1- Prepare Kubeadm env (EC2s on AWS)
2- Docker (Manual Installation)
3- Nodejs app
4- Prometheus & Grafana(Manual Installation)

```
ubuntu@masternode:~$ kubectl get ns
NAME              STATUS   AGE
default           Active   3h22m
kube-flannel      Active   3h21m
kube-node-lease   Active   3h22m
kube-public       Active   3h22m
kube-system       Active   3h22m
monitoring        Active   3h17m
```

Open the required ports using kubectl expose commands.

```
ubuntu@masternode:~$ kubectl get svc -n monitoring
NAME                                        TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                        AGE
alertmanager-operated                       ClusterIP   None             <none>        9093/TCP,9094/TCP,9094/UDP     3h17m
prometheus-grafana                          ClusterIP   10.105.94.63     <none>        80/TCP                         3h17m
prometheus-grafana-ext                      NodePort    10.111.141.235   <none>        80:30403/TCP                   3h7m
prometheus-kube-prometheus-alertmanager     ClusterIP   10.103.161.117   <none>        9093/TCP,8080/TCP              3h17m
prometheus-kube-prometheus-alertmanager-ext NodePort    10.102.253.183   <none>        9093:32677/TCP,8080:30930/TCP  171m
prometheus-kube-prometheus-operator         ClusterIP   10.107.215.193   <none>        443/TCP                        3h17m
prometheus-kube-prometheus-prometheus       ClusterIP   10.99.167.195    <none>        9090/TCP,8080/TCP              3h17m
prometheus-kube-prometheus-prometheus-ext   NodePort    10.108.204.107   <none>        9090:30238/TCP,8080:31353/TCP  3h11m
prometheus-kube-state-metrics               ClusterIP   10.98.129.203    <none>        8080/TCP                       3h17m
prometheus-operated                         ClusterIP   None             <none>        9090/TCP                       3h17m
prometheus-prometheus-node-exporter         ClusterIP   10.98.156.62     <none>        9100/TCP                       3h17m
```

Access the Prometheus and Grafana UIs using IP:NodePort. *(Grafana password: prom--operator)*

```
ubuntu@masternode:~$ mkdir nodejs
ubuntu@masternode:~$ cd nodejs/
ubuntu@masternode:~/nodejs$
```

```
ubuntu@masternode:~$ vim index.js

const express = require('express');                    // web framework to create a lightweigh server
const client = require('prom-client');               // prometheus library to collect metrcis

const app = express();
const port = 3000;                                                    // open port 3000

// Create a Registry to register the metrics
const register = new client.Registry();

// Add a default label which is added to all metrics
register.setDefaultLabels({
  app: 'nodejs_system_app'
});

// Enable the collection of default metrics
client.collectDefaultMetrics({ register });                          // Memory – Cpu – event loop lag – gc state

// Define a custom metric for total HTTP requests to the root path
const rootHttpRequestCounter = new client.Counter({
  name: 'http_requests_root_total',
  help: 'Total number of HTTP requests to the root path',
});

// Register the custom metric
register.registerMetric(rootHttpRequestCounter);

// Middleware to count every request to the root path
app.use((req, res, next) => {
  if (req.path === '/') {
    rootHttpRequestCounter.inc();
  }
  next();
});

// Define a route for Prometheus to scrape
app.get('/metrics', async (req, res) => {                        // open the endpoint /metrics that store information
  res.set('Content-Type', register.contentType);
  res.end(await register.metrics());
});

// Define the root route
app.get('/', (req, res) => {
  res.send('Hello From Node.js app Test');
});

// Start the server
app.listen(port, () => {
  console.log(`Example app listening at http://localhost:${port}`);
});
```

```
ubuntu@masternode:~$ vim Dockerfile

FROM node:lts
WORKDIR /usr/src/app
COPY . .
RUN npm install express prom-client
EXPOSE 3000
CMD  ["node", "index.js"]
```

```
ubuntu@masternode:~/nodejs$ docker build -t nodejs-app-prom .
[+] Building 5.1s (9/9) FINISHED                                                                                                                                 docker:default
 => [internal] load build definition from Dockerfile                                                                                                                       0.0s
 => => transferring dockerfile: 156B                                                                                                                                       0.0s
 => [internal] load metadata for docker.io/library/node:lts                                                                                                                0.4s
 => [internal] load .dockerignore                                                                                                                                          0.0s
 => => transferring context: 2B                                                                                                                                            0.0s
 => [1/4] FROM docker.io/library/node:lts@sha256:e515259afd26f60db74957c62203c93d45760f2ba864d94accfa2edfc1ac17cf                                                          0.0s
 => [internal] load build context                                                                                                                                          0.0s
 => => transferring context: 184B                                                                                                                                          0.0s
 => CACHED [2/4] WORKDIR /usr/src/app                                                                                                                                      0.0s
 => [3/4] COPY . .                                                                                                                                                         0.0s
 => [4/4] RUN npm install express prom-client                                                                                                                              4.1s
 => exporting to image                                                                                                                                                     0.4s
 => => exporting layers                                                                                                                                                    0.4s
 => => writing image sha256:21391370b13e5047165456ae10376cce0ea6f1161ece3b2f4d73639e5bbb09dd                                                                               0.0s
 => => naming to docker.io/library/nodejs-app-prom                                                                                                                         0.0s
```

```
ubuntu@masternode:~/nodejs$ docker images
REPOSITORY        TAG       IMAGE ID       CREATED              SIZE
nodejs-app-prom   latest    21391370b13e   About a minute ago   1.14GB
```

```
ubuntu@masternode:~/nodejs$ docker tag nodejs-app-prom:latest aliaceofali/nodejs-app-prom:v1

ubuntu@masternode:~/nodejs$ docker images
REPOSITORY                    TAG       IMAGE ID       CREATED              SIZE
aliaceofali/nodejs-app-prom   v1        21391370b13e   About a minute ago   1.14GB ## check for tag
nodejs-app-prom               latest    21391370b13e   About a minute ago   1.14GB
```

```
docker login
Login Succeeded
## verification
```

```
ubuntu@masternode:~/nodejs$ docker push aliaceofali/nodejs-app-prom:v1
The push refers to repository [docker.io/aliaceofali/nodejs-app-prom]
f515febcb2f4: Pushed
7617c43270a7: Pushed
8735979c33fc: Pushed
e5b77f05d0e5: Mounted from library/node
bd8201ae8f5d: Mounted from library/node
16af5fc601df: Mounted from library/node
2f10455f8757: Mounted from library/node
1b90aaaeb596: Mounted from library/node
a9b908787288: Mounted from library/node
20c8aca21338: Mounted from library/node
175a19836175: Mounted from library/node
v1: digest: sha256:6b99fe3af88641a2eb4d0de2f8bc95122127efa37d93a188ca4ebc915880afe1 size: 2630
```

Check your image on your DockerHub account.

Name	Last Pushed	Contains	Visibility	Scout
aliaceofali/nodejs-app-prom	less than a minute ag	IMAGE	Public	Inactive

Deployment Steps

- **Create a deployment.**
- **Expose it and check access using the NodePort.**

```
ubuntu@masternode:~/nodejs$ kubectl get deploy
No resources found in default namespace.


ubuntu@masternode:~/nodejs$ kubectl get po
No resources found in default namespace.


ubuntu@masternode:~$ vim nodejs-app.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  name: nodejs-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nodejs
  template:
    metadata:
      labels:
        app: nodejs
    spec:
      containers:
        - name: nodejs
          image: aliaceofali/nodejs-app-prom:v1
          ports:
            - containerPort: 3000

ubuntu@masternode:~/nodejs$ kubectl apply -f nodejs-app.yaml
deployment.apps/nodejs-app created

ubuntu@masternode:~/nodejs$ kubectl get deploy
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
nodejs-app   3/3     3            3           27s

ubuntu@masternode:~/nodejs$ kubectl get rs
NAME                    DESIRED   CURRENT   READY   AGE
nodejs-app-8549c8b8db   3         3         3       29s

ubuntu@masternode:~/nodejs$ kubectl get po
NAME                         READY   STATUS    RESTARTS   AGE
nodejs-app-8549c8b8db-cjj7v   1/1     Running   0          23s
nodejs-app-8549c8b8db-twmrn   1/1     Running   0          23s
nodejs-app-8549c8b8db-zgk5q   1/1     Running   0          23s

----

apiVersion: v1
kind: Service
metadata:
  name: nodejs-svc
  labels:
    app: nodejs
  annotations:
    prometheus.io/scrape: "true" # Enable scraping for Prometheus
spec:
  type: NodePort
    selector:
      app: nodejs
   ports:
    - port: 3000
      targetPort: 3000
      name: http-nodejs-app


ubuntu@masternode:~/nodejs$ kubectl apply -f nodejs-svc.yaml
service/nodejs-svc created

kubectl get svc
ubuntu@masternode:~/nodejs$ kubectl get svc
NAME         TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)          AGE
kubernetes   ClusterIP   10.96.0.1     <none>        443/TCP          3h45m
nodejs-svc   NodePort    10.97.14.36   <none>        3000:31557/TCP   11s


ip : 31557(NodePort)
```

Hello From Node.js app Test

Enable Prometheus to Monitor the Application (Not Just the Pods)

```
ubuntu@masternode:~/nodejs$ kubectl get po
NAME                        READY   STATUS    RESTARTS   AGE
nodejs-app-8549c8b8db-cjj7v   1/1     Running   0          3m7s
nodejs-app-8549c8b8db-twmrn   1/1     Running   0          3m7s
nodejs-app-8549c8b8db-zgk5q   1/1     Running   0          3m7s
```

In the Prometheus UI, execute the query: container_cpu_usage_seconds_total

- Use the Pod ID to search for results.
- You'll find how much CPU the application is using.
- Prometheus **did not** scrape metrics from the application — only from the pod itself.

```
container_cpu_usage_seconds_total{cpu="total", endpoint="https-metrics", id="/kubepods.slice/kubepods-besteffort.slice/kubepods-besteffort-poddf1d95a6_a39b_4536_a85a_437eb1add8ef.slice/cri-containerd-3a601598f37386d5bb4412dc4e9f71c0af235e577f0ba863c1510b515ff56a1e.scope", image="k8s.gcr.io/pause:3.5", instance="172.31.38.31:10250", job="kubelet", metrics_path="/metrics/cadvisor", name="3a601598f37386d5bb4412dc4e9f71c0af235e577f0ba863c1510b515ff56a1e", namespace="default", node="node1", pod="nodejs-app-8549c8b8db-cjj7v", service="prometheus-kube-prometheus-kubelet"}    0.030135
```

So let's tell Prometheus about the application by adding it as a **new target**.

To do this, we need to create a **ServiceMonitor** component in the **Prometheus namespace.**

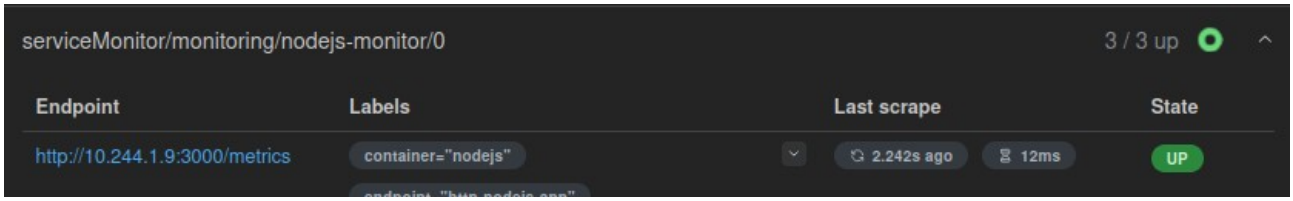
```
ubuntu@masternode:~/nodejs$ kubectl get ns
NAME              STATUS   AGE
default           Active   3h49m
kube-flannel      Active   3h49m
kube-node-lease   Active   3h49m
kube-public       Active   3h49m
kube-system       Active   3h49m
monitoring        Active   3h44m
```

```
ubuntu@masternode:~/nodejs$ kubectl apply -f nodejs-monitor.yaml
```

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: nodejs-monitor
  namespace: monitoring # Ensure this matches your monitoring namespace
  labels:
    release: prometheus # Adjust if you use a different labels ( depends on system )
spec:
  selector:
    matchLabels:
      app: nodejs # Match the label used in your nodejs-app.yaml
  namespaceSelector:
    matchNames:
      - default # where your nodejs-svc is deployed
  endpoints:
    - port: http-nodejs-app # Match the name of the port in your nodejs-svc.yaml
      path: /metrics # Path to scrape metrics from
```

```
ubuntu@masternode:~/nodejs$ kubectl apply -f nodejs-monitor.yaml
servicemonitor.monitoring.coreos.com/nodejs-monitor created
```

Open Prometheus and check the **Targets** page to see if the new application endpoint is added.

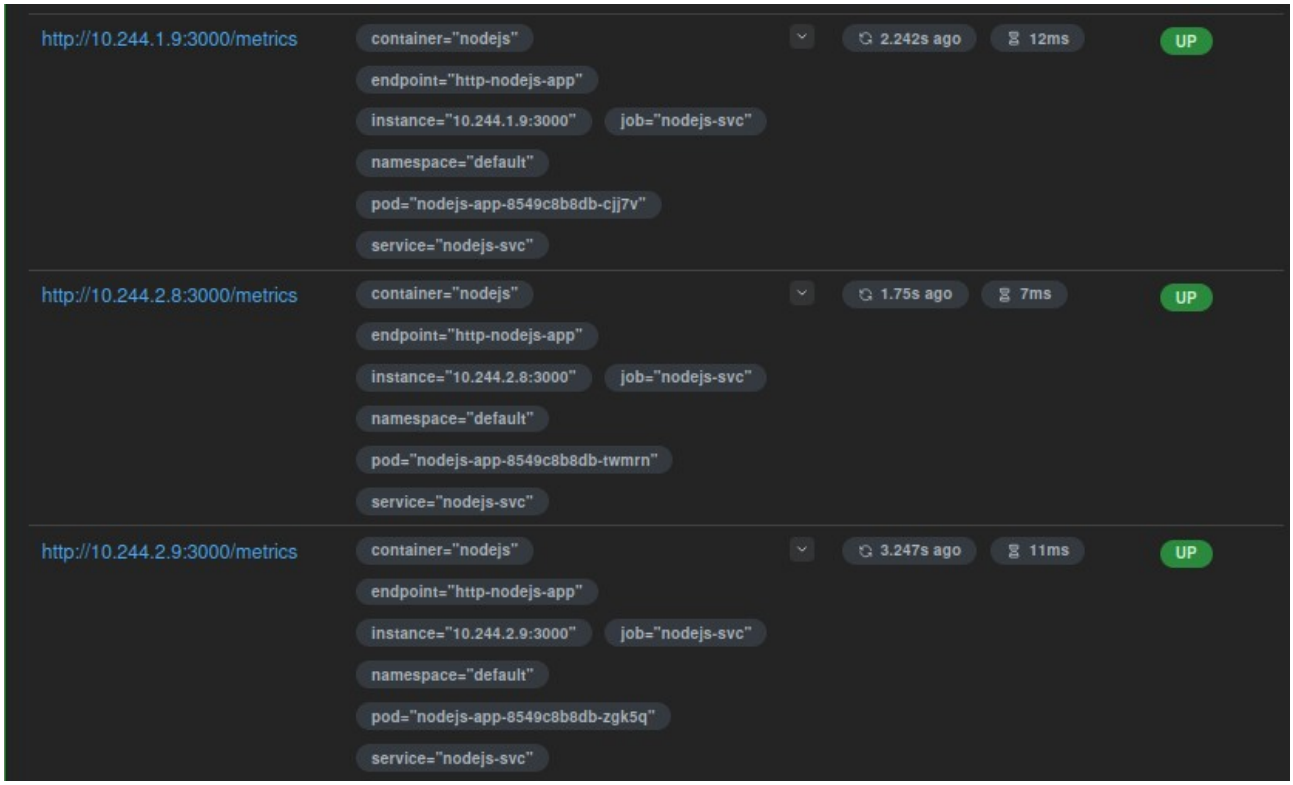


```
ubuntu@masternode:~/nodejs$ kubectl get po -o wide
NAME                          READY   STATUS    RESTARTS   AGE   IP           NODE    NOMINATED NODE   READINESS GATES
nodejs-app-8549c8b8db-cjj7v   1/1     Running   0          11m   10.244.1.9   node1   <none>           <none>
nodejs-app-8549c8b8db-twmrn   1/1     Running   0          11m   10.244.2.8   node2   <none>           <none>
nodejs-app-8549c8b8db-zgk5q   1/1     Running   0          11m   10.244.2.9   node2   <none>           <none>
```

You'll find the endpoint listed under **Targets**, along with its IP. Now Prometheus can monitor both the pod **and** the application inside it.

Add a New Target Using ServiceMonitor

- The prometheus-operator will automatically discover and start scraping the new target.

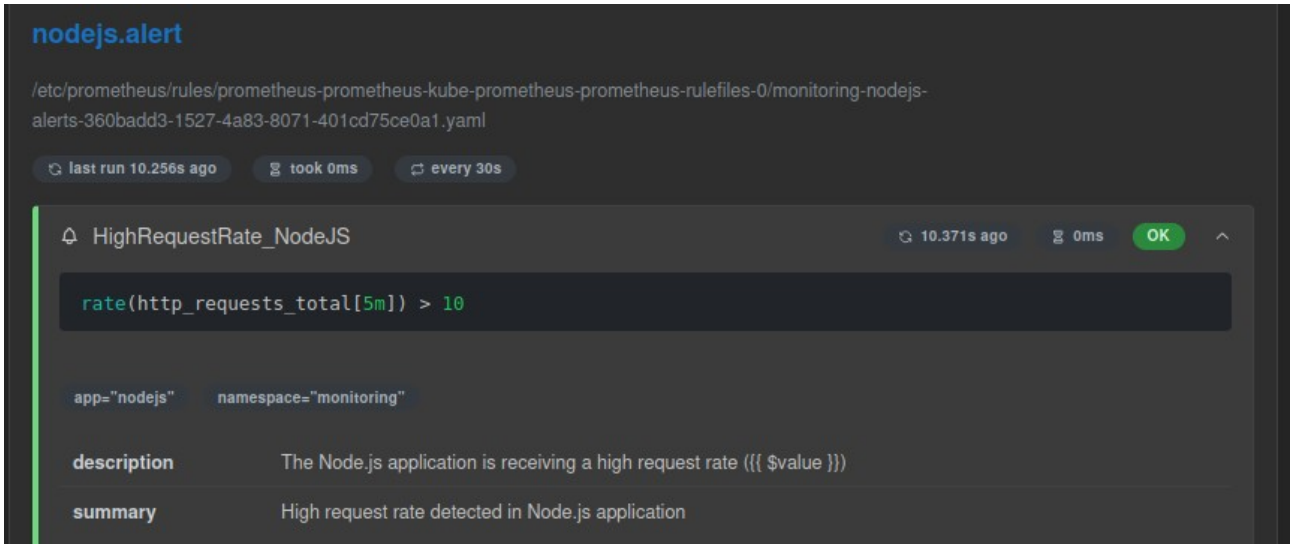


Create a Rule

- Let’s now create an alerting rule.
- You can check configured rules in the **UI under Alerts/Rules.**
- Configure the rule to send alerts to **Slack.**
- Create a Bash script to send many requests to the application, simulating **stress or load.**
- Observe the behavior in the UI — even in networking namespaces if applicable.

```yaml
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  name: nodejs-alerts
  namespace: monitoring # Ensure this matches your monitoring namespace
  labels:
    app: kube-prometheus-stack
    release: prometheus # it must match the label used in your Prometheus default configuration
spec:
  groups:
    - name: nodejs.alert  # Name of the alert group
      rules:
        - alert: HighRequestRate_NodeJS
          expr: rate(http_requests_root_total[5m]) > 10 # expression means high request rate
          for: 0m
          labels:
            app: nodejs
            namespace: monitoring # Ensure this matches your monitoring namespace
          annotations:
            summary: "High request rate detected in Node.js application"
            description: "The Node.js application is receiving a high request rate ({{ $value }})"
```

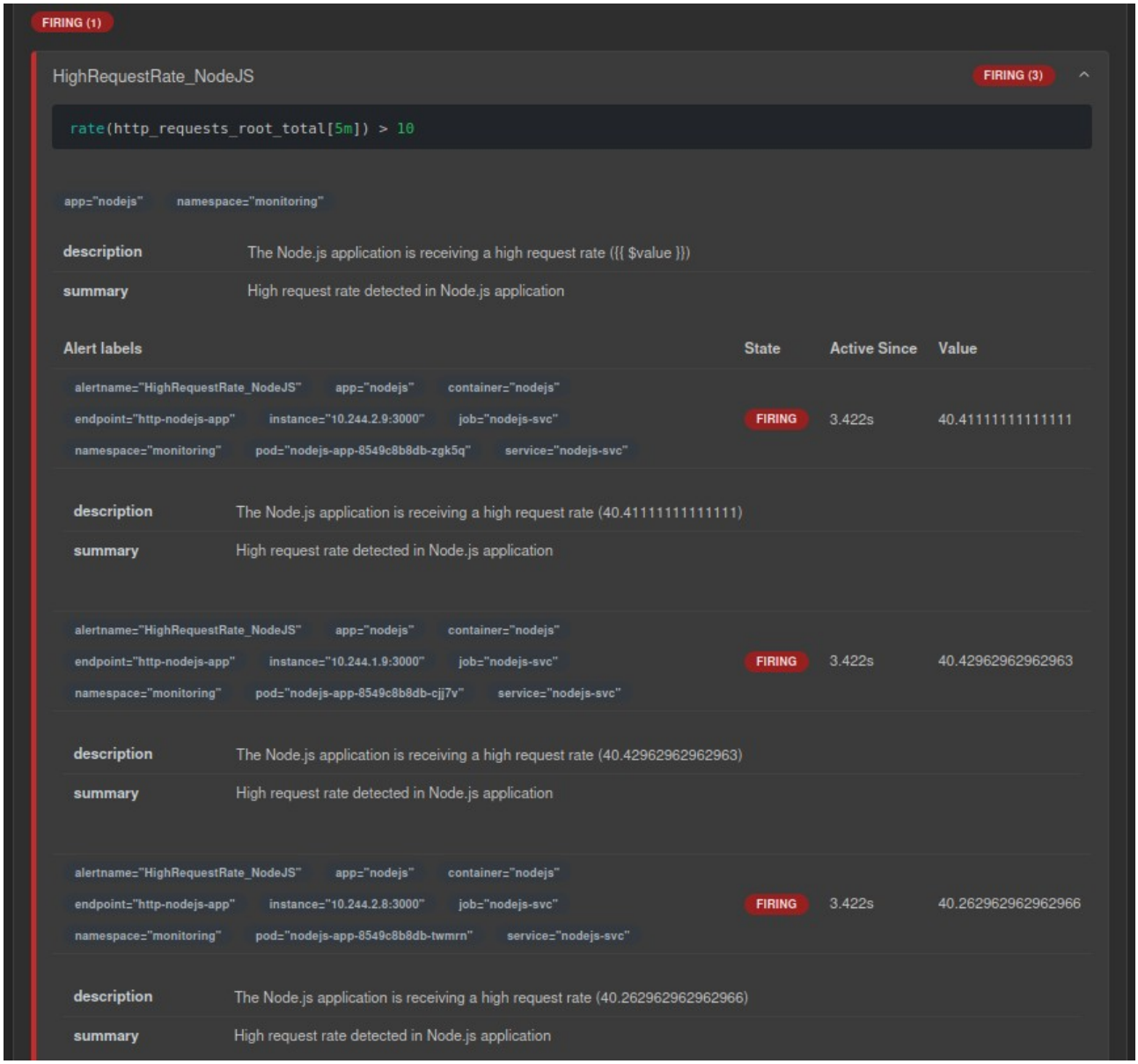
```
ubuntu@masternode:~/nodejs$ kubectl apply -f nodejs-rule.yaml
prometheusrule.monitoring.coreos.com/nodejs-alerts created
```

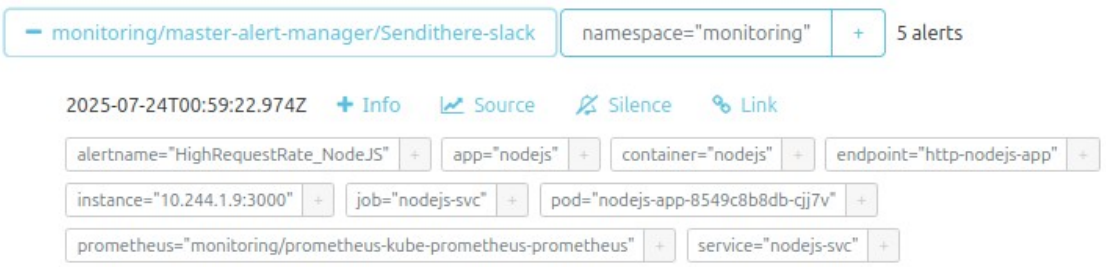


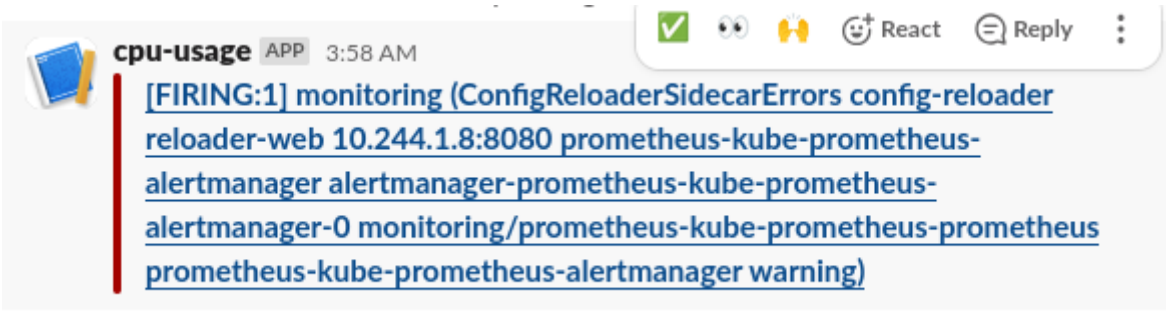
```
ubuntu@masternode:~/nodejs$ kubectl get secret -n monitoring
NAME                                                                  TYPE                  DATA   AGE
alertmanager-prometheus-kube-prometheus-alertmanager                  Opaque                1      3h56m
alertmanager-prometheus-kube-prometheus-alertmanager-cluster-tls-config  Opaque                1      3h56m
alertmanager-prometheus-kube-prometheus-alertmanager-generated        Opaque                1      3h56m
alertmanager-prometheus-kube-prometheus-alertmanager-tls-assets-0     Opaque                0      3h56m
alertmanager-prometheus-kube-prometheus-alertmanager-web-config       Opaque                1      3h56m
prometheus-grafana                                                    Opaque                3      3h56m
prometheus-kube-prometheus-admission                                  Opaque                3      3h56m
prometheus-prometheus-kube-prometheus-prometheus                      Opaque                1      3h56m
prometheus-prometheus-kube-prometheus-prometheus-thanos-prometheus-http-client-file   Opaque   1      3h56m
prometheus-prometheus-kube-prometheus-prometheus-tls-assets-0         Opaque                1      3h56m
prometheus-prometheus-kube-prometheus-prometheus-web-config           Opaque                1      3h56m
sh.helm.release.v1.prometheus.v1                                      helm.sh/release.v1    1      3h56m
slack-secret                                                          Opaque                1      3h6m


ubuntu@masternode:~/nodejs$ kubectl describe secret slack-secret -n monitoring
Name:         slack-secret
Namespace:    monitoring
Labels:       <none>
Annotations:  <none>

Type:  Opaque

Data
====
webhook:  81 bytes


ubuntu@masternode:~/nodejs$ vim nodejs-alert-manager.yaml

apiVersion: monitoring.coreos.com/v1alpha1
kind: AlertmanagerConfig
metadata:
  name: nodejs-alert-manager
  namespace: monitoring # Ensure this matches your monitoring namespace
spec:
  route:
    receiver: 'nodejs-slack'
    repeatInterval: 30m
    routes:
      - matchers:
          - name: alert-https
            value: HighRequestRate_NodeJS # Match the alert name you created in nodejs-rule.yaml
        repeatInterval: 10m
  receivers:
    - name: 'nodejs-slack'
      slackConfigs:
        - apiURL:
            key: webhook
            name: slack-secret # Ensure this matches your secret name
          channel: '#https_requests' # Adjust to your Slack channel
          sendResolved: true

ubuntu@masternode:~/nodejs$ kubectl apply -f nodejs-alert-manager.yaml
alertmanagerconfig.monitoring.coreos.com/nodejs-alert-manager created


ubuntu@masternode: kubectl get alertmanagerconfig -n monitoring

ubuntu@masternode:~/nodejs$ kubectl get alertmanagerconfig -n monitoring
NAME                   AGE
nodejs-alert-manager   35s


ubuntu@masternode:~/nodejs$ kubectl get svc
NAME         TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)          AGE
kubernetes   ClusterIP   10.96.0.1     <none>        443/TCP          4h12m
nodejs-svc   NodePort    10.97.14.36   <none>        3000:31557/TCP   27m


ubuntu@masternode:~/nodejs$ vim forcerequests.sh

#!/bin/bash

send_requests() {
    while true; do
        curl -sS http://13.61.174.226:31557/  > /dev/null
        echo "Request sent"
        sleep 0.0667
    done
}

for ((i=1; i<=15; i++)); do
    send_requests &
    pids[$i]=$!
done

trap 'echo "Exiting..."; kill ${pids[*]}; exit' INT

wait

ubuntu@masternode:~/nodejs$ sudo chmod +x forcerequests.sh
ubuntu@masternode:~/nodejs$ ./forcerequests.sh
Request sent
Request sent
Request sent
```

Add Visualization in Grafana

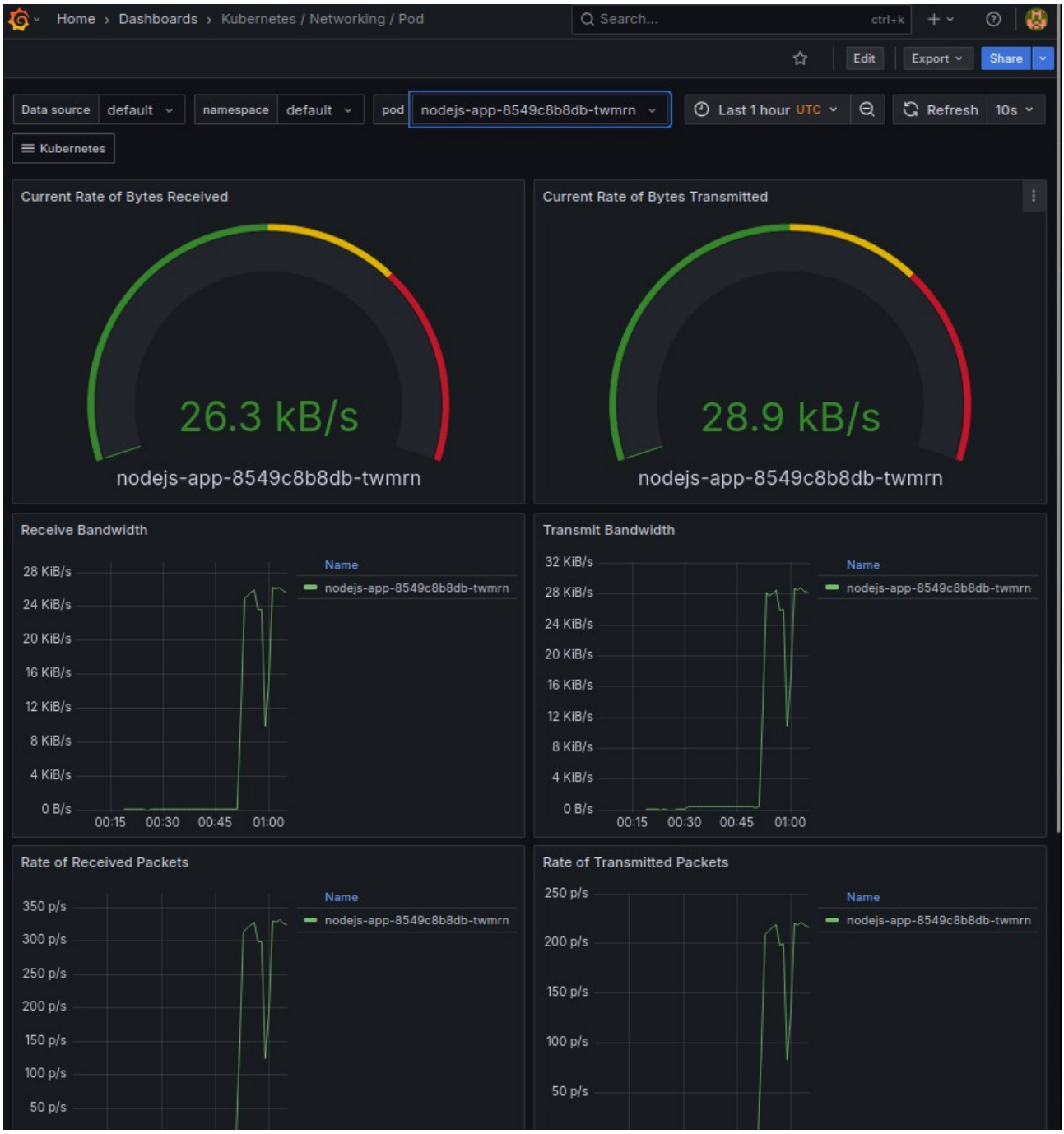
- Take the Prometheus query expression and add it to **Grafana.**
- Watch the graph showing your application's metrics in real-time.
- This helps you visualize request traffic and performance data.





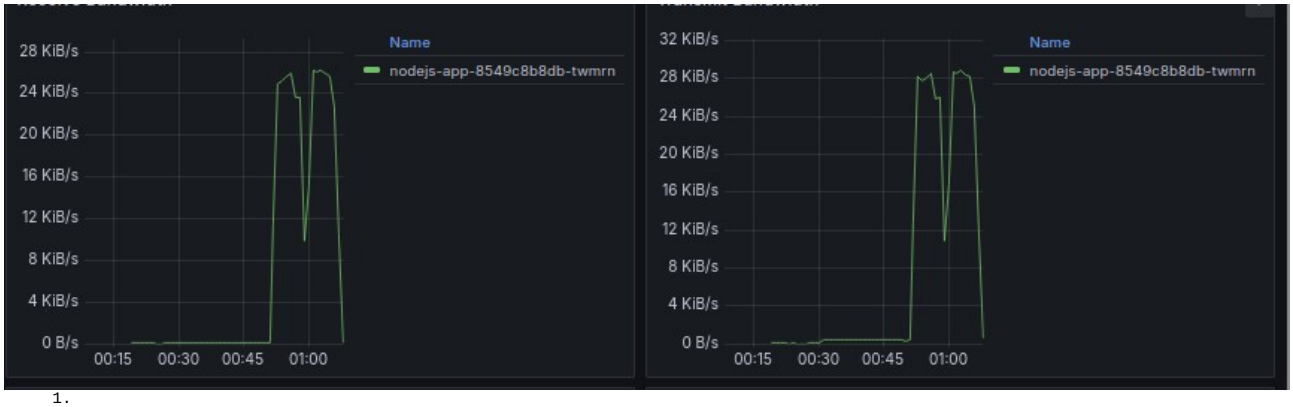


let’s stop the script


Home › Dashboards › Kubernetes / Networking / Pod
Search...
ctrl+k
Edit
Export
Share
Data source
default
namespace
default
pod
nodejs-app-8549c8b8db-twmrn
Last 1 hour UTC
Refresh
10s
Kubernetes
Current Rate of Bytes Received
26.3 kB/s
nodejs-app-8549c8b8db-twmrn
Current Rate of Bytes Transmitted
28.9 kB/s
nodejs-app-8549c8b8db-twmrn
Receive Bandwidth
28 KiB/s
24 KiB/s
20 KiB/s
16 KiB/s
12 KiB/s
8 KiB/s
4 KiB/s
0 B/s
00:15
00:30
00:45
01:00
Name
nodejs-app-8549c8b8db-twmrn
Transmit Bandwidth
32 KiB/s
28 KiB/s
24 KiB/s
20 KiB/s
16 KiB/s
12 KiB/s
8 KiB/s
4 KiB/s
0 B/s
00:15
00:30
00:45
01:00
Name
nodejs-app-8549c8b8db-twmrn
Rate of Received Packets
350 p/s
300 p/s
250 p/s
200 p/s
150 p/s
100 p/s
50 p/s
Name
nodejs-app-8549c8b8db-twmrn
Rate of Transmitted Packets
250 p/s
200 p/s
150 p/s
100 p/s
50 p/s
Name
nodejs-app-8549c8b8db-twmrn

Stop the Bash Script

- Once the stress test is done, stop the script.
- Then check the **Grafana graphs** again to observe how the application metrics respond.



1.