

# Задача 10. Мониторинг

## Задача 1: Установка и Настройка Prometheus и Grafana

### Установка Prometheus в кластере Minikube, используя Helm Chart.

1. Запустим minikube

```
minikube start
```

2. Добавим репозиторий prometheus

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

3. Установим готовый Helm чарт

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

4. Экспонируем сервис prometheus-server с помощью NodePort `kubectl expose service prometheus-server --type=NodePort --target-port=9090 --name=prometheus-server-np`

5. Проверим список сервисов

```
ruslan@ruslan-Z690-UD ~/IT-2023/task 10$ kubectl expose service prometheus-server --type=NodePort --target-port=9090 --name=prometheus-server-np
service/prometheus-server-np exposed
ruslan@ruslan-Z690-UD ~/IT-2023/task 10$ kubectl get svc
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	3m39s
prometheus-alertmanager	ClusterIP	10.96.53.92	<none>	9093/TCP	54s
prometheus-alertmanager-headless	ClusterIP	None	<none>	9093/TCP	54s
prometheus-kube-state-metrics	ClusterIP	10.97.18.88	<none>	8080/TCP	54s
prometheus-prometheus-node-exporter	ClusterIP	10.101.42.200	<none>	9100/TCP	54s
prometheus-prometheus-pushgateway	ClusterIP	10.103.199.219	<none>	9091/TCP	54s
prometheus-server	ClusterIP	10.101.17.187	<none>	80/TCP	54s
prometheus-server-np	NodePort	10.101.10.207	<none>	80:30769/TCP	16s

```
ruslan@ruslan-Z690-UD ~/IT-2023/task 10$ kubectl get svc
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	16d
prometheus-alertmanager	ClusterIP	10.106.23.114	<none>	9093/TCP	2m21s
prometheus-alertmanager-headless	ClusterIP	None	<none>	9093/TCP	2m21s
prometheus-kube-state-metrics	ClusterIP	10.105.162.128	<none>	8080/TCP	2m21s
prometheus-prometheus-node-exporter	ClusterIP	10.111.71.214	<none>	9100/TCP	2m21s
prometheus-prometheus-pushgateway	ClusterIP	10.109.113.171	<none>	9091/TCP	2m21s
prometheus-server	ClusterIP	10.102.220.42	<none>	80/TCP	2m21s
prometheus-server-np	NodePort	10.111.21.48	<none>	80:32317/TCP	46s

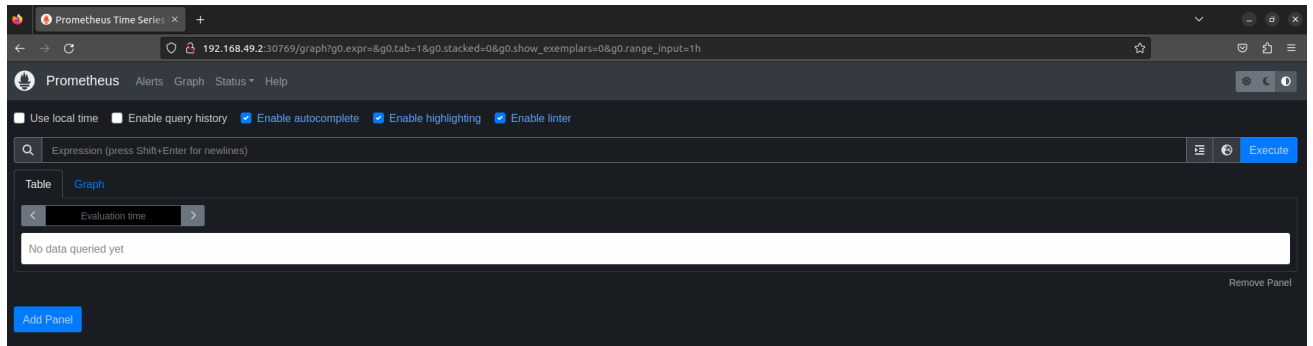
6. Экспортируем URL нашего сервиса

```
minikube service prometheus-server-np --url
```

7. На выходе получаем URL, который будет доступен в браузере

```
ruslan@ruslan-Z690-UD ~/IT-2023/task 10$ minikube service prometheus-server-np --url
http://192.168.49.2:30769
ruslan@ruslan-Z690-UD ~/IT-2023/task 10$
```

## 8. Открываем UI



## Установка Grafana с использованием Helm Chart.

1. Добавим репозиторий grafana

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

2. Установим готовый Helm чарт

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

3. Экспонируем сервис grafana с помощью NodePort

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

4. Проверим список сервисов

```
ruslan@ruslan-Z690-UD ~/IT-2023/task 10$ kubectl get services  
NAME                TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE  
grafana              ClusterIP   10.97.165.34   <none>         80/TCP           37s  
grafana-np           NodePort    10.103.177.185 <none>         80:31618/TCP     26s  
kubernetes           ClusterIP   10.96.0.1     <none>         443/TCP          5m31s  
prometheus-alertmanager ClusterIP   10.96.53.92   <none>         9093/TCP         2m46s  
prometheus-alertmanager-headless ClusterIP   None          <none>         9093/TCP         2m46s  
prometheus-kube-state-metrics ClusterIP   10.97.18.88   <none>         8080/TCP         2m46s  
prometheus-prometheus-node-exporter ClusterIP   10.101.42.200 <none>         9100/TCP         2m46s  
prometheus-prometheus-pushgateway ClusterIP   10.103.199.219 <none>         9091/TCP         2m46s  
prometheus-server    ClusterIP   10.101.17.187 <none>         80/TCP           2m46s  
prometheus-server-np NodePort    10.101.10.207 <none>         80:30769/TCP     2m8s  
ruslan@ruslan-Z690-UD ~/IT-2023/task 10$
```

5. Экспортируем URL сервиса grafana-np

```
minikube service grafana-np --url
```

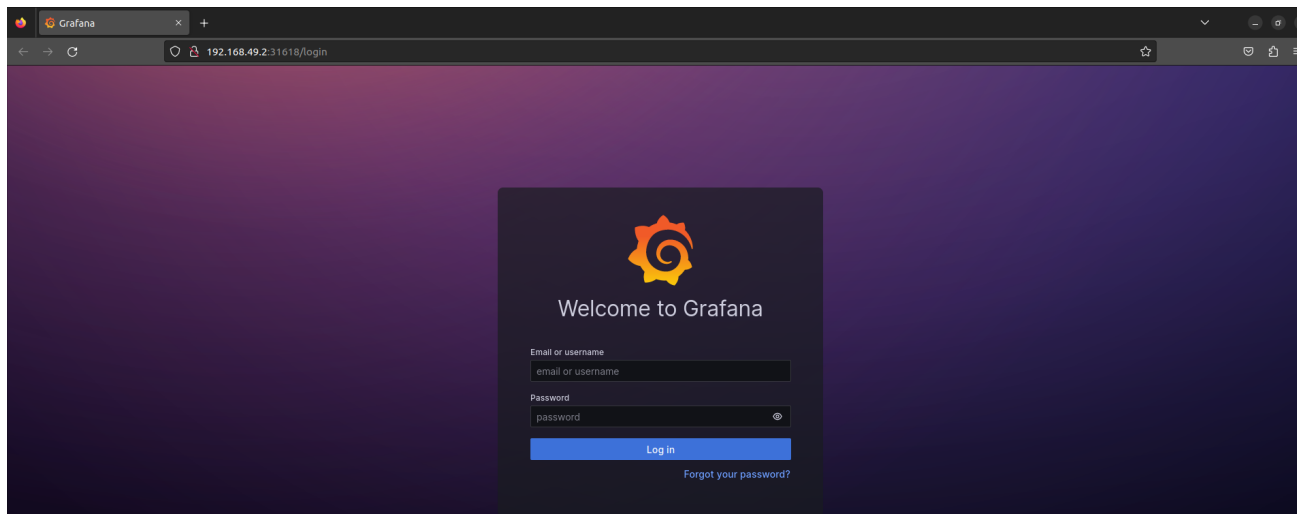
6. На выходе также получаем URL grafana, который будет доступен в браузере

```
ruslan@ruslan-Z690-UD ~/IT-2023/task 10$ minikube service grafana-np --url  
http://192.168.49.2:31651  
ruslan@ruslan-Z690-UD ~/IT-2023/task 10$
```

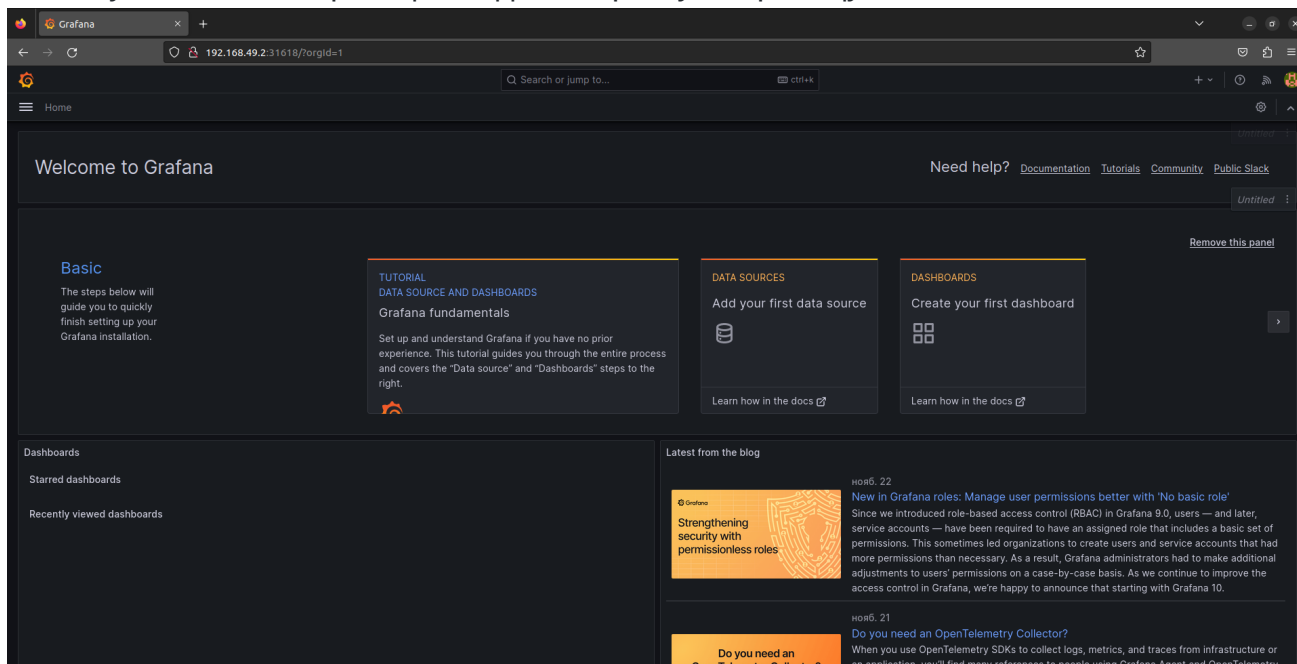
7. Получим пароль для входа в grafana

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

## 8. Откроем UI в браузере и введем логин и пароль

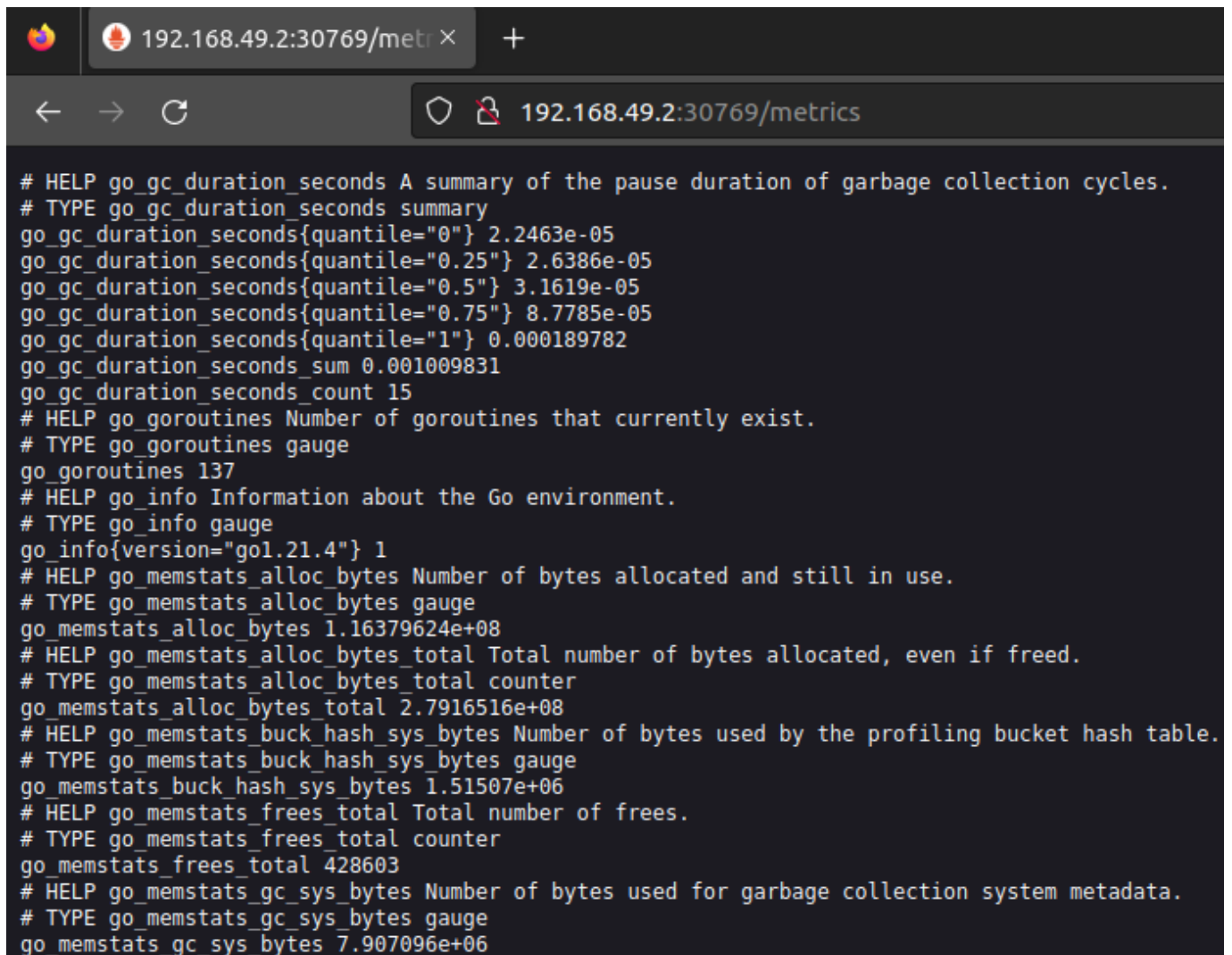


## 9. После успешной авторизации видим стартовую страницу



## Настройка Prometheus для мониторинга самого себя (self-monitoring).

Prometheus предоставляет метрики о себе в конечной точке `/metrics`, следовательно, он может мониторить свое собственное состояние.

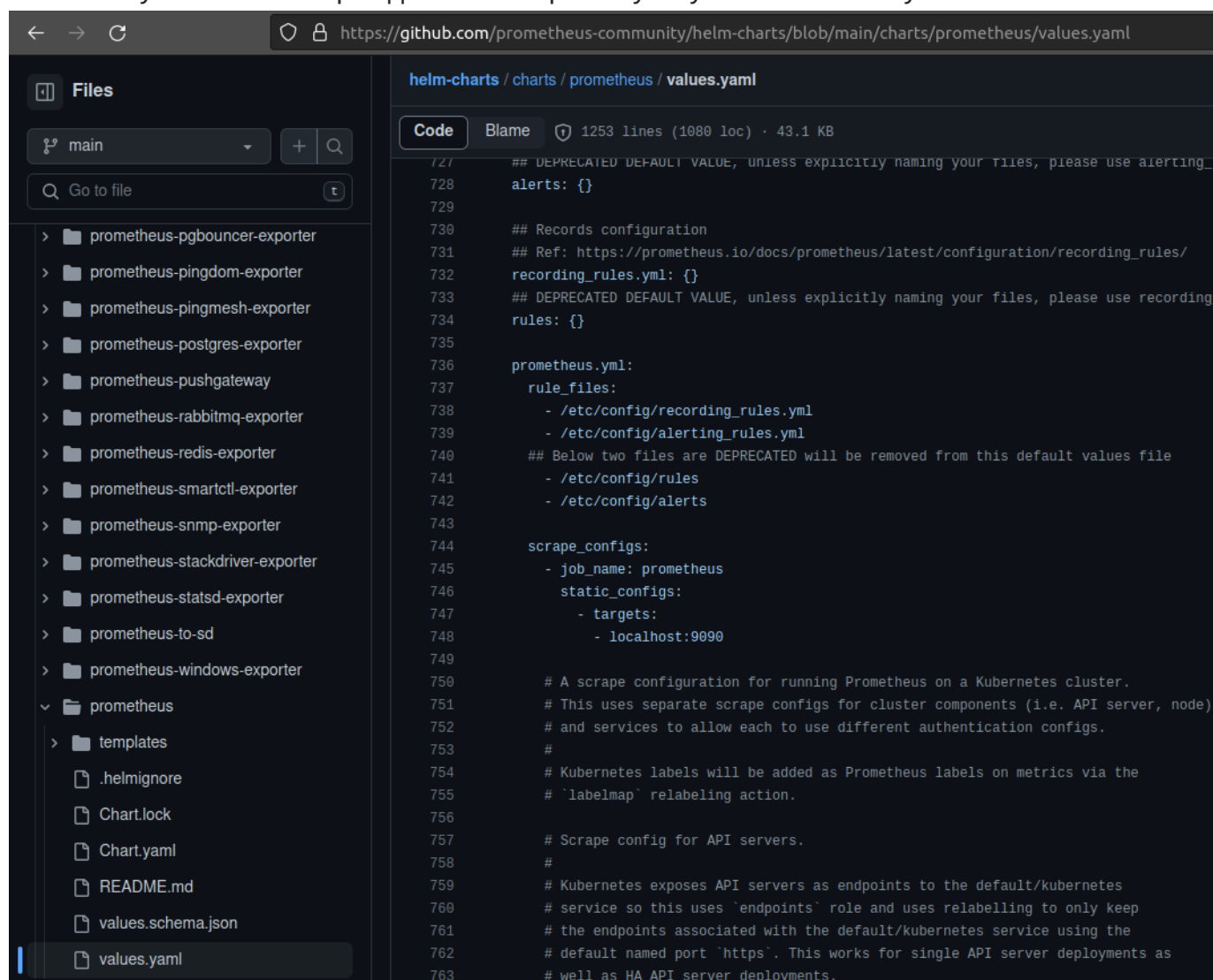


```
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 2.2463e-05
go_gc_duration_seconds{quantile="0.25"} 2.6386e-05
go_gc_duration_seconds{quantile="0.5"} 3.1619e-05
go_gc_duration_seconds{quantile="0.75"} 8.7785e-05
go_gc_duration_seconds{quantile="1"} 0.000189782
go_gc_duration_seconds_sum 0.001009831
go_gc_duration_seconds_count 15
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 137
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.21.4"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 1.16379624e+08
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 2.7916516e+08
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.51507e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 428603
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 7.907096e+06
```

Соответственно, в конфиге `prometheus.yml` должна быть следующая настройка:

```
scrape_configs:
  - job_name: prometheus
    static_configs:
      - targets:
        - localhost:9090
```

В используемом Helm чарте данная настройка уже установлена по-умолчанию:

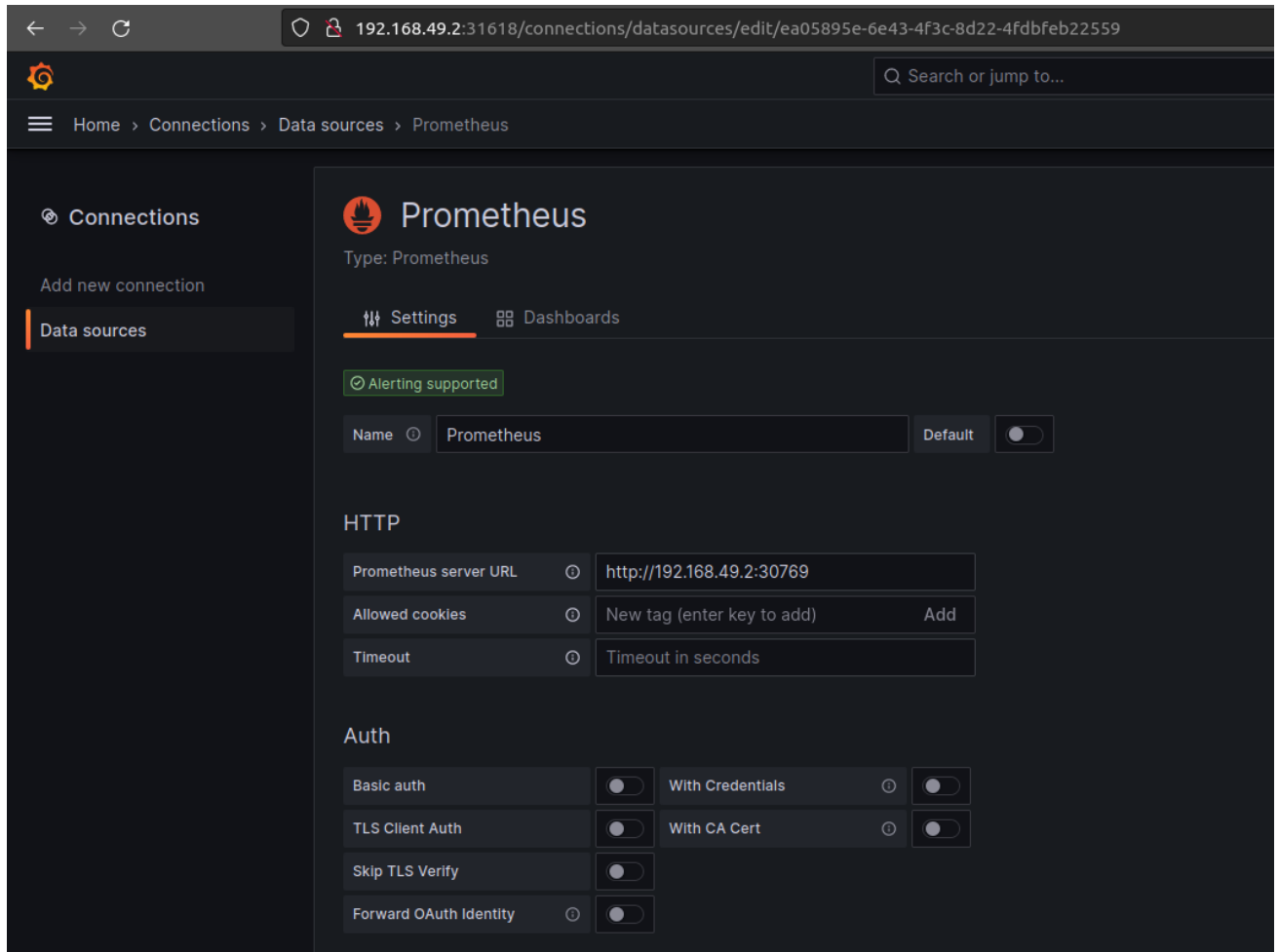


The screenshot shows a web browser displaying the GitHub repository `https://github.com/prometheus-community/helm-charts/blob/main/charts/prometheus/values.yaml`. The left sidebar shows a file explorer with the `prometheus` directory expanded, highlighting `values.yaml`. The main area shows the content of `values.yaml` with line numbers 727 to 763. The code defines default configurations for Prometheus, including alerting rules, recording rules, and scrape configurations.

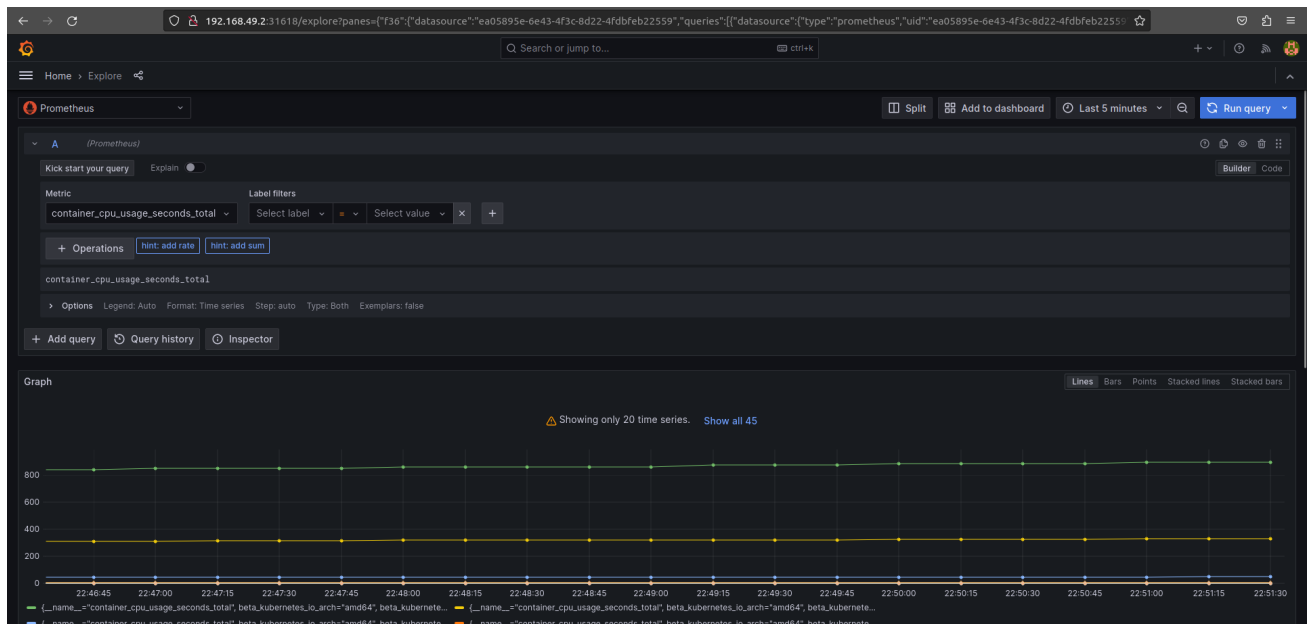
```
727  ## DEPRECATED DEFAULT VALUE, unless explicitly naming your files, please use alerting_
728  alerts: {}
729
730  ## Records configuration
731  ## Ref: https://prometheus.io/docs/prometheus/latest/configuration/recording_rules/
732  recording_rules.yml: {}
733  ## DEPRECATED DEFAULT VALUE, unless explicitly naming your files, please use recording_
734  rules: {}
735
736  prometheus.yml:
737    rule_files:
738      - /etc/config/recording_rules.yml
739      - /etc/config/alerting_rules.yml
740    ## Below two files are DEPRECATED will be removed from this default values file
741    - /etc/config/rules
742    - /etc/config/alerts
743
744  scrape_configs:
745    - job_name: prometheus
746      static_configs:
747        - targets:
748            - localhost:9090
749
750  # A scrape configuration for running Prometheus on a Kubernetes cluster.
751  # This uses separate scrape configs for cluster components (i.e. API server, node)
752  # and services to allow each to use different authentication configs.
753  #
754  # Kubernetes labels will be added as Prometheus labels on metrics via the
755  # 'labelmap' relabeling action.
756
757  # Scrape config for API servers.
758  #
759  # Kubernetes exposes API servers as endpoints to the default/kubernetes
760  # service so this uses 'endpoints' role and uses relabelling to only keep
761  # the endpoints associated with the default/kubernetes service using the
762  # default named port 'https'. This works for single API server deployments as
763  # well as HA API server deployments.
```

**Создание источника данных (data source) Prometheus в Grafana.**

## 1. Перейдем в интерфейс grafana и добавим новый источник данных для prometheus



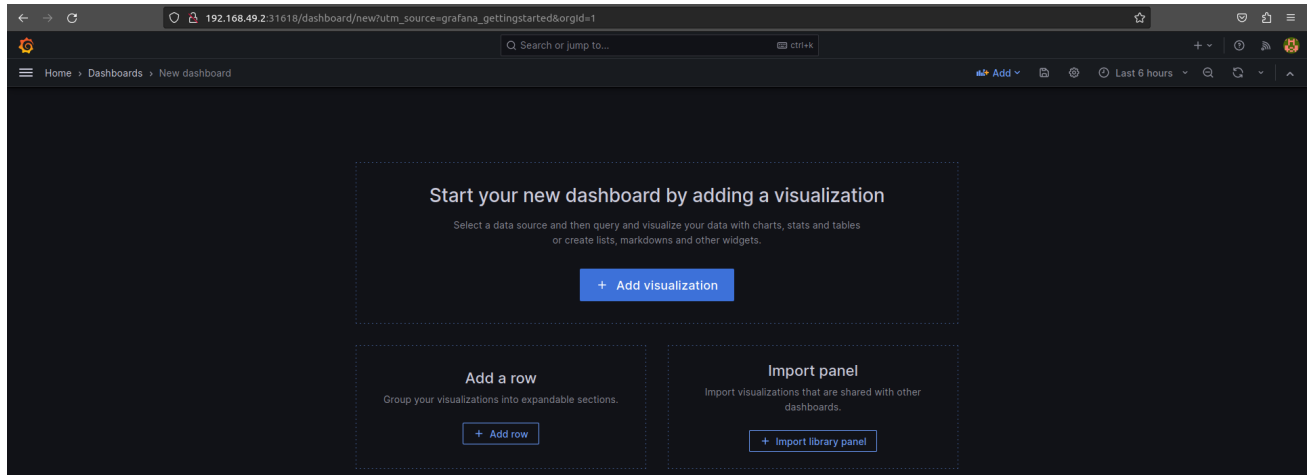
## 2. Убедимся в работоспособности, выведя одну из метрик в качестве графика



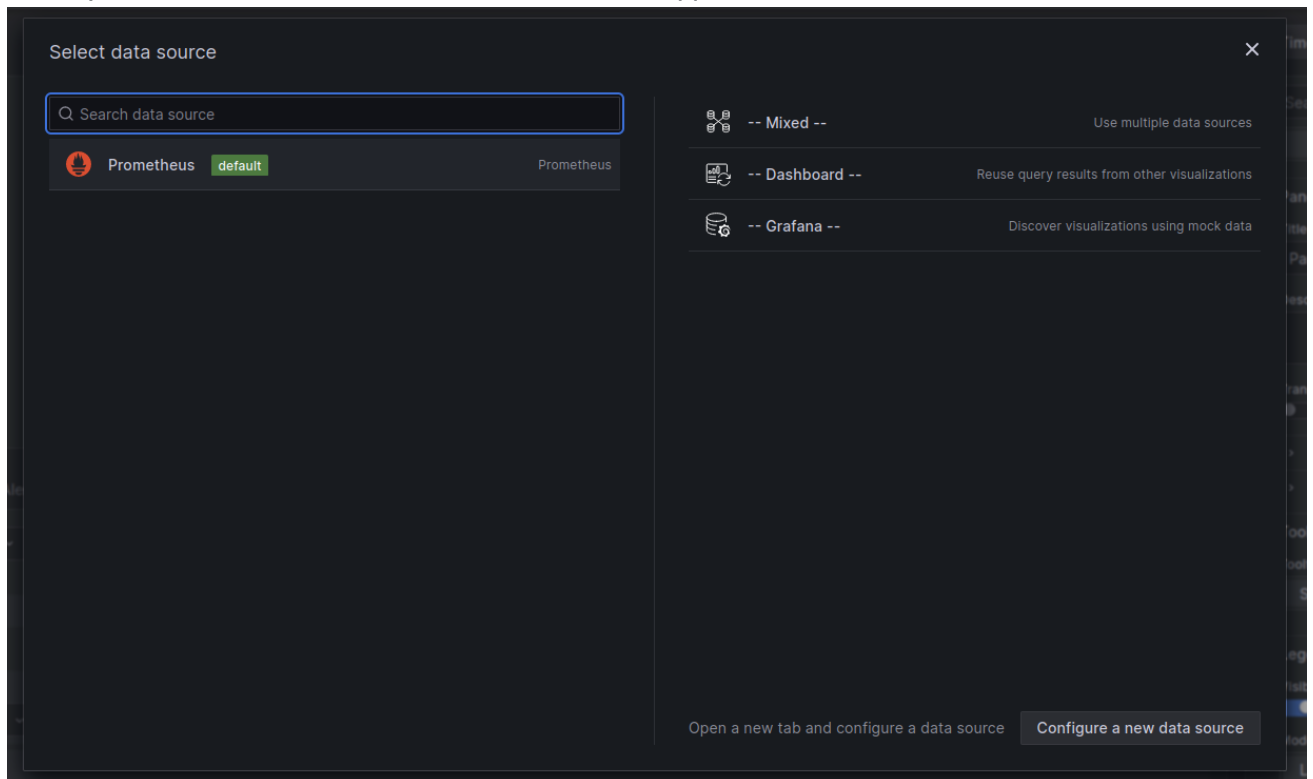
## Задача 2: Создание Дашборда в Grafana

# Создание пользовательского дашборда в интерфейсе Grafana

## 1. Переходим в интерфейс и создаем новый дашборд



## 2. Выбираем Prometheus в качестве источника данных



### 3. Добавим график временного ряда для метрики `prometheus_http_requests_total`

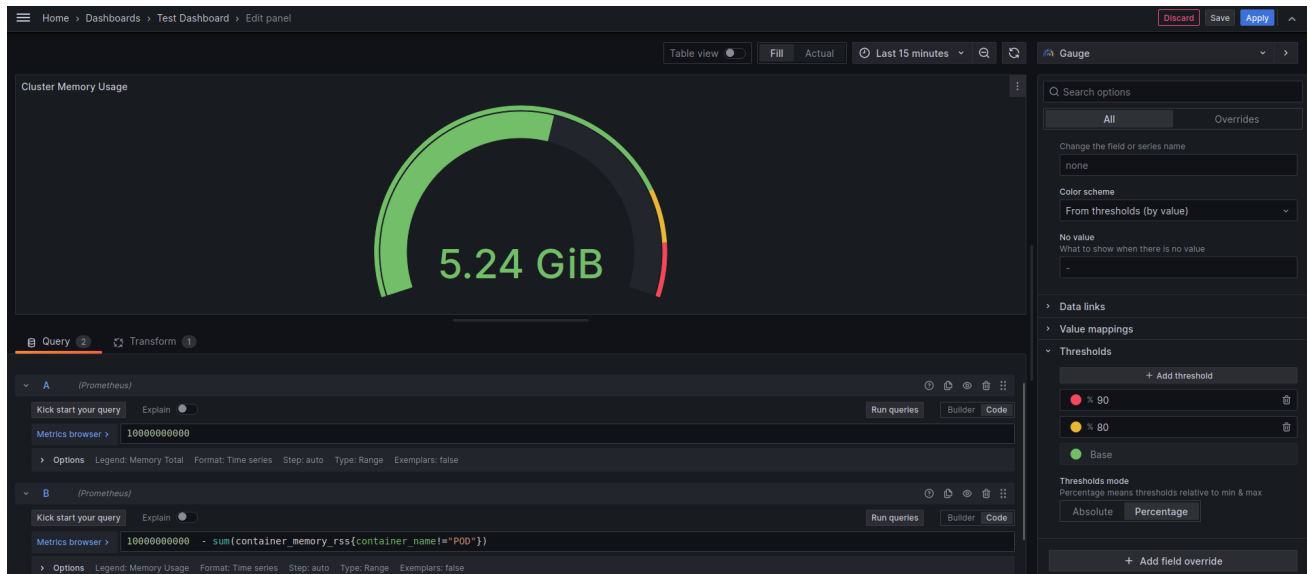


### 4. Добавим график для мониторинга нагрузки на CPU для подов в %

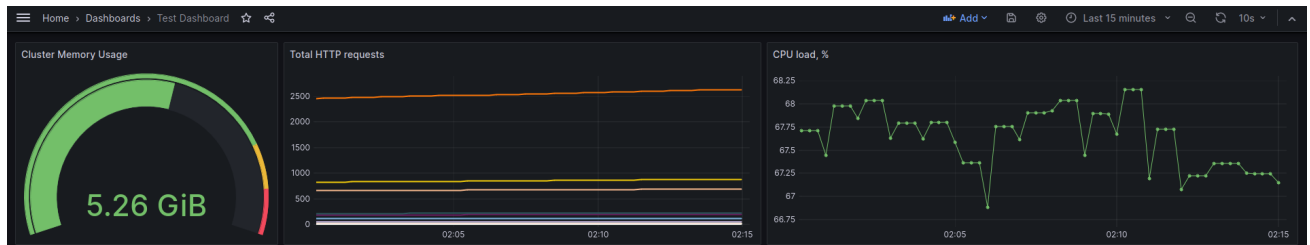


### 5. Также добавим график в виде измерительного прибора (gauge) для мониторинга нагрузки на оперативную память





## 6. Настроим основную панель дашборда для отображения созданных графиков



# Задача 3: Создание Алертов в Prometheus и Интеграция с Grafana

## Создание правила алерта в файле конфигурации Prometheus

1. Из репозитория установленного Prometheus сохраняем локально файл `values.yaml` и добавляем в него правила алертов в разделе `serverFiles`. В качестве тестового примера реализуем алерт для высокой нагрузки на CPU и RAM.

```
## Prometheus server ConfigMap entries
##
serverFiles:
  ## Alerts configuration
  ## Ref: https://prometheus.io/docs/prometheus/latest/configuration/alerting_rules/
  alerting_rules.yml:
    groups:
      - name: Main
        rules:
          - alert: HighCpuLoad
            expr: 100 * sum(rate(container_cpu_usage_seconds_total{container_name!="POD"}[15m])) > 67
            for: 1m
            annotations:
              summary: High CPU load on {{ $labels.instance }}
          - alert: HighMemoryLoad
            expr: 100 * (sum(container_memory_rss{container_name!="POD"}) / 10000000000) > 80
            for: 1m
            annotations:
              summary: High RAM load on {{ $labels.instance }}
```

2. С помощью Helm обновляем файл конфигурации

```
helm upgrade prometheus -f values.yaml prometheus-community/prometheus
```

3. Убеждаемся в успешности в выполнения

```
ruslan@ruslan-Z690-UD: ~/IT-2023/task 10$ helm upgrade prometheus -f values.yaml prometheus-community/prometheus
Release "prometheus" has been upgraded. Happy Helming!
NAME: prometheus
LAST DEPLOYED: Sun Nov 26 18:38:56 2023
NAMESPACE: default
STATUS: deployed
REVISION: 8
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
```

4. Далее, переходим в интерфейс Prometheus и видим наши настройки в разделе rules

192.168.49.2:30769/rules

Prometheus Alerts Graph Status Help

### Rules

Rule	Interval: 1m 0s	3.473s ago	3.222ms	
<b>Rule</b>	<b>State</b>	<b>Error</b>	<b>Last Evaluation</b>	<b>Evaluation Time</b>
<b>alert:</b> HighCpuLoad <b>expr:</b> 100 * sum(rate(container_cpu_usage_seconds_total{container_name!="POD"}[15m])) > 67 <b>for:</b> 1m <b>annotations:</b> summary: High CPU load on {{ \$labels.instance }}	OK		3.475s ago	1.889ms
<b>alert:</b> HighMemoryLoad <b>expr:</b> 100 * (sum(container_memory_rss{container_name!="POD"}) / 1e+10) > 80 <b>for:</b> 1m <b>annotations:</b> summary: High RAM load on {{ \$labels.instance }}	OK		3.474s ago	1.287ms

5. В разделе alerts отображается состояние алертов

192.168.49.2:30769/alerts?search=

Prometheus Alerts Graph Status Help

☒ Inactive (1) ☒ Pending (1) ☒ Firing (0)  ☐ Show annotations

/etc/config/alerting\_rules.yml > Main Inactive pending (1)

HighCpuLoad (1 active)

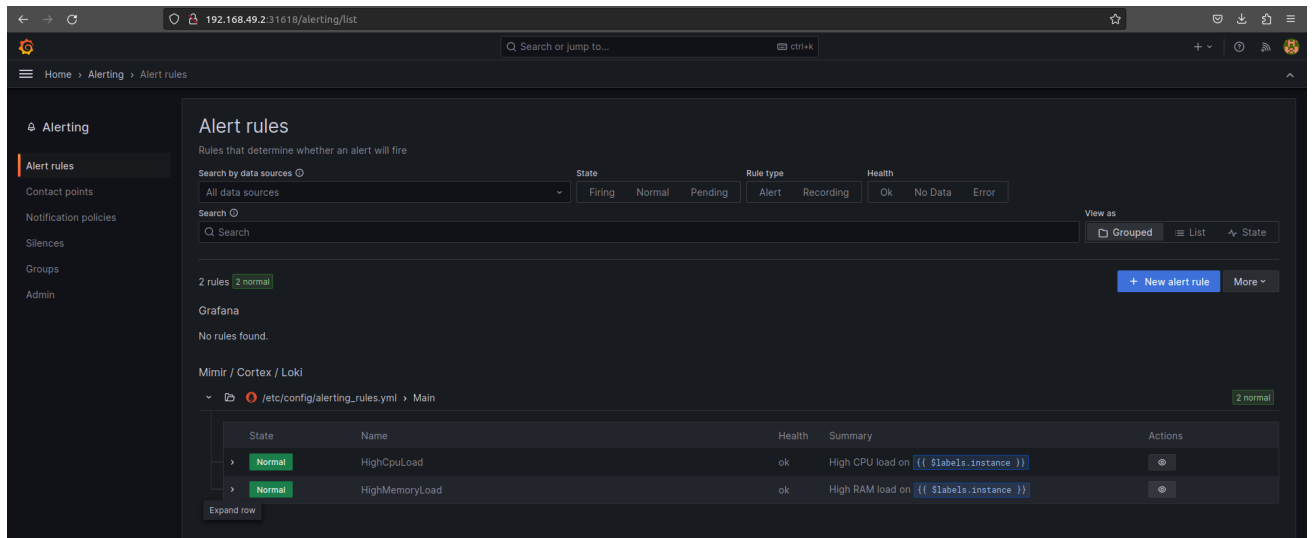
**name:** HighCpuLoad  
**expr:** 100 \* sum(rate(container\_cpu\_usage\_seconds\_total{container\_name!="POD"}[15m])) > 67  
**for:** 1m  
**annotations:**  
summary: High CPU load on {{ \$labels.instance }}

Labels	State	Active Since	Value
alertname:HighCpuLoad	PENDING	2023-11-26T21:20:47.298041106Z	68.19029493737716

HighMemoryLoad (0 active)

**name:** HighMemoryLoad  
**expr:** 100 \* (sum(container\_memory\_rss{container\_name!="POD"}) / 1e+10) > 80  
**for:** 1m  
**annotations:**  
summary: High RAM load on {{ \$labels.instance }}

6. Также наши алерты автоматически появились в интерфейсе Grafana в разделе Alert Rules



Создание панели в дашборде Grafana, которая будет отображать статус алертов.

1. Переходим в дашборд, и создаем новый график с типом Alert List, который будет

Alert list

Q Search options

Alert name

Filter for alerts containing this text

Alert instance label

Filter alert instances using label querying, ex: {severity="critical", instance=~"cluster-us-.\*"}

Datasource

Filter alerts from selected datasource

Select data source

Folder

Filter for alerts in the selected folder (only for Grafana alerts)

Choose

Alert state filter

Alerting / Firing

☒

Pending

☐

No Data

☐

Normal

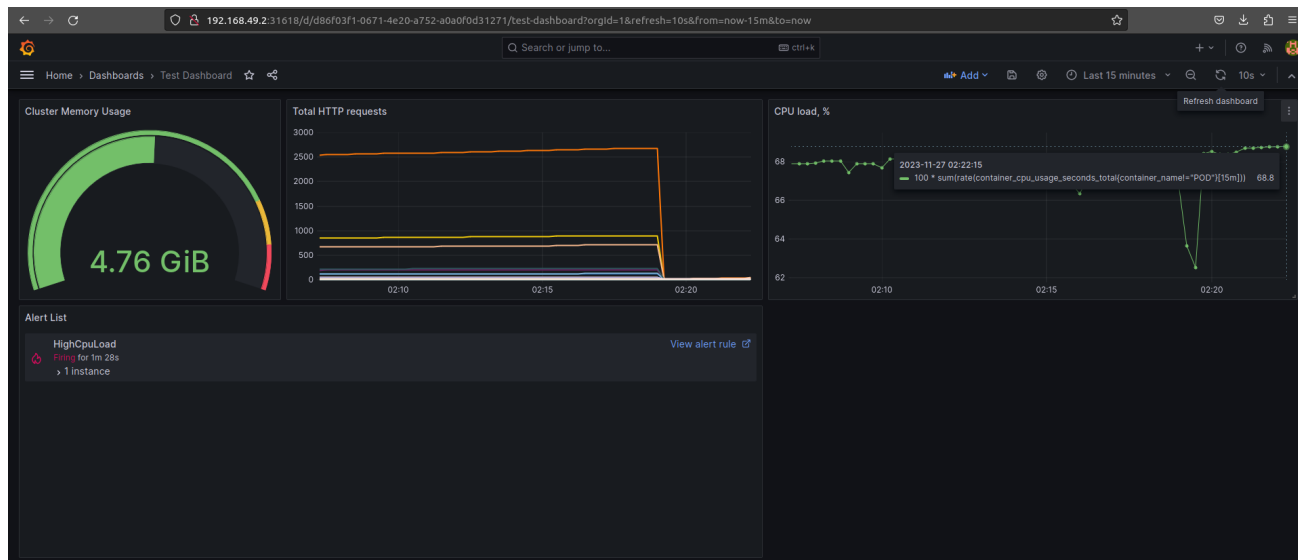
☐

Error

☐

отображать 'firing' алерты.

## 2. Добавляем список на панель дашборда



## Задача 4: Масштабирование Minikube и Мониторинг с Prometheus

### Увеличение количества реплик пода в Minikube кластере и их влияние на отображение метрик в Grafana

Масштабирование в k8s выполняется с помощью изменения количества реплик в деплойменте.

#### 1. Посмотрим текущее состояние деплойментов и подов

```
ruslan@ruslan-Z690-UD ~/IT-2023/task 10$ kubectl get deployments
NAME                                READY    UP-TO-DATE    AVAILABLE    AGE
grafana                             1/1      1              1            22h
prometheus-kube-state-metrics       1/1      1              1            22h
prometheus-prometheus-pushgateway  1/1      1              1            22h
prometheus-server                   1/1      1              1            22h
ruslan@ruslan-Z690-UD ~/IT-2023/task 10$ kubectl get pods
NAME                                READY    STATUS      RESTARTS      AGE
grafana-77b7f5dc85-9wmcg            1/1      Running     1 (19h ago)   22h
prometheus-alertmanager-0           1/1      Running     1 (19h ago)   22h
prometheus-kube-state-metrics-85596bfd6-9n44f  1/1      Running     2 (7h16m ago) 22h
prometheus-prometheus-node-exporter-hhpx4     1/1      Running     1 (19h ago)   22h
prometheus-prometheus-pushgateway-79745d4495-mbq85  1/1      Running     1 (19h ago)   22h
prometheus-server-68f4567df8-c59fg           2/2      Running     0              2m38s
ruslan@ruslan-Z690-UD ~/IT-2023/task 10$
```

#### 2. Далее отмасштабируем деплоймент grafana до 4 реплик. Для этого воспользуемся командой kubectl scale, для которой укажем тип объекта - деплоймент, его название и количество желаемых экземпляров

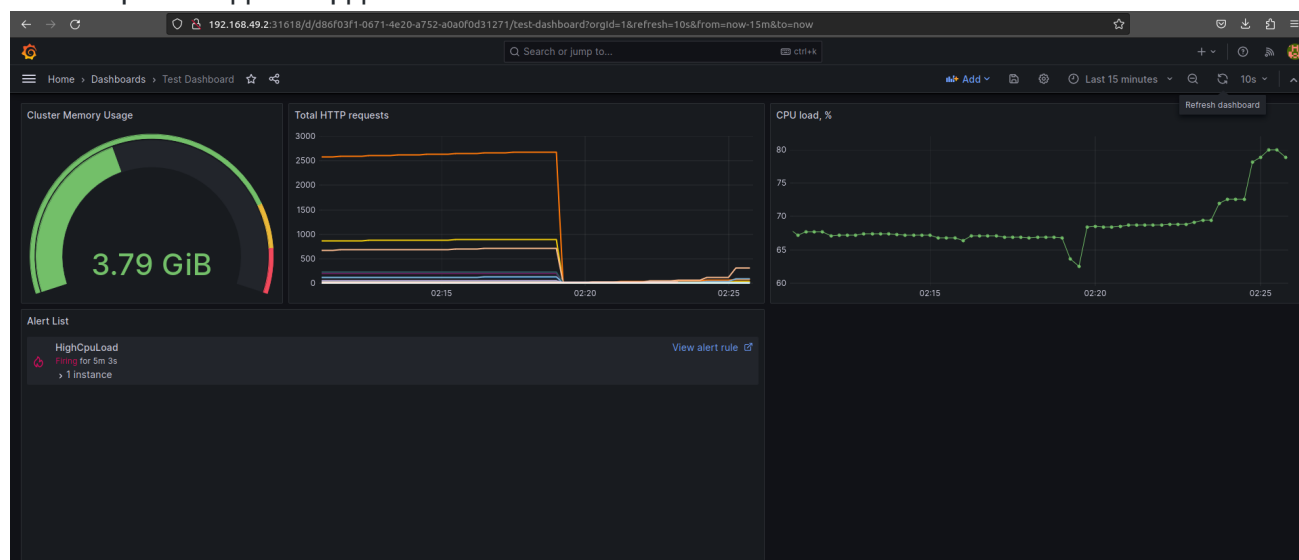
```
kubectl scale deploy grafana --replicas=4
```

### 3. Проверим состояние после масштабирования

```
ruslan@ruslan-Z690-UD ~/IT-2023/task 10$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
grafana-77b7f5dc85-9wmcp           1/1     Running   1 (20h ago) 24h
grafana-77b7f5dc85-htbw4           1/1     Running   0           15s
grafana-77b7f5dc85-lqrmr           1/1     Running   0           15s
grafana-77b7f5dc85-vbdn7           1/1     Running   0           15s
prometheus-alertmanager-0           1/1     Running   1 (20h ago) 24h
prometheus-kube-state-metrics-85596bfd6-9n44f 1/1     Running   2 (8h ago)  24h
prometheus-prometheus-node-exporter-hhpx4 1/1     Running   1 (20h ago) 24h
prometheus-prometheus-pushgateway-79745d4495-mbq85 1/1     Running   1 (20h ago) 24h
prometheus-server-68f4567df8-8htm4 2/2     Running   0           10m

ruslan@ruslan-Z690-UD ~/IT-2023/task 10$ kubectl get deployments
NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
grafana                             4/4     4             4           24h
prometheus-kube-state-metrics        1/1     1             1           24h
prometheus-prometheus-pushgateway    1/1     1             1           24h
prometheus-server                    1/1     1             1           24h
```

### 4. Посмотрим на дашборд



### 5. Видим повышение объема используемого CPU

