Lab 5

DevOps Lab Manual: Monitoring and Logging with Prometheus and Grafana

Objective

- Learn how to set up Prometheus and Grafana for monitoring and logging.
- Monitor a Python application deployed in Kubernetes.
- Visualize metrics using Grafana dashboards.

1. Install Kubectl

```
- curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
```

Make the binary file executable

```
- chmod +x kubectl
```

Move binary to executable directory

```
- sudo mv kubectl /usr/local/bin/
```

Verify installation

```
- kubectl version --client --output=yaml
```

Configure kubectl for minikube

Do check if minikube is installed and working. If not then go to previous software installation manual and install kubernetes on system. It will work more properly if you restart system after installation

```
- minikube start
```

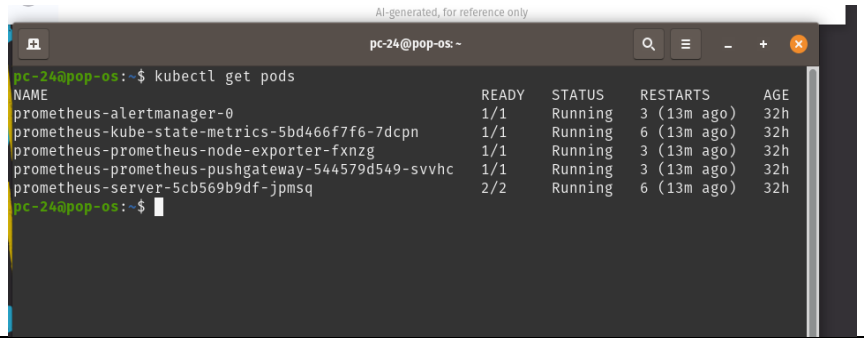
Verify minikube cluster

Check cluster information

- Kubectl cluster-info

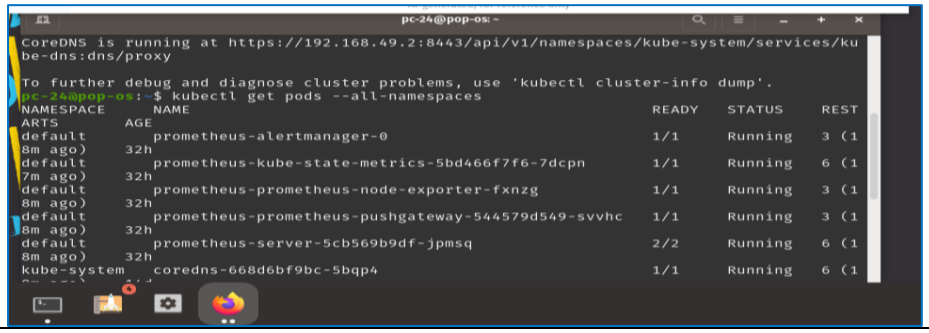
List all running nodes

- kubectl get nodes



Check the namesapce that kubectl has created for Promethous , which is 'default' by default.

- kubectl get pods --all-namespaces



2. Install Helm " required for both Prometheus and Grafana"

- curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash
- 3helm version

3. Install Prometheus using Helm

Add repository for Prometheus Helm

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

No need to Install Prometheus as it is already running in kubectle

4. Install Grafana using Helm

   Add helm Bitnami repository

   - helm repo add bitnami https://charts.bitnami.com/bitnami

   Add Grafana Hem repository

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

Update the Helm reposirty

- helm repo update

To access Grafana , get admin password and save it

- kubectl get secret grafana-admin --namespace default -o jsonpath="{.data.GF_SECURITY_ADMIN_PASSWORD}" | base64 --decode
-

Expose grafana

- kubectl port-forward service/grafana 3000:3000

Access Grafana at http://localhost:3000

Create application to deploy:

Create a directory and add the following files of applications:

Layout:

```
python-app/
├── app.py
├── requirements.txt
├── Dockerfile
└── deployment.yaml
```

App.py:

```
from flask import Flask from prometheus_client import start_http_server, Counter

app = Flask(name) REQUEST_COUNT = Counter('app_request_count', 'Total number of requests')

@app.route('/') def hello_world(): REQUEST_COUNT.inc() # Increment the request count return 'Hello, DevOps!'

if name == 'main': start_http_server(8000) # Start Prometheus metrics server on port 8000 app.run(host='0.0.0.0', port=5000) # Run the Flask app on port 5000
```

Dockerfile

```
Use an official Python runtime as a parent image

FROM python:3.9-slim

Set the working directory in the container

WORKDIR /app

Copy the requirements file into the container

COPY requirements.txt .

Install any needed packages specified in requirements.txt

RUN pip install --no-cache-dir -r requirements.txt

Copy the rest of the application code

COPY . .

Make port 5000 available to the world outside this container

EXPOSE 5000

Run the application

CMD ["python", "app.py"]
```

Requirements.txt

```
Flask==2.0.1
prometheus-client==0.12.0
```

Deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: python-app
spec:
  replicas: 2
  selector:
```

```
    matchLabels:
      app: python-app
  template:
    metadata:
      labels:
        app: python-app
    spec:
      containers:
      - name: python-app
        image: python-app
        ports:
        - containerPort: 5000
          name: http
        - containerPort: 8000
          name: metrics
---
apiVersion: v1
kind: Service
metadata:
  name: python-app
spec:
  selector:
    app: python-app
  ports:
    - protocol: TCP
      port: 5000
      targetPort: 5000
      name: http
    - protocol: TCP
      port: 8000
      targetPort: 8000
      name: metrics
  type: NodePort
```

rebuild application and deploy the application

Rebuild docker image

```
- docker build -t python-app .
```

Update the kubernetes deployment

```
- kubectl apply -f deployment.yaml
```

Lab Task:

Configure Grafana dashboard to visualize metrics using any query:

- Add Prometheus as a Data Source
  - In Grafana, go to **Connection > Data Sources**.
  - Click **Add data source** and select **Prometheus**.
  - Set the URL to http://prometheus-server:80.
  - Click **Save & Test**.
- Create a Dashboard in Grafana:
  - Go to **Create > Dashboard**.
  - Add a new panel and select the Prometheus data source.
  - Use the query app_request_count to visualize the request count.
  - Customize the panel and save the dashboard.