1. **Setup minikube.**
2. **Create namespace.**

```
kubectl create namespace monitoring
```

3. **Deploy Prometheus on minikube.**

```
apiVersion: v1
kind: Namespace
metadata:
  name: monitoring

---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: prometheus
  namespace: monitoring

---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: prometheus
rules:
- apiGroups: [""]
  resources:
  - nodes
  - nodes/metrics
  - services
  - endpoints
  - pods
  verbs: ["get", "list", "watch"]
- apiGroups: [""]
  resources:
  - configmaps
  verbs: ["get"]
- nonResourceURLs: ["/metrics"]
  verbs: ["get"]

---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: prometheus
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: prometheus
subjects:
- kind: ServiceAccount
  name: prometheus
  namespace: monitoring

---
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-server-conf
  namespace: monitoring
data:
  prometheus.yml: |-
    global:
      scrape_interval: 15s

    scrape_configs:
      - job_name: 'prometheus'
        static_configs:
          - targets: ['localhost:9090']

      - job_name: 'kube-state-metrics'
        static_configs:
          - targets: ['kube-state-metrics:8080']

      - job_name: 'node-exporter'
        static_configs:
          - targets: ['192.168.59.100:30000']

      - job_name: 'alertmanager'
        static_configs:
          - targets: ['alertmanager:31000']

    rule_files:
      - prometheus.rules

---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: prometheus-server
  namespace: monitoring
spec:
  replicas: 1
  selector:
    matchLabels:
      app: prometheus-server
  template:
    metadata:
      labels:
        app: prometheus-server
    spec:
      serviceAccountName: prometheus
      containers:
      - name: prometheus
        image: prom/prometheus:v2.30.0
        args:
          - "--config.file=/etc/prometheus/prometheus.yml"
          - "--storage.tsdb.path=/prometheus/"
        ports:
```

```
        - containerPort: 9090
        volumeMounts:
        - name: config-volume
          mountPath: /etc/prometheus
        - name: prometheus-data
          mountPath: /prometheus
      volumes:
      - name: config-volume
        configMap:
          name: prometheus-server-conf
      - name: prometheus-data
        emptyDir: {}

---
apiVersion: v1
kind: Service
metadata:
  name: prometheus
  namespace: monitoring
spec:
  type: NodePort
  selector:
    app: prometheus-server
  ports:
    - port: 9090
      targetPort: 9090
      nodePort: 30900 # Choose a port number within the valid range (30000-32767)
```

4. **Deploy Grafana on minikube.**

```
apiVersion: v1
kind: Namespace
metadata:
  name: monitoring

---
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:latest
        ports:
        - containerPort: 3000
        env:
        - name: GF_SECURITY_ADMIN_PASSWORD
          value: "yourpassword"
        volumeMounts:
        - name: grafana-storage
          mountPath: /var/lib/grafana
      volumes:
      - name: grafana-storage
        emptyDir: {}

---
apiVersion: v1
kind: Service
metadata:
  name: grafana
  namespace: monitoring
spec:
  type: NodePort
  selector:
    app: grafana
  ports:
    - port: 80
      targetPort: 3000
      nodePort: 30901 # Choose a port number within the valid range (30000-32767)
```

5. **Apply the deployment-service files.**

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

kubectl apply -f grafana.yaml -n monitoring

kubectl get services -n monitoring
```

6. **In case of an error, use the following commands:**

```
kubectl get services --all-namespaces -o wide

kubectl describe service prometheus -n monitoring

kubectl describe service grafana -n monitoring

kubectl logs <prometheus-pod-name> -n monitoring

kubectl logs <grafana-pod-name> -n monitoring
```

7. The output from **kubectl describe service prometheus -n monitoring** indicated that the service is of type **NodePort** and is listening on port **30900/TCP**. However, the **<nodePort>** field is still **<unset>** which indicated that Kubernetes did not assign a specific node port automatically. To fix this, we need to explicitly define the **nodePort** in the service section.

8. By verifying the **data source configuration in Grafana**, you can confirm that the data you're viewing is indeed coming from your Minikube cluster's Prometheus instance. If you're not sure which data source is being used, you can also check the queries or dashboards to see if they reference specific data sources.

9. **Getting minikube metrics:**

```
# Clone kube-state-metrics repository
git clone https://github.com/kubernetes/kube-state-metrics.git
cd kube-state-metrics/deploy/kubernetes/

# Apply manifests
kubectl apply -f .

# Verify deployment
kubectl get pods -n kube-system
```

*This will deploy **kube-state-metrics** in the **kube-system** namespace by default.*

```
sed -i 's/namespace: kube-system/namespace: monitoring/g' kube-state-
metrics/examples/standard/*
kubectl apply -f kube-state-metrics/examples/standard
kubectl get pods -n monitoring
```

10. **Create node-exporter deployment and service files.**

Node-exporter-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: node-exporter
  namespace: monitoring
spec:
  replicas: 1
  selector:
    matchLabels:
      app: node-exporter
  template:
    metadata:
      labels:
        app: node-exporter
    spec:
      containers:
      - name: node-exporter
        image: prom/node-exporter:v1.2.2
        imagePullPolicy: Always   # Set image pull policy to Always
        ports:
        - containerPort: 9100
```

Node-exporter-service.yaml

```
apiVersion: v1
```

```
kind: Service
metadata:
  name: node-exporter
  namespace: monitoring
spec:
  selector:
    app: node-exporter
  ports:
  - protocol: TCP
    port: 9100
    targetPort: 9100
    nodePort: 30000  # Specify the desired NodePort value
  type: NodePort  # Set the service type to NodePort
```

11. node-exporter NodePort 10.111.230.234 <none> 9100:30000/TCP 2m19s

    Node-exporter is running on 30000 but not 9100.

    This indicates that the **NodePort mapping** was successful, but it's not mapping to the expected port.

    To resolve this issue, updated your Prometheus configuration to scrape metrics from port 30000 instead of 9100.

12. **Alerting:**

    Create alert.rule.yml file

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-alert
  namespace: monitoring
data:
  alert.rule.yaml: |
    groups:
    - name: kubernetes-pods
      rules:
      - alert: PodDown
        expr: absent(up{job="kubelet"} == 1)
        for: 1m
        labels:
          severity: critical
        annotations:
          summary: "Pod {{ $labels.pod }} is down"
          description: "The pod {{ $labels.pod }} is not reporting any data, indicating it may be
down or unreachable."

      - alert: ContainerDown
        expr:
absent(container_memory_usage_bytes{container_name!="POD",container_name!=""})
        for: 1m
        labels:
          severity: warning
        annotations:
          summary: "Container {{ $labels.container_name }} is down in Pod {{ $labels.pod }}"
          description: "The container {{ $labels.container_name }} in pod {{ $labels.pod }} is not
      reporting any memory usage, indicating it may be down or unreachable."
```

Include rules_file in Prometheus ConfigMap

Include the rules in volumes and volumeMounts in Prometheus Deployment

```
kubectl apply -f alert.rule.yaml -n monitoring
kubectl delete deployment prometheus-server -n monitoring
kubectl apply -f prometheus.yaml -n monitoring
```