Lab Manual: Deploying Prometheus and Grafana Monitoring for NGINX on Kubernetes (without Helm)

# 1. Prerequisites

* - 3-node Kubernetes cluster with kubectl access
* - You can run as admin (cluster-admin)
* - Internet access for your cluster nodes

# 2. Create Monitoring Namespace

kubectl create namespace monitoring

# 3. Deploy NGINX

nginx-deployment.yaml

apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: nginx-deployment  
 namespace: monitoring  
spec:  
 replicas: 2  
 selector:  
 matchLabels:  
 app: nginx  
 template:  
 metadata:  
 labels:  
 app: nginx  
 spec:  
 containers:  
 - name: nginx  
 image: nginx:latest  
 ports:  
 - containerPort: 80  
---  
apiVersion: v1  
kind: Service  
metadata:  
 name: nginx-service  
 namespace: monitoring  
spec:  
 selector:  
 app: nginx  
 ports:  
 - port: 80  
 targetPort: 80  
 type: NodePort

Apply NGINX deployment:

kubectl apply -f nginx-deployment.yaml

kubectl get pods -n monitoring -l app=nginx

kubectl get svc -n monitoring nginx-service

# 4. Deploy Prometheus

A. ConfigMap for Prometheus

prometheus-config.yaml

apiVersion: v1  
kind: ConfigMap  
metadata:  
 name: prometheus-config  
 namespace: monitoring  
data:  
 prometheus.yml: |  
 global:  
 scrape\_interval: 15s  
 scrape\_configs:  
 - job\_name: 'nginx'  
 kubernetes\_sd\_configs:  
 - role: pod  
 relabel\_configs:  
 - source\_labels: [\_\_meta\_kubernetes\_pod\_label\_app]  
 action: keep  
 regex: nginx  
 - job\_name: 'kubernetes-nodes'  
 kubernetes\_sd\_configs:  
 - role: node  
 - job\_name: 'kubelet'  
 kubernetes\_sd\_configs:  
 - role: node  
 metrics\_path: /metrics  
 scheme: http

B. Prometheus Deployment

prometheus-deployment.yaml

apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: prometheus-server  
 namespace: monitoring  
spec:  
 replicas: 1  
 selector:  
 matchLabels:  
 app: prometheus  
 template:  
 metadata:  
 labels:  
 app: prometheus  
 spec:  
 containers:  
 - name: prometheus  
 image: prom/prometheus:v2.52.0  
 args:  
 - "--config.file=/etc/prometheus/prometheus.yml"  
 - "--storage.tsdb.path=/prometheus/"  
 ports:  
 - containerPort: 9090  
 volumeMounts:  
 - name: config-volume  
 mountPath: /etc/prometheus/  
 volumes:  
 - name: config-volume  
 configMap:  
 name: prometheus-config

C. Prometheus Service

prometheus-service.yaml

apiVersion: v1  
kind: Service  
metadata:  
 name: prometheus-service  
 namespace: monitoring  
spec:  
 type: NodePort  
 ports:  
 - port: 9090  
 targetPort: 9090  
 nodePort: 30090  
 selector:  
 app: prometheus

Deploy Prometheus:

kubectl apply -f prometheus-config.yaml

kubectl apply -f prometheus-deployment.yaml

kubectl apply -f prometheus-service.yaml

kubectl get pods -n monitoring -l app=prometheus

kubectl get svc -n monitoring prometheus-service

# 5. Deploy Grafana

A. Grafana Deployment

grafana-deployment.yaml

apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: grafana  
 namespace: monitoring  
spec:  
 replicas: 1  
 selector:  
 matchLabels:  
 app: grafana  
 template:  
 metadata:  
 labels:  
 app: grafana  
 spec:  
 containers:  
 - name: grafana  
 image: grafana/grafana:11.0.0  
 ports:  
 - containerPort: 3000

B. Grafana Service

grafana-service.yaml

apiVersion: v1  
kind: Service  
metadata:  
 name: grafana-service  
 namespace: monitoring  
spec:  
 type: NodePort  
 ports:  
 - port: 3000  
 targetPort: 3000  
 nodePort: 30300  
 selector:  
 app: grafana

Deploy Grafana:

kubectl apply -f grafana-deployment.yaml

kubectl apply -f grafana-service.yaml

kubectl get pods -n monitoring -l app=grafana

kubectl get svc -n monitoring grafana-service

# 6. Access Prometheus and Grafana UIs

Prometheus: Open http://<NodeIP>:30090

Grafana: Open http://<NodeIP>:30300

Default login: admin / admin (change password on first login)

# 7. Configure Grafana to Use Prometheus

Open Grafana UI (http://<NodeIP>:30300)

Go to Settings > Data Sources > Add data source

Choose Prometheus

Set URL to:

http://prometheus-service.monitoring.svc.cluster.local:9090

Click Save & Test

# 8. Import Dashboards in Grafana

In Grafana, go to + > Import

Use dashboard ID 315 for Kubernetes cluster monitoring, or search for NGINX dashboards

# 9. (Optional) Add Alerts with Prometheus

example-alert-rules.yaml

apiVersion: monitoring.coreos.com/v1  
kind: PrometheusRule  
metadata:  
 name: custom-alerts  
 namespace: monitoring  
spec:  
 groups:  
 - name: custom.rules  
 rules:  
 - alert: NginxPodDown  
 expr: kube\_deployment\_status\_replicas\_unavailable{deployment="nginx-deployment"} > 0  
 for: 2m  
 labels:  
 severity: warning  
 annotations:  
 summary: "Some NGINX pods are unavailable"  
 description: "Nginx deployment pods are down for more than 2 minutes."

Apply alert rules:

kubectl apply -f example-alert-rules.yaml

(You need Prometheus Operator for CRDs like PrometheusRule, or use simple alerting configs in the configmap for plain Prometheus.)

# 10. (Optional) Clean Up

kubectl delete -f nginx-deployment.yaml

kubectl delete -f prometheus-config.yaml

kubectl delete -f prometheus-deployment.yaml

kubectl delete -f prometheus-service.yaml

kubectl delete -f grafana-deployment.yaml

kubectl delete -f grafana-service.yaml

kubectl delete namespace monitoring

## Summary Table: Key Commands

|  |  |
| --- | --- |
| Task | Command |
| Create monitoring namespace | kubectl create namespace monitoring |
| Deploy NGINX app | kubectl apply -f nginx-deployment.yaml |
| Deploy Prometheus config/deployment | kubectl apply -f prometheus-\*.yaml |
| Deploy Grafana deployment/service | kubectl apply -f grafana-\*.yaml |
| Access Prometheus UI | http://<NodeIP>:30090 |
| Access Grafana UI | http://<NodeIP>:30300 |
| Configure Grafana data source | (via Grafana UI) |
| Import dashboards | (via Grafana UI) |
| Clean up lab | kubectl delete -f ... && kubectl delete namespace monitoring |

End of Document