Option 1 - Setup Prometheus Monitoring On Kubernetes Cluster

1. **Cloning the repo that contains the configuration files needed for this lab.**

```
C:\Users\V&M\Desktop>git clone https://github.com/techiescamp/kubernetes-prometheus
Cloning into 'kubernetes-prometheus'...
remote: Enumerating objects: 249, done.
remote: Counting objects: 100% (163/163), done.
remote: Compressing objects: 100% (79/79), done.
Receiving objects: 100% (249/249), 63.43 KiB | 1.38 MiB/s, done.eused 86 (delta 84), pack-reused 86

Resolving deltas: 100% (141/141), done.
```

2. **Create a Namespace & ClusterRole**

```
C:\Users\V&M\Desktop>kubectl create namespace monitoring
namespace/monitoring created
```

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: prometheus
rules:
- apiGroups: [""]
  resources:
  - nodes
  - nodes/proxy
  - services
  - endpoints
  - pods
  verbs: ["get", "list", "watch"]
- apiGroups:
  - extensions
  resources:
  - ingresses
  verbs: ["get", "list", "watch"]
- 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: default
  namespace: monitoring
```

```
C:\Users\V&M\Desktop\kubernetes-prometheus>kubectl create -f clusterRole.yaml
clusterrole.rbac.authorization.k8s.io/prometheus created
clusterrolebinding.rbac.authorization.k8s.io/prometheus created
```

3. **Create a Config Map To Externalize Prometheus Configurations**

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-server-conf
  labels:
    name: prometheus-server-conf
  namespace: monitoring
data:
  prometheus.rules: |-
    groups:
    - name: devopscube demo alert
      rules:
      - alert: High Pod Memory
        expr: sum(container_memory_usage_bytes) > 1
        for: 1m
        labels:
          severity: slack
        annotations:
          summary: High Memory Usage
  prometheus.yml: |-
    global:
      scrape_interval: 5s
      evaluation_interval: 5s
    rule_files:
      - /etc/prometheus/prometheus.rules
    alerting:
      alertmanagers:
      - scheme: http
        static_configs:
        - targets:
          - "alertmanager.monitoring.svc:9093"
    scrape_configs:
```

```
C:\Users\V&M\Desktop\kubernetes-prometheus>kubectl create -f config-map.yaml
configmap/prometheus-server-conf created
```

4. **Create a Prometheus Deployment**

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: prometheus-deployment
  namespace: monitoring
  labels:
    app: prometheus-server
spec:
  replicas: 1
  selector:
    matchLabels:
      app: prometheus-server
  template:
    metadata:
      labels:
        app: prometheus-server
    spec:
      containers:
        - name: prometheus
          image: prom/prometheus
          args:
            - "--config.file=/etc/prometheus/prometheus.yml"
            - "--storage.tsdb.path=/prometheus/"
          ports:
            - containerPort: 9090
          volumeMounts:
            - name: prometheus-config-volume
              mountPath: /etc/prometheus/
            - name: prometheus-storage-volume
              mountPath: /prometheus/
      volumes:
        - name: prometheus-config-volume
          configMap:
            defaultMode: 420
            name: prometheus-server-conf

        - name: prometheus-storage-volume
          emptyDir: {}
```

```
C:\Users\V&M\Desktop\kubernetes-prometheus>kubectl create  -f prometheus-deployment.yaml
deployment.apps/prometheus-deployment created
C:\Users\V&M\Desktop\kubernetes-prometheus>kubectl get deployments --namespace=monitoring
NAME                    READY   UP-TO-DATE   AVAILABLE   AGE
prometheus-deployment   1/1     1            1           2m2s
```

5. **Connecting To Prometheus Dashboard**

Method 1: Using Kubectl port forwarding

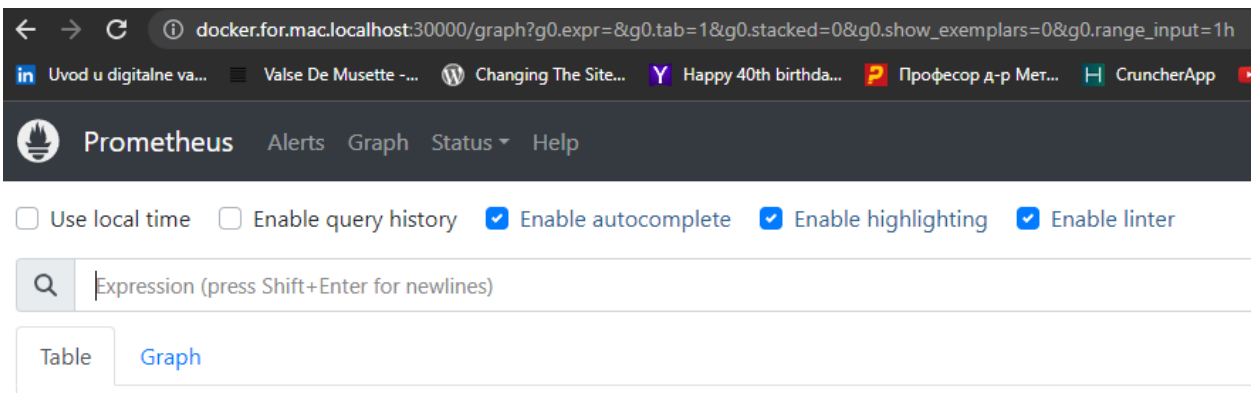
```
C:\Users\V&M\Desktop\kubernetes-prometheus>kubectl port-forward
prometheus-deployment-74dc6c7466-fjzb9 9090:9090 -n monitoring
Forwarding from 127.0.0.1:9090 -> 9090
Forwarding from [::1]:9090 -> 9090
Handling connection for 9090
Handling connection for 9090
Handling connection for 9090
```

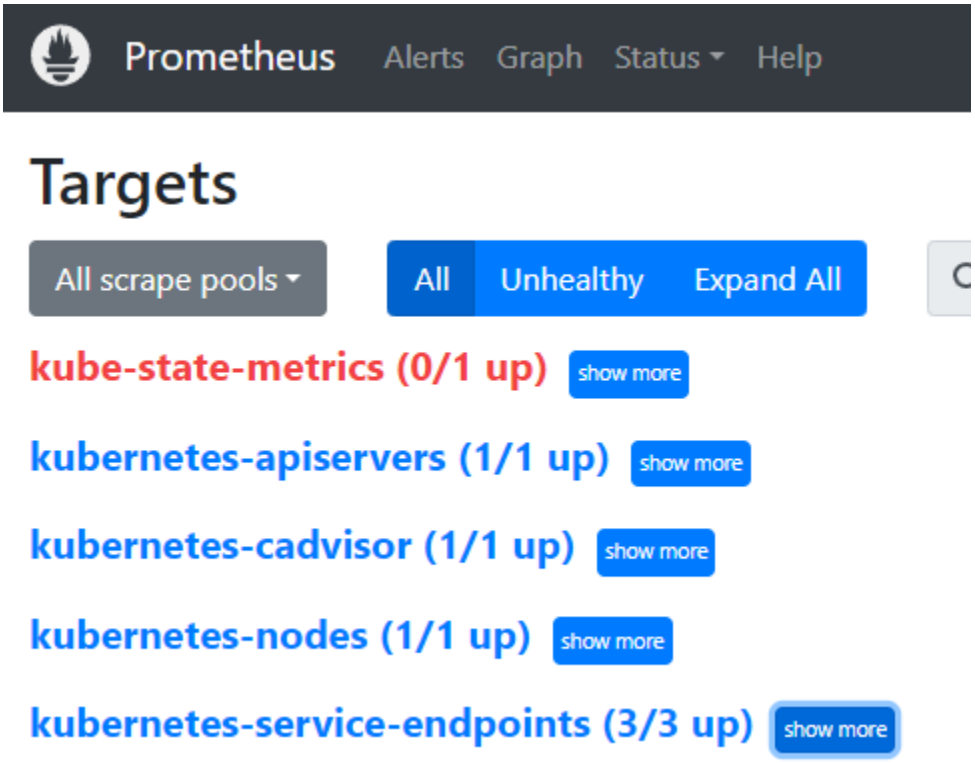
Method 2: Exposing Prometheus as a Service [NodePort & LoadBalancer]

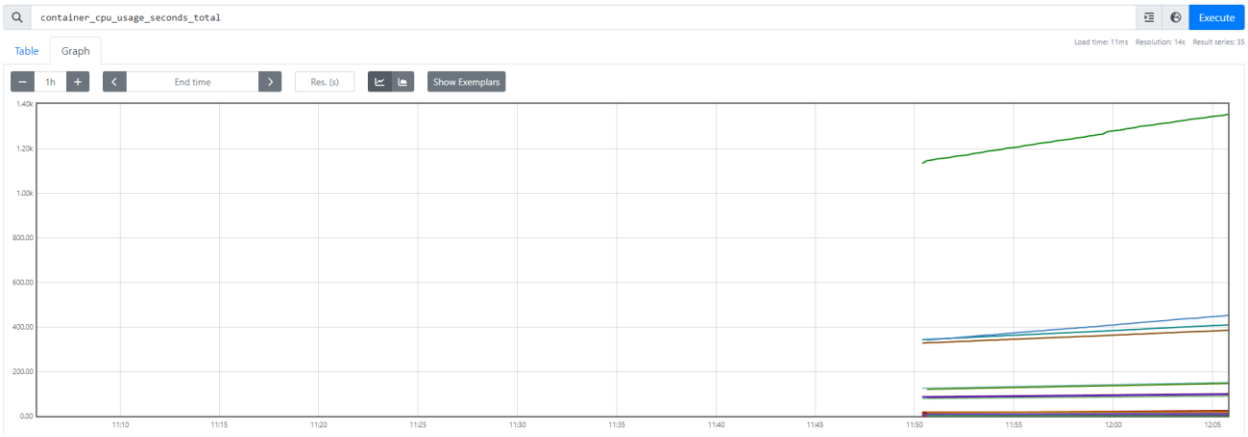
```yaml
apiVersion: v1
kind: Service
metadata:
  name: prometheus-service
  namespace: monitoring
  annotations:
        prometheus.io/scrape: 'true'
        prometheus.io/port:   '9090'

spec:
  selector:
    app: prometheus-server
  type: NodePort
  ports:
    - port: 8080
      targetPort: 9090
      nodePort: 30000
```

```
C:\Users\V&M\Desktop\kubernetes-prometheus>kubectl create -f prometheus-service.yaml --namespace=monitoring
service/prometheus-service created
```

- **Accessing the Prometheus dashboard using any the Kubernetes node's address on port 30000**




Prometheus
Alerts
Graph
Status
Help
Targets
All scrape pools
All
Unhealthy
Expand All
kube-state-metrics (0/1 up)
show more
kubernetes-apiservers (1/1 up)
show more
kubernetes-cadvisor (1/1 up)
show more
kubernetes-nodes (1/1 up)
show more
kubernetes-service-endpoints (3/3 up)
show more


container_cpu_usage_seconds_total
Execute
Table
Graph
1h
End time
Res. (s)
Show Exemplars

Method 3: Exposing Prometheus Using Ingress

```yaml
## Nginx Ingress
## Follow https://devopscube.com/setup-ingress-kubernetes-nginx-controller/

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: prometheus-ui
  namespace: monitoring
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  rules:
  # Use the host you used in your kubernetes Ingress Configurations
  - host: prometheus.example.com
    http:
      paths:
      - backend:
          service:
            name: prometheus-service
            port:
              number: 8080
        path: /
        pathType: Prefix
  tls:
  - hosts:
    - prometheus.apps.shaker242.lab
    secretName: prometheus-secret
---
apiVersion: v1
kind: Secret
metadata:
  name: prometheus-secret
  namespace: monitoring
data:
# USe base64 in the certs
  tls.crt: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUZpVENDQkhHZ0F3SUJBZ0lCQVR
  tls.key: LS0tLS1CRUdJTiBQUklWQVRFIEtFWS0tLS0tCk1JSUV2d0lCQURBTkJna3Foa2lHOXc
```

6. **Kube State Metrics Setup**

```
C:\Users\V&M\Desktop\kubernetes-prometheus>kubectl apply -f kube-state-metrics-configs/
clusterrolebinding.rbac.authorization.k8s.io/kube-state-metrics created
clusterrole.rbac.authorization.k8s.io/kube-state-metrics created
deployment.apps/kube-state-metrics created
serviceaccount/kube-state-metrics created
service/kube-state-metrics created
```

7. **Setting Up Alertmanager**

```yaml
kind: ConfigMap
apiVersion: v1
metadata:
  name: alertmanager-config
  namespace: monitoring
data:
  config.yml: |-
    global:
    templates:
    - '/etc/alertmanager/*.tmpl'
    route:
      receiver: alert-emailer
      group_by: ['alertname', 'priority']
      group_wait: 10s
      repeat_interval: 30m
      routes:
        - receiver: slack_demo
        # Send severity=slack alerts to slack.
          match:
            severity: slack
          group_wait: 10s
          repeat_interval: 1m

    receivers:
    - name: alert-emailer
      email_configs:
      - to: demo@devopscube.com
        send_resolved: false
        from: from-email@email.com
        smarthost: smtp.eample.com:25
        require_tls: false
    - name: slack_demo
      slack_configs:
      - api_url: https://hooks.slack.com/services/T
        channel: '#devopscube-demo'
```

```
C:\Users\V&M\Desktop\kubernetes-prometheus>kubectl create -f AlertManagerConfigmap.yaml
configmap/alertmanager-config created
```

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  creationTimestamp: null
  name: alertmanager-templates
  namespace: monitoring
data:
  default.tmpl: |
    {{ define "__alertmanager" }}A
    {{ define "__alertmanagerURL"
    {{ define "__subject" }}[{{ .S
    {{ define "__description" }}{{
    {{ define "__text_alert_list"
```

```
C:\Users\V&M\Desktop\kubernetes-prometheus>kubectl create -f AlertTemplateConfigMap.yaml
configmap/alertmanager-templates created
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: alertmanager
  namespace: monitoring
spec:
  replicas: 1
  selector:
    matchLabels:
      app: alertmanager
  template:
    metadata:
      name: alertmanager
      labels:
        app: alertmanager
    spec:
      containers:
      - name: alertmanager
        image: prom/alertmanager:latest
        args:
          - "--config.file=/etc/alertmanager/config.yml"
          - "--storage.path=/alertmanager"
        ports:
        - name: alertmanager
          containerPort: 9093
```

```
C:\Users\V&M\Desktop\kubernetes-prometheus>kubectl create -f Deployment.yaml
deployment.apps/alertmanager created
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: alertmanager
  namespace: monitoring
  annotations:
      prometheus.io/scrape: 'true'
      prometheus.io/port:   '9093'
spec:
  selector:
    app: alertmanager
  type: NodePort
  ports:
    - port: 9093
      targetPort: 9093
      nodePort: 31000
```

```
C:\Users\V&M\Desktop\kubernetes-prometheus>kubectl create -f Service.yaml
service/alertmanager created
```

8. **Setting up Grafana**

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: grafana-datasources
  namespace: monitoring
data:
  prometheus.yaml: |-
    {
        "apiVersion": 1,
        "datasources": [
            {
               "access":"proxy",
                "editable": true,
                "name": "prometheus",
                "orgId": 1,
                "type": "prometheus",
                "url": "http://prometheus-service.monitoring.svc:8080",
                "version": 1
            }
        ]
    }
```

```
C:\Users\V&M\Desktop\kubernetes-prometheus>kubectl create -f grafana-datasource-config.yaml
configmap/grafana-datasources created
```

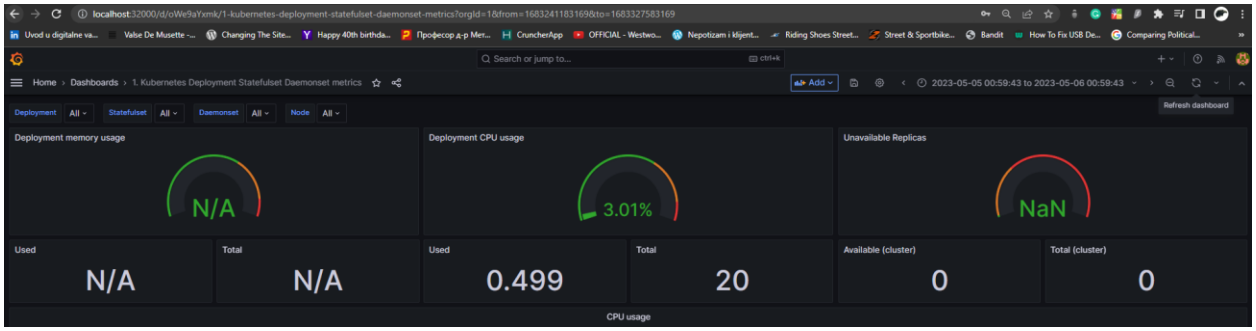
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: grafana
  namespace: monitoring
spec:
  replicas: 1
  selector:
    matchLabels:
      app: grafana
  template:
    metadata:
      name: grafana
      labels:
        app: grafana
    spec:
      containers:
      - name: grafana
        image: grafana/grafana:latest
        ports:
        - name: grafana
          containerPort: 3000
        resources:
          limits:
            memory: "1Gi"
            cpu: "1000m"
          requests:
            memory: 500M
            cpu: "500m"
        volumeMounts:
          - mountPath: /var/lib/grafana
            name: grafana-storage
          - mountPath: /etc/grafana/provisioning/datasources
            name: grafana-datasources
            readOnly: false
      volumes:
        - name: grafana-storage
          emptyDir: {}
        - name: grafana-datasources
          configMap:
              defaultMode: 420
              name: grafana-datasources
```

```
C:\Users\V&M\Desktop\kubernetes-prometheus>kubectl create -f deployment-grafana.yaml
deployment.apps/grafana created
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: grafana
  namespace: monitoring
  annotations:
      prometheus.io/scrape: 'true'
      prometheus.io/port:   '3000'
spec:
  selector:
    app: grafana
  type: NodePort
  ports:
    - port: 3000
      targetPort: 3000
      nodePort: 32000
```

```
C:\Users\V&M\Desktop\kubernetes-prometheus>kubectl create -f service-grafana.yaml
service/grafana created
```

```
C:\Users\V&M\Desktop\kubernetes-prometheus>kubectl port-forward -n monitoring grafana-5469c64c7d-k8pf9 3000 &
Forwarding from 127.0.0.1:3000 -> 3000
Forwarding from [::1]:3000 -> 3000
```



Option 2 - Monitoring Windows OS with Prometheus

1. After I installed the Prometheus and Windows Exporter, I configured the Prometheus configuration file /opt/prometheus/prometheus.yml like following:

```
vladimir@DESKTOP-O80TR5A:~$ cat /opt/prometheus/prometheus.yml
# my global config
global:
  scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets:
          # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.

scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"
  - job_name: 'win10' #THIS IS LINE 24
    static_configs:
    - targets: ['192.168.1.106:9182']
```

2. Than, I accessed Prometheus web interface using the computer's IP address where I have installed Prometheus, and tried some expressions for monitoring Windows computer.