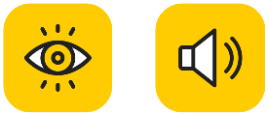
OTUS

Микросервисная архитектура на основе Kafka

otus.ru

Меня хорошо видно && слышно?



Защита проекта

Тема: Микросервисная архитектура на основе Kafka

Астраханцев Виктор

Должность: разработчик
Компания: МТС Диджитал

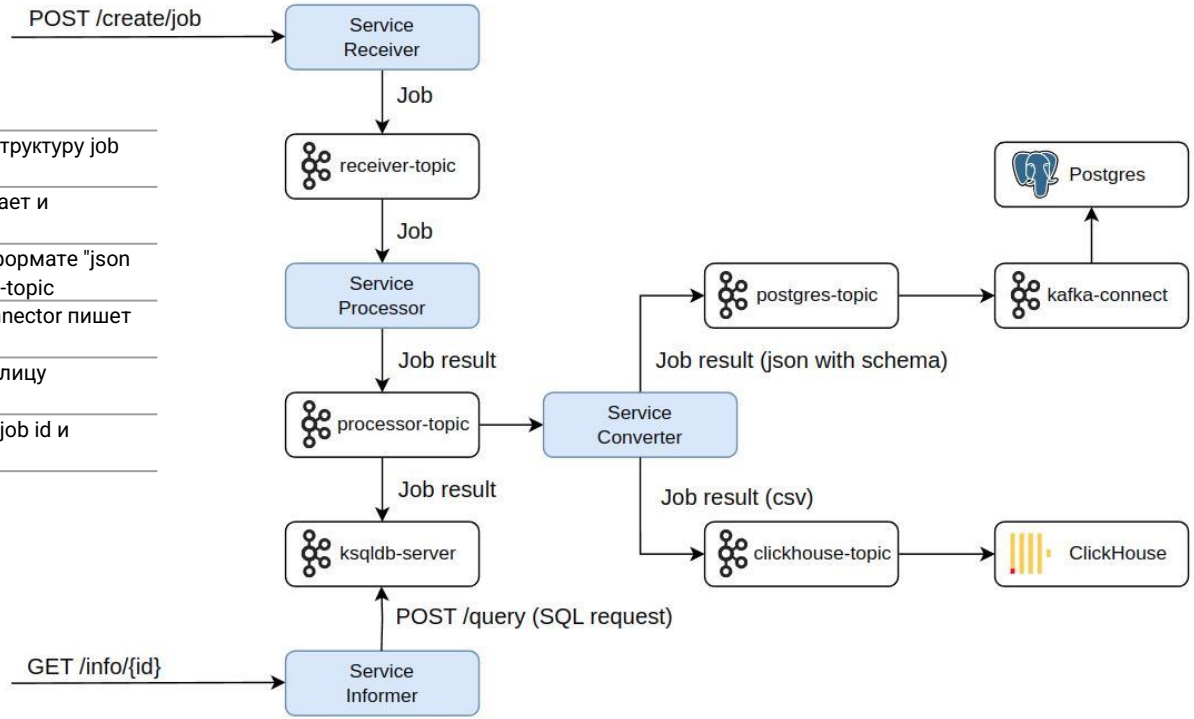
astrviktor@gmail.com
https://github.com/astrviktor

Архитектура

- Service Receiver получает запрос через REST, создает структуру job и сохраняет в receiver-topic
- Service Processor читает job из receiver-topic, обрабатывает и сохраняет job result в processor-topic
- Service Converter job result из processor-topic и пишет в формате "json with schema" в postgres-topic и формате "csv" в clickhouse-topic
- Kafka Connect читает postgres-topic и через JdbcSinkConnector пишет данные в таблицу Postgres
- Clickhouse читает clickhouse-topic и пишет данные в таблицу Clickhouse
- Service Informer через REST получает запрос статуса по job id и запрашивают информацию через REST ksqldb-server

Используемые технологии

- Kafka, Kafka Connect, ksqlDB Server
- Микросервисы на golang
- Postgres, ClickHouse
- Docker, Dockerfile, Docker-Compose
- Makefile и Bash скрипты



Service Receiver

Сервис получает запрос через REST

```
curl --request POST 'http://127.0.0.1:8001/create/job'

{"id":"b31b7446-20f7-4add-adea-157bac7fe73b","create_date":"2024-01-27T12:34:50.594423357Z","sleep":99,"data_size":1024}
```

Создает структуру Job

```
type Job struct {
 Id         string    `json:"id"`
 CreateDate time.Time `json:"create_date"`
 Sleep      int64     `json:"sleep"`
 Data       string    `json:"data"`
}
```

Записывает Job в **receiver-topic**

```
{
 "id": "b31b7446-20f7-4add-adea-157bac7fe73b",
 "create_date": "2024-01-27T12:34:50.594423357Z",
 "sleep": 99,
 "data": "…"
}
```

Service Processor

Сервис читает Job из **receiver-topic**

```
type Job struct {
 Id         string    `json:"id"`
 CreateDate time.Time `json:"create_date"`
 Sleep      int64     `json:"sleep"`
 Data       string    `json:"data"`
}
```

Делает Sleep на значение поля Sleep (~ 100 мс)

Создает структуру JobDone

```
type JobDone struct {
 Id         string    `json:"id"`
 Status     JobStatus `json:"status"`
 CreateDate time.Time `json:"create_date"`
 FinishDate time.Time `json:"finish_date"`
}
```

Записывает JobDone в **processor-topic**

```
{
 "id": "b31b7446-20f7-4add-adea-157bac7fe73b",
 "status": "finish",
 "create_date": "2024-01-27T12:34:50.594423357Z",
 "finish_date": "2024-01-27T12:34:50.704464721Z"
}
```

Service Converter

Сервис читает JobDone из **processor-topic**

```
type JobDone struct {
 Id         string    `json:"id"`
 Status     JobStatus `json:"status"`
 CreateDate time.Time `json:"create_date"`
 FinishDate time.Time `json:"finish_date"`
}
```

Создает структуру PostgresMessage с json схемой и данными

```
type PostgresMessage struct {
 Schema  SchemaType      `json:"schema"`
 Payload PostgresPayload `json:"payload"`
}
```

Создает структуру ClickhouseMessage с данными для csv

```
type ClickhouseMessage struct {
 Id         string    `json:"id"`
 Status     JobStatus `json:"status"`
 CreateDate time.Time `json:"create_date"`
}
```

Service Converter

Записывает PostgresMessage в **postgres-topic**


```
{
 "schema": {
  "type": "struct",
  "fields": [
   {
    "type": "string",
    "optional": false,
    "field": "id"
   },
   {
    "type": "string",
    "optional": false,
    "field": "status"
   },
   {
    "type": "int64",
    "optional": false,
    "field": "create_date"
   },
   {
    "type": "int64",
    "optional": false,
    "field": "finish_date"
   }
  ]
 },
 "payload": {
  "id": "b31b7446-20f7-4add-adea-157bac7fe73b",
  "status": "finish",
  "create_date": 1706358890,
  "finish_date": 1706358890
 }
}
```


Записывает 2 строчки ClickhouseMessage в **clickhouse-topic**

```
b31b7446-20f7-4add-adea-157bac7fe73b,create,2024-01-27T12:34:50.594423357Z
b31b7446-20f7-4add-adea-157bac7fe73b,finish,2024-01-27T12:34:50.704464721Z
```

Service Informer

Сервис получает запрос через REST

```
curl --request GET --url 'http://127.0.0.1:8004/info/b31b7446-20f7-4add-adea-157bac7fe73b'

{
  "id":"b31b7446-20f7-4add-adea-157bac7fe73b",
  "status":"finish",
  "create_date":"2024-01-27T12:34:50.594423357Z",
  "finish_date":"2024-01-27T12:34:50.704464721Z"
}
```

Если id не существует, status = unknown

```
curl --request GET --url 'http://127.0.0.1:8004/info/00000000-0000-0000-0000-000000000000'

{
  "id":"00000000-0000-0000-0000-000000000000",
  "status":"unknown"
}
```

Service Informer

Для получения результата сервис делает REST запрос в ksqldb-server

```
curl --request POST --url 'http://localhost:8088/query' \
  --header 'Content-Type: application/vnd.ksql.v1+json; charset=utf-8' \
  --data '{ "ksql": "SELECT * FROM jobs_stream WHERE id = ' b31b7446-20f7-4add-adea-157bac7fe73b ';",
            "streamsProperties": {} }'
```

Ответ от ksqldb-server

```
[
  {
    "header": {
      "queryId": "transient_JOBS_STREAM_7461873452658691048",
      "schema": "`ID` STRING, `STATUS` STRING, `CREATE_DATE` STRING, `FINISH_DATE` STRING"
    }
  },
  {
    "row": {
      "columns": [
        "b31b7446-20f7-4add-adea-157bac7fe73b",
        "finish",
        "2024-01-27T12:34:50.594423357Z",
        "2024-01-27T12:34:50.704464721Z"
      ]
    }
  },
  {
    "finalMessage": "Query Completed"
  }
]
```

Создание стрима для топика processor-topic

```
CREATE STREAM jobs_stream (
  id VARCHAR KEY,
  status VARCHAR,
  create_date VARCHAR,
  finish_date VARCHAR
) WITH (
  KAFKA_TOPIC = 'processor-topic',
  VALUE_FORMAT = 'JSON'
);
```

Kafka Connect и Postgres

Создание "postgres-connector"

```
{
    "name": "postgres-connector",
    "config": {
        "connector.class": "io.confluent.connect.jdbc.JdbcSinkConnector",
        "connection.url": "jdbc:postgresql://postgres:5432/postgres",
        "connection.user": "postgres",
        "connection.password": "password",
        "connection.ds.pool.size": 5,
        "topics": "postgres-topic",
        "auto.create": "true",
        "insert.mode.databaselevel": true,
        "pk.mode" : "record_value",
        "pk.fields": "id",
        "tasks.max": "1",
        "table.name.format": "jobs"
    }
}
```

Создание таблицы

```
CREATE TABLE jobs
(
    id varchar(36) PRIMARY KEY,
    status text,
    create_date bigint default (EXTRACT(epoch FROM now()))::bigint,
    finish_date bigint default (EXTRACT(epoch FROM now()))::bigint
);
```

Kafka Connect и Postgres: проверка данных

```
docker exec -ti postgres psql -U postgres
psql (16.1 (Debian 16.1-1.pgdg120+1))
Type "help" for help.

postgres=# \d
        List of relations
 Schema | Name | Type  |  Owner
--------+------+-------+----------
 public | jobs | table | postgres
(1 row)

postgres=# SELECT * FROM jobs;
                  id                  | status | create_date | finish_date
--------------------------------------+--------+-------------+-------------
 b31b7446-20f7-4add-adea-157bac7fe73b | finish |  1706358890 |  1706358890
(1 row)

postgres=# \q
```

Clickhouse Kafka Engine

В ClickHouse предусмотрена возможность потоковой заливки данных из Kafka

Настройки

```
CREATE TABLE IF NOT EXISTS jobs_kafka (
  id String,
  status String,
  timestamp String
) ENGINE = Kafka SETTINGS
            kafka_broker_list = 'kafka:9191',
            kafka_topic_list = 'clickhouse-topic',
            kafka_group_name = 'clickhouse',
            kafka_format = 'CSV',
            kafka_num_consumers = 1,
            kafka_skip_broken_messages = 10;
```

```
CREATE TABLE IF NOT EXISTS jobs (
  id String,
  status String,
  timestamp DateTime
) ENGINE = MergeTree()
ORDER BY timestamp;
```

```
CREATE MATERIALIZED VIEW IF NOT EXISTS jobs_mv
TO jobs
(
  id String,
  status String,
  timestamp DateTime
)
AS
SELECT
  id,
  status,
  parseDateTimeBestEffortOrNull(timestamp) AS timestamp
FROM jobs_kafka;
```

Clickhouse Kafka Engine: проверка данных

```
docker exec -ti clickhouse clickhouse-client

clickhouse :)
clickhouse :) SHOW TABLES FROM default;

SHOW TABLES FROM default

Query id: 45362a4b-e4be-421b-a91b-4138735508d4

┌─name─────────┐
│ jobs         │
│ jobs_kafka   │
│ jobs_mv      │
└──────────────┘

3 rows in set. Elapsed: 0.001 sec.

clickhouse :) SELECT * FROM default.jobs;

SELECT *
FROM default.jobs

Query id: cc143814-5a65-40c5-a92e-9858e0595221

┌─id─────────────────────────────────────┬─status─┬─────────timestamp─┐
│ b31b7446-20f7-4add-adea-157bac7fe73b │ create │ 2024-01-27 12:34:50 │
│ b31b7446-20f7-4add-adea-157bac7fe73b │ finish │ 2024-01-27 12:34:50 │
└────────────────────────────────────────┴────────┴─────────────────────┘
```

Запуск, остановка, нагрузка, мониторинг

Запуск через docker-compose

```
make compose-up
```

Остановка

```
make compose-down
```

Нагрузка

Нагрузка на POST /create/job создается через yandex-tank

Запускается нагрузка 25 RPS на 300 секунд

```
make yandex-tank
docker exec -it yandex-tank /bin/bash
Yandex.Tank Docker image

[tank]root@minikube: /var/loadtest # sh post.sh
```

Prometheus и Grafana

Сервисы пишут метрики для Prometheus

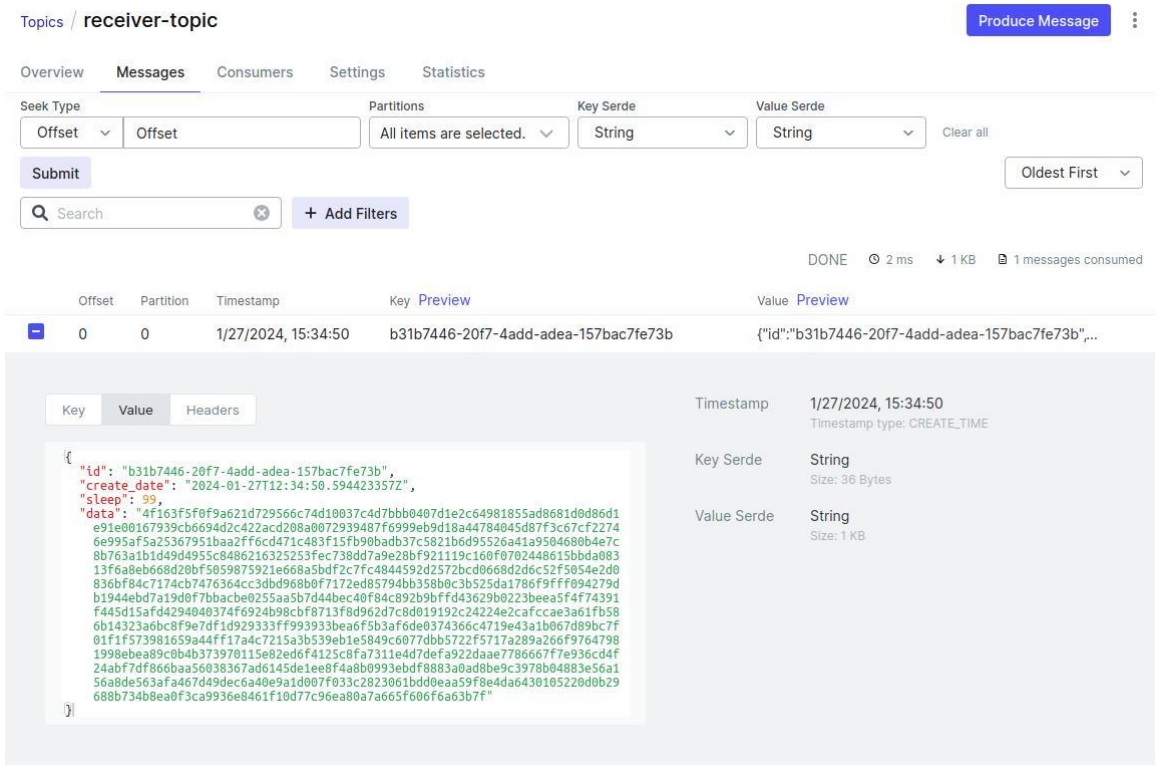
```
Prometheus: http://127.0.0.1:9090

Grafana: http://127.0.0.1:3000

логин   пароль
admin / password
```

В Grafana есть старнадртные дашборды для Kafka

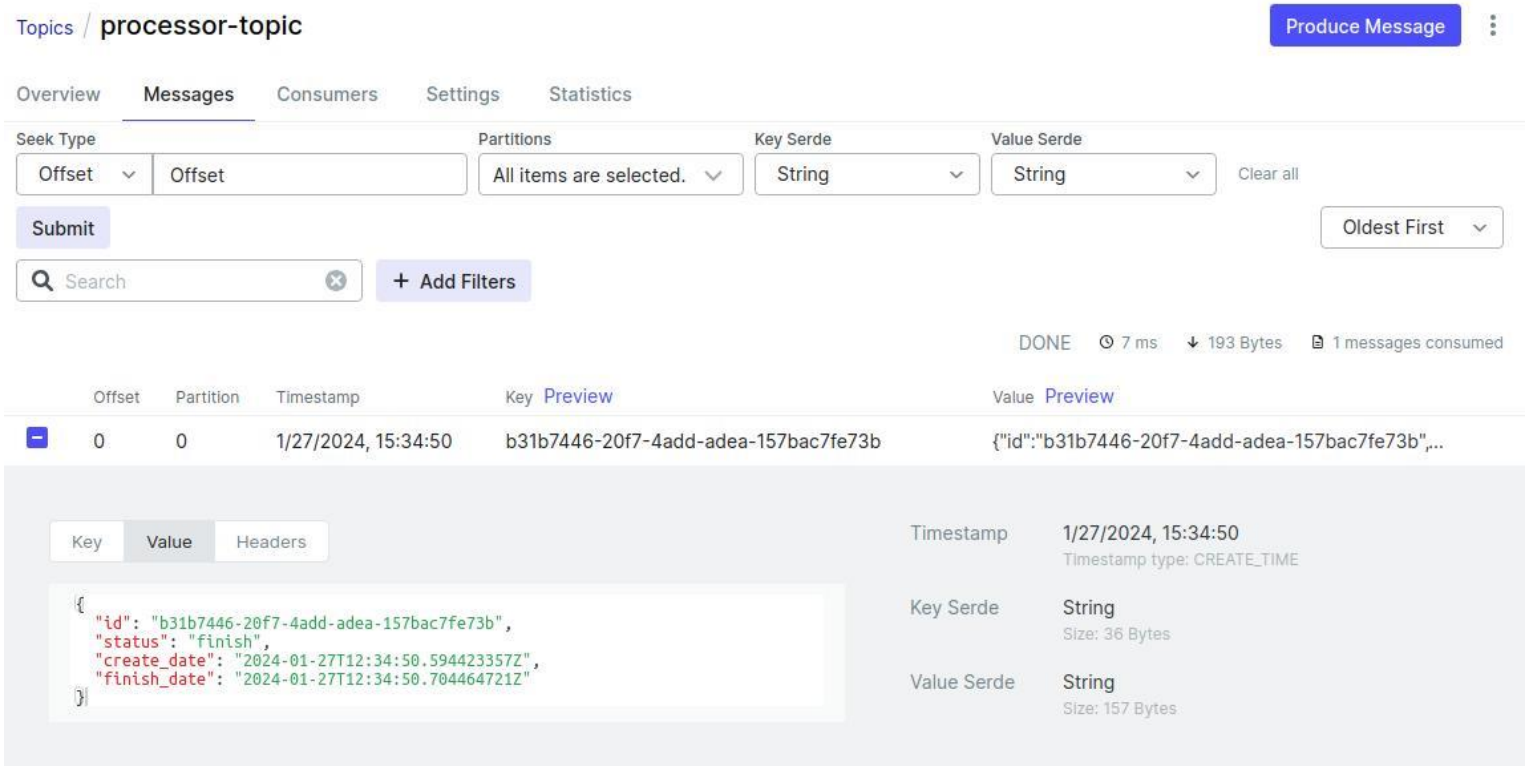
Создан дашборд Services для сервисов

Спасибо за внимание!

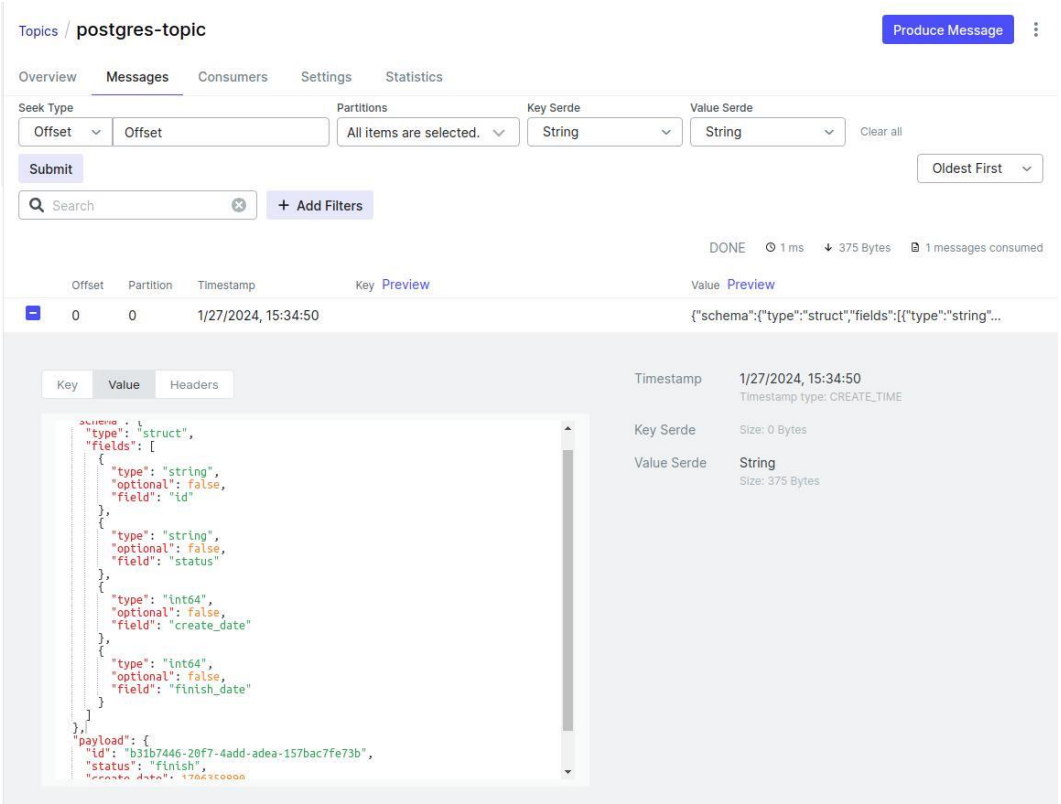
Скриншоты


Topics / receiver-topic
Produce Message
Overview
Messages
Consumers
Settings
Statistics
Seek Type
Offset
Offset
Partitions
All items are selected.
Key Serde
String
Value Serde
String
Clear all
Submit
Oldest First
Search
+ Add Filters
DONE
2 ms
1 KB
1 messages consumed
Offset
Partition
Timestamp
Key Preview
Value Preview
0
0
1/27/2024, 15:34:50
b31b7446-20f7-4add-adea-157bac7fe73b
{"id":"b31b7446-20f7-4add-adea-157bac7fe73b",...
Key
Value
Headers
{
"id": "b31b7446-20f7-4add-adea-157bac7fe73b",
"create_date": "2024-01-27T12:34:50.594423357Z",
"sleep": 99,
"data": "4f163f5f0f9a621d729566c74d10037c4d7bbb0407d1e2c64981855ad8681d0d86d1
e91e00167939cb6694d2c422acd208a0072939487f6999eb9d18a44784045d87f3c67cf2274
6e995af5a25367951baa2ff6cd471c483f15fb90badb37c5821b6d95526a41a9504680b4e7c
8b763a1b1d49d4955c8486216325253fec738dd7a9e28bf921119c160f0702448615bbda083
13f6a8eb668d20bf5059875921e668a5bdf2c7fc4844592d2572bcd0668d2d6c52f5054e2d0
836bf84c7174cb7476364cc3dbd968b0f7172ed85794bb358b0c3b525da1786f9fff094279d
b1944ebd7a19d0f7bbacbe0255aa5b7d44bec40f84c892b9bffd43629b0223beea5f4f74391
f445d15afd4294040374f6924b98cbf8713f8d962d7c8d019192c24224e2cafccae3a61fb58
6b14323a6bc8f9e7df1d929333ff993933bea6f5b3af6de0374366c4719e43a1b067d89bc7f
01f1f573981659a44ff17a4c7215a3b539eb1e5849c6077dbb5722f5717a289a266f9764798
1998ebea89c0b4b373970115e82ed6f4125c8fa7311e4d7defa922daae7786667f7e936cd4f
24abf7df866baa56038367ad6145de1ee8f4a8b0993ebdf8883a0ad8be9c3978b04883e56a1
56a8de563afa467d49dec6a40e9a1d007f033c2823061bdd0eaa59f8e4da6430105220d0b29
688b734b8ea0f3ca9936e8461f10d77c96ea80a7a665f606f6a63b7f"
}
Timestamp
1/27/2024, 15:34:50
Timestamp type: CREATE_TIME
Key Serde
String
Size: 36 Bytes
Value Serde
String
Size: 1 KB

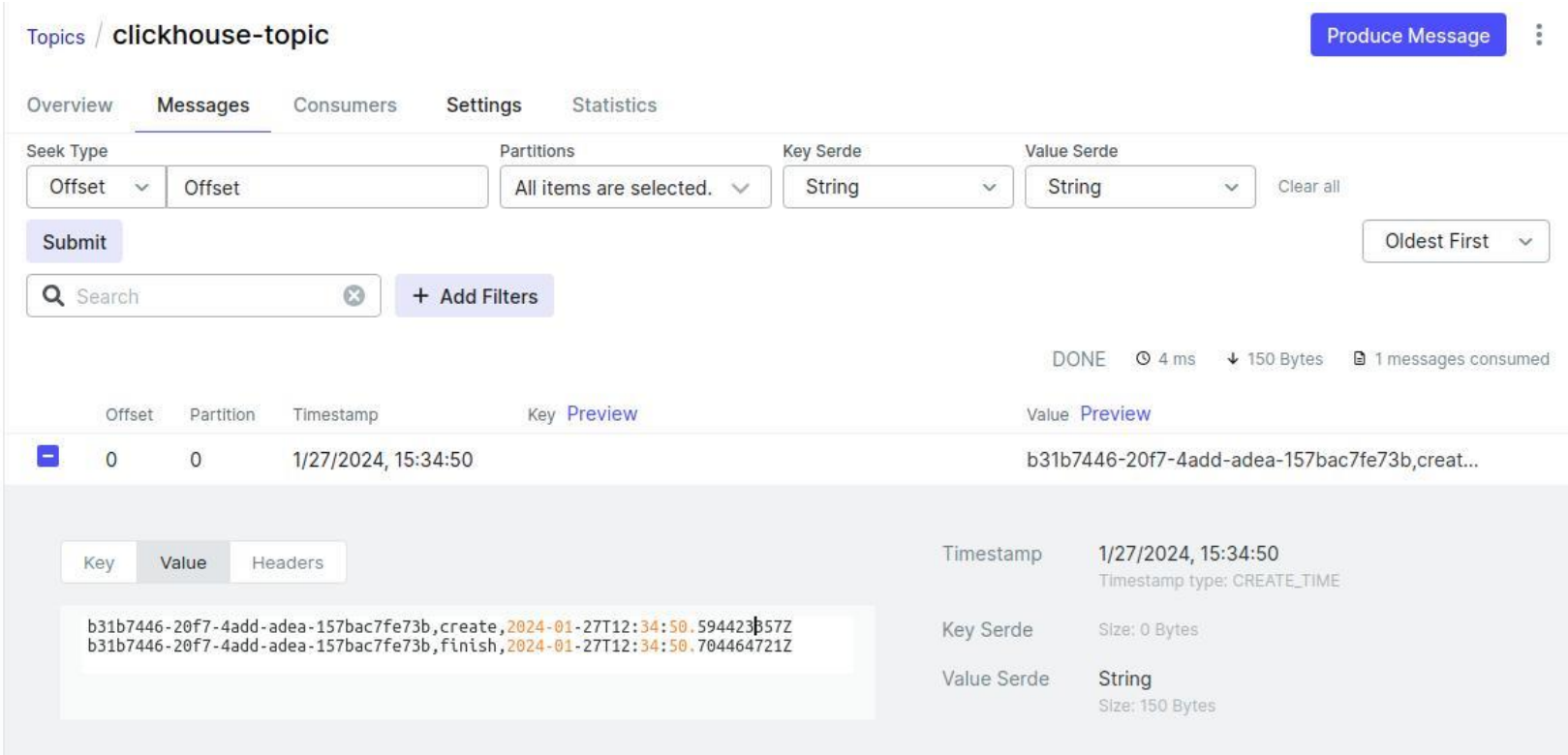
Скриншоты


Topics / processor-topic
Produce Message
Overview
Messages
Consumers
Settings
Statistics
Seek Type
Offset
Offset
Partitions
All items are selected.
Key Serde
String
Value Serde
String
Clear all
Submit
Oldest First
Search
+ Add Filters
DONE
7 ms
193 Bytes
1 messages consumed
Offset
Partition
Timestamp
Key Preview
Value Preview
0
0
1/27/2024, 15:34:50
b31b7446-20f7-4add-adea-157bac7fe73b
{"id":"b31b7446-20f7-4add-adea-157bac7fe73b",...
Key
Value
Headers
{
"id": "b31b7446-20f7-4add-adea-157bac7fe73b",
"status": "finish",
"create_date": "2024-01-27T12:34:50.594423357Z",
"finish_date": "2024-01-27T12:34:50.704464721Z"
}
Timestamp
1/27/2024, 15:34:50
Timestamp type: CREATE_TIME
Key Serde
String
Size: 36 Bytes
Value Serde
String
Size: 157 Bytes

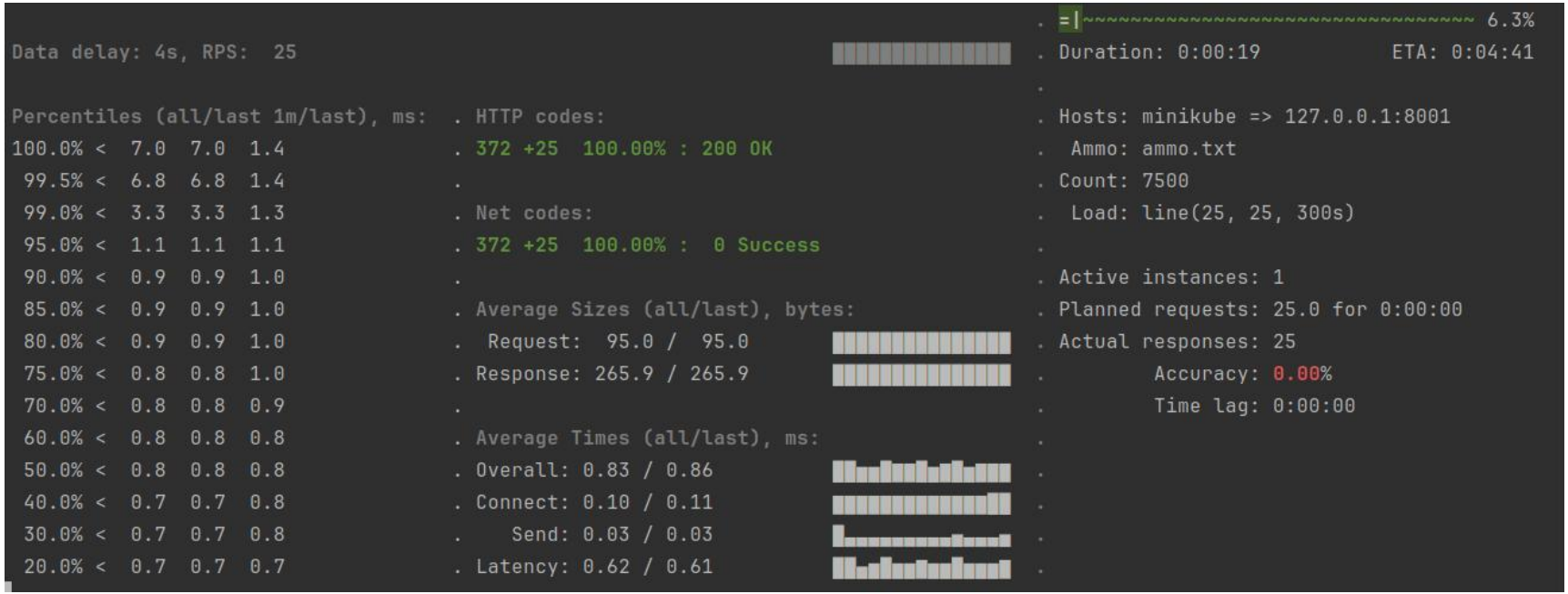
Скриншоты


Topics / postgres-topic
Produce Message
Overview
Messages
Consumers
Settings
Statistics
Seek Type
Offset
Offset
Partitions
All items are selected.
Key Serde
String
Value Serde
String
Clear all
Submit
Oldest First
Search
Add Filters
DONE
1 ms
375 Bytes
1 messages consumed
Offset
Partition
Timestamp
Key Preview
Value Preview
0
0
1/27/2024, 15:34:50
{"schema":{"type":"struct","fields":[{"type":"string"...
Key
Value
Headers
"type": "struct",
"fields": [
{
"type": "string",
"optional": false,
"field": "id"
},
{
"type": "string",
"optional": false,
"field": "status"
},
{
"type": "int64",
"optional": false,
"field": "create_date"
},
{
"type": "int64",
"optional": false,
"field": "finish_date"
}
]
},
"payload": {
"id": "b31b7446-20f7-4add-adea-157bac7fe73b",
"status": "finish",
Timestamp
1/27/2024, 15:34:50
Timestamp type: CREATE_TIME
Key Serde
Size: 0 Bytes
Value Serde
String
Size: 375 Bytes

Скриншоты


Topics / clickhouse-topic
Produce Message
Overview
Messages
Consumers
Settings
Statistics
Seek Type
Offset
Offset
Partitions
All items are selected.
Key Serde
String
Value Serde
String
Clear all
Submit
Oldest First
Search
+ Add Filters
DONE
4 ms
150 Bytes
1 messages consumed
Offset
Partition
Timestamp
Key Preview
Value Preview
0
0
1/27/2024, 15:34:50
b31b7446-20f7-4add-adea-157bac7fe73b,creat...
Key
Value
Headers
b31b7446-20f7-4add-adea-157bac7fe73b,create,2024-01-27T12:34:50.594423357Z
b31b7446-20f7-4add-adea-157bac7fe73b,finish,2024-01-27T12:34:50.704464721Z
Timestamp
1/27/2024, 15:34:50
Timestamp type: CREATE_TIME
Key Serde
Size: 0 Bytes
Value Serde
String
Size: 150 Bytes

Скриншоты


Data delay: 4s, RPS: 25
Percentiles (all/last 1m/last), ms:
100.0% < 7.0 7.0 1.4
99.5% < 6.8 6.8 1.4
99.0% < 3.3 3.3 1.3
95.0% < 1.1 1.1 1.1
90.0% < 0.9 0.9 1.0
85.0% < 0.9 0.9 1.0
80.0% < 0.9 0.9 1.0
75.0% < 0.8 0.8 1.0
70.0% < 0.8 0.8 0.9
60.0% < 0.8 0.8 0.8
50.0% < 0.8 0.8 0.8
40.0% < 0.7 0.7 0.8
30.0% < 0.7 0.7 0.8
20.0% < 0.7 0.7 0.7
HTTP codes:
372 +25 100.00% : 200 OK
Net codes:
372 +25 100.00% : 0 Success
Average Sizes (all/last), bytes:
Request: 95.0 / 95.0
Response: 265.9 / 265.9
Average Times (all/last), ms:
Overall: 0.83 / 0.86
Connect: 0.10 / 0.11
Send: 0.03 / 0.03
Latency: 0.62 / 0.61
6.3%
Duration: 0:00:19 ETA: 0:04:41
Hosts: minikube => 127.0.0.1:8001
Ammo: ammo.txt
Count: 7500
Load: line(25, 25, 300s)
Active instances: 1
Planned requests: 25.0 for 0:00:00
Actual responses: 25
Accuracy: 0.00%
Time lag: 0:00:00

Скриншоты


Receiver
Requests RPS
20
10
0
19:08:00
19:09:00
19:10:00
19:11:00
19:12:00
19:13:00
19:14:00
19:15:00
19:16:00
19:17:00
19:18:00
19:19:00
19:20:00
19:21:00
200 POST /api/v1/create/job
RAM
42.9 MiB
38.1 MiB
33.4 MiB
28.6 MiB
23.8 MiB
19:10
19:15
19:20
receiver:8001
Goroutines
9.0
8.8
8.6
8.4
8.2
8.0
19:10
19:15
19:20
receiver:8001
Resident memory (RSS) bytes
38.1 MiB
28.6 MiB
19.1 MiB
19:10
19:15
19:20
receiver:8001
CPU
0.030 mz
0.025 mz
0.020 mz
0.015 mz
0.010 mz
0.005 mz
0.000 mz
19:10
19:15
19:20
receiver:8001

Скриншоты


Processor
Kafka RPS
20
15
10
5
0
19:08:00
19:09:00
19:10:00
19:11:00
19:12:00
19:13:00
19:14:00
19:15:00
19:16:00
19:17:00
19:18:00
19:19:00
19:20:00
19:21:00
19:22:00
processor-1: done success
processor-2: done success
RAM
28.6 MiB
26.7 MiB
24.8 MiB
22.9 MiB
19:10
19:15
19:20
build-processor-1:8002
build-processor-2:8002
Goroutines
15
10
5
0
19:10
19:15
19:20
build-processor-1:8002
build-processor-2:8002
Resident memory (RSS) bytes
33.4 MiB
28.6 MiB
23.8 MiB
19.1 MiB
14.3 MiB
19:10
19:15
19:20
build-processor-1:8002
build-processor-2:8002
CPU
0.0150 mz
0.0125 mz
0.0100 mz
0.0075 mz
0.0050 mz
0.0025 mz
0.0000 mz
19:10
19:15
19:20
build-processor-1:8002
build-processor-2:8002

Скриншоты


Converter
Kafka RPS
20
15
10
5
0
19:09:00
19:10:00
19:11:00
19:12:00
19:13:00
19:14:00
19:15:00
19:16:00
19:17:00
19:18:00
19:19:00
19:20:00
19:21:00
19:22:00
19:23:00
done success
RAM
76.3 MiB
66.8 MiB
57.2 MiB
47.7 MiB
19:10
19:15
19:20
converter:8003
Goroutines
20
15
10
5
0
19:10
19:15
19:20
converter:8003
Resident memory (RSS) bytes
57.2 MiB
47.7 MiB
38.1 MiB
28.6 MiB
19.1 MiB
19:10
19:15
19:20
converter:8003
CPU
0.03 mz
0.02 mz
0.01 mz
0.00 mz
19:10
19:15
19:20
converter:8003

Скриншоты


Informer
Requests count
2.0
1.5
1.0
0.5
0.0
19:00
19:05
19:10
19:15
19:20
19:25
200 GET /info/00000000-0000-0000-0000-000000000000
200 GET /info/b31b7446-20f7-4add-adea-157bac7fe73b
RAM
1.91 MiB
1.43 MiB
977 KiB
488 KiB
19:00
19:05
19:10
19:15
19:20
19:25
informer:8004
Goroutines
10
9
8
7
6
5
19:00
19:05
19:10
19:15
19:20
19:25
informer:8004
Resident memory (RSS) bytes
10.5 MiB
10.0 MiB
9.54 MiB
9.06 MiB
8.58 MiB
8.11 MiB
19:00
19:05
19:10
19:15
19:20
19:25
informer:8004
CPU
0.0004 mz
0.0003 mz
0.0002 mz
0.0001 mz
0.0000 mz
19:00
19:05
19:10
19:15
19:20
19:25
informer:8004