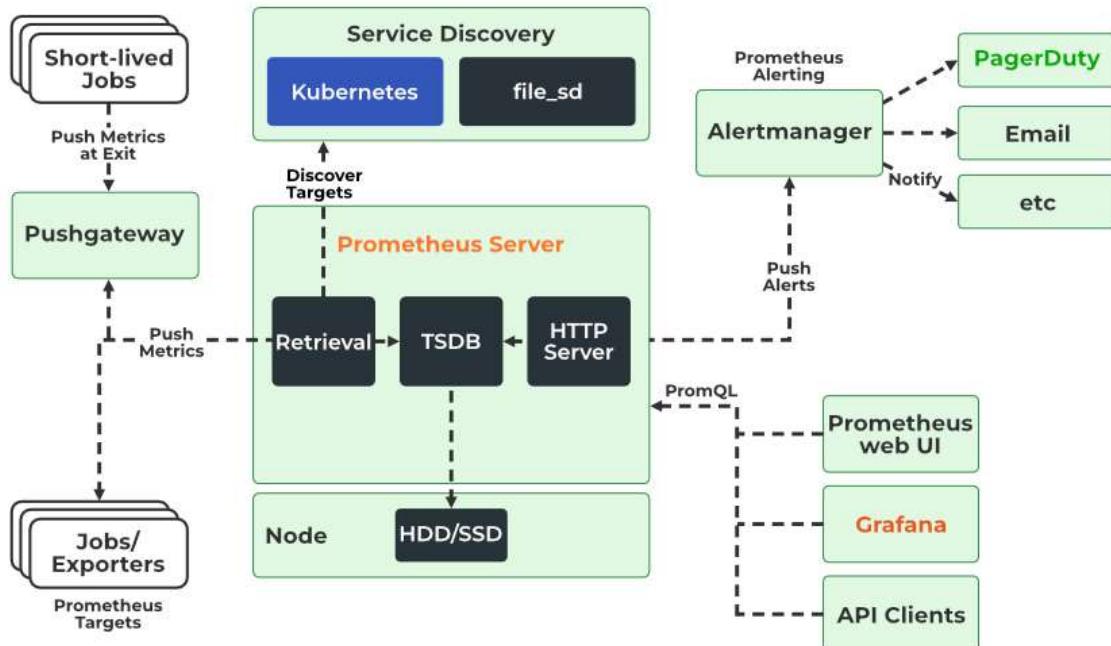
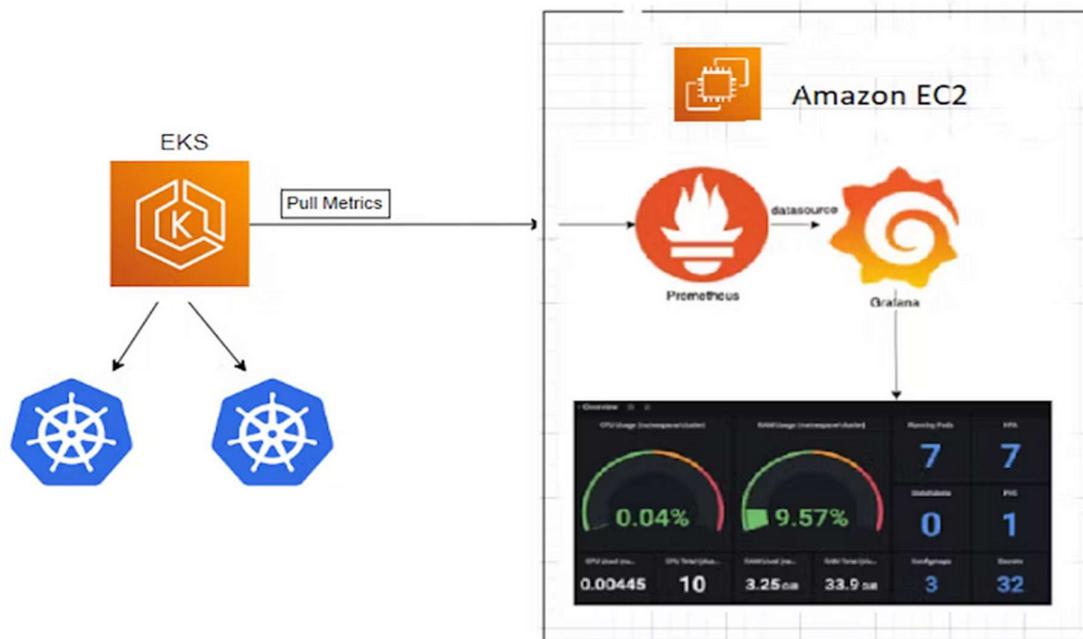


EKS monitoring using Helm, Prometheus and Grafana Dashboard



This blog explains how to set up Prometheus and Grafana in Amazon EKS.

1. What is Prometheus?

- Prometheus is an open-source monitoring tool
- Provides out-of-the-box monitoring capabilities for the Kubernetes container orchestration platform. It can monitor servers and databases as well.
- Collects and stores metrics as time-series data, recording information with a timestamp
- It is based on pull and collects metrics from targets by scraping metrics HTTP endpoints.

2. What is Grafana?

- Grafana is an open-source visualization and analytics software.
- It allows you to query, visualize, alert on, and explore your metrics no matter where they are stored.

3. Why use helm?

- Helm is a package manager for Kubernetes. Helm simplifies the installation of all components in one command. install using helm is recommended as you will not be missing any configuration steps and very efficient

Why do we need System Monitoring?

- To detect and prevent failures, it is very convenient to have a good monitoring tool that provides you with a solid monitoring system.
- Monitoring systems are responsible for controlling the technology used by companies in order to analyze their operation and performance.
- Understanding the state of your infrastructure and systems is essential for ensuring the reliability and stability of your services.
- Information about the health and performance of your deployments not only helps your team react to issues but also gives them the security to make changes with confidence.

Steps:-

1. Set up an AWS EC2 Instance and
2. Install AWS CLI and Configure AWS Credentials
3. Install and setup kubectl
4. Install and setup eksctl
5. Install Helm Charts
6. Creating an Amazon EKS cluster using eksctl
7. We need to add the Helm Stable Charts for your local client
8. Add Prometheus Helm repo
9. Create Prometheus namespace
10. Install Prometheus
11. Install Node JS application and monitor it on Grafana
12. Cleanup/Deprovision

Now, lets get started in detail

Step 1 — Set up an AWS T2 Medium Ubuntu EC2 Instance.

You can select an existing key pair, and enable HTTP and HTTPS Traffic. Launch the instance and once it is launched you can connect to it using the key pair. Name it as Main-Server

The screenshot shows the AWS EC2 Instances page. The instance summary for 'i-08f072f075b76fcf8 (EKS monitoring)' is displayed. Key details include:

- Instance ID: i-08f072f075b76fcf8 (EKS monitoring)
- Public IPv4 address: 13.229.225.215
- Instance state: Running
- Private IP DNS name (IPv4 only): ip-172-31-35-103.ap-southeast-1.compute.internal
- Instance type: t2.medium
- VPC ID: vpc-03c1ae9fa1980e9be
- Subnet ID: subnet-01a05c5bf69936566

The left sidebar shows navigation options like EC2 Dashboard, EC2 Global View, Events, Limits, Instances, Images, and Elastic Block Store.

Inbound rules (4)						
	Name	Security group rule...	IP version	Type	Protocol	Port range
<input type="checkbox"/>	-	sgr-0c745f696a28d921b	IPv4	HTTP	TCP	80
<input type="checkbox"/>	-	sgr-040ef851052f325ac	IPv4	SSH	TCP	22
<input type="checkbox"/>	-	sgr-080fc81c2b1acf373	IPv4	All traffic	All	All
<input type="checkbox"/>	-	sgr-0b29053bb0dbbb...	IPv4	HTTPS	TCP	443

Once it is launched, connect to the instance via console or using SSH Key Pair. Once you have connected to your EC2 Instance, you need to install and configure AWS CLI using the below command

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
sudo apt install unzip
unzip awscliv2.zip
sudo ./aws/install
aws --version
```

```
ubuntu@ip-172-31-35-103:~$ sudo ./aws/install
You can now run: /usr/local/bin/aws --version
ubuntu@ip-172-31-35-103:~$ aws --version
aws-cli/2.12.0 Python/3.11.3 Linux/5.19.0-1025-aws exe/x86_64/ubuntu.22 prompt/off
ubuntu@ip-172-31-35-103:~$ |
```

Step 2— Now, let's configure AWS Credentials so that it can authenticate and communicate with AWS Environment.
Here are the instructions for fetching the AWS credentials -

A) To set up the AWS credentials you first need to fetch them from your AWS account.

1. Goto your AWS account
2. Then Goto Security Credentials
3. Click on Access Keys
4. Then *Create New Access Key*

Access keys (1)					
Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. Learn more [L]					
Actions		Create access key			
Access key ID	Created on	Access key last used	Region last used	Service last used	Status
AKIA3GZODAO7GS7QFHL5	21 hours ago	2 hours ago	ap-southeast-1	cloudformation	Active

Creta IAM Rol with administration access

The screenshot shows the AWS IAM console. On the left, there's a sidebar with navigation links like Dashboard, Access management (Users, Roles, Policies, Identity providers, Account settings), and Access reports (Access analyzer, Archive rules, Analyzers, Settings, Credential report, Organization activity). The main area is titled 'Permissions policies (1)'. It shows a single policy named 'AdministratorAccess' which is an 'AWS managed - job function' attached directly to the user. There are sections for 'Permissions boundary (not set)' and 'Generate policy based on CloudTrail events'.

Go to AWS console, click on EC2, select EC2 instance, Choose Security. Click on Modify IAM Role

The screenshot shows the AWS EC2 Instances page. It lists one instance named 'EKS monitoring' with the ID 'i-08f072f075b76fcf8', which is running. On the right, there's a context menu for the instance, with 'Security' selected. Below it, the 'Modify IAM role' dialog is open. It shows the instance ID 'i-08f072f075b76fcf8 (EKS monitoring)' and a dropdown menu where 'eks-administration-role' is selected. At the bottom, there are 'Cancel' and 'Update IAM role' buttons.

Setup the AWS Credentials -

1. Open terminal/command prompt
2. Run the command **aws configure** and then enter **AWS Access key ID** and **AWS secret access key**

```
ubuntu@ip-172-31-35-103:~$ aws configure
AWS Access Key ID [None]: AWSAccessKeyID
AWS Secret Access Key [None]: AWSSecretKey
Default region name [None]: ap-southeast-1
Default output format [None]:
ubuntu@ip-172-31-35-103:~$
```

Step 3 – Install and setup kubectl

```
sudo curl --silent --location -o /usr/local/bin/kubectl https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.6/2022-03-09/bin/linux/amd64/kubectl
sudo chmod +x /usr/local/bin/kubectl
kubectl version --short --client
```

It should return you back with the Kubernetes version -

```
ubuntu@ip-172-31-35-103:~$ sudo curl --silent --location -o /usr/local/bin/kubectl https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.6/2022-03-09/bin/linux/amd64/kubectl
ubuntu@ip-172-31-35-103:~$ sudo chmod +x /usr/local/bin/kubectl
ubuntu@ip-172-31-35-103:~$ kubectl version --short --client
Client Version: v1.22.6-eks-7d68063
ubuntu@ip-172-31-35-103:~$ |
```

Step 4— Install and set up eksctl

Download and extract the latest release of eksctl with below command

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

Move the extracted binary to /usr/local/bin

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

Check whether the installation has been successfully done using

```
eksctl version
```

```
ubuntu@ip-172-31-35-103:~$ curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp  
ubuntu@ip-172-31-35-103:~$ sudo mv /tmp/eksctl /usr/local/bin  
ubuntu@ip-172-31-35-103:~$ eksctl version  
0.144.0  
ubuntu@ip-172-31-35-103:~$ |
```

Step 5— Install Helm Chart

The next tool we need is Helm Chart. Helm is a package manager for Kubernetes, an open-source container orchestration platform. Helm helps you manage Kubernetes applications by making it easy to install, update, and delete them. Use the below script to install helm

```
curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3  
chmod 700 get_helm.sh  
../get_helm.sh
```

```
ubuntu@ip-172-31-35-103:~$ curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3  
ubuntu@ip-172-31-35-103:~$ chmod 700 get_helm.sh  
ubuntu@ip-172-31-35-103:~$ ./get_helm.sh  
Downloading https://get.helm.sh/helm-v3.12.1-linux-amd64.tar.gz  
Verifying checksum... Done.  
Preparing to install helm into /usr/local/bin  
helm installed into /usr/local/bin/helm  
ubuntu@ip-172-31-35-103:~$ |
```

You can verify the helm installation using

```
helm version
```

```
ubuntu@ip-172-31-35-103:~$ helm version  
version.BuildInfo{Version:"v3.12.1", GitCommit:"f32a527a060157990e2aa86bf45010dfb3cc8b8d", GitTreeState:"clean", GoVersion:"go1.20.4"}  
ubuntu@ip-172-31-35-103:~$ |
```

Now, here we are done with the installation of **AWS**, **CLI**, **kubectl**, **eksctl** and **Helm**

Step 6— Now we will configure the Amazon EKS cluster using eksctl. You can refer to the detailed documentation

here <https://docs.aws.amazon.com/eks/latest/userguide/clusters.html>

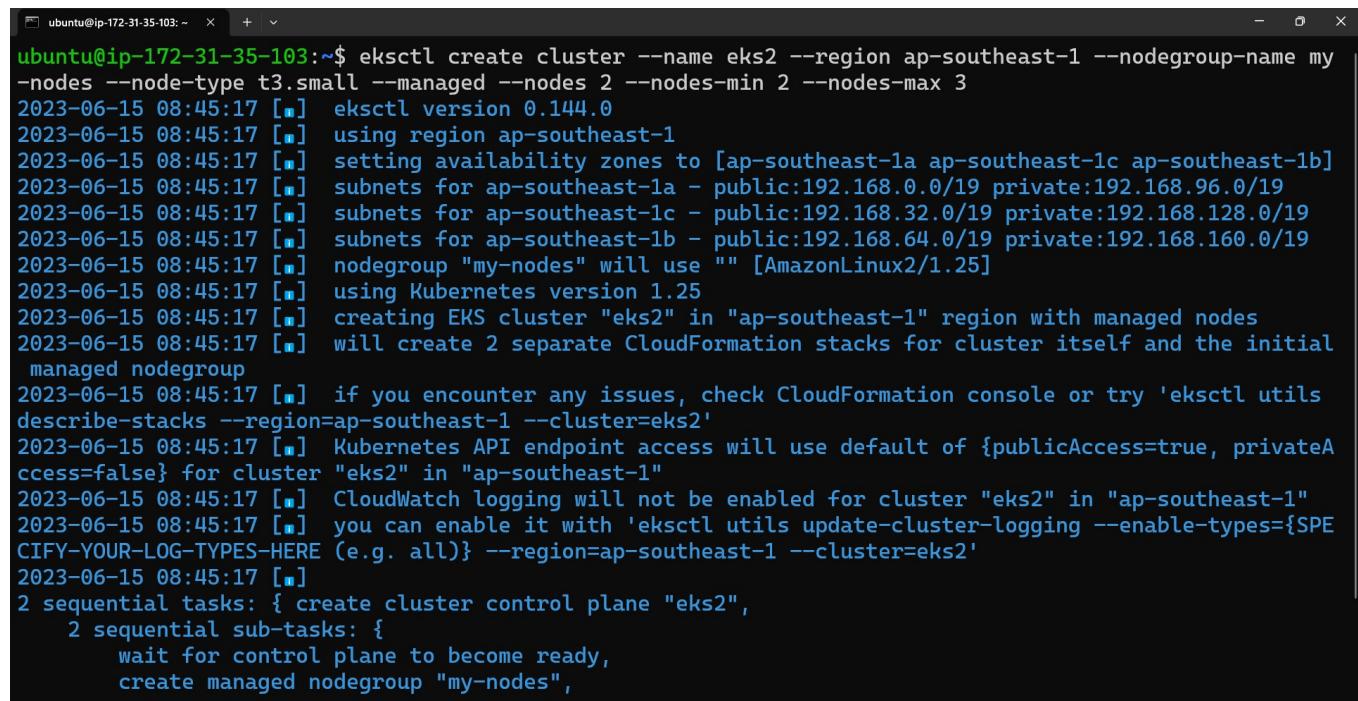
You need the following in order to run the eksctl command

1. Name of the cluster : — eks4
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 t3.small
6. Number of nodes: — nodes 2
7. Minimum Number of nodes: — nodes-min 2
8. Maximum Number of nodes: — nodes-max 3

Here is the eksctl command

```
eksctl create cluster --name eks2 --region us-east-1 --nodegroup-name my-nodes --node-type t3.small --managed --nodes 2 --nodes-min 2 --nodes-max 3
```

Kindly note that it would take 15–20 minutes for this installation to complete. Once it is done, you can go to your AWS Console and look for the eksctl clusters



```
ubuntu@ip-172-31-35-103:~$ eksctl create cluster --name eks2 --region ap-southeast-1 --nodegroup-name my-nodes --node-type t3.small --managed --nodes 2 --nodes-min 2 --nodes-max 3
2023-06-15 08:45:17 [] eksctl version 0.144.0
2023-06-15 08:45:17 [] using region ap-southeast-1
2023-06-15 08:45:17 [] setting availability zones to [ap-southeast-1a ap-southeast-1c ap-southeast-1b]
2023-06-15 08:45:17 [] subnets for ap-southeast-1a - public:192.168.0.0/19 private:192.168.96.0/19
2023-06-15 08:45:17 [] subnets for ap-southeast-1c - public:192.168.32.0/19 private:192.168.128.0/19
2023-06-15 08:45:17 [] subnets for ap-southeast-1b - public:192.168.64.0/19 private:192.168.160.0/19
2023-06-15 08:45:17 [] nodegroup "my-nodes" will use "" [AmazonLinux2/1.25]
2023-06-15 08:45:17 [] using Kubernetes version 1.25
2023-06-15 08:45:17 [] creating EKS cluster "eks2" in "ap-southeast-1" region with managed nodes
2023-06-15 08:45:17 [] will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2023-06-15 08:45:17 [] if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=ap-southeast-1 --cluster=eks2'
2023-06-15 08:45:17 [] Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "eks2" in "ap-southeast-1"
2023-06-15 08:45:17 [] CloudWatch logging will not be enabled for cluster "eks2" in "ap-southeast-1"
2023-06-15 08:45:17 [] you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=ap-southeast-1 --cluster=eks2'
2023-06-15 08:45:17 []
2 sequential tasks: { create cluster control plane "eks2",
  2 sequential sub-tasks: {
    wait for control plane to become ready,
    create managed nodegroup "my-nodes",
```

```

ubuntu@ip-172-31-35-103:~ + 
2023-06-15 08:51:17 [.] waiting for CloudFormation stack "eksctl-eks2-cluster"
2023-06-15 08:52:17 [.] waiting for CloudFormation stack "eksctl-eks2-cluster"
2023-06-15 08:53:17 [.] waiting for CloudFormation stack "eksctl-eks2-cluster"
2023-06-15 08:55:18 [.] building managed nodegroup stack "eksctl-eks2-nodegroup-my-nodes"
2023-06-15 08:55:18 [.] deploying stack "eksctl-eks2-nodegroup-my-nodes"
2023-06-15 08:55:18 [.] waiting for CloudFormation stack "eksctl-eks2-nodegroup-my-nodes"
2023-06-15 08:55:49 [.] waiting for CloudFormation stack "eksctl-eks2-nodegroup-my-nodes"
2023-06-15 08:56:46 [.] waiting for CloudFormation stack "eksctl-eks2-nodegroup-my-nodes"
2023-06-15 08:57:19 [.] waiting for CloudFormation stack "eksctl-eks2-nodegroup-my-nodes"
2023-06-15 08:58:51 [.] waiting for CloudFormation stack "eksctl-eks2-nodegroup-my-nodes"
2023-06-15 08:58:51 [.] waiting for the control plane to become ready
2023-06-15 08:58:52 [.] saved kubeconfig as "/home/ubuntu/.kube/config"
2023-06-15 08:58:52 [.] no tasks
2023-06-15 08:58:52 [.] all EKS cluster resources for "eks2" have been created
2023-06-15 08:58:52 [.] nodegroup "my-nodes" has 2 node(s)
2023-06-15 08:58:52 [.] node "ip-192-168-24-210.ap-southeast-1.compute.internal" is ready
2023-06-15 08:58:52 [.] node "ip-192-168-51-50.ap-southeast-1.compute.internal" is ready
2023-06-15 08:58:52 [.] waiting for at least 2 node(s) to become ready in "my-nodes"
2023-06-15 08:58:52 [.] nodegroup "my-nodes" has 2 node(s)
2023-06-15 08:58:52 [.] node "ip-192-168-24-210.ap-southeast-1.compute.internal" is ready
2023-06-15 08:58:52 [.] node "ip-192-168-51-50.ap-southeast-1.compute.internal" is ready
2023-06-15 08:58:54 [.] kubectl command should work with "/home/ubuntu/.kube/config", try 'kubectl get nodes'
2023-06-15 08:58:54 [.] EKS cluster "eks2" in "ap-southeast-1" region is ready
ubuntu@ip-172-31-35-103:~$ 

```

We can verify the cluster by logging into the AWS Console

The screenshot shows the AWS EKS Clusters page. On the left sidebar, under 'Clusters', there is a link to 'New'. Under 'Related services', links to 'Amazon ECR' and 'AWS Batch' are shown. The main content area displays a table titled 'Clusters (1) Info' with one row for 'eks2'. The table includes columns for 'Cluster name', 'Status', 'Kubernetes version', and 'Provider'. The status is 'Active', the Kubernetes version is '1.25', and the provider is 'EKS'. A message at the top of the table says 'New Kubernetes versions are available for 1 cluster.'

Cluster name	Status	Kubernetes version	Provider
eks2	Active	1.25 Update now	EKS

The screenshot shows the AWS EKS Cluster details page for 'eks2'. The left sidebar is identical to the previous screenshot. The main content area shows the 'eks2' cluster details. At the top, a message says 'New Kubernetes versions are available for this cluster. Learn more'. Below this, the 'Cluster info' section shows the Kubernetes version as '1.25', status as 'Active', and provider as 'EKS'. The 'Compute' tab is selected. In the 'Nodes (2) Info' section, two nodes are listed: 'ip-192-168-24-210.ap-southeast-1.compute.internal' and 'ip-192-168-51-50.ap-southeast-1.compute.internal', both of which are 't3.small' instances in the 'my-nodes' group and were created 3 minutes ago, with a status of 'Ready'.

```
eksctl get cluster --name eks2 --region ap-southeast-1
```

This should confirm that EKS cluster is up and running.

```
ubuntu@ip-172-31-35-103:~$ eksctl get cluster --name eks2 --region ap-southeast-1
NAME      VERSION STATUS   CREATED          VPC           SUBNETS
SECURITYGROUPS    PROVIDER
eks2       1.25    ACTIVE  2023-06-15T08:45:54Z  vpc-0964b7eab5c60e6a3  subnet-006c9c11486693721,subnet-02a52be97875a08a5,subnet-04ffd66d93a838da9,subnet-074356b6ac6c00728,subnet-08be8563916152f34,subnet-0d075b3bae70c3263  sg-02484972dc9eaf508  EKS
ubuntu@ip-172-31-35-103:~$ |
```

Update Kube config by entering below command:

```
aws eks update-kubeconfig --name eks2 --region us-east-1
```

```
ubuntu@ip-172-31-35-103:~$ aws eks update-kubeconfig --name eks2 --region ap-southeast-1
Updated context arn:aws:eks:ap-southeast-1:770506425278:cluster/eks2 in /home/ubuntu/.kube/config
ubuntu@ip-172-31-35-103:~$ |
```

Connect to EKS cluster using kubectl commands

To view the list of worker nodes as part of EKS cluster.

```
kubectl get nodes
```

```
kubectl get ns
```

```
ubuntu@ip-172-31-35-103:~$ kubectl get nodes
NAME                  STATUS  ROLES   AGE   VERSION
ip-192-168-24-210.ap-southeast-1.compute.internal  Ready   <none>  13m  v1.25.9-eks-0a21954
ip-192-168-51-50.ap-southeast-1.compute.internal  Ready   <none>  13m  v1.25.9-eks-0a21954
ubuntu@ip-172-31-35-103:~$ |
```

```
ubuntu@ip-172-31-35-103:~$ kubectl get ns
NAME      STATUS  AGE
default   Active  19m
kube-node-lease Active  20m
kube-public Active  20m
kube-system Active  20m
ubuntu@ip-172-31-35-103:~$ |
```

Step 7— We need to add the Helm Stable Charts for your local client. Execute the below command:

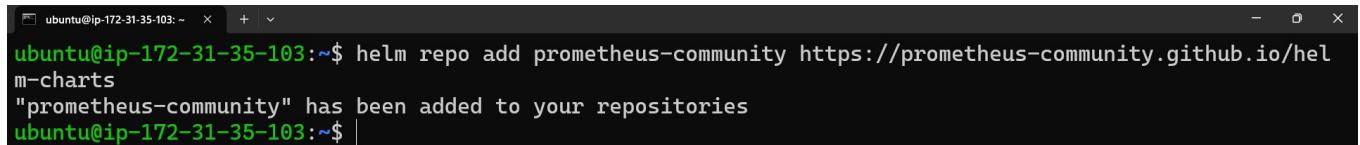
```
helm repo add stable https://charts.helm.sh/stable
```



```
ubuntu@ip-172-31-35-103:~$ helm repo add stable https://charts.helm.sh/stable
"stable" has been added to your repositories
ubuntu@ip-172-31-35-103:~$ |
```

Step 8— Add Prometheus Helm repo

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



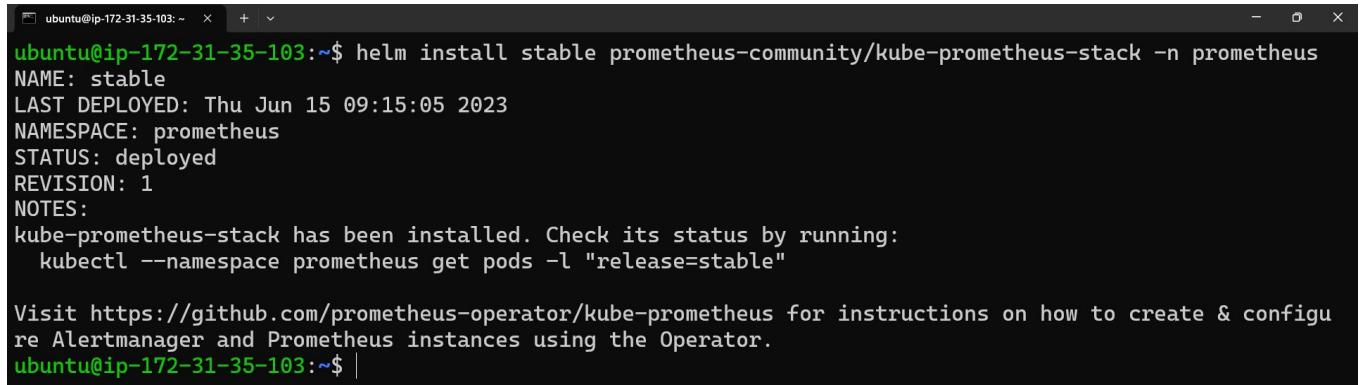
```
ubuntu@ip-172-31-35-103:~$ helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
"prometheus-community" has been added to your repositories
ubuntu@ip-172-31-35-103:~$ |
```

Step 9— Create Prometheus namespace

```
kubectl create namespace prometheus
```

Step 10— Install Prometheus

```
helm install stable prometheus-community/kube-prometheus-stack -n prometheus
```



```
ubuntu@ip-172-31-35-103:~$ helm install stable prometheus-community/kube-prometheus-stack -n prometheus
NAME: stable
LAST DEPLOYED: Thu Jun 15 09:15:05 2023
NAMESPACE: prometheus
STATUS: deployed
REVISION: 1
NOTES:
kube-prometheus-stack has been installed. Check its status by running:
  kubectl --namespace prometheus get pods -l "release=stable"

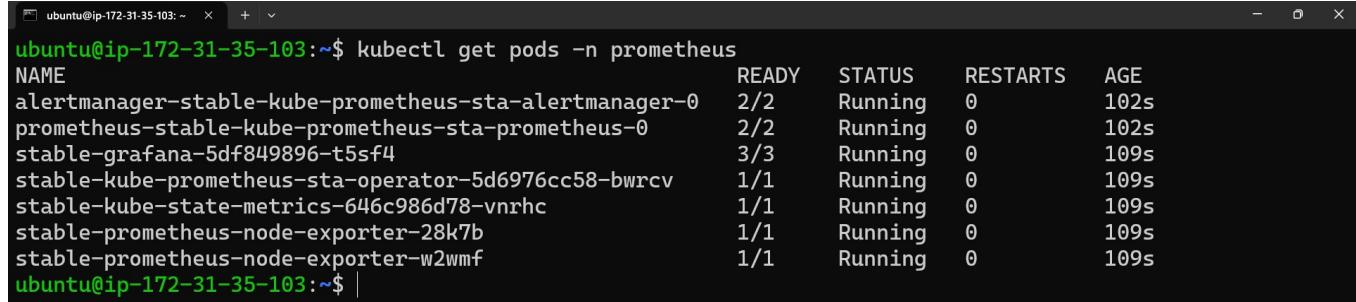
Visit https://github.com/prometheus-operator/kube-prometheus for instructions on how to create & configu
re Alertmanager and Prometheus instances using the Operator.
ubuntu@ip-172-31-35-103:~$ |
```

- above command is used to install kube-Prometheus-stack. The helm repo kube-stack-Prometheus comes with a Grafana deployment embedded (as the default one).

Now Prometheus is installed using helm in the ec2 instance

- To check whether Prometheus is installed or not use the below command

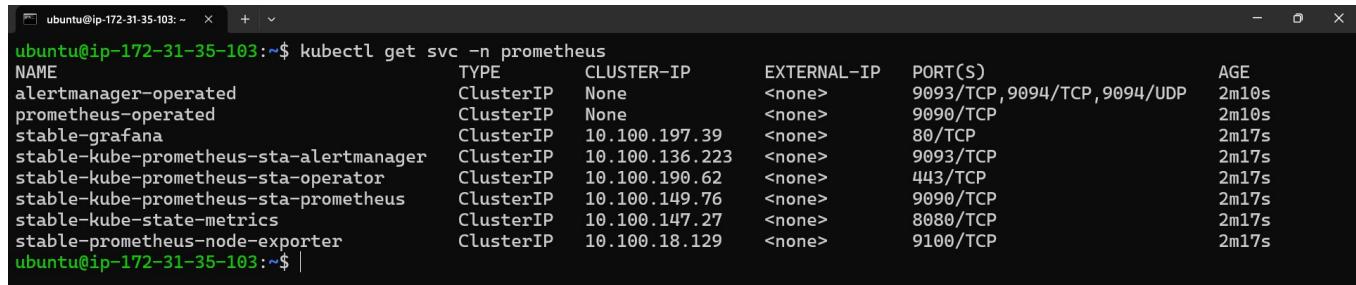
```
kubectl get pods -n prometheus
```



```
ubuntu@ip-172-31-35-103:~$ kubectl get pods -n prometheus
NAME                               READY   STATUS    RESTARTS   AGE
alertmanager-stable-kube-prometheus-sta-alertmanager-0   2/2     Running   0          102s
prometheus-stable-kube-prometheus-sta-prometheus-0       2/2     Running   0          102s
stable-grafana-5df849896-t5sf4                         3/3     Running   0          109s
stable-kube-prometheus-sta-operator-5d6976cc58-bwrcv      1/1     Running   0          109s
stable-kube-state-metrics-646c986d78-vnrhc              1/1     Running   0          109s
stable-prometheus-node-exporter-28k7b                   1/1     Running   0          109s
stable-prometheus-node-exporter-w2wmf                  1/1     Running   0          109s
ubuntu@ip-172-31-35-103:~$ |
```

to check the services file (svc) of the Prometheus

```
kubectl get svc -n prometheus
```



```
ubuntu@ip-172-31-35-103:~$ kubectl get svc -n prometheus
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
alertmanager-operated   ClusterIP  None        <none>      9093/TCP,9094/TCP,9094/UDP  2m10s
prometheus-operated    ClusterIP  None        <none>      9090/TCP                2m10s
stable-grafana         ClusterIP  10.100.197.39 <none>      80/TCP                 2m17s
stable-kube-prometheus-sta-alertmanager   ClusterIP  10.100.136.223 <none>      9093/TCP                2m17s
stable-kube-prometheus-sta-operator        ClusterIP  10.100.190.62  <none>      443/TCP                 2m17s
stable-kube-prometheus-sta-prometheus     ClusterIP  10.100.149.76  <none>      9090/TCP                2m17s
stable-kube-state-metrics               ClusterIP  10.100.147.27 <none>      8080/TCP                2m17s
stable-prometheus-node-exporter         ClusterIP  10.100.18.129  <none>      9100/TCP                2m17s
ubuntu@ip-172-31-35-103:~$ |
```

Grafana will be coming along with Prometheus as the stable version

This output is conformation that our Prometheus is installed successfully there is no need of installing Grafana as a separate tool it comes along with Prometheus

let's expose Prometheus and Grafan to the external world

there are 2 ways to expose

1. through Node Port

2. through LoadBalancer

let's go with the LoadBalancer

to attach the load balancer we need to change from ClusterIP to LoadBalancer
command to get the svc file

```
kubectl edit svc stable-kube-prometheus-sta-prometheus -n prometheus
```

```
ubuntu@ip-172-31-35-103:~ % + | v
name: stable-kube-prometheus-sta-prometheus
namespace: prometheus
resourceVersion: "3885"
uid: a484d903-0837-4b7e-a59d-e52e531dbeae
spec:
  clusterIP: 10.100.149.76
  clusterIPs:
  - 10.100.149.76
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - name: http-web
    port: 9090
    protocol: TCP
    targetPort: 9090
  selector:
    app.kubernetes.io/name: prometheus
    prometheus: stable-kube-prometheus-sta-prometheus
  sessionAffinity: None
  type: LoadBalancer
status:
  loadBalancer: {}
-- INSERT --
```

```
ubuntu@ip-172-31-35-103:~ % + | v
ubuntu@ip-172-31-35-103:~$ kubectl edit svc stable-kube-prometheus-sta-prometheus -n prometheus
service/stable-kube-prometheus-sta-prometheus edited
ubuntu@ip-172-31-35-103:~$ |
```

change it from Cluster IP to LoadBalancer after changing make sure you save the file

```
kubectl get svc -n prometheus
```

```

ubuntu@ip-172-31-35-103:~$ kubectl edit svc stable-kube-prometheus-sta-prometheus -n prometheus
service/stable-kube-prometheus-sta-prometheus edited
ubuntu@ip-172-31-35-103:~$ kubectl get svc prometheus
NAME                TYPE            CLUSTER-IP      EXTERNAL-IP
alertmanager-operated   ClusterIP     None           <none>
prometheus-operated    ClusterIP     None           <none>
stable-grafana          ClusterIP     10.100.197.39  <none>
stable-kube-prometheus-sta-alertmanager ClusterIP  10.100.136.223  <none>
stable-kube-prometheus-sta-operator       LoadBalancer  10.100.149.76   aa484d98308374b7ea59de52e531dbea-555742178.ap-southeast-1.elb.amazonaws.com
stable-kube-state-metrics      ClusterIP     10.100.147.27  <none>
stable-prometheus-node-exporter     ClusterIP     10.100.18.129  <none>

PORT(S)          AGE
9093/TCP, 9094/TCP, 9094/UDP  5m38s
9090/TCP                   5m38s
80/TCP                      5m45s
9093/TCP                   5m45s
404/TCP                     5m45s
9090-31258/TCP             5m45s
8088/TCP                   5m45s
9100/TCP                   5m45s

```

you guys can see I have a load balancer for my Prometheus which I can access from that link port 9090

The screenshot shows the Prometheus UI interface. At the top, there's a navigation bar with links for Prometheus, Alerts, Graph, Status, and Help. Below the navigation bar is a search bar with the placeholder "Expression (press Shift+Enter for newlines)". To the right of the search bar are several configuration and execution buttons: "Use local time", "Enable query history", "Enable autocomplete" (checked), "Enable highlighting" (checked), "Enable linter" (checked), and a large blue "Execute" button. Below the search bar, there are two tabs: "Table" (selected) and "Graph". Under the "Table" tab, there's a timestamp selector with arrows and a "Evaluation time" label. A message "No data queried yet" is displayed. In the bottom right corner of the main area, there's a "Remove Panel" link.

This screenshot shows another view of the Prometheus UI. It has a similar layout to the first one, with a navigation bar, search bar, and configuration buttons. The "Table" tab is selected, and it shows a timestamp selector with a "2h" button and a "Res. (s)" dropdown. A message "No data queried yet" is visible. In the bottom right corner, there's a "Remove Panel" link.

we can use a Prometheus UI for monitoring the EKS but the UI of Prometheus is not a convent for the user Grafana will extract the matrix from the Prometheus UI and show it in a user-friendly manner

let's change the SVC file of the Grafana and expose it to the outer world

command to edit the SVC file of grafana

```
kubectl edit svc stable-grafana -n prometheus
```

```

ubuntu@ip-172-31-35-103:~ % + 
spec:
  clusterIP: 10.100.197.39
  clusterIPs:
    - 10.100.197.39
  internalTrafficPolicy: Cluster
  ipFamilies:
    - IPv4
  ipFamilyPolicy: SingleStack
  ports:
    - name: http-web
      port: 80
      protocol: TCP
      targetPort: 3000
  selector:
    app.kubernetes.io/instance: stable
    app.kubernetes.io/name: grafana
  sessionAffinity: None
  type: LoadBalancer
status:
  loadBalancer: {}

```

the Grafana LoadBalancer also exposed

```

ubuntu@ip-172-31-35-103:~ % + 
ubuntu@ip-172-31-35-103:~$ kubectl edit svc stable-grafana -n prometheus
service/stable-grafana edited
kubectl get svc -n prometheus

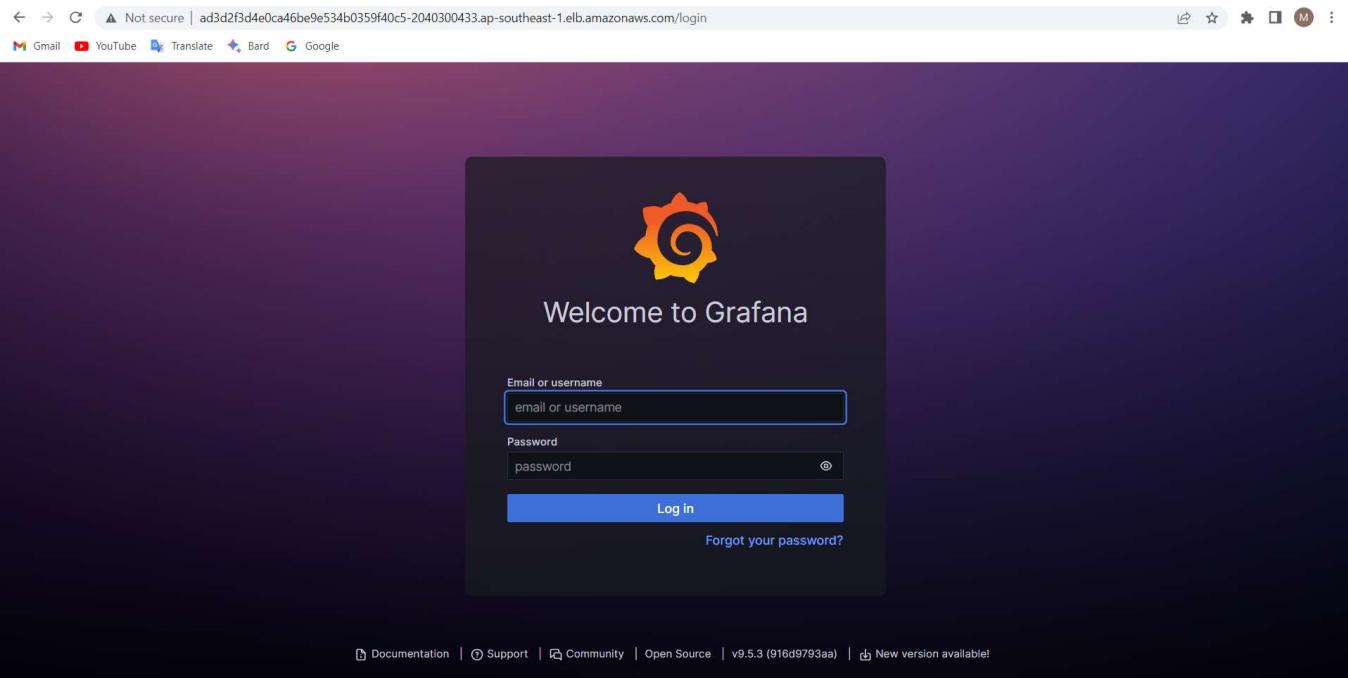
```

```

ubuntu@ip-172-31-35-103:~ % + 
ubuntu@ip-172-31-35-103:~$ kubectl get svc -n prometheus
NAME           TYPE        CLUSTER-IP   EXTERNAL-IP                                     PORT(S)          AGE
alertmanager-operated   ClusterIP  None         <none>
prometheus-operated   ClusterIP  None         <none>
stable-grafana        LoadBalancer 10.100.197.39  ad3d2f3d4e0ca46be9e534b0359f40c5-2040300433.ap-southeast-1.elb.amazonaws.com  80:30362/TCP  17m
stable-kube-prometheus-sta-alertmanager   ClusterIP  10.100.136.223 <none>
stable-kube-prometheus-sta-operator       ClusterIP  10.100.198.62  <none>
stable-kube-prometheus-sta-prometheus   LoadBalancer 10.100.149.76  aa484d90308374b7ea59de52e531dbea-555742178.ap-southeast-1.elb.amazonaws.com  9093/TCP, 4413/TCP, 9099:31258/TCP, 8088/TCP, 9100/TCP  17m
stable-kube-state-metrics      ClusterIP  10.100.147.27 <none>
stable-prometheus-node-exporter     ClusterIP  10.100.18.129 <none>
ubuntu@ip-172-31-35-103:~ |

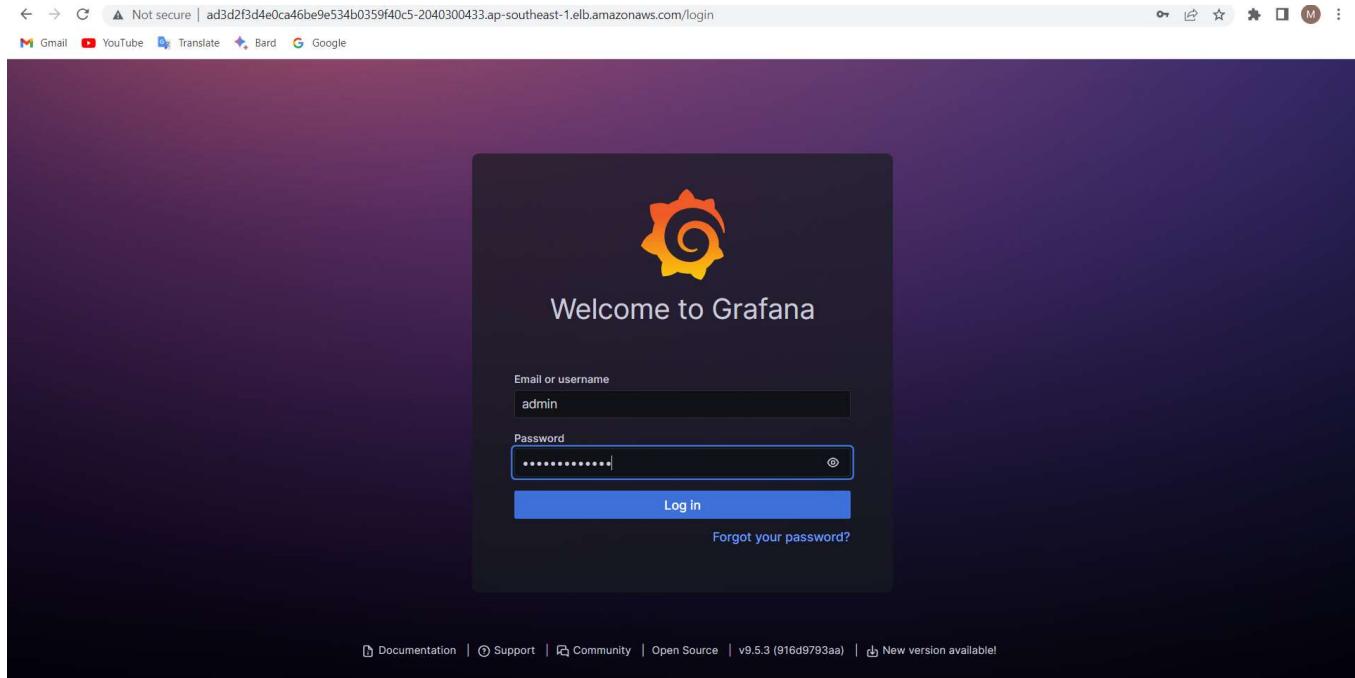
```

use the link of the LoadBalancer and access from the Browser



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

use the above command to get the password
the user name is admin



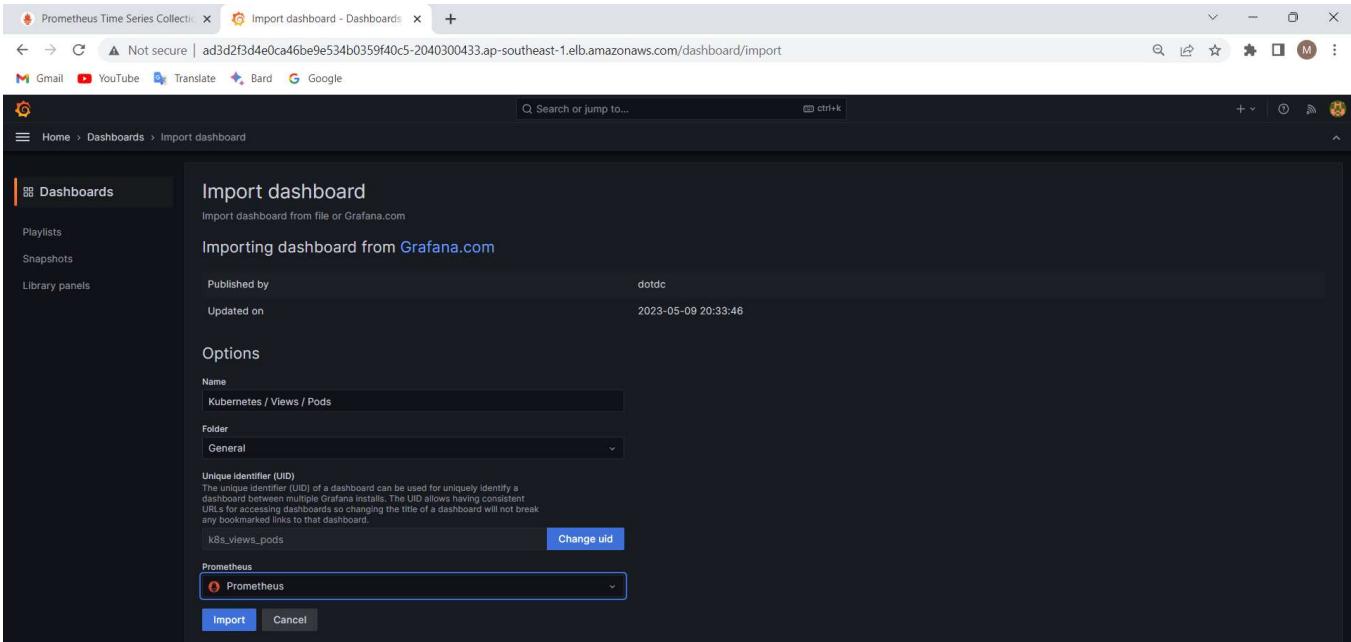
Create a Dashboard in Grafana

let's create a Dashboard by importing
click on Import and import the dashboard with numbers

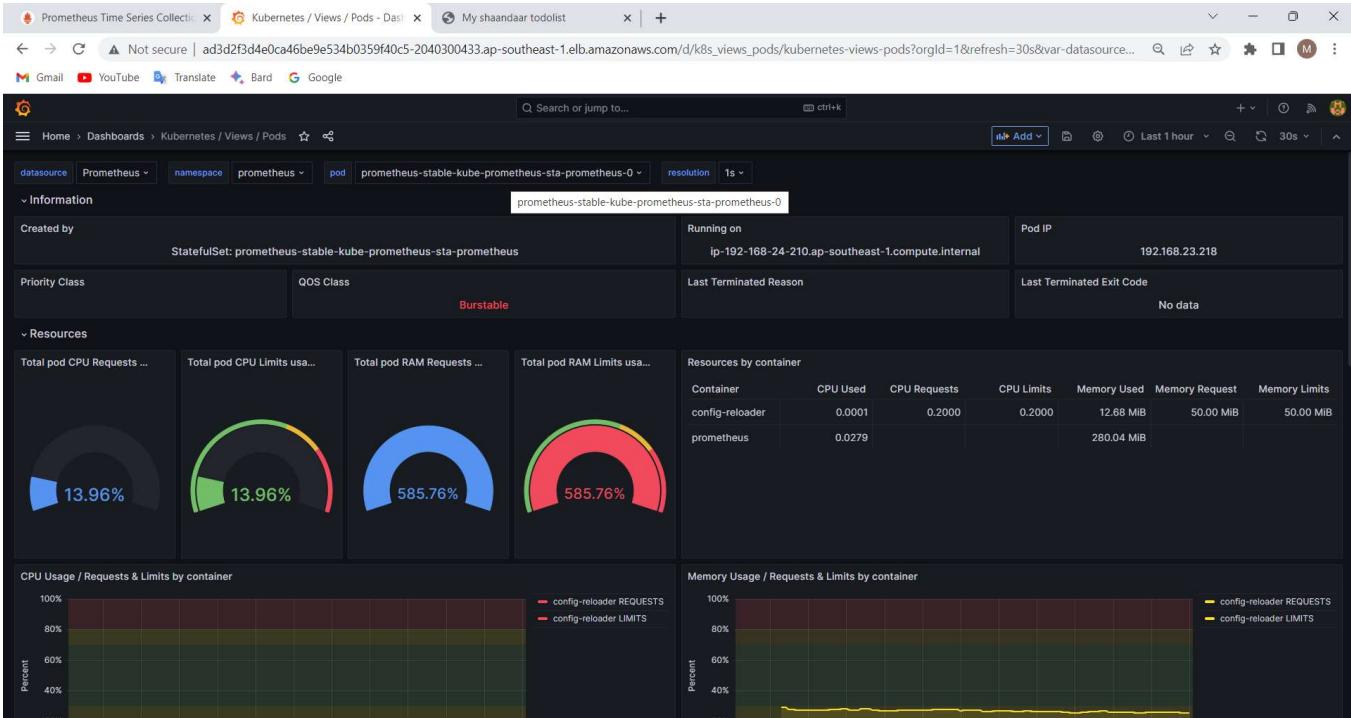
The screenshot shows the Grafana welcome screen. On the left, there's a sidebar with sections like 'Basic', 'Dashboards', 'Starred dashboards', and 'Recently viewed dashboards'. The main area has a search bar at the top and a list of recent dashboards. A vertical sidebar on the right contains links for 'Alertmanager / Overview', 'CoreDNS', 'Grafana Overview', 'Kubernetes / API server', and 'Kubernetes / Compute Resources / Multi-Cluster'. Below these are 'Actions' (New dashboard, Import dashboard, Create alert rule), 'Pages' (Home, Starred, Dashboards, Explore, Alerting, Overview), and a 'COMPLETE' section with a 'Create your first dashboard' button. The URL in the address bar is `ad3d2f3d4e0ca46be9e534b0359f40c5-2040300433.ap-southeast-1.elb.amazonaws.com/dashboard/import`.

there are plenty of ready templates to use the pre-existing templates and modify
based on our desired
it uses a Prometheus

The screenshot shows the 'Import dashboard' dialog in Grafana. On the left, there's a sidebar with 'Dashboards', 'Playlists', 'Snapshots', and 'Library panels'. The main area has a search bar at the top and a large central input field with a dashed border and an upward arrow icon, labeled 'Upload dashboard JSON file'. Below it is a note 'Accepted file types: .json, .txt'. There are two sections at the bottom: 'Import via grafana.com' with a file input field containing '15760' and a 'Load' button, and 'Import via panel json' with a large empty text area. At the bottom are 'Load' and 'Cancel' buttons.



click on the Import

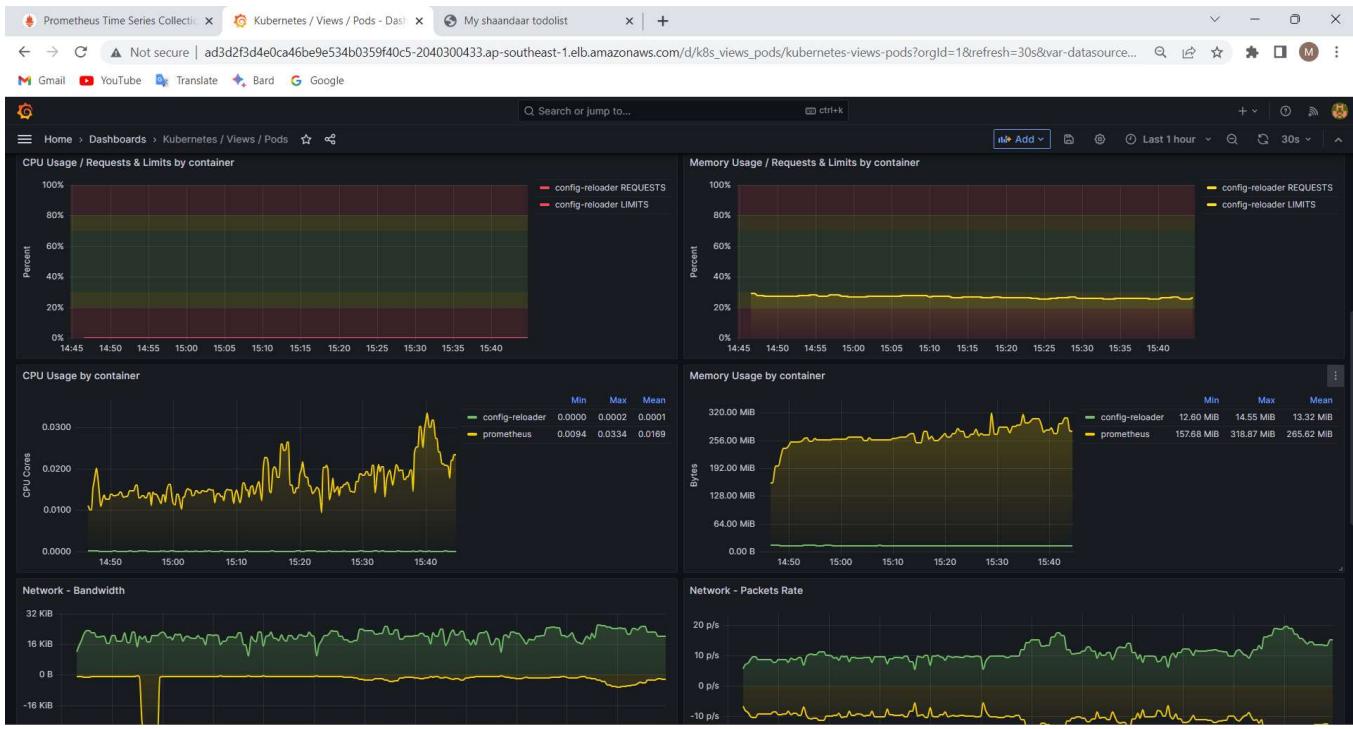


the Entire data of the cluster

where we can able to see the entire data of the EKS cluster

1. CPU and RAM use

2. pods in a specific namespace



3. Pod up history

4. HPA

5. Resources by Container

CPU used by container & limits

network bandwidth & packet rate

Step 11— Install Node JS application and monitor it on Grafana

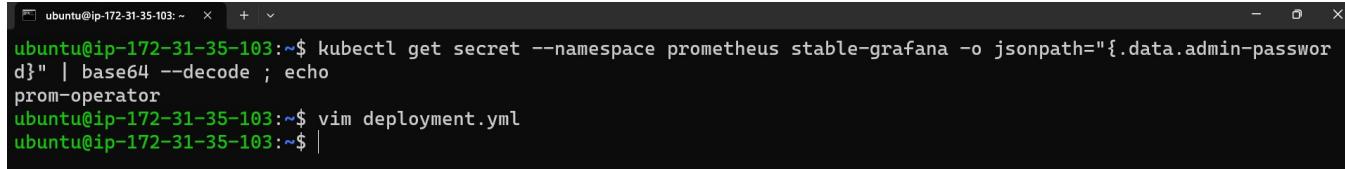
To make use of Grafana dashboard, we will deploy Node.js application on Kubernetes. Download deployment.yml file from the below.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: node-todo-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: node-todo-app
  template:
    metadata:
      labels:
        app: node-todo-app
    spec:
      containers:
```

```

- name: node-todo-app
  image: mahesh8887/node-todo-app:latest
  ports:
    - containerPort: 8000
  env:
    - name: PORT
      value: "8000"
---
apiVersion: v1
kind: Service
metadata:
  name: node-todo-app
spec:
  type: LoadBalancer
  ports:
    - port: 80
      targetPort: 8000
  selector:
    app: node-todo-app

```



```

ubuntu@ip-172-31-35-103:~$ kubectl get secret --namespace prometheus stable-grafana -o jsonpath="{.data.admin-password}" | base64 --decode ; echo
prom-operator
ubuntu@ip-172-31-35-103:~$ vim deployment.yml
ubuntu@ip-172-31-35-103:~$ 

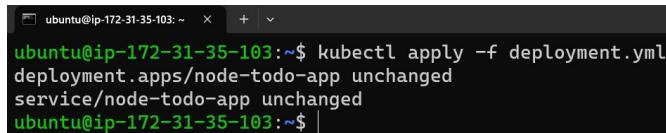
```

To deploy the Node.js application on kubernetes cluster user the following `kubectl` command. Verify the deployment by running the following `kubectl` command

```

kubectl apply -f deployment.yml
kubectl get deployment
kubectl get pods

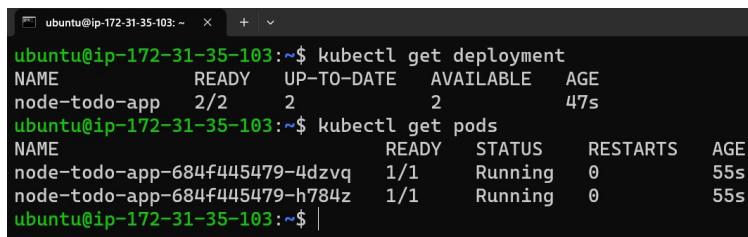
```



```

ubuntu@ip-172-31-35-103:~$ kubectl apply -f deployment.yml
deployment.apps/node-todo-app unchanged
service/node-todo-app unchanged
ubuntu@ip-172-31-35-103:~$ 

```



```

ubuntu@ip-172-31-35-103:~$ kubectl get deployment
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
node-todo-app   2/2       2          2           47s
ubuntu@ip-172-31-35-103:~$ kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
node-todo-app-684f445479-4dzvq   1/1     Running   0          55s
node-todo-app-684f445479-h784z   1/1     Running   0          55s
ubuntu@ip-172-31-35-103:~$ 

```

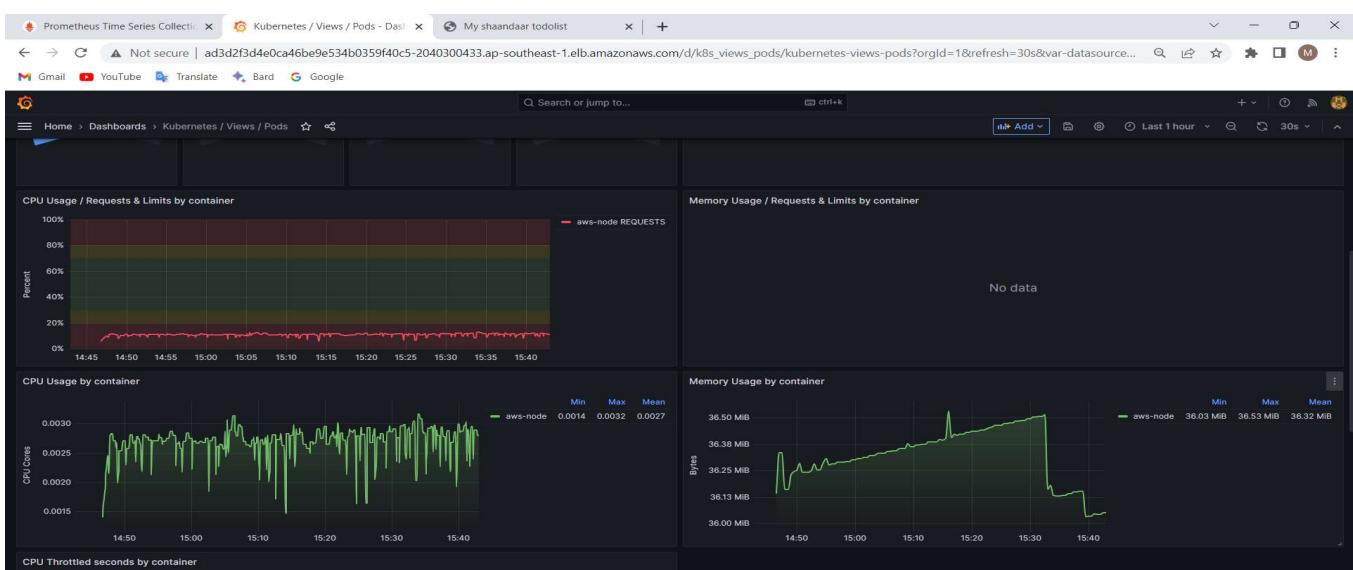
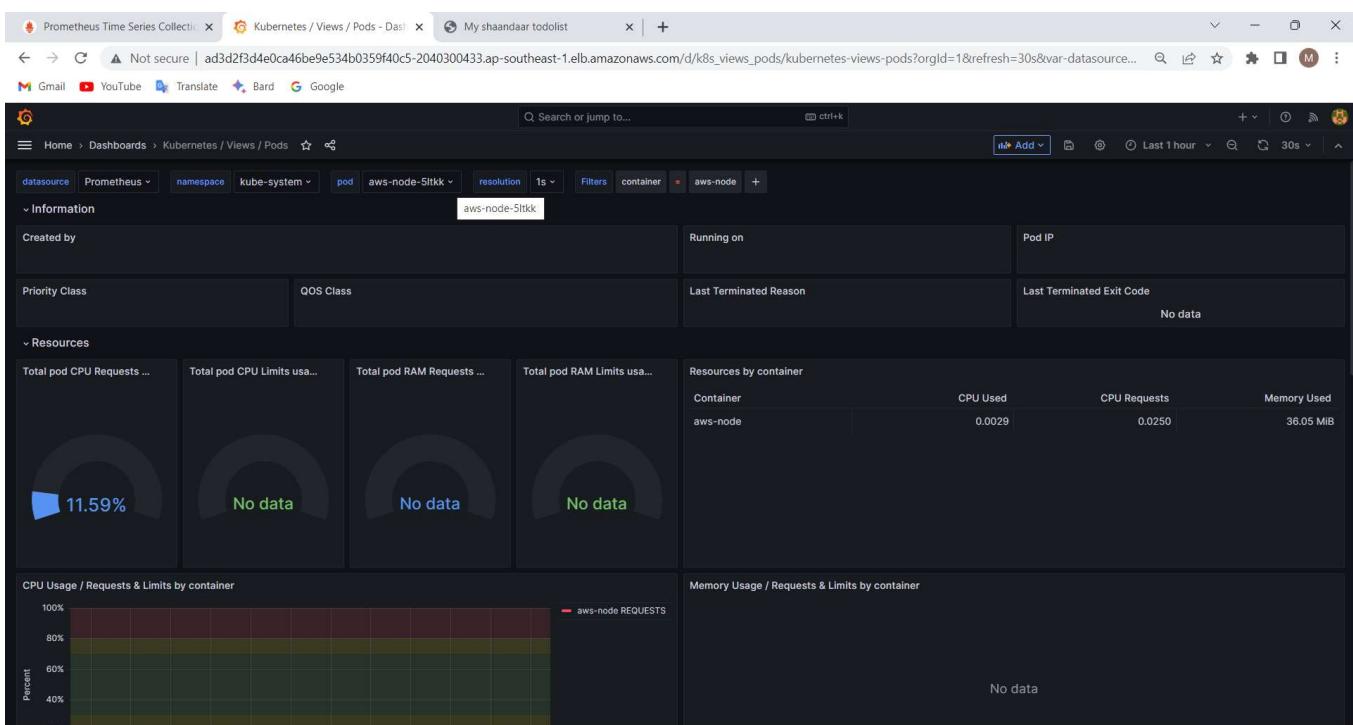
The Node.js Application is running successfully.

```
ubuntu@ip-172-31-35-103:~ $ kubectl get svc
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP
kubernetes     ClusterIP  10.100.0.1  <none>
node-todo-app LoadBalancer 10.100.239.252 a8b301f343bf94f799ae2fcc1a30aabb-1244414291.ap-southeast-1.elb.amazonaws.com
ubuntu@ip-172-31-35-103:~ $
```

@Devops Community is Super Awesome@

What should I do? Add

Refresh the Grafana dashboard to verify the deployment



Step 12 – Clean up/Deprovision

Now we will deprovision all our resources. First, lets start with EKS cluster

```
eksctl delete cluster --name eks2
```

Note: we can deploy the Grafana and Prometheus with a different method

1. Create all configuration files of both Prometheus and Grafana and execute them in the right order.
2. Prometheus Operator — to simplify and automate the configuration and management of the Prometheus monitoring stack running on a Kubernetes cluster
3. Helm chart — Using helm to install Prometheus Operator including Grafana
Helm is the best Practice among all the methods.

Hope you found this useful. Please feel free to reach out to me if you need any assistance while doing EKS Monitoring using Prometheus and Grafana with the help of AWS CLI, kubectl, eksctl and helm utility.

Thank you !