Kafka UI краткий гайд

**Предыстория**

Буквально недавно я начал создавать свой аналог сервиса Discord, используя веб-сокеты и передавая сообщения через очередь. В один из моментов мне понадобилось проверить, корректно ли проходят сообщения в этой очереди. Тогда я осознал, как неудобно каждый раз заходить в docker контейнер и вручную вводить команду для просмотра всех сообщений в топиках. После дня анализа и советов от моего коллеги, опытного DevOps-инженера Паши, я пришел к выводу, что kafka-ui от команды provectus — это наилучший и наиболее удобный инструмент для визуализации данных в Kafka.

**Docker-Compose**

Для начала давайте настроем наш docker-compose под современные реалии kafka. Для этого у нас будет использоваться 2 основных образа

1. [zookeeper](https://habr.com/ru/articles/565976/)
2. kafka

services:

zookeeper:

image: confluentinc/cp-zookeeper:7.2.1

hostname: zookeeper

container\_name: zookeeper

ports:

- "2181:2181"

environment:

ZOOKEEPER\_CLIENT\_PORT: 2181

ZOOKEEPER\_TICK\_TIME: 2000

kafka:

image: confluentinc/cp-server:7.2.1

hostname: kafka

container\_name: kafka

depends\_on:

- zookeeper

ports:

- "9092:9092"

- "9997:9997"

environment:

KAFKA\_BROKER\_ID: 1

KAFKA\_ZOOKEEPER\_CONNECT: 'zookeeper:2181'

KAFKA\_LISTENER\_SECURITY\_PROTOCOL\_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT\_HOST:PLAINTEXT

KAFKA\_ADVERTISED\_LISTENERS: PLAINTEXT://kafka:29092,PLAINTEXT\_HOST://localhost:9092

KAFKA\_OFFSETS\_TOPIC\_REPLICATION\_FACTOR: 1

KAFKA\_GROUP\_INITIAL\_REBALANCE\_DELAY\_MS: 0

KAFKA\_CONFLUENT\_LICENSE\_TOPIC\_REPLICATION\_FACTOR: 1

KAFKA\_CONFLUENT\_BALANCER\_TOPIC\_REPLICATION\_FACTOR: 1

KAFKA\_TRANSACTION\_STATE\_LOG\_MIN\_ISR: 1

KAFKA\_TRANSACTION\_STATE\_LOG\_REPLICATION\_FACTOR: 1

KAFKA\_JMX\_PORT: 9997

KAFKA\_JMX\_HOSTNAME: kafka

В параметрах образа для ZooKeeper основное — это указание рабочего порта. Я также добавил параметр tick\_time, который, хоть и не является обязательным, я по привычке всегда указываю. Вкратце, tick\_time используется для синхронизации реплик и задания таймаутов различных операций.

KAFKA\_BROKER\_ID — свойство broker.id — это уникальное и постоянное имя каждого узла в кластере.

KAFKA\_AUTO\_CREATE\_TOPICS\_ENABLE – если значение истинно, то оно позволяет брокерам создавать темы, когда на них впервые ссылается производитель или потребитель. Если для значения установлено значение false, тему следует сначала создать с помощью команды Kafka, а затем использовать.

KAFKA\_ZOOKEEPER\_CONNECT — указывает Кафке, как связаться с Zookeeper.

KAFKA\_OFFSETS\_TOPIC\_REPLICATION\_FACTOR – требуется, когда мы работаем с кластером с одним узлом. Если у вас три или более узлов, мы можем использовать значение по умолчанию.

KAFKA\_LISTENER\_SECURITY\_PROTOCOL\_MAP – определяет пары ключ/значение для протокола безопасности, которые будут использоваться для каждого имени прослушивателя.

KAFKA\_ADVERTISED\_LISTENERS — KAFKA\_ADVERTISED\_LISTENERS — это то, что будет возвращать брокер кафки клиентам, говоря как с ним (или, в зависимости от темы, с необходимым клиенту брокером) связаться и делает Kafka доступным снаружи контейнера, объявляя его местоположение на хосте Docker.

По конфигурации Kafka ситуация аналогичная: основное — это рабочий порт и базовые конфигурационные параметры, связанные с ZooKeeper. Но ключевым моментом здесь является работа с JMX (Java Management Extensions). JMX — это стандартные средства для мониторинга и управления приложениями, работающими на Java. В рамках Apache Kafka, JMX позволяет мониторить различные метрики работы брокера. Для наших нужд достаточно указать порт и имя хоста для JMX, который, как вы возможно догадались, в дальнейшем будет использоваться в kafka-ui.

Последний этап — это настройка образа для kafka-ui. Ознакомившись с [официальной документац](https://github.com/provectus/kafka-ui/blob/master/documentation/compose/DOCKER_COMPOSE.md)ией команды provectus, вы найдете множество шаблонов для kafka-ui в комбинации с другими сервисами. Однако, стартовый шаблон для связки Kafka, ZooKeeper и kafka-ui на официальном сайте отсутствует. Так что я предоставлю вам свою рабочую версию файла compose.

version: '2'

services:

zookeeper:

image: confluentinc/cp-zookeeper:7.2.1

hostname: zookeeper

container\_name: zookeeper

ports:

- "2181:2181"

environment:

ZOOKEEPER\_CLIENT\_PORT: 2181

ZOOKEEPER\_TICK\_TIME: 2000

kafka:

image: confluentinc/cp-server:7.2.1

hostname: kafka

container\_name: kafka

depends\_on:

- zookeeper

ports:

- "9092:9092"

- "9997:9997"

environment:

KAFKA\_BROKER\_ID: 1

KAFKA\_ZOOKEEPER\_CONNECT: 'zookeeper:2181'

KAFKA\_LISTENER\_SECURITY\_PROTOCOL\_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT\_HOST:PLAINTEXT

KAFKA\_ADVERTISED\_LISTENERS: PLAINTEXT://kafka:29092,PLAINTEXT\_HOST://localhost:9092

KAFKA\_OFFSETS\_TOPIC\_REPLICATION\_FACTOR: 1

KAFKA\_GROUP\_INITIAL\_REBALANCE\_DELAY\_MS: 0

KAFKA\_CONFLUENT\_LICENSE\_TOPIC\_REPLICATION\_FACTOR: 1

KAFKA\_CONFLUENT\_BALANCER\_TOPIC\_REPLICATION\_FACTOR: 1

KAFKA\_TRANSACTION\_STATE\_LOG\_MIN\_ISR: 1

KAFKA\_TRANSACTION\_STATE\_LOG\_REPLICATION\_FACTOR: 1

KAFKA\_JMX\_PORT: 9997

KAFKA\_JMX\_HOSTNAME: kafka

kafka-ui:

container\_name: kafka-ui

image: provectuslabs/kafka-ui:latest

ports:

- 8082:8080

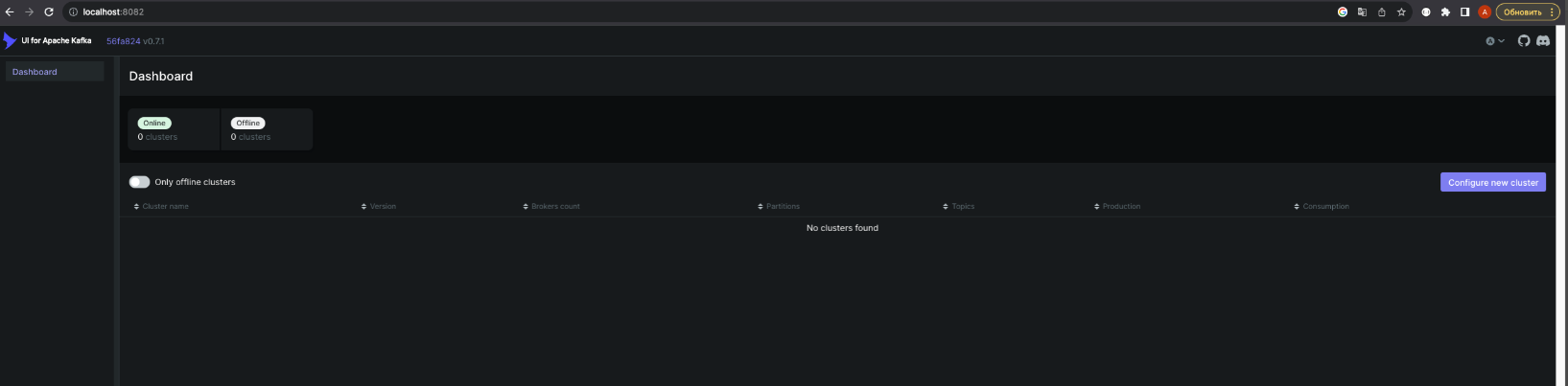
environment:

DYNAMIC\_CONFIG\_ENABLED: true

Здесь, как вы можете увидеть, мы указываем рабочий порт, образ (image), который Docker должен загрузить, а также переменную "DYNAMIC\_CONFIG\_ENABLED". Она нужна, что неудивительно, для возможности изменения конфигурации вашего kafka-ui в реальном времени. С полным списком переменных окружения вы можете ознакомиться, перейдя по предоставленной [ссылке](https://docs.kafka-ui.provectus.io/configuration/misc-configuration-properties).

**Подключение брокера к Kafka-UI**

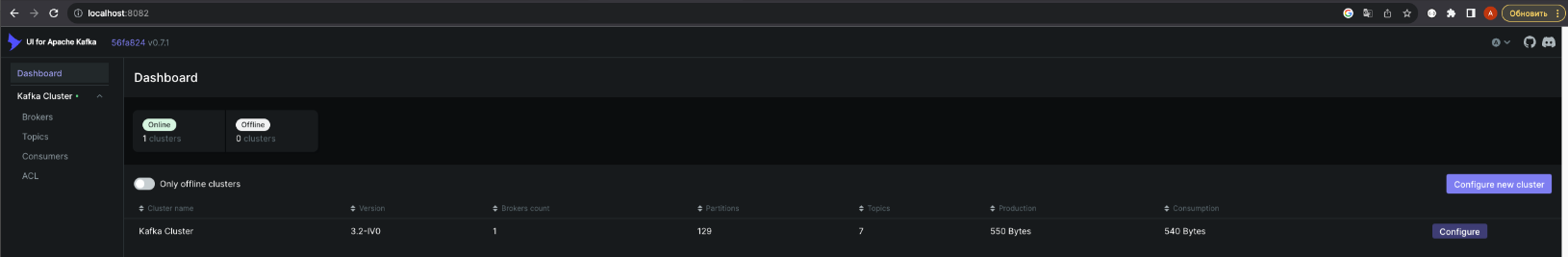
После успешного запуска вашего docker-compose файла, вам нужно перейти на заданный вами порт для kafka-ui и увидеть следующую картину:



Теперь давайте добавим нашу kafka в kafka-ui. Для этого давайте нажмем на кнопку **configure new cluster**где вам нужно заполнить следующие поля:

1. Cluster name - Можете указать просто как "Kafka Cluster"
2. Bootstrap Servers - Суда вам внужно вписать PLAINTEXT://kafka:29092 ну или другое наименование в зависимости от вашей конфигурации параметра "KAFKA\_ADVERTISED\_LISTENERS" в kafka image. Соотвественно если у вас поднято несколько реплик кафки, вам нужно их всех вписать. Apache рекомендуют иметь 3 ноды с kafka на вашем проекте.
3. Metrics
   1. metrics type -> JMX
   2. port -> 9997 ну или как вы указали в своей конфигурации

После этого ваш Kafka кластер должен отобразиться в вашем GUI. Выглядит это примерно так:

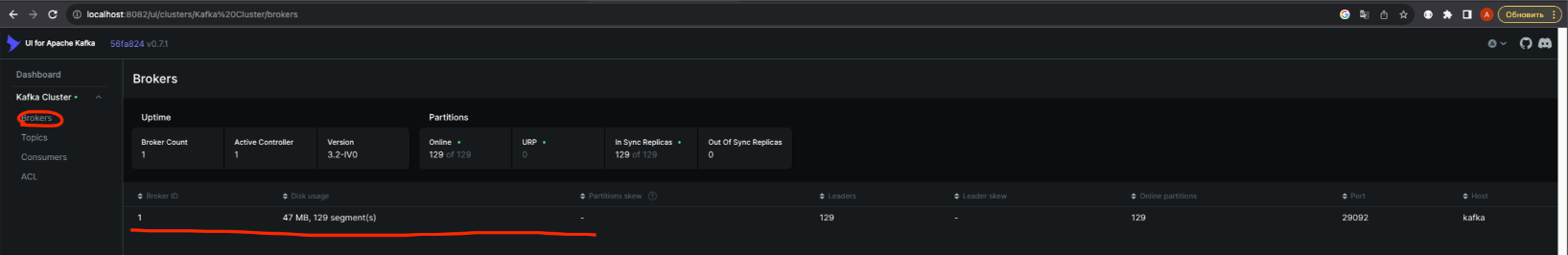


**Функционал**

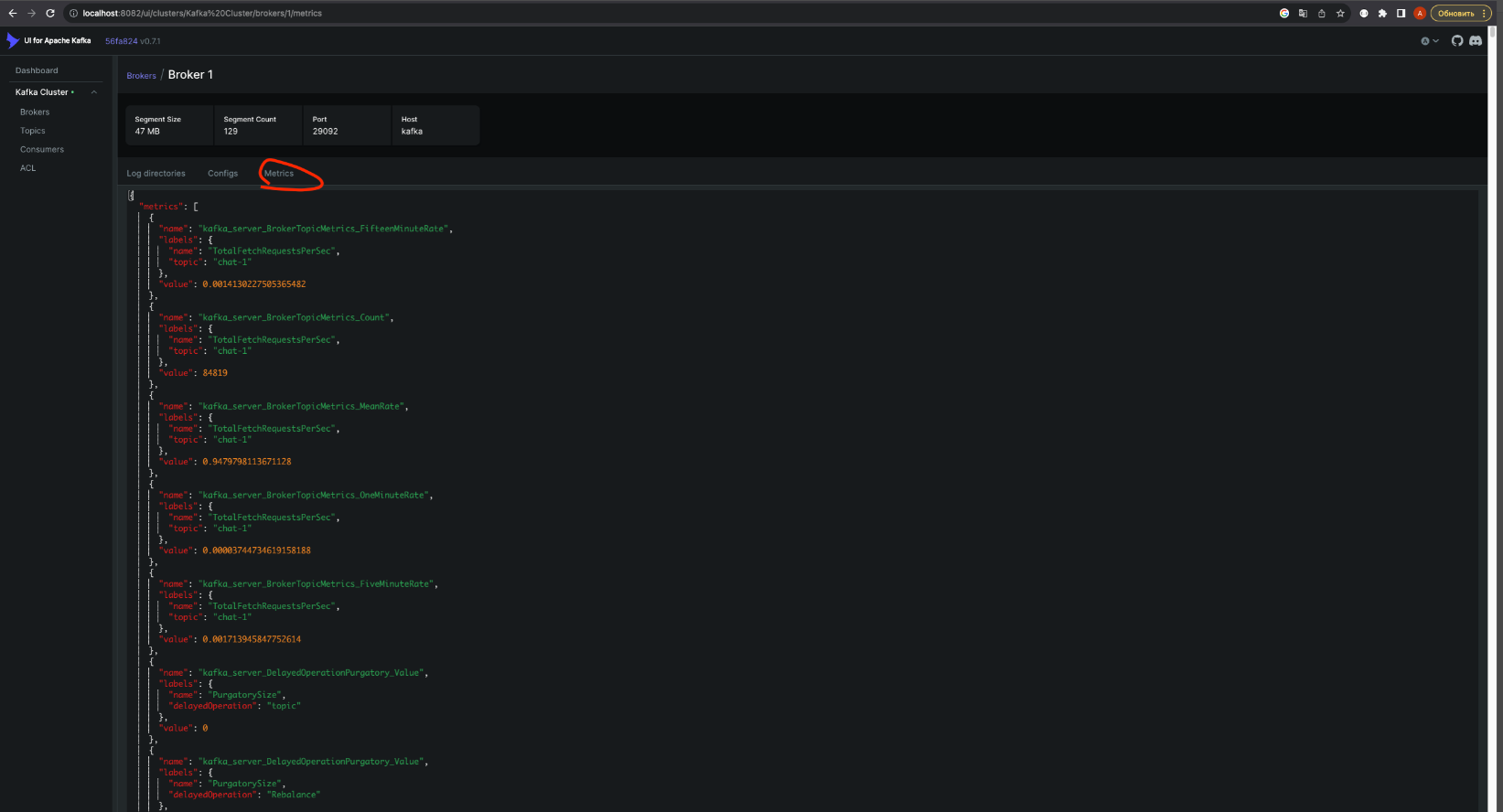
Самое важное в данном сервисе как по мне это функционал по просмотру топика внутри вашей кафки. Для этого нам нужно перейти во вкладку Topics и далее перейти во вкладку messages где мы сможем лицезреть полную информацию по каждому сообщению и фильтр для поиска специфичный сообщений.

Так же вы можете просматривать ваших consumer'ов и их группы во вкладке consumers.

Для просмотра конфигурации вам нужно перейти во вкладку Brokers и выбрать нужный вам брокер



Далее переходим во вкладку Metrics и просматриваем всю нужную нам информацию.



Итог

В процессе создания своего аналога сервисов Discord я столкнулся с необходимостью мониторить свою очередь сообщений в Kafka. Я искал удобный инструмент для этого, и благодаря совету моего коллеги Паши, DevOps инженера, я обратил внимание на kafka-ui от команды provectus. Этот интерфейс позволил мне легко отслеживать все сообщения в топиках без необходимости вручную входить в docker контейнер. Особо хочу отметить удобство и функциональность kafka-ui: просмотр сообщений в топиках, мониторинг consumer'ов и их групп, а также возможность просмотра конфигурации и метрик брокера. Используя этот инструмент, я значительно упростил процесс работы и диагностики своей системы. Если кто-то из вас работает с Kafka, рекомендую обратить внимание на kafka-ui!