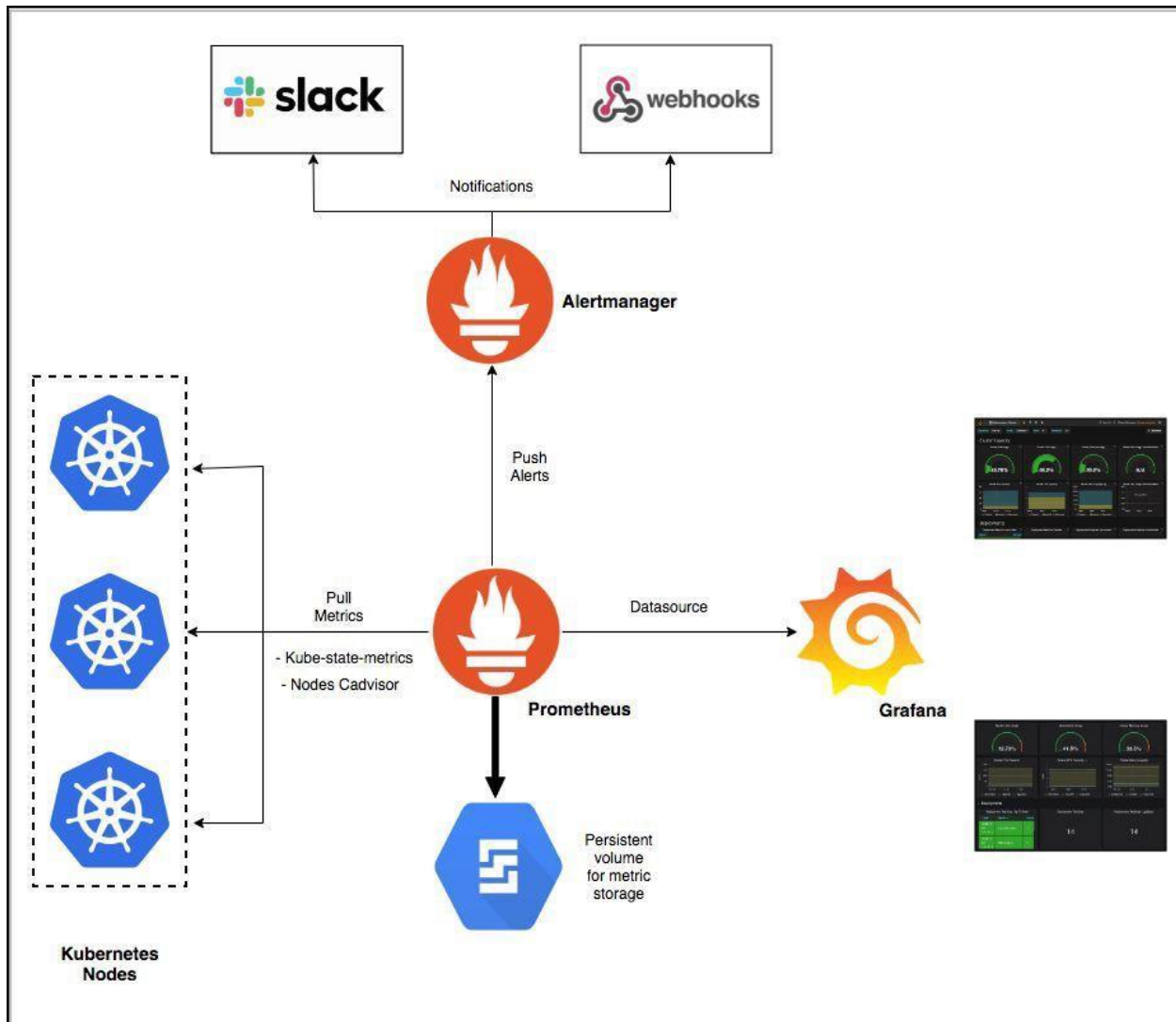


Monitoring of EKS cluster resources using Grafana & Prometheus

Architectural Diagram:



By,
Vaishal Shah
vaishal2611@gmail.com

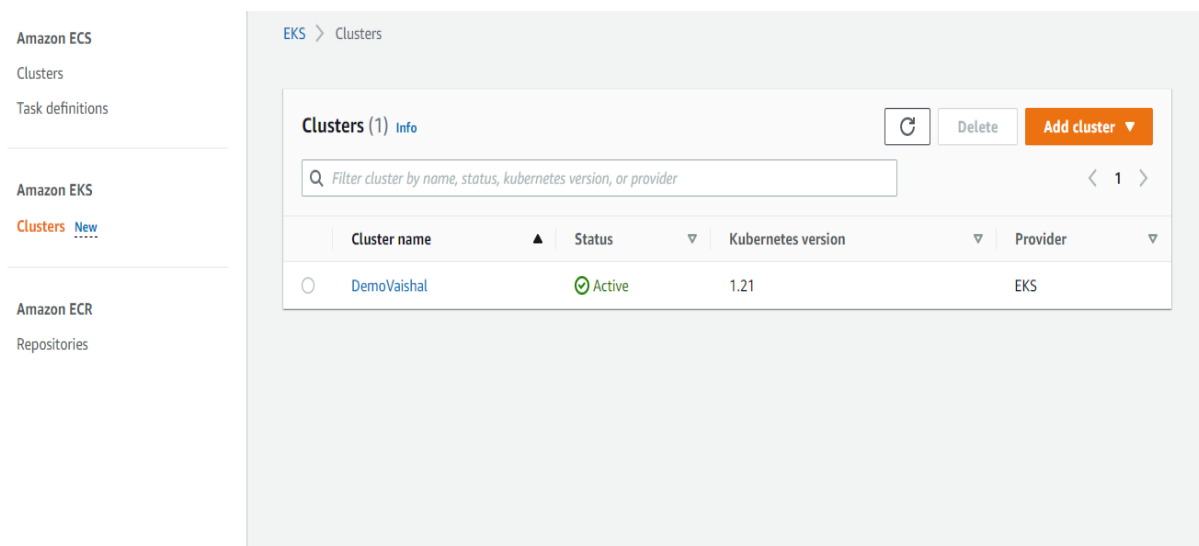
In this demo, I have created monitoring and visualizing infrastructure for EKS resources. Grafana and Prometheus are used for the setup. Along with it, Slack is integrated with Grafana. So, when an event hits a threshold configured, Grafana sends an alert message in the Slack channel.

Tech Stack:

EKS:	Elastic Kubernetes Service
Terraform:	For creating infrastructure as a code
Helm:	For deploying containerized application
Nginx-Ingress Controller:	For exposing services to external world
DNS management:	Local
Slack:	For getting alerts when conditional events occur.

EKS:

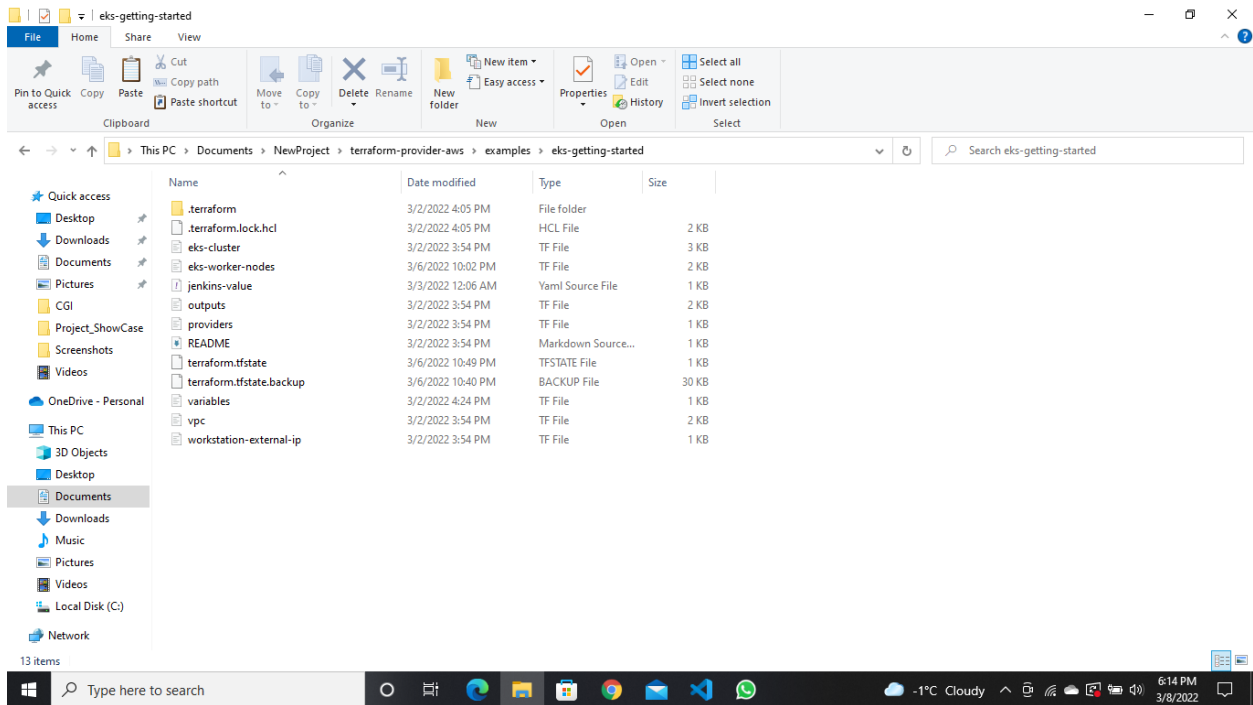
Cluster is created using the terraform, cluster has one working node and can expand up to maximum 3 nodes. Nodes running in AWS environments are spot instances with size of t3.medium.



Terraform:

It is used to create the entire Kubernetes infrastructure over AWS. Below shown are the files used for creating resources on cloud.

GitHub repo for code: <https://github.com/Vaishalshah2611/terraform-eks>



Helm Charts are used to deploy containerized resources on Kubernetes. Nginx-Ingress Controller, Prometheus and Grafana are deployed using helm.

Commands to install Nginx-Ingress Controller:

```
helm repo add ingress-nginx
https://kubernetes.github.io/ingress-nginx
helm repo update
helm install nginx-ingress ingress-nginx/ingress-nginx -values
customvalues.yaml
```

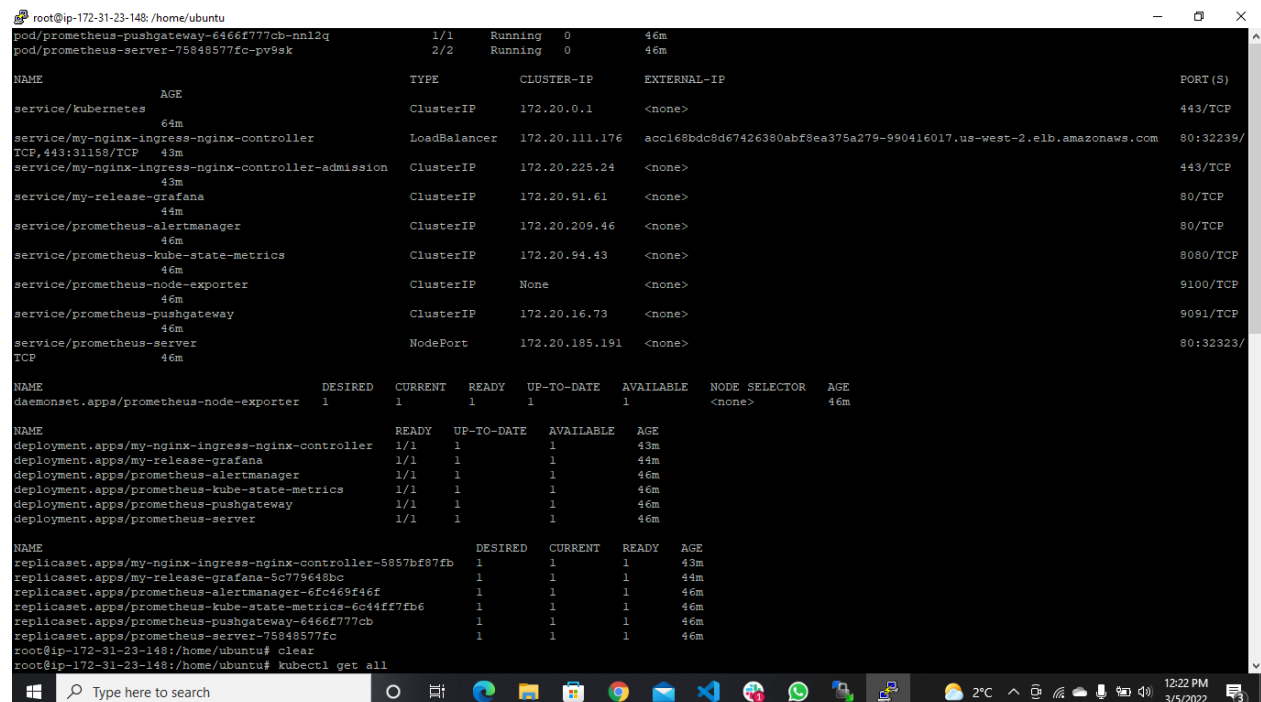
Commands used to install Prometheus:

```
helm repo add prometheus-community https://prometheus-  
community.github.io/helm-charts  
helm repo update  
helm install prometheus prometheus-community/prometheus -values  
customvalues.yaml
```

Commands to install Grafana:

```
helm repo add grafana https://grafana.github.io/helm-charts  
helm repo update  
helm install my-release grafana/grafana -values  
customvalues.yaml
```

Below images shows, everything is deployed on cluster along with ingress are configured for every service.



```
root@ip-172-31-23-148: /home/ubuntu  
pod/prometheus-pushgateway-6466f777cb-nn12q      1/1      Running    0          46m  
pod/prometheus-server-75848577fc-pv9sk          2/2      Running    0          46m
```

NAME	AGE	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
service/kubernetes	64m	ClusterIP	172.20.0.1	<none>	443/TCP
service/my-nginx-ingress-nginx-controller	43m	LoadBalancer	172.20.111.176	accl68bdc8d67426380abf9ea375a279-990416017.us-west-2.elb.amazonaws.com	80:32239/ TCP,443:31158/TCP
service/my-nginx-ingress-nginx-controller-admission	43m	ClusterIP	172.20.225.24	<none>	443/TCP
service/my-release-grafana	44m	ClusterIP	172.20.91.61	<none>	80/TCP
service/prometheus-alertmanager	46m	ClusterIP	172.20.209.46	<none>	80/TCP
service/prometheus-kube-state-metrics	46m	ClusterIP	172.20.94.43	<none>	8080/TCP
service/prometheus-node-exporter	46m	ClusterIP	None	<none>	9100/TCP
service/prometheus-pushgateway	46m	ClusterIP	172.20.16.73	<none>	9091/TCP
service/prometheus-server	46m	NodePort	172.20.185.191	<none>	80:32323/ TCP

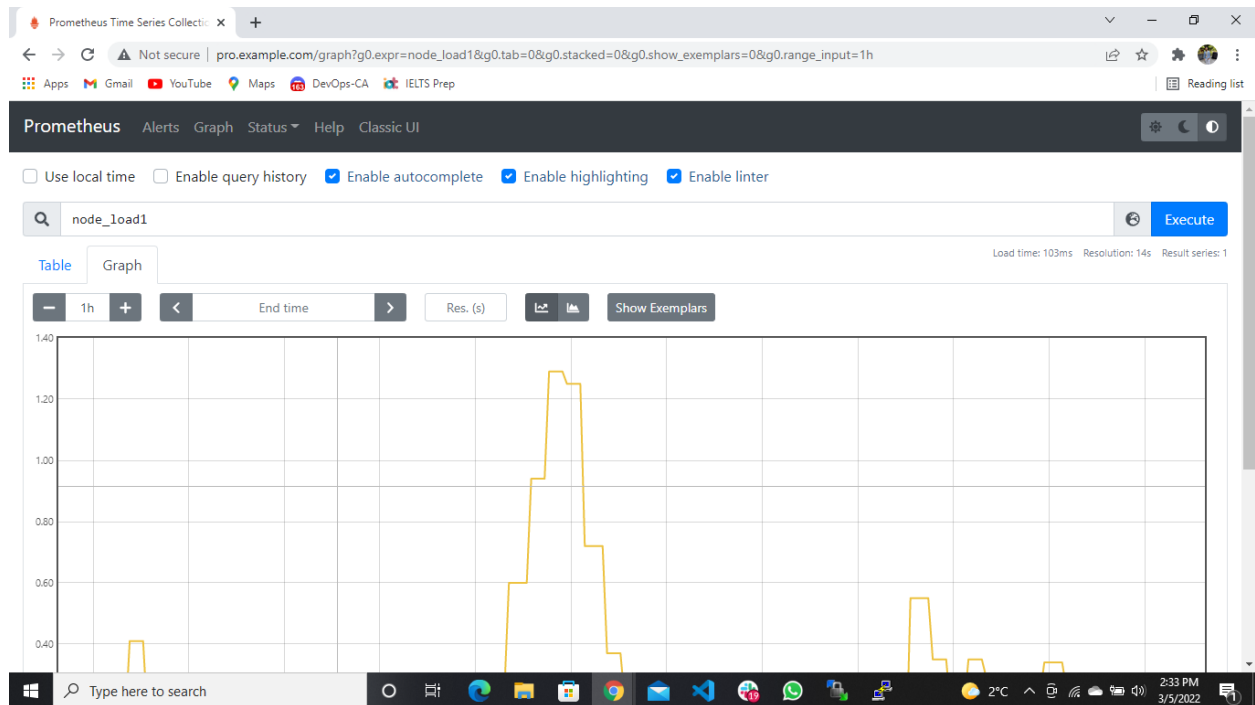
NAME	DESIRED	CURRENT	READY	UP-TO-DATE	AVAILABLE	NODE SELECTOR	AGE
daemonset.apps/prometheus-node-exporter	1	1	1	1	1	<none>	46m

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/my-nginx-ingress-nginx-controller	1/1	1	1	43m
deployment.apps/my-release-grafana	1/1	1	1	44m
deployment.apps/prometheus-alertmanager	1/1	1	1	46m
deployment.apps/prometheus-kube-state-metrics	1/1	1	1	46m
deployment.apps/prometheus-pushgateway	1/1	1	1	46m
deployment.apps/prometheus-server	1/1	1	1	46m

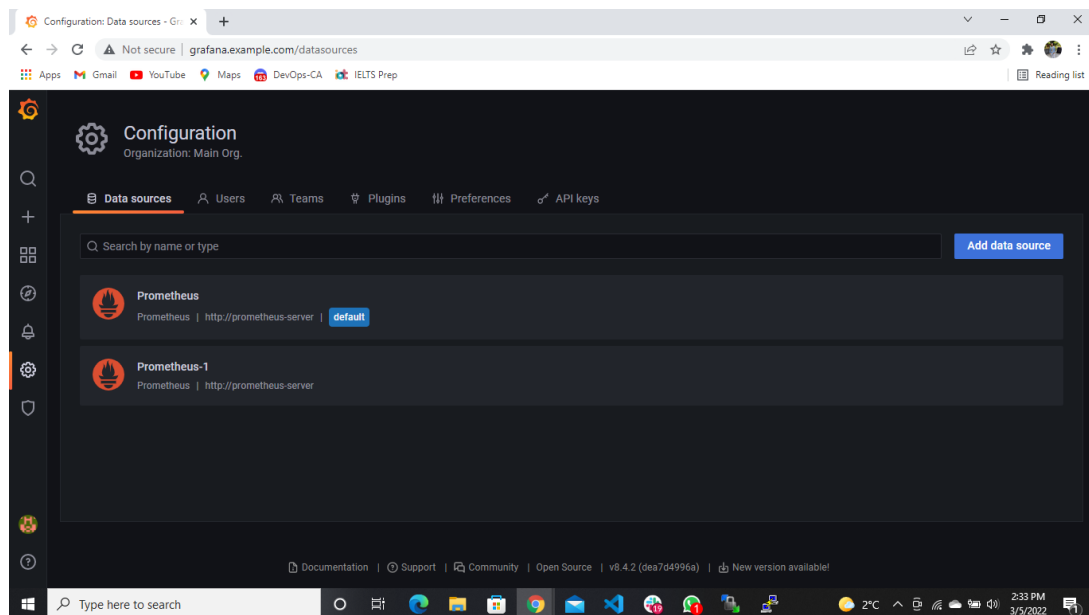
NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/my-nginx-ingress-nginx-controller-5857bf97fb	1	1	1	43m
replicaset.apps/my-release-grafana-5c779648bc	1	1	1	44m
replicaset.apps/prometheus-alertmanager-6fc469f46f	1	1	1	46m
replicaset.apps/prometheus-kube-state-metrics-6c44ff7fb6	1	1	1	46m
replicaset.apps/prometheus-pushgateway-6466f777cb	1	1	1	46m
replicaset.apps/prometheus-server-75848577fc	1	1	1	46m

```
root@ip-172-31-23-148: /home/ubuntu# clear  
root@ip-172-31-23-148: /home/ubuntu# kubectl get all
```

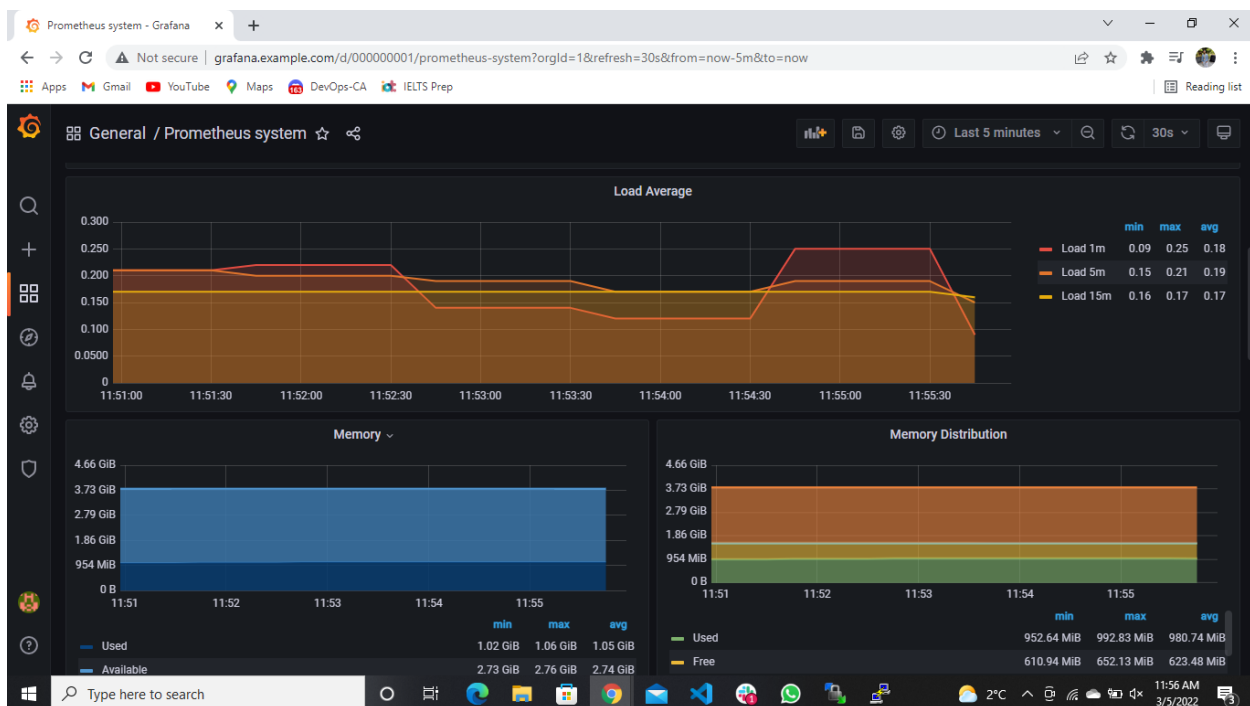
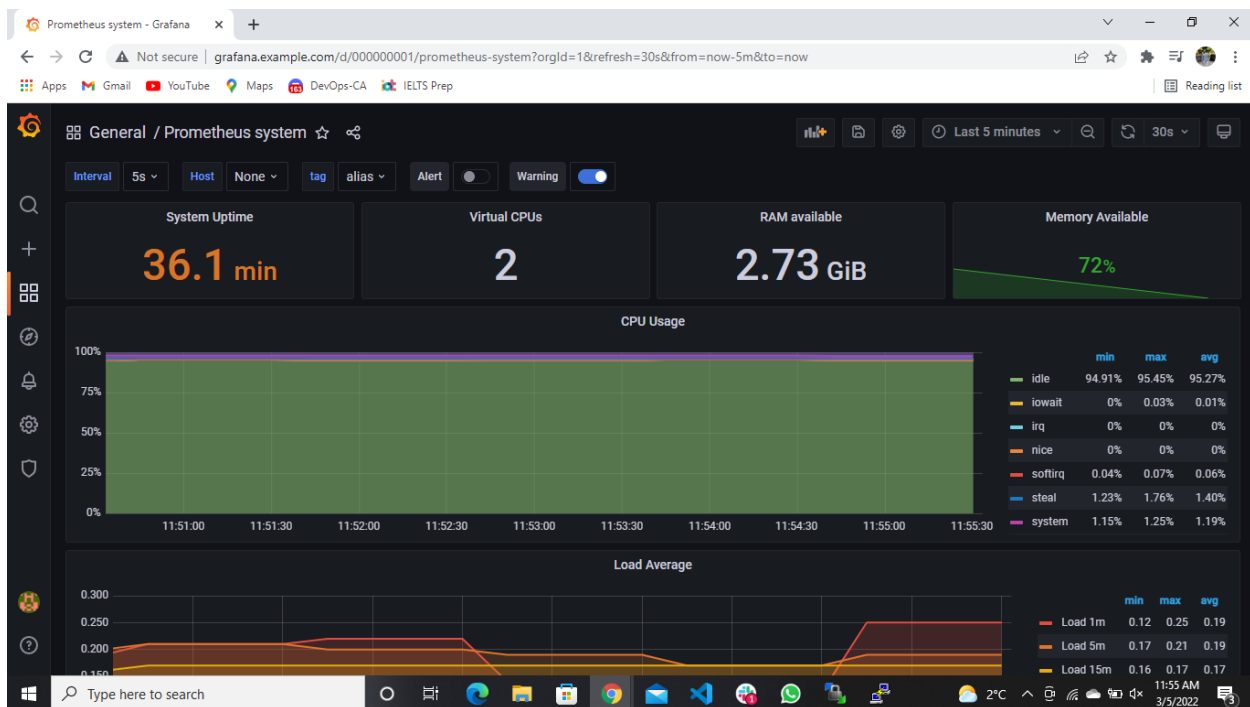
After successfully deploying all the resources on the cluster. We can see Prometheus is receiving the data as expected. It can be seen from the image below.

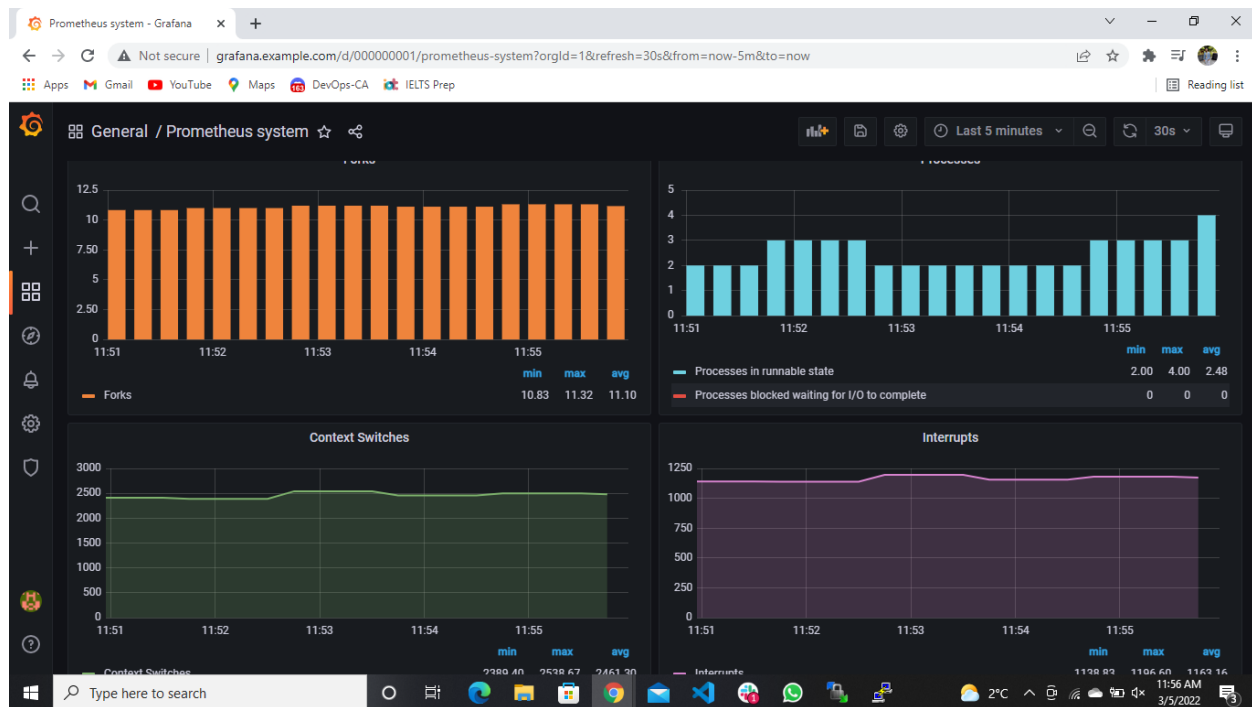


Once data is being observed from the Prometheus. Configure the Data Source section on Grafana and start visualizing over it.



Below Images shows Grafana is configured properly and using preferred dashboard, data is being visualized on Grafana.





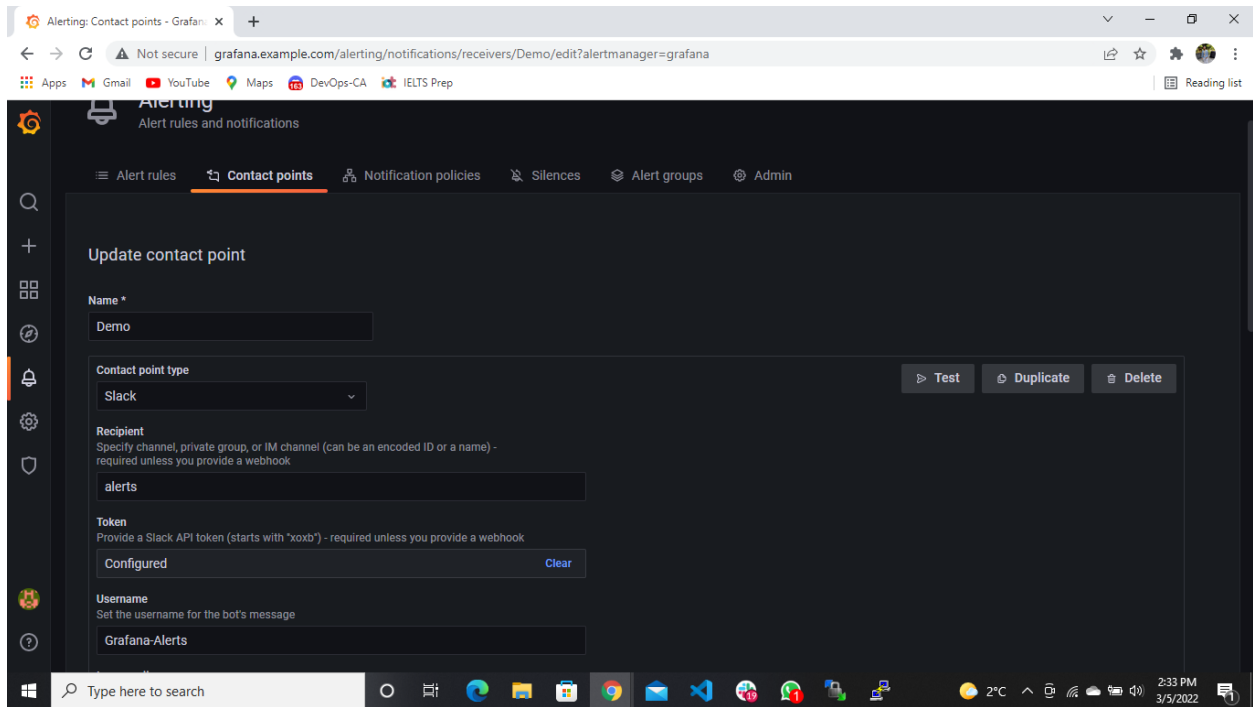
Almost done, here it is clearly seen that required data is flowing in Grafana and we can visualize it in a beautiful dashboard.

Let's configure Slack with Grafana so that, when a specific event occurs, we can get its message in Slack Channel.

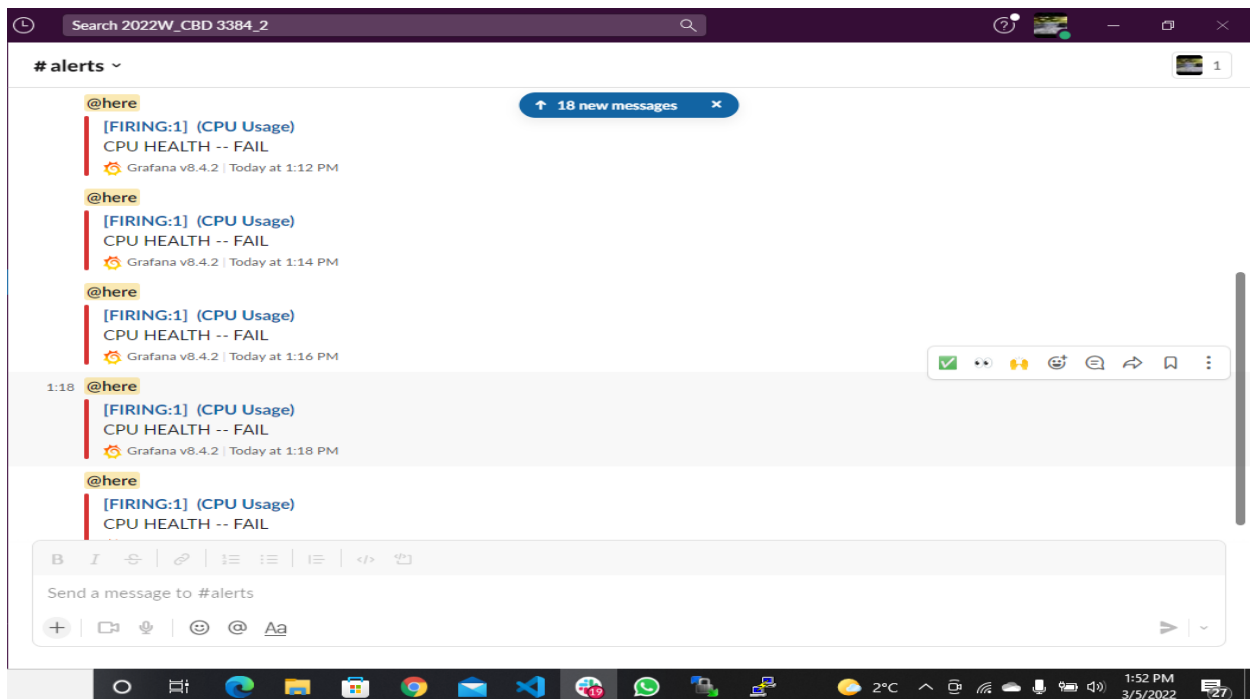
I have used this handy document to integrate Slack with Grafana.

["https://medium.com/@_oleksii_/grafana-alerting-and-slack-notifications-3affe9d5f688"](https://medium.com/@_oleksii_/grafana-alerting-and-slack-notifications-3affe9d5f688)

In contact points, I configured the Slack Webhook as shown in the image below.



Now, when my custom events trigger. It sends the message over slack channel. As shown in the image below.



Thank you,
Vaishal Shah