Instalación de Grafana y Prometheus en Minikube mediante Helm

Benjamin Martinez para Zulunity

1 Instalación de Grafana

Este documento proporciona los pasos para desplegar Grafana y Prometheus en un clúster de Minikube utilizando Helm.

- **Paso 1: Agregar el repositorio de Helm de Grafana**
  Se ejecuta el siguiente comando para agregar el repositorio oficial de Grafana:

  ```
  helm repo add grafana https://grafana.github.io/helm-charts
  ```

- **Paso 2: Actualizar los repositorios de Helm**
  Se actualizan los repositorios de Helm para asegurarte de tener la última versión de los charts:

  ```
  helm repo update
  ```

- **Paso 3: Crear un Namespace para Grafana**
  Se crea un namespace específico para Grafana:

  ```
  kubectl create namespace monitoring
  ```

- **Paso 4: Desplegar Grafana en Minikube**
  Se ejecuta el siguiente comando para desplegar Grafana en el namespace `monitoring`:

  ```
  helm install grafana grafana/grafana --namespace monitoring
  ```

- **Paso 5: Verificar el Estado del Despliegue**
  Se puede verificar el estado del despliegue de Grafana con el siguiente comando:

  ```
  kubectl get pods -n monitoring
  ```

- **Paso 6: Exponer Grafana en Minikube**
  Si se desea exponer Grafana a través de un puerto en Minikube, se ejecuta:

  ```
  kubectl port-forward svc/grafana 3000:80 -n monitoring
  ```

  Luego, se accede a Grafana desde el navegador en `http://localhost:3000`.

2 Instalación de Prometheus

- **Paso 1: Agregar el repositorio de Helm de Prometheus**

  Para instalar Prometheus, primero se agrega el repositorio oficial de Helm de Prometheus:

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

- **Paso 2: Actualizar los repositorios de Helm**

  Se ejecuta el siguiente comando para asegurarse de que los repositorios de Helm estén actualizados:

  ```
  helm repo update
  ```

- **Paso 3: Crear un Namespace para Prometheus**

  En este caso, usaremos el mismo namespace `monitoring` creado para Grafana:

  ```
  kubectl create namespace monitoring
  ```

- **Paso 4: Desplegar Prometheus en Minikube**

  Para desplegar Prometheus, se usa el siguiente comando en el namespace `monitoring`:

  ```
  helm install prometheus prometheus-community/prometheus --namespace monitoring
  ```

- **Paso 5: Verificar el Estado del Despliegue**

  Luego se verifica el estado de Prometheus con el siguiente comando:

  ```
  kubectl get pods -n monitoring
  ```

3 Consultas de Monitoreo en Prometheus

Esta sección proporciona algunas consultas útiles para monitorear el uso de recursos en el clúster, como CPU, memoria y el estado de los pods.

- **Uso de CPU por Nodo**

  ```
  sum(rate(container_cpu_usage_seconds_total{job="kubelet", image!="",
      container!="POD"}[5m])) by (node)
  ```

- **Memoria Usada por Nodo**

  ```
  sum(container_memory_usage_bytes{job="kubelet", image!=""}) by (node)
  ```

- **Uso de CPU por Pod**

  ```
  sum(rate(container_cpu_usage_seconds_total{namespace="<namespace>"}[5m])) by (pod)
  ```

- **Memoria Usada por Pod**

  ```
  sum(container_memory_usage_bytes{namespace="<namespace>"}) by (pod)
  ```

- **Estatus del Pod**

```
count(kube_pod_status_phase{namespace="<namespace>", phase="Running"}) by (pod)
```

- **Número de Pods en un Namespace**

```
count(kube_pod_info{namespace="<namespace>"}) by (pod)
```

- **Uso de Almacenamiento de Persistent Volumes**

```
kubelet_volume_stats_used_bytes{namespace="<namespace>"}
```

- **Pods en Estado CrashLoopBackOff**

```
count(kube_pod_container_status_restarts_total{namespace="<namespace>",
    reason="CrashLoopBackOff"}) by (pod)
```

Nota: Se tiene que reemplazar `<namespace>` con el nombre específico de tu namespace y `<deployment>` con el nombre de tu despliegue al usar estas consultas.