



Deploying Prometheus And Grafana in Kubernetes in one CLICK...

Before explanation of the task, let's first see something about Prometheus and Grafana...

Prometheus: Prometheus is a free software application used for event monitoring and alerting. It records real-time metrics in a time series database built using an HTTP pull model, with flexible queries and real-time alerting.

Grafana: **Grafana** is an open-source platform for data visualization, monitoring, and analysis. Most of the time, we use it with Prometheus. **Grafana** focuses on presenting time-series charts based on specific metrics such as CPU and I/O utilization.

So let's see what we are going to do...

Integrate Prometheus and Grafana and perform in the following way:

1. Deploy them as pods on top of Kubernetes by creating resources Deployment, ReplicaSet, Pods, or Services.
2. Make their data remain persistent and both of them should be exposed to the outside world.

Github Link for reference: <https://github.com/Anuddeeph/prometheus-grafana-persistent.git>

Let's start now...

- The first thing which we need to set up Kubernetes locally. Check this as a reference for setting [Kubernetes with Minikube](#)...
- For launch the whole setup I created yaml file and one kustomization file. Let me explain all the file one by one...
- For download image, you can check my docker hub account... <https://hub.docker.com/u/anuddeeph>
- **I created one ConfigMap for attaching config file with prometheus server...**

```
• apiVersion: v1
• kind: ConfigMap
• metadata:
•   name: prom-config
•   labels:
•     app: prom
• data:
•   prometheus.yml: |
•     global:
•       scrape_interval: 15s
•       evaluation_interval: 15s
•       scrape_configs:
•         - job_name: 'prometheus'
•           static_configs:
•
•         - targets: ['localhost:9090']
```

A **ConfigMap** is an API object that lets you store configuration for other objects to use.

Unlike most **Kubernetes** objects that have a spec , a **ConfigMap** has a data section to store items (keys) and their values.

- **This is the code for creating one PVC resource to make Prometheus data persistent...**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
```

```

    name: task-prom-claim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:

      storage: 3Gi

```

- **This is for creating one service and deployment to launch and expose Prometheus...**

```

apiVersion: v1
kind: Service
metadata:
  name: prometheus
  labels:
    app: prom
spec:
  ports:
    - port: 9090
  selector:
    app: prom
  type: NodePort
---

apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: prom-deploy
  labels:
    app: prom
spec:
  replicas: 1
  selector:
    matchLabels:
      app: prom
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: prom
    spec:
      containers:
        - image: anuddeeph/Prometheus:v1
          name: prom
          ports:
            - containerPort: 9090
              name: prom
          volumeMounts:
            - name: prom-config
              mountPath: /prometheus-2.19.2.linux-amd64/prometheus.yml

```

```

    subPath: prometheus.yml
  - name: task-prom-claim
    mountPath: /prometheus-2.19.2.linux-amd64/data
volumes:
  - name: prom-config
    configMap:
      name: prom-config
  - name: task-prom-claim
    persistentVolumeClaim:
      claimName: task-prom-claim

```

- **This is the code for creating one service, PVC, and deployment resource for launch, expose and make Grafana persistent...**

```

apiVersion: v1
kind: Service
metadata:
  name: grafana
  labels:
    app: graf
spec:
  ports:
    - port: 3000
  selector:
    app: graf
  type: NodePort
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: task-graf-claim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 3Gi
---
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: graf-deploy
  labels:
    app: graf
spec:
  replicas: 1
  selector:
    matchLabels:
      app: graf
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: graf

```


```
spec:
  containers:
  - image: anuddeeph/Grafana:v1
    name: graf
    ports:
    - containerPort: 3000
      name: graf
    volumeMounts:
    - name: task-graf
      mountPath: "/usr/share/grafana/data"
  volumes:
  - name: task-graf
    persistentVolumeClaim:
      claimName: task-graf-claim
```

- **Now the last thing, we are going to create a kustomization file for launch the whole setup in one click...**

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
resources:
- configmap.yml
- promvolume.yml
- prometheus.yml

- grafana.yml
```

Let's check the output now...

 Command Prompt

```
C:\Users\Anuddeeph Nalla\Desktop\Devops\DevopsTask5>dir
Volume in drive C has no label.
Volume Serial Number is E023-1724

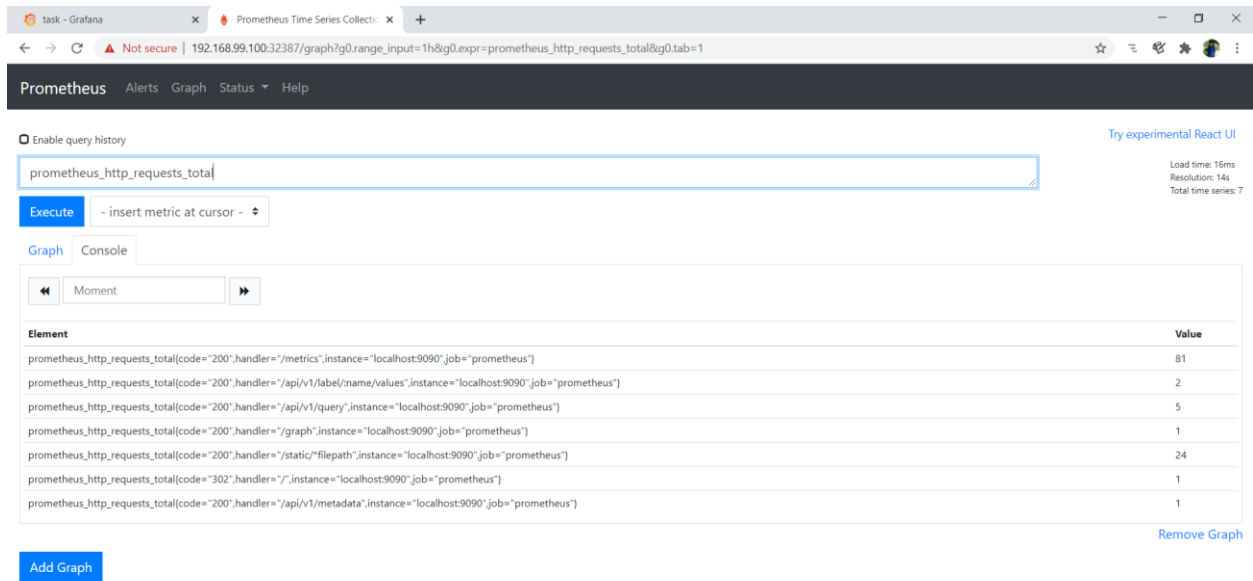
Directory of C:\Users\Anuddeeph Nalla\Desktop\Devops\DevopsTask5

21-07-2020  22:43    <DIR>          .
21-07-2020  22:43    <DIR>          ..
30-06-2020  07:33             312 configmap.yml
21-07-2020  22:27           1,026 grafana.yml
30-06-2020  07:33             152 kustomization.yml
21-07-2020  22:28           1,091 prometheus.yml
30-06-2020  07:33             164 promvolume.yml
               5 File(s)              2,745 bytes
               2 Dir(s)  22,857,048,064 bytes free

C:\Users\Anuddeeph Nalla\Desktop\Devops\DevopsTask5>kubectl apply -k .
configmap/prom-config created
service/grafana created
service/prometheus created
deployment.apps/graf-deploy created
deployment.apps/prom-deploy created
persistentvolumeclaim/task-graf-claim created
persistentvolumeclaim/task-prom-claim created
```

You can check here that all the resources are launch by single command...

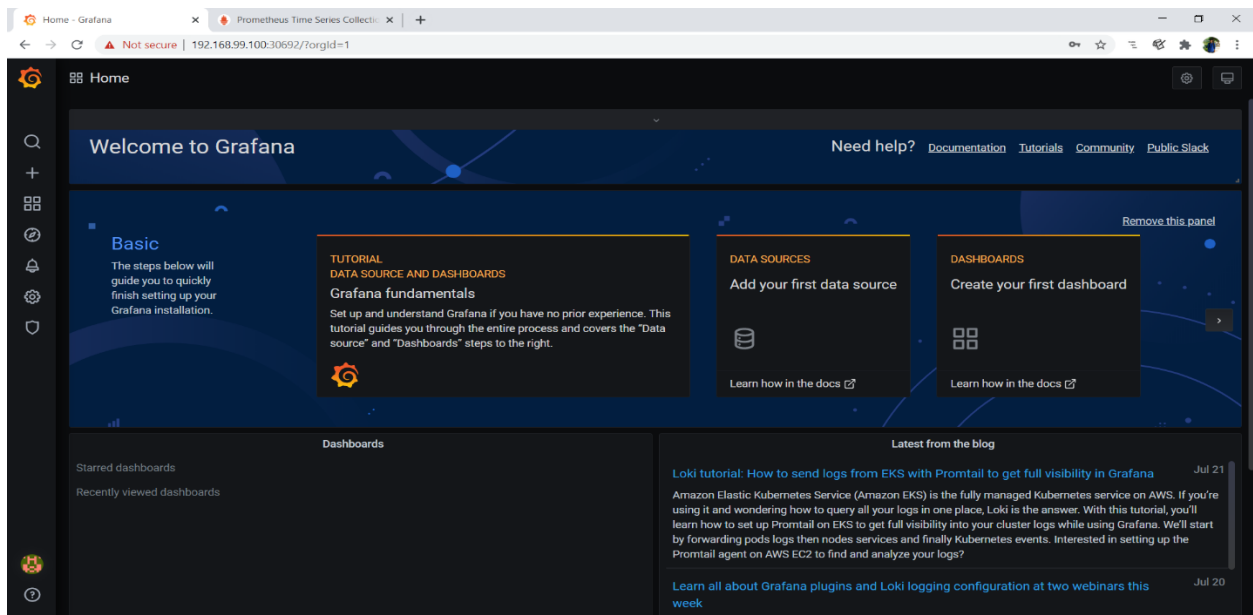
kubectl apply -k .



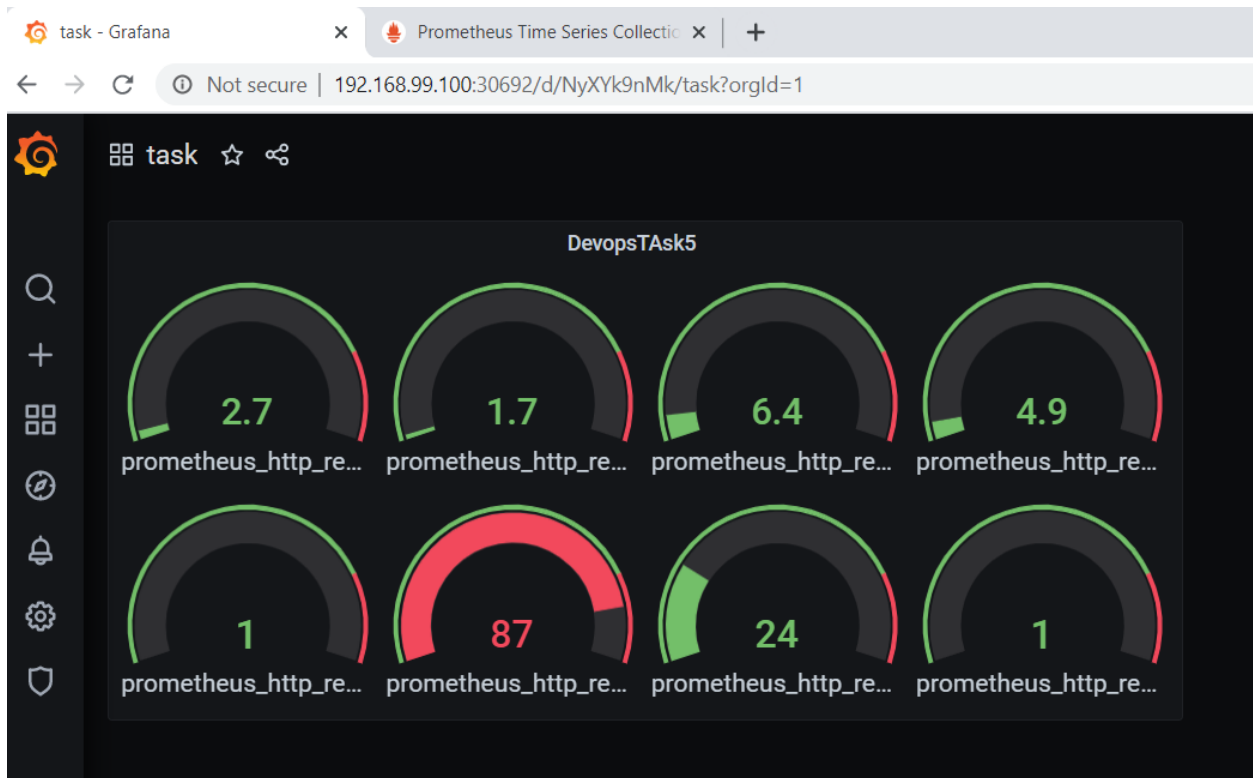
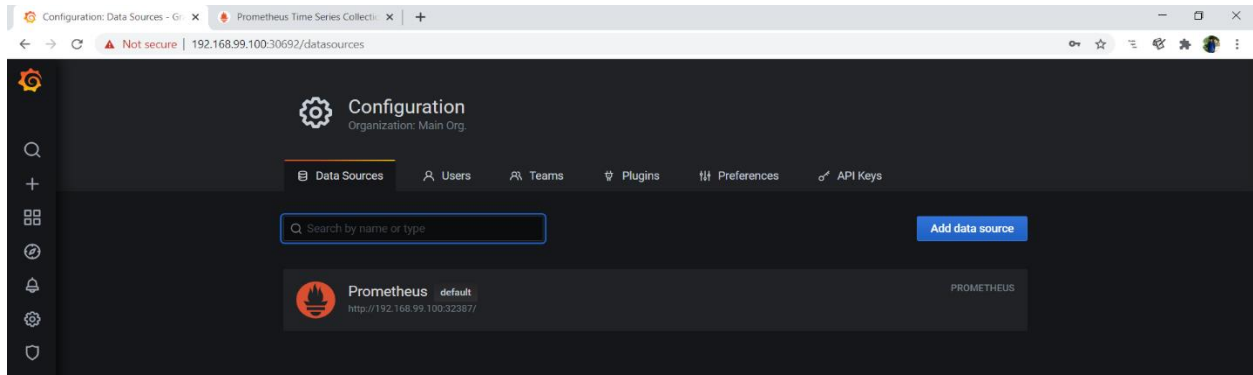
The screenshot shows the Prometheus web interface. At the top, there's a navigation bar with 'Prometheus', 'Alerts', 'Graph', 'Status', and 'Help'. Below this, there's a search bar with the query 'prometheus_http_requests_total' and an 'Execute' button. To the right of the search bar, it says 'Try experimental React UI' and shows 'Load time: 16ms', 'Resolution: 14s', and 'Total time series: 7'. Below the search bar, there's a 'Graph' tab and a 'Console' tab. The 'Graph' tab is selected, and it shows a table of results for the query. The table has two columns: 'Element' and 'Value'. The results are as follows:

Element	Value
prometheus_http_requests_total[code="200",handler="/metrics",instance="localhost:9090",job="prometheus"]	81
prometheus_http_requests_total[code="200",handler="/api/v1/label/name/values",instance="localhost:9090",job="prometheus"]	2
prometheus_http_requests_total[code="200",handler="/api/v1/query",instance="localhost:9090",job="prometheus"]	5
prometheus_http_requests_total[code="200",handler="/graph",instance="localhost:9090",job="prometheus"]	1
prometheus_http_requests_total[code="200",handler="/static/FilePath",instance="localhost:9090",job="prometheus"]	24
prometheus_http_requests_total[code="302",handler="/",instance="localhost:9090",job="prometheus"]	1
prometheus_http_requests_total[code="200",handler="/api/v1/metadata",instance="localhost:9090",job="prometheus"]	1

At the bottom of the table, there's a 'Remove Graph' button. Below the table, there's an 'Add Graph' button.



The screenshot shows the Grafana web interface. At the top, there's a navigation bar with 'Home', 'Need help?', 'Documentation', 'Tutorials', 'Community', and 'Public Slack'. Below this, there's a 'Welcome to Grafana' message. The main content area is divided into several sections. On the left, there's a 'Basic' section with the text 'The steps below will guide you to quickly finish setting up your Grafana installation.' In the center, there's a 'TUTORIAL DATA SOURCE AND DASHBOARDS Grafana fundamentals' section with the text 'Set up and understand Grafana if you have no prior experience. This tutorial guides you through the entire process and covers the "Data source" and "Dashboards" steps to the right.' To the right of the tutorial, there's a 'DATA SOURCES' section with the text 'Add your first data source' and a 'DASHBOARDS' section with the text 'Create your first dashboard'. Below these sections, there's a 'Dashboards' section with 'Starred dashboards' and 'Recently viewed dashboards'. On the right, there's a 'Latest from the blog' section with two articles: 'Loki tutorial: How to send logs from EKS with Promtail to get full visibility in Grafana' and 'Learn all about Grafana plugins and Loki logging configuration at two webinars this week'.



Now if you either delete any deployment or pod, your data remain persistent..

Command Prompt

```
C:\Users\Anuddeeph Nalla\Desktop\Devops\DevopsTask5>kubect1 delete pods --all
pod "graf-deploy-584d56674-9m8lr" deleted
pod "prom-deploy-5555b65c49-1wrbc" deleted
```

```
C:\Users\Anuddeeph Nalla\Desktop\Devops\DevopsTask5>kubect1 get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/graf-deploy-584d56674-xcrdr     1/1     Running   0           17s
pod/prom-deploy-5555b65c49-tm5c9    1/1     Running   0           17s
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/grafana	NodePort	10.104.239.245	<none>	3000:30692/TCP	48m
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	49m
service/prometheus	NodePort	10.102.140.39	<none>	9090:32387/TCP	48m

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/graf-deploy	1/1	1	1	48m
deployment.apps/prom-deploy	1/1	1	1	48m

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/graf-deploy-584d56674	1	1	1	48m
replicaset.apps/prom-deploy-5555b65c49	1	1	1	48m

task - Grafana x Prometheus Time Series Collect: x +

← → ↺ 🔒 Not secure | 192.168.99.100:32387/graph?g0.range_input=1h&g0.expr=prometheus_http_requests_total&g0.tab=1 ☆ ⚙ ⚙ ⚙ ⚙

Prometheus Alerts Graph Status ▾ Help

☐ Enable query history [Try experimental React UI](#)

prometheus_http_requests_total

Execute - insert metric at cursor - ▾

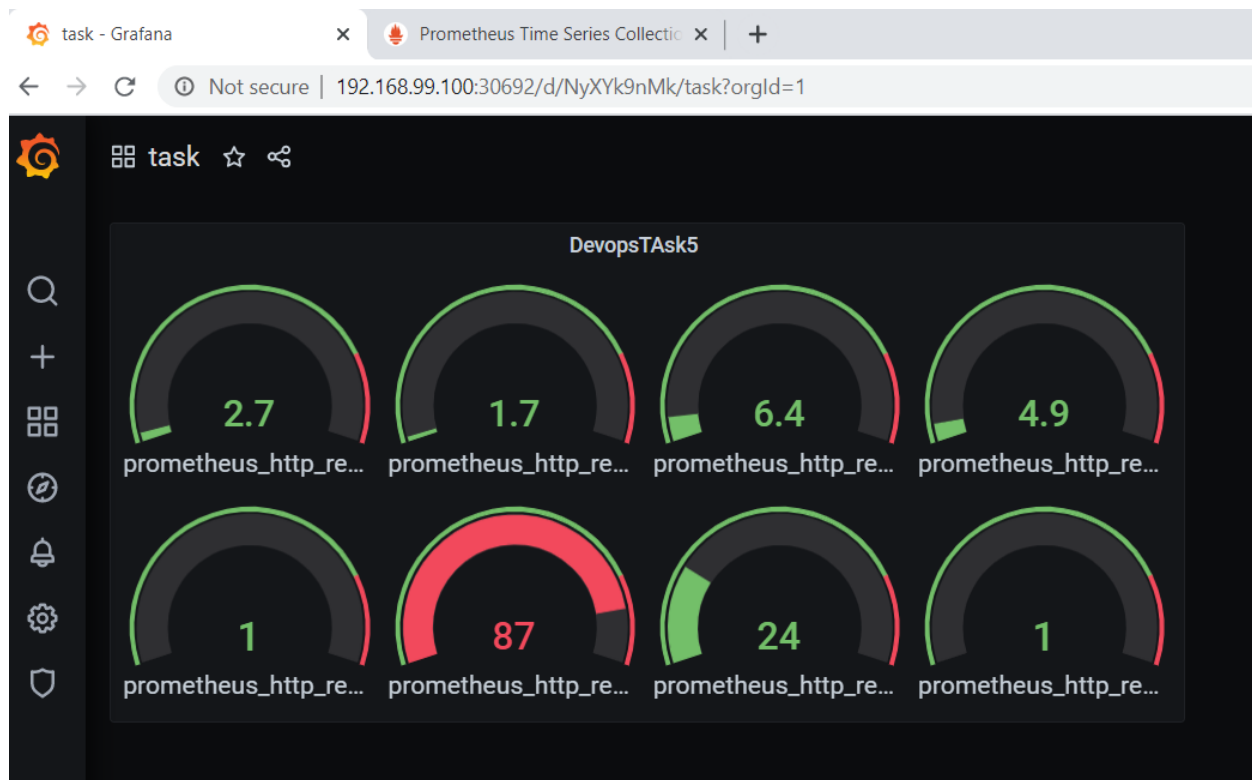
Graph Console

⏪ Moment ⏩

Element	Value
prometheus_http_requests_total{code="200",handler="/metrics",instance="localhost:9090",job="prometheus"}	81
prometheus_http_requests_total{code="200",handler="/api/v1/label/name/values",instance="localhost:9090",job="prometheus"}	2
prometheus_http_requests_total{code="200",handler="/api/v1/query",instance="localhost:9090",job="prometheus"}	5
prometheus_http_requests_total{code="200",handler="/graph",instance="localhost:9090",job="prometheus"}	1
prometheus_http_requests_total{code="200",handler="/static/filepath",instance="localhost:9090",job="prometheus"}	24
prometheus_http_requests_total{code="302",handler="/",instance="localhost:9090",job="prometheus"}	1
prometheus_http_requests_total{code="200",handler="/api/v1/metadata",instance="localhost:9090",job="prometheus"}	1

[Remove Graph](#)

Add Graph



Feel free to DM me if you have any query regarding this setup...

Thanks for reading... 😊