

Deploy Prometheus and Grafana on Kubernetes using Helm

Gayatripawar · [Follow](#)

12 min read · Dec 19, 2023

[Listen](#)[Share](#)

In the dynamic world of Kubernetes, effective monitoring and visualization are crucial for maintaining the health and performance of your applications. In this blog, we'll explore the seamless deployment of Prometheus and Grafana on the minikube cluster using Helm.

Overview of Prometheus, Grafana, and Helm:

Prometheus: An open-source monitoring and alerting toolkit designed for Kubernetes, Prometheus excels in collecting and querying real-time metrics.

Grafana: A powerful analytics and monitoring platform that integrates seamlessly with Prometheus, offering visually stunning dashboards.

Helm: The Kubernetes package manager that simplifies the deployment and management of applications through charts.

Prerequisites:

1. Minikube

2. kubectl

3. Helm installed

Let's get Started...

Prometheus Installation

Step 1: Search for the Prometheus helm chart using the below command

```
helm search hub Prometheus
```

This command will list the following Prometheus helm chart

URL	CHART VERSION	APP VER
https://artifacthub.io/packages/helm/prometheus...	25.8.2	v2.48.1
https://artifacthub.io/packages/helm/truecharts...	13.1.0	2.48.1
https://artifacthub.io/packages/helm/prometheus...	13.0.0	2.22.1
https://artifacthub.io/packages/helm/saurabh6-p...	0.2.0	1.1
https://artifacthub.io/packages/helm/cloudposse...	0.2.1	
https://artifacthub.io/packages/helm/mach1el-ch...	1.0.1	v2.47.0
https://artifacthub.io/packages/helm/bitnami/pr...	0.5.1	2.48.1
https://artifacthub.io/packages/helm/wener/prom...	25.8.2	v2.48.1
https://artifacthub.io/packages/helm/stakater/p...	1.0.32	

Or else you can go to the [Artifact Hub](#) repository and search for the Prometheus helm chart.

The screenshot shows the Artifact Hub website with the search bar set to "Search packages". The results for "prometheus" are displayed, featuring the official Prometheus logo and a brief description: "Prometheus is a monitoring system and time series database." Below the description are four small icons: a blue square with a white flame, a blue square with a white gear, a blue square with a white checkmark, and a green square with a white star. To the right of these icons are three statistics: "SUBSCRIPTIONS: 130", "WEBHOOKS: 9", and "PRODUCTION USERS: 5". The main content area contains a section titled "Prometheus" with a brief overview of the project, its purpose, and how it's deployed using Helm. It also lists prerequisites: "Kubernetes 1.19+" and "Helm 3.7+".

Run the following commands to add the Prometheus helm chart

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-
helm repo update
```

Output:

```
helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "hashicorp" chart repository
...Successfully got an update from the "grafana" chart repository
...Successfully got an update from the "prometheus-community" chart repository
Update Complete. *Happy Helming!*
```

We have downloaded the latest version of the Prometheus.

Install Prometheus Helm Chart on Kubernetes Cluster

To install the Prometheus helm chart we need to run the “helm install” command as below shown

```
helm install prometheus prometheus-community/prometheus
```

Output:

```
helm install prometheus prometheus-community/prometheus
NAME: prometheus
LAST DEPLOYED: Tue Dec 19 11:04:13 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The Prometheus server can be accessed via port 80 on the following DNS name from
prometheus-server.default.svc.cluster.local
```

Get the Prometheus server URL by running these commands in the same shell:

```
export POD_NAME=$(kubectl get pods --namespace default -l "app.kubernetes.io/name=prometheus")
kubectl --namespace default port-forward $POD_NAME 9090
```

The Prometheus alertmanager can be accessed via port 9093 on the following DNS name from

Get the Alertmanager URL by running these commands in the same shell:

```
export POD_NAME=$(kubectl get pods --namespace default -l "app.kubernetes.io/name=alertmanager")
kubectl --namespace default port-forward $POD_NAME 9093
```

```
#####
##### WARNING: Pod Security Policy has been disabled by default since #####
##### it deprecated after k8s 1.25+. use #####
##### (index .Values "prometheus-node-exporter" "rbac" #####
##### "pspEnabled") with (index .Values #####
##### "prometheus-node-exporter" "rbac" "pspAnnotations") #####
##### in case you still need it. #####
#####
```

The Prometheus PushGateway can be accessed via port 9091 on the following DNS name from

Get the PushGateway URL by running these commands in the same shell:

```
export POD_NAME=$(kubectl get pods --namespace default -l "app=prometheus-pushgateway" --namespace default port-forward $POD_NAME 9091)
```

For more information on running Prometheus, visit:

<https://prometheus.io/>

We have successfully installed Prometheus on Kubernetes. Now to check the deployed Kubernetes resources by running the 'kubectl' command..

Output:

```
D:\k8s-lab>kubectl get all
NAME                                         READY   STATUS    RESTARTS   AGE
pod/prometheus-alertmanager-0                1/1     Running   0          3m29s
pod/prometheus-kube-state-metrics-6b464f5b88-9k5mp 1/1     Running   0          3m29s
pod/prometheus-prometheus-node-exporter-dhtgw 1/1     Running   0          3m29s
pod/prometheus-prometheus-pushgateway-7857c44f49-jpv6m 1/1     Running   0          3m29s
pod/prometheus-server-6b68fdb54b-1k4kl        2/2     Running   0          3m29s

NAME                           TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)      AGE
service/kubernetes            ClusterIP   10.96.0.1      <none>       443/TCP     24h
service/prometheus-alertmanager ClusterIP   10.103.177.50 <none>       9093/TCP    3m30s
service/prometheus-alertmanager-headless ClusterIP   None           <none>       9093/TCP    3m30s
service/prometheus-kube-state-metrics ClusterIP   10.100.246.82 <none>       8080/TCP    3m30s
service/prometheus-prometheus-node-exporter ClusterIP   10.106.54.11 <none>       9100/TCP    3m30s
service/prometheus-prometheus-pushgateway ClusterIP   10.102.227.21 <none>       9091/TCP    3m30s
service/prometheus-server        ClusterIP   10.105.79.116  <none>       80/TCP      3m30s

NAME              DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
daemonset.apps/prometheus-prometheus-node-exporter  1         1         1         1             1           kubernetes.io/os=linux   3m29s

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/prometheus-kube-state-metrics  1/1     1           1           3m29s
deployment.apps/prometheus-prometheus-pushgateway 1/1     1           1           3m29s
deployment.apps/prometheus-server               1/1     1           1           3m29s

NAME                           DESIRED   CURRENT   READY   AGE
replicaset.apps/prometheus-kube-state-metrics-6b464f5b88  1         1         1         3m29s
replicaset.apps/prometheus-prometheus-pushgateway-7857c44f49 1         1         1         3m29s
replicaset.apps/prometheus-server-6b68fdb54b            1         1         1         3m29s

NAME   READY   AGE
statefulset.apps/prometheus-alertmanager  1/1   3m29s
```

When you install the Prometheus Helm chart, it creates several Kubernetes resources to set up the Prometheus monitoring system. Here's a brief list of the key resources that are typically created:

ConfigMaps:

Prometheus-server: Contains the main Prometheus configuration.

Prometheus-rule files: Stores Prometheus alerting and recording rules.

Secrets:

Prometheus-server-TLS: Contains TLS certificates for secure communication.

ServiceAccounts:

Prometheus-server: Defines the service account used by the Prometheus server

components.

ClusterRole and ClusterRoleBinding:

Prometheus-server: Grants necessary permissions to the Prometheus server components.

StatefulSet:

Prometheus-server: Manages the stateful deployment of Prometheus server pods.

Service:

Prometheus-server: Exposes the Prometheus server within the cluster.

The next step is to access and launch the Prometheus Kubernetes application. You'll access the application using the Kubernetes services for Prometheus. To get all the Kubernetes Services for Prometheus, run this command:

```
kubectl get service
```

Output:

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP
kubernetes	ClusterIP	10.96.0.1	<none>
prometheus-alertmanager	ClusterIP	10.103.177.50	<none>
prometheus-alertmanager-headless	ClusterIP	None	<none>
prometheus-kube-state-metrics	ClusterIP	10.100.246.82	<none>
prometheus-prometheus-node-exporter	ClusterIP	10.106.54.11	<none>
prometheus-prometheus-pushgateway	ClusterIP	10.102.227.21	<none>
prometheus-server	ClusterIP	10.105.79.116	<none>

- **Prometheus-alert manager(ClusterIP):** Alertmanager is a component of Prometheus that manages and handles alerts. This service provides the ClusterIP for communication within the cluster on port 9093.
- **Prometheus-alert manager-headless(ClusterIP):** This is a headless service for Alertmanager, meaning it does not provide a ClusterIP. It is used for discovery

purposes, typically when other services need to discover the IP addresses of Alertmanager instances.

- **Prometheus-kube-state-metrics (ClusterIP):** Kube-state-metrics is an add-on service for Prometheus that exposes Kubernetes resource metrics. This service provides a ClusterIP for communication within the cluster on port 8080.
- **Prometheus-prometheus-node-exporter (ClusterIP):** Node Exporter is a Prometheus exporter that collects system-level metrics from nodes in the cluster. This service provides a ClusterIP for communication within the cluster on port 9100.
- **Prometheus-prometheus-pushgateway (ClusterIP):** Pushgateway allows ephemeral and batch jobs to expose their metrics to Prometheus. This service provides a ClusterIP for communication within the cluster on port 9091.
- **Prometheus-server (ClusterIP):** Prometheus server is the core component that collects, stores, and queries metrics. This service provides a ClusterIP for communication within the cluster.

Exposing the Prometheus-server service on Kubernetes

To expose the Kubernetes Prometheus-server service, run this command

```
kubectl expose service prometheus-server --type=NodePort --target-port=9090 --r
```

Output:

```
D:\k8s-lab>kubectl expose service prometheus-server --type=NodePort --target-port=9090 --name=prometheus-server-ext  
service/prometheus-server-ext exposed
```

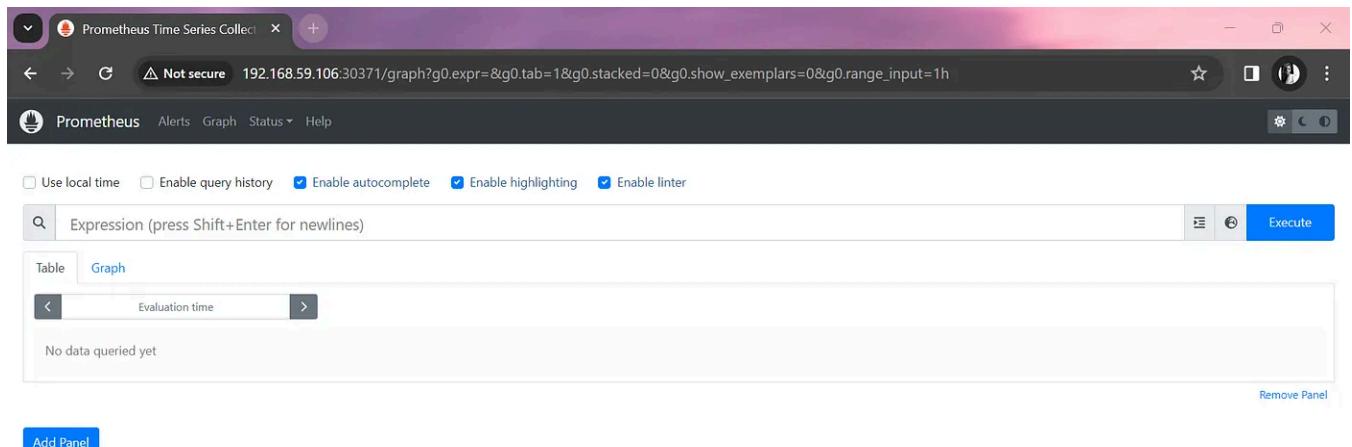
If you need to access Prometheus from outside the Kubernetes cluster, exposing it as a NodePort allows you to do so. You can use the node's IP and the allocated NodePort to reach Prometheus. If you have external monitoring or visualization tools that need to connect to Prometheus, exposing it as a NodePort facilitates this external integration.

After making the Prometheus-server Kubernetes service accessible, let's reach the Prometheus application by employing the provided command.

```
minikube service prometheus-server-ext
```

NAMESPACE	NAME	TARGET PORT	URL
default	prometheus-server-ext	80	http://192.168.59.106:30371

Opening service default/prometheus-server-ext in default browser...



Our Prometheus web UI is available now. With the installation of Prometheus on Kubernetes via Helm, the Prometheus instance is now operational within the cluster, and we can reach it by navigating to a browser or using a specific URL.

Grafana Installation

To begin, let's initiate the Grafana installation. Subsequently, we'll establish integration between Prometheus and Grafana, with Grafana configured to utilize Prometheus as its primary data source. Lastly, leveraging Grafana, we'll craft insightful dashboards to monitor and observe the Kubernetes cluster.

Now search for the Grafana helm chart by running the below command

```
helm search hub grafana
```

Output:

```
helm search hub grafana
URL                                              CHART VERSION
https://artifacthub.io/packages/helm/grafana/gr... 7.0.19
https://artifacthub.io/packages/helm/surajwarbh... 0.1.0
https://artifacthub.io/packages/helm/saurabh6-g... 0.2.0
https://artifacthub.io/packages/helm/romanow-he... 1.5.0
https://artifacthub.io/packages/helm/kube-ops/g... 1.0.2
https://artifacthub.io/packages/helm/flagger/gr... 1.7.0
https://artifacthub.io/packages/helm/stakater/g... 1.0.39
```

An alternative approach is to navigate to the [Artifact Hub](#) repository and explore the official Grafana Helm Chart.

The screenshot shows the Artifact Hub interface. On the left, there is a sidebar with filters for 'Official', 'Verified publishers', and 'CNCF'. Below that are sections for 'KIND' (Helm charts, KCL modules, Keptn integrations, OLM operators) and 'CATEGORY' (AI / Machine learning, Integration and delivery, Monitoring and logging, Storage). The main area displays search results for 'grafana'. The first result is a chart by 'Grafana' (version 7.0.19), described as 'The leading tool for querying and visualizing time series and metrics.' It has a 'Monitoring and logging' badge. The second result is a chart by 'Bitnami' (version 9.6.5), described as 'Grafana is an open source metric analytics and visualization suite for visualizing time series data that supports various types of data sources.' It also has a 'Monitoring and logging' badge. The top navigation bar includes links for 'DOCS', 'STATS', 'SIGN UP', 'SIGN IN', and a settings gear icon. The search bar at the top is empty.

To get the Grafana Helm chart, run this command

```
helm repo add grafana https://grafana.github.io/helm-charts
helm repo update
```

Output:

```
D:\k8s-lab>helm repo add grafana https://grafana.github.io/helm-charts
"grafana" has been added to your repositories

D:\k8s-lab>helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "hashicorp" chart repository
...Successfully got an update from the "grafana" chart repository
...Successfully got an update from the "prometheus-community" chart repository
Update Complete. *Happy Helming!*
```

As you can see the Grafana repo is added successfully. Now we need to install the Grafana.

Install Grafana Helm Chart on Kubernetes Cluster

For installing the Grafana on Kubernetes, Use “helm install” command

```
helm install grafana grafana/grafana
```

Output:

```
helm install grafana grafana/grafana
NAME: grafana
LAST DEPLOYED: Tue Dec 19 12:36:38 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
1. Get your 'admin' user password by running:
```

```
kubectl get secret --namespace default grafana -o jsonpath="{.data.admin-pas
```

- The Grafana server can be accessed via port 80 on the following DNS name `grafana.default.svc.cluster.local`

`Get the Grafana URL to visit by running these commands in the same shell:`

```
export POD_NAME=$(kubectl get pods --namespace default -l "app.kubernetes."
kubectl --namespace default port-forward $POD_NAME 3000
```

3. Login with the password from step 1 and the username: admin

Having successfully installed Grafana on the Kubernetes Cluster, the Grafana server is now accessible through port 80. To retrieve the complete list of Kubernetes Services associated with Grafana, execute the following command:

```
kubectl get service
```

Output:

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP
grafana	ClusterIP	10.104.22.18	<none>

Exposing the Grafana Kubernetes Service

To expose the Grafana-service on Kubernetes we need to run this command

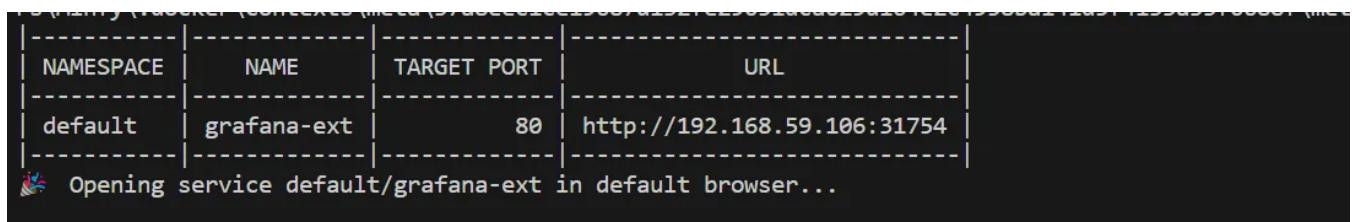
```
kubectl expose service grafana --type=NodePort --target-port=3000 --name=grafana-ext
```

```
D:\k8s-lab>kubectl expose service grafana --type=NodePort --target-port=3000 --name=grafana-ext
service/grafana-ext exposed
```

By executing this command, we transition the service type from ClusterIP to NodePort, enabling external access to Grafana beyond the Kubernetes Cluster. The service will now be reachable on port 3000.

This below command will generate the following URL to access the Grafana dashboard

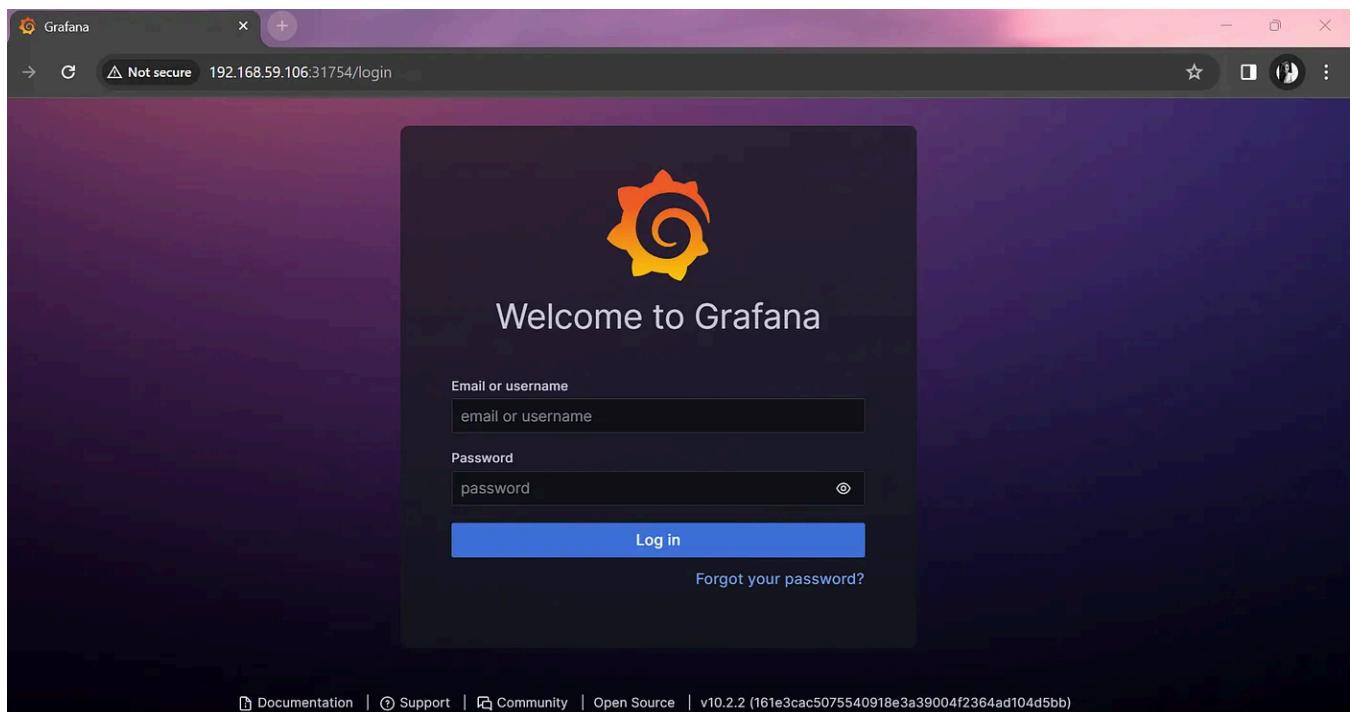
```
minikube service grafana-ext
```



NAMESPACE	NAME	TARGET PORT	URL
default	grafana-ext	80	http://192.168.59.106:31754

Opening service default/grafana-ext in default browser...

Grafana UI:



The image above shows the Grafana Login page. To get the password for `admin`, run this command on a new terminal.

```
kubectl get secret --namespace default grafana -o jsonpath=".data.admin-password" | base64 --decode | tr -d '\n'
```

NOTE: You need to open a new terminal to run this process to leave the Grafana tunnel running

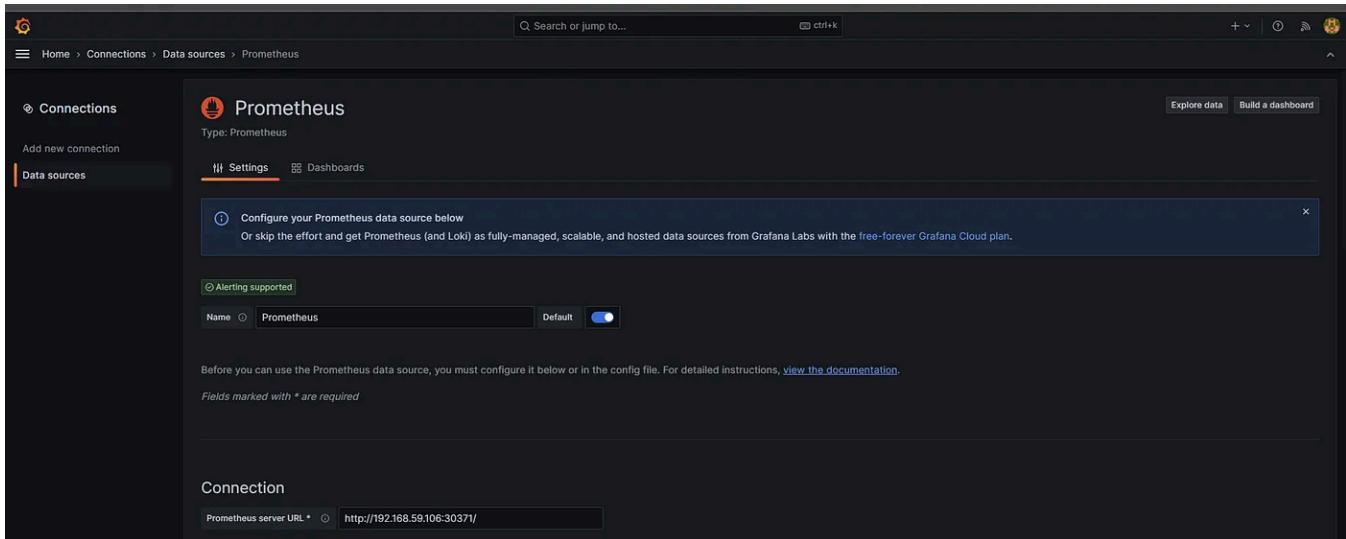
Login to Grafana

Now let's add the data source as Prometheus. To add Prometheus as the data source, follow these steps:

- On the Welcome to Grafana Home page, click Add your first data source :
- Select Prometheus as the data source:

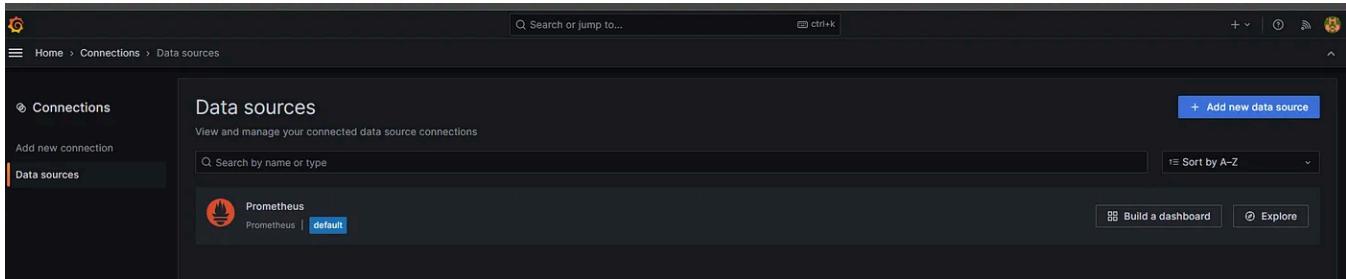
Include the internal cluster URL of your Prometheus application by referring to the initial URL displayed when executing the 'minikube service prometheus-server-ext'

command earlier.



Click on ‘Save and Next’.

You have successfully added the Prometheus as Data source.



Grafana Dashboard

In this section, our focus will be on importing a Grafana Dashboard to streamline the process.

To import a Grafana Dashboard, follow these steps:

- Get the Grafana Dashboard ID from the [Grafana public Dashboard library](#)

The screenshot shows the Grafana Labs Dashboards page. At the top, there's a banner with the text "From heatmaps to histograms, graphs to geomaps: fast and furious visualizations any way you want. Check out new visualizations in Grafana 10!" Below the banner are four featured dashboard cards:

- Monitor your Kubernetes deployment**
- Visualize your MongoDB data**
- Visualize your Jira data**
- InfluxDB dashboards for telegraf metrics**

Below these cards are filtering options: "Filter by:" dropdowns for Category (set to All), Panel (set to All), and Data Source (set to All). There's also a search bar labeled "Search dashboards" and a count of 7266 results.

- Now go to the search dashboard, and search for Kubernetes :

The screenshot shows the search results for "Kubernetes" on the Grafana Dashboards page. The results are displayed in a grid format:

Dashboard Title	Rating	Downloads	Source
K8s / Storage / Volumes / Namespace	No ratings	3.91M downloads	Prometheus
K8s / Storage / Volumes / Cluster	4/5 2 ratings	3.73M downloads	Prometheus
1 K8S for Prometheus Dashboard 20211010 EN	No ratings	3.30M downloads	Prometheus
Fluentd 1.x	5/5 2 ratings	3.15M downloads	Prometheus
Kubernetes cluster monitoring (via Prometheus)	4.36/5 11 ratings	2.05M downloads	Prometheus
cert-manager	No ratings	1.80M downloads	Prometheus
Cluster Autoscaler Stats	No ratings	N/A	kubernetes
Kubernetes Cluster (Prometheus)	3.67/5 3 ratings	N/A	Prometheus
Kubernetes Nodes	5/5 1 rating	N/A	Prometheus

A red arrow points to the "Kubernetes cluster monitoring (via Prometheus)" card.

- Select Dashboard and copy the Dashboard ID:

[← All dashboards](#)

Kubernetes cluster monitoring (via Prometheus)

Monitors Kubernetes cluster using Prometheus. Shows overall cluster CPU / Memory / Filesystem usage as well as individual pod, containers, systemd services statistics. Uses cAdvisor metrics only.

[Overview](#) [Revisions](#) [Reviews](#)

Node CPU usage

Used: 50% Total: 6.00%

Network IO pressure

Cluster memory usage

Used: 1.00 GB Total: 2.00 GB

Full K8 solution with dashboards, alerts and KPIs available in Grafana Cloud free tier (with 10K free Prometheus series metrics and more): <https://grafana.com/signup/cloud/connect-account>

release v3 license MIT kubernetes ^1.4.0 prometheus ^1.3.0 grafana ^3.1.1 grafana.net link

Initial idea was taken from [this dashboard](#) and improved to exclude `node-exporter` dependency and to give

[Sign up for Grafana Cloud](#) (?)

Easily monitor your deployment of Kubernetes with Grafana Cloud's out-of-the-box monitoring solution.

Free tier includes 10k metrics, 50GB logs, 50GB traces, and more.

[Create free account →](#)

Get this dashboard

Data source: Prometheus 1.3.0

Dependencies: grafana 3.1.1 Graph (old) Singlestat

Import the dashboard template:

[Copy ID to clipboard](#) or [Download JSON](#)

Docs: Importing dashboards

- Go back to the Grafana home, and go to the dashboard on left corner

Grafana

Not secure 192.168.59.106:31754/?orgId=1

Home

- Home
- Starred
- Dashboards
- Explore** (New) Import
- Alerting
- Connections
- Administration

Search or jump to...

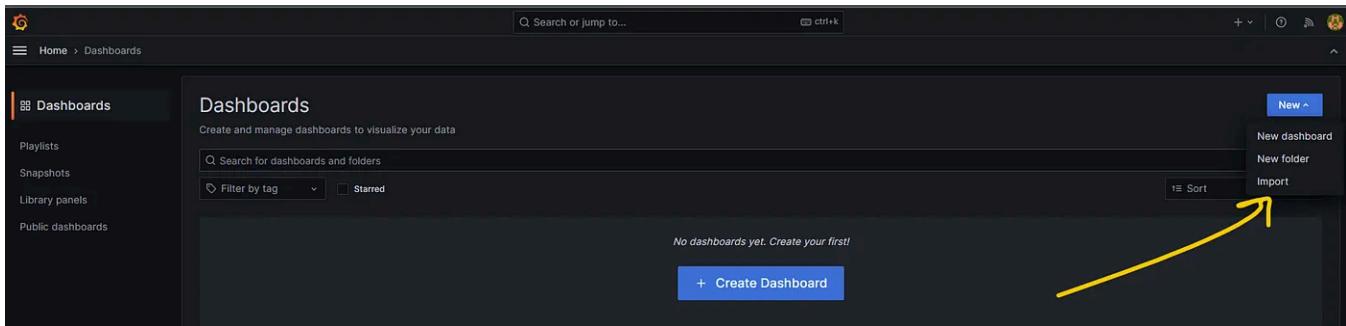
TUTORIAL

[DATA SOURCE AND DASHBOARDS](#)

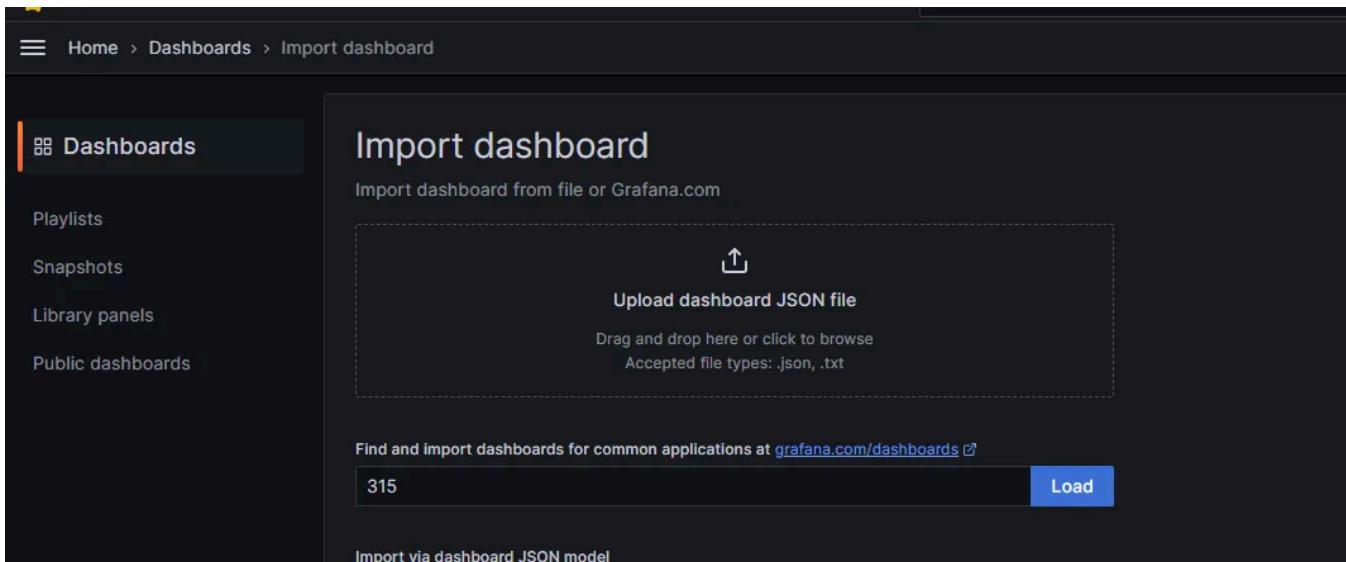
Grafana fundamentals

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.

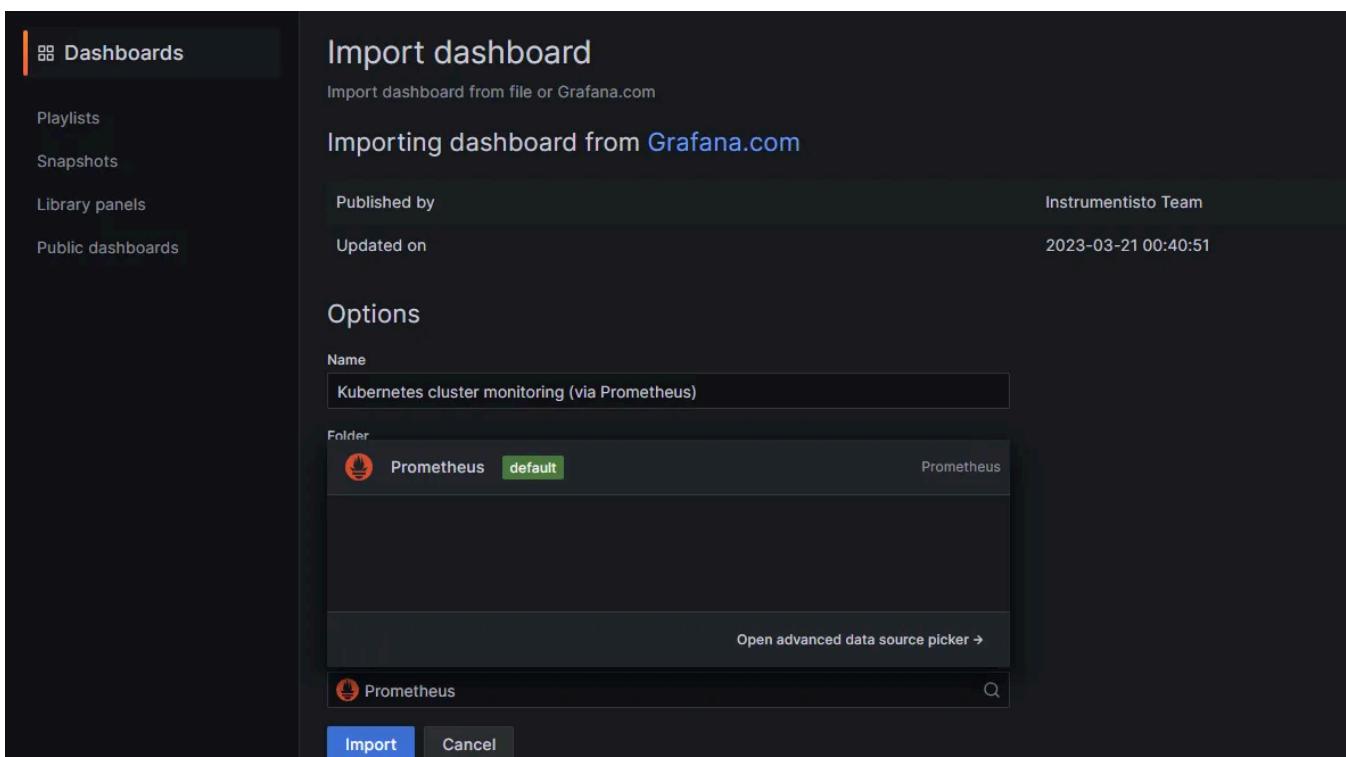
- Now click on the new->import



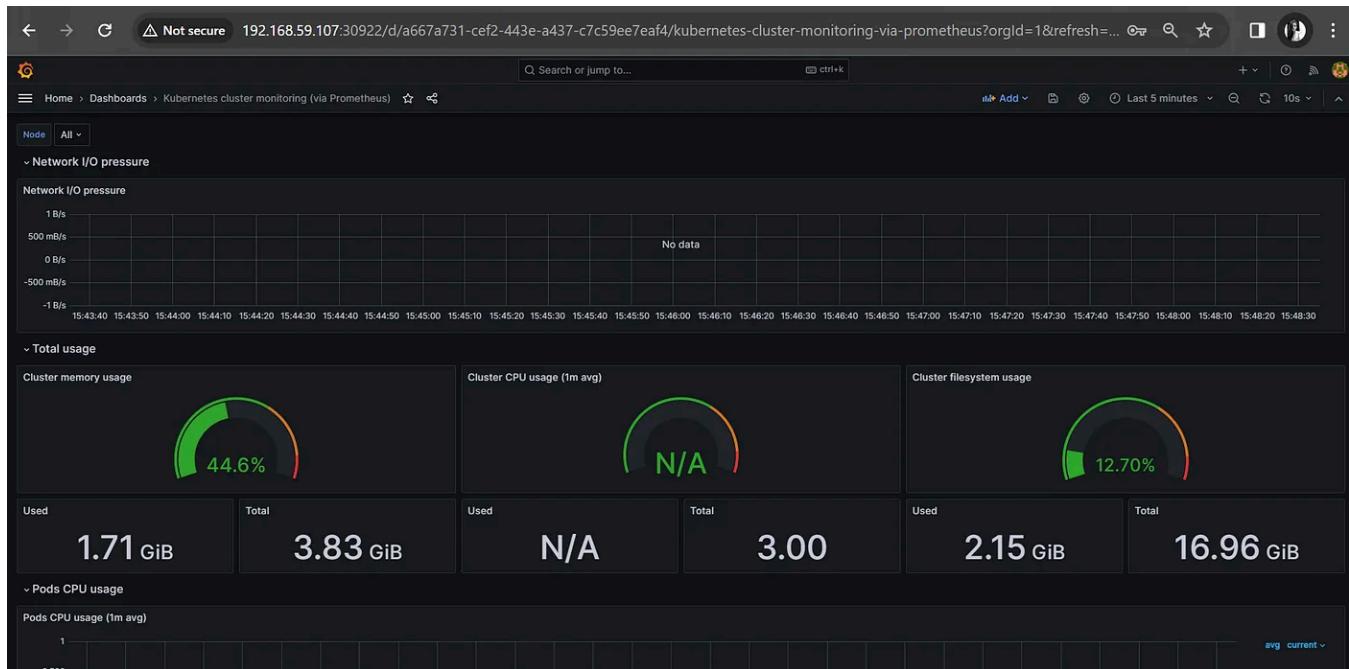
- Add the Grafana Id, and click on 'Load'



- Select a Prometheus Data Source and Click Import



- It will launch the Dashboard shown below:



You use this dashboard to monitor and observe the Kubernetes cluster metrics. It displays the following Kubernetes cluster metrics:

- Network I/O pressure.
- Cluster CPU usage.
- Cluster Memory usage.
- Cluster filesystem usage.
- Pods CPU usage.

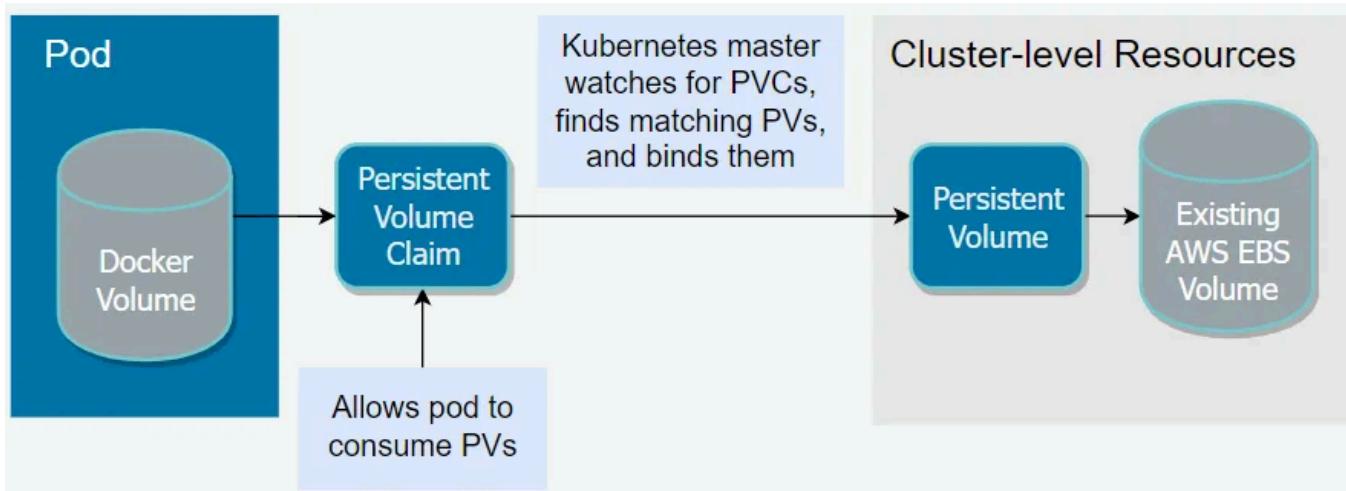
Implementing Persistent Volume and Persistent Volume claim for Prometheus and Grafana

In this section will see what is PV and PVC

what is PV?

A *persistent volume*(PV) is a piece of storage in the cluster that has been provisioned by an administrator or dynamically provisioned using Storage Classes. It is a resource in the cluster just like a node is a cluster resource. PVs are volume plugins like Volumes but have a lifecycle independent of any individual Pod that uses the PV.

This API object captures the details of the implementation of the storage, be that NFS, iSCSI, or a cloud-provider-specific storage system.



what is PVC?

A *PersistentVolumeClaim* (PVC) is a request for storage by a user. It is similar to a Pod. Pods consume node resources and PVCs consume PV resources. Pods can request specific levels of resources (CPU and Memory). Claims can request specific size and access modes (e.g., they can be mounted ReadWriteOnce, ReadOnlyMany, ReadWriteMany, or ReadWriteOncePod, see [Access mode](#)).

1. Create Persistent Volume for Prometheus

```

apiVersion: v1
kind: PersistentVolume
metadata:
  name: prometheus-pv
spec:
  capacity:
    storage: 5Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: local-storage
  local:
    path: /prometheus-data
  nodeAffinity:
    required:
      nodeSelectorTerms:
        - matchExpressions:
          - key: kubernetes.io/hostname
  
```

```
operator: In
values:
- <NODE_PROMETHEUS_RUNS>
```

Now Create a .yaml file by adding the above code and apply using kubectl

```
kubectl apply -f pv-prom.yaml
persistentvolume/prometheus-pv created
```

2. Create a Persistent Volume Claim for Prometheus

Create another .yaml file for PVC and apply it using kubectl

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: prometheus-pvc
spec:
  storageClassName: local-storage
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
```

```
kubectl apply -f pvc-prom.yaml
persistentvolumeclaim/prometheus-pvc created
```

3. Now Create a PV for Grafana

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: grafana-pv
spec:
  capacity:
```

```

storage: 5Gi
volumeMode: Filesystem
accessModes:
- ReadWriteOnce
persistentVolumeReclaimPolicy: Retain
storageClassName: local-storage
local:
path: /grafana-data

nodeAffinity:
required:
nodeSelectorTerms:
- matchExpressions:
- key: kubernetes.io/hostname
operator: In
values:
- <NODE_GRAFANA_RUNS>

```

```
kubectl apply -f pv.yaml
persistentvolume/grafana-pv created
```

4. Create a PVC for Grafana

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: grafana-pvc
spec:
storageClassName: local-storage
accessModes:
- ReadWriteOnce
resources:
requests:
storage: 5Gi

```

```
kubectl apply -f pvc.yaml
persistentvolumeclaim/grafana-pvc created
```

- Update Prometheus to use the persistent storage:

```
helm upgrade prometheus prometheus-community/prometheus --set server.persistent
```

- Update Grafana to use the persistent storage:

```
helm upgrade grafana grafana/grafana --set persistence.enabled=true,persistence
```

Conclusion

In this blog we saw how to integrate the Prometheus and Grafana on Kubernetes using helm. Prometheus, with its ability to collect, store, and query metrics efficiently, becomes the vigilant guardian of your applications and infrastructure. Also saw how to create a sample Grafana dashboard for Kubernetes that collects metrics of the cluster. And we have Created the Persistent volumes and claims for both Prometheus and Grafana.

Monitoring is crucial in DevOps. It keeps track of the Kubernetes cluster and microservices performance. The real-time insights it offers enable SREs to identify bottlenecks, troubleshoot issues, and proactively address potential pitfalls before they impact end-users.

----- * *** * *** *

Thanks !!

[Follow](#)

Written by **Gayatripawar**

72 Followers · 4 Following

Responses (4)



What are your thoughts?

[Respond](#)

MOHIT MISHRA

about 1 year ago

...

Hi Gayatri, this is your first article and you have written so wonderfully !! Kudos to you.

As an extension to this article you can write 2 more article if you can :

1. The dashboard dont show you details of each specific pods in your cluster. For.....

[Read More](#)

 2  1 reply

[Reply](#)

 Akshaynagdive
about 2 months ago

...

Informative article

 1

[Reply](#)

 Patrick
9 months ago

...

Hi Gayatri. Very nice article. I was going to pass on it until I seen the 12min read time. Most of the articles I see here are copy a pasta and have little to no solid information in them. Eg. The quality is extremely low, and duplicated. Yours on.....

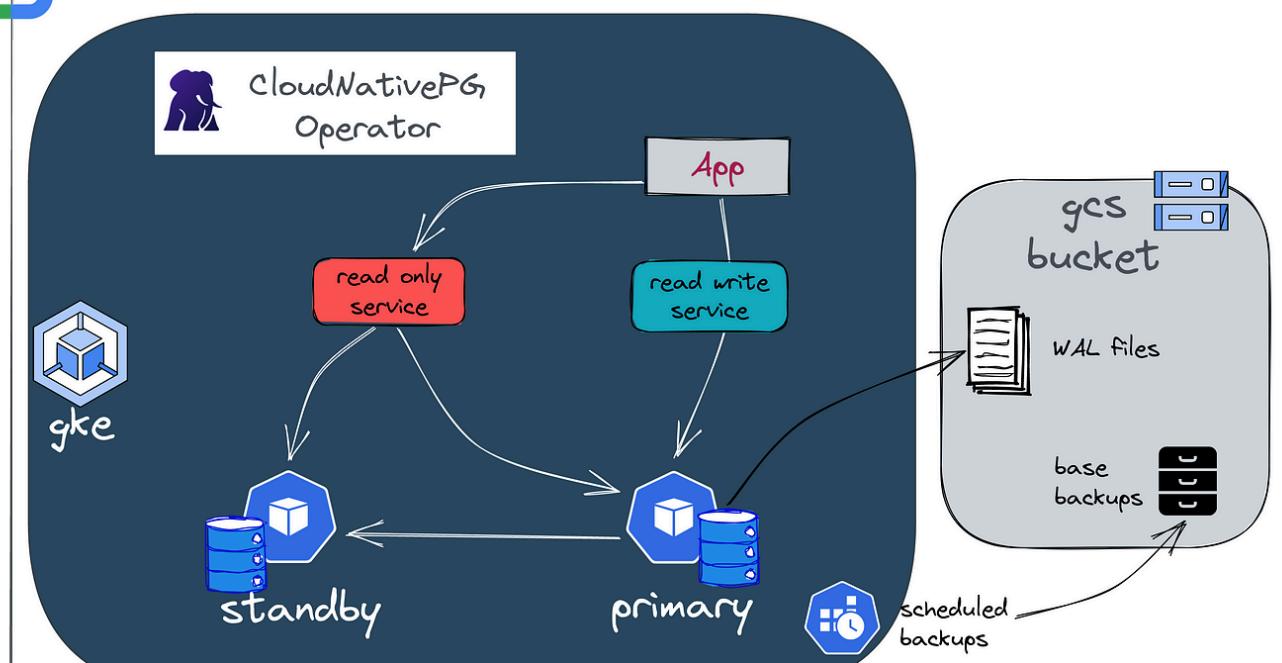
[Read More](#)

 2  1 reply

[Reply](#)

[See all responses](#)

Recommended from Medium



Mr.PlanB

Local Path Provider or Rook-Ceph? Storage Solutions for Cloud Native PG in Kubernetes

When setting up a Kubernetes (K8S) environment for production workloads, choosing the right storage solution is critical, particularly for...

Nov 2, 2024 6



1:01:02.123456789Z {app="nginx",cluster="us-west1"} G

nd precision

Prometheus-style Labels
key-value pairs

indexed

Mano karan

Kubernetes Observability using the Grafana Loki, Promtail and Grafana Dashboard using Helm in the...

Introduction: Hi, If you are a container enthusiastic you will be in need of the observability to troubleshoot the container. Here I am...

Aug 24, 2024  2



Lists



Staff picks

797 stories · 1562 saves



Stories to Help You Level-Up at Work

19 stories · 915 saves



Self-Improvement 101

20 stories · 3207 saves



Productivity 101

20 stories · 2712 saves

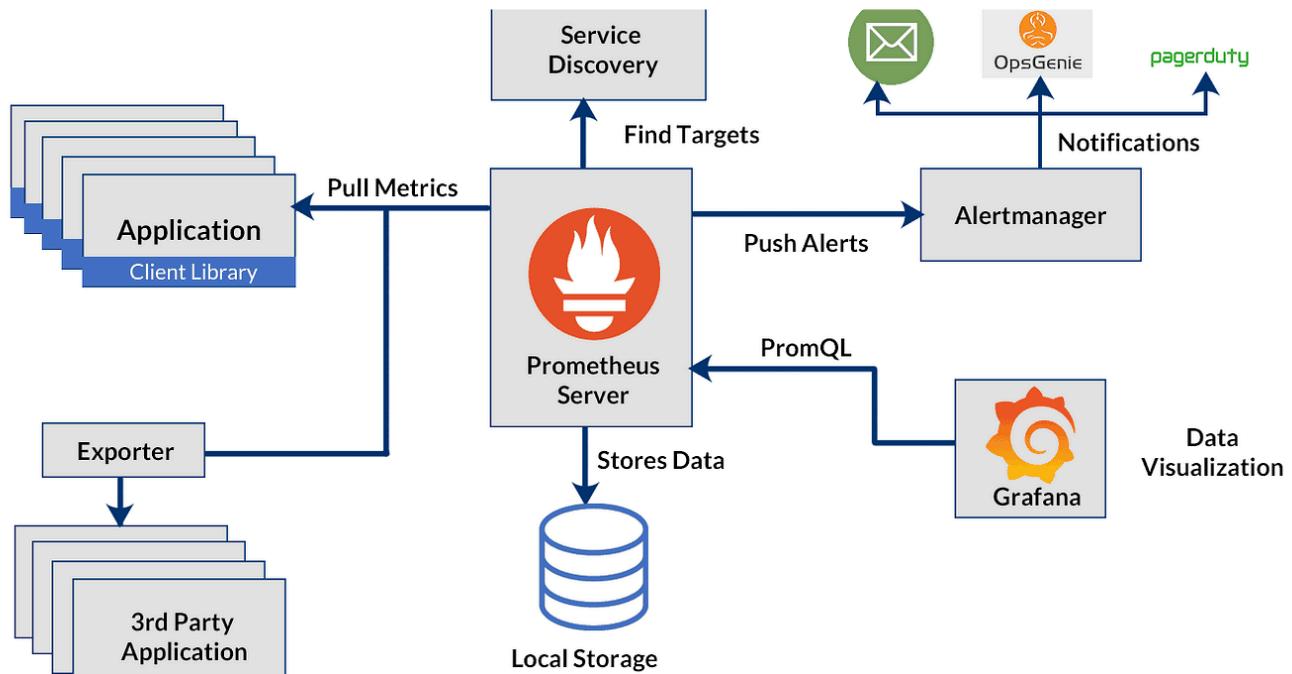


 In Spacelift by Sumeet Ninawe

Configuring Prometheus with Helm Chart on Kubernetes

Originally published at <https://spacelift.io>.

◆ Sep 16, 2024  2



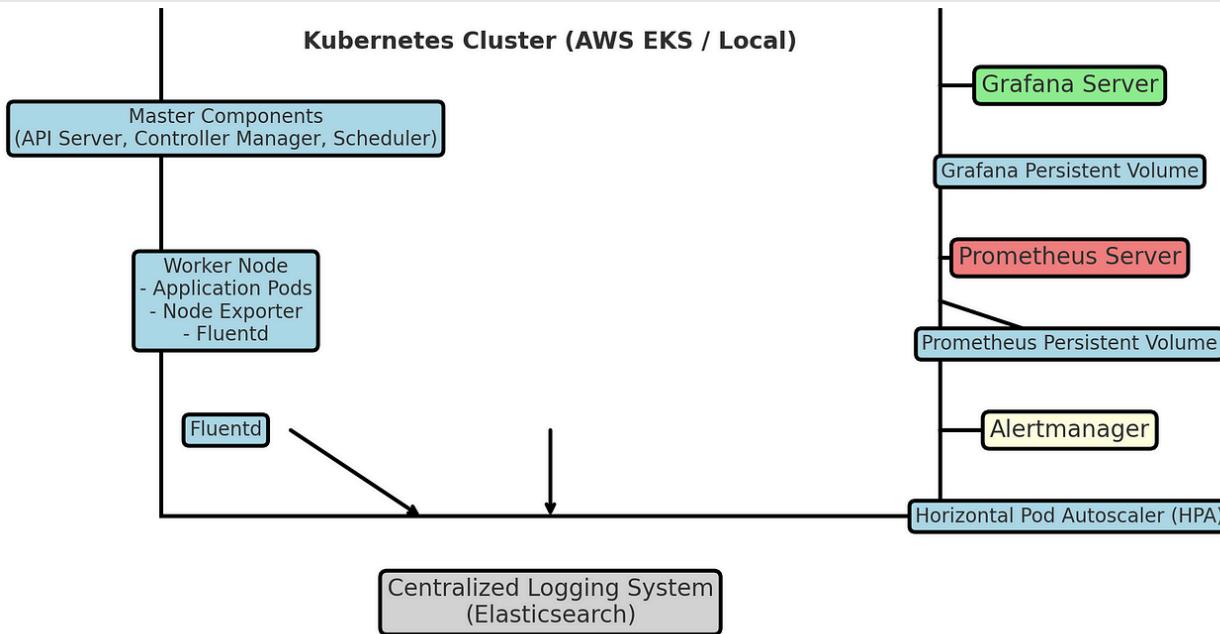
 Platform Engineers

Setting Up a Prometheus and Grafana Monitoring Stack from Scratch

This guide will walk you through the process of setting up a monitoring stack using Prometheus and Grafana from scratch. This setup will be...

Oct 8, 2024

8



Ram Vadranam

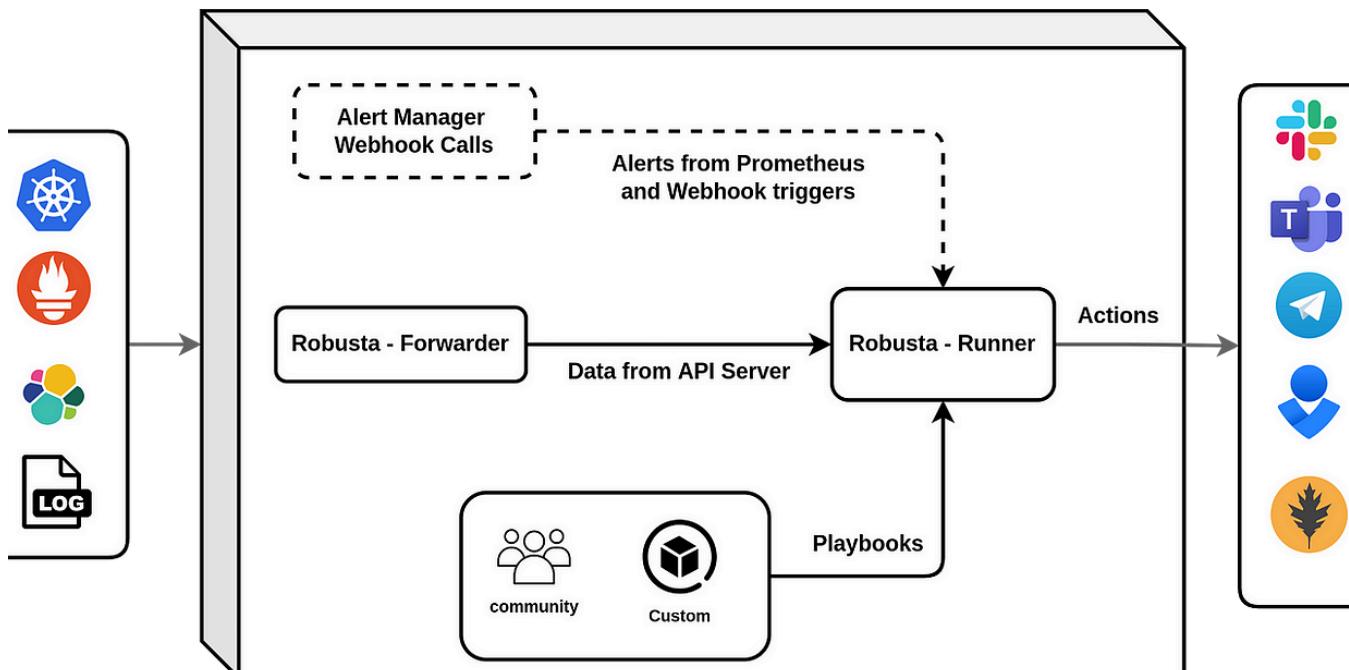
Comprehensive Guide to Monitoring AWS EKS and Local Kubernetes Clusters with Prometheus and Grafana

Optimize Performance and Visibility Across AWS EKS and Local Kubernetes Clusters Using Prometheus and Grafana

Aug 13, 2024

63

1



In MyCloudSeries by Ewere Diagboya

Single-Deployment Observability on Kubernetes: Introducing Robusta for Kubernetes Monitoring

Single Deployment Observability for your Kubernetes cluster

◆ Jul 14, 2024 🙌 16



See more recommendations