

Серикова Виктория

Подготовка

Включаем эндпоинт прометеуса в актуаторе, добавляем аннотации в поды, чтобы прометеус мог скрапить:

```
management:
  exposure: "health, info, prometheus"
```

```
annotations:
prometheus.io/scrape: "true"
prometheus.io/path: "/actuator/prometheus"
prometheus.io/port: "{{ .Values.container.port }}"
```

Также включаем скрэп для постгреса:

```
metrics:
  enabled: true
  service:
    annotations:
      prometheus.io/scrape: "true"
      prometheus.io/port: "9187"
      prometheus.io/path: "/metrics"
```

Прометеус будем поднимать через хельм чарт:

```
- name: prometheus
  namespace: monitoring
  chart: prometheus-community/prometheus
  values:
    - ./prometheus/values.yaml
```

для него еще создадим неймспейс monitoring.

```
kubectl create namespace monitoring
```

(можно в целом и в дефолтном)

далее поднимаем хельм чарт командой

```
helmfile apply
```

ждем, пока все поднимется.

```

kubectl get pods -n monitoring

```

NAME	READY	STATUS	RESTARTS	AGE
prometheus-alertmanager-0	1/1	Running	0	16m
prometheus-kube-state-metrics-8459ccf44c-29w9c	1/1	Running	0	16m
prometheus-prometheus-node-exporter-2qpxf	1/1	Running	0	16m
prometheus-prometheus-pushgateway-68757884b8-fxmhj	1/1	Running	0	16m
prometheus-server-f7bd75bb6-sd5z8	2/2	Running	0	16m

image-25.png

```

(base) vikas@MacBook-Air-Viktoria prometheus_sem % kubectl get pods

```

NAME	READY	STATUS
muffin-currency-5695f68474-pv5hd	1/1	Running
muffin-wallet-59d8dcdb5-2cnb8	1/1	Running
postgres-postgresql-0	2/2	Running

image-26.png

идем на <https://muffin.local/actuator/prometheus>, видим метрики, которые будет отдавать актуатор.**

```
# HELP application_ready_time_seconds Time taken for the application to be ready to service requests
# TYPE application_ready_time_seconds gauge
application_ready_time_seconds{main_application_class="ru.hse.muffin.wallet.server.MuffinWalletApplication"} 68.414
# HELP application_started_time_seconds Time taken to start the application
# TYPE application_started_time_seconds gauge
application_started_time_seconds{main_application_class="ru.hse.muffin.wallet.server.MuffinWalletApplication"} 68.162
# HELP disk_free_bytes Usable space for path
# TYPE disk_free_bytes gauge
disk_free_bytes{path="/."} 2.0655972352E10
# HELP disk_total_bytes Total space for path
# TYPE disk_total_bytes gauge
disk_total_bytes{path="/."} 6.2671097856E10
# HELP executor_active_threads The approximate number of threads that are actively executing tasks
# TYPE executor_active_threads gauge
executor_active_threads{name="applicationTaskExecutor"} 0.0
# HELP executor_completed_tasks_total The approximate total number of tasks that have completed execution
# TYPE executor_completed_tasks_total counter
executor_completed_tasks_total{name="applicationTaskExecutor"} 0.0
# HELP executor_pool_core_threads The core number of threads for the pool
# TYPE executor_pool_core_threads gauge
executor_pool_core_threads{name="applicationTaskExecutor"} 8.0
# HELP executor_pool_max_threads The maximum allowed number of threads in the pool
# TYPE executor_pool_max_threads gauge
executor_pool_max_threads{name="applicationTaskExecutor"} 2.147483647E9
# HELP executor_pool_size_threads The current number of threads in the pool
# TYPE executor_pool_size_threads gauge
executor_pool_size_threads{name="applicationTaskExecutor"} 0.0
# HELP executor_queue_remaining_tasks The number of additional elements that this queue can ideally accept without blocking
# TYPE executor_queue_remaining_tasks gauge
executor_queue_remaining_tasks{name="applicationTaskExecutor"} 2.147483647E9
# HELP executor_queued_tasks The approximate number of tasks that are queued for execution
# TYPE executor_queued_tasks gauge
executor_queued_tasks{name="applicationTaskExecutor"} 0.0
# HELP hikaricp_connections Total connections
# TYPE hikaricp_connections gauge
hikaricp_connections{pool="HikariPool-1"} 10.0
# HELP hikaricp_connections_acquire_seconds Connection acquire time
# TYPE hikaricp_connections_acquire_seconds summary
hikaricp_connections_acquire_seconds_count{pool="HikariPool-1"} 10
hikaricp_connections_acquire_seconds_sum{pool="HikariPool-1"} 0.008391127
# HELP hikaricp_connections_acquire_seconds_max Connection acquire time
# TYPE hikaricp_connections_acquire_seconds_max gauge
hikaricp_connections_acquire_seconds_max{pool="HikariPool-1"} 0.001888334
# HELP hikaricp_connections_active Active connections
# TYPE hikaricp_connections_active gauge
hikaricp_connections_active{pool="HikariPool-1"} 0.0
# HELP hikaricp_connections_creation_seconds Connection creation time
# TYPE hikaricp_connections_creation_seconds summary
hikaricp_connections_creation_seconds_count{pool="HikariPool-1"} 2
hikaricp_connections_creation_seconds_sum{pool="HikariPool-1"} 0.158
# HELP hikaricp_connections_creation_seconds_max Connection creation time
```

Пробрасываем порт для прометеуса:

```
kubectl port-forward -n monitoring svc/prometheus-server 9090:80
```

и идем на локалхост 9090, видим что прометеус собирает метрики с приложения

← ↻ Не защищено localhost:9090 Prometheus Time Series Collection and Processing Server

Crio Blog

Prometheus

Query Alerts Status > Target health

Select scrape pool Filter by target health muffin

ⓘ Hiding pools with no matching targets

Hiding 9 empty pools due to filters or no targets. [Show empty pools](#)

kubernetes-pods 1/1 up

Endpoint	Labels	Last scrape	State
http://10.244.1.184:8081/actuator/prometheus	<div>app_kubernetes_io_instance="muffin-wallet" app_kubernetes_io_name="muffin-wallet"</div> <div>instance="10.244.1.184:8081" job="kubernetes-pods" namespace="default"</div> <div>node="minikube" pod="muffin-wallet-59d8dcdb5-2cnb8" pod_template_hash="59d8dcdb5"</div>	<div>2.136s ago</div> <div>37ms</div>	UP

image-27.png

Запросы

Количество запросов в секунду по каждому методу REST API вашего приложения.

```
sum by (method, uri) (  
rate(http_server_requests_seconds_count[5m])  
)
```

Проверить можно следующим скриптом:

```
for i in {1..30}; do curl -s -o /dev/null http://muffin.local/v1/muffin-wallets; done
```

Я просто поспамила пост и гет запросами через сваггер, было видно, что метрика растет.

К сожалению, почему-то ингресс создает на маке сильную задержку: все запросы очень долго грузят, а пробрасывать порты напрямую к поду не по заданию.



image-29.png

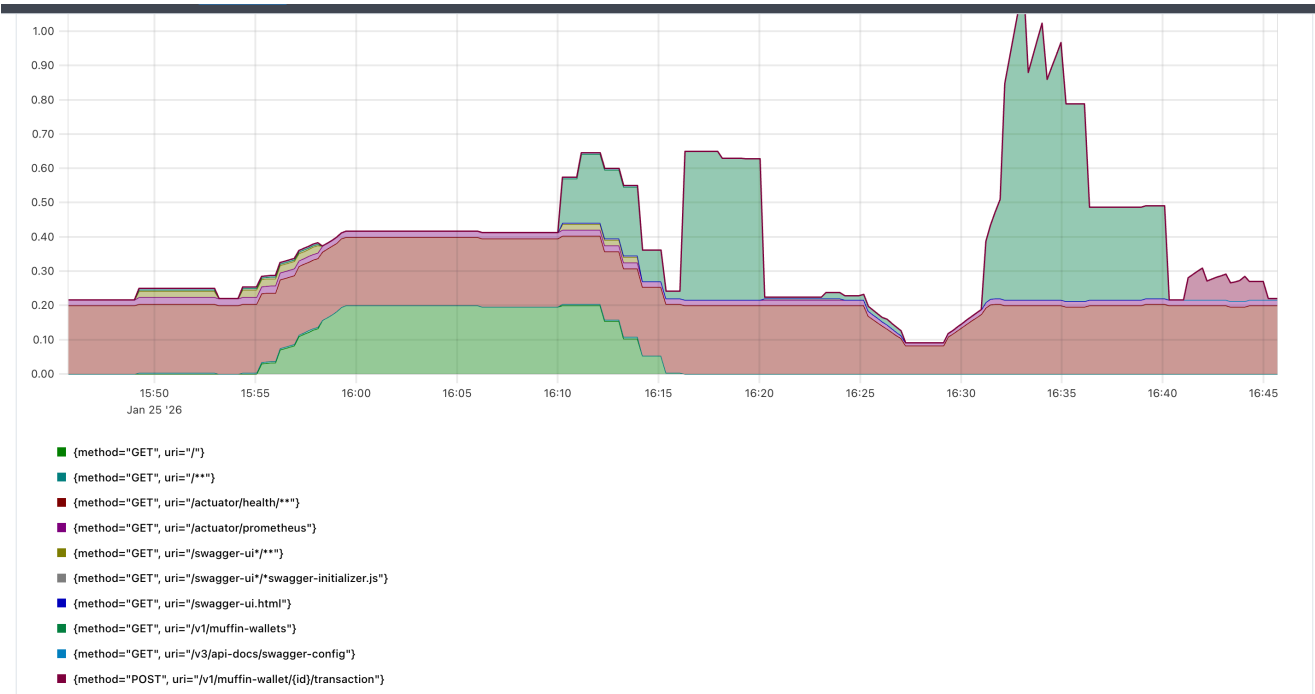
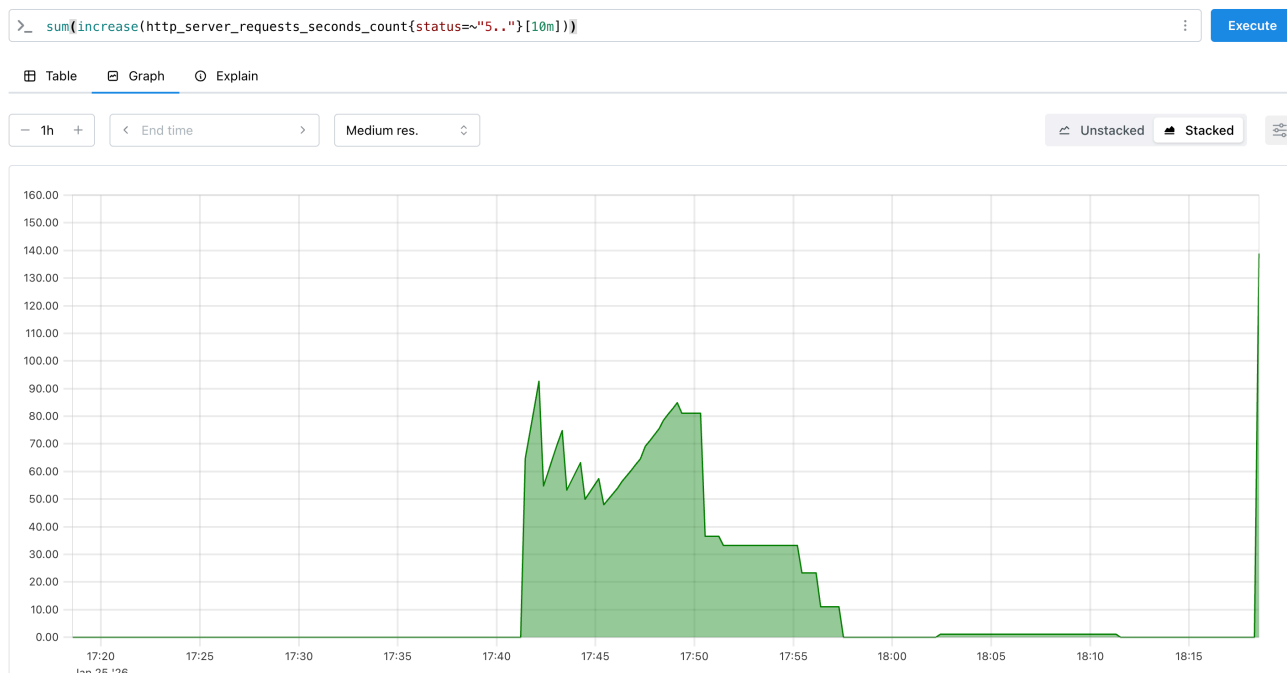


image-30.png

Количество ошибок в логах приложения.

```
sum(increase(http_server_requests_seconds_count{status=~"5.."}[10m]))
```

я много раз отослала пост запрос на транзакцию между одним и тем же счетом (это вызывает 500). Предварительно надо создать счет.



скрипт для проверки:

```
for i in {1..30}; do
  curl -s -o /dev/null -w "%{http_code}\n" \
  -X POST "http://muffin.local/v1/muffin-wallet/51044ea0-35c5-469c-b681-
d0aca5ae4a30/transaction" \
  -H "accept: application/json" \
  -H "Content-Type: application/json" \
  -d '{"to_muffin_wallet_id":"51044ea0-35c5-469c-b681-
d0aca5ae4a30","amount":0}'
done
```

99-й персентиль времени ответа HTTP (обработка запросов)

```
histogram_quantile(
0.99,
sum by (le, method, uri) (
rate(http_server_requests_seconds_bucket[5m])
)
)
```

или так

```
max by (method, uri) (  
http_server_requests_seconds{quantile="0.99"}  
)
```

для проверки нужно создать нагрузку

```
seq 1 200 | xargs -I{} -P 20 curl -s -o /dev/null  
"http://muffin.local/v1/muffin-wallets?owner_name=string&page=0&size=1"
```

У меня к сожалению были пустые результаты, возможно гистрограммы не включены в настройках приложения.

Количество активных соединений к базе данных PostgreSQL.

```
sum(pg_stat_activity_count{datname="muffin_wallet", state="active"})
```

Для проверки нужно создать нагрузку, которая будет создавать конекшены к бд

```
URL="http://muffin.local/v1/muffin-wallets?  
owner_name=string&page=0&size=1"  
seq 1 500 | xargs -I{} -P 50 curl -s -o /dev/null "$URL"
```