

Deploy Prometheus and Grafana on Kubernetes (Major Project)

Created by: Pradip Ramesh Malik

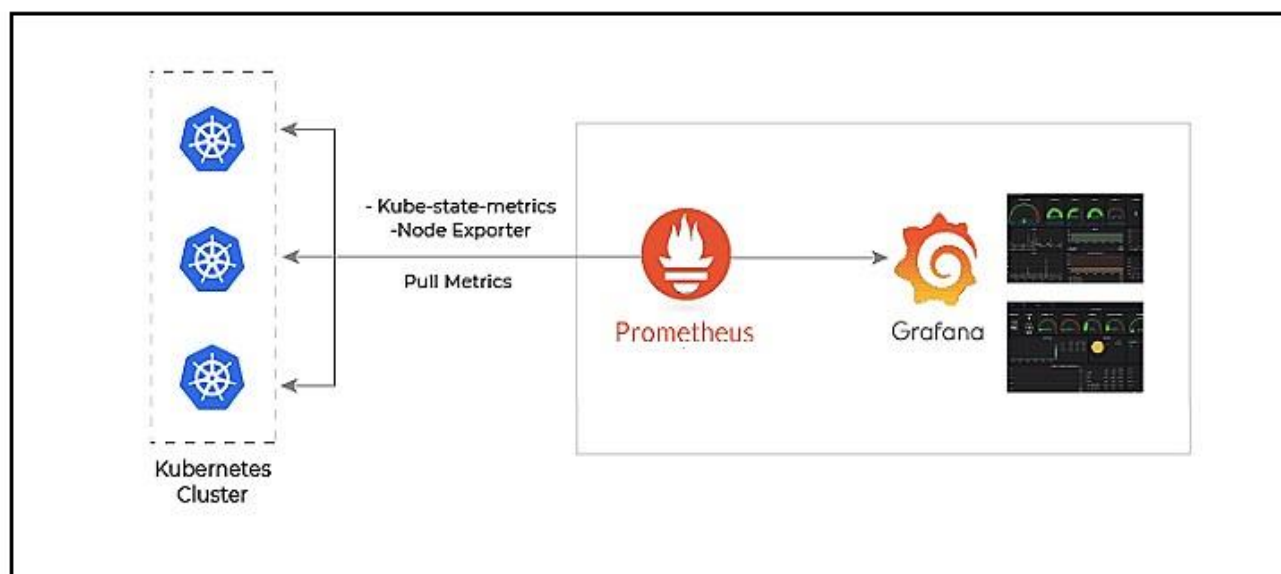
Date: 15/07/2024

Prometheus and Grafana

Prometheus is an open-source monitoring and alerting toolkit designed specifically for reliability and scalability. It collects metrics from configured targets at given intervals, evaluates rule expressions, displays results, and triggers alerts if certain conditions are met. In a Kubernetes environment, Prometheus is widely used to gather detailed metrics on cluster components, application performance, and resource usage.

Grafana is an open-source analytics and interactive visualization tool that integrates seamlessly with Prometheus. It allows users to create, explore, and share dashboards that can display real-time data from Prometheus. Grafana's powerful query capabilities and customizable visualizations make it ideal for monitoring Kubernetes clusters.

Architecture



1. **Prerequisites:** Ensure you have kubernetes (v1.30.0) installed and running.

```
aws Services Search [Alt+S] N. Virginia
root@ip-172-31-22-135:~# kubectl get nodes
NAME                STATUS    ROLES    AGE   VERSION
ip-172-31-20-46     Ready    <none>   10s   v1.30.2
ip-172-31-22-135    Ready    control-plane 15m   v1.30.2
root@ip-172-31-22-135:~#
```

i-0cf145e1fd5880c4d (Mater)
PublicIPs: 54.226.151.237 PrivateIPs: 172.31.22.135

2. Create Namespace “monitoring” for deployments of Prometheus and Grafana:

```
aws Services Search [Alt+S] N. Virginia
root@ip-172-31-22-135:~# kubectl create namespace monitoring
namespace/monitoring created
root@ip-172-31-22-135:~#
```

i-0cf145e1fd5880c4d (Mater)
PublicIPs: 54.226.151.237 PrivateIPs: 172.31.22.135

3. Git clone the Prometheus files into system:

```
aws Services Search [Alt+S] N. Virginia
root@ip-172-31-22-135:~# git clone https://github.com/bibinwilson/kubernetes-prometheus
Cloning into 'kubernetes-prometheus'...
remote: Enumerating objects: 277, done.
remote: Counting objects: 100% (191/191), done.
remote: Compressing objects: 100% (90/90), done.
remote: Total 277 (delta 151), reused 124 (delta 101), pack-reused 86
Receiving objects: 100% (277/277), 69.91 KiB | 6.35 MiB/s, done.
Resolving deltas: 100% (160/160), done.
root@ip-172-31-22-135:~# ls
custom-resources.yaml kubernetes-prometheus m.sh snap
root@ip-172-31-22-135:~# cd kubernetes-prometheus/
root@ip-172-31-22-135:~/kubernetes-prometheus# ls
README.md clusterRole.yaml config-map.yaml prometheus-deployment.yaml prometheus-ingress.yaml prometheus-service.yaml
root@ip-172-31-22-135:~/kubernetes-prometheus#
```

i-0cf145e1fd5880c4d (Mater)
PublicIPs: 54.226.151.237 PrivateIPs: 172.31.22.135

4. In kubernetes-prometheus folder you can see all the Prometheus files have been downloaded:

- prometheus-deployment.yaml
- prometheus-service.yaml
- clusterRole.yaml
- config-map.yaml

```
aws Services Search [Alt+S] N. Virginia
root@ip-172-31-22-135:~/kubernetes-prometheus# ls
clusterRole.yaml config-map.yaml prometheus-deployment.yaml prometheus-service.yaml
root@ip-172-31-22-135:~/kubernetes-prometheus#
```

5. Create Kubernetes resources from configuration files in the current directory:

- `kubectl create -f .`
- `kubectl get all -n monitoring` (to see all objects in that namespace)

```
aws Services Search [Alt+S] N. Virginia
root@ip-172-31-22-135:~/kubernetes-prometheus# kubectl create -f .
clusterrole.rbac.authorization.k8s.io/prometheus created
clusterrolebinding.rbac.authorization.k8s.io/prometheus created
configmap/prometheus-server-conf created
deployment.apps/prometheus-deployment created
service/prometheus-service created
root@ip-172-31-22-135:~/kubernetes-prometheus# kubectl get all -n monitoring
NAME                                READY   STATUS    RESTARTS   AGE
pod/prometheus-deployment-854d8f784f-8rtcv   1/1     Running   0           41s

NAME                                TYPE          CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
service/prometheus-service           NodePort      10.104.107.95   <none>         8080:30000/TCP  40s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/prometheus-deployment  1/1     1             1           41s

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/prometheus-deployment-854d8f784f  1         1         1       41s
root@ip-172-31-22-135:~/kubernetes-prometheus#
```

i-0cf145e1fd5880c4d (Mater)
PublicIPs: 54.226.151.237 PrivateIPs: 172.31.22.135

6. Create a yaml file for Grafana including all the objects in it:

vi 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
        }
      ]
    }
```

```
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
```

```
    containerPort: 3000
  resources:
    limits:
      memory: "2Gi"
      cpu: "1000m"
    requests:
      memory: "1Gi"
      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

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
```

7. Then apply the code by:

- `kubectl create -f grafana.yml`
- `kubectl get pods -n monitoring` (to see pods)
- `kubectl get svc -n monitoring` (to see services)

```
aws Services Search [Alt+S] N. Virginia
root@ip-172-31-22-135:~/kubernetes-prometheus# kubectl create -f grafana.yml
configmap/grafana-datasources created
deployment.apps/grafana created
service/grafana created
root@ip-172-31-22-135:~/kubernetes-prometheus# kubectl get pods -n monitoring
NAME                                READY   STATUS    RESTARTS   AGE
grafana-5dc695bbff-rrbf9            1/1    Running   0          31s
prometheus-deployment-854d8f784f-8rtcv 1/1    Running   0          5m34s
root@ip-172-31-22-135:~/kubernetes-prometheus# kubectl get svc -n monitoring
NAME                TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
grafana             NodePort    10.104.162.207 <none>        3000:32000/TCP 43s
prometheus-service NodePort    10.104.107.95 <none>        8080:30000/TCP 5m45s
root@ip-172-31-22-135:~/kubernetes-prometheus#
```

i-0cf145e1fd5880c4d (Mater)
PublicIPs: 54.226.151.237 PrivateIPs: 172.31.22.135

8. Use the **Public-IP** of worker node and the node ports to access the Prometheus and Grafana dashboards outside the cluster environment:

<http://<IP Address>:NodePort>

Prometheus Dashboard and metrics:

The screenshot shows the Prometheus Alerts page. At the top, there are navigation links for Alerts, Graph, Status, and Help. Below the navigation, there are status indicators for Inactive (0), Pending (0), and Firing (1). A search bar is present with the text 'Filter by name or labels'. The main content area shows a firing alert for the rule '/etc/prometheus/prometheus.rules > devopscube demo alert'. The alert is titled 'High Pod Memory (1 active)'. The alert details include the name 'High Pod Memory', the expression 'sum(container_memory_usage_bytes) > 1', and labels 'severity: slack'. The alert is shown as firing with a red 'FIRING' button. Below the alert details, there is a table with columns for Labels, State, Active Since, and Value.

Labels	State	Active Since	Value
alertname=High Pod Memory severity=slack	FIRING	2024-07-15T10:09:07.454309151Z	9039491072

kubernetes-service-endpoints [show less](#)

Discovered Labels

- address="192.168.1.4:3000"
- meta_kubernetes_endpoint_address_target_kind="Pod"
- meta_kubernetes_endpoint_address_target_name="grafana-5dc695bbff-mrdvc"
- meta_kubernetes_endpoint_node_name="node01"
- meta_kubernetes_endpoint_port_protocol="TCP"
- meta_kubernetes_endpoint_ready="true"
- meta_kubernetes_annotation_endpoints_kubernetes_io_last_change_trigger_time="2024-07-15T10:08:57Z"
- meta_kubernetes_endpoints_annotationpresent_endpoints_kubernetes_io_last_change_trigger_time="true"
- meta_kubernetes_endpoints_name="grafana"
- meta_kubernetes_namespace="monitoring"
- meta_kubernetes_pod_annotation_cni_projectcalico_org_containerID="34dcf56d9e64fc214783af3be3e5a90cc870821d7f9fc4858b914ea1ba4111e4"
- meta_kubernetes_pod_annotation_cni_projectcalico_org_podIP="192.168.1.4/32"
- meta_kubernetes_pod_annotation_cni_projectcalico_org_podIP="192.168.1.4/32"
- meta_kubernetes_pod_annotationpresent_cni_projectcalico_org_containerID="true"

Target Labels

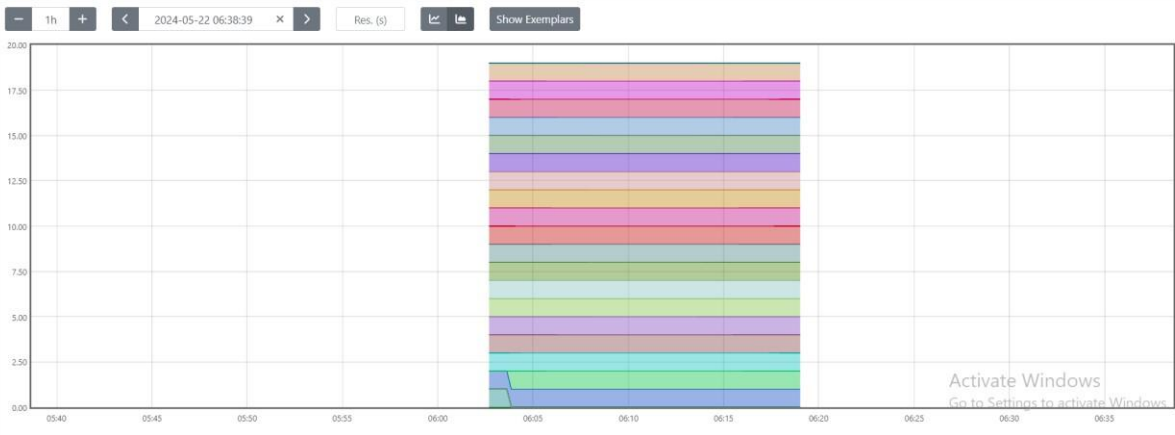
- instance="192.168.1.4:3000"
- job="kubernetes-service-endpoints"
- kubernetes_name="grafana"
- kubernetes_namespace="monitoring"

Q kube_pod_status_ready

Execute

Table Graph

Load time: 104ms Resolution: 14s Result series: 57



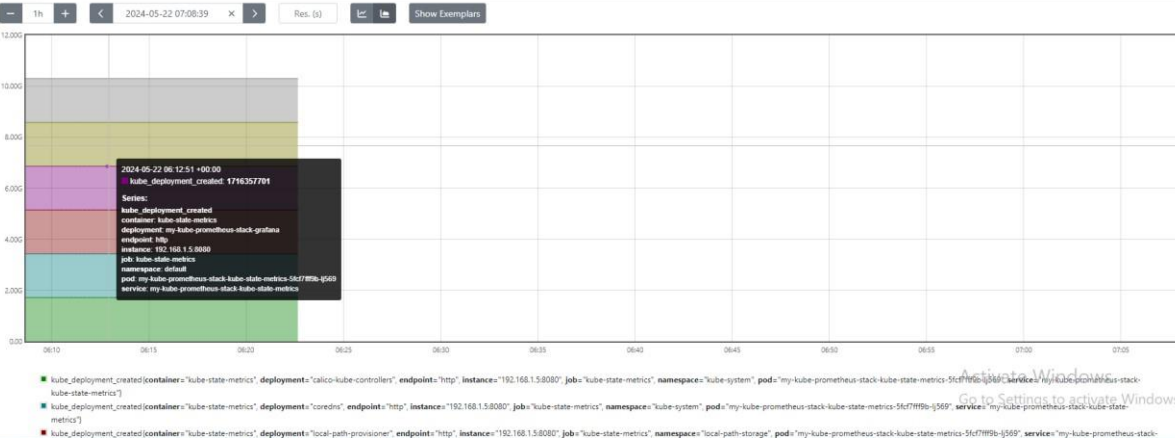
- Use local time
- Enable query history
- Enable autocomplete
- Enable highlighting
- Enable linter

Q kube_deployment_created

Execute

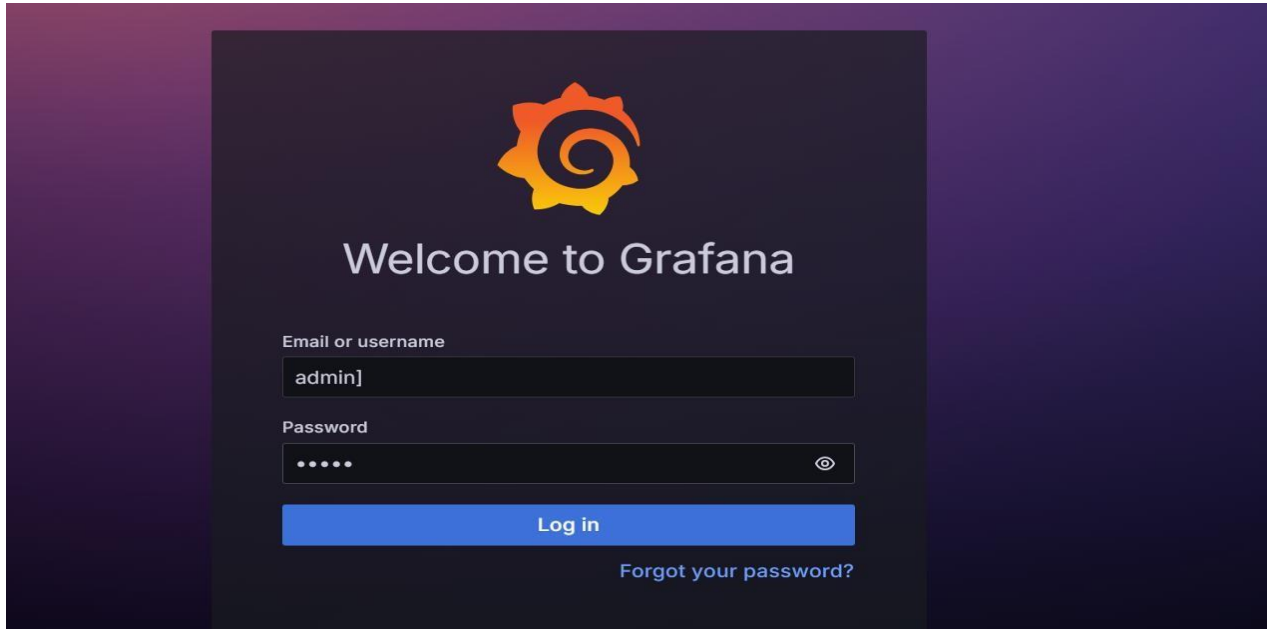
Load time: 89ms Resolution: 14s Result series: 6

Table Graph



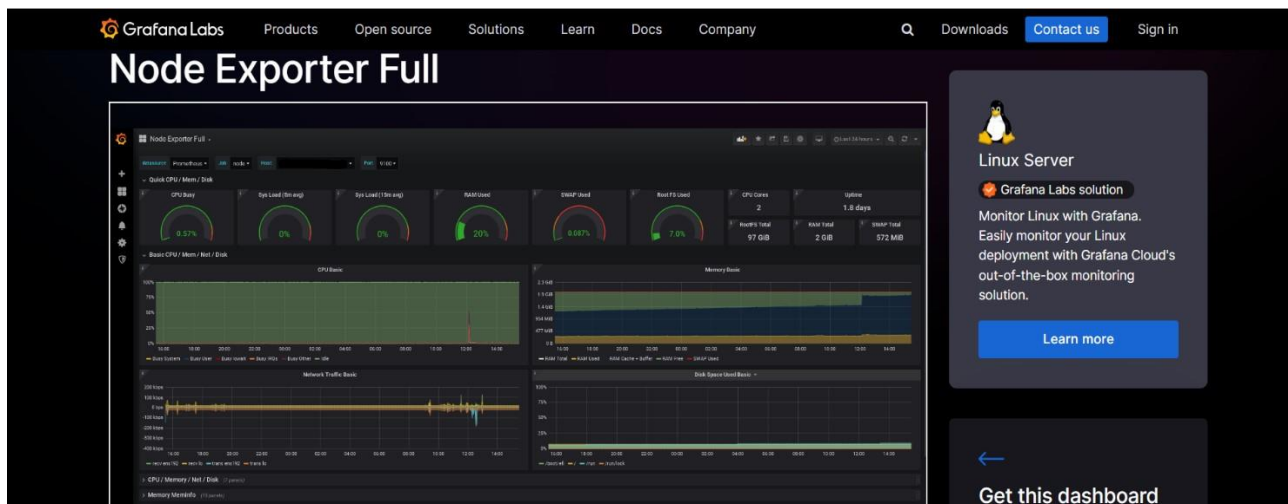
9. Grafana Dashboard :

The default username & password is “admin” for both, then also you can change your password.

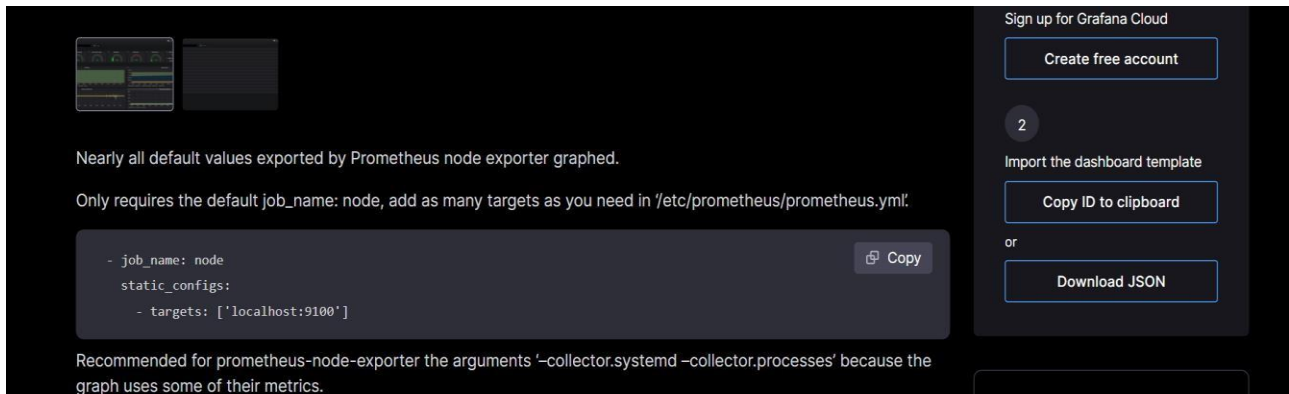


We also have the option of creating our dashboards from scratch as well as importing multiple Grafana dashboards provided by the [Grafana library](#).

We can use this Node Exporter dashboard to monitor & observe the health of our nodes present in our Kubernetes Cluster.



Select Dashboard and copy the Dashboard ID



Nearly all default values exported by Prometheus node exporter graphed.

Only requires the default job_name: node, add as many targets as you need in /etc/prometheus/prometheus.yml.

```
- job_name: node
  static_configs:
    - targets: ['localhost:9100']
```

Recommended for prometheus-node-exporter the arguments '-collector.systemd -collector.processes' because the graph uses some of their metrics.

Sign up for Grafana Cloud

Create free account

2

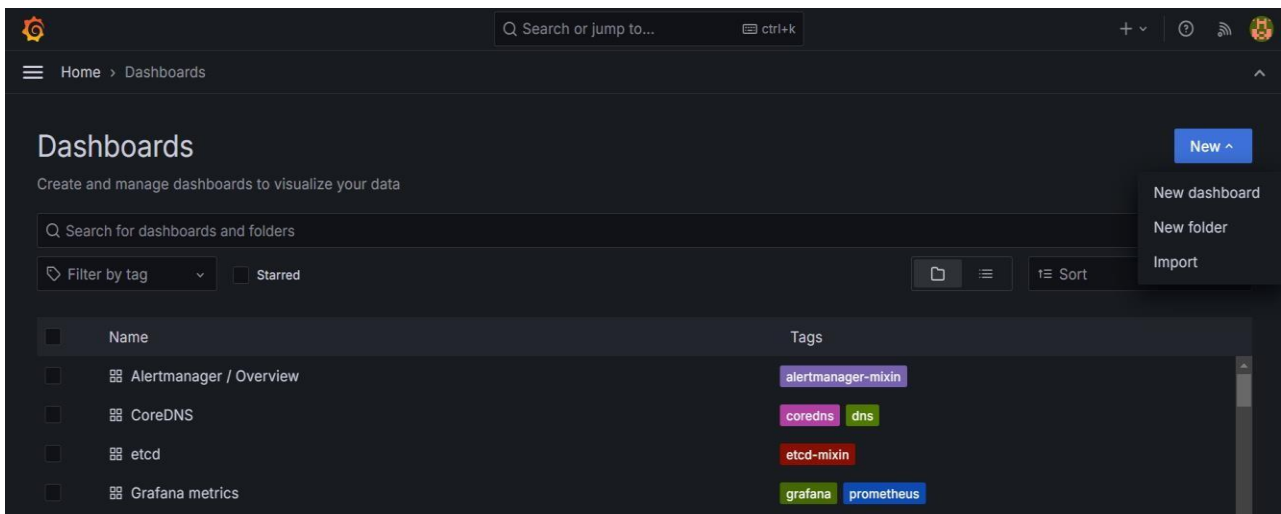
Import the dashboard template

Copy ID to clipboard

or

Download JSON

Under **Dashboards** page we can get the **Import** option:



Home > Dashboards

Dashboards

Create and manage dashboards to visualize your data

Search for dashboards and folders

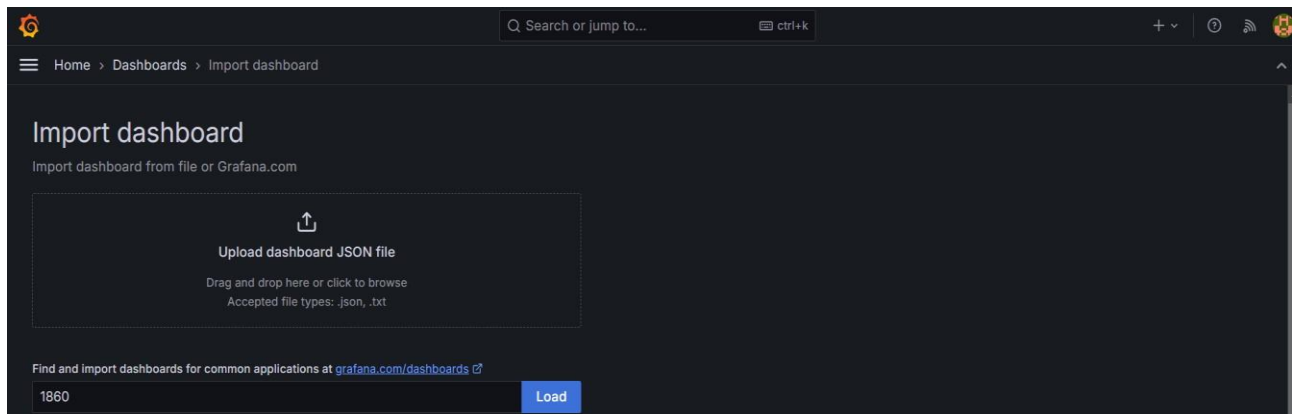
Filter by tag Starred

Sort

New dashboard
New folder
Import

Name	Tags
Alertmanager / Overview	alertmanager-mixin
CoreDNS	coredns dns
etcd	etcd-mixin
Grafana metrics	grafana prometheus

Under "Import Dashboard" page, we need to paste the Dashboard ID that we copied earlier & click on the **Load** button.



Home > Dashboards > Import dashboard

Import dashboard

Import dashboard from file or Grafana.com

Upload dashboard JSON file

Drag and drop here or click to browse

Accepted file types: json, .txt

Find and import dashboards for common applications at grafana.com/dashboards

1860 **Load**

After clicking on the **Load** button, it will auto-load the dashboard from the library after which we can import the dashboard by clicking on the **Import** button.

Options

Name
Node Exporter Full

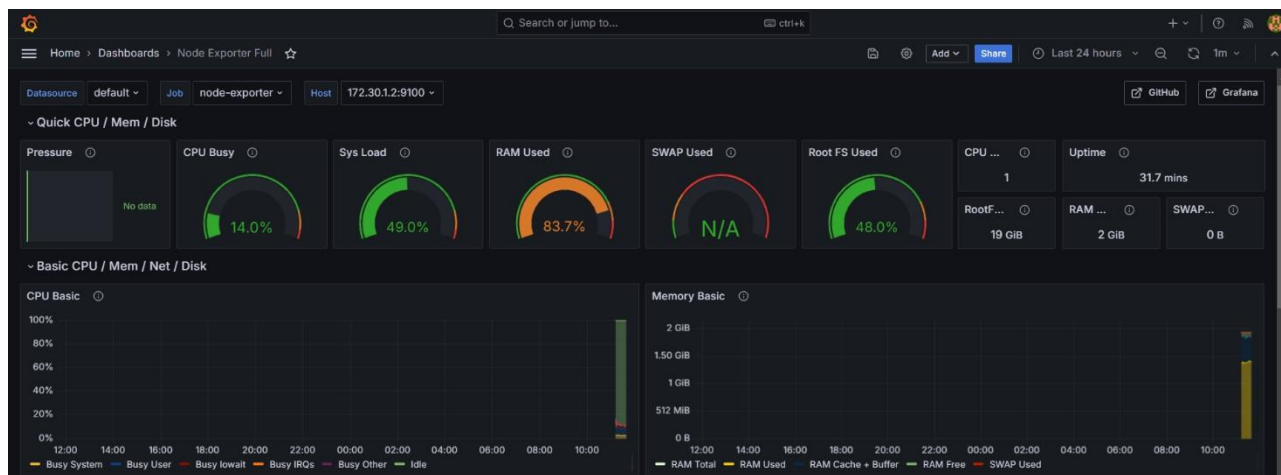
Folder
Dashboards

Unique identifier (UID)
The unique identifier (UID) of a dashboard can be used to uniquely identify a dashboard between multiple Grafana installs. The UID allows having consistent URLs for accessing dashboards so changing the title of a dashboard will not break any bookmarked links to that dashboard.
rYdddIPWk Change uid

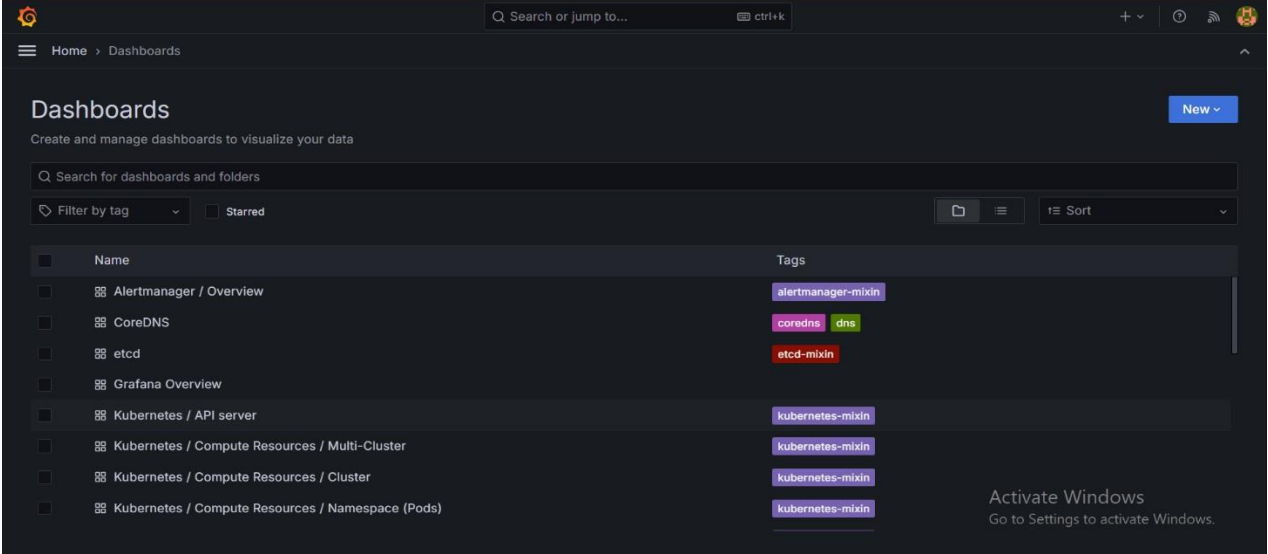
Prometheus
Prometheus

Import Cancel

New imported dashboard:



It has multiple dashboards to monitor the health of the Kubernetes cluster and its resources:



Monitor & observe the resources present in our Kubernetes Cluster:

