# Ab testing semi-online pipeline

*Teacher: Professor* [*Peter B. Panfilov*](https://www.hse.ru/en/org/persons/14608534)*. HSE Graduate School of Business / Department of Business Informatics.*

*Student: Maxim Movshin, group 213.*

## About this project

Purpose of this project is to practice big data pipeline creation with online data processing.

Plot of this project is implementing an A/B testing system. Business requirements can be formulated as follows:

*We need create a way to check business hypotheses about our service. In order to do it we want to integrate a new feature testing instrument — A/B testing.*

*A/B testing*

*Each feature must be turned on only on a part of traffic. Then we log Information about each user session, performed product actions and experimental features that were turned on. At last we analyze this information: compare metric with and without experimental feature, and identifying whether the feature was useful or not.*

*Time is money and we want to analyze all the data that we collect as soon as it is possible.*

*There must be a UI friendly instrument for our analysts and even managers to work with data. This instrument must provide us a way to easily explore basic information about experiments in real time.*

## Technology stack and pipeline architecture



Each element of the system runs in separate docker container. There is a single docker compose file that allows to start system in just one command.

It all starts from a product. Python script emulates a service for users. It generates logs and passes it to the Apache Kafka brokers.

Apache Kafka stores data and then streams it to Apache Spark Structured Streaming server.

Spark streaming server performs some actions with data and saves it to ClickHouse database.

ClickHouse database simply stores data and gives a metabase aggregated information on request.

Metabase prepares requests to database, and requests data from clickhouse. Then it shows data in UI.

## Data processing architecture

Logs in webserver are created in almost unstructured way. It was done on purpose to simulate real life services.

Kafka is just storing and transmitting data further through pipeline.

Spark structured streaming instance is prettifying logs data, extracting most important information from it. It gets data about main user actions, such as purchases, and information about testids. Then it saves data to the clickhouse database.

Metabase then aggregates this primary processed data to show basic information in BI interface. This is info such as comparison of values of main metrics for each selection from experiment.

All of the above is performed in realtime (except metabase dashboard: its refresh take about a few seconds, and performs by clicking a button), so analysts can view main results in real time.

## A/B testing methodology and structure

First of all, it is important to understand that commonly A/B testing infrastructure requires a lot of work and business logic related to experiment versions distribution. There should be a separate service that returns a list of selections (roughly speaking flags about which logic to use) for a request with user session id. But this is, actually, not really related to the theme of a project (big data), so selections are distributed on a server itself.

Holding several experiments is a rough task. There are many ways to do it. For example, it is possible to distribute selection so that experiments can overlap. But the easiest way is to just make all experiments nonoverlapping. So in this example splitting task solved this way:

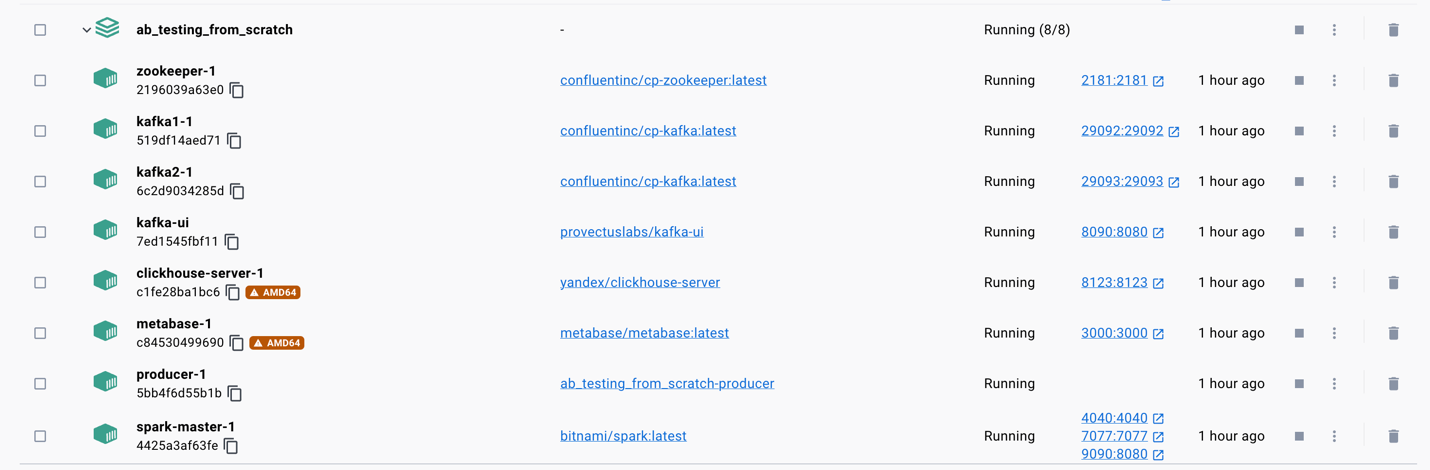
We just split traffic to hundred buckets. All traffic that goes to buckets 0-39 refers to the first experiment. 0-19 — control (no modifications), 20-39 — test (modified). All traffic that goes to buckets 40-79 goes to the second experiment. 40-59 — control (no modifications), 60-79 — test (modified). Buckets 80-99 is a clean production.

Real A/B testing uses complex statistics instruments to make metrics more accurate. But for simplicity we will just calculate average values of metrics per session in each group. It will be a way less accurate, and we will not be able to detect small changes between metrics. But we can deal with it by just making mock impact on user behavior huge.

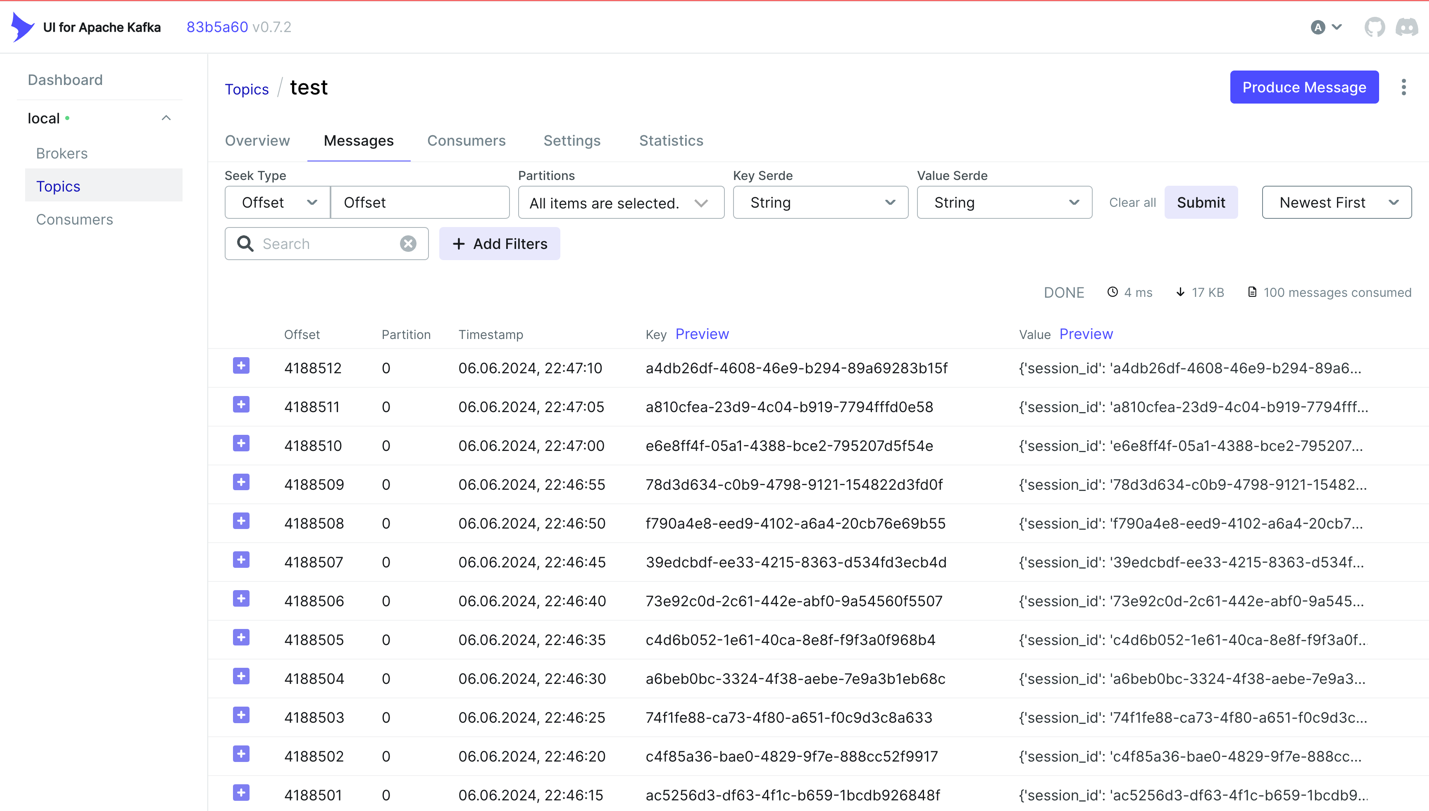
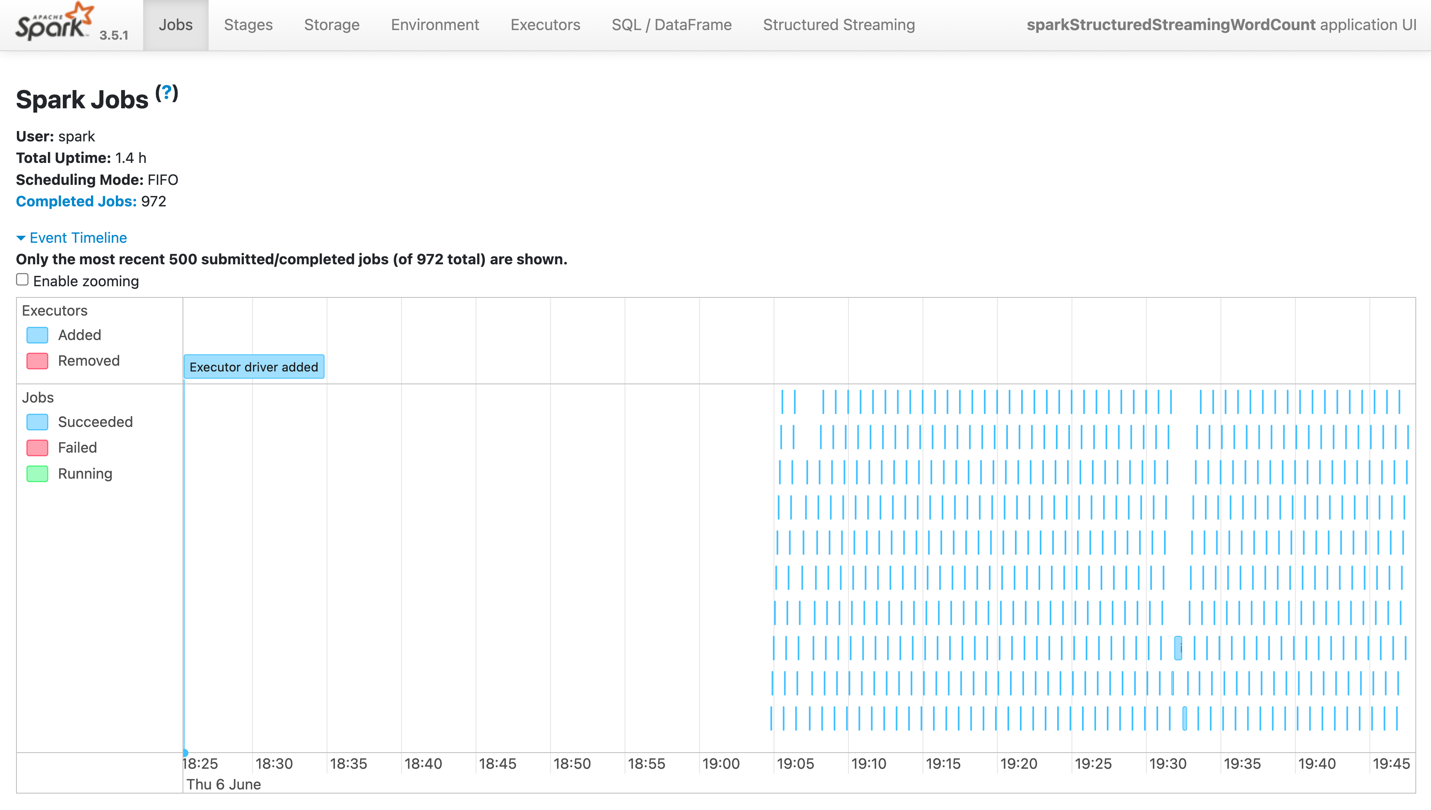
We will make two experiments. Both of them will try to increase main product metric — purchase probability. Default probability written in server code is 0.3. First experiment’s feature will increase it by 0.1, and second feature will decrease it by 0.15. We assume finding it on a dashboard.

## Presentation

Project is available on a github. Simply run docker compose up to start the pipeline.



How to check if everything goes okay?

* KafkaUI is available on <localhost:8090> . Shows kafka brokers health. Topics data can be listed. It is possible to look at new messages in a live mode.
* Spark UI is available on <localhost:4040>. Shows jobs graph and health state.
* Metabase is available on <localhost:3000>. This is a BI UI to investigate data.

Let us look on the result in Metabase

Not so accurate at first. Let’s generate more logs.



Horray! Exactly what we expected!

## Possible improvements and extensions

* All the systems could be deployed on different machines in cloud. It would be a good way to get familiar with k8s.
* Some of the subsystems doesn’t have several pods, so they are at risk. It would be good to add replication for the database and other stuff.
* In my mind there must be one more process around this pipeline. Semi-processed data that is stored in database have to be processed by regular DAG from airflow with a cluster of Apache Spark machines performing jobs. It can, for example, go deeper in logs or join them with anti-fraud tables. Metabase will have additional dashboards and graphs based on this deep information. So analysts will have a dashboard of main data updated in runtime and deeper information updated a few times a day.