`

Prometheus Complete Documentation

Version History

Version Number	Date	Description of Changes	Author
v1.0	15/01/2025	draft	Mritunjay Kumar

Version No.	Date	Changed By	Reason for Plan Revision*	Changes Made	Approved By
V1.0	15-01-2025		First draft	N/A	
				-	
				-	

Contents

Objective

This document aims to guide the setup of a robust monitoring system using Prometheus and Alertmanager for servers, Kubernetes clusters, and Sophos firewalls. It covers installation, configuration, visualization through Grafana, alert setup, and best practices to ensure reliable performance and proactive issue resolution.

Emp ID	
DOJ	
Emp Designation	
Employee First Name	
Employee Last Name	
Display Name	***First Name and *Last Name Only**
DL's to be added	
Emp Department	
Reporting Manager	
Emp Work Location	
Emp Office Address	
Emp Mobile Number	
Emp Personal Email ID	

2. Installation of Prometheus and Alert Manager

2.1 Prometheus Installation

Step 1 - Update System Packages

$ sudo apt update

Step 2 - Create a System User for Prometheus
Now create a group and a system user for Prometheus. To create a group and then add a user to the group, run the following command:

$ sudo groupadd --system prometheus
$ sudo useradd -s /sbin/nologin --system -g prometheus Prometheus

Step 3 - Create Directories for Prometheus
To store configuration files and libraries for Prometheus, you need to create a few directories. The directories will be in the /etc and the /var/lib directory respectively. Use the commands below to create the directories:

$ sudo mkdir /etc/prometheus
$ sudo mkdir /var/lib/prometheus

Step 4 - Download Prometheus and Extract Files
To download the latest update, go to the Prometheus official downloads site and copy the download

$ wget https://github.com/prometheus/prometheus/releases/download/v2.43.0/prometheus-2.43.0.linux-amd64.tar.gz

After the download has been completed, run the following command to extract the contents of the downloaded file:

$ tar vxf prometheus*.tar.gz

Step 5- Navigate to the Prometheus Directory
After extracting the files, navigate to the newly extracted Prometheus directory using the following command:

$ cd prometheus*/

Changing to the Prometheus directory allows for easier management and configuration of the installation. Subsequent steps will be performed within the context of the Prometheus directory.

Step 6 - Move the Binary Files & Set Owner
First, you need to move some binary files (**prometheus** and **promtool**) and change the ownership of the files to the "**prometheus"** user and group. You can do this with the following commands:

$ sudo mv prometheus /usr/local/bin
$ sudo mv promtool /usr/local/bin
$ sudo chown prometheus:prometheus /usr/local/bin/prometheus

$ sudo chown prometheus:prometheus /usr/local/bin/promtool

Step 7 - Move the Configuration Files & Set Owner
Next, move the configuration files and set their ownership so that Prometheus can access them. To do this, run the following commands:

$ sudo mv consoles /etc/prometheus
$ sudo mv console_libraries /etc/prometheus
$ sudo mv prometheus.yml /etc/prometheus
$ sudo chown prometheus:prometheus /etc/prometheus
$ sudo chown -R prometheus:prometheus /etc/prometheus/consoles
$ sudo chown -R prometheus:prometheus /etc/prometheus/console_libraries
$ sudo chown -R prometheus:prometheus /var/lib/prometheus
$ sudo nano /etc/prometheus/prometheus.yml

For prometheus.yml file check in Gitea devops-Monitoring repositort
https://192.168.61.88:3000/Toucan_Payments_India/devops-monitoring

Step 8 - Create Prometheus Systemd Service
$ sudo nano /etc/systemd/system/prometheus.service
Include these settings to the file, save, and exit:

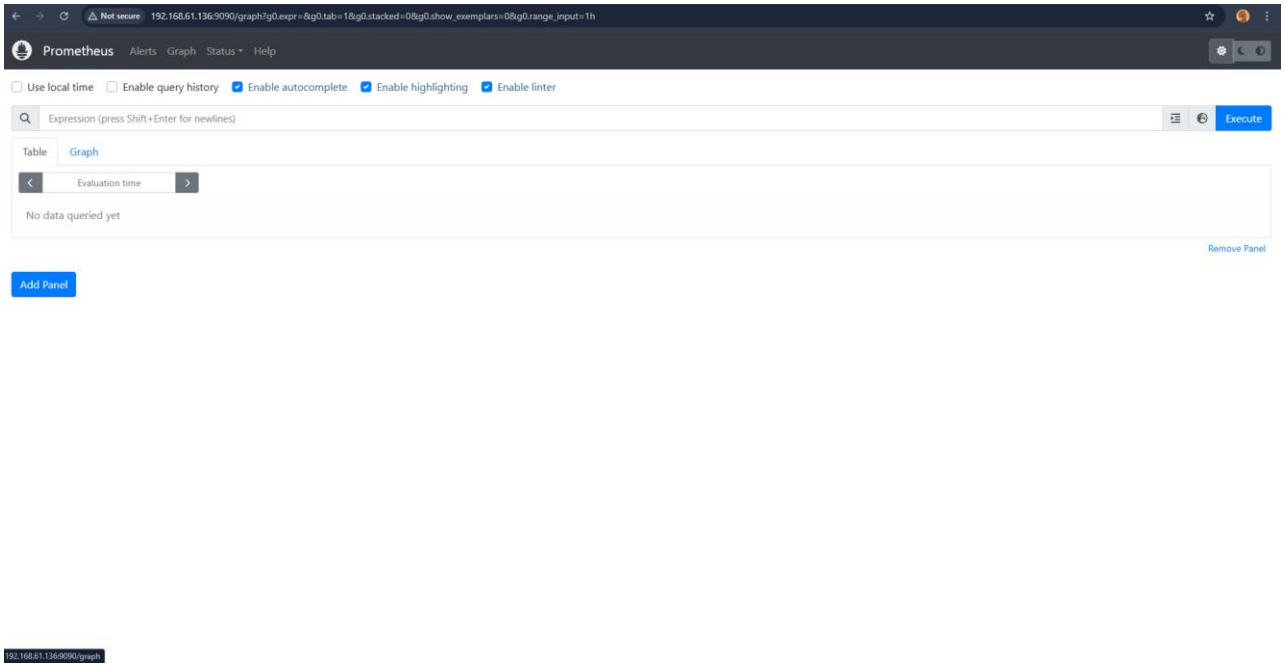
[Unit]
Description=Prometheus
Wants=network-online.target
After=network-online.target
[Service]
User=prometheus
Group=prometheus
Type=simple
ExecStart=/usr/local/bin/prometheus
--config.file /etc/prometheus/prometheus.yml
--storage.tsdb.path /var/lib/prometheus/
--web.console.templates=/etc/prometheus/consoles
--web.console.libraries=/etc/prometheus/console_libraries
[Install] WantedBy=multi-user.target

Step 9 - Reload Systemd
$ sudo systemctl daemon-reload

Step 10 - Start Prometheus Service
$ sudo systemctl enable prometheus
$ sudo systemctl start prometheus

Step 11 - Check Prometheus Status
$ sudo systemctl status prometheus

With Prometheus running successfully, you can access it via your web browser using localhost:9090 or <ip_address>:9090



2.2 Alert Manager Installation

Download Prometheus AlertManager
$ wget https://github.com/prometheus/alertmanager/releases/download/v0.21.0/alertmanager-0.21.0.linux-amd64.tar.gz

Create User
Create a Prometheus user, required directories, and make prometheus user as the owner of those directories.

$ sudo groupadd -f alertmanager
$ sudo useradd -g alertmanager --no-create-home --shell /bin/false alertmanager
$ sudo mkdir -p /etc/alertmanager/templates
$ sudo mkdir /var/lib/alertmanager
$ sudo chown alertmanager:alertmanager /etc/alertmanager
$ sudo chown alertmanager:alertmanager /var/lib/alertmanager

Unpack Prometheus AlertManager Binary
$ tar -xvf alertmanager-0.21.0.linux-amd64.tar.gz
$ mv alertmanager-0.21.0.linux-amd64 alertmanager-files

Install Prometheus AlertManager
$ sudo cp alertmanager-files/alertmanager /usr/bin/
$ sudo cp alertmanager-files/amtool /usr/bin/
$ sudo chown alertmanager:alertmanager /usr/bin/alertmanager

Install Prometheus AlertManager Configuration File
$ sudo cp alertmanager-files/alertmanager.yml /etc/alertmanager/alertmanager.yml
$ sudo chown alertmanager:alertmanager /etc/alertmanager/alertmanager.yml

Setup Prometheus AlertManager Service
$ sudo vi /usr/lib/systemd/system/alertmanager.service

$ sudo chmod 664 /usr/lib/systemd/system/alertmanager.service

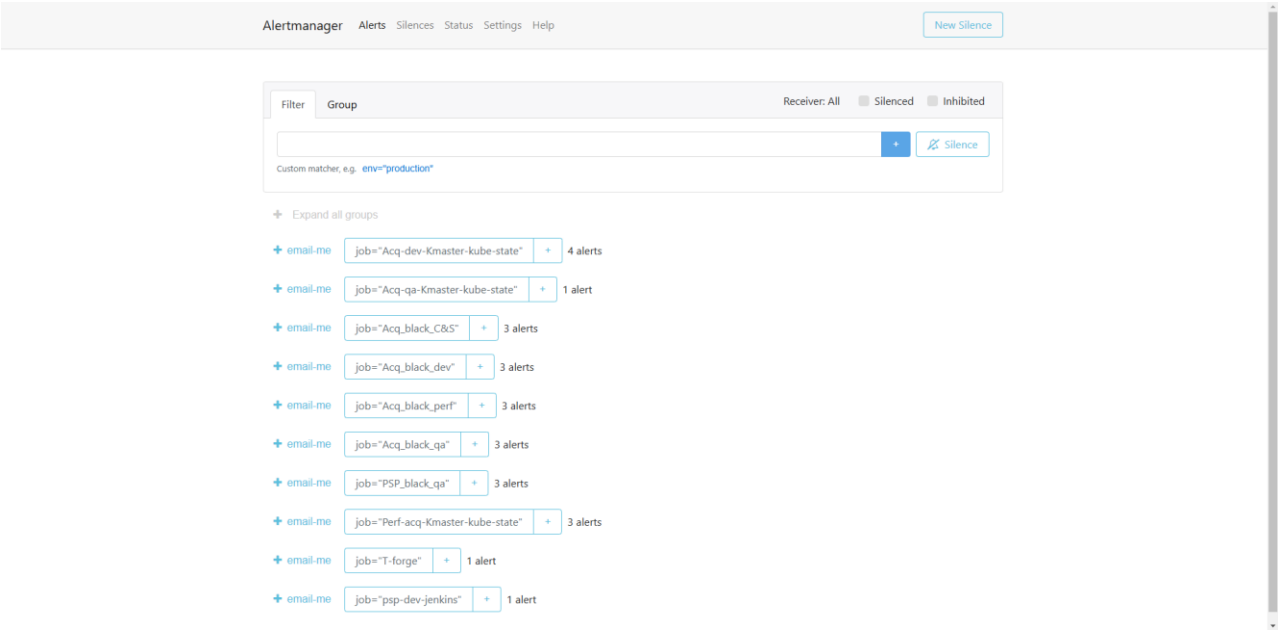
Reload systemd and Register Prometheus AlertManager
$ sudo systemctl daemon-reload
$ sudo systemctl start alertmanager

Configure Prometheus AlertManager to start at boot
$ sudo systemctl enable alertmanager.service

Check the alertmanager service status using the following command.
$ sudo systemctl status alertmanager

Access Prometheus AlertManager UI
http://<alertmanager-ip>:9093

With Prometheus running successfully, you can access it via your web browser using localhost:9093 or <ip_address>:9090



3. Monitoring Servers

Monitoring server resources such as CPU utilization, disk utilization, memory utilization, and server availability. To achieve this, we use **Node Exporter** and configure alert rules in the alert_rules.yaml file located in the Prometheus directory.

Steps for Server Monitoring:

1. **Install Node Exporter:**
   - Install Node Exporter on each server to be monitored.
   - Node Exporter collects system-level metrics such as CPU, memory, disk, and network usage.

1.1 Steps to Install Node Exporter

Download Node Exporter
$ wget https://github.com/prometheus/node_exporter/releases/download/v1.0.1/node_exporter-1.0.1.linux-amd64.tar.gz

Create User
Create a Node Exporter user, required directories, and make prometheus user as the owner of those directories.

```
$ sudo groupadd -f node_exporter
$ sudo useradd -g node_exporter --no-create-home --shell /bin/false node_exporter
$ sudo mkdir /etc/node_exporter
$ sudo chown node_exporter:node_exporter /etc/node_exporter
```

Unpack Node Exporter Binary
```
$ tar -xvf node_exporter-1.0.1.linux-amd64.tar.gz
$ mv node_exporter-1.0.1.linux-amd64 node_exporter-files
```

Install Node Exporter
```
$ sudo cp node_exporter-files/node_exporter /usr/bin/
$ sudo chown node_exporter:node_exporter /usr/bin/node_exporter
```

Setup Node Exporter Service
Create a node_exporter service file.
$ sudo vi /usr/lib/systemd/system/node_exporter.service

Add the following configuration

```
[Unit]
Description=Node Exporter
Documentation=https://prometheus.io/docs/guides/node-exporter/
Wants=network-online.target
After=network-online.target

[Service]
User=node_exporter
Group=node_exporter
Type=simple
Restart=on-failure
ExecStart=/usr/bin/node_exporter \
  --web.listen-address=:9200

[Install]
WantedBy=multi-user.target
```

$ sudo chmod 664 /usr/lib/systemd/system/node_exporter.service

Reload systemd and Start Node Exporter
Reload the systemd service to register the prometheus service and start the prometheus service.
```
$ sudo systemctl daemon-reload
$ sudo systemctl start node_exporter
```

Check the node exporter service status using the following command.

$ sudo systemctl status node_exporter

Configure node_exporter to start at boot
$ sudo systemctl enable node_exporter.service

If firewalld is enabled and running, add a rule for port 9200
$ sudo firewall-cmd --permanent --zone=public --add-port=9200/tcp
$ sudo firewall-cmd –reload

Verify Node Exporter is Running
Verify the exporter is running by visiting the /metrics endpoint on the node on port 9200
http://<node_exporter-ip>:9200/metrics

2. **Data Collection by Prometheus:**
   - Configure Prometheus to scrape metrics from Node Exporter by adding the server's details to the prometheus.yml file.

```
- job_name: 'vcip-kmaster'
  static_configs:
    - targets:
        - '192.168.61.80:9200'
- job_name: 'vcip-kworker1'
  static_configs:
    - targets:
        - '192.168.61.81:9200'
- job_name: 'vcip-kworker2'
  static_configs:
    - targets:
        - '192.168.63.113:9200'
- job_name: 'acq-dev-kworker2'
  static_configs:
    - targets:
        - '192.168.61.63:9200'
```

3. **Configured Alerts:**
   - **CPU Utilization:** Alerts for high CPU usage to ensure optimal performance.
   - **Disk Utilization:** Monitors disk usage to avoid running out of storage.
   - **Memory Utilization:** Tracks memory usage to prevent system slowdowns.
   - **Server Down (VM Down):** Triggers alerts when a server becomes unreachable to quickly address outages.

4. **Alert Rules:**
   - These alert rules are defined in the alert_rules.yaml file, which resides in the Prometheus directory.
   - Integrate the alert_rules.yaml file with Prometheus by specifying its location in the prometheus.yml configuration.

5. **Integration and Validation**

Update the prometheus.yml File:

- Add the path to the alert_rules.yaml file in the prometheus.yml configuration:

```
rule_files:
  - 'alert-rules/*.yaml'  # File containing alert rules
  # Uncomment and add more rule files if needed
```

Create the alert_rules.yaml File:

- Place the following alert rules in the alert_rules.yaml file located in the Prometheus directory.

Alert Rules for Metrics

CPU Utilization Alert:

```
groups:
- name: CPU-Utilisation-Alert
  rules:
  # Critical Alert: CPU Utilization > 85%
  - alert: CPU-Utilisation-Critical
    expr: 100 - (avg by (instance, job) (rate(node_cpu_seconds_total{mode="idle"}[5m]) * 100)) > 85
    for: 5m
    labels:
      severity: 'critical'
    annotations:
      summary: "{{ $labels.job }}-{{ $labels.instance }} CPU utilizing more than 85. Current value is {{ $value }}%."
```

Disk Utilization Alert:

```
groups:
- name: Disk-Utilisation-Alert
  rules:
  # Critical Alert: Disk Utilization above 85%
  - alert: Disk-Utilisation-Critical
    expr: 100 - ((node_filesystem_avail_bytes{mountpoint="/"} / node_filesystem_size_bytes{mountpoint="/"}) * 100) > 85
    for: 5m
    labels:
      severity: 'critical'
    annotations:
      summary: "Disk usage on {{ $labels.job }}-{{ $labels.instance }} ({{ $labels.fstype }}) has exceeded 85%. Current value is {{ $value }}%."
```

Memory Utilization Alert:

```
groups:
- name: Memory-Utilisation-Alert
  rules:
  # Critical Alert: Memory Utilization above 85%
  - alert: Memory-Utilisation-Critical
    expr: 100 - ((node_memory_MemAvailable_bytes / node_memory_MemTotal_bytes) * 100) > 85
    for: 5m
    labels:
      severity: 'critical'
    annotations:
      summary: "{{ $labels.job }}-{{ $labels.instance }} memory utilization more than 85%. Current value is {{ $value }}%."
```

Server Down (Instance Down) Alert:

```
groups:
  - name: InstanceDown
    rules:
      - alert: InstanceDownAlert
        expr: up == 0
        for: 5m
        labels:
          severity: critical
        annotations:
          summary: "Instance is Down"
```

4. Monitoring Kubernetes Clusters

Kube-state-metrics

Open-source service that listens to the Kubernetes API server and generates metrics about the state of Kubernetes objects. These metrics are designed to provide detailed insights into the cluster's state and are used primarily for monitoring and alerting purposes.

Metrics Overview

The metrics generated by Kube-state-metrics include information about the following Kubernetes resources:

- **Pods**:

Status, restarts, conditions, etc.

- **Deployments**:

Desired replicas, available replicas, updated replicas, etc.

- **Nodes**:

Node conditions (e.g., Ready, Disk Pressure), capacity, and allocatable resources.

- **Persistent Volume Claims (PVCs)**:

Status, capacity, and storage class details.

Step 1: - Download from the Gitea code for Kube state metric.

Gitea link: - https://192.168.61.88:3000/Toucan_Payments_India/devops-monitoring.git

$ cd devops-monitoring/kube-exporter/

Step 2: - Create a namespace with the name of monitoring.

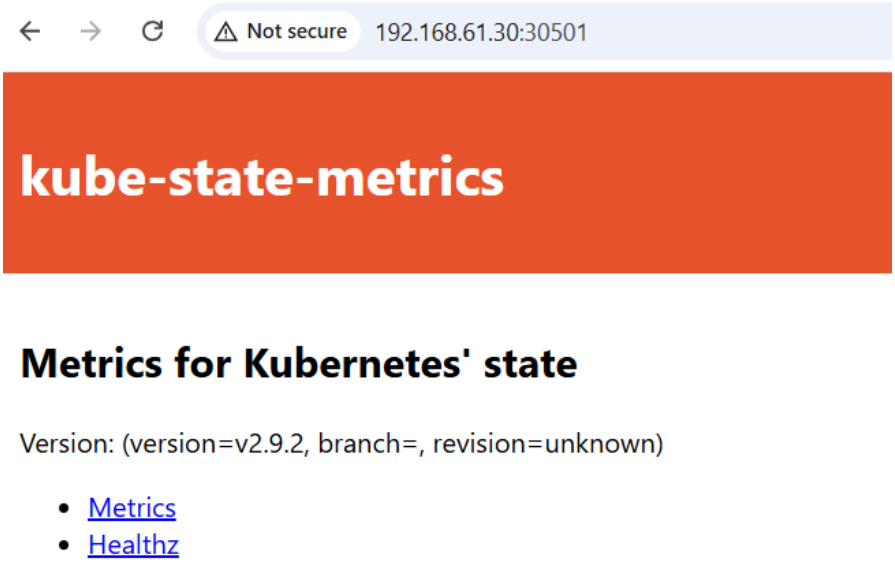
$ kubectl create ns monitoring

$ kubectl apply –f kube-state-exporter.yaml

```
touadmin@acq-kmaster:~$ kubectl get all -n monitoring
NAME                                      READY   STATUS    RESTARTS   AGE
pod/blackbox-exporter-6c76c844c-6zqpv     1/1     Running   0          30d
pod/kube-state-metrics-78c5c9f66b-j5gkr   1/1     Running   0          29d

NAME                          TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
service/blackbox-exporter     NodePort   10.99.147.33    <none>        9115:30115/TCP   30d
service/kube-state-metrics    NodePort   10.105.39.177   <none>        8080:30501/TCP   29d

NAME                                  READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/blackbox-exporter     1/1     1            1           30d
deployment.apps/kube-state-metrics    1/1     1            1           29d

NAME                                            DESIRED   CURRENT   READY   AGE
replicaset.apps/blackbox-exporter-6c76c844c     1         1         1       30d
replicaset.apps/kube-state-metrics-78c5c9f66b   1         1         1       29d
touadmin@acq-kmaster:~$
```

Step 3: - For the Kube state metric enter the IP address and port number.

Eg. http://192.168.61.30:30501/



Note: - Install the above processes in required k8 master cluster for the kube-state metrics.

Step 4: - To get the Kube state metric in the Prometheus. We need to add in the Prometheus yaml file.

$ cd /etc/prometheus

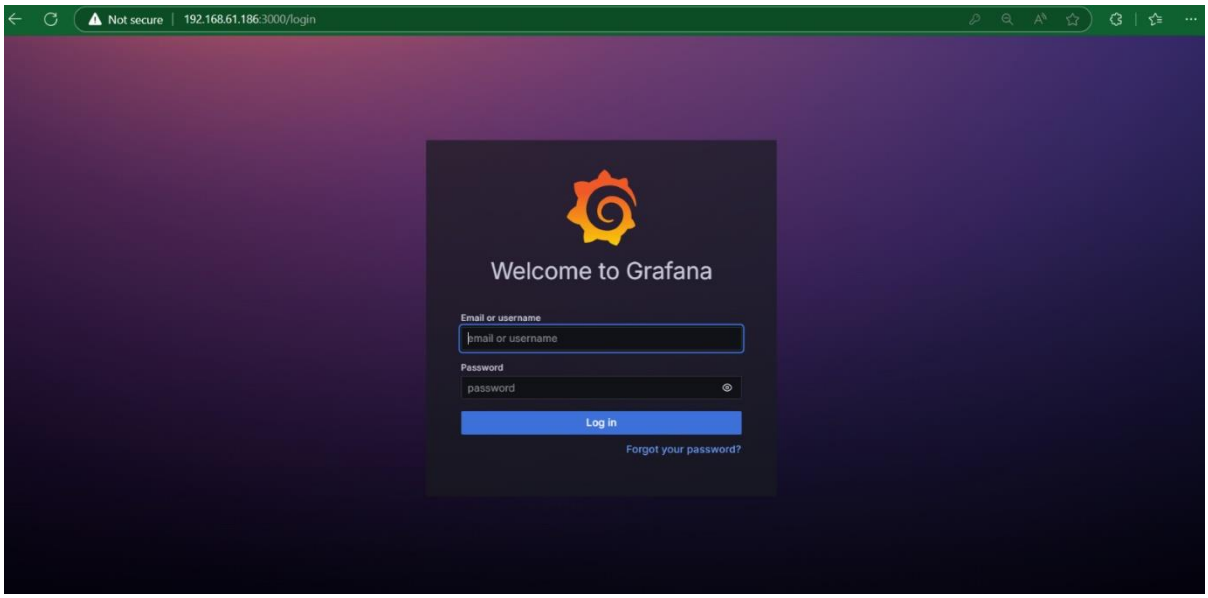
```
sanath@prometheus:/etc/prometheus$ ls
alert-rules  console_libraries  consoles  prometheus.yml  targets.yaml
sanath@prometheus:/etc/prometheus$
```

Adding the below lines of code in prometheus.yaml file for Kube state metrics for the single cluster.

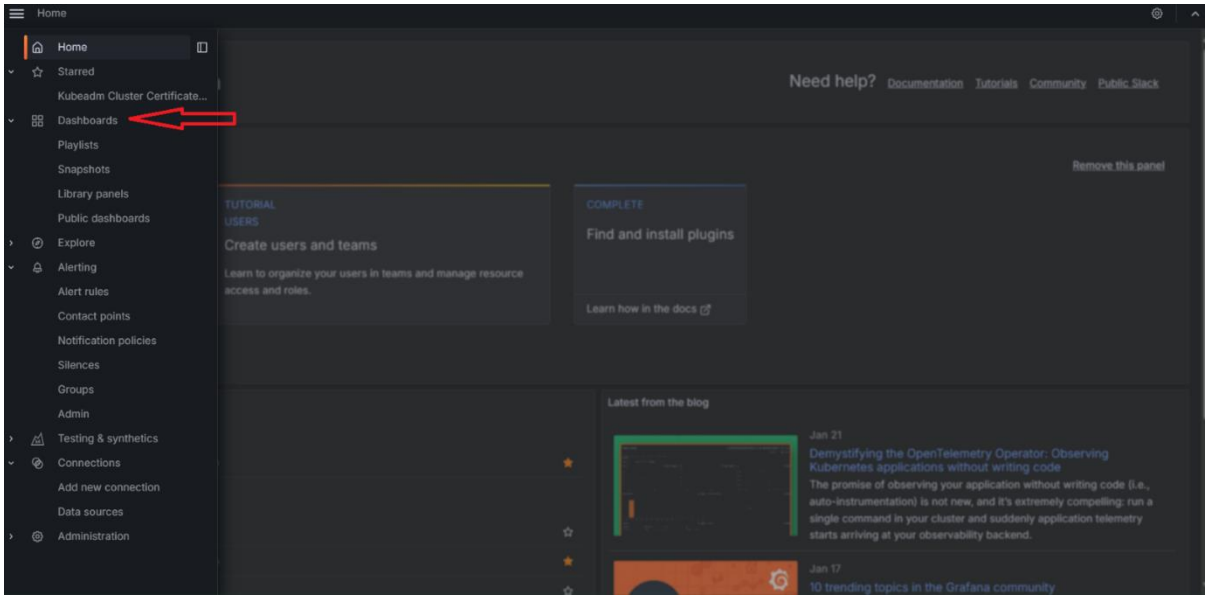
```
- job_name: 'lyra-Kmaster-kube-state'
  honor_timestamps: true
  metrics_path: /metrics
  scheme: http
  static_configs:
    - targets: ['192.168.61.30:30501']
  metric_relabel_configs:
    - target_label: cluster
      replacement: lyra-Kmaster-kube-state
```

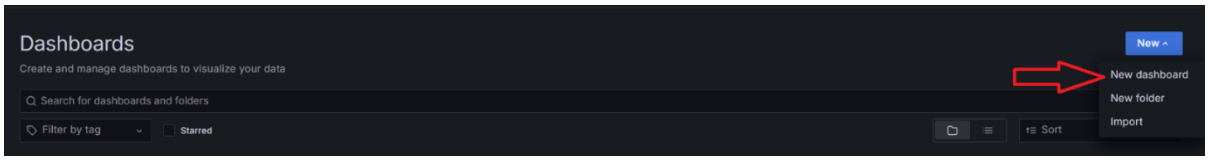
Step 5: - We need to import Kube state metrics dashboard in Grafana. Login to the Grafana dashboard with specific user credential.

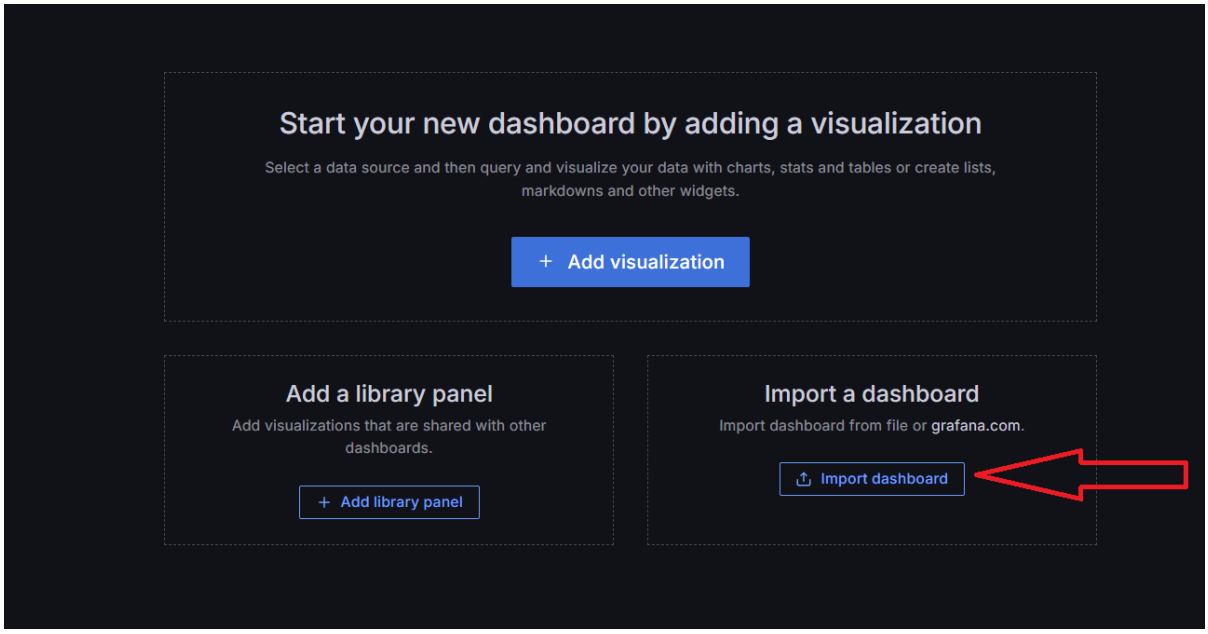
Eg. 192.168.61.186:3000

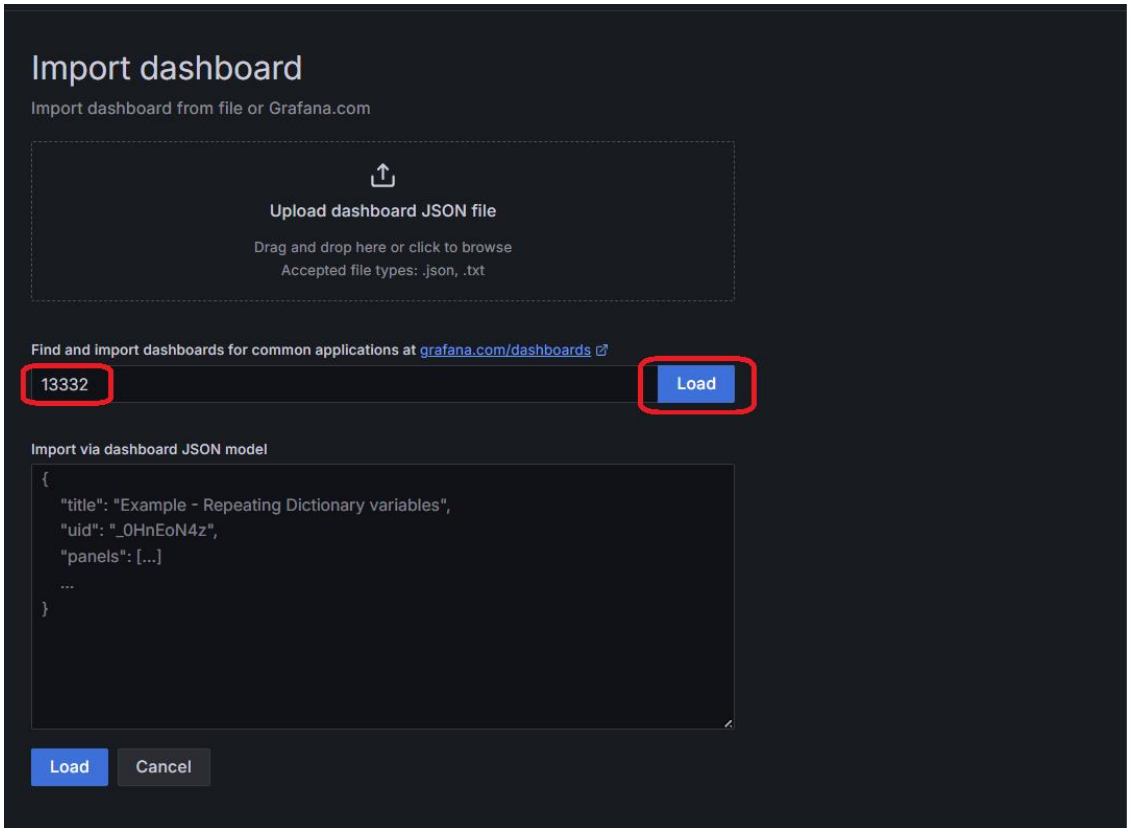


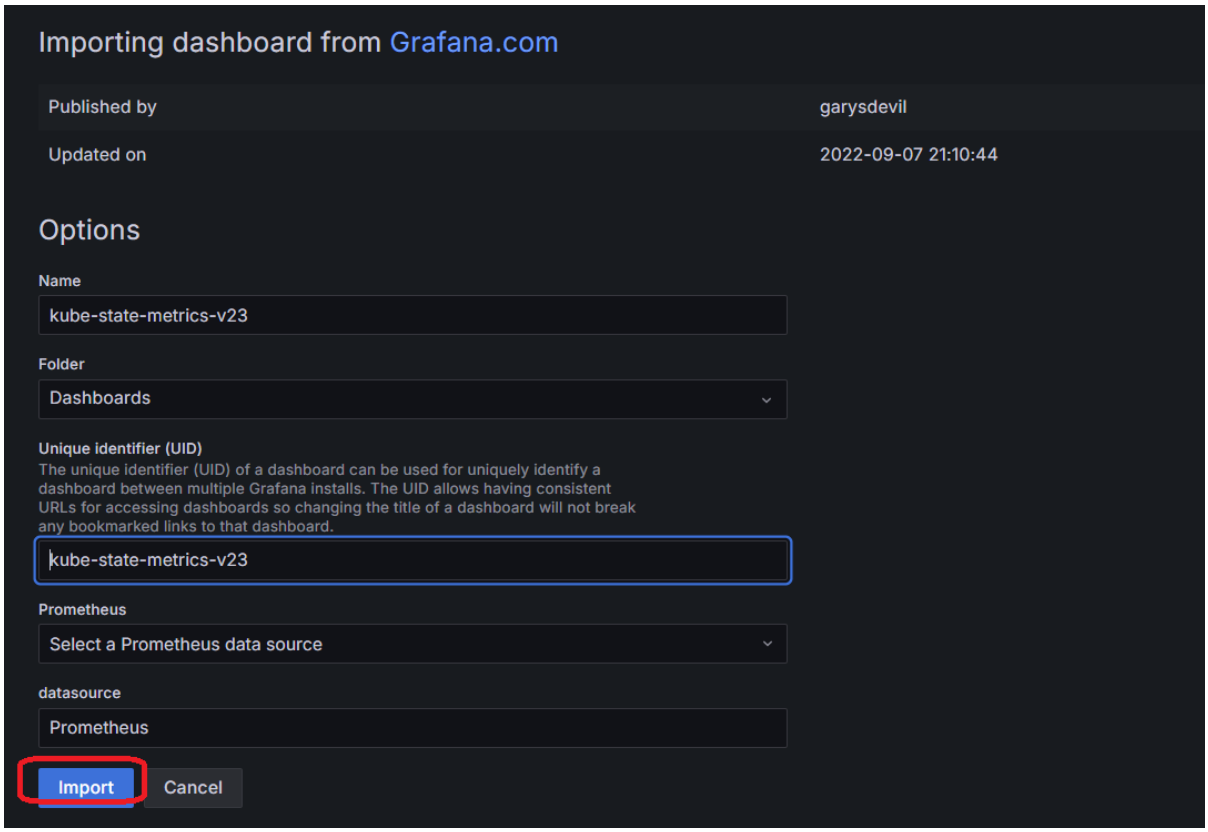
Step 6: - Click on the **dashboards** and select the **new dashboard** option. In the Import section select the **import dashboard** option In the Import dashboard section enter the **Grafana URL or ID (13332)** of the Kube state metric dashboard click on load option.


Home
Home
Starred
Kubeadm Cluster Certificate...
Dashboards
Playlists
Snapshots
Library panels
Public dashboards
Explore
Alerting
Alert rules
Contact points
Notification policies
Silences
Groups
Admin
Testing & synthetics
Connections
Add new connection
Data sources
Administration
Need help? Documentation Tutorials Community Public Slack
Remove this panel
TUTORIAL
USERS
Create users and teams
Learn to organize your users in teams and manage resource access and roles.
COMPLETE
Find and install plugins
Learn how in the docs
Latest from the blog
Jan 21
Demystifying the OpenTelemetry Operator: Observing Kubernetes applications without writing code
The promise of observing your application without writing code (i.e., auto-instrumentation) is not new, and it's extremely compelling: run a single command in your cluster and suddenly application telemetry starts arriving at your observability backend.
Jan 17
10 trending topics in the Grafana community

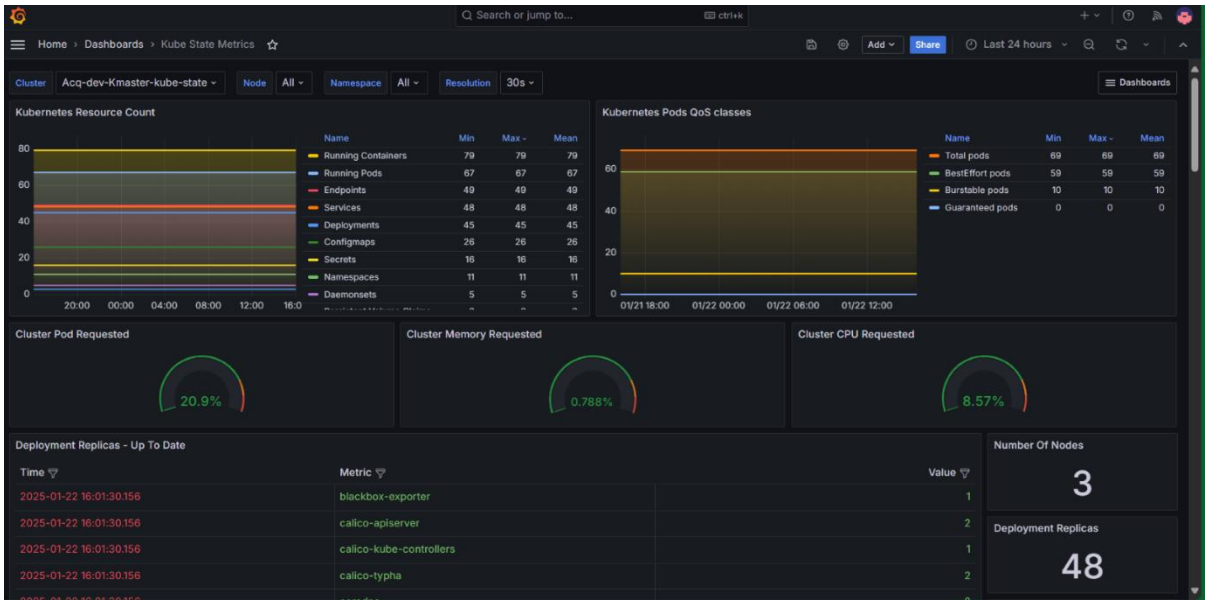

Dashboards
Create and manage dashboards to visualize your data
New
New dashboard
New folder
Import
Search for dashboards and folders
Filter by tag
Starred
Sort


Start your new dashboard by adding a visualization
Select a data source and then query and visualize your data with charts, stats and tables or create lists, markdowns and other widgets.
+ Add visualization
Add a library panel
Add visualizations that are shared with other dashboards.
+ Add library panel
Import a dashboard
Import dashboard from file or grafana.com.
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
13332
Load
Import via dashboard JSON model
{
"title": "Example - Repeating Dictionary variables",
"uid": "_0HnEoN4z",
"panels": [...]
...
}
Load
Cancel


Importing dashboard from Grafana.com
Published by
garysdevil
Updated on
2022-09-07 21:10:44
Options
Name
kube-state-metrics-v23
Folder
Dashboards
Unique identifier (UID)
The unique identifier (UID) of a dashboard can be used for 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.
kube-state-metrics-v23
Prometheus
Select a Prometheus data source
datasource
Prometheus
Import
Cancel

Step 7: - Final dashboard for the Kube state metric.



Flow charts

4.1. Certificate Exporter

Used to track the expiration dates of kubeadm certificates, ensuring the stability and security of Kubernetes clusters. This exporter is particularly tailored to monitor critical certificates managed by kubeadm.

Key Features

1. **Kubeadm Certificate Monitoring**:
2. Tracks the expiration of essential certificates generated and managed by kubeadm, including:
    a. Kubernetes API Server (`apiserver.crt`)
    b. Controller Manager
    c. Scheduler
    d. etcd certificates (`etcd/server.crt`, `etcd/peer.crt`, etc.)

Step 1: - Download from the Gitea code for Kube state metric.

Gitea link:- https://192.168.61.88:3000/Toucan_Payments_India/devops-monitoring.git

$ cd devops-monitoring/ cert-exporter /

Step 2: - Apply the gitea files of (service.yaml , x509.yaml) in the cluster. The pods are run in the **kube-sysem namespace**.

$ Kubcetl apply –f x509.yaml

$ $ Kubcetl apply –f service.yaml

```
touadmin@qa-shared-kmaster:/etc/kubernetes/manifests$ kubectl get po -n kube-system
NAME                                          READY   STATUS    RESTARTS        AGE
coredns-765dddf964-jlkrq                      1/1     Running   0               76d
coredns-765dddf964-sc7m8                      1/1     Running   0               76d
etcd-qa-shared-kmaster                        1/1     Running   1 (93d ago)     145d
kube-apiserver-qa-shared-kmaster              1/1     Running   1 (93d ago)     145d
kube-controller-manager-qa-shared-kmaster     1/1     Running   15 (5d1h ago)   145d
kube-proxy-44mq8                              1/1     Running   1 (93d ago)     145d
kube-proxy-jn49x                              1/1     Running   1 (93d ago)     145d
kube-proxy-wtgf8                              1/1     Running   1 (93d ago)     145d
kube-scheduler-qa-shared-kmaster              1/1     Running   11 (5d1h ago)   145d
kube-state-metrics-69dd4877d5-nfghh           1/1     Running   0               64d
nfs-pod-provisioner-6f5cb5568b-24vmd          1/1     Running   13 (11d ago)    142d
x509-certificate-exporter                     1/1     Running   0               49d
x509-certificate-exporter-qa-shared-kmaster   1/1     Running   0               49d
touadmin@qa-shared-kmaster:/etc/kubernetes/manifests$
```

Note: - Install the above processes in required k8 master cluster for the certification expire of kubeadm.

Step 3: - To get the certification expire of kubeadm for the cluster in the Prometheus. We need to add in the Prometheus yaml file.

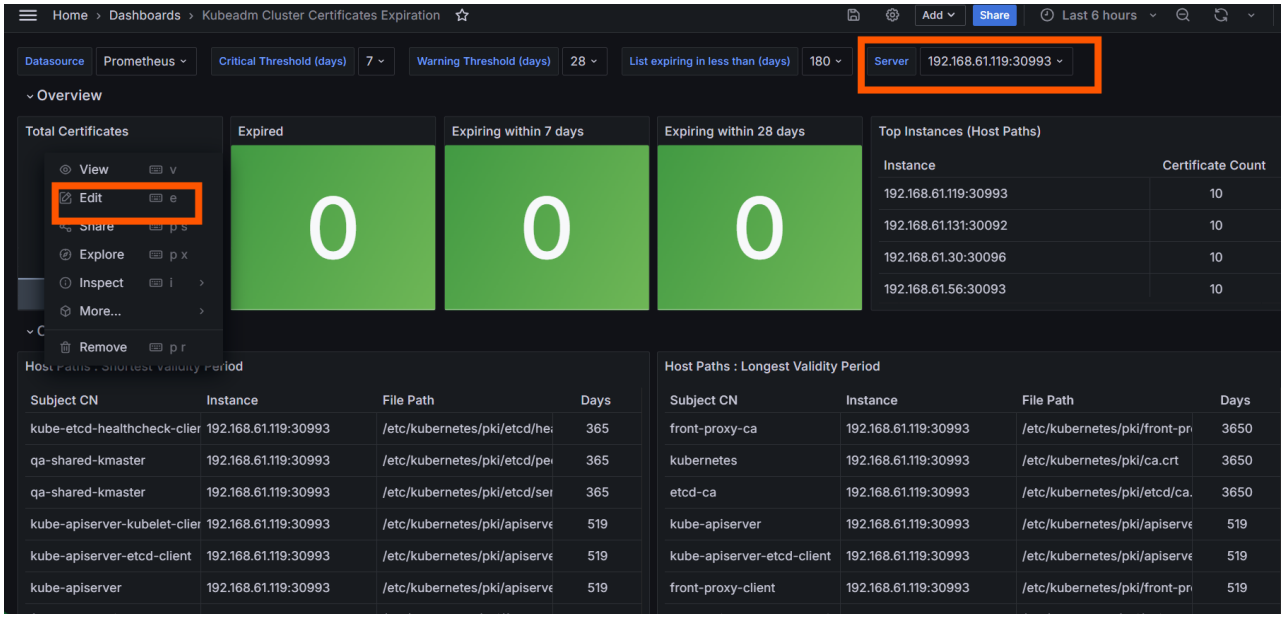
$ cd /etc/prometheus/

```
sanath@prometheus:/etc/prometheus$ ls
alert-rules  console_libraries  consoles  prometheus.yml  targets.yaml
sanath@prometheus:/etc/prometheus$
```

Adding the below lines of code in prometheus.yaml file for certification expire of kubeadm for single cluster.

```
- job_name: 'x509-certs-expiry-Dev-switch-kmaster'
  static_configs:
    - targets:
        - '192.168.63.135:30099'
```

Step 4: - Follow the above steps from s**tep5 and step6 in the Kube state metrics** for the dashboard setup. Use **Grafana URL or ID (13922)** for the kubeadm certificate expire dashboard.



For server label goto the edit panel for each panel replace for the old query with new query

Total Certificates

old query - count(x509_cert_not_after)
new query - count(x509_cert_not_after{instance=~"$Server"})

Host Paths : Shortest Validity Period

old query - bottomk(10, (x509_cert_not_after{filepath!=""} - x509_cert_not_before) / 86400)
new query - bottomk(10, ((x509_cert_not_after{filepath!=""} - x509_cert_not_before{filepath!="", instance="$Server"}) / 86400))

Host Paths : Longest Validity Period

old query - topk(10, (x509_cert_not_after{filepath!=""} - x509_cert_not_before) / 86400
new query - topk(10, ((x509_cert_not_after{filepath!="" , instance=~"$Server"} - x509_cert_not_before{instance=~"$Server"}) / 86400))

Expired

old query - sum(((x509_cert_not_after - time()) / 86400) < bool 0)
new query - sum(((x509_cert_not_after{instance=~"$Server"} - time()) / 86400) < bool 0)

Expiring within 7 days

old query - sum(0 < ((x509_cert_not_after - time()) / 86400) < bool $critical_threshold)
new query - sum(0 < ((x509_cert_not_after{instance=~"$Server"} - time()) / 86400) < bool $critical_threshold)

Expiring within 28 days

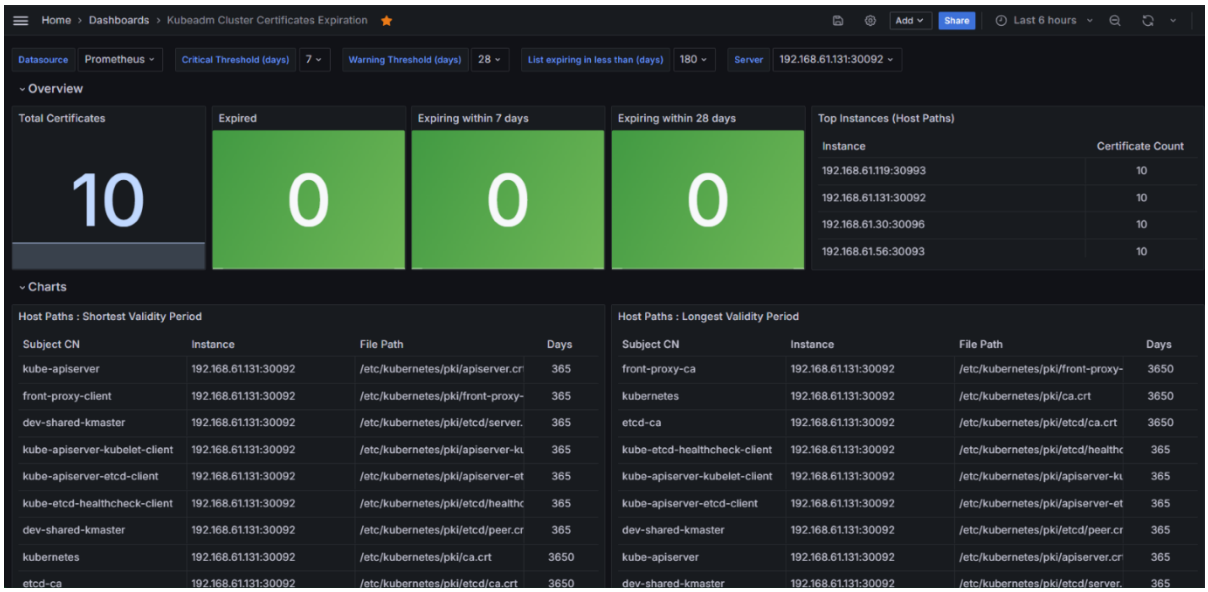
old query - sum(0 < ((x509_cert_not_after - time()) / 86400) < bool $warning_threshold)

new query - sum(0 < ((x509_cert_not_after{instance=~"$Server"} - time()) / 86400) < bool $warning_threshold)

Top Issuers

old quey - topk(10, sort_desc(count by (issuer_CN) (x509_cert_not_after)))
new query - topk(10, sort_desc(count by (issuer_CN) (x509_cert_not_after{instance=~"$Server"})))

Step 5: - Final dashboard for the kubeadm certificate expires.



4.2. Blackbox Exporter

Overview

The blackbox exporter that monitoring and probing of Jenkins, Spira, JFrog, Nexus, Grafana, and Prometheus. It provides performance of these services, ensuring their reliability and responsiveness.

Key Features

- Multi-protocol support (HTTP, HTTPS, DNS)
- SSL/TLS verification

Configuration Guide

- Add a blackbox user

$ sudo useradd --no-create-home blackbox

- Download and extract the Blackbox binary:

$ wget https://github.com/prometheus/blackbox_exporter/releases/download/v0.21.0-rc.0/blackbox_exporter-0.21.0-rc.0.linux-amd64.tar.gztar -xvf blackbox_exporter-0.21.0-rc.0.linux-amd64.tar.gzsudo mkdir /etc/blackbox

- Copy files from the blackbox setup:

$ sudo cp blackbox_exporter-0.21.0-rc.0.linux-amd64/blackbox_exporter /usr/local/bin/

$ sudo cp blackbox_exporter-0.21.0-rc.0.linux-amd64/blackbox.yml /etc/blackbox/

- Adding content to blackbox's configuration file

$ **sudo vim /etc/blackbox/blackbox.yml**

```
---
modules:
http_prometheus:
  prober: http
  timeout: 5s
  http:
    method: GET
    valid_http_versions: ["HTTP/1.1", "HTTP/2"]
    fail_if_ssl: false
    fail_if_not_ssl: false
 ---
```

- Give the user 'blackbox' permission to the file used to run the blackbox binary.

$ sudo chown blackbox:blackbox /usr/local/bin/blackbox_exporter
$ sudo chown -R blackbox:blackbox /etc/blackbox/*

- Add the blackbox startup configs in the service script.This will let us start, stop, restart & check its status easily.

$ sudo vim /etc/systemd/system/blackbox.service

```
---
[Unit]
Description=Blackbox
Wants=network-online.target
After=network-online.target

[Service]
```

User=blackbox
Group=blackbox
Type=simple
ExecStart=/usr/local/bin/blackbox_exporter --config.file=/etc/blackbox/blackbox.yml --web.listen-address="0.0.0.0:9115"

[Install]
WantedBy=multi-user.target

- Run the following to add the above service unit and start blackbox.

$ sudo systemctl daemon-reload

$ sudo systemctl enable blackbox

$ sudo systemctl start blackbox

$ sudo systemctl status blackbox

- Check if the process is running or failing.

$ sudo systemctl status blackbox

```
● blackbox.service - Blackbox
     Loaded: loaded (/etc/systemd/system/blackbox.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2025-01-30 12:25:56 IST; 1 week 5 days ago
   Main PID: 141400 (blackbox_export)
      Tasks: 10 (limit: 9394)
     Memory: 16.4M
        CPU: 1h 45min 25.125s
     CGroup: /system.slice/blackbox.service
             └─141400 /usr/local/bin/blackbox_exporter --config.file=/etc/blackbox/blackbox.yml --web.listen-address=0.0.0.0:9115

Jan 30 12:25:56 grafana systemd[1]: Started Blackbox.
Jan 30 12:25:56 grafana blackbox_exporter[141400]: ts=2025-01-30T06:55:56.542Z caller=main.go:255 level=info msg="Starting blackbox_exporter" version="(version=0.21.0-rc.0, branch=HEAD, revision=6b655cdaf87
Jan 30 12:25:56 grafana blackbox_exporter[141400]: ts=2025-01-30T06:55:56.542Z caller=main.go:256 level=info build_context="(go=go1.18.1, user=root@8cc5ba78995d, date=20220510-12:08:07)"
Jan 30 12:25:56 grafana blackbox_exporter[141400]: ts=2025-01-30T06:55:56.543Z caller=main.go:268 level=info msg="Loaded config file"
Jan 30 12:25:56 grafana blackbox_exporter[141400]: ts=2025-01-30T06:55:56.543Z caller=main.go:416 level=info msg="Listening on address" address=0.0.0.0:9115
Jan 30 12:25:56 grafana blackbox_exporter[141400]: ts=2025-01-30T06:55:56.543Z caller=tls_config.go:195 level=info msg="TLS is disabled." http2=false
```

Black occupies port 9115 here. Check response http://I.P address:9115/metrics.

Blackbox Exporter

Probe prometheus.io for http_2xx

Debug probe prometheus.io for http_2xx

Metrics

Configuration

Recent Probes

Module	Target	Result	Debug
http_2xx	https://lyra-jenkins.toucanint.com/login?from=%2F	Success	Logs
http_2xx	https://acq-perf-jenkins.toucanint.com/login?from=%2F	Success	Logs
http_2xx	https://switch-dev-jenkins.toucanint.com/login?from=%2F	Success	Logs
http_2xx	https://nexus.toucanint.com/	Success	Logs
http_2xx	https://synopsys-jenkins.toucanint.com/login?from=%2F	Success	Logs
http_2xx	https://master-slave-jenkins.toucanint.com/login?from=%2F	Success	Logs
http_2xx	https://toucan.spiraservice.net/Login.aspx?ReturnUrl=%2f	Success	Logs
http_2xx	https://artifacts.toucanint.com/ui/login/	Success	Logs
http_2xx	https://acq-jenkins.toucanint.com/login?from=%2F	Success	Logs
http_2xx	https://psp-dev-jenkins.toucanint.com/login?from=%2F	Success	Logs
http_2xx	https://switch-qa-jenkins.toucanint.com/login?from=%2F	Success	Logs
http_2xx	https://grafana.toucanint.com/login	Success	Logs
http_2xx	https://lyra-jenkins.toucanint.com/login?from=%2F	Success	Logs
http_2xx	https://acq-perf-jenkins.toucanint.com/login?from=%2F	Success	Logs
http_2xx	https://switch-dev-jenkins.toucanint.com/login?from=%2F	Success	Logs
http_2xx	https://nexus.toucanint.com/	Success	Logs
http_2xx	https://synopsys-jenkins.toucanint.com/login?from=%2F	Success	Logs
http_2xx	https://master-slave-jenkins.toucanint.com/login?from=%2F	Success	Logs

Metrics Overview

1. `probe_success`

    a. Description: Indicates if the probe was successful
    b. Values: 0 (failure) or 1 (success)

2. `probe_duration_seconds`

    a. Description: Duration of the probe
    b. Labels: phase (resolve, connect, tls, processing)

3. `probe_http_status_code`

    Description: Response status code for HTTP probes

Prometheus Integration

```
- job_name: 'blackbox'
  metrics_path: /probe
  params:
    module: [http_2xx]  # Look for a HTTP 200 response.
  static_configs:
    - targets:
      - https://acq-jenkins.toucanint.com/login?from=%2F
      - https://psp-dev-jenkins.toucanint.com/login?from=%2F
      - https://master-slave-jenkins.toucanint.com/login?from=%2F
      - https://switch-dev-jenkins.toucanint.com/login?from=%2F
      - https://lyra-jenkins.toucanint.com/login?from=%2F
      - https://switch-qa-jenkins.toucanint.com/login?from=%2F
      - https://synopsys-jenkins.toucanint.com/login?from=%2F
      - https://acq-perf-jenkins.toucanint.com/login?from=%2F
      - https://nexus.toucanint.com/
      - https://artifacts.toucanint.com/ui/login/
      - https://toucan.spiraservice.net/Login.aspx?ReturnUrl=%2f
      - https://grafana.toucanint.com/login
  relabel_configs:
  - source_labels: [__address__]
    target_label: __param_target
  - source_labels: [__param_target]
    target_label: instance
  - target_label: __address__
    replacement: 192.168.61.186:9115  # The Blackbox Exporter's real hostname:port.
```

Step1 : **-** Follow the above steps from s**tep5 and step6 in the Kube state metrics** for the dashboard setup. Use **Grafana URL or ID (13922)** for the kubeadm certificate expire dashboard.

Step 2: - Final dashboard for the Blackbox exporter.

Job All Instance All

HTTP Probe Overview

Instance	Status	Code	SSL	TLS Version	SSL Cert Expiry (days)	Probe Duration (s)	DNS Lookup Duration (s)
https://acq-jenkins.toucanint.com/...	UP	200	OK	TLS 1.3	41	0.03	0.001
https://acq-perf-jenkins.toucanint....	UP	200	OK	TLS 1.3	41	0.02	0.001
https://artifacts.toucanint.com/ui/l...	UP	200	OK	TLS 1.3	41	0.02	0.001
https://grafana.toucanint.com/login	UP	200	OK	TLS 1.3	41	0.02	0.001
https://lyra-jenkins.toucanint.com/...	UP	200	OK	TLS 1.3	41	0.01	0.001
https://master-slave-jenkins.touca...	UP	200	OK	TLS 1.3	41	0.02	0.001
https://nexus.toucanint.com/	UP	200	OK	TLS 1.3	41	0.01	0.001
https://psp-dev-jenkins.toucanint....	UP	200	OK	TLS 1.3	41	0.02	0.001
https://switch-dev-jenkins.toucani...	UP	200	OK	TLS 1.3	41	0.02	0.001
https://switch-qa-jenkins.toucanin...	UP	200	OK	TLS 1.3	41	0.01	0.001
https://synopsys-jenkins.toucanin...	UP	200	OK	TLS 1.2	41	0.02	0.001
https://toucan.spiraservice.net/Lo...	UP	200	OK	TLS 1.2	219	0.19	0.021

Alert Rules for Kubernetes Clusters

<u>Podnotrunning</u>(Alert)

```
groups:
- name: pod.rules
  rules:
  - alert: PodNotRunning
    expr: |
      sum by (namespace, pod, job, instance) (
        kube_pod_status_phase{phase=~"Pending|Failed|Unknown"}
      ) > 0
    for: 2m
    labels:
      severity: warning
    annotations:
      summary: "Pod {{ $labels.pod }} (Job: {{ $labels.job }}) not running"
      description: "Pod {{ $labels.pod }} in namespace {{ $labels.namespace }} on instance {{ $labels.instance }} for job {{ $labels.job }} has been in non-running state for more than 2
minutes"
```

ImagePullBackOff(Alert)

```
groups:
- name: pod.rules
  rules:
  - alert: PodImagePullBackOff
    expr: |
      sum by (namespace, pod, container, job, instance) (
        kube_pod_container_status_waiting_reason{reason="ImagePullBackOff"}
      ) > 0
    for: 2m
    labels:
      severity: critical
    annotations:
      summary: "Container {{ $labels.container }} in Pod {{ $labels.pod }} has ImagePullBackOff (Job: {{ $labels.job }})"
      description: "Container {{ $labels.container }} in Pod {{ $labels.pod }} in namespace {{ $labels.namespace }} on instance {{ $labels.instance }} has been unable to pull its image
for more than 2 minutes"

  - alert: PodErrImagePull
    expr: |
      sum by (namespace, pod, container, job, instance) (
        kube_pod_container_status_waiting_reason{reason="ErrImagePull"}
      ) > 0
    for: 2m
    labels:
      severity: critical
    annotations:
      summary: "Container {{ $labels.container }} in Pod {{ $labels.pod }} has ErrImagePull (Job: {{ $labels.job }})"
      description: "Container {{ $labels.container }} in Pod {{ $labels.pod }} in namespace {{ $labels.namespace }} on instance {{ $labels.instance }} encountered error while pulling im
age"
```

PodCrashLoopBackOff(Alert)

```
groups:
- name: pod.rules
  rules:
  - alert: PodCrashLoopBackOff
    expr: |
      sum by (namespace, pod, job, instance) (
        kube_pod_container_status_waiting_reason{reason="CrashLoopBackOff"}
      ) > 0
    for: 2m
    labels:
      severity: critical
    annotations:
      summary: "Pod {{ $labels.pod }} (Job: {{ $labels.job }}) in CrashLoopBackOff"
      description: "Pod {{ $labels.pod }} in namespace {{ $labels.namespace }} on instance {{ $labels.instance }} for job {{ $labels.job }} has been in CrashLoopBackOff state for more t
han 2 minutes"
```

HTTPServiceDown(Alert)

```
groups:
- name: site_availability.rules
  rules:
  - alert: HTTPServiceDown
    expr: |
      (probe_http_status_code != 200 and probe_http_status_code > 0) and probe_success == 0
    for: 2m
    labels:
      severity: critical
    annotations:
      summary: "HTTP Service Down - {{ $labels.instance }} (Job: {{ $labels.job }})"
      description: "HTTP service at {{ $labels.instance }} is returning status code {{ $value }} for more than 2 minutes"
```

sitedown(Alert)

```
groups:
- name: site_availability.rules
  rules:
  - alert: SiteDown
    expr: |
      probe_success == 0 and up == 1
    for: 2m
    labels:
      severity: critical
    annotations:
      summary: "Site Down - {{ $labels.instance }} (Job: {{ $labels.job }})"
      description: "Site {{ $labels.instance }} has been down for more than 2 minutes"
```

Alertmanager

Alertmanager is a component of the Prometheus monitoring system responsible for managing alerts. It handles alerts sent by Prometheus notification integrations such as email.