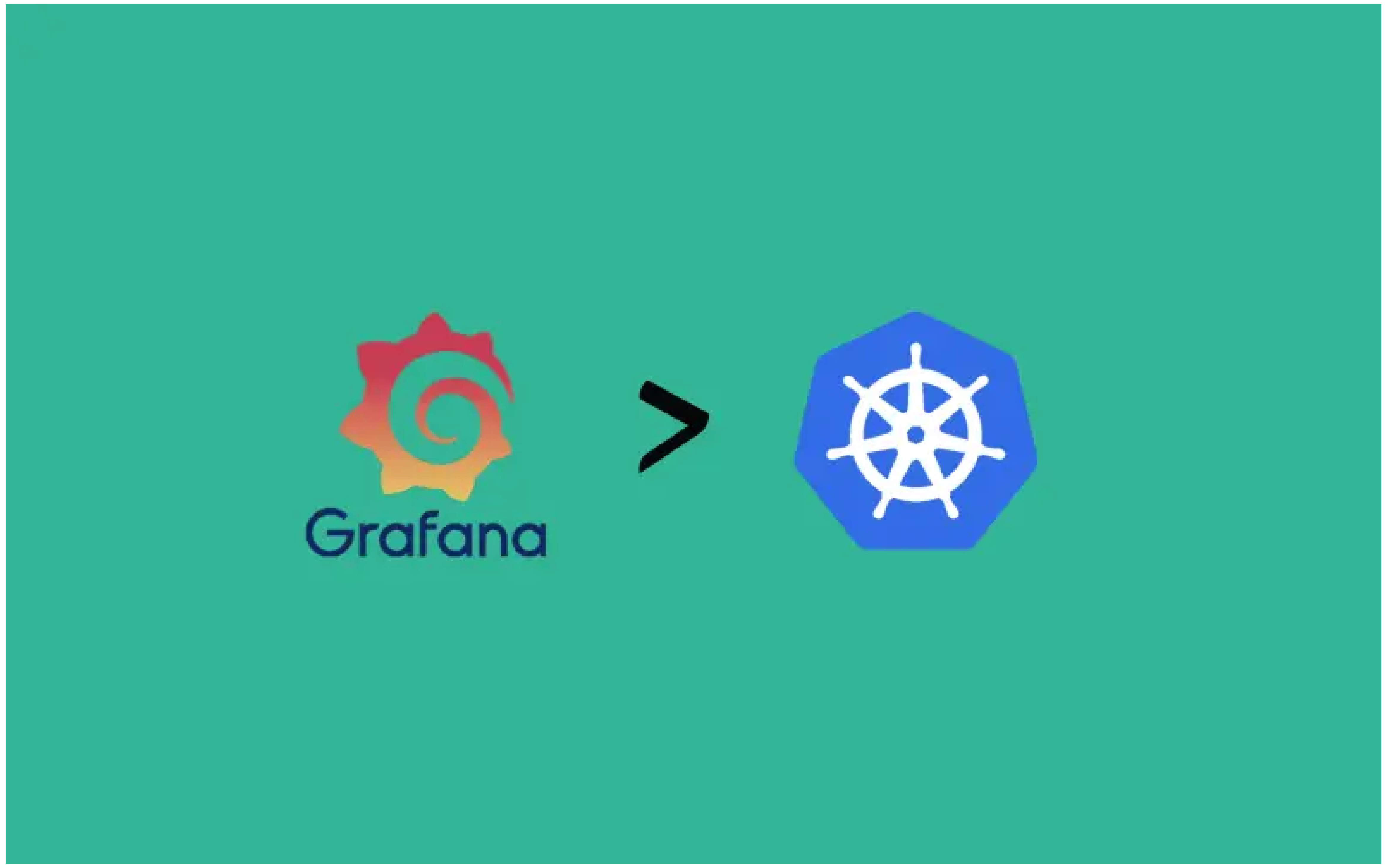
H — HOW TO

# How To Setup Grafana On Kubernetes

by **Bibin Wilson** · April 23, 2021



Grafana is an open-source lightweight dashboard tool. It can be integrated with many data sources like Prometheus, AWS cloud watch, Stackdriver, etc. Running Grafana on Kubernetes

When Grafana is used with Prometheus, it caused PromQL to query metrics from Prometheus

In our previous posts, we have looked at the following.

This tutorial explains how to run Grafana on Kubernetes cluster. Using Grafana you can simplify Kubernetes monitoring dashboards from Prometheus metrics.

## Grafana Kubernetes Manifests

All the Kubernetes manifests (YAML files) used in this tutorial are hosted on Github as well. You can clone it and use it for the setup.

```
git clone https://github.com/bibinwilson/kubernetes-grafana.git
```

## Deploy Grafana On Kubernetes

Let's look at the Grafana setup in detail.

**Step 1:** Create file named `grafana-datasource-config.yaml`

```
vi grafana-datasource-config.yaml
```

Copy the following contents.

**Note:** The following data source configuration is for Prometheus. If you have more data sources, you can add more data sources with different YAMLs under the data section.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: grafana-datasources
  namespace: monitoring
data:
  prometheus.yaml: |-
    {
        "apiVersion": 1,
        "datasources": [
            {
               "access":"proxy",
                "editable": true,
                "name": "prometheus",
                "orgId": 1,
            }
        ]
    }
```

**Step 2:** Create the configmap using the following command.

```
kubectl create -f grafana-datasource-config.yaml
```

**Step 3:** Create a file named `deployment.yaml`

```
vi deployment.yaml
```

Copy the following contents on the file.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: grafana
  namespace: monitoring
spec:
  replicas: 1
  selector:
    matchLabels:
      app: grafana
  template:
    metadata:
      name: grafana
      labels:
        app: grafana
    spec:
      containers:
      - name: grafana
        image: grafana/grafana:latest
        ports:
        - name: grafana
          containerPort: 3000
        resources:
          limits:
            memory: "1Gi"
            cpu: "1000m"
          requests:
            memory: 500M
            cpu: "500m"
        volumeMounts:
          - mountPath: /var/lib/grafana
            name: grafana-storage
          - mountPath: /etc/grafana/provisioning/datasources
            name: grafana-datasources
            readOnly: false
      volumes:
        - name: grafana-storage
          emptyDir: {}
        - name: grafana-datasources
          configMap:
              defaultMode: 420
              name: grafana-datasources
```

if you are deploying Grafana for your project requirements. It will persist all the configs and data that Grafana uses.

**Step 4:** Create the deployment

```
kubectl create -f deployment.yaml
```

**Step 5:** Create a service file named `service.yaml`

```
vi service.yaml
```

Copy the following contents. This will expose Grafana on `NodePort` 32000. You can also expose it using ingress or a Loadbalancer based on your requirement.

```
apiVersion: v1
kind: Service
metadata:
  name: grafana
  namespace: monitoring
  annotations:
      prometheus.io/scrape: 'true'
      prometheus.io/port:   '3000'
spec:
  selector:
    app: grafana
  type: NodePort
  ports:
    - port: 3000
      targetPort: 3000
      nodePort: 32000
```

**Step 6:** Create the service.

```
kubectl create -f service.yaml
```

Now you should be able to access the Grafana dashboard **using any node IP** on port `32000`. Make sure the port is allowed in the firewall to be accessed from your workstation.

```
http://<your-node-ip>:32000
```

You can also use port forwarding using the following command.

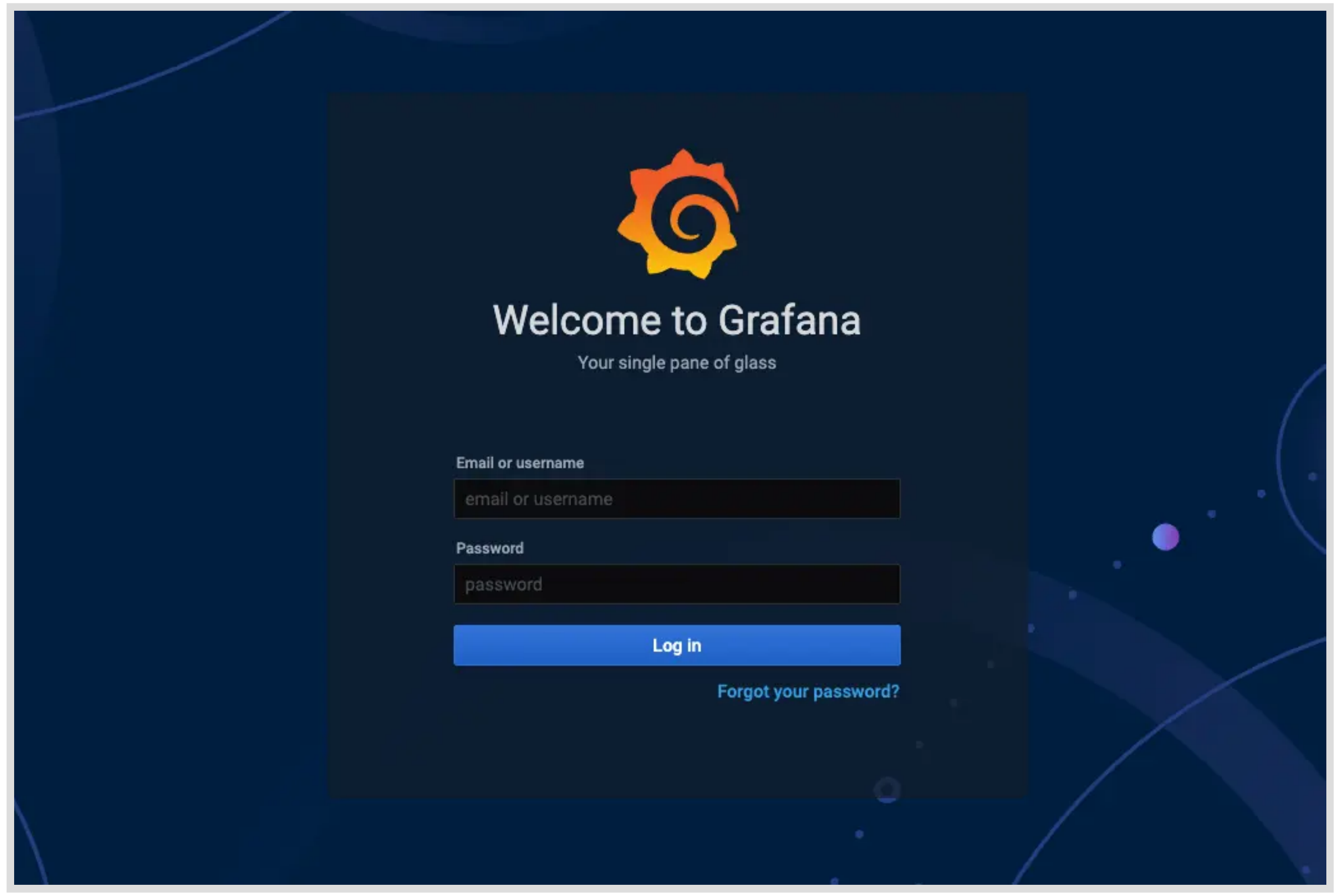
```
kubectl port-forward -n monitoring <grafana-pod-name> 3000 &
```

```
vagrant@dcubelab:~$ kubectl get po -n monitoring
NAME                       READY   STATUS    RESTARTS   AGE
grafana-64c89f57f7-kjqrb   1/1     Running   0          10m
vagrant@dcubelab:~$ kubectl port-forward -n monitoring grafana-64c89f57f7-kjqrb
3000 &
```

You will be able to access Grafana a from `http://localhost:3000`

Use the following default username and password to log in. Once you log in with default credentials, it will prompt you to change the default password.

```
User: admin
Pass: admin
```

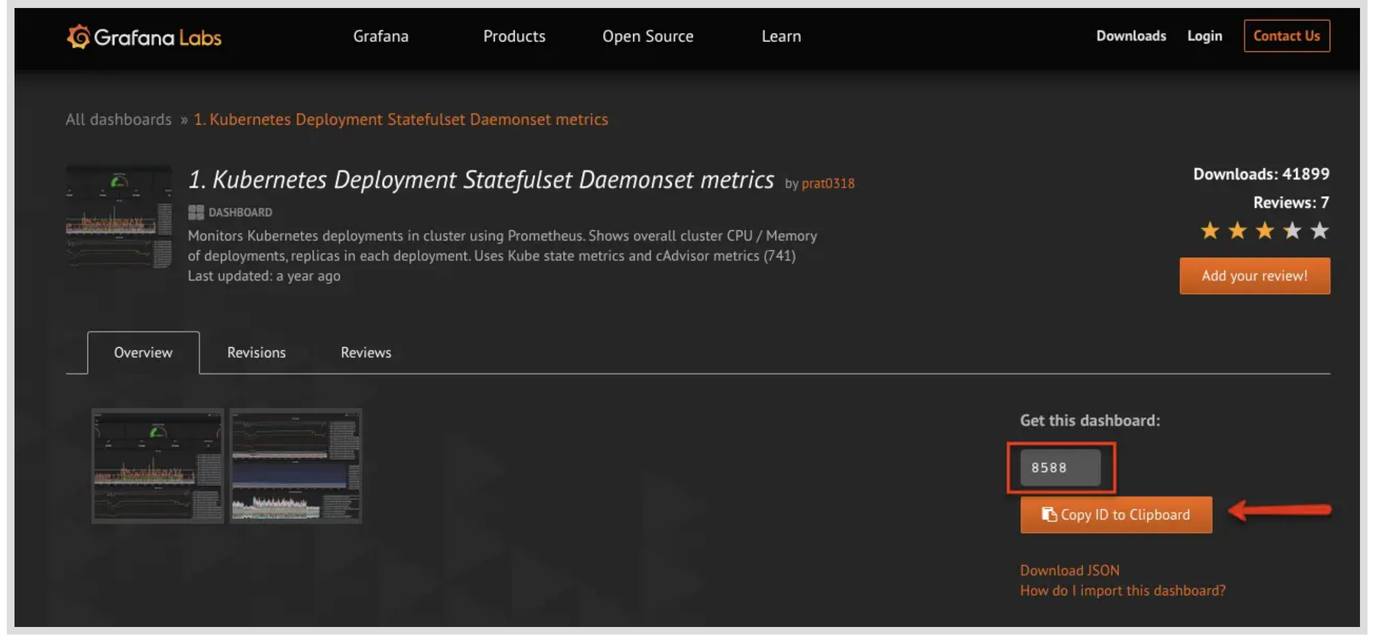


## Setup Kubernetes Dashboards on Grafana

There are many prebuilt Grafana templates available for Kubernetes. To know more, see Grafana Kubernetes Dashboard templates

Setting up a dashboard from a template is pretty easy. Follow the steps given below to set up a Grafana dashboard to monitor kubernetes deployments.

**Step 1:** Get the template ID from grafana public template. as shown below.



All dashboards » 1. Kubernetes Deployment Statefulset Daemonset metrics

1. Kubernetes Deployment Statefulset Daemonset metrics by prat0318

DASHBOARD

Monitors Kubernetes deployments in cluster using Prometheus. Shows overall cluster CPU / Memory of deployments, replicas in each deployment. Uses Kube state metrics and cAdvisor metrics (741)

Last updated: a year ago

Downloads: 41899

Reviews: 7

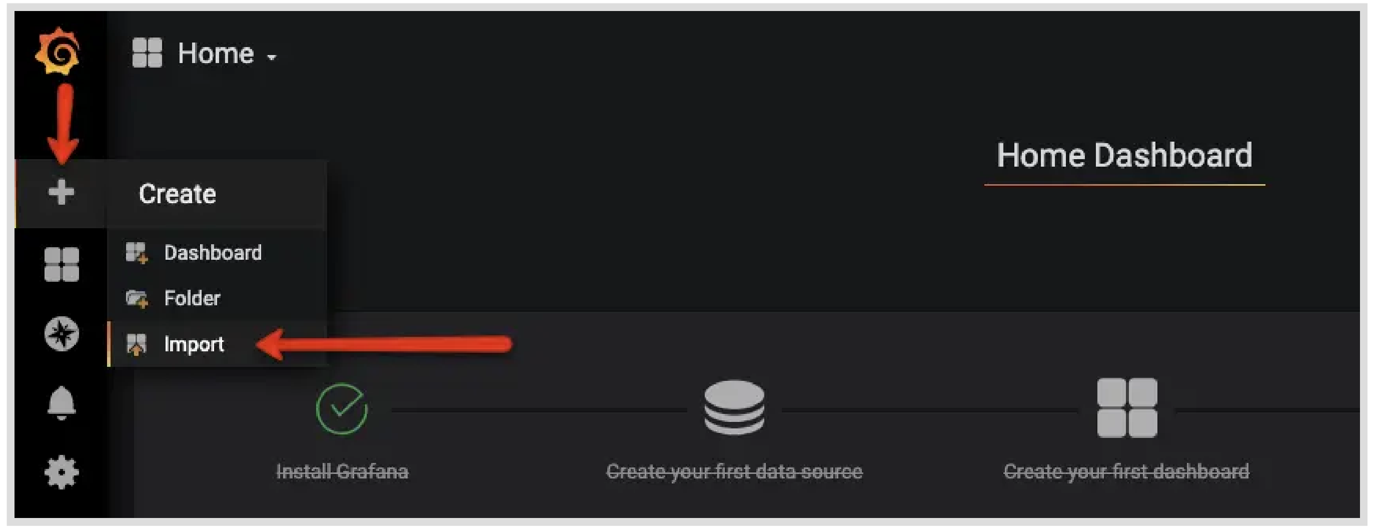
Get this dashboard:

8588

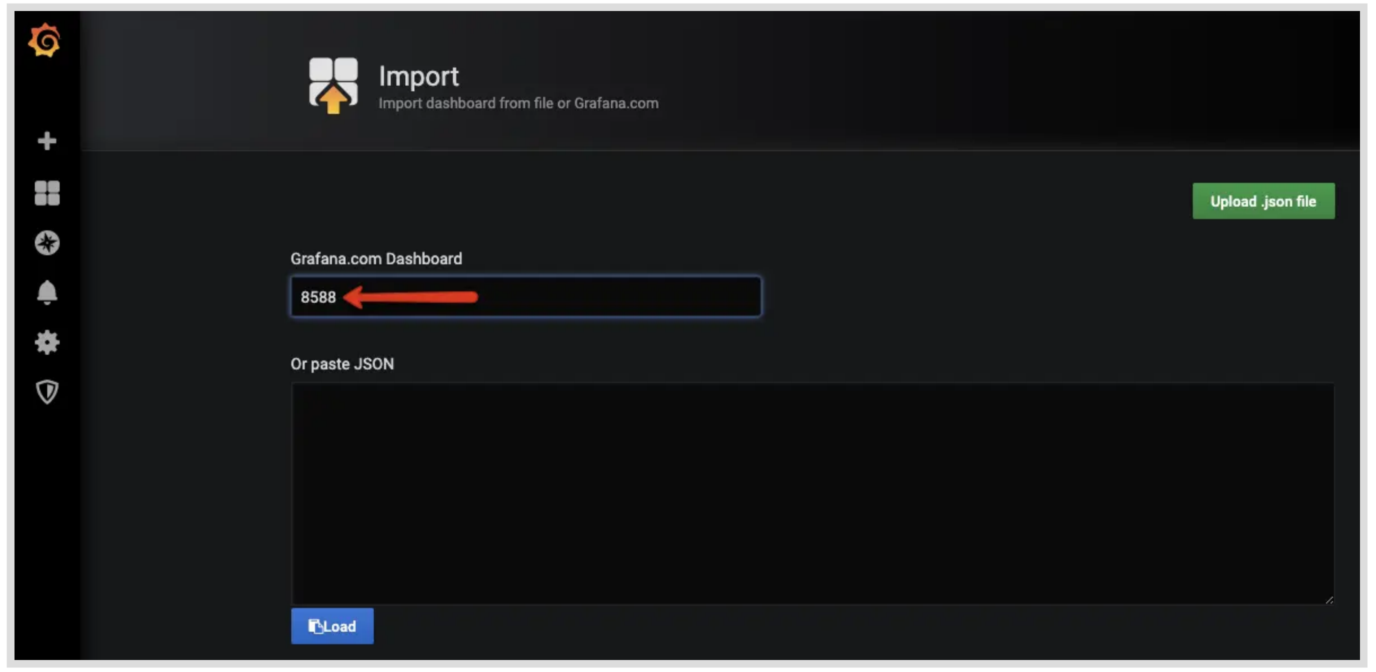
Download JSON

How do I import this dashboard?

**Step 2:** Head over to grafana and select the import option.


Home
Home Dashboard
Create
Dashboard
Folder
Import
~~Install Grafana~~
~~Create your first data source~~
~~Create your first dashboard~~

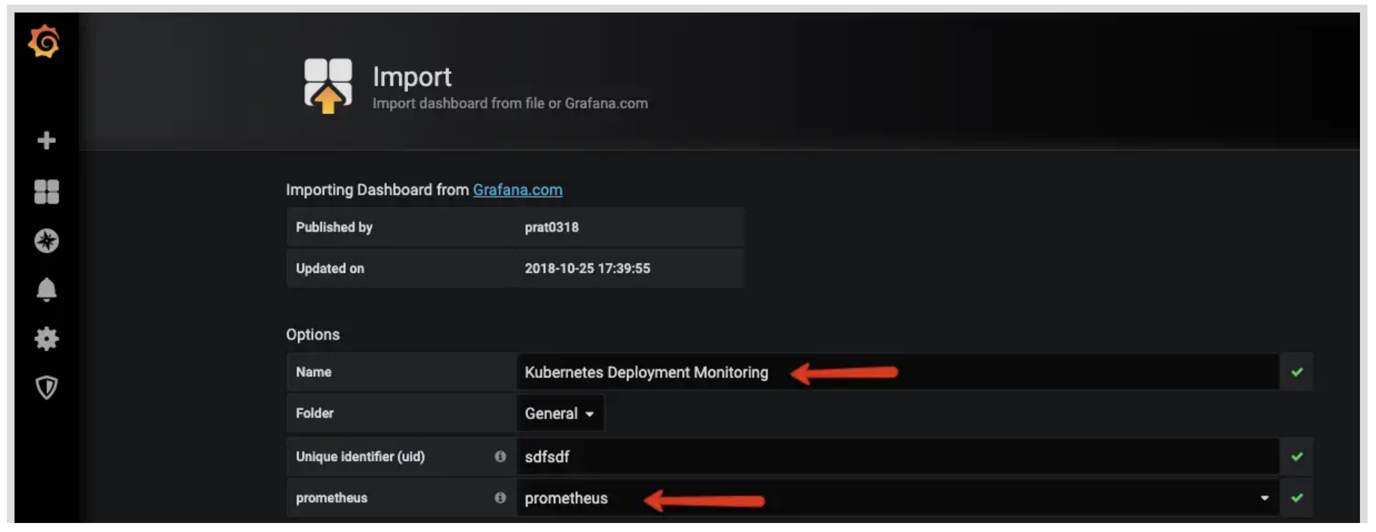
**Step 3:** Enter the dashboard ID you got it step 1



Grafana.com Dashboard

8588

**Step 4:** Grafana will automatically fetch the template from Grafana website. You can change the values as shown in the image below and click import.



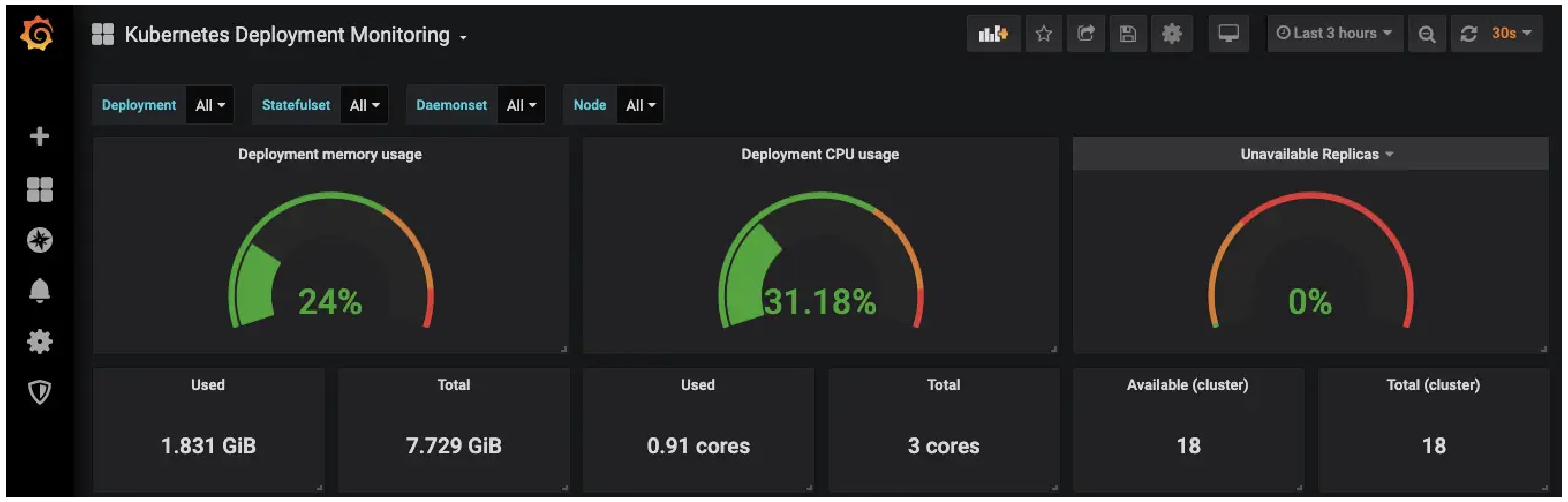
Importing Dashboard from Grafana.com

| | |
|---|---|
| Published by | prat0318 |
| Updated on | 2018-10-25 17:39:55 |

Options

| | |
|---|---|
| Name | Kubernetes Deployment Monitoring |
| Folder | General |
| Unique identifier (uid) | sdfsdf |
| prometheus | prometheus |

You should see the dashboard immediately.



## Conclusion

Grafana is a very powerful tool when it comes to monitoring dashboards.

It is used by many organizations to monitor their workloads.

Let us know how you are using Grafana in your organization. Also, let us know if you want to add more information to this article.

**TAGS:** Grafana Kubernetes Prometheus

Bibin Wilson

An author, blogger and DevOps practitioner. In spare time, he loves to try out the latest open source technologies. He works as an Associate Technical Architect

by **devopscube** · April 6, 2021

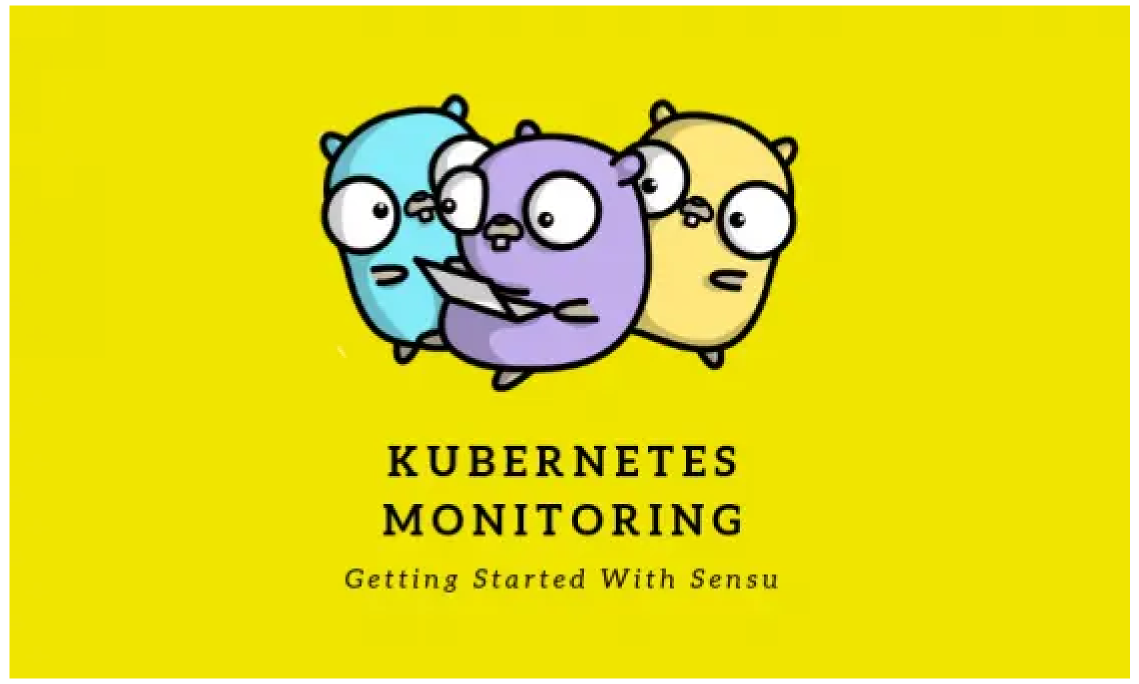
If you want to know how the Kubernetes nodes perform or monitor system-level insights of kubernetes nodes, you...

D — DEVOPS

## How To Create Jenkins Shared Library For Pipelines

by **devopscube** · July 7, 2019

In this tutorial, you will learn how to create a basic shared library and integrate it with Jenkins...



K — KUBERNETES

## Kubernetes monitoring With Sensu: Setting up Container Sidecar Agent

by **Jef Spaleta** · August 27, 2019

The rise of containerized infrastructure has caused us to rethink the way we build and deploy our applications....

K — KUBERNETES

## How To Setup Kubernetes Cluster On Google Cloud (GKE)

by **Bibin Wilson** · June 13, 2021

This guide walks you through deploying a Kubernetes Cluster on google cloud using the Google Kubernetes Engine (GKE)....

D — DEVOPS

## – Managed Cloud FileStore Service

by **devopscube** · August 16, 2018

Google Cloud Filestore is a managed NFS implementation on google cloud. This is one of the awaited features…

C — CLOUD

## How to Setup a Replicated GlusterFS Cluster on AWS EC2

by **devopscube** · October 6, 2016

GlusterFS  is one of the best open source distributed file systems. If you want a highly available distributed…

DevopsCube

Privacy Policy About Site Map Disclaimer

Contribute Advertise Archives