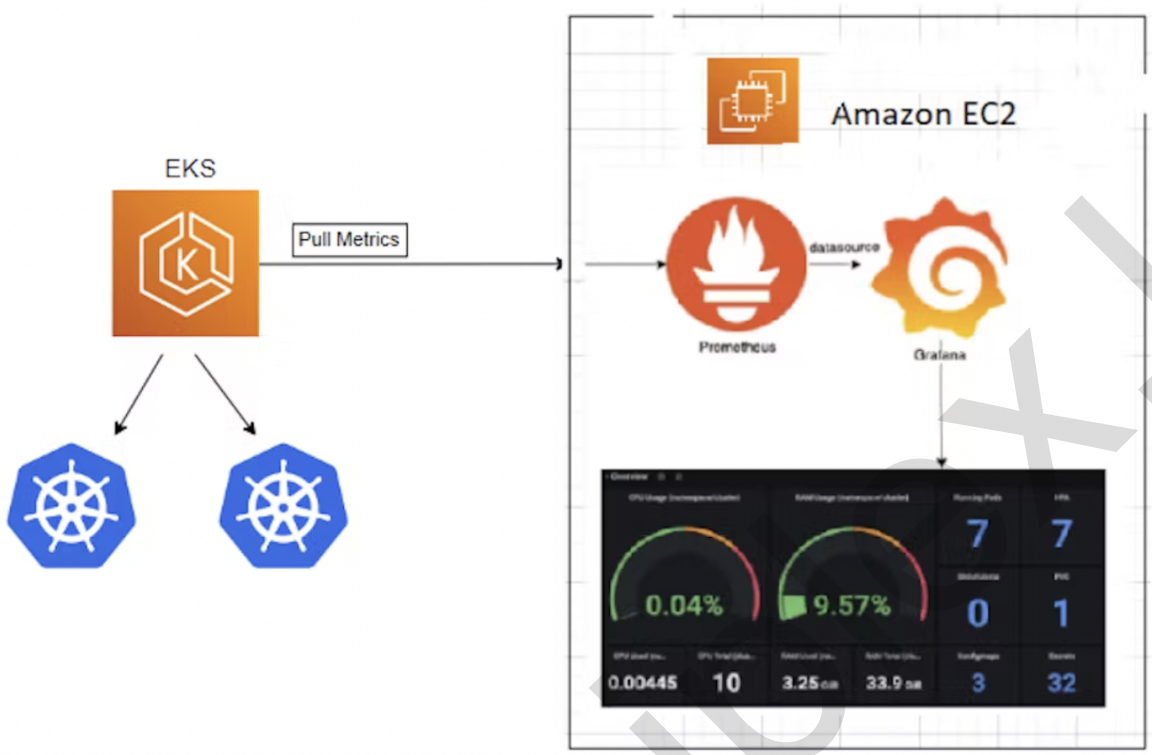
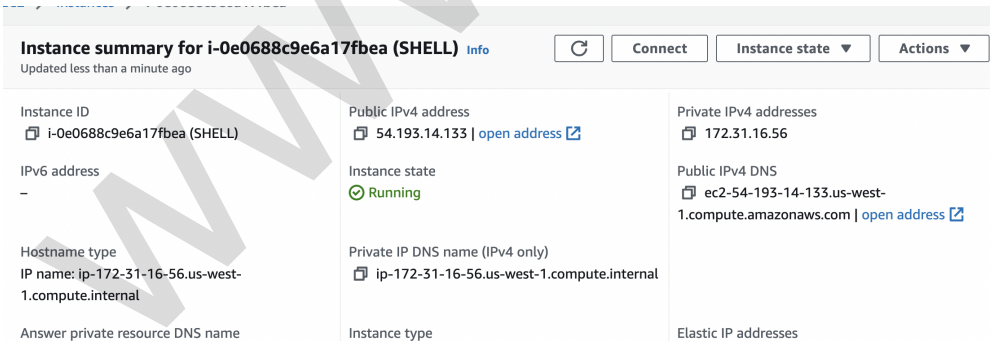
REAL TIME MONITORING HANDSON

Prometheus and Grafana Dashboard on EKS Cluster using Helm Chart



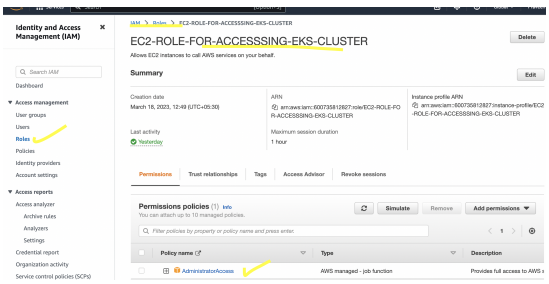
Step 1 – Setup EC2 Instance

Instance Type as t2.medium
AMIs as Ubuntu
US-EAST-1



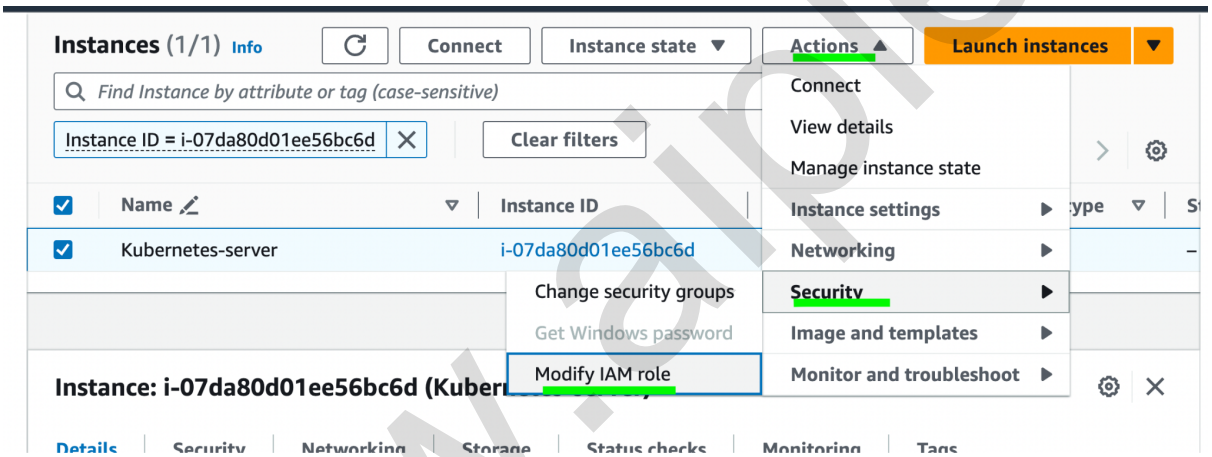
Step 1.1 – Create the IAM role having full access

Go to IAM -> Create role -> Select EC2 -> Give Full admin access



Step 1.2 – Attach the IAM role having full access

Go to EC2 -> Click on Actions on the left hand side -> Security -> Modify IAM role



Step 2 - Install AWS CLI and Configure

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"

sudo apt install unzip
unzip awscliv2.zip
sudo ./aws/install

Step 3 - Install and Setup Kubectl

curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl

chmod +x ./kubectl

sudo mv ./kubectl /usr/local/bin

kubectl version

kubectl version --short

Step 4 - Install and Setup eksctl

curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp

sudo mv /tmp/eksctl /usr/local/bin

eksctl version

Step 5 - Install Helm chart

curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3

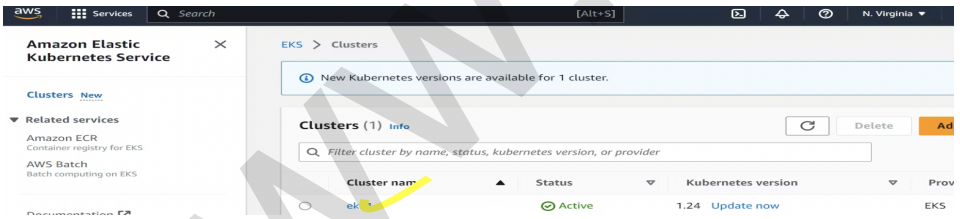
chmod 700 get_helm.sh
./get_helm.sh

helm version

Step 6 - Creating an Amazon EKS cluster using eksctl

1. **Name of the cluster :** --eks2
2. **Version of Kubernetes :** --version 1.24
3. **Region :** --region us-east-1
4. **Nodegroup name/worker nodes :** --nodegroup-name worker-nodes
5. **Node Type :** --nodegroup-type t2.medium
6. **Number of nodes:** --nodes 2
7. **Minimum Number of nodes:** --nodes-min 2
8. **Maximum Number of nodes:** --nodes-max 3

eksctl create cluster --name eks2 --version 1.24 --region us-east-1 --nodegroup-name worker-nodes --node-type t2.medium --nodes 2 --nodes-min 2 --nodes-max 3



Step 6.1 - IF ANY ERROR:

aws eks update-kubeconfig --region <region-code> --name <cluster-name>

Step 7 - Installing the Kubernetes Metrics Server

kubectl apply -f
https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml

Step 7.1 - Verify that the metrics-server deployment is running the desired number of pods with the following command

kubectl get deployment metrics-server -n kube-system

Step 8 - Install Prometheus

Now install the Prometheus using the helm chart.

Add Prometheus helm chart repository

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

Step 8.1 - Update helm chart repository

helm repo update
helm repo list

Step 8.2 Create prometheus namespace

kubectl create namespace prometheus

Step 8.3 - Install Prometheus

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

```
    --namespace prometheus \
    --set alertmanager.persistentVolume.storageClass="gp2" \
    --set server.persistentVolume.storageClass="gp2"
NAME: prometheus
LAST DEPLOYED: Wed Mar 22 08:05:28 2023
NAMESPACE: prometheus
STATUS: deployed
REVISION: 1
NOTES:
The Prometheus server can be accessed via port 80 on the following DNS name from within your cluster:
prometheus-server.prometheus.svc.cluster.local


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].metadat
  kubectl --namespace prometheus port-forward $POD_NAME 9090


The Prometheus alertmanager can be accessed via port  on the following DNS name from within your cluster:
```

Step 9 - Create IAM OIDC Provider

Your cluster has an OpenID Connect (OIDC) issuer URL associated with it. To use AWS Identity and Access Management (IAM) roles for service accounts, an IAM OIDC provider must exist for your cluster's OIDC issuer URL.

oidc_id=$(aws eks describe-cluster --name eks2 --region us-east-1 --query "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)

aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4

eksctl utils associate-iam-oidc-provider --cluster eks2 --approve --region us-east-1

```
5-aa875f532a01, ResourceNotFoundException: No cluster found for name: eks2.
root@ip-172-31-16-112:/home/ubuntu# eksctl utils associate-iam-oidc-provider --cluster eks2 --approve --region us-east-1
2023-03-31 13:23:09 [ℹ]  will create IAM Open ID Connect provider for cluster "eks2" in "us-east-1"
2023-03-31 13:23:10 [✔]  created IAM Open ID Connect provider for cluster "eks2" in "us-east-1"
```

Step 10 – Create iamserviceaccount with role

eksctl create iamserviceaccount --name ebs-csi-controller-sa --namespace kube-system --cluster eks2 --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy --approve --role-only --role-name AmazonEKS_EBS_CSI_DriverRole --region us-east-1

```
root@ip-172-31-16-112:/home/ubuntu# eksctl create iamserviceaccount \
  --name ebs-csi-controller-sa \
  --namespace kube-system \
  --cluster eks2 \
  --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \
  --approve \
  --role-only \
  --role-name AmazonEKS_EBS_CSI_DriverRole \
--region us-east-1
2023-03-31 13:25:27 [ℹ]  1 iamserviceaccount (kube-system/ebs-csi-controller-sa) was included (based on the include/exclude rules)
2023-03-31 13:25:27 [!]  serviceaccounts in Kubernetes will not be created or modified, since the option --role-only is used
2023-03-31 13:25:27 [ℹ]  1 task: { create IAM role for serviceaccount "kube-system/ebs-csi-controller-sa" }
2023-03-31 13:25:27 [ℹ]  building iamserviceaccount stack "eksctl-eks2-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa"
2023-03-31 13:25:27 [ℹ]  deploying stack "eksctl-eks2-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa"
2023-03-31 13:25:27 [ℹ]  waiting for CloudFormation stack "eksctl-eks2-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa"
2023-03-31 13:25:57 [ℹ]  waiting for CloudFormation stack "eksctl-eks2-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa"
```

Step 10.1 - Then attach ROLE to eks by running the following command

Enter your account ID and cluster name.

eksctl create addon --name aws-ebs-csi-driver --cluster eks2 --service-account-role-arn arn:aws:iam::164297528770:role/AmazonEKS_EBS_CSI_DriverRole --force --region us-east-1

```
root@ip-172-31-16-112:/home/ubuntu# eksctl create addon --name aws-ebs-csi-driver --cluster eks2 --service-account-role-arn arn:aws:iam::600735812827:role/Ama
zonEKS_EBS_CSI_DriverRole --force --region us-east-1
2023-03-31 13:29:15 [ℹ]  Kubernetes version "1.24" in use by cluster "eks2"
2023-03-31 13:29:15 [ℹ]  using provided ServiceAccountRoleARN "arn:aws:iam::600735812827:role/AmazonEKS_EBS_CSI_DriverRole"
2023-03-31 13:29:15 [ℹ]  creating addon
```

Step 10.2 - kubectl get pods -n prometheus

```
:~$ kubectl get all -n prometheus
NAME                                                READY   STATUS    RESTARTS   AGE
pod/prometheus-alertmanager-0                       1/1     Running   0          5m57s
pod/prometheus-kube-state-metrics-6fcf5978bf-vs8mn  1/1     Running   0          5m57s
pod/prometheus-prometheus-node-exporter-qmqnw       1/1     Running   0          5m57s
pod/prometheus-prometheus-node-exporter-qwsdk       1/1     Running   0          5m57s
pod/prometheus-prometheus-pushgateway-fdb75d75f-x4z4z   1/1     Running   0          5m57s
pod/prometheus-server-646ccf9b67-2rl7m              2/2     Running   0          5m57s

NAME                                          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
service/prometheus-alertmanager               ClusterIP   10.100.252.20    <none>        9093/TCP   5m57s
service/prometheus-alertmanager-headless      ClusterIP   None             <none>        9093/TCP   5m57s
service/prometheus-kube-state-metrics         ClusterIP   10.100.78.81     <none>        8080/TCP   5m57s
service/prometheus-prometheus-node-exporter   ClusterIP   10.100.140.104   <none>        9100/TCP   5m57s
service/prometheus-prometheus-pushgateway     ClusterIP   10.100.103.210   <none>        9091/TCP   5m57s
service/prometheus-server                     ClusterIP   10.100.151.93    <none>        80/TCP     5m57s

NAME                                                DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
daemonset.apps/prometheus-prometheus-node-exporter  2         2         2       2            2           <none>          5m57s
```

Step 10.3 - View the Prometheus dashboard by forwarding the deployment ports

kubectl port-forward deployment/prometheus-server 9090:9090 -n prometheus

Step 10.4 - Open different browser and connect to your EC2 instance and run curl localhost:9090/graph

Step 11 - Install Grafana

helm repo add grafana https://grafana.github.io/helm-charts

helm repo update

Step 11.1 - Create a namespace Grafana

kubectl create namespace grafana

```
root@ip-172-31-16-112:/home/ubuntu# vim prometheus-datasource.yaml
root@ip-172-31-16-112:/home/ubuntu# vim prometheus-datasource.yaml
root@ip-172-31-16-112:/home/ubuntu# kubectl create namespace grafana
namespace/grafana created
root@ip-172-31-16-112:/home/ubuntu#
```

Step 11.2 - Install the Grafana

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

This command will create the Grafana service with an external load balancer to get the public view.

Step 11.3 - Verify the Grafana installation by using the following kubectl command

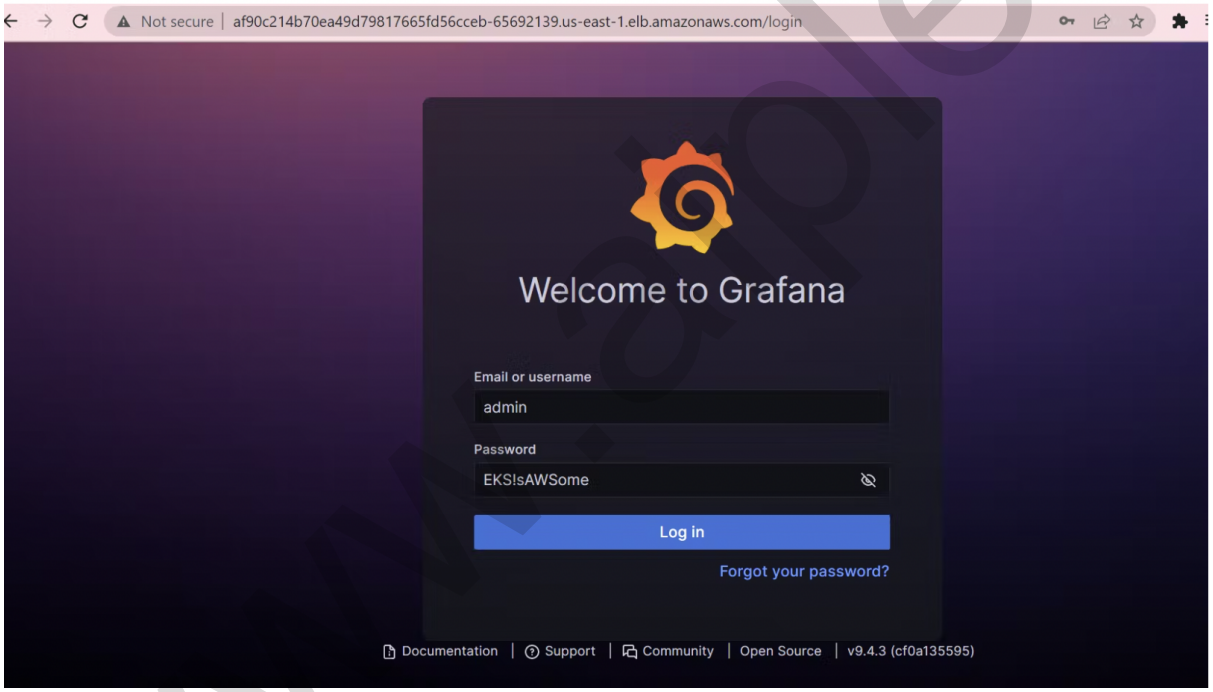
kubectl get pods -n grafana

kubectl get service -n grafana

```
[redacted]$ kubectl get all -n grafana
NAME                           READY   STATUS    RESTARTS   AGE
pod/grafana-64784d5b57-2d4wb   1/1     Running   0          62s

NAME              TYPE           CLUSTER-IP       EXTERNAL-IP                                                               PORT(S)        AGE
service/grafana   LoadBalancer   10.100.186.204   af90c214b70ea49d79817665fd56cceb-65692139.us-east-1.elb.amazonaws.com   80:30355/TCP   62s

NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/grafana   1/1     1            1           62s

NAME                                 DESIRED   CURRENT   READY   AGE
replicaset.apps/grafana-64784d5b57   1         1         1       62s
ubuntu@ip-172-31-22-24:~$ kubectl get service -n grafana
NAME      TYPE           CLUSTER-IP       EXTERNAL-IP                                                               PORT(S)        AGE
grafana   LoadBalancer   10.100.186.204   af90c214b70ea49d79817665fd56cceb-65692139.us-east-1.elb.amazonaws.com   80:30355/TCP   99s
[redacted]~$
```

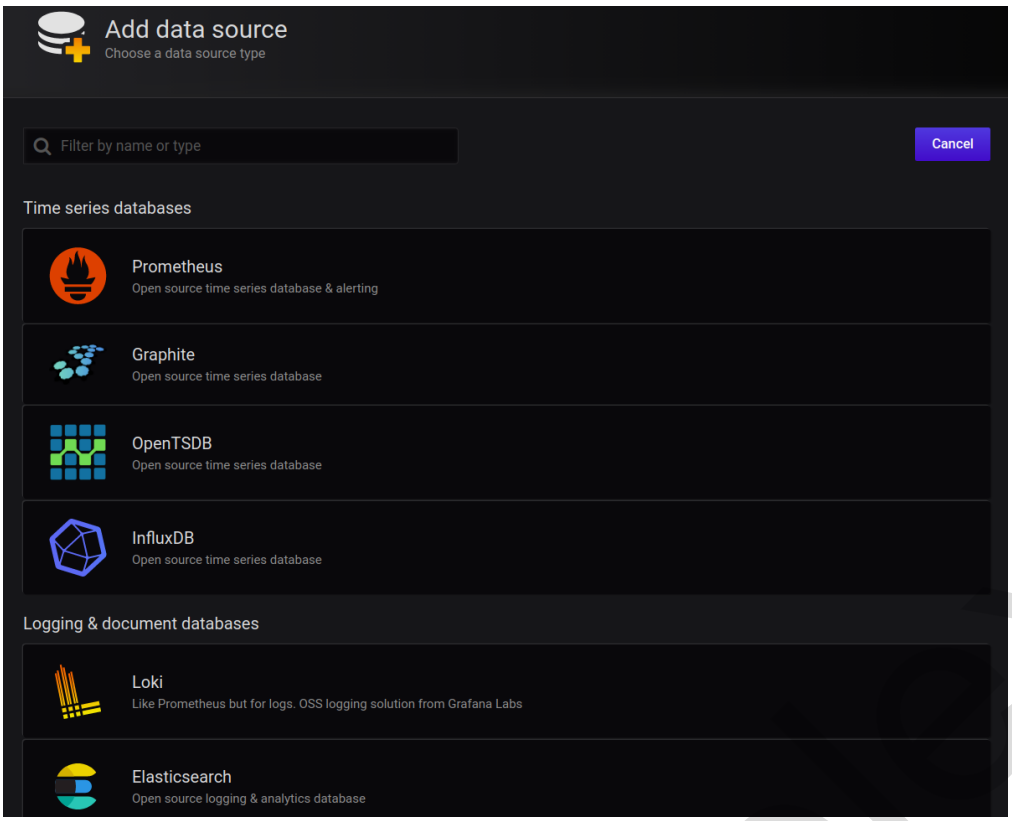
Step 11.4 - Copy the EXTERNAL-IP and paste in browser

Password you mentioned as EKS!sAWSome while creating Grafana



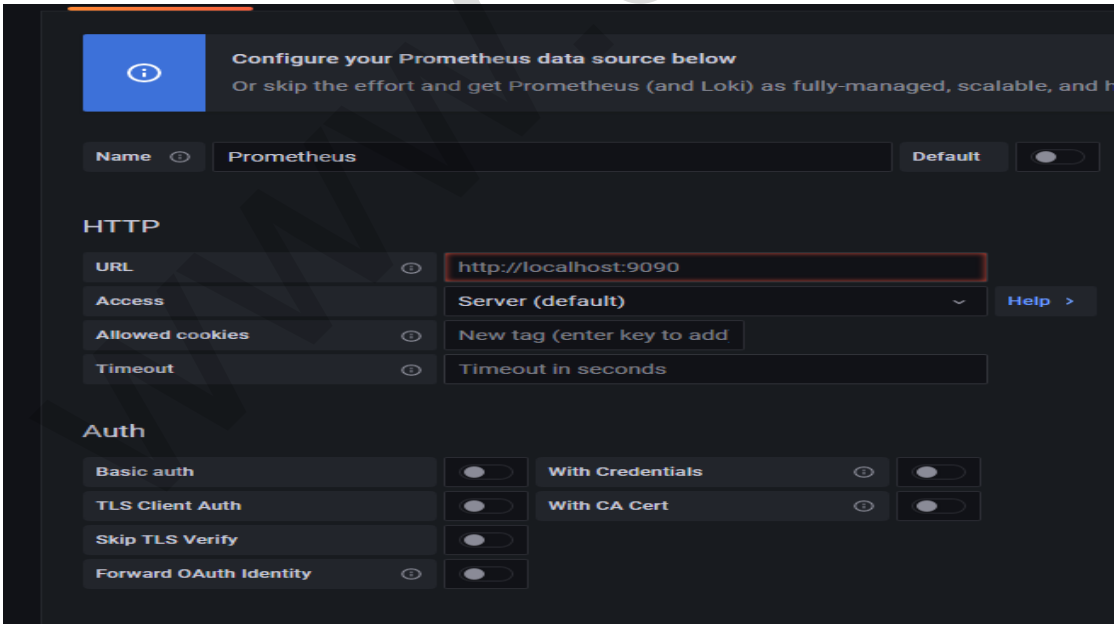
Step 11.5 - Add the Prometheus as the datasource to Grafana

Go to Grafana Dashboard -> Add the Datasource -> Select the Prometheus



Step 11.6 - Configure the endpoints of Prometheus and save

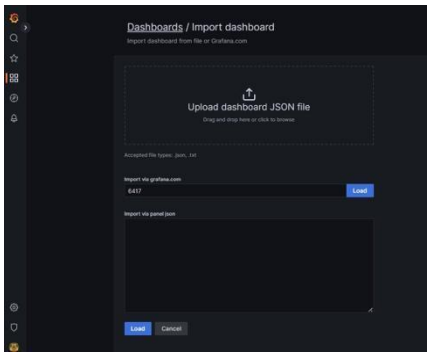
URL - http://prometheus-server.prometheus.svc.cluster.local



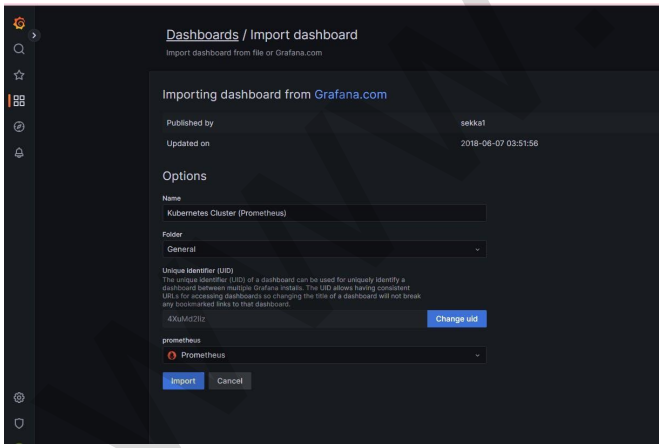
Step 11.6 - Import Grafana dashboard from Grafana Labs

Now we have set up everything in terms of Prometheus and Grafana. For the custom Grafana Dashboard, we are going to use the open source grafana dashboard. For this session, I am going to import a dashboard **6417**

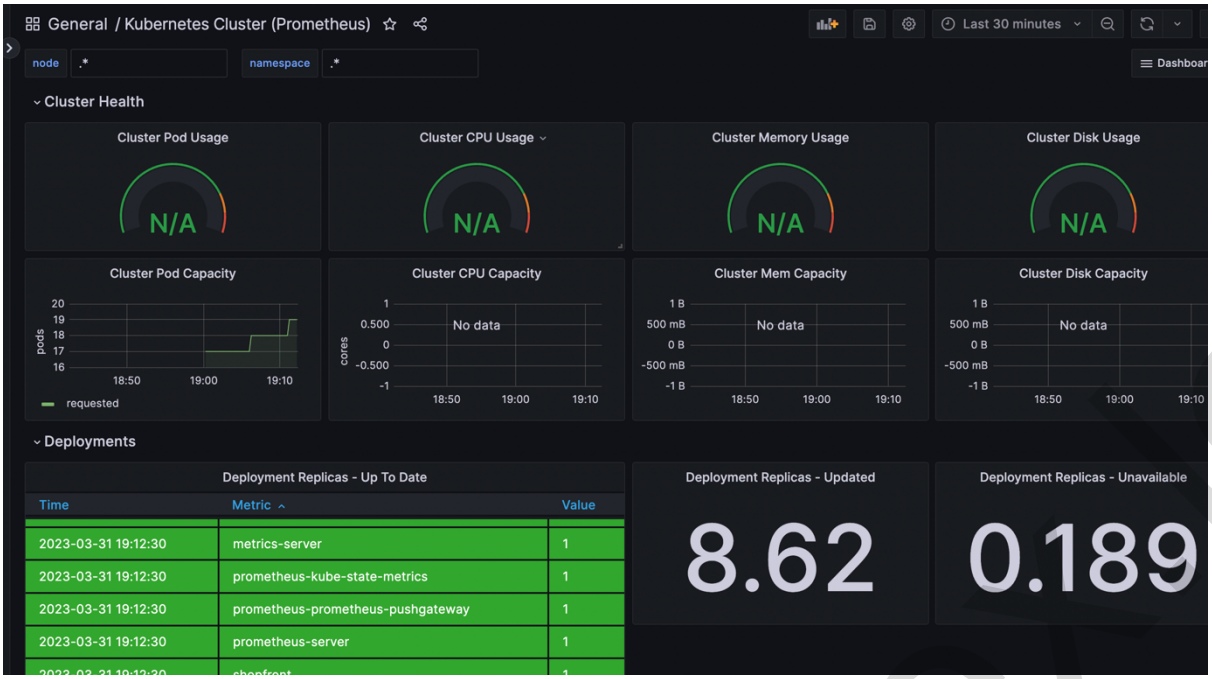
Go to left side -> click on dashboards -> Click on New -> Import



Load and select the source as Prometheus



Step 12 – Visualise the java application



Step 13 - Deploy a Java application and monitor it on Grafana

git clone
https://github.com/praveen1994dec/kubernetes_java_deployment.git

cd **/kubernetes_java_deployment/Kubernetes/**

kubectl apply -f shopfront-service.yaml
kubectl get deployment
kubectl get pods

kubectl logs shopfront-7868468c56-4r2kk -c shopfront

Step 14 - Clean Up

eksctl delete cluster --name eks2 --region us-east-1