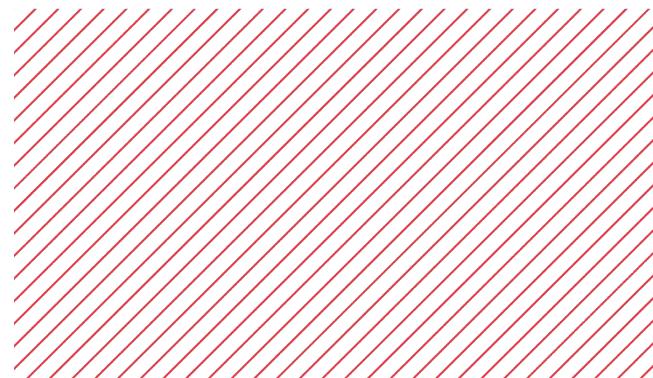


академия
больших
данных

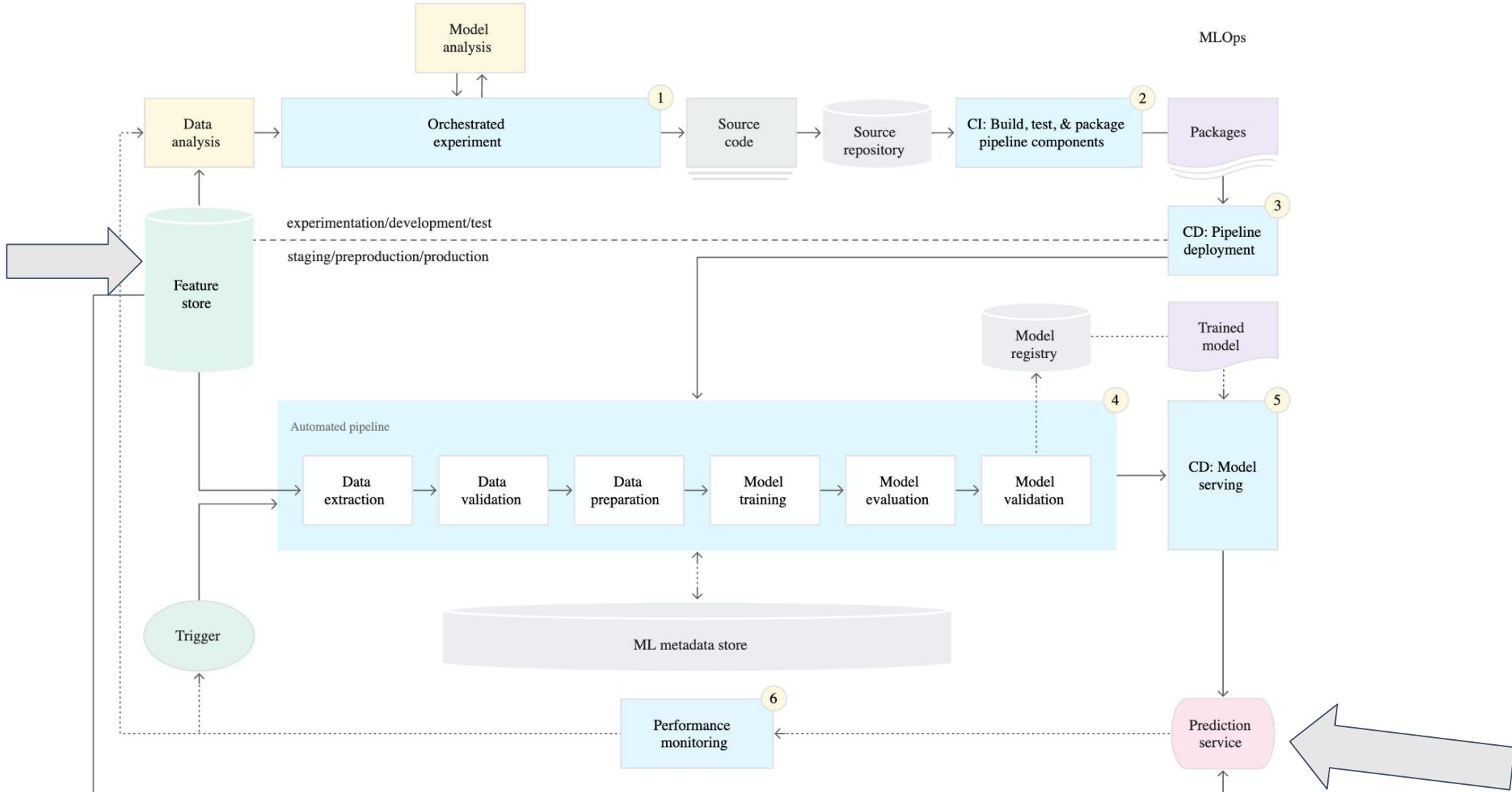


Очереди сообщений и Feast

Михаил Марюфич, MLE



О чём мы сегодня?



Удаление спам сообщений в Instagram



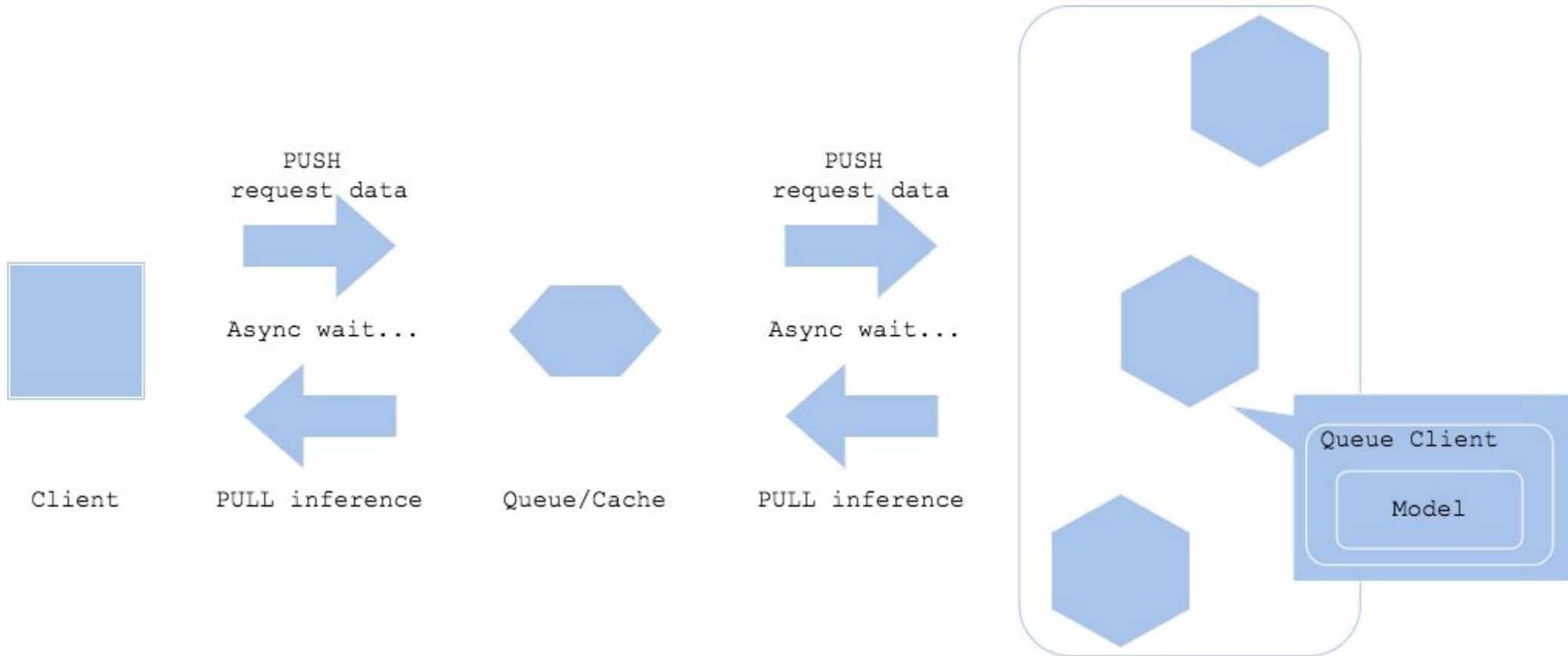
pavelvolyaofficial • Подписаться

irina_almazova_proff Классные вы ❤️
_777777755 Шоколад и мармелад ❤️
anikeeva600 Вы классные!!! Любите
друг друга.... И будьте счастливы!!!
iiodas ❤️
sweetypresentbouquet Очень
красивая пара 😊 ❤️
nastyawol ❤️❤️❤️⭐⭐⭐
marinavictory ❤️
cornershop_777 🔥🔥🔥
expressbeautykiev Привет!
Подпишитесь на нашу страницу и Вы
окунетесь в атмосферу
совершенства, красоты и
привлекательности. 💁‍♀️蝶

master_school_socchi СЧАСТЬЯ ВАМ
РЕБЯТА ❤️❤️❤️

273 933 отметок "Нравится"
17 ЯНВАРЯ
Добавьте комментарий... ⋮

Асинхронный(near realtime) паттерн



https://github.com/mercari/ml-system-design-pattern/blob/master/Serving-patterns/Asynchronous-pattern/design_en.md

Зачем нужны очереди сообщений?

Очередь сообщений

Компонент распределенной системы, обеспечивающий асинхронный обмен сообщениями



Примеры

- Kafka
- RabbitMQ
- NATS
- Tarantool





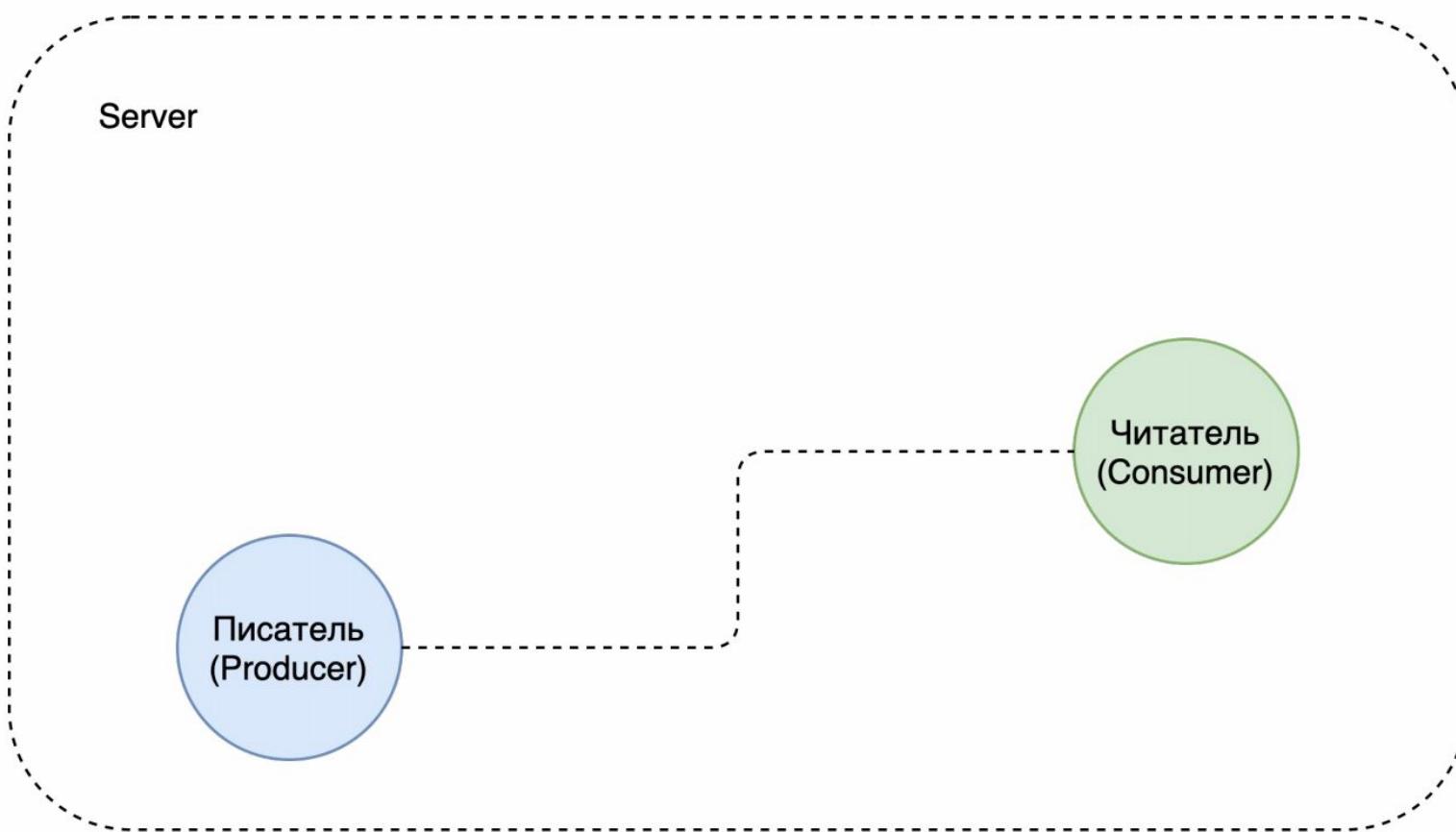
ФУНКЦИИ

Асинхронная/распределенная
обработка
Роутинг сообщений

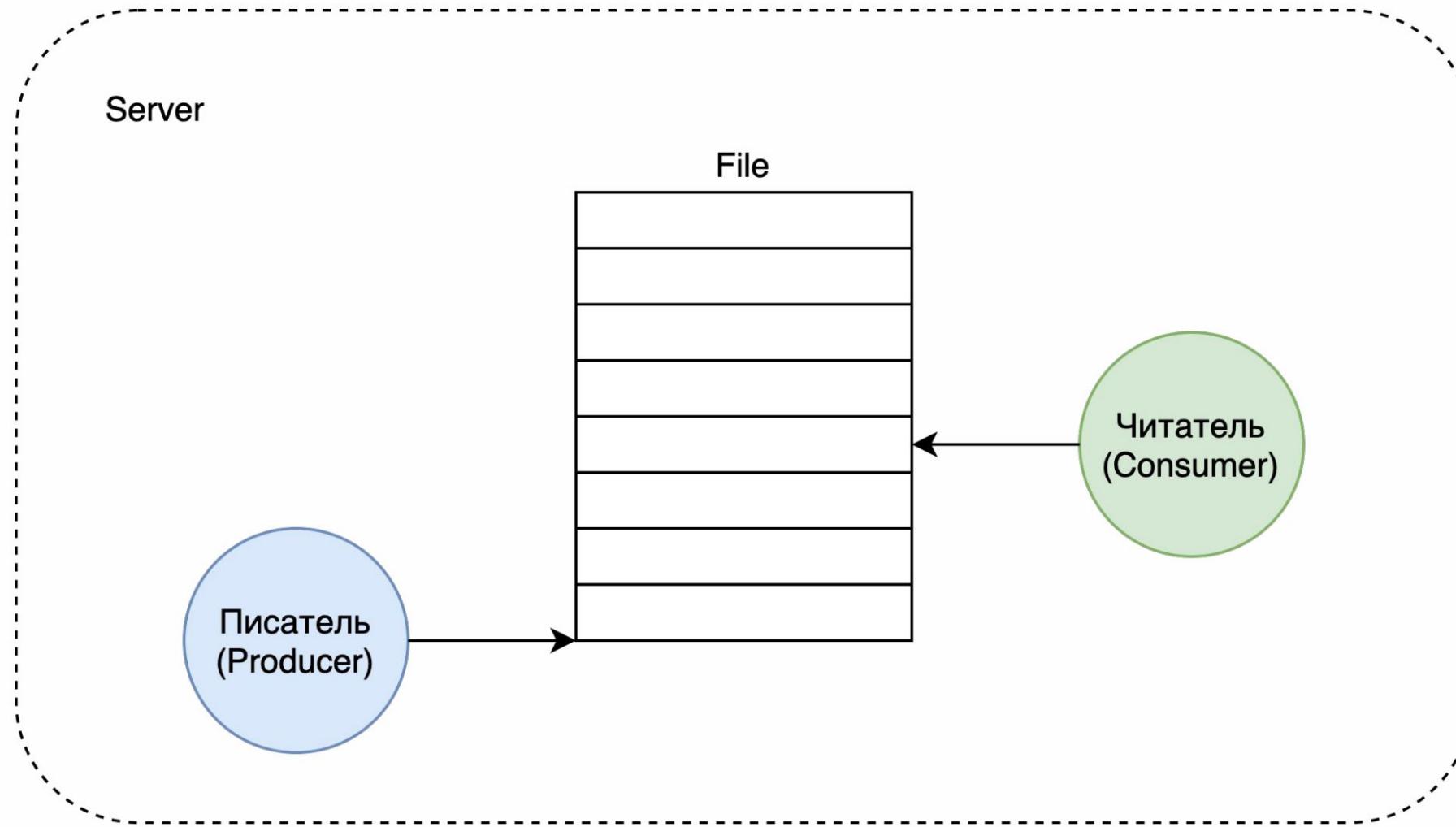
Что такое APACHE KAFKA?

Kafka — простыми словами

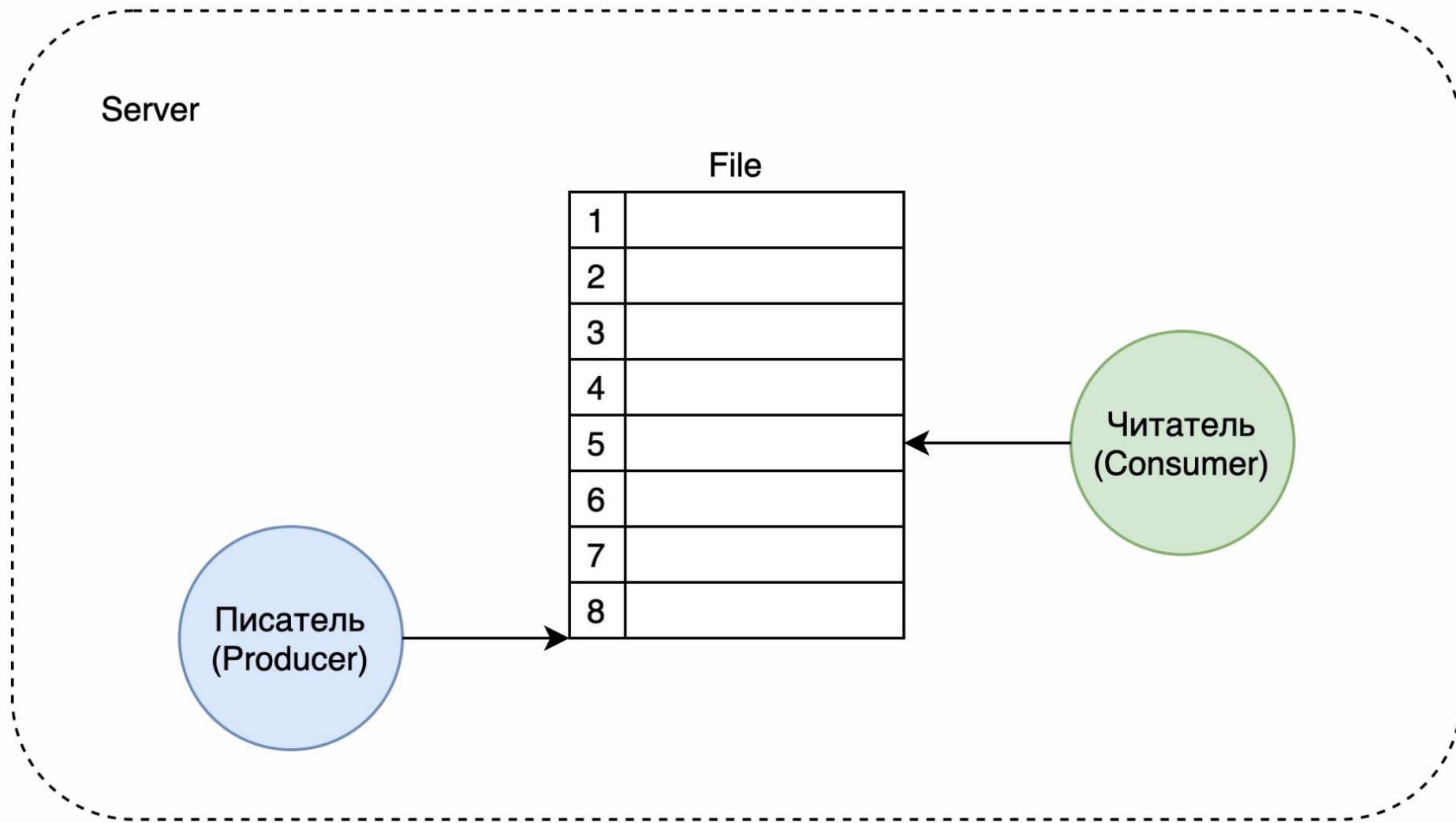
Из одного приложения нужно передать в другое сообщение.



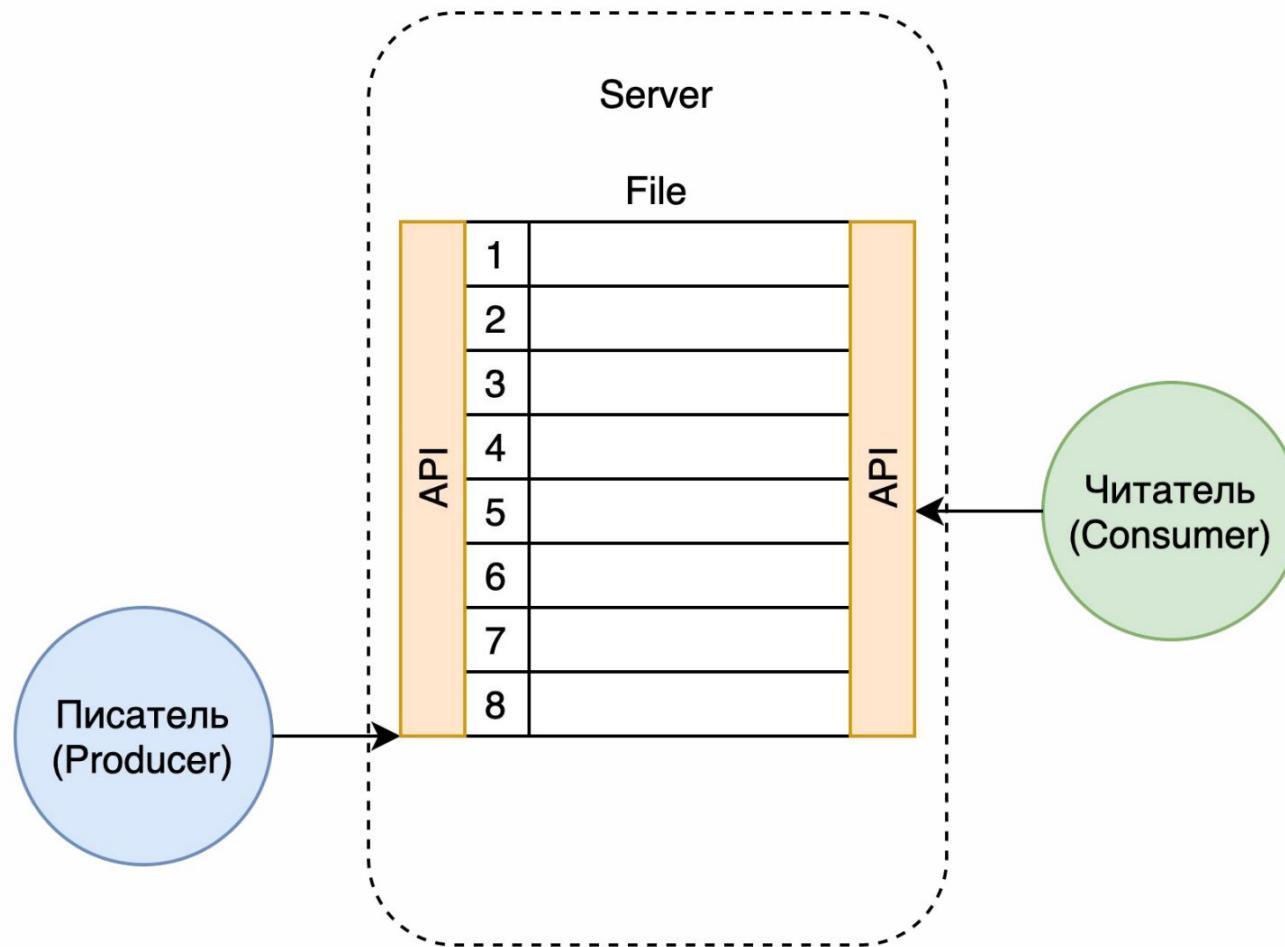
Kafka — простыми словами



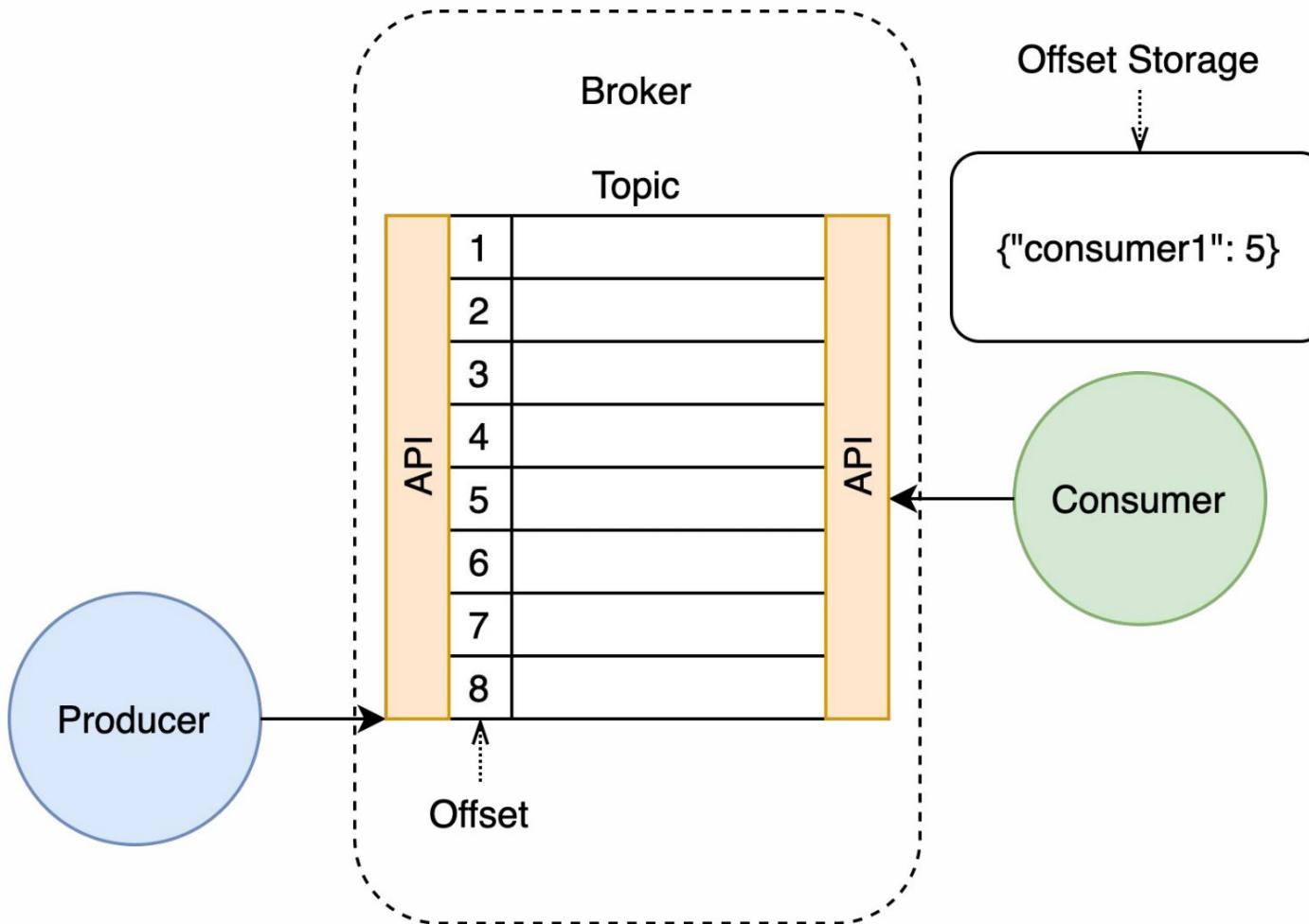
Kafka — простыми словами



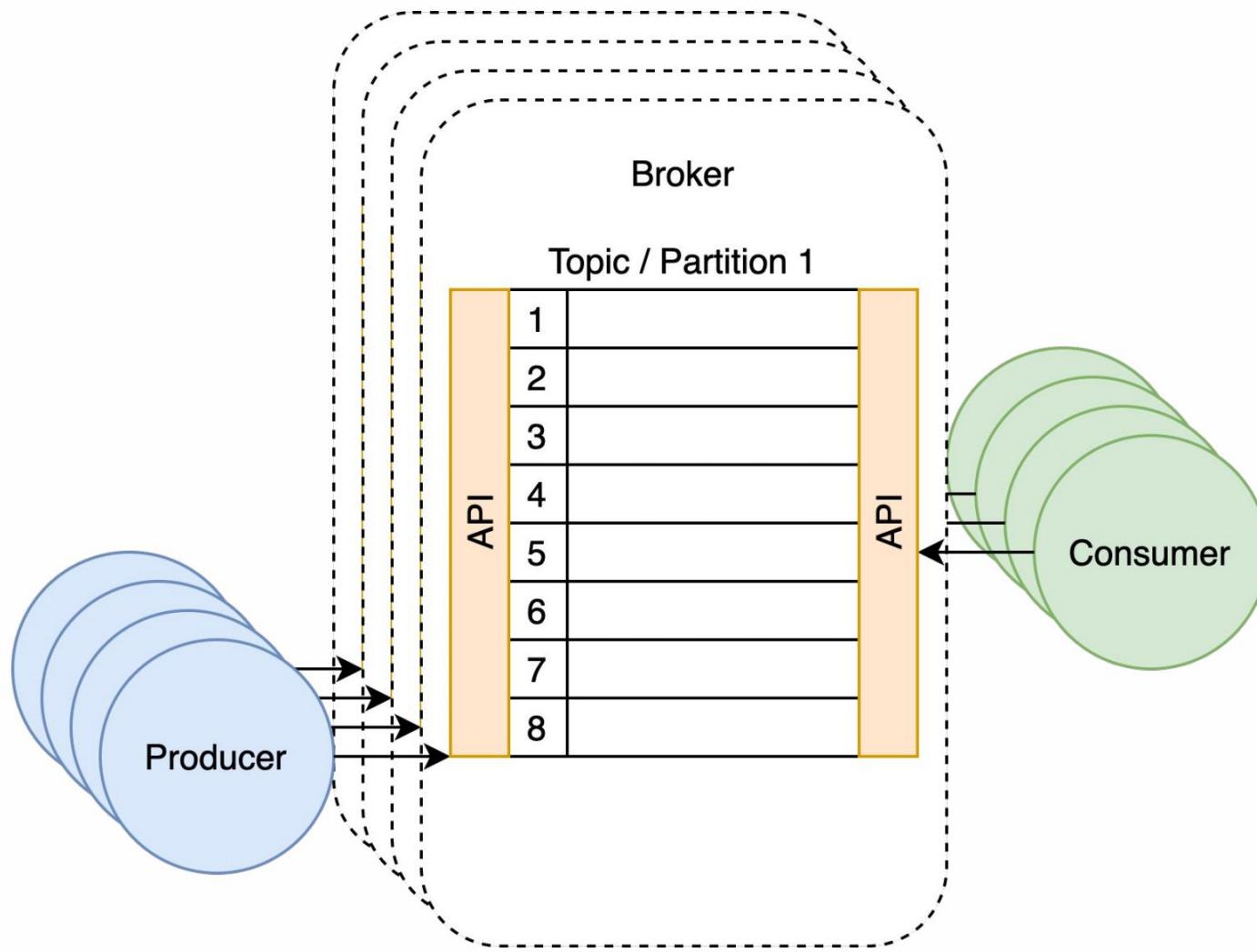
Kafka — простыми словами



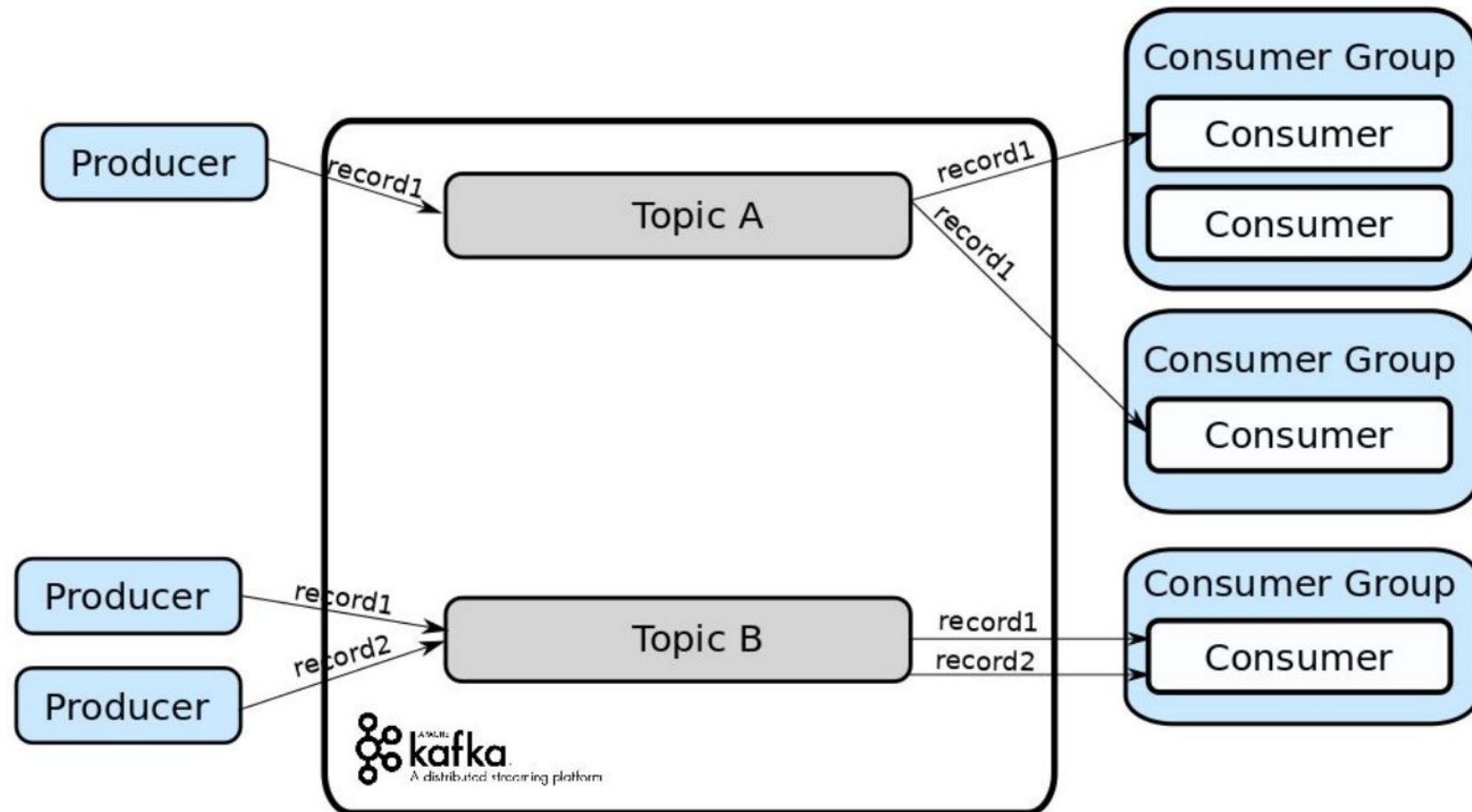
Kafka — простыми словами



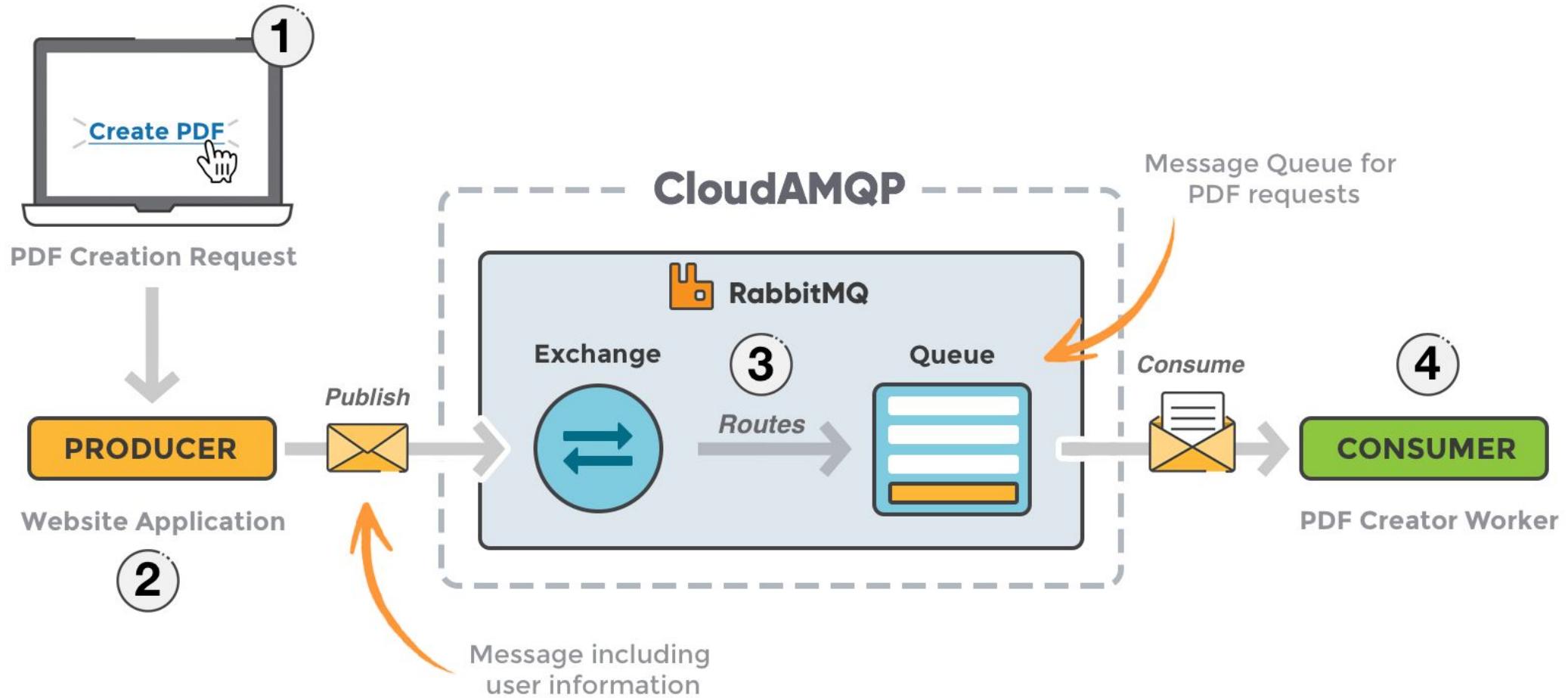
Kafka — простыми словами



Архитектура Kafka

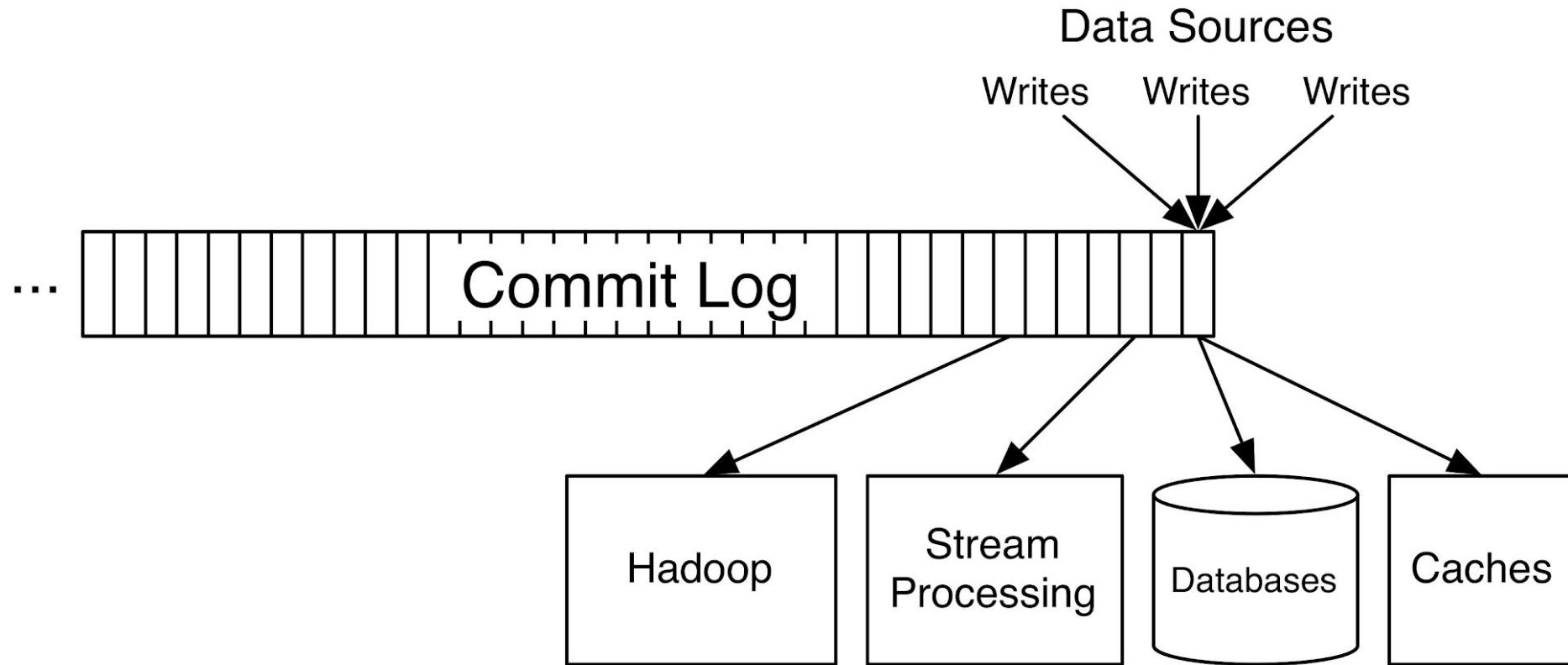


Архитектура RabbitMQ



Commit log

Kafka — это распределенный горизонтально масштабируемый отказоустойчивый журнал коммитов.





Достоинства commit-log

Простота использования

Высокая надежность

Высокая пропускная способность

Kafka — термины

Record – элемент данных

Topic – имя потока с данными

Producer – процесс, публикующий записи

Consumer – процесс, читающий записи

Offset – позиция записи

Partition – единица параллелизма темы

Broker – один сервер, обслуживающий несколько партиций

Cluster – множество broker



Еще раз — достоинства очередей сообщений

Буффер для нагрузки — можно налить данных на большой скорость и влиять на скорость разгребания

Абстракция — читатели мало что знают о писателях, связка по формату данных

Репликация — при выпадении компонентов, данные будут продублированы в kafka, можно их перезачитать
(Сами данные внутри kafka тоже реплицируются)



Взаимодействие с Kafka

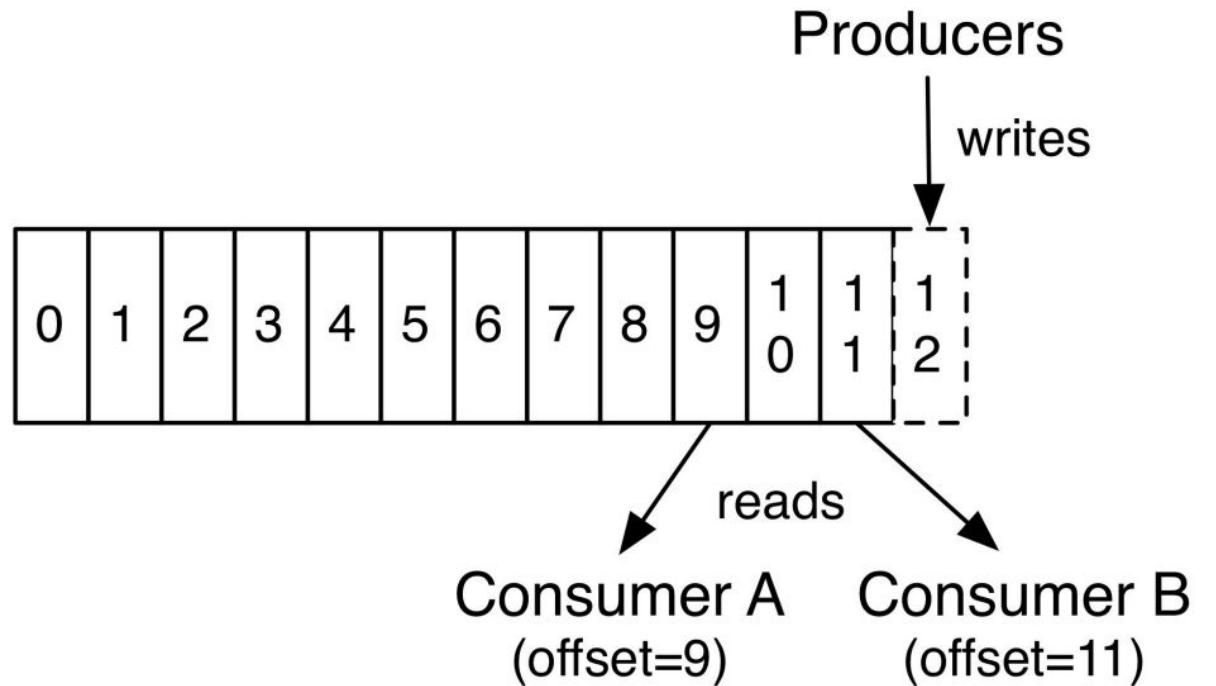
Каждый **record** — пара ключ-значение $\langle K, V \rangle$

И ключ и значение — произвольный набор байт

Каждое сообщение имеет timestamp

Взаимодействие с Kafka

Kafka гарантирует, что все сообщения в пределах **partition** будут упорядочены именно в той последовательности, в которой поступили



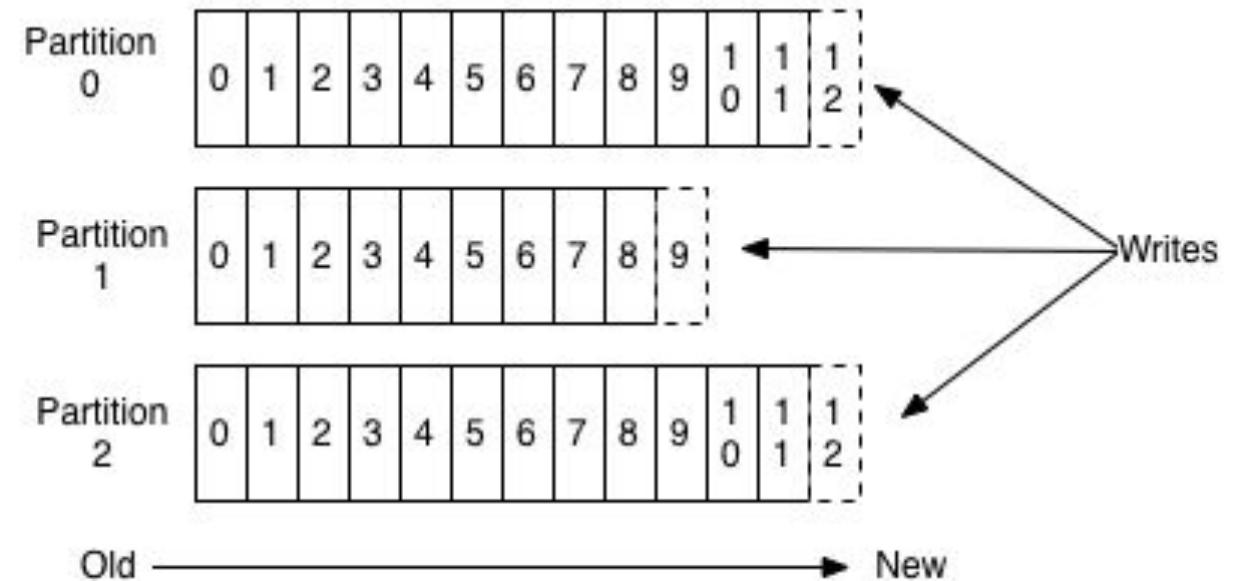
Producer

Producer (которых может быть несколько) может распределять сообщения по партициям.

2 стратегии:

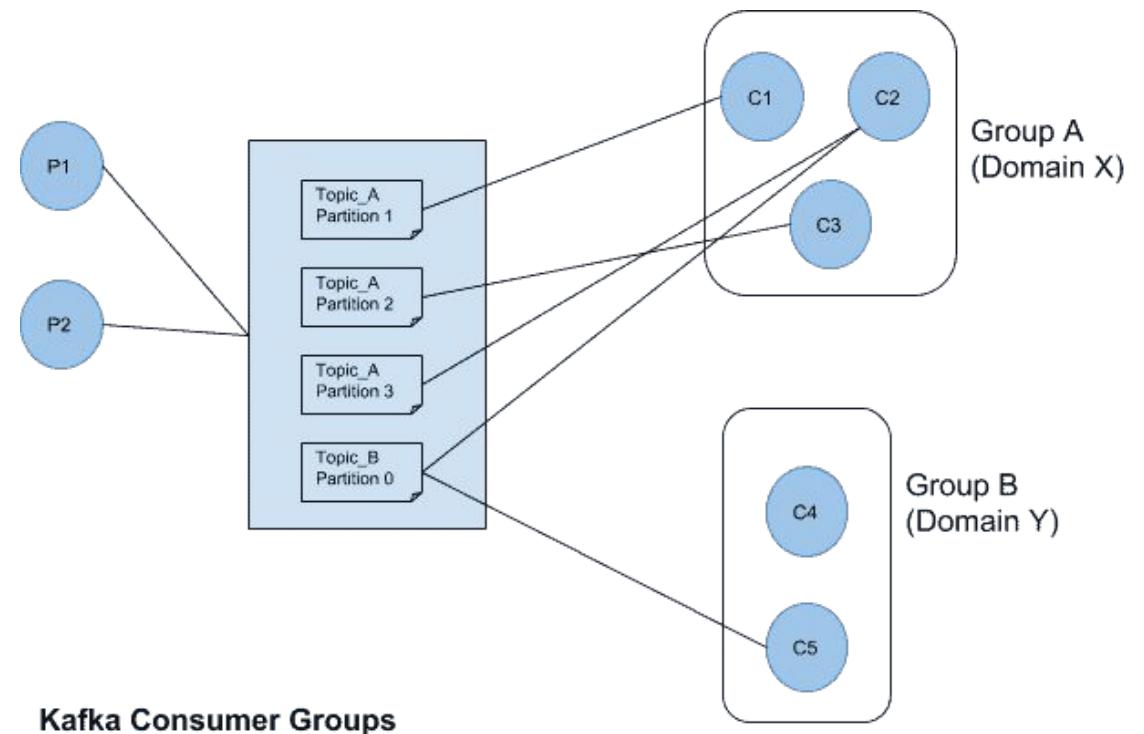
- round robin
- партиционирование на основе значения ключа

Anatomy of a Topic



Consumer

Consumer может быть несколькими потоками / приложениями, которые объединены group_id

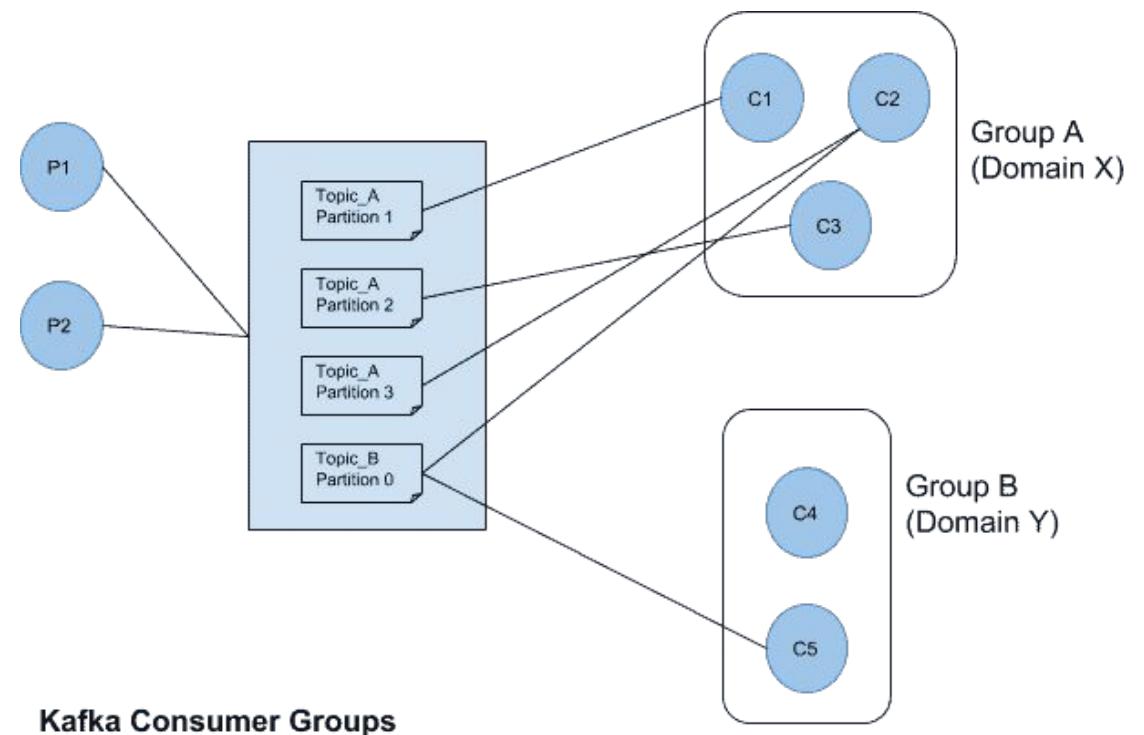


Consumer

Consumer внутри одной группы не читают из одной партиции.

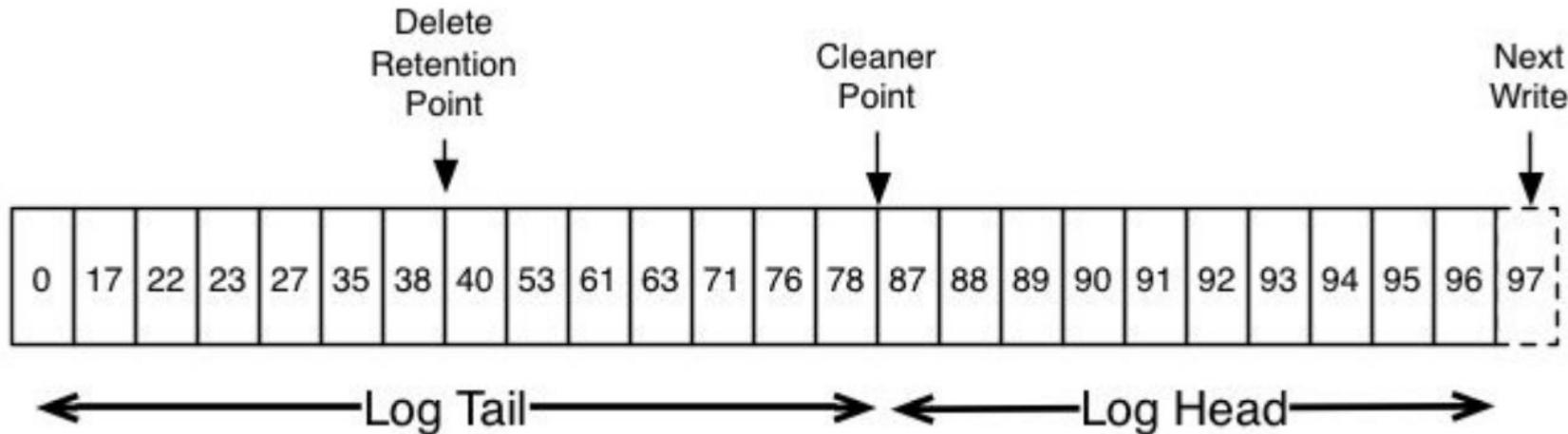
Consumer может читать из нескольких партиций.

Иначе одно и то же сообщение могут прочесть разные consumer



Dumb broker - smart consumer

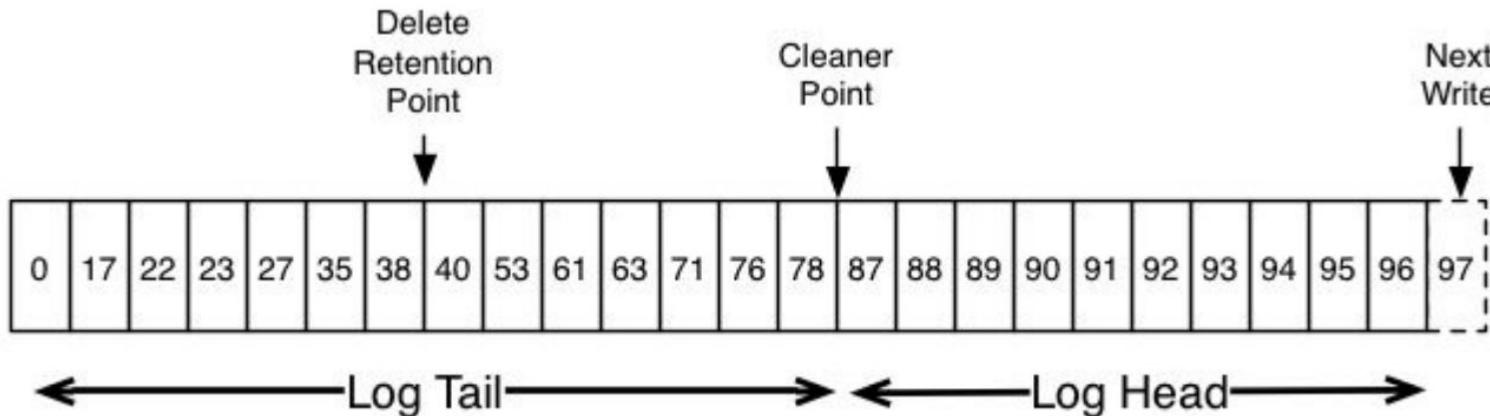
Kafka не отслеживает, какие записи считываются потребителем и после этого удаляются,



Log retention

У каждого топика есть retentions period — время после которого сообщение удаляется.

Кроме того, есть стратегия compaction, которая позволяет оставлять последнее значение для каждого ключа



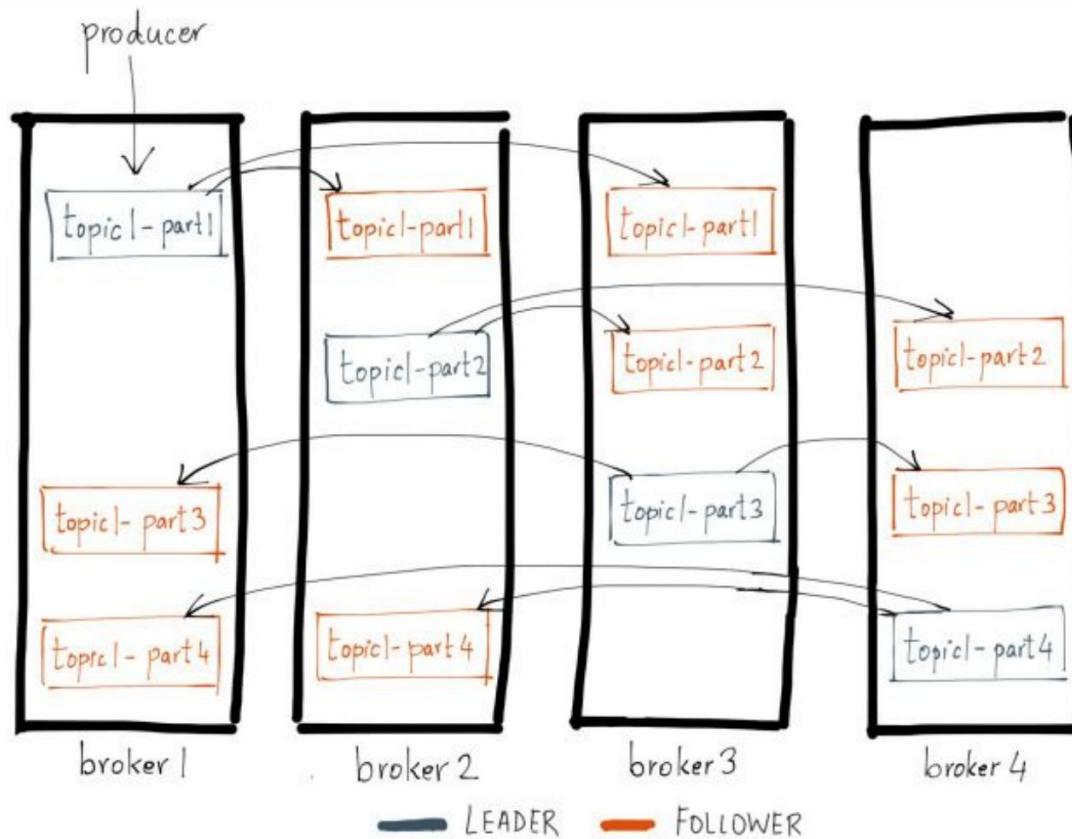


Compression

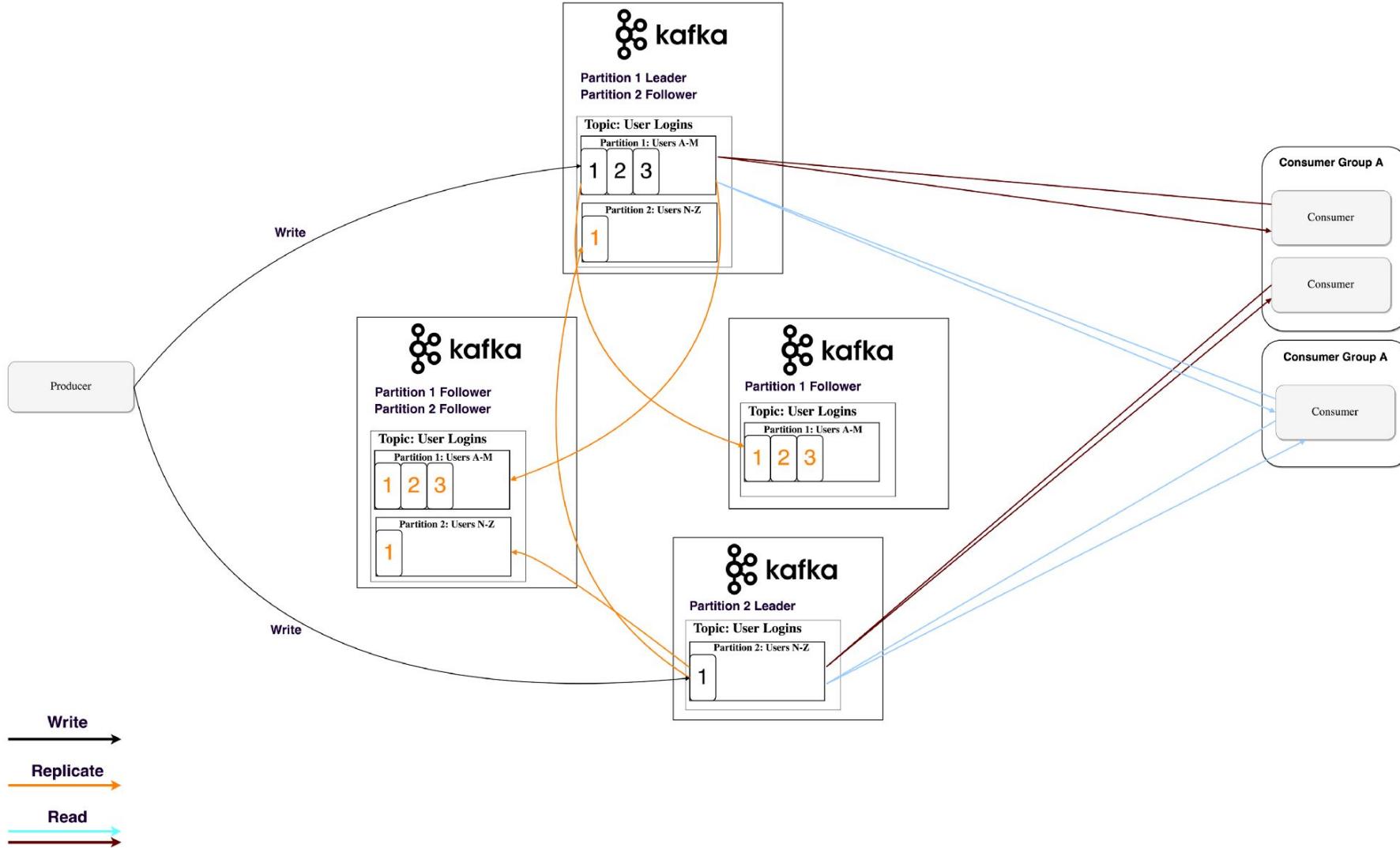
- no compression
- gzip
- snappy
- lz4
- zstd

Replication

Данные в **Kafka** дублируются по партициям



Replication

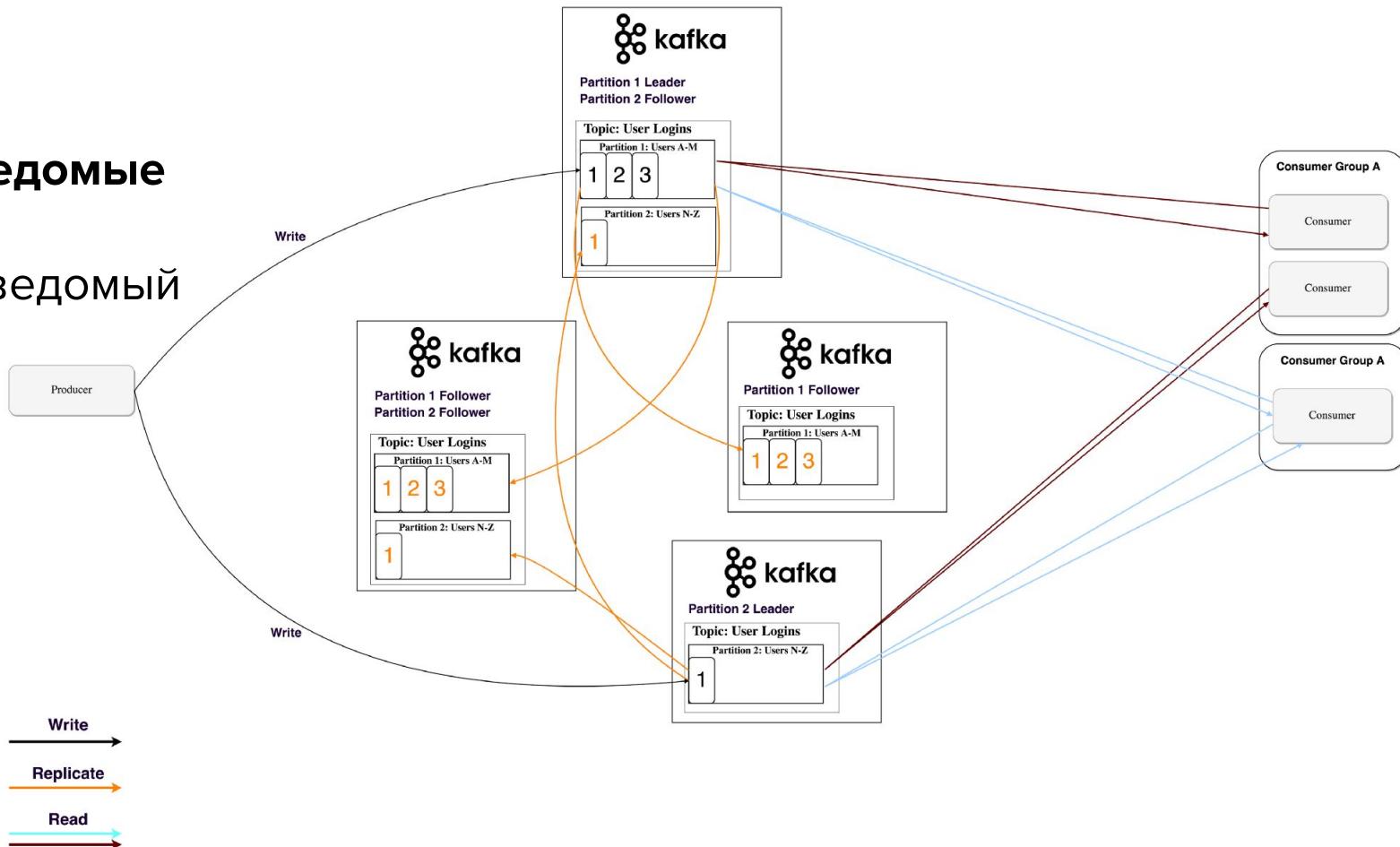


Replication

У нас есть несколько **брокеров**, один из них
лидер,
остальные **ведомые**

Лидер дублирует данные на **ведомые**

Если **ведущий** отказывает, то ведомый
становится **лидером**

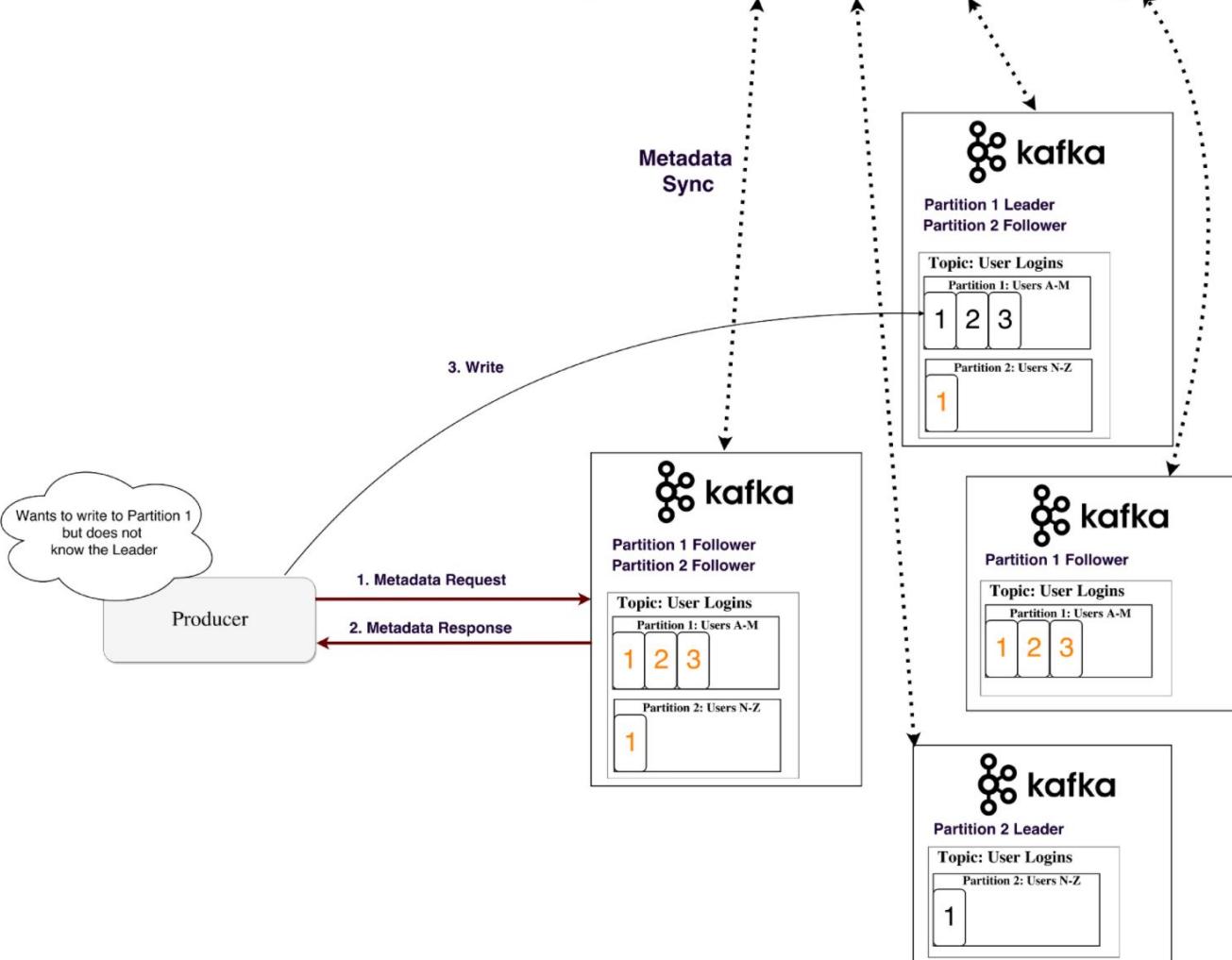
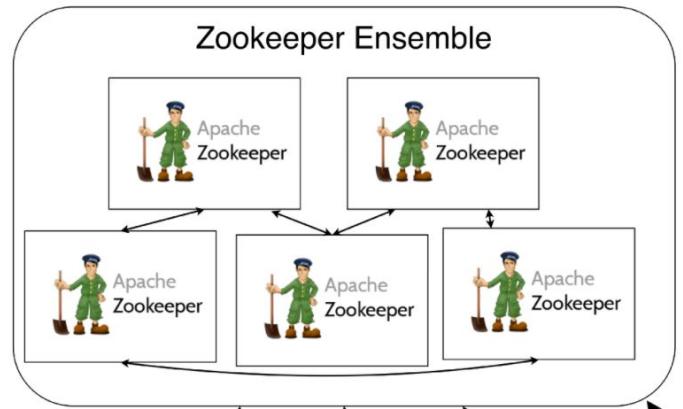


Zookeeper

Используется в **kafka** для
Хранения информации:

- Смещения групп потребителей в рамках секции
- **Ведущих** партиций

Выбора лидера



ДЕМО
Поднимаем kafka
Создаем топики
Читаем, пишем

<https://kafka.apache.org/quickstart>



Confluent Platform

- Панель управления
- Schema Registry
- Kafka Connect
- Kafka Streams
- ksqlDB

Control Center

The screenshot shows the Confluent Control Center interface. At the top left is the Confluent logo. Below it, the navigation bar includes 'HOME' and 'CONTROLCENTER.CLUSTER'. A sidebar on the left lists cluster components: Brokers, Topics, Connect, ksqlDB, Consumers, Replicators, and Cluster settings. The 'Cluster overview' tab is selected, highlighted in light blue.

Overview

Brokers

1	2.44K		87
Total	Production (bytes / second)		Consumption (bytes / second)

Topics

58	154	0	0
Total	Partitions	Under replicated partitions	Out-of-sync replicas

Connect

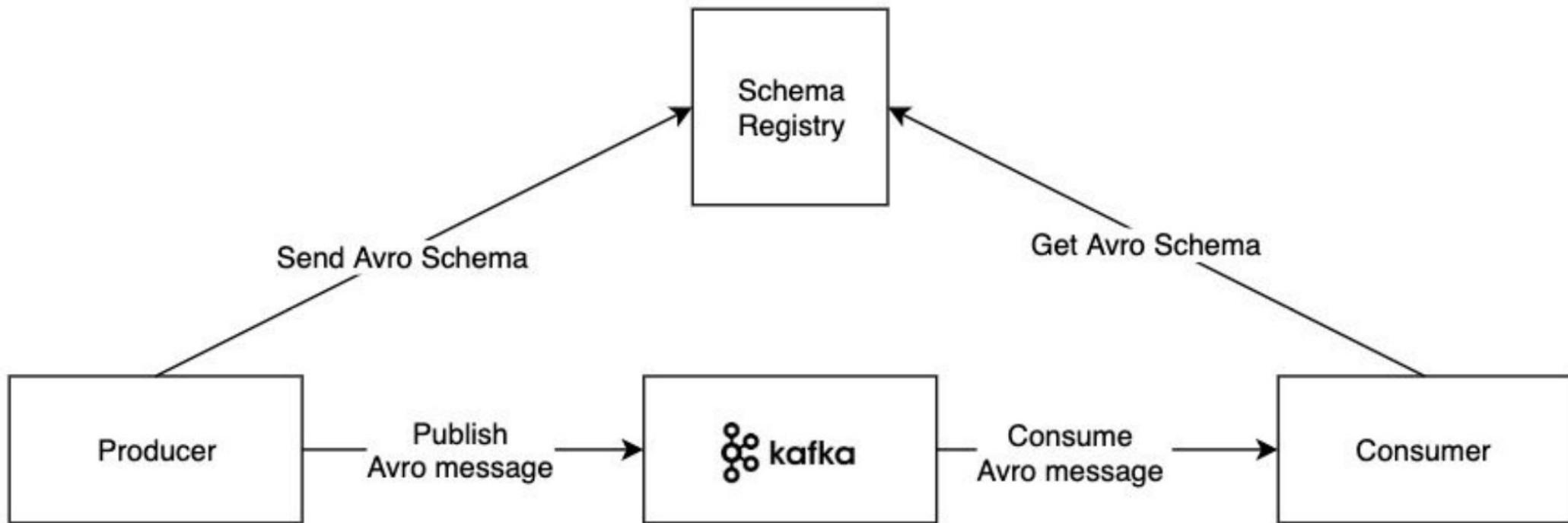
0	0	0	0	0
Clusters	Running	Paused	Degraded	Failed

Avro

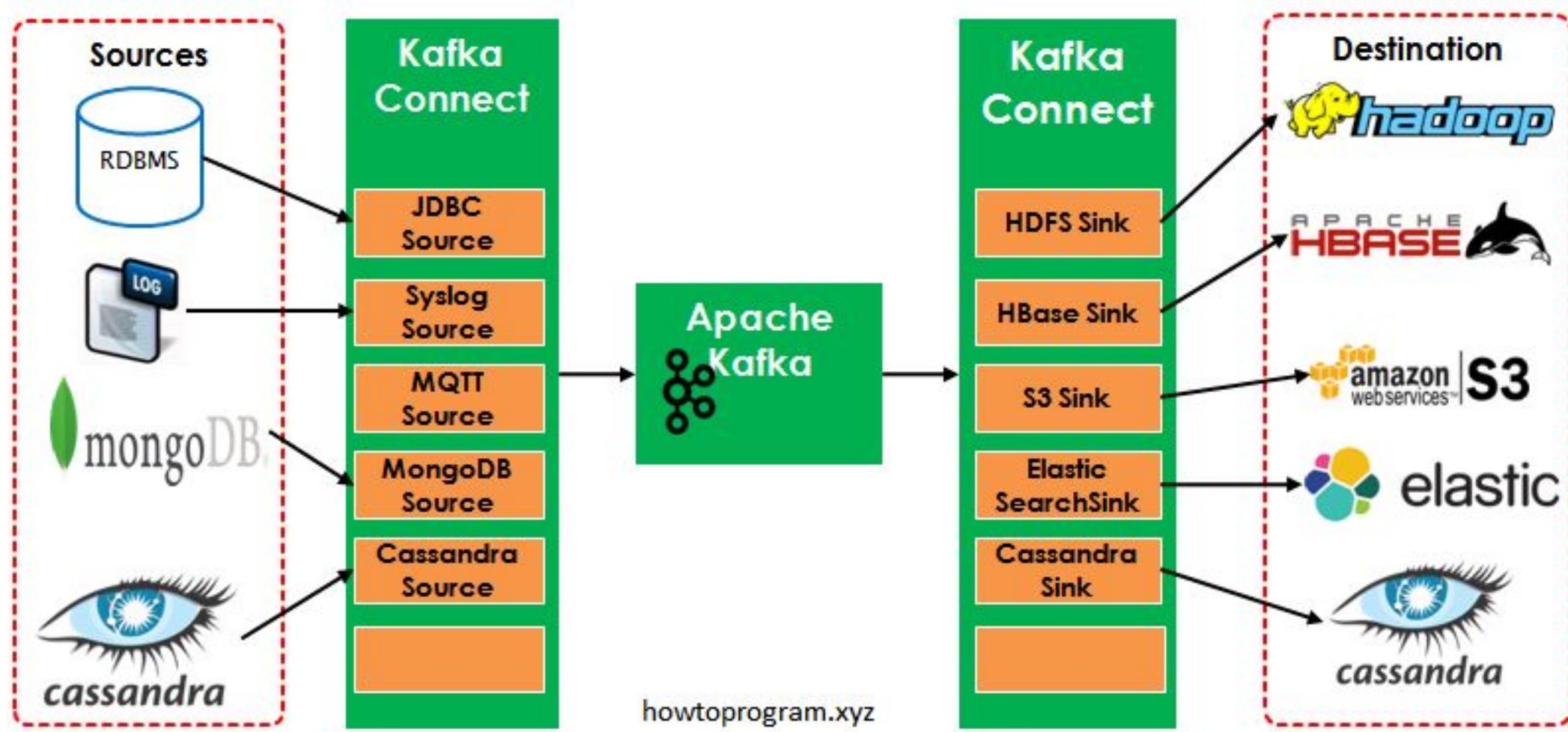
Бинарный протокол, описывающий формат
данных в схемах

```
{  
    "type": "record",  
    "name": "LongList",  
    "fields" : [  
        {"name": "value", "type": "long"},  
        {"name": "next", "type": ["null", "LongList"]}  
    ]  
}
```

Schema Registry



Kafka Connect



```
background-color: #F5F5F5;
text-shadow: 0px -1px 0px #EAEAEA;
filter: dropshadow(color:#EAEAEA);
color:#777;

}

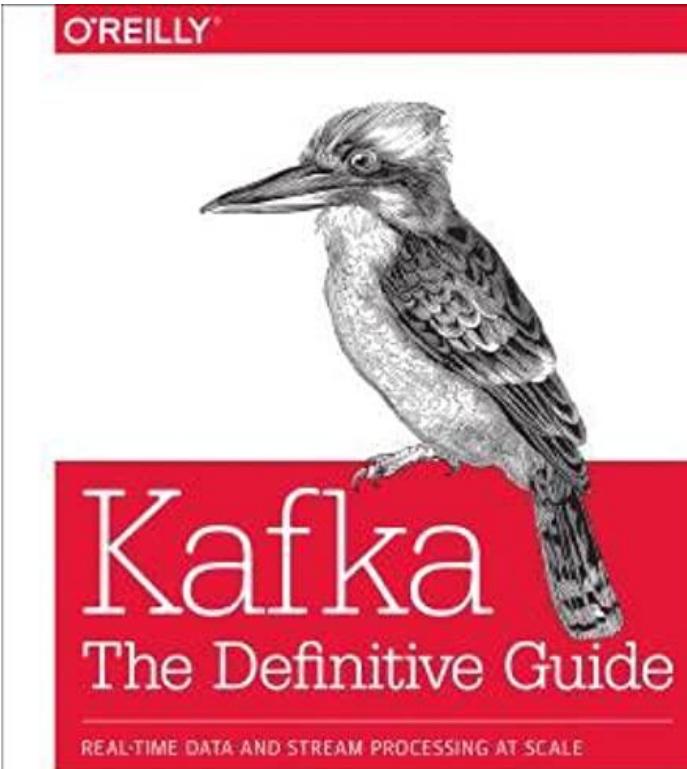
header #main-navigation ul li span:hover,
header #main-navigation ul li span:active {
  border-bottom: 1px solid #F5F5F5;
  background-color: #F5F5F5;
  box-shadow: 0px 0px 1px #EAEAEA;
  -webkit-box-shadow: 0px 0px 2px #EAEAEA;
  -moz-box-shadow: 0px 0px 1px #EAEAEA;
  color:#777;
}
```

DEMO

Смотрим confluent platform Producer, consumer на питоне

<https://docs.confluent.io/clients-confluent-kafka-python/current/overview.html>

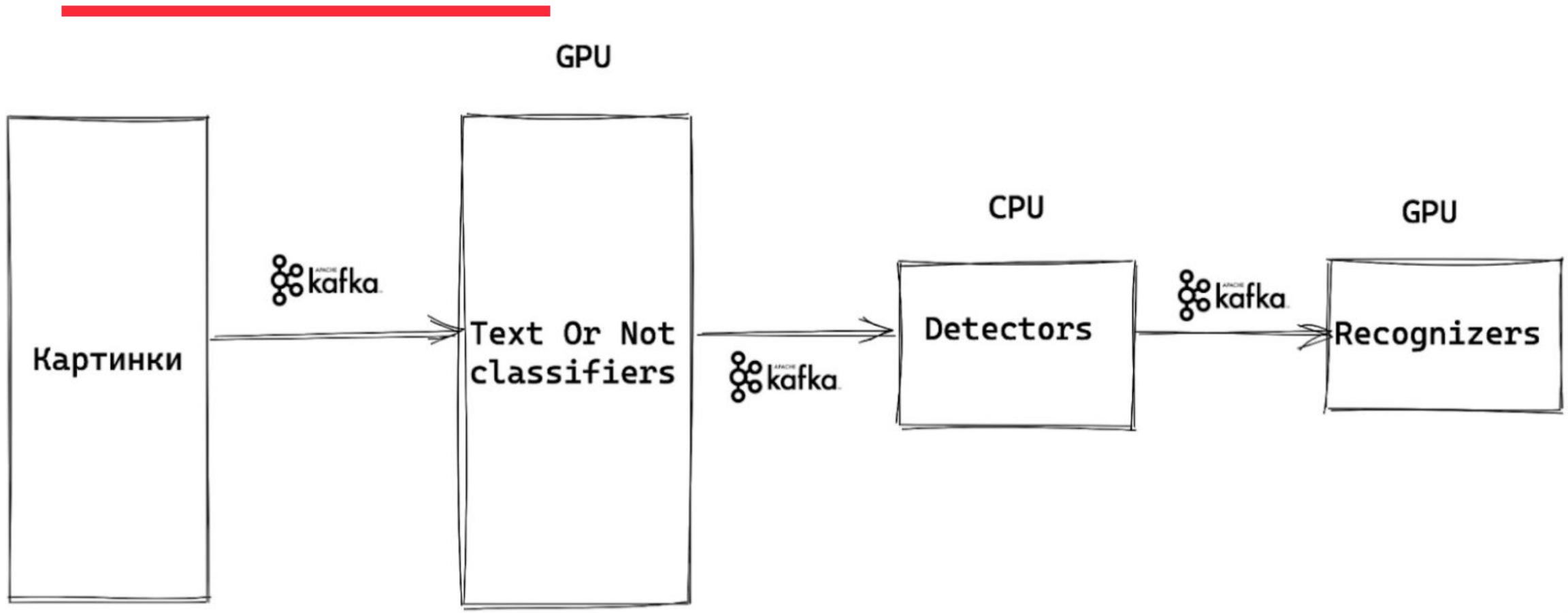
Дополнительно про Kafka



Neha Narkhede,
Gwen Shapira & Todd Palino

<https://habr.com/ru/company/piter/blog/352978/>

Распознавание картинок в ОК



Высокоуровневая архитектура системы распознавания текста с картинки

Feature Stores

Удаление спам сообщений в Instagram



A couple wearing sunglasses and a straw hat are smiling on a beach. The woman has red nail polish. The man is wearing a white shirt.

pavelvolyaofficial • Подписаться

irina_almazova_proff Классные вы ❤️
_777777755 Шоколад и мармелад ❤️
anikeeva600 Вы классные!!! Любите друг друга.... И будьте счастливы!!!
iiodas ❤️
sweetypresentbouquet Очень красивая пара 😊 ❤️
nastyawol ❤️❤️❤️⭐⭐
marinavictory ❤️
cornershop_777 🔥🔥
expressbeautykiev Привет!
Подпишитесь на нашу страницу и Вы окунетесь в атмосферу совершенства, красоты и привлекательности. 🌸😊🦋
master_school_socchi СЧАСТЬЯ ВАМ РЕБЯТА ❤️❤️❤️

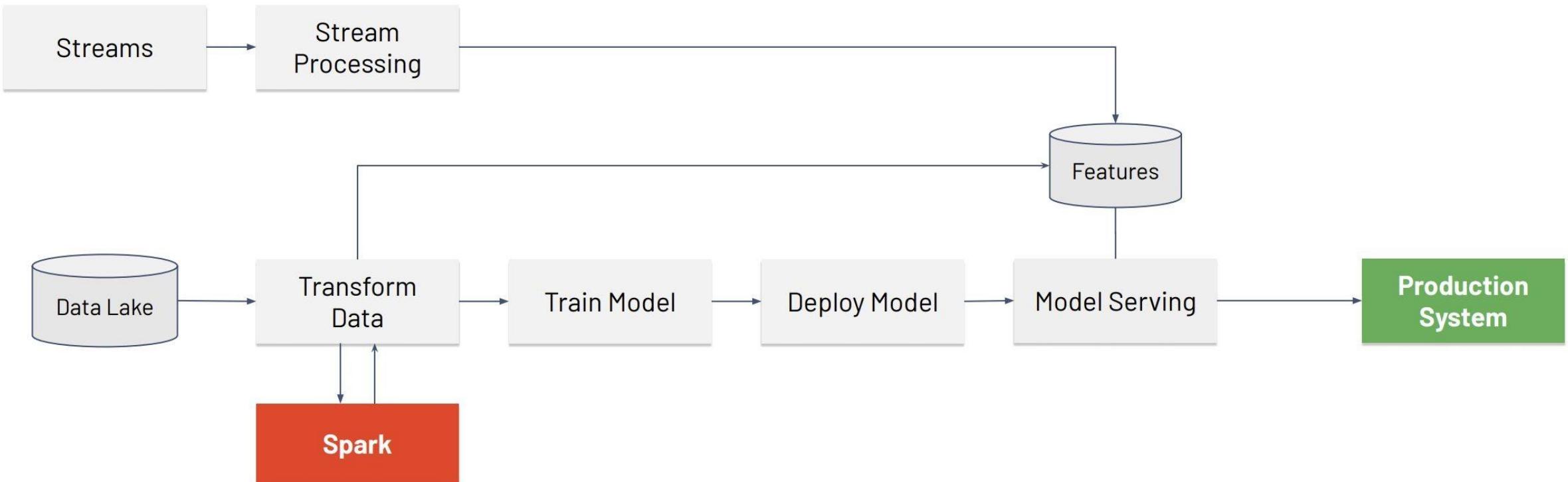
273 933 отметок "Нравится"
17 ЯНВАРЯ
Добавьте комментарий... ⋮



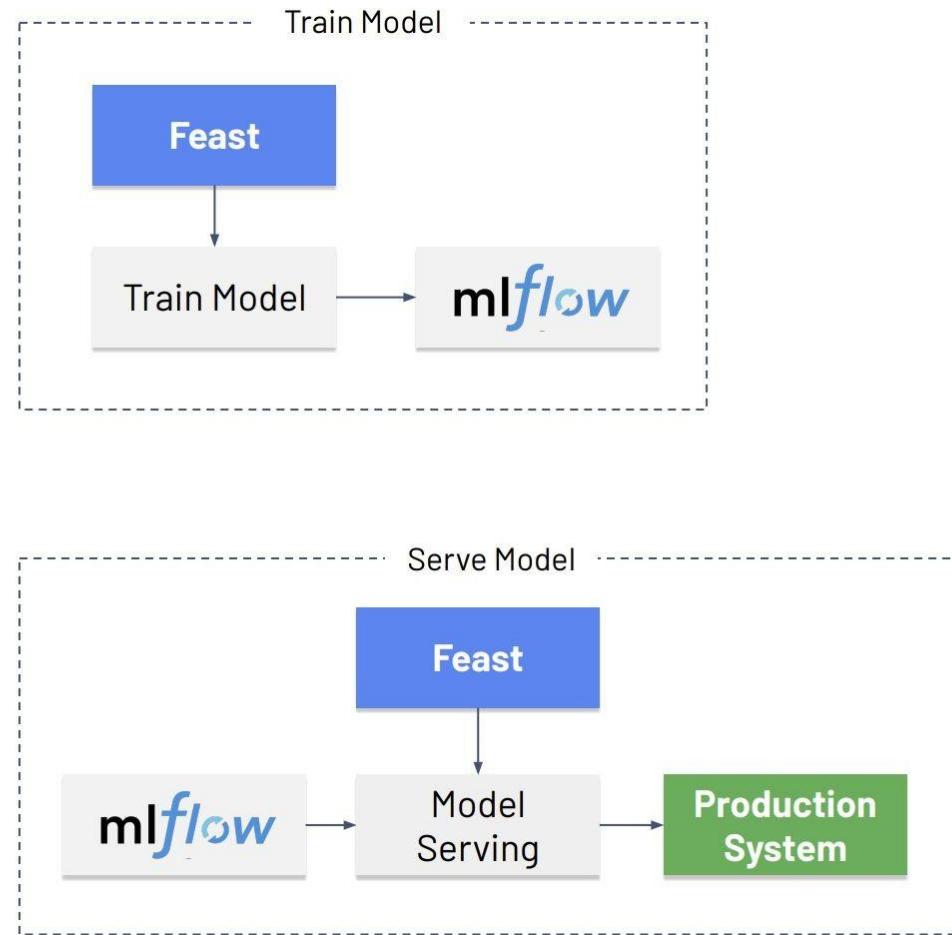
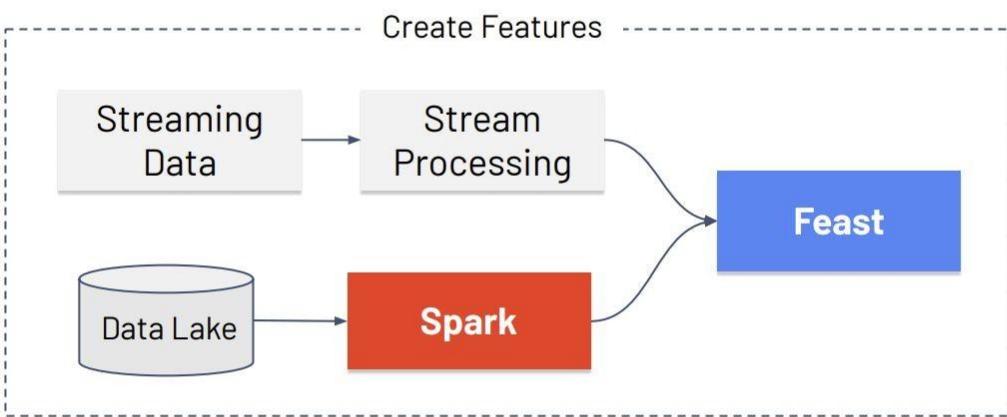
Возможные фичи

- 1) Сам текст коммента
- 2) Счетчики взаимодействия этого пользователя с разного рода объектами
 - пользователь написал сегодня 100500 комментариев
 - т из них было помечено другими пользователями как спам
 - пролайкал n записей
- 3) Коллаборативный эмбеддинг пользователя(считается раз в пару суток)

Проблемы?



Идея feature-store





Train vs Inference

DATA FOR TRAINING	DATA FOR INFERENCE
Можем подождать какое-то время	Нужно прямо сейчас(<10 ms)
Нужно много данных за какой-то временной период	Нужно данных, чтобы сделать предсказание для одного объекта



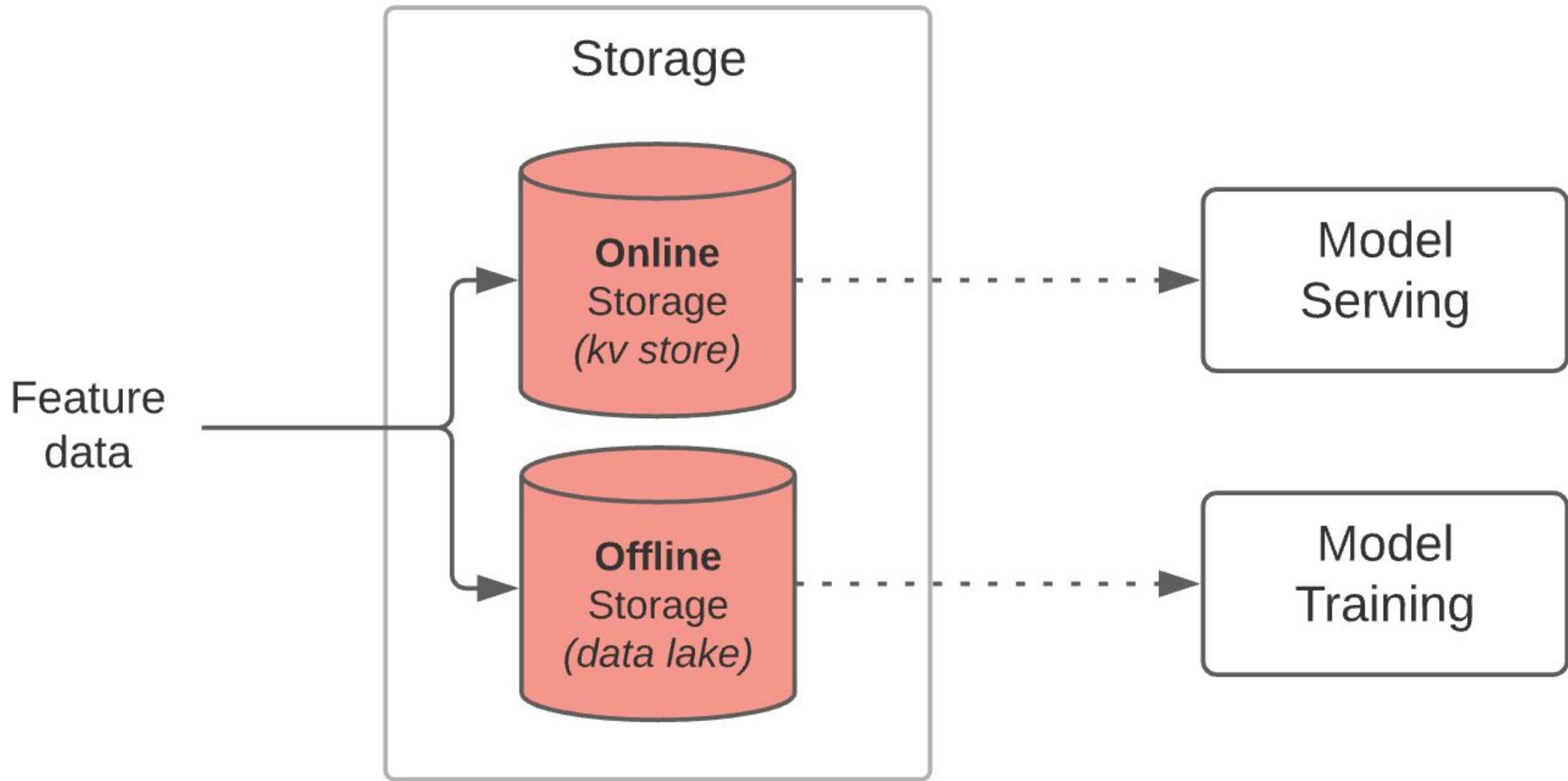
Train vs Inference

DATA FOR TRAINING	DATA FOR INFERENCE
Можем подождать какое-то время	Нужно прямо сейчас(<10 ms)
Нужно много данных за какой-то временной период	Нужно данных, чтобы сделать предсказание для одного объекта
Хранятся в hdfs(hive), etc	Хранятся в высокодоступном хранилище(лучше in-memory) - redis,

Train vs Inference

CHALLENGE: Должны быть консистентны между собой

DATA FOR TRAINING	DATA FOR INFERENCE
Можем подождать какое-то время	Нужно прямо сейчас(<10 ms)
Нужно много данных за какой-то временной период	Нужно данных, чтобы сделать предсказание для одного объекта
Хранятся в hdfs(hive), etc	Хранятся в высокодоступном хранилище(лучше in-memory) - redis,



Путь данных в Feast

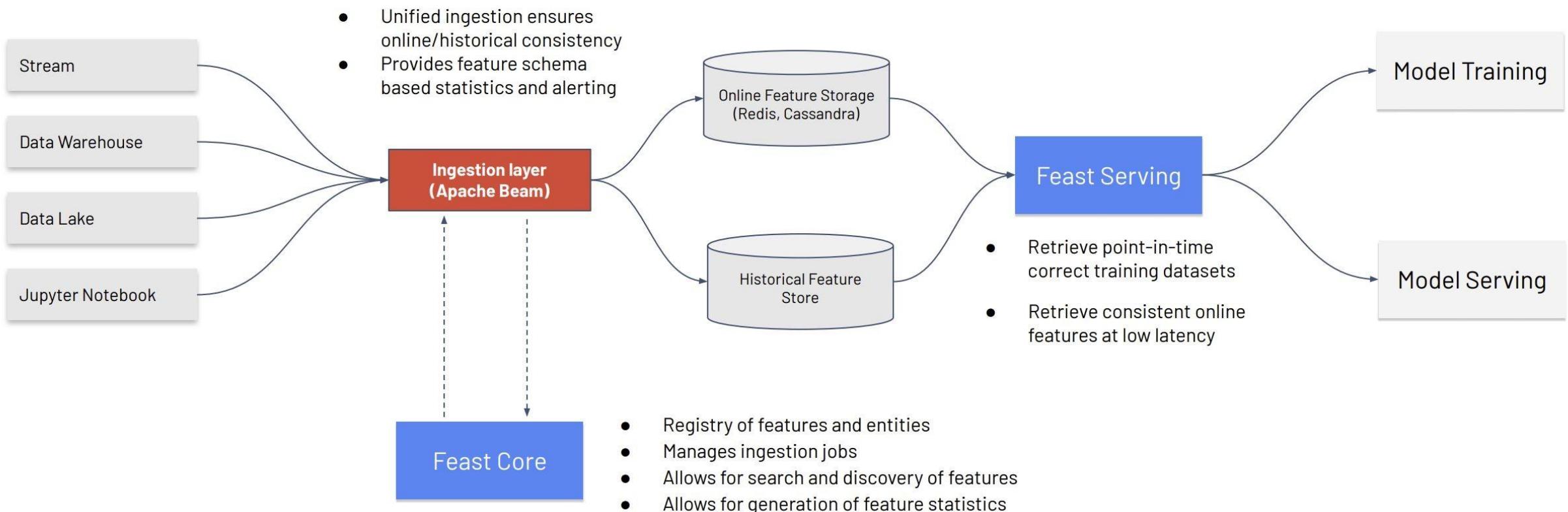
Your data

Ingestion

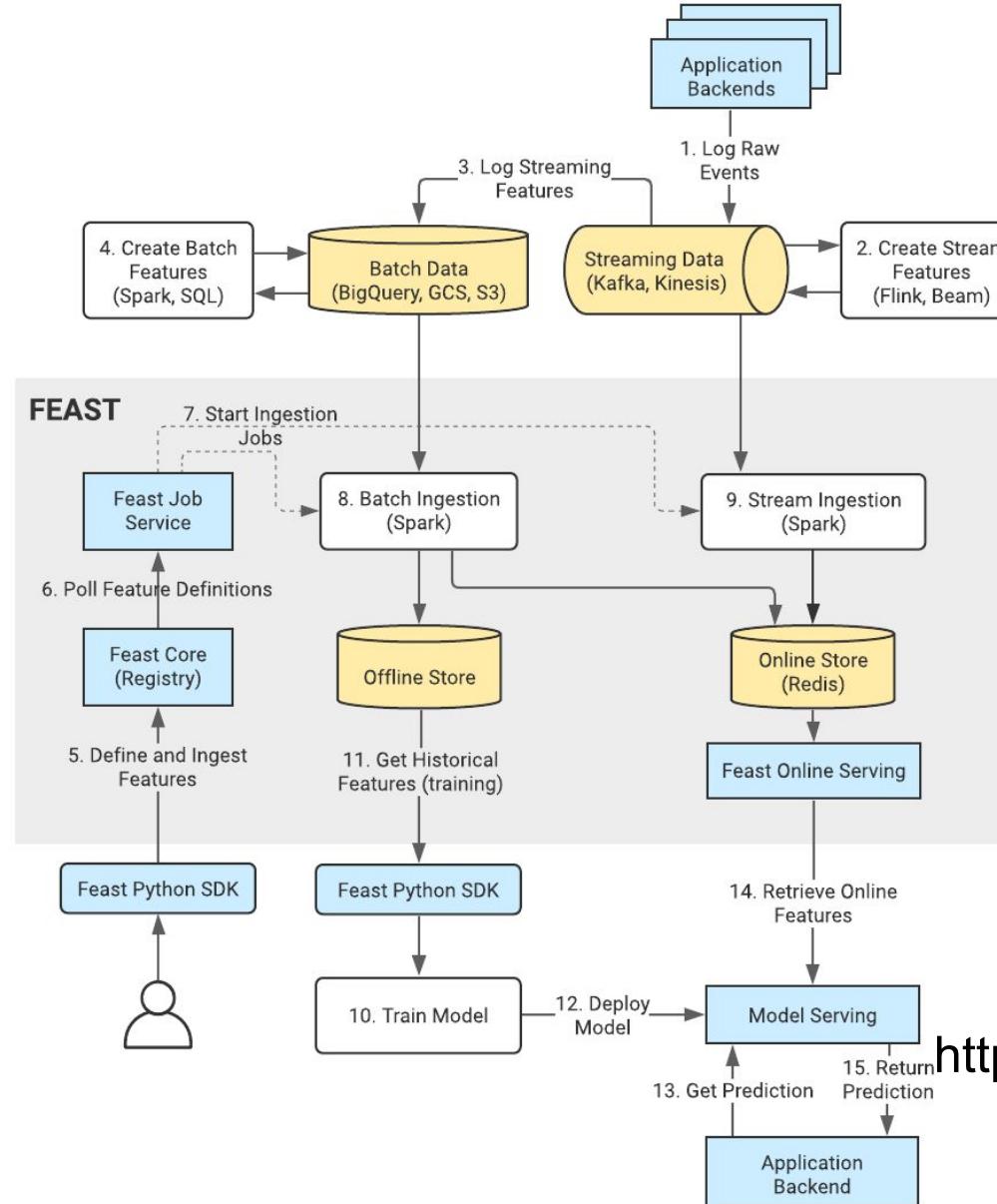
Storage

Serving

Production



Путь данных в Feast



Entites

customer_id.yaml

```
1 # Entity name
2 name: customer_id
3
4 # Entity value type
5 value_type: INT64
```



```
1 from feast import Entity, ValueType, FeatureSet
2
3 # Create a customer entity
4 customer = Entity("customer_id", ValueType.INT64)
5
6 # Create a feature set with only a single entity
7 customer_feature_set = FeatureSet("customer_fs", entities=[customer])
8
9 # Register the feature set with Feast
10 client.apply(customer_feature_set)
```



Feast

total_trips_feature.yaml

```
1 # Feature name
2 name: total_trips_24h
3
4 # Feature value type
5 value_type: INT64
```



```
1 from feast import Entity, Feature, ValueType, FeatureSet
2
3 # Create a driver entity
4 driver = Entity("driver_id", ValueType.INT64)
5
6 # Create a total trips 24h feature
7 total_trips_24h = Feature("total_trips_24h", ValueType.INT64)
8
9 # Create a feature set with a single entity and a single feature
10 driver_fs = FeatureSet("driver_fs", entities=[driver], features=[total_trips_24h])
11
12 # Register the feature set with Feast
13 client.apply(driver_fs)
```



Feature set

customer_transactions_feature_set.yaml

```
1 name: customer_transactions
2 entities:
3 - name: customer_id
4   valueType: INT64
5 features:
6 - name: daily_transactions
7   valueType: FLOAT
8 - name: total_transactions
9   valueType: FLOAT
```



The dataframe below (`customer_data.csv`) contains the features and entities of the above feature set.

datetime	customer_id	daily_transactions	total_transactions
2019-01-01 01:00:00	20001	5.0	14.0
2019-01-01 01:00:00	20002	2.6	43.0
2019-01-01 01:00:00	20003	4.1	154.0
2019-01-01 01:00:00	20004	3.4	74.0

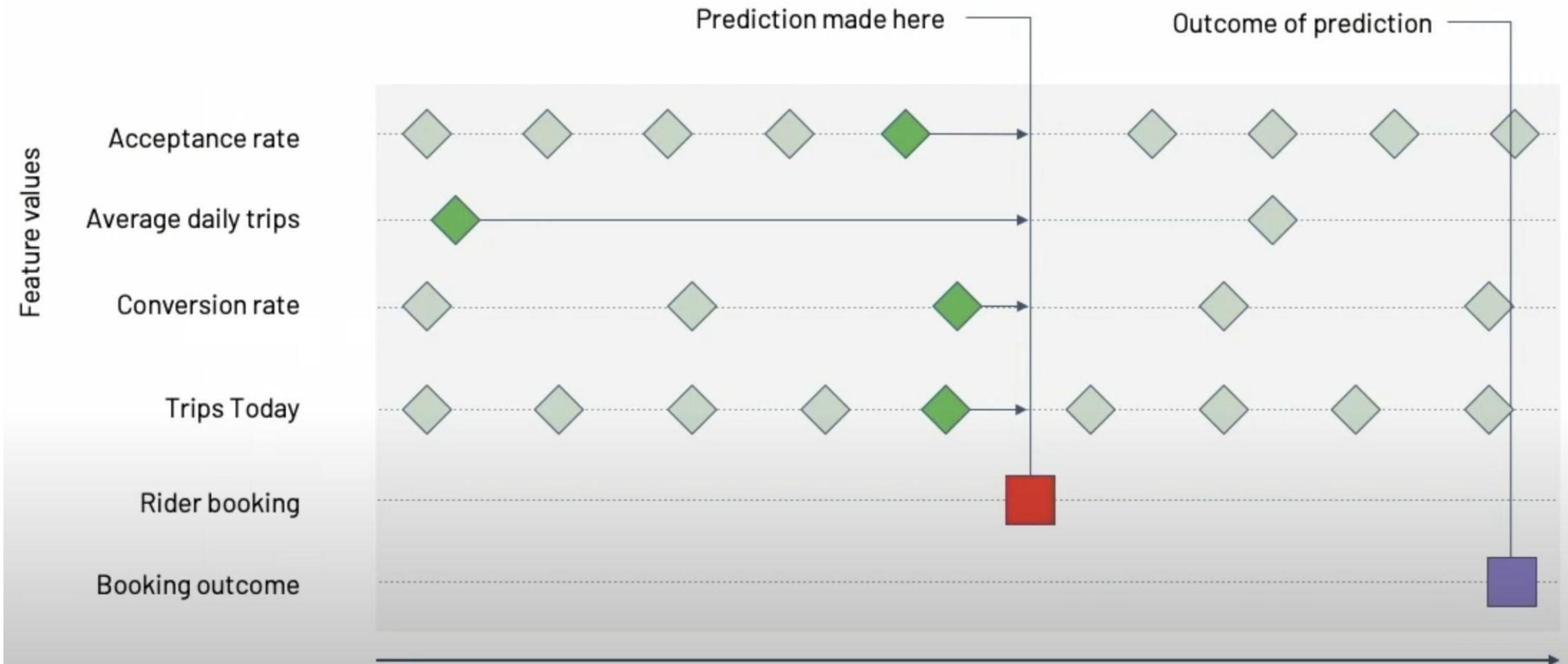
Online feature retrieval

```
feature_vector = store.get_online_features(  
    feature_refs=[  
        'driver_hourly_stats:conv_rate',  
        'driver_hourly_stats:acc_rate',  
        'driver_hourly_stats:avg_daily_trips'  
    ],  
    entity_rows=[{"driver_id": 1001}]  
).to_dict()  
  
pprint(feature_vector)
```

Offline feature retrieval

```
training_df = store.get_historical_features(  
    entity_df=entity_df,  
    feature_refs = [  
        'driver_hourly_stats:conv_rate',  
        'driver_hourly_stats:acc_rate',  
        'driver_hourly_stats:avg_daily_trips'  
    ],  
).to_df()  
  
training_df.head()
```

Point-in-time join



Point-in-time joins

Entity DataFrame

ride_id	datetime	customer_id	driver_id	trip_completed
10001	2020-01-01 12:00:00	2008	1001	True
10002	2020-01-01 12:12:12	2001	1002	False
10003	2020-01-01 13:13:13	2003	1003	True
10004	2020-01-01 15:15:15	2010	1004	False

Driver DataFrame

driver_id	datetime	average_daily_rides	maximum_daily_rides	rating
1001	2020-01-01 00:00:00	1.3	3	4.2
1002	2020-01-01 00:00:00	4.3	5	3.8
1003	2020-01-01 00:00:00	2.9	4	4.5
1004	2020-01-01 00:00:00	6.8	11	4.9

Point-in-time join

Joined DataFrame

average_daily_rides	maximum_daily_rides	rating	trip_completed
1.3	3	4.2	True
4.3	5	3.8	False
2.9	4	4.5	True
6.8	11	4.9	False

Point-in-time joins

```
# Do point in-time-join between entity_df and feature dataframe
entity_df_with_features = pd.merge_asof(
    entity_df_with_features,
    df_to_join,
    left_on=entity_df_event_timestamp_col,
    right_on=event_timestamp_column,
    by=right_entity_columns,
    tolerance=feature_view.ttl,
)
```

DEMO
Смотрим FEAST локально
и на ноутбук в облаке



Feast is not!

Is Feast a feature computation system?

No. Even though some feature stores include transformations, Feast purely manages retrieval. Feast is used alongside a separate system that computes feature values. Most often, these are pipelines written in SQL or a Python Dataframe library and scheduled to run periodically.

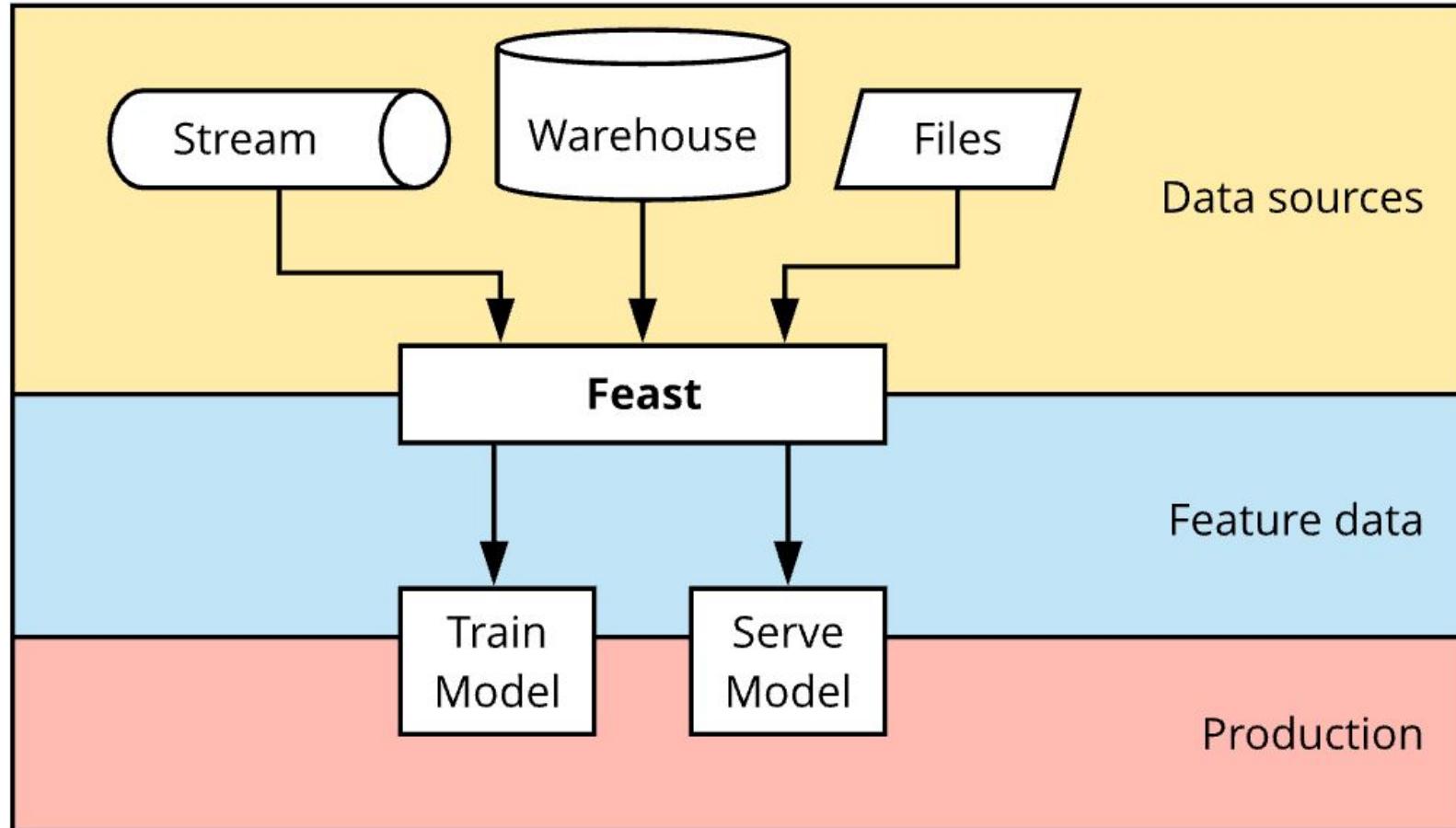
If you need a managed feature store that provides feature computation, check out [Tecton](#).

Is Feast a database?

No. Feast is a tool that manages data stored in other systems, (e.g. BigQuery or Cloud Firestore.) It is not a database, but it helps manage data stored in other systems.



Feast is



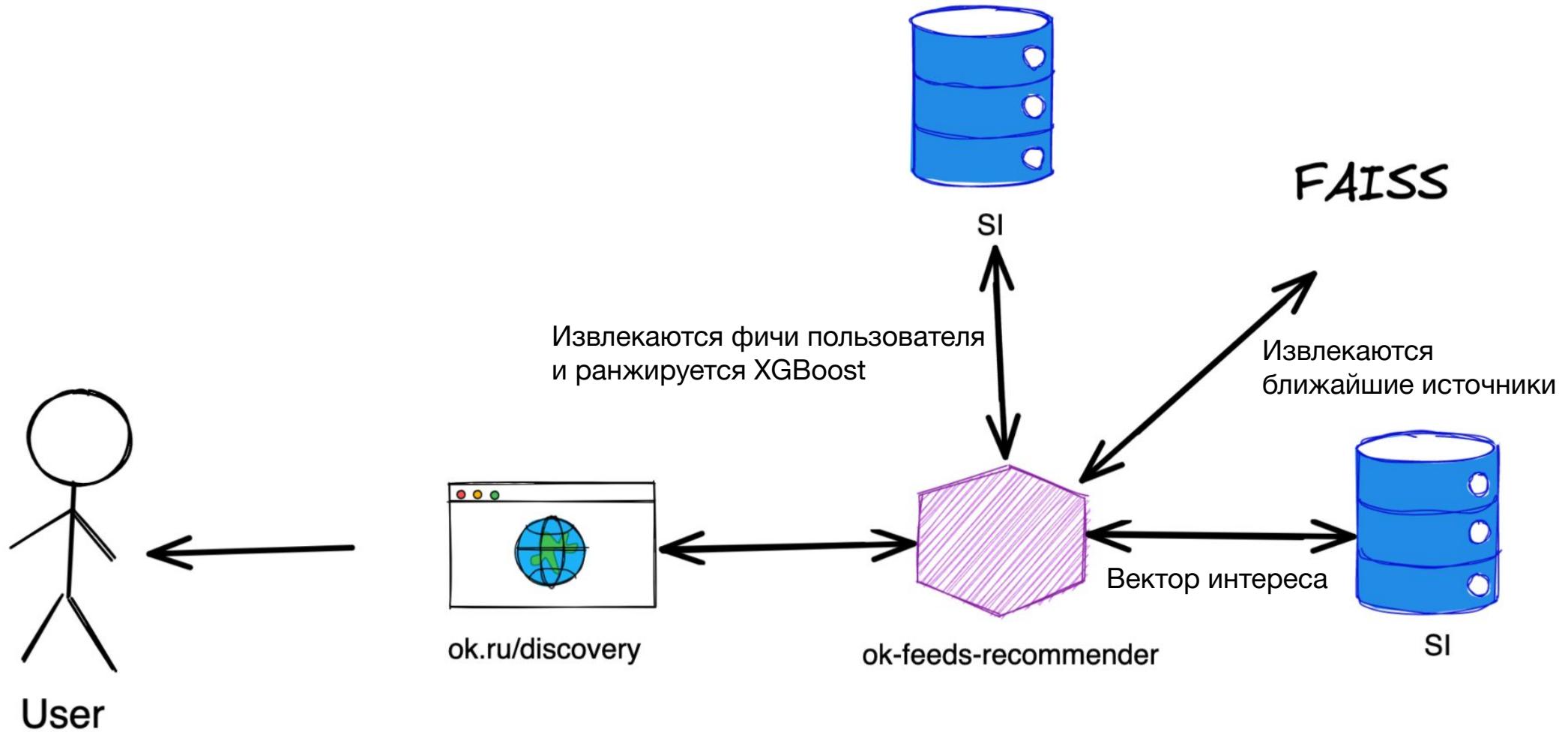
Platform	Open-Source	Offline	Online	Metadata	Feature Engineering	Supported Platforms	TimeTravel / Point-in-Time Queries	Training Data
Hopsworks	AGPL-V3	Hudi/Hive	MySQL Cluster	DB Tables, Elasticsearch	(Py)Spark, Python	AWS, GCP, On-Prem	SQL Join or Hudi Queries	.tfrecords, .csv, .npy, .parquet, .hf5, etc
Michelangelo	N/A	Hive	Cassandra	Content	Spark, DSL	Proprietary	SQL Join	Streamed to models?
Feast	Apache V2	BigQuery	BigTable/Redis	DB Tables	Beam, Python	GCP	SQL Join	Streamed to models
Conde Nast	N/A	Kafka/Cassandra	Kafka/ Cassandra	Protocol Buffers	Shared libraries	Proprietary	?	Protobuf
Zipline	N/A	Hive	KV Store	KV Entries	Flink, Spark, DSL	Proprietary	Schema	Streamed to models?
Comcast	N/A	HDFS, Cassandra	Kafka / Redis	Github	Flink, Spark	Proprietary	No?	Unknown
Netflix Metaflow	N/A	Kafka & S3	Kafka & Microservices	Protobufs	Spark, shared libraries	Proprietary	Custom	Protobuf
Twitter	N/A	HDFS	Strato / Manhatten	Scala shared feature libraries	Scala DSL, Scalding, shared libraries	Proprietary	No	Unknown
Facebook FB Learner	N/A	?	Yes, no details	Yes, no details	?	Proprietary	?	Unknown
Pinterest Galaxy	Pinterest Galaxy	Content	S3/Hive	Yes, no details	Yes, no details	DSL (Linchpin), Spark	Proprietary	?
Iguazio data science	N/A	Parquet	Yes, in mem database	Yes, no details	Spark, Python, Nuclio	AWS, Azure, GCP, on-prem	Yes, native time series or SQL	Yes, no details

<https://mlops.community/learn/feature-store/>

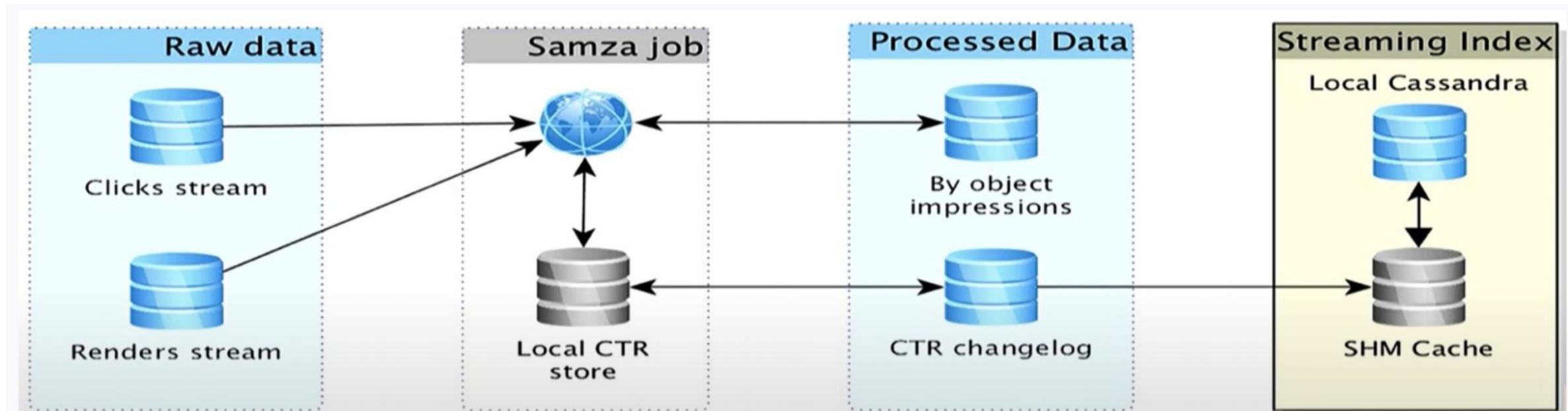
<https://www.featurestore.org/>

Кейсы из OK

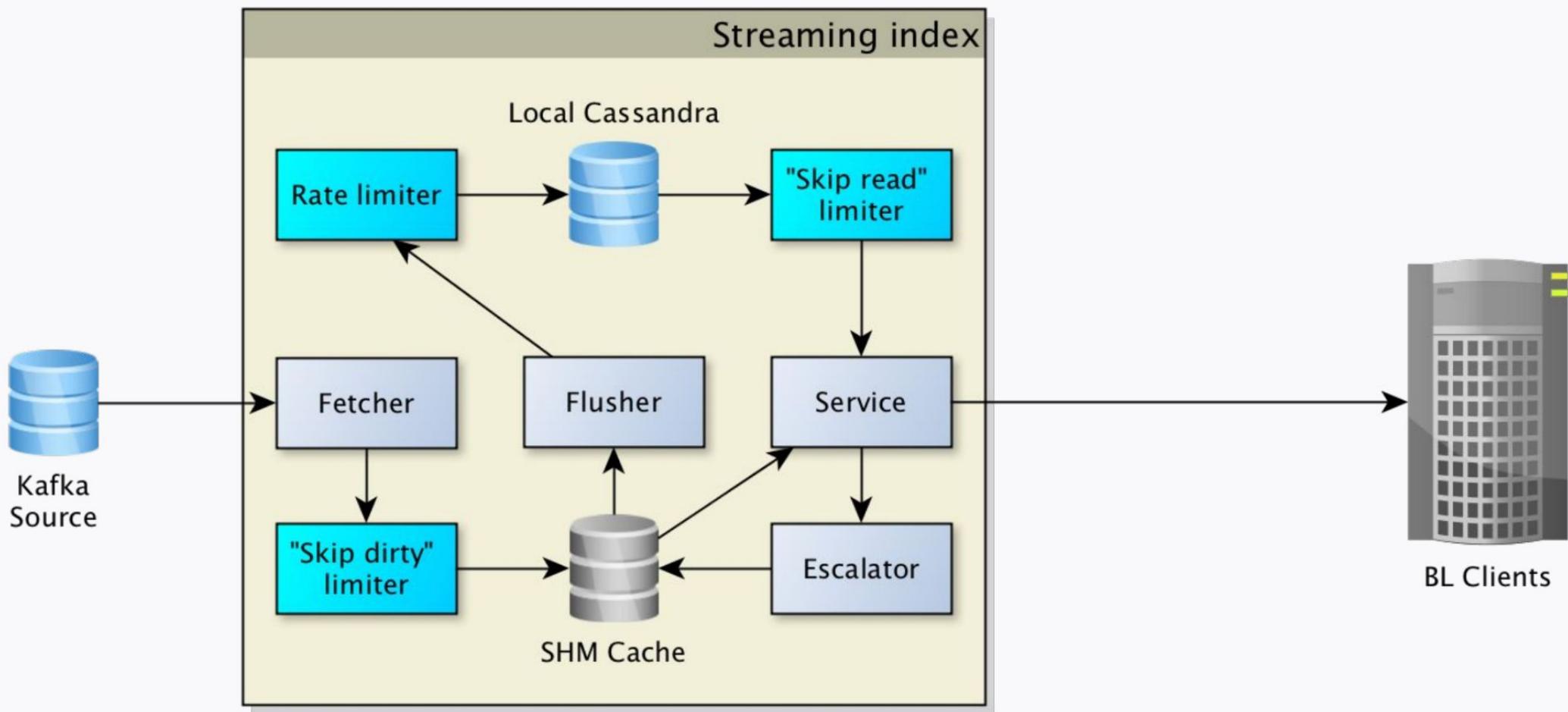
Рекомендации в ОК



OK Streaming Index Architecture

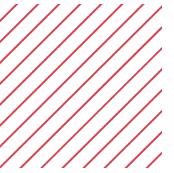


OK Streaming Index Architecture



Доклад про наш Streaming Index





Ссылка на опрос

<https://forms.gle/L1RXAmdWn9ji7ohLA>

академия
больших
данных



Очереди сообщений и Feast

Михаил Марюфич, MLE

