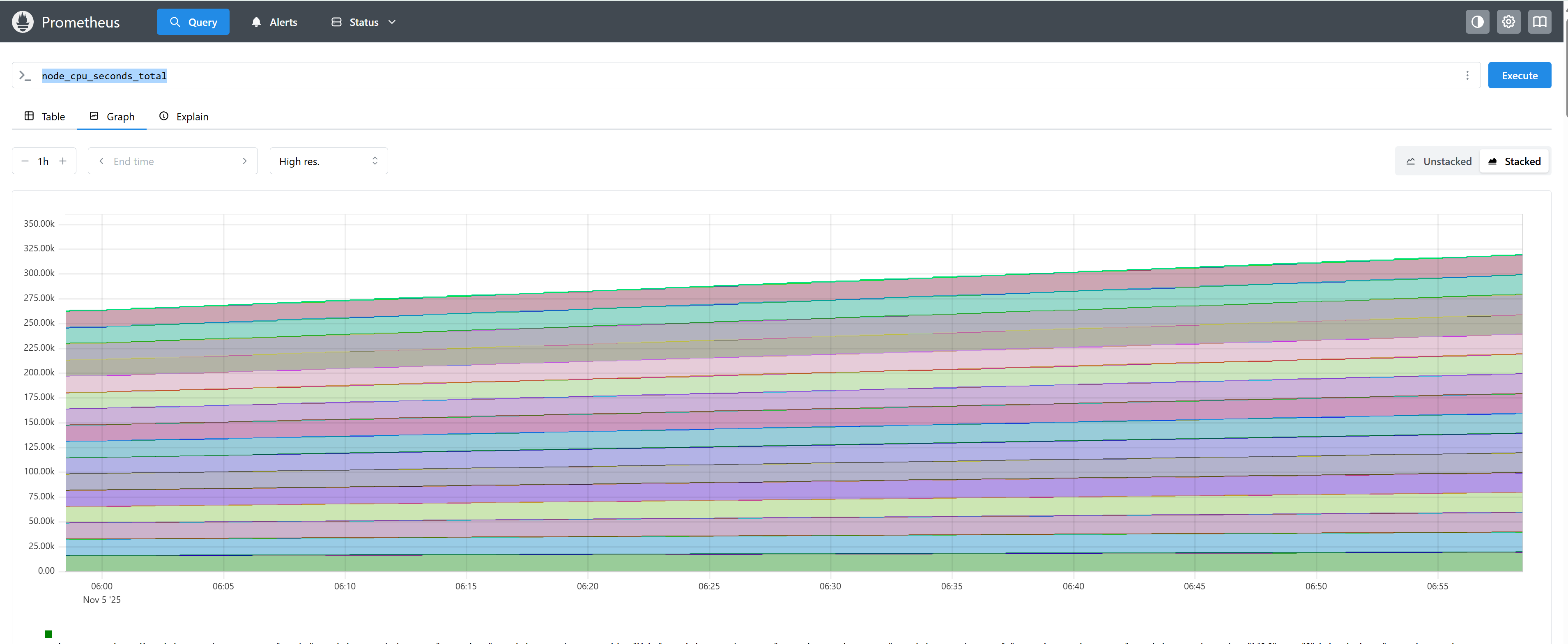
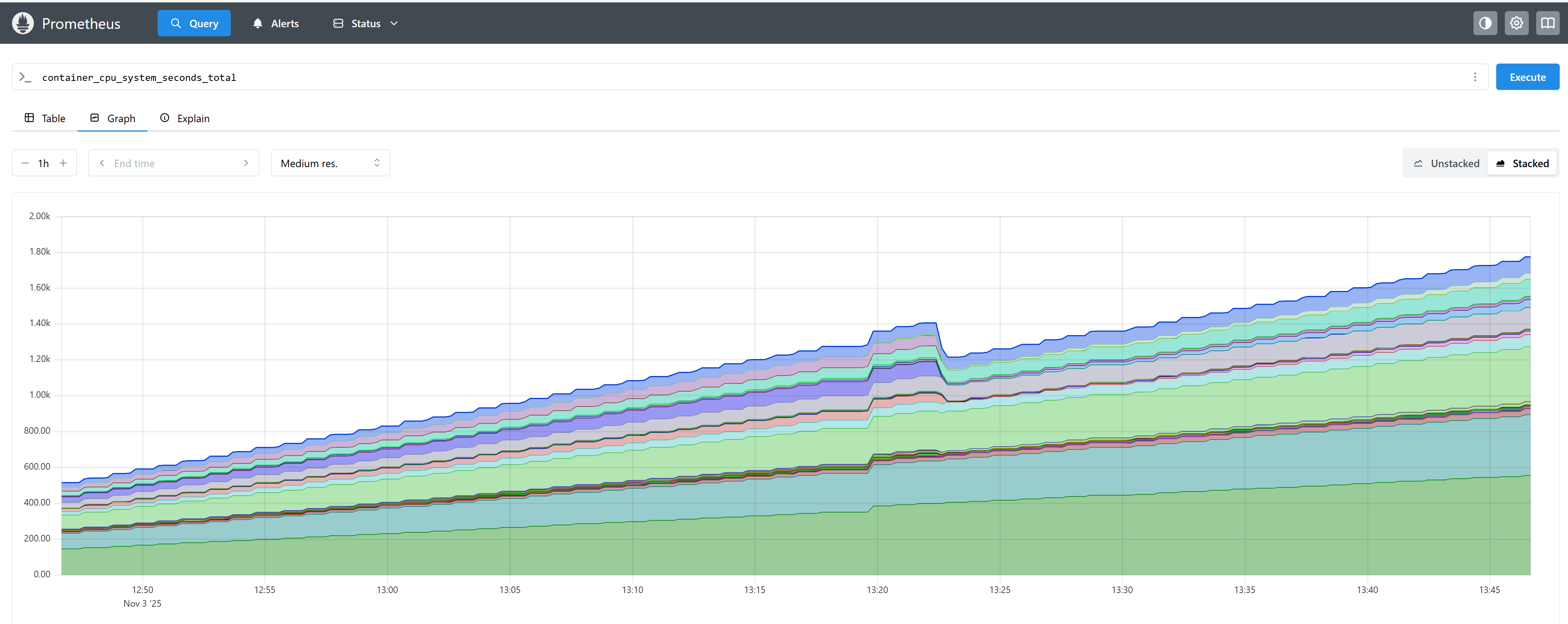
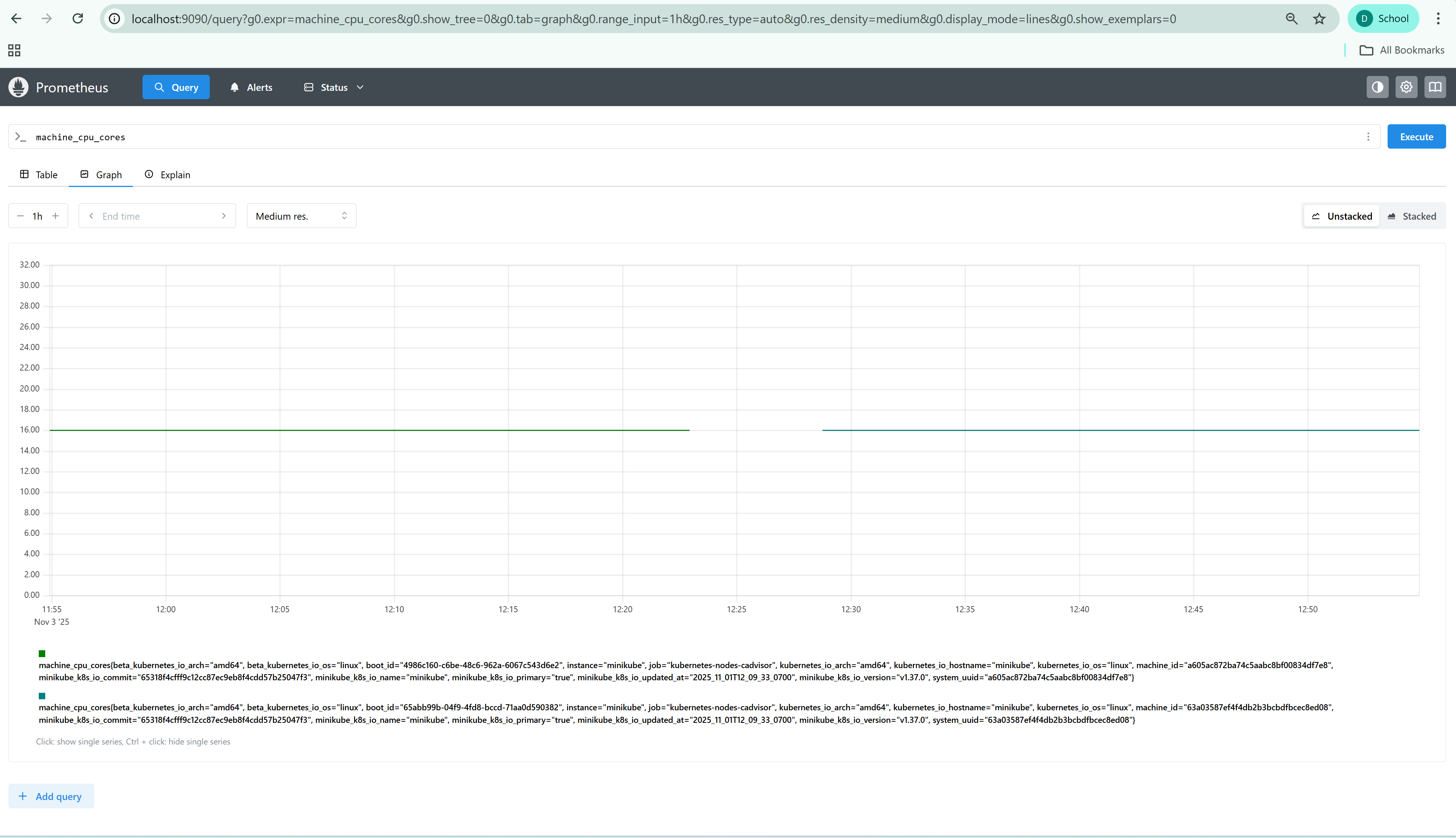
**Screenshot:**   
  
**Show CPU**  


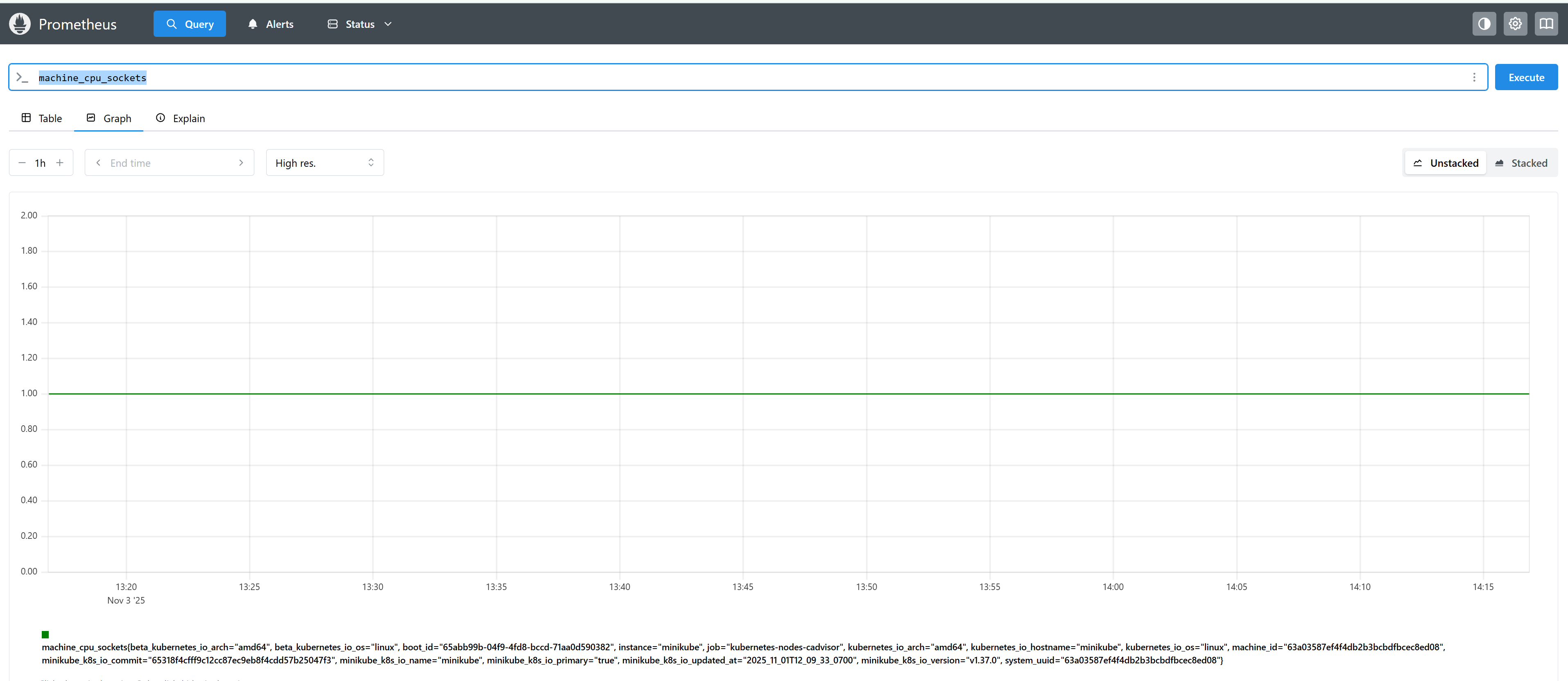
Request Count vs Latency

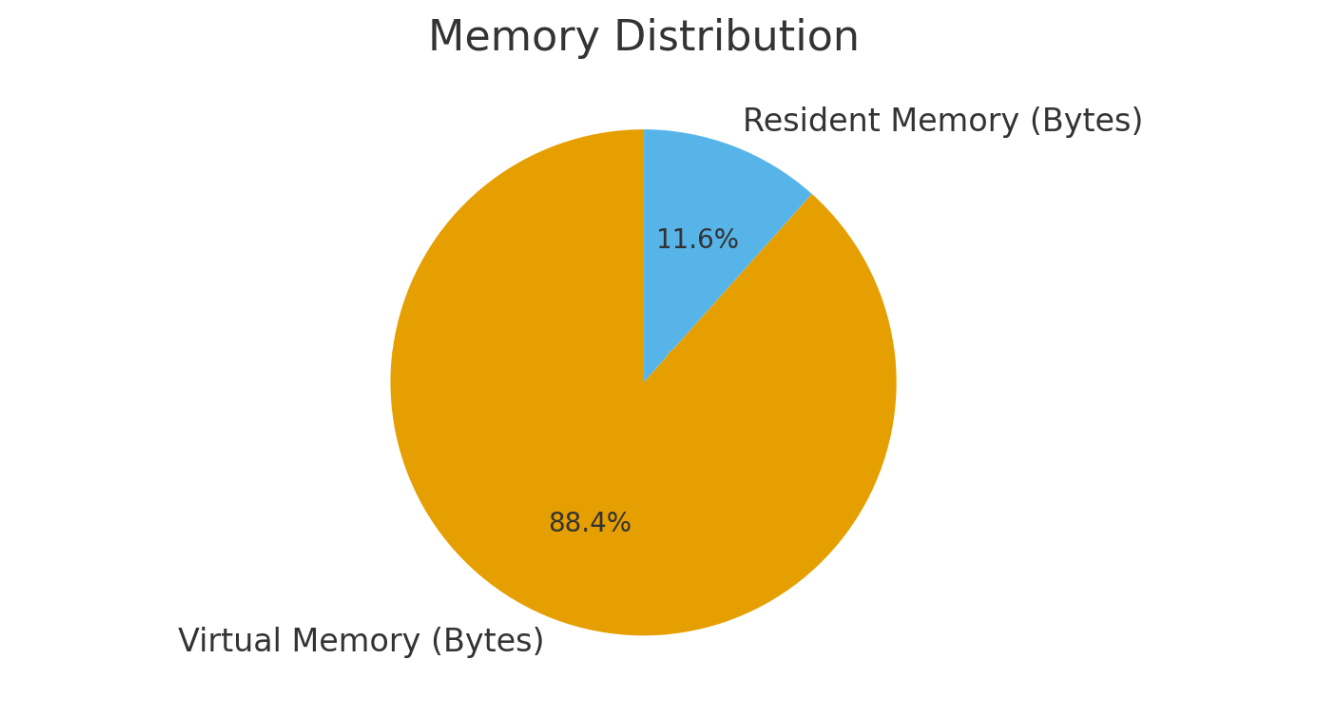
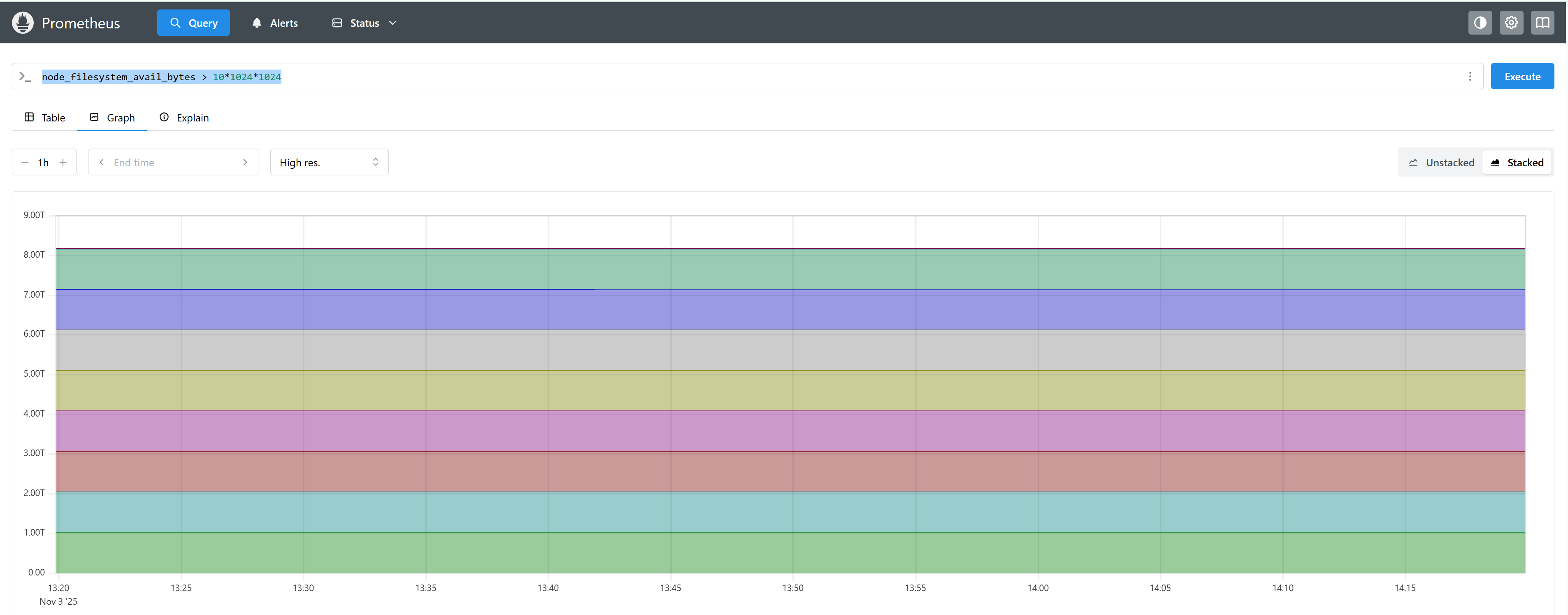
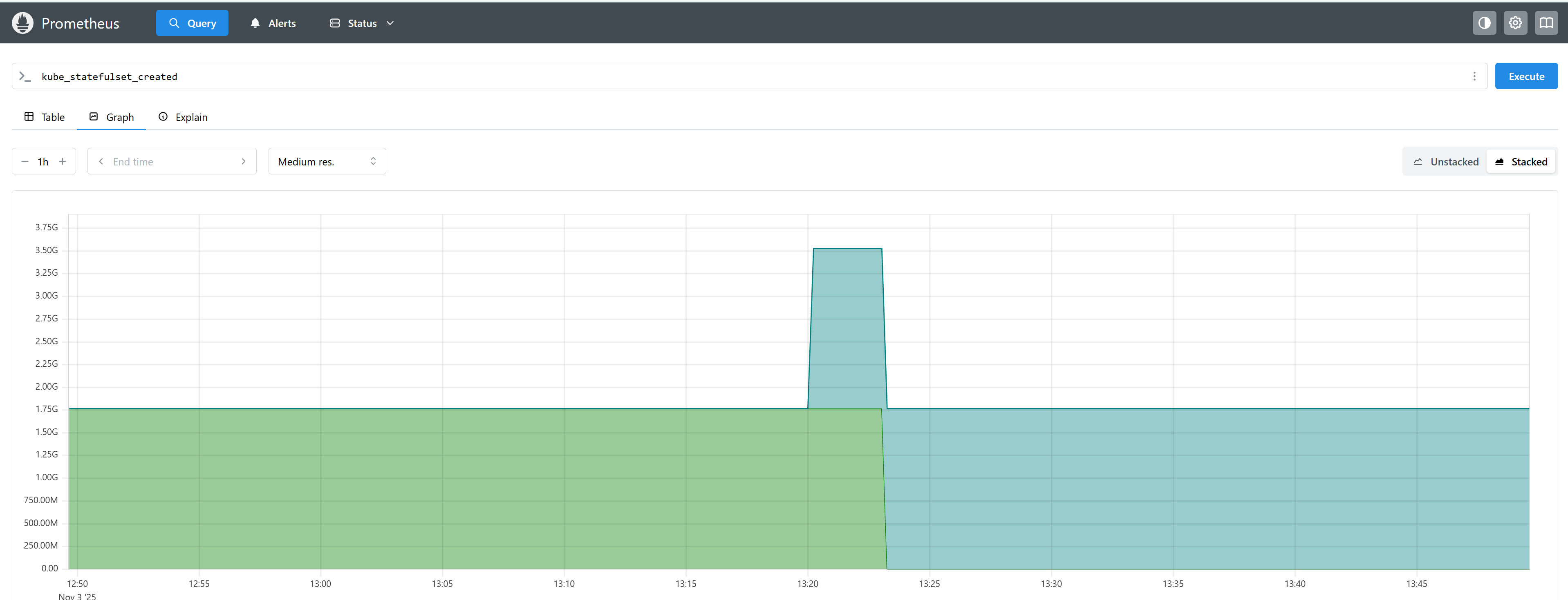
****

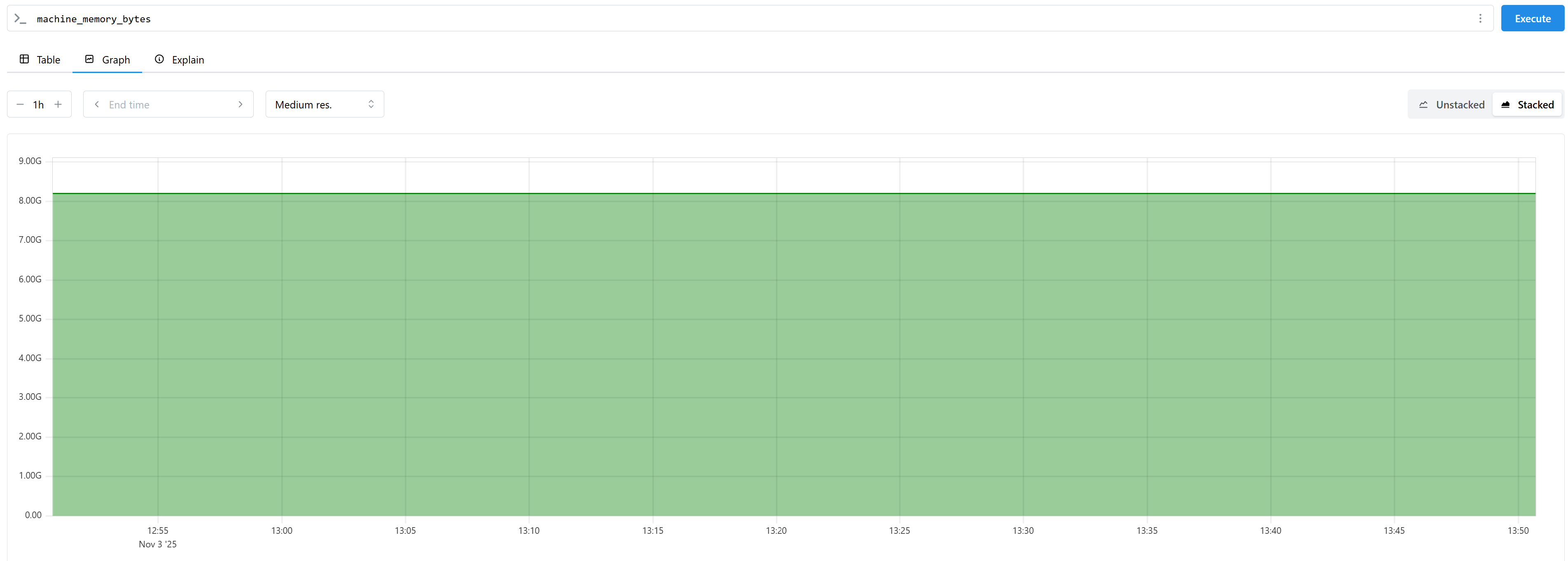
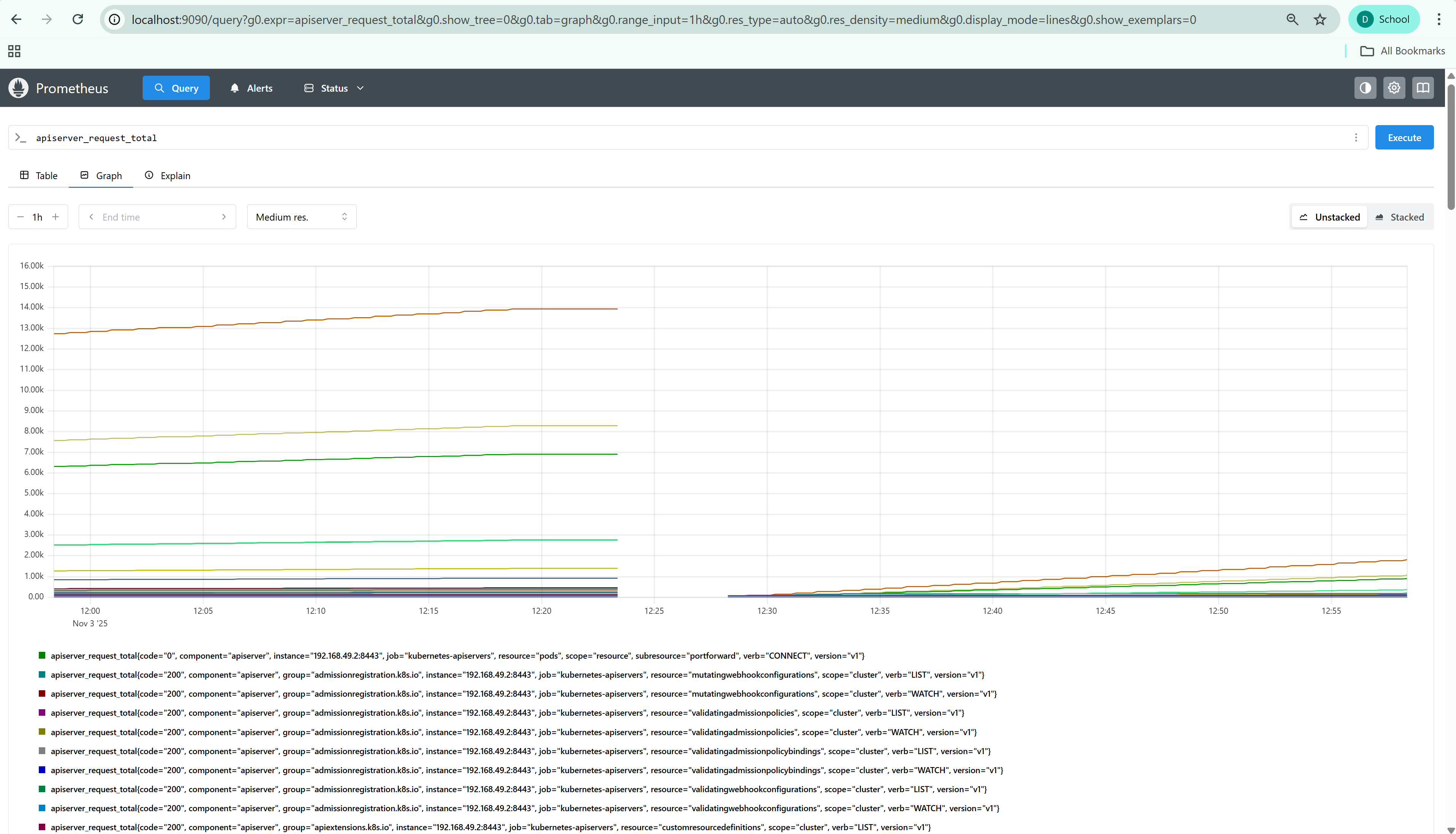
Up

****node\_cpu\_seconds\_total

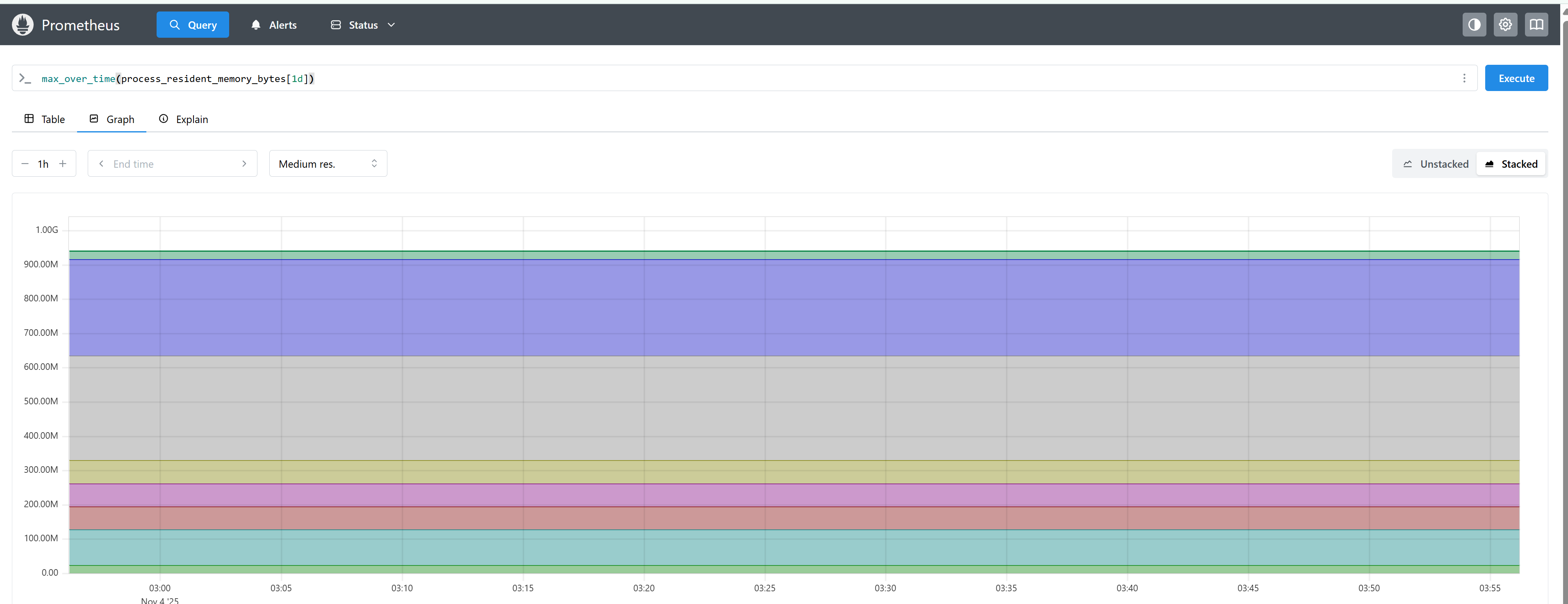
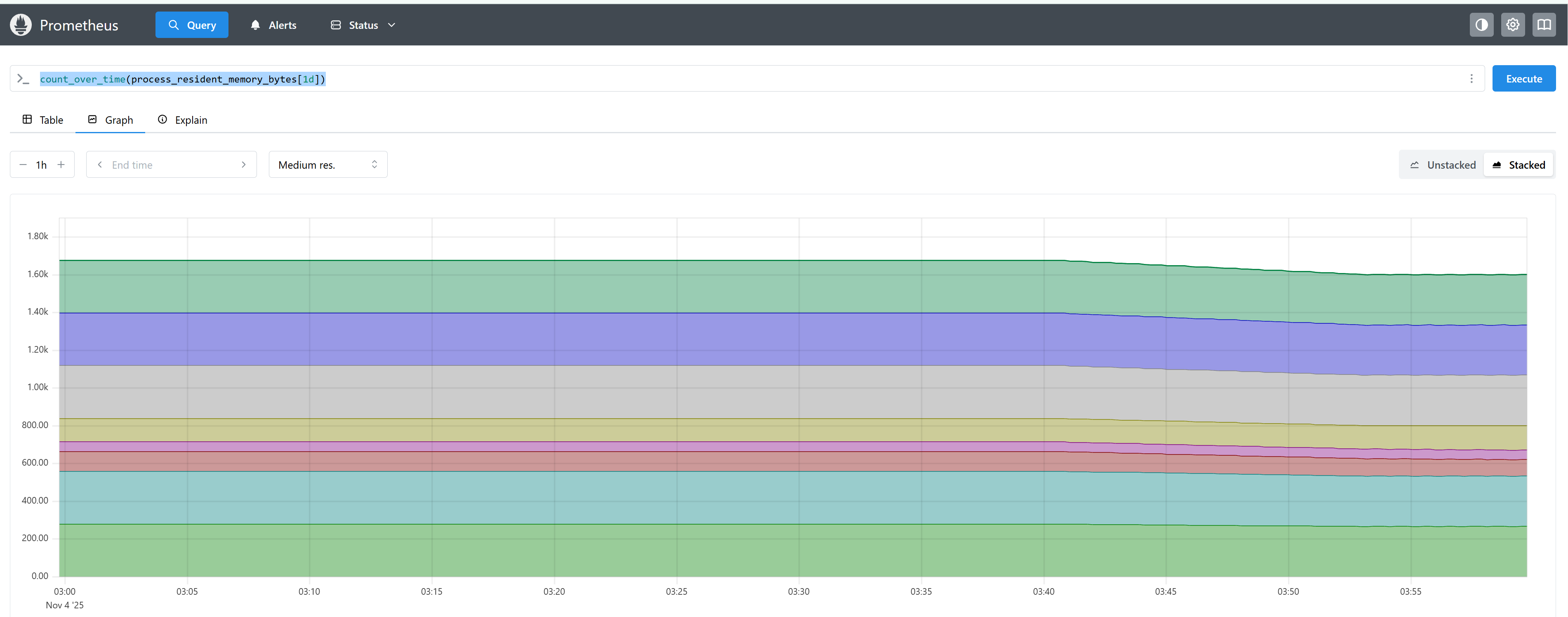
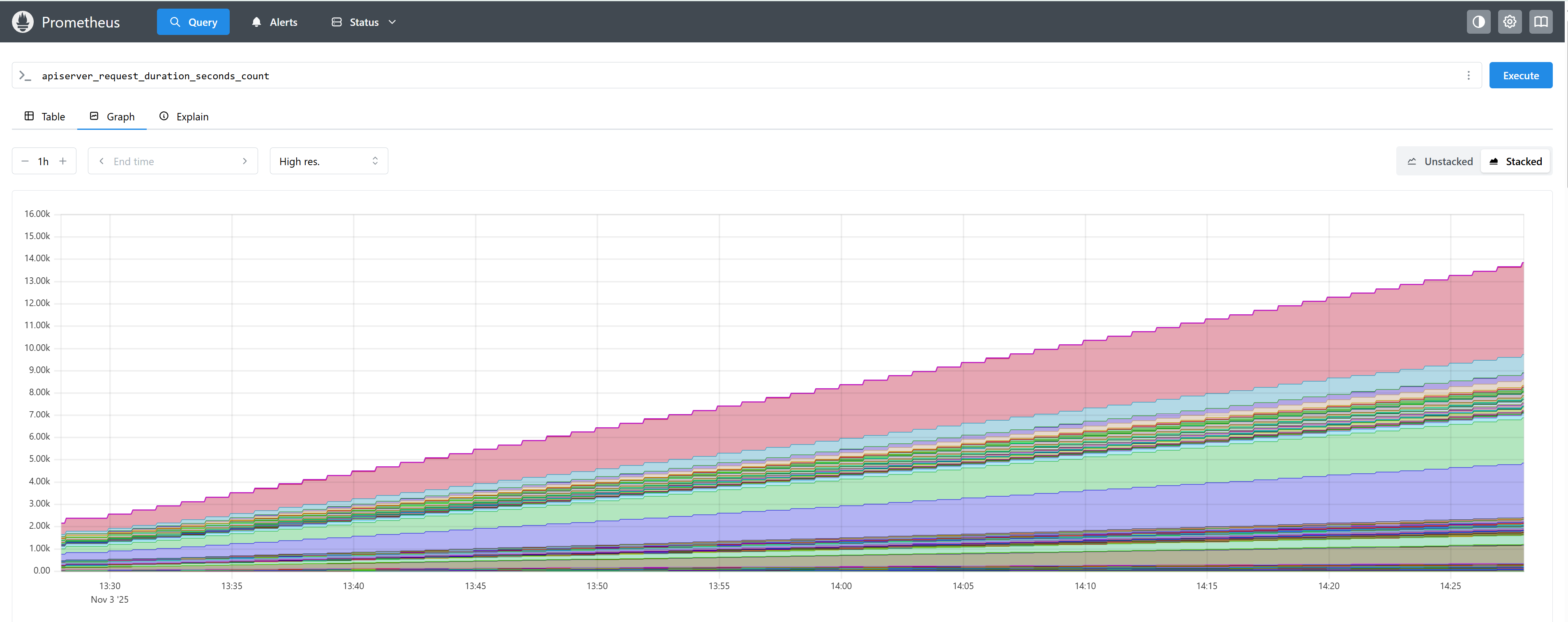
****  
CPU seconds total  
  
  
machine\_cpu\_cores  
  
  
  
  
  
  
machine\_cpu\_sockets

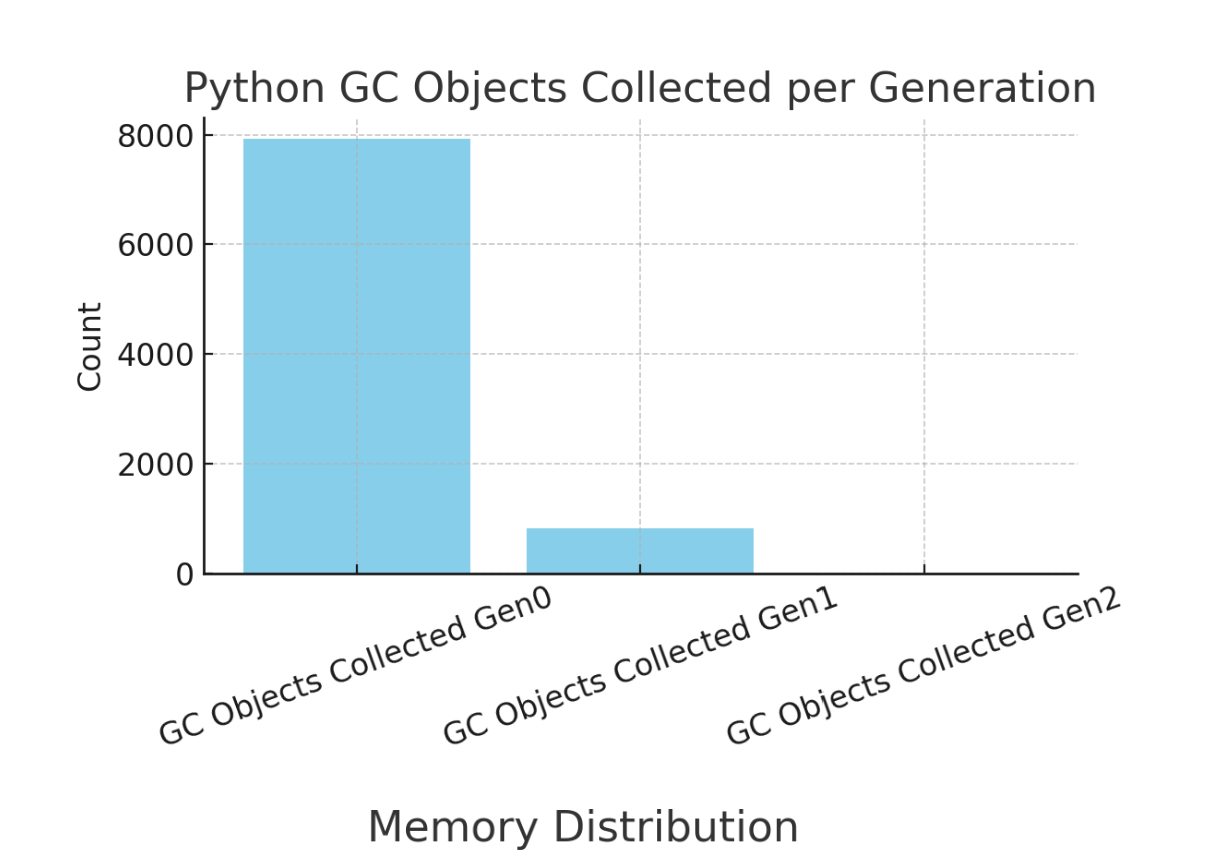


**Memory  
  
**  
node\_filesystem\_avail\_bytes > 10\*1024\*1024  
  
kube\_statefulset\_created  


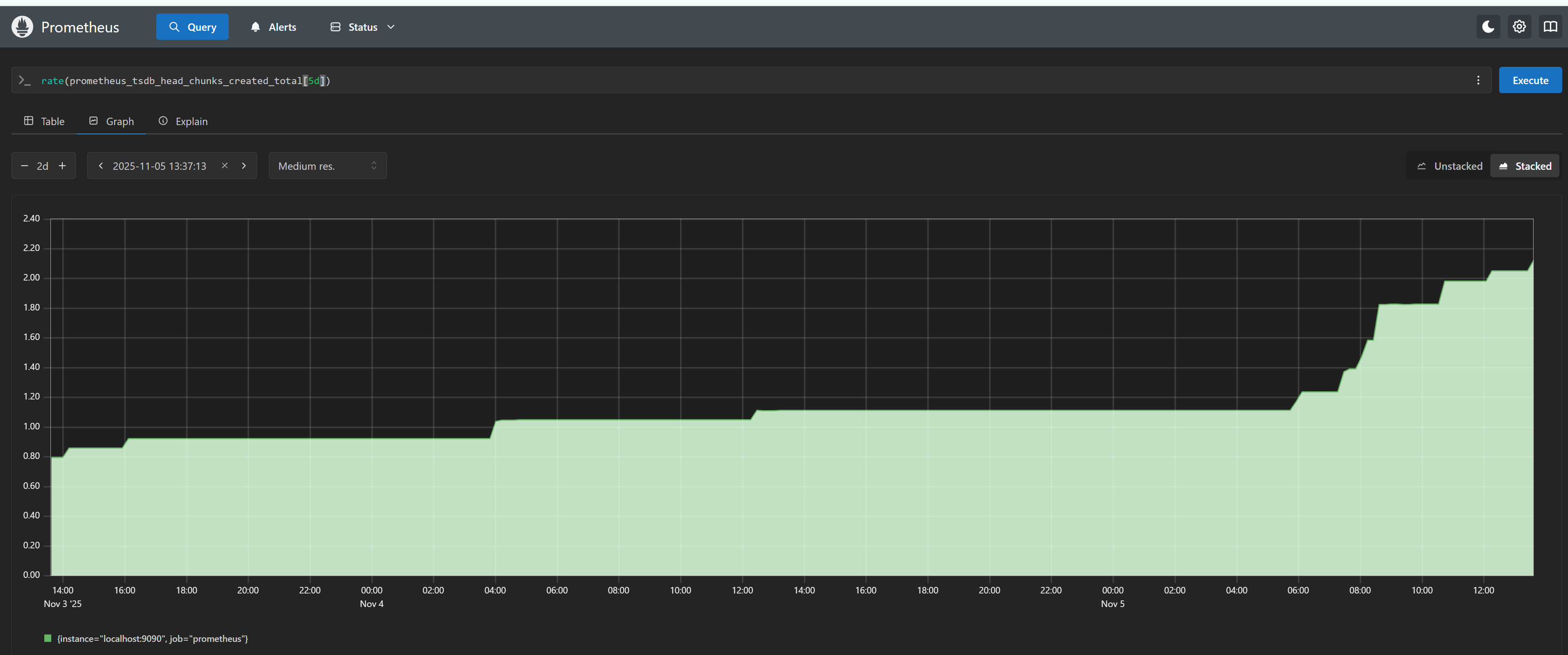
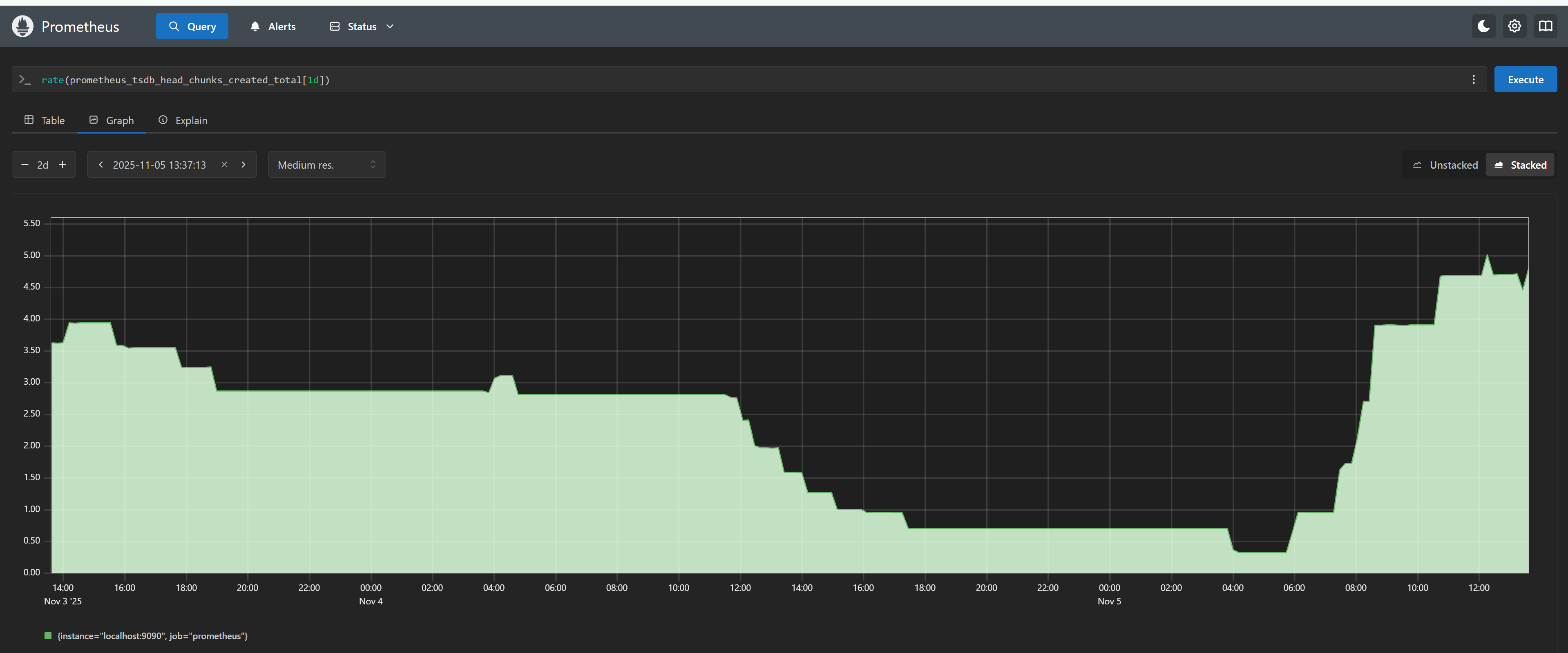
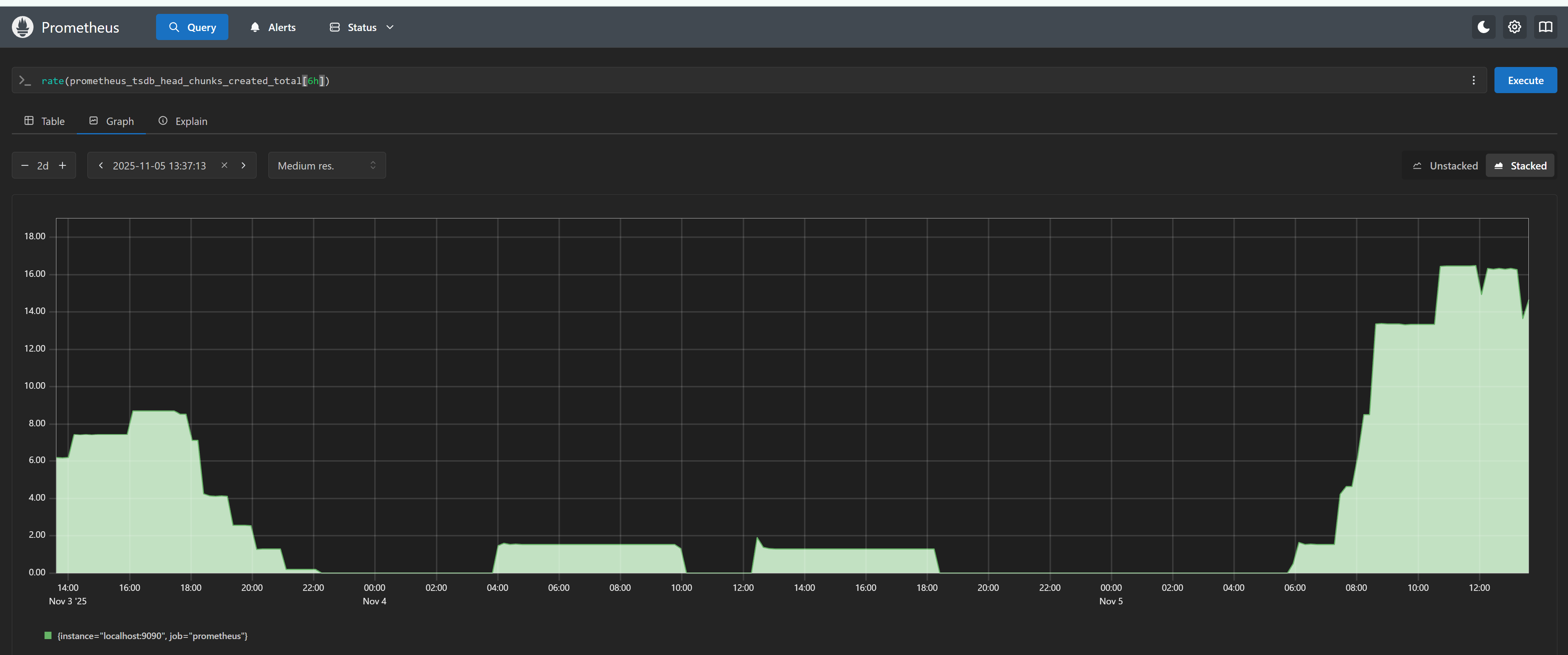
Machine memory in bytes  
  
  
request count per pod.  


**Graph:** Requests over time:  
avg\_over\_time(go\_goroutines[1d])  


max\_over\_time(process\_resident\_memory\_bytes[1d])  
  
  
count\_over\_time(process\_resident\_memory\_bytes[1d])  
  
  
  
apiserver\_request\_duration\_seconds\_count  
  




**TSDB Head count**

**  
  
  
**

**CPU seconds Total**

****