Opensense Monitoring Through Grafana and Prometheus

🟩 Step 1 — Create VNet Peering Between Both VNets

You must peer:

- **VNet A** → where Grafana + Prometheus pods run
- **VNet B** → where OPNsense lives

✓ In Azure Portal:

VNet A → Peerings → Add:

- Remote VNet: **VNet B**
- ✓ Allow **Virtual network access**
- ✓ Allow **Forwarded traffic**
- ❌ Gateway transit → leave disabled (unless using VPN Gateway)

VNet B → Peerings → Add:

- Remote VNet: **VNet A**
- ✓ Allow **Virtual network access**
- ✓ Allow **Forwarded traffic**
- ❌ Gateway transit → disabled

This enables **routing** between Prometheus Pod and OPNsense.

🟩 Step 2 — Update NSG Rules (If using Azure NSG)

Prometheus Pod must reach OPNsense's Prometheus exporter port.

Assume exporter port = **9100**.

On OPNsense VNet/subnet NSG:

Add:

Setting	Value
Source	**VNet A address range (CIDR)**
Destination	**OPNsense private IP**
Protocol	TCP
Port	**9100**
Action	Allow

Example:

Source: 10.10.0.0/16 (Grafana/Prom VNet)

Destination: 10.20.1.5 (OPNsense)

Port: 9100

Action: Allow

🟩 Step 3 — Configure OPNsense Firewall Rules

OPNsense must allow traffic from Prometheus VNet.

Go to:

Firewall → Rules → LAN (or interface OPNsense is on)

Add allow rule:

Source: <Grafana/Prometheus VNet CIDR>

Destination: This firewall

Port: 9100

🟩 Step 4 — Prometheus Pod Configuration

ConfigMap (prometheus.yml with OPNsense target)

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

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

    - job_name: 'opnsense'
      metrics_path: /metrics
      static_configs:
        - targets:
['<OPNSENSE_PRIVATE_IP:9100']
```

✅ **Service (internal access only)**

```
apiVersion: v1
kind: Service
metadata:
  name: opensense-prometheus
  namespace: monitoring
spec:
  type: ClusterIP
  selector:
    app: opensense-prometheus
  ports:
    - port: 9090
      targetPort: 9090
      protocol: TCP
```

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

🔥 **After Deploying These Files**

Apply everything:

kubectl apply -f configmap.yaml

kubectl apply -f deployment.yaml

kubectl apply -f service.yaml

Then verify Prometheus is running:

kubectl get pods -n monitoring

Test reachability:

kubectl exec -it <prometheus-pod> -n monitoring -- curl http://<OPNSENSE_PRIVATE_IP>:9100/metrics

If peering + NSG + OPNsense firewall are correct → you will see metrics.

🟩 **Step 5 — Test Connectivity From the Pod**

Open a shell inside Prometheus pod:

kubectl exec -it <prometheus-pod> -- sh

Test reachability:

curl http://10.20.1.5:9100/metrics

If VNet peering + NSG + OPNsense firewall are correct, this will show metrics text.

🟩 **Step 6 — Grafana**

Grafana is in **same VNet** → so no network issues.

Just add Prometheus as data source.