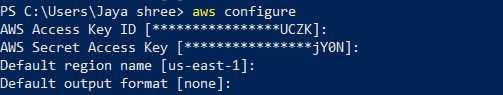
**Stepwise approach to deploy a wordpress application using a docker image and monitor the same using prometheus and Grafana.**

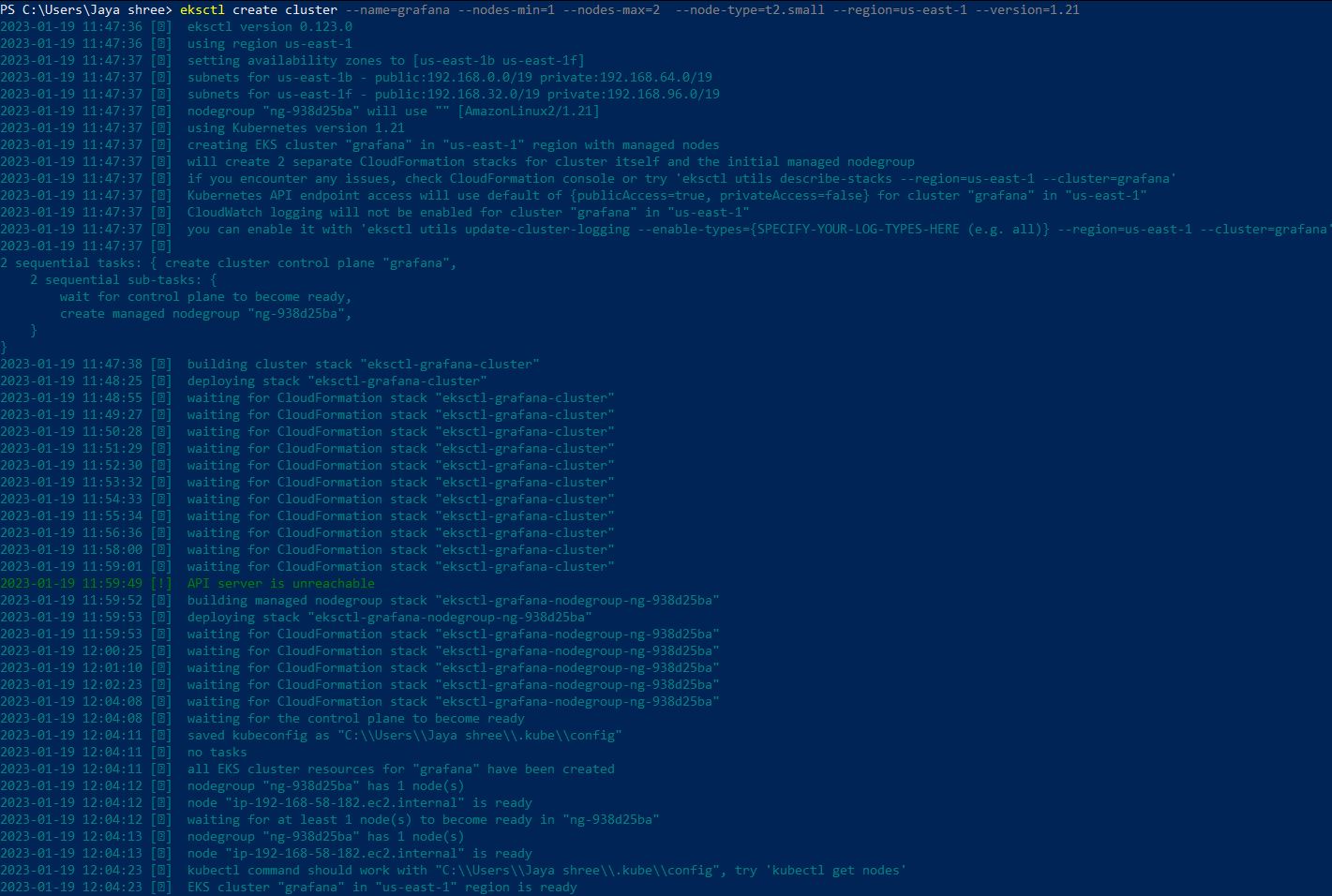
1. Firstly, we will configure our powershell with the aws account that we are using.

Command- **aws configure**

****

1. Now, we will create the cluster as per the requirement, For this time I have used the following command for creating the cluster.

Command- **eksctl create cluster –name=grafana –nodes-min=1 –nodes-max=2 –node-type=t2.small –region=us-east-1 –version=1.21**

****

(This Process will take out most 30 min for creation of the deployment)

1. We will now check if the nodes are running and up

Command- **kubectl get nodes**

****

1. Firstly we will create a namespace where will keep the deployment yaml file

Command - **kubectl create namespace testing**



1. Now We will deploy the wordpress application. We will use deployment.yaml file here under the namepace testing.

Command- **kubectl apply -f deployment.yml -n testing**



Code of deployment.yml:

apiVersion: apps/v1

kind: Deployment

metadata:

name: server-demo

spec:

replicas: 1

selector:

matchLabels:

app: web

template:

metadata:

labels:

app: web

spec:

containers:

- name: back-end

image: 655682474236.dkr.ecr.us-east-1.amazonaws.com/wordpress:latest

ports:

- containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

name: backend-service

spec:

type: NodePort

selector:

app: web

ports:

- nodePort: 31479

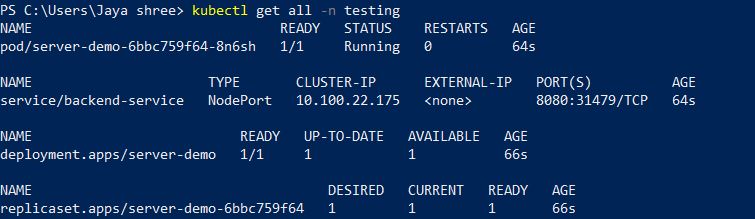
port: 8080

targetPort: 80

(Here we are using the ECR image in the file which we can get through the aws console.)

1. We will now check the running status of the deployed pods of the application.

Command - **kubectl get deployments**



1. To get the external ip address so that we can connect to it.

Command - **kubectl get nodes -o wide**

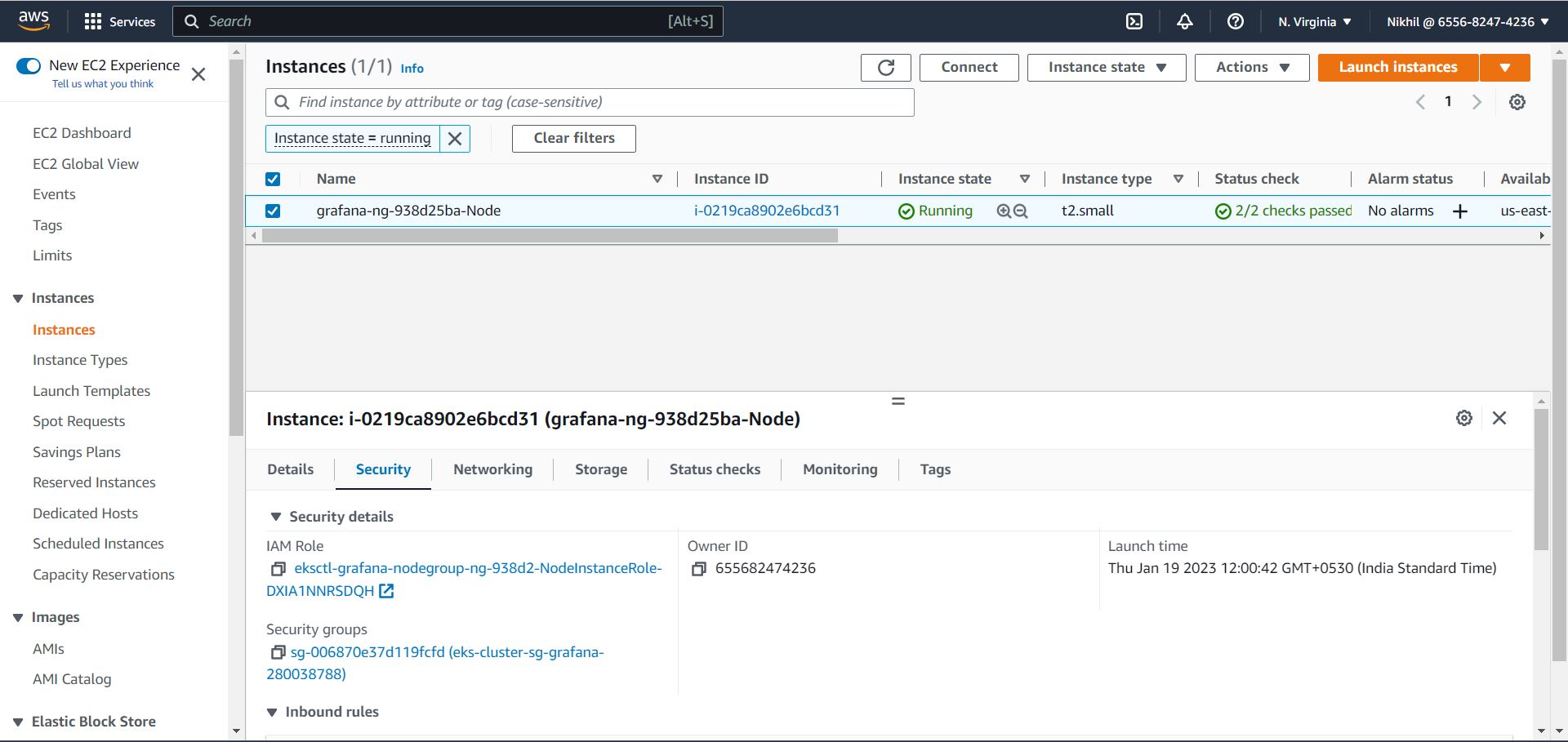


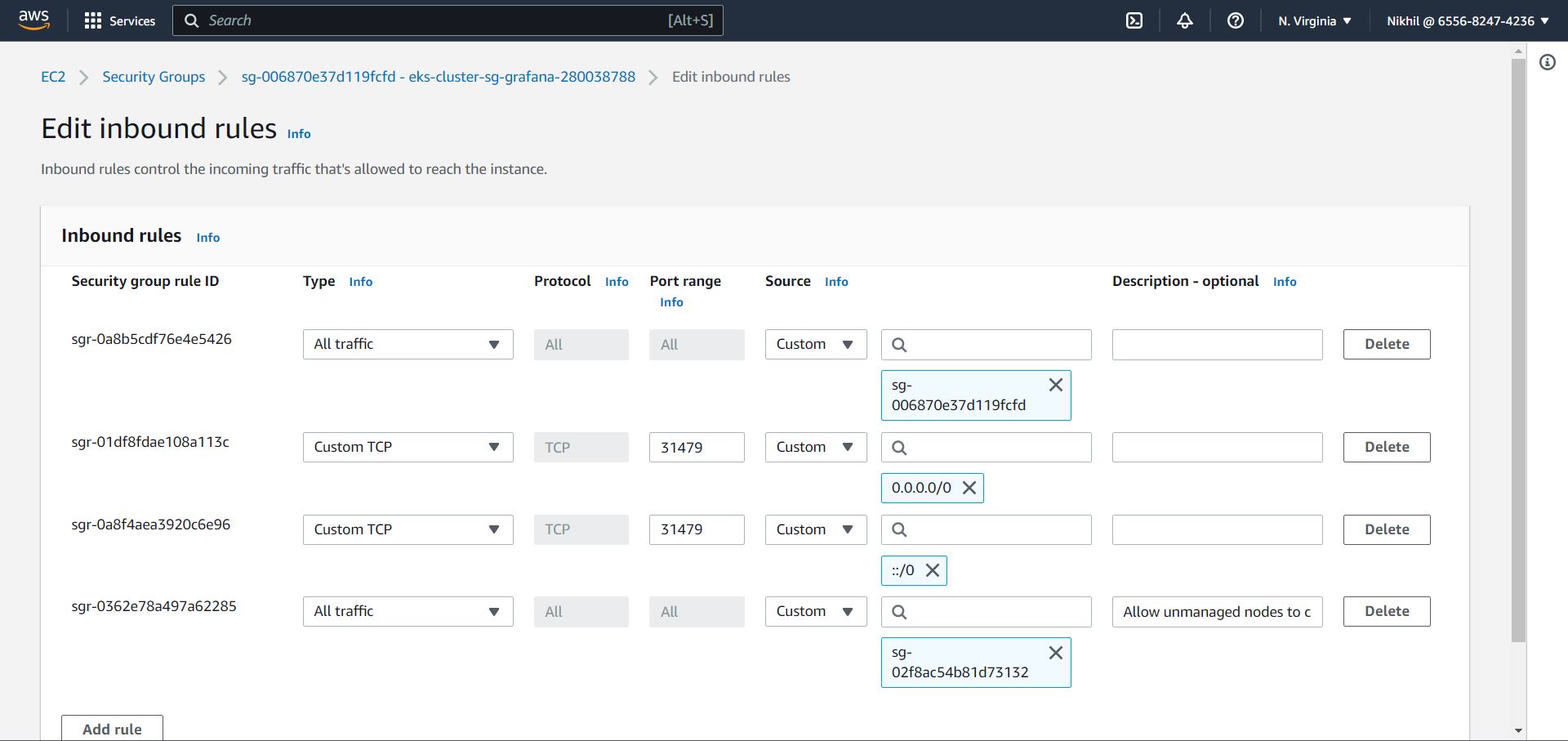
1. For Connecting the Nodeport to public Workergroup. For this time there is only one instance.

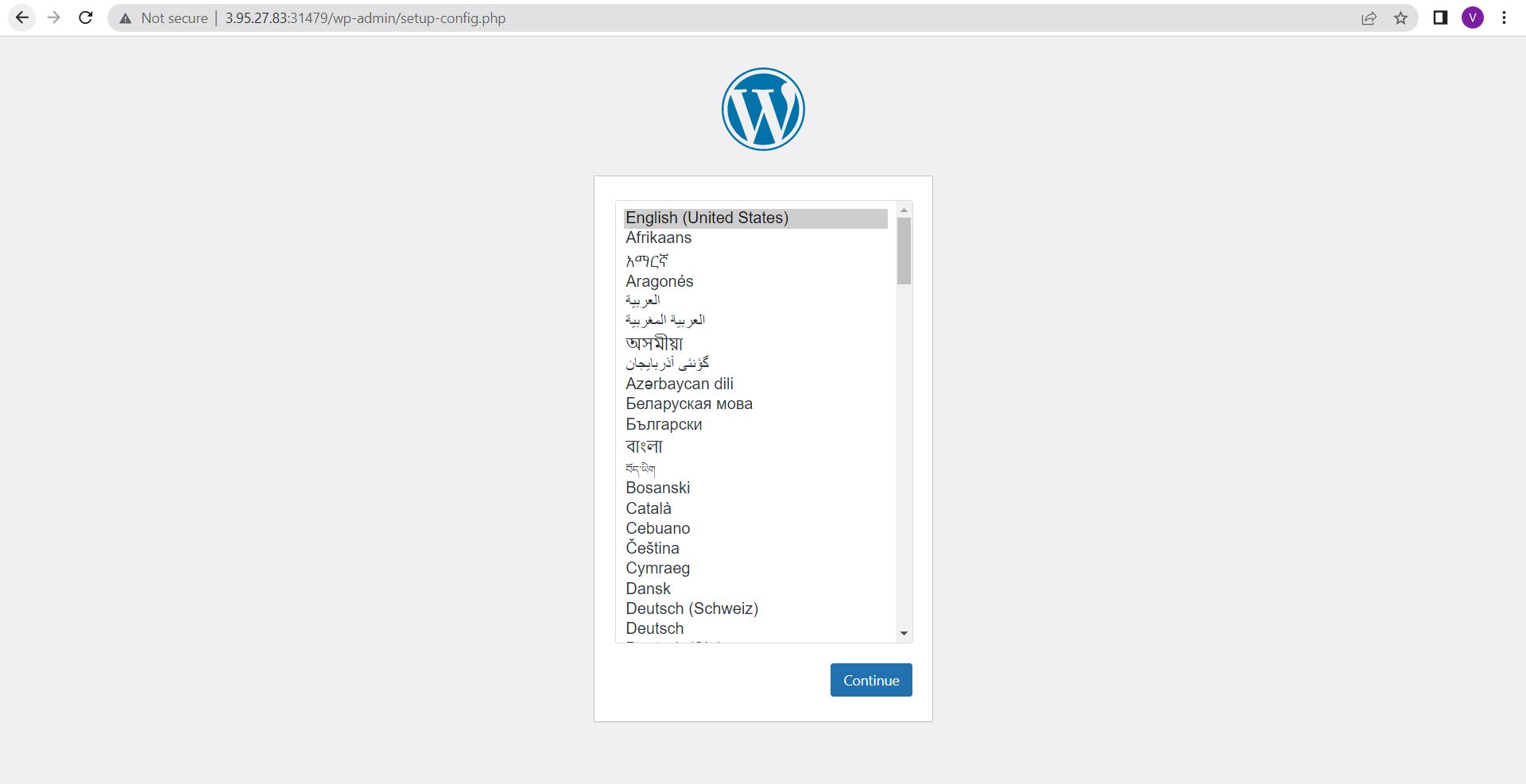
Firstly, we will go in the **aws console**. Then go **EC2** instance, select the instance, after that go in the **security group**, select the **public worker group**.

We need open the **inbound rules** so as to connect.

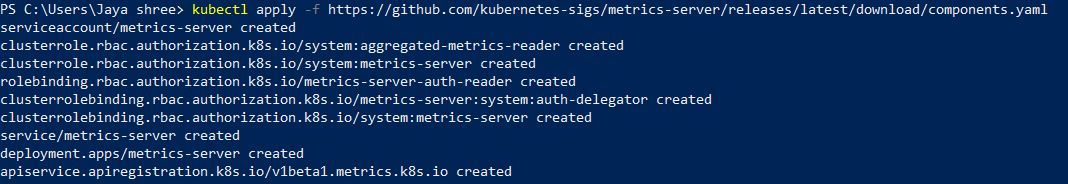
Click on **edit inbound rules**, Select custom ip,TCP, **port range=31479**, IPv4 and IPv6.





1. Now we will use the external ip and the nodeport provided by us to connect to the wordpress application.
2. Now, we will deploy the Kubernetes metric server which will be used by the prometheus server later for collecting the data

Command - **kubectl apply -f** [**https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml**](https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml)



1. We will Check weather the metric server got deployed or not.

Command - **kubectl get deployment metrics-server -n kube-system**



1. Now we will create namespace for prometheus

Command - **kubectl create namespace prometheus**



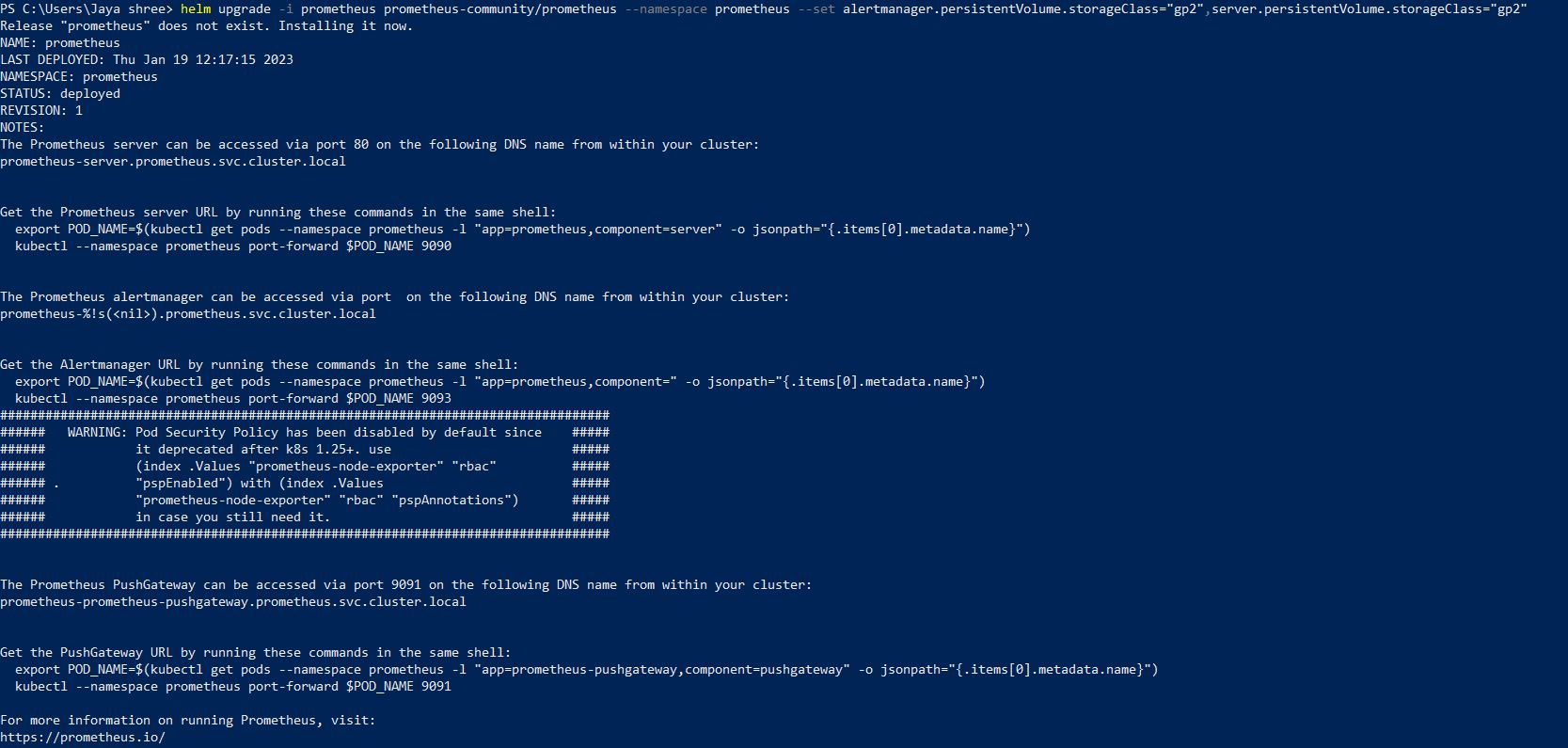
1. Firstly we will add prometheus-community chart repository

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



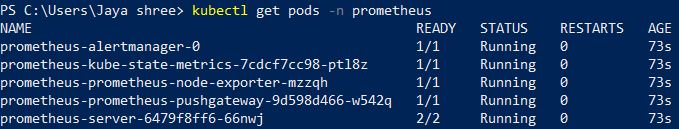
1. Using Helm charts we will deploy the prometheus monitoring tool

Command - **helm upgrade -i prometheus prometheus-community/prometheus --namespace prometheus --set alertmanager.persistentVolume.storageClass="gp2",server.persistentVolume.storageClass="gp2"**

****

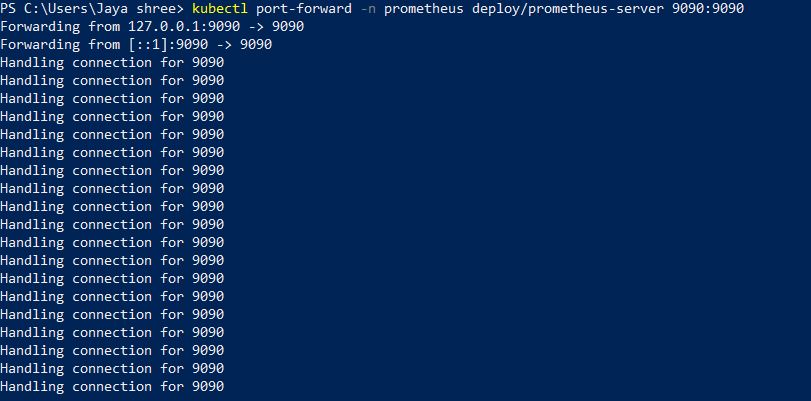
1. Check for the pods of prometheus are running or not.

Command - **kubectl get pods -n prometheus**

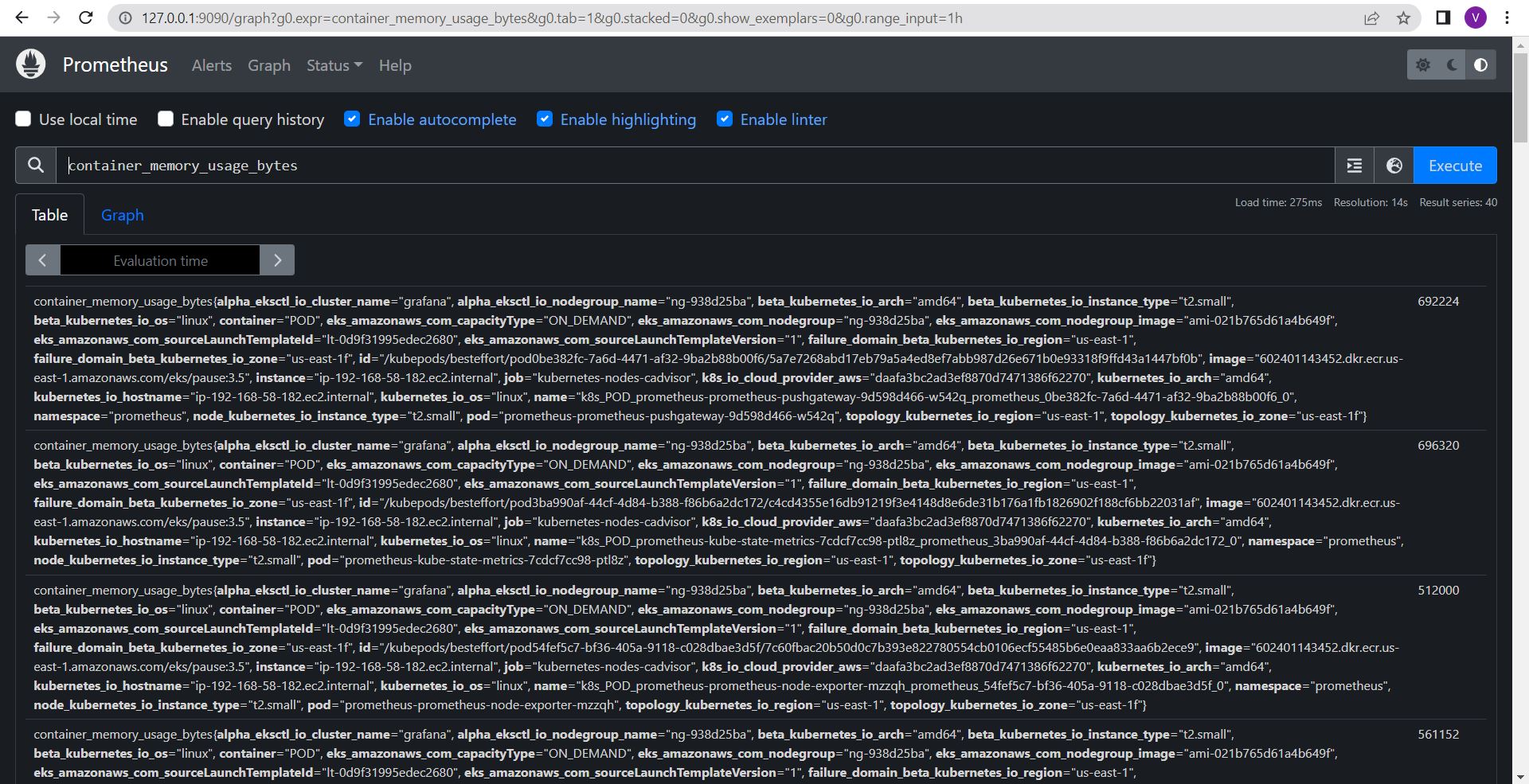


1. After this for accessing the prometheus server we will port forward the same to 9090 port.

Command - **kubectl port-forward -n prometheus deploy/prometheus-server 9090:9090**



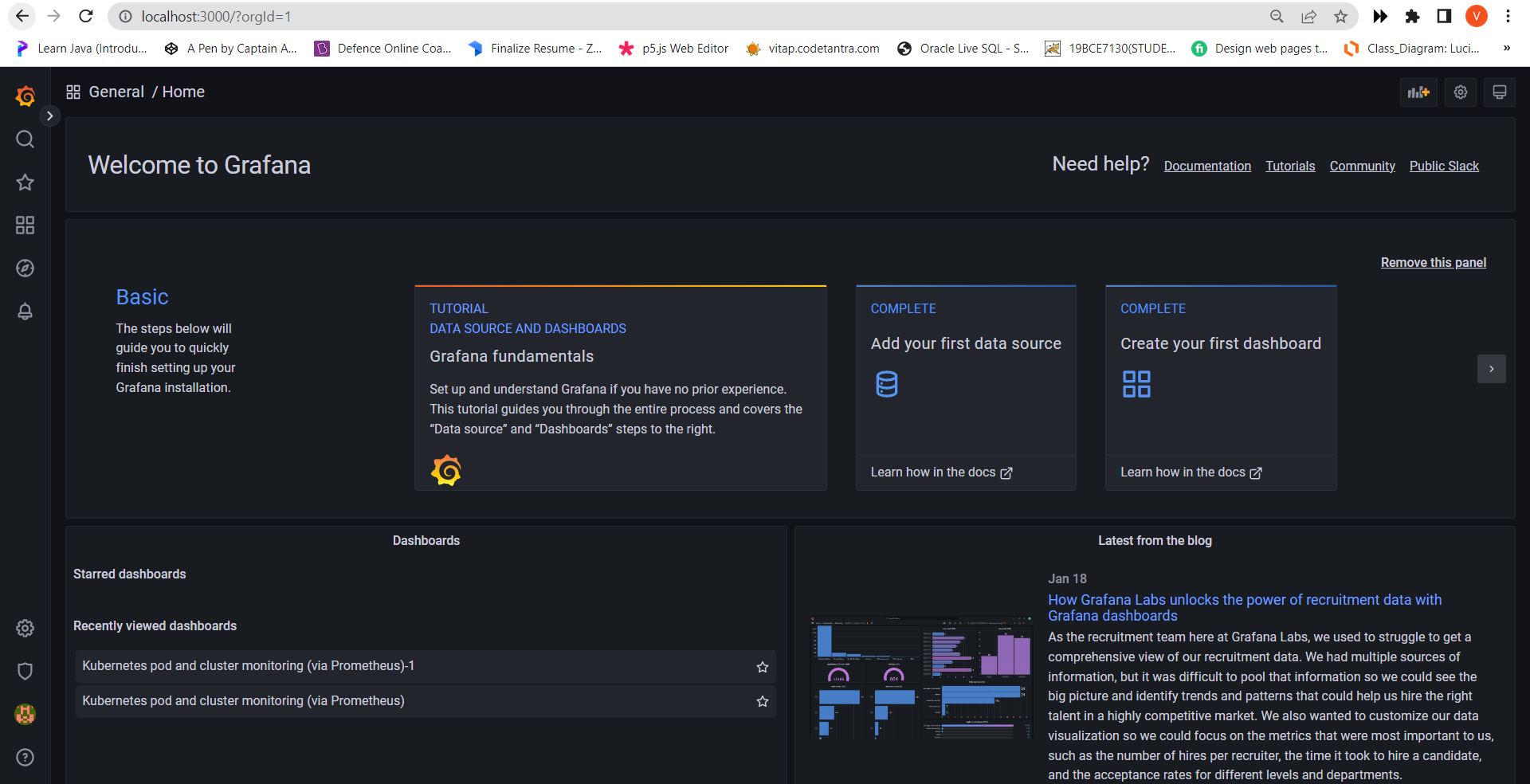
1. Check for the deployment of prometheus server on **127.0.0.1:9090**



1. Now we will use the grafana for converting the prometheus log into grafana visuals.

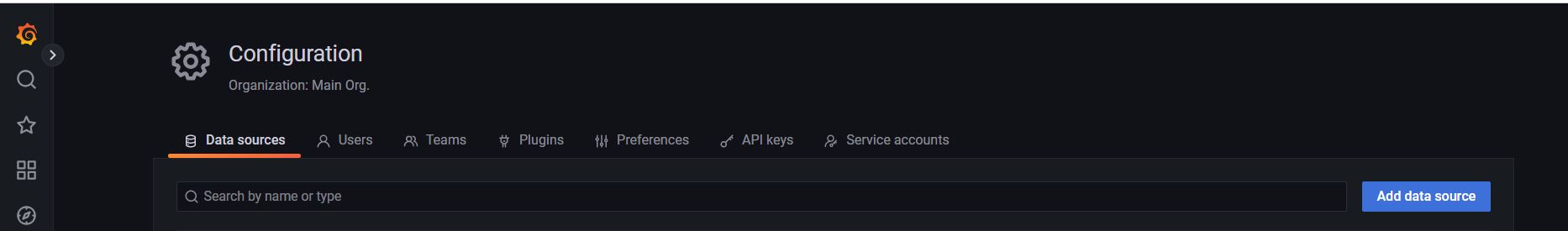
Install the windows on your local from the grafana website.

Later go to **localhost:3000** for accessing the same



1. Now we configure grafana so as to connect with the prometheus server.

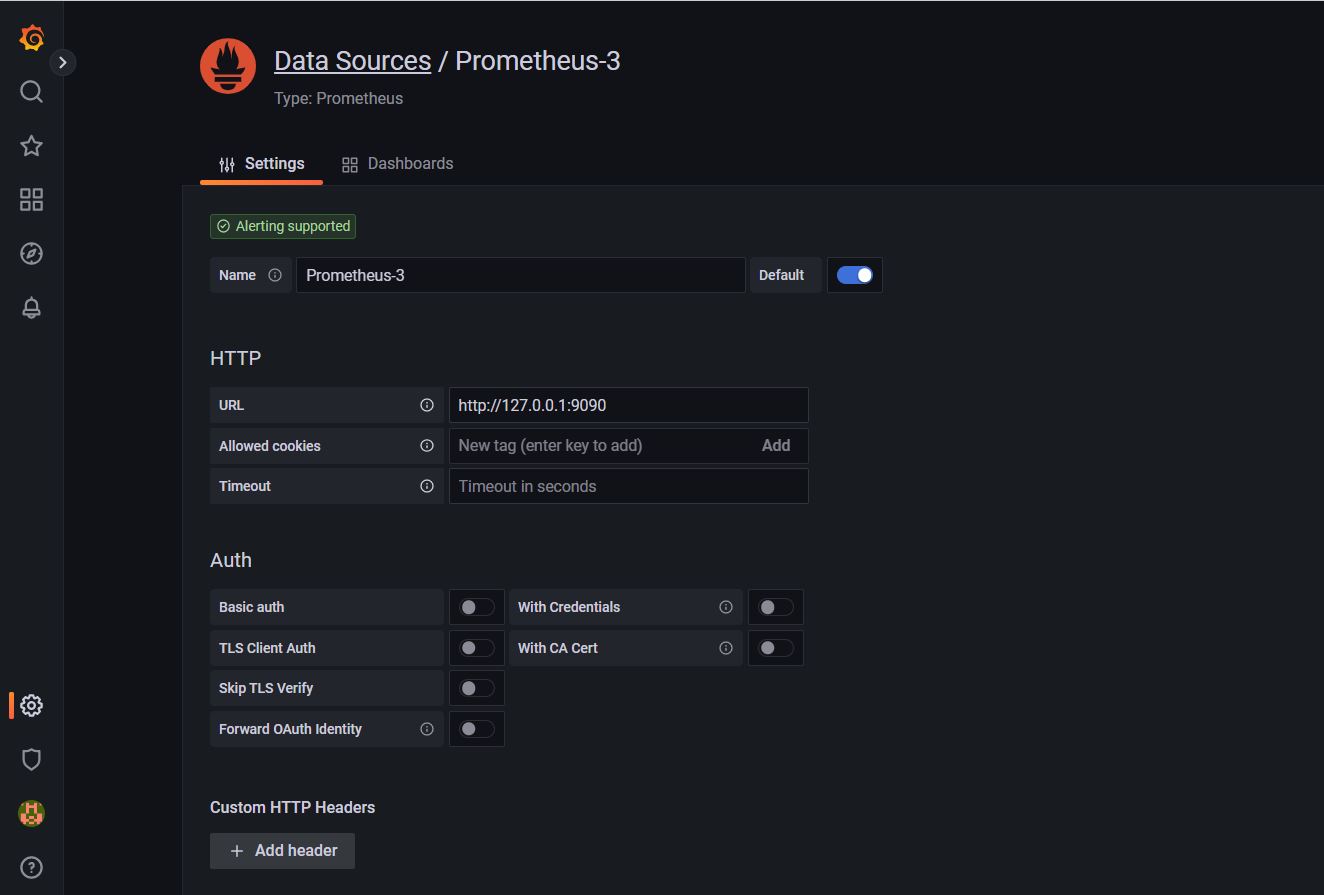
Firstly we will **go to settings -> configure**



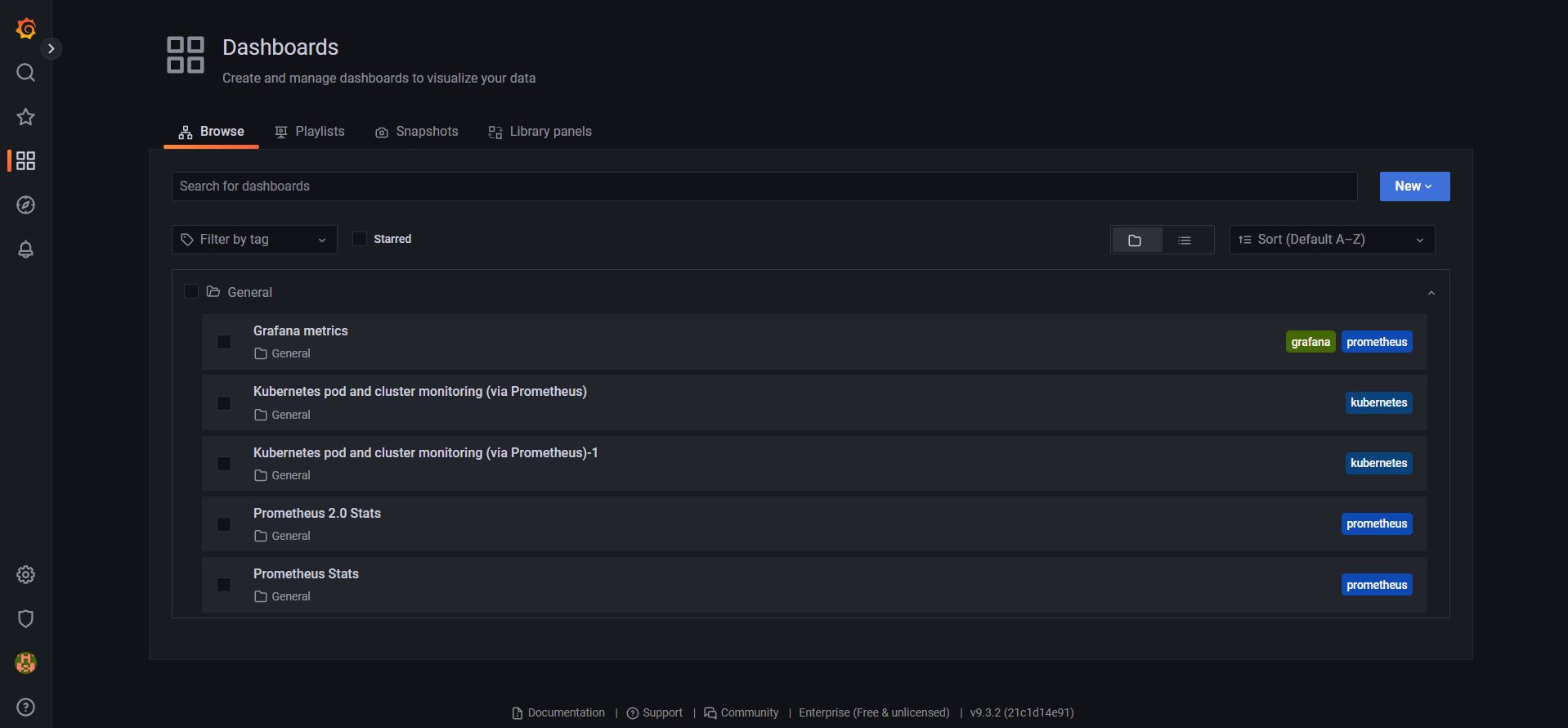
1. **Configure -> datasource -> prometheus ->add URL->select as default->click save & test**

Fill up the details like for URL we will use the URL that we used to check weather we access the prometheus server that is 127.0.0.1:9090

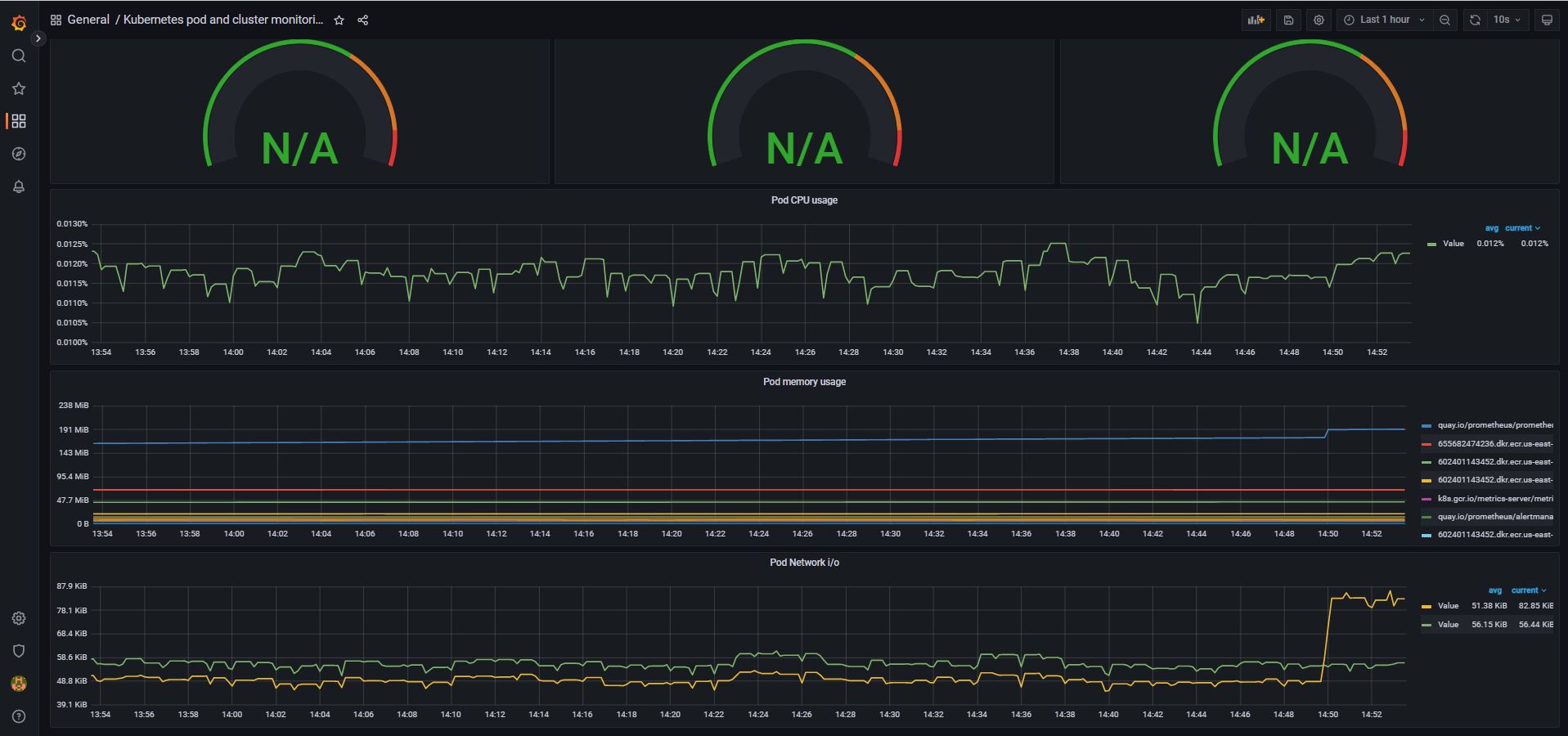
click save and test.



1. **Go to Dashboard -> select the dashboard from the general menu**



1. Now we will check weather the monitoring of cluster and pod is working properly or not



1. After so, we will delete the cluster so as to avoid the extra cost.

Command - **eksctl delete cluster –region=us-east-1 –name grafana**

