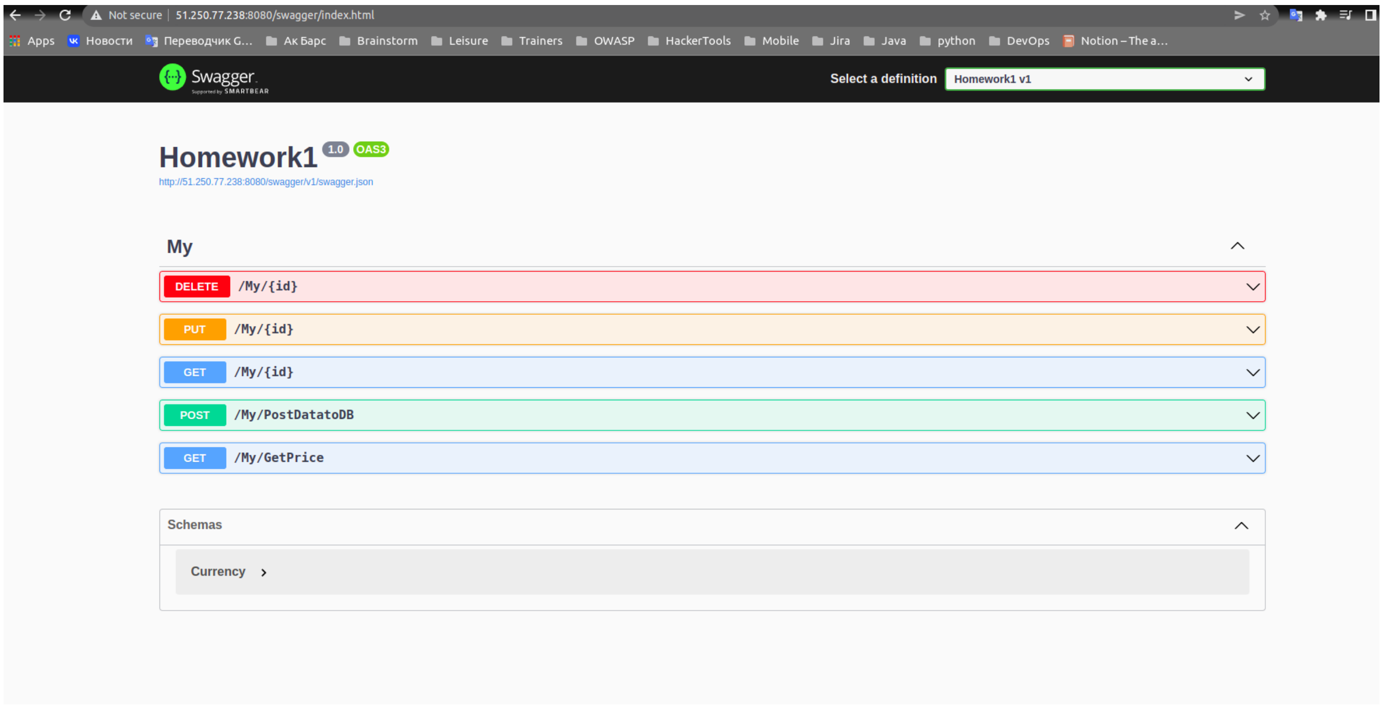
4.11 Централизованная обработка логов Loki, ELK

Для начала, на двух тачках я поднял свое приложение, все через докер композ, для этого у меня есть ветка с пайплайном (https://gitlab.com/xokage/exchangeservice/-/tree/loki):

1. 

Not secure | 51.250.77.238:8080/swagger/index.html

Swagger

Select a definition: Homework1 v1

Homework1 1.0 OAS3

http://51.250.77.238:8080/swagger/v1/swagger.json

My

DELETE `/My/{id}`

PUT `/My/{id}`

GET `/My/{id}`

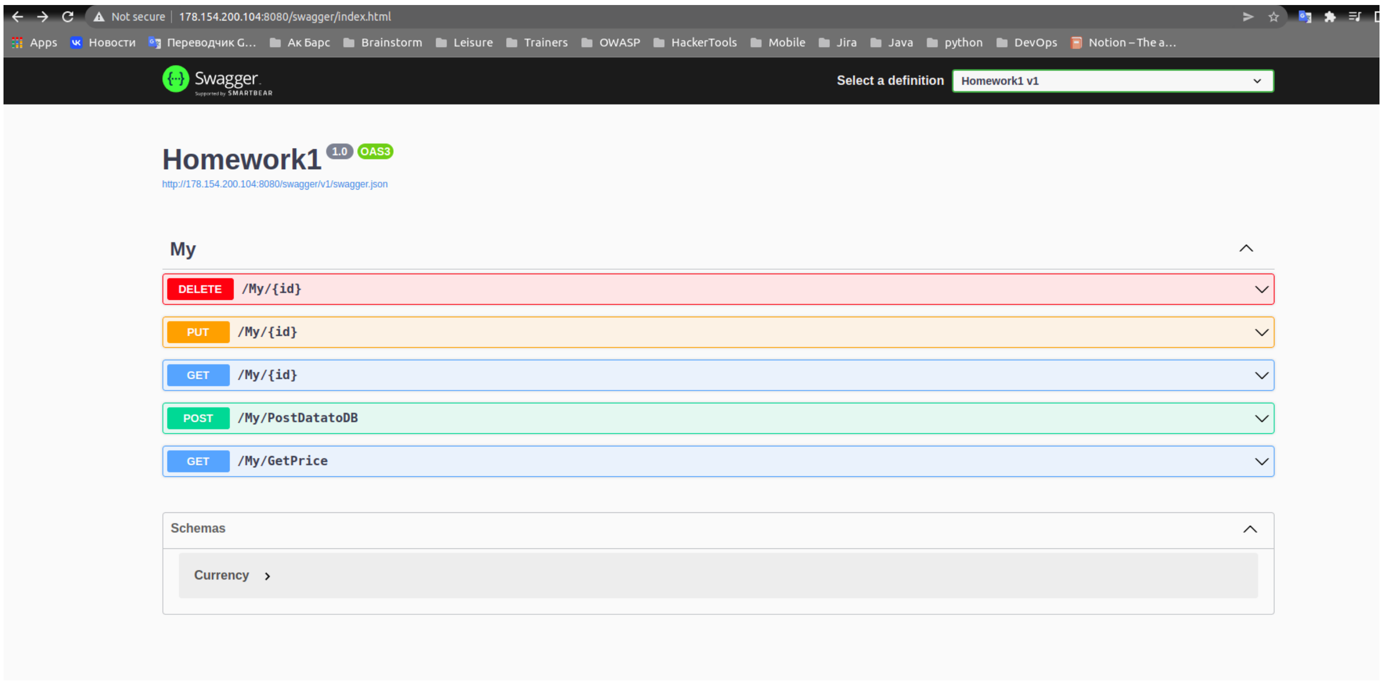
POST `/My/PostDatatoDB`

GET `/My/GetPrice`

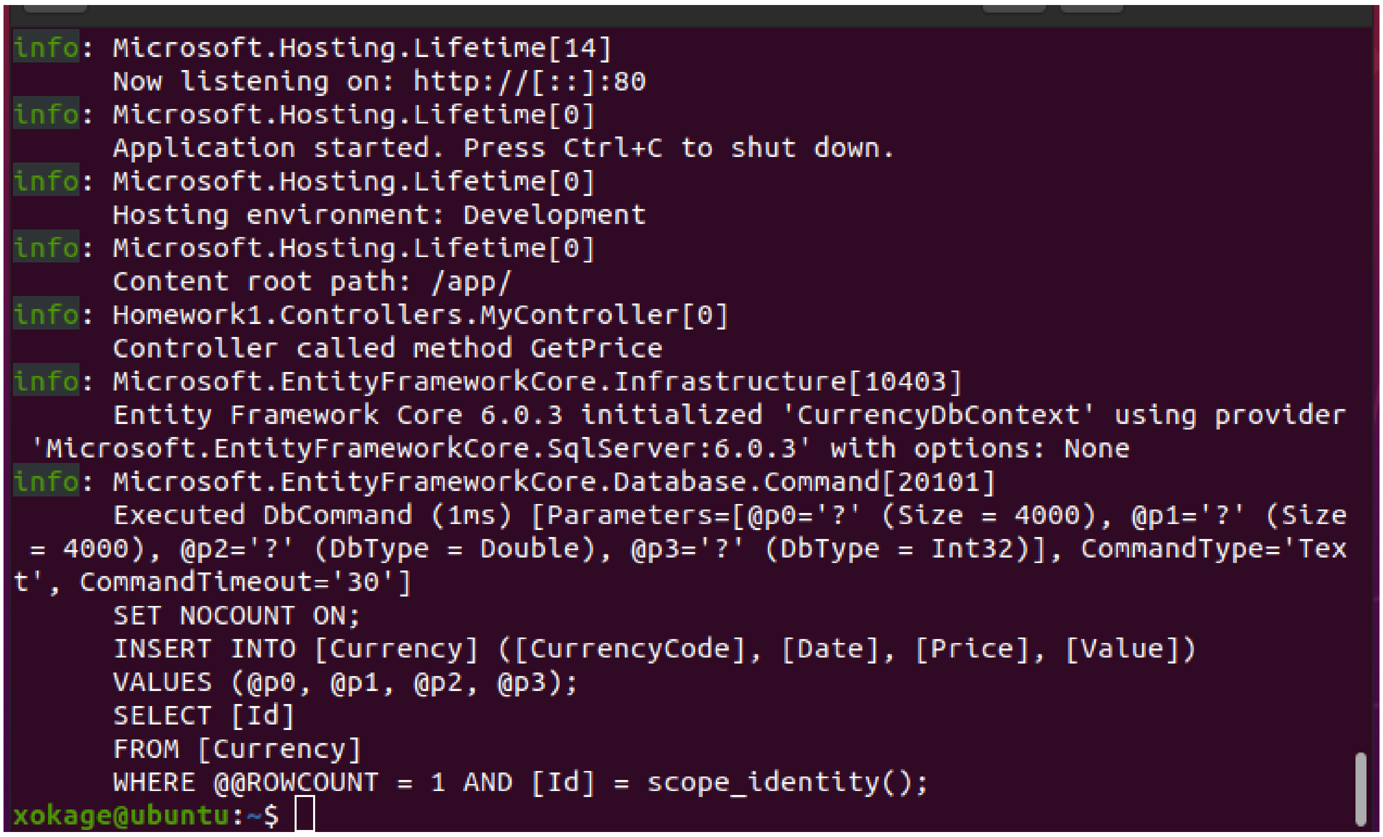
Schemas

Currency

2.

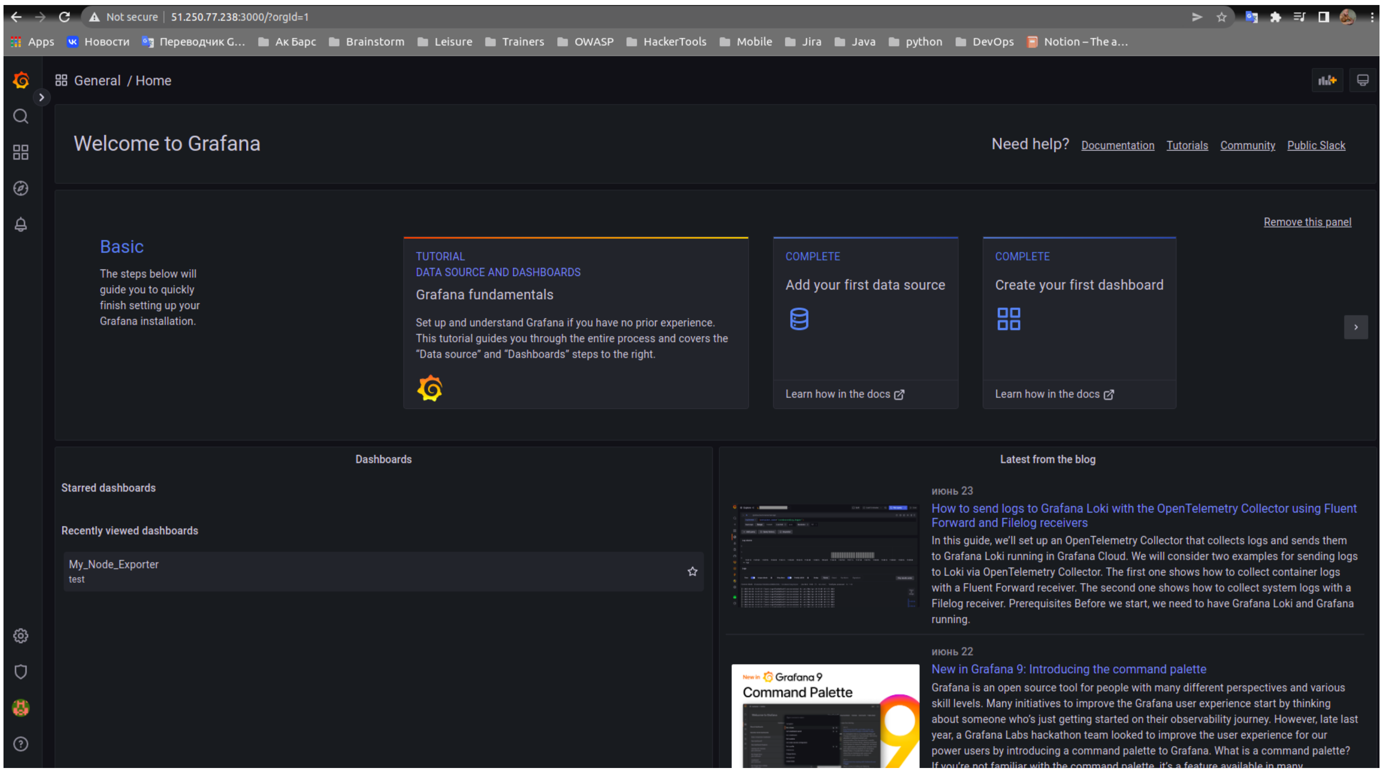


Запросы логируются внутри контейнера:



```
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://[::]:80
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /app/
info: Homework1.Controllers.MyController[0]
      Controller called method GetPrice
info: Microsoft.EntityFrameworkCore.Infrastructure[10403]
      Entity Framework Core 6.0.3 initialized 'CurrencyDbContext' using provider
 'Microsoft.EntityFrameworkCore.SqlServer:6.0.3' with options: None
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (1ms) [Parameters=[@p0='?' (Size = 4000), @p1='?' (Size
 = 4000), @p2='?' (DbType = Double), @p3='?' (DbType = Int32)], CommandType='Tex
t', CommandTimeout='30']
      SET NOCOUNT ON;
      INSERT INTO [Currency] ([CurrencyCode], [Date], [Price], [Value])
      VALUES (@p0, @p1, @p2, @p3);
      SELECT [Id]
      FROM [Currency]
      WHERE @@ROWCOUNT = 1 AND [Id] = scope_identity();
xokage@ubuntu:~$
```

Далее с прошлого дз у меня развернута графана на тачке http://51.250.77.238:3000 (http://51.250.77.238:3000)


Not secure | 51.250.77.238:3000/?orgId=1
Apps Новости Переводчик G... Ак Барс Brainstorm Leisure Trainers OWASP HackerTools Mobile Jira Java python DevOps Notion – The a...
General / Home

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.

COMPLETE
Add your first data source
Learn how in the docs

COMPLETE
Create your first dashboard
Learn how in the docs

Dashboards

Starred dashboards

Recently viewed dashboards

My_Node_Exporter
test

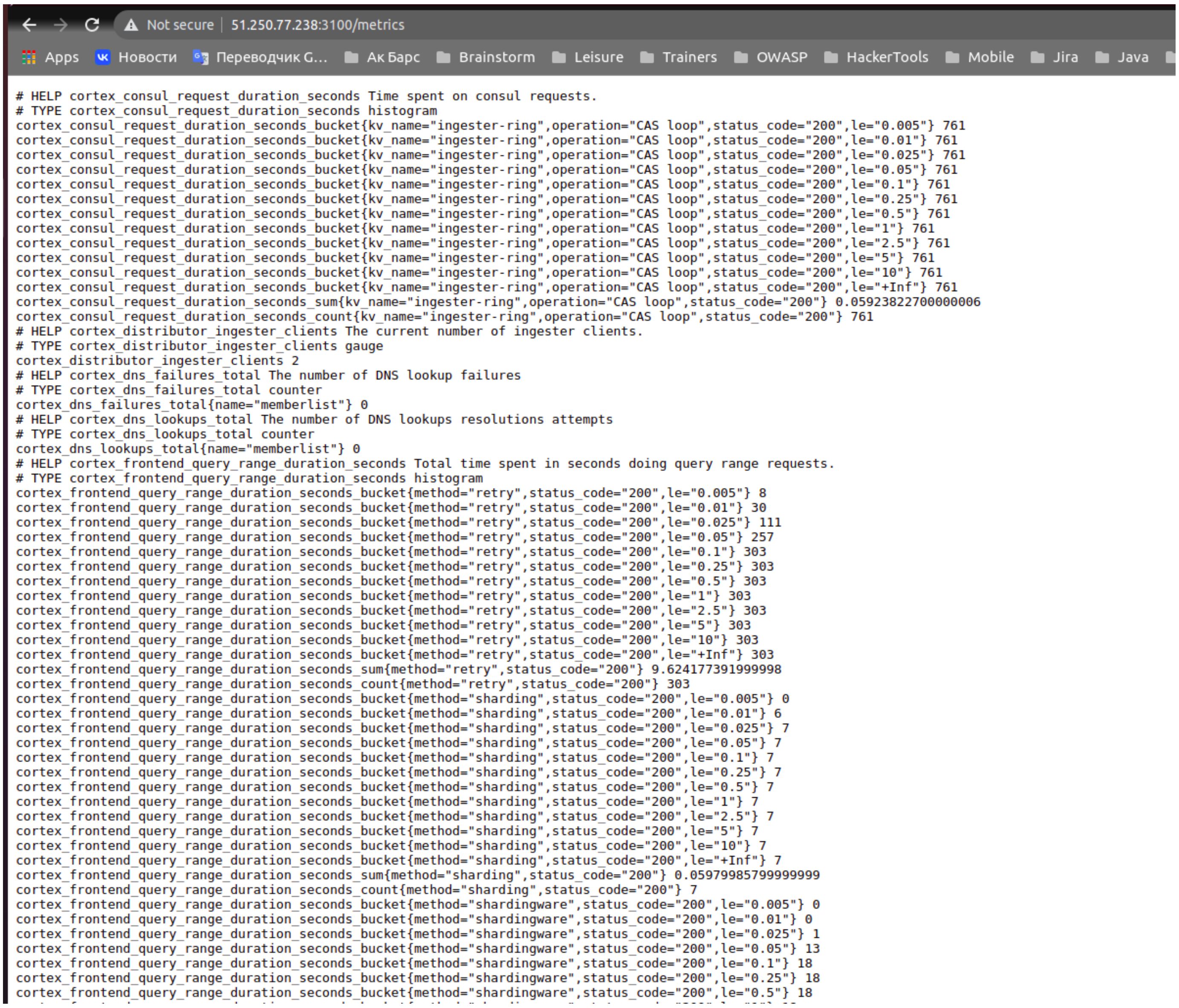
Latest from the blog

июнь 23
How to send logs to Grafana Loki with the OpenTelemetry Collector using Fluent Forward and Filelog receivers
In this guide, we'll set up an OpenTelemetry Collector that collects logs and sends them to Grafana Loki running in Grafana Cloud. We will consider two examples for sending logs to Loki via OpenTelemetry Collector. The first one shows how to collect container logs with a Fluent Forward receiver. The second one shows how to collect system logs with a Filelog receiver. Prerequisites Before we start, we need to have Grafana Loki and Grafana running.

июнь 22
New in Grafana 9: Introducing the command palette
Grafana is an open source tool for people with many different perspectives and various skill levels. Many initiatives to improve the Grafana user experience start by thinking about someone who's just getting started on their observability journey. However, late last year, a Grafana Labs hackathon team looked to improve the user experience for our power users by introducing a command palette to Grafana. What is a command palette? If you're not familiar with the command palette, it's a feature available in many

Ставлю на тачку с графаной Loki через контейнер:

```
wget https://raw.githubusercontent.com/grafana/loki/v2.5.0/cmd/loki/loki-local-
docker run --name loki -d -v $(pwd):/mnt/config -p 3100:3100 grafana/loki:2.5.0
```



```
# HELP cortex_consul_request_duration_seconds Time spent on consul requests.
# TYPE cortex_consul_request_duration_seconds histogram
cortex_consul_request_duration_seconds_bucket{kv_name="ingester-ring",operation="CAS loop",status_code="200",le="0.005"} 761
cortex_consul_request_duration_seconds_bucket{kv_name="ingester-ring",operation="CAS loop",status_code="200",le="0.01"} 761
cortex_consul_request_duration_seconds_bucket{kv_name="ingester-ring",operation="CAS loop",status_code="200",le="0.025"} 761
cortex_consul_request_duration_seconds_bucket{kv_name="ingester-ring",operation="CAS loop",status_code="200",le="0.05"} 761
cortex_consul_request_duration_seconds_bucket{kv_name="ingester-ring",operation="CAS loop",status_code="200",le="0.1"} 761
cortex_consul_request_duration_seconds_bucket{kv_name="ingester-ring",operation="CAS loop",status_code="200",le="0.25"} 761
cortex_consul_request_duration_seconds_bucket{kv_name="ingester-ring",operation="CAS loop",status_code="200",le="0.5"} 761
cortex_consul_request_duration_seconds_bucket{kv_name="ingester-ring",operation="CAS loop",status_code="200",le="1"} 761
cortex_consul_request_duration_seconds_bucket{kv_name="ingester-ring",operation="CAS loop",status_code="200",le="2.5"} 761
cortex_consul_request_duration_seconds_bucket{kv_name="ingester-ring",operation="CAS loop",status_code="200",le="5"} 761
cortex_consul_request_duration_seconds_bucket{kv_name="ingester-ring",operation="CAS loop",status_code="200",le="10"} 761
cortex_consul_request_duration_seconds_bucket{kv_name="ingester-ring",operation="CAS loop",status_code="200",le="+Inf"} 761
cortex_consul_request_duration_seconds_sum{kv_name="ingester-ring",operation="CAS loop",status_code="200"} 0.05923822700000006
cortex_consul_request_duration_seconds_count{kv_name="ingester-ring",operation="CAS loop",status_code="200"} 761
# HELP cortex_distributor_ingester_clients The current number of ingester clients.
# TYPE cortex_distributor_ingester_clients gauge
cortex_distributor_ingester_clients 2
# HELP cortex_dns_failures_total The number of DNS lookup failures
# TYPE cortex_dns_failures_total counter
cortex_dns_failures_total{name="memberlist"} 0
# HELP cortex_dns_lookups_total The number of DNS lookups resolutions attempts
# TYPE cortex_dns_lookups_total counter
cortex_dns_lookups_total{name="memberlist"} 0
# HELP cortex_frontend_query_range_duration_seconds Total time spent in seconds doing query range requests.
# TYPE cortex_frontend_query_range_duration_seconds histogram
cortex_frontend_query_range_duration_seconds_bucket{method="retry",status_code="200",le="0.005"} 8
cortex_frontend_query_range_duration_seconds_bucket{method="retry",status_code="200",le="0.01"} 30
cortex_frontend_query_range_duration_seconds_bucket{method="retry",status_code="200",le="0.025"} 111
cortex_frontend_query_range_duration_seconds_bucket{method="retry",status_code="200",le="0.05"} 257
cortex_frontend_query_range_duration_seconds_bucket{method="retry",status_code="200",le="0.1"} 303
cortex_frontend_query_range_duration_seconds_bucket{method="retry",status_code="200",le="0.25"} 303
cortex_frontend_query_range_duration_seconds_bucket{method="retry",status_code="200",le="0.5"} 303
cortex_frontend_query_range_duration_seconds_bucket{method="retry",status_code="200",le="1"} 303
cortex_frontend_query_range_duration_seconds_bucket{method="retry",status_code="200",le="2.5"} 303
cortex_frontend_query_range_duration_seconds_bucket{method="retry",status_code="200",le="5"} 303
cortex_frontend_query_range_duration_seconds_bucket{method="retry",status_code="200",le="10"} 303
cortex_frontend_query_range_duration_seconds_bucket{method="retry",status_code="200",le="+Inf"} 303
cortex_frontend_query_range_duration_seconds_sum{method="retry",status_code="200"} 9.624177391999998
cortex_frontend_query_range_duration_seconds_count{method="retry",status_code="200"} 303
cortex_frontend_query_range_duration_seconds_bucket{method="sharding",status_code="200",le="0.005"} 0
cortex_frontend_query_range_duration_seconds_bucket{method="sharding",status_code="200",le="0.01"} 6
cortex_frontend_query_range_duration_seconds_bucket{method="sharding",status_code="200",le="0.025"} 7
cortex_frontend_query_range_duration_seconds_bucket{method="sharding",status_code="200",le="0.05"} 7
cortex_frontend_query_range_duration_seconds_bucket{method="sharding",status_code="200",le="0.1"} 7
cortex_frontend_query_range_duration_seconds_bucket{method="sharding",status_code="200",le="0.25"} 7
cortex_frontend_query_range_duration_seconds_bucket{method="sharding",status_code="200",le="0.5"} 7
cortex_frontend_query_range_duration_seconds_bucket{method="sharding",status_code="200",le="1"} 7
cortex_frontend_query_range_duration_seconds_bucket{method="sharding",status_code="200",le="2.5"} 7
cortex_frontend_query_range_duration_seconds_bucket{method="sharding",status_code="200",le="5"} 7
cortex_frontend_query_range_duration_seconds_bucket{method="sharding",status_code="200",le="10"} 7
cortex_frontend_query_range_duration_seconds_bucket{method="sharding",status_code="200",le="+Inf"} 7
cortex_frontend_query_range_duration_seconds_sum{method="sharding",status_code="200"} 0.05979985799999999
cortex_frontend_query_range_duration_seconds_count{method="sharding",status_code="200"} 7
cortex_frontend_query_range_duration_seconds_bucket{method="shardingware",status_code="200",le="0.005"} 0
cortex_frontend_query_range_duration_seconds_bucket{method="shardingware",status_code="200",le="0.01"} 0
cortex_frontend_query_range_duration_seconds_bucket{method="shardingware",status_code="200",le="0.025"} 1
cortex_frontend_query_range_duration_seconds_bucket{method="shardingware",status_code="200",le="0.05"} 13
cortex_frontend_query_range_duration_seconds_bucket{method="shardingware",status_code="200",le="0.1"} 18
cortex_frontend_query_range_duration_seconds_bucket{method="shardingware",status_code="200",le="0.25"} 18
cortex_frontend_query_range_duration_seconds_bucket{method="shardingware",status_code="200",le="0.5"} 18
```

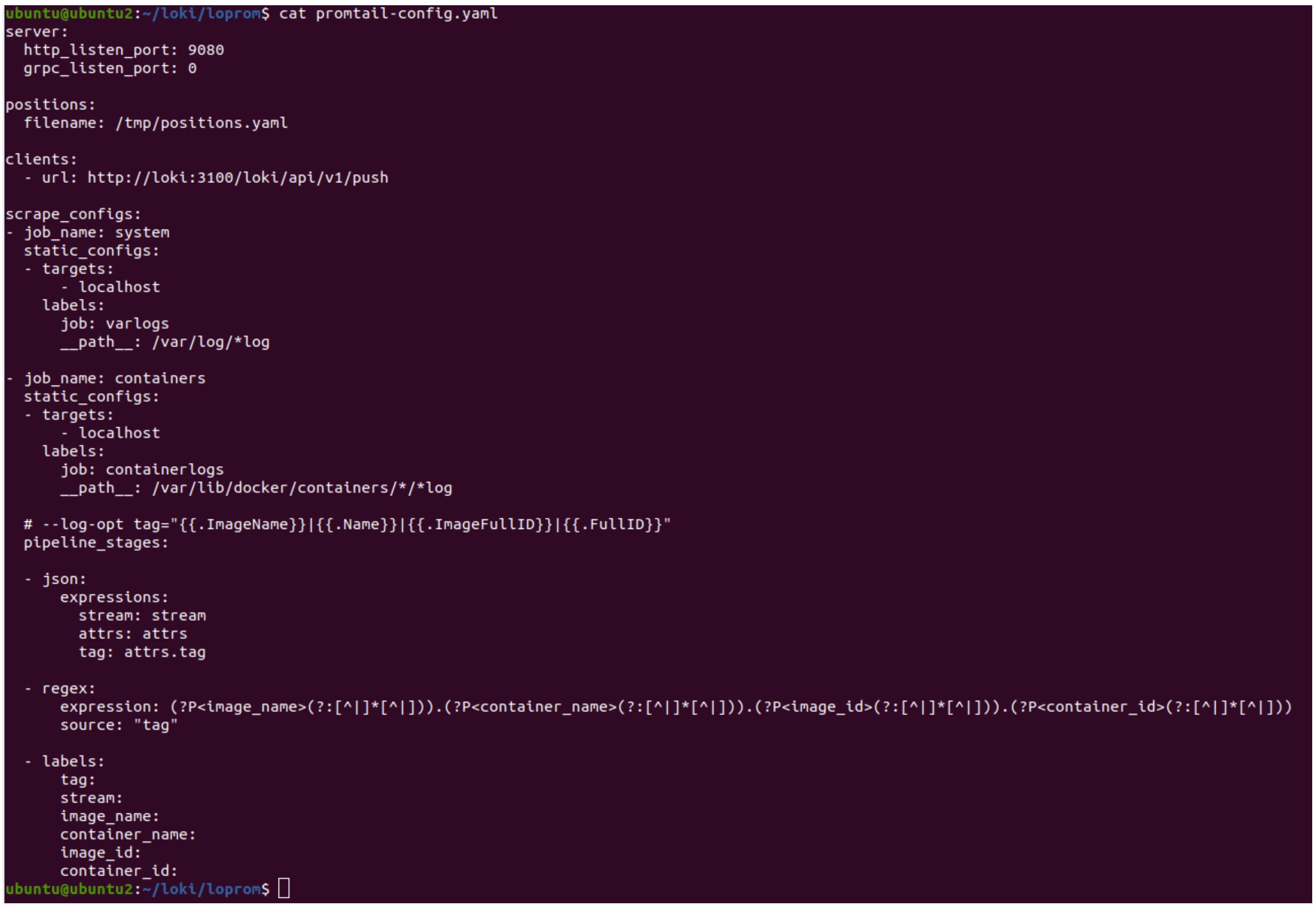
Все работает

Далее нужно поставить промтейл:

```
wget https://raw.githubusercontent.com/grafana/loki/v2.5.0/clients/cmd/promtail

docker run --name promtail --rm -d -v $(pwd):/mnt/config -v /var/log:/var/log -
```

Для того чтобы собирались логи с контейнеров, меняем конфиг промтейла:



```
ubuntu@ubuntu2:~/loki/loprom$ cat promtail-config.yaml
server:
  http_listen_port: 9080
  grpc_listen_port: 0

positions:
  filename: /tmp/positions.yaml

clients:
  - url: http://loki:3100/loki/api/v1/push

scrape_configs:
- job_name: system
  static_configs:
  - targets:
      - localhost
    labels:
      job: varlogs
      __path__: /var/log/*log

- job_name: containers
  static_configs:
  - targets:
      - localhost
    labels:
      job: containerlogs
      __path__: /var/lib/docker/containers/*/*log

  # --log-opt tag="{{.ImageName}}|{{.Name}}|{{.ImageFullID}}|{{.FullID}}"
  pipeline_stages:

  - json:
      expressions:
        stream: stream
        attrs: attrs
        tag: attrs.tag

  - regex:
      expression: (?P<image_name>(?:[^|]*[^|])).(?P<container_name>(?:[^|]*[^|])).(?P<image_id>(?:[^|]*[^|])).(?P<container_id>(?:[^|]*[^|]))
      source: "tag"

  - labels:
      tag:
      stream:
      image_name:
      container_name:
      image_id:
      container_id:
ubuntu@ubuntu2:~/loki/loprom$
```

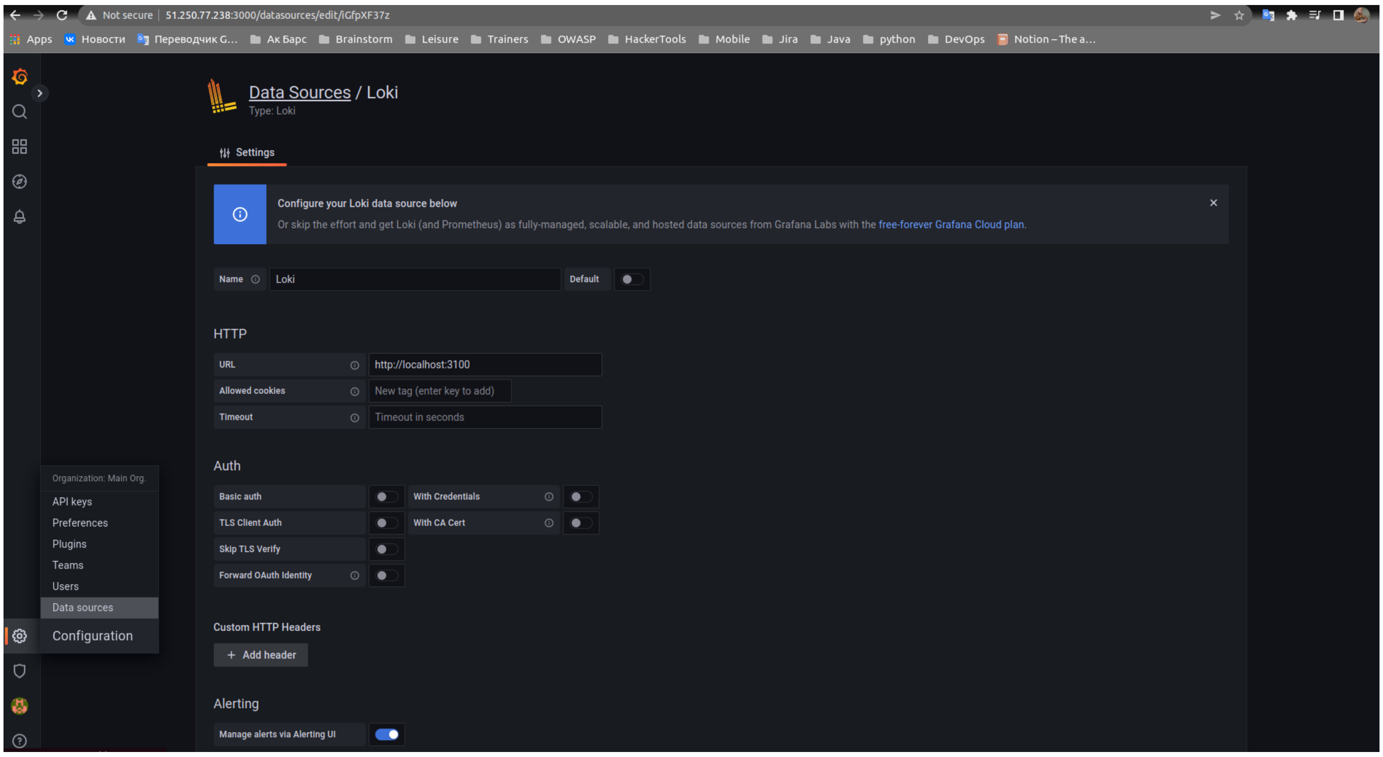
Здесь используется алиас loki, поскольку запуск локи и промтейла на одной тачке.

На другой машине качаем конфиг промтейла, меняем его и запускаем в докере:

```
wget https://raw.githubusercontent.com/grafana/loki/v2.5.0/clients/cmd/promtail

docker run --name promtail --rm -d -v $(pwd):/mnt/config -v /var/log:/var/log -
```

В графане настраиваем data source from Loki:


Not secure | 51.250.77.238:3000/datasources/edit/iGfpXF37z
Apps Новости Переводчик G... Ак Барс Brainstorm Leisure Trainers OWASP HackerTools Mobile Jira Java python DevOps Notion – The a...
Organization: Main Org.
API keys
Preferences
Plugins
Teams
Users
Data sources
Configuration
Settings
Add header

Data Sources / Loki
Type: Loki

Configure your Loki data source below
Or skip the effort and get Loki (and Prometheus) as fully-managed, scalable, and hosted data sources from Grafana Labs with the free-forever Grafana Cloud plan.

Name: Loki
Default: off

HTTP
URL: http://localhost:3100
Allowed cookies: New tag (enter key to add)
Timeout: Timeout in seconds

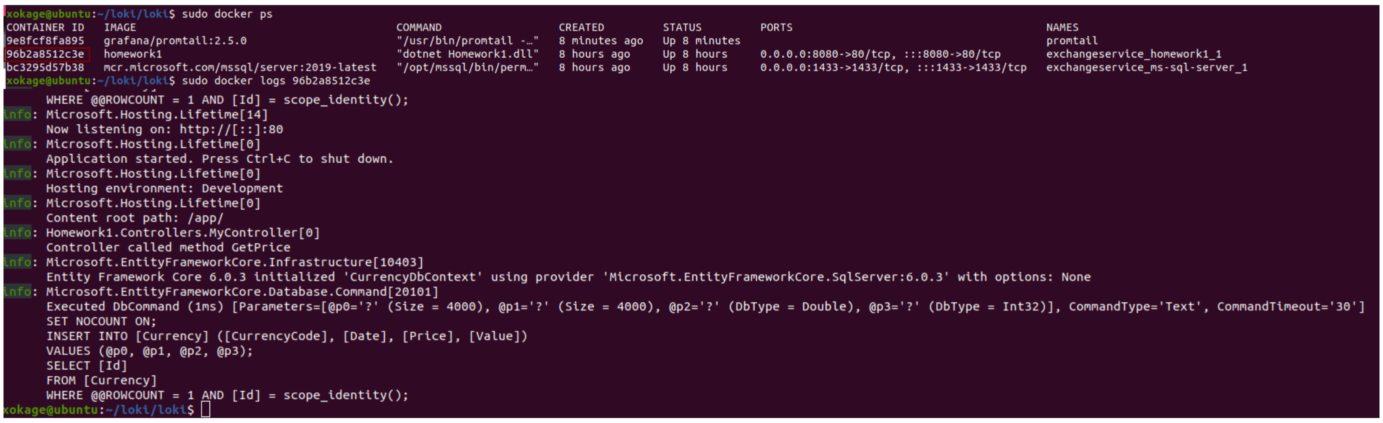
Auth
Basic auth: off
With Credentials: off
TLS Client Auth: off
With CA Cert: off
Skip TLS Verify: off
Forward OAuth Identity: off

Custom HTTP Headers

Alerting
Manage alerts via Alerting UI: on

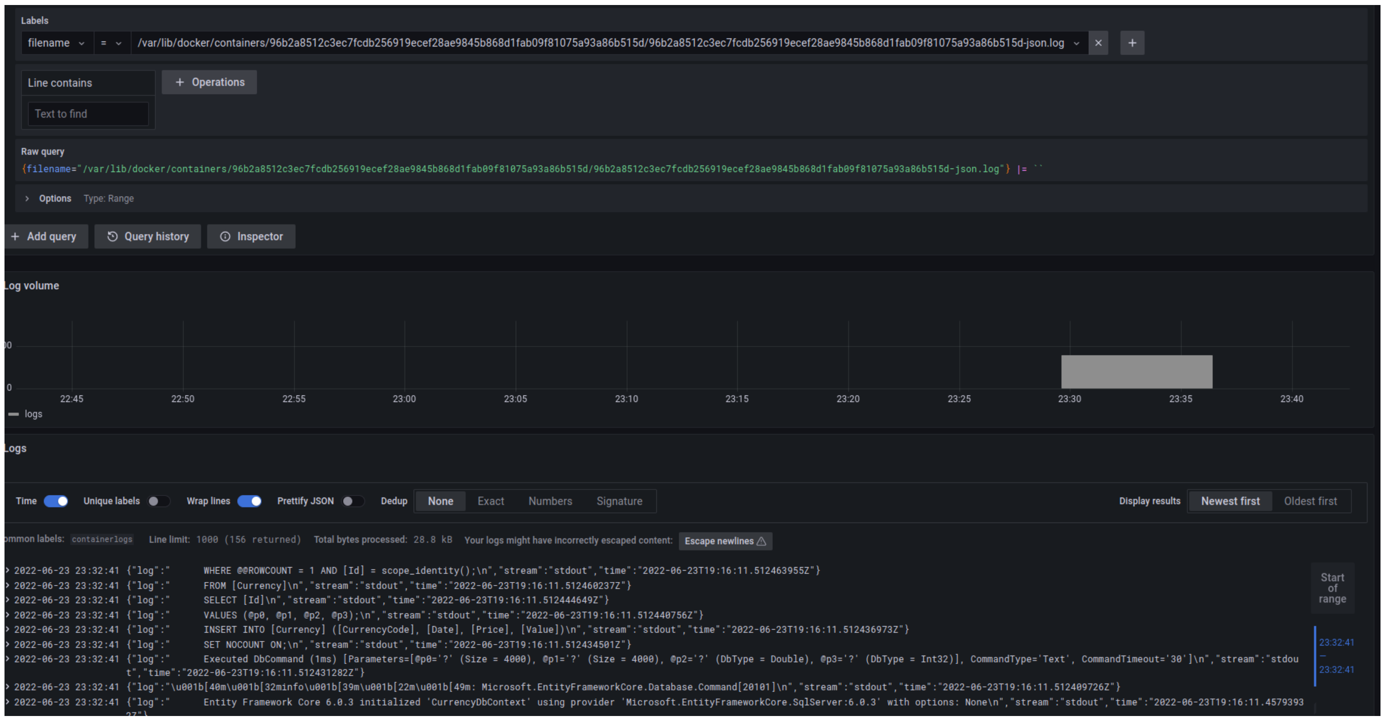
Далее можно искать логи:

Первая тачка



```
xokage@ubuntu:~/loki/loki$ sudo docker ps
CONTAINER ID   IMAGE                                        COMMAND                  CREATED          STATUS          PORTS                                          NAMES
9e8fcf8fa895   grafana/promtail:2.5.0                       "/usr/bin/promtail -…"   8 minutes ago    Up 8 minutes                                                   promtail
96b2a8512c3e   homework1                                    "dotnet Homework1.dll"   8 hours ago      Up 8 hours      0.0.0.0:8080->80/tcp, :::8080->80/tcp          exchangeservice_homework1_1
bc3295d57b38   mcr.microsoft.com/mssql/server:2019-latest   "/opt/mssql/bin/perm…"   8 hours ago      Up 8 hours      0.0.0.0:1433->1433/tcp, :::1433->1433/tcp      exchangeservice_ms-sql-server_1
xokage@ubuntu:~/loki/loki$ sudo docker logs 96b2a8512c3e
      WHERE @@ROWCOUNT = 1 AND [Id] = scope_identity();
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://[::]:80
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /app/
info: Homework1.Controllers.MyController[0]
      Controller called method GetPrice
info: Microsoft.EntityFrameworkCore.Infrastructure[10403]
      Entity Framework Core 6.0.3 initialized 'CurrencyDbContext' using provider 'Microsoft.EntityFrameworkCore.SqlServer:6.0.3' with options: None
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (1ms) [Parameters=[@p0='?' (Size = 4000), @p1='?' (Size = 4000), @p2='?' (DbType = Double), @p3='?' (DbType = Int32)], CommandType='Text', CommandTimeout='30']
      SET NOCOUNT ON;
      INSERT INTO [Currency] ([CurrencyCode], [Date], [Price], [Value])
      VALUES (@p0, @p1, @p2, @p3);
      SELECT [Id]
      FROM [Currency]
      WHERE @@ROWCOUNT = 1 AND [Id] = scope_identity();
xokage@ubuntu:~/loki/loki$ 
```

Последние логи выглядят так, в графане:



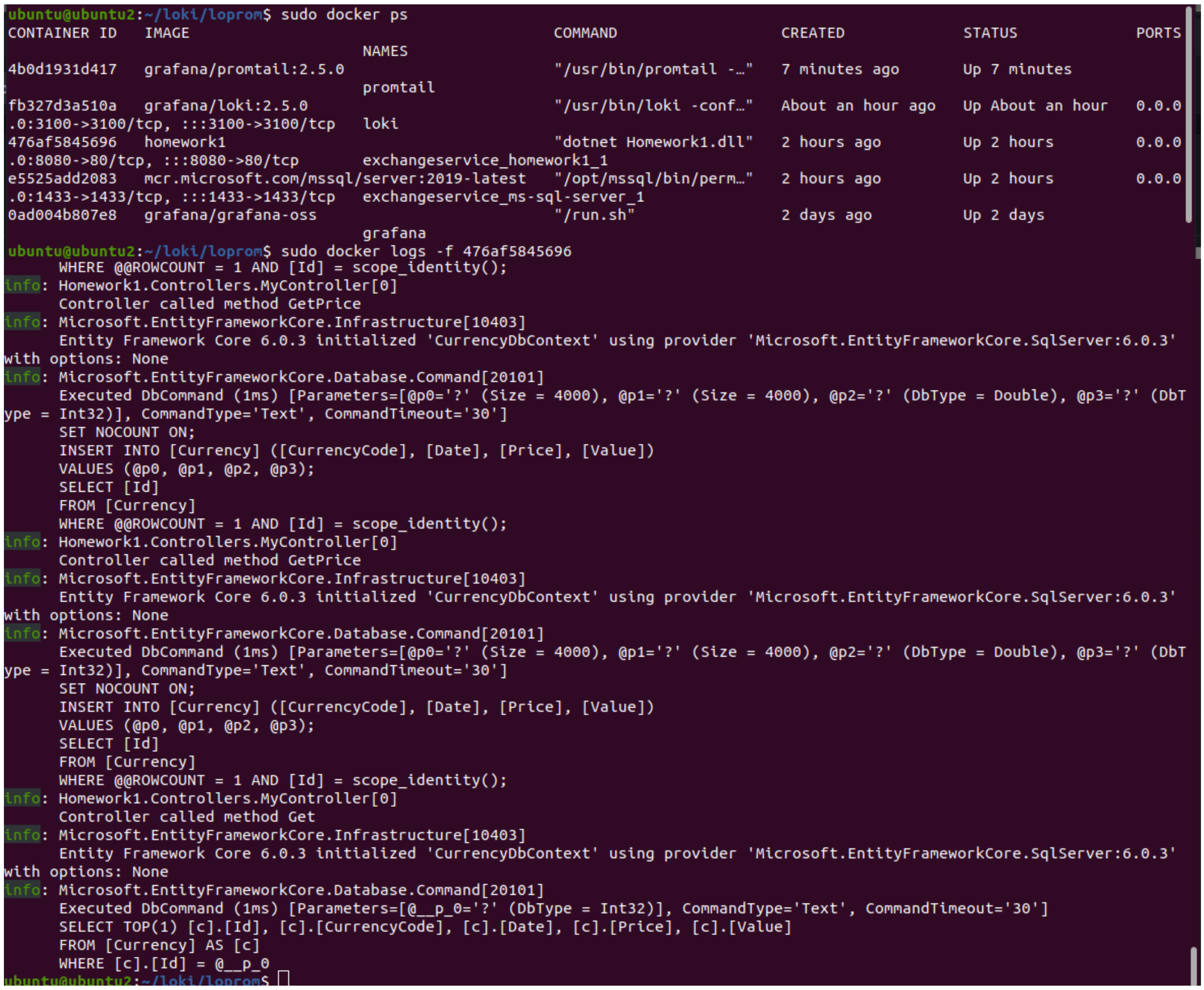
```
Raw query
{filename="/var/lib/docker/containers/96b2a8512c3ec7fcdb256919ecef28ae9845b868d1fab09f81075a93a86b515d/96b2a8512c3ec7fcdb256919ecef28ae9845b868d1fab09f81075a93a86b515d-json.log"} |= ``
Options  Type: Range

ommon labels: containerlogs   Line limit: 1000 (156 returned)   Total bytes processed: 28.8 kB   Your logs might have incorrectly escaped content:

> 2022-06-23 23:32:41 {"log":"      WHERE @@ROWCOUNT = 1 AND [Id] = scope_identity();\n","stream":"stdout","time":"2022-06-23T19:16:11.512463955Z"}
> 2022-06-23 23:32:41 {"log":"      FROM [Currency]\n","stream":"stdout","time":"2022-06-23T19:16:11.512460237Z"}
> 2022-06-23 23:32:41 {"log":"      SELECT [Id]\n","stream":"stdout","time":"2022-06-23T19:16:11.512444649Z"}
> 2022-06-23 23:32:41 {"log":"      VALUES (@p0, @p1, @p2, @p3);\n","stream":"stdout","time":"2022-06-23T19:16:11.512440756Z"}
> 2022-06-23 23:32:41 {"log":"      INSERT INTO [Currency] ([CurrencyCode], [Date], [Price], [Value])\n","stream":"stdout","time":"2022-06-23T19:16:11.512436973Z"}
> 2022-06-23 23:32:41 {"log":"      SET NOCOUNT ON;\n","stream":"stdout","time":"2022-06-23T19:16:11.512434501Z"}
> 2022-06-23 23:32:41 {"log":"      Executed DbCommand (1ms) [Parameters=[@p0='?' (Size = 4000), @p1='?' (Size = 4000), @p2='?' (DbType = Double), @p3='?' (DbType = Int32)], CommandType='Text', CommandTimeout='30']\n","stream":"stdou
  t","time":"2022-06-23T19:16:11.512431282Z"}
> 2022-06-23 23:32:41 {"log":"\u001b[40m\u001b[32minfo\u001b[39m\u001b[22m\u001b[49m: Microsoft.EntityFrameworkCore.Database.Command[20101]\n","stream":"stdout","time":"2022-06-23T19:16:11.512409726Z"}
> 2022-06-23 23:32:41 {"log":"      Entity Framework Core 6.0.3 initialized 'CurrencyDbContext' using provider 'Microsoft.EntityFrameworkCore.SqlServer:6.0.3' with options: None\n","stream":"stdout","time":"2022-06-23T19:16:11.4579393
```

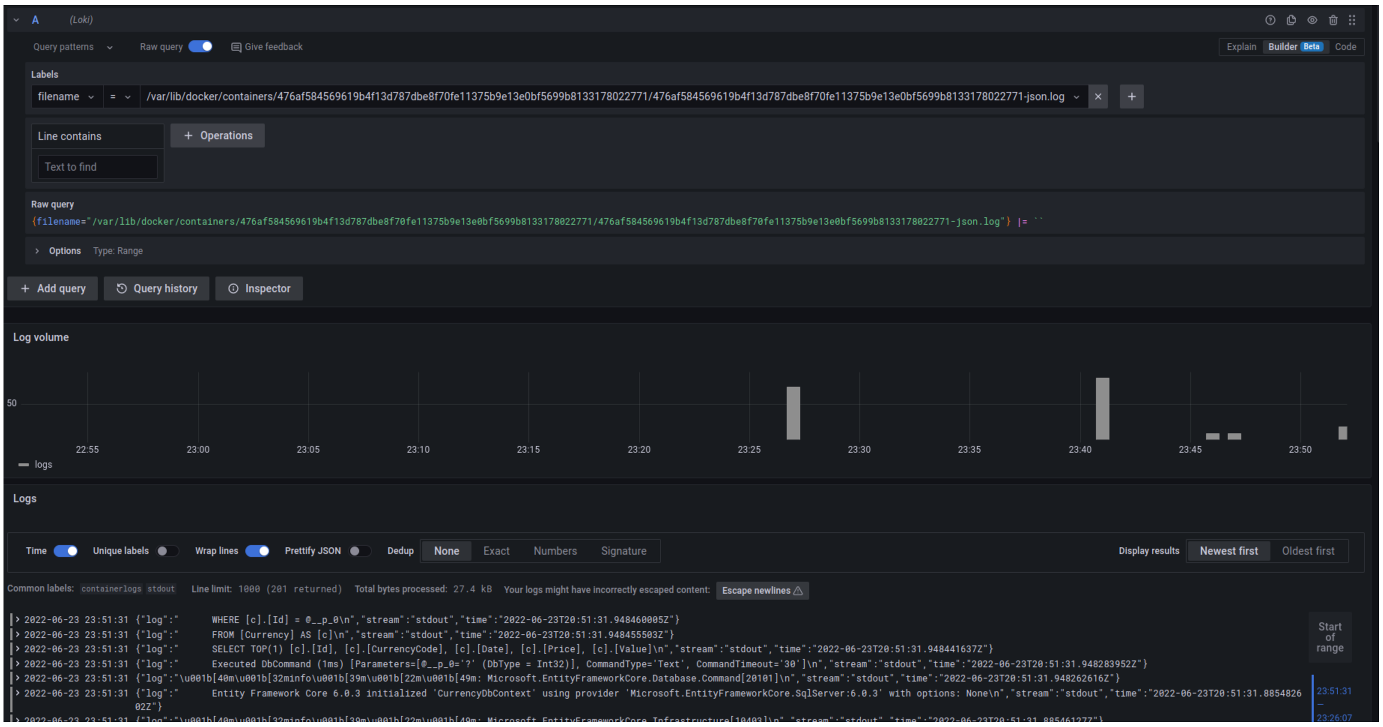
Выбрал в лейбле filename тот самый контейнер, вижу те же логи, что последние через docker logs

Аналогичная история со второй тачкой:



```
ubuntu@ubuntu2:~/loki/loprom$ sudo docker ps
CONTAINER ID   IMAGE                                        COMMAND                  CREATED             STATUS             PORTS
                                              NAMES
4b0d1931d417   grafana/promtail:2.5.0                       "/usr/bin/promtail -…"   7 minutes ago       Up 7 minutes
                                              promtail
fb327d3a510a   grafana/loki:2.5.0                           "/usr/bin/loki -conf…"   About an hour ago   Up About an hour   0.0.0
.0:3100->3100/tcp, :::3100->3100/tcp    loki
476af5845696   homework1                                    "dotnet Homework1.dll"   2 hours ago         Up 2 hours         0.0.0
.0:8080->80/tcp, :::8080->80/tcp        exchangeservice_homework1_1
e5525add2083   mcr.microsoft.com/mssql/server:2019-latest   "/opt/mssql/bin/perm…"   2 hours ago         Up 2 hours         0.0.0
.0:1433->1433/tcp, :::1433->1433/tcp    exchangeservice_ms-sql-server_1
0ad004b807e8   grafana/grafana-oss                          "/run.sh"                2 days ago          Up 2 days
                                              grafana
ubuntu@ubuntu2:~/loki/loprom$ sudo docker logs -f 476af5845696
      WHERE @@ROWCOUNT = 1 AND [Id] = scope_identity();
info: Homework1.Controllers.MyController[0]
      Controller called method GetPrice
info: Microsoft.EntityFrameworkCore.Infrastructure[10403]
      Entity Framework Core 6.0.3 initialized 'CurrencyDbContext' using provider 'Microsoft.EntityFrameworkCore.SqlServer:6.0.3'
with options: None
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (1ms) [Parameters=[@p0='?' (Size = 4000), @p1='?' (Size = 4000), @p2='?' (DbType = Double), @p3='?' (DbT
ype = Int32)], CommandType='Text', CommandTimeout='30']
      SET NOCOUNT ON;
      INSERT INTO [Currency] ([CurrencyCode], [Date], [Price], [Value])
      VALUES (@p0, @p1, @p2, @p3);
      SELECT [Id]
      FROM [Currency]
      WHERE @@ROWCOUNT = 1 AND [Id] = scope_identity();
info: Homework1.Controllers.MyController[0]
      Controller called method GetPrice
info: Microsoft.EntityFrameworkCore.Infrastructure[10403]
      Entity Framework Core 6.0.3 initialized 'CurrencyDbContext' using provider 'Microsoft.EntityFrameworkCore.SqlServer:6.0.3'
with options: None
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (1ms) [Parameters=[@p0='?' (Size = 4000), @p1='?' (Size = 4000), @p2='?' (DbType = Double), @p3='?' (DbT
ype = Int32)], CommandType='Text', CommandTimeout='30']
      SET NOCOUNT ON;
      INSERT INTO [Currency] ([CurrencyCode], [Date], [Price], [Value])
      VALUES (@p0, @p1, @p2, @p3);
      SELECT [Id]
      FROM [Currency]
      WHERE @@ROWCOUNT = 1 AND [Id] = scope_identity();
info: Homework1.Controllers.MyController[0]
      Controller called method Get
info: Microsoft.EntityFrameworkCore.Infrastructure[10403]
      Entity Framework Core 6.0.3 initialized 'CurrencyDbContext' using provider 'Microsoft.EntityFrameworkCore.SqlServer:6.0.3'
with options: None
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (1ms) [Parameters=[@__p_0='?' (DbType = Int32)], CommandType='Text', CommandTimeout='30']
      SELECT TOP(1) [c].[Id], [c].[CurrencyCode], [c].[Date], [c].[Price], [c].[Value]
      FROM [Currency] AS [c]
      WHERE [c].[Id] = @__p_0
ubuntu@ubuntu2:~/loki/loprom$
```

Последние логи выглядят так, в графане:



Labels: filename = /var/lib/docker/containers/476af584569619b4f13d787dbe8f70fe11375b9e13e0bf5699b8133178022771/476af584569619b4f13d787dbe8f70fe11375b9e13e0bf5699b8133178022771-json.log

Raw query

```
{filename="/var/lib/docker/containers/476af584569619b4f13d787dbe8f70fe11375b9e13e0bf5699b8133178022771/476af584569619b4f13d787dbe8f70fe11375b9e13e0bf5699b8133178022771-json.log"} |= ``
```

Common labels: containerlogs stdout   Line limit: 1000 (201 returned)   Total bytes processed: 27.4 kB   Your logs might have incorrectly escaped content:

```
> 2022-06-23 23:51:31 {"log":"      WHERE [c].[Id] = @__p_0\n","stream":"stdout","time":"2022-06-23T20:51:31.948460005Z"}
> 2022-06-23 23:51:31 {"log":"      FROM [Currency] AS [c]\n","stream":"stdout","time":"2022-06-23T20:51:31.948455503Z"}
> 2022-06-23 23:51:31 {"log":"      SELECT TOP(1) [c].[Id], [c].[CurrencyCode], [c].[Date], [c].[Price], [c].[Value]\n","stream":"stdout","time":"2022-06-23T20:51:31.948441637Z"}
> 2022-06-23 23:51:31 {"log":"      Executed DbCommand (1ms) [Parameters=[@__p_0='?' (DbType = Int32)], CommandType='Text', CommandTimeout='30']\n","stream":"stdout","time":"2022-06-23T20:51:31.948283952Z"}
> 2022-06-23 23:51:31 {"log":"\u001b[40m\u001b[32minfo\u001b[39m\u001b[22m\u001b[49m: Microsoft.EntityFrameworkCore.Database.Command[20101]\n","stream":"stdout","time":"2022-06-23T20:51:31.948262616Z"}
> 2022-06-23 23:51:31 {"log":"      Entity Framework Core 6.0.3 initialized 'CurrencyDbContext' using provider 'Microsoft.EntityFrameworkCore.SqlServer:6.0.3' with options: None\n","stream":"stdout","time":"2022-06-23T20:51:31.8854826
                     02Z"}
```

Выбрал в лейбле filename тот самый контейнер, вижу те же логи, что последние через docker logs