**What to monitor in an AWS EKS Cluster :**

* To monitore AWS resource limits, you should monitor them proactively and set alerts to trigger once you reach a threshold, such as eighty percent of the limit, to give you sufficient time to react.

You can count on AWS Trusted Advisor to keep an eye on those limits, as well as a handful of recommendations in terms of cost optimizations, performance, security, fault tolerance, and service limits

**CloudWatch Agent**

- can analyze resource metrics in AWS CloudWatch.

- On top of the resource utilization, you will gain access to the Node health statuses (Ready, MemoryPressure, etc.).

One option also worth mentioning is AWS Elastic Service, the managed Elasticsearch solution provided by AWS. The ELK stack (Elastic-Logstash-Kibana) centralizing the solutions in one spot.

Listen for Events

* CrashLoopBackOff: a Pod keeps crashing again and again
* ImagePullBackOff: a Pod is unable to start because the node is unable to pull the container image
* Evicted: a Pod has been killed by Kubernetes (because it uses too much RAM, for instance)
* can create CloudWatch Alarms to be notified when an important event occurs.

**Prometheus.**

Monitoring an application involves several elements: logs, metrics, traces, and health checks

Prometheus is a widely popular, open-source monitoring tool that does just that. Thanks to the large community helping to maintain it, this tool has matured over time and can capture real-time metrics from multiple source systems. Prometheus saves these metrics in a time series database and alerts users when metric values cross preconfigured thresholds. Even dynamic environments, like Kubernetes, can benefit from such features.

I use prometheus and rafana to monitor the eks cluster

Prometheus is an open-source systems monitoring and alerting toolkit

Grafana is open source visualization and analytics software. It allows you to query, visualize, alert on, and explore your metrics, it provides you with tools to turn your time-series database (TSDB) data into beautiful graphs and visualizations.

Add the repo as follows:

> helm search repo prometheus-community

> helm repo add prometheus-community https://prometheus-community.github.io/helm-charts

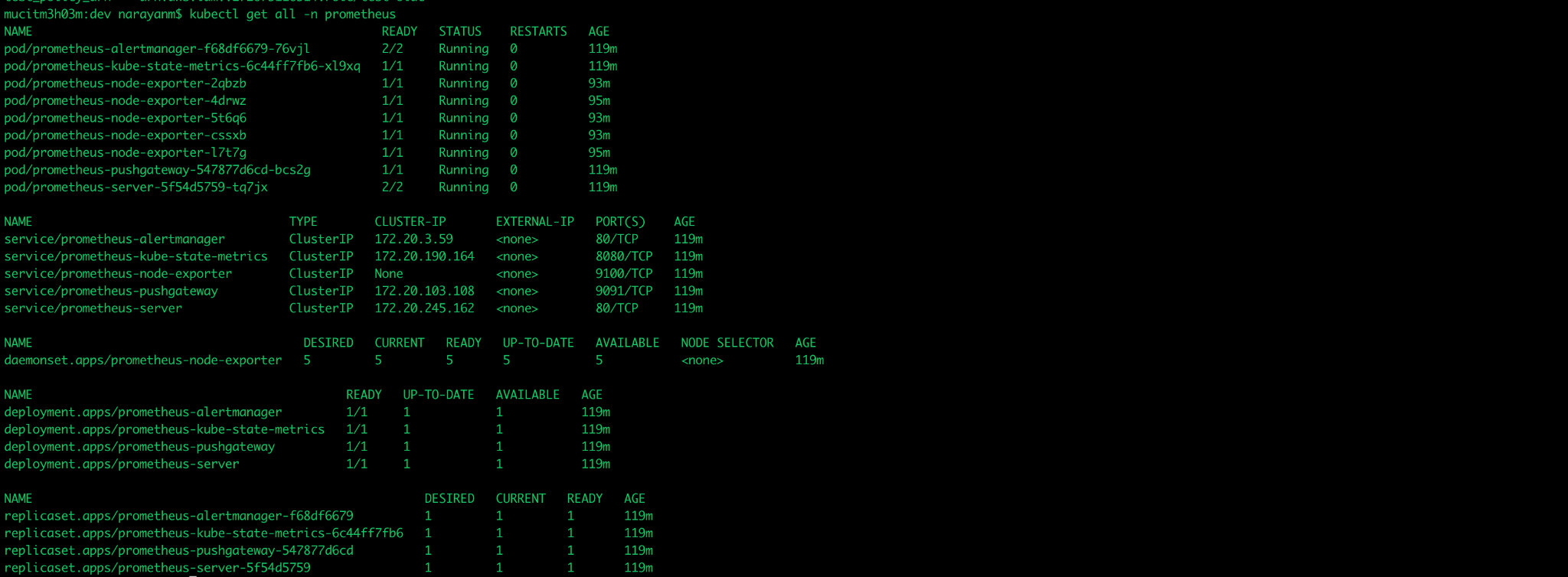
You can then run helm search repo prometheus-community to see the charts.

Deploy Prometheus

helm install prometheus prometheus-community/prometheus --namespace prometheus --set alertmanager.persistentVolume.storageClass="gp2" --set server.persistentVolume.storageClass="gp2"

Check if Prometheus components deployed as expected

> kubectl get all -n prometheus



In order to access the Prometheus server URL, we are going to use the kubectl port-forward command to access the application

> kubectl port-forward -n prometheus deploy/prometheus-server 8080:9090

Get the Prometheus server URL by running these commands in the same shell:

export POD\_NAME=$(kubectl get pods --namespace prometheus -l "app=prometheus,component=server" -o jsonpath="{.items[0].metadata.name}")

kubectl --namespace prometheus port-forward $POD\_NAME 9090

Get the Alertmanager URL by running these commands in the same shell:

export POD\_NAME=$(kubectl get pods --namespace prometheus -l "app=prometheus,component=alertmanager" -o jsonpath="{.items[0].metadata.name}")

kubectl --namespace prometheus port-forward $POD\_NAME 9093

Get the PushGateway URL by running these commands in the same shell:

export POD\_NAME=$(kubectl get pods --namespace prometheus -l "app=prometheus,component=pushgateway" -o jsonpath="{.items[0].metadata.name}")

kubectl --namespace prometheus port-forward $POD\_NAME 9091

**Grafana :**

As with Prometheus, we are setting the storage class to gp2, admin password, configuring the datasource to point to Prometheus and creating an external loadbalancer for the service.

Create YAML file called grafana.yaml

cat << EoF > grafana.yaml

datasources:

datasources.yaml:

apiVersion: 1

datasources:

- name: Prometheus

type: prometheus

url: http://prometheus-server.prometheus.svc.cluster.local

access: proxy

isDefault: true

EoF

Create a grafana namespace

> kubectl create namespace grafana

Search repo

> helm search repo grafana

> helm install grafana grafana/grafana --namespace grafana --set persistence.storageClassName="gp2" --set persistence.enabled=true --set adminPassword='EKS!sAWSome' --values grafana.yaml --set service.type=LoadBalancer

> helm install grafana stable/grafana --namespace grafana --set persistence.storageClassName="gp2" --set persistence.enabled=true --set adminPassword='EKS!sAWSome' --values grafana.yaml --set service.type=LoadBalancer

1. Get your 'admin' user password by running:

kubectl get secret --namespace grafana grafana -o jsonpath="{.data.admin-password}" | base64 --decode ; echo

2. The Grafana server can be accessed via port 80 on the following DNS name from within your cluster:

grafana.grafana.svc.cluster.local

Get the Grafana URL to visit by running these commands in the same shell:

NOTE: It may take a few minutes for the LoadBalancer IP to be available.

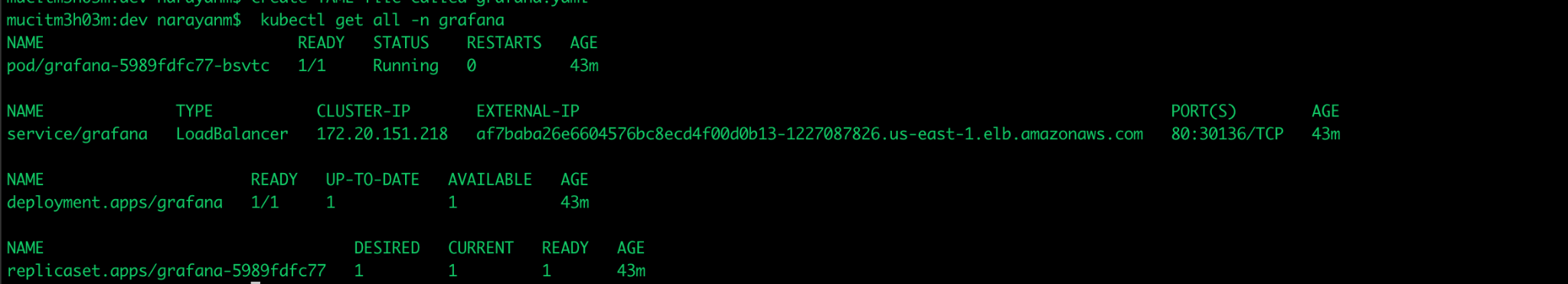
You can watch the status of by running 'kubectl get svc --namespace grafana -w grafana'

export SERVICE\_IP=$(kubectl get svc --namespace grafana grafana -o jsonpath='{.status.loadBalancer.ingress[0].ip}')

http://$SERVICE\_IP:80

3. Login with the password from step 1 and the username: admin

mucitm3h03m:monitoring-grafana narayanm$ kubectl get all -n grafana

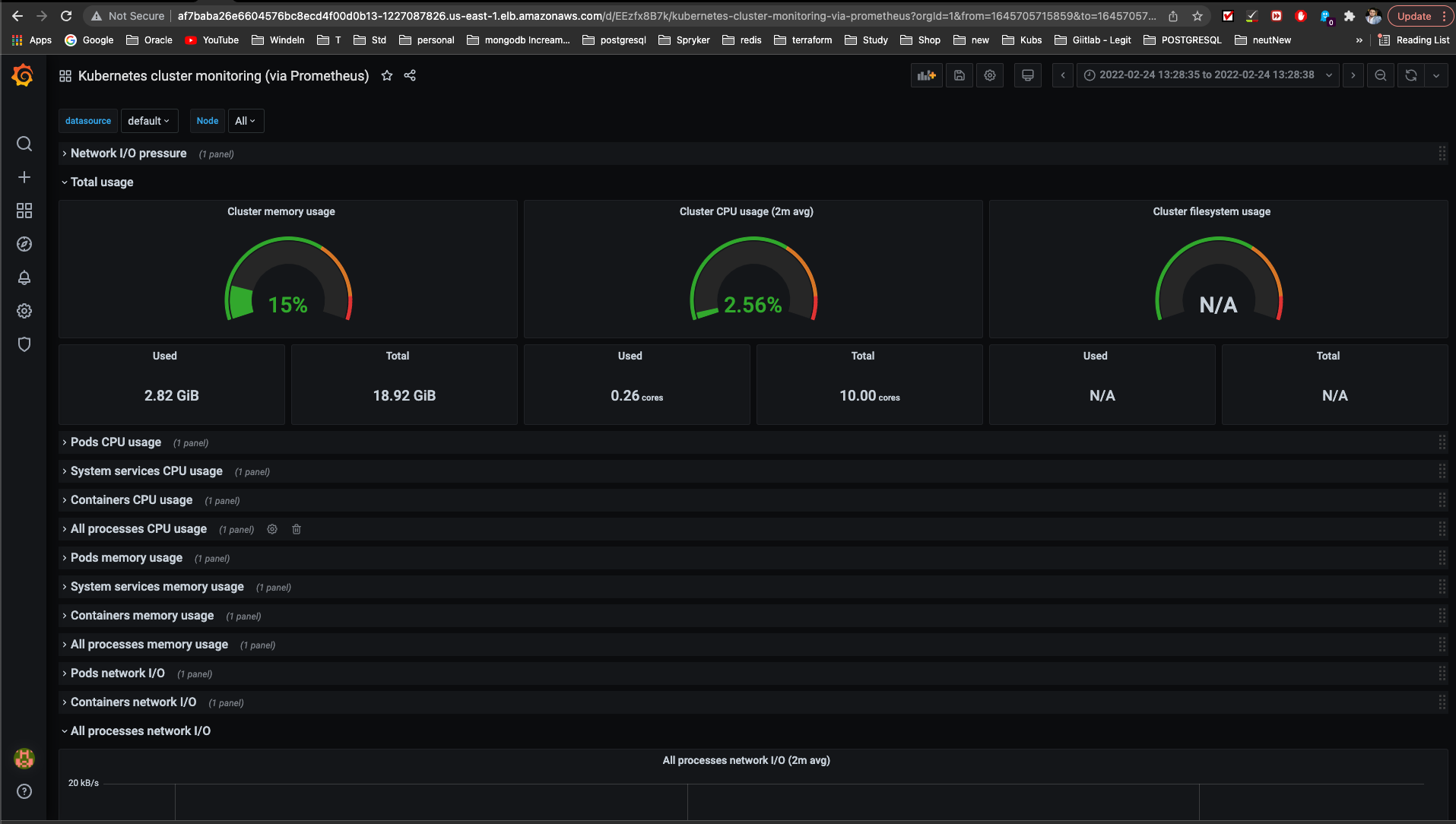


**Cluster Monitoring Dashboard**

- left panel and select ‘Import’.

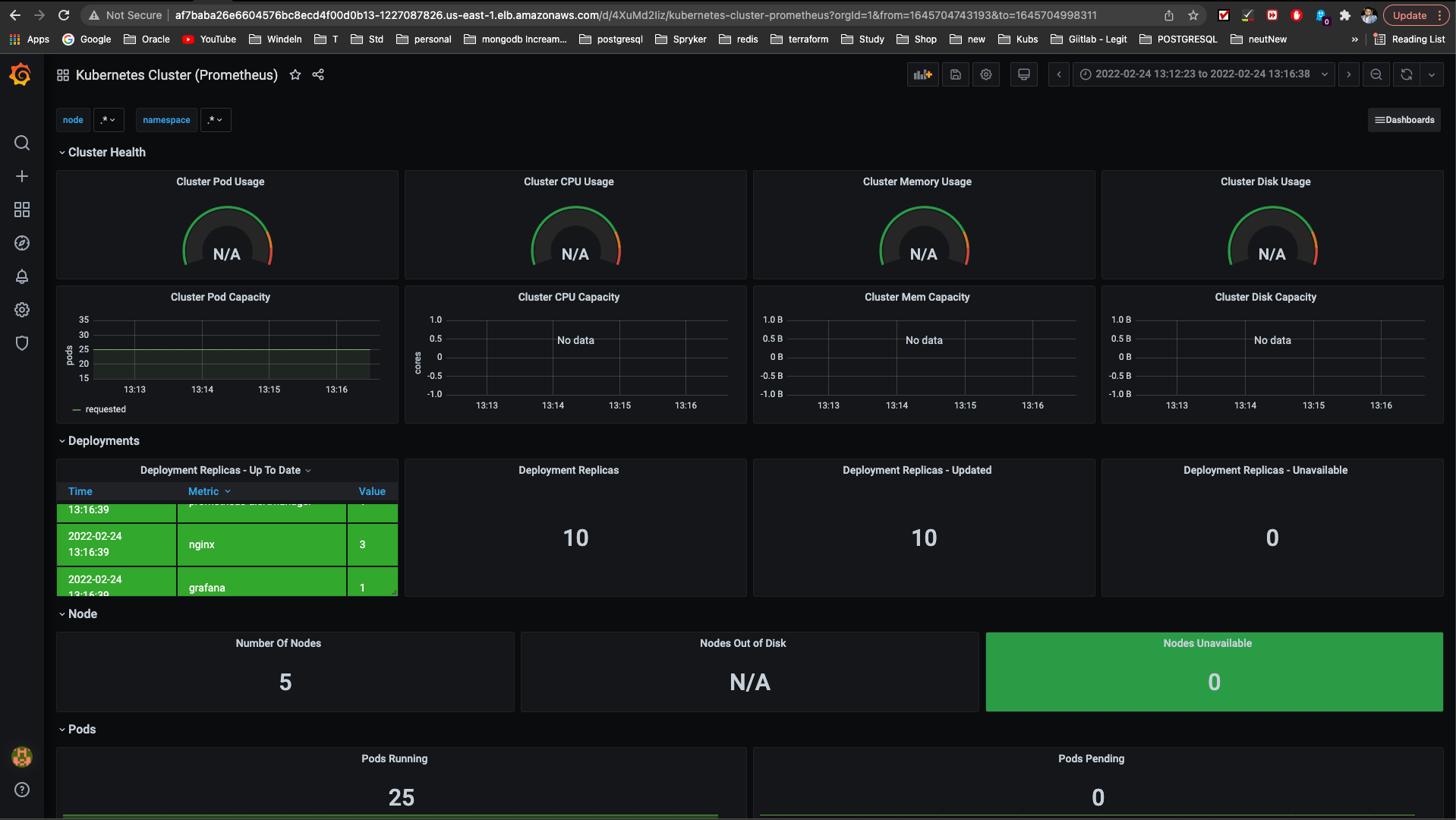
- Enter 3119 dashboard id under Grafana.com Dashboard.

- Click ‘Load’ - ‘Prometheus’ as the endpoint under prometheus data sources drop down - ‘Import’.



Pods Monitoring Dashboard

import > 6417 > Load > kubernetes pods monitring > change to uid > Prometheus > import



And final services I am running with following task shown below in the screen shot

