Deploy Prometheus and Grafana for Monitoring Your Kubernetes Cluster

Table of Contents

Introduction

Prometheus and Grafana are essential tools for monitoring the health and performance of Kubernetes clusters. **Prometheus** is an open-source monitoring system that collects metrics from different sources and stores them in a time-series database. **Grafana**, on the other hand, is a powerful tool for creating dynamic dashboards based on data collected by Prometheus.

In this guide, you will deploy Prometheus and Grafana on a Kubernetes cluster using Minikube, enabling real-time monitoring of cluster metrics.

Problem Statement

As Kubernetes clusters grow in complexity, it becomes increasingly important to monitor the state of your nodes, workloads, and resources. Prometheus can scrape metrics from your Kubernetes components, and Grafana allows you to visualize those metrics through customizable dashboards. This lab will walk you through deploying these tools in your Minikube cluster.

Prerequisites

Completion of all previous lab guides (up to Lab Guide-07) is required before proceeding with Lab Guide-08.

- A working **Minikube** cluster on Windows.
- **kubectl** installed and configured to interact with your cluster.
- Basic understanding of Kubernetes objects (Deployments, Services, etc.).

Software Requirements

- **Minikube**: v1.19 or later
- **kubectl**: Latest version compatible with your Kubernetes setup

- **Helm**: Latest version (for easy Prometheus and Grafana installation)

Hardware Requirements

- Minimum 2 CPU cores
- 4GB RAM for Minikube cluster

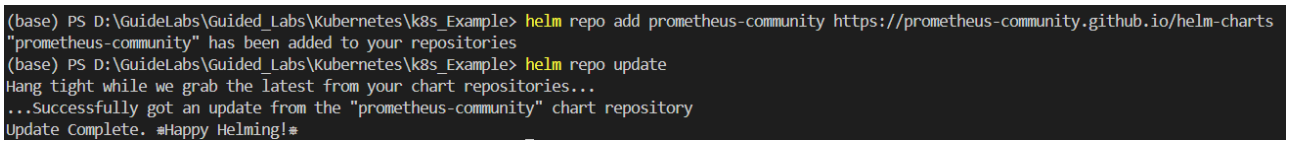
Lab Guide: Deploying Prometheus and Grafana

Step 1: Set Up Prometheus

1. **Install Helm (if not installed)**
   Helm simplifies the installation of Kubernetes applications. Download and install Helm from the official website here.

2. **Add Prometheus Helm Repository**
   Run the following command to add the Prometheus Helm charts repository:

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



3. **Install Prometheus**
   Now, install Prometheus using the Helm chart:

```
helm install prometheus prometheus-community/prometheus --namespace
monitoring --create-namespace
```

```
(base) PS D:\GuideLabs\Guided_Labs\Kubernetes\k8s_Example> helm install prometheus prometheus-community/prometheus --namespace monitoring --create-namespace
NAME: prometheus
LAST DEPLOYED: Wed Oct 16 09:17:47 2024
NAMESPACE: monitoring
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The Prometheus server can be accessed via port 80 on the following DNS name from within your cluster:
prometheus-server.monitoring.svc.cluster.local


Get the Prometheus server URL by running these commands in the same shell:
  export POD_NAME=$(kubectl get pods --namespace monitoring -l "app.kubernetes.io/name=prometheus,app.kubernetes.io/instance=prometheus" -o jsonpath="{.items[0].metadata.name}")
  kubectl --namespace monitoring port-forward $POD_NAME 9090

The Prometheus alertmanager can be accessed via port 9093 on the following DNS name from within your cluster:
prometheus-alertmanager.monitoring.svc.cluster.local


Get the Alertmanager URL by running these commands in the same shell:
  export POD_NAME=$(kubectl get pods --namespace monitoring -l "app.kubernetes.io/name=alertmanager,app.kubernetes.io/instance=prometheus" -o jsonpath="{.items[0].metadata.name}")
  kubectl --namespace monitoring 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.monitoring.svc.cluster.local


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

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

This will create a `monitoring` namespace and install Prometheus in your cluster.

4. **Verify Prometheus Deployment**
   Check if the Prometheus pods are running:

```
kubectl get pods -n monitoring
```

```
(base) PS D:\GuideLabs\Guided_Labs\Kubernetes\k8s_Example> kubectl get pods -n monitoring
NAME                                             READY   STATUS    RESTARTS   AGE
prometheus-alertmanager-0                        1/1     Running   0          58s
prometheus-kube-state-metrics-75b5bb4bf8-79bpw   1/1     Running   0          58s
prometheus-prometheus-node-exporter-9fslj        1/1     Running   0          58s
prometheus-prometheus-pushgateway-84557d6c79-lgmhm   1/1     Running   0          58s
prometheus-server-644d686bc6-k7s8k               2/2     Running   0          58s
```

You should see several pods, including `prometheus-server`, `alertmanager`, and `node-exporter`.

Step 2: Set Up Grafana

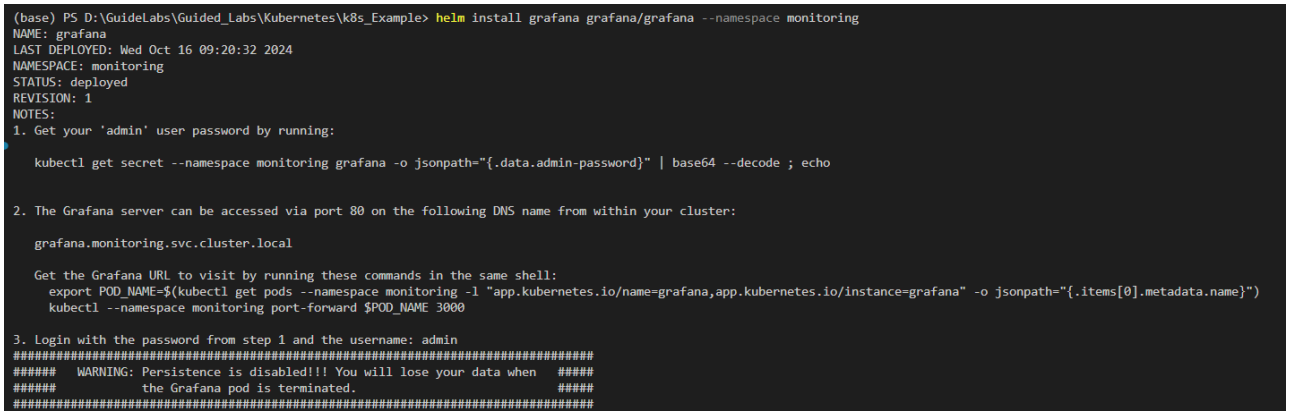
1. **Add Grafana Helm Repository**
   Run the following command to add the Grafana Helm charts repository:

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

```
(base) PS D:\GuideLabs\Guided_Labs\Kubernetes\k8s_Example> helm repo add grafana https://grafana.github.io/helm-charts
"grafana" has been added to your repositories
(base) PS D:\GuideLabs\Guided_Labs\Kubernetes\k8s_Example> helm repo update
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
Update Complete. ⎈Happy Helming!⎈
```

2. **Install Grafana**
   Use the Helm chart to install Grafana:

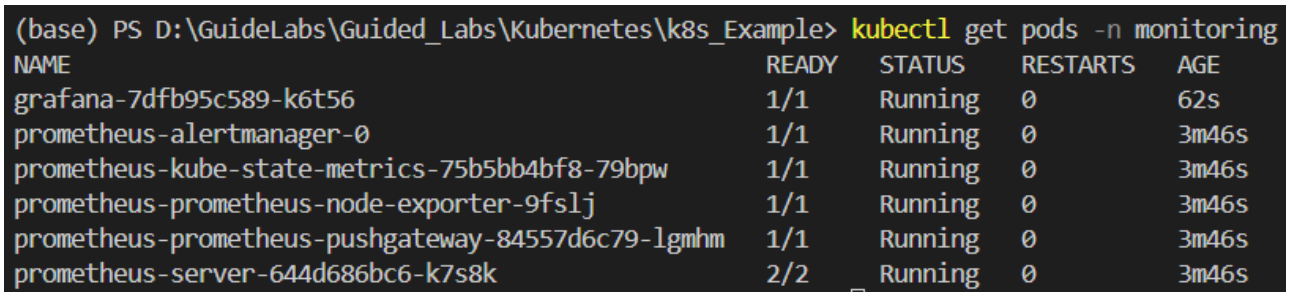
```
helm install grafana grafana/grafana --namespace monitoring
```



   This will deploy Grafana in the same `monitoring` namespace.

3. **Verify Grafana Deployment**
   Check the status of the Grafana pod:

```
kubectl get pods -n monitoring
```

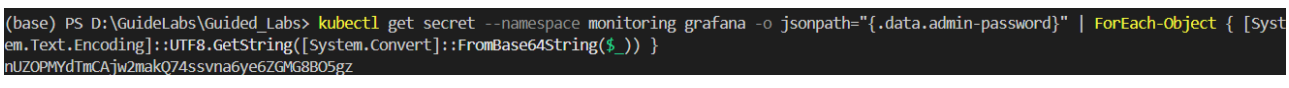


   You should see a `grafana` pod running.

4. **Get Grafana Admin Password**

   The default Grafana admin password is stored in a Kubernetes secret. Retrieve it with:

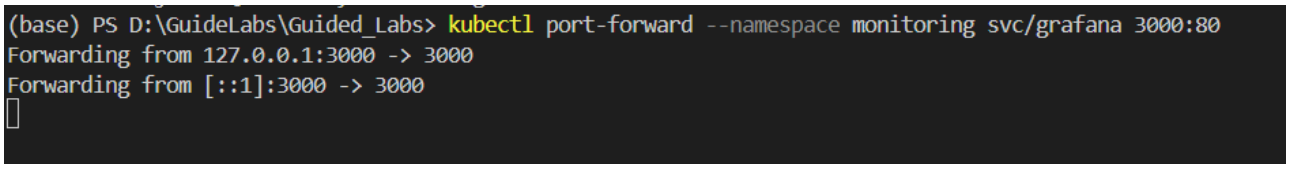
```
kubectl get secret --namespace monitoring grafana -o jsonpath="{.data.admin-password}" | ForEach-Object { [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($_)) }
```



5. **Expose Grafana for External Access**

To access Grafana from your local machine, run:

```
kubectl port-forward --namespace monitoring svc/grafana 3000:80
```



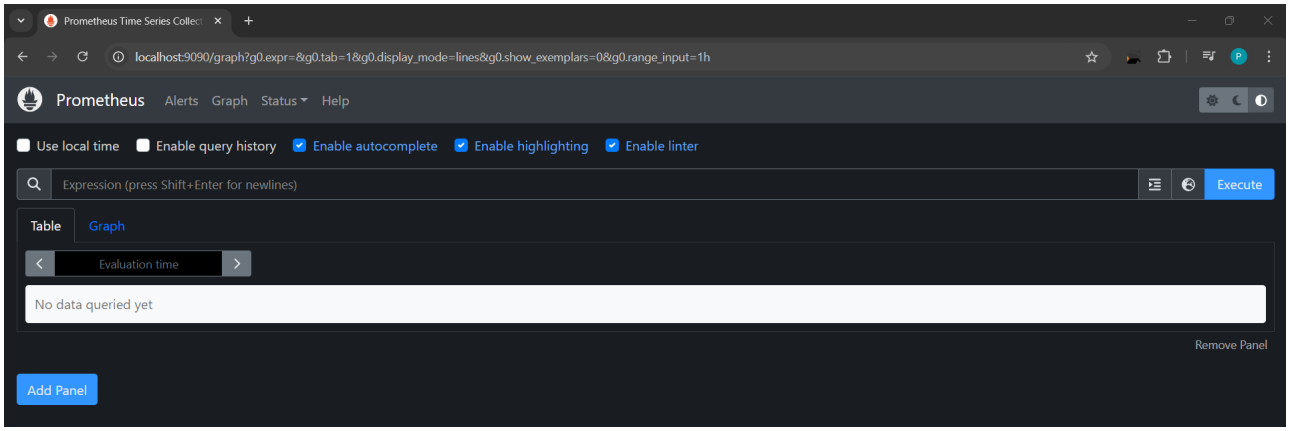
Now, open your browser and navigate to `http://localhost:3000`. The default username is `admin`, and the password is the one you retrieved in the previous step.

Step 3: Access Prometheus and Grafana Dashboards

1. **Access Prometheus UI**

   Forward the Prometheus service to access it locally:

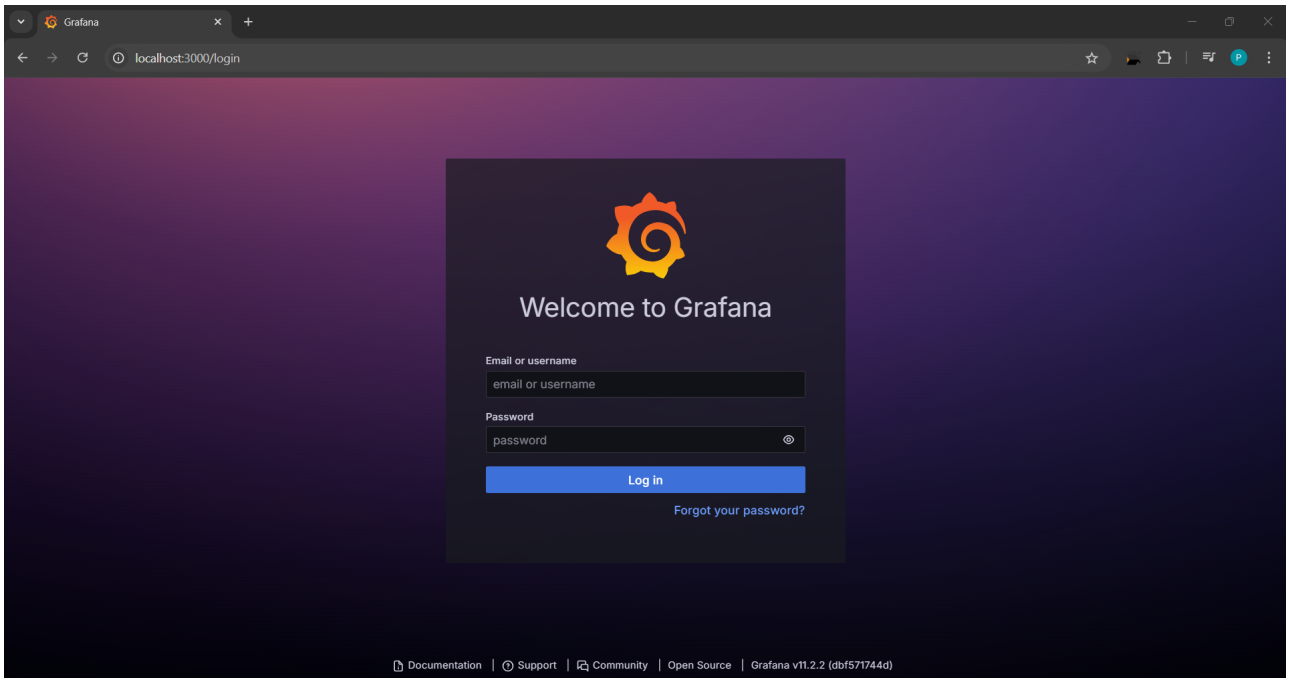
   ```
   kubectl port-forward --namespace monitoring svc/prometheus-server 9090:80
   ```



   Visit `http://localhost:9090` to access the Prometheus dashboard. You can explore the metrics that are being scraped from your cluster.

2. **Access Grafana UI**

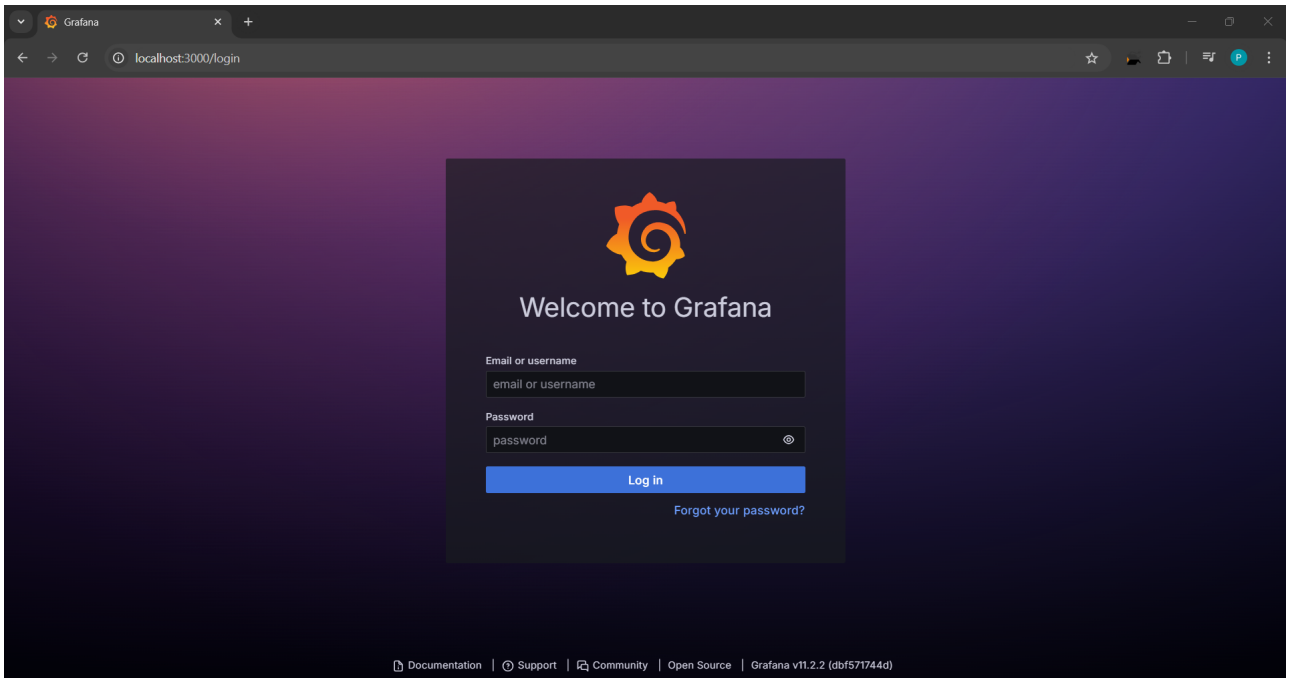
   Open your browser and navigate to `http://localhost:3000`. Log in using your admin credentials (username: `admin`, password: the one you retrieved earlier).



Step 4: Configure Grafana Dashboards for Kubernetes Metrics

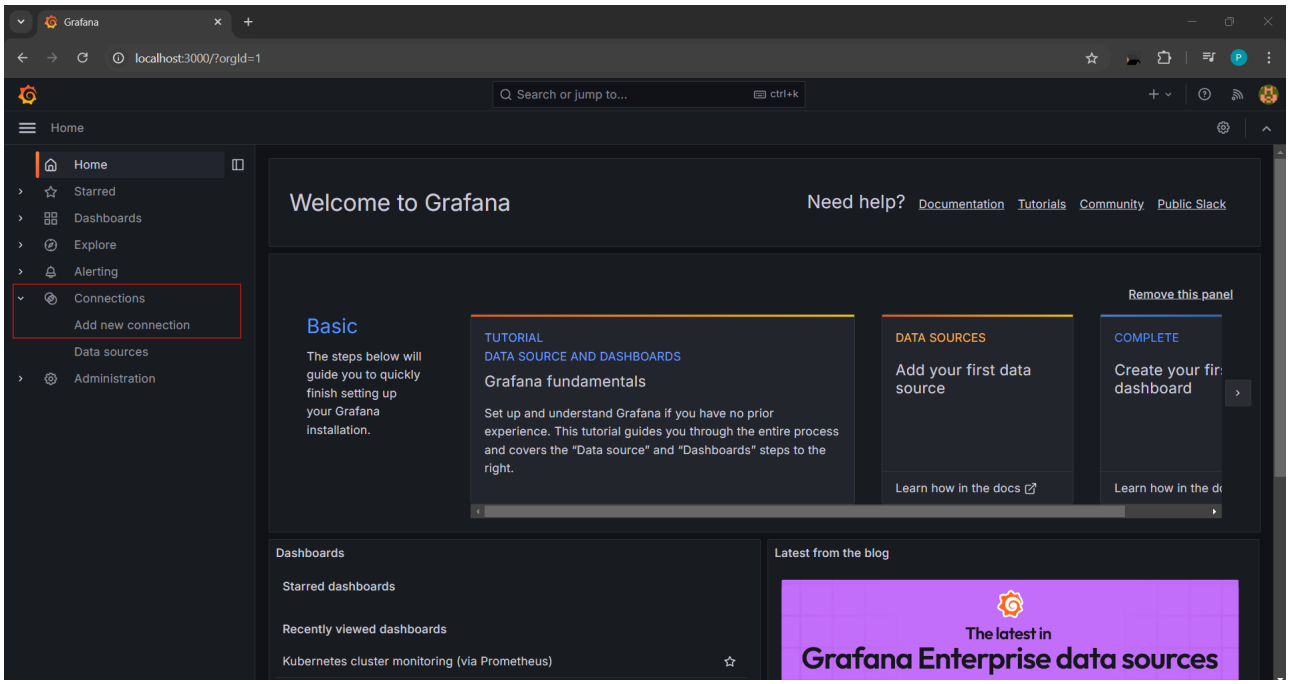
1. **Log in to Grafana**

   Open your browser and navigate to `http://localhost:3000`. Log in using your admin credentials (username: `admin`, password: the one you retrieved earlier).



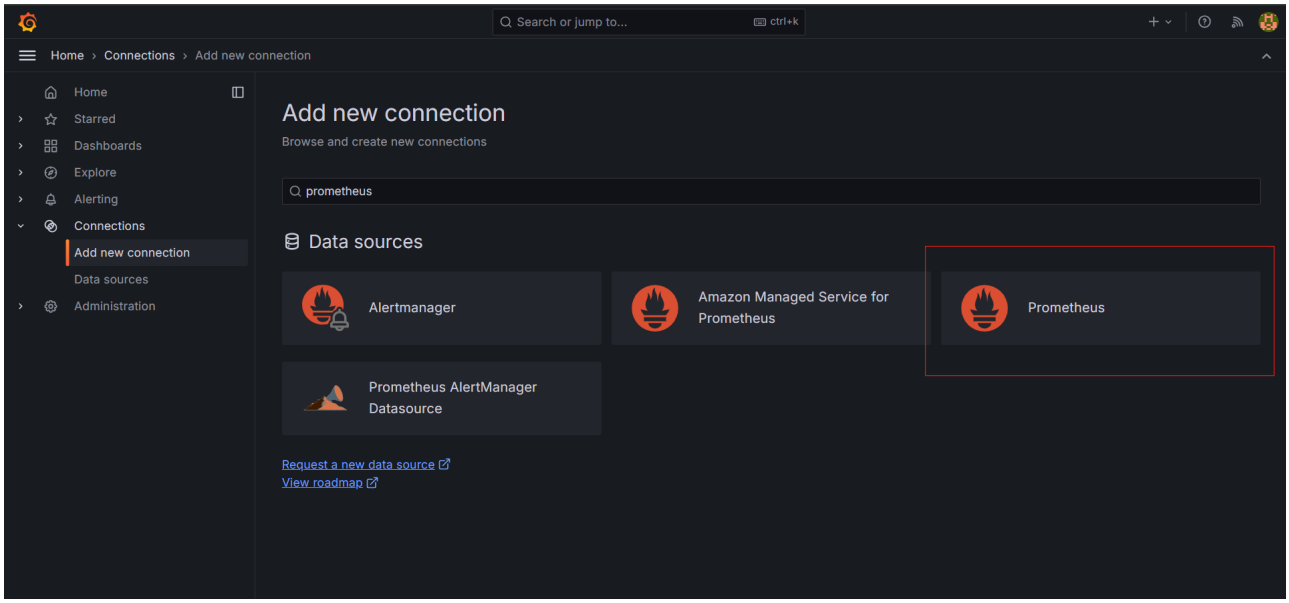
2. **Navigate to "Connections" Section**

   - From the left-hand menu, click on **Connections** (this might be called "Connections" or "Data Sources" depending on your version).
   - In the **Connections** tab, click on **Add new connection**.



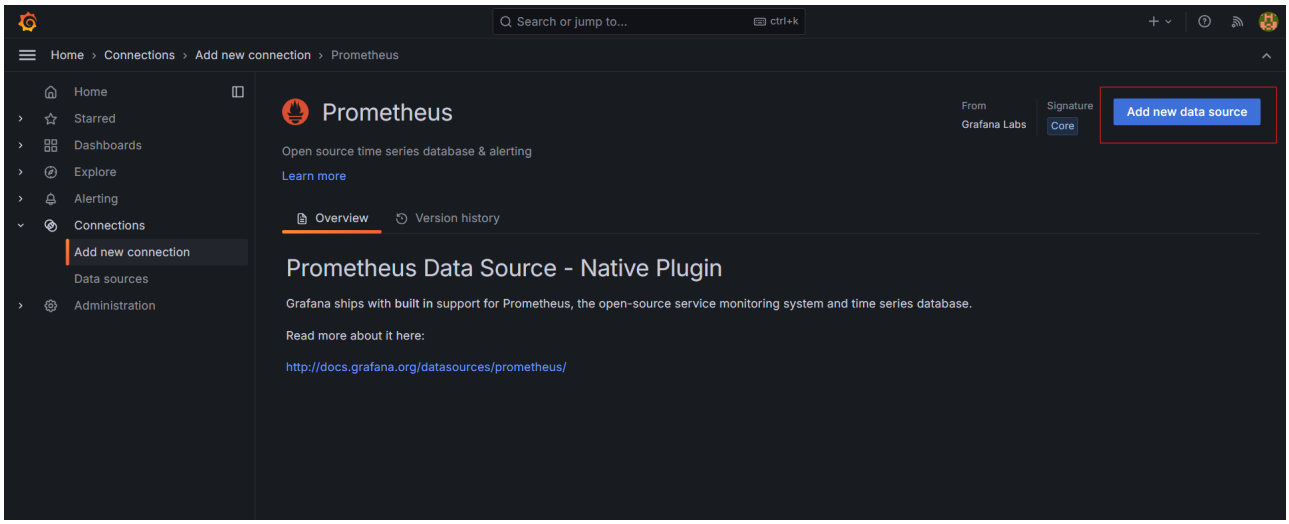
3. **Search for Prometheus**

   - In the search bar that appears, type **Prometheus**.
   - Select **Prometheus** from the list of available data sources.



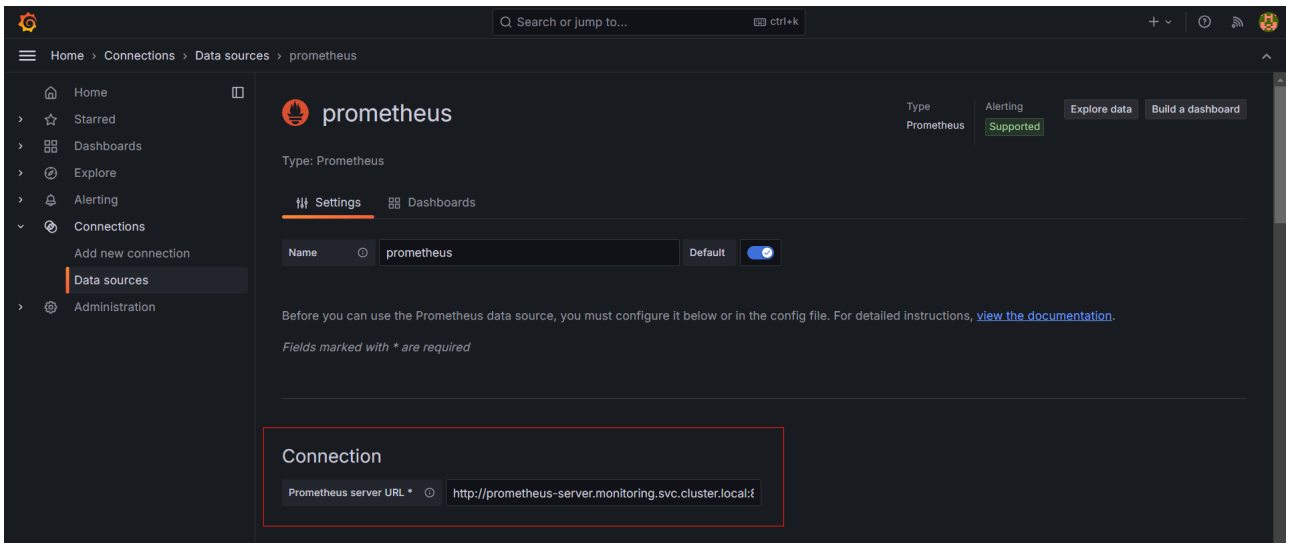
4. **Configure the Prometheus Data Source**

   - On the right corner of the screen, click **Add new data source**.



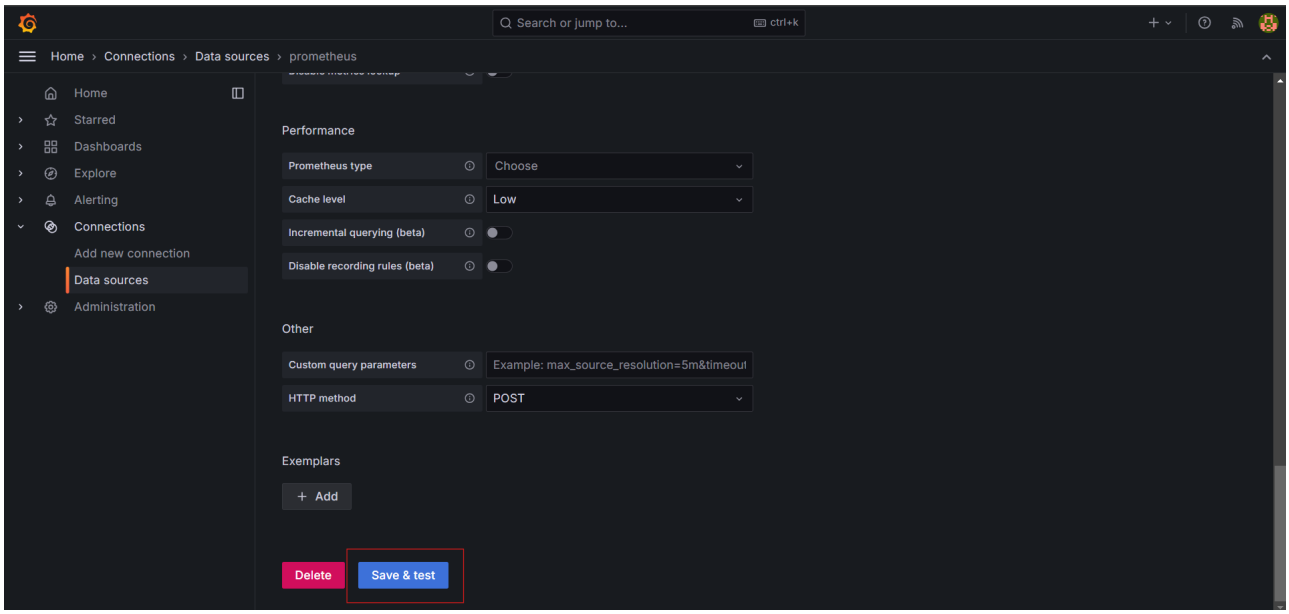
- In the **HTTP URL** field (under **Connection details**),

  If you are accessing Prometheus via its service within the cluster, you should use the service URL instead: `http://prometheus-server.monitoring.svc.cluster.local:80`.

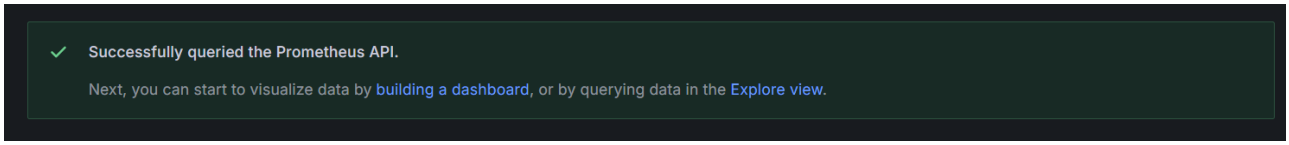


5. **Save & Test**

   - Scroll to the bottom of the page and click **Save & Test** to verify the connection to Prometheus.



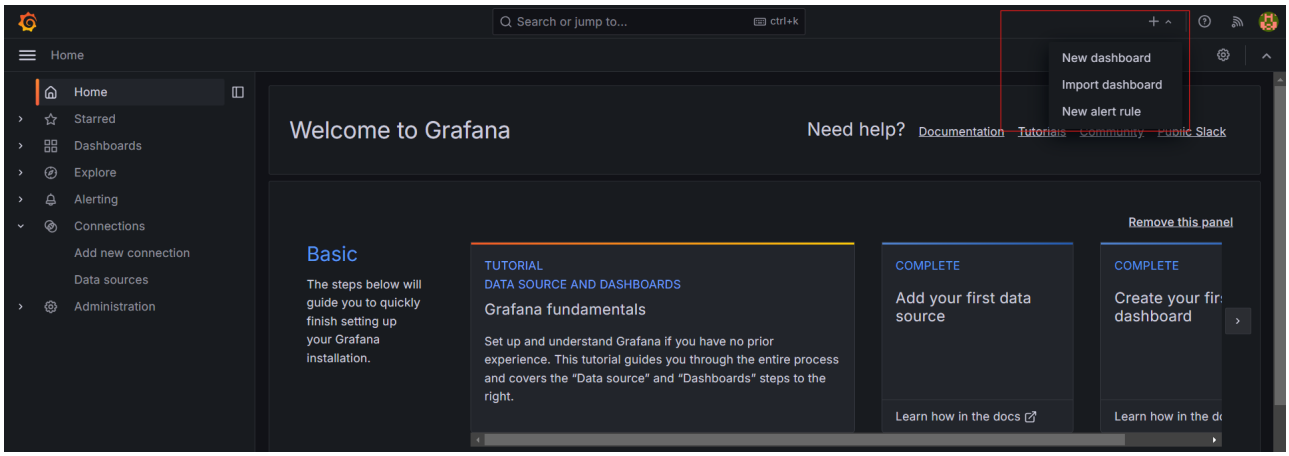
- You should see a success message indicating that Grafana successfully connected to Prometheus.



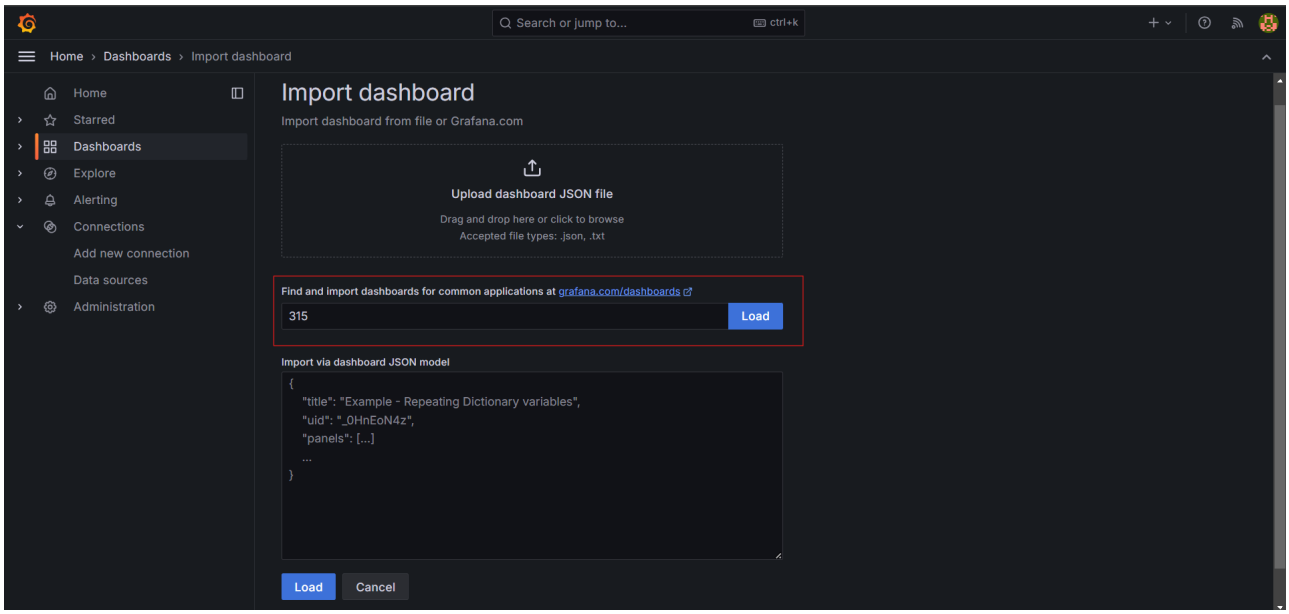
6. **Import Pre-built Kubernetes Dashboards**

Grafana has many pre-built Kubernetes monitoring dashboards. To import one:

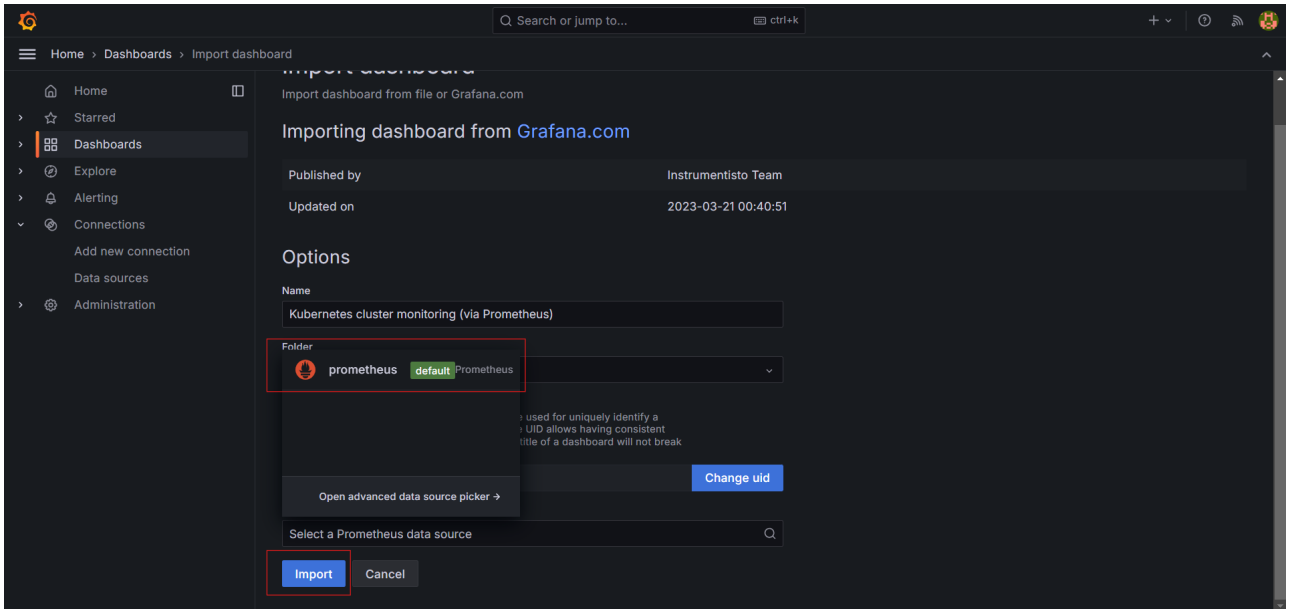
- In the Grafana UI, click **+ > Import Dashboard**.



- Use the following dashboard ID: `315` and click `load`. This is a popular Kubernetes cluster monitoring dashboard.

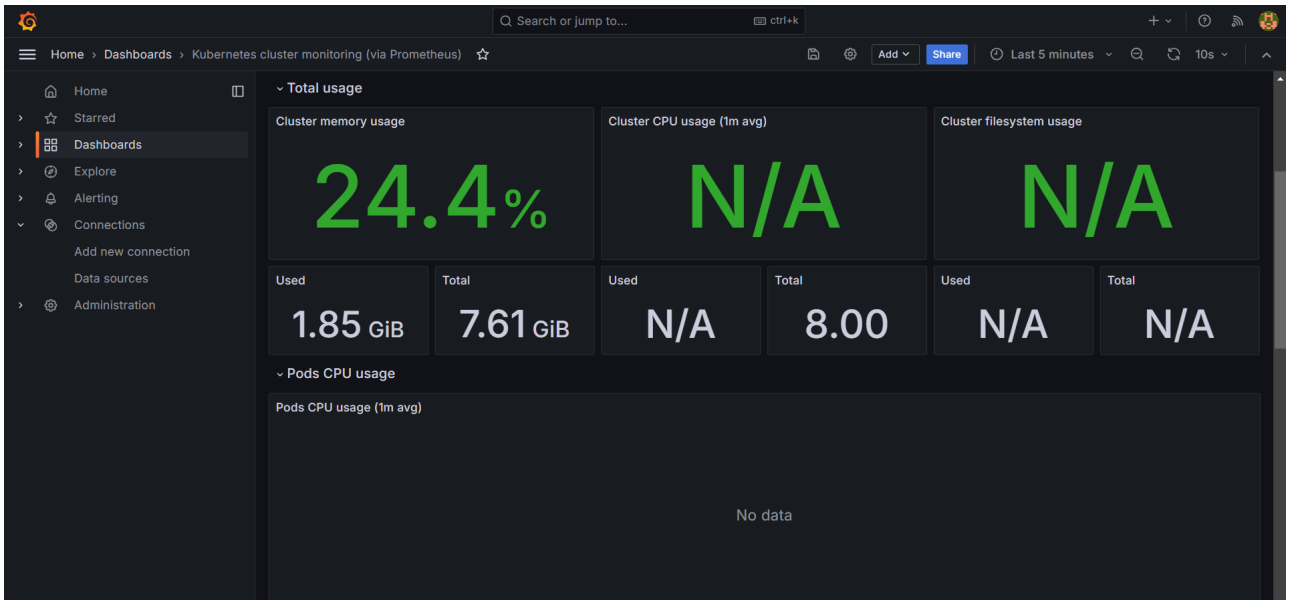


- select the Prometheus data source, and click **Import**.



7. **View Your Kubernetes Metrics**

   After importing the dashboard, you'll be able to see metrics like CPU usage, memory utilization, pod status, and more for your Kubernetes cluster.



References

- Prometheus Documentation
- Grafana Documentation
- Helm Documentation
