

## Проектная работа на тему: Сбор и обработка данных по валютам с помощью ClickHouse.

Система ежедневно получает данные с сайта банка России по api в формате xml:

<https://www.cbr.ru/development/SXML/>

### 1. Сборка компонентов для получения и хранения данных.

Переходим в папку проекта - OtusClickHouse-Project\.

Собираем и запускаем контейнер PostgreSQL с помощью скрипта docker/build\_docker\_postgres.sh.

- build\_docker\_postgres.sh:

```
#!/bin/bash
docker stop postgresql
docker rm postgresql

docker run --name postgresql \
-e POSTGRES_PASSWORD=passwordpg1234 \
-e POSTGRES_USER=userpg \
-e POSTGRES_DB=pgdb \
-p 5433:5432 \
-v "/postgresql/data":"/var/lib/postgresql/data" \
-d postgres:latest
```

Подключаемся к базе и создаем схему с таблицей:

```
CREATE SCHEMA pgclick;

CREATE TABLE IF NOT EXISTS pgclick.valute_data(
  id bigint NOT NULL PRIMARY KEY,
  date VARCHAR(36) NOT NULL,
  name VARCHAR(200) NOT NULL,
  str_id VARCHAR(20) NOT NULL,
  num_code VARCHAR(20) NOT NULL,
  char_code VARCHAR(20) NOT NULL,
  nominal VARCHAR(20) NOT NULL,
  value VARCHAR(50) NOT null
);

CREATE UNIQUE INDEX idx_valute_pk
ON pgclick.valute_data(id);

CREATE UNIQUE INDEX idx_valute_data
ON pgclick.valute_data(date, str_id);

CREATE SEQUENCE IF NOT EXISTS pgclick.seq_valute_id;
```

Собираем приложение xml-parser с помощью скрипта xml-parser/build\_app.sh. Для сборки потребуется установленный maven. Docker-контейнер собираем и запускаем с помощью скрипта xml-parser/build\_docker.sh.

- build\_app.sh:

```
#!/bin/bash
git pull
mvn clean install
```

- build\_docker.sh:

```
#!/bin/bash
```

```
docker build -f app.Dockerfile -t uoles/xml-parser:1.0.1 .
```

```
docker stop xml-parser
```

```
docker rm xml-parser
```

```
docker run -d --name xml-parser \
```

```
--network=host \
```

```
--publish 8090:8090 \
```

```
uoles/xml-parser:1.0.1
```

- app.Dockerfile:

```
FROM docker.io/library/openjdk:17
```

```
MAINTAINER Maksim Kulikov <max.uoles@rambler.ru>
```

```
COPY target/xml-parser.jar xml-parser.jar
```

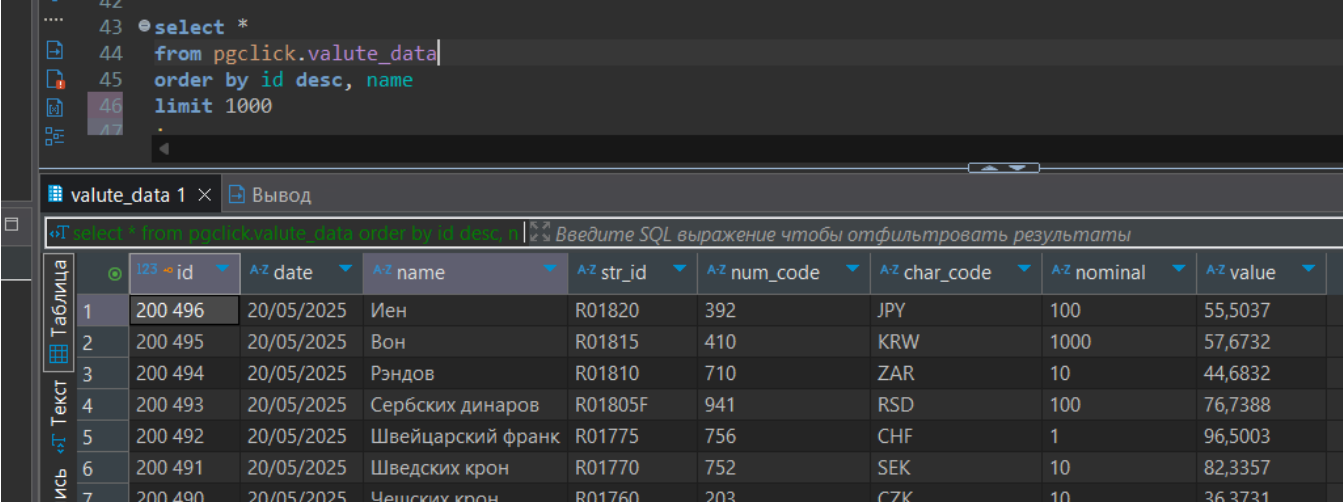
```
ENV TZ=Europe/Moscow
```

```
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone
```

```
EXPOSE 8090
```

```
CMD ["java","-jar","xml-parser.jar"]
```

После запуска базы и приложения начинается сбор данных с 1 декабря 2010 года по текущее время. Так же, ежедневно в 12:30 подгружаются данные на текущие сутки.



	id	date	name	str_id	num_code	char_code	nominal	value
1	200 496	20/05/2025	Иен	R01820	392	JPY	100	55,5037
2	200 495	20/05/2025	Вон	R01815	410	KRW	1000	57,6732
3	200 494	20/05/2025	Рэндов	R01810	710	ZAR	10	44,6832
4	200 493	20/05/2025	Сербских динаров	R01805F	941	RSD	100	76,7388
5	200 492	20/05/2025	Швейцарский франк	R01775	756	CHF	1	96,5003
6	200 491	20/05/2025	Шведских крон	R01770	752	SEK	10	82,3357
7	200 490	20/05/2025	Чешских крон	R01760	203	CZK	10	36,3731

## 2. Сборка компонентов для обработки данных.

Переходим в каталог OtusClickHouse-Project\docker и запускаем сборку docker-compose.yml, в котором запускаются clickhouse, kafka и superset.

Для этого выполняем команду: `docker-compose up -d`

Superset – <http://localhost:8088>

Kafka-ui – <http://localhost:8089>

docker-compose.yml:

```
x-superset-image: &superset-image apache/superset:${TAG:-latest-dev}
x-superset-depends-on: &superset-depends-on
  - db
  - redis
x-superset-only-depends-on: &superset-only-depends-on
  - db
  - redis
  - clickhouse-server
x-superset-volumes: &superset-volumes
  - ./docker:/app/docker
  - superset_home:/app/superset_home

version: "3.7"
services:
  clickhouse-server:
    container_name: clickhouse-server
    image: uoles/clickhouse:25.2.1
    build:
      context: .
      dockerfile: clickhouse-25.2.1.Dockerfile
    environment:
      CLICKHOUSE_DB: my_database
      CLICKHOUSE_USER: username
      CLICKHOUSE_DEFAULT_ACCESS_MANAGEMENT: 1
      CLICKHOUSE_PASSWORD: password
    ports:
      - "18123:8123"
      - "19000:9000"
    volumes:
      - ./clickhouse/config.xml:/etc/clickhouse-server/config.xml
      - ./clickhouse/zookeeper-servers.xml:/etc/clickhouse-server/conf.d/zookeeper-servers.xml
    ulimits:
      nofile:
        soft: 262144
        hard: 262144
    depends_on:
      - kafka
    links:
      - kafka

  zookeeper:
    container_name: zookeeper
    image: confluentinc/cp-zookeeper:6.2.4
    healthcheck:
      test: [ "CMD", "nc", "-vz", "localhost", "2181" ]
      interval: 10s
      timeout: 3s
      retries: 3
    environment:
```

ZOOKEEPER\_CLIENT\_PORT: 2181

ZOOKEEPER\_TICK\_TIME: 2000

ports:

- 22181:2181

kafka:

container\_name: kafka

image: confluentinc/cp-kafka:6.2.4

depends\_on:

zookeeper:

condition: service\_healthy

ports:

- "29092:29092"

- "29093:29093"

healthcheck:

test: [ "CMD", "nc", "-vz", "localhost", "9092" ]

interval: 10s

timeout: 3s

retries: 3

environment:

KAFKA\_BROKER\_ID: 1

KAFKA\_ZOOKEEPER\_CONNECT: zookeeper:2181

KAFKA\_LISTENERS: OUTSIDE://:29092,INTERNAL://:9092,EXTERNAL\_DIFFERENT\_HOST://:29093

KAFKA\_ADVERTISED\_LISTENERS:

OUTSIDE://localhost:29092,INTERNAL://kafka:9092,EXTERNAL\_DIFFERENT\_HOST://89.169.3.137:29093

KAFKA\_LISTENER\_SECURITY\_PROTOCOL\_MAP:

INTERNAL:PLAINTEXT,OUTSIDE:PLAINTEXT,EXTERNAL\_DIFFERENT\_HOST:PLAINTEXT

KAFKA\_INTER\_BROKER\_LISTENER\_NAME: INTERNAL

KAFKA\_OFFSETS\_TOPIC\_REPLICATION\_FACTOR: 1

kafka-ui:

image: provectuslabs/kafka-ui

container\_name: kafka-ui

ports:

- "8089:8080"

restart: always

depends\_on:

kafka:

condition: service\_healthy

environment:

KAFKA\_CLUSTERS\_0\_NAME: local

KAFKA\_CLUSTERS\_0\_BOOTSTRAPSERVERS: kafka:9092

redis:

image: redis:7

container\_name: superset\_cache

restart: unless-stopped

volumes:

- redis:/data

db:

env\_file: docker/.env-non-dev

image: postgres:14

container\_name: superset\_db

restart: unless-stopped

volumes:

- db\_home:/var/lib/postgresql/data

superset:

env\_file: docker/.env-non-dev

image: \*superset-image

container\_name: superset\_app  
command: [ sh, -c, "pip install clickhouse-connect && /app/docker/docker-bootstrap.sh app-gunicorn" ]  
user: "root"  
restart: unless-stopped  
ports:  
- 8088:8088  
depends\_on: \*superset-only-depends-on  
volumes: \*superset-volumes  
links:  
- clickhouse-server

superset-init:  
image: \*superset-image  
container\_name: superset\_init  
command: [ "/app/docker/docker-init.sh" ]  
env\_file: docker/.env-non-dev  
depends\_on: \*superset-depends-on  
user: "root"  
volumes: \*superset-volumes  
healthcheck:  
disable: true

superset-worker:  
image: \*superset-image  
container\_name: superset\_worker  
command: [ "/app/docker/docker-bootstrap.sh", "worker" ]  
env\_file: docker/.env-non-dev  
restart: unless-stopped  
depends\_on: \*superset-depends-on  
user: "root"  
volumes: \*superset-volumes  
healthcheck:  
test: [ "CMD-SHELL", "celery inspect ping -A superset.tasks.celery\_app:app -d celery@\$HOSTNAME" ]

superset-worker-beat:  
image: \*superset-image  
container\_name: superset\_worker\_beat  
command: [ "/app/docker/docker-bootstrap.sh", "beat" ]  
env\_file: docker/.env-non-dev  
restart: unless-stopped  
depends\_on: \*superset-depends-on  
user: "root"  
volumes: \*superset-volumes  
healthcheck:  
disable: true

volumes:  
superset\_home:  
external: false  
db\_home:  
external: false  
redis:  
external: false

- Подключаемся к clickhouse и проливаем таблицы и выюхи:

```
CREATE TABLE valute_data
(
    id UInt64,
    date Date,
    name String,
    str_id String,
    num_code String,
    char_code String,
    nominal UInt32,
    value Float32
)
ENGINE = MergeTree
ORDER BY (id, date, str_id);

CREATE TABLE valute_data_queue
(
    id UInt64,
    date String,
    name String,
    str_id String,
    num_code String,
    char_code String,
    nominal String,
    value String
)
ENGINE = Kafka
SETTINGS
    kafka_broker_list = 'kafka:9092',
    kafka_topic_list = 'valute_topic',
    kafka_group_name = 'valute_group',
    kafka_format = 'JSONEachRow',
    kafka_row_delimiter = '\n',
    kafka_num_consumers = 1,
    kafka_skip_broken_messages = 10,
    kafka_thread_per_consumer = 0;

CREATE MATERIALIZED VIEW valute_data_queue_mv TO valute_data
(
    id UInt64,
    date Date,
    name String,
    str_id String,
    num_code String,
    char_code String,
    nominal UInt32,
    value Float32
) AS
SELECT
    id,
    parseDateTimeBestEffortOrNull(date, 'Europe/Moscow') AS date,
    name,
    str_id,
    num_code,
    char_code,
    toInt32OrNull(nominal) as nominal,
    toFloat32OrNull(replaceOne(value, ',', '.')) / nominal as value
FROM valute_data_queue
WHERE name not in ('Евро', 'Дирхам ОАЭ', 'Тенге');
```

```

CREATE MATERIALIZED VIEW valute_data_evro_mv TO valute_data_evro
(
    id UInt64,
    date Date,
    name String,
    str_id String,
    num_code String,
    char_code String,
    nominal UInt32,
    value Float32
) AS
SELECT
    id,
    parseDateTimeBestEffortOrNull(date, 'Europe/Moscow') AS date,
    name,
    str_id,
    num_code,
    char_code,
    toInt32OrNull(nominal) AS nominal,
    toFloat32OrNull(replaceOne(value, ',', '.')) / nominal AS value
FROM valute_data_queue
WHERE name = 'Евро';

CREATE TABLE valute_data_evro
(
    id UInt64,
    date Date,
    name String,
    str_id String,
    num_code String,
    char_code String,
    nominal UInt32,
    value Float32
)
ENGINE = MergeTree
ORDER BY (id, date, str_id);

CREATE MATERIALIZED VIEW valute_data_dirham_mv TO valute_data_dirham
(
    id UInt64,
    date Date,
    name String,
    str_id String,
    num_code String,
    char_code String,
    nominal UInt32,
    value Float32
) AS
SELECT
    id,
    parseDateTimeBestEffortOrNull(date, 'Europe/Moscow') AS date,
    name,
    str_id,
    num_code,
    char_code,
    toInt32OrNull(nominal) AS nominal,
    toFloat32OrNull(replaceOne(value, ',', '.')) / nominal AS value
FROM valute_data_queue
WHERE name = 'Дирхам ОАЭ';

CREATE TABLE valute_data_dirham
(
    id UInt64,
    date Date,
    name String,
    str_id String,

```

```

        num_code String,
        char_code String,
        nominal UInt32,
        value Float32
    )
ENGINE = MergeTree
ORDER BY (id, date, str_id);

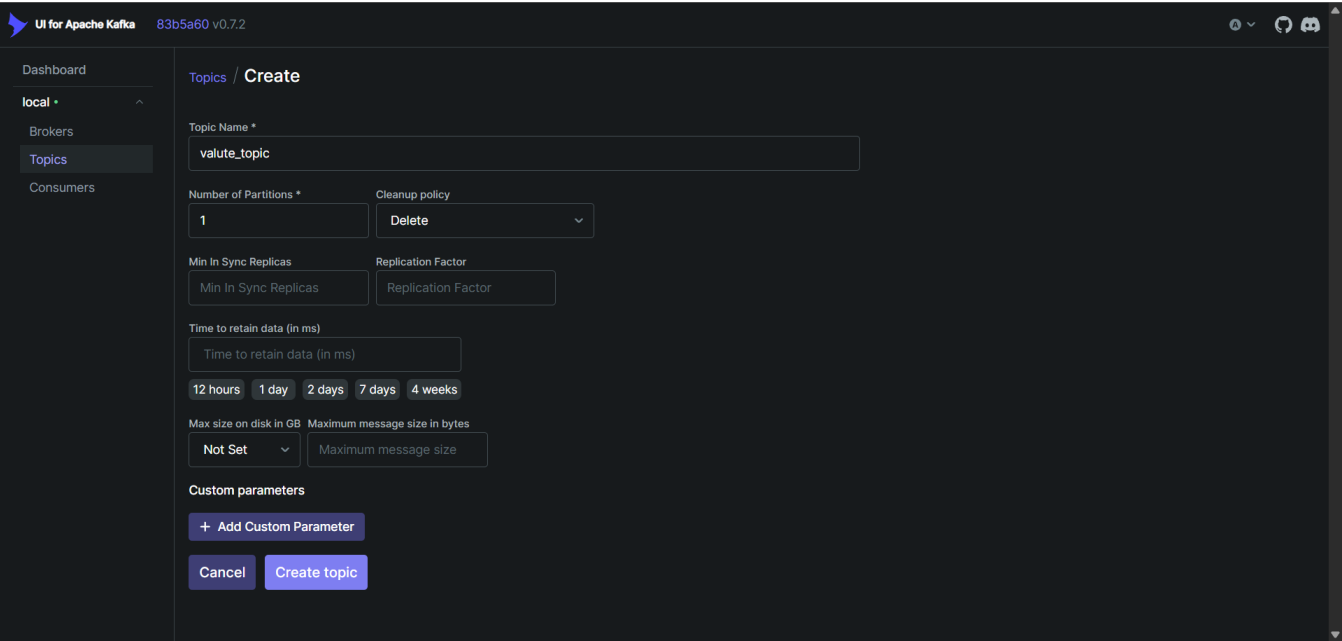
CREATE MATERIALIZED VIEW valute_data_tenge_mv TO valute_data_tenge
(
    id UInt64,
    date Date,
    name String,
    str_id String,
    num_code String,
    char_code String,
    nominal UInt32,
    value Float32
) AS
SELECT
    id,
    parseDateTimeBestEffortOrNull(date, 'Europe/Moscow') AS date,
    name,
    str_id,
    num_code,
    char_code,
    toInt32OrNull(nominal) as nominal,
    toFloat32OrNull(replaceOne(value, ',', '.')) / nominal as value
FROM valute_data_queue
WHERE name = 'Тенге';

CREATE TABLE valute_data_tenge
(
    id UInt64,
    date Date,
    name String,
    str_id String,
    num_code String,
    char_code String,
    nominal UInt32,
    value Float32
)
ENGINE = MergeTree
ORDER BY (id, date, str_id);

```



- Подключаемся к kafka-ui и создаем топик “valute\_topic” куда будут приходить обновления от debezium:



Теперь kafka и clickhouse могут принимать данные.

### 3. Настраиваем PostgreSQL и разворачиваем приложение с компонентом debezium.

Подключаемся к ранее развернутому контейнеру PostgreSQL и подготавливаем ее к подключению приложения. Для этого нужно выставить:

- уровень репликации “logical” и перезапустить базу;
- создать публикацию, с описанием таблиц и операций, которые будем обрабатывать;
- нужен пользователь со свойством REPLICATION (есть по умолчанию у пользователя при создании контейнера).

Скрипты:

```
ALTER SYSTEM SET wal_level = logical;

SHOW wal_level; -- проверяем уровень репликации после рестарта

CREATE PUBLICATION pgclick_publication
  FOR TABLES IN SCHEMA pgclick
  with(publish = 'insert');
```

Для запуска приложения переходим в каталог OtusClickHouse-Project\debezium\. Так же, как и xml-parser собираем и разворачиваем debezium скриптами build\_app.sh и build\_docker.sh.

- build\_app.sh:

```
#!/bin/bash
git pull
mvn clean install
```

- build\_docker.sh:

```
#!/bin/bash
docker build -f app.Dockerfile -t uoles/debezium:1.0.1 .
```

```
docker stop debezium
docker rm debezium
```

```
docker run -d --name debezium \
  --network=host \
  --publish 8091:8091 \
  uoles/debezium:1.0.1
```

- app.Dockerfile:

```
FROM docker.io/library/openjdk:17
MAINTAINER Maksim Kulikov <max.woles@rambler.ru>
```

```
COPY target/debezium.jar debezium.jar
```

```
EXPOSE 8091
```

```
CMD ["java", "-jar", "debezium.jar"]
```

Приложение подключается к бд PostgreSQL, получает все изменения в базе из wal-журнала через библиотеку debezium и отправляет данные в kafka в формате json.

Конфигурация debezium:

```

* Time: 23:20
*/
@Configuration no usages 1 uoles
public class DebeziumConfig {

    @Bean no usages 1 uoles
    public io.debezium.config.Configuration connector(Environment env) throws IOException {
        return io.debezium.config.Configuration.create()
            .with("name", "pgclick-postgres-connector")
            .with("connector.class", "io.debezium.connector.postgresql.PostgresConnector")
            .with("offset.storage", "ru.uoles.debezium.offset.PostgreOffsetBackingStore")
            .with("offset.flush.interval.ms", "5000")
            .with("database.hostname", PropertiesConfig.getHost())
            .with("database.port", PropertiesConfig.getPort())
            .with("database.user", PropertiesConfig.getUsername())
            .with("database.password", PropertiesConfig.getPassword())
            .with("database.dbname", PropertiesConfig.getDatabaseName())
            .with("database.schema", PropertiesConfig.getSchema())
            .with("database.server.id", "10181")
            .with("database.server.name", "pgclick-postgres-db")
            .with("database.history", "io.debezium.relational.history.MemoryDatabaseHistory")
            .with("table.include.list", "pgclick_valute_data")
            .with("publication.autocreate.mode", "filtered")
            .with("plugin.name", "pgoutput")
            .with("publication.name", PUBLICATION_NAME)
            .with("slot.name", SLOT_NAME)
            .with("snapshot.mode", "initial")
            .build();
    }
}

```

```

debezium\...\application.properties x DayLoadScheduler.java x
Spring Boot configuration files are supported by IntelliJ IDEA Ultimate
1 server.port=8091
2
3 logging.level.ru.uoles=INFO
4 logging.level.io.debezium=INFO
5
6 debezium.datasource.host=localhost
7 debezium.datasource.port=5433
8 debezium.datasource.database=pgdb
9 debezium.datasource.username=userpg
10 debezium.datasource.password=passwordpg1234
11 debezium.datasource.schema=pgclick
12 debezium.snapshot.initial=0
13
14 kafka.bootstrap.server=localhost:29092

```

Скрины логами и данными:

Ask Gordon BETA

Containers

Images

Volumes

Builds

Docker Hub

Docker Scout

Extensions

Containers / xml-parser

xml-parser

9479ce0cb295 uoles/xml-parser:1.0.1 8090:8090

STATUS Running (5 hours ago)

Logs

Inspect

Bind mounts

Exec

Files

Stats

2025-05-20 12:24:37.066 INFO 1 --- [ main ] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.

2025-05-20 12:24:37.083 INFO 1 --- [ main ] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 6 ms. Found 0 JPA repository interfaces.

2025-05-20 12:24:37.885 INFO 1 --- [ main ] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8090 (http)

2025-05-20 12:24:37.897 INFO 1 --- [ main ] o.apache.catalina.core.StandardService : Starting service [Tomcat]

2025-05-20 12:24:37.898 INFO 1 --- [ main ] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.64]

2025-05-20 12:24:37.985 INFO 1 --- [ main ] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext

2025-05-20 12:24:37.985 INFO 1 --- [ main ] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1357 ms

2025-05-20 12:24:38.393 INFO 1 --- [ main ] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]

2025-05-20 12:24:38.455 INFO 1 --- [ main ] org.hibernate.Version : HHH0000412: Hibernate ORM core version 5.6.9.Final

2025-05-20 12:24:38.650 INFO 1 --- [ main ] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations {5.1.2.Final}

2025-05-20 12:24:38.840 INFO 1 --- [ main ] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect.PostgreSQL10Dialect

2025-05-20 12:24:39.063 INFO 1 --- [ main ] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform implementation:

[org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]

2025-05-20 12:24:39.082 INFO 1 --- [ main ] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'

2025-05-20 12:24:39.212 WARN 1 --- [ main ] JpaBaseConfiguration\$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning

2025-05-20 12:24:39.591 INFO 1 --- [ main ] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8090 (http) with context path ''

2025-05-20 12:24:39.607 INFO 1 --- [ main ] ru.uoles.Application : Started Application in 3.576 seconds (JVM running for 3.953)

2025-05-20 12:24:42.603 INFO 1 --- [ scheduling-1 ] ru.uoles.service.XmlProcessService : --- Current date 20, 5, 2025

2025-05-20 12:24:42.612 INFO 1 --- [ scheduling-1 ] ru.uoles.schedulers.OnceLoadScheduler : --- OnceLoadScheduler. Get data to current date: 20/05/2025

2025-05-20 12:24:43.434 INFO 1 --- [ scheduling-1 ] ru.uoles.service.XmlProcessService : Process date: 19/05/2025

2025-05-20 12:24:43.747 INFO 1 --- [ scheduling-1 ] ru.uoles.service.XmlProcessService : Process date: 20/05/2025

2025-05-20 12:29:59.988 INFO 1 --- [ scheduling-1 ] ru.uoles.service.XmlProcessService : Process date: 20/05/2025

2025-05-20 12:29:59.989 INFO 1 --- [ scheduling-1 ] ru.uoles.schedulers.DayLoadScheduler : --- DayLoadScheduler. Get data to current date: 20/05/2025

2025-05-20 12:30:00.639 INFO 1 --- [ scheduling-1 ] ru.uoles.service.XmlProcessService : Process date: 20/05/2025

Engine running

RAM 7.29 GB CPU 1.00% Disk: 15.65 GB used (limit 1006.85 GB)

Terminal New version available

Ask Gordon BETA

Containers

Images

Volumes

Builds

Docker Hub

Docker Scout

Extensions

Containers [Give feedback](#)

View all your running containers and applications. [Learn more](#)

Container CPU usage 23.64% / 1600% (16 CPUs available)

Container memory usage 5.59GB / 7.49GB

Show charts

Search

Only show running containers

	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	xml-parser	5d2aeaa8793d	uoles/xml-parser:1.0.1	8090:8090	0.14%	8 minutes ago	<div><div></div><div></div><div></div></div>
<input type="checkbox"/>	debezium	02a7c649c8da	uoles/debezium:1.0.1	8091:8091	1.34%	7 minutes ago	<div><div></div><div></div><div></div></div>
<input type="checkbox"/>	postgresql	78ada6182a86	postgres:latest	5433:5432	0.64%	8 minutes ago	<div><div></div><div></div><div></div></div>
<input type="checkbox"/>	docker	-	-	-	21.52%	5 hours ago	<div><div></div><div></div><div></div></div>

Showing 4 items

Engine running

RAM 7.28 GB CPU 4.32% Disk: 15.57 GB used (limit 1006.85 GB)

Terminal New version available

docker desktopPERSONAL

Q SearchCtrl+K

2

U

Ask GordonBETA

Containers

Images

Volumes

Builds

Docker Hub

Docker Scout

Extensions

ContainersGive feedback

View all your running containers and applications. Learn more

Container CPU usage

20.97% / 1600% (16 CPUs available)

Container memory usage

5.59GB / 7.49GB

Show charts

Q Search

Only show running containers

	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
	superset_db	b0c4d21a1a84	postgres:14		0%	5 hours ago	<div></div> <div></div> <div></div>
	zookeeper	b0730f26dbad	confluentinc/cp-zookeeper	22181:2181	0.15%	5 hours ago	<div></div> <div></div> <div></div>
	superset_cache	93e8345174ad	redis:7		0.28%	5 hours ago	<div></div> <div></div> <div></div>
	kafka	f29440c7e2bf	confluentinc/cp-kafka:6.2	29092:29092	0.83%	5 hours ago	<div></div> <div></div> <div></div>
	kafka-ui	8378b04da349	provectuslabs/kafka-ui	8089:8080	0.09%	5 hours ago	<div></div> <div></div> <div></div>
	superset_init	7b1034d7b273	apache/superset:latest-dev		0%	5 hours ago	<div></div> <div></div> <div></div>
	superset_worker	4a4da6f1d123	apache/superset:latest-de		0.05%	5 hours ago	<div></div> <div></div> <div></div>
	superset_worker_beat	72f5ffbaf52d	apache/superset:latest-de		0%	5 hours ago	<div></div> <div></div> <div></div>
	clickhouse-server	6c8d9b9a27dc	uoles/clickhouse:25.2.1	18123:8123	18.07%	5 hours ago	<div></div> <div></div> <div></div>
	superset_app	bb1ce14f98c7	apache/superset:latest-de	8088:8088	0.03%	5 hours ago	<div></div> <div></div> <div></div>

Showing 14 items

Engine runningRAM 7.25 GB CPU 3.81% Disk: 15.57 GB used (limit 1006.85 GB)TerminalNew version available

docker desktopPERSONAL

Q SearchCtrl+K

2

U

Ask GordonBETA

Containers

Images

Volumes

Builds

Docker Hub

Docker Scout

Extensions

Containers / debezium

debezium

02a7c649c8da

uoles/debezium:1.0.1

8091:8091

STATUS

Running (7 hours ago)

Logs

Inspect

Bind mounts

Exec

Files

Stats

2025-05-20 09:24:44.253 INFO 1 --- [pool-1-thread-1] ru.uoles.kafka.KafkaManager : --- ADD. Row: ValuteDto(id=200490, date=20/05/2025, name=Чешских крон, strId=R01760, numCode=203, charCode=CKZ, nominal=10, value=36.3731)

2025-05-20 09:24:44.253 INFO 1 --- [ad | producer-1] ru.uoles.kafka.KafkaManager : --- Received new metadata. Topic:valute\_topic Partition: 0 Offset: 199876 Timestamp: 1747733084250

2025-05-20 09:24:44.253 INFO 1 --- [ad | producer-1] ru.uoles.kafka.KafkaManager : --- Received new metadata. Topic:valute\_topic Partition: 0 Offset: 199877 Timestamp: 1747733084250

2025-05-20 09:24:44.253 INFO 1 --- [ad | producer-1] ru.uoles.kafka.KafkaManager : --- Received new metadata. Topic:valute\_topic Partition: 0 Offset: 199878 Timestamp: 1747733084251

2025-05-20 09:24:44.253 INFO 1 --- [pool-1-thread-1] ru.uoles.kafka.KafkaManager : --- ADD. Row: ValuteDto(id=200491, date=20/05/2025, name=Шведских крон, strId=R01770, numCode=752, charCode=SEK, nominal=10, value=82.3357)

2025-05-20 09:24:44.253 INFO 1 --- [ad | producer-1] ru.uoles.kafka.KafkaManager : --- Received new metadata. Topic:valute\_topic Partition: 0 Offset: 199879 Timestamp: 1747733084251

2025-05-20 09:24:44.253 INFO 1 --- [pool-1-thread-1] ru.uoles.kafka.KafkaManager : --- ADD. Row: ValuteDto(id=200492, date=20/05/2025, name=Швейцарский франк, strId=R01775, numCode=756, charCode=CHF, nominal=1, value=96.5003)

2025-05-20 09:24:44.253 INFO 1 --- [pool-1-thread-1] ru.uoles.kafka.KafkaManager : --- ADD. Row: ValuteDto(id=200493, date=20/05/2025, name=Сербских динаров, strId=R01805F, numCode=941, charCode=RSD, nominal=100, value=76.7308)

2025-05-20 09:24:44.253 INFO 1 --- [pool-1-thread-1] ru.uoles.kafka.KafkaManager : --- ADD. Row: ValuteDto(id=200494, date=20/05/2025, name=Рэндов, strId=R01810, numCode=710, charCode=ZAR, nominal=10, value=44.6832)

2025-05-20 09:24:44.254 INFO 1 --- [pool-1-thread-1] ru.uoles.kafka.KafkaManager : --- ADD. Row: ValuteDto(id=200495, date=20/05/2025, name=Бон, strId=R01815, numCode=410, charCode=KRM, nominal=1000, value=57.6732)

2025-05-20 09:24:44.254 INFO 1 --- [ad | producer-1] ru.uoles.kafka.KafkaManager : --- Received new metadata. Topic:valute\_topic Partition: 0 Offset: 199880 Timestamp: 1747733084251

2025-05-20 09:24:44.254 INFO 1 --- [ad | producer-1] ru.uoles.kafka.KafkaManager : --- Received new metadata. Topic:valute\_topic Partition: 0 Offset: 199881 Timestamp: 1747733084251

2025-05-20 09:24:44.254 INFO 1 --- [pool-1-thread-1] ru.uoles.kafka.KafkaManager : --- ADD. Row: ValuteDto(id=200496, date=20/05/2025, name=Иен, strId=R01820, numCode=392, charCode=JPY, nominal=100, value=55.5037)

Engine runningRAM 7.22 GB CPU 1.88% Disk: 15.65 GB used (limit 1006.85 GB)TerminalNew version available

Данные в топике kafka-ui:

UI for Apache Kafka 83b5a60 v0.7.2

Dashboard

local

Brokers

Topics

Consumers

### Topics

Search by Topic Name

Show Internal Topics

Delete selected topics Copy selected topic Purge messages of selected topics

Topic Name	Partitions	Out of sync replicas	Replication Factor	Number of messages	Size
__consumer_offsets	50	0	1	32	6 KB
valute_topic	1	0	1	199516	31 MB

UI for Apache Kafka 83b5a60 v0.7.2

Dashboard

local

Brokers

Topics

Consumers

### Topics / valute\_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 12 ms 74 KB 500 messages consumed

Offset	Partition	Timestamp	Key	Value
0	0	20.05.2025, 10:09:49		{"id":1,"date":"01/01/2010","name":"Австралийски...
1	0	20.05.2025, 10:09:49		{"id":2,"date":"01/01/2010","name":"Фунт стерлин...
2	0	20.05.2025, 10:09:49		{"id":3,"date":"01/01/2010","name":"Белорусских ...
3	0	20.05.2025, 10:09:49		{"id":4,"date":"01/01/2010","name":"Датских крон...
4	0	20.05.2025, 10:09:49		{"id":5,"date":"01/01/2010","name":"Доллар США",...
5	0	20.05.2025, 10:09:49		{"id":6,"date":"01/01/2010","name":"Евро","str_id":...
6	0	20.05.2025, 10:09:49		{"id":7,"date":"01/01/2010","name":"Исландских к...
7	0	20.05.2025, 10:09:49		{"id":8,"date":"01/01/2010","name":"Тенге","str_id":...
8	0	20.05.2025, 10:09:49		{"id":9,"date":"01/01/2010","name":"Канадский до...
9	0	20.05.2025, 10:09:49		{"id":10,"date":"01/01/2010","name":"Канадский до...

Данные в clickhouse:

```
73
74 select *
75 from valute_data_queue_mv
76 order by id desc;
77
```

valute\_data\_queue\_mv 1

Введите SQL выражение чтобы отфильтровать результаты

	id	date	name	str_id	num_code	char_code	nominal	value
1	200 496	2025-05-20	Иен	R01820	392	JPY	100	0,555037
2	200 495	2025-05-20	Вон	R01815	410	KRW	1 000	0,057673197
3	200 494	2025-05-20	Рэндов	R01810	710	ZAR	10	4,46832
4	200 493	2025-05-20	Сербских динаров	R01805F	941	RSD	100	0,767388
5	200 492	2025-05-20	Швейцарский франк	R01775	756	CHF	1	96,5003
6	200 491	2025-05-20	Шведских крон	R01770	752	SEK	10	8,23357
7	200 490	2025-05-20	Чешских крон	R01760	203	CZK	10	3,63731

```

182
183 select *
184 from valute_data_evro_mv
185 order by id desc;
186

```

valute\_data\_evro\_mv 1 ×

select \* from valute\_data\_evro\_mv order by id desc | Введите SQL выражение чтобы отфильтровать результаты

	123 id	date	A-Z name	A-Z str_id	A-Z num_code	A-Z char_code	123 nominal	123 value
1	200 468	2025-05-20	Евро	R01239	978	EUR	1	90,5098
2	200 057	2025-05-19	Евро	R01239	978	EUR	1	90,7062
3	199 444	2025-05-18	Евро	R01239	978	EUR	1	90,7062
4	199 401	2025-05-17	Евро	R01239	978	EUR	1	90,7062
5	199 358	2025-05-16	Евро	R01239	978	EUR	1	90,1011
6	199 315	2025-05-15	Евро	R01239	978	EUR	1	90,3821
7	199 272	2025-05-14	Евро	R01239	978	EUR	1	89,6956

```

233
234 select *
235 from valute_data_dirham_mv
236 order by date;
237

```

valute\_data\_dirham\_mv 1 ×

select \* from valute\_data\_dirham\_mv order by date | Введите SQL выражение чтобы отфильтровать результаты

	123 id	date	A-Z name	A-Z str_id	A-Z num_code	A-Z char_code	123 nominal	123 value
1	162 892	2023-01-19	Дирхам ОАЭ	R01230	784	AED	1	18,7511
2	162 935	2023-01-20	Дирхам ОАЭ	R01230	784	AED	1	18,744
3	162 978	2023-01-21	Дирхам ОАЭ	R01230	784	AED	1	18,6952
4	163 021	2023-01-22	Дирхам ОАЭ	R01230	784	AED	1	18,6952
5	163 064	2023-01-23	Дирхам ОАЭ	R01230	784	AED	1	18,6952
6	163 107	2023-01-24	Дирхам ОАЭ	R01230	784	AED	1	18,6847
7	163 150	2023-01-25	Дирхам ОАЭ	R01230	784	AED	1	18,7185

```

280
281 select *
282 from valute_data_tenge_mv
283 order by date;
284

```

valute\_data\_tenge\_mv 1 ×

select \* from valute\_data\_tenge\_mv order by date | Введите SQL выражение чтобы отфильтровать результаты

	123 id	date	A-Z name	A-Z str_id	A-Z num_code	A-Z char_code	123 nominal	123 value
1	8	2010-01-01	Тенге	R01335	398	KZT	100	0,20323901
2	26	2010-01-02	Тенге	R01335	398	KZT	100	0,20323901
3	44	2010-01-03	Тенге	R01335	398	KZT	100	0,20323901
4	62	2010-01-04	Тенге	R01335	398	KZT	100	0,20323901
5	80	2010-01-05	Тенге	R01335	398	KZT	100	0,20323901
6	98	2010-01-06	Тенге	R01335	398	KZT	100	0,20323901
7	116	2010-01-07	Тенге	R01335	398	KZT	100	0,20323901

## 4. Создание графиков.

После разворачивания инфраструктуры заходим в Superset и создаем графики.

Графики с отдельными вьюхами:

- курс Евро с 2010 года;
- курс Тенге с 2010 года;
- курс Дирхам ОАЭ с 2010 года;

Графики на общей вьюхе:

- курсы Доллара США и Австралийского доллара;
- курс Вон, минимальный и максимальные значения за месяц;
- курсы Норвежской кроны, Датской кроны и Белорусских рублей.

