Create a Namespace & ClusterRole

First we need to created namespace for our monitoring using the following command, after we start the minikube

```
minikube start
```

create new namespace using

```
kubectl create namespace monitoring
```

To start the dashboard for kubernetis cluster run **minikube dashboard** and navigate to **Cluster / Namespaces** and we can see that the namespace that we create.

To verify that namespace **monitoring** is created in command line type:

kubectl get namespaces

```
ds@ds-HP-ProBook-440-G6:~$ kubectl get namespaces
NAME                   STATUS   AGE
default                Active   18m
kube-node-lease        Active   18m
kube-public            Active   18m
kube-system            Active   18m
kubernetes-dashboard   Active   16m
monitoring             Active   5m43s
ds@ds-HP-ProBook-440-G6:~$
```

Now lets create new file caled **clusterRole.yaml**

Note: In the role, given below, you can see that we have added ***get, list, and watch*** permissions to nodes, services endpoints, pods, and ingresses. The role binding is bound to the monitoring namespace. If you have any use case to retrieve metrics from any other object, you need to add that in this cluster role.

clusterRole.yaml

```
! clusterRole.yaml ×
monitoring > ! clusterRole.yaml > [ ] subjects > {} 0 > ▣ namespace
        io.k8s.api.rbac.v1.ClusterRoleBinding (v1@clusterrolebinding.json) | io.k8s.api.rbac.v1.ClusterRole (v1@clusterrole.json)
  1     apiVersion: rbac.authorization.k8s.io/v1
  2     kind: ClusterRole
  3     metadata:
  4       name: prometheus
  5     rules:
  6     - apiGroups: [""]
  7       resources:
  8       - nodes
  9       - nodes/proxy
 10       - services
 11       - endpoints
 12       - pods
 13       verbs: ["get", "list", "watch"]
 14     - apiGroups:
 15       - extensions
 16       resources:
 17       - ingresses
 18       verbs: ["get", "list", "watch"]
 19     - nonResourceURLs: ["/metrics"]
 20       verbs: ["get"]
 21     ---
 22     apiVersion: rbac.authorization.k8s.io/v1
 23     kind: ClusterRoleBinding
 24     metadata:
 25       name: prometheus
 26     roleRef:
 27       apiGroup: rbac.authorization.k8s.io
 28       kind: ClusterRole
 29       name: prometheus
 30     subjects:
 31     - kind: ServiceAccount
 32       name: default
 33       namespace: monitoring
```

run the following command to create the role

```
kubectl create -f clusterRole.yaml
```



```
ds@ds-HP-ProBook-440-G6:~/Documents/monitoring$ kubectl create -f clusterRole.yaml
clusterrole.rbac.authorization.k8s.io/prometheus created
clusterrolebinding.rbac.authorization.k8s.io/prometheus created
ds@ds-HP-ProBook-440-G6:~/Documents/monitoring$
```

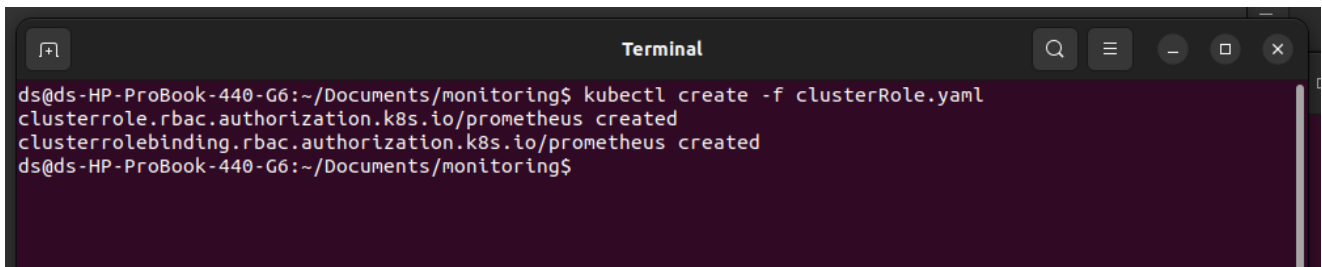# Create a Config Map To Externalize Prometheus Configurations

All configurations for Prometheus are part of prometheus.yaml file and all the alert rules for Alertmanager are configured in prometheus.rules.

prometheus.yaml: This is the main Prometheus configuration which holds all the scrape configs, service discovery details, storage locations, data retention configs, etc)

prometheus.rules: This file contains all the Prometheus alerting rules

Lets create new file **config-map.yaml** with the following content config-map.yaml and excute **kubectl create -f config-map.yaml** this command will create two new file **prometheus.yaml:** and **prometheus.rules:**

```
kubectl create -f config-map.yaml
```


The prometheus.yaml contains all the configurations to discover pods and services running in the Kubernetes cluster dynamically. We have the following scrape jobs in our Prometheus scrape configuration.

- **kubernetes-apiservers:** It gets all the metrics from the API servers.
- **kubernetes-nodes:** It collects all the kubernetes node metrics.
- **kubernetes-pods:** All the pod metrics get discovered if the pod metadata is annotated with prometheus.io/scrape and prometheus.io/port annotations.
- **kubernetes-cadvisor:** Collects all cAdvisor metrics.
- **kubernetes-service-endpoints:** All the Service endpoints are scrapped if the service metadata is annotated with prometheus.io/scrape and prometheus.io/port annotations. It can be used for black-box monitoring.

Create a Prometheus Deployment

Let`s create new file called **prometheus-deployment.yaml** with the following content.

```yaml
! prometheus-deployment.yaml 1 ×

! prometheus-deployment.yaml > {} spec > {} template > {} spec > [ ] containers > {} 0 > [ ] volumeMounts > {} 1 > ▣ name
    io.k8s.api.apps.v1.Deployment (v1@deployment.json)
 1   apiVersion: apps/v1
 2   kind: Deployment
 3   metadata:
 4     name: prometheus-deployment
 5     namespace: monitoring
 6     labels:
 7       app: prometheus-server
 8   spec:
 9     replicas: 1
10     selector:
11       matchLabels:
12         app: prometheus-server
13     template:
14       metadata:
15         labels:
16           app: prometheus-server
17       spec:
18         containers:
19           - name: prometheus
20             image: prom/prometheus
21             args:
22               - "--storage.tsdb.retention.time=12h"
23               - "--config.file=/etc/prometheus/prometheus.yml"
24               - "--storage.tsdb.path=/prometheus/"
25             ports:
26               - containerPort: 9090
27             resources:
28               requests:
29                 cpu: 500m
30                 memory: 500M
31               limits:
32                 cpu: 1
33                 memory: 1Gi
34             volumeMounts:
35               - name: prometheus-config-volume
36                 mountPath: /etc/prometheus/
37               - name: prometheus-storage-volume
38                 mountPath: /prometheus/
39         volumes:
40           - name: prometheus-config-volume
41             configMap:
42               defaultMode: 420
43               name: prometheus-server-conf
44
45           - name: prometheus-storage-volume
46             emptyDir: {}
```

Run the command ***kubectl create -f prometheus-deployment.yaml*** to create the deplyoment

```
kubectl create  -f prometheus-deployment.yaml
```

```
● ds@ds-HP-ProBook-440-G6:~/Documents/monitoring$ kubectl create  -f prometheus-deployment.yaml
  deployment.apps/prometheus-deployment created
○ ds@ds-HP-ProBook-440-G6:~/Documents/monitoring$ ▮
```

You can check the created deployment using the following command or you can check also from minikube dashboard.

```
kubectl get deployments --namespace=monitoring
```

```
deployment.apps/prometheus-deployment created
ds@ds-HP-ProBook-440-G6:~/Documents/monitoring$ kubectl get deployments --namespace=monitoring
NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
prometheus-deployment     1/1     1            1           3m27s
ds@ds-HP-ProBook-440-G6:~/Documents/monitoring$
```

Minikube dashboard
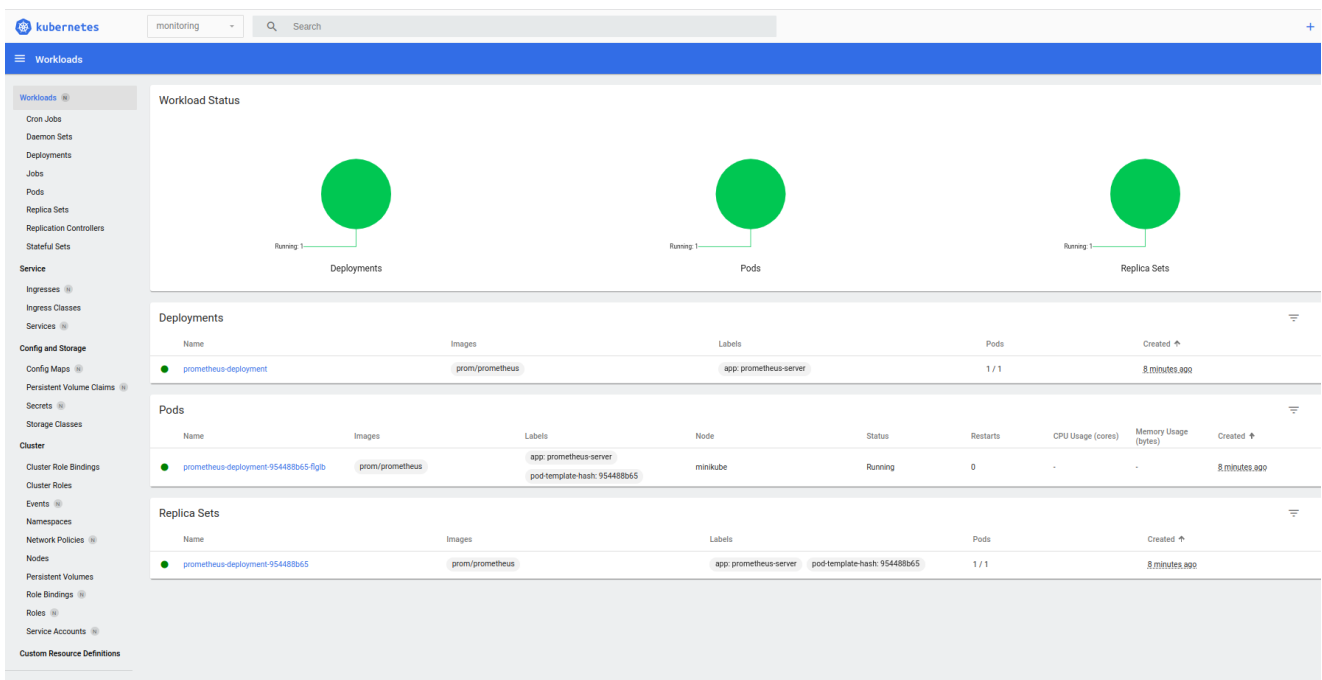
Connecting To Prometheus Dashboard

1. First get the pods name

   kubectl get pods --namespace=monitoring

```
error: the path "clusterRole.yaml" does not exist
ds@ds-HP-ProBook-440-G6:/media/ds/New Volume/sfa-homework/Prometheus Monitoring$ kubectl get pods --namespace=monitoring
NAME                                   READY   STATUS    RESTARTS   AGE
prometheus-deployment-954488b65-flglb  1/1     Running   0          10m
ds@ds-HP-ProBook-440-G6:/media/ds/New Volume/sfa-homework/Prometheus Monitoring$
```

2. execute the following command to start monitoring dashboard

   kubectl port-forward prometheus-deployment-954488b65-flglb 8080:9090 -n monitoring

```
ds@ds-HP-ProBook-440-G6:/media/ds/New Volume/sfa-homework/Prometheus Monitoring$ kubectl port-forward prometheus-deployment-954488b65-flglb 8080:9090 -n monitoring
Forwarding from 127.0.0.1:8080 -> 9090
Forwarding from [::1]:8080 -> 9090
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
```

You can open you browser to **http://localhost:8080** and view the dashboard
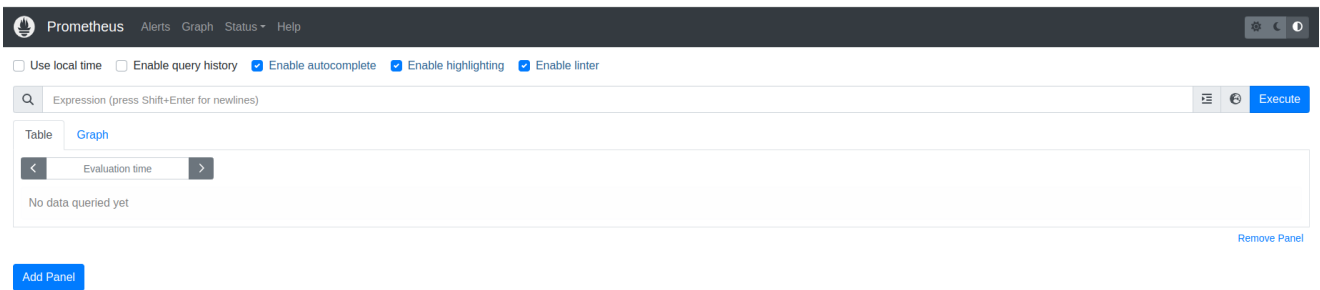
Method 2: Exposing Prometheus as a Service [NodePort & LoadBalancer]

To access the Prometheus dashboard over a IP or a DNS name, you need to expose it as a Kubernetes service.

Step 1: Create a file named prometheus-service.yaml and copy the following contents. We will expose Prometheus on all kubernetes node IP's on port 30000.

```yaml
apiVersion: v1
kind: Service
metadata:
name: prometheus-service
namespace: monitoring
annotations:
    prometheus.io/scrape: 'true'
    prometheus.io/port:   '9090'
spec:
selector:
    app: prometheus-server
type: NodePort
ports:
    - port: 8080
    targetPort: 9090
    nodePort: 30000
```

Step 2: Create the service using the following command.

```
kubectl create -f prometheus-service.yaml --namespace=monitoring
```

```
ds@ds-HP-ProBook-440-G6:~/Documents/monitoring$ kubectl create -f prometheus-service.yaml --namespace=monitoring
service/prometheus-service created
ds@ds-HP-ProBook-440-G6:~/Documents/monitoring$ 
```