Prometheus Monitoring On Kubernetes Cluster

Connect to the Kubernetes Cluster

```
nikki@DESKTOP-I298DGN:~$ minikube start
😄  minikube v1.30.1 on Ubuntu 20.04
✨  Using the docker driver based on existing profile
👍  Starting control plane node minikube in cluster minikube
🚜  Pulling base image ...
🔄  Restarting existing docker container for "minikube" ...
🐳  Preparing Kubernetes v1.26.3 on Docker 23.0.2 ...
🔗  Configuring bridge CNI (Container Networking Interface) ...
🔎  Verifying Kubernetes components...
    ▪ Using image docker.io/kubernetesui/dashboard:v2.7.0
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
    ▪ Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
💡  Some dashboard features require the metrics-server addon. To enable all features please run:

        minikube addons enable metrics-server


🌟  Enabled addons: storage-provisioner, default-storageclass, dashboard
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

Create a Namespace & ClusterRole

1. Create namespace

```
nikki@DESKTOP-I298DGN:~$ kubectl create namespace monitoring
namespace/monitoring created
nikki@DESKTOP-I298DGN:~$
```

2. Create role

```
nikki@DESKTOP-I298DGN:~/kubernetes-monitoring$ kubectl create -f clusterRole.yaml
clusterrole.rbac.authorization.k8s.io/prometheus created
clusterrolebinding.rbac.authorization.k8s.io/prometheus created
nikki@DESKTOP-I298DGN:~/kubernetes-monitoring$
```

Create a Config Map To Externalize Prometheus Configurations

```
nikki@DESKTOP-I298DGN:~/kubernetes-monitoring$ kubectl create -f config-map.yaml
configmap/prometheus-server-conf created
nikki@DESKTOP-I298DGN:~/kubernetes-monitoring$
```

Create a Prometheus Deployment

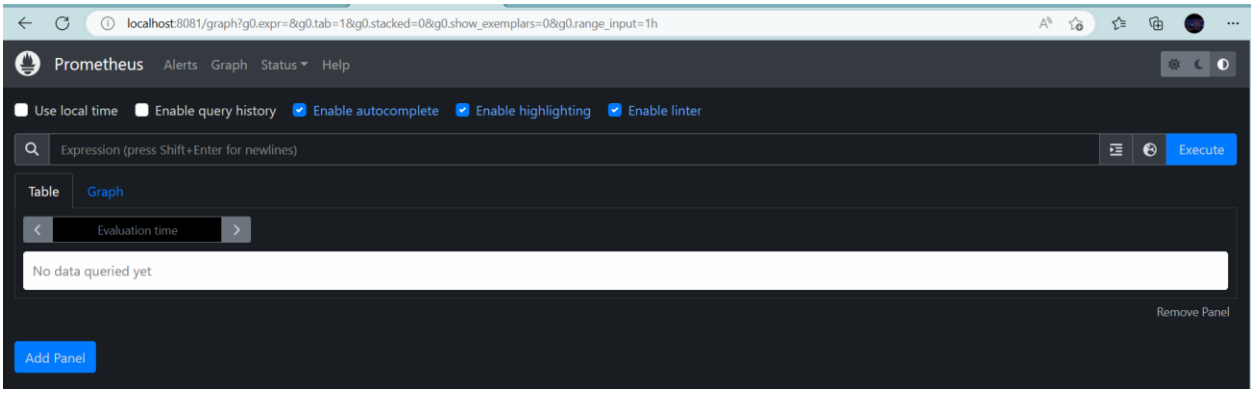
```
nikki@DESKTOP-I298DGN:~/kubernetes-monitoring$ kubectl create -f prometheus-deployment.yaml
deployment.apps/prometheus-deployment created
nikki@DESKTOP-I298DGN:~/kubernetes-monitoring$ kubectl get deployments --namespace=monitoring
NAME                    READY   UP-TO-DATE   AVAILABLE   AGE
prometheus-deployment   0/1     1            0           45s
nikki@DESKTOP-I298DGN:~/kubernetes-monitoring$
```

Connecting To Prometheus Dashboard

Method 1: Using Kubectl port forwarding

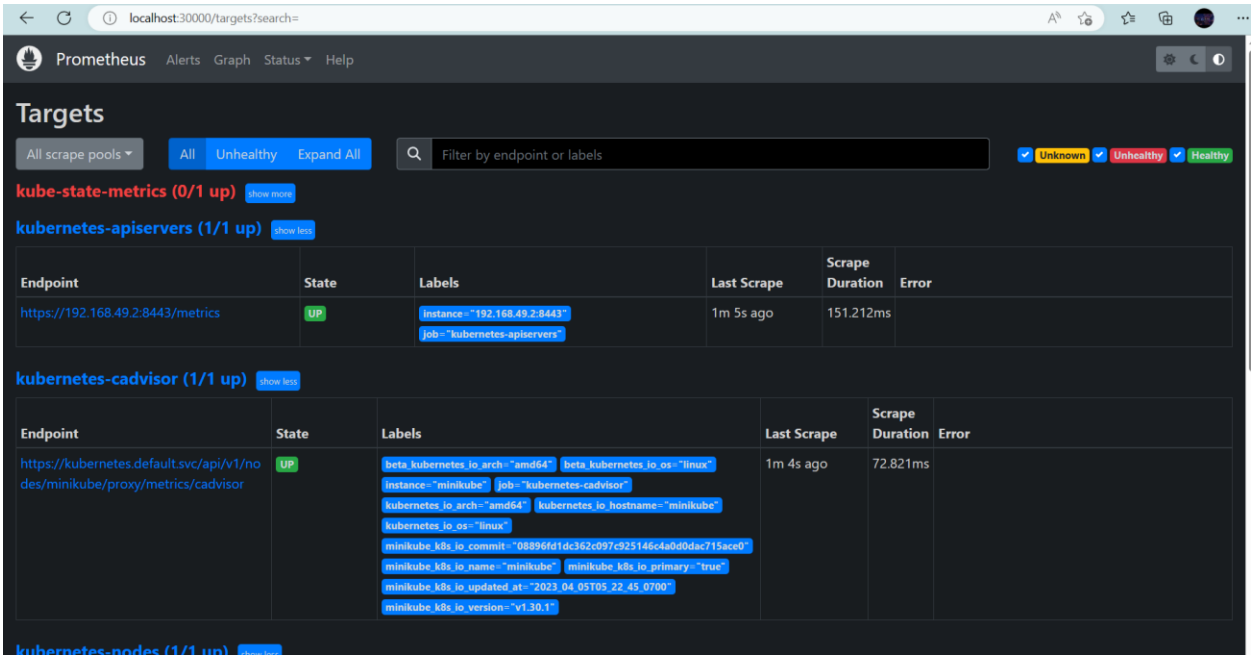
- I used port 8081, because port 8080 is occupied by Jenkins.

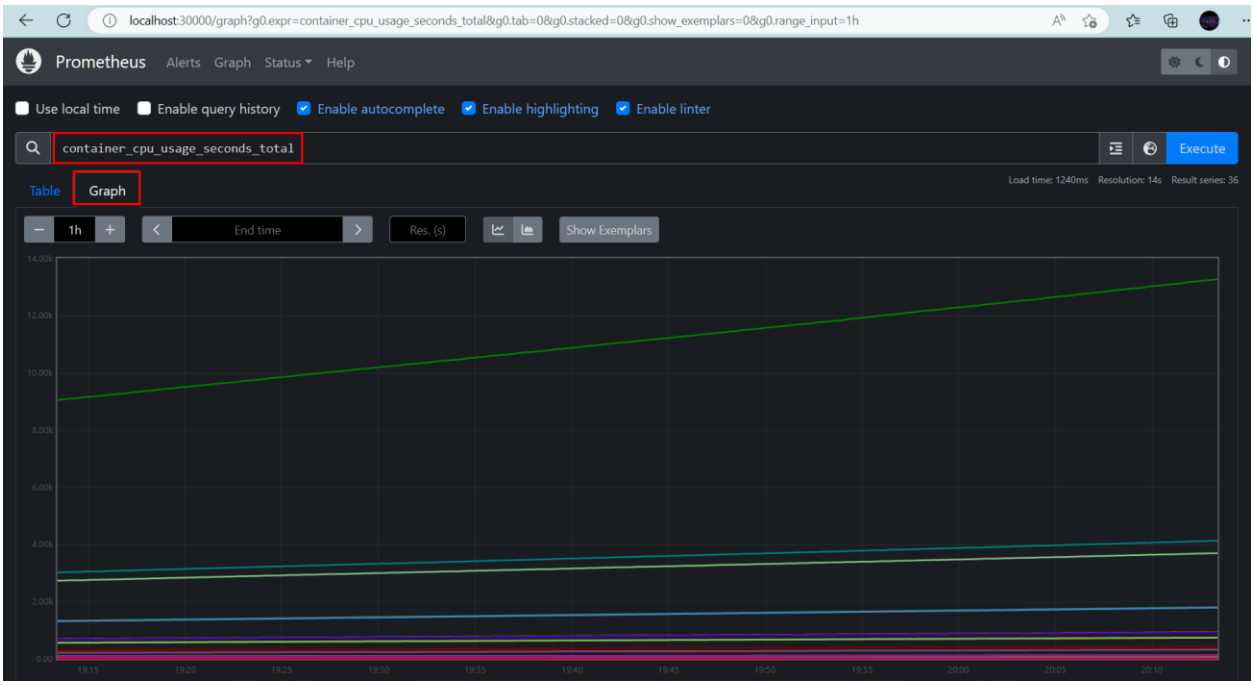
```
nikki@DESKTOP-I298DGN:~/kubernetes-monitoring$ kubectl port-forward prometheus-deployment-67cf879cc4-j6s59 8081:9090 -n monitoring
Forwarding from 127.0.0.1:8081 -> 9090
Forwarding from [::1]:8081 -> 9090
Handling connection for 8081
Handling connection for 8081
Handling connection for 8081
```



Method 2: Exposing Prometheus as a Service [NodePort & LoadBalancer]

```
^X^Cnikki@DESKTOP-I298DGN:~/kubernetes-monitoring$ kubectl create -f prometheus-service.yaml --namespace=monitoring
service/prometheus-service created
```

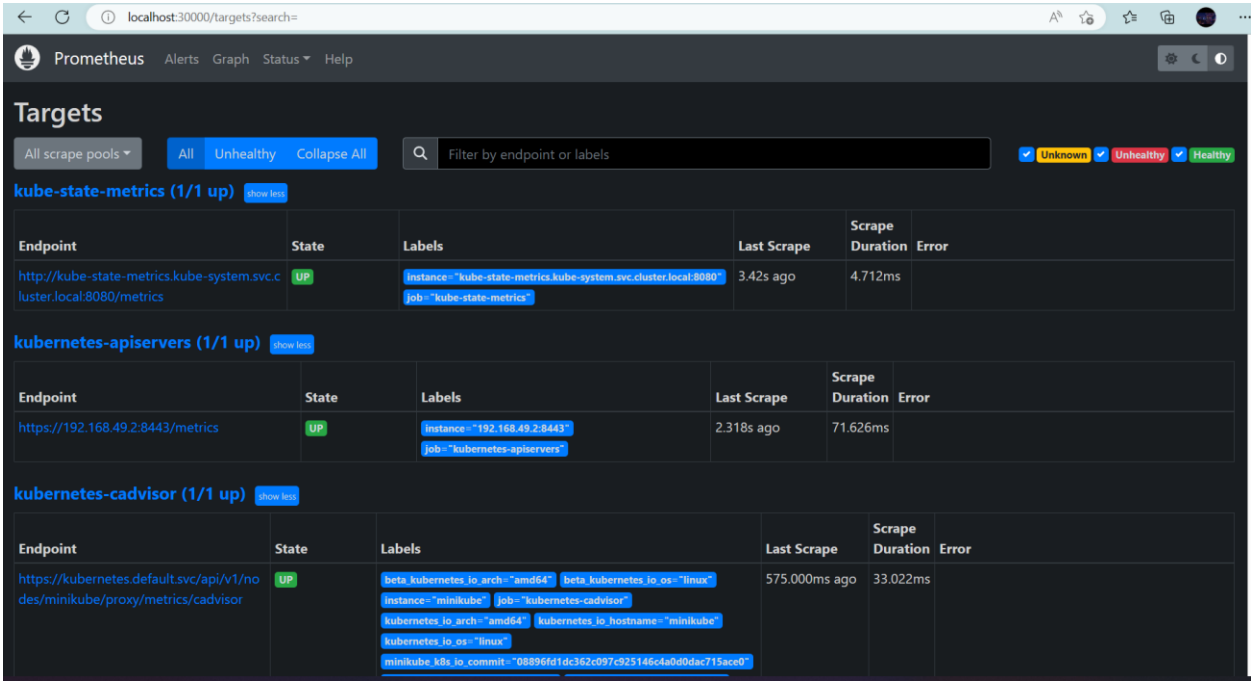




Setting Up Kube State Metrics

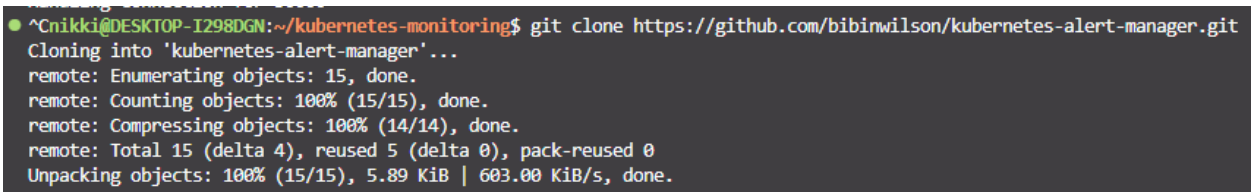
- I cloned the GitHub repository and applied the commands.

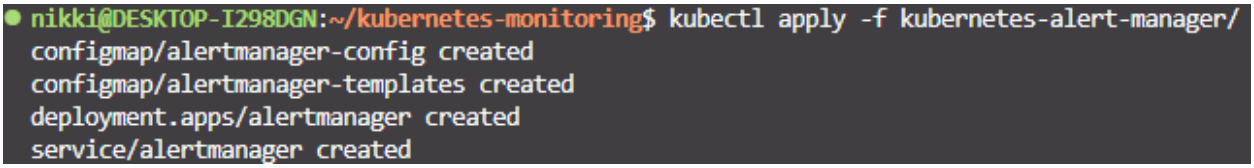
```
^Cnikki@DESKTOP-I298DGN:~/kubernetes-monitoring$ git clone https://github.com/devopscube/kube-state-metrics-configs.git
Cloning into 'kube-state-metrics-configs'...
remote: Enumerating objects: 25, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (24/24), done.
remote: Total 25 (delta 10), reused 1 (delta 0), pack-reused 0
Unpacking objects: 100% (25/25), 5.70 KiB | 389.00 KiB/s, done.
nikki@DESKTOP-I298DGN:~/kubernetes-monitoring$ kubectl apply -f kube-state-metrics-configs/
clusterrolebinding.rbac.authorization.k8s.io/kube-state-metrics created
clusterrole.rbac.authorization.k8s.io/kube-state-metrics created
deployment.apps/kube-state-metrics created
serviceaccount/kube-state-metrics created
service/kube-state-metrics created
nikki@DESKTOP-I298DGN:~/kubernetes-monitoring$ kubectl get deployments kube-state-metrics -n kube-system
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kube-state-metrics    0/1     1            0           10s
nikki@DESKTOP-I298DGN:~/kubernetes-monitoring$ kubectl port-forward prometheus-deployment-67cf879cc4-j6s59 30000:9090 -n monitoring
Forwarding from 127.0.0.1:30000 -> 9090
Forwarding from [::1]:30000 -> 9090
Handling connection for 30000
Handling connection for 30000
Handling connection for 30000
```

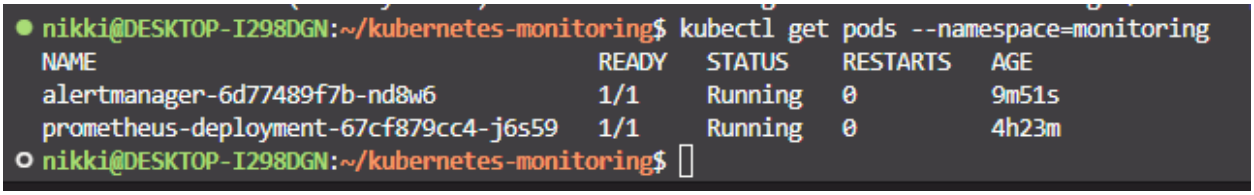


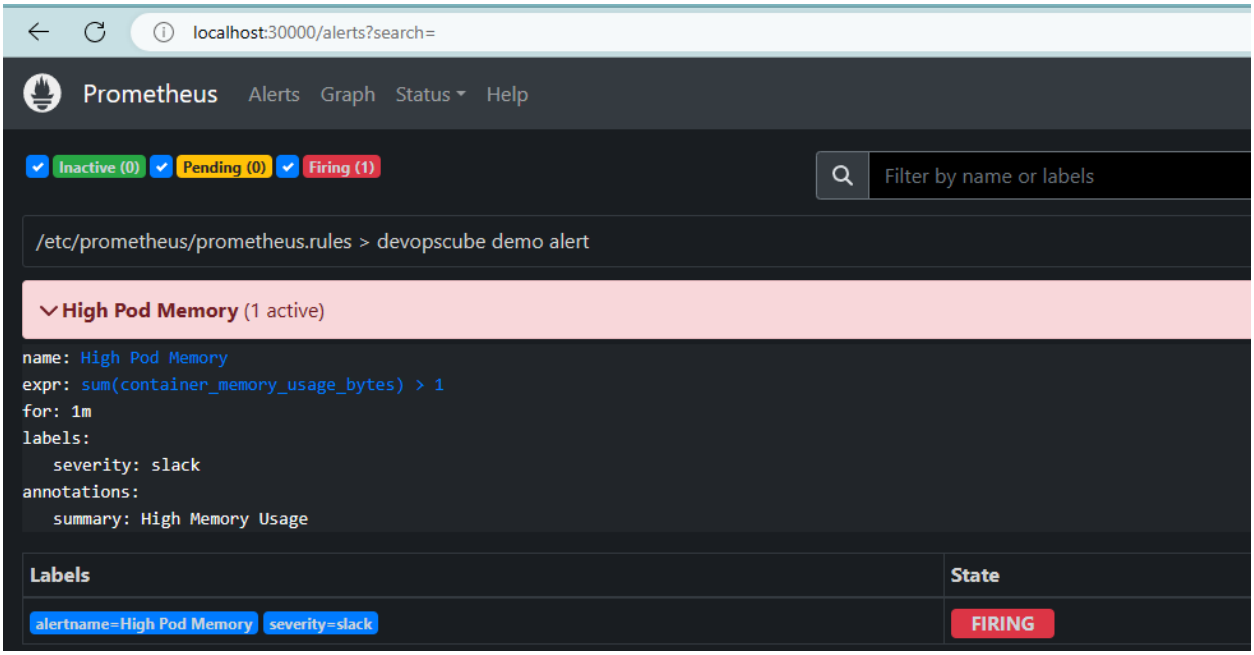
Setting Up Alertmanager

- I cloned the GitHub repository and applied the commands.



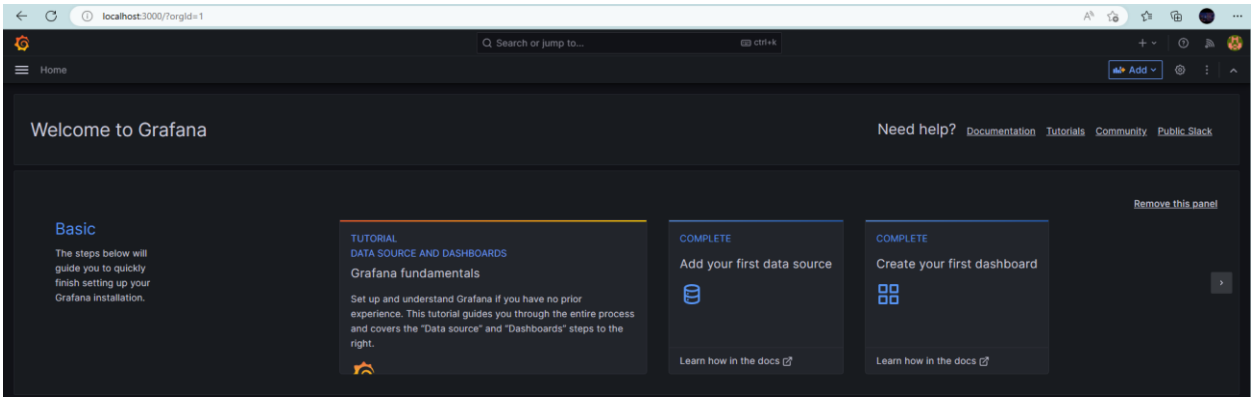


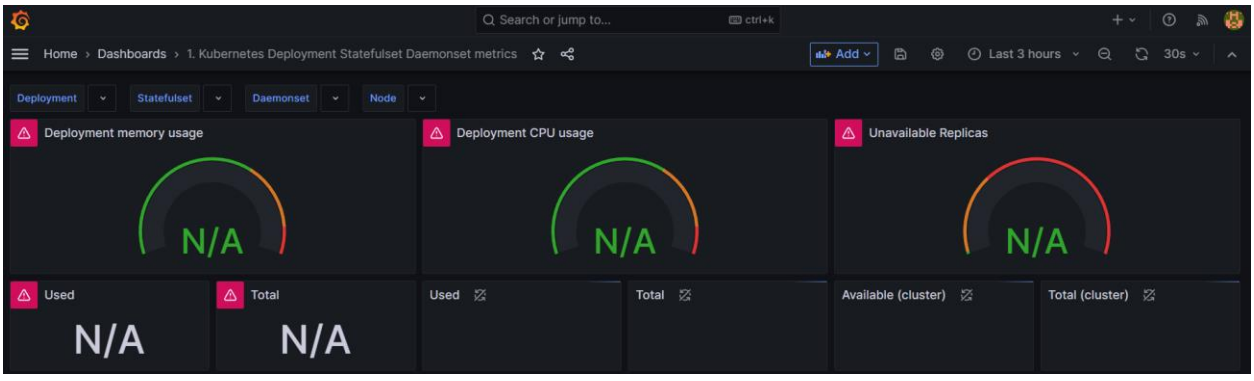




Setting Up Grafana

```
^Cnikki@DESKTOP-I298DGN:~/kubernetes-monitoring$ git clone https://github.com/bibinwilson/kubernetes-grafana.git
Cloning into 'kubernetes-grafana'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 9 (delta 0), reused 9 (delta 0), pack-reused 0
Unpacking objects: 100% (9/9), 1.37 KiB | 176.00 KiB/s, done.
nikki@DESKTOP-I298DGN:~/kubernetes-monitoring$ kubectl apply -f kubernetes-grafana/
deployment.apps/grafana created
configmap/grafana-datasources created
service/grafana created
nikki@DESKTOP-I298DGN:~/kubernetes-monitoring$ kubectl get pods --namespace=monitoring
NAME                                   READY   STATUS    RESTARTS   AGE
alertmanager-6d77489f7b-nd8w6          1/1     Running   0          22m
grafana-68b7b49968-hqfcv               1/1     Running   0          4m12s
prometheus-deployment-67cf879cc4-j6s59   1/1     Running   0          4h36m
nikki@DESKTOP-I298DGN:~/kubernetes-monitoring$ kubectl port-forward -n monitoring grafana-68b7b49968-hqfcv 3000
Forwarding from 127.0.0.1:3000 -> 3000
Forwarding from [::1]:3000 -> 3000
Handling connection for 3000
Handling connection for 3000
Handling connection for 3000
Handling connection for 3000
Handling connection for 3000
```





- I kept getting error for the Grafana Kubernetes dashboard.