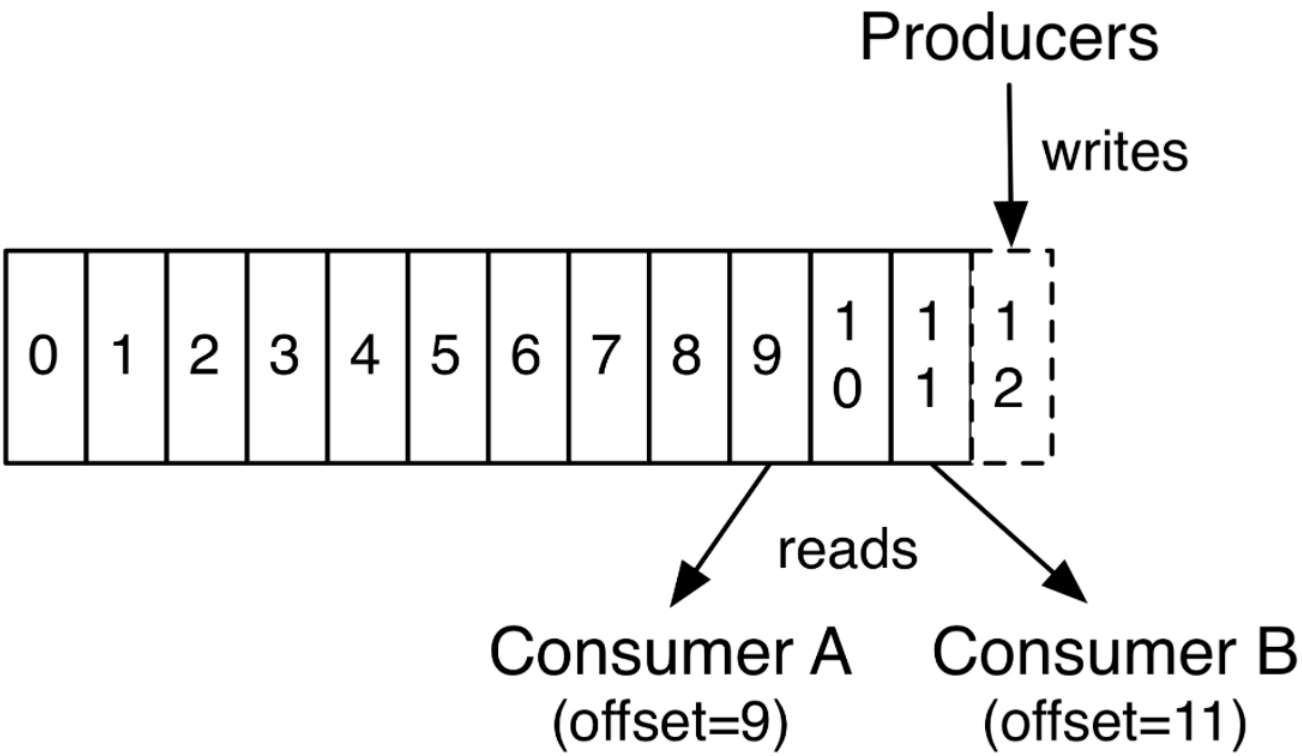
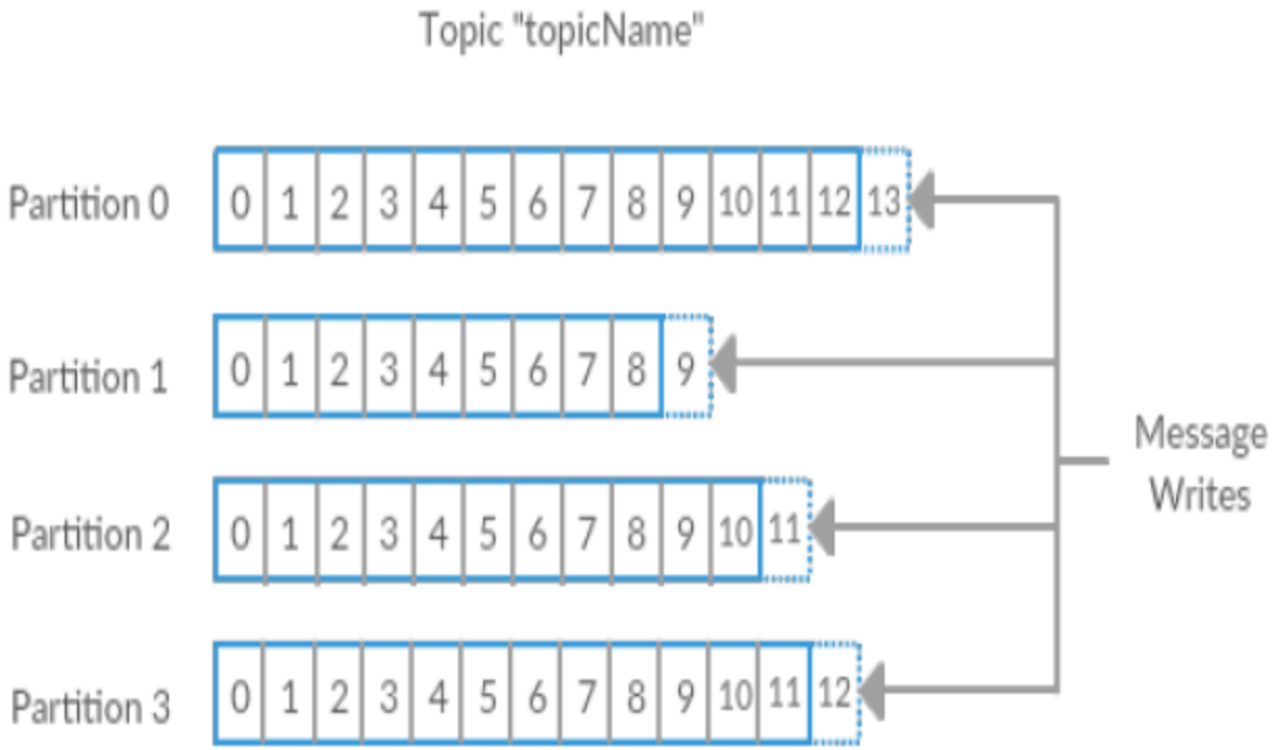
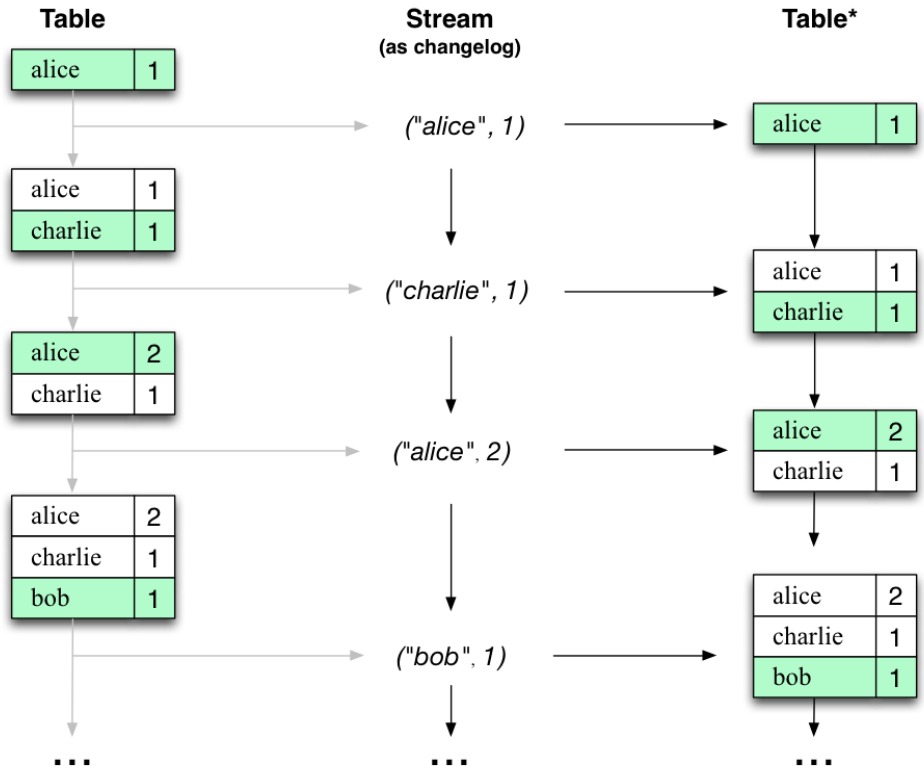
Kafka / Kafka Ecosystem / Design patterns

Realtime data stream processing

- Apache Kafka is a stream-processing software platform
- Created by LinkedIn in 2011
- Open source under Apache. `Confluent` is main contributor
- Used globally:
  - Pinterest - Ads
  - Netflix - Events
  - DataDog - Input data
  - Twitter - Input for Storm
- Used locally:
  - Unity - Events
  - Nordea - ETL (Extract, Transorm, Load)
  - Zalando - ESB (Enterprise service bus)




Topic "topicName"
Partition 0
0 1 2 3 4 5 6 7 8 9 10 11 12 13
Partition 1
0 1 2 3 4 5 6 7 8 9
Partition 2
0 1 2 3 4 5 6 7 8 9 10 11
Partition 3
0 1 2 3 4 5 6 7 8 9 10 11 12
Message
Writes



- Consume again and again
- Persistent by default - Kafka is a DB
- Ordering guarantees
- 1M/s is common
- Clear data - data retention / compaction / tombstone
- Contracts - Avro, other binary protocols
- You know your clients - consumer ID

Kafka is not a queue

- Actually, it is a queue - everything else is not!
- No selective ack
- Accept duplicates
- Possible to ack every message, but not usual
- Shouldn't replace `[your favorite *MQ]`
- Event? => Kafka
- Data for action ? => `*MQ`

Kafka Streams

```
val stream = builder.stream("words");
val pattern = Pattern.compile("\\W+");
val counts = source.flatMapValues(value -> pattern.split())
              .mapValues(value -> value.toLowerCase())
              .filter((key, value) -> value != "the"))
              .groupByKey()
              .count("CountStore")
              .toStream()
counts.to("counts")
```

- KStream vs KTable

Join operands	Type	(INNER) JOIN	LEFT JOIN	OUTER JOIN
KStream-to-KStream	Windowed	Supported	Supported	Supported
KTable-to-KTable	Non-windowed	Supported	Supported	Supported
KStream-to-KTable	Non-windowed	Supported	Supported	Not Supported

- Ex: User action <> Company user
- Ex: Ad shown <> Ad clicked

- Java lib - it's still about Consumers/Producers
- Realtime stream processing
- Scalable, masterless, rolling restarts
- Join/Split/Aggregate/GroupBy streams
- Stateful with failover(!) if needed
  - Local state using RocksDB (or custom implementation)
  - Remote state in Kafka
- No backpressure needed as Kafka itself handles that

Kafka KSQL

- Streaming SQL - streams without code, but with UDF

```
SELECT user_id, page, action FROM clickstream c
LEFT JOIN users u ON c.user_id = u.user_id
WHERE u.level = 'Platinum';
```

- Realtime continuous queries. CLI/WebUI
- Observe data, change it - output it to new output topic

Kafka Connect

- Stream to/from [TARGET] from/to Kafka
- Mongo, Elastic, MySQL, Postgres, Cassandra, RabbitMQ, Redis
- FTP, Files, SalesForce. Products from AWS, GCP, Oracle
- Writing custom connectors is simple
- Scalable - many workers

- RedHat Debezium: MqSQL/Mongo/Postgres commit logs -> kafka

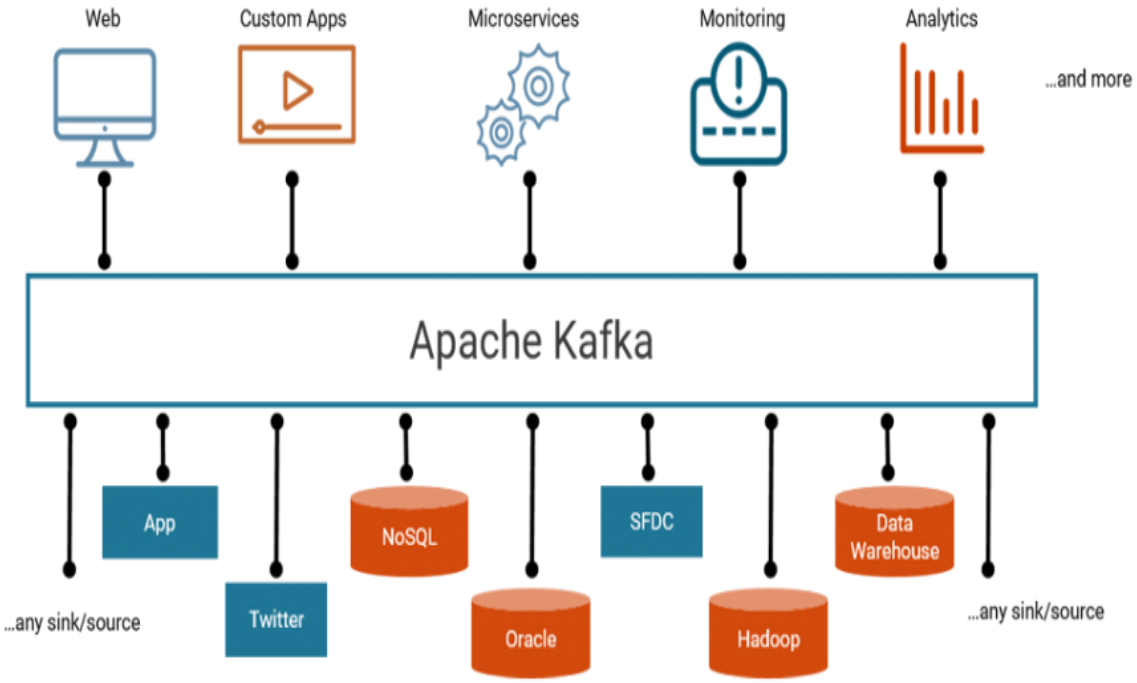
Kafka REST Proxy

- Consume/Produce messages via REST call
- Scalable
- Ex: Create new topic from KSQL query, start consuming it via REST right away
- Ex: CLI, scripts
- Ex: Introduce Kafka without introducing Kafka

Batch vs Stream

- Run every X minutes VS Put event into stream every X minutes
  - Decouple schedule from the action
  - Scalable
- Batch cannot be reactive
- Every hour make 20k req VS continuously processing 20000/60/60 = 5 req/s
  - One liner to create KStream with needed data
- History for free
  - Don't do work if fresh enough
  - Avoid duplicate processing

Extract > Transform > Load



How to split a monolith

- Integration on top is hard
- Integration on the data level
  - Expose DB table as a stream
  - Create REST wrapper on top of a stream
  - Stop adding more clients to the monolith and point to new service
  - Lower the load on the monolith
  - Resilient to monolith errors
  - Only read, but possible to proxy write to the monolith
  - Few seconds lag (you cannot read your own writes!)

Microservices using Kafka Streams

- Dataset is small? Cache it all using KTable and get 0ms joins
- Stateful stateless services
- Deliver configuration without restarts
- Audit log

Event driven microservices without HTTP

- HTTP on the edge - inside only events

```
Batch job => POST /company/register
```

VS

```
    Batch
        => Event CompanyRegistrationRequest
            => Event CompanyRegistration
                => SendEmailToCompanyAdmin
                => CreateCompanyDB
                => UpdateLandingPageCompaniesNum
```

Cache invalidation - solved

Questions?