

Kubernetes Monitoring on Minikube using Prometheus, Grafana & New Relic

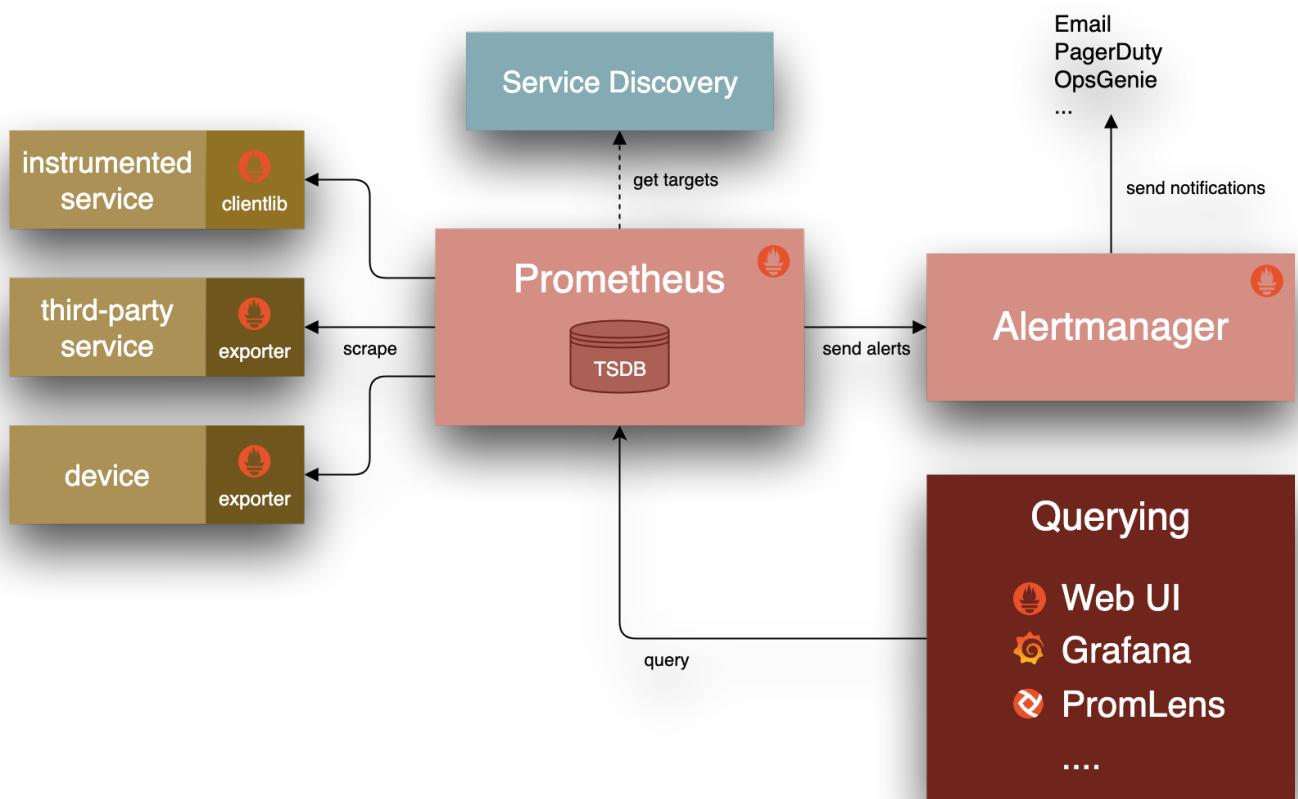
Project Overview

This project showcases how to set up monitoring stack using Newrelic, Prometheus and Grafana on a local Kubernetes cluster created with Minikube(using Docker Desktop), Installing monitoring tools using Helm.

Prerequisites

1. Minikube
2. Docker Desktop
3. Minikube
4. Kubectl (kubernetes CLI)
5. Helm
6. Newrelic account

Prometheus & Grafana:



source: web

Prometheus

Step-by-Step Setup

1. Create a Minikube Cluster (linux/macOS - Docker Driver)

```
minikube start --driver=docker
```

```
[akhilrao@Akhils-MacBook-Air prom % minikube start
  minikube v1.35.0 on Darwin 15.5 (arm64)
  Automatically selected the docker driver
  Using Docker Desktop driver with root privileges
  Starting "minikube" primary control-plane node in "minikube" cluster
  Pulling base image v0.0.46 ...
  Creating docker container (CPUs=2, Memory=2200MB) ...
  Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
    ■ Generating certificates and keys ...
    ■ Booting up control plane ...
    ■ Configuring RBAC rules ...
  ⚡ Configuring bridge CNI (Container Networking Interface) ...
  🔍 Verifying Kubernetes components...
    ■ Using image gcr.io/k8s-minikube/storage-provisioner:v5
  🌟 Enabled addons: storage-provisioner, default-storageclass
  🎉 Done! kubectl is now configured to use "minikube" cluster and "default" name
space by default
[akhilrao@Akhils-MacBook-Air prom % kubectl get pods
  No resources found in default namespace.
[akhilrao@Akhils-MacBook-Air prom % kubectl get pods -a
  error: unknown shorthand flag: 'a' in -a
  See 'kubectl get --help' for usage.
[akhilrao@Akhils-MacBook-Air prom % kubectl get pods -A
  NAME          READY   STATUS    RESTARTS   AGE
  kube-system   coredns-668d6bf9bc-gks4w   1/1     Running   0          88s
  kube-system   etcd-minikube             1/1     Running   0          94s
  kube-system   kube-apiserver-minikube  1/1     Running   0          94s
  kube-system   kube-controller-manager-minikube  1/1     Running   0          94s
  kube-system   kube-proxy-hn427         1/1     Running   0          88s
  kube-system   kube-scheduler-minikube  1/1     Running   0          94s
  kube-system   storage-provisioner      1/1     Running   0          92s
```

2. Add the Prometheus Helm Chart & Install

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

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

```
akhilrao@Akhils-MacBook-Air prom % helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo update
"prometheus-community" already exists with the same configuration, skipping
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "prometheus-community" chart repository
Update Complete. *Happy Helmning!*
```

```
akhilrao@Akhils-MacBook-Air prom % helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "prometheus-community" chart repository
Update Complete. *Happy Helmning!*
akhilrao@Akhils-MacBook-Air prom % helm install prometheus prometheus-community/prometheus
NAME: prometheus
LAST DEPLOYED: Sun Jun  8 15:04:50 2025
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The Prometheus server can be accessed via port 80 on the following DNS name from 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.kubernetes.io/name=prometheus,app.kubernetes.io/instance=prometheus" -o jsonpath="{.it
[0].metadata.name}")
kubectl --namespace default 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.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.kubernetes.io/name=alertmanager,app.kubernetes.io/instance=prometheus" -o jsonpath=".i
ms[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
```

3. Expose Prometheus via NodePort to Access in Browser

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

Changes made:

type: ClusterIP

To:

type: NodePort

Then:

```
minikube ip
```

```

[akhilrao@Akhils-MacBook-Air prom % k get svc
  NAME          TYPE        CLUSTER-IP      EXTERNAL-IP
  PORT(S)      AGE
kubernetes     ClusterIP   10.96.0.1       <none>
  443/TCP      12m
prometheus-alertmanager ClusterIP   10.104.25.65    <none>
  9093/TCP     5m37s
prometheus-alertmanager-headless ClusterIP   None         <none>
  9093/TCP     5m37s
prometheus-kube-state-metrics ClusterIP   10.110.215.138 <none>
  8080/TCP     5m37s
prometheus-prometheus-node-exporter ClusterIP   10.96.142.153 <none>
  9100/TCP     5m37s
prometheus-prometheus-pushgateway ClusterIP   10.102.49.189 <none>
  9091/TCP     5m37s
prometheus-server ClusterIP   10.96.3.15       <none>
  80/TCP       5m37s
[akhilrao@Akhils-MacBook-Air prom % kubectl expose service prometheus-server --type=NodePort --target-port=9090 --name=prometheus-server-ext
service/prometheus-server-ext exposed
[akhilrao@Akhils-MacBook-Air prom % k get svc
  NAME          TYPE        CLUSTER-IP      EXTERNAL-IP
  PORT(S)      AGE
kubernetes     ClusterIP   10.96.0.1       <none>
  443/TCP      23m
prometheus-alertmanager ClusterIP   10.104.25.65    <none>
  9093/TCP     16m
prometheus-alertmanager-headless ClusterIP   None         <none>
  9093/TCP     16m
prometheus-kube-state-metrics ClusterIP   10.110.215.138 <none>
  8080/TCP     16m
prometheus-prometheus-node-exporter ClusterIP   10.96.142.153 <none>
  9100/TCP     16m
prometheus-prometheus-pushgateway ClusterIP   10.102.49.189 <none>
  9091/TCP     16m
prometheus-server ClusterIP   10.96.3.15       <none>
  80/TCP       16m
prometheus-server-ext NodePort    10.110.30.117    <none>
  80:30539/TCP 5m53s

```

kubectl get svc

Access Prometheus at:

***As I'm using docker desktop as driver in the project, I have to run below command
Minikube service prometheus-server-ext

*** Once you stop the tunnel for service. It won't be able to access. it is better to use virtual box(windows)/ hyperkit(mac) as driver for minikube to access prometheus
Show less

```
Last login: Thu Jun  5 12:15:53 on ttys000
You have new mail.
[akhilrao@Akhils-MacBook-Air ~ % minikube service prometheus-server-ext
|-----|-----|-----|-----|
| NAMESPACE | NAME | TARGET PORT | URL |
|-----|-----|-----|-----|
| default | prometheus-server-ext | 80 | http://192.168.49.2:30539 |
|-----|-----|-----|-----|
🏃 Starting tunnel for service prometheus-server-ext.
|-----|-----|-----|-----|
| NAMESPACE | NAME | TARGET PORT | URL |
|-----|-----|-----|-----|
| default | prometheus-server-ext | | http://127.0.0.1:52644 |
|-----|-----|-----|-----|
🎉 Opening service default/prometheus-server-ext in default browser...
❗ Because you are using a Docker driver on darwin, the terminal needs to be open to run it.
```

If using other drivers, access using:

`http://<minikube-ip>:<node-port>`

The screenshot shows the Prometheus UI with a single query result. The query is `>_ node_memory_MemTotal_bytes - node_memory_MemAvailable_bytes`. The results table has one row with the following details:

Labels	Value
app_kubernetes_io_component	"metrics"
app_kubernetes_io_instance	"prometheus"
app_kubernetes_io_managed_by	"Heim"
app_kubernetes_io_name	"prometheus-node-exporter"
app_kubernetes_io_part_of	"prometheus-node-exporter"
app_kubernetes_io_version	"1.9.1"
helm_sh_chart	"prometheus-node-exporter-4.46.1"
instance	"192.168.49.2:9100"
job	"kubernetes-service-endpoints"
namespace	"default"
node	"minikube"
service	"prometheus-prometheus-node-exporter"

Load time: 30ms Result series: 1

Query used to view Memory usage of nodes:

`node_memory_MemTotal_bytes - node_memory_MemAvailable_bytes`

Grafana

4. Add the Grafana Helm Chart & Install

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

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

Command to get password for grafana UI:

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

```
akhilrao@Akhils-MacBook-Air prom % helm repo add grafana https://grafana.github.io/helm-charts
"grafana" has been added to your repositories
akhilrao@Akhils-MacBook-Air prom % kubectl
NAME          READY   STATUS    RESTARTS   AGE
prometheus-alertmanager-0   1/1     Running   0          39m
prometheus-kube-state-metrics-5b9cfb448c-75kw7   1/1     Running   0          39m
prometheus-prometheus-node-exporter-ptfnt        1/1     Running   0          39m
prometheus-prometheus-pushgateway-5dd6b84f88-mx8rj  1/1     Running   0          39m
prometheus-server-7b4cd97c8-mjqwf                2/2     Running   0          39m
akhilrao@Akhils-MacBook-Air prom % 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 Helm-ing!*
akhilrao@Akhils-MacBook-Air prom % helm install grafana grafana/grafana
NAME: grafana
LAST DEPLOYED: Sun Jun 8 15:44:26 2025
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. #####
#####
akhilrao@Akhils-MacBook-Air prom % kubectl get secret --namespace default grafana -o jsonpath=".data.admin-password" | base64 --decode ; echo
```

5. Expose Grafana via NodePort

```
kubectl get svc
```

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

```
minikube ip
```

Access Grafana at:

<http://<minikube-ip>:<grafana-node-port>>

Or, if you are using docker desktop as driver -

```
Minikube service grafana-ext
```

```
akhilrao@Akhils-MacBook-Air prom % kubectl expose service grafana --type=NodePort --target-port=3000 --name=grafana-ext
Error from server (AlreadyExists): services "grafana-ext" already exists
akhilrao@Akhils-MacBook-Air prom % kubectl get svc
NAME          TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)      AGE
grafana       ClusterIP  10.106.104.38  <none>        80/TCP       3m53s
grafana-ext   NodePort   10.103.17.254   <none>        80:30671/TCP 39s
kubernetes    ClusterIP  10.96.0.1    <none>        443/TCP      50m
prometheus-alertmanager  ClusterIP  10.104.25.65  <none>        9093/TCP      43m
prometheus-alertmanager-headless ClusterIP  None        <none>        9093/TCP      43m
prometheus-kube-state-metrics  ClusterIP  10.110.215.138 <none>        8080/TCP      43m
prometheus-prometheus-node-exporter ClusterIP  10.96.142.153 <none>        9100/TCP      43m
prometheus-prometheus-pushgateway  ClusterIP  10.102.49.189  <none>        9091/TCP      43m
prometheus-server    ClusterIP  10.96.3.15    <none>        80/TCP       43m
prometheus-server-ext  NodePort   10.110.30.117   <none>        80:30539/TCP 33m
```

```
akhilrao — ssh - minikube service grafana-ext — 80x24
Last login: Sun Jun  8 15:27:47 on ttys001
You have new mail.
[akhilrao@Akhils-MacBook-Air ~ % minikube service grafana-ext
-----|-----|-----|-----|
| NAMESPACE | NAME | TARGET PORT | URL |
-----|-----|-----|-----|
| default   | grafana-ext | 80 | http://192.168.49.2:30671 |
-----|-----|-----|-----|
🏃 Starting tunnel for service grafana-ext.
-----|-----|-----|-----|
| NAMESPACE | NAME | TARGET PORT | URL |
-----|-----|-----|-----|
| default   | grafana-ext |                | http://127.0.0.1:53145 |
-----|-----|-----|-----|
🎉 Opening service default/grafana-ext in default browser...
❗ Because you are using a Docker driver on darwin, the terminal needs to be open to run it.
```

The screenshot shows a web browser window with the URL `127.0.0.1:53145/?orgId=1&from=now-6h&to=now&timezone=browser`. The page is titled "Welcome to Grafana". On the left, there is a sidebar with navigation links: Home, Bookmarks, Starred, Dashboards, Explore, Drilldown (with a "New!" badge), Alerting, Connections, and Administration. The main content area has three panels: "Basic" (with a sub-section "TUTORIAL DATA SOURCE AND DASHBOARDS" about Grafana fundamentals), "DATA SOURCES" (with a sub-section "Add your first data source"), and "DASHBOARD" (with a sub-section "Create yo dashboard"). At the bottom, there are sections for "Dashboards" (Starred dashboards, Recently viewed dashboards) and "Latest from the blog" (an article about Database monitoring with a chart showing client operation duration).

6. + Add Prometheus as a Data Source in Grafana

- Go to connections > Data Sources
- Choose Prometheus
- Enter the URL: http://<minikube-ip>:<prometheus-node-port>
- Click Save & Test

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

Learn how in the docs ↗

COMPLETE
Create your first dashboard

Learn how in the docs ↗

Remove this panel

Latest from the blog

Jun 06 Database observability: How OpenTelemetry semantic conventions improve consistency across signals

Databases are a crucial part of modern systems, which means database observability

Add data source

Choose a data source type

Q Filter by name or type

Time series databases

Prometheus Open source time series database & alerting Core

Graphite Open source time series database

prometheus-1

Type: Prometheus

Settings Dashboards

Configure your Prometheus data source below

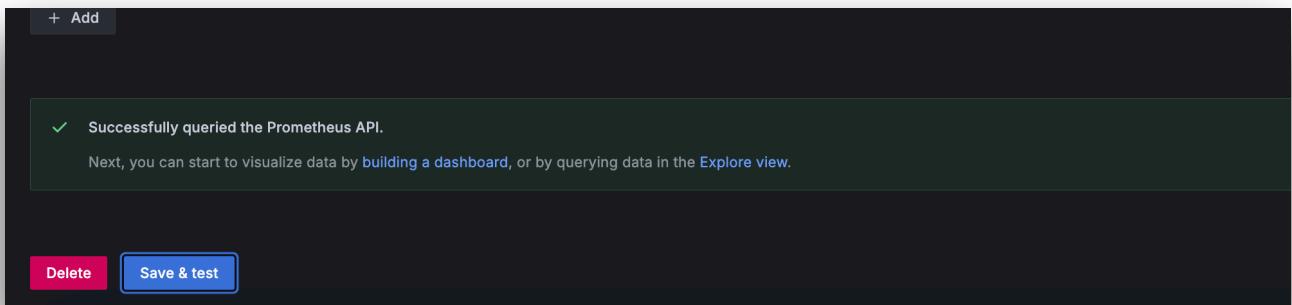
Name prometheus-1 Default

Prometheus server URL * http://192.168.49.2:30539/

Before you can use the Prometheus data source, you must configure it below or in the config file. For detailed instructions, [view the documentation](#).

Fields marked with * are required

Connection



A successfully added data source says^

7. Build Dashboards:

- Go to Dashboard > new > import
- Enter ID: 3662 (#this is a pre-defined dashboard for a Kubernetes cluster)
- Select the Prometheus data source
- Click Import

The screenshot shows the 'Import dashboard' dialog. At the top, it says 'Import dashboard' and 'Import dashboard from file or Grafana.com'. Below that is a dashed box with an upward arrow icon and the text 'Upload dashboard JSON file'. It also says 'Drag and drop here or click to browse' and 'Accepted file types: .json, .txt'. Underneath this is a search bar with '3662' and a 'Load' button. A small circular icon with a downward arrow is next to the search bar. Below the search bar is a section titled 'Import via dashboard JSON model' containing a JSON code block:

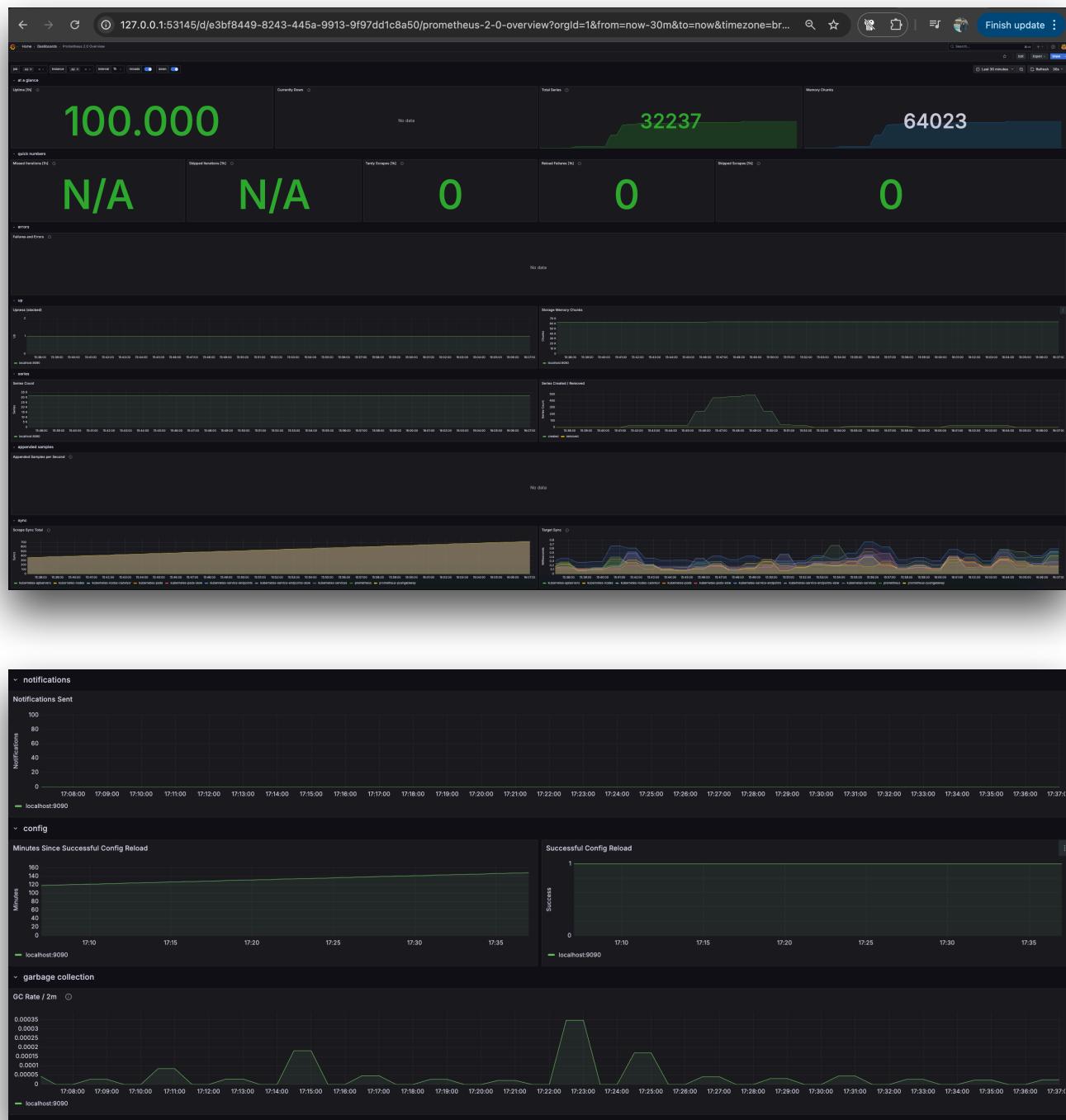
```
{  
  "title": "Example - Repeating Dictionary variables",  
  "uid": "_0HnEoN4z",  
  "panels": [...]  
  ...  
}
```

At the bottom of the dialog are two buttons: 'Load' (blue) and 'Cancel'.

View Live Metrics in Grafana

Explore the pre-built Kubernetes dashboard to view:

- Node usage
- Pod status
- CPU/memory graphs
- Cluster health



> Expose kube-state-metrics for More Cluster Insights

kube-state-metrics exports detailed Kubernetes object-level metrics, such as:

- Number of running/deleted pods
- StatefulSets, DaemonSets, Deployments status
- Node conditions
- Persistent volume statuses

These are not exposed by the default Prometheus metrics, so this adds depth to your monitoring.

Expose kube-state-metrics via NodePort

```
kubectl expose service prometheus-kube-state-metrics --type=NodePort --target-port=8080 --name=prometheus-kube-state-metrics-ext
```

Access it in your browser to confirm it's reachable:

http://<minikube-ip>:<prometheus-kube-state-ext_node-port>/metrics

Or if using docker desktop, use

Minikube service prometheus-kube-state-metrics-ext

```
akhilrao@Akhils-MacBook-Air prom % kubectl expose service prometheus-kube-state-metrics --type=NodePort --target-port=8080 --name=prometheus-kube-state-metrics-ext exposed
akhilrao@Akhils-MacBook-Air prom % k get svc
NAME           TYPE      CLUSTER-IP    EXTERNAL-IP   PORT(S)        AGE
grafana        ClusterIP 10.106.104.38 <none>        80/TCP         33m
grafana-ext    NodePort   10.103.17.254  <none>        80:30671/TCP  30m
kubernetes     ClusterIP 10.96.0.1     <none>        443/TCP       79m
prometheus-alertmanager ClusterIP 10.104.25.65 <none>        9093/TCP      72m
prometheus-alertmanager-headless ClusterIP None          <none>        9093/TCP      72m
prometheus-kube-state-metrics ClusterIP 10.110.215.138 <none>        8080/TCP      72m
prometheus-kube-state-metrics-ext NodePort   10.109.130.155 <none>        8080:31103/TCP 39s
prometheus-prometheus-node-exporter ClusterIP 10.96.142.153 <none>        9100/TCP      72m
prometheus-prometheus-pushgateway   ClusterIP 10.102.49.189 <none>        9091/TCP      72m
prometheus-server     ClusterIP 10.96.3.15   <none>        80/TCP        72m
prometheus-server-ext  NodePort   10.110.30.117  <none>        80:30539/TCP  62m
akhilrao@Akhils-MacBook-Air prom %
```

```
akhilrao@Akhils-MacBook-Air ~ % minikube service prometheus-kube-state-metrics-ext
|-----|-----|-----|-----|
| NAMESPACE |     NAME      | TARGET PORT |     URL      |
|-----|-----|-----|-----|
| default   | prometheus-kube-state-metrics-ext | 8080 | http://192.168.49.2:31103 |
|-----|-----|-----|-----|
🏃 Starting tunnel for service prometheus-kube-state-metrics-ext.
|-----|-----|-----|-----|
| NAMESPACE |     NAME      | TARGET PORT |     URL      |
|-----|-----|-----|-----|
| default   | prometheus-kube-state-metrics-ext |          | http://127.0.0.1:53951 |
|-----|-----|-----|-----|
🎉 Opening service default/prometheus-kube-state-metrics-ext in default browser...
❗ Because you are using a Docker driver on darwin, the terminal needs to be open to run it.
```

kube-state-metrics

Metrics for Kubernetes' state

Version: (version=v2.15.0, branch=, revision=unknown)

- Metrics
- Healthz
- Livez

Download a detailed report of resource usage (pprof format, from the Go runtime):

- heap_usage (memory)
- CPU_usage (60 second profile)

To visualize and share profiles you can upload to [pprof.me](#)

/metrics

```
# HELP kube_certificatesigningrequest_annotations Kubernetes annotations converted to Prometheus labels.
# TYPE kube_certificatesigningrequest_annotations gauge
# HELP kube_certificatesigningrequest_labels [STABLE] Kubernetes labels converted to Prometheus labels.
# TYPE kube_certificatesigningrequest_labels gauge
# HELP kube_certificatesigningrequest_unix_timestamp [STABLE] Unix creation timestamp
# TYPE kube_certificatesigningrequest_unix_timestamp gauge
# TYPE kube_certificatesigningrequest_created gauge
kube_certificatesigningrequest{signer_name="kubernetes.io/kube-apiserver-client-kubelet"} 1.74937488e+09
# HELP kube_certificatesigningrequest_condition [STABLE] The number of each certificatesigningrequest condition
# TYPE kube_certificatesigningrequest_condition gauge
kube_certificatesigningrequest{condition="approved"} 1
kube_certificatesigningrequest{condition="denied"} 0
# HELP kube_certificatesigningrequest_cert_length [STABLE] Length of the issued cert
# TYPE kube_certificatesigningrequest_cert_length gauge
kube_certificatesigningrequest{cert_length=887} 887
# HELP kube_configmap_annotations Kubernetes annotations converted to Prometheus labels.
# TYPE kube_configmap_annotations gauge
# HELP kube_configmap_labels [STABLE] Kubernetes labels converted to Prometheus labels.
# TYPE kube_configmap_labels gauge
# HELP kube_configmap_info [STABLE] Information about configmap.
# TYPE kube_configmap_info gauge
kube_configmap{namespace="monitoring",configmap="kube-root-ca.crt"} 1
kube_configmap{info(namespace="kube-system",configmap="prometheus-alertmanager") } 1
kube_configmap{info(namespace="kube-system",configmap="kube-proxy") } 1
kube_configmap{info(namespace="kube-system",configmap="kube-apiserver-legacy-service-account-token-tracking") } 1
kube_configmap{info(namespace="kube-system",configmap="kubeadm-config") } 1
kube_configmap{info(namespace="kube-system",configmap="kube-root-ca.crt") } 1
kube_configmap{info(namespace="kube-system",configmap="kubelet-config") } 1
kube_configmap{info(namespace="kube-system",configmap="extension-apiserver-authentication") } 1
# HELP kube_configmap_created [STABLE] Unix creation timestamp
# TYPE kube_configmap_created gauge
kube_configmap{created(namespace="kube-system",configmap="kube-apiserver-legacy-service-account-token-tracking") } 1.74937488e+09
kube_configmap{created(namespace="monitoring",configmap="kube-root-ca.crt") } 1.4937518e+09
kube_configmap{created(namespace="default",configmap="prometheus-alertmanager") } 1.74937529e+09
kube_configmap{created(namespace="kube-system",configmap="kubelet-config") } 1.74937488e+09
kube_configmap{created(namespace="kube-system",configmap="kubeadm-config") } 1.74937482e+09
kube_configmap{created(namespace="kube-system",configmap="kube-root-ca.crt") } 1.74937482e+09
kube_configmap{created(namespace="kube-node-lease",configmap="grafana") } 1.74937766e+09
kube_configmap{created(namespace="kube-system",configmap="coredns") } 1.74937482e+09
kube_configmap{created(namespace="kube-public",configmap="cluster-info") } 1.74937482e+09
kube_configmap{created(namespace="kube-public",configmap="kube-root-ca.crt") } 1.74937488e+09
# HELP kube_configmap_extension_apiserver_authentication [STABLE] Unix creation timestamp
# TYPE kube_configmap_extension_apiserver_authentication gauge
kube_configmap{extension_apiserver_authentication="grafana"} 2819
kube_configmap{extension_apiserver_authentication="coredns"} 286
kube_configmap{extension_apiserver_authentication="kubeadm-config"} 360
kube_configmap{extension_apiserver_authentication="kube-root-ca.crt"} 360
kube_configmap{extension_apiserver_authentication="kube-public",configmap="cluster-info"} 357
kube_configmap{extension_apiserver_authentication="kube-public",configmap="kube-root-ca.crt"} 361
kube_configmap{extension_apiserver_authentication="extension-apiserver-authentication"} 28
```

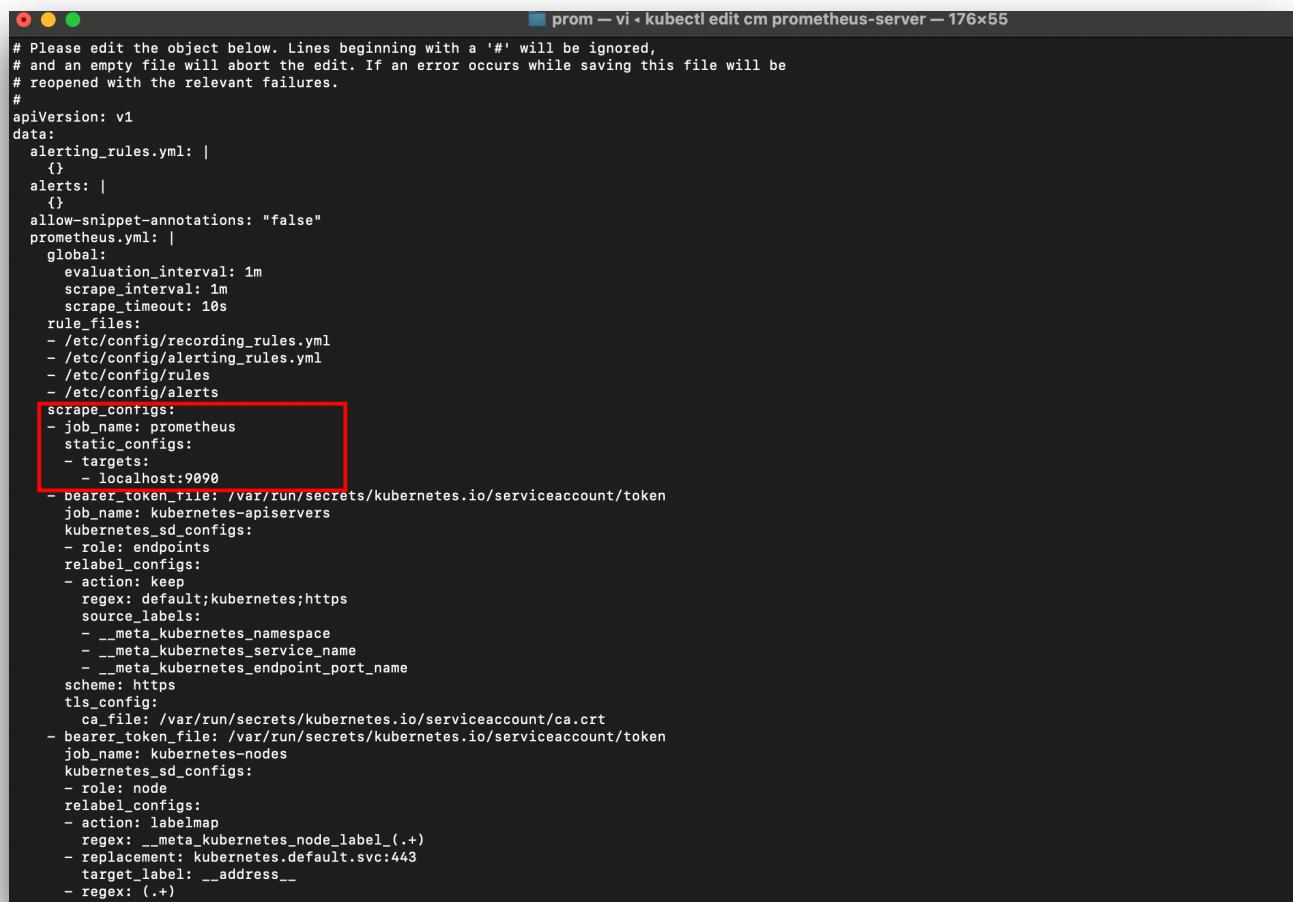
- > Add kube-state-metrics to Prometheus Scrape Config
 - Edit the prometheus-server config map:

Kubectl get cm

kubectl edit configmap prometheus-server

- Add the following under scrape_configs:

```
- job_name: 'kube-state-metrics'
  static_configs:
    - targets: ['<minikube-ip>:<kube-state-node-port>']
```



```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  alerting_rules.yml: |
    {}
  alerts: |
    {}
  allow-snippet-annotations: "false"
  prometheus.yml: |
    global:
      evaluation_interval: 1m
      scrape_interval: 1m
      scrape_timeout: 10s
    rule_files:
      - /etc/config/recording_rules.yml
      - /etc/config/alerting_rules.yml
      - /etc/config/rules
      - /etc/config/alerts
    scrape_configs:
      - job_name: prometheus
        static_configs:
          - targets:
              - localhost:9090
        bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
        job_name: kubernetes-apiservers
        kubernetes_sd_configs:
          - role: endpoints
        relabel_configs:
          - action: keep
            regex: default;kubernetes;https
            source_labels:
              - __meta_kubernetes_namespace
              - __meta_kubernetes_service_name
              - __meta_kubernetes_endpoint_port_name
            scheme: https
            tls_config:
              ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
        job_name: kubernetes-nodes
        kubernetes_sd_configs:
          - role: node
        relabel_configs:
          - action: labelmap
            regex: __meta_kubernetes_node_label_(.+)
          - replacement: kubernetes.default.svc:443
            target_label: __address__
          - regex: (.+)
```

- Restart the Prometheus server pod to reload config:

kubectl delete pod <prometheus-pod-name>

> What is kube-state-metrics and Why It Matters?

kube-state-metrics is a service that listens to the Kubernetes API and generates metrics about the state of Kubernetes objects — like how many pods are running, whether deployments are available, what nodes are ready, etc.

Unlike system metrics (CPU, memory, etc.), these metrics are about the state of Kubernetes itself, not the node.

You can think of it this way:

- Prometheus Node Exporter = resource usage
- kube-state-metrics = Kubernetes health/state

This is critical for:

- Alerting on failed deployments or pending pods
 - Tracking object counts over time
 - Monitoring cluster stability
-

> Want to Monitor Your Own Application?

To monitor a custom app:

1. Add a /metrics endpoint using Prometheus client library (e.g., `prometheus_client` in Python or `prom-client` in Node.js)
 2. Expose the service using NodePort
 3. Add that endpoint to Prometheus `scrape_configs` in the same way as kube-state-metrics
-

Conclusion

This project demonstrates how to build a local Kubernetes monitoring stack using:

- Minikube
- Prometheus
- Grafana
- kube-state-metrics

With this setup, you're ready to monitor your Kubernetes clusters and applications effectively!

Newrelic

1. Create your New relic account

2. Integrations & Agents > Helm > Provide New relic License key > copy url > run in your cluster

```
KSM_IMAGE_VERSION="v2.13.0" && helm repo add newrelic https://helm-charts.newrelic.com && helm repo update && kubectl create namespace newrelic ; helm upgrade --install newrelic-newrelic/nri-bundle --set global.licenseKey=<your_license_key_new_relic_goes_here> --set global.cluster= --namespace=newrelic --set newrelic-infrastructure.privileged=true --set global.lowDataMode=true --set kube-state-metrics.image.tag=${KSM_IMAGE_VERSION} --set kube-state-metrics.enabled=true --set kubeEvents.enabled=true --set newrelic-prometheus-agent.enabled=true --set newrelic-prometheus-agent.lowDataMode=true --set newrelic-prometheus-agent.config.kubernetes.integrations_filter.enabled=false --set k8s-agents-operator.enabled=true --set logging.enabled=true --set newrelic-logging.lowDataMode=true
```

```
akhilrao@Akhils-MacBook-Air deploy % KSM_IMAGE_VERSION="v2.13.0" && helm repo add newrelic https://helm-charts.newrelic.com && helm repo update && kubectl create namespace newrelic ; helm upgrade --install newrelic-newrelic/nri-bundle --set global.licenseKey=<your_license_key_new_relic_goes_here> --set global.cluster= --namespace=newrelic --set newrelic-infrastructure.privileged=true --set global.lowDataMode=true --set kube-state-metrics.image.tag=${KSM_IMAGE_VERSION} --set kube-state-metrics.enabled=true --set kubeEvents.enabled=true --set newrelic-prometheus-agent.enabled=true --set newrelic-prometheus-agent.lowDataMode=true --set newrelic-prometheus-agent.config.kubernetes.integrations_filter.enabled=false --set k8s-agents-operator.enabled=true --set logging.enabled=true --set newrelic-logging.lowDataMode=true
```

3. Configure APM auto-instrumentation:

> copy the instrumentation.yaml to your cluster

> kubectl apply -f ./instrumentation.yaml -n newrelic

The screenshot shows the Newrelic Kubernetes integration configuration interface. At the top, there's a sidebar with a 'Data source' section and a 'Kubernetes' icon. Below that, a summary states: 'Our Kubernetes monitoring solution gives you visibility into your Kubernetes clusters and workloads in minutes, whether your clusters are hosted on-premises or in the cloud.' On the left, a vertical list of steps is shown with green checkmarks: 'Select instrumentation method', 'Enter your credentials', 'Configure the Kubernetes integration', 'Select additional data', 'Enable eAPM instrumentation', 'Enable APM auto-instrumentation', 'Gather Log data', 'Install the Kubernetes integration', 'Configure APM auto-instrumentation' (which is currently selected), and 'Test the connection'. To the right of this list, there's a note: 'To enable APM auto-instrumentation deploy the following manifest with the command specified.' Below this note is a section titled 'Please select how to filter applications to be auto-instrumented.' It contains two checkboxes: 'Namespace label' (which is checked) and 'Pod label'. Underneath these checkboxes is a 'Namespace to monitor' field containing 'default'. At the bottom of the page, there's a code snippet labeled 'instrumentation.yaml' with a size of '32 lines 1.1 KB'. It includes download and copy-to-clipboard buttons. The code itself starts with:

```
1 apiVersion: newrelic.com/v1alpha2
2 kind: Instrumentation
3 metadata:
```

4. Test the connection > successful

The screenshot shows the 'Integrations & Agents' section of a monitoring tool. On the left, a sidebar lists steps for setting up the Kubernetes integration, each marked with a green checkmark. On the right, a 'Test the connection' panel displays a table with one row. The table has three columns: 'Connection type' (Kubernetes integration), 'Status' (Successful), and 'Details' (Successfully installed). Below the table are two buttons: 'Test connection' and 'See your data'.

Connection type	Status	Details
Kubernetes integration	Successful	Successfully installed

5. New relic > All Entities > check for your cluster

The screenshot shows the 'All Entities' dashboard in New Relic. The left sidebar shows a hierarchical list of entities, including 'All entities (76)', 'Your system', 'Other entities', 'Kubernetes', and 'Log Collectors'. The main area lists various Kubernetes entities with their counts: Hosts (3), Containers (27), Corednses (1), Kubernetes Clusters (1), Kubernetes DaemonSets (5), Kubernetes Deployments (10), Kubernetes PersistentVolumes (2), Kubernetes PersistentVolumeClaims (2), Kubernetes Pods (22), Kubernetes StatefulSets (2), and Fluent Bit - Kubernetes (1).

Kubernetes/Clusters local_cluster:

Pod	Namespace	Node	CPU Used (cores)	Memory Used (bytes)	Pod Status	Alert Status
coredns-668d0f9c-gk4dw	kube-system	minikube	0.00144 cores	31.2 MB	Running	NOT_CONFIGURED
etcd-minikube	kube-system	minikube	0.03932 cores	69.1 MB	Running	NOT_CONFIGURED
kube-system-vet-minikube	kube-system	minikube	0.07179 cores	202 MB	Running	NOT_CONFIGURED
kube-controller-manager-minikube	kube-system	minikube	0.02886 cores	75.4 MB	Running	NOT_CONFIGURED
kube-proxy-hm427	kube-system	minikube	0.00347 cores	22.1 MB	Running	NOT_CONFIGURED
kube-scheduler-minikube	kube-system	minikube	0.01133 cores	33.7 MB	Running	NOT_CONFIGURED
storage-provisioner	kube-system	minikube	0.00024 cores	28.9 MB	Running	NOT_CONFIGURED
crashloop	default	minikube	-	-	Running	NOT_CONFIGURED
grafana-59fbcfb7-trad4	default	minikube	0.02228 cores	153 MB	Running	NOT_CONFIGURED
mysqld-684558667-8ydh	default	minikube	-	-	Running	NOT_CONFIGURED
mysqld-684558667-rgtq	default	minikube	-	-	Running	NOT_CONFIGURED
mysqld-684558667-756	default	minikube	-	-	Running	NOT_CONFIGURED
mysqld-644d559c-2qnt7	default	minikube	0.01 μ cores	34.2 MB	Running	NOT_CONFIGURED
prometheus-k8smanager-0	default	minikube	0.00016 cores	22.7 MB	Running	NOT_CONFIGURED

Name	Account	CPU usage (%)	Memory usage (%)
local_cluster	Account 6429287	0.34%	54.8%

Now that the Local Kubernetes cluster has been successfully added to New Relic, let's setup an alert.

Requirement: Get an alert when an app's pod fails/changes its state from running

The screenshot shows the New Relic Data Explorer interface. A query is run against the K8sContainerSample database:

```
1 From K8sContainerSample SELECT max(restartCount) WHERE podName LIKE '%myapp%' FACET podName SINCE 30 minutes ago
```

The results table displays pod names and their maximum restart counts:

Pod Name	Max Restart Count
myapp-74cbb6bf5-rmz	0
myapp-84dcfd5f9c-2qnt7	0
myapp-84dcfd5f9c-blbq	0
myapp-74cbb6bf5-w285l	0
myapp-74cbb6bf5-8pybt	0
myapp-84dcfd5f9c-bflwz	0

1. Check the pod status of the app using NRQL(newrelic query language) to validate the data ingestion:

Query: From K8sContainerSample SELECT max(restartCount) WHERE podName LIKE '%myapp%' FACET podName SINCE 30 minutes ago

As seen above, the data ingestion looks good and we can see the restart count of the pods for app=myapp is up and running.

```
Kubectl get pods | grep my app
```

myapp-84dcfd5f9c-2qnt7	1/1	Running	0	4m16s
myapp-84dcfd5f9c-bflwz	1/1	Running	0	3m19s
myapp-84dcfd5f9c-19blq	1/1	Running	0	3m47s

2. Now, create an alert policy and alert condition:

Alerts > Alert Policies > New Alert Policy > Kubernetes

The screenshot shows the New Relic Alerts > Alert Policies interface. The 'Alert Policies' tab is selected. A red arrow points to the '+ New alert condition' button in the top right corner.

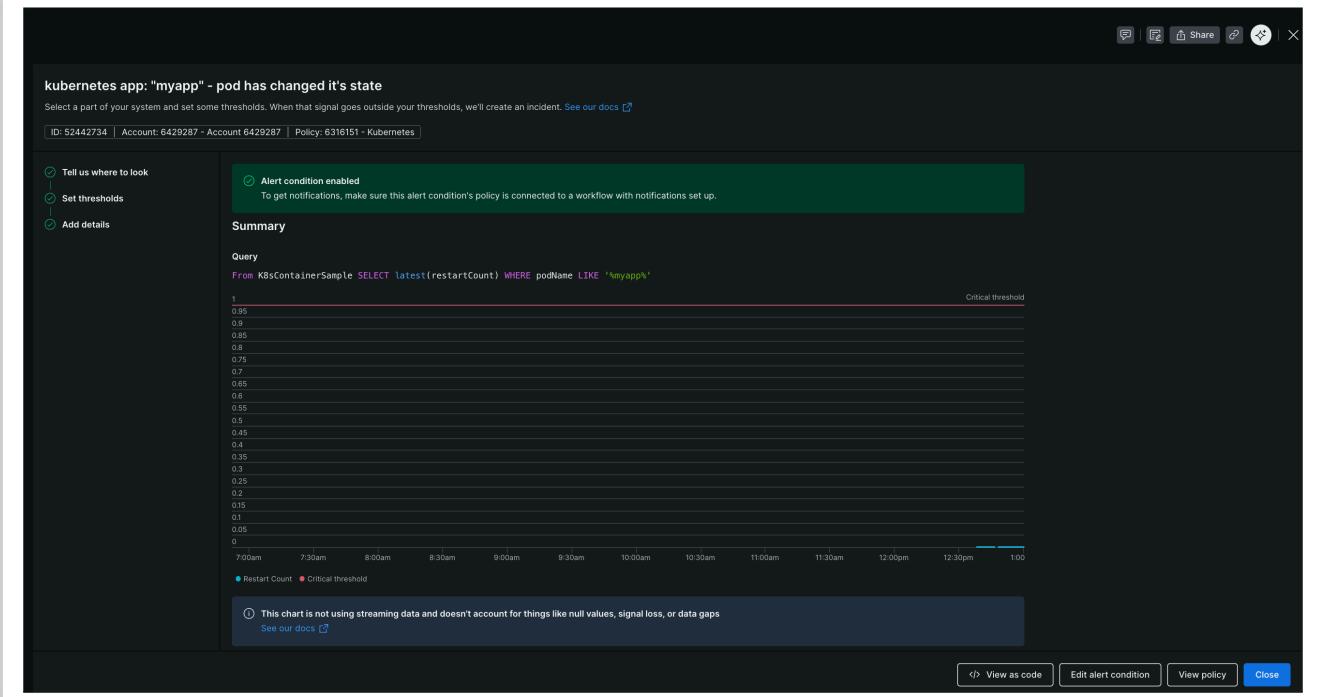
Kubernetes > New alert condition

The screenshot shows the New Relic Kubernetes > New alert condition interface. The 'Alerts / Alert Policies' tab is selected. A red arrow points to the 'Condition Name' input field at the bottom of the page.

3. Alert condition setup:

Query: From K8sContainerSample SELECT latest(restartCount) WHERE podName LIKE '%myapp%'

Use set condition threshold > static > above '1' for at least '5' minutes



Alert condition	Policy	Query	Threshold	Type	Open issues	Last modified
kubernetes app: "myapp" - pod has changed its state	Kubernetes	From K8sContainerSample ...	Critical: above 1 for at least 5 minutes Create a warning threshold	NRQL Query	0	Jun 8, 2025, 1:04pm
crond service monitor	Initial policy	SELECT filter(uniqueCount(...	Critical: above or equals 1 Create a warning threshold	NRQL Query	0	Jun 4, 2025, 1:37pm
load_aws	Initial policy	SELECT average(cpuPerce...	Critical: above 15 for at least 1 minute Warning: above 10 for at least 1 minute	NRQL Query	0	Jun 4, 2025, 12:41pm
Memory_aws	Initial policy	SELECT average(cpuPerce...	Critical: above 50 for at least 1 minute Create a warning threshold	NRQL Query	0	Jun 4, 2025, 10:35am
nginx service monitor	Initial policy	SELECT filter(uniqueCount(...	Critical: above or equals 1 Create a warning threshold	NRQL Query	0	Jun 4, 2025, 1:41pm

4. Lets make changes to 'myapp' deployment to test the alert condition.

```
akhilrao@Akhils-MacBook-Air deploy % cat deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp
spec:
  replicas: 3
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
        - name: myapp
          image: busybox
          command: ["/bin/sh", "-c"]
          args: ["echo 'Starting...'; sleep 5; echo 'Now crashing...'; exit 1"]
```

```
Kubectl apply -f deployment.yaml
```

5. Check current state of the pods > pods are now in crashloop

```
akhilrao@Akhils-MacBook-Air deploy % kubectl get pods | grep myapp
myapp-684558667-9gn8h           0/1     CrashLoopBackOff   17 (4m8s ago)   69m
myapp-684558667-rgxfr           0/1     CrashLoopBackOff   17 (5m9s ago)   71m
myapp-684558667-vn75s           0/1     CrashLoopBackOff   17 (4m14s ago)  69m
myapp-84dcfd5f9c-2qnt7          1/1     Running          0            88m
akhilrao@Akhils-MacBook-Air deploy %
```

6. Wait for Alert to trigger: Newrelic > Alerts > Issues & Activity

Entity name	Notified	Contains	Actions taken
K8sContainerSample query result is > 1.0 for 5 minutes on 'kubernetes app: "myapp" - pod has changed its state'	1 incident		...

Active Issue

Critical Activated on Jun 8, 2025 1:38pm Duration: 1m Last updated: Jun 8, 2025 1:38pm

K8sContainerSample query result is > 1.0 for 5 minutes on 'kubernetes' app: "myapp" - pod has changed its state

Source: Issue payload

Current state: issue created

Acknowledge ...

Overview Potential causes

> AI summary [Preview](#)

Incidents (1)

Critical Incident opened on Jun 8, 2025 1:38pm Duration: 1m

Description [New](#); Add description

Alert Policy: Kubernetes Alert Condition: kubernetes app: "myapp" - pod has changed its state

From: K8sContainerSample SELECT latest(restartCount) WHERE podName LIKE '%myapp%' TIMESERIES 1 SECOND SINCE '2025-06-08 13:38:00 Z' UNTIL '2025-06-08 13:39:35 Z'

Query this data ...

Incident period

Tags (12)

accountid: 6429287 account: Account 6429287 enabled: true id: 52442734 nr.alerts.conditionId: 52442734 nr.alerts.enabled: true nr.alerts.policyId: 6316151 nr.alerts.type: NRQL_Query policyId: 6316151 trustedAccountId: 6429287 type: NRQL_Query

Issue timeline & event log

ALERT TRIGGERED SUCCESSFULLY!!

All Kubernetes entities of the cluster that are currently added to Newrelic:

The screenshot shows the New Relic All Entities dashboard with the URL <https://one.newrelic.com/nr1-core/navigator/home?account=6429287&duration=1800000&state=5cda9f17-d346-fd69-9337-325ddcd17fb9f>. The left sidebar is collapsed, showing various monitoring categories like Integrations & Agents, Dashboards, APM & Services, Logs, Traces, Synthetic Monitoring, Alerts, Infrastructure, and Kubernetes. The main area displays the following data:

- Alerting groups:** Issues (1 red hexagon)
- Operational groups:**
 - Containers (40) - 40 grey hexagons
 - Kubernetes Pods (35) - 35 grey hexagons
 - Kubernetes Deployments (11) - 11 grey hexagons
 - Kubernetes DaemonSets - 6 grey hexagons
 - Kubernetes PersistentVolumeClaims - 2 green hexagons
 - Kubernetes PersistentVolume - 2 grey hexagons
 - Kubernetes Pod - 35 grey hexagons
 - Kubernetes StatefulSets - 2 grey hexagons
 - Coredns - 1 grey hexagon
 - Fluent Bit - 1 grey hexagon
 - Kubernetes Clusters - 1 grey hexagon

At the top right, there are buttons for "Create a workload" and "Add data". The status bar indicates "Since 30 minutes ago (UTC)".