Kubernetes Monitoring on Minikube using Prometheus, Grafana & New Relic

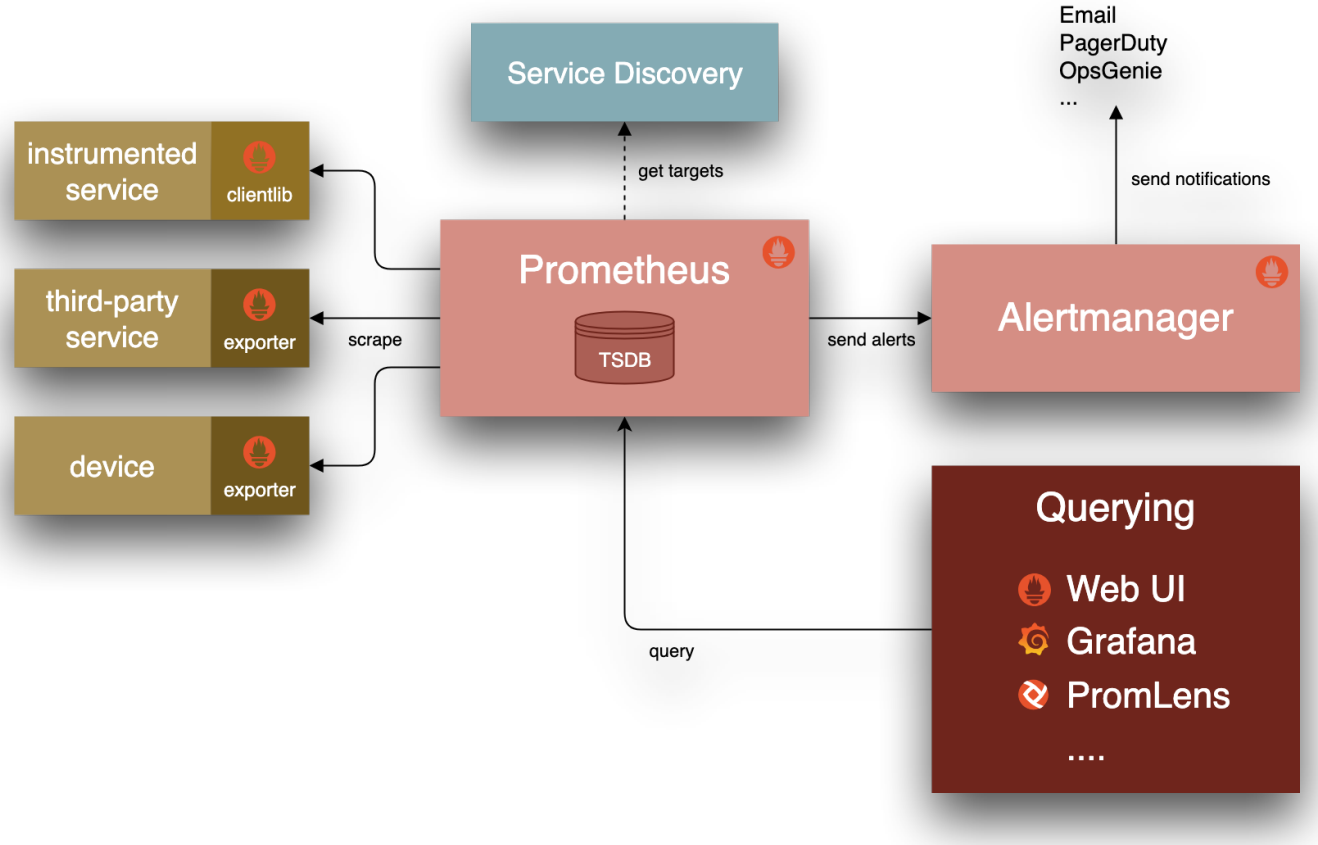
Project Overview

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

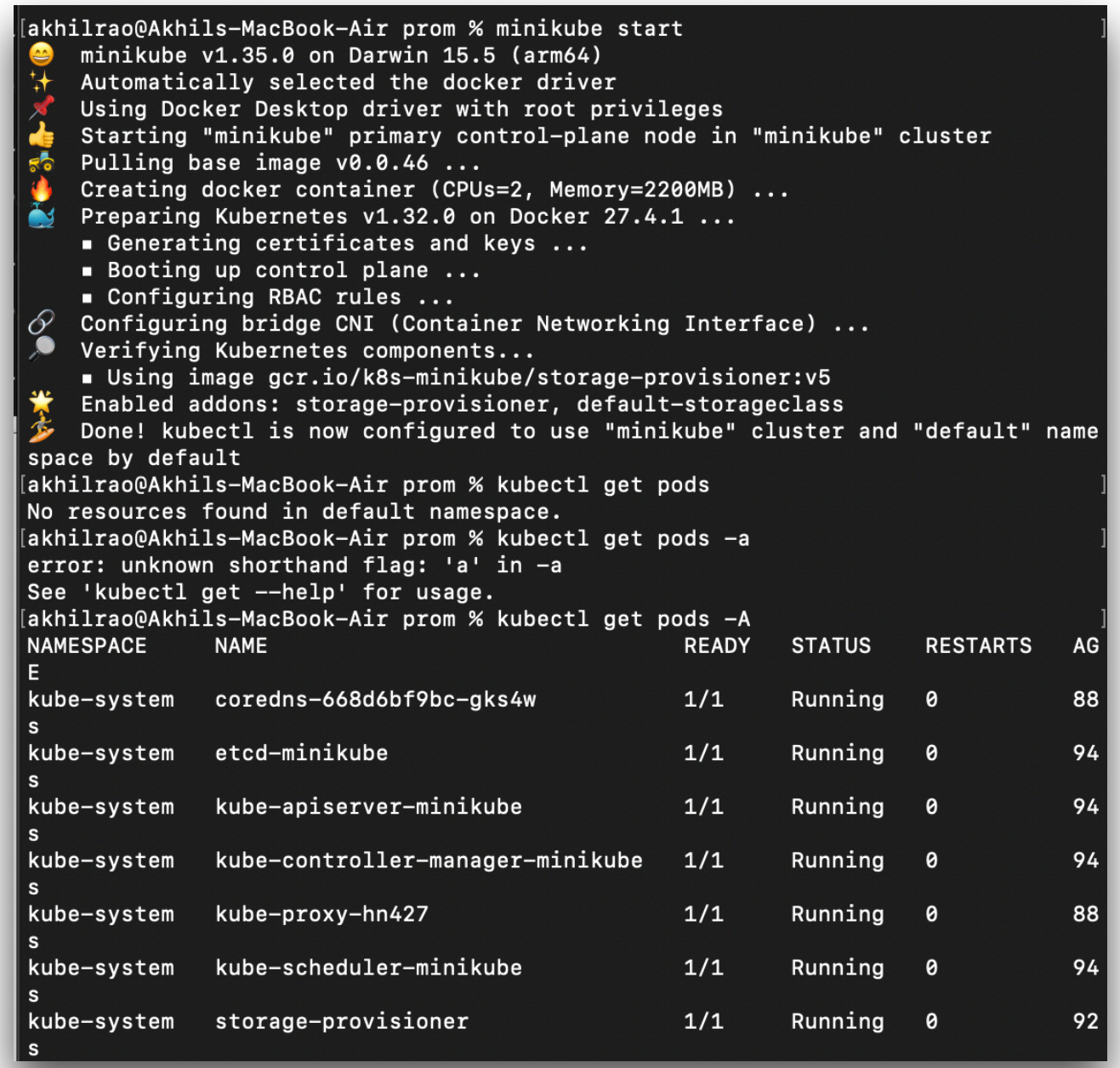
Prometheus

Step-by-Step Setup

---

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

minikube start --driver=docker



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 Helming!⎈
```

```
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 Helming!⎈
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="{.
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 --ty
pe=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:

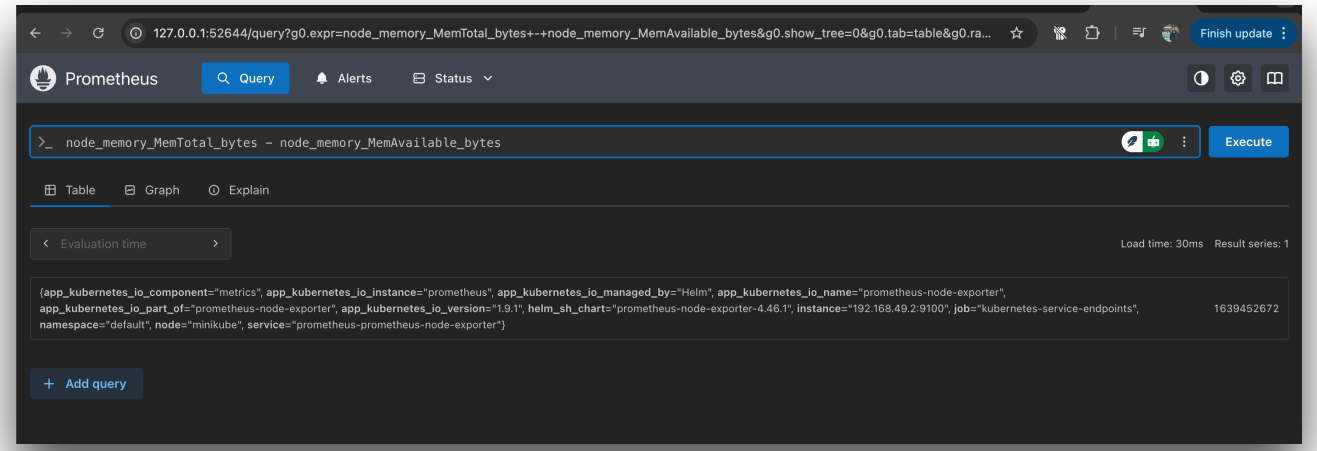
***As I'm using docker desktop as driver in the project, I have to run below command
Minikube service prometheus-server-ext

*** 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 op
en to run it.
```

If using other drivers, access using:

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



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 % kgp
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-ptfmt        1/1     Running   0          39m
prometheus-prometheus-pushgateway-56d6b84f88-mx8rj   1/1     Running   0          39m
prometheus-server-7b4cd9b7c8-mjwqf               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 Helming!⎈
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
```

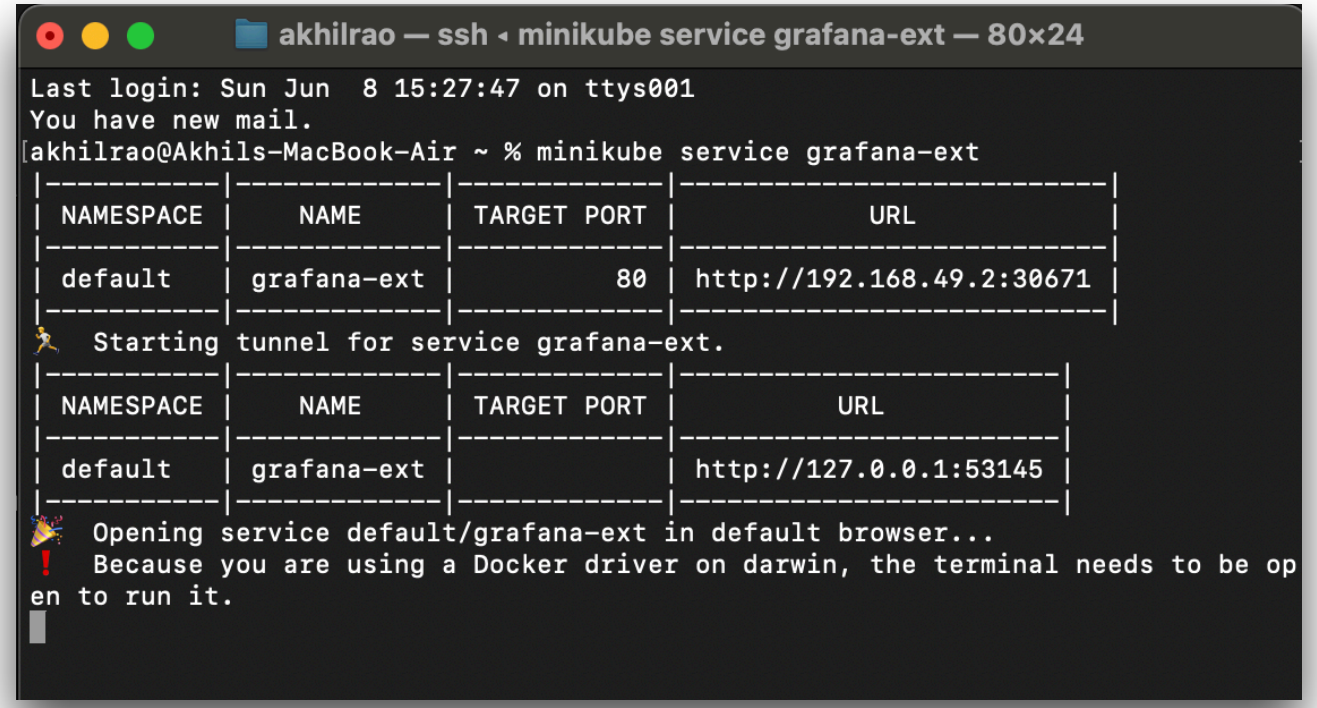
Access Grafana at:

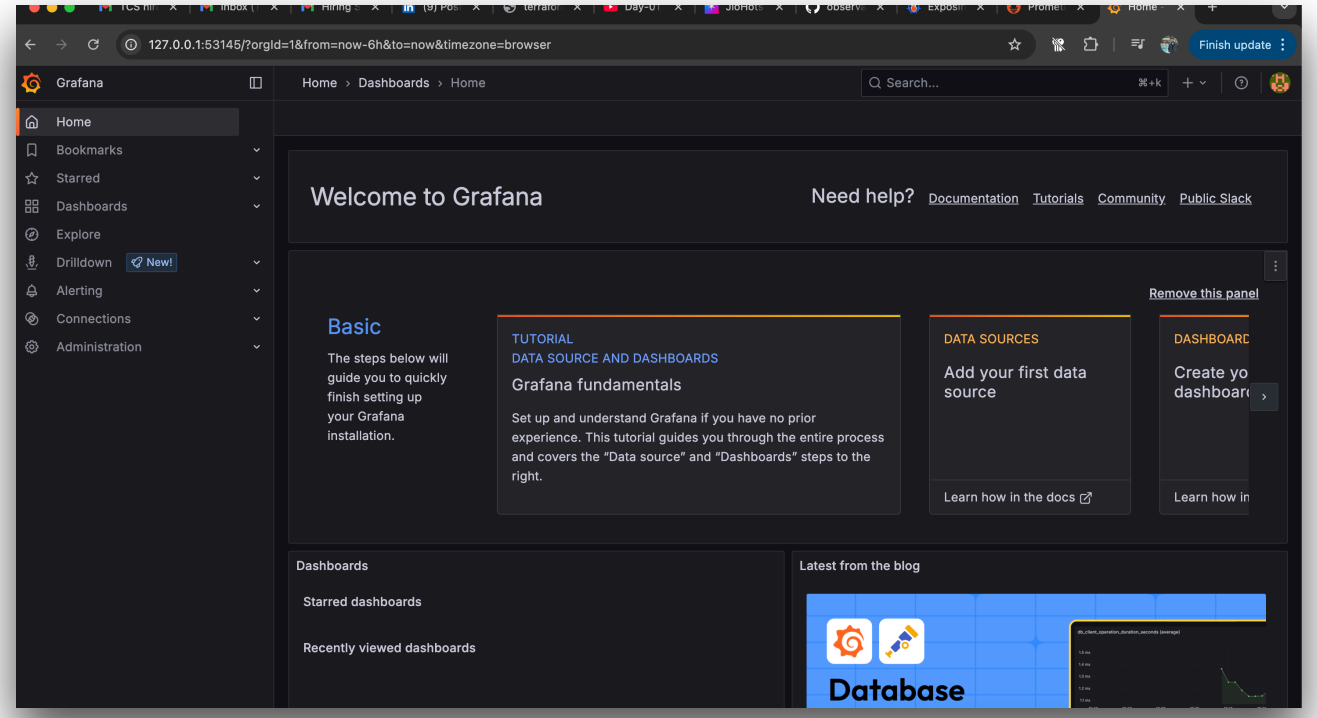
http://<minikube-ip>:<grafana-node-port>

Or, if you are using docker desktop as driver -

```
Minikube service grafana-ext
```

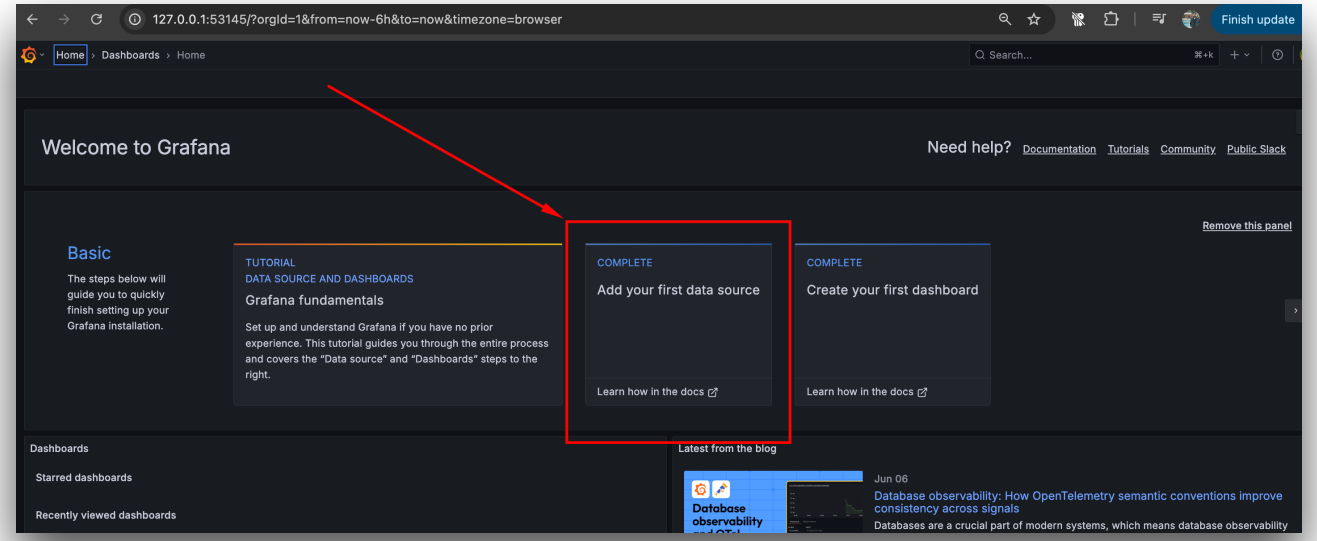
```
akhilrao@Akhils-MacBook-Air prom % kubectl expose service grafana --type=NodePort --target-port=3000 --name=grafana-ext
Error from server (AlreadyExists): services "grafana-ext" already exists
akhilrao@Akhils-MacBook-Air prom % k get svc
NAME                                  TYPE        CLUSTER-IP       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
```

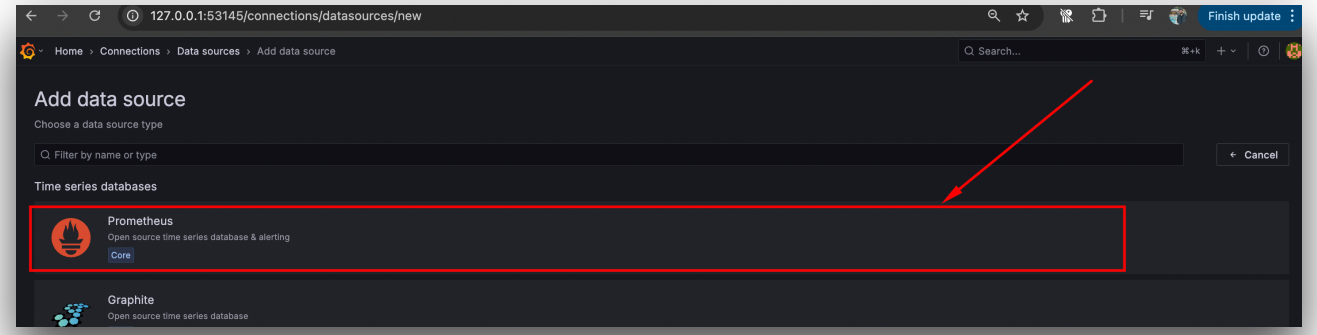

akhilrao — ssh ◂ minikube service grafana-ext — 80×24
Last login: Sun Jun 8 15:27:47 on ttys001
You have new mail.
akhilrao@Akhils-MacBook-Air ~ % minikube service grafana-ext
| NAMESPACE | NAME | TARGET PORT | URL |
| default | grafana-ext | 80 | http://192.168.49.2:30671 |
Starting tunnel for service 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.

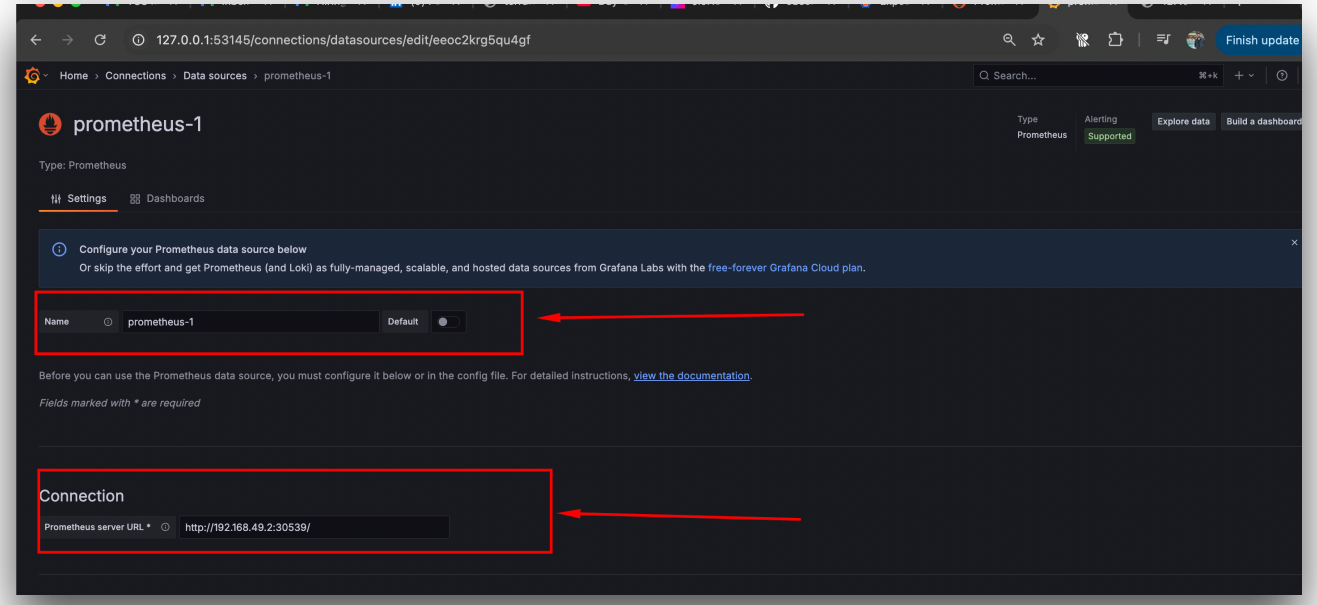

127.0.0.1:53145/?orgId=1&from=now-6h&to=now&timezone=browser
Finish update
Grafana
Home › Dashboards › Home
Search...
Home
Bookmarks
Starred
Dashboards
Explore
Drilldown
New!
Alerting
Connections
Administration
Welcome to Grafana
Need help?
Documentation
Tutorials
Community
Public Slack
Remove this panel
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.
DATA SOURCES
Add your first data source
Learn how in the docs
DASHBOARD
Create your dashboard
Learn how in
Dashboards
Starred dashboards
Recently viewed dashboards
Latest from the blog
Database

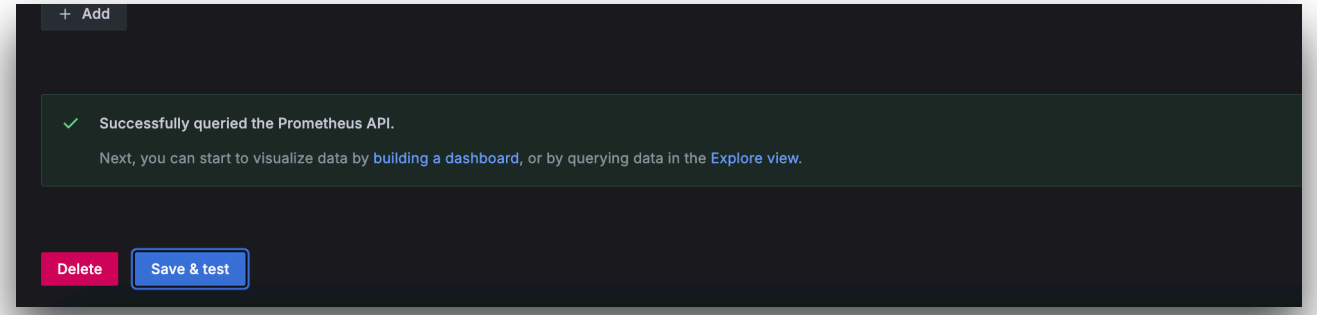
6. ➕ Add Prometheus as a Data Source in Grafana

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







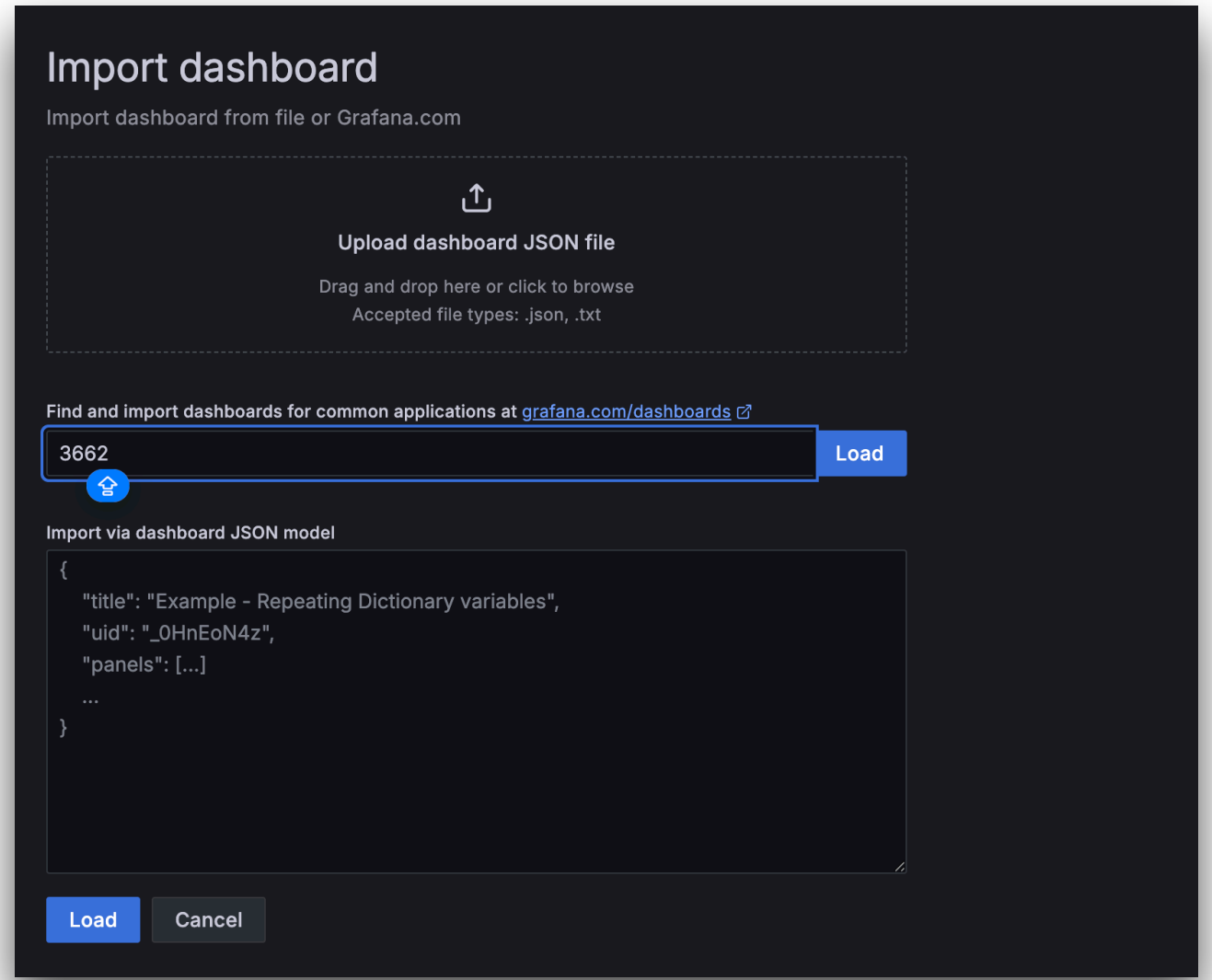


A successfully added data source says^

---

7. Build Dashboards:

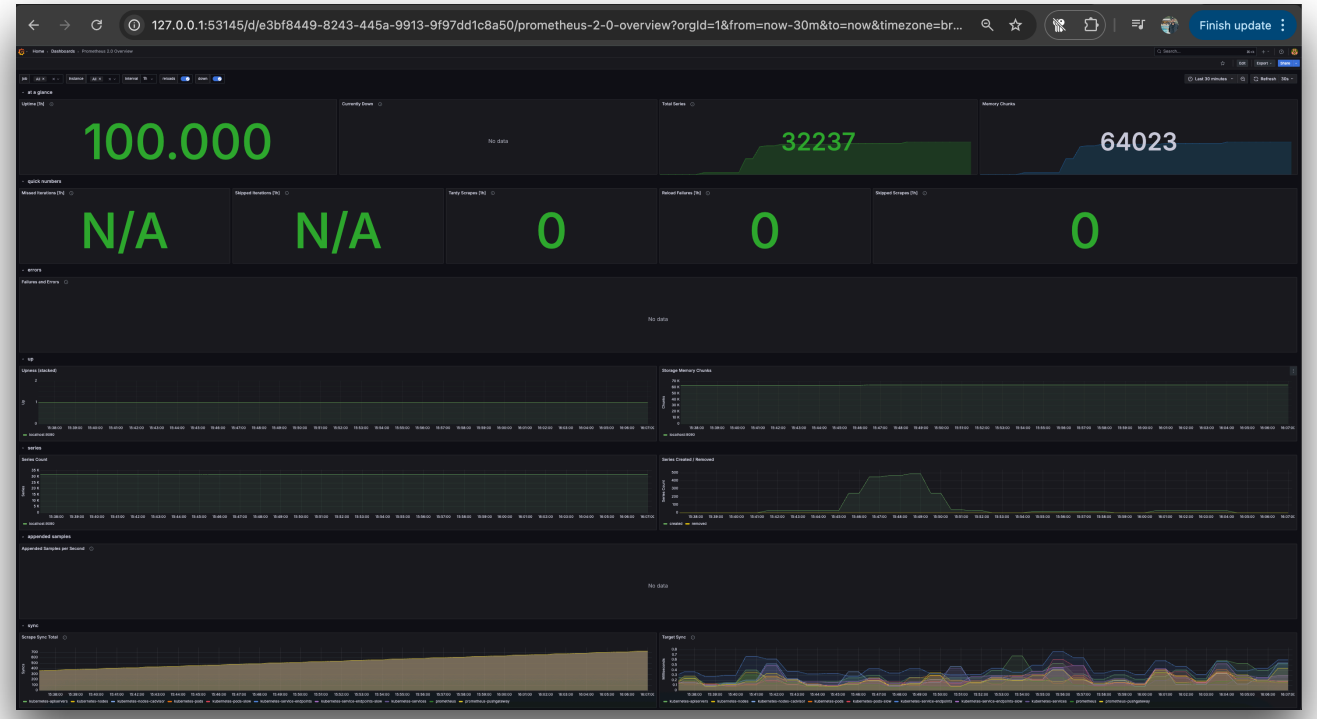
- Go to Dashboard > new > import
- Enter ID: 3662 (#this is a pre-defined dashboard for a Kubernetes cluster)
- Select the Prometheus data source
- Click Import

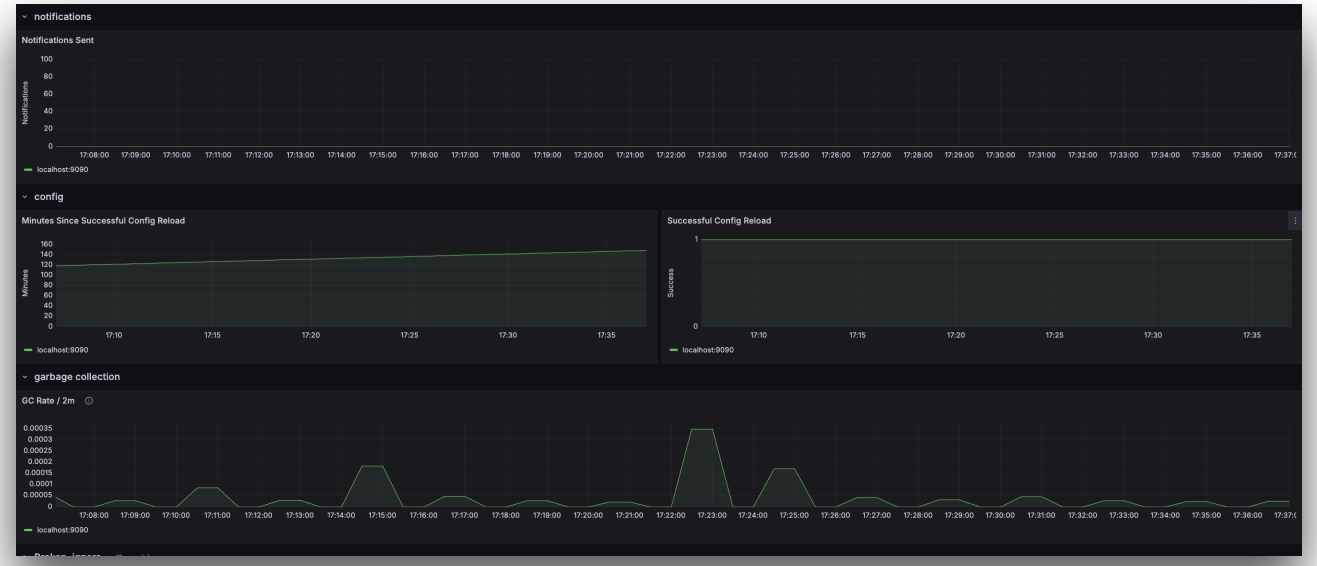


View Live Metrics in Grafana

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.

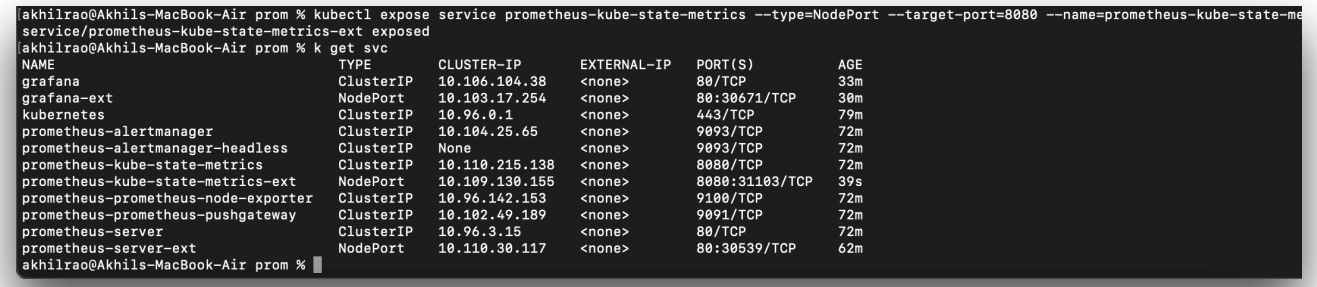
———

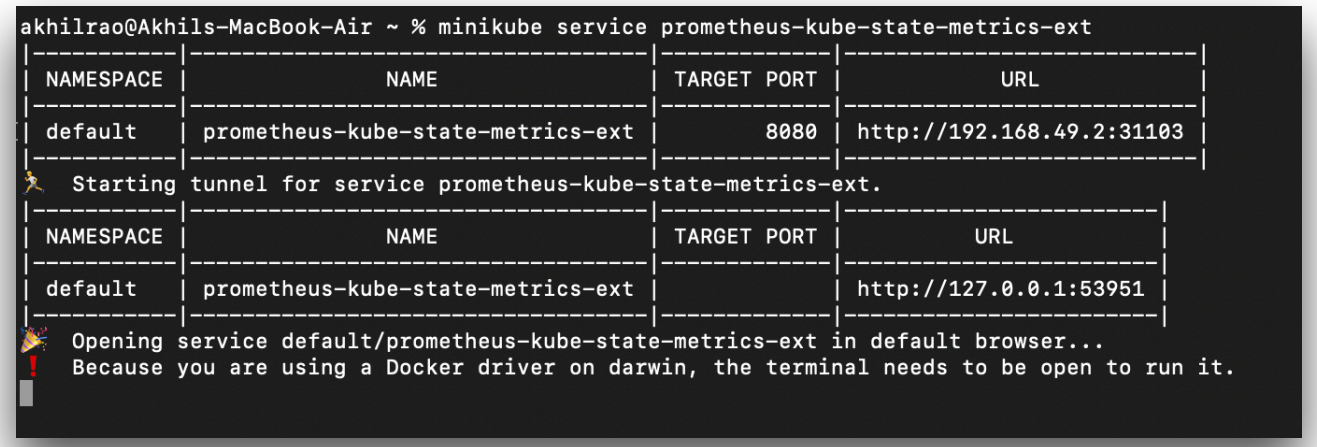
Expose kube-state-metrics via NodePort

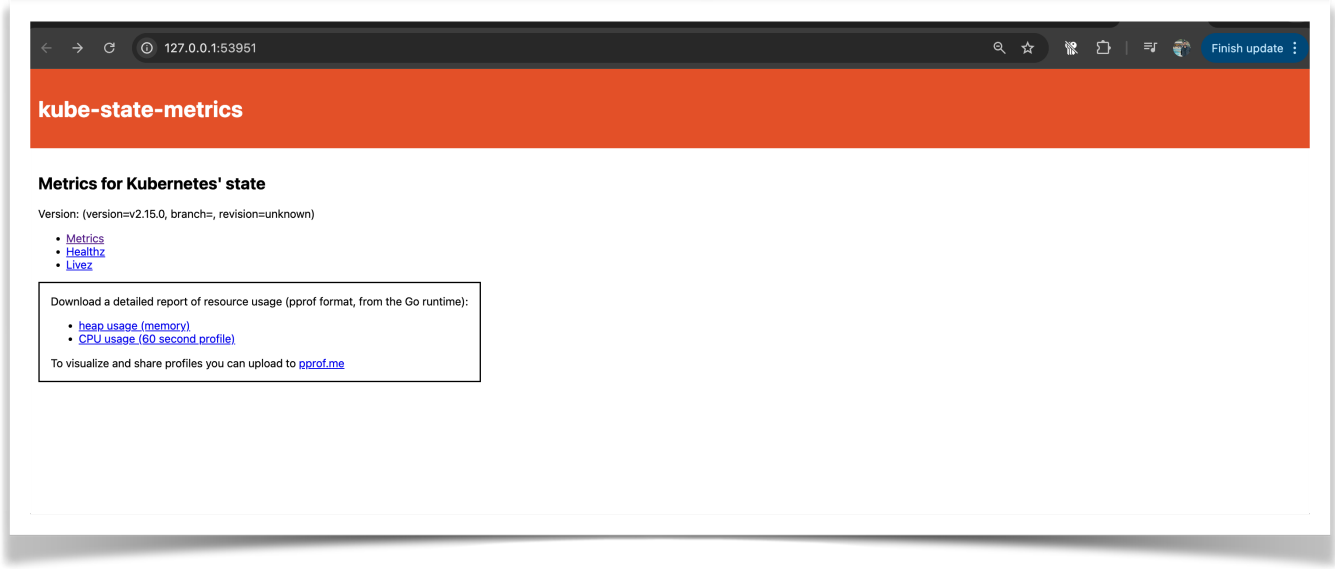
```
kubectl expose service prometheus-kube-state-metrics --type=NodePort --
target-port=8080 —name=prometheus-kube-state-metrics-ext
```

Access it in your browser to confirm it's reachable:
http://<minikube-ip>:<prometheus-kube-state-ext_node-port>/metrics

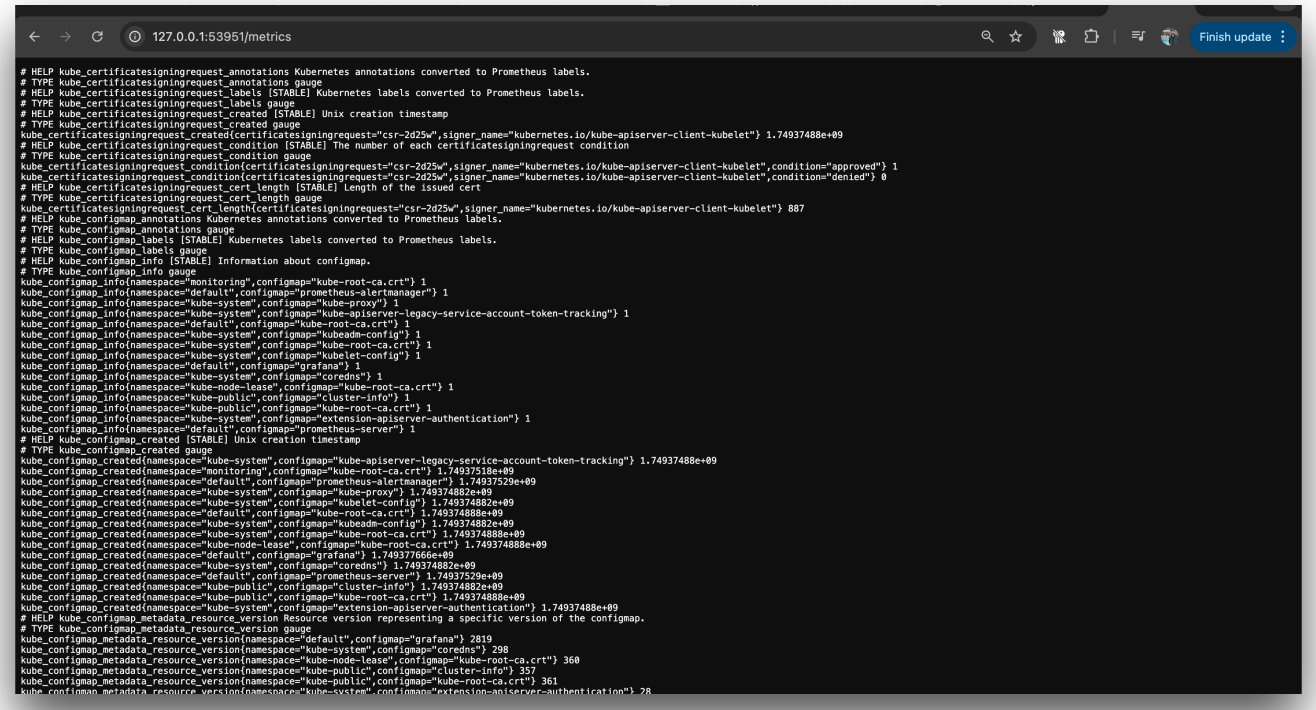
Or if using docker desktop, use
Minikube service prometheus-kube-state-metrics-ext






127.0.0.1:53951
Finish update
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


127.0.0.1:53951/metrics
Finish update
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_created [STABLE] Unix creation timestamp
TYPE kube_certificatesigningrequest_created gauge
kube_certificatesigningrequest_created{certificatesigningrequest="csr-2d25w",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{certificatesigningrequest="csr-2d25w",signer_name="kubernetes.io/kube-apiserver-client-kubelet",condition="approved"} 1
kube_certificatesigningrequest_condition{certificatesigningrequest="csr-2d25w",signer_name="kubernetes.io/kube-apiserver-client-kubelet",condition="denied"} 0
HELP kube_certificatesigningrequest_cert_length [STABLE] Length of the issued cert
TYPE kube_certificatesigningrequest_cert_length gauge
kube_certificatesigningrequest_cert_length{certificatesigningrequest="csr-2d25w",signer_name="kubernetes.io/kube-apiserver-client-kubelet"} 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_info{namespace="monitoring",configmap="kube-root-ca.crt"} 1
kube_configmap_info{namespace="default",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="default",configmap="kube-root-ca.crt"} 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="default",configmap="grafana"} 1
kube_configmap_info{namespace="kube-system",configmap="coredns"} 1
kube_configmap_info{namespace="kube-node-lease",configmap="kube-root-ca.crt"} 1
kube_configmap_info{namespace="kube-public",configmap="cluster-info"} 1
kube_configmap_info{namespace="kube-public",configmap="kube-root-ca.crt"} 1
kube_configmap_info{namespace="kube-system",configmap="extension-apiserver-authentication"} 1
kube_configmap_info{namespace="default",configmap="prometheus-server"} 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"} 1.74937518e+09
kube_configmap_created{namespace="default",configmap="prometheus-alertmanager"} 1.74937529e+09
kube_configmap_created{namespace="kube-system",configmap="kube-proxy"} 1.749374882e+09
kube_configmap_created{namespace="kube-system",configmap="kubelet-config"} 1.749374882e+09
kube_configmap_created{namespace="default",configmap="kube-root-ca.crt"} 1.749374888e+09
kube_configmap_created{namespace="kube-system",configmap="kubeadm-config"} 1.749374882e+09
kube_configmap_created{namespace="kube-system",configmap="kube-root-ca.crt"} 1.749374888e+09
kube_configmap_created{namespace="kube-node-lease",configmap="kube-root-ca.crt"} 1.749374888e+09
kube_configmap_created{namespace="default",configmap="grafana"} 1.749377666e+09
kube_configmap_created{namespace="kube-system",configmap="coredns"} 1.749374882e+09
kube_configmap_created{namespace="default",configmap="prometheus-server"} 1.74937529e+09
kube_configmap_created{namespace="kube-public",configmap="cluster-info"} 1.749374882e+09
kube_configmap_created{namespace="kube-public",configmap="kube-root-ca.crt"} 1.749374888e+09
kube_configmap_created{namespace="kube-system",configmap="extension-apiserver-authentication"} 1.74937488e+09
HELP kube_configmap_metadata_resource_version Resource version representing a specific version of the configmap.
TYPE kube_configmap_metadata_resource_version gauge
kube_configmap_metadata_resource_version{namespace="default",configmap="grafana"} 2819
kube_configmap_metadata_resource_version{namespace="kube-system",configmap="coredns"} 298
kube_configmap_metadata_resource_version{namespace="kube-node-lease",configmap="kube-root-ca.crt"} 360
kube_configmap_metadata_resource_version{namespace="kube-public",configmap="cluster-info"} 357
kube_configmap_metadata_resource_version{namespace="kube-public",configmap="kube-root-ca.crt"} 361
kube_configmap_metadata_resource_version{namespace="kube-system",configmap="extension-apiserver-authentication"} 28

> Add kube-state-metrics to Prometheus Scrape Config

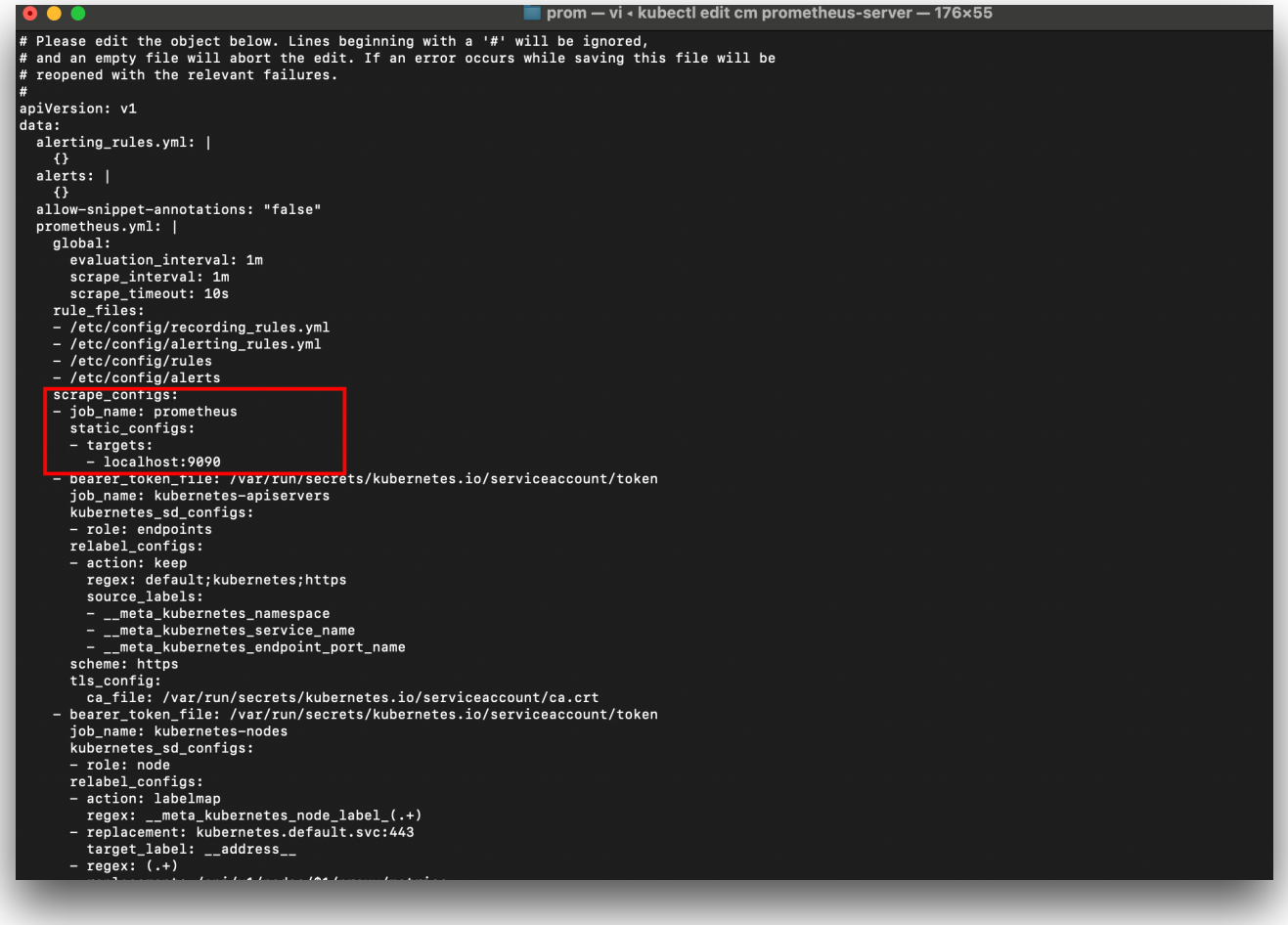
- Edit the prometheus-server config map:

```
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>']
```



- 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:

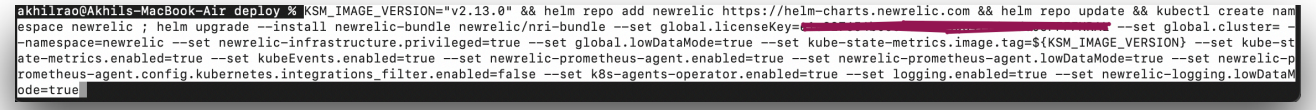
- Minikube
- Prometheus
- Grafana
- kube-state-metrics

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

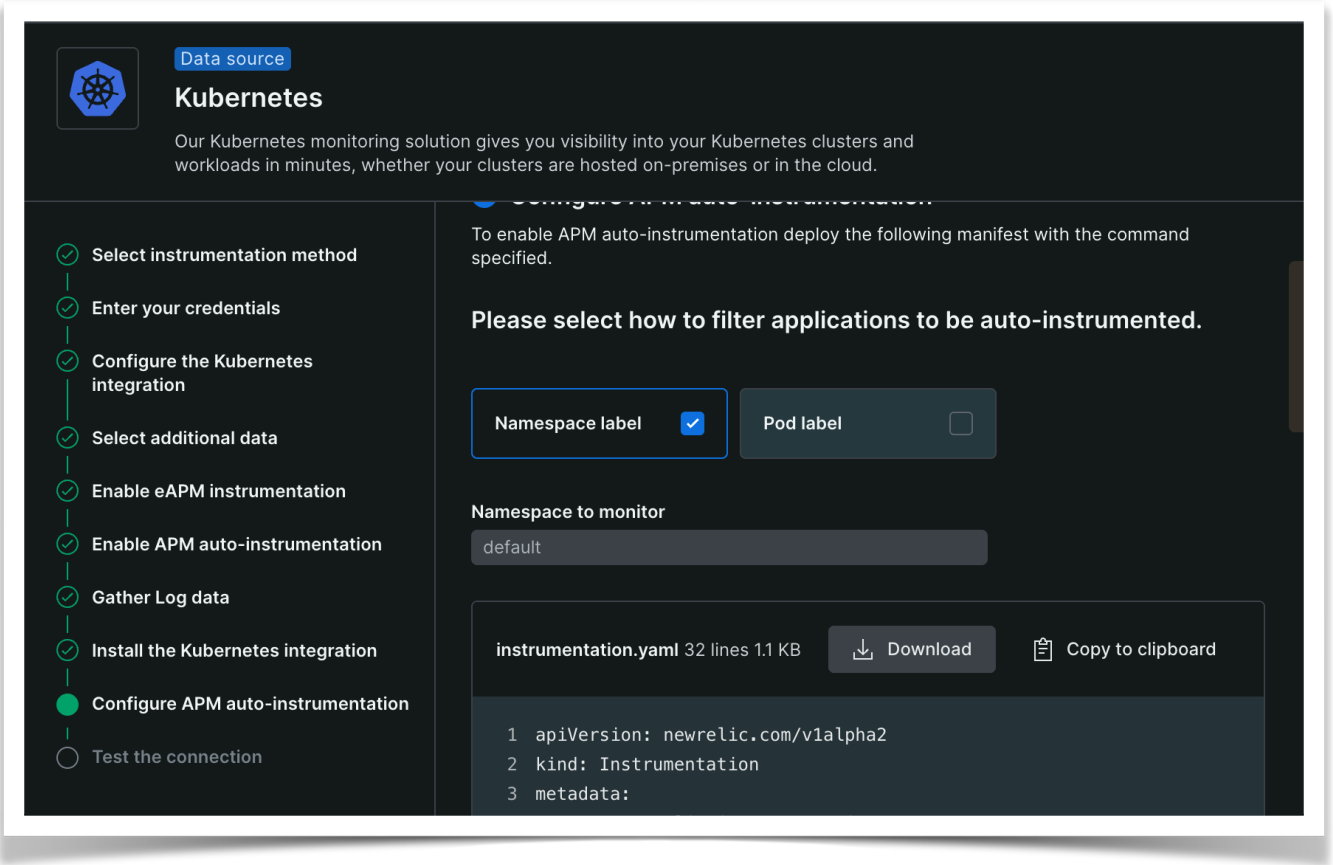
Newrelic

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

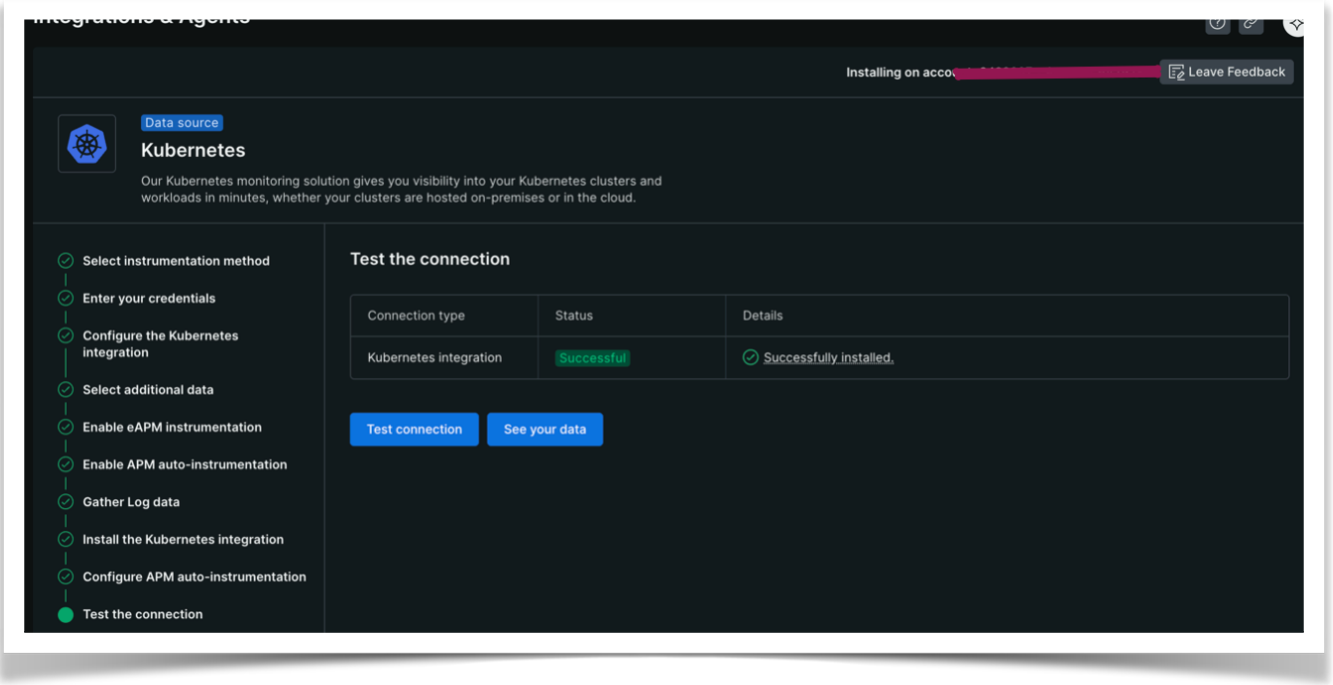
```
KSM_IMAGE_VERSION="v2.13.0" && helm repo add newrelic https://helm-
charts.newrelic.com && helm repo update && kubectl create namespace
newrelic ; helm upgrade --install newrelic-bundle newrelic/nri-bundle --
set global.licenseKey=<your_license_key_new_relic_goes_here> --set
global.cluster= --namespace=newrelic --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
```



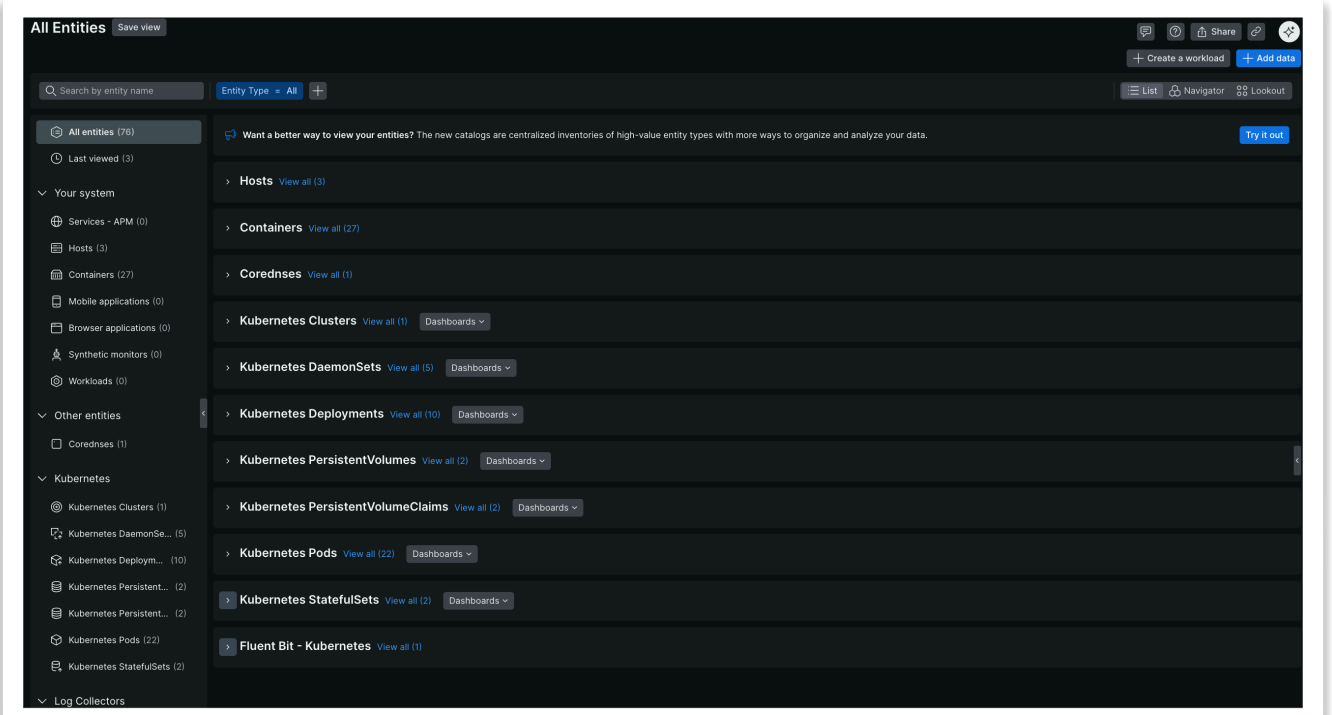
3. Configure APM auto-instrumentation:
   > copy the instrumentation.yaml to your cluster
   > `kubectl apply -f ./instrumentation.yaml -n newrelic`



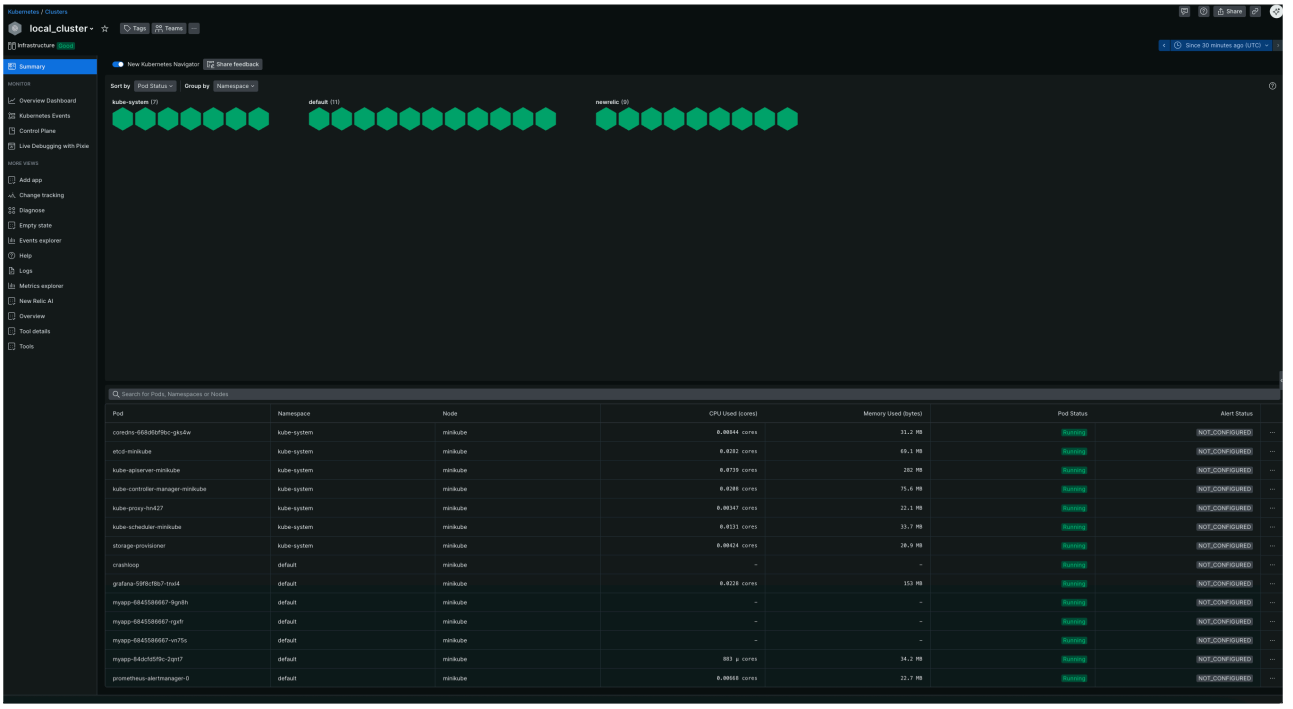
4. Test the connection > successful



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

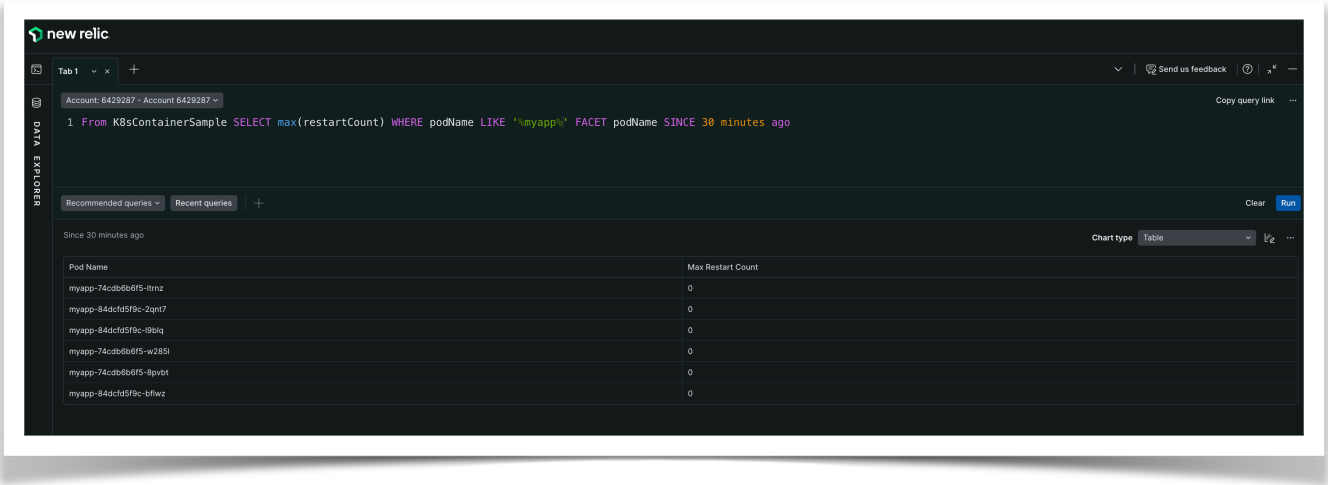


Kubernetes/Clusters
local_cluster:



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

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

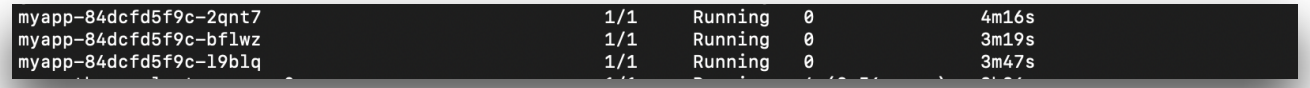


1. **Check the pod status of the app using NRQL(newrelic query language) to validate the data ingestion:**

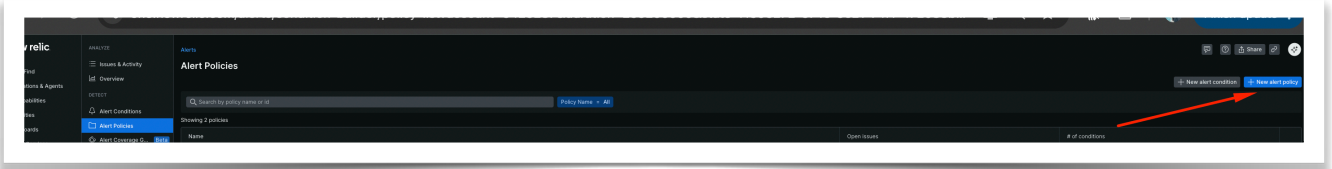
Query: `From K8sContainerSample SELECT max(restartCount) WHERE podName LIKE '%myapp%' FACET podName SINCE 30 minutes ago`

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.

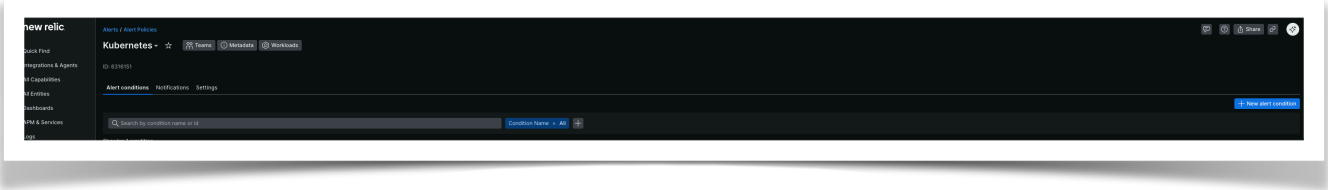
```
Kubectl get pods | grep my app
```



2. Now, **create an alert policy and alert condition**:
Alerts > Alert Policies > New Alert Policy > Kubernetes



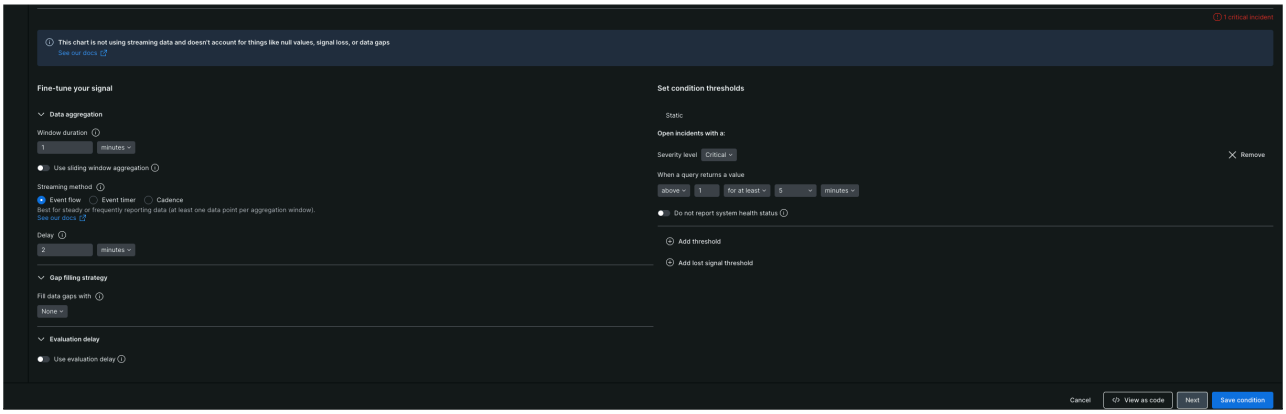
Kubernetes > New alert condition

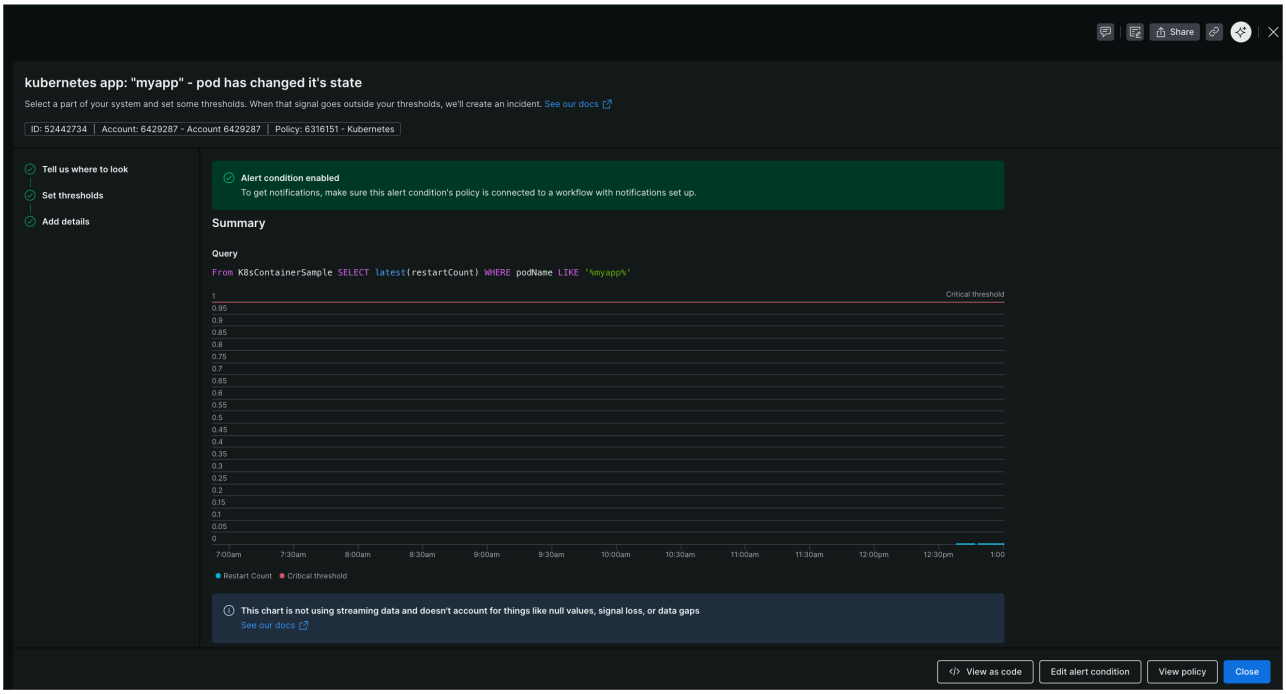


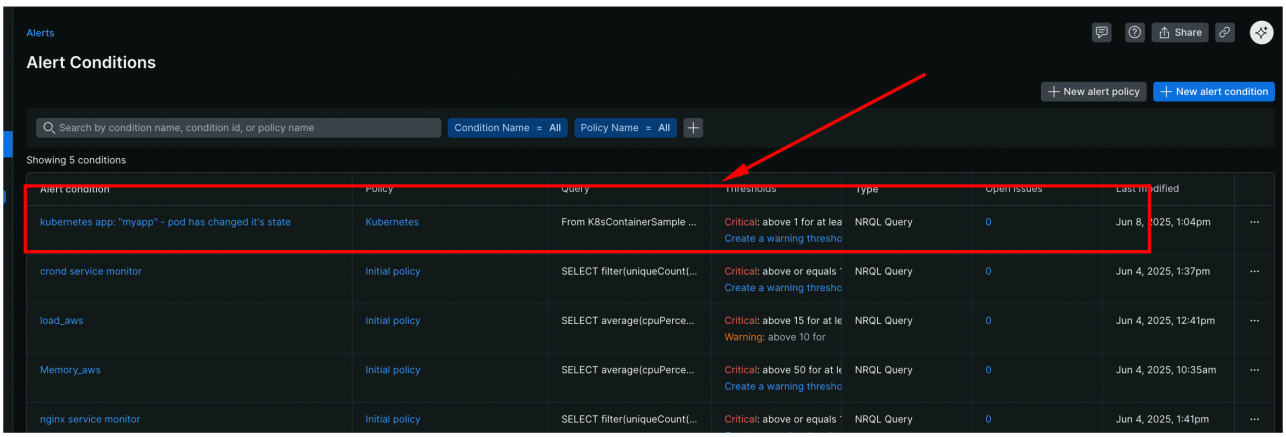
3. **Alert condition setup**:

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

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







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

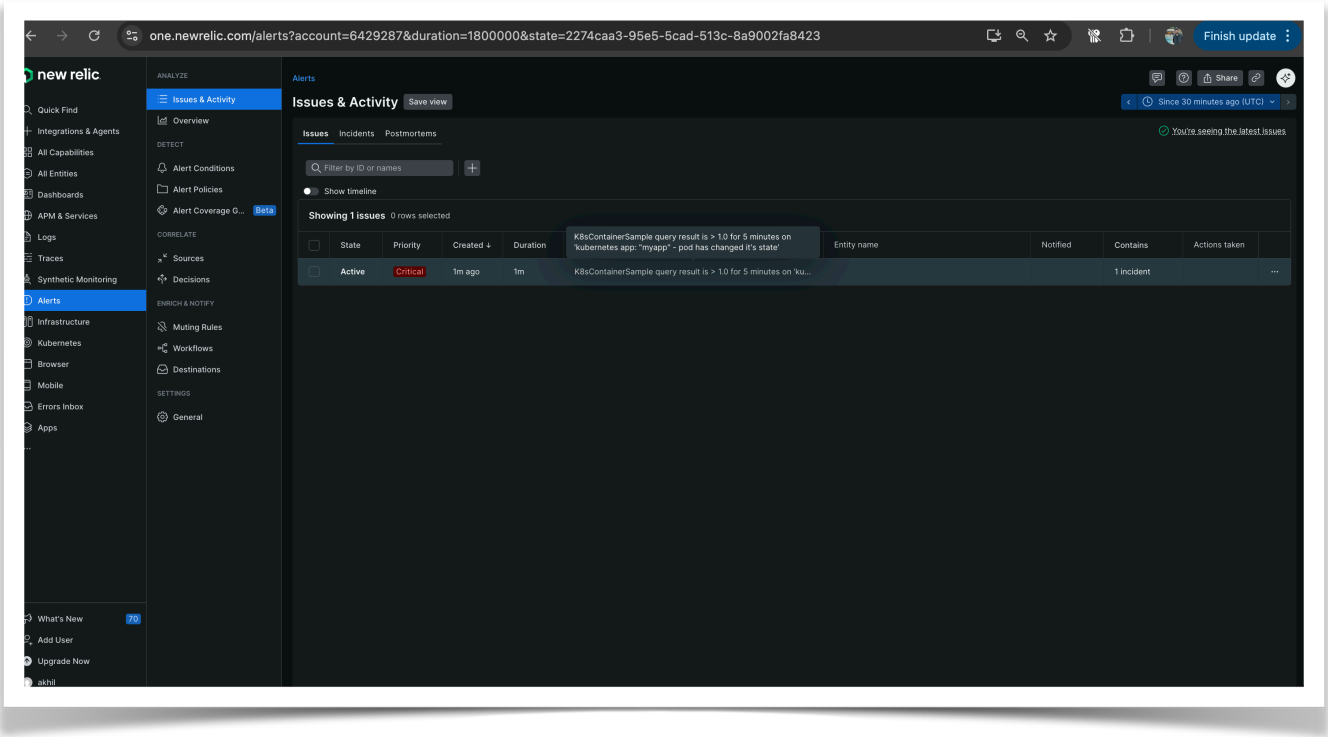
```
akhilrao@Akhils-MacBook-Air deploy % cat deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp
spec:
  replicas: 3
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
      - name: myapp
     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

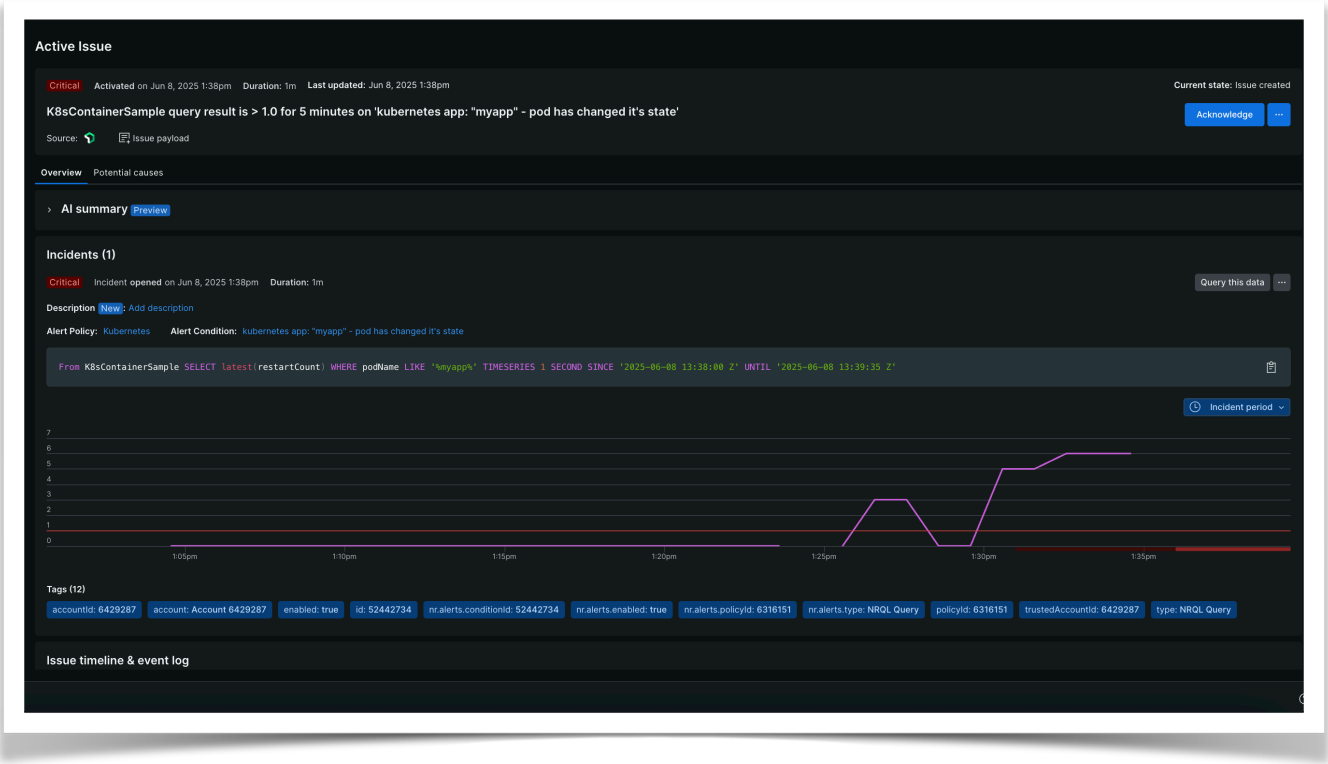
`Kubectl apply -f deployment.yaml`

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

```
[akhilrao@Akhils-MacBook-Air deploy % kubectl get pods | grep myapp
myapp-6845586667-9gn8h                    0/1     CrashLoopBackOff   17 (4m8s ago)    69m
myapp-6845586667-rgxfr                    0/1     CrashLoopBackOff   17 (5m9s ago)    71m
myapp-6845586667-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





ALERT TRIGGERED SUCCESSFULLY!!

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