16 - Monitoring with Prometheus

Exercises for Module "Monitoring with Prometheus"

> Use repository: https://gitlab.com/devops-bootcamp3/bootcamp-java-mysql/-/tree/feature/monitoring

Context

You and your team are running the following setup in the K8s cluster:

- Java application that uses Mysql DB and is accessible from browser using Ingress. It's all running fine, but sometimes you have issues where Mysql DB is not accessible or Ingress has some issues and users can't access your Java application. And when this happens, you and your team spend a lot of time figuring out what the issue is and troubleshooting within the cluster. Also, most of the time when these issues happen, you are not even aware of them until an application user writes to the support team that the application isn't working or developers write you an email that things need to be fixed.
- As an improvement, you have decided to increase visibility in your cluster to know immediately when such issues happen and proactively fix them. Also, you want a more efficient way to pinpoint the issues right away, without hours of troubleshooting. And maybe even prevent such issues from happening by staying alert to any possible issues and fixing them before they even happen.
- Your manager suggested using Prometheus, since it's a well known tool with a large community and is widely used, especially in K8s environment.
- So you and your team are super motivated to improve the application observability using Prometheus monitoring.

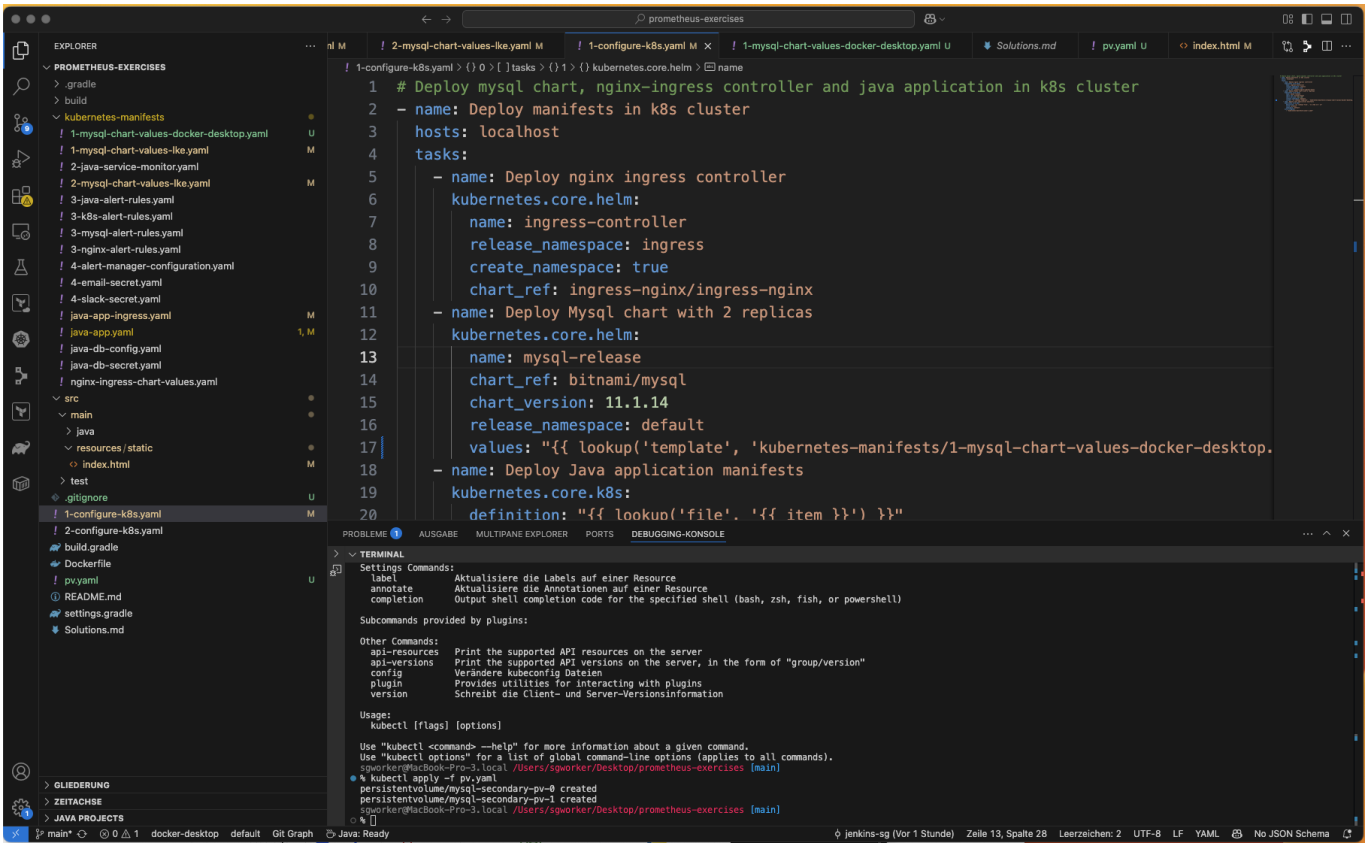
I used my repo on github: https://github.com/Saban39/prometheus-exercises

▶ Solution 1: Deploy your Application and Prepare the Setup

> EXERCISE 1: Deploy your Application and Prepare the Setup

- Create a K8s cluster
- Deploy Mysql database for your Java application with 2 replicas (You can use the following helm chart: https://github.com/bitnami/charts/tree/master/bitnami/mysql)
- Deploy Java Maven application with 3 replicas that talks to the Mysql DB
- Deploy Nginx Ingress Controller (You can use the following helm chart: https://github.com/kubernetes/ingress-nginx/tree/master/charts/ingress-nginx)
- Now configure access to your Java application using an Ingress rule
- You can use the Ansible playbook from Ansible exercises 7 & 8 with a few adjustments to configure this setup.

I decided to create the cluster using Docker Desktop.



This YAML file defines two PersistentVolumes (PVs) in Kubernetes. Each PersistentVolume provides 8 GiB of storage and is configured to be mounted using the hostPath method, meaning the data will be stored on the local filesystem of the host machine (useful for local development like with Docker Desktop).

```
kubectl apply -f pv.yaml
persistentvolume/mysql-secondary-pv-0 created
persistentvolume/mysql-secondary-pv-1 created
```

my pv.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mysql-secondary-pv-0
spec:
  capacity:
    storage: 8Gi
  accessModes:
    - ReadWriteOnce
  storageClassName: standard
  hostPath:
    path: /Users/sgworker/Desktop/ansible_exercises/ansible-
exercises/mysql-secondary-0

---
```

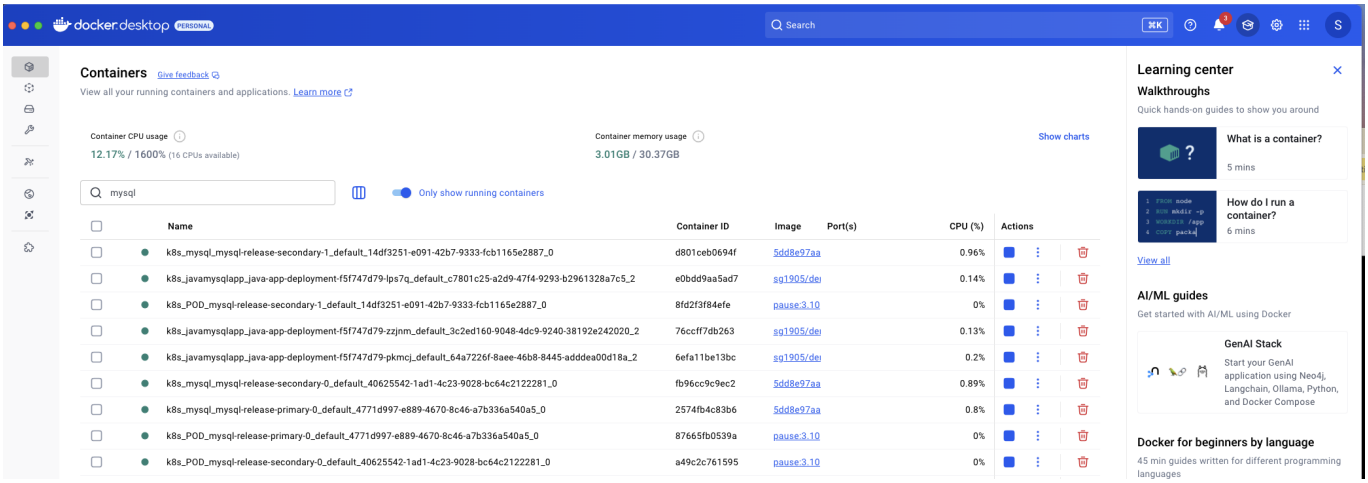
```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mysql-secondary-pv-1
spec:
  capacity:
    storage: 8Gi
  accessModes:
    - ReadWriteOnce
  storageClassName: standard
  hostPath:
    path: /Users/sgworker/Desktop/ansible_exercises/ansible-
exercises/mysql-secondary-1
```

```
% kubectl apply -f pv.yaml
persistentvolume/mysql-secondary-pv-0 created
persistentvolume/mysql-secondary-pv-1 created
sgworker@MacBook-Pro-3.local /Users/sgworker/Desktop/prometheus-exercises
[main]
% ansible-playbook 1-configure-k8s.yaml

PLAY [Deploy manifests in k8s cluster]
*********************************************************************************
*************************************************************************

TASK [Gathering Facts]
*********************************************************************************
*********************************************************************************
**************
ok: [localhost]

TASK [Deploy nginx ingress controller]
*********************************************************************************
*************************************************************************
changed: [localhost]

TASK [Deploy Mysql chart with 2 replicas]
*********************************************************************************
*************************************************************************
changed: [localhost]

TASK [Deploy Java application manifests]
*********************************************************************************
*************************************************************************
changed: [localhost] => (item=/Users/sgworker/Desktop/prometheus-
exercises/kubernetes-manifests/java-db-config.yaml)
changed: [localhost] => (item=/Users/sgworker/Desktop/prometheus-
exercises/kubernetes-manifests/java-app-ingress.yaml)
```

```
changed: [localhost] => (item=/Users/sgworker/Desktop/prometheus-
exercises/kubernetes-manifests/java-db-secret.yaml)
changed: [localhost] => (item=/Users/sgworker/Desktop/prometheus-
exercises/kubernetes-manifests/java-app.yaml)

PLAY RECAP
*********************************************************************************
*********************************************************************************
*************************
localhost                  : ok=4    changed=3    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0

sgworker@MacBook-Pro-3.local /Users/sgworker/Desktop/prometheus-exercises
[main]
% kubectl get svc
NAME                              TYPE        CLUSTER-IP       EXTERNAL-
IP   PORT(S)             AGE
java-app-service                  ClusterIP   10.99.40.39      <none>
8080/TCP,8081/TCP   5m9s
kubernetes                        ClusterIP   10.96.0.1        <none>
443/TCP             2d22h
mysql-release-primary             ClusterIP   10.105.215.192   <none>
3306/TCP            5m15s
mysql-release-primary-headless    ClusterIP   None             <none>
3306/TCP            5m15s
mysql-release-secondary           ClusterIP   10.101.118.114   <none>
3306/TCP            5m15s
mysql-release-secondary-headless  ClusterIP   None             <none>
3306/TCP            5m15s
sgworker@MacBook-Pro-3.local /Users/sgworker/Desktop/prometheus-exercises
[main]
%
sgworker@MacBook-Pro-3.local /Users/sgworker/Desktop/prometheus-exercises
[main]
% kubectl get pods
NAME                                  READY   STATUS    RESTARTS       AGE
java-app-deployment-f5f747d79-lps7q   1/1     Running   2 (5m7s ago)
5m31s
java-app-deployment-f5f747d79-pkmcj   1/1     Running   2 (5m9s ago)
5m31s
java-app-deployment-f5f747d79-zzjnm   1/1     Running   2 (5m8s ago)
5m31s
mysql-release-primary-0               1/1     Running   0
5m37s
mysql-release-secondary-0             1/1     Running   0
5m37s
mysql-release-secondary-1             1/1     Running   0
4m55s
sgworker@MacBook-Pro-3.local /Users/sgworker/Desktop/prometheus-exercises
[main]
%
```

My running cluster on Docker Desktop:

Java Maven application with 3 replicas and Mysql database for my Java application with 2 replicas and nginx ingress controller is also deployed.



▶ Solution 2: Start Monitoring your Applications

> EXERCISE 2: Start Monitoring your Applications

> Note: as you've learned, we deploy separate exporter applications for different services to monitor third party applications. But, some cloud native applications may have the metrics scraping configuration inside and not require an addition exporter application. So check whether the chart of that application supports scraping configuration before deploying a separate exporter for it.

- Deploy Prometheus Operator in your cluster (You can use the following helm chart: https://github.com/prometheus-community/helm-charts/tree/main/charts/kube-prometheus-stack)
- Configure metrics scraping for Nginx Controller
- Configure metrics scraping for Mysql
- Configure metrics scraping for Java application (Note: Java application exposes metrics on port 8081, NOT on /metrics endpoint)
- Check in Prometheus UI, that all three application metrics are being collected

First, I added the Prometheus Helm repository.

```
sgworker@MacBook-Pro-3.local /Users/sgworker/Desktop/prometheus-exercises
[main]
% helm repo add prometheus-community https://prometheus-
community.github.io/helm-charts
"prometheus-community" has been added to your repositories
sgworker@MacBook-Pro-3.local /Users/sgworker/Desktop/prometheus-exercises
[main]
```

after that i executed helm upgrade command and created the namespace monitoring.

```
% helm repo update
Hang tight while we grab the latest from your chart repositories...
```

```
...Successfully got an update from the "ingress-nginx" chart repository
...Successfully got an update from the "my-chart-repo" chart repository
...Successfully got an update from the "my-chart-repo-2" chart repository
...Successfully got an update from the "myhelmrepo" chart repository
...Successfully got an update from the "prometheus-community" chart
repository
...Successfully got an update from the "bitnami" chart repository
Update Complete. ⎈Happy Helming!⎈
sgworker@MacBook-Pro-3.local /Users/sgworker/Desktop/prometheus-exercises
[main]
% kubectl create namespace monitoring
namespace/monitoring created
sgworker@MacBook-Pro-3.local /Users/sgworker/Desktop/prometheus-exercises
[main]
```

Unfortunately, the installation on Docker Desktop didn't work, and I had to adjust the Helm install command for Prometheus. I added this setting to my helm install command: --set prometheus-node-exporter.hostRootFsMount.enabled=false

```
Events:
  Type     Reason     Age                    From               Message
  ----     ------     ----                   ----               -------
  Normal   Scheduled  3m49s                  default-scheduler
Successfully assigned monitoring/monitoring-stack-prometheus-node-
exporter-69482 to docker-desktop
  Normal   Pulling    3m48s                  kubelet            Pulling
image "quay.io/prometheus/node-exporter:v1.9.1"
  Normal   Pulled     3m44s                  kubelet
Successfully pulled image "quay.io/prometheus/node-exporter:v1.9.1" in
4.255s (4.255s including waiting). Image size: 24978622 bytes.
  Normal   Created    59s (x6 over 3m44s)    kubelet            Created
container: node-exporter
  Warning  Failed     59s (x6 over 3m44s)    kubelet            Error:
failed to start container "node-exporter": Error response from daemon:
path / is mounted on / but it is not a shared or slave mount
  Normal   Pulled     59s (x5 over 3m43s)    kubelet            Container
image "quay.io/prometheus/node-exporter:v1.9.1" already present on machine
  Warning  BackOff    39s (x19 over 3m38s)   kubelet            Back-off
restarting failed container node-exporter in pod monitoring-stack-
prometheus-node-exporter-69482_monitoring(8c29c02a-9656-426f-b07a-
0194d7d62202)
```

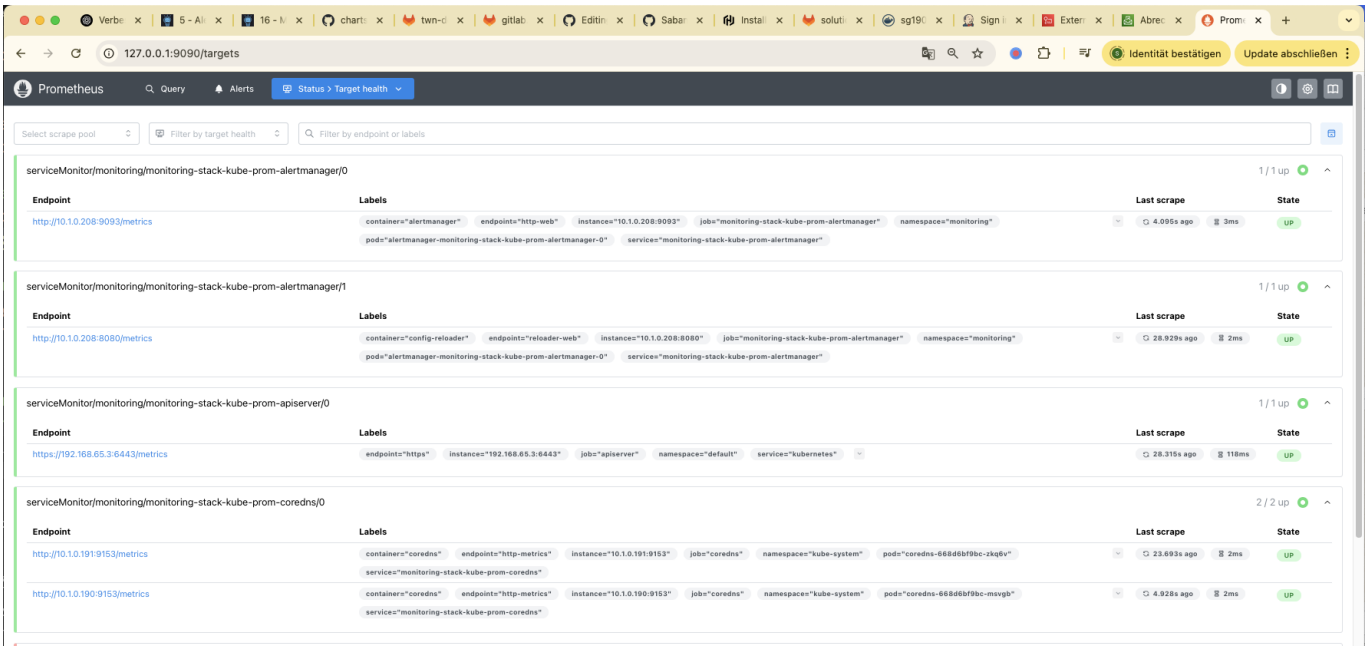
```
sgworker@MacBook-Pro-3.local /Users/sgworker/Desktop/prometheus-exercises
[main]
% helm upgrade --install monitoring-stack prometheus-community/kube-
prometheus-stack \
  --namespace monitoring \
  --create-namespace \
  --set prometheus-node-exporter.hostRootFsMount.enabled=false
```

```
Release "monitoring-stack" has been upgraded. Happy Helming!
NAME: monitoring-stack
LAST DEPLOYED: Mon Jun 30 16:13:00 2025
NAMESPACE: monitoring
STATUS: deployed
REVISION: 2
NOTES:
kube-prometheus-stack has been installed. Check its status by running:
  kubectl --namespace monitoring get pods -l "release=monitoring-stack"

Get Grafana 'admin' user password by running:

  kubectl --namespace monitoring get secrets monitoring-stack-grafana -o
jsonpath="{.data.admin-password}" | base64 -d ; echo

Access Grafana local instance:

  export POD_NAME=$(kubectl --namespace monitoring get pod -l
"app.kubernetes.io/name=grafana,app.kubernetes.io/instance=monitoring-
stack" -oname)
  kubectl --namespace monitoring port-forward $POD_NAME 3000

Visit https://github.com/prometheus-operator/kube-prometheus for
instructions on how to create & configure Alertmanager and Prometheus
instances using the Operator.
```

```
sgworker@MacBook-Pro-3.local /Users/sgworker/Desktop/prometheus-exercises
[main]
% kubectl --namespace monitoring get pods -l "release=monitoring-stack"
NAME                                                     READY   STATUS
RESTARTS   AGE
monitoring-stack-kube-prom-operator-745cdb7785-n6sph   1/1     Running   0
5m47s
monitoring-stack-kube-state-metrics-6546b9fcb4-cg996   1/1     Running   0
5m47s
monitoring-stack-prometheus-node-exporter-2mlgp        1/1     Running   0
24s
```

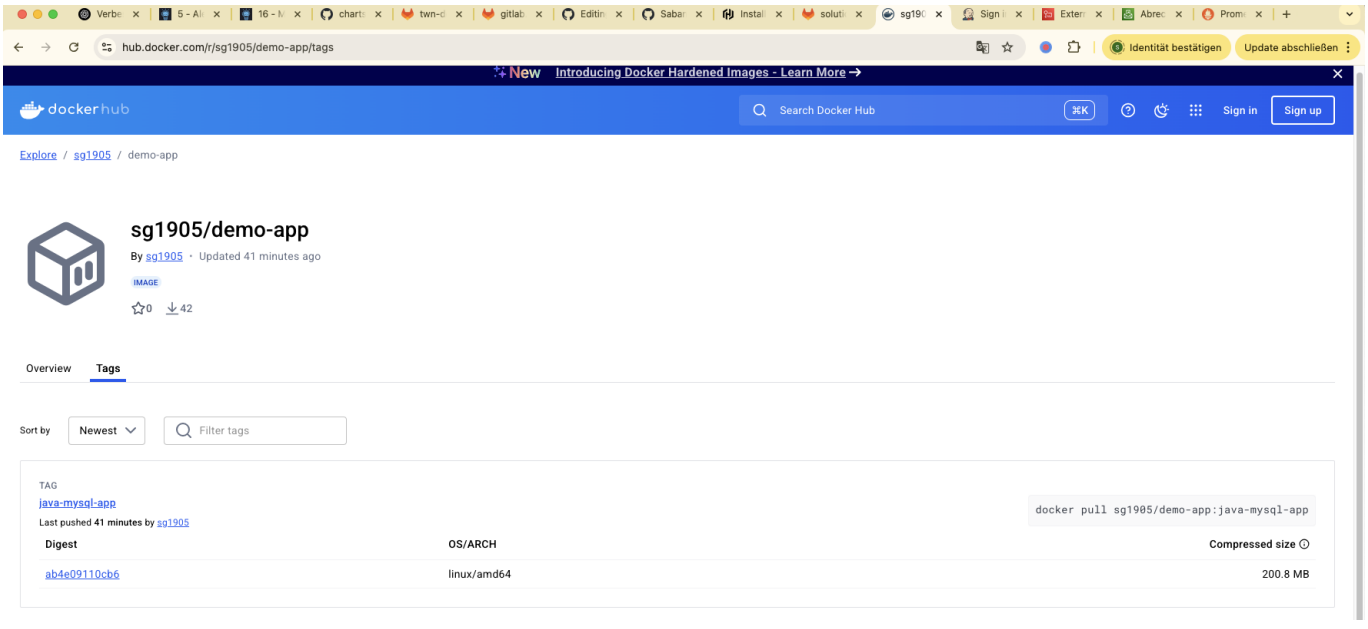
```
sgworker@MacBook-Pro-3.local /Users/sgworker/Desktop/prometheus-exercises
[main]
% kubectl -n monitoring port-forward svc/monitoring-stack-kube-prom-
prometheus 9090:9090
Forwarding from 127.0.0.1:9090 -> 9090
Forwarding from [::1]:9090 -> 9090
```

I was successfully connected to the Prometheus UI :




```
sgworker@MacBook-Pro-3.local /Users/sgworker/Desktop/prometheus-exercises
[main]
% kubectl get prometheuses.monitoring.coreos.com
No resources found in default namespace.
sgworker@MacBook-Pro-3.local /Users/sgworker/Desktop/prometheus-exercises
[main]
% kubectl get prometheuses.monitoring.coreos.com -n monitoring
NAME                                    VERSION   DESIRED   READY
RECONCILED   AVAILABLE   AGE
monitoring-stack-kube-prom-prometheus   v3.4.2    1         1       True
True         17m
```

I built the java-mysql-app again with prometheus stuff and pushed into my docker repository:

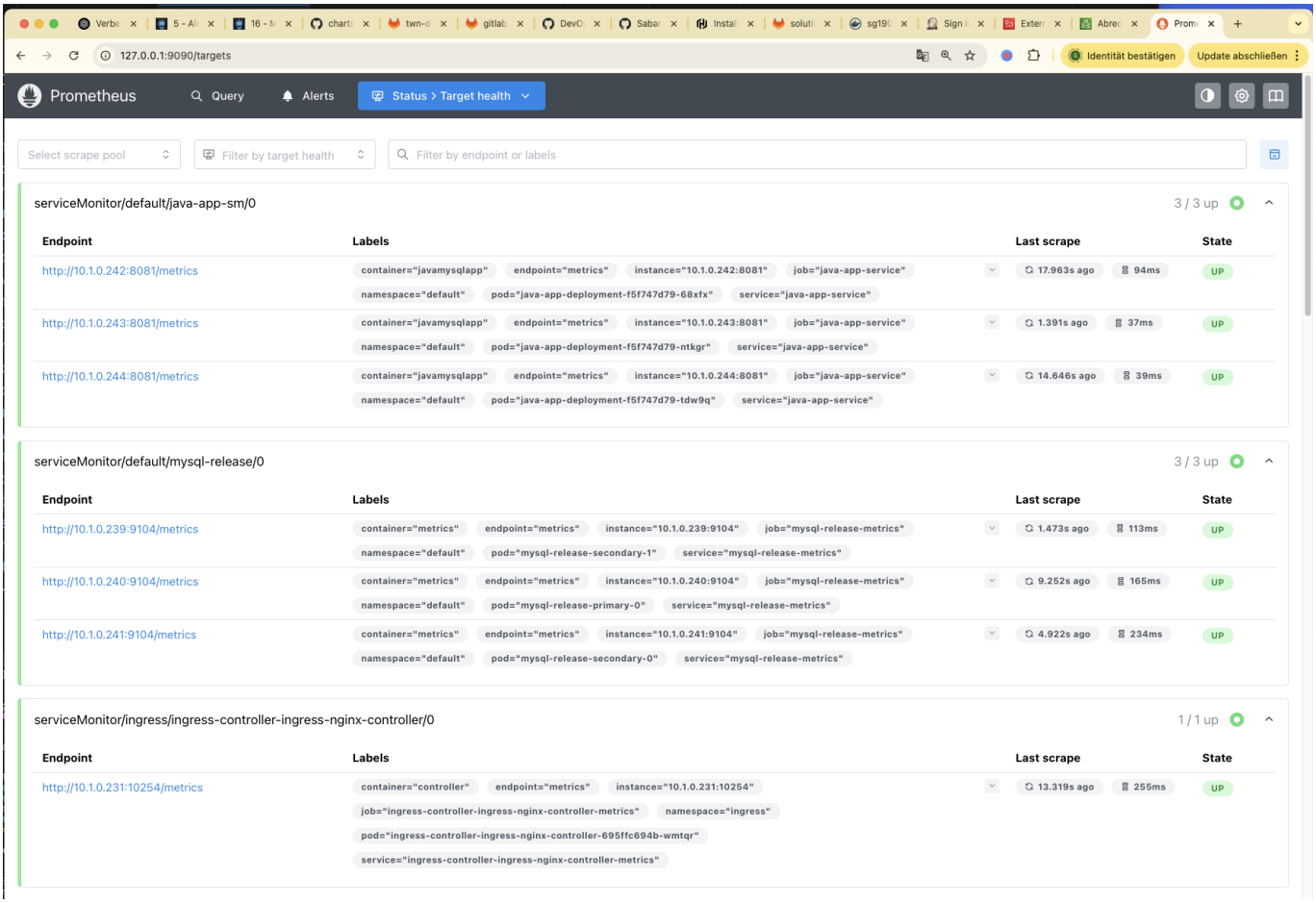


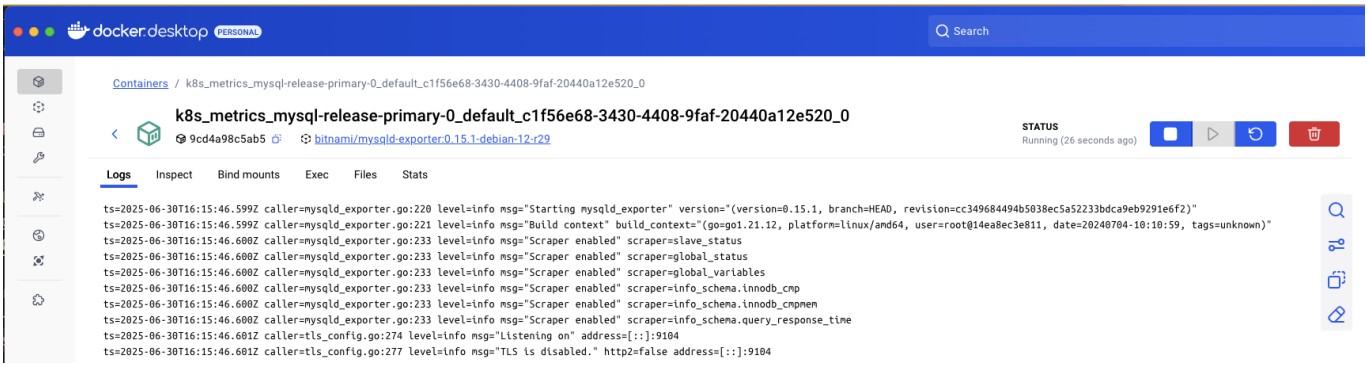
I uninstalled mysql-release and ingress-controller. After that executed the ansible-playbook 2-configure-k8s.yaml

```
sgworker@MacBook-Pro-3.local /Users/sgworker/Desktop/prometheus-exercises
[main]
% helm uninstall mysql-release
helm uninstall ingress-controller -n ingress
release "mysql-release" uninstalled
release "ingress-controller" uninstalled
sgworker@MacBook-Pro-3.local /Users/sgworker/Desktop/prometheus-exercises
[main]
% ansible-playbook 2-configure-k8s.yaml

PLAY [Deploy manifests in k8s cluster]
*********************************************************************************
*********************************************************************************
*************************************************************

TASK [Gathering Facts]
*********************************************************************************
*********************************************************************************
*********************************************************************
ok: [localhost]

TASK [Deploy nginx ingress controller]
*********************************************************************************
*********************************************************************************
*************************************************************
changed: [localhost]

TASK [Deploy Mysql chart with 2 replicas]
*********************************************************************************
*********************************************************************************
*************************************************************
changed: [localhost]

TASK [Deploy Java application manifests]
*********************************************************************************
*********************************************************************************
*************************************************************
ok: [localhost] => (item=/Users/sgworker/Desktop/prometheus-
exercises/kubernetes-manifests/java-db-config.yaml)
ok: [localhost] => (item=/Users/sgworker/Desktop/prometheus-
exercises/kubernetes-manifests/java-service-monitor.yaml)
ok: [localhost] => (item=/Users/sgworker/Desktop/prometheus-
exercises/kubernetes-manifests/java-app-ingress.yaml)
ok: [localhost] => (item=/Users/sgworker/Desktop/prometheus-
exercises/kubernetes-manifests/java-db-secret.yaml)
ok: [localhost] => (item=/Users/sgworker/Desktop/prometheus-
exercises/kubernetes-manifests/java-app.yaml)

PLAY RECAP
*********************************************************************************
*********************************************************************************
*********************************************************************************
```

```
**********
localhost                  : ok=4    changed=2    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0

sgworker@MacBook-Pro-3.local /Users/sgworker/Desktop/prometheus-exercises
[main]
% kubectl get svc java-app-service -n default

NAME               TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
AGE
java-app-service   ClusterIP   10.99.127.212   <none>
8080/TCP,8081/TCP   38m

sgworker@MacBook-Pro-3.local /Users/sgworker/Desktop/prometheus-exercises
[main]
% kubectl -n monitoring port-forward svc/prometheus-operated 9090

Forwarding from 127.0.0.1:9090 -> 9090
Forwarding from [::1]:9090 -> 9090
Handling connection for 9090
Handling connection for 9090
```

The monitoing was enabled successfully for my java-app, mysql and ingress controller:





▶ Solution 3: Configure Alert Rules

> EXERCISE 3: Configure Alert Rules

Now it's time to configure alerts for critical issues that may happen with any of the applications.

- Configure an alert rule for nginx-ingress: More than 5% of HTTP requests have status 4xx
- Configure alert rules for Mysql: All Mysql instances are down & Mysql has too many connections
- Configure alert rule for the Java application: Too many requests
- Configure alert rule for a K8s component: StatefulSet replicas mismatch (Since Mysql is deployed as a StatefulSet, if one of the replicas goes down, we want to be notified)

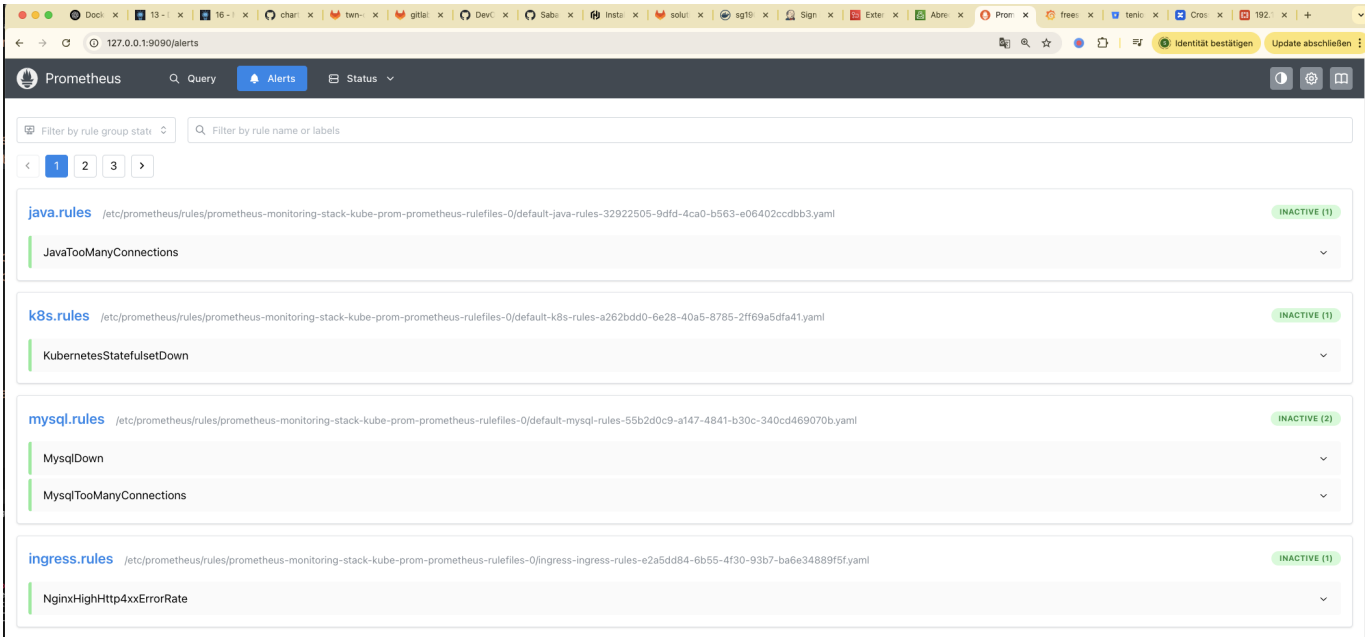
```
kubectl get prometheuses.monitoring.coreos.com -A

NAMESPACE    NAME                                    VERSION   DESIRED
READY   RECONCILED   AVAILABLE   AGE
monitoring   monitoring-stack-kube-prom-prometheus   v3.4.2    1
1       True         True        24h

kubectl -n monitoring get prometheuses.monitoring.coreos.com monitoring-
stack-kube-prom-prometheus -o yaml | grep ruleSelector -A 5

  ruleSelector:
    matchLabels:
      release: monitoring-stack
  scrapeConfigNamespaceSelector: {}
  scrapeConfigSelector:
    matchLabels:
```

```
!!!!! ❌ Problem
There is one closing parenthesis ) too many.
The rate() function is correctly closed, but there's an additional closing
) afterward that was never opened.
I fixed it.
```

```
sgworker@MacBook-Pro-3.local /Users/sgworker/Desktop/prometheus-exercises
[main]
% kubectl apply -f kubernetes-manifests/3-nginx-alert-rules.yaml
kubectl apply -f kubernetes-manifests/3-mysql-alert-rules.yaml
kubectl apply -f kubernetes-manifests/3-java-alert-rules.yaml
kubectl apply -f kubernetes-manifests/3-k8s-alert-rules.yaml
prometheusrule.monitoring.coreos.com/ingress-rules unchanged
prometheusrule.monitoring.coreos.com/mysql-rules unchanged
prometheusrule.monitoring.coreos.com/java-rules created
prometheusrule.monitoring.coreos.com/k8s-rules unchanged
```

I was then able to successfully deploy the alert rules.

```
kubectl -n monitoring port-forward svc/prometheus-operated 9090
```



▶ Solution 4: Send Alert Notifications

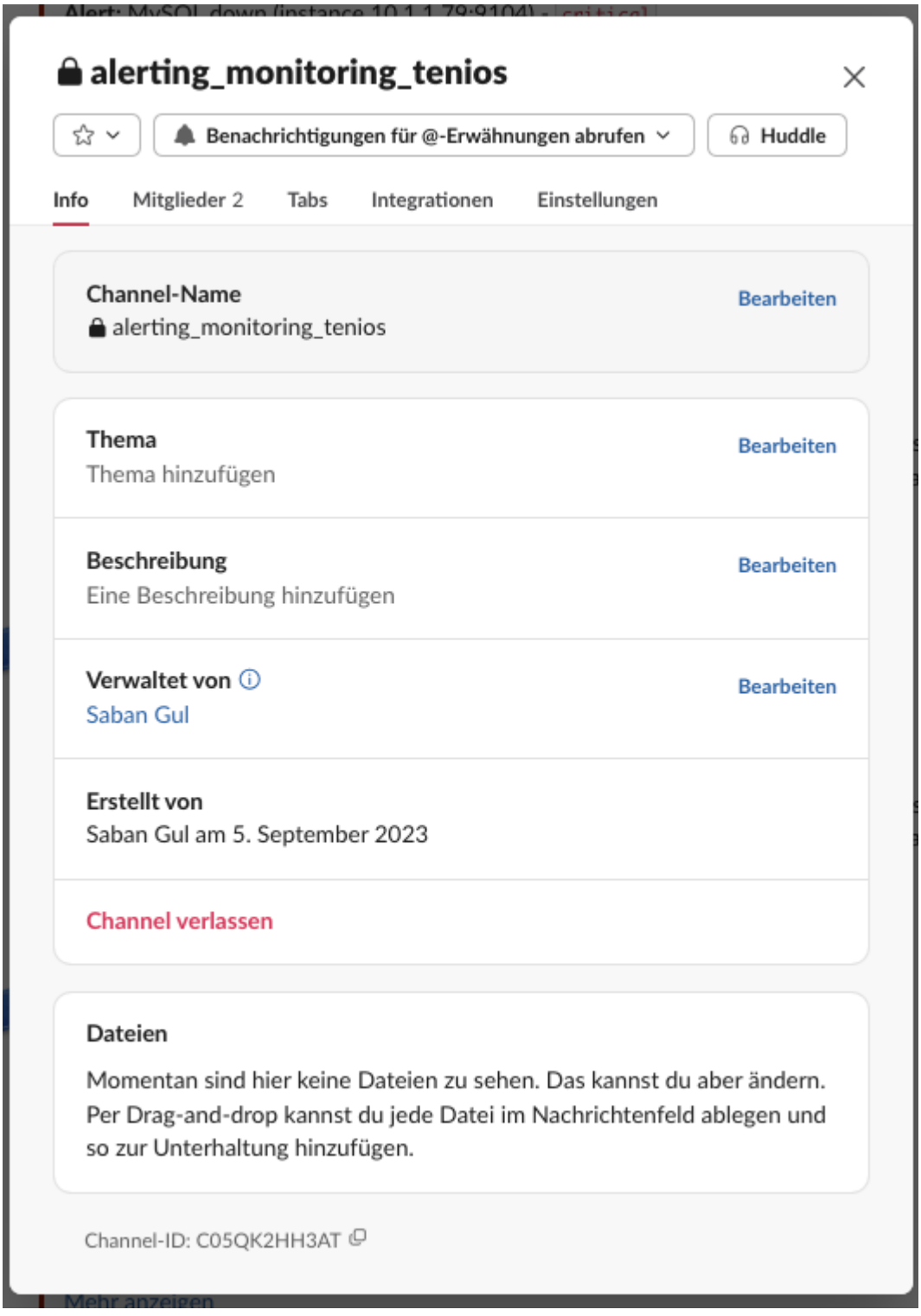
> EXERCISE 4: Send Alert Notifications

Great job! You have added observability to your cluster, and you have configured your monitoring with all the important alerts. Now when issues happen in the cluster, you want to automatically notify people who are responsible for fixing the issue or at least observing the issue, so it doesn't break the cluster.

- Configure alert manager to send all issues related to Java or Mysql application to the developer team's Slack channel. (Hint: You can use the following guide to set up a Slack channel for the notifications: https://www.freecodecamp.org/news/what-are-github-actions-and-how-can-you-automate-tests-and-slack-notifications/#part-2-post-new-pull-requests-to-slack)
- Configure alert manager to send all issues related Nginx Ingress Controller or K8s components to K8s administrator's email address.

> Note: Of course, in your case, this can be your own email address or your own Slack channel.

I have configured the following Slack channel with the ID: XXX and the Gmail address.




```
apiVersion: monitoring.coreos.com/v1alpha1
kind: AlertmanagerConfig
metadata:
  name: main-rules-alert-config
  namespace: monitoring
  labels:
    alertmanagerConfig: main
spec:
```

```
    route:
      receiver: 'null'          # Default-Receiver "null" (keine Aktion)
      groupWait: 30s
      groupInterval: 5m
      repeatInterval: 4h
      continue: true            # Weiter in den Subroutes suchen
      routes:
      - receiver: 'email'
        matchers:
        - name: alertname
          value: NginxHighHttp4xxErrorRate
        continue: false         # Bei Match stoppen
      - receiver: 'slack'
        matchers:
        - name: alertname
          value: MysqlDown
        continue: false
      - receiver: 'slack'
        matchers:
        - name: alertname
          value: MysqlTooManyConnections
        continue: false
      - receiver: 'slack'
        matchers:
        - name: alertname
          value: JavaTooManyConnections
        continue: false
      - receiver: 'email'
        matchers:
        - name: alertname
          value: KubernetesStatefulsetDown
        - name: namespace
          value: monitoring
        continue: false
      - receiver: 'email'
        matchers:
        - name: alertname
          value: AlwaysFire
        - name: namespace
          value: monitoring
        continue: false

    receivers:
    - name: 'null'              # Null-Receiver: verwirft Alerts
    - name: 'email'
      emailConfigs:
      - to: 'sg1905w1@gmail.com'
        from: 'sg1905w1@gmail.com'
        smarthost: 'smtp.gmail.com:587'
        authUsername: 'sg1905w1@gmail.com'
        authIdentity: 'sg1905w1@gmail.com'
        authPassword:
          name: gmail-auth
          key: password
```

```
    - name: 'slack'
      slackConfigs:
      - channel: 'C05QK2HH3AT'
        sendResolved: false
        apiURL:
          name: slack-auth
          key: slack_url
        title: '[{{ .Status | toUpper }}{{ if eq .Status "firing" }}:{{
.Alerts.Firing | len }}{{ end }}] Monitoring Event Notification'
        text: >-
          {{ range .Alerts }}
            *Alert:* {{ .Annotations.summary }} - `{{ .Labels.severity }}`
            *Description:* {{ .Annotations.description }}
            *Graph:* <{{ .GeneratorURL }}|:chart_with_upwards_trend:>
*Runbook:* <{{ .Annotations.runbook }}|:spiral_note_pad:>
            *Details:*
            {{ range .Labels.SortedPairs }} • *{{ .Name }}:* `{{ .Value }}`
            {{ end }}
          {{ end }}
```

Lastly, I executed the following steps:

```
kubectl apply -f kubernetes-manifests/4-email-secret.yaml
kubectl apply -f kubernetes-manifests/4-slack-secret.yaml
kubectl apply -f kubernetes-manifests/4-alert-manager-configuration.yaml
```

▶ Solution 5: Test the Alerts

> EXERCISE 5: Test the Alerts

- Of course, you want to check now that your whole setup works, so try to simulate issues and trigger 1 alert for each notification channel (Slack and E-mail).
- For this, you can simply, kubectl delete one of the stateful set pods, or Mysql pods or try accessing your java applications on a /path-that-doesnt-exist etc.

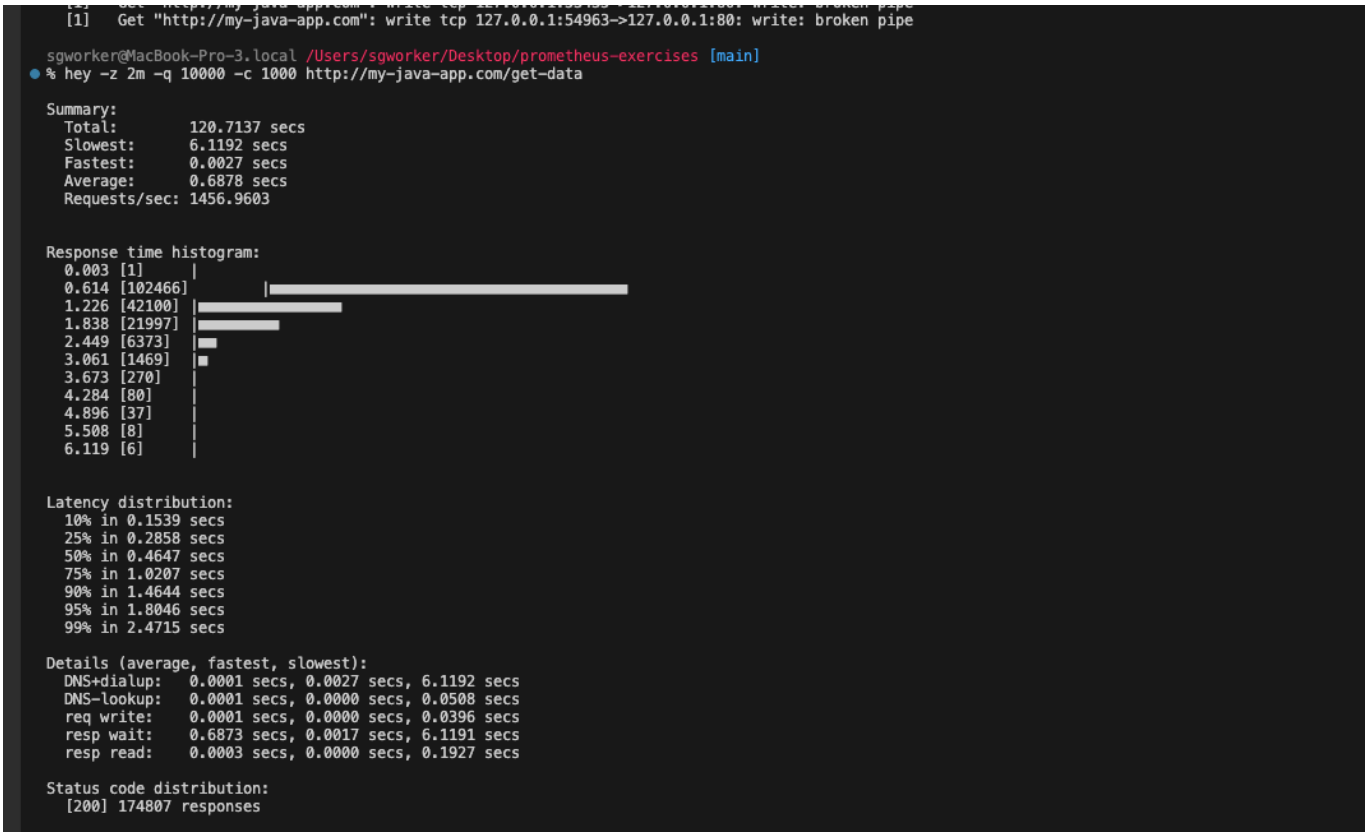
FYI: !!! Unfortunately, no matter what I did, I did not receive the alert emails or Slack notifications until I added the label namespace: monitoring to the PrometheusRule configurations. Before that, the alerts were routed to the null receiver and thus discarded.

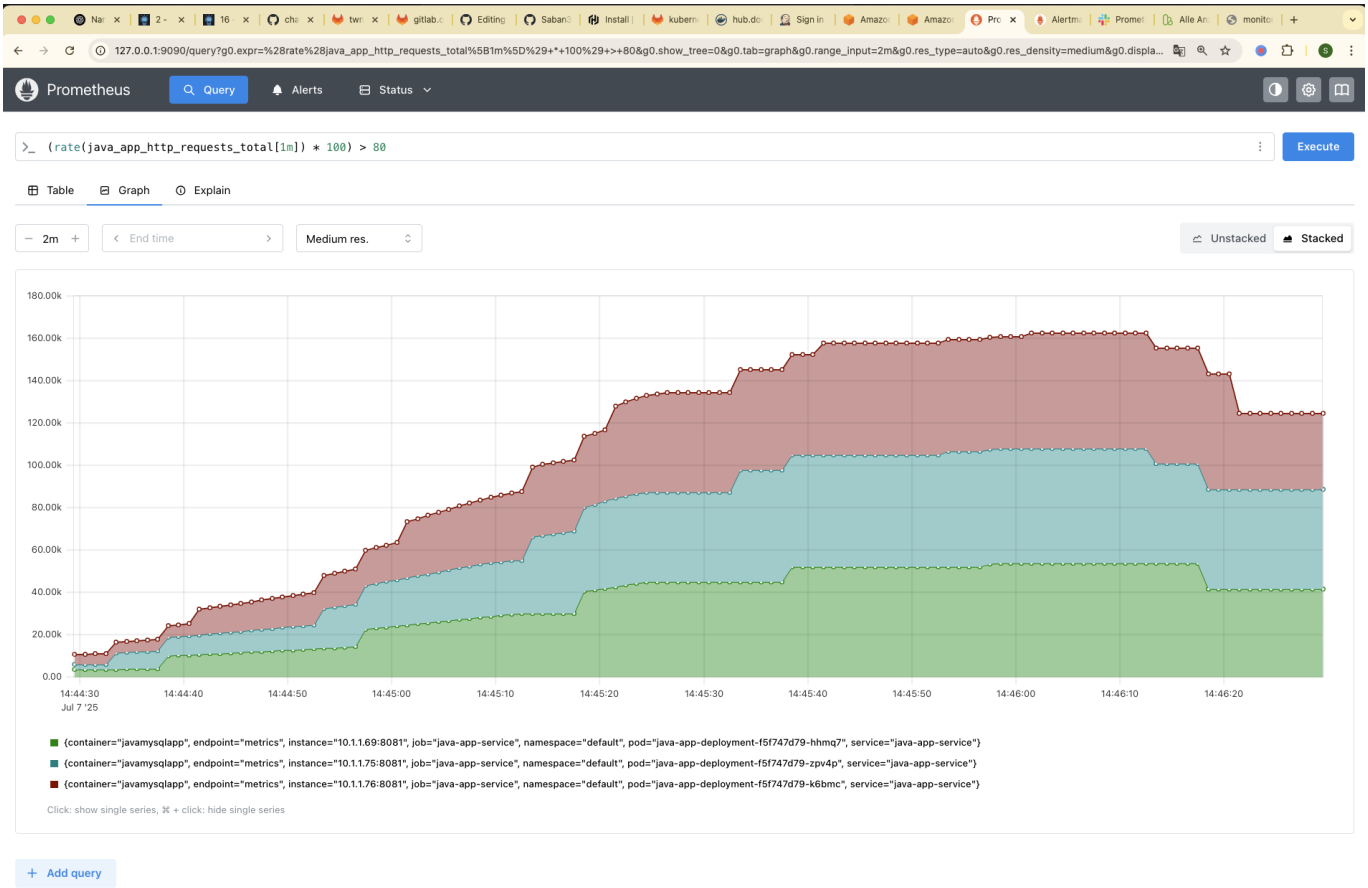
To trigger the 'Java application too many connections' alert, I used the following command:

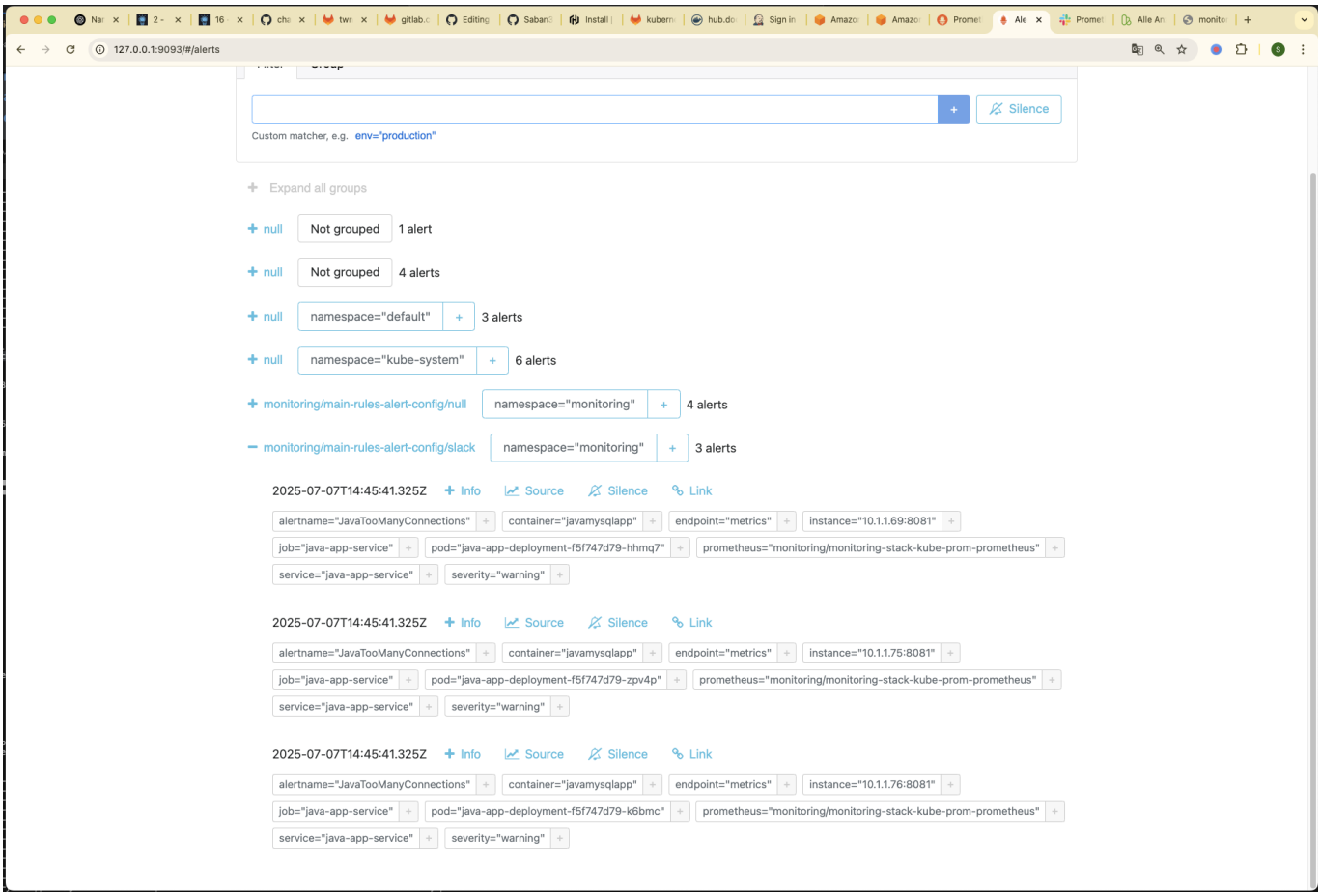
```
hey -z 2m -q 10000 -c 1000 http://my-java-app.com/get-data
```

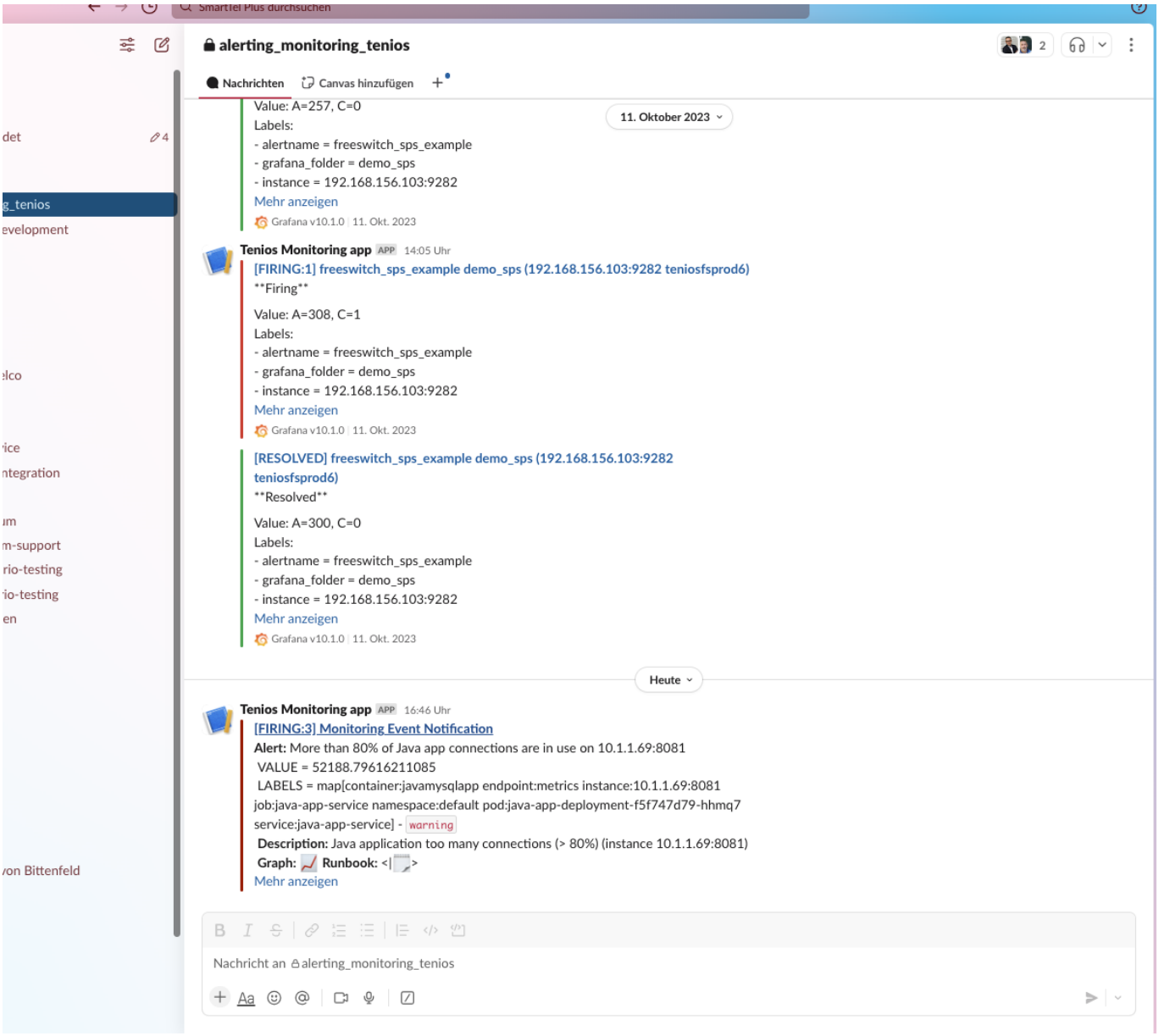
and then checked:

```
kubectl -n monitoring port-forward svc/prometheus-operated 9090
kubectl -n monitoring port-forward svc/monitoring-stack-kube-prom-
alertmanager 9093:9093
```



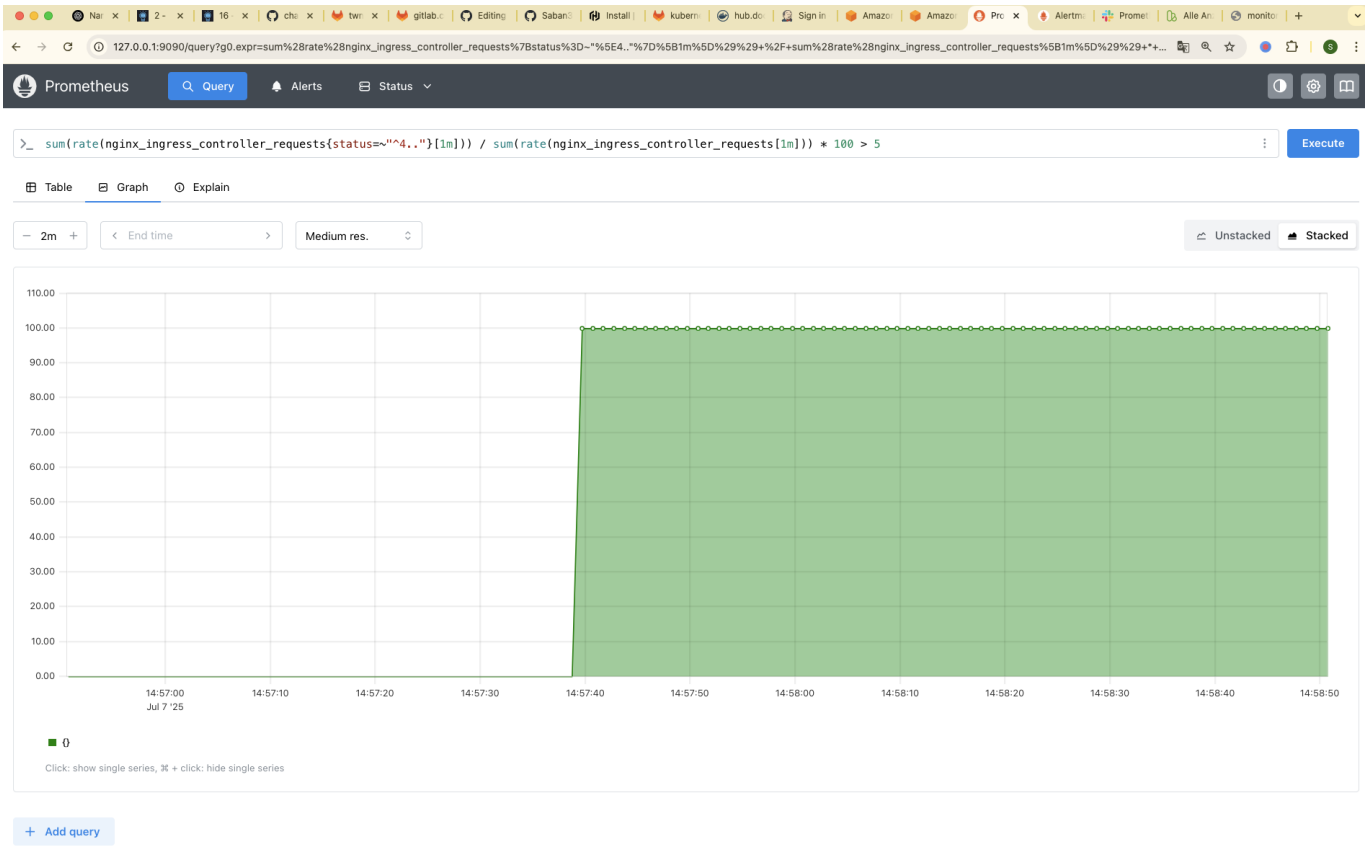


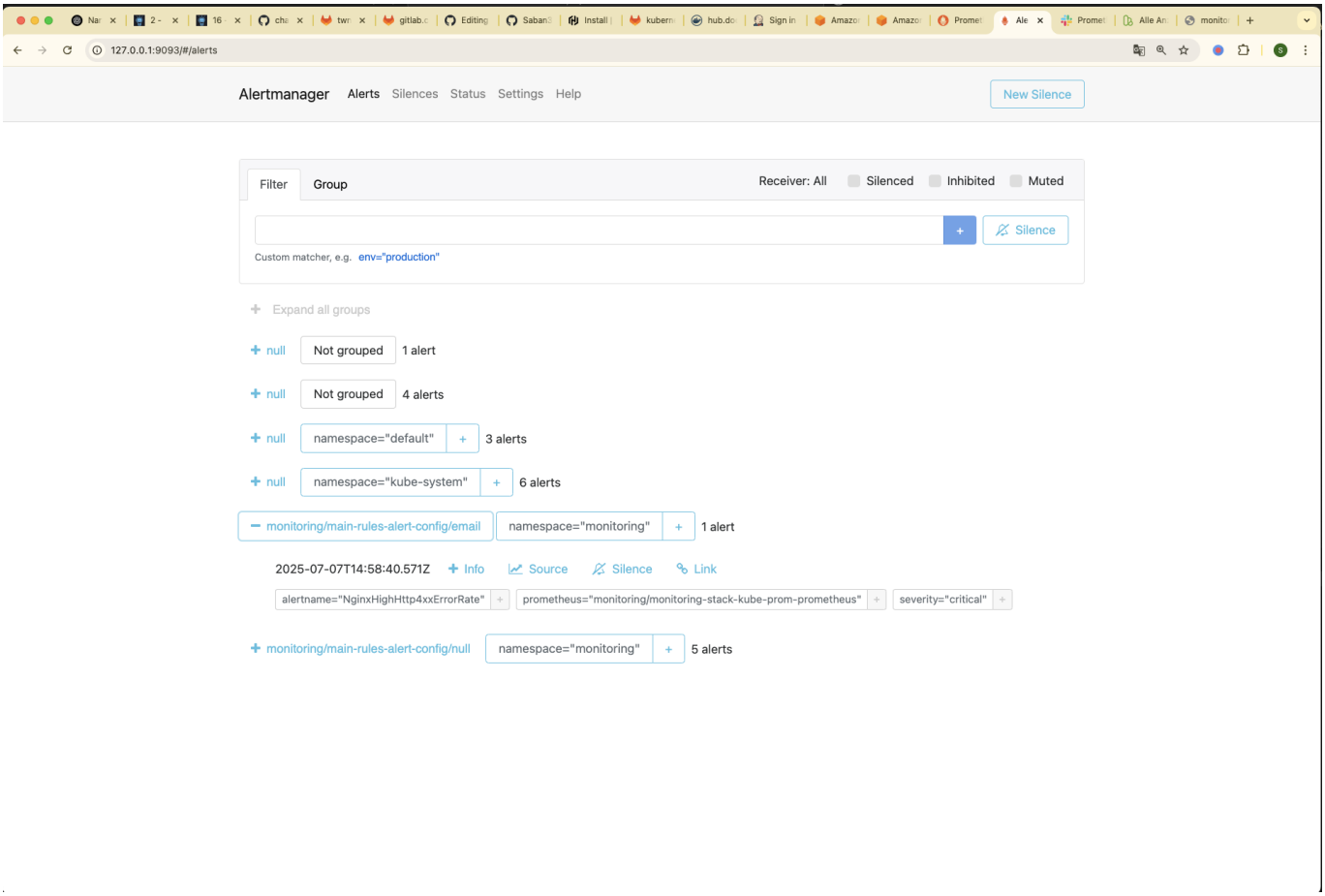

127.0.0.1:9093/#/alerts
Silence
Custom matcher, e.g. env="production"
Expand all groups
null Not grouped 1 alert
null Not grouped 4 alerts
null namespace="default" 3 alerts
null namespace="kube-system" 6 alerts
monitoring/main-rules-alert-config/null namespace="monitoring" 4 alerts
monitoring/main-rules-alert-config/slack namespace="monitoring" 3 alerts
2025-07-07T14:45:41.325Z Info Source Silence Link
alertname="JavaTooManyConnections" container="javamysqlapp" endpoint="metrics" instance="10.1.1.69:8081"
job="java-app-service" pod="java-app-deployment-f5f747d79-hhmq7" prometheus="monitoring/monitoring-stack-kube-prom-prometheus"
service="java-app-service" severity="warning"
2025-07-07T14:45:41.325Z Info Source Silence Link
alertname="JavaTooManyConnections" container="javamysqlapp" endpoint="metrics" instance="10.1.1.75:8081"
job="java-app-service" pod="java-app-deployment-f5f747d79-zpv4p" prometheus="monitoring/monitoring-stack-kube-prom-prometheus"
service="java-app-service" severity="warning"
2025-07-07T14:45:41.325Z Info Source Silence Link
alertname="JavaTooManyConnections" container="javamysqlapp" endpoint="metrics" instance="10.1.1.76:8081"
job="java-app-service" pod="java-app-deployment-f5f747d79-k6bmc" prometheus="monitoring/monitoring-stack-kube-prom-prometheus"
service="java-app-service" severity="warning"

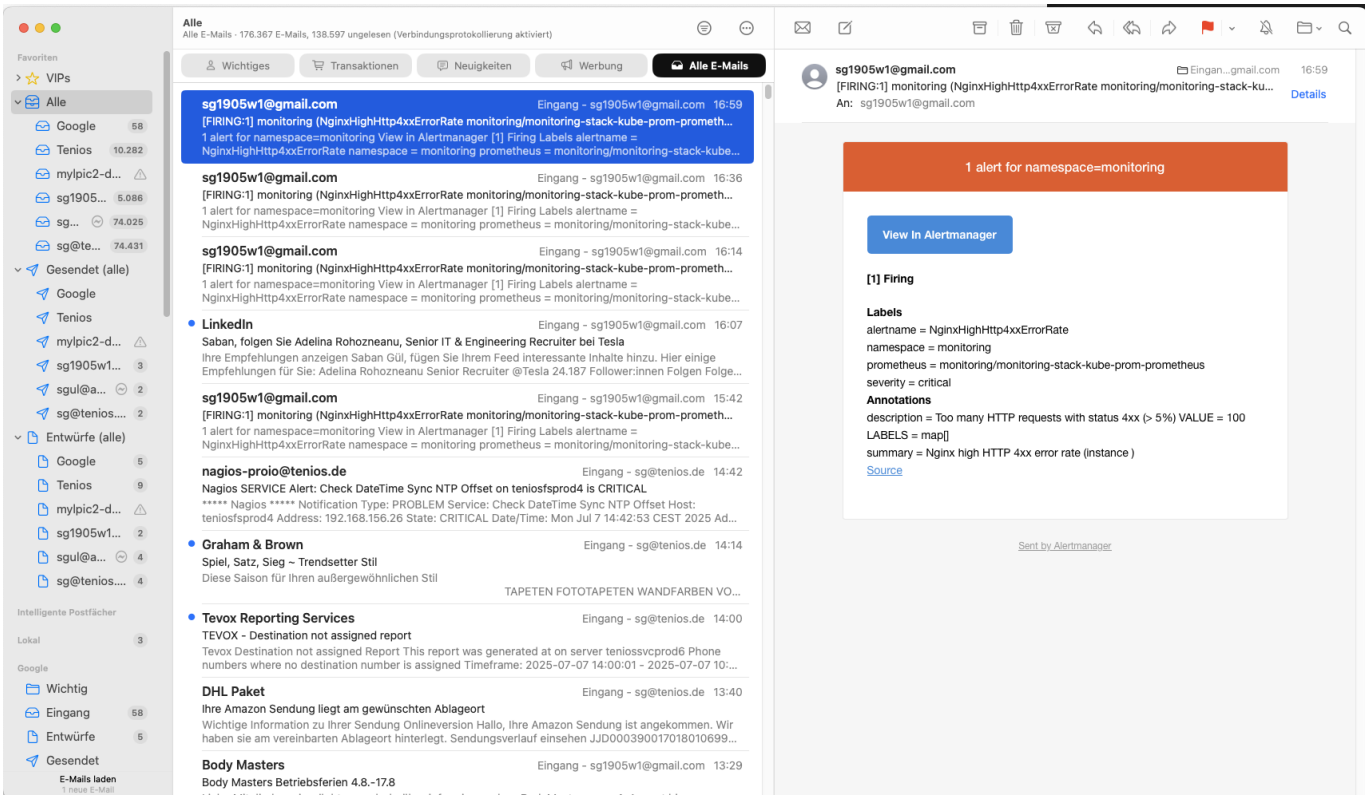


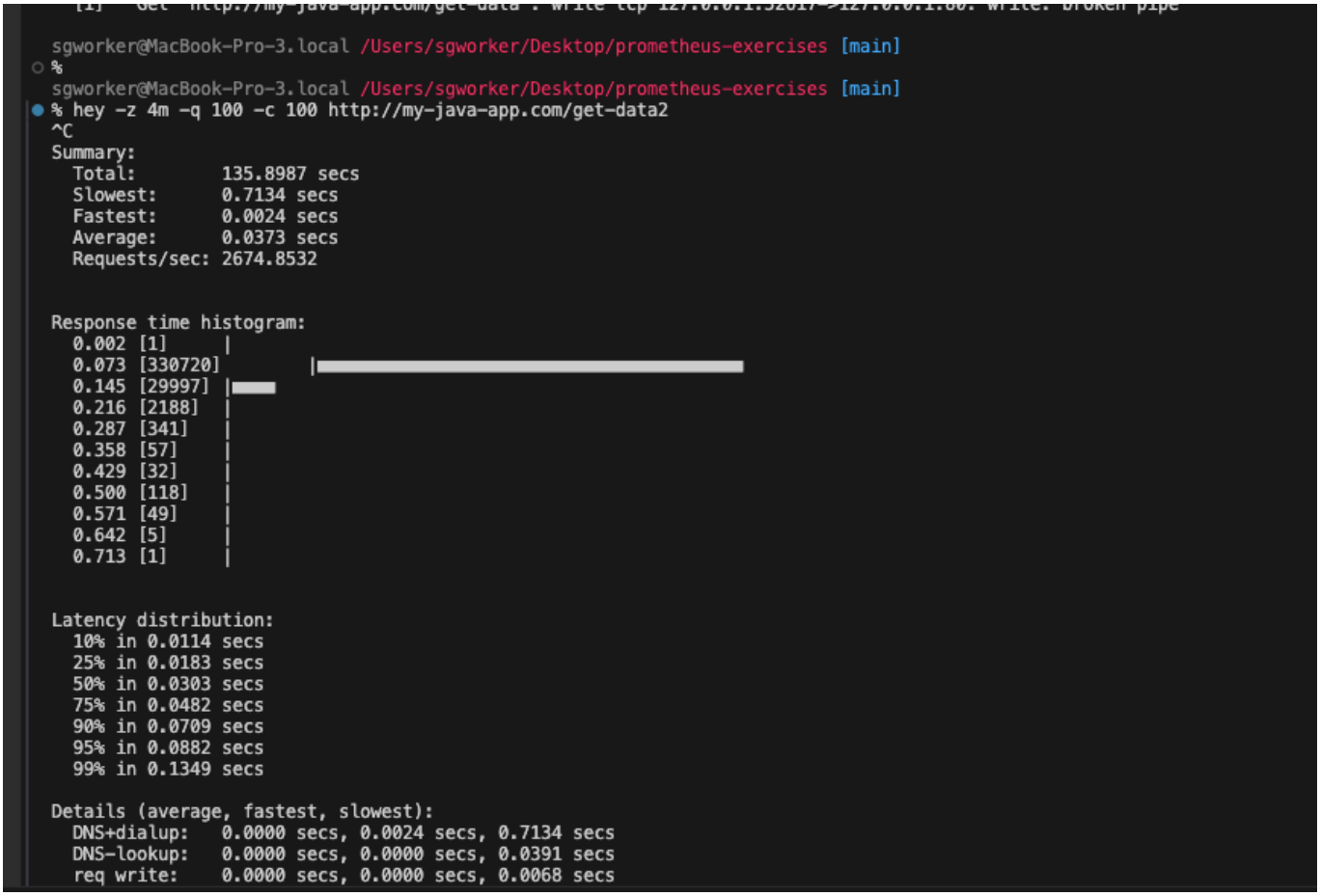
To trigger the 'NginxHighHttp4xxErrorRate' alert, I used the following command to call a non-existent URL

```
hey -z 4m -q 100 -c 100 http://my-java-app.com/get-data2
```


127.0.0.1:9090/query?g0.expr=sum%28rate%28nginx_ingress_controller_requests%7Bstatus%3D~"%5E4.."%7D%5B1m%5D%29%29+%2F+sum%28rate%28nginx_ingress_controller_requests%5B1m%5D%29%29+*+...
Prometheus
Query
Alerts
Status
sum(rate(nginx_ingress_controller_requests{status=~"^4.."}[1m])) / sum(rate(nginx_ingress_controller_requests[1m])) * 100 > 5
Execute
Table
Graph
Explain
2m
End time
Medium res.
Unstacked
Stacked
110.00
100.00
90.00
80.00
70.00
60.00
50.00
40.00
30.00
20.00
10.00
0.00
14:57:00
Jul 7 '25
14:57:10
14:57:20
14:57:30
14:57:40
14:57:50
14:58:00
14:58:10
14:58:20
14:58:30
14:58:40
14:58:50
{}
Click: show single series, ⌘ + click: hide single series
Add query


127.0.0.1:9093/#/alerts
Alertmanager
Alerts
Silences
Status
Settings
Help
New Silence
Filter
Group
Receiver: All
Silenced
Inhibited
Muted
Silence
Custom matcher, e.g. env="production"
Expand all groups
null
Not grouped
1 alert
null
Not grouped
4 alerts
null
namespace="default"
3 alerts
null
namespace="kube-system"
6 alerts
monitoring/main-rules-alert-config/email
namespace="monitoring"
1 alert
2025-07-07T14:58:40.571Z
Info
Source
Silence
Link
alertname="NginxHighHttp4xxErrorRate"
prometheus="monitoring/monitoring-stack-kube-prom-prometheus"
severity="critical"
monitoring/main-rules-alert-config/null
namespace="monitoring"
5 alerts





To trigger the MysqlTooManyConnections alert, I first verified the maximum number of allowed MySQL connections with the following command:

```
kubectl exec -n default -it mysql-release-primary-0 -c mysql -- \
  mysql -u root -psecret-root-pass -e "SHOW VARIABLES LIKE
```

```
'max_connections';"
```

This returned:

```
mysql: [Warning] Using a password on the command line interface can be
insecure.
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| max_connections | 151   |
+-----------------+-------+
```

Then, I applied a Kubernetes Job that simulates many parallel MySQL connections to overload the server and trigger the alert:

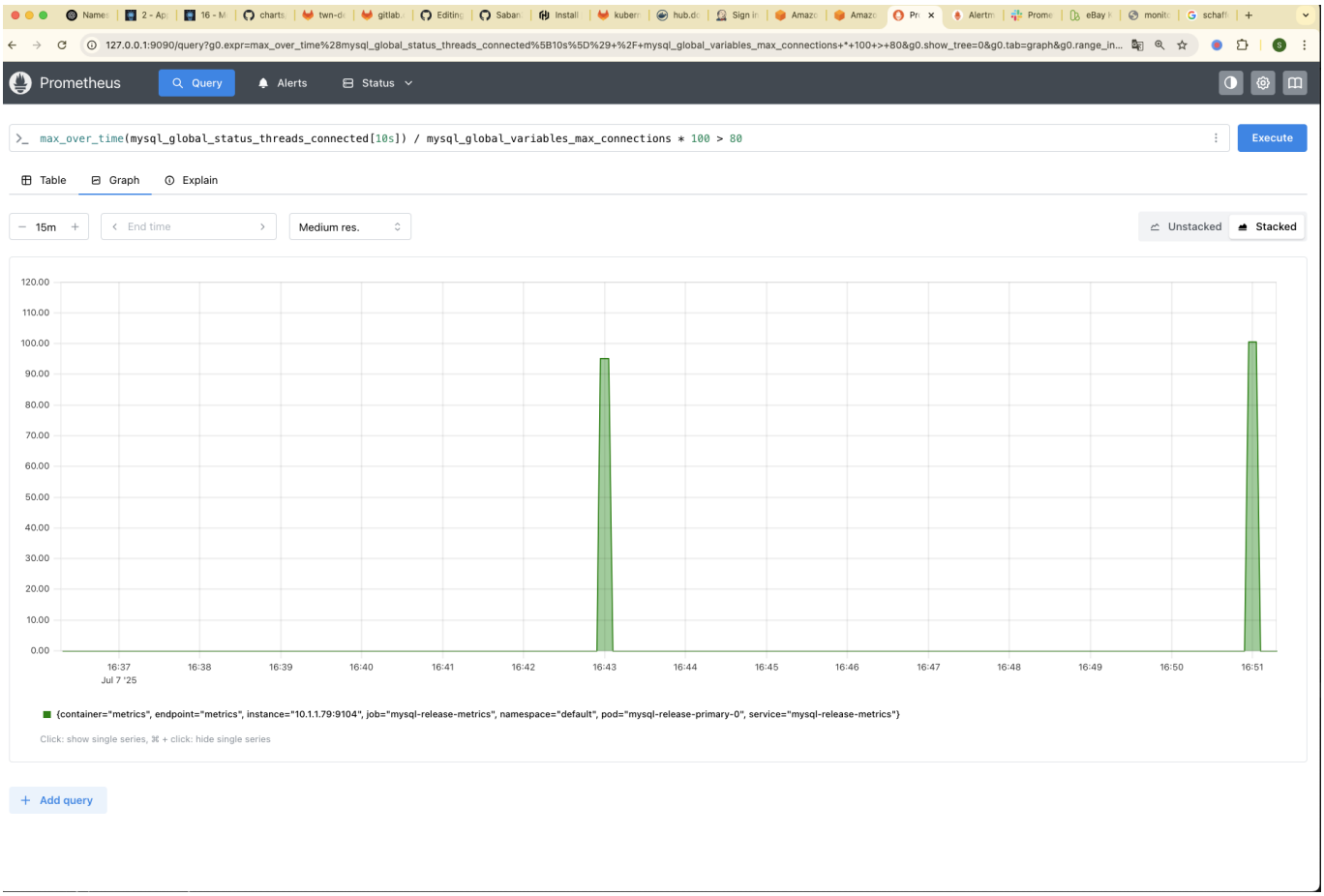
```
apiVersion: batch/v1
kind: Job
metadata:
  name: mysql-connection-stress
  namespace: default
spec:
  completions: 10
  parallelism: 10
  template:
    spec:
      containers:
      - name: mysql-client
        image: mysql:8
        command: ["sh", "-c"]
        args:
        - |
          for i in $(seq 1 100); do
            mysql -h mysql-release-primary.default.svc.cluster.local \
                  -u root -pXXXXXXXX \
                  -e "SELECT SLEEP(300);" &
          done
          wait
        resources:
          limits:
            memory: "64Mi"
            cpu: "50m"
          requests:
            memory: "32Mi"
            cpu: "25m"
      restartPolicy: Never
```

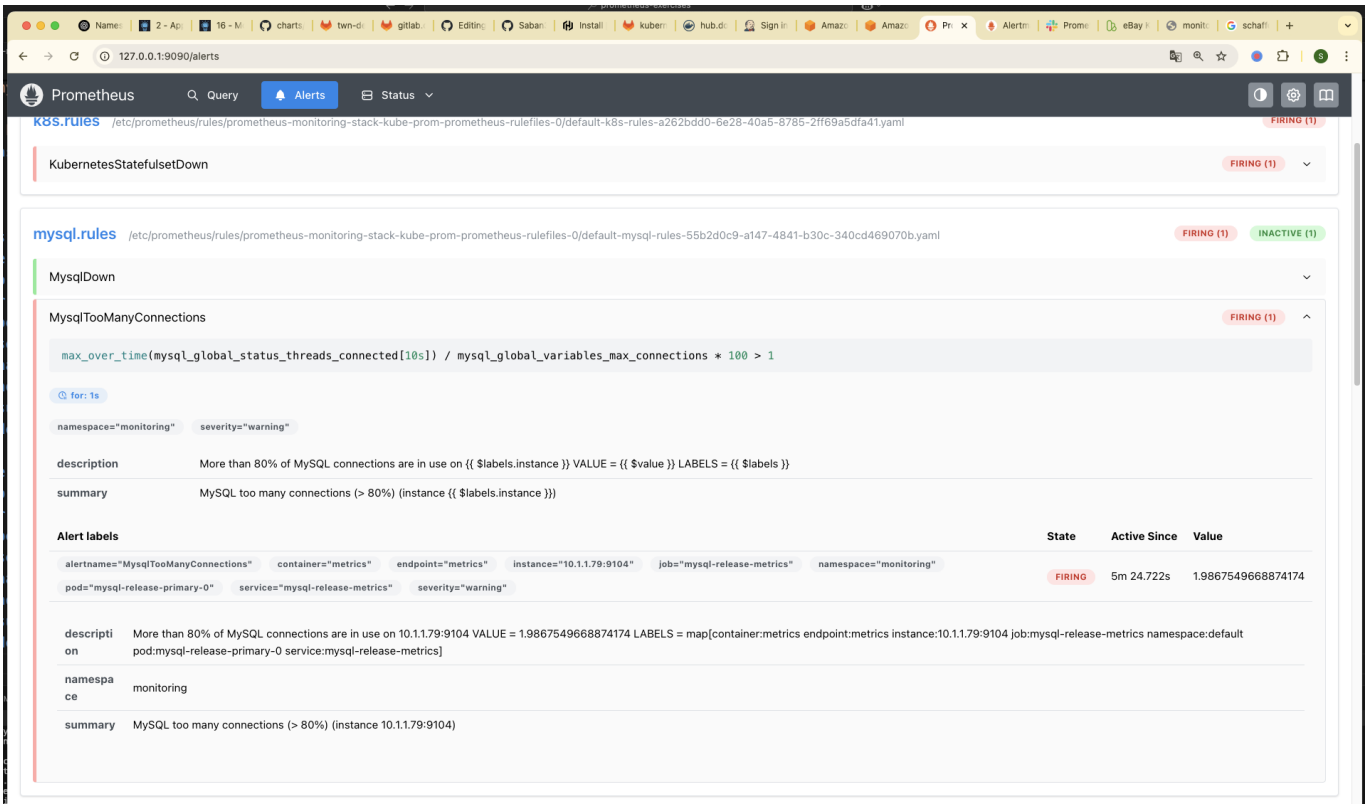
Finally, I created the job using:

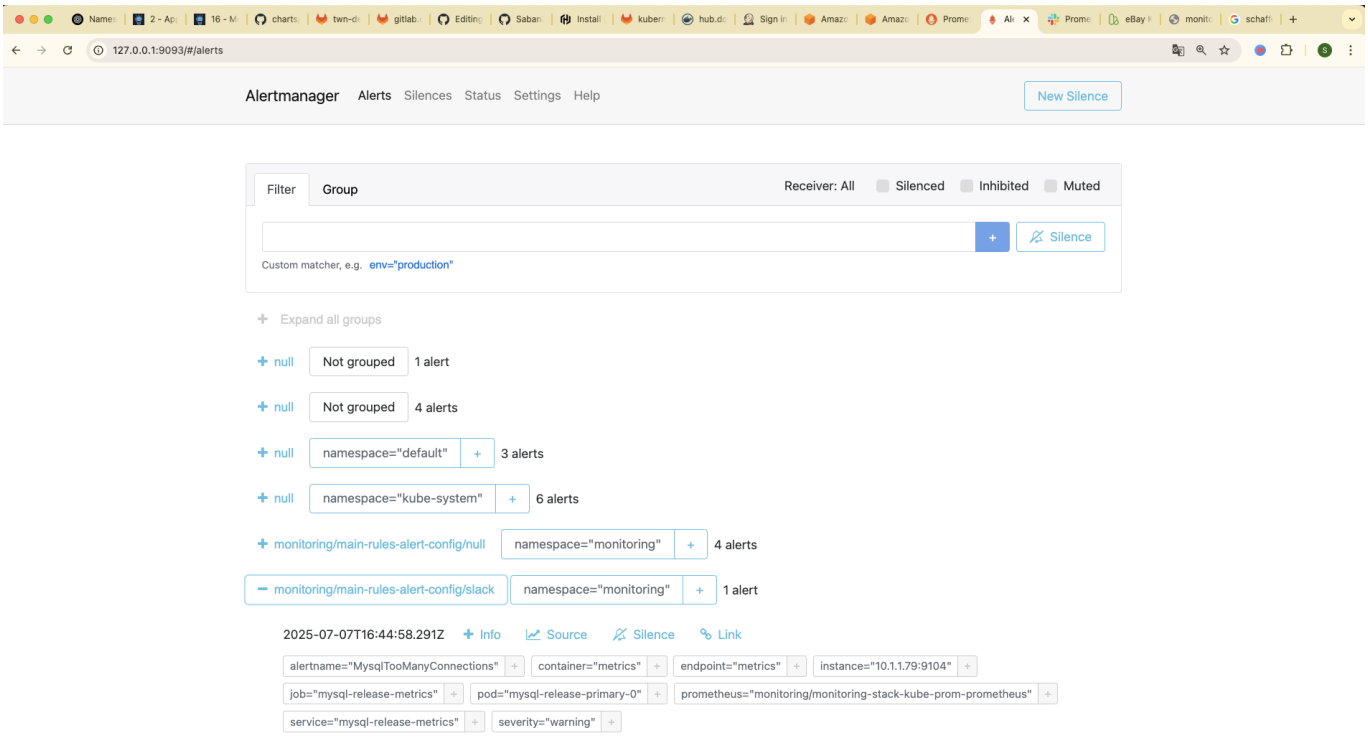
```
kubectl apply -f kubernetes-manifests/mysql-connection-stress.yaml
```

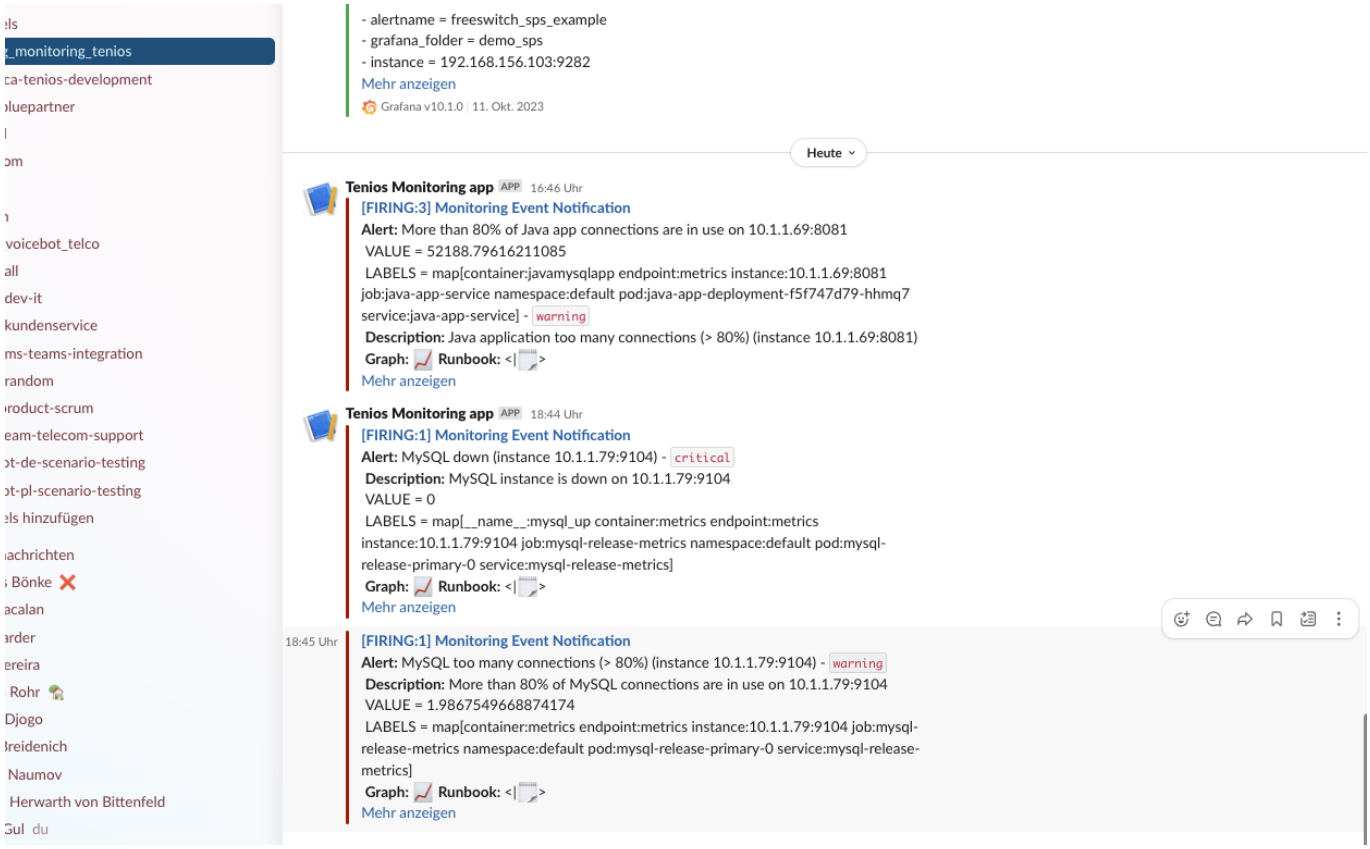
Returned:

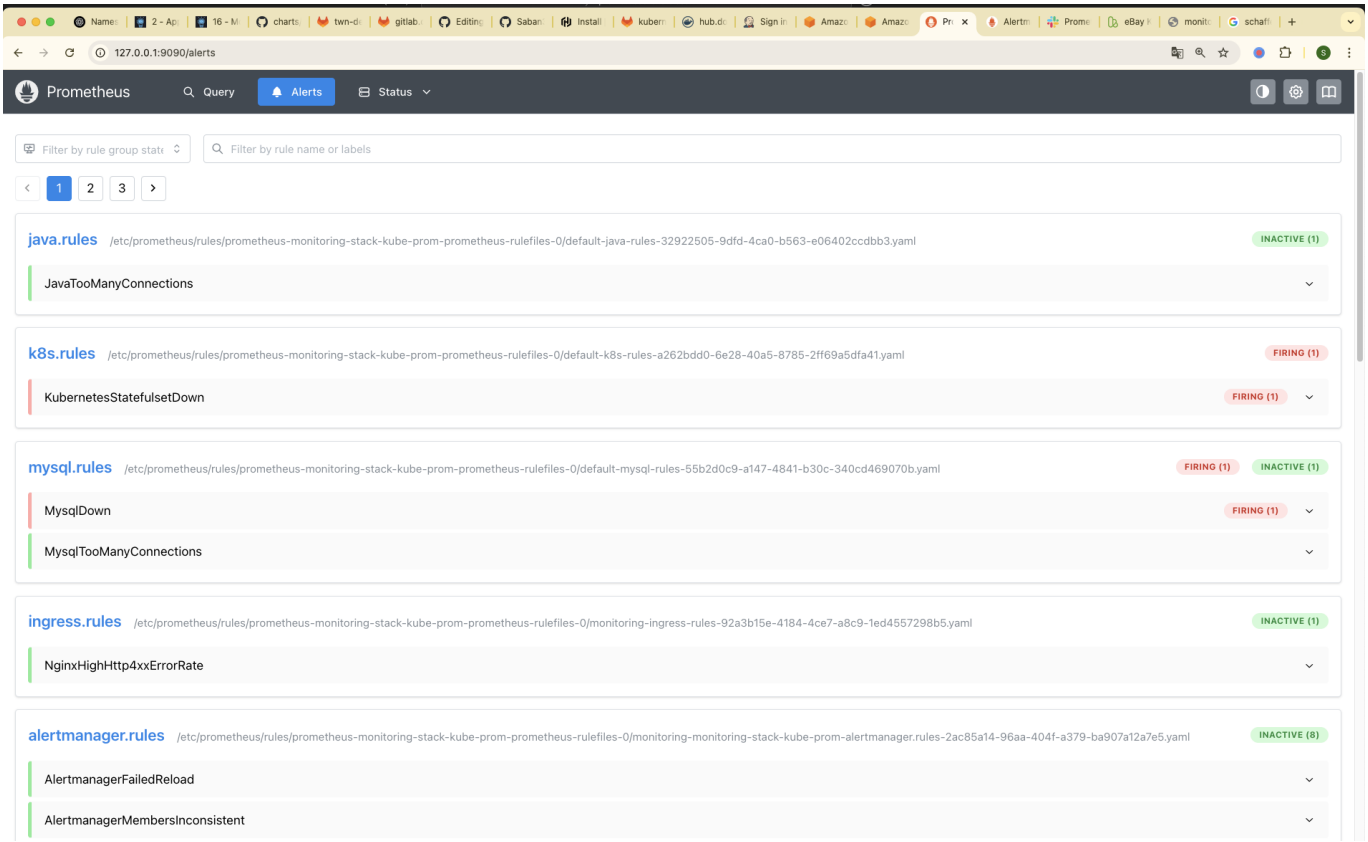
```
job.batch/mysql-connection-stress created
```




127.0.0.1:9090/alerts
Prometheus
Query
Alerts
Status
k8s.rules
/etc/prometheus/rules/prometheus-monitoring-stack-kube-prom-prometheus-rulefiles-0/default-k8s-rules-a262bdd0-6e28-40a5-8785-2ff69a5dfa41.yaml
FIRING (1)
KubernetesStatefulsetDown
FIRING (1)
mysql.rules
/etc/prometheus/rules/prometheus-monitoring-stack-kube-prom-prometheus-rulefiles-0/default-mysql-rules-55b2d0c9-a147-4841-b30c-340cd469070b.yaml
FIRING (1)
INACTIVE (1)
MysqlDown
MysqlTooManyConnections
FIRING (1)
max_over_time(mysql_global_status_threads_connected[10s]) / mysql_global_variables_max_connections * 100 > 1
for: 1s
namespace="monitoring"
severity="warning"
description
More than 80% of MySQL connections are in use on {{ $labels.instance }} VALUE = {{ $value }} LABELS = {{ $labels }}
summary
MySQL too many connections (> 80%) (instance {{ $labels.instance }})
Alert labels
State
Active Since
Value
alertname="MysqlTooManyConnections"
container="metrics"
endpoint="metrics"
instance="10.1.1.79:9104"
job="mysql-release-metrics"
namespace="monitoring"
pod="mysql-release-primary-0"
service="mysql-release-metrics"
severity="warning"
FIRING
5m 24.722s
1.9867549668874174
description
More than 80% of MySQL connections are in use on 10.1.1.79:9104 VALUE = 1.9867549668874174 LABELS = map[container:metrics endpoint:metrics instance:10.1.1.79:9104 job:mysql-release-metrics namespace:default pod:mysql-release-primary-0 service:mysql-release-metrics]
namespace
monitoring
summary
MySQL too many connections (> 80%) (instance 10.1.1.79:9104)


127.0.0.1:9093/#/alerts
Alertmanager
Alerts
Silences
Status
Settings
Help
New Silence
Filter
Group
Receiver: All
Silenced
Inhibited
Muted
Silence
Custom matcher, e.g. env="production"
Expand all groups
null
Not grouped
1 alert
null
Not grouped
4 alerts
null
namespace="default"
3 alerts
null
namespace="kube-system"
6 alerts
monitoring/main-rules-alert-config/null
namespace="monitoring"
4 alerts
monitoring/main-rules-alert-config/slack
namespace="monitoring"
1 alert
2025-07-07T16:44:58.291Z
Info
Source
Silence
Link
alertname="MysqlTooManyConnections"
container="metrics"
endpoint="metrics"
instance="10.1.1.79:9104"
job="mysql-release-metrics"
pod="mysql-release-primary-0"
prometheus="monitoring/monitoring-stack-kube-prom-prometheus"
service="mysql-release-metrics"
severity="warning"



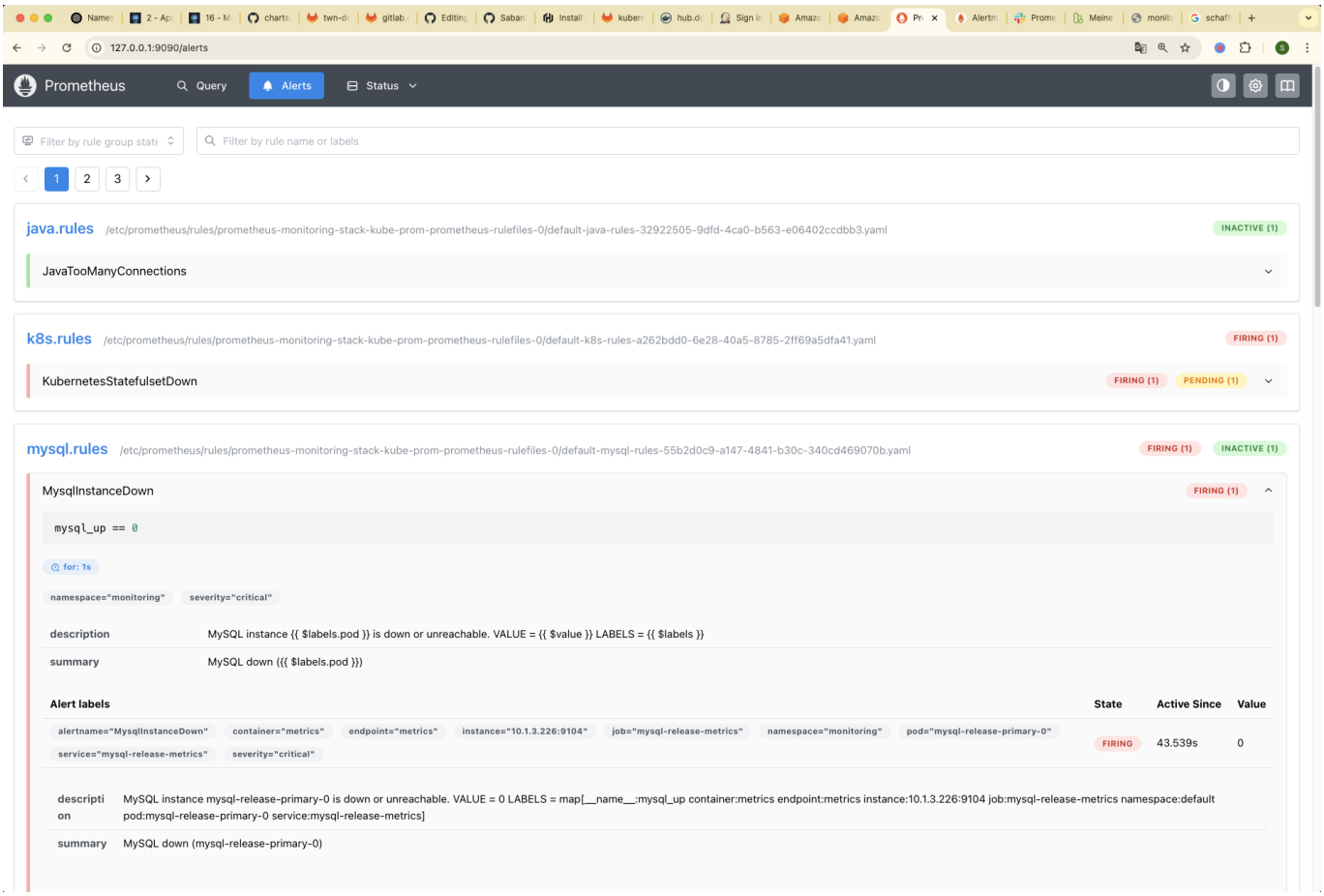


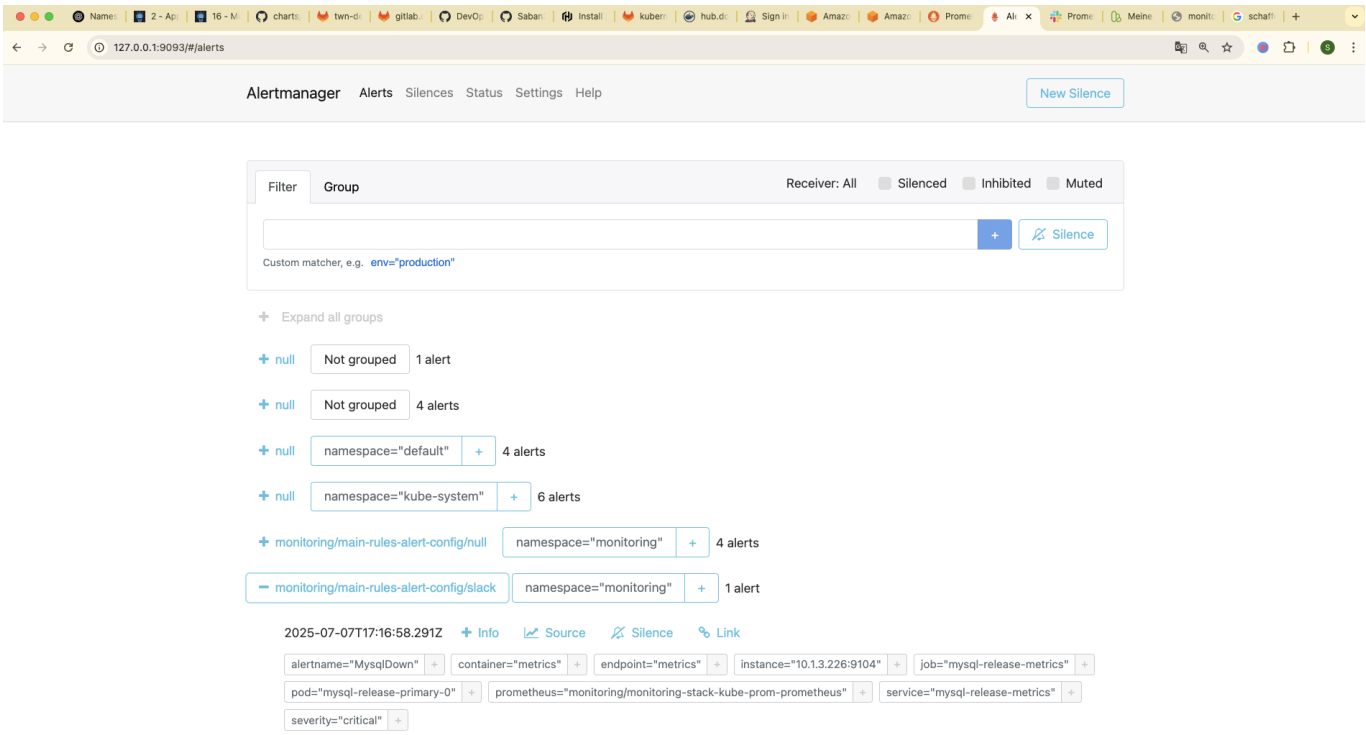
after testing I deleted it:

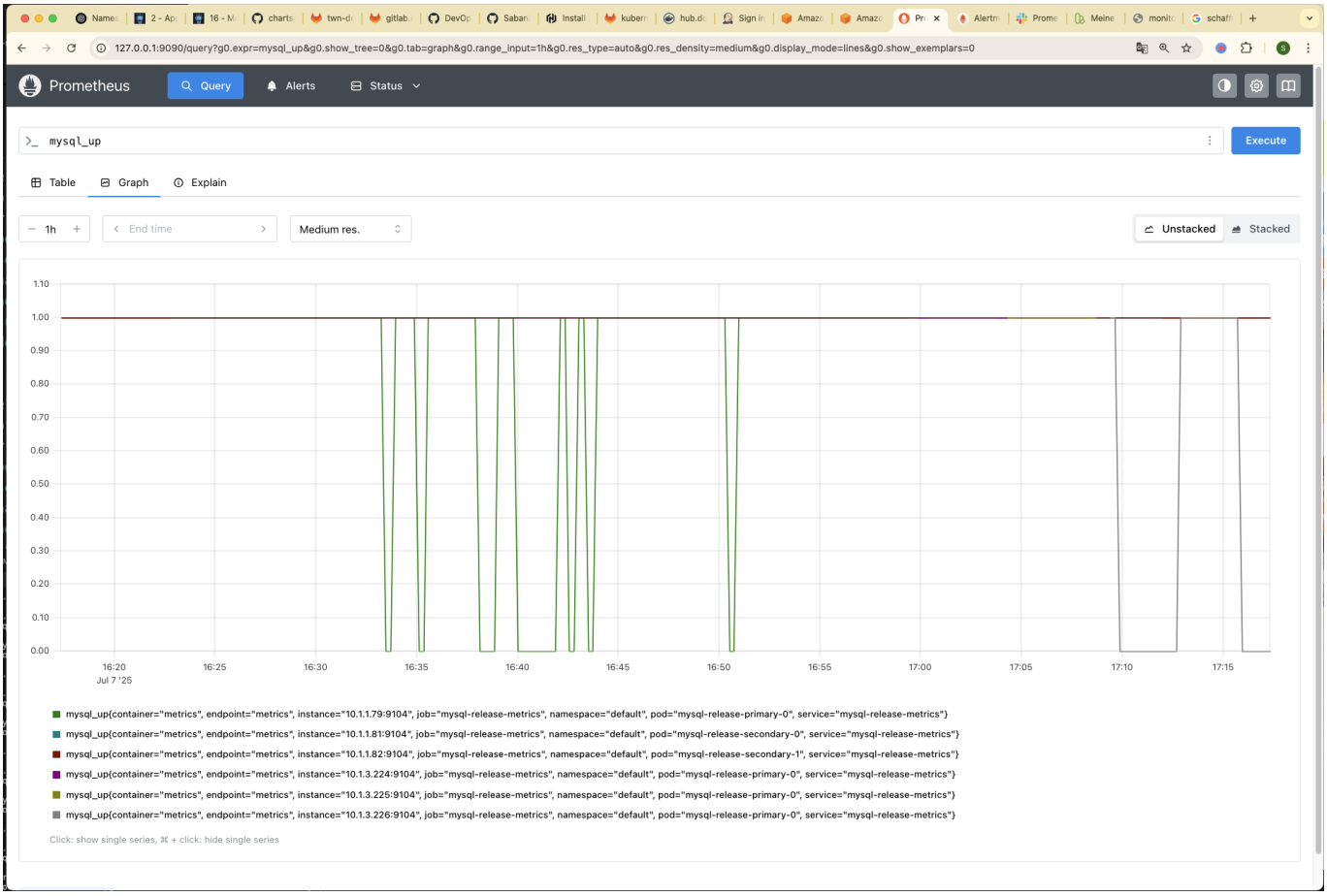
```
kubectl delete job mysql-connection-stress
```

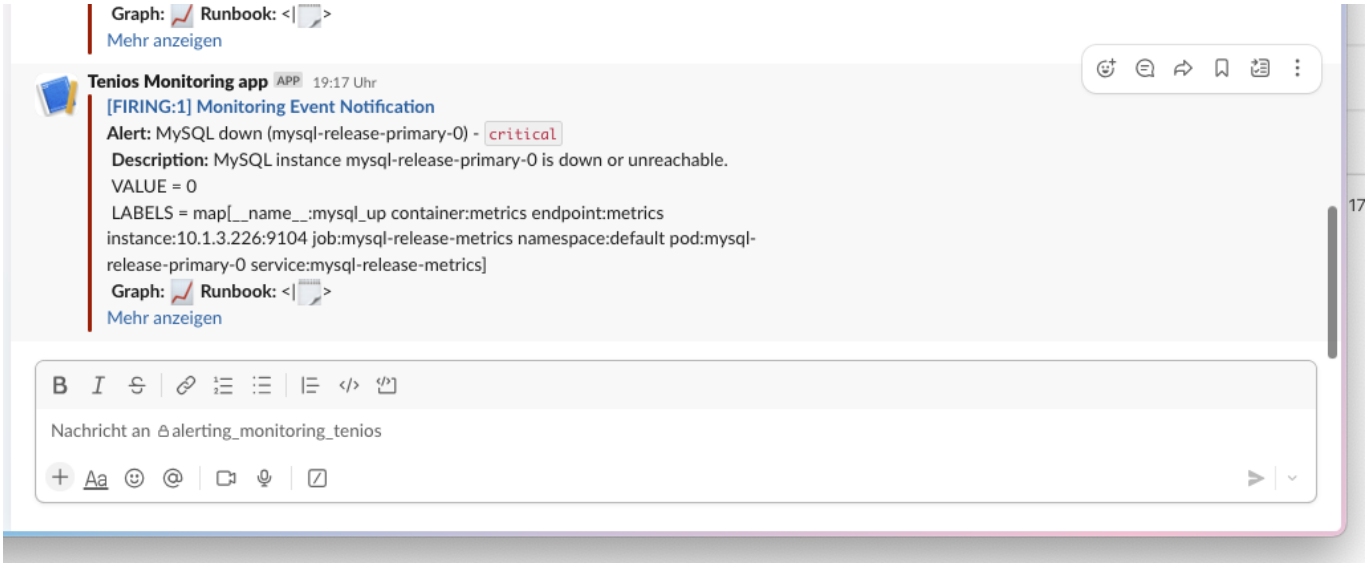
To trigger the MysqlDown alert, I executed the following command to kill the MySQL process inside the primary pod:

```
sgworker@MacBook-Pro-3.local /Users/sgworker/Desktop/prometheus-exercises
[main]
% kubectl exec -it mysql-release-primary-0 -- pkill mysqld

Defaulted container "mysql" out of: mysql, metrics, preserve-logs-symlinks
(init), volume-permissions (init)
command terminated with exit code 1
```




127.0.0.1:9093/#/alerts
Alertmanager
Alerts
Silences
Status
Settings
Help
New Silence
Filter
Group
Receiver: All
Silenced
Inhibited
Muted
Silence
Custom matcher, e.g. env="production"
Expand all groups
null
Not grouped
1 alert
null
Not grouped
4 alerts
null
namespace="default"
4 alerts
null
namespace="kube-system"
6 alerts
monitoring/main-rules-alert-config/null
namespace="monitoring"
4 alerts
monitoring/main-rules-alert-config/slack
namespace="monitoring"
1 alert
2025-07-07T17:16:58.291Z
Info
Source
Silence
Link
alertname="MysqlDown"
container="metrics"
endpoint="metrics"
instance="10.1.3.226:9104"
job="mysql-release-metrics"
pod="mysql-release-primary-0"
prometheus="monitoring/monitoring-stack-kube-prom-prometheus"
service="mysql-release-metrics"
severity="critical"


127.0.0.1:9090/query?g0.expr=mysql_up&g0.show_tree=0&g0.tab=graph&g0.range_input=1h&g0.res_type=auto&g0.res_density=medium&g0.display_mode=lines&g0.show_exemplars=0
Prometheus
Query
Alerts
Status
mysql_up
Execute
Table
Graph
Explain
1h
End time
Medium res.
Unstacked
Stacked
1.10
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
16:20
Jul 7 '25
16:25
16:30
16:35
16:40
16:45
16:50
16:55
17:00
17:05
17:10
17:15
mysql_up{container="metrics", endpoint="metrics", instance="10.1.1.79:9104", job="mysql-release-metrics", namespace="default", pod="mysql-release-primary-0", service="mysql-release-metrics"}
mysql_up{container="metrics", endpoint="metrics", instance="10.1.1.81:9104", job="mysql-release-metrics", namespace="default", pod="mysql-release-secondary-0", service="mysql-release-metrics"}
mysql_up{container="metrics", endpoint="metrics", instance="10.1.1.82:9104", job="mysql-release-metrics", namespace="default", pod="mysql-release-secondary-1", service="mysql-release-metrics"}
mysql_up{container="metrics", endpoint="metrics", instance="10.1.3.224:9104", job="mysql-release-metrics", namespace="default", pod="mysql-release-primary-0", service="mysql-release-metrics"}
mysql_up{container="metrics", endpoint="metrics", instance="10.1.3.225:9104", job="mysql-release-metrics", namespace="default", pod="mysql-release-primary-0", service="mysql-release-metrics"}
mysql_up{container="metrics", endpoint="metrics", instance="10.1.3.226:9104", job="mysql-release-metrics", namespace="default", pod="mysql-release-primary-0", service="mysql-release-metrics"}
Click: show single series, ⌘ + click: hide single series


Graph: Runbook: <|>
Mehr anzeigen
Tenios Monitoring app APP 19:17 Uhr
[FIRING:1] Monitoring Event Notification
Alert: MySQL down (mysql-release-primary-0) - critical
Description: MySQL instance mysql-release-primary-0 is down or unreachable.
VALUE = 0
LABELS = map[__name__:mysql_up container:metrics endpoint:metrics instance:10.1.3.226:9104 job:mysql-release-metrics namespace:default pod:mysql-release-primary-0 service:mysql-release-metrics]
Graph: Runbook: <|>
Mehr anzeigen
Nachricht an alerting_monitoring_tenios
17