Prometheus & Grafana Monitoring Setup on EC2 using Minikube (or K3s)

Overview

This setup involves running Prometheus and Grafana on an Ubuntu EC2 instance using Minikube or K3s. Additionally, a Node.js application exposing Prometheus metrics is deployed and monitored via Grafana.

Tools & Versions

- **EC2 OS:** Ubuntu 24.04
- **Kubernetes:** K3s or Minikube (latest)
- **Prometheus & Grafana:** Deployed via Helm
- **App Container:** pradeepaanandh/node-prom-app

Step-by-Step Setup

1. Install K3s Kubernetes (Alternative to Minikube)

```
curl -sfL https://get.k3s.io | sh -
```

- To check status:

```
sudo systemctl status k3s
```

- Check nodes:

```
kubectl get nodes
```

> ✅**Note:** `kubectl` is already available in K3s by default. K3s internally maps `kubectl` to `k3s kubectl`, so no need for separate `kubectl` installation or alias setup.

2. Install Helm

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

- Verify:

```
helm version
```

3. Create Monitoring Namespace

```
kubectl create namespace monitoring
```

4. Prometheus & Grafana Installation

- Add Helm repo:

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

- Install Prometheus Stack:

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

5. Expose Services via NodePort

- Edited Grafana and Prometheus services:

```
kind: Service
metadata:
  name: grafana
  namespace: monitoring
spec:
  type: NodePort
  ports:
    - port: 80
      targetPort: 3000
      nodePort: 30877
---
kind: Service
metadata:
  name: prometheus-server
  namespace: monitoring
spec:
  type: NodePort
  ports:
    - port: 80
      targetPort: 9090
      nodePort: 32683
```

- Final NodePorts:

- Grafana: `30877`
- Prometheus: `32683`

6. Node.js Application Deployment

- **Deployment.yaml:**

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: node-prom-app
  labels:
    app: node-prom
spec:
  replicas: 1
  selector:
    matchLabels:
      app: node-prom
  template:
    metadata:
      labels:
        app: node-prom
      annotations:
        prometheus.io/scrape: "true"
        prometheus.io/port: "3000"
    spec:
      containers:
        - name: node-prom
          image: pradeepaanandh/node-prom-app
          ports:
            - containerPort: 3000
          livenessProbe:
            httpGet:
              path: /
              port: 3000
            initialDelaySeconds: 5
            periodSeconds: 10
```

- **Service.yaml:**

```
apiVersion: v1
kind: Service
metadata:
  name: node-prom-svc
spec:
  type: NodePort
  selector:
    app: node-prom
  ports:
    - port: 3000
      targetPort: 3000
      nodePort: 30081
```

7. Grafana Dashboard Setup

- Access Grafana: `http://<EC2-IP>:30877`
- Login default: `admin/admin` (then reset password)
- ⚫Retrieve autogenerated Grafana admin password (if needed):

```
kubectl get secret --namespace monitoring prometheus-grafana -o
jsonpath="{.data.admin-password}" | base64 -d
```

- Add Prometheus data source:
- URL: `http://prometheus-server.monitoring.svc.cluster.local`
- Create new dashboard → Panel → Query: `up`

Errors & Troubleshooting

❌Node.js app not showing in Prometheus UI

Fix:

- Ensure proper annotations are set in Deployment.yaml:

```
annotations:
  prometheus.io/scrape: "true"
  prometheus.io/port: "3000"
```

- Validate the service and pod are labeled `app: node-prom`

❌NodePort mismatch

Fix:

- Verified with `kubectl get svc -n monitoring`
- Ensure correct `nodePort` value assigned in `Service.yaml`

❌Service not found error

Fix:

- Ran:

```
kubectl apply -f service.yaml -n monitoring
```

to make sure service is created in right namespace.

❌Prometheus label match parse error

```
Error: parse error: unexpected identifier "node" in label matching
```

Fix:

- Incorrect query syntax.
- Correct usage:

```
up{job="node-prom"}
```

or simply start with `up` to validate target status.

❌No data in Grafana panel

Fix:

- Wait few seconds after panel creation
- Ensure Prometheus is added as Data Source
- Cross-check Prometheus targets at: `http://<EC2-IP>:32683/targets`

❌NodePort changed by Helm default

Fix:

- Modified Prometheus & Grafana service YAML to explicitly specify `nodePort`
- Confirmed NodePort using:

```
kubectl get svc -n monitoring
```

🔯Port-Forwarding Alternative

Usage:

```
kubectl port-forward svc/grafana -n monitoring 3000:80
kubectl port-forward svc/prometheus-server -n monitoring 9090:80
```

Then access locally via:

- `http://localhost:3000` (Grafana)
- `http://localhost:9090` (Prometheus)

Result

- Node.js app is successfully deployed and monitored.
- Metrics are being scraped by Prometheus.
- Grafana panel displays metric graphs.

Next Steps (Planned)

- Automate this setup using Helm and Terraform.
- Create custom alerts in Prometheus.
- Set up dashboards with multiple metrics.
- Integrate Slack alert channel.

Author: pradeepaanandh\ **Date:** 2025-07-16