K8s Troubleshooting Deployments

Three steps to troubleshoot Kubernetes deployments

Since there are three components in every deployment, you should debug all of them in order, starting from the bottom.

1. You should make sure that your Pods are running, then
2. Focus on getting the Service to route traffic to the Pods and then
3. Check that the Ingress is correctly configured

Common Pods errors

Pods can have startup and runtime errors.

Startup errors include:

- ImagePullBackoff
- ImageInspectError
- ErrImagePull
- ErrImageNeverPull
- RegistryUnavailable
- InvalidImageName

Runtime errors include:

- CrashLoopBackOff
- RunContainerError
- KillContainerError
- VerifyNonRootError
- RunInitContainerError
- CreatePodSandboxError
- ConfigPodSandboxError
- KillPodSandboxError
- SetupNetworkError
- TeardownNetworkError

Troubleshooting Services

Services are designed to route the traffic to Pods based on their labels.
So the first thing that you should check is how many Pods are targeted by the Service.
You can do so by checking the Endpoints in the Service:

```
kubectl describe service <service-name> | grep Endpoints
```

An endpoint is a pair of <ip address:port>, and there should be at least one — when the Service targets (at least) a Pod.

If the "Endpoints" section is empty, there are two explanations:

1. You don't have any Pod running with the correct label (hint: you should check if you are in the right namespace)
2. You have a typo in the selector labels of the Service

Troubleshooting Ingress

If you have reached this section, then:

- the Pods are *Running* and *Ready*
- the Service distributes the traffic to the Pod

But you still can't see a response from your app.
It means that most likely, the Ingress is misconfigured.

The Ingress uses the serviceName and servicePort to connect to the Service.
You should check that those are correctly configured.
You can inspect that the Ingress is correctly configured with:

```
kubectl describe ingress <ingress-name>
```

If the *Backend* column is empty, then there must be an error in the configuration.
You can isolate infrastructure issues from Ingress by connecting to the Ingress Pod directly.
First, retrieve the Pod for your Ingress controller (which could be located in a different namespace):

```
kubectl get pods --all-namespaces
```

Describe it to retrieve the port:

```
kubectl describe pod nginx-ingress-controller-6fc5bcc
```

Finally, connect to the Pod:

```
kubectl port-forward nginx-ingress-controller-6fc5bcc 3000:80 --namespace kube-system
```

At this point, every time you visit port 3000 on your computer, the request is forwarded to port 80 on the Pod.
Does it works now?

- If it works, the issue is in the infrastructure. You should investigate how the traffic is routed to your cluster.
- If it doesn't work, the problem is in the Ingress controller. You should debug the Ingress.

Since Ingress Nginx is the most popular Ingress controller, we included a few tips for it in the next section.

Debugging Ingress Nginx

The Ingress-nginx project has an official plugin for Kubectl.

You can use `kubectl ingress-nginx` to:

- inspect logs, backends, certs, etc.
- connect to the Ingress
- examine the current configuration

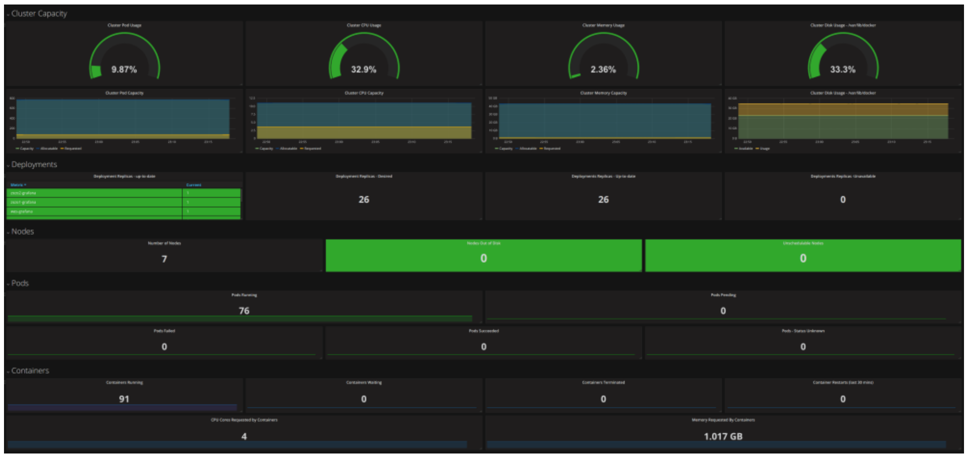
The three commands that you should try are:

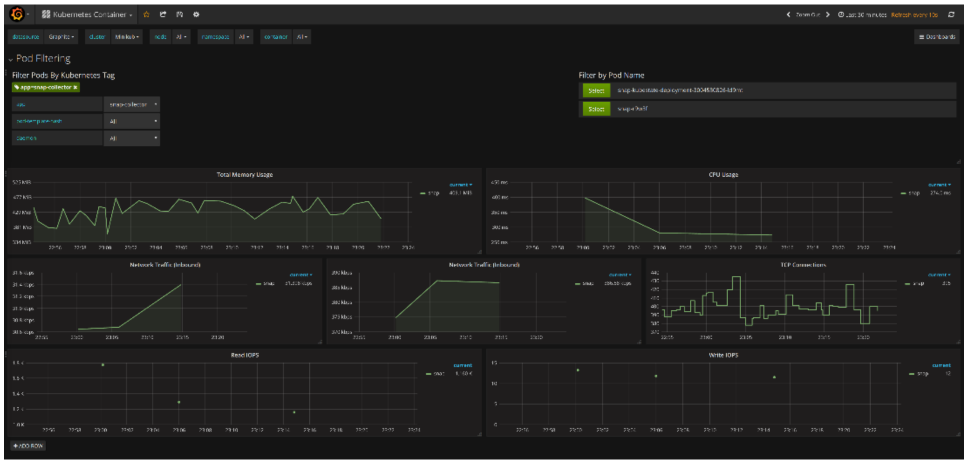
- kubectl ingress-nginx lint, which checks the nginx.conf
- kubectl ingress-nginx backend, to inspect the backend (similar to kubectl describe ingress <ingress-name>)
- kubectl ingress-nginx logs, to check the logs

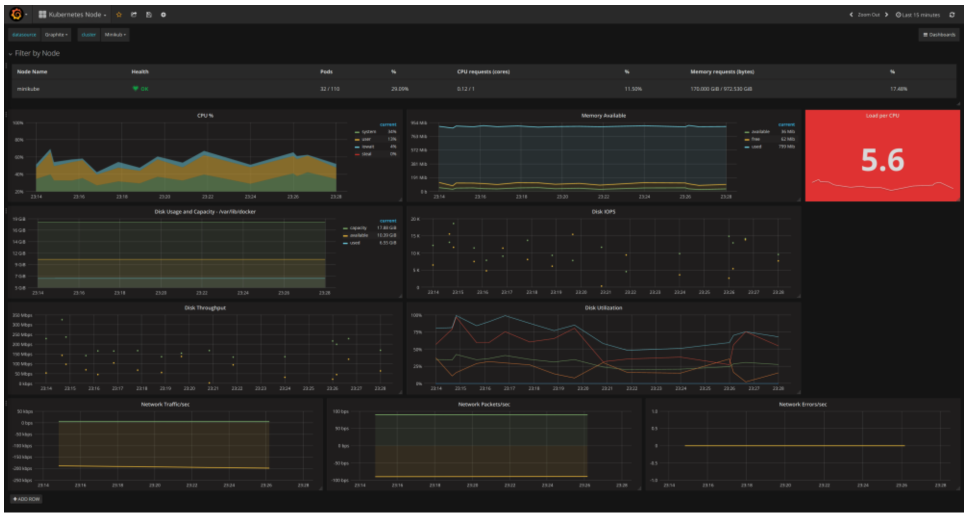
How to monitor Kubernetes with Grafana

Grafana App for Kubernetes

The Grafana Kubernetes App allows you to monitor your Kubernetes cluster's performance. It includes four dashboards, Cluster, Node, Pod/Container and Deployment. It allows for the automatic deployment of the required Prometheus exporters and a default scrape config to use with your in cluster Prometheus deployment. The metrics collected are high-level cluster and node stats as well as lower level pod and container stats. Use the high-level metrics to alert on and the low-level metrics to **troubleshoot**.







Node Name	Health	Pods	%	CPU requests (cores)	%	Memory requests (bytes)	%
minikube	OK	32 / 110	29.09%	0.12 / 1	11.50%	170.000 GiB / 972.530 GiB	17.48%

Metrics available

Cluster Metrics

- Pod Capacity/Usage
- Memory Capacity/Usage
- CPU Capacity/Usage
- Disk Capacity/Usage
- Overview of Nodes, Pods and Containers

Node Metrics

- CPU
- Memory Available
- Load per CPU
- Read IOPS
- Write IOPS
- %Util
- Network Traffic/second
- Network Packets/second
- Network Errors/second

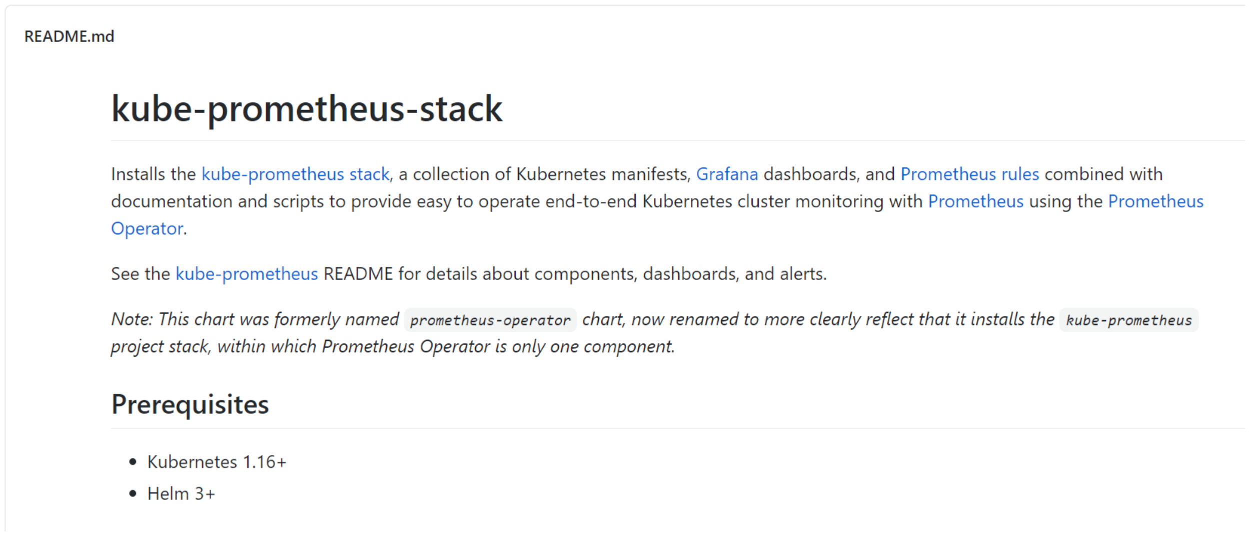
Pod/Container Metrics

- Memory Usage
- Network Traffic
- CPU Usage
- Read IOPS
- Write IOPS

Implementing

I used Minikube and this helm chart to do the deploy of the cluster that includes, Prometheus and Grafana with preloaded dashboard for monitoring the cluster.

https://github.com/prometheus-community/helm-charts/tree/main/charts/kube-prometheus-stack



README.md

kube-prometheus-stack

Installs the kube-prometheus stack, a collection of Kubernetes manifests, Grafana dashboards, and Prometheus rules combined with documentation and scripts to provide easy to operate end-to-end Kubernetes cluster monitoring with Prometheus using the Prometheus Operator.

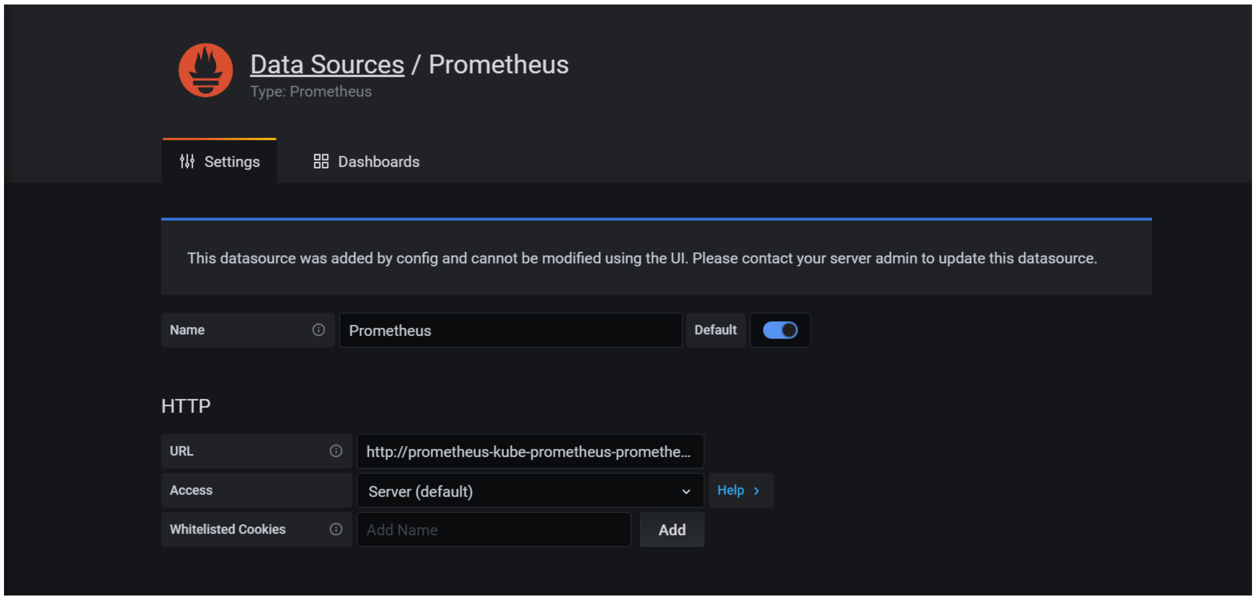
See the kube-prometheus README for details about components, dashboards, and alerts.

Note: This chart was formerly named `prometheus-operator` chart, now renamed to more clearly reflect that it installs the `kube-prometheus` project stack, within which Prometheus Operator is only one component.

Prerequisites

- Kubernetes 1.16+
- Helm 3+

Datasource used



Data Sources / Prometheus
Type: Prometheus

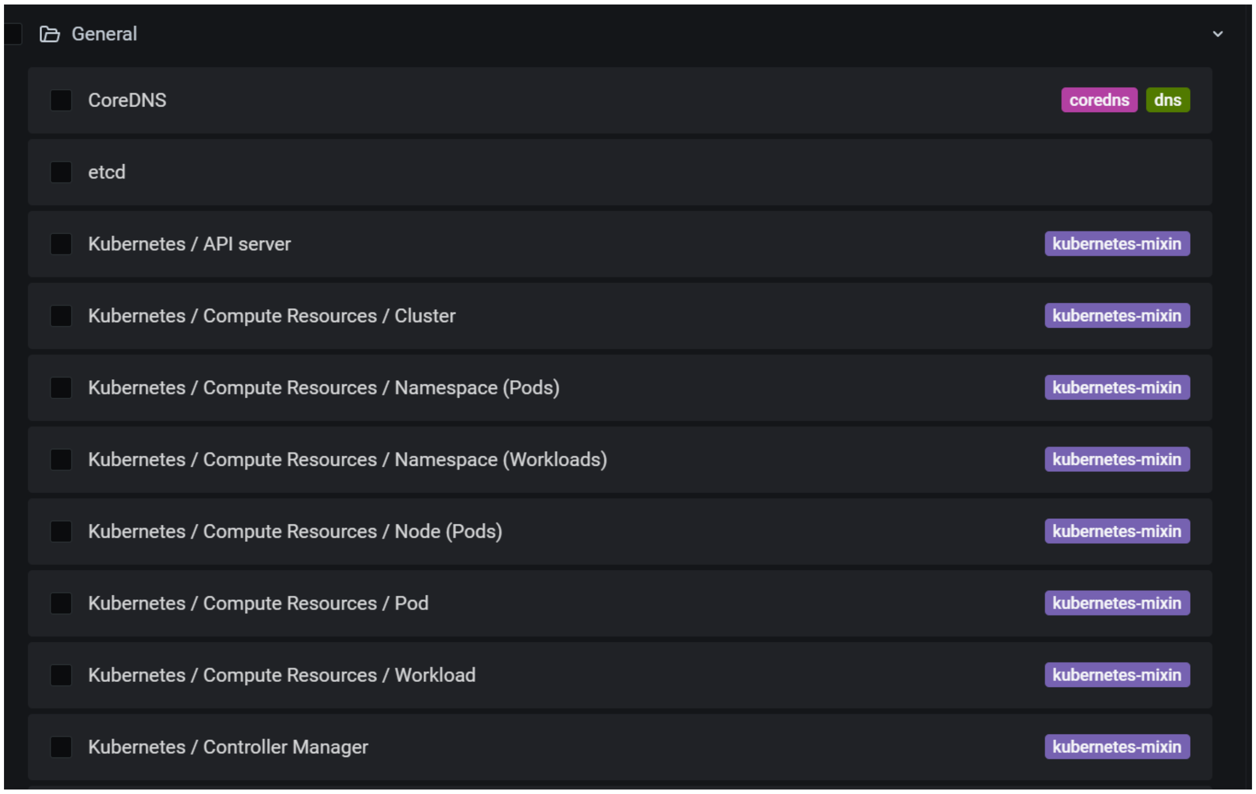
This datasource was added by config and cannot be modified using the UI. Please contact your server admin to update this datasource.

Name: Prometheus Default: on

HTTP

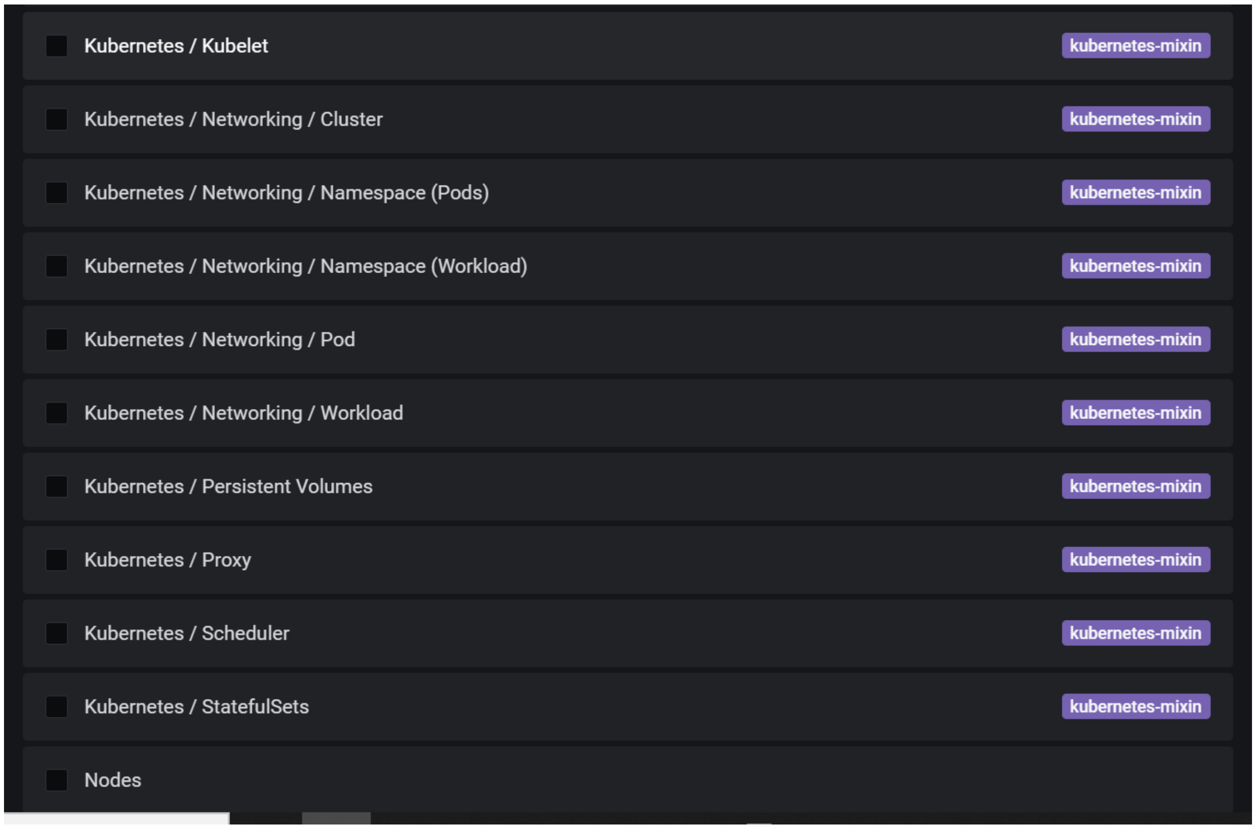
URL: http://prometheus-kube-prometheus-promethe...
Access: Server (default)
Whitelisted Cookies: Add Name

Dashboards included

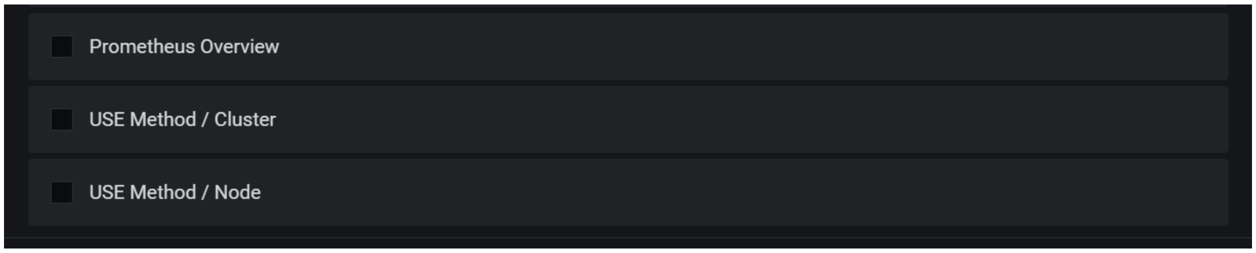


General

CoreDNS	coredns dns
etcd	
Kubernetes / API server	kubernetes-mixin
Kubernetes / Compute Resources / Cluster	kubernetes-mixin
Kubernetes / Compute Resources / Namespace (Pods)	kubernetes-mixin
Kubernetes / Compute Resources / Namespace (Workloads)	kubernetes-mixin
Kubernetes / Compute Resources / Node (Pods)	kubernetes-mixin
Kubernetes / Compute Resources / Pod	kubernetes-mixin
Kubernetes / Compute Resources / Workload	kubernetes-mixin
Kubernetes / Controller Manager	kubernetes-mixin



Kubernetes / Kubelet	kubernetes-mixin
Kubernetes / Networking / Cluster	kubernetes-mixin
Kubernetes / Networking / Namespace (Pods)	kubernetes-mixin
Kubernetes / Networking / Namespace (Workload)	kubernetes-mixin
Kubernetes / Networking / Pod	kubernetes-mixin
Kubernetes / Networking / Workload	kubernetes-mixin
Kubernetes / Persistent Volumes	kubernetes-mixin
Kubernetes / Proxy	kubernetes-mixin
Kubernetes / Scheduler	kubernetes-mixin
Kubernetes / StatefulSets	kubernetes-mixin
Nodes	



☐ Prometheus Overview
☐ USE Method / Cluster
☐ USE Method / Node