How to Integrate Prometheus and Grafana on Kubernetes with Helm

Paul Odhiambo

Prometheus is an open-source automated monitoring and alerting system. It has become a widely accepted tool for monitoring highly dynamic container environments such as Kubernetes and Docker Swarm. It can also monitor traditional non-container infrastructures such as monitoring Linux/windows servers, single applications, Apache Server, and Database services.

When running Prometheus to monitor Kubernetes, it scrapes metrics data from the Kubernetes Cluster for monitoring the system. These metrics are CPU status, memory/disk space storage, request duration, number of running processes, server resources, and node status.

Grafana is a multi-platform that gets data from a data source such as Prometheus and transforms it into visualizations charts. This article will teach you how to integrate Prometheus and Grafana on Kubernetes using Helm.

Let’s start by discussing how Prometheus and Grafana work.

How Promethetheus works

When you install Prometheus in Kubernetes, it sets up Prometheus Server and Alert Manager.

The Prometheus Server

This is the core of Prometheus. It does the actual monitoring work and has three components:

Data Retrieval Worker.

Time Series Database.

HTTP Server.

Data Retrieval Worker

This component scrapes and pulls the metrics data from the application. It then transforms the metrics data into time series data.

Time Series Database

This component stores the metric data in time series format.

HTTP Server

This component accepts queries for the stored time series data and displays the data in a web UI or dashboard. It can use the inbuilt/default Prometheus Web UI or a third-party platform.

We will use Grafana as the third-party platform for displaying the time series data. We will talk about Grafana later.

Alert Manager

It provides automated alerting through email or slack to the system administrator. It enables the administrator to know any changes in the system that Prometheus is monitoring in real-time.

Why Prometheus Monitoring is Important in DevOps

Prometheus is essential in monitoring highly dynamic container development environments and microservices infrastructure. Modern DevOps is becoming more and more complex. DevOps engineers are building complex infrastructures that are becoming difficult to handle/maintain and monitor manually. It, therefore, needs more automation such as Prometheus monitoring.

When you use Prometheus to monitor a Kubernetes cluster, it automatically manages all the microservices. It can then easily detect any failures in the processes running the microservices. It also provides quicker debugging of the Kubernetes Cluster to identify errors and failures.

When Prometheus detects an error in one of the microservices, it uses the Alert Manager component to notify the system administrator. The system administrator will then take the necessary measures to resolve the error.

Let’s summarize the importance of Prometheus as follows:

Constantly monitoring the microservices running in the Kubernetes Cluster.

Automatically alert the administrator when a microservice crashes.

It reports on the cluster’s health and performance.

Display the metric data on the Prometheus Web UI or a third-party platform.

How Grafana Works

Grafana is an open-source multi-platform that gets data from a data source such as Prometheus and transforms it into visualizations charts. Grafana can create interactive charts, graphs, dashboards, and web UIs. You can see all the content on a web page once you connect your data source.

Grafana cannot create these visualization graphs, charts, and dashboards without a data source. It has to connect to a data source such as Prometheus.

Prometheus will send the metrics data it collects from the Kubernetes Cluster, and Grafana uses it to create the visualizations. It allows developers to create alerts, run queries, create visualizations, and understand their metrics no matter the data source.

How to Integrate Prometheus and Grafana on Kubernetes

The are two ways of integrating Prometheus and Grafana on Kubernetes:

1. Manually Deploy all Deployments, Services, ConfigMaps, and Secrets configurations files.

These configuration files are in YAML format. Kubernetes will read these files and deploy the application with all the Kubernetes resources.

This method of manually creating these files is efficient when you are dealing with a simple application. Prometheus and Grafana are complex applications that may take time to configure all the files. This is why most developers prefer the second method of using Helm. It makes their work easier.

2. Using Helm

Helm is popularly known as the package manager for Kubernetes. Helm helps to configure and install any application to Kubernetes. It bundles a Kubernetes application into a package known as a Helm Chart.

Getting Started with Helm and Charts

Helm makes it easy to deploy applications to Kubernetes. It is a collection of YAML files that defines all the Kubernetes resources such as Kubernetes Deployments, Services, ConfigMaps, and Secrets for the Kubernetes application. Helm interacts and communicates with Kubernetes to deploy the Helm Charts to the Kubernetes Cluster.

Helm Charts will help us to define, install, upgrade and integrate even the most complex application on Kubernetes using a few Helm commands. You can easily download and use existing Helm Charts.

We will use Helm to download the Helm Charts for Prometheus and Grafana. We will then deploy the downloaded Helm Charts to Kubernetes. It will save the time of building the entire application from scratch.

The Helm Community creates and maintains the Helm charts for various applications. Helm charts are reusable, easy to install, and up to date. Helm charts are distributed in public and private repositories such as ArtifactHub.

Installing Helm

Helm runs on a Kubernetes Cluster. Before you start installing Helm, ensure you have a Kubernetes cluster that is running. You will install Helm on that Kubernetes Cluster.

In this article, we will install Helm on Minikube. Minikube is a local Kubernetes Cluster that allows the Kubernetes Engine to run our machine using the memory and CPU of the computer. To install Helm on Minikube, follow these steps:

Step One: Tools

You must have the following tool already set up in your machine:

Docker Engine: This is the virtualization platform that will enable us to run Minikube. Minikube requires Minikube to run locally on our computer.

Step Two: Start Minikube

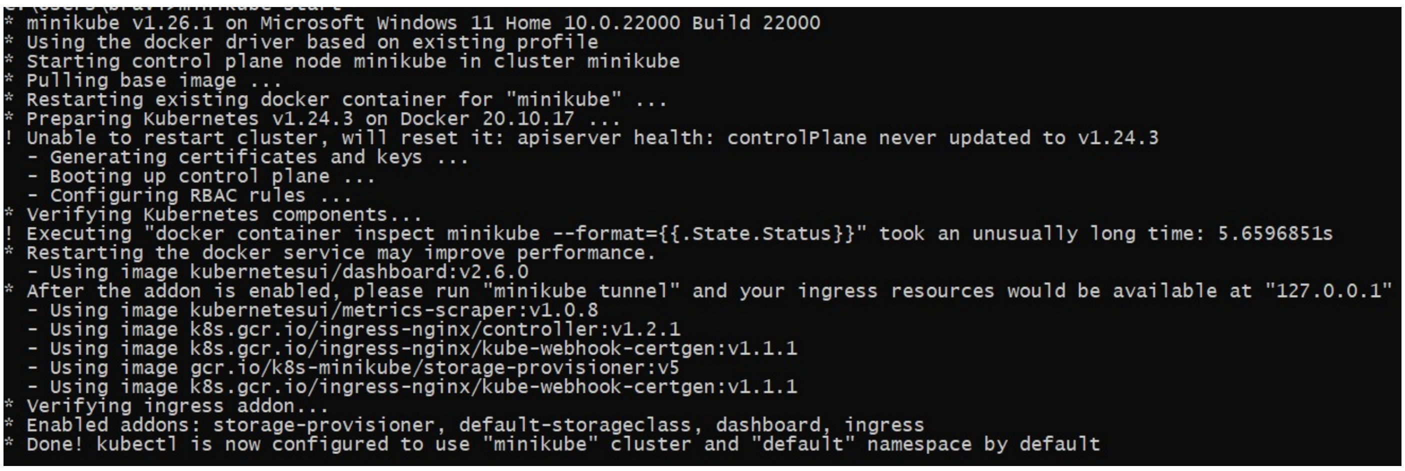
Minikube comes together with the Docker Engine. After installing Docker, run this command to start the Minikube Kubernetes Cluster.

minikube start --driver=docker

minikube start --driver=docker

minikube start --driver=docker

The command will start your cluster as shown below:



```
* minikube v1.26.1 on Microsoft Windows 11 Home 10.0.22000 Build 22000
* Using the docker driver based on existing profile
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Restarting existing docker container for "minikube" ...
* Preparing Kubernetes v1.24.3 on Docker 20.10.17 ...
! Unable to restart cluster, will reset it: apiserver health: controlPlane never updated to v1.24.3
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Verifying Kubernetes components...
! Executing "docker container inspect minikube --format={{.State.Status}}" took an unusually long time: 5.6596851s
* Restarting the docker service may improve performance.
  - Using image kubernetesui/dashboard:v2.6.0
* After the addon is enabled, please run "minikube tunnel" and your ingress resources would be available at "127.0.0.1"
  - Using image kubernetesui/metrics-scraper:v1.0.8
  - Using image k8s.gcr.io/ingress-nginx/controller:v1.2.1
  - Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.1.1
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
  - Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.1.1
* Verifying ingress addon...
* Enabled addons: storage-provisioner, default-storageclass, dashboard, ingress
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

Step Three: Installing Helm

Now that our cluster is running, I will show you how to install Helm.

Installing Helm on Windows

On the Windows terminal, run this command:

choco install kubernetes-helm

choco install kubernetes-helm

choco install kubernetes-helm

We will use the Chocolatey package manager for windows applications to install Helm.

Installing Helm on Linux

On the Linux terminal, run this command:

sudo apt-get install helm

sudo apt-get install helm

sudo apt-get install helm

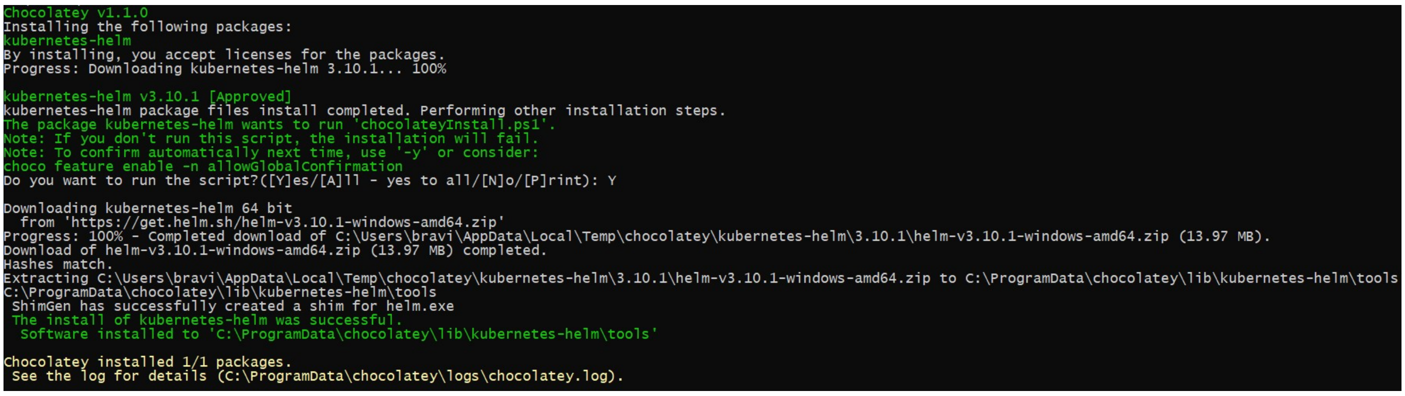
We will use the Apt package manager for Ubuntu applications to install Helm.

Installing Helm on macOS

On the macOS terminal, run this command:

brew install helm

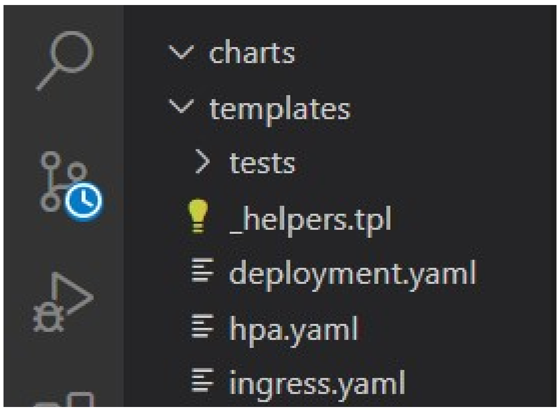
We will use the Homebrew package manager for macOS applications to install Helm. I am running Windows. The Helm installation process produces the following output:



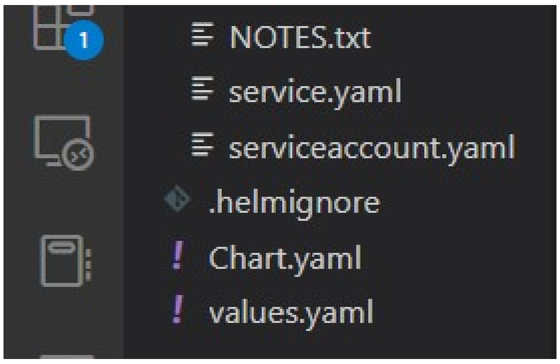
```
Chocolatey v1.1.0
Installing the following packages:
kubernetes-helm
By installing, you accept licenses for the packages.
Progress: Downloading kubernetes-helm 3.10.1... 100%

kubernetes-helm v3.10.1 [Approved]
kubernetes-helm package files install completed. Performing other installation steps.
The package kubernetes-helm wants to run 'chocolateyInstall.ps1'.
Note: If you don't run this script, the installation will fail.
Note: To confirm automatically next time, use '-y' or consider:
choco feature enable -n allowGlobalConfirmation
Do you want to run the script?([Y]es/[A]ll - yes to all/[N]o/[P]rint): Y

Downloading kubernetes-helm 64 bit
  from 'https://get.helm.sh/helm-v3.10.1-windows-amd64.zip'
Progress: 100% - Completed download of C:\Users\bravi\AppData\Local\Temp\chocolatey\kubernetes-helm\3.10.1\helm-v3.10.1-windows-amd64.zip (13.97 MB).
Download of helm-v3.10.1-windows-amd64.zip (13.97 MB) completed.
Hashes match.
Extracting C:\Users\bravi\AppData\Local\Temp\chocolatey\kubernetes-helm\3.10.1\helm-v3.10.1-windows-amd64.zip to C:\ProgramData\chocolatey\lib\kubernetes-helm\tools
C:\ProgramData\chocolatey\lib\kubernetes-helm\tools
 ShimGen has successfully created a shim for helm.exe
 The install of kubernetes-helm was successful.
  Software installed to 'C:\ProgramData\chocolatey\lib\kubernetes-helm\tools'

Chocolatey installed 1/1 packages.
 See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).
```

Helm Chart Structure

The general structure for a Helm Chart is shown in the image below:



```
v charts
v templates
  > tests
  _helpers.tpl
  deployment.yaml
  hpa.yaml
  ingress.yaml
```


NOTES.txt
service.yaml
serviceaccount.yaml
.helmignore
Chart.yaml
values.yaml

A Helm Chart consists of a Chart folder, Template folder, and chart.yaml file and Values.yaml file.

Chart Folder

It contains the chart dependencies for the downloaded Helm Chart.

Chart.yaml

This file contains the meta-information about the downloaded Helm Chart such as name, description version number, and list of dependencies.

Template Folder

This folder contains all the configuration YAML files you are deploying to the Kubernetes Cluster with the Helm Chart. These files are the Deployments, Services, ConfigMaps, and Secrets configurations YAML files.

Values.yaml

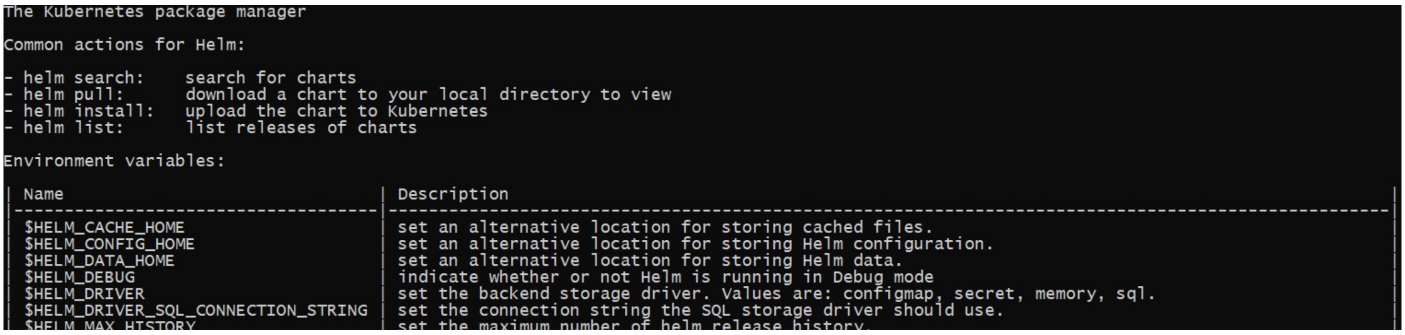
This file is where all the values for the configurations of YAML files are configured and set. The configuration YAML files read their values from this file.

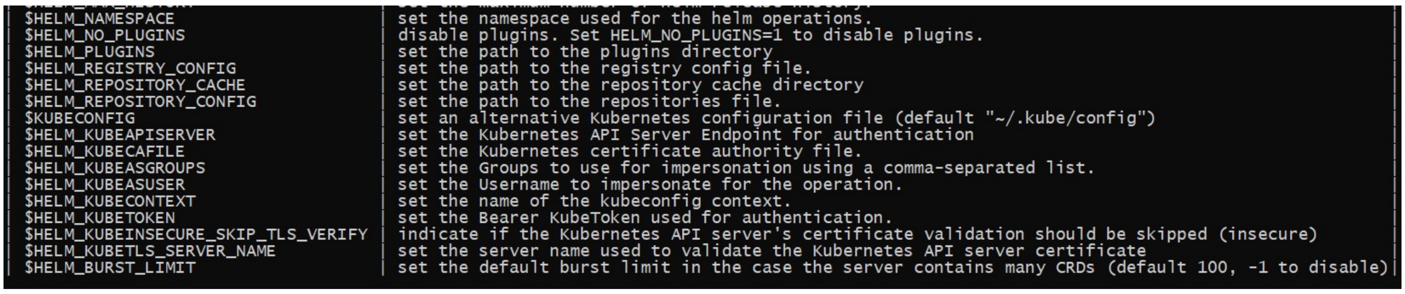
Helm Commands

To check the most common Helm commands, run this command in your terminal:

helm

It displays the following information:


```
The Kubernetes package manager

Common actions for Helm:

- helm search:    search for charts
- helm pull:      download a chart to your local directory to view
- helm install:   upload the chart to Kubernetes
- helm list:      list releases of charts

Environment variables:

| Name                               | Description                                                                            |
|------------------------------------|----------------------------------------------------------------------------------------|
| $HELM_CACHE_HOME                   | set an alternative location for storing cached files.                                  |
| $HELM_CONFIG_HOME                  | set an alternative location for storing Helm configuration.                            |
| $HELM_DATA_HOME                    | set an alternative location for storing Helm data.                                     |
| $HELM_DEBUG                        | indicate whether or not Helm is running in Debug mode                                  |
| $HELM_DRIVER                       | set the backend storage driver. Values are: configmap, secret, memory, sql.            |
| $HELM_DRIVER_SQL_CONNECTION_STRING | set the connection string the SQL storage driver should use.                           |
| $HELM_MAX_HISTORY                  | set the maximum number of helm release history.                                        |
```



```
| $HELM_NAMESPACE                    | set the namespace used for the helm operations.                                       |
| $HELM_NO_PLUGINS                   | disable plugins. Set HELM_NO_PLUGINS=1 to disable plugins.                            |
| $HELM_PLUGINS                      | set the path to the plugins directory                                                 |
| $HELM_REGISTRY_CONFIG              | set the path to the registry config file.                                             |
| $HELM_REPOSITORY_CACHE             | set the path to the repository cache directory                                        |
| $HELM_REPOSITORY_CONFIG            | set the path to the repositories file.                                                |
| $KUBECONFIG                        | set an alternative Kubernetes configuration file (default "~/.kube/config")           |
| $HELM_KUBEAPISERVER                | set the Kubernetes API Server Endpoint for authentication                             |
| $HELM_KUBECAFILE                   | set the Kubernetes certificate authority file.                                        |
| $HELM_KUBEASGROUPS                 | set the Groups to use for impersonation using a comma-separated list.                 |
| $HELM_KUBEASUSER                   | set the Username to impersonate for the operation.                                    |
| $HELM_KUBECONTEXT                  | set the name of the kubeconfig context.                                               |
| $HELM_KUBETOKEN                    | set the Bearer KubeToken used for authentication.                                     |
| $HELM_KUBEINSECURE_SKIP_TLS_VERIFY | indicate if the Kubernetes API server's certificate validation should be skipped (insecure) |
| $HELM_KUBETLS_SERVER_NAME          | set the server name used to validate the Kubernetes API server certificate            |
| $HELM_BURST_LIMIT                  | set the default burst limit in the case the server contains many CRDs (default 100, -1 to disable)|
```

These are the four Helm Commands:

helm search: We will use this command to search for our Prometheus and Grafana Helm Charts.

helm pull: It will pull and download the Prometheus and Grafana Helm Charts from ArtifactHub public repository.

helm install: We will use this command to upload or deploy the Prometheus and Grafana Helm Charts to Minikube Kubernetes Cluster.

helm list: We will use this command to list the installed Helm Charts in Kubernetes.

Let's start by searching for Prometheus and Grafana Helm Charts.

Searching for Prometheus Helm Chart

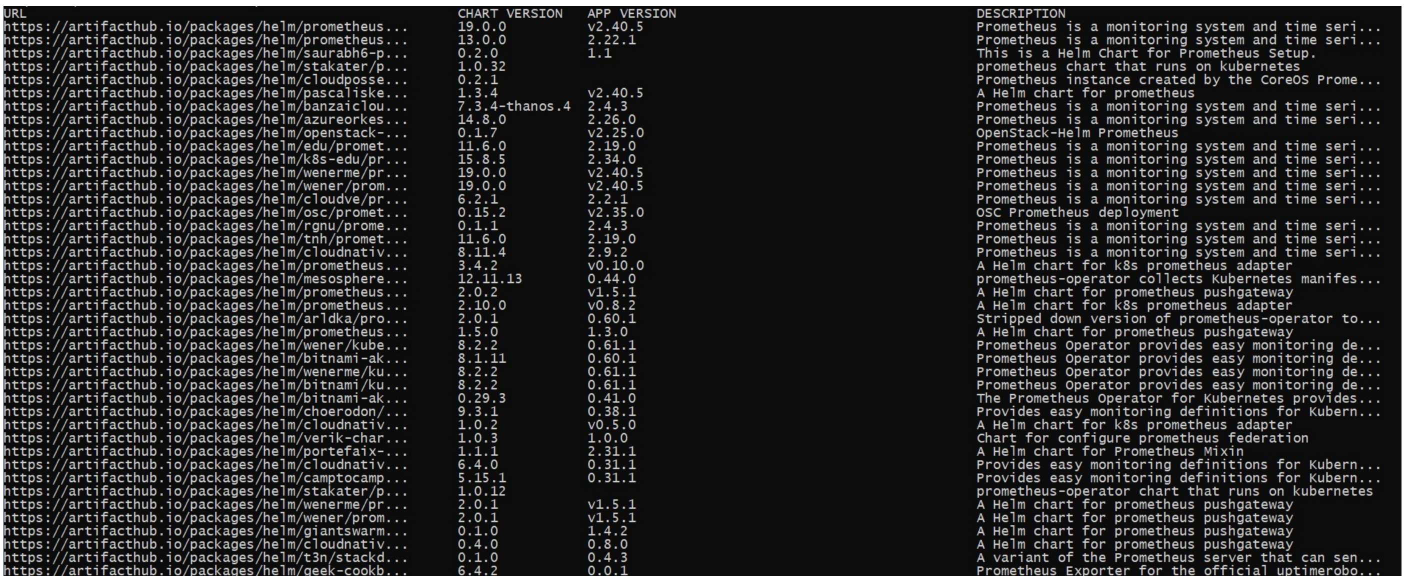
You can search Prometheus Helm from the command line as follows:

helm search hub prometheus

helm search hub prometheus

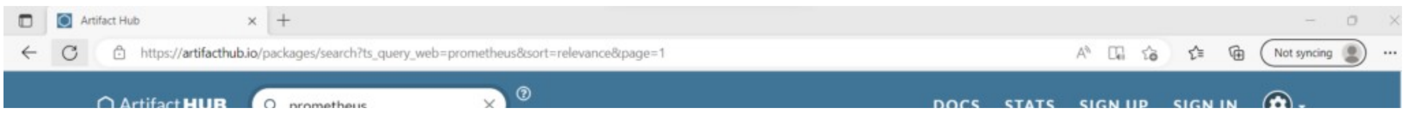
helm search hub prometheus

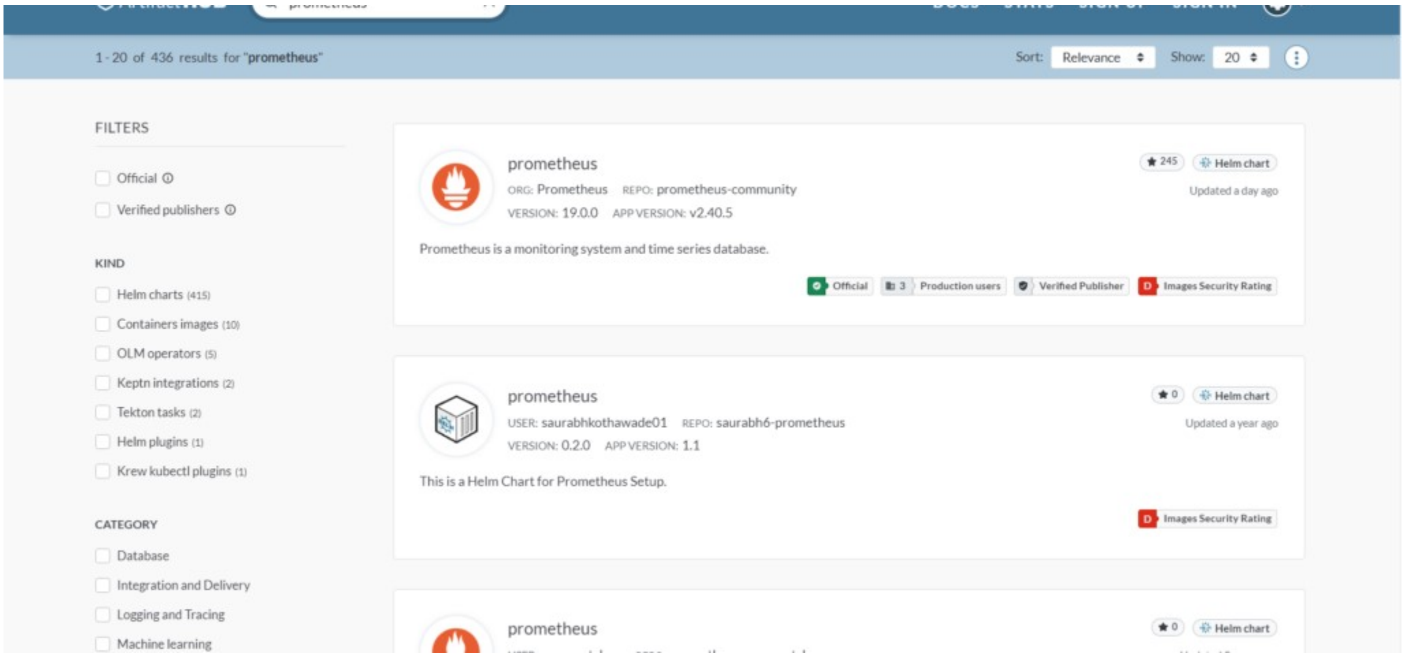
The command displays the following Helm Charts:



```
URL                                                     CHART VERSION   APP VERSION                                     DESCRIPTION
https://artifacthub.io/packages/helm/prometheus...      19.0.0          v2.40.5                                         Prometheus is a monitoring system and time seri...
https://artifacthub.io/packages/helm/prometheus...      13.0.0          2.22.1                                          Prometheus is a monitoring system and time seri...
https://artifacthub.io/packages/helm/saurabh6-p...      0.2.0           1.1                                             This is a Helm Chart for Prometheus Setup.
https://artifacthub.io/packages/helm/stakater/p...      1.0.32                                                          prometheus chart that runs on kubernetes
https://artifacthub.io/packages/helm/cloudposse...      0.2.1                                                           Prometheus instance created by the CoreOS Prome...
https://artifacthub.io/packages/helm/pascaliske...      1.3.4           v2.40.5                                         A Helm chart for prometheus
https://artifacthub.io/packages/helm/banzaiclou...      7.3.4-thanos.4  2.4.3                                           Prometheus is a monitoring system and time seri...
https://artifacthub.io/packages/helm/azureorkes...      14.8.0          2.26.0                                          Prometheus is a monitoring system and time seri...
https://artifacthub.io/packages/helm/openstack-...      0.1.7           v2.25.0                                         OpenStack-Helm Prometheus
https://artifacthub.io/packages/helm/edu/promet...      11.6.0          2.19.0                                          Prometheus is a monitoring system and time seri...
https://artifacthub.io/packages/helm/k8s-edu/pr...      15.8.5          2.34.0                                          Prometheus is a monitoring system and time seri...
https://artifacthub.io/packages/helm/wenerme/pr...      19.0.0          v2.40.5                                         Prometheus is a monitoring system and time seri...
https://artifacthub.io/packages/helm/wener/prom...      19.0.0          v2.40.5                                         Prometheus is a monitoring system and time seri...
https://artifacthub.io/packages/helm/cloudve/pr...      6.2.1           2.2.1                                           Prometheus is a monitoring system and time seri...
https://artifacthub.io/packages/helm/osc/promet...      0.15.2          v2.35.0                                         OSC Prometheus deployment
https://artifacthub.io/packages/helm/rgnu/prome...      0.1.1           2.4.3                                           Prometheus is a monitoring system and time seri...
https://artifacthub.io/packages/helm/tnh/promet...      11.6.0          2.19.0                                          Prometheus is a monitoring system and time seri...
https://artifacthub.io/packages/helm/cloudnativ...      8.11.4          2.9.2                                           Prometheus is a monitoring system and time seri...
https://artifacthub.io/packages/helm/prometheus...      3.4.2           v0.10.0                                         A Helm chart for k8s prometheus adapter
https://artifacthub.io/packages/helm/mesosphere...      12.11.13        0.44.0                                          prometheus-operator collects Kubernetes manifes...
https://artifacthub.io/packages/helm/prometheus...      2.0.2           v1.5.1                                          A Helm chart for prometheus pushgateway
https://artifacthub.io/packages/helm/prometheus...      2.10.0          v0.8.2                                          A Helm chart for k8s prometheus adapter
https://artifacthub.io/packages/helm/arldka/pro...      2.0.1           0.60.1                                          Stripped down version of prometheus-operator to...
https://artifacthub.io/packages/helm/prometheus...      1.5.0           1.3.0                                           A Helm chart for prometheus pushgateway
https://artifacthub.io/packages/helm/wener/kube...      8.2.2           0.61.1                                          Prometheus Operator provides easy monitoring de...
https://artifacthub.io/packages/helm/bitnami-ak...      8.1.11          0.60.1                                          Prometheus Operator provides easy monitoring de...
https://artifacthub.io/packages/helm/wenerme/ku...      8.2.2           0.61.1                                          Prometheus Operator provides easy monitoring de...
https://artifacthub.io/packages/helm/bitnami/ku...      8.2.2           0.61.1                                          Prometheus Operator provides easy monitoring de...
https://artifacthub.io/packages/helm/bitnami-ak...      0.29.3          0.41.0                                          The Prometheus Operator for Kubernetes provides...
https://artifacthub.io/packages/helm/choerodon/...      9.3.1           0.38.1                                          Provides easy monitoring definitions for Kubern...
https://artifacthub.io/packages/helm/cloudnativ...      1.0.2           v0.5.0                                          A Helm chart for k8s prometheus adapter
https://artifacthub.io/packages/helm/verik-char...      1.0.3           1.0.0                                           Chart for configure prometheus federation
https://artifacthub.io/packages/helm/portefaix-...      1.1.1           2.31.1                                          A Helm chart for Prometheus Mixin
https://artifacthub.io/packages/helm/cloudnativ...      6.4.0           0.31.1                                          Provides easy monitoring definitions for Kubern...
https://artifacthub.io/packages/helm/camptocamp...      5.15.1          0.31.1                                          Provides easy monitoring definitions for Kubern...
https://artifacthub.io/packages/helm/stakater/p...      1.0.12                                                          prometheus-operator chart that runs on kubernetes
https://artifacthub.io/packages/helm/wenerme/pr...      2.0.1           v1.5.1                                          A Helm chart for prometheus pushgateway
https://artifacthub.io/packages/helm/wener/prom...      2.0.1           v1.5.1                                          A Helm chart for prometheus pushgateway
https://artifacthub.io/packages/helm/giantswarm...      0.1.0           1.4.2                                           A Helm chart for prometheus pushgateway
https://artifacthub.io/packages/helm/cloudnativ...      0.4.0           0.8.0                                           A Helm chart for prometheus pushgateway
https://artifacthub.io/packages/helm/t3n/stackd...      0.1.0           0.4.3                                           A variant of the Prometheus server that can sen...
https://artifacthub.io/packages/helm/geek-cookb...      6.4.2           0.0.1                                           Prometheus Exporter for the official uptimerobo...
```

You can also search the Prometheus Helm Chart from the ArtifactHub UI as shown below:


Artifact Hub
https://artifacthub.io/packages/search?ts_query_web=prometheus&sort=relevance&page=1
Not syncing
ArtifactHUB
prometheus
DOCS
STATS
SIGN UP
SIGN IN


1 - 20 of 436 results for "prometheus"

Sort: Relevance Show: 20

FILTERS
- Official
- Verified publishers

KIND
- Helm charts (415)
- Containers images (10)
- OLM operators (5)
- Keptn integrations (2)
- Tekton tasks (2)
- Helm plugins (1)
- Krew kubectl plugins (1)

CATEGORY
- Database
- Integration and Delivery
- Logging and Tracing
- Machine learning

prometheus
ORG: Prometheus REPO: prometheus-community
VERSION: 19.0.0 APP VERSION: v2.40.5
★ 245 Helm chart
Updated a day ago
Prometheus is a monitoring system and time series database.
Official | 3 Production users | Verified Publisher | D Images Security Rating

prometheus
USER: saurabhkothawade01 REPO: saurabh6-prometheus
VERSION: 0.2.0 APP VERSION: 1.1
★ 0 Helm chart
Updated a year ago
This is a Helm Chart for Prometheus Setup.
D Images Security Rating

prometheus
★ 0 Helm chart

We will use the official Prometheus Helm Chart from Prometheus. Let's add the 'prometheus-community' repository for our Prometheus Helm Chart as follows:

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

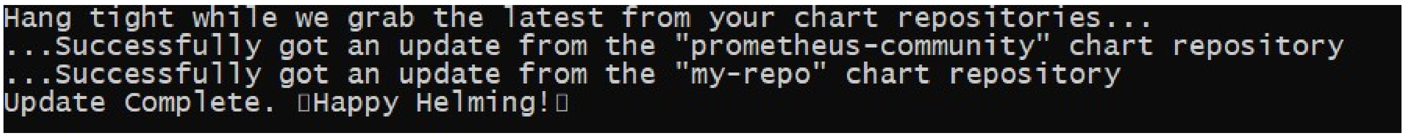
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts

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

To update the added repository, run this command:

helm repo update

It gives the following output:


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

Installing the Prometheus Helm Chart

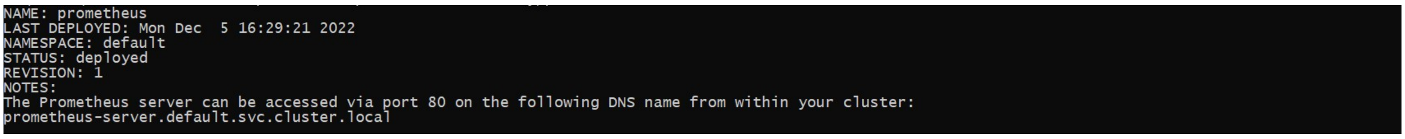
Installing the Prometheus Helm Chart will upload or deploy the Prometheus Helm Chart to Minikube Kubernetes Cluster. To install the Prometheus Helm Chart, use this 'helm install' command:

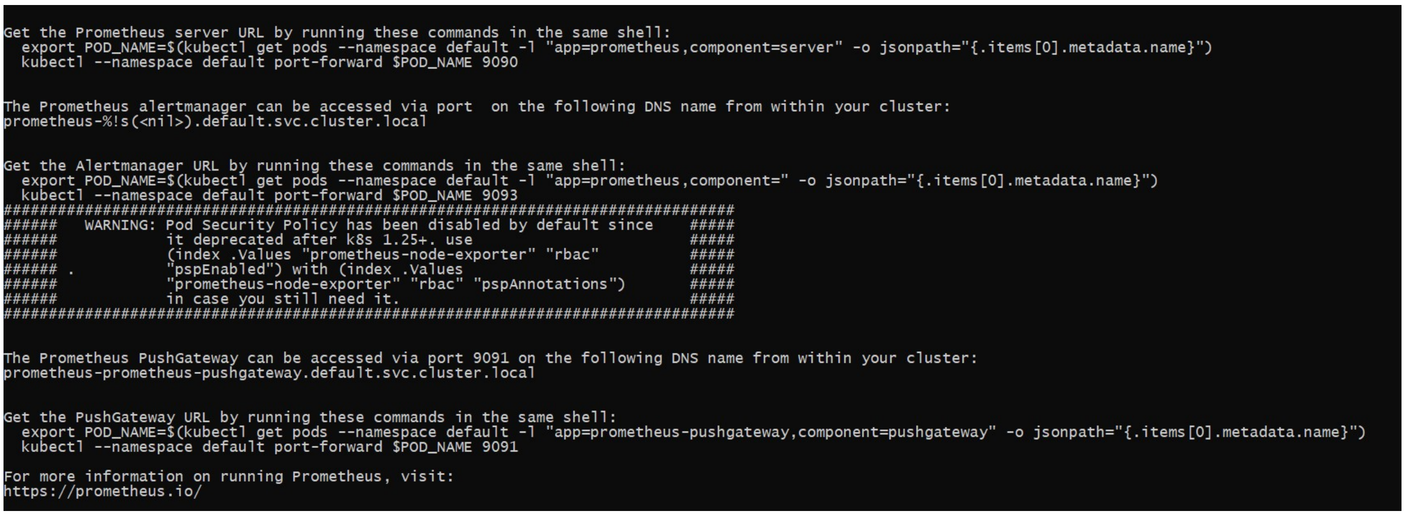
helm install prometheus prometheus-community/prometheus

helm install prometheus prometheus-community/prometheus

helm install prometheus prometheus-community/prometheus

Installing the Prometheus Helm Chart gives the following output:


```
NAME: prometheus
LAST DEPLOYED: Mon Dec  5 16:29:21 2022
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
The Prometheus server can be accessed via port 80 on the following DNS name from within your cluster:
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=prometheus,component=server" -o jsonpath="{.items[0].metadata.name}")
  kubectl --namespace default port-forward $POD_NAME 9090


The Prometheus alertmanager can be accessed via port  on the following DNS name from within your cluster:
prometheus-%!s(<nil>).default.svc.cluster.local


Get the Alertmanager URL by running these commands in the same shell:
  export POD_NAME=$(kubectl get pods --namespace default -l "app=prometheus,component=" -o jsonpath="{.items[0].metadata.name}")
  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 within your cluster:
prometheus-prometheus-pushgateway.default.svc.cluster.local


Get the PushGateway URL by running these commands in the same shell:
  export POD_NAME=$(kubectl get pods --namespace default -l "app=prometheus-pushgateway,component=pushgateway" -o jsonpath="{.items[0].metadata.name}")
  kubectl --namespace default port-forward $POD_NAME 9091

For more information on running Prometheus, visit:
https://prometheus.io/
```

When you run the ‘helm install’ command:

Kubernetes reads the configuration YAML files in the Helm Chart.

It then sends the Deployments, Services, ConfigMaps, and Secrets configuration YAML files to the Kubernetes API server.

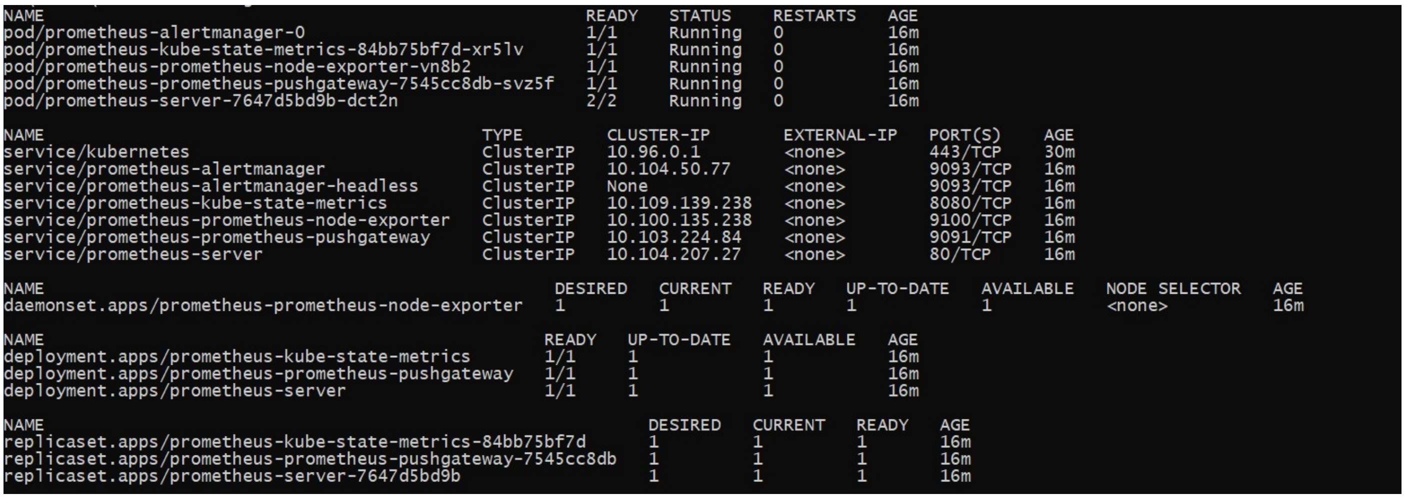
Kubernetes takes these files and creates the Prometheus application inside the Minikube Kubernetes Cluster.

It will install a Prometheus Server, an Alert Manager, and monitoring components.

To get all the deployed Kubernetes resources for the Prometheus Kubernetes application, run this ‘kubectl’ command:

kubectl get all

The command will display all the created Kubernetes objects/components in the Kubernetes cluster as shown below:


```
NAME                                                   READY   STATUS    RESTARTS   AGE
pod/prometheus-alertmanager-0                          1/1     Running   0          16m
pod/prometheus-kube-state-metrics-84bb75bf7d-xr5lv     1/1     Running   0          16m
pod/prometheus-prometheus-node-exporter-vn8b2          1/1     Running   0          16m
pod/prometheus-prometheus-pushgateway-7545cc8db-svz5f  1/1     Running   0          16m
pod/prometheus-server-7647d5bd9b-dct2n                 2/2     Running   0          16m

NAME                                        TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
service/kubernetes                          ClusterIP   10.96.0.1        <none>        443/TCP    30m
service/prometheus-alertmanager             ClusterIP   10.104.50.77     <none>        9093/TCP   16m
service/prometheus-alertmanager-headless    ClusterIP   None             <none>        9093/TCP   16m
service/prometheus-kube-state-metrics       ClusterIP   10.109.139.238   <none>        8080/TCP   16m
service/prometheus-prometheus-node-exporter ClusterIP   10.100.135.238   <none>        9100/TCP   16m
service/prometheus-prometheus-pushgateway   ClusterIP   10.103.224.84    <none>        9091/TCP   16m
service/prometheus-server                   ClusterIP   10.104.207.27    <none>        80/TCP     16m

NAME                                                 DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
daemonset.apps/prometheus-prometheus-node-exporter   1         1         1       1            1           <none>          16m

NAME                                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/prometheus-kube-state-metrics       1/1     1            1           16m
deployment.apps/prometheus-prometheus-pushgateway   1/1     1            1           16m
deployment.apps/prometheus-server                   1/1     1            1           16m

NAME                                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/prometheus-kube-state-metrics-84bb75bf7d       1         1         1       16m
replicaset.apps/prometheus-prometheus-pushgateway-7545cc8db    1         1         1       16m
replicaset.apps/prometheus-server-7647d5bd9b                   1         1         1       16m
```

We have created pods, Kubernetes deployments, and services. Helm simplifies creating all these Kubernetes objects only using a single command.

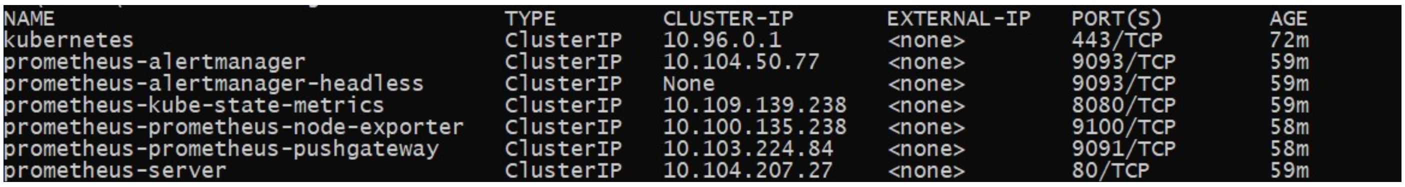
You can read more about Helm here, on Sweetcode.

Get All the Created Kubernetes Services

To get all the created Kubernetes Services, run this command:

kubectl get service

It displays the following Kubernetes Services:



```
NAME                                  TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)      AGE
kubernetes                            ClusterIP   10.96.0.1        <none>        443/TCP      72m
prometheus-alertmanager               ClusterIP   10.104.50.77     <none>        9093/TCP     59m
prometheus-alertmanager-headless      ClusterIP   None             <none>        9093/TCP     59m
prometheus-kube-state-metrics         ClusterIP   10.109.139.238   <none>        8080/TCP     59m
prometheus-prometheus-node-exporter   ClusterIP   10.100.135.238   <none>        9100/TCP     58m
prometheus-prometheus-pushgateway     ClusterIP   10.103.224.84    <none>        9091/TCP     58m
prometheus-server                     ClusterIP   10.104.207.27    <none>        80/TCP       59m
```

The Prometheus Helm Chart has created six Kubernetes services. We will access the Prometheus application using the ‘prometheus-server’ Kubernetes service.

The ‘prometheus-server’ service has a ‘ClusterIP’ Kubernetes Service type. It allows you to access the Prometheus application only inside the Minikube Kubernetes Cluster.

Accessing the Prometheus Kubernetes Application

To access the Prometheus Kubernetes application outside the Minikube Kubernetes Cluster, you will run the following command:

kubectl expose service prometheus-server --type=NodePort --target-port=9090 --name=prometheus-server-ext

kubectl expose service prometheus-server --type=NodePort --target-port=9090 --name=prometheus-server-ext

kubectl expose service prometheus-server --type=NodePort --target-port=9090 --name=prometheus-server-ext

The command will convert the ‘prometheus-server’ Kubernetes Service from the ‘ClusterIP’ type to the ‘NodePort’ type.

It will allow us to access the Prometheus application outside the Minikube cluster on port ‘9090’.

The command outputs the following:

service/prometheus-server-ext exposed

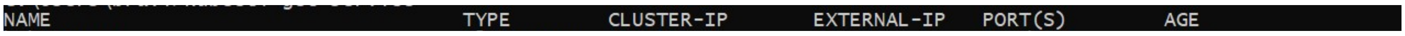
service/prometheus-server-ext exposed

service/prometheus-server-ext exposed

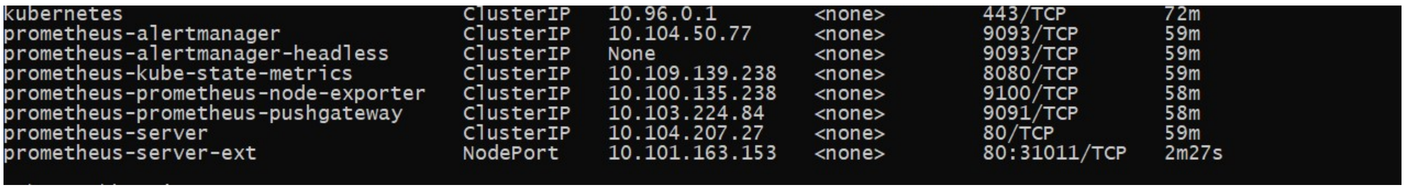
To check the newly created ‘prometheus-server-ext’ Kubernetes Service, run this command:

kubectl get service

The command outputs the following:



```
NAME                                  TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
```



```
kubernetes                           ClusterIP   10.96.0.1        <none>        443/TCP        72m
prometheus-alertmanager              ClusterIP   10.104.50.77     <none>        9093/TCP       59m
prometheus-alertmanager-headless     ClusterIP   None             <none>        9093/TCP       59m
prometheus-kube-state-metrics        ClusterIP   10.109.139.238   <none>        8080/TCP       59m
prometheus-prometheus-node-exporter  ClusterIP   10.100.135.238   <none>        9100/TCP       58m
prometheus-prometheus-pushgateway    ClusterIP   10.103.224.84    <none>        9091/TCP       58m
prometheus-server                    ClusterIP   10.104.207.27    <none>        80/TCP         59m
prometheus-server-ext                NodePort    10.101.163.153   <none>        80:31011/TCP   2m27s
```

You should be able to access the 'prometheus-server-ext' Kubernetes Service outside the Minikube Cluster using an URL.

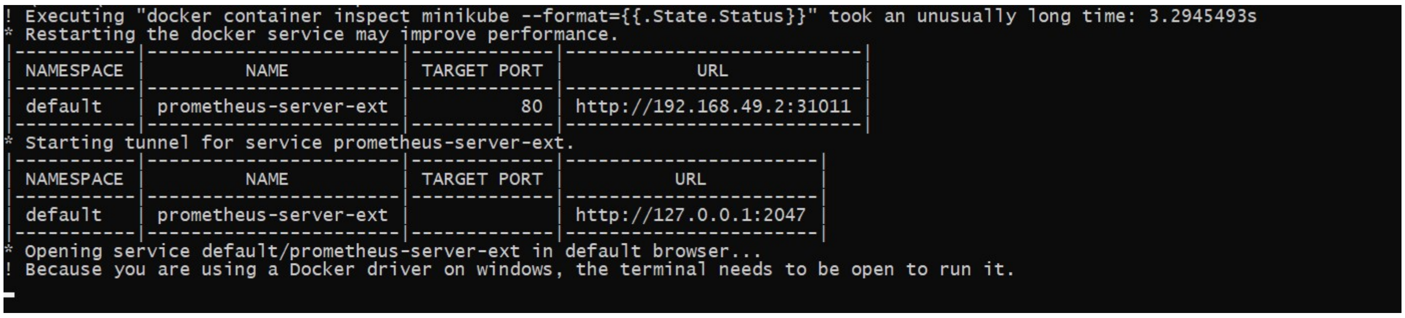
To generate the URL for accessing this service, run this command:

minikube service prometheus-server-ext

minikube service prometheus-server-ext

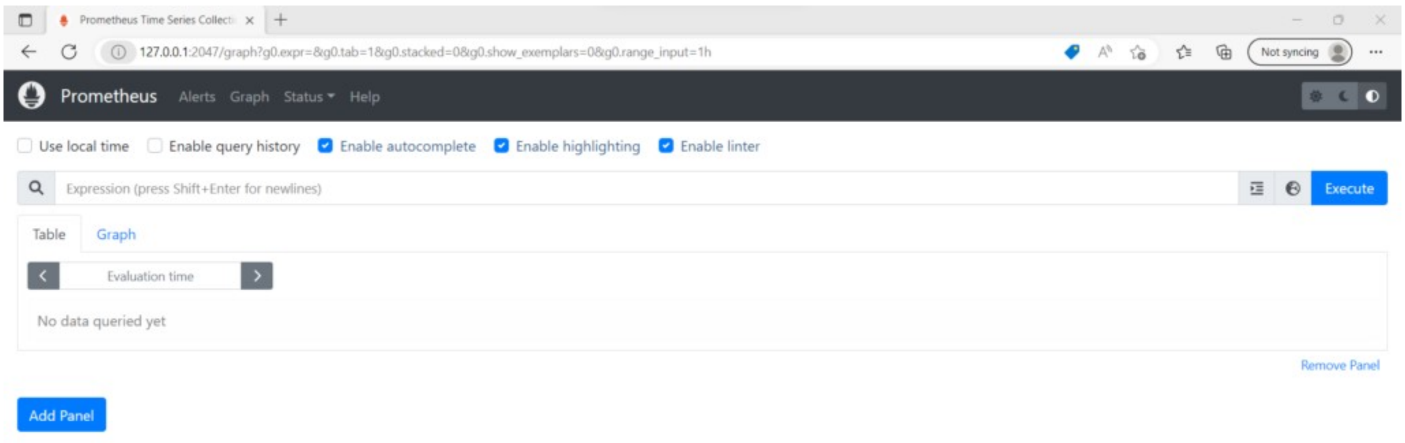
minikube service prometheus-server-ext

The command outputs the following URL:



```
! Executing "docker container inspect minikube --format={{.State.Status}}" took an unusually long time: 3.2945493s
* Restarting the docker service may improve performance.
|-----------|-----------------------|-------------|---------------------------|
| NAMESPACE |         NAME          | TARGET PORT |            URL            |
|-----------|-----------------------|-------------|---------------------------|
| default   | prometheus-server-ext |          80 | http://192.168.49.2:31011 |
|-----------|-----------------------|-------------|---------------------------|
* Starting tunnel for service prometheus-server-ext.
|-----------|-----------------------|-------------|------------------------|
| NAMESPACE |         NAME          | TARGET PORT |          URL           |
|-----------|-----------------------|-------------|------------------------|
| default   | prometheus-server-ext |             | http://127.0.0.1:2047  |
|-----------|-----------------------|-------------|------------------------|
* Opening service default/prometheus-server-ext in default browser...
! Because you are using a Docker driver on windows, the terminal needs to be open to run it.
```

To access the Prometheus application, input the URL in your browser as shown below:



127.0.0.1:2047/graph?g0.expr=&g0.tab=1&g0.stacked=0&g0.show_exemplars=0&g0.range_input=1h

From the image above, we have installed the Prometheus application inside the Minikube Kubernetes cluster. This is the Prometheus web UI.

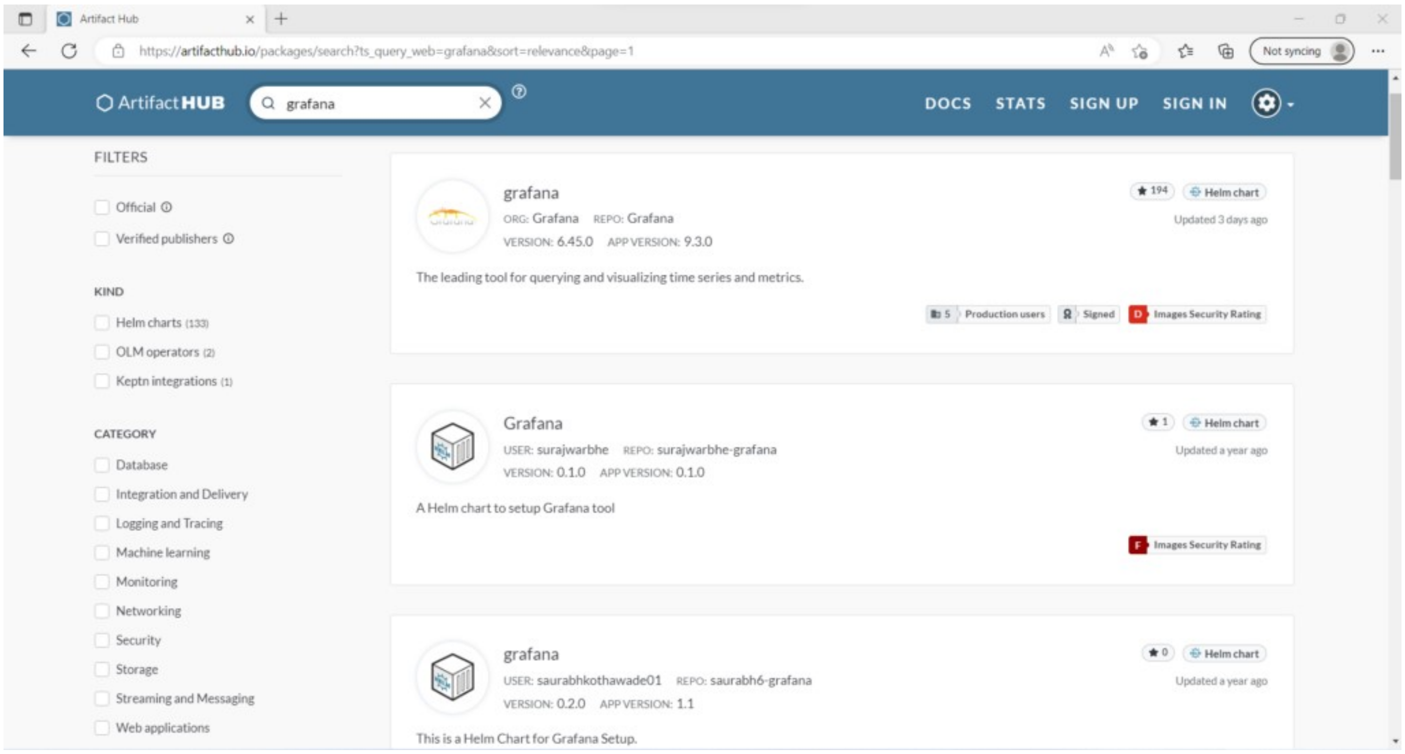
Exploring the Prometheus UI

Let’s explore this Prometheus UI as follows:

These are some of the components on the Prometheus web UI. The next step is to install Grafana on the Minikube Kubernetes cluster. We will then integrate it to use Prometheus as the data source. Grafana will create dashboards and charts from the data source.

Searching for Grafana Helm Chart

We will search the Grafana Helm Chart from the Artifact Hub as shown below:



grafana
ORG: Grafana REPO: Grafana
VERSION: 6.45.0 APP VERSION: 9.3.0
★ 194 Helm chart
Updated 3 days ago
The leading tool for querying and visualizing time series and metrics.
5 Production users | Signed | D Images Security Rating

Grafana
USER: surajwarbhe REPO: surajwarbhe-grafana
VERSION: 0.1.0 APP VERSION: 0.1.0
★ 1 Helm chart
Updated a year ago
A Helm chart to setup Grafana tool
F Images Security Rating

grafana
USER: saurabhkothawade01 REPO: saurabh6-grafana
VERSION: 0.2.0 APP VERSION: 1.1
★ 0 Helm chart
Updated a year ago
This is a Helm Chart for Grafana Setup.

We will use the official Grafana Helm from Grafana. Let’s add the `grafana` repository for our Grafana Helm Chart as follows:

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

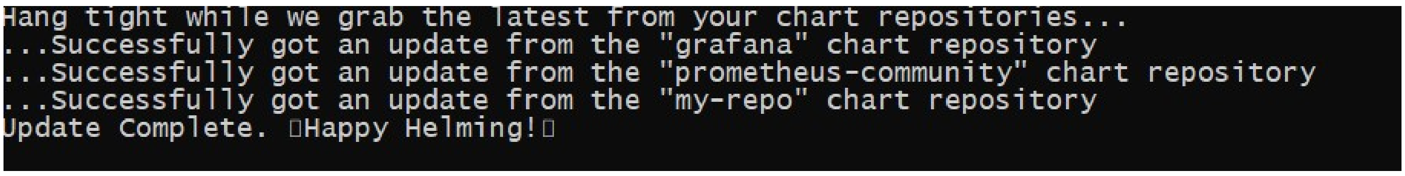
helm repo add grafana https://grafana.github.io/helm-charts

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

To update the added Grafana repository, run this command:

helm repo update

It gives the following output:



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

Installing the Grafana Helm Chart

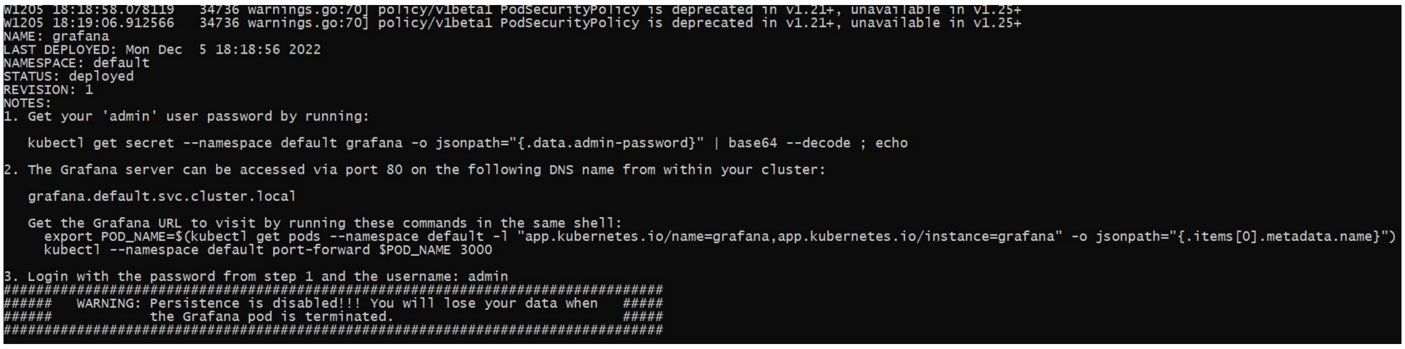
Installing the Grafana Helm Chart will upload or deploy the Grafana Helm Chart to Minikube Kubernetes Cluster. To install the Grafana Helm Chart, use this ‘helm install’ command:

helm install grafana grafana/grafana

helm install grafana grafana/grafana

helm install grafana grafana/grafana

Installing the Grafana Helm Chart gives the following output:



```
W1205 18:18:58.078119   34736 warnings.go:70] policy/v1beta1 PodSecurityPolicy is deprecated in v1.21+, unavailable in v1.25+
W1205 18:19:06.912566   34736 warnings.go:70] policy/v1beta1 PodSecurityPolicy is deprecated in v1.21+, unavailable in v1.25+
NAME: grafana
LAST DEPLOYED: Mon Dec  5 18:18:56 2022
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-password}" | base64 --decode ; echo

2. The Grafana server can be accessed via port 80 on the following DNS name from within your cluster:

   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.io/name=grafana,app.kubernetes.io/instance=grafana" -o jsonpath="{.items[0].metadata.name}")
     kubectl --namespace default port-forward $POD_NAME 3000

3. Login with the password from step 1 and the username: admin
#################################################################################
######   WARNING: Persistence is disabled!!! You will lose your data when   #####
######            the Grafana pod is terminated.                            #####
#################################################################################
```

When you run the ‘helm install’ command:

Kubernetes reads the configuration YAML files in the Helm Chart.

It then sends the Deployments, Services, ConfigMaps, and Secrets configuration YAML files to the Kubernetes API server.

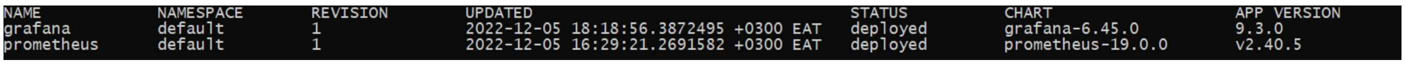
Kubernetes takes these files and creates the Grafana application inside the Minikube Kubernetes Cluster.

Listing the Installed Helm Charts

To list the two installed Helm charts, run this command:

helm list

The command gives the following output:

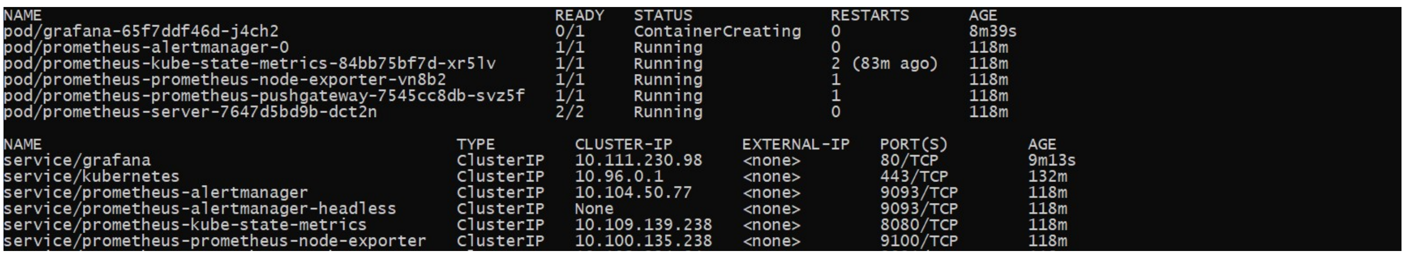


```
NAME        NAMESPACE   REVISION    UPDATED                                 STATUS      CHART               APP VERSION
grafana     default     1           2022-12-05 18:18:56.3872495 +0300 EAT   deployed    grafana-6.45.0      9.3.0
prometheus  default     1           2022-12-05 16:29:21.2691582 +0300 EAT   deployed    prometheus-19.0.0   v2.40.5
```

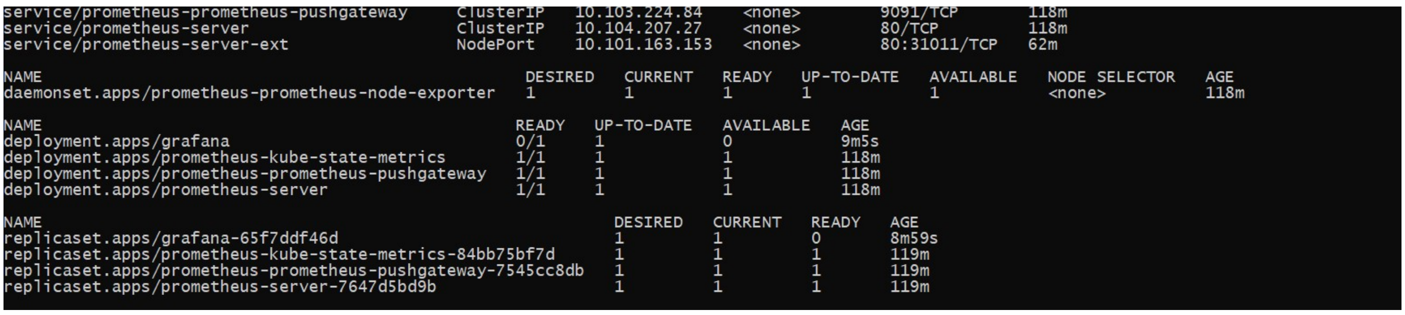
We have ‘grafana’ and ‘prometheus’ deployed to the Minikube Kubernetes Cluster. To get the Kubernetes objects created after deploying the Grafana Helm Chart, run this command:

kubectl get all

The command gives the following output:



```
NAME                                                 READY   STATUS              RESTARTS      AGE
pod/grafana-65f7ddf46d-j4ch2                         0/1     ContainerCreating   0             8m39s
pod/prometheus-alertmanager-0                        1/1     Running             0             118m
pod/prometheus-kube-state-metrics-84bb75bf7d-xr5lv   1/1     Running             2 (83m ago)   118m
pod/prometheus-prometheus-node-exporter-vn8b2        1/1     Running             1             118m
pod/prometheus-prometheus-pushgateway-7545cc8db-svz5f   1/1  Running             1             118m
pod/prometheus-server-7647d5bd9b-dct2n               2/2     Running             0             118m

NAME                                         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)       AGE
service/grafana                              ClusterIP   10.111.230.98    <none>        80/TCP        9m13s
service/kubernetes                           ClusterIP   10.96.0.1        <none>        443/TCP       132m
service/prometheus-alertmanager              ClusterIP   10.104.50.77     <none>        9093/TCP      118m
service/prometheus-alertmanager-headless     ClusterIP   None             <none>        9093/TCP      118m
service/prometheus-kube-state-metrics        ClusterIP   10.109.139.238   <none>        8080/TCP      118m
service/prometheus-prometheus-node-exporter  ClusterIP   10.100.135.238   <none>        9100/TCP      118m
```



```
service/prometheus-prometheus-pushgateway     ClusterIP   10.103.224.84    <none>        9091/TCP       118m
service/prometheus-server                     ClusterIP   10.104.207.27    <none>        80/TCP         118m
service/prometheus-server-ext                 NodePort    10.101.163.153   <none>        80:31011/TCP   62m

NAME                                           DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
daemonset.apps/prometheus-prometheus-node-exporter   1         1         1       1            1           <none>          118m

NAME                                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/grafana                       0/1     1            0           9m5s
deployment.apps/prometheus-kube-state-metrics       1/1     1            1           118m
deployment.apps/prometheus-prometheus-pushgateway   1/1     1            1           118m
deployment.apps/prometheus-server             1/1     1            1           118m

NAME                                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/grafana-65f7ddf46d                    1         1         0       8m59s
replicaset.apps/prometheus-kube-state-metrics-84bb75bf7d         1         1         1       119m
replicaset.apps/prometheus-prometheus-pushgateway-7545cc8db   1         1         1       119m
replicaset.apps/prometheus-server-7647d5bd9b          1         1         1       119m
```

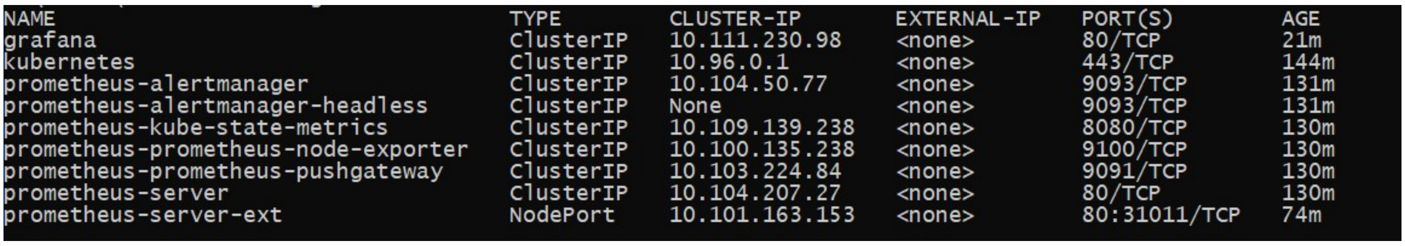
We have created pods, Kubernetes deployments, and services for the Grafana Kubernetes application.

Get All the Created Kubernetes Services for the Grafana Kubernetes Application

To get all the created Kubernetes Services, run this command:

kubectl get service

It displays the following Kubernetes Services:



```
NAME                                  TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
grafana                               ClusterIP   10.111.230.98    <none>        80/TCP         21m
kubernetes                            ClusterIP   10.96.0.1        <none>        443/TCP        144m
prometheus-alertmanager               ClusterIP   10.104.50.77     <none>        9093/TCP       131m
prometheus-alertmanager-headless      ClusterIP   None             <none>        9093/TCP       131m
prometheus-kube-state-metrics         ClusterIP   10.109.139.238   <none>        8080/TCP       130m
prometheus-prometheus-node-exporter   ClusterIP   10.100.135.238   <none>        9100/TCP       130m
prometheus-prometheus-pushgateway     ClusterIP   10.103.224.84    <none>        9091/TCP       130m
prometheus-server                     ClusterIP   10.104.207.27    <none>        80/TCP         130m
prometheus-server-ext                 NodePort    10.101.163.153   <none>        80:31011/TCP   74m
```

The Grafana Helm Chart has created a ‘grafana’ Kubernetes Service. We will access the Grafana application using the ‘grafana’ Kubernetes service.

The ‘grafana’ service has a ‘ClusterIP’ Kubernetes Service type. It allows you to access the Grafana application only inside the Minikube Kubernetes Cluster.

Accessing the Grafana Kubernetes Application

To access the Grafana Kubernetes application outside the Minikube Kubernetes Cluster, you will run the following command:

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

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

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

The command will also convert the ‘Grafana-server’ Kubernetes Service from the ‘ClusterIP’ type to the ‘NodePort’ type. This will allow us to access the Grafana application outside the Minikube cluster on port ‘3000’.

The command outputs the following:

service/grafana-ext exposed

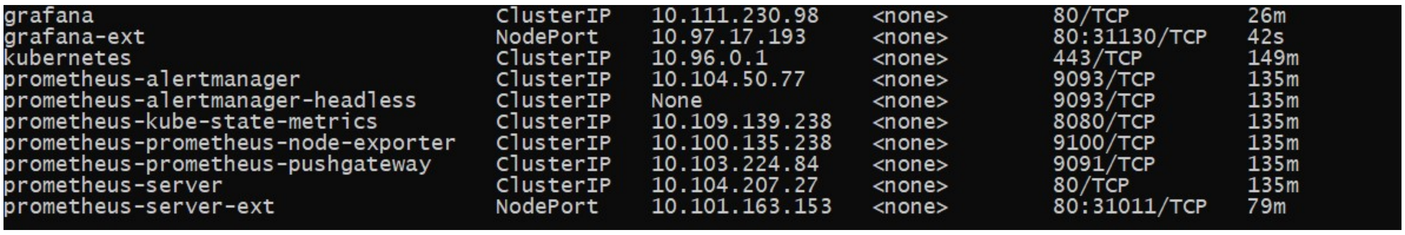
service/grafana-ext exposed

service/grafana-ext exposed

To check the newly created ‘grafana-ext’ Kubernetes Service, run this command:

kubectl get service

The command outputs the following:



```
grafana                               ClusterIP   10.111.230.98    <none>        80/TCP         26m
grafana-ext                           NodePort    10.97.17.193     <none>        80:31130/TCP   42s
kubernetes                            ClusterIP   10.96.0.1        <none>        443/TCP        149m
prometheus-alertmanager               ClusterIP   10.104.50.77     <none>        9093/TCP       135m
prometheus-alertmanager-headless      ClusterIP   None             <none>        9093/TCP       135m
prometheus-kube-state-metrics         ClusterIP   10.109.139.238   <none>        8080/TCP       135m
prometheus-prometheus-node-exporter   ClusterIP   10.100.135.238   <none>        9100/TCP       135m
prometheus-prometheus-pushgateway     ClusterIP   10.103.224.84    <none>        9091/TCP       135m
prometheus-server                     ClusterIP   10.104.207.27    <none>        80/TCP         135m
prometheus-server-ext                 NodePort    10.101.163.153   <none>        80:31011/TCP   79m
```

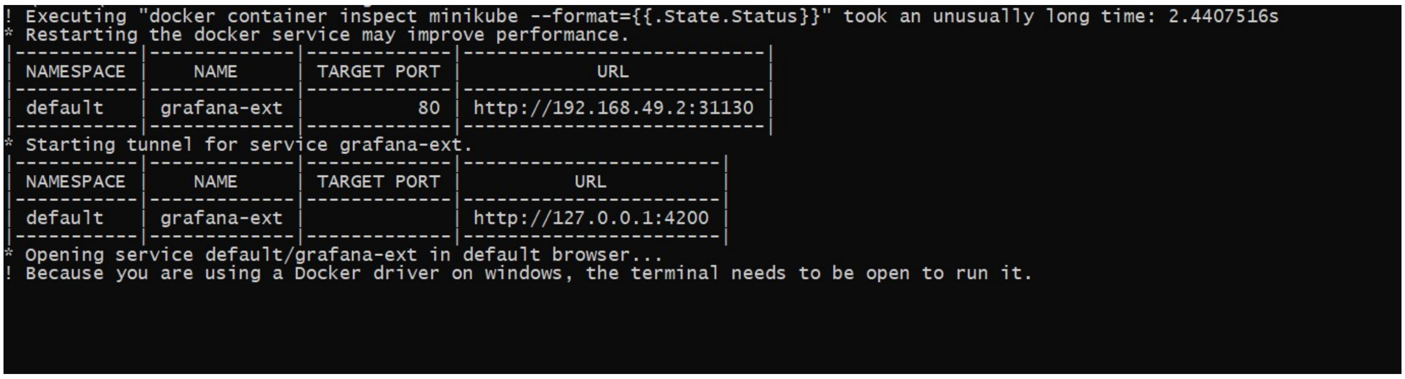
You should be able to access the ‘grafana-ext’ Kubernetes Service outside the Minikube Cluster using an URL. To generate the URL for accessing this service, run this command:

minikube service grafana-ext

minikube service grafana-ext

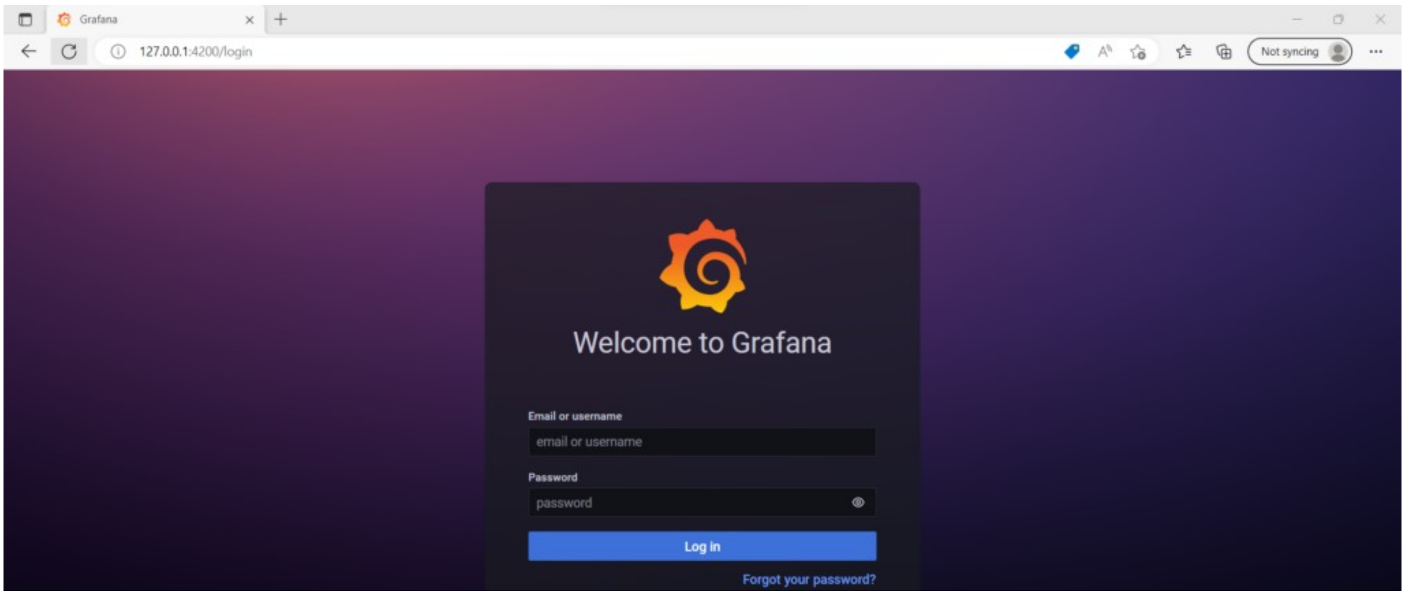
minikube service grafana-ext

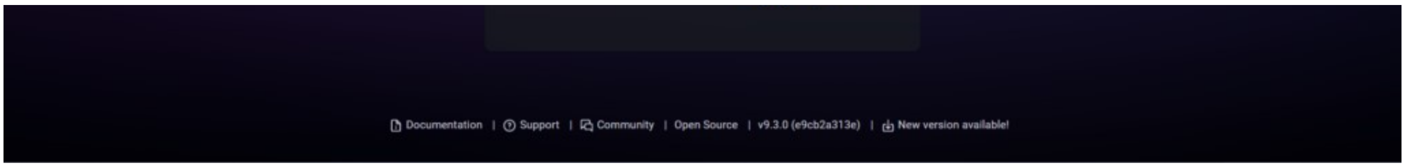
The command outputs the following URL:



```
! Executing "docker container inspect minikube --format={{.State.Status}}" took an unusually long time: 2.4407516s
* Restarting the docker service may improve performance.
|-----------|-------------|-------------|---------------------------|
| NAMESPACE |    NAME     | TARGET PORT |            URL            |
|-----------|-------------|-------------|---------------------------|
| default   | grafana-ext |          80 | http://192.168.49.2:31130 |
|-----------|-------------|-------------|---------------------------|
* Starting tunnel for service grafana-ext.
|-----------|-------------|-------------|------------------------|
| NAMESPACE |    NAME     | TARGET PORT |          URL           |
|-----------|-------------|-------------|------------------------|
| default   | grafana-ext |             | http://127.0.0.1:4200  |
|-----------|-------------|-------------|------------------------|
* Opening service default/grafana-ext in default browser...
! Because you are using a Docker driver on windows, the terminal needs to be open to run it.
```

To access the Grafana application, input the URL in your browser as shown below:




Documentation | Support | Community | Open Source | v9.3.0 (e9cb2a313e) | New version available!

From the image above, we have installed the Grafana Helm Chart application inside the Minikube Kubernetes cluster. The next thing is to login into the Grafana application.

Login Into Grafana

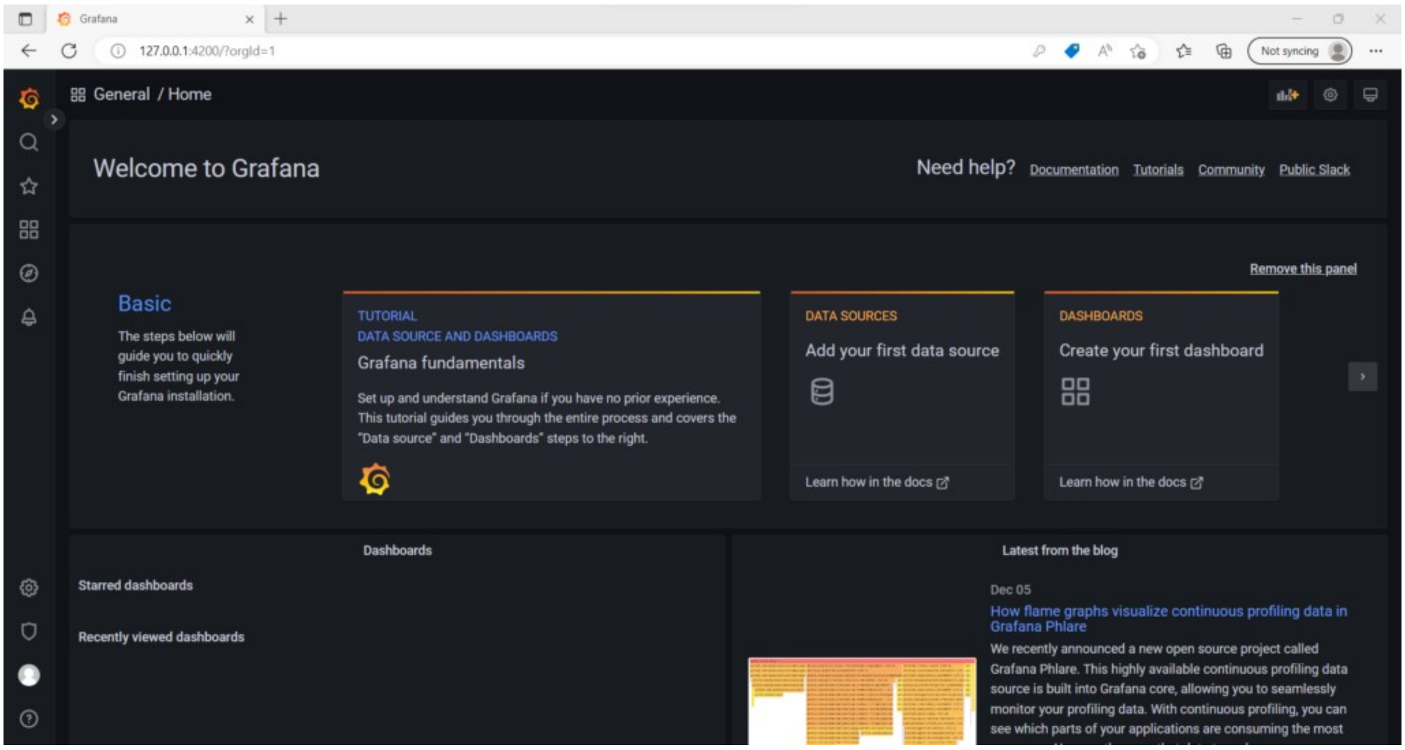
To login into Grafana, we need to get the password for the `admin` user using the following command:

```
kubectl get secret --namespace default grafana -o jsonpath="{.data.admin-password}" | base64
--decode ; echo
```

```
kubectl get secret --namespace default grafana -o jsonpath="{.data.admin-password}" | base64
--decode ; echo
```

```
kubectl get secret --namespace default grafana -o jsonpath="{.data.admin-password}" | base64
--decode ; echo
```

On the Grafana login page, input the username as 'admin' and your decoded password to access the Grafana application. After logging in, you will be redirected to the Grafana homepage as shown below:



The steps below will guide you to quickly finish setting up your Grafana installation.

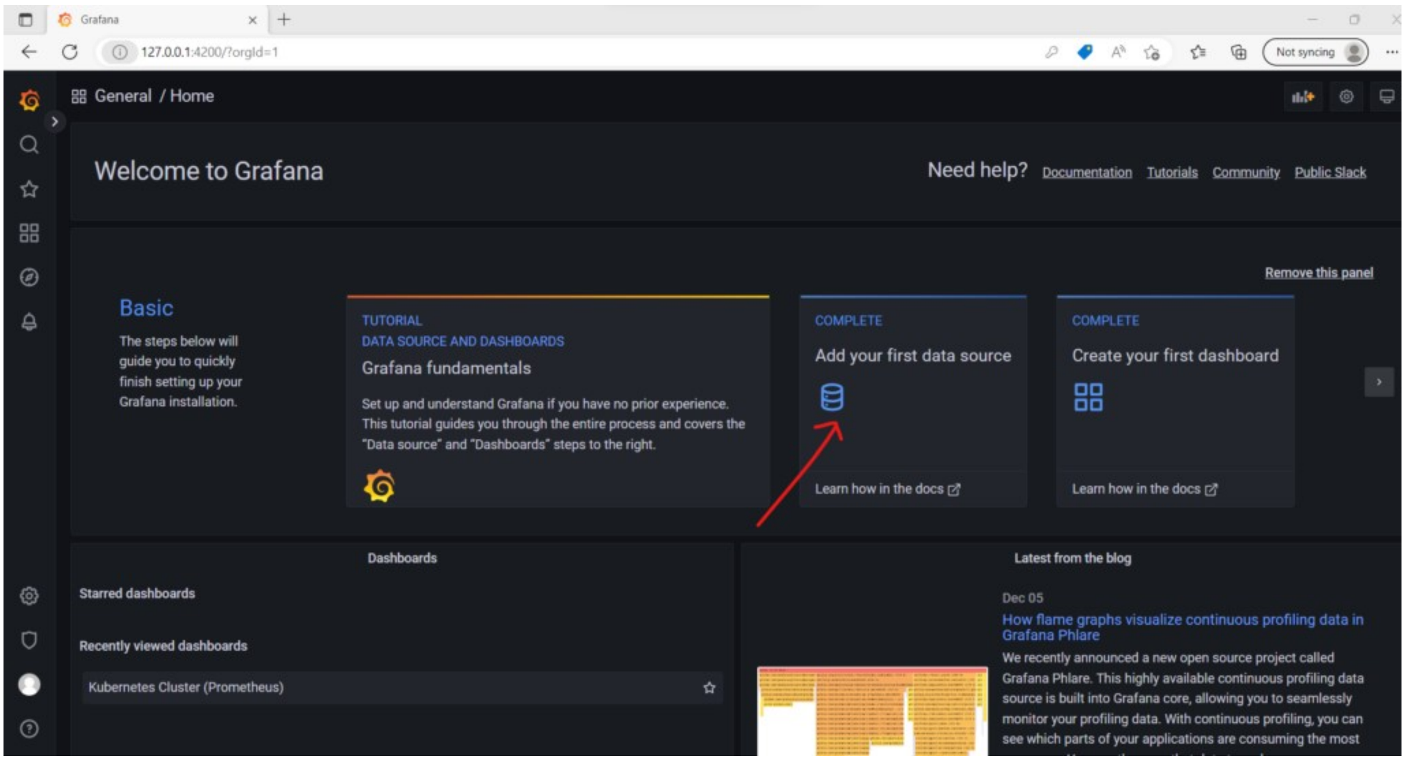
Set up and understand Grafana if you have no prior experience. This tutorial guides you through the entire process and covers the "Data source" and "Dashboards" steps to the right.

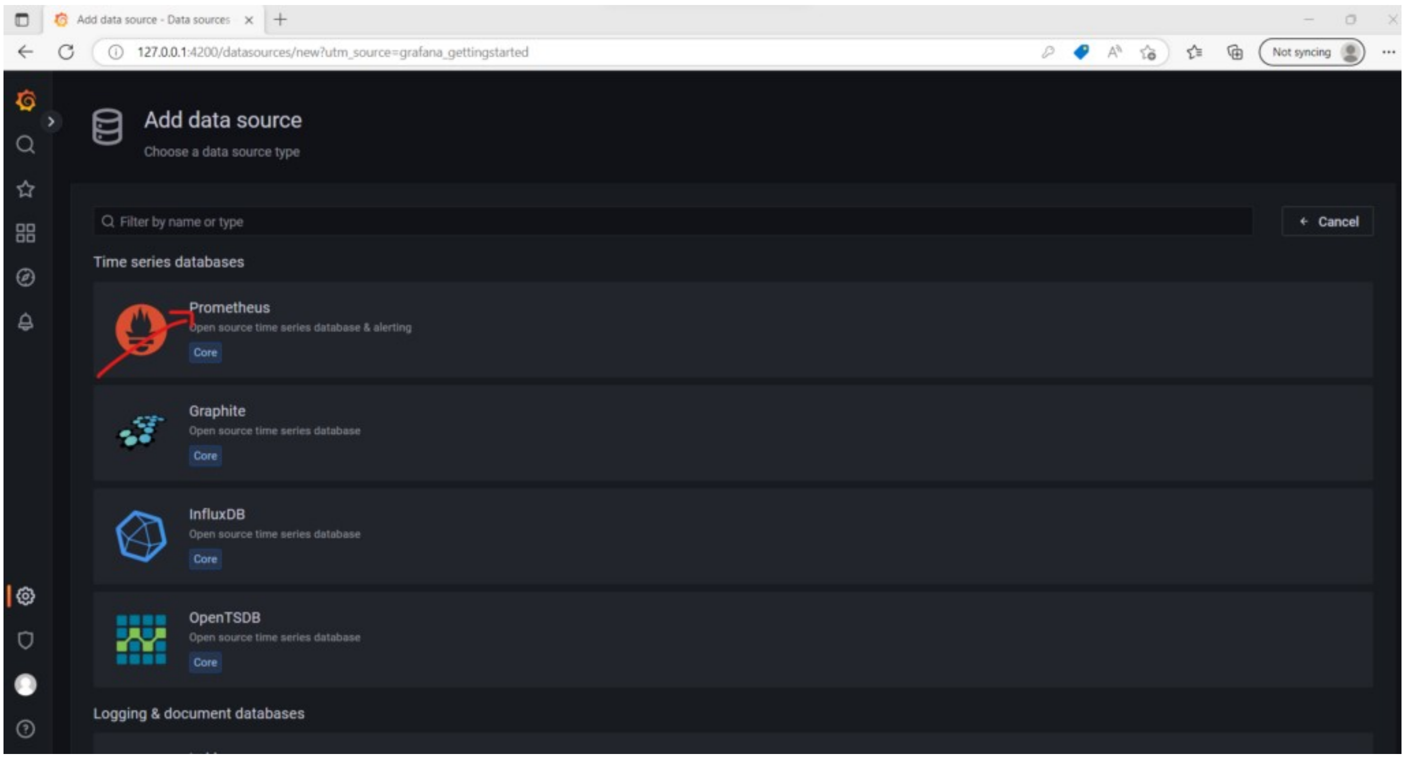
We recently announced a new open source project called Grafana Phlare. This highly available continuous profiling data source is built into Grafana core, allowing you to seamlessly monitor your profiling data. With continuous profiling, you can see which parts of your applications are consuming the most

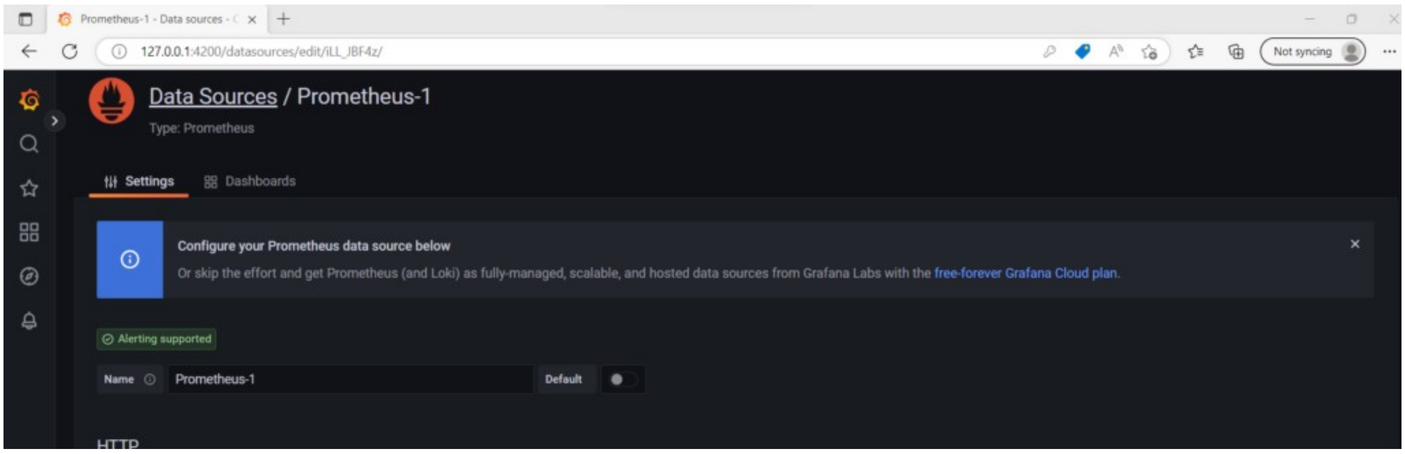
Let's integrate Grafana with Prometheus. Grafana will start using Prometheus as the data source.

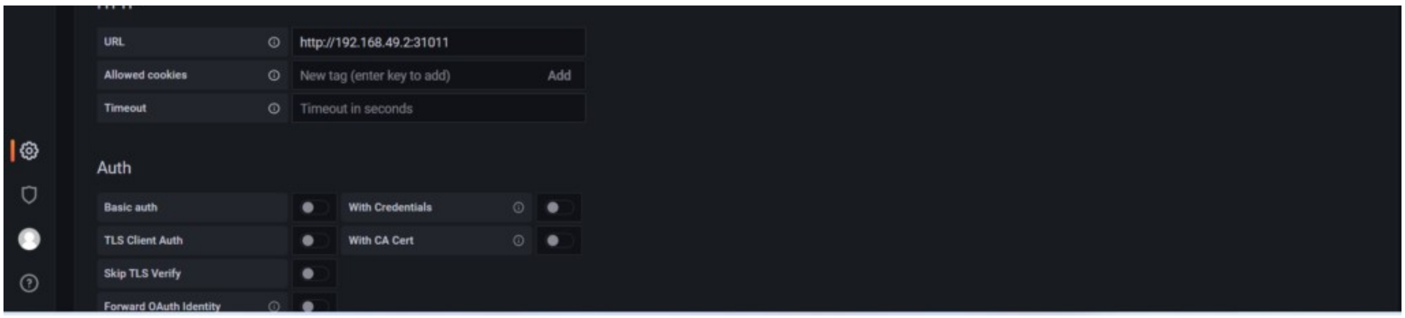
Integrating Grafana and Prometheus

To integrate Grafana and Prometheus, add Prometheus as the data source. Prometheus is running on http://192.168.49.2:31011. The steps are shown in the images below:


Grafana
127.0.0.1:4200/?orgId=1
Not syncing
General / Home
Welcome to Grafana
Need help? Documentation Tutorials Community Public Slack
Remove this panel
Basic
The steps below will guide you to quickly finish setting up your Grafana installation.
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.
COMPLETE
Add your first data source
Learn how in the docs
COMPLETE
Create your first dashboard
Learn how in the docs
Dashboards
Starred dashboards
Recently viewed dashboards
Kubernetes Cluster (Prometheus)
Latest from the blog
Dec 05
How flame graphs visualize continuous profiling data in Grafana Phlare
We recently announced a new open source project called Grafana Phlare. This highly available continuous profiling data source is built into Grafana core, allowing you to seamlessly monitor your profiling data. With continuous profiling, you can see which parts of your applications are consuming the most


Add data source - Data sources
127.0.0.1:4200/datasources/new?utm_source=grafana_gettingstarted
Add data source
Choose a data source type
Filter by name or type
Cancel
Time series databases
Prometheus
Open source time series database & alerting
Core
Graphite
Open source time series database
Core
InfluxDB
Open source time series database
Core
OpenTSDB
Open source time series database
Core
Logging & document databases


Prometheus-1 - Data sources
127.0.0.1:4200/datasources/edit/iLL_JBF4z/
Data Sources / Prometheus-1
Type: Prometheus
Settings
Dashboards
Configure your Prometheus data source below
Or skip the effort and get Prometheus (and Loki) as fully-managed, scalable, and hosted data sources from Grafana Labs with the free-forever Grafana Cloud plan.
Alerting supported
Name
Prometheus-1
Default
HTTP


URL: http://192.168.49.2:31011

Allowed cookies: New tag (enter key to add) Add

Timeout: Timeout in seconds

Auth

Basic auth | With Credentials

TLS Client Auth | With CA Cert

Skip TLS Verify

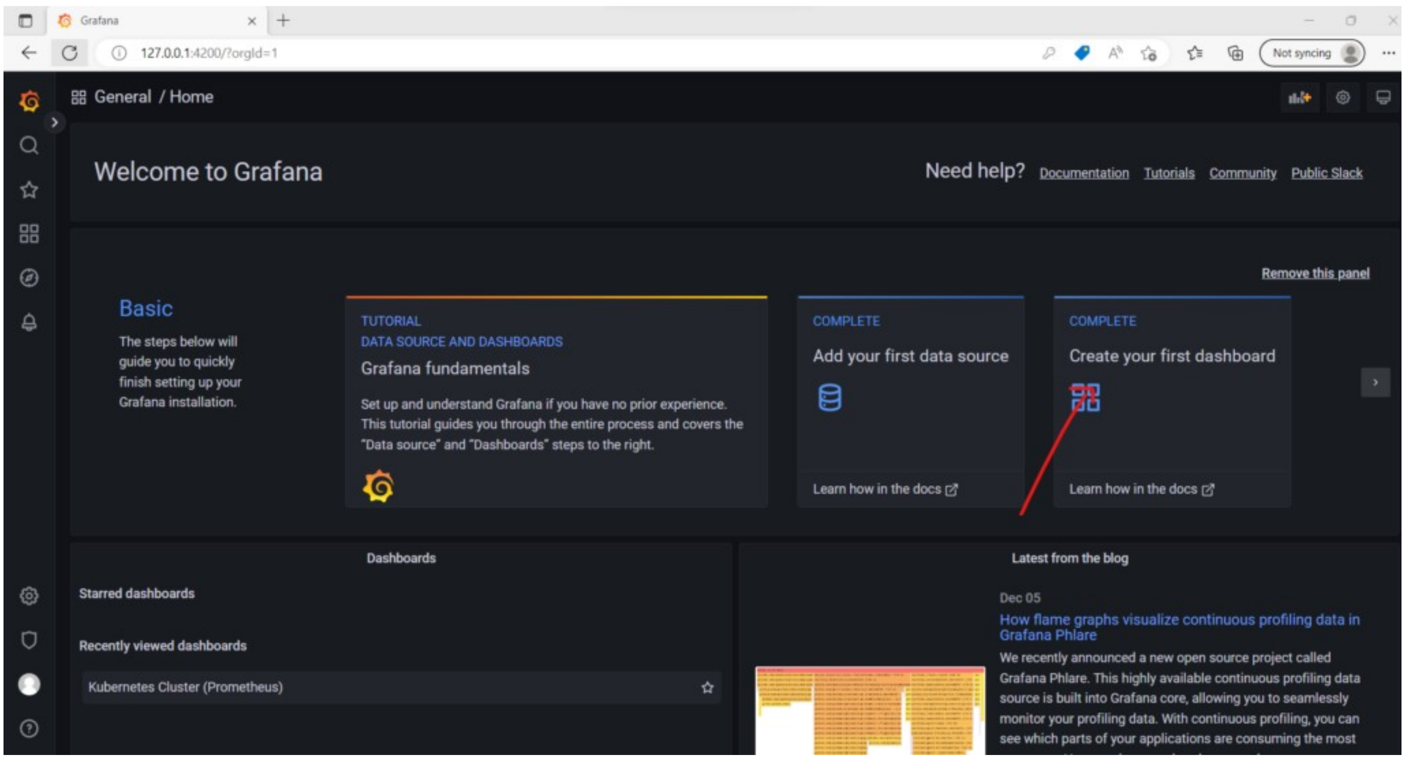
Forward OAuth Identity

Once you have connected with Prometheus, let’s create a visualization dashboard. We will use the visualization dashboard to monitor the Minikube cluster. It will show the statistics of the cluster usage and running applications in the cluster.

Creating a Visualization Dashboard

To create a visualization dashboard, follow the steps below:

1. Click ‘Create your first Dashboard’



Welcome to Grafana

Basic

The steps below will guide you to quickly finish setting up your Grafana installation.

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.

COMPLETE
Add your first data source

COMPLETE
Create your first dashboard

Dashboards

Starred dashboards

Recently viewed dashboards

Kubernetes Cluster (Prometheus)

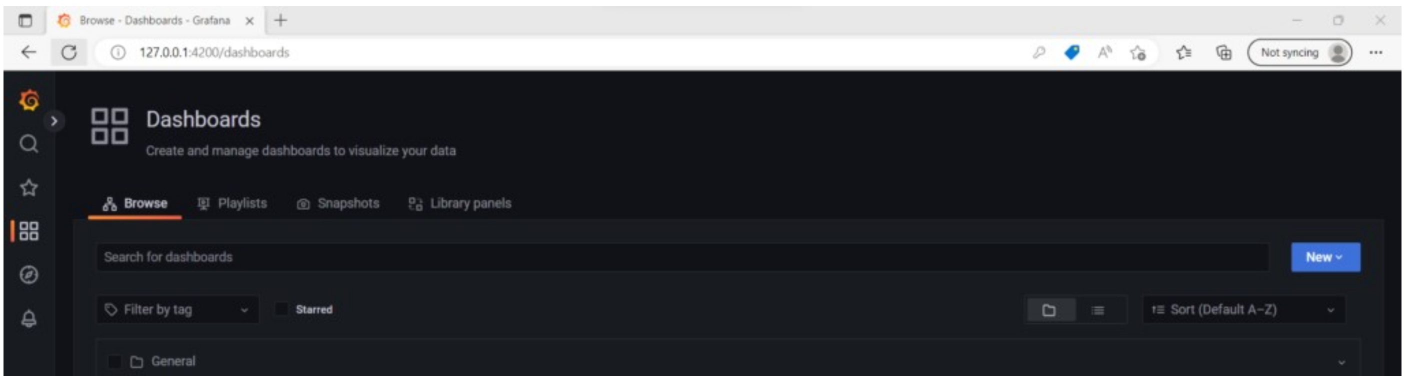
Latest from the blog

Dec 05

How flame graphs visualize continuous profiling data in Grafana Phlare

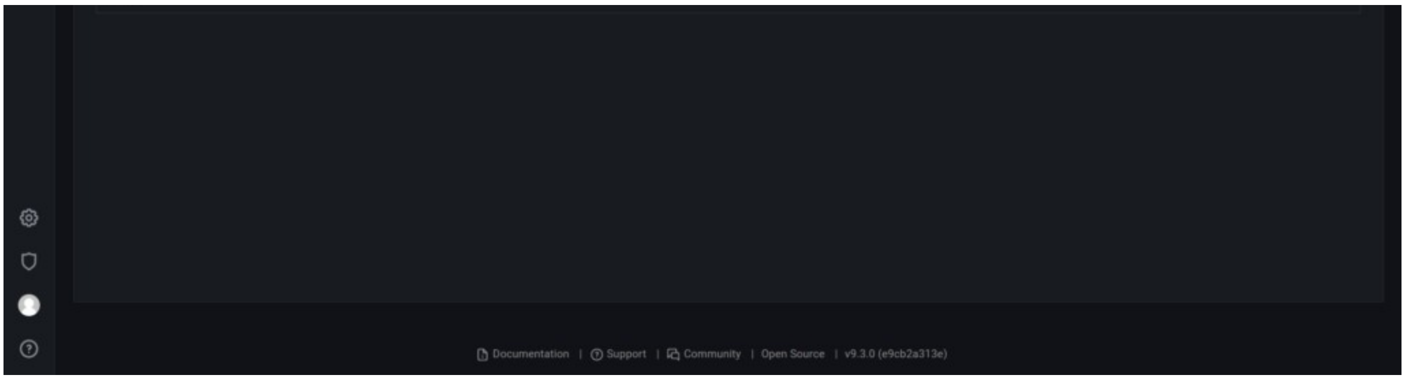
We recently announced a new open source project called Grafana Phlare. This highly available continuous profiling data source is built into Grafana core, allowing you to seamlessly monitor your profiling data. With continuous profiling, you can see which parts of your applications are consuming the most

2. Browse to find a Grafana Dashboard



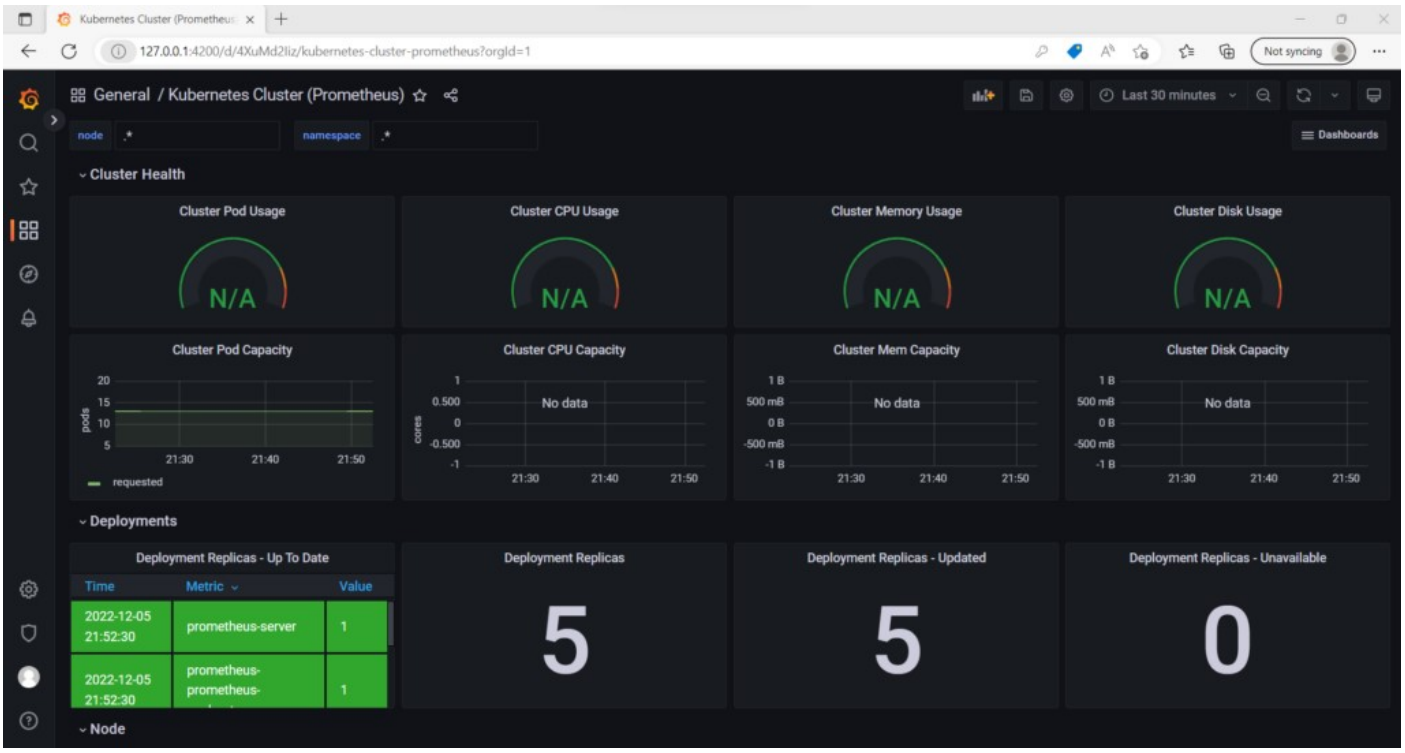
Dashboards

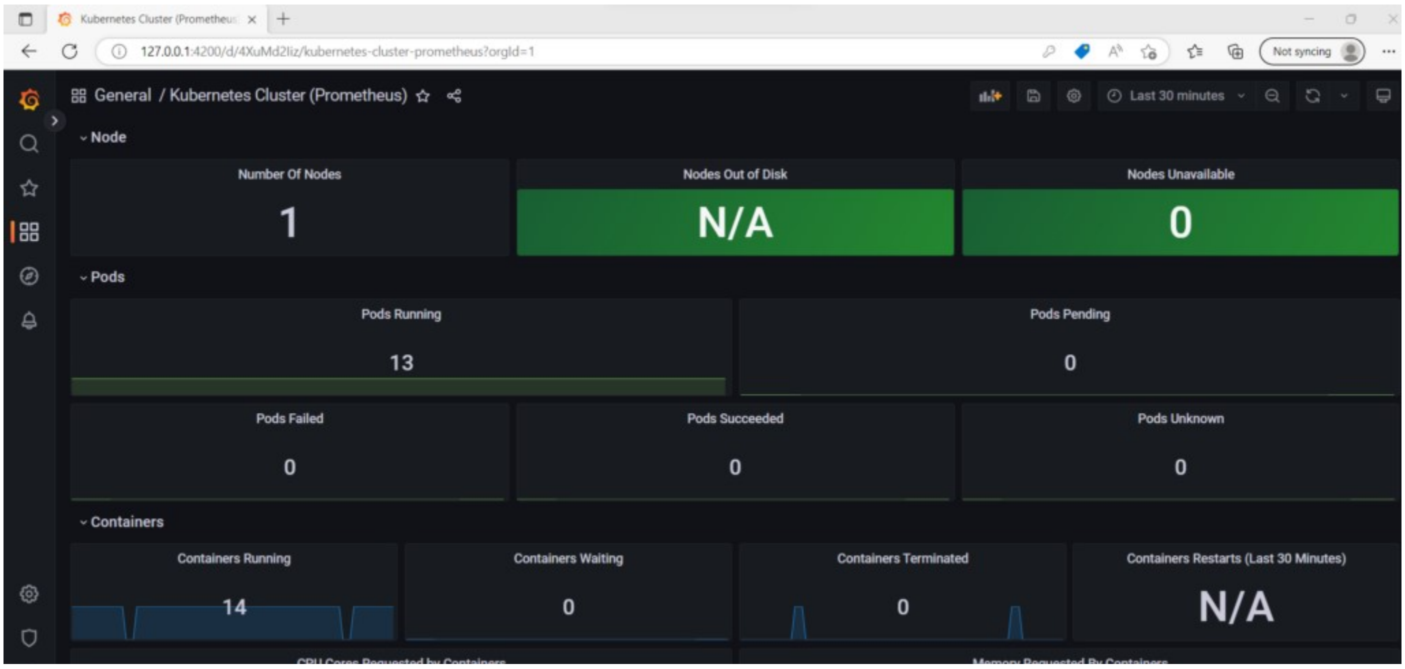
Create and manage dashboards to visualize your data

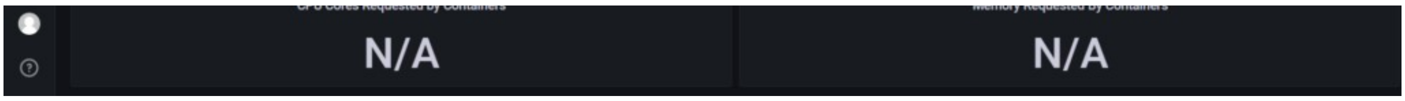

Documentation | Support | Community | Open Source | v9.3.0 (e9cb2a313e)

3. Select Prometheus as the Data Source

After selecting Prometheus as the data source, Grafana creates a Dashboard shown in the images below:


Kubernetes Cluster (Prometheus)
127.0.0.1:4200/d/4XuMd2liz/kubernetes-cluster-prometheus?orgId=1
Not syncing
General / Kubernetes Cluster (Prometheus)
Last 30 minutes
node .*
namespace .*
Dashboards
Cluster Health
Cluster Pod Usage N/A
Cluster CPU Usage N/A
Cluster Memory Usage N/A
Cluster Disk Usage N/A
Cluster Pod Capacity
requested
Cluster CPU Capacity No data
Cluster Mem Capacity No data
Cluster Disk Capacity No data
Deployments
Deployment Replicas - Up To Date
Time Metric Value
2022-12-05 21:52:30 prometheus-server 1
2022-12-05 21:52:30 prometheus- prometheus- 1
Deployment Replicas 5
Deployment Replicas - Updated 5
Deployment Replicas - Unavailable 0
Node




CPU Cores Requested by Containers
N/A
Memory Requested by Containers
N/A

From the image above, we have set Prometheus as the data source. Grafana will use the Kubernetes cluster metrics to monitor its performance.

These Grafana dashboards show the ‘Cluster Pod Usage’, ‘Cluster CPU Usage’, ‘Cluster Memory Usage’, and ‘Cluster Disk Usage’. It also shows the status of the ‘Nodes’, ‘Pods’, and ‘Containers’.

It will enable easy management and monitoring of the cluster. We have successfully integrated Prometheus and Grafana on Kubernetes using Helm.

Conclusion

In this article, you have learned how to integrate Prometheus and Grafana on Kubernetes using Helm. We started by discussing how Prometheus and Grafana work. Next, we discussed the three components of Prometheus and the importance of Prometheus monitoring in DevOps. We also covered how to install Helm in your machine and how to create Helm Charts.

Then we installed Prometheus and Grafana Helm Charts on the Minikube Kubernetes Cluster. We used Kubernetes Services to access these two applications. Our two applications were running inside the Minikube Kubernetes Cluster. After logging into Grafana, we integrated Grafana and Prometheus. We then created a visualization dashboard using the Prometheus data source. We used the Grafana dashboard to display Kubernetes cluster usage and monitor its performance.

This article is just the beginning of getting started with Prometheus and Grafana. Feel free to explore more on this topic to gain more exposure and experience. Thank you all for reading this article and all the best in your DevOps journey.