Demo Project: Install Prometheus Stack in Kubernetes

Project Description

In this project, we will:

- **Set up an EKS cluster** using `eksctl`.
- **Deploy Prometheus, Alert Manager, and Grafana** as part of the Prometheus Operator using a Helm chart.

Step 1: Create EKS cluster

- **Create an EKS cluster** in the AWS default region using default AWS credentials with 2 worker nodes: `eksctl create cluster`
- **Confirm the nodes** are created: `kubectl get node`

Step 2: Deploy Microservices Application

- **Clone the microservices repository**: `git@gitlab.com:twn-devops-projects/prometheus/monitoring.git`
- **Navigate into the project directory**: `cd monitoring`

- **Deploy the microservices:** `kubectl apply -f config-microservices.yaml`
- **Verify the deployed microservices:** `kubectl get pod`

  Example output:

```
NAME                                 READY   STATUS    RESTARTS   AGE
adservice-7f5dc4b75f-ns9pl           1/1     Running   0          29s
cartservice-7986dbd956-smr4w         1/1     Running   0          29s
checkoutservice-86bdcfbbfc-tzqgz     1/1     Running   0          28s
currencyservice-6f7fd6989-g6gj8      1/1     Running   0          31s
emailservice-6df48986c8-hxtd6        1/1     Running   0          33s
frontend-6f7d9ff6bc-bjf94            1/1     Running   0          28s
paymentservice-c5776df6f-8r8gz       1/1     Running   0          32s
productcatalogservice-7856db7589-vsw47   1/1     Running   0          32s
recommendationservice-6bd89f88fc-5mssn   1/1     Running   0          33s
redis-cart-8c5bbbccf-qgcnp           1/1     Running   0          27s
shippingservice-5cc877bc4c-pw6ml     1/1     Running   0          30s
```

Step 3: Deploy Prometheus Stack using Helm

1. **Add the Prometheus Helm repository and update it**:

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

2. **Create a dedicated namespace for montoring:**

```
kubectl create namespace monitoring
```

3. **Deploy Prometheus stack**:

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

4. **Confirm the deployment:** `kubectl get all -n monitoring`
5. **Check StatefulSets to see where Prometheus and Alert Manager store data:**

   `kubectl get statefulset -n monitoring`

   Example output:

```
NAME                                               READY   AGE
alertmanager-monitoring-kube-prometheus-alertmanager   1/1     2m4s
prometheus-monitoring-kube-prometheus-prometheus       1/1     2m4s
```

6. **Describe the StatefulSets and generate yamls for more details:**
   - `kubectl describe statefulset prometheus-monitoring-kube-prometheus-prometheus -n monitoring > prom.yaml`
   - `kubectl describe statefulset alertmanager-monitoring-kube-prometheus-alertmanager -n monitoring > alerts.yaml`
7. **Check deployed Deployments:** `kubectl get deployment -n monitoring`

   Example output:

```
NAME                                  READY   UP-TO-DATE   AVAILABLE   AGE
monitoring-grafana                    1/1     1            1           4m38s
monitoring-kube-prometheus-operator   1/1     1            1           4m38s
```

8. **Describe the deployment for further insights:** `kubectl describe deployment monitoring-kube-prometheus-operator -n monitoring > oper.yaml`

Step 4: Access Prometheus and Grafana Dashboards

- **Check Prometheus Stack Pods:**

  `kubectl get all -n monitoring`

```
NAME                                                 TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                      AGE
service/alertmanager-operated                        ClusterIP   None             <none>        9093/TCP,9094/TCP,9094/UDP   14m
service/monitoring-grafana                           ClusterIP   10.100.133.64    <none>        80/TCP                       14m
service/monitoring-kube-prometheus-alertmanager      ClusterIP   10.100.195.150   <none>        9093/TCP,8080/TCP            14m
service/monitoring-kube-prometheus-operator          ClusterIP   10.100.114.109   <none>        443/TCP                      14m
service/monitoring-kube-prometheus-prometheus        ClusterIP   10.100.132.60    <none>        9090/TCP,8080/TCP            14m
service/monitoring-kube-state-metrics                ClusterIP   10.100.192.221   <none>        8080/TCP                     14m
service/monitoring-prometheus-node-exporter          ClusterIP   10.100.108.12    <none>        9100/TCP                     14m
service/prometheus-operated                          ClusterIP   None             <none>        9090/TCP                     14m
```

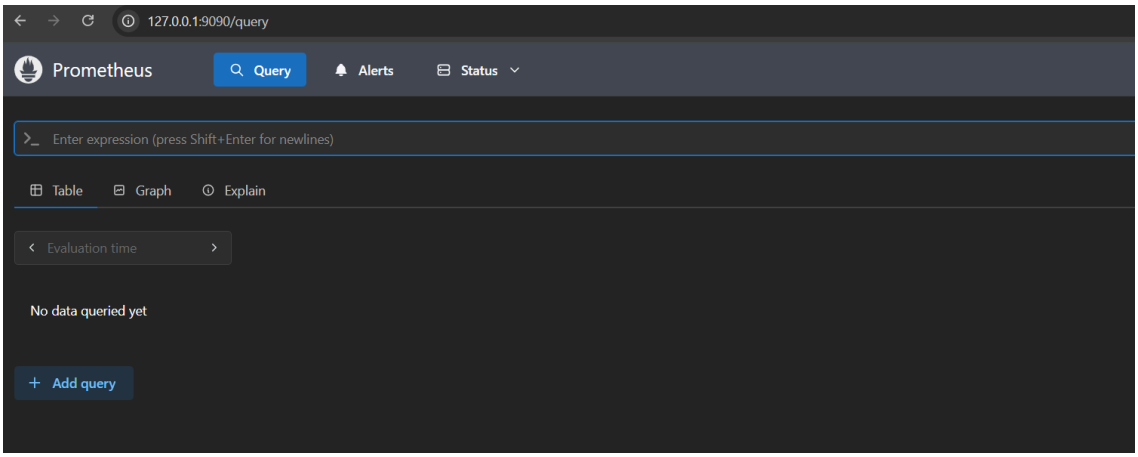
1. Access Prometheus UI

1. **Port forward Prometheus service:**

   ```
   kubectl port-forward svc/monitoring-kube-prometheus-prometheus 9090:9090 -n monitoring &
   ```

2. Access **Prometheus UI**:

   - Open a browser and go to `127.0.0.1:9090`



4. **Check monitored targets**:

   - Click **Status → Targets** to see what Prometheus is monitoring.

5. **View Metrics**:

   - From the main page, type `cpu` in the search bar or explore the **list of all available metrics**.

2. Access Grafana UI

1. **Port forward Grafana service**:

```
kubectl port-forward svc/monitoring-grafana 8080:80 -n monitoring &
```

2. **Access Grafana UI**:
    - Open a browser and go to `127.0.0.1:8080`
3. **Log in to Grafana**:
    - user: `admin`
    - pwd: `prom-operator`