

# 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/kubernetes/dashboard:v2.7.0
  ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
  ▪ Using image docker.io/kubernetes/metrics-server: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

```
ClusterRoleBinding PrometheusRole:authn.k8s.io/prometheus created
● 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

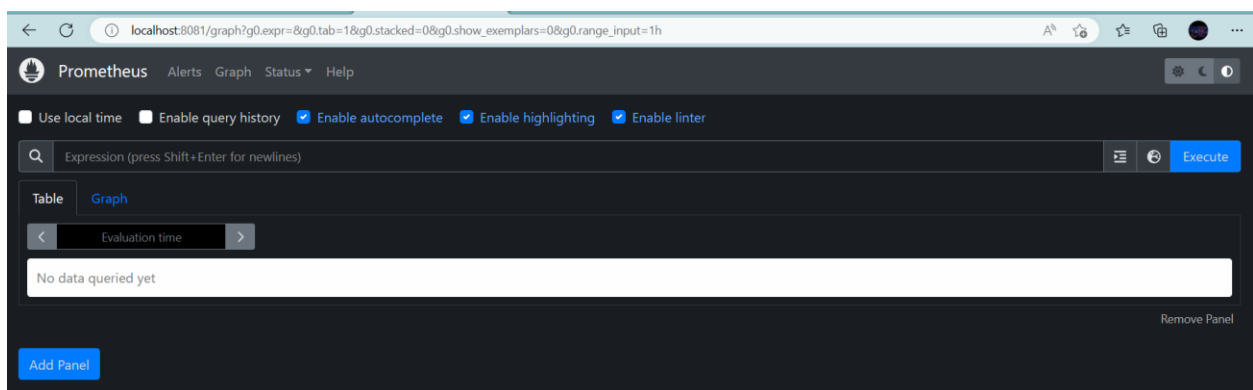
```
● nikki@DESKTOP-I298DGN:~/kubernetes-monitoring$ kubectl create -f prometheus-deployment.yaml
deployment.apps/prometheus-deployment created
● nikki@DESKTOP-I298DGN:~/kubernetes-monitoring$ kubectl get deployments --namespace=monitoring
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
prometheus-deployment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

## Connecting To Prometheus Dashboard

### Method 1: Using Kubectl port forwarding

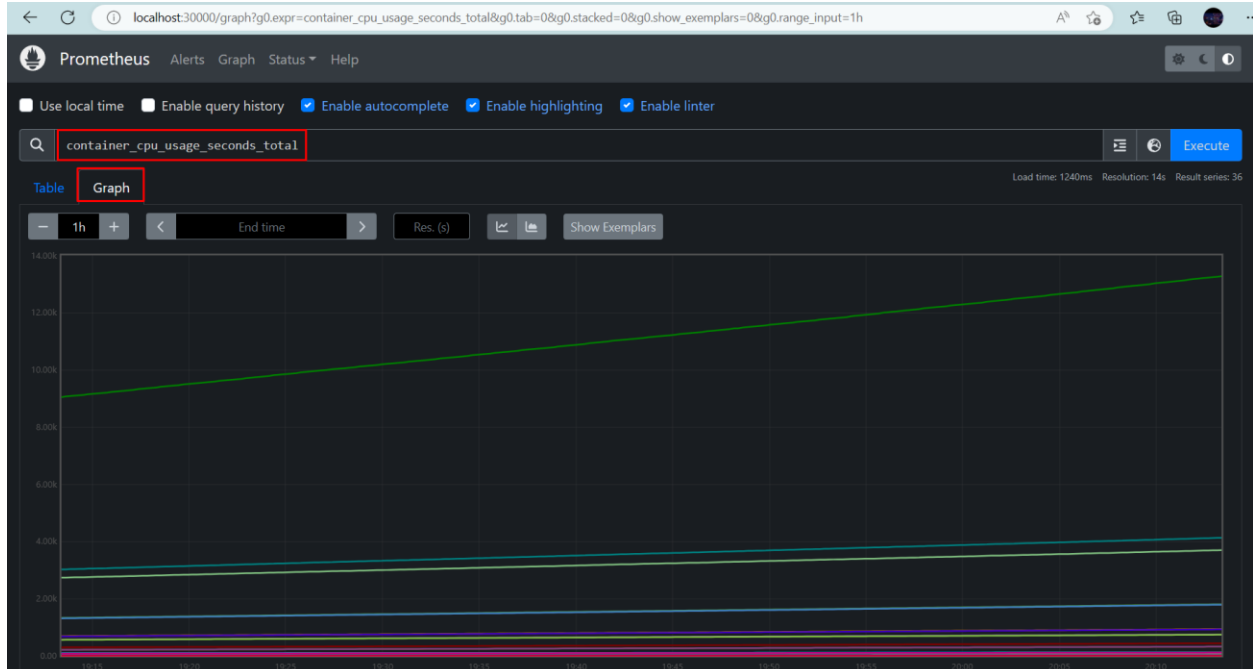
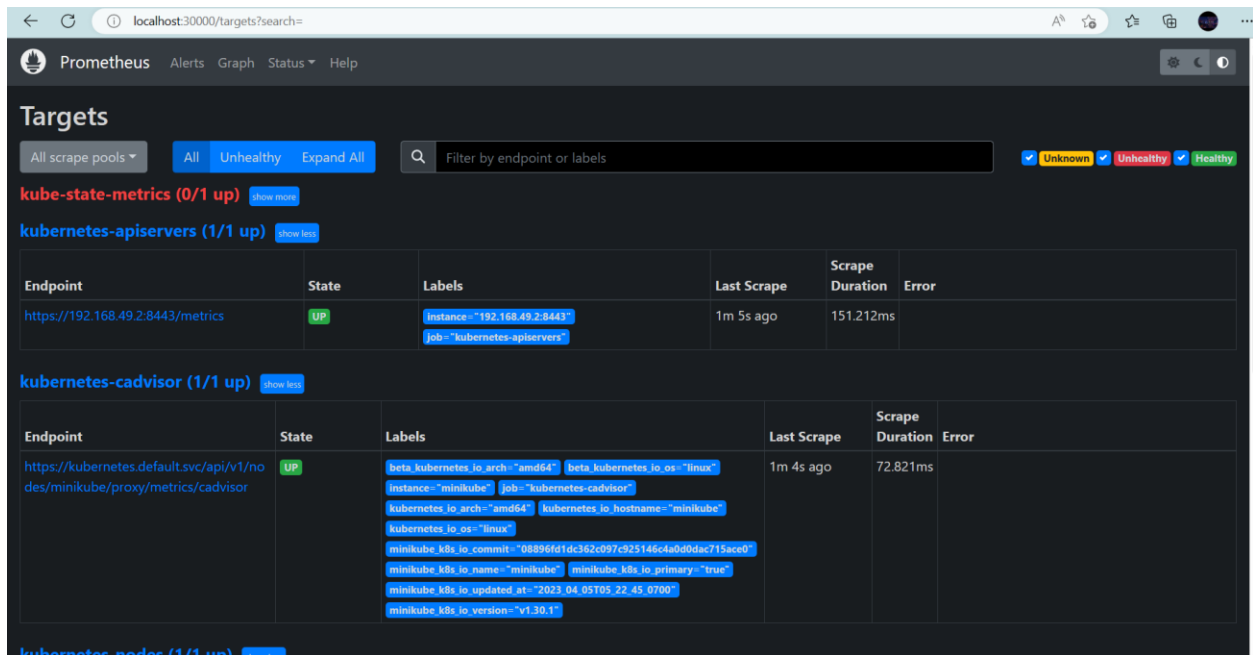
- I used port 8081, because port 8080 is occupied by Jenkins.

```
○ 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
[]
```

The screenshot shows the Prometheus web interface at localhost:30000. The 'Targets' tab is active, displaying a list of scrape pools. Each pool has a table of targets with columns for Endpoint, State, Labels, Last Scrape, Scrape Duration, and Error.

### kube-state-metrics (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://kube-state-metrics.kube-system.svc.cluster.local:8080/metrics	UP	instance="kube-state-metrics.kube-system.svc.cluster.local:8080" job="kube-state-metrics"	3.42s ago	4.712ms	

### kubernetes-apiservers (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
https://192.168.49.2:8443/metrics	UP	instance="192.168.49.2:8443" job="kubernetes-apiservers"	2.318s ago	71.626ms	

### kubernetes-cadvisor (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
https://kubernetes.default.svc/api/v1/nodes/minikube/proxy/metrics/cadvisor	UP	beta_kubernetes_io_arch="amd64" beta_kubernetes_io_os="linux" instance="minikube" job="kubernetes-cadvisor" kubernetes_io_arch="amd64" kubernetes_io_hostname="minikube" kubernetes_io_os="linux" minikube_k8s_io_commit="08896f61dc362c097c925146c4a00dac715ace0"	575.000ms ago	33.022ms	

## Setting Up Alertmanager

- I cloned the GitHub repository and applied the commands.

```
^nikki@DESKTOP-I298DGN:~/kubernetes-monitoring$ git clone https://github.com/bibinwilson/kubernetes-alert-manager.git
Cloning into 'kubernetes-alert-manager'...
remote: Enumerating objects: 15, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 15 (delta 4), reused 5 (delta 0), pack-reused 0
Unpacking objects: 100% (15/15), 5.89 KiB | 603.00 KiB/s, done.

nikki@DESKTOP-I298DGN:~/kubernetes-monitoring$ kubectl apply -f kubernetes-alert-manager/
configmap/alertmanager-config created
configmap/alertmanager-templates created
deployment.apps/alertmanager created
service/alertmanager created

nikki@DESKTOP-I298DGN:~/kubernetes-monitoring$ kubectl get pods --namespace=monitoring
NAME                                READY   STATUS    RESTARTS   AGE
alertmanager-6d77489f7b-nd8w6       1/1     Running   0           9m51s
prometheus-deployment-67cf879cc4-j6s59 1/1     Running   0           4h23m
nikki@DESKTOP-I298DGN:~/kubernetes-monitoring$
```

The screenshot shows the Prometheus Alertmanager web interface at `localhost:30000/alerts?search=`. The interface has a top navigation bar with 'Prometheus', 'Alerts', 'Graph', 'Status', and 'Help'. Below the navigation bar, there are filters for alert states: 'Inactive (0)', 'Pending (0)', and 'Firing (1)'. A search bar is also present. The main content area shows a breadcrumb trail: `/etc/prometheus/prometheus.rules > devopscube demo alert`. Below this, a section titled 'High Pod Memory (1 active)' is expanded, showing the alert details:

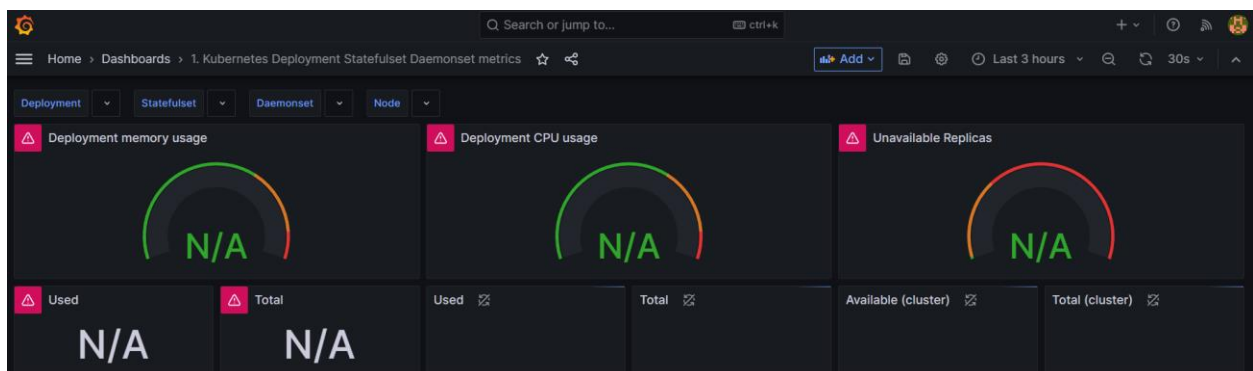
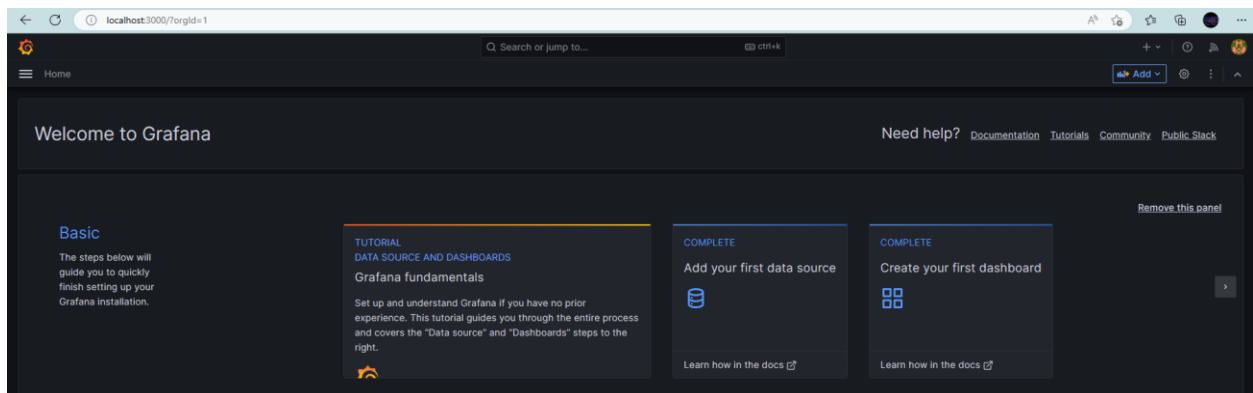
- name: High Pod Memory
- expr: `sum(container_memory_usage_bytes) > 1`
- for: 1m
- labels:
  - severity: slack
- annotations:
  - summary: High Memory Usage

At the bottom, there is a table with two columns: 'Labels' and 'State'. The 'Labels' column shows `alertname=High Pod Memory` and `severity=slack`. The 'State' column shows 'FIRING'.

Labels	State
<code>alertname=High Pod Memory</code> <code>severity=slack</code>	FIRING

## 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.